Analogical proportion-based methods for recommendation - First investigations

Nicolas Hug, Henri Prade, Gilles Richard, Mathieu Serrurier

Analogical proportion-based methods for recommendation First investigations

Nicolas Hug [a], Henri Prade [a,b,*], Gilles Richard [a,c], Mathieu Serrurier [a]

[a] *IRIT, UPS-CNRS, 118 route de Narbonne, 31062 Toulouse Cedex 09, France*
[b] *QCIS, University of Technology, Sydney, Australia*
[c] *BITE, London, UK*

Abstract

Analogy making is widely recognized as a powerful kind of common-sense reasoning. This paper primarily addresses the relevance of analogical reasoning for recommender systems, which aim at providing suggestions of interest for end-users. A well-known form of analogy is that of analogical proportions, which are statements of the form "*a* is to *b* as *c* is to *d*". Encouraged by good results obtained in classification by analogical proportion-based techniques, we study the potential use of analogy as the main underlying principle for implementing rating prediction algorithms. We investigate two ways of using analogical proportions for that purpose. First, we exploit proportions between four users, each described by their respective ratings on items that they have commonly rated. The second prediction method only relies on pairs of users and pairs of items, and leads to better performances. Finally, in order to know to what extent it may be meaningful to apply analogical methods in data analysis, we address the problem of mining analogical proportions between users (or items) in ratings datasets. Altogether, this paper initiates a general investigation of the potential use of analogical proportions for recommendation purposes.

Keywords: Analogical proportion; Recommendation; Analogy mining

1. Introduction

Recommendation systems take advantage of a variety of information sources [1]. They propose items or products on the basis of their descriptions to users whose preferences are known, and they also take advantage of the behavior of other users that are similar to the recommendee. Exploiting preferences may call for fuzzy set methods, which may handle both preferences and similarities; see e.g. [2] for an early example. Similarity is also a graded notion that underlies case-based reasoning, which can be embedded in a fuzzy rule-based approach and be related to k-nearest neighbor approaches [3–5].

* Corresponding author.
E-mail addresses: nicolas.hug@irit.fr (N. Hug), henri.prade@irit.fr (H. Prade), gilles.richard@irit.fr (G. Richard), mathieu.serrurier@irit.fr (M. Serrurier).

https://doi.org/10.1016/j.fss.2018.11.007

The recommendation problem considered in this paper is the prediction of missing ratings on the basis of known ratings. Namely, given a history of past ratings R expressing the preferences of a set of users towards a set of items, the goal is to infer the preferences of any user towards any item. The principal challenge is that the set of known ratings R is extremely small compared to the number of potential predictions.

We more precisely explore the idea of applying analogical reasoning to this problem. Indeed, analogy-making is considered as a useful form of heuristic reasoning, and recent works [6,7] have shown that it can be applied to classification problems. Analogy is used here in terms of analogical proportions, i.e. statements of the form "a is to b as c is to d". In case-based reasoning, situations with known conclusions are put in parallel *one by one*, with a new pair (situation 0, conclusion 0) where 'conclusion 0' is unknown. Then case-based reasoning can be viewed as a particular instance of analogical reasoning since one can say that "conclusion 0 should be to conclusion i as situation 0 is to situation i". However, there is a more sophisticated way that relies on analogical proportions of the form "(situation 0, conclusion 0) is to (situation 3, conclusion 3) as (situation 2, conclusion 2) is to (situation 1, conclusion 1)". This requires to put the situation on which one wants to conclude in parallel with *three* other situations where the corresponding conclusion is known [8]. Then using a formal model of an analogical proportion [9,10], and observing that analogical proportions hold on various features describing the four situations, one concludes that "conclusion 1 is to conclusion 2 as conclusion 3 is to conclusion 0" should hold as well, which leads to compute 'conclusion 0' from this latter relation.

The idea of applying analogy to recommendation is not entirely new. Sakaguchi et al. [11] use four-terms analogy in a case-based reasoning style for proposing dishes to users. In this paper, we investigate the use of analogical proportions in three different directions.

We first present an analogy-based algorithm for predicting ratings. This algorithm is a direct application of an analogical inference principle, and is strongly related to the analogical classifiers [12]. The basic underlying idea is that if four users (a, b, c, d) are in analogy regarding their common items, then it should also be the case for other items that d has not yet rated. The idea of completing missing data by using analogical proportions has been already experienced in Boolean databases [13], in low sparsity situations. In contrast, recommender system databases are extremely sparse: often more than 99% of the values are missing.

Next, we investigate a different way of using analogical proportions for solving the same problem. This second approach relies on proportions of the form "the rating of user u for item i is to the rating of user v for item i as the rating of user u for item j is to the rating of user v for item j". These proportions are different from the previous ones since they involve two users u and v, and two items i and j. Interestingly enough, this leads to an estimation process quite close to the one used in Takagi–Sugeno fuzzy rule-based controllers [14] where similarity-based weighted averages are performed.

Lastly, we take inspiration from mining algorithms for association rules, and devise an algorithm for the mining of analogical proportions in incomplete databases, like the ones encountered in a rating prediction setting. This may be viewed as a preliminary step for checking if there are enough high-quality analogical proportions before applying one of the previous prediction methods. Moreover, this also illustrates what kind of analogies can be extracted between items.

The present paper expands the contents of three conference papers [15–17], providing preliminary studies of the three above issues. Altogether, it provides a synthetic view of the different ways one may think of applying analogical proportions in the setting of recommendation systems. Moreover, explanations on the different issues have been reworked and more detailed, while the data mining part, now available in English, substantially expands [17].

This paper is structured as follows. Section 2 provides the necessary background on the rating prediction task that we plan to address. Section 3 introduces analogical proportions and explains how to use them for inference tasks. Section 4 describes our first approach to the rating prediction problem using proportions between users, and reports experimental results. Section 5 presents our second approach which is more scalable and that relies on the idea that some users make a more or less severe use of the rating scale. Finally, Section 6 is devoted to the mining of analogical proportions in incomplete databases. The obtained results will allow us to retrospectively explain the modest performance of our first algorithm for rating prediction.

2. Background on recommender systems

This first section provides the necessary background on recommender systems, and describes the problem of rating prediction that we address in this paper.

2.1. Taxonomy

To provide an organized view of the vast recommendation landscape, various taxonomies for recommender systems have been proposed. Using that of [18], we will briefly describe *content-based*, *collaborative* and *knowledge-based* approaches. In real systems, the frontiers between these three families are not that sharp, and practical recommendation solutions usually fall into more than one single category. We refer to such systems as *hybrid* recommenders.

Content-based methods. The general idea of content-based algorithms is as follows: they try to recommend to a user some items that are *similar* to those that the user has liked in the past. For example, if the renting history of a user mainly contains science fiction movies, the aim of the system is to learn these preferences and to recommend some other science fiction movies to the user. See [19] for a recent overview of content-based methods.

To find out similar items to those that a user has liked, these systems rely on a similarity measure whose nature strongly depends on the representation of the items. In most cases, the similarity measure is based on some metadata describing the items (hence the name *content*-based). In the case of movies, the metadata could be for example the genre, main actors, film director, etc.

A well-known drawback of content-based recommenders is their tendency to overfit and recommend only items that users may already know or do not need anymore. The recommendations tend to lack in novelty, surprise and diversity. Also, content-based systems tend to output much less accurate predictions than the collaborative filtering methods (the way accuracy is computed will be described in Section 2.3). This is why collaborative filtering methods have gained in popularity.

Collaborative filtering. The main idea behind collaborative filtering [20] algorithms is to recommend items to a user that other users with similar tastes have liked in the past. In its most general form, collaborative filtering methods (also called social filtering) try to model the social environment of users and to output recommendations based on the behavior of their peers.

Probably the most common way of modeling the social environment of a user is to use historical transactions stored in the system, for example using user-item ratings: two users are peers if their ratings are similar. If Bob and Alice usually agree on their movie ratings, i.e. they like the same movies and dislike the same movies, and if Alice has not seen a movie that Bob has liked, then it will be recommended to Alice.

A main advantage of collaborative filtering methods over content-based ones is that they do not require any item or user knowledge (metadata): only user-item feedback (ratings) is needed to output a prediction. All the recommendation algorithms that we will propose in this paper are of a collaborative nature. Section 2.4 will be devoted to short description of two popular collaborative techniques: the neighborhood approach and matrix factorization-based methods.

Knowledge-based systems. The two families we have described so far (content-based and collaborative methods) are only suitable for certain kinds of items. While movies, books, music or news can all fit quite well within these two frameworks, this is not the case for some other items such as cars, bank loans, or real estates.

In broad terms, we refer to knowledge-based systems as any method that is neither content-based nor collaborative. In general, these systems are used in settings where the domain and contextual knowledge (i.e. the knowledge about the items that are available in the system) prevails over any other source of information, such as social information for example. A practical instance of knowledge-based recommendation takes the form of a recommendation session, where users indicate their needs in an interactive manner, and the systems tries to match these needs as best as possible to provide useful recommendations.

According to [21], there exist two kinds of knowledge-based systems. The first category is made up of the *case-based* systems: the system will try to recommend items that are similar to an ideal item expressed by the user's needs.

The second kind of knowledge-based recommenders are the so-called *constraint-based* systems (see [22] for a complete review). In such systems, users and items are described by a set of properties, along with a set of constraints. The problem of recommending an item to a user then takes the form of a classical constraint satisfaction problem.

2.2. Recommendation as a rating prediction problem

Let U be a set of users and I a set of items. For some pairs $(u, i) \in U \times I$, a rating r_{ui} is supposed to have been given by user u to express their preferences towards the item i. The way the preferences are expressed depends on the rating scale that is used, but the numerical scale $[1, 5]$ is very common.

The algorithms we will design will be of a collaborative nature, and therefore they will rely on a dataset containing user-item interactions (in our case, user-item ratings). Let us denote by R the set of known ratings recorded in the system. In real systems, the size of R is very small with regard to the potential number of ratings which is $|U| \times |I|$, as many ratings are missing. The set U_i will denote the set of users that have rated item i, and $U_{ij} \stackrel{\text{def}}{=} U_i \cap U_j$ is the set of users that have rated both items i and j. Similarly, I_u is the set of items that user u has rated, and $I_{uv} \stackrel{\text{def}}{=} I_u \cap I_v$ is the set of items that both users u and v have rated.

A very common way of providing personalized recommendations to a target user is to estimate their tastes regarding the items that the system provides. The taste of a user u for a given item i is usually represented as the rating that u would give to i. Once these estimations are made, a simple option is to recommend the items with the highest ratings among all the estimated scores for the considered user (using the implicit assumption that a user should be interested in the items with high scores).

More formally, a recommender system usually proceeds as follows:

1. Using a prediction algorithm A, estimate the unknown ratings r_{ui} (i.e., $r_{ui} \notin R$). This estimation $A(u, i)$ will here be denoted $\hat{r}_{ui}$.
2. Using a recommendation strategy S and in the light of the previously estimated ratings, recommend items to users. For instance, a basic yet common strategy is to suggest to user u the items $i \notin I_u$ with the highest estimation $\hat{r}_{ui}$.

We now describe how the quality of a prediction algorithm and of a recommendation strategy can be evaluated.

2.3. Recommender system evaluation

The most common evaluation protocol is probably to perform *off-line evaluation*: in this setting, we dispose of a dataset of past transactions (the set of all ratings R), and try to simulate user behavior to compute different measures. In general, we will use k-folds cross-validation procedures to reliably evaluate each of the performance measures. The set of all ratings R is divided into k (typically 5) disjoint sets of equal sizes; at each of the k iterations, the test set R_{test} is set to the kth subset and the training set R_{train} is set as the union of the $k - 1$ remaining subsets: the system will be trained on R_{train} and tested on R_{test}. The reported performances are then averaged over the k folds.

Providing an accurate measure of the overall quality of a recommender system is not a simple task, and diverse viewpoints have to be considered.

Accuracy. The performance of the algorithm A is usually evaluated in terms of accuracy, which measures how close the rating predictions $\hat{r}_{ui}$ are to the true ratings r_{ui}, for every possible prediction. The Root Mean Squared Error (RMSE) is probably the most common indicator of how accurate an algorithm is, and is calculated as follows:

$$\text{RMSE}(A) = \sqrt{\frac{1}{|R_{\text{test}}|} \cdot \sum_{r_{ui} \in R_{\text{test}}} (\hat{r}_{ui} - r_{ui})^2}.$$

Another common indicator for accuracy is the Mean Absolute Error (MAE), where important errors are not penalized more than small ones:

$$\text{MAE}(A) = \frac{1}{|R_{\text{test}}|} \cdot \sum_{r_{ui} \in R_{\text{test}}} |\hat{r}_{ui} - r_{ui}|.$$

From a mathematical viewpoint, *MAE* is a lower bound of *RMSE* and

$$RMSE \leq MAE * \sqrt{n}$$

where n is the sample size. On top of that, MAE is not a derivable function because of the use of absolute values. This is probably why MAE is less commonly used than RMSE, but it is still easier to interpret than RMSE, as it directly reflects the error range.

To better reflect the user-system interaction, other precision-oriented metrics are sometimes used in order to provide a more informed view.

Precision and recall. Precision and recall help measure the ability of a system to provide relevant recommendations, and are therefore indicators of the performance of the recommendation strategy S. They are defined by means of the number of true positives, true negatives, false positives and false negatives.

In the following, we denote by I^S the set of items that the strategy S will suggest to the users using the predictions coming from A. For ratings in the interval $[1, 5]$, a simple strategy could be for example to recommend an item i to user u if the estimation rating $\hat{r}_{ui}$ is greater than 4:

$$I^S \stackrel{\text{def}}{=} \{i \in I | \exists u \in U,\ \hat{r}_{ui} \geq 4,\ r_{ui} \in R_{\text{test}}\}.$$

We also define I^* as the set of items that are *actually* relevant to the users, i.e. the set of items that would have been recommended to the users if all the predictions made by A were exact:

$$I^* \stackrel{\text{def}}{=} \{i \in I | \exists u \in U,\ r_{ui} \geq 4,\ r_{ui} \in R_{\text{test}}\}.$$

The *precision* of the system is defined as the fraction of recommended items that are relevant to the users, and the *recall* is defined as the fraction of relevant recommended items over all relevant items:

$$\text{Precision} \stackrel{\text{def}}{=} \frac{|\ I^S \cap I^*\ |}{|\ I^S\ |},\quad \text{Recall} \stackrel{\text{def}}{=} \frac{|\ I^S \cap I^*\ |}{|\ I^*\ |}.$$

Precision and recall are complementary metrics: it would not make sense to evaluate an algorithm performance by only looking at the precision, without considering the recall. It is indeed quite easy to obtain a high recall (by simply recommending all of the items), but that would lead to a terrible precision. Precision and recall are commonly summarized into the F-measure, which is their harmonic mean:

$$\text{F} \stackrel{\text{def}}{=} 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Coverage. In its simplest form, coverage is used to measure the ability of a system to recommend a large amount of items: it is quite easy indeed to create a recommender system that would only recommend very popular items. Such a recommender system would drop to zero added value. We here define the *coverage* as the proportion of recommended items out of all existing items (I):

$$\text{Coverage} \stackrel{\text{def}}{=} \frac{|\ I^S\ |}{|\ I\ |}.$$

2.4. Two collaborative filtering techniques: neighborhood-based, and matrix factorization

We will here present two families of collaborative filtering algorithms: the neighborhood approach based on the well known k-NN algorithm, and the matrix factorization techniques. These two families of algorithms will serve as baselines against which we will compare the performances of our own algorithms in the next sections.

Neighborhood methods. The motto of collaborative filtering methods is to recommend some items that are appreciated by other users having the same tastes. This principle is carried out to the letter in neighborhood methods. Neighborhood approaches are instances of the general k-NN scheme. To estimate the rating r_{ui} of a user u for an item i, the most basic method consists in computing the set of k users that are most similar to u and that have rated i. We will denote this set $N_i^k(u)$. The computation of $N_i^k(u)$ depends of course on a similarity measure between users,

which is based on their respective ratings. The estimation $\hat{r}_{ui}$ of r_{ui} is then computed as an aggregate of the ratings r_{vi}, where v is one of the neighbors of u in $N_i^k(u)$. Usually, the aggregation is simply a mean weighted by the similarity between u and v:

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} r_{vi} \cdot \mathrm{sim}(u, v)}{\sum\limits_{v \in N_i^k(u)} \mathrm{sim}(u, v)},$$

where $N_i^k(u)$ is the set of users having rated i with the highest k values of sim with u. As can be seen, the above formula looks very similar to the interpolation principle underlying Takagi–Sugeno fuzzy controller where similarity degree is viewed as a fuzzy membership grade [14].

There are many ways to define the similarity metric between two users (or items).

- The *Cosine similarity* between two users u and v is defined as:

$$\text{Cosine sim}(u, v) \stackrel{\text{def}}{=} \frac{\sum\limits_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum\limits_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum\limits_{i \in I_{uv}} r_{vi}^2}}.$$

  Here, users u and v are considered as vectors in a vector space defined by the items they have both rated (the set of common items is I_{uv}). Their Cosine similarity simply is the cosine of the angle between the two vectors.
- The Pearson similarity can be viewed as a mean-centered version of the Cosine similarity:

$$\text{Pearson sim}(u, v) \stackrel{\text{def}}{=} \frac{\sum\limits_{i \in I_{uv}} (r_{ui} - \mu_u) \cdot (r_{vi} - \mu_v)}{\sqrt{\sum\limits_{i \in I_{uv}} (r_{ui} - \mu_u)^2} \cdot \sqrt{\sum\limits_{i \in I_{uv}} (r_{vi} - \mu_v)^2}},$$

  where μ_u and μ_v are the average rating of users u and v respectively.
- One last similarity metric that we will use is the Mean Squared Difference (MSD)[1]: The *Mean Squared Difference* between two users u and v is defined as:

$$\text{MSD}(u, v) \stackrel{\text{def}}{=} \frac{1}{|I_{uv}|} \cdot \sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2.$$

Matrix factorization techniques. The matrix factorization approach is heavily inspired by singular value decomposition (SVD): it postulates the existence of f factors/criteria (whose nature is not necessarily known) that determine the value of any rating r_{ui}. A user u is modeled as a vector $p_u \in \mathbb{R}^f$, where each component of p_u models the importance of the corresponding factor for u. Similarly, an item i is modeled as a vector $q_i \in \mathbb{R}^f$, where each component of q_i models how well i fits the corresponding criteria. Then, a rating prediction $\hat{r}_{ui}$ is calculated as the dot product of the two vectors p_u and q_i:

$$\hat{r}_{ui} = q_i^t \cdot p_u$$

The number of factors being set, the problem is here to estimate the vectors p_u and q_i for every possible users and items. In [23], this problem was studied from a Bayesian perspective, and was named Probabilistic Matrix Factorization (PMF), leading to the following regularized least squares problem:

$$\min_{p_u, q_i} \sum_{r_{ui} \in R} \left(r_{ui} - {q_i}^t \cdot p_u\right)^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2\right),$$

where λ is a regularization term. Not surprisingly, Stochastic Gradient Descent (SGD) tends to perform really well when it comes to solving this problem, and is quite simple to implement. Another optimization technique can be used,

[1] Strictly speaking MSD is actually a distance rather than a similarity metric, so we could take its inverse.

by noting that if either p_u or q_i is fixed, we obtain a convex problem that can be solved using classical least squares methods. The idea is then to first consider all q_i as constants and solve the associated linear problem. Then, the p_u are considered constant and the associated problem is solved. Repeating these steps an arbitrary number of times will also converge to a local solution, and this method is known as Alternating Least Squares [24].

3. Analogical reasoning with proportions

The following section provides the necessary background on analogical reasoning that will be used throughout this paper.

3.1. Formal definitions

An analogical proportion "a is to b as c is to d" states analogical relations between the pairs (a, b) and (c, d), as well as between the pairs (a, c) and (b, d). There are numerous examples of such statements, with which everybody will more or less agree, such as "calf is to cow as foal is to mare", or "brush is to painter as chalk is to teacher". Note that in the first example, all the four items refer to the same universe (which can thus be described in terms of the same set of attributes), while in the second case, two universes are involved (*tools* and *activities*). In the following, we only consider analogical proportions of the first kind. It is only rather recently that formal definitions have been proposed for analogical proportions, in different settings [25–27]. For more details, see [10,28,29].

It has been agreed since Aristotle time, taking lesson from geometrical proportions, that an analogical proportion T, as a quaternary relation, satisfies the three following characteristic properties:

1. $T(a, b, a, b)$ (reflexivity)
2. $T(a, b, c, d) \implies T(c, d, a, b)$ (symmetry)
3. $T(a, b, c, d) \implies T(a, c, b, d)$ (central permutation)

Note that the third property is more debatable when the items belong to two distinct universes (e.g. "brush is to chalk as painter is to teacher" hardly makes sense).

Starting from the proportion $T(a, b, c, d)$, the alternate application of the symmetry and central permutation properties leads to 8 equivalent expressions of the analogical proportion, namely $T(a, b, c, d) \iff T(c, d, a, b) \iff T(c, a, d, b) \iff T(d, b, c, a) \cdots$ There are two other equivalence classes of 8 elements. The second one is *represented* by $T(a, b, d, c)$ and the third one is *represented* by $T(a, d, c, b)$. This makes a total of 24 possible proportions from the 4-tuple (a, b, c, d). The existence of these three equivalence classes will be useful to us in Section 6.

There are various models of analogical proportions, depending on the target domain. When the underlying domain is fixed, $T(a, b, c, d)$ is simply denoted by $a : b :: c : d$. Standard examples are:

- Domain $\mathbb{R}$: $a : b :: c : d$ iff $a - b = c - d$ iff $a + d = b + c$ (arithmetic proportion)
- Domain $\mathbb{R}^n$: $\vec{a} : \vec{b} :: \vec{c} : \vec{d}$ iff $\vec{a} - \vec{b} = \vec{c} - \vec{d}$. This is just the extension of arithmetic proportion to real vectors. In that case, the 4 vectors $\vec{a}, \vec{b}, \vec{c}, \vec{d}$ build up a parallelogram.
- Boolean domain $\mathbb{B} = \{0, 1\}$:
  $a : b :: c : d$ iff $(a \wedge d \equiv b \wedge c) \wedge (a \vee d \equiv b \vee c)$

In the following, we will be mostly interested in the arithmetic proportions in $\mathbb{R}$ or in $\mathbb{R}^n$, and will work with analogies between ratings. The Boolean expression can be extended to the unit interval $[0, 1]$ in the multiple-valued logic setting [30,31]:

$$A(a, b, c, d) = \begin{cases} 1 - |(a - b) - (c - d)| & \text{if } a \geq b \text{ and } c \geq d, \text{ or } a \leq b \text{ and } c \leq d \\ 1 - \max(|a - b|, |c - d|) & \text{if } a \geq b \text{ and } c \leq d, \text{ or } a \leq b \text{ and } c \geq d. \end{cases}$$

This provides a basis for evaluating to what extent an analogical proportion holds in the numerical case.

3.2. Using analogical proportion for inference

To understand how one can infer new information on the basis of analogical proportions, we need to define the equation solving process. The equation solving problem amounts to finding a fourth element x to make the incompletely stated proportion $a : b :: c : x$ to hold. As expected, the solution of this problem depends on the target model, and may not always exist. For instance, in the case of arithmetic proportions in $\mathbb{R}$, the solution exists and is unique: $x = b - a + c$. In terms of geometry, this simply tells us that given 3 points, we can always find a fourth one (aligned with, or in the same plan as a, b, c) to build a parallelogram. However, a Boolean proportion is not always solvable: e.g. the equation $0 : 1 : 1 : x$ has no solution.

The analogical inference principle is, logically speaking, an unsound inference principle, but providing plausible conclusions [32]. It postulates that, given 4 vectors $\vec{a}, \vec{b}, \vec{c}, \vec{d}$ such that the proportion holds on some components, then it should also hold on the remaining ones. This can be stated as (where $\vec{a} = (a_1, a_2, \cdots a_n)$, and $J \subset [1, n]$):

$$\frac{\forall j \in J, a_j : b_j :: c_j : d_j}{\forall i \in [1, n] \setminus J, a_i : b_i :: c_i : d_i} \quad (\textit{analogical inference})$$

This principle leads to a prediction rule in the following context:

- 4 vectors $\vec{a}, \vec{b}, \vec{c}, \vec{d}$ are given where $\vec{d}$ is partially known: only the components of $\vec{d}$ with indexes in J are known.
- Using analogical inference, we can predict the missing components of $\vec{d}$ by solving (w.r.t. d_i) the set of equations (in the case they are solvable):

$$\forall i \in [1, n] \setminus J, \quad a_i : b_i :: c_i : d_i .$$

In the case where the items are such that their last component is a label, applying this principle to a new element $\vec{d}$ whose label is unknown leads to predict a candidate label for $\vec{d}$.

This prediction technique has been successfully applied to classification problems in both Boolean [12,33] and numerical settings [34,7], thus suggesting promising results in the recommendation task.

4. A rating prediction algorithm using proportions between users

We will here describe our first attempt to use an analogical proportion-based algorithm for recommender systems. The main idea of our algorithm is quite simple and is directly inspired by the process described in Section 3.2.

4.1. Algorithm

Here, we will make use of the analogical inference principle, but we will state it in slightly different terms. So far, the analogical inference principle stated that if four vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ are in proportion, then their labels should also be in proportion (in a classification context). Here, we will say that if an analogical proportion stands between four users a, b, c, d, meaning that for each item j that they have commonly rated, the analogical proportion $r_{aj} : r_{bj} :: r_{cj} : r_{dj}$ holds, then it should also hold for an item i that a, b, c have rated but d has not (i.e. r_{di} is the missing component):

$$\frac{r_{aj} : r_{bj} :: r_{cj} : r_{dj} \quad \forall j \in I_a \cap I_b \cap I_c \cap I_d}{r_{ai} : r_{bi} :: r_{ci} : r_{di} \quad \forall i \in I_a \cap I_b \cap I_c, \text{ and } i \notin I_d}$$

This leads us to estimate r_{di} as the solution y of the following analogical equation:

$$r_{ai} : r_{bi} :: r_{ci} : y.$$

Naturally, we want y to be in the same range as the other ratings. In our experiments, we deal with a $[1, 5]$ rating scale, so we will restrict solutions to this range.

We may find many 3-tuples (a, b, c) that are in proportion with u, so we will need to aggregate all the candidate solutions y. Given a pair (u, i) such that $r_{ui} \notin R$, the main procedure to estimate r_{ui} is as follows:

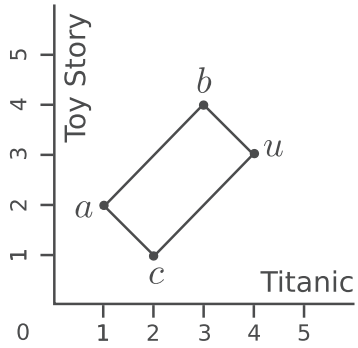

Fig. 1. Four users a, b, c, u that are in proportion.

Table 1
The four users a, b, c, u are in proportion for every item j that they have commonly rated. For an item i that u has not rated, the prediction $\hat{r}_{ui}$ is set as the solution of the analogical equation $2 : 4 :: 3 :?$, i.e. $\hat{r}_{ui} = 3-2+4 = 5$, using the arithmetic proportion.

	j_1	j_2	j_3	$\cdots$	i
a	1	4	3	$\cdots$	2
b	5	2	3	$\cdots$	4
c	1	5	3	$\cdots$	3
u	5	3	3	$\cdots$	?

1. Find the set of 3-tuples of users a, b, c such that an analogical proportion stands between a, b, c, and u and such that the equation $r_{ai} : r_{bi} :: r_{ci} : y$ is solvable.
2. Solve the equation $r_{ai} : r_{bi} :: r_{ci} : y$ and consider the solution y as a candidate rating for r_{ui}.
3. Set $\hat{r}_{ui}$ as an aggregate of all candidate ratings.

In our implementation, we have used the arithmetic proportion in $\mathbb{R}^m$: $a : b :: c : u \iff a - b = c - u$. The four users a, b, c, u are here considered as vectors of ratings in a space defined by their common items. For the sake of clarity though, we will not write them with boldface letters and a will denote both the user a as well as its vectorial representation. Let us consider Fig. 1: the four users a, b, c and u are in proportion, i.e. they make up a parallelogram in the 2-dimensional space of their common items, namely the two movies Titanic and Toy story. If the three users a, b, c have rated a movie i that u has not rated, the estimation of r_{ui} will be such that a, b, c and u still make up a parallelogram, but this time in a 3-dimensional space. This process is also explained in Table 1.

In practice, we may not find any 3-tuple (a, b, c) such that a perfect proportion stands between a, b, c and u. We will thus allow some distortion of shape for the parallelogram $abcu$ by choosing another condition, for example that $\|(a - b) - (c - u)\| \leq \lambda$, where λ is a suitable threshold and $\|\cdot\|$ is any p-norm. The expression $\|(a - b) - (c - u)\|$ is also known as the *analogical dissimilarity* between four vectors a, b, c and u, and indicates how far are the four vectors from being in perfect analogical proportion. Analogical dissimilarity was introduced in [6] for real vectors and for sequences. Let us note that such a view of *approximate* analogical proportion as a skewed parallelogram is closely related to the view of analogy advocated earlier in the works of Rumelhart and Abrahamsen [35].

Our analogical proportion-based algorithm for recommendation is described by Algorithm 1.

We have considered a strict condition for the solvability of the equation $r_{ai} - r_{bi} = r_{ci} - y$: as the exact arithmetic result $y = r_{ci} + r_{bi} - r_{ai}$ does not necessarily belong to the rating scale used in our experiments (which is $[1, 5]$), we have considered that the equation is solvable only when $r_{ai} = r_{bi}$ or $r_{ai} = r_{ci}$. In both cases, we ensure that the solution y is in $[1, 5]$. We also tried another option where an equation is considered solvable when the solution belongs to $[1, 5]$. This option led to extremely close results, where differences were not significant. Note also that it is possible to invert the roles of users and items, and instead look for proportions between items. We will only report our investigations when the proportions are user-based, as the item-based case yields very similar results.

Algorithm 1: Analogical proportion-based algorithm for recommendation.

Input: A set of known ratings R, a user u, and an item i such that $r_{ui} \notin R$.

Output: $\hat{r}_{ui}$, an estimation of r_{ui}.
Init:
$C \leftarrow \varnothing$ // The set of candidate ratings
for all users a, b, c in U_i such that:

- $\|(a - b) - (c - u)\| \leq \lambda$
- $r_{ai} : r_{bi} :: r_{ci} : y$ is solvable

do
$y \leftarrow r_{ci} - r_{ai} + r_{bi}$
$C \leftarrow C \cup \{y\}$ // Add y as a candidate rating
end for
$\hat{r}_{ui} \leftarrow \text{mean}_{y \in C}\ y$

Table 2
Performances of recommendation algorithms on the MovieLens-100k dataset.

Algorithm	Details		RMSE	MAE	Cov	Prec	Rec	F	Time
Parall	Sample	$n = 100$	1.18	.93	57	95	68	79	10 m
		$n = 1000$	1.04	.83	23	95	27	42	1 h
	k-NN	$k = 20$	1.00	.79	31	97	38	54	6 h
		$k = 30$	0.99	.79	25	96	31	47	19 h
k-NN	MSD	$k = 40$	0.98	.76	23	96	28	43	30 s
	Cos	$k = 40$	1.02	.81	21	96	26	41	30 s
	Pears	$k = 40$	1.01	.82	25	95	30	46	30 s
PMF		$f = 100$	0.95	.75	38	99	47	64	45 s
Mean			1.13	.90					1 s
Random			1.52	1.22	81	86	89	88	1 s

In the next section, we will check if our analogical recommender can compete with other standard recommendation algorithms. The goal here is not necessarily to show that analogical reasoning can outperform other well-optimized methods, but rather to establish that it is still a relevant way of dealing with recommendation.

4.2. Experiments and results

The performances of our analogical recommender are summed up in Table 2. Various options were considered, that we will describe in a moment. We also report the performances of other standard recommendation algorithms that we have already mentioned:

- The basic neighborhood approach (denoted k-NN) with different similarity metrics, namely Mean Squared Difference, Cosine similarity and Pearson similarity. For these three algorithms, the size of the neighborhood was set to $k = 40$.
- The matrix-factorization algorithm (denoted PMF), as described in Section 2.4. The number of factors was set to $f = 100$, and the optimization problem was solved by a stochastic gradient descent of 20 iterations with a constant learning rate of 0.005 and a regularization penalty of 0.02. These values are those recommended by the authors in [20] for the Netflix dataset, and turned out to be quite efficient for our experiments.
- For the sake of completeness we also report results for an algorithm that always predicts the average of all ratings ($\hat{r}_{ui} = \mu$ for all u, i). In addition, we also give the performances of a random algorithm that predicts random ratings based on the distribution of the dataset, which is assumed to be normal.

The reported metrics are RMSE, MAE, precision, recall, F-measure, and coverage. Precision, recall, F-measure and coverage are reported as percentages. Remember that for these dimensions high values mean high performance,

while for RMSE and surprise low values are better. RMSE and MAE are the only measure that only depends on the prediction algorithm. All the other dimensions depend on the items that we actually choose to recommend, and rely on the recommendation strategy S. Here, we have chosen to recommend i to u if $\hat{r}_{ui} \geq 4$.

All reported results are averaged over a 5-fold cross-validation procedure. Obviously, the 5 folds are the same for all of the algorithms, to allow for meaningful comparisons. The dataset that we used is the MovieLens-100K dataset,[2] composed of 100,000 ratings from 1000 users and 1700 movies. Each rating belongs to the interval $[1, 5]$, and the sparsity of this taste is of about 94%, i.e. only 6% of all possible ratings are actually known. We also report the computation time of each algorithm (roughly estimated), which is not an average but the total over the 5 folds. All our experiments have been carried out using Surprise [36], a Python recommendation library that we have developed specifically for the occasion.

We have considered various alternatives for our analogical recommender. In fact, when trying to predict a single rating r_{ui}, looking for *all* the 3-tuples of users in U_i as described in Algorithm 1 is simply impractical[3]: there are often too many users in U_i. As the complexity of this search is in $\mathcal{O}(\mid U_i \mid^3)$ and as $\mid U_i \mid$ can be quite large for some items, strictly sticking to Algorithm 1 would lead to even worse computation times. We thus have chosen two different strategies:

- The first one is to randomly sample n times a 3-tuple in U_i^3. As in the strict version of Algorithm 1 the prediction is an average from all the 3-tuples in U_i^3, choosing a large value of n should lead to a fairly good estimate. We have reported the results for $n = 100$ and $n = 1000$. Note though that $\mid U_i \mid^3$ is usually much larger than 1000, so we only have a very rough estimation here. This option is referred to as *Parall Sample*.
- The second strategy is to consider the 3-tuples (a, b, c) in the neighborhood of u, using the assumption that the neighbors of u are probably more reliable to yield a prediction where u is involved. We have used the MSD metric to compute the neighborhood, and we have considered various sizes of neighborhood, namely $k = 20$ and $k = 30$. Unfortunately, values of k greater than 30 lead to basically never-ending algorithms. This option is referred to as *Parall k-NN*.

The λ threshold has been set using a grid search strategy on each of the 5 folds. Note that this does not positively bias the results for the analogical algorithms, as the test fold is never used for training or for grid search. Empirically, we observed that the algorithms was not really sensitive to small shifting of the λ parameter.

RMSE analysis. Out of all the algorithms, the PMF algorithm is by far the most accurate, with an RMSE of 0.95. Note however that even though the matrix factorization algorithms cannot be overlooked when it comes to performance comparison, their philosophy remains quite different from that of other classical neighborhood approaches. They tend to model data structures at a high-level of abstraction, while neighborhood methods tend to model local features of the data. As such, it makes more sense to compare the performances of analogical recommenders to those of neighborhood-based techniques, rather than use matrix factorization-based models as a baseline.

As expected, the RMSE of the Parall Sample algorithm gets better while n grows, but still remains quite far from that of the k-NN algorithms. By paying attention to the Parall k-NN algorithms, we see that looking for the 3-tuples in the neighborhood of the users improves the accuracy of our analogy-based algorithms. Their RMSE is better than that of the neighborhood approaches when using Cosine and Pearson similarity, and it may be expected that using a neighborhood size of $k = 40$ would lead to an RMSE close to that of the k-NN algorithm with MSD similarity.

Coverage. Let us now consider the coverage measure. At first sight, it may appear that the Random and Parall Sample 100 algorithm have the best recall values. But this would be missing a very important detail: any measure that evaluates the fitness of the recommendation strategy S (such as the coverage) greatly depends *also* on the fitness of the prediction algorithm A, simply because S highly depends on A. Therefore, the coverage of the Random algorithm cannot be taken very seriously, because its accuracy (RMSE) is disastrous: it is only by chance that some items were estimated with a

[2] http://grouplens.org/datasets/movielens/.

[3] Remember that U_i is the set of users that rated item i.

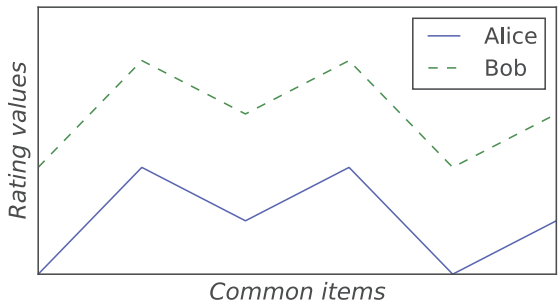

Fig. 2. Bob is a perfect clone of Alice, and Alice is a perfect clone of Bob.

rating greater than 4, and were then recommended. The same goes for the Parall Sample 100 algorithm, which has a quite bad accuracy. Actually, the most reasonable choice would probably be the PMF algorithm again, which has the best RMSE and very decent coverage. With respect to coverage, our other analogy-based algorithms yield comparable performances to the neighborhood-based methods, with a slight advantage for analogy-based methods.

Precision and recall. As for precision, recall and F-measure, we somehow have the same situation: the Random algorithm seems to be the best one, but this needs to be taken very carefully. Here again, the PMF algorithm probably yields the best actual trade-off. Comparatively, analogy-based algorithms have a slightly better F-measure than the neighborhood approaches. This may describe the fact that our Parall algorithms tend to produce more diverse recommendations, as would also suggest the results on the coverage.

Other remarks. As a side note, we have reported only the RMSE for the Mean algorithm, because this algorithm always outputs a prediction of $\hat{r}_{ui} = \mu = 3.53$ which is lower than the threshold we have chosen for our recommendation strategy S. Therefore, not a single item is recommended with this algorithm, and the precision, recall, etc. are not defined (or are null).

We are now led to computation time. All of the other algorithms can manage through the 5-folds cross-validation procedure in less than a minute, but our analogy-based algorithms need hours to yield only decent performances. This issue is linked to the cubic complexity of the analogy-based learners, which cannot cope with big amounts of data. For now, this limitation prevents analogy-based learners to be relevant in real-world applications such as recommender systems, where computation time is one of the most decisive factor.

Nonetheless, it is still possible to design analogy-inspired solutions for recommendation. In the next section, we describe other algorithms that rely on the concept of *clones* and that generalize the classical neighborhood approach.

5. A "clone"-based view of analogical recommendation

Considering analogies between four users has shown to be computationally intensive, thus not really suitable for recommendation purposes, where time is a highly critical dimension. Yet, other forms of analogy can be addressed in the recommendation task, based on the observation that some users may be more inclined to give good (or bad) ratings than others. Indeed, ratings are in no way absolute and greatly depend on the subjective appreciation each user has about the rating scale. In the $[1, 5]$ scale for example, two users u and v might semantically agree on an item i describing it as *bad*, but there is a chance that this agreement is not perfectly reflected in the ratings: u might have rated i with $r_{ui} = 1$ and v with $r_{vi} = 3$, simply because from v's point of view 3 is a *bad* rating, while for u a rating of 3 would simply mean *decent* or *good enough*. In the following, we refer to such users that *semantically* agree on their common items (but not necessarily *numerically*) as *clones*, as illustrated in Fig. 2. Please note that the word *clone* is not used here to mean *strictly identical*, but rather in the sense that two clones are two users following parallel paths: their ratings are aligned.

It is obvious that in collaborative filtering, clones are of great interest when it comes to predicting a user's ratings, and yet the information they provide are often discarded. Indeed, in Fig. 2, Alice and Bob would definitely not be considered as neighbors, so Bob would not be used to predict Alice's ratings, and Alice would not be used to predict Bob's ratings. The principle underlying the analogical clone-based view is the following: for predicting a missing rating for u, we not only look at its nearest neighbors but also at the users v whose ratings are such that $r_{ui} = r_{vi} + t_{vu}$

where t_{vu} is a constant (up to some bounded variations) *correction term* that can be either positive or negative. This correction term is the difference between Bob's ratings and those of Alice. When two users u and v are clones, we can come back to an analogical proportion-based viewpoint by noticing that we have:

$$r_{ui} : r_{vi} :: r_{uj} : r_{vj}, \quad r_{uj} : r_{vj} :: r_{uk} : r_{vk}, \ldots$$

where $i, j, k, \cdots$ are the common items rated by u and v. The algorithms we will propose are more efficient than those of the previous section (in terms of computation time), because they will not rely on an extensive search of 3-tuples of users.

5.1. Two new analogical algorithms

In the following, $C_i(u)$ will denote the set of users that are clones of u and that have rated item i. From the previous informal definitions, one can easily derive a very general collaborative filtering framework for predicting a user's rating by taking into account its clones:

$$\hat{r}_{ui} = \underset{v \in C_i(u)}{\text{aggr}} \ (r_{vi} + t_{vu}),$$

where t_{vu} is a *correction term* that we need to add to v's ratings so that they correspond to those of u, and $aggr$ is any of the candidate aggregation functions previously mentioned (weighted mean, majority vote, etc.). We clearly have a generalization of the neighborhood approach defined in Section 2.4, which could be rewritten as:

$$\hat{r}_{ui} = \underset{\substack{v \in C_i(u), \\ t_{vu}=0}}{\text{aggr}} \ (r_{vi} + t_{vu}).$$

Following this general framework, one can construct a great variety of algorithms with various levels of complexity. In the next subsections, we will propose a very straightforward algorithm, and a more efficient one.

5.1.1. A straightforward prediction algorithm

We will here design a very basic clone-based algorithm, to show that even a basic method can outperform the classical neighborhood approach.

Let us first introduce the notion of t-clone. In its most simple form, a user v can be considered to be a t-clone of u if the ratings of v exactly differ from those of u by a constant term t:

$$t\text{-}C(u) \stackrel{\text{def}}{=} \{v \in U | \forall i \in I_{uv}, \ r_{ui} = r_{vi} + t\}.$$

From then on, computing $\hat{r}_{ui}$ amounts to finding all the users v that satisfy this criterion, and computing an aggregation of their ratings for i, which can simply be an average. We implemented this basic algorithm described by Algorithm 2, and referred to as *brute-force*.

Of course, one may want to relax the definition of a t-clone, as the current one is too strict and only very few users will satisfy this criterion. In our implementation, we chose the following condition:

Algorithm 2: A brute-force algorithm for clone-based recommendation.

Input: A set of known ratings R, a user u, an item i such that $r_{ui} \notin R$.
Output: $\hat{r}_{ui}$, an estimation of r_{ui}.
Init:
$C \leftarrow \varnothing$ // list of candidate ratings
for all users $v \in U_i$ **do**
  for all t **do**
    if $v \in t$-Clones(u) **then**
      $C \leftarrow C \cup \{r_{vi} + t\}$ // add $r_{vi} + t$ as a candidate rating
    end if
  end for
end for
$\hat{r}_{ui} \leftarrow \underset{c \in C}{\text{aggr}}\, c$

$$t\text{-}C(u) \stackrel{\text{def}}{=} \left\{ v \in U \;\middle|\; \sum_{i \in I_{uv}} |(r_{ui} - r_{vi}) - t| \leq | I_{uv} | \right\},$$

which amounts to accept v as a t-clone of u if on average, the difference $|r_{ui} - r_{vi}|$ is equal to t with a margin of 1. The values of t clearly depend on the rating scale. The datasets on which we tested our algorithms use the $[1, 5]$ interval, so possible values for t that we have considered are integer values in $[-4, 4]$.

This is obviously a very rough algorithm, to which one could point out numerous flaws. The first obvious one is its time complexity which is very high: it is about $\mathcal{O}(|U| \cdot T \cdot |I|)$, where T is the number of values that t can take. But the purpose of this brute-force algorithm is simply to show that even such a basic clone-based approach can lead to better results than a basic neighborhood method, as we will see in the experiments section.

5.1.2. Modeling clones with the similarity measure

Another option to consider clones is to use the well known neighborhood-based formula, and capture their effects using an appropriate similarity measure. Recall that the general neighborhood formula is as follows:

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} r_{vi} \cdot \text{sim}(u, v)}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)}.$$

We have seen that this formula is commonly used with classical similarity metrics such as Pearson similarity, Cosine similarity, or inverse of MSD. However, these similarities are not fully satisfactory when it comes to clones. Indeed with these metrics, two users are considered to be close if their common ratings are often the same, but two perfect clones u and v with a significant correction term t_{vu} would be considered as being far from each other, thus involving a loss of information.

We propose the following simple choice of metric to measure how two users relate as clones:

$$\text{Clone\_dist}(u, v) \stackrel{\text{def}}{=} \frac{1}{| I_{uv} |} \cdot \sum_{i \in I_{uv}} [(r_{ui} - r_{vi}) - \mu_{uv}]^2,$$

where μ_{vu} is the mean difference between ratings of u and v:

$$\mu_{uv} \stackrel{\text{def}}{=} \frac{1}{| I_{uv} |} \sum_{i \in I_{uv}} (r_{ui} - r_{vi}).$$

We can understand this distance in two ways:

- it can be regarded as the variance of the difference of ratings between u and v,
- or it can be regarded as a simple MSD measure (defined in Section 2.4) to which the mean difference of ratings between u and v has been subtracted.

As our measure Clone_dist is a distance, it needs to be turned into a similarity measure. A common choice is to take its inverse (while accounting for zero division):

$$\text{Clone\_sim}(u, v) = \frac{1}{\text{Clone\_dist}(u, v) + 1}.$$

Once we know how to find the clones of a user, it is easy to output a prediction using the classical neighborhood approach, now extended to these clones:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} (r_{vi} + \mu_{uv}) \cdot \text{Clone\_sim}(u, v)}{\sum_{v \in N_i^k(u)} \text{Clone\_sim}(u, v)}.$$

This algorithm will be referred to as *Clone*. For the sake of completeness, we also tried the same formula but with a more basic similarity metric that does not care about clones: MSD.

5.2. Current advances in neighborhood-based techniques

What we have seen so far in terms of neighborhood methods are the rough, basic techniques that have existed for a long time. Actually, more sophisticated approaches have been developed, in particular during the Netflix competition. The one we will describe here has been popularized in [37], and makes use of *baseline predictors*:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} (r_{vi} - b_{vi}) \cdot \mathrm{sim}(u, v)}{\sum_{v \in N_i^k(u)} \mathrm{sim}(u, v)},$$

where b_{ui} is a baseline (or bias) related to user u and item i. Its expression is $b_{ui} = \mu + b_u + b_i$, where b_u is supposed to model how u tends to give higher (or lower) ratings than the average of all ratings μ, and b_i is supposed to model how i tends to be rated higher or lower than μ. For example, if the mean of all ratings is $\mu = 3$, and the ratings of a user are $(2, 2, 1)$, its bias b_u would be close to -1.

Baselines are computed by solving a regularized least squares problem:

$$\min_{b_u, b_i} \sum_{r_{ui} \in R} [r_{ui} - (\mu + b_u + b_i)]^2 + \lambda \left(b_u^2 + b_i^2 \right),$$

which can be achieved efficiently by stochastic gradient descent, or alternating least squares. The regularization terms are here to avoid overfitting: they allow to give more confidence to biases that are computed on a high number of ratings. In our previous example, the user had only rated 3 items so we cannot reliably say that its real bias is close to -1. The regularization term will allow to give a value closer to 0 for this user bias.

In their work, the authors have used this particular similarity metric, that is in perfect accordance with their prediction formula:

$$\mathrm{sim}(u, v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - b_{ui}) \cdot (r_{vi} - b_{vi})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - b_{ui})^2} \cdot \sqrt{\sum_{i \in I_{uv}} (r_{vi} - b_{vi})^2}}.$$

It is simply a Pearson correlation coefficient, except that instead of centering ratings with their averages, they are centered with the baseline predictors. This seemingly simple tweak actually has noteworthy consequences. An intuitive and illuminating way to look at this algorithm as a whole is to see that it conceptually follows these steps:

1. Compute R', the set of all ratings normalized by the corresponding baselines: $r'_{ui} = r_{ui} - b_{ui}$. R' can be regarded as the set where all ratings are given from the same frame of reference (in this case 0), thus discarding any bias coming from the users or from the items. In R', ratings can then be considered as absolute, in the sense that they are not spoiled by the users' moods or the items inherent popularity.
2. Using R', compute similarities between users using the Cosine similarity (the Cosine similarity is the same as the Pearson correlation coefficient, except that quantities are not centered).
3. Output a prediction using the basic neighborhood formula. As this prediction belongs to the same space of R' where ratings have no bias, it needs to be transposed back to the space of R for performance evaluation purposes. This is why b_{ui} and b_{vi} are added (or subtracted) in the prediction formula of $\hat{r}_{ui}$.

In what follows, this algorithm will be referred to as k-NN*.

It is very clear that the use of the baseline predictors and the use of clone-based recommendation are motivated by the same reason: they both come from the fact that users (and items) tend to interpret the rating scale differently. This means that k-NN* implicitly takes the idea of clones into account, and thus a form of analogical reasoning. Differences and resemblances of these two approaches will be discussed in the next section.

5.3. Experiments and discussion

To assess the suitability of our clone-based view of recommendation, we have evaluated the accuracy of the brute-force and Clone algorithms, and compared them to the previously mentioned approaches: the basic neighborhood

Table 3
RMSE and MAE of our clone-based algorithms on the MovieLens-100k and 1 M datasets.

			ML-100k		ML-1 M	
Algorithm	Details		RMSE	MAE	RMSE	MAE
k-NN	MSD	$k = 40$	.979	.773	.921	.725
Brute-Force			.948	.737		
Clone	Clone_sim	$k = 40$	.936	.733	.899	.705
	MSD	$k = 40$	.931	.732	.897	.707
k-NN*		$k = 40$	.921	.721	.869	.680

method (k-NN), and the neighborhood method taking into account user and item biases (k-NN*). The evaluation protocol is the same as that of Section 4.2, i.e. results are averaged over a 5-folds cross-validation procedure. In addition to the MovieLens-100k dataset presented earlier, we also used the MovieLens-1 M dataset containing 1 million ratings with 6000 users and 4000 movies. For each of these algorithms, the number of neighbors or clones used to output a prediction is $k = 40$, except for the brute-force algorithm where the number of clones cannot be controlled. The RMSE and MAE of the algorithms are reported on Table 3. We have chosen not to add other metrics as in Table 2, as they do not bring much to the discussion.

It is clear that even a very straightforward approach of the clone-based recommendation principle outperforms the most basic k-NN algorithm, and thus validates the need to take into account biases between users. The brute-force is however a lot heavier to compute ($\mathcal{O}(\mid U \mid^3)$ as printed out in Section 4), and thus not very suitable for real world recommendation purposes (its performances on the MovieLens-1 M dataset simply could not be computed). The two other clone-based algorithms, however, have the exact same complexity of any k-NN-based algorithm ($\mathcal{O}(\mid U \mid^2)$), which is a important improvement from the algorithm described in Section 4.

Our two Clone algorithms output (almost) exactly the same accuracies. This may seem a bit surprising, because the Clone MSD algorithm does not take into account clones in the similarity measure, and we would expect it to yield a lower accuracy. But this result still gives a further reason to consider clones as useful predictors: the only difference between the Clone MSD algorithm and the k-NN MSD algorithm (whose accuracy is much worse) is that in the prediction of Clone MSD, the mean differences μ_{uv} between the ratings of u and its neighbors are taken into account. If this (seemingly) simple change can make such a large difference in the accuracies, this means that the way users relate as clones is an important feature of the dataset and should not be ignored.

Performances of the Clone algorithms are close to those of the state of the art k-NN* algorithm, yet the difference is more striking on the MovieLens-1 M dataset than on MovieLens-100K. It is however important to understand that these algorithms differ on the following points:

- The Clone algorithms do not address item bias, which is a serious drawback. It is not unreasonable to believe that incorporating item bias in the prediction would lead to better results.
- There is a subtle yet meaningful difference of interpretation between the biases induced by both algorithms. In the Clone algorithm, biases are all pairwise, meaning that they involve two users, and they are computed on items that both users have rated. As for the k-NN* algorithm, there is no such thing as a pairwise bias. Bias for a given user is computed using only its own ratings, and is a result of a global optimization problem involving the global mean of all ratings, which means that every single rating in R has an impact on the bias. As baselines are computed on the whole training set, they tend to capture most of the noise when the training set gets bigger. This may explain why the difference between the Clone algorithms and k-NN* is more striking on the MovieLens-1 M dataset.

This closes for now our contributions to rating prediction. We will now focus on the mining of analogical proportions in rating databases (such as MovieLens). As we will see, mining analogical proportions may be a way of checking the relevance and the suitability of analogical methods for recommendation, in addition to the illustrative interest of exhibiting proportions that hold in a dataset.

6. Mining analogical proportions

In Section 4, we described our first attempt to design an analogical-based prediction algorithm. This algorithm relied on finding proportions between users. But we have so far neglected some basic questions: can we find enough 4-tuples of users such that their common ratings are in proportion? And if we can, how good are these proportions? We already acknowledged that absolutely perfect proportions are actually hard to find, and this is why we have relaxed the condition for an analogy to hold in Algorithm 1.

In this section, we will design an algorithm that is able to answer these questions, by extracting all the analogical proportions underlying a database that satisfy some given quality criterion. We will apply this algorithm to the MovieLens database, and we will look for analogies between items (movies in our case) instead of looking for analogies between users, because the interpretation of a movie-based proportion is actually easier, and because the parallel with association rules (described below) will be more natural. Nevertheless, both problems are symmetric, and our method can be adapted to user proportions in a straightforward manner.

Let us mention that the mining of analogical proportions has already been addressed in e.g. [38], though in a different context. This work takes place in the setting of relation databases without any missing values and relies on clustering techniques for pairing tuples of vectors and then building analogical proportions. In contrast, we will look for analogies in a rating dataset, which can be viewed as a very sparse matrix.

Because our method for mining analogical proportions is inspired from the mining of association rules, we will first review this topic in the next subsection.

6.1. Association rules

Association rules are pieces of information that one can extract from a database, and that reveal dependencies between items. Recommendation is one of the principal application of association rules. Starting from the association rule $i \implies j$, which means that users that buy i tend to buy j with high probability, a recommendation system can suggest j as soon as we bought i (but not yet j). A well-known example of association rule is the famous beer $\implies$ diapers association, which was revealed after mining association rules in a supermarket selling history, suggesting that people tend to buy beer and diapers altogether. We now formally introduce the problem of *association rule mining*.

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of m items, and let $T = \{t_1, t_2, ..., t_n\}$ be a multiset of transactions, where each transaction is a subset of I: $\forall i,\ t_i \subseteq I$. A transaction simply is a set of items that are purchased together. An association rule is expressed in the form $X \implies Y$, where $X \subset I,\ Y \subset I$, and $X \cap Y = \varnothing$. X and Y are sets of items that we will call *itemsets*, and usually Y is restricted to a single item. We define the *support* $\text{supp}(X)$ of an itemset X as the proportion of transactions that contain it:

$$\text{supp}(X) \stackrel{\text{def}}{=} \frac{|\ \{t \in T | X \subseteq t\}\ |}{|\ T\ |}.$$

Sometimes, the support is not defined as a proportion but rather as the absolute number $|\ \{t \in T | X \subseteq t\}\ |$. Various measures can be used to evaluate the quality of an association rule, such as the *confidence* which can be expressed as:

$$\text{Conf}(X \implies Y) \stackrel{\text{def}}{=} \frac{\text{supp}(X \cup Y)}{\text{supp}(X)}.$$

As could be naturally expected, $\text{Conf}(X \implies Y)$ is equal to 1 when the items of X and Y are systematically bought together, and decreases if the set X is sometimes found in a transaction that does not contain Y.

The mining of association rules is a popular research topic, and has been extensively studied. The most famous algorithm for association rule mining probably is the *Apriori* algorithm introduced in [39], that we will briefly review. Using Apriori allows to scan the itemset lattice in an efficient way, avoiding many useless nodes.

Ultimately, we are only interested in association rules where the support of the involved itemsets is high. An itemset whose support is above some given threshold α is called a *frequent* itemset. The downward-closure property of support states that if I_S is a frequent itemset, then all of its subsets are also frequent itemsets. Taking advantage of this fact, a basic version of the Apriori can be described in the following steps:

1. Consider all itemsets of size 1 whose support is above α.
2. By joining these 1-itemsets, build all possible 2-itemsets and only keep those whose support is above α.
3. By joining these 2-itemsets, build all possible 3-itemsets and only keep those whose support is above α.
4. Repeat the process: all the frequent k-itemsets are built by joining the frequent $k-1$ itemsets.
5. Once all the frequent itemsets have been computed compute all partitions $\{X, Y\}$ of each frequent itemset and calculate the confidence associated with the rule $X \implies Y$. If the confidence is above a given threshold, then keep the association rule, else discard it.

In the end, we are provided with a set of association rules that comply with some quality requirements, and that give us insightful information that link the elements of our database.

Just like association rules, the identification of analogical proportions in a database is an additional information source, and deserves to be addressed. We will now get to the heart of the matter.

6.2. Looking for analogies in an incomplete database

In order to avoid any ambiguity, we will formally define our problem first.

6.2.1. Problem definition and links to association rules

We are here looking for analogies between items in the MovieLens database, but our method can naturally be extended to any other database with the same structure. We dispose of a set of users U and a set of items I. Each user u has rated (or purchased) a given set of movies $I_u \subseteq I$. In this setting, each I_u can be considered as a transaction t, as defined in the previous section (items play the same role as before). We are then in the exact same setting as that of association rules mining, and we will use the Apriori algorithm to find 4-itemsets with a minimal support threshold, which will help us find relevant proportions between such items.

Just like with association rules, the notion of support of an itemset I_S remains, and can be defined as the proportion of users that have rated all the items in I_S.

$$\text{supp}(I_S) \stackrel{\text{def}}{=} \frac{|\{u \in U | I_S \subseteq I_u\}|}{|U|}.$$

The Apriori algorithm will filter itemsets with low support: the underlying idea is that we prefer finding proportions involving four items that have a lot of common raters.

Apriori will provide us with all 4-itemsets that satisfy a support criterion (which are all potential proportions), but we now need a way to assess the quality of a proportion. To do so, we will use a function f that plays a similar role to the confidence function for association rules. Then, it will be natural to only consider analogies that are above some given quality threshold. Simply put, we would consider $a : b :: c : d$ as a valid analogy if $f(a : b :: c : d) \geq \beta$.

In our experiments, the quality function that we will use is the definition of analogy in the multi-valued case, described Section 3.1:

$$A(a, b, c, d) = \begin{cases} 1 - |(a - b) - (c - d)| & \text{if } a \geq b \text{ and } c \geq d, \text{ or } a \leq b \text{ and } c \leq d \\ 1 - \max(|a - b|, |c - d|) & \text{if } a \geq b \text{ and } c \leq d, \text{ or } a \leq b \text{ and } c \geq d. \end{cases}$$

This function A evaluates the quality of a proportion between scalar values. We will instead deal with proportion between vectors. Hence, to assess the quality of a vector proportion, we can choose various kinds of aggregations for the component-wise qualities: mean, max, min, or also compare two proportions by lexicographic order. We will give further details in the experiments section.

Note that when Apriori provides us with a 4-itemset (a, b, c, d), we need to decide if the proportion that we will evaluate is $a : b :: c : d$, or $a : b :: d : c$, or any of the 24 (4!) combinations of these four elements. Fortunately, we do not have to test all the 24 orderings. We know from Section 3.1 that there are exactly 3 equivalent classes of analogies, which are represented by:

Algorithm 3: Analogical proportion mining.

Input: A set of known ratings R, a quality function f, and two thresholds α and β.
Output: A set $\mathcal{P}$ of analogical proportions between items.
$\mathcal{P} \leftarrow \varnothing$
Candidate retrieval:
Using Apriori, derive all the 4-itemsets whose support is greater than α.
Quality evaluation:
for all (a, b, c, d) in the set of 4-itemsets **do**
  for all prop $\in \{(a : b :: c : d), (a : b :: d : c), (a : d :: c : b)\}$ **do**
    if f(prop) $\geq \beta$ **then**
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{\text{prop}\}$
    end if
  end for
end for

- $a : b :: c : d$,
- $a : b :: d : c$,
- $a : d :: c : b$.

Thus, testing these three orderings is enough to find out about the 24 possible forms of analogies.

6.2.2. Algorithm

Our algorithm for mining analogical proportions imitates an association rule mining process: we first set a threshold α for the support, and a quality evaluation f along with a threshold β.

After having built all the 4-itemsets whose support is above the threshold α with the Apriori algorithm, we compute the quality of the proportions associated with the three equivalent classes, and keep those that satisfy our criterion. These steps are described in Algorithm 3.

In theory, it is possible to end up with two non-equivalent proportions in $\mathcal{P}$ that still relate to the same four items, i.e. we could find in $\mathcal{P}$ the proportion $a : b :: c : d$ as well as the non-equivalent proportion $a : b :: d : c$. It should not seem natural to have these two proportions in $\mathcal{P}$, so if this happens, it is probably because the quality function f is too permissive or because the threshold β is not correctly tuned.

We also want to stress the point that the actual rating values are only used in the second part of the algorithm, i.e. when we evaluate the quality of the proportions. In the first part involving the Apriori algorithm, the only information that matter are that a rating exists. Its value is not taken into account.

6.3. Experiments and discussion

As previously indicated, we have considered the MovieLens-100k dataset for our experiments: 100,000 ratings in $[1, 5]$ from 1000 users and 1700 movies.

For our purpose, we actually do not need to set a quality threshold β, which would by the way be quite arbitrary. Instead, we will only be interested in *comparing* the quality of the proportions. We have chosen to compare two proportions by first computing the truth value of each of their component-wise proportions using the graded-view of analogical proportions described in Section 3.1, and then by comparing these truth values in lexicographic order. This allows to compare proportions that have different dimensions in a more meaningful way.

Using this comparison procedure, we have looked for the 10 best proportions with a minimum support of 200 common users. The results are reported on Table 4.

The first obvious observation is that only a few movies (exactly 7) make up these 10 proportions. This is not really surprising, as we have set the support threshold quite high, and only a few movies are rated by more than 200 users. In fact, we found 331 4-itemsets with more than 200 common users, but they only contain 37 unique movies. Clearly here the involved movies (Star Wars, Indiana Jones…) are extremely popular, and people tend to give them high ratings. Using notions introduced in Section 5.2, we can say that these movies have a high item bias b_i. To confirm this claim, we can check that in average, the mean rating of these 7 movies is of 4.09, while the average rating of all movies is only 3.53.

Table 4
The ten best item proportions with a support of more than 200 common ratings, using A.

	i_1	i_2	i_3	i_4
1	Star Wars	The Emp. Strikes Back	Raid. of the Lost Ark	Return of the Jedi
2	Star Wars	Return of the Jedi	Raid. of the Lost Ark	The Emp. Strikes Back
3	Star Wars	The Emp. Strikes Back	Return of the Jedi	Raid. of the Lost Ark
4	Star Wars	Raid. of the Lost Ark	Return of the Jedi	I.J. and the Last Crus.
5	Star Wars	Return of the Jedi	Raid. of the Lost Ark	The Fugitive
6	Star Wars	Raid. of the Lost Ark	Return of the Jedi	Back to the Future
7	Star Wars	The Emp. Strikes Back	Raid. of the Lost Ark	I.J. and the Last Crus.
8	Star Wars	The Emp. Strikes Back	Return of the Jedi	I.J. and the Last Crus.
9	Star Wars	The Emp. Strikes Back	I.J. and the Last Crus.	Return of the Jedi
10	Star Wars	Raid. of the Lost Ark	The Emp. Strikes Back	The Fugitive

Some of the analogies are actually quite good though. For example the fourth one *Star Wars (1977)* is to *Raiders of the Lost Ark (1981)* as *Return of the Jedi (1983)* is to *Indiana Jones and the Last Crusade (1989)*, or the seventh one which is similar: *Star Wars (1977)* is to *The Empire Strikes Back* as *Raiders of the Lost Ark (1981)* is to *Indiana Jones and the Last Crusade (1989)*

Looking at the first and third proportions, we see that the two forms $i_1 : i_2 :: i_3 : i_4$ and $i_1 : i_2 :: i_4 : i_3$ are present in the table. These two forms are non-equivalent, and it does not seem natural to consider these two proportions as (almost) equally valid. But this result can be explained by looking at the actual rating values: a great number of them are of the form $r : r :: r : r$ (where r is usually a high rating, given that these movies are popular). In the two cases, a switch between i_3 and i_4 has absolutely no effect on the truth values of the analogies.

This leads us to another point: all these analogies are, after all, quite trivial. They all concern quite similar movies. We do not mean to offend any cinema fan, but we believe it is still fair to say that the differences between a Star Wars movie and an Indiana Jones movie are quite shallow. And these similarities are reflected in the ratings, where the most common pattern is $r : r :: r : r$ (or close to it), suggesting that people like all these movies the same.

To try to discover some more surprising analogies, we basically have two options. The first one is to tweak a bit our quality function f (and in our case, our comparison function) by penalizing proportions with the pattern $r : r :: r : r$, or inversely by promoting the proportions $r : r' :: r : r'$ where r and r' are quite different. The second option is to lower the minimum support threshold, to allow movies that are rated by a reasonable (and not necessarily very high) number of users. As we have seen, movies with a very high number of ratings are likely to lead to trivial proportions precisely because they are popular, and because people like them all equally well in general. If we really want to ban popular movies, we can also set a *maximum* support threshold. This will also have the benefit to limit the number of itemsets that we consider, leading to a huge improvement in computation time. These two options will be explored.

Promoting disagreement in the proportions. Table 5 shows the 10 best proportions with a minimum support of 200, with a tweaked comparison function. Here, the truth value of a component-proportion $r_1 : r_2 :: r_3 : r_4$ is defined by:

$$A'(r_1, r_2, r_3, r_4) = \frac{1}{2}A(r_1, r_2, r_3, r_4) + \frac{1}{4}|r_1 - r_2| + \frac{1}{4}|r_3 - r_4|.$$

We are here promoting proportions where rating values tend to disagree, which should lead to different proportions from that of Table 4. Proportions are still compared using the lexicographic order of all the A' values.

We still have a lot of Star Wars movies (and a few Indiana Jones), but the movies seem more diverse: we now have 11 unique movies, instead of just 7 before. 9 of these proportions comply with the pattern *Star Wars movie* is to *Other Star Wars movie* what *movie A* is to *movie B*. This suggests that we should find strong links underlying the connection between *Movie A* and *Movie B*, because the two Star Wars movies are deeply related. Is *Pulp Fiction* related to *The Fugitive*? Is *Twister* related to *Independence Day*? Is *Fargo* related to *The Silence of the Lambs*? We will not venture to answer these questions, and leave them to the reader's appreciation. All we can say is that some forms of analogy seem to emerge from their respective ratings. One thing we can note however, is that even though all these movies are very popular just like in Table 4, their genres are however quite different: while we have considered Star Wars and Indiana Jones to be of the same kind, we can fairly claim that a Star Wars movie is very far from Fargo, Pulp Fiction or The Silence of the Lambs.

Table 5
The ten best item proportions with a support of more than 200 common ratings, using A'.

	i_1	i_2	i_3	i_4
1	Star Wars	Pulp Fiction	The Empire Strikes Back	The Fugitive
2	Star Wars	Pulp Fiction	Return of the Jedi	The Fugitive
3	Star Wars	Pulp Fiction	The Empire Strikes Back	The Silence of the Lambs
4	Star Wars	Twister	Return of the Jedi	Independence Day
5	Star Wars	Return of the Jedi	The Silence of the Lambs	Fargo
6	Star Wars	The Terminator	Return of the Jedi	Pulp Fiction
7	Star Wars	Pulp Fiction	Return of the Jedi	The Terminator
8	Star Wars	The Fugitive	The Empire Strikes Back	The Silence of the Lambs
9	Star Wars	Pulp Fiction	Return of the Jedi	The Silence of the Lambs
10	Star Wars	Pulp Fiction	Raiders of the Lost Ark	The Silence of the Lambs

Table 6
The ten best item proportions with a support between 10 and 50.

	i_1	i_2	i_3	i_4
1	To Catch a Thief	Laura	Gigi	An American in Paris
2	Nick of Time	It Could Happen to You	Milk Money	Only You
3	To Catch a Thief	An American in Paris	Gigi	Meet John Doe
4	Dangerous Minds	Money Train	Higher Learning	With Honors
5	Judge Dredd	Under Siege 2: Dark Territory	The Shadow	Mortal Kombat
6	Terminal Velocity	Under Siege 2: Dark Territory	Money Train	Drop Zone
7	Judge Dredd	Under Siege 2: Dark Territory	Mortal Kombat	Coneheads
8	To Catch a Thief	An American in Paris	Gigi	Laura
9	To Catch a Thief	Meet John Doe	Gigi	An American in Paris
10	Nick of Time	It Could Happen to You	Only You	Milk Money

Lowering support threshold. Table 6 illustrates the ten best proportions we have found after setting the minimum support at 10, and a maximum support at 50. This drastically reduces the number of 4-tuples that are explored, and avoids any highly popular movie. Now, we find up to 20 unique movies building up these 10 proportions, which is quite an improvement with respect to the previous settings. It seems that our proportions exhibit some sort of clustering: movies of proportions 1, 3, 8 and 9 are all movies from the 40'-50's. Clearly the users that rated the corresponding movies must be some fans of the old movie-making era. Similarly, proportions 2 and 10 are all movies from 1994 or 1995, suggesting a niche. Here again it is not really surprising that we observe the two patterns $a : b :: c : d$ and $a : b :: d : c$. This comes from the fact that these movies are often given similar (and high) ratings. The other proportions involve movies that are still from the 90's and that are given pretty bad reviews from critics: they seem to belong to the *so bad it's good* kind.

7. Conclusion

In this paper we have proposed two frameworks based on analogical proportions for the design of rating prediction algorithms, and also proposed a method for mining analogical proportions in an incomplete database.

The first prediction algorithm, which exploits analogies between users, yields decent accuracy compared to neighborhood methods, but is extremely expensive in terms of computational resources. It is clear that this simple and straightforward application of the analogical inference principle cannot be used in real-world settings.

Acknowledging the fact that each user tends to have a very personal idea of what a good (or bad) rating should be, we introduced the notion of clones which captures how two users tend to agree semantically on a rating. This semantic agreement is not necessarily reflected in the numerical rating values: for a user u, the value 3 may be a good rating, while for v it may be a bad one. We derived two algorithms relying on this clone-based view of the ratings.

We finally proposed an algorithm for the mining of analogical proportions in a partially described database. This general method perfectly applies to the problem of mining analogies between users or items in a rating dataset. Based on the Apriori algorithm, our method is inspired by the mining of association rules and allows us to extract the

potential analogies in an efficient way. It turned out that the best analogies that could be extracted between items were involving users with similar tastes: the four movies of a proportion were mostly neighbors, in the sense that users tend to agree on their respective ratings. As could have been expected, we observed that the symmetric problem of mining analogical proportions between users leads to similar conclusions: the only proportions that could be found were actually of quite poor quality.

This allows us to retrospectively interpret the results of our first analogy-based prediction algorithm. We have seen that its performances were close to that of a neighborhood-based technique, but in the light of what has just been explained we can fairly say that there were no analogies to find in the database, except for those involving four neighbors. So after all, our analogy-based inference principle was reduced to that of the classical k-NN approach. Clearly in that case, paying the price of the cubic search in the set of all users is really not worth it. This is entirely dependent on the database that was used (in this case, the MovieLens dataset), and we may still presume that there may exist a dataset where the use of analogical proportion is beneficial over a simple neighborhood approach.

Moreover, we have essentially discussed in this paper the interest of analogical proportions-based methods for rating prediction in collaborative filtering. The investigations reported here do not preclude the suitability of analogical reasoning in other recommendation tasks. Indeed, as a tool involving both similarities and dissimilarities between objects, analogical proportions might be used for overcoming the lack of novelty and diversity in content-based and case-based methods. In fact, analogical proportions establish bridges between 4-tuples of objects in such a way that these objects are allowed to be all quite different.

Altogether, the reported studies have tried to assess to what extent analogical reasoning is suitable as an underlying inference tool for the task of recommendation. Results show that it indeed makes sense, even if in our experiments results are of limited practical interest, since fully fledged analogical proportions are very difficult to find in such scarce data settings.

Generally speaking, the reported results on the use of analogical proportion-based methods suggest that the formal definition of analogy may be too strict and that a more flexible definition would be worth investigating.

References

F. Ricci, L. Rokach, B. Shapira, P. B. Kantor (Eds.), Recommender Systems Handbook, Springer, 2011.

P. Perny, J.-D. Zucker, Preference-based search and machine learning for collaborative filtering: the “film-conseil” movie recommender system, Inf. Interact. Intell. 1 (1) (2001) 9–48.

D. Dubois, E. Hullermeier, H. Prade, Fuzzy set-based methods in instance-based reasoning, IEEE Trans. Fuzzy Syst. 10 (2002) 322–332.

E. Hüllermeier, D. Dubois, H. Prade, Model adaptation in possibilistic instance-based reasoning, IEEE Trans. Fuzzy Syst. 10 (3) (2002) 333–339.

D. Dubois, E. Hüllermeier, H. Prade, Fuzzy methods for case-based recommendation and decision support, J. Intell. Inf. Syst. 27 (2) (2006) 95–115.

L. Miclet, S. Bayoudh, A. Delhay, Analogical dissimilarity: definition, algorithms and two experiments in machine learning, J. Artif. Intell. Res. 32 (2008) 793–824.

M. Bounhas, H. Prade, G. Richard, Analogy-based classifiers for nominal or numerical data, Int. J. Approx. Reason. 91 (2017) 36–55.

D. Dubois, H. Prade, G. Richard, Case-based and analogical reasoning, in: C. Sierra (Ed.), Festschrift for Ramon López de Mantaras’ 60 years’ birthday, Artificial Intelligence Research Institute, Barcelona, 2012, pp. 59–77.

L. Miclet, H. Prade, Logical definition of analogical proportion and its fuzzy extensions, in: Proc. Annual Meeting of the North Amer. Fuzzy Information Proc. Soc. (NAFIPS), New-York, May, 19-22, IEEE, 2008, pp. 1–6.

H. Prade, G. Richard, From analogical proportion to logical proportions, Log. Univers. 7 (4) (2013) 441–505.

T. Sakaguchi, Y. Akaho, K. Okada, T. Date, T. Takagi, N. Kamimaeda, M. Miyahara, T. Tsunoda, Recommendation system with multi-dimensional and parallel-case four-term analogy, in: Systems, Man, and Cybernetics (SMC), 2011 IEEE Int. Conf, IEEE, 2011, pp. 3137–3143.

S. Bayoudh, L. Miclet, A. Delhay, Learning by analogy: a classification rule for binary and nominal data, in: Proc. Int. Joint Conf. on Artificial Intelligence IJCAI07, IEEE, 2007, pp. 678–683.

W. Correa Beltran, H. Jaudoin, O. Pivert, Analogical prediction of null values: the numerical attribute case, in: Y. Manolopoulos, G. Trajcevski, Kon-Popovska (Eds.), Advances in Databases and Information Systems, 18th East European Conf., ADBIS, Ohrid, Macedonia, Sep. 7–10. Proc, in: LNCS, vol. 8716, Springer, 2014, pp. 323–336.

T. Takagi, M. Sugeno, Fuzzy identification of systems and its application to modelling and control, IEEE Trans. Syst. Man Cybern. 15 (1)(1985) 116–132.

N. Hug, H. Prade, G. Richard, Experimenting analogical reasoning in recommendation, in: F. Esposito, O. Pivert, M. Hacid, Z.W. Ras, S. Ferilli (Eds.), Proc. 22nd Int. Symp. on Foundations of Intelligent Systems (ISMIS’15), Lyon, Oct. 21–23, in: LNCS, vol. 9384, Springer, Oct 2015, pp. 69–78.

N. Hug, H. Prade, G. Richard, M. Serrurier, Analogy in recommendation. numerical vs. ordinal: a discussion, in: 2016 IEEE Int. Conf. on Fuzzy Systems, FUZZ-IEEE 2016, Vancouver, BC, Canada, July 24–29, IEEE, July 2016, pp. 2220–2226.ü

N. Hug, H. Prade, G. Richard, M. Serrurier, Proportions analogiques: créativité et fouille de données, in: Rencontres francophones sur la Logique Floue et ses Applications, Cépaduès, 2016, pp. 105–112.
R. Burke, M. Ramezani, Matching Recommendation Technologies and Domains, Springer US, Boston, MA, 2011, pp. 367–386.
P. Lops, M. de Gemmis, G. Semeraro, Content-based Recommender Systems: State of the Art and Trends, Springer US, Boston, MA, 2011, pp. 73–105.
Y. Koren, R. Bell, Advances in Collaborative Filtering, in: F. Ricci, et al. (Eds.), Recommender Systems Handbook, Springer US, Boston, MA, 2011, pp. 145–186.
A. Felfernig, R. Burke, Constraint-based recommender systems: technologies and research issues, in: Proc. of the 10th Int. Conf. on Electronic Commerce, CEC '08, ACM, New York, NY, USA, 2008, pp. 1–10.
A. Felfernig, G. Friedrich, D. Jannach, M. Zanker, Developing Constraint-based Recommenders, in: F. Ricci, et al. (Eds.), Recommender Systems Handbook, Springer US, Boston, MA, 2011, pp. 187–215.
R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: NIPS Proceedings: Advances in Neural Information Processing Systems 20, 2007, pp. 1257–1264.
R. Bell, Y. Koren, Scalable collaborative filtering with jointly derived neighborhood interpolation weights, in: Data Mining, ICDM 2007, Seventh IEEE Int. Conf., IEEE, 2007, pp. 43–52.
F. Yvon, N. Stroppa, Formal models of analogical proportions, Tech. Rep. D008, Ecole Nationale Supérieure des Télécommunications, Paris, 2006.
Y. Lepage, De l'analogie rendant compte de la commutation en linguistique, Habilitation à diriger des recherches, Université Joseph-Fourier - Grenoble I, https://tel.archives-ouvertes.fr/tel-00004372, May 2003.
L. Miclet, H. Prade, Handling analogical proportions in classical logic and fuzzy logics settings, in: European Conf. on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Springer, 2009, pp. 638–650.
H. Prade, G. Richard, Analogical proportions and multiple-valued logics, in: European Conf. on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Springer, 2013, pp. 497–509.
H. Prade, G. Richard, Homogenous and heterogeneous logical proportions, IfCoLog J. Log. Appl. 1 (1) (2014) 1–51.
H. Prade, G. Richard, Multiple-valued logic interpretations of analogical, reverse analogical, and paralogical proportions, in: Proc. 40th IEEE Int. Symp. on Multiple-Valued Logic, IEEE Computer Society, Barcelona, 2010, pp. 258–263.
D. Dubois, H. Prade, G. Richard, Multiple-valued extensions of analogical proportions, Fuzzy Sets Syst. 292 (2016) 193–202.
H. Prade, G. Richard, Reasoning with logical proportions, in: F. Lin, U. Sattler, M. Truszczynski (Eds.), Principles of Knowledge Representation and Reasoning: Proc. of the Twelfth Int. Conf., KR 2010, AAAI Press, Toronto, Ontario, Canada, May 2010, pp. 9–13, 2010, pp. 545–555.
M. Bounhas, H. Prade, G. Richard, Analogical classification: a new way to deal with examples, in: ECAI, in: Frontiers in Artificial Intelligence and Applications, vol. 263, IOS Press, 2014, pp. 135–140.
H. Prade, G. Richard, B. Yao, Enforcing regularity by means of analogy-related proportions-a new approach to classification, Int. J. Comput. Inf. Syst. Ind. Manag. Appl. 4 (2012) 648–658.
D.E. Rumelhart, A.A. Abrahamsen, A model for analogical reasoning, Cogn. Psychol. 5 (2005) 1–28.
N. Hug, Surprise, a Python library for recommender systems, http://surpriselib.com, 2017.
R. Bell, Y. Koren, Lessons from the Netflix prize challenge, SIGKDD Explor. Newsl. 9 (2) (2007) 75–79.
W. Correa Beltran, H. Jaudoin, O. Pivert, A clustering-based approach to the mining of analogical proportions, in: ICTAI, IEEE Computer Society, 2015, pp. 125–131.
R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proc 20th Int Conf on Very Large Data Bases, VLDB '94, Morgan Kaufmann Pub, San Francisco, 1994, 1994, pp. 487–499.