Part IV

Applications

It is through science that we prove, but through intuition that we discover.
Jules Henri Poincaré

Chapter 7

Dynamic recommender ensembles

Hybrid recommender systems – and recommender ensembles as a particular case – have become a very popular strategy for making recommendations, since they help alleviate most of the shortcomings of the individual recommenders combined. They have, however, specific problems such as the need of deciding which information sources should be exploited, which recommenders should exploit each of these sources, and how the combination of recommenders should be configured.

In this chapter we propose a framework to decide how dynamic hybridisation should be balanced, by estimating its expected improvements on individual recommendations. Furthermore, we provide some requirements to decide when to build such hybridisation. Within the spectrum of hybrid recommendation approaches, we focus on those that linearly combine the output from several recommenders, and use different weights for generating a particular aggregation of the individual recommendations. In the standard approach, these weights are typically fixed regardless of the user for which recommendations are produced, or the recommended items. In this context we investigate the use of performance predictors to assign those weights dynamically depending on the target user or item. We evaluate our approach using the predictors proposed in the previous chapter. The results obtained show that the generated dynamic ensembles are capable of outperforming their static counterparts. Furthermore, they also show that dynamic ensembles can be improved if predictors with stronger predictive power (higher correlation values as observed in the previous chapter) are used.

In Section 7.1 we present and formulate the research problem of recommendation hybridisation. Next, in Section 7.2 we describe our proposed performance prediction framework for dynamic hybrid recommendation. Section 7.3 describes the experiments conducted and provide an overall discussion of the obtained results. Finally, in Section 7.4 some conclusions are given.

7.1 Problem statement

As described in Chapter 2, hybrid recommenders are built by the combination of different recommendation methods. In the simplest and typical case, hybrid recommendations are produced by weighting and summing the utility values output by some recommenders, forming a so called recommender ensemble where an arbitrary number of algorithms of different kinds (content-based, user-based collaborative filtering, item-based collaborative filtering, social-based, demographics-based, etc.) can be combined.

Researchers in Machine Learning have known for long that the combination of classifiers usually achieves better results than each method separately, which is also true in Recommender Systems – the Netflix prize has been a paradigmatic example of this, where all the top classified teams used large recommender ensembles. We focus on weighted hybrid approaches, as an option that begets a simple and general formulation of the dynamic balance of the combined methods R_k by just setting the weights λ_k of each method in the hybrid combination. This approach can be expressed as follows:

$$rat(u,i) = \sum_k \lambda_k * rat_{R_k}(u,i) \text{ s.t. } \sum_k \lambda_k = 1 \tag{7.1}$$

In this chapter we investigate whether the performance predictors proposed in the previous chapter – where we have already found degrees of correlation between the ambiguity (clarity) of the user's preferences and the accuracy of the system's recommendations – can be useful for hybridisation. Specifically, we aim to use these predictors to build **dynamic hybrid recommenders** in such a way that the weight λ_k depends not only on the recommender but also on the current user u, or potentially other variables such as the item i or other available context information. We propose to specify such weights according to the ambiguity of the user's preferences or item's patterns, that is, we aim to use the performance predictors defined in the Chapter 6 to estimate those weights.

In the next section we propose a framework to perform dynamic hybrid recommendation where we use recommendation performance predictors and we analyse different requirements related to the adaptation of such predictors to produce weights in a hybrid recommender combination. After that, three different experiments are presented, where the predictors proposed in Chapter 6 are used as dynamic weights in the combination.

7.2 A performance prediction framework for ensemble recommendation

Let us simplify Equation (7.1) to the case where only two recommenders $R1$ and $R2$ are used. In this situation, only one weighting factor λ is needed (because of the constraint for the weights to sum to one) and we would have the following formulation:

$$rat(u,i) = \lambda * \ rat_{R1}(u,i) + (1-\lambda) * rat_{R2}(u,i) \tag{7.2}$$

In this case, since the λ weight is the same for every user u and item i we refer to such a recommender as a *static hybrid*. However, a single value of the combination parameter λ is not generally the optimal for each (user, item) pair. Therefore, instead of Equation (7.2), we may want to consider:

$$rat(u,i) = \gamma_{R1}(u,i) * \ rat_{R1}(u,i) + \gamma_{R2}(u,i) * rat_{R2}(u,i) \tag{7.3}$$

where γ_R is the combination parameter which may depend on the current user, item, or both, and probably also depending on the recommender R. In this case we refer to such method as a *dynamic hybrid*.

A suitable assignment of the $\gamma(u,i)$ parameters is a difficult task. In our approach, however, we propose to use the performance prediction methodology developed in the previous chapter, whenever the predictors show some correlation with the performance of a recommender. In this way, since we have some evidence that the performance predictors are able to estimate in advance the performance of a user in a user or item basis, we can use such estimations to weight accordingly the ratings predicted for a given user and item pair by each recommender.

In this context, it is not granted in general to obtain improvements whenever a performance predictor is used in a dynamic ensemble. We have to devise a set of conditions in which such predictors may be used; moreover, the ensemble problem has to be well defined, which is not always true as we shall show. Hence, we define a framework for dynamic hybrid recommendation based on recommendation performance predictors, characterised by some prerequisites, a specific normalisation strategy, and a weighting distribution among recommenders. In this framework, the weights γ_R are obtained by transformations of the values obtained by a performance predictor, in a similar way as the work presented in (Yom-Tov et al., 2005b) on rank aggregation in Information Retrieval, but in the context of Recommender Systems.

7.2.1 Requirements

A first requirement to use a performance predictor for weighting the recommenders of an ensemble, is that it should correlate positively with the performance of not all but some of such recommenders, or with the performance of all the recommenders but to different degrees. If a performance predictor correlates positively with all the recommenders in an ensemble to a similar extent, it does not provide a discriminative criteria to weight the recommenders any differently.

A predictor should be used to assign weights to those recommenders of the ensemble with which it correlates for performance. These assignments also alter the weights of the uncorrelated recommenders, since the weights of all the recommenders in the ensemble need to sum to 1. However, this should not affect the overall performance contribution of these recommenders, as the resulting weight should correspond randomly with their performance (hence the unpredicted recommenders' weight can be expected to change for good as much as for bad, whereas the weight of predicted recommenders should change more often for good).

Figure 7.1 shows which correlations can be considered valid according to the statements presented above, for an ensemble with two recommenders R1 and R2. The horizontal axis depicts the correlation with respect R1 and the vertical axis with R2. Hence, the dotted area represents those situations where a predictor's correlation for R1 is higher than for R2, and thus, the predictor should weight R1. Analogously, the striped area represents the candidate situations where the predictor should weight R2. Furthermore, when correlations with R1 and R2 are too similar (diagonal) no weighting assignment is preferred, and thus, if a predictor lies in the white area it should be used for weighting neither R1 nor R2 for the reasons described above.

Another requirement is that a recommender should not have an always superior or always inferior performance to those of the rest of the ensemble's recommenders. Otherwise the problem is distorted by the fact that the best weight is the one that gets closest to 0 for the recommenders that systematically perform worse (or 1 for the best), regardless of how excellent or terribly bad is the applied strategy, or the predictive power of the approach, since a biased predictor (either towards 0 or 1, depending on which recommender (the worst or the best) such predictor is weighting) would obtain very good results. This issue is recognised in (van Setten, 2005) where the author presents the situation where all recommenders produce item suggestions that are all too low or all too high with respect to the true user's preferences, and then the recommender ensemble is less accurate than the best individual recommender. In summary, underperforming recommenders are useless in an ensemble to begin with, or equivalently, the over performing one(s) should be used alone, and thus, there is no true weighting problem to solve.

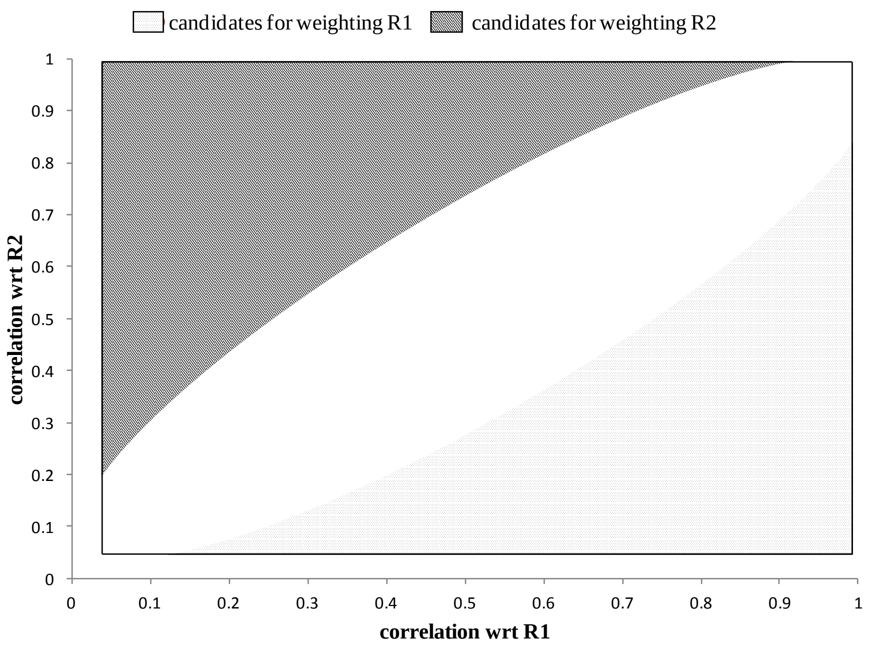

Figure 7.1. Valid predictor correlation regions for a recommender ensemble of size 2.

7.2.2 Predictor normalisation

The output of a predictor is required to correlate with the performance of a recommender, but it is not necessarily by itself a good value for weighting the recommender in an ensemble, as already pointed out in (Hauff et al., 2009). In order to generate appropriate weights, the predictor output should be transformed by a monotonic function into values on a comparable scale, such as simply $[0,1]$. We shall call this transformation "normalisation."

In this context, different transformations can be applied. Mapping the minimum value to 0 and the maximum to 1 is the simplest transformation, also known as *min-max* score normalisation (Renda and Straccia, 2003). Another common approach is to map (named *rank-sim* by Renda and Straccia, 2003) the predictor scores onto evenly distributed points in the $[0,1]$, preserving their order. Min-max preserves the original predictor score distribution, while rank-sim maps it onto a uniform distribution. There is no obvious a priori reason to decide which case is preferable, to preserve the original distribution, or to equalise it somehow, and in fact more complex normalisation techniques could be used, like the one proposed in (Fernández et al., 2006b).

7.2.3 Weight distribution among recommenders

Once the predictor output has been normalised, it still needs a final adjustment to ensure, among other things, that the sum of the weights assigned to the ensemble's recommenders is 1. How this step is done depends, mainly, on how many recommenders are weighted by predictors, more specifically on whether all or only some of the combined recommenders are treated by performance predictors. Hence, we consider two options for the distribution of the weights among the recommenders:

a) Only some of the recommenders in the ensemble are given dynamic weights. The rest of the recommenders receive the same weight, ensuring the weights of the ensemble's recommenders sum up to 1. This can be done in different ways:
   - Assigning a weight of 0.5 to the unpredicted recommenders, and dividing all weights by the total sum. This strategy is named as *fixed weight* or **FW**.
   - Assigning the dynamic weights to the corresponding recommenders, if we assume that their sum is ≤ 1, then we divide 1 minus the sum of dynamic coefficients equally among the unpredicted recommenders. We denote this strategy as *one minus* or **OM**. If the sum is greater than 1, we have to divide by the total sum and normalise it by the total number of predictors.

b) All recommenders are weighted using a specific predictor per recommender. This is not easy to grant in general, as there may not be predictors for all the recommenders combined. In case this option is taken, the weights can be simply normalised by the sum of weights.

Furthermore, if the output of each recommender has a different range, it would be necessary to apply an additional normalisation step to the recommender scores. The most usual strategies are the ones described in the previous section: score or rank normalisation (Renda and Straccia, 2003).

7.3 Experimental results

We next report experiments assessing the usefulness of the proposed predictors for adjusting the weights of a recommender ensemble, once their predictive power has been confirmed against the recommenders' actual performance, as reported in the previous chapter. We identify the combinations of recommenders that meet the conditions stated in the previous section for the dynamic combination problem to make sense and select the performance predictors to be applied based on their observed correlation with the performance of the recommenders (as reported in Section 6.5), and the requirements proposed in this chapter, i.e., that one recommender in the ensemble should have a positive correlation with the predictor, and the other should have an opposite or near neutral correlation. Then, we compare dynamic against static ensembles.

Among the different ways to set up static ensembles of two recommenders we take as baselines a) the best performing one in test, and b) the best theoretical static one without prior information, i.e., one with $\lambda = 0.5$. Intuitively, an even weighting is the optimum over the – theoretical – set of all recommender ensembles: if say $\lambda_B = 0.3$ was the best weight for the combination of two recommenders R1+R2, then $\lambda = 0.3$ should be fairly bad for the permutation R2+R1 ($\lambda = 1 - 0.3 = 0.7$ being best). If we assume that performance loss is convex with respect to $|\lambda - \lambda_B|$ – it can be seen that otherwise the hybrid may underperform its constituents –, then $\lambda = 0.5$ is the best compromise for R1+R2 and R2+R1. Since the set of all possible ensembles includes all the permutations of the combined recommenders, $\lambda = 0.5$ is the best (theoretical) overall weight.

We also take as "skylines" (upper bound baselines) an oracle performance predictor consisting of the performance of the recommender itself. We shall refer to this method as 'perfect correlation', where the true performance of both recommenders is used as a weight for hybridisation (hence, such predictor would have a correlation of 1.0 with the recommender's performance), whereas we shall refer to it as 'PC-OM' and 'PC-FW' when the performance of only one recommender is used (the same recommender being weighted by the predictors) along with the one minus or the fixed weight strategy for weight distribution (see Section 7.2.3). In all cases we apply a rank normalisation technique on the recommenders' scores.

In the subsequent sections we present three experiments conducted to evaluate the proposed performance predictors. In the first experiment we use the rating-based predictors and test both user- and item-based performance predictors presented in Section 6.2.1. We use the MovieLens dataset, and compare the results with four of the evaluation methodologies presented in Chapter 4, i.e., AR, 1R, P1R, and U1R. In the second experiment we use predictors based on log data. We evaluate the predictors presented in Section 6.2.2 on the two versions of the Last.fm dataset using the 1R methodology. Finally, in the third experiment we test the social-based predictors presented in Section 6.3 on the CAMRa dataset and the AR methodology.

7.3.1 Dynamic recommender ensembles on rating data

As a first instantiation of our framework for building dynamic recommender ensembles described in Section 7.2, we first have to identify the recommenders to combine, that is: one of the recommenders should have a positive correlation with the predictor, while the other should have an opposite or near neutral correlation; besides, they should not perform very differently.

According to the correlation results presented in Section 6.5.1, we identify the pairs of recommenders presented in Table 7.1 as combinations meeting the conditions stated above. The first three ensembles are combinations of a collaborative filtering with a content-based recommendation method. The last ensemble combines a user-based collaborative filtering method with a non-personalised method, and the rest of the ensembles are combinations of two collaborative filtering methods. Al-

	R1	R2
HRU1	TFL1	CB
HRU2	TFL2	CB
HRU3	kNN	CB
HRU4	kNN	IB
HRU5	kNN	pLSA
HRU6	kNN	ItemPop

Table 7.1. Selected recommenders for building dynamic ensemble using user performance predictors that exploit rating-based information (MovieLens dataset).

though some of these combinations have not been typical in the recommender systems literature, in our study they serve as a proof of concept to check whether the proposed dynamic recommender ensemble framework is useful in general or not. We refer the reader to Appendix A.2 for more details about the implementation of the recommenders.

The first two rows of Table 7.2, Table 7.3, Table 7.4, and Table 7.5 show the P@10 values for each of the combined recommenders obtained using the AR, 1R, U1R, and P1R methodologies, respectively. In Appendix A.5.1 we report results with other evaluation metrics. Note that, as mentioned in Chapter 4, in the AR methodology the absolute values are not meaningful since they depend on the amount of relevant information in test; on the other hand, for the 1R related methodologies (i.e., 1R, U1R, and P1R) the precision at 10 metric has an upper bound on 0.1, since there is only one relevant item in each ranking.

In these tables we may observe that among the six considered ensembles, there are cases where the first recommender (with respect to which the performance is predicted) performs better, worse, or similarly to the second recommender. This situation changes accross methodologies and provides for a comparison of the resulting effects when the stated requirements are not met. Analogously, the predictors' correlations may change depending on the evaluation methodology followed, as observed in Section 6.5.1. Specifically, the recommenders presented in Table 7.1 where chosen according to the correlation results obtained for the AR methodology, and we may observe that some of the conditions stated above do not hold for some of the selected cases, for instance, correlation between most of the predictors and kNN recommender is negligible in the 1R, U1R, and P1R methodologies, in contrast with the results found for the AR methodology.

In the tables we may also observe that the best static ensemble is different depending on the evaluation methodology and the combined recommenders. The performance values of the best static ensembles, on the other hand, show an interesting situation that does depend on the specific considered ensemble, namely, whether the (best) static ensembles outperform or not both recommenders. For the AR methodology (Table 7.2), in the case of HRU1, HRU3, HRU5, and HRU6, the best static

	HRU1	HRU2	HRU3	HRU4	HRU5	HRU6
R1 (λ=1.0)	0.0024	0.0696	0.0307	0.0307	0.0307	0.0307
R2 (λ=0.0)	0.0163	0.0163	0.0163	0.0001	0.1454	0.0897
Baseline (λ=0.5)	0.0106	0.0473	0.0363	0.0008	0.1142	0.0808
Best static	0.0180	0.0668	0.0392	0.0078	0.1475	0.0937
(best λ)	(0.1)	(0.9)	(0.9)	(0.9)	(0.1)	(0.1)
Perfect correlation	<u>0.0189</u>	<u>0.0732</u>	<u>0.0401</u>	<u>0.0311</u>	0.1469	<u>0.0980</u>
PC-OM	0.0176	0.0721	0.0434	0.0091	0.1489	0.0958
PC-FW	0.0177	0.0541	0.0379	0.0025	0.1478	0.0958
Entropy-OM	**0.0110**$^{\blacktriangledown}_{\vartriangle}$	**0.0685**$^{\vartriangle}_{\blacktriangle}$	**0.0388**$^{\triangledown}_{\blacktriangle}$	**0.0069**$^{\blacktriangledown}_{\blacktriangle}$	0.1126$^{\blacktriangledown}_{\triangledown}$	0.0791$^{\blacktriangledown}_{\blacktriangledown}$
ItemSimple-OM	**0.0170**$^{\blacktriangledown}_{\blacktriangle}$	**0.0685**$^{\vartriangle}_{\blacktriangle}$	**0.0390**$^{\triangledown}_{\blacktriangle}$	**0.0072**$^{\triangledown}_{\blacktriangle}$	**0.1496**$^{\blacktriangle}_{\blacktriangle}$	**0.0919**$^{\blacktriangledown}_{\blacktriangle}$
ItemUser-OM	**0.0172**$^{\blacktriangledown}_{\blacktriangle}$	**0.0680**$^{\vartriangle}_{\blacktriangle}$	**0.0386**$^{\triangledown}_{\blacktriangle}$	**0.0068**$^{\blacktriangledown}_{\blacktriangle}$	**0.1513**$^{\blacktriangle}_{\blacktriangle}$	**0.0924**$^{\blacktriangledown}_{\blacktriangle}$
RatUser-OM	**0.0177**$^{\triangledown}_{\blacktriangle}$	**0.0687**$^{\vartriangle}_{\blacktriangle}$	**0.0393**$^{\vartriangle}_{\blacktriangle}$	**0.0072**$^{\triangledown}_{\blacktriangle}$	**0.1535**$^{\blacktriangle}_{\blacktriangle}$	**0.0931**$^{\triangledown}_{\blacktriangle}$
RatItem-OM	**0.0178**$^{\triangledown}_{\blacktriangle}$	**0.0674**$^{\vartriangle}_{\blacktriangle}$	**0.0389**$^{\triangledown}_{\blacktriangle}$	**0.0066**$^{\blacktriangledown}_{\blacktriangle}$	**<u>0.1542</u>**$^{\blacktriangle}_{\blacktriangle}$	**0.0928**$^{\triangledown}_{\blacktriangle}$
IRUser-OM	**0.0169**$^{\blacktriangledown}_{\blacktriangle}$	**0.0668**$_{\blacktriangle}$	**0.0387**$^{\triangledown}_{\blacktriangle}$	**0.0066**$^{\blacktriangledown}_{\blacktriangle}$	**0.1487**$^{\blacktriangle}_{\blacktriangle}$	**0.0922**$^{\blacktriangledown}_{\blacktriangle}$
IRItem-OM	**0.0172**$^{\blacktriangledown}_{\blacktriangle}$	**0.0655**$^{\triangledown}_{\blacktriangle}$	**0.0378**$^{\blacktriangledown}_{\blacktriangle}$	**0.0061**$^{\blacktriangledown}_{\blacktriangle}$	**0.1500**$^{\blacktriangle}_{\blacktriangle}$	**0.0918**$^{\blacktriangledown}_{\blacktriangle}$
IRUserItem-OM	**0.0170**$^{\blacktriangledown}_{\blacktriangle}$	**0.0665**$^{\triangledown}_{\blacktriangle}$	**0.0388**$^{\triangledown}_{\blacktriangle}$	**0.0066**$^{\blacktriangledown}_{\blacktriangle}$	**0.1498**$^{\blacktriangle}_{\blacktriangle}$	**0.0916**$^{\blacktriangledown}_{\blacktriangle}$
Entropy-FW	**0.0111**$^{\blacktriangledown}_{\blacktriangle}$	**0.0528**$^{\blacktriangledown}_{\blacktriangle}$	**0.0369**$^{\blacktriangledown}_{\vartriangle}$	**0.0027**$^{\blacktriangledown}_{\blacktriangle}$	**0.1156**$^{\blacktriangledown}_{\blacktriangle}$	0.0807$^{\blacktriangledown}_{\triangledown}$
ItemSimple-FW	**0.0156**$^{\blacktriangledown}_{\blacktriangle}$	**0.0529**$^{\blacktriangledown}_{\blacktriangle}$	**0.0369**$^{\blacktriangledown}_{\vartriangle}$	**0.0027**$^{\blacktriangledown}_{\blacktriangle}$	**0.1433**$^{\blacktriangledown}_{\blacktriangle}$	**0.0908**$^{\blacktriangledown}_{\blacktriangle}$
ItemUser-FW	**0.0166**$^{\blacktriangledown}_{\blacktriangle}$	**0.0529**$^{\blacktriangledown}_{\blacktriangle}$	**0.0368**$^{\blacktriangledown}_{\vartriangle}$	**0.0028**$^{\blacktriangledown}_{\blacktriangle}$	**0.1468**$^{\triangledown}_{\blacktriangle}$	**0.0915**$^{\blacktriangledown}_{\blacktriangle}$
RatUser-FW	**0.0170**$^{\blacktriangledown}_{\blacktriangle}$	**0.0528**$^{\blacktriangledown}_{\blacktriangle}$	**0.0370**$^{\blacktriangledown}_{\vartriangle}$	**0.0028**$^{\blacktriangledown}_{\blacktriangle}$	**0.1498**$^{\blacktriangle}_{\blacktriangle}$	**0.0919**$^{\blacktriangledown}_{\blacktriangle}$
RatItem-FW	**0.0170**$^{\blacktriangledown}_{\blacktriangle}$	**0.0529**$^{\blacktriangledown}_{\blacktriangle}$	**0.0369**$^{\blacktriangledown}_{\vartriangle}$	**0.0027**$^{\blacktriangledown}_{\blacktriangle}$	**0.1499**$^{\blacktriangle}_{\blacktriangle}$	**0.0918**$^{\blacktriangledown}_{\blacktriangle}$
IRUser-FW	**0.0161**$^{\blacktriangledown}_{\blacktriangle}$	**0.0526**$^{\blacktriangledown}_{\blacktriangle}$	**0.0371**$^{\blacktriangledown}_{\vartriangle}$	**0.0029**$^{\blacktriangledown}_{\blacktriangle}$	**0.1420**$^{\blacktriangledown}_{\blacktriangle}$	**0.0912**$^{\blacktriangledown}_{\blacktriangle}$
IRItem-FW	**0.0163**$^{\blacktriangledown}_{\blacktriangle}$	**0.0525**$^{\blacktriangledown}_{\blacktriangle}$	**0.0367**$^{\blacktriangledown}_{\vartriangle}$	**0.0027**$^{\blacktriangledown}_{\blacktriangle}$	**0.1459**$^{\triangledown}_{\blacktriangle}$	**0.0909**$^{\blacktriangledown}_{\blacktriangle}$
IRUserItem-FW	**0.0164**$^{\blacktriangledown}_{\blacktriangle}$	**0.0527**$^{\blacktriangledown}_{\blacktriangle}$	**0.0372**$^{\blacktriangledown}_{\vartriangle}$	**0.0028**$^{\blacktriangledown}_{\blacktriangle}$	**0.1452**$^{\triangledown}_{\blacktriangle}$	**0.0908**$^{\blacktriangledown}_{\blacktriangle}$

Table 7.2. Dynamic ensemble performance values (P@10) using <u>AR methodology</u> and user predictors (MovieLens dataset). Improvements over the baseline are in bold, the best result for each column is underlined. The value a of each dynamic hybrid is marked with a_y^x, where x and y indicate, respectively, statistical difference with respect to the best static (upper, x) and with respect to the baseline (lower, y). Moreover, $\blacktriangle$ and $\vartriangle$ indicate, respectively, significant and non-significant improvements over the corresponding recommender. A similar convention with $\blacktriangledown$ and $\triangledown$ indicates values below the recommender performance. Statistical significance is established by paired Wilcoxon $p < 0.05$ in all cases.

outperforms both recommenders, but this is not observed for HRU2 nor for HRU4. In the latter scenarios, thus, it seems hybridisation would not be so useful for combination.

Additionally, regarding the normalisation of the predictor's output we evaluate two normalisation techniques: rank and score normalisation. Since there is no prior information about which normalisation technique would provide better results, we test both, and report the best results in each situation, which are usually achieved by the rank-sim normalisation technique. Finally, the weigh strategy is also included as a parameter of the experiments. Since we only have a predictor for one of the recommenders in the ensemble (denoted as R1), as we explained in Section 7.2.3, we may weight the unpredicted recommender as one minus the predictor value (OM), or as 0.5 and then divide the weights of the two recommenders by the sum of weights (FW).

	HRU1	HRU2	HRU3	HRU4	HRU5	HRU6
R1 (λ=1.0)	0.0221	0.0690	0.0437	0.0437	0.0437	0.0437
R2 (λ=0.0)	0.0221	0.0221	0.0221	0.0074	0.0836	0.0649
Baseline (λ=0.5)	0.0338	0.0536	0.0469	0.0327	0.0749	0.0658
Best static	0.0338	0.0720	0.0514	0.0455	0.0856	0.0696
(best λ)	(0.4)	(0.9)	(0.8)	(0.9)	(0.1)	(0.2)
Perfect correlation	0.0370	0.0715	0.0553	0.0458	0.0840	0.0723
PC-OM	0.0358	0.0683	0.0507	0.0353	0.0811	0.0709
PC-FW	0.0343	0.0592	0.0482	0.0344	0.0803	0.0699
Entropy-OM	0.0332$^{\blacktriangledown}_{\blacktriangledown}$	**0.0662**$^{\blacktriangledown}_{\blacktriangle}$	**0.0472**$^{\blacktriangledown}_{\vartriangle}$	**0.0382**$^{\blacktriangledown}_{\blacktriangle}$	0.0709$^{\blacktriangledown}_{\blacktriangledown}$	0.0626$^{\blacktriangledown}_{\blacktriangledown}$
ItemSimple-OM	0.0304$^{\blacktriangledown}_{\blacktriangledown}$	**0.0666**$^{\blacktriangledown}_{\blacktriangle}$	**0.0473**$^{\blacktriangledown}_{\blacktriangle}$	**0.0384**$^{\blacktriangledown}_{\blacktriangle}$	**0.0844**$^{\blacktriangledown}_{\blacktriangle}$	**0.0681**$^{\blacktriangledown}_{\blacktriangle}$
ItemUser-OM	0.0305$^{\blacktriangledown}_{\blacktriangledown}$	**0.0660**$^{\blacktriangledown}_{\blacktriangle}$	**0.0471**$^{\blacktriangledown}_{\vartriangle}$	**0.0381**$^{\blacktriangledown}_{\blacktriangle}$	**0.0847**$^{\blacktriangledown}_{\blacktriangle}$	**0.0680**$^{\blacktriangledown}_{\blacktriangle}$
RatUser-OM	0.0307$^{\blacktriangledown}_{\blacktriangledown}$	**0.0666**$^{\blacktriangledown}_{\blacktriangle}$	**0.0478**$^{\blacktriangledown}_{\blacktriangle}$	**0.0386**$^{\blacktriangledown}_{\blacktriangle}$	**0.0850**$^{\blacktriangledown}_{\blacktriangle}$	**0.0680**$^{\blacktriangledown}_{\blacktriangle}$
RatItem-OM	0.0305$^{\blacktriangledown}_{\blacktriangledown}$	**0.0663**$^{\blacktriangledown}_{\blacktriangle}$	**0.0475**$^{\blacktriangledown}_{\blacktriangle}$	**0.0385**$^{\blacktriangledown}_{\blacktriangle}$	**0.0849**$^{\blacktriangledown}_{\blacktriangle}$	**0.0678**$^{\blacktriangledown}_{\blacktriangle}$
IRUser-OM	0.0304$^{\blacktriangledown}_{\blacktriangledown}$	**0.0655**$^{\blacktriangledown}_{\blacktriangle}$	**0.0470**$^{\blacktriangledown}_{\vartriangle}$	**0.0381**$^{\blacktriangledown}_{\blacktriangle}$	**0.0839**$^{\blacktriangledown}_{\blacktriangle}$	**0.0675**$^{\blacktriangledown}_{\blacktriangle}$
IRItem-OM	0.0298$^{\blacktriangledown}_{\blacktriangledown}$	**0.0644**$^{\blacktriangledown}_{\blacktriangle}$	0.0457$^{\blacktriangledown}_{\blacktriangledown}$	**0.0370**$^{\blacktriangledown}_{\blacktriangle}$	**0.0839**$^{\blacktriangledown}_{\blacktriangle}$	**0.0671**$^{\blacktriangledown}_{\blacktriangle}$
IRUserItem-OM	0.0305$^{\blacktriangledown}_{\blacktriangledown}$	**0.0655**$^{\blacktriangledown}_{\blacktriangle}$	**0.0471**$^{\blacktriangledown}_{\vartriangle}$	**0.0381**$^{\blacktriangledown}_{\blacktriangle}$	**0.0841**$^{\blacktriangledown}_{\blacktriangle}$	**0.0674**$^{\blacktriangledown}_{\blacktriangle}$
Entropy-FW	**0.0339**$^{\vartriangle}_{\vartriangle}$	**0.0594**$^{\blacktriangledown}_{\blacktriangle}$	**0.0472**$^{\blacktriangledown}_{\vartriangle}$	**0.0356**$^{\blacktriangledown}_{\blacktriangle}$	0.0686$^{\blacktriangledown}_{\blacktriangledown}$	0.0650$^{\blacktriangledown}_{\blacktriangledown}$
ItemSimple-FW	0.0321$^{\blacktriangledown}_{\blacktriangledown}$	**0.0596**$^{\blacktriangledown}_{\blacktriangle}$	**0.0473**$^{\blacktriangledown}_{\blacktriangle}$	**0.0358**$^{\blacktriangledown}_{\blacktriangle}$	**0.0837**$^{\blacktriangledown}_{\blacktriangle}$	**0.0684**$^{\blacktriangledown}_{\blacktriangle}$
ItemUser-FW	0.0320$^{\blacktriangledown}_{\blacktriangledown}$	**0.0594**$^{\blacktriangledown}_{\blacktriangle}$	**0.0471**$^{\blacktriangledown}_{\vartriangle}$	**0.0356**$^{\blacktriangledown}_{\blacktriangle}$	**0.0843**$^{\blacktriangledown}_{\blacktriangle}$	**0.0683**$^{\blacktriangledown}_{\blacktriangle}$
RatUser-FW	0.0321$^{\blacktriangledown}_{\blacktriangledown}$	**0.0596**$^{\blacktriangledown}_{\blacktriangle}$	**0.0475**$^{\blacktriangledown}_{\blacktriangle}$	**0.0359**$^{\blacktriangledown}_{\blacktriangle}$	**0.0848**$^{\blacktriangledown}_{\blacktriangle}$	**0.0684**$^{\blacktriangledown}_{\blacktriangle}$
RatItem-FW	0.0321$^{\blacktriangledown}_{\blacktriangledown}$	**0.0595**$^{\blacktriangledown}_{\blacktriangle}$	**0.0473**$^{\blacktriangledown}_{\blacktriangle}$	**0.0358**$^{\blacktriangledown}_{\blacktriangle}$	**0.0847**$^{\blacktriangledown}_{\blacktriangle}$	**0.0684**$^{\blacktriangledown}_{\blacktriangle}$
IRUser-FW	0.0320$^{\blacktriangledown}_{\blacktriangledown}$	**0.0592**$^{\blacktriangledown}_{\blacktriangle}$	**0.0471**$^{\blacktriangledown}_{\vartriangle}$	**0.0356**$^{\blacktriangledown}_{\blacktriangle}$	**0.0834**$^{\blacktriangledown}_{\blacktriangle}$	**0.0680**$^{\blacktriangledown}_{\blacktriangle}$
IRItem-FW	0.0318$^{\blacktriangledown}_{\blacktriangledown}$	**0.0588**$^{\blacktriangledown}_{\blacktriangle}$	0.0465$^{\blacktriangledown}_{\blacktriangledown}$	**0.0349**$^{\blacktriangledown}_{\blacktriangle}$	**0.0835**$^{\blacktriangledown}_{\blacktriangle}$	**0.0674**$^{\blacktriangledown}_{\blacktriangle}$
IRUserItem-FW	0.0320$^{\blacktriangledown}_{\blacktriangledown}$	**0.0592**$^{\blacktriangledown}_{\blacktriangle}$	**0.0471**$^{\blacktriangledown}_{\vartriangle}$	**0.0356**$^{\blacktriangledown}_{\blacktriangle}$	**0.0837**$^{\blacktriangledown}_{\blacktriangle}$	**0.0678**$^{\blacktriangledown}_{\blacktriangle}$

Table 7.3. Dynamic ensemble performance values (P@10) using 1R methodology and user predictors (MovieLens dataset).

Table 7.2 shows the results obtained following the AR methodology. We may observe how, except in three cases, dynamic ensembles outperform the baseline. Interestingly, for HRU5, the best performing method is not the one obtained with the 'perfect correlation' approach, as we may expect, but with our dynamic ensembles based on the user clarity performance predictors. This is due to the fact that the corresponding predictor for the first recommender (P@10 values for kNN) also has a strong correlation with the performance of the second recommender (pLSA), and thus, it does not satisfy the requirement that the correlation values should not be too similar for both recommenders.

Table 7.3 shows the results obtained with the 1R methodology. Note that in this case the correlations were consistently lower than those obtained with the AR methodology. In particular, this is emphasised in the results of the dynamic ensemble HRU1, which do not outperform the baseline for almost any predictor. This can be explained with the results reported in Table 6.9, where the TFL1 recommender obtains a near-zero correlation, and thus, the correlation requirement of our framework is not satisfied. Specifically, this fact highlights the importance of the strength in the correlation between the predictor and the recommender performance, as stated in Section 7.2.1. Furthermore, we may observe in the table that for two combinations

	HRU1	HRU2	HRU3	HRU4	HRU5	HRU6
R1 (λ=1.0)	0.0294	0.0524	0.0381	0.0381	0.0381	0.0381
R2 (λ=0.0)	0.0223	0.0223	0.0223	0.0068	0.0718	0.0406
Baseline (λ=0.5)	0.0345	0.0440	0.0396	0.0283	0.0639	0.0493
Best static	0.0351	0.0536	0.0424	0.0384	0.0732	0.0493
(best λ)	(0.6)	(0.9)	(0.7)	(0.9)	(0.1)	(0.5)
Perfect correlation	<u>0.0389</u>	<u>0.0552</u>	<u>0.0493</u>	<u>0.0396</u>	<u>0.0742</u>	<u>0.0559</u>
PC-OM	0.0373	0.0485	0.0471	0.0332	0.0732	0.0548
PC-FW	0.0355	0.0459	0.0429	0.0307	0.0722	0.0535
Entropy-OM	0.0345$^{\blacktriangledown}$	**0.0518**$^{\blacktriangledown}_{\blacktriangle}$	**0.0404**$^{\blacktriangledown}_{\blacktriangle}$	**0.0337**$^{\blacktriangledown}_{\blacktriangle}$	0.0615$^{\blacktriangledown}_{\blacktriangledown}$	0.0471$^{\blacktriangledown}_{\blacktriangledown}$
ItemSimple-OM	0.0333$^{\blacktriangledown}_{\blacktriangledown}$	**0.0519**$^{\blacktriangledown}_{\blacktriangle}$	**0.0403**$^{\blacktriangledown}_{\blacktriangle}$	**0.0339**$^{\blacktriangledown}_{\blacktriangle}$	**0.0723**$^{\blacktriangledown}_{\blacktriangle}$	0.0444$^{\blacktriangledown}_{\blacktriangledown}$
ItemUser-OM	0.0334$^{\blacktriangledown}_{\blacktriangledown}$	**0.0517**$^{\blacktriangledown}_{\blacktriangle}$	**0.0403**$^{\blacktriangledown}_{\blacktriangle}$	**0.0336**$^{\blacktriangledown}_{\blacktriangle}$	**0.0726**$^{\blacktriangledown}_{\blacktriangle}$	0.0438$^{\blacktriangledown}_{\blacktriangledown}$
RatUser-OM	0.0335$^{\blacktriangledown}_{\blacktriangledown}$	**0.0521**$^{\blacktriangledown}_{\blacktriangle}$	**0.0410**$^{\blacktriangledown}_{\blacktriangle}$	**0.0341**$^{\blacktriangledown}_{\blacktriangle}$	**0.0728**$^{\blacktriangledown}_{\blacktriangle}$	0.0435$^{\blacktriangledown}_{\blacktriangledown}$
RatItem-OM	0.0334$^{\blacktriangledown}_{\blacktriangledown}$	**0.0516**$^{\blacktriangledown}_{\blacktriangle}$	**0.0406**$^{\blacktriangledown}_{\blacktriangle}$	**0.0341**$^{\blacktriangledown}_{\blacktriangle}$	**0.0726**$^{\blacktriangledown}_{\blacktriangle}$	0.0434$^{\blacktriangledown}_{\blacktriangledown}$
IRUser-OM	0.0333$^{\blacktriangledown}_{\blacktriangledown}$	**0.0511**$^{\blacktriangledown}_{\blacktriangle}$	**0.0401**$^{\blacktriangledown}_{\blacktriangle}$	**0.0336**$^{\blacktriangledown}_{\blacktriangle}$	**0.0718**$^{\blacktriangledown}_{\blacktriangle}$	0.0440$^{\blacktriangledown}_{\blacktriangledown}$
IRItem-OM	0.0326$^{\blacktriangledown}_{\blacktriangledown}$	**0.0504**$^{\blacktriangledown}_{\blacktriangle}$	0.0388$^{\blacktriangledown}_{\blacktriangledown}$	**0.0325**$^{\blacktriangledown}_{\blacktriangle}$	**0.0714**$^{\blacktriangledown}_{\blacktriangle}$	0.0430$^{\blacktriangledown}_{\blacktriangledown}$
IRUserItem-OM	0.0334$^{\blacktriangledown}_{\blacktriangledown}$	**0.0511**$^{\blacktriangledown}_{\blacktriangle}$	**0.0401**$^{\blacktriangledown}_{\blacktriangle}$	**0.0336**$^{\blacktriangledown}_{\blacktriangle}$	**0.0719**$^{\blacktriangledown}_{\blacktriangle}$	0.0437$^{\blacktriangledown}_{\blacktriangledown}$
Entropy-FW	**0.0347**$^{\blacktriangledown}_{\blacktriangle}$	**0.0472**$^{\blacktriangledown}_{\blacktriangle}$	**0.0402**$^{\blacktriangledown}_{\blacktriangle}$	**0.0308**$^{\blacktriangledown}_{\blacktriangle}$	0.0636$^{\blacktriangledown}_{\blacktriangledown}$	0.0486$^{\blacktriangledown}_{\blacktriangledown}$
ItemSimple-FW	0.0342$^{\blacktriangledown}_{\blacktriangledown}$	**0.0473**$^{\blacktriangledown}_{\blacktriangle}$	**0.0402**$^{\blacktriangledown}_{\blacktriangle}$	**0.0309**$^{\blacktriangledown}_{\blacktriangle}$	**0.0720**$^{\blacktriangledown}_{\blacktriangle}$	0.0467$^{\blacktriangledown}_{\blacktriangledown}$
ItemUser-FW	0.0342$^{\blacktriangledown}_{\blacktriangledown}$	**0.0471**$^{\blacktriangledown}_{\blacktriangle}$	**0.0401**$^{\blacktriangledown}_{\blacktriangle}$	**0.0308**$^{\blacktriangledown}_{\blacktriangle}$	**0.0724**$^{\blacktriangledown}_{\blacktriangle}$	0.0467$^{\blacktriangledown}_{\blacktriangledown}$
RatUser-FW	0.0343$^{\blacktriangledown}_{\triangledown}$	**0.0474**$^{\blacktriangledown}_{\blacktriangle}$	**0.0405**$^{\blacktriangledown}_{\blacktriangle}$	**0.0310**$^{\blacktriangledown}_{\blacktriangle}$	**0.0727**$^{\blacktriangledown}_{\blacktriangle}$	0.0469$^{\blacktriangledown}_{\blacktriangledown}$
RatItem-FW	0.0342$^{\blacktriangledown}_{\blacktriangledown}$	**0.0472**$^{\blacktriangledown}_{\blacktriangle}$	**0.0403**$^{\blacktriangledown}_{\blacktriangle}$	**0.0309**$^{\blacktriangledown}_{\blacktriangle}$	**0.0725**$^{\blacktriangledown}_{\blacktriangle}$	0.0469$^{\blacktriangledown}_{\blacktriangledown}$
IRUser-FW	0.0341$^{\blacktriangledown}_{\blacktriangledown}$	**0.0470**$^{\blacktriangledown}_{\blacktriangle}$	**0.0401**$^{\blacktriangledown}_{\blacktriangle}$	**0.0308**$^{\blacktriangledown}_{\blacktriangle}$	**0.0714**$^{\blacktriangledown}_{\blacktriangle}$	0.0469$^{\blacktriangledown}_{\blacktriangledown}$
IRItem-FW	0.0338$^{\blacktriangledown}_{\blacktriangledown}$	**0.0467**$^{\blacktriangledown}_{\blacktriangle}$	0.0393$^{\blacktriangledown}_{\triangledown}$	**0.0302**$^{\blacktriangledown}_{\blacktriangle}$	**0.0712**$^{\blacktriangledown}_{\blacktriangle}$	0.0464$^{\blacktriangledown}_{\blacktriangledown}$
IRUserItem-FW	0.0341$^{\blacktriangledown}_{\blacktriangledown}$	**0.0471**$^{\blacktriangledown}_{\blacktriangle}$	**0.0401**$^{\blacktriangledown}_{\blacktriangle}$	**0.0308**$^{\blacktriangledown}_{\blacktriangle}$	**0.0716**$^{\blacktriangledown}_{\blacktriangle}$	0.0469$^{\blacktriangledown}_{\blacktriangledown}$

Table 7.4. Dynamic ensemble performance values (P@10) using the <u>U1R methodology</u> and user predictors (MovieLens dataset)

(HRU2 and HRU5) the best performance results are not obtained by dynamic approaches, but by the best static approaches in contrast with what we found for the AR methodology. This situation is different to the one obtained when we evaluate using MAP@10 (see Appendix A.4.1), where the best results are always obtained by dynamic ensembles.

Table 7.4 and Table 7.5 show the performance values obtained with the unbiased methodologies proposed in Chapter 4, that is, U1R and P1R. Following the U1R methodology (Table 7.4) we obtain similar results to those obtained in the 1R methodology except for HRU6. In contrast, with the P1R methodology (Table 7.5) our framework does not show improvements over any baseline. We may see that the 'perfect correlation' methods are able to obtain better, although very close, values than those of the best static ensemble. This means that there is room for improvement in this methodology, and that the performance of the dynamic recommender ensembles could be improved if better performance predictors were found.

	HRU1	HRU2	HRU3	HRU4	HRU5	HRU6
R1 (λ=1.0)	0.0203	0.0348	0.0265	0.0265	0.0265	0.0265
R2 (λ=0.0)	0.0197	0.0197	0.0197	0.0208	0.0604	0.0282
Baseline (λ=0.5)	0.0470	0.0579	0.0539	0.0269	0.0763	0.0560
Best static	0.0470	0.0593	0.0541	0.0278	0.0796	0.0560
(best λ)	(0.5)	(0.6)	(0.6)	(0.7)	(0.4)	(0.5)
Perfect correlation	0.0464	0.0579	0.0546	0.0314	0.0767	0.0564
PC-OM	0.0425	0.0554	0.0528	0.0296	0.0746	0.0537
PC-FW	0.0429	0.0542	0.0504	0.0282	0.0764	0.0522
Entropy-OM	0.0431▼	0.0564▼	0.0502▼	0.0261▼	0.0698▼	0.0521▼
ItemSimple-OM	0.0358▼	0.0509▼	0.0429▼	0.0261▼	0.0689▼	0.0441▼
ItemUser-OM	0.0361▼	0.0512▼	0.0431▼	0.0261▼	0.0675▼	0.0444▼
RatUser-OM	0.0362▼	0.0514▼	0.0436▼	0.0263▼	0.0663▼	0.0446▼
RatItem-OM	0.0361▼	0.0511▼	0.0432▼	0.0262▼	0.0661▼	0.0444▼
IRUser-OM	0.0365▼	0.0513▼	0.0435▼	0.0263▼	0.0687▼	0.0447▼
IRItem-OM	0.0357▼	0.0504▼	0.0421▼	0.0257▼	0.0669▼	0.0439▼
IRUserItem-OM	0.0365▼	0.0513▼	0.0434▼	0.0263▼	0.0675▼	0.0447▼
Entropy-FW	0.0457▼	0.0577▼	0.0524▼	0.0265▼	0.0745▼	0.0546▼
ItemSimple-FW	0.0410▼	0.0540▼	0.0475▼	0.0266▼	0.0720▼	0.0498▼
ItemUser-FW	0.0409▼	0.0538▼	0.0473▼	0.0265▼	0.0706▼	0.0497▼
RatUser-FW	0.0410▼	0.0540▼	0.0477▼	0.0267▼	0.0691▼	0.0499▼
RatItem-FW	0.0411▼	0.0541▼	0.0476▼	0.0266▼	0.0688▼	0.0499▼
IRUser-FW	0.0410▼	0.0538▼	0.0474▼	0.0266▼	0.0721▼	0.0496▼
IRItem-FW	0.0406▼	0.0534▼	0.0467▼	0.0263▼	0.0699▼	0.0491▼
IRUserItem-FW	0.0409▼	0.0538▼	0.0474▼	0.0266▼	0.0706▼	0.0496▼

Table 7.5. Dynamic ensemble performance values (P@10) using the P1R methodology and user predictors (MovieLens dataset).

In summary, the **results show that our methods significantly outperform static ensembles for different recommender combinations in most of the evaluation methodologies**. Moreover, in most cases our methods also achieve the best results for each ensemble, let aside the performance of the oracle performance prediction (perfect correlation) and best static approaches, which use groundtruth (test) information, differently to the clarity- and entropy-based performance predictors.

Nevertheless, we observe that in those cases where the dynamic ensembles do not perform better than the static ensembles, the best static approaches use values of λ close to 0.5. We hypothesise that our framework may be biased towards favouring those ensembles whose recommender combination is highly unbalanced. Interestingly, although the predictors only weight one of the recommenders (not always the better performing one) a dynamic ensemble is usually able to find the optimal combination in the unbalanced cases. In particular, this could help to answer why our dynamic ensembles underperform static approaches for the U1R and P1R methodologies, since the best static in these cases seem to be often very close to 0.5.

	R1	R2
HRI1	pLSA	CB
HRI2	pLSA	kNN
HRI3	ItemPop	CB
HRI4	ItemPop	kNN

Table 7.6. Selected recommenders for building dynamic ensembles using item predictors that exploit rating data (MovieLens dataset).

Using item-based predictors

As we noted in Section 6.5.2, item-based predictors could also be valuable since they also obtain high correlations with respect to item perfomance. Table 7.6 shows the selected recommenders that satisfy the correlation requirements with item predictors. Table 7.7, Table 7.8, and Table 7.9 show the results obtained when these recommender combinations are evaluated and compared against dynamic versions (using our proposed item predictors), and using the 1R, U1R, and uuU1R methodologies. In this case, ensemble predictions are computed by means of Equation (7.3) with values $\gamma(u,i)$ only depending on the current item, that is, $\gamma(i)$.

When measuring the performance of dynamic ensembles that use item-based performance predictors, we do not compute the perfect correlation predictors because we do not have a standard metric for item performance. Apart from that, the

	HRI1	HRI2	HRI3	HRI4
R1 (λ=1.0)	0.0836	0.0836	0.0649	0.0649
R2 (λ=0.0)	0.0221	0.0437	0.0221	0.0437
Baseline (λ=0.5)	0.0909	0.0924	0.0886	0.0907
Best static (best λ)	0.0909 (0.5)	0.0924 (0.5)	0.0886 (0.5)	0.0907 (0.5)
Entropy-OM	0.0708▼▼	0.0858▼▼	0.0684▼▼	0.0831▼▼
UserSimple-OM	0.0761▼▼	0.0905▼▼	0.0723▼▼	0.0837▼▼
UserItem-OM	0.0776▼▼	0.0903▼▼	0.0749▼▼	0.0843▼▼
RatItem-OM	0.0751▼▼	0.0893▼▼	0.0712▼▼	0.0824▼▼
RatUser-OM	0.0759▼▼	0.0892▼▼	0.0674▼▼	0.0789▼▼
URItem-OM	0.0776▼▼	0.0911▼▼	0.0797▼▼	0.0885▼▼
URUser-OM	0.0781▼▼	0.0906▼▼	0.0721▼▼	0.0820▼▼
URItemUser-OM	0.0777▼▼	0.0909▼▼	0.0777▼▼	0.0869▼▼
Entropy-FW	0.0798▼▼	0.0923▼▼	0.0771▼▼	0.0895▼▼
UserSimple-FW	**0.0946**▲▲	**0.0979**▲▲	**0.0916**▲▲	**0.0949**▲▲
UserItem-FW	**0.0949**▲▲	**0.0980**▲▲	**0.0920**▲▲	**0.0950**▲▲
RatItem-FW	**0.0944**▲▲	**0.0979**▲▲	**0.0913**▲▲	**0.0948**▲▲
RatUser-FW	**0.0946**▲▲	**0.0978**▲▲	**0.0908**▲▲	**0.0942**▲▲
URItem-FW	**0.0940**▲▲	**0.0981**▲▲	**0.0923**▲▲	**0.0958**▲▲
URUser-FW	**0.0946**▲▲	**0.0978**▲▲	**0.0912**▲▲	**0.0945**▲▲
URItemUser-FW	**0.0944**▲▲	**0.0980**▲▲	**0.0921**▲▲	**0.0954**▲▲

Table 7.7. Dynamic ensemble performance values (P@10) using 1R methodology with item predictors (MovieLens dataset).

rest of the experimental settings is the same as those described above for dynamic hybrids with user-based performance predictors.

Table 7.7 shows the results obtained by using item-based predictors and the 1R methodology. We may observe that if the predictors are weighted using the FW strategy, dynamic ensembles outperform static combinations in every situation, except for the Entropy predictor. It is interesting to note that, differently to user-based predictors, the dynamic ensembles are able to outperform the best static ensemble even when they are close to the baseline with $\lambda = 0.5$. The reader may compare Table 7.4 and Table 7.7 to observe these differences.

In Table 7.8, where the methodology U1R is used, a very similar situation occurs, although not all dynamic ensembles outperform the static approach with the FW strategy. Specifically, the dynamic hybrid weighted by the URItem clarity predictor clearly obtains better performance than the rest of the dynamic and static ensembles, in particular the HRI3 and HRI4 combinations.

Finally, the performance results found for the uuU1R methodology are presented in Table 7.9, in which the test ratings – i.e., the users – are uniformly distributed over the items, items previously uniformly distributed in the test (like in the U1R methodology). In this experiment, the performance of the dynamic ensemble is much better than in the previous experiments, since **all the rating-based item predictors (except for the Entropy predictor) outperform the static baseline no matter the weighting strategy in three out of four recommender combinations**.

	HRI1	HRI2	HRI3	HRI4
R1 (λ=1.0)	0.0718	0.0718	0.0406	0.0406
R2 (λ=0.0)	0.0223	0.0381	0.0223	0.0381
Baseline (λ=0.5)	0.0764	0.0812	0.0630	0.0689
Best static (best λ)	0.0764 (0.5)	0.0812 (0.5)	0.0630 (0.5)	0.0689 (0.5)
Entropy-OM	0.0571▼	0.0652▼	0.0435▼	0.0508▼
UserSimple-OM	0.0657▼	0.0716▼	0.0399▼	0.0450▼
UserItem-OM	0.0671▼	0.0721▼	0.0425▼	0.0462▼
RatItem-OM	0.0645▼	0.0699▼	0.0392▼	0.0435▼
RatUser-OM	0.0620▼	0.0671▼	0.0335▼	0.0382▼
URItem-OM	0.0705▼	0.0757▼	0.0496▼	0.0532▼
URUser-OM	0.0650▼	0.0699▼	0.0372▼	0.0414▼
URItemUser-OM	0.0690▼	0.0741▼	0.0462▼	0.0500▼
Entropy-FW	0.0668▼	0.0757▼	0.0518▼	0.0595▼
UserSimple-FW	**0.0840**▲	**0.0886**▲	0.0601▼	0.0658▼
UserItem-FW	**0.0844**▲	**0.0887**▲	0.0609▼	0.0663▼
RatItem-FW	**0.0839**▲	**0.0883**▲	0.0598▼	0.0653▼
RatUser-FW	**0.0831**▲	**0.0876**▲	0.0573▼	0.0630▼
URItem-FW	**0.0851**▲	**0.0897**▲	**0.0642**▲	**0.0698**▲
URUser-FW	**0.0836**▲	**0.0881**▲	0.0585▼	0.0642▼
URItemUser-FW	**0.0848**▲	**0.0893**▲	0.0625▼	0.0680▼

Table 7.8. Dynamic ensemble performance values (P@10) using U1R methodology with item predictors (MovieLens dataset).

	HRI1	HRI2	HRI3	HRI4
R1 (λ=1.0)	0.0536	0.0536	0.0225	0.0225
R2 (λ=0.0)	0.0198	0.0275	0.0198	0.0275
Baseline (λ=0.5)	0.0374	0.0440	0.0239	0.0256
Best static	0.0491	0.0502	0.0239	0.0271
(best λ)	(0.9)	(0.9)	(0.6)	(0.2)
Entropy-OM	0.0324▼▼	0.0385▼▼	0.0236▽▽	**0.0280**▲▲
UserSimple-OM	**0.0510**△△	**0.0548**▲▲	0.0237▽▽	**0.0282**▲▲
UserItem-OM	**0.0514**▲▲	**0.0547**▲▲	0.0236▽▽	**0.0280**▲▲
RatItem-OM	**0.0516**▲▲	**0.0547**▲▲	0.0237▽▽	**0.0281**▲▲
RatUser-OM	**0.0523**▲▲	**0.0551**▲▲	0.0237▽▽	**0.0282**▲▲
URItem-OM	**0.0498**▲▲	**0.0536**▲▲	0.0234▼▼	**0.0280**▲▲
URUser-OM	**0.0518**▲▲	**0.0551**▲▲	0.0234▼▼	**0.0279**▲▲
URItemUser-OM	**0.0505**▲▲	**0.0542**▲▲	0.0235▽▼	**0.0280**▲▲
Entropy-FW	0.0344▼▼	0.0410▼▼	**0.0241**△△	**0.0275**▲▲
UserSimple-FW	**0.0435**△△	**0.0503**△▲	**0.0244**▲▲	**0.0276**▲▲
UserItem-FW	**0.0435**▼▲	**0.0501**△▲	**0.0245**▲▲	**0.0275**▲▲
RatItem-FW	**0.0436**▼▲	**0.0504**△▲	**0.0244**▲▲	**0.0275**▲▲
RatUser-FW	**0.0440**▼▲	**0.0509**▲▲	**0.0245**▲▲	**0.0276**▲▲
URItem-FW	**0.0429**▼▲	**0.0494**▲▲	**0.0244**▲▲	**0.0273**△▲
URUser-FW	**0.0438**▼▲	**0.0506**△▲	**0.0245**▲▲	**0.0274**△▲
URItemUser-FW	**0.0432**▼▲	**0.0498**△▲	**0.0245**▲▲	**0.0274**△▲

Table 7.9. Dynamic ensemble performance values (P@10) using uuU1R methodology with item predictors (MovieLens dataset).

In the other combination (HRI3) the best strategy is FW, the same as with the other evaluation methodologies.

7.3.2 Dynamic recommender ensembles on log data

In this section we present experiments in which log-based predictors are used to dynamically weight an ensemble's recommenders. As with rating-based information, in this case we first have to select suitable recommenders to combine according to the requirements established in our framework. Hence, we choose the combinations HL1, HL2 and HL3 presented in Table 7.10, where, as before, the performance predictors weight the recommender denoted as R1.

The Last.fm dataset contains timestamped log-based information. As noted in Chapter 4, for efficiency reasons, we only use the 1R methodology in this dataset. Table 7.11 shows the results obtained with a temporal split of the data, and Table 7.12 shows the results obtained with a random split (five-fold) of the data.

	R1	R2
HL1	kNN	CB
HL2	kNN	ItemPop
HL3	pLSA	kNN

Table 7.10. Selected recommenders for building dynamic ensembles using performance predictors that exploit log-based information (Last.fm dataset).

	HL1	HL2	HL3
R1 (λ=1.0)	0.0603	0.0603	$\underline{0.0926}$
R2 (λ=0.0)	$\underline{0.0916}$	0.0797	0.0603
Baseline (λ=0.5)	0.0852	0.0755	0.0820
Best static	0.0914	$\underline{0.0812}$	0.0925
(best λ)	(0.2)	(0.1)	(0.9)
Perfect correlation	0.0890	0.0783	0.0863
PC-OM	0.0869	0.0771	0.0851
PC-FW	0.0849	0.0751	0.0826
ItemSimple-OM	**0.0904**$^{\blacktriangledown}_{\blacktriangle}$	**0.0804**$^{\blacktriangledown}_{\blacktriangle}$	**0.0901**$^{\blacktriangledown}_{\blacktriangle}$
Autocorrelation-OM	0.0815$^{\blacktriangledown}_{\blacktriangledown}$	0.0722$^{\blacktriangledown}_{\blacktriangledown}$	0.0781$^{\blacktriangledown}_{\blacktriangledown}$
TimeSimple-OM	**0.0905**$^{\blacktriangledown}_{\blacktriangle}$	**0.0789**$^{\blacktriangledown}_{\blacktriangle}$	**0.0898**$^{\blacktriangledown}_{\blacktriangle}$
ItemTime-OM	**0.0906**$^{\blacktriangledown}_{\blacktriangle}$	**0.0804**$^{\blacktriangledown}_{\blacktriangle}$	**0.0902**$^{\blacktriangledown}_{\blacktriangle}$
ItemPriorTime-OM	**0.0885**$^{\blacktriangledown}_{\blacktriangle}$	**0.0778**$^{\blacktriangledown}_{\blacktriangle}$	**0.0863**$^{\blacktriangledown}_{\blacktriangle}$
ItemSimple-FW	**0.0903**$^{\blacktriangledown}_{\blacktriangle}$	**0.0802**$^{\blacktriangledown}_{\blacktriangle}$	**0.0891**$^{\blacktriangledown}_{\blacktriangle}$
Autocorrelation-FW	0.0842$^{\blacktriangledown}_{\blacktriangle}$	0.0746$^{\blacktriangledown}_{\blacktriangledown}$	0.0809$^{\blacktriangledown}_{\blacktriangledown}$
TimeSimple-FW	**0.0901**$^{\blacktriangledown}_{\blacktriangle}$	**0.0785**$^{\blacktriangledown}_{\blacktriangle}$	**0.0884**$^{\blacktriangledown}_{\blacktriangle}$
ItemTime-FW	**0.0904**$^{\blacktriangledown}_{\blacktriangle}$	**0.0800**$^{\blacktriangledown}_{\blacktriangle}$	**0.0891**$^{\blacktriangledown}_{\blacktriangle}$
ItemPriorTime-FW	**0.0883**$^{\blacktriangledown}_{\blacktriangle}$	**0.0775**$^{\blacktriangledown}_{\blacktriangle}$	**0.0855**$^{\blacktriangledown}_{\blacktriangle}$

Table 7.11. Dynamic ensemble performance values (P@10) using the 1R methodology with the log-based user predictors (Last.fm, temporal split).

	HL1	HL2	HL3
R1 (λ=1.0)	0.0204	0.0204	0.0836
R2 (λ=0.0)	$\underline{0.0828}$	$\underline{0.0767}$	0.0204
Baseline (λ=0.5)	0.0764	0.0643	0.0704
Best static	0.0818	$\underline{0.0767}$	$\underline{0.0837}$
(best λ)	(0.2)	(0.1)	(0.9)
Perfect correlation	0.0818	0.0760	0.0829
PC-OM	0.0816	0.0755	0.0823
PC-FW	0.0815	0.0745	0.0811
ItemSimple-OM	**0.0799**$^{\blacktriangledown}_{\blacktriangle}$	**0.0730**$^{\blacktriangledown}_{\blacktriangle}$	**0.0771**$^{\blacktriangledown}_{\blacktriangle}$
Autocorrelation-OM	0.0717$^{\blacktriangledown}_{\blacktriangledown}$	0.0596$^{\blacktriangledown}_{\blacktriangledown}$	0.0686$^{\blacktriangledown}_{\blacktriangledown}$
TimeSimple-OM	**0.0814**$^{\blacktriangledown}_{\blacktriangle}$	**0.0762**$^{\blacktriangledown}_{\blacktriangle}$	0.0518$^{\blacktriangledown}_{\blacktriangledown}$
ItemTime-OM	**0.0806**$^{\blacktriangledown}_{\blacktriangle}$	**0.0734**$^{\blacktriangledown}_{\blacktriangle}$	**0.0761**$^{\blacktriangledown}_{\blacktriangle}$
ItemPriorTime-OM	**0.0770**$^{\blacktriangledown}_{\blacktriangle}$	**0.0658**$^{\blacktriangledown}_{\blacktriangle}$	**0.0743**$^{\blacktriangledown}_{\blacktriangle}$
ItemSimple-FW	**0.0804**$^{\blacktriangledown}_{\blacktriangle}$	**0.0726**$^{\blacktriangledown}_{\blacktriangle}$	**0.0739**$^{\blacktriangledown}_{\blacktriangle}$
Autocorrelation-FW	0.0756$^{\blacktriangledown}_{\blacktriangledown}$	0.0631$^{\blacktriangledown}_{\blacktriangledown}$	0.0697$^{\blacktriangledown}_{\blacktriangledown}$
TimeSimple-FW	**0.0814**$^{\blacktriangledown}_{\blacktriangle}$	**0.0753**$^{\blacktriangledown}_{\blacktriangle}$	0.0579$^{\blacktriangledown}_{\blacktriangledown}$
ItemTime-FW	**0.0808**$^{\blacktriangledown}_{\blacktriangle}$	**0.0728**$^{\blacktriangledown}_{\blacktriangle}$	**0.0732**$^{\blacktriangledown}_{\blacktriangle}$
ItemPriorTime-FW	**0.0783**$^{\blacktriangledown}_{\blacktriangle}$	**0.0671**$^{\blacktriangledown}_{\blacktriangle}$	**0.0719**$^{\blacktriangledown}_{\blacktriangle}$

Table 7.12. Dynamic ensemble performance values (P@10) using the 1R methodology with log-based user predictors (Last.fm, five-fold random split).

We can see that the results of both tables are analogous. **The dynamic ensembles weighted by the log-based performance predictors outperform the baseline static ensemble in all cases, except with the Autocorrelation predictor**. This result is consistent with the correlations presented in Table 6.14 and Table 6.15, where autocorrelation obtained the lowest (absolute) correlation value for the kNN recommender on both versions of the dataset. Regarding the pLSA recommender (in the combination HL3), the Autocorrelation and TimeSimple predictors obtain com-

	R1	R2
HS1	Personal	pLSA
HS2	Personal	kNN
HS3	PureSocial	pLSA
HS4	PureSocial	kNN

Table 7.13. Selected recommenders for building dynamic ensembles using social-based user predictors (CAMRa dataset).

parable correlations with the combined recommenders, yet the performance of the corresponding dynamic ensembles is very different, thus suggesting that, although we have found a dependence between the predictors' power in terms of correlation, and their effectiveness in weighting hybrids, this is not a strict necessary condition to obtain improvements over the static ensembles.

The best performance values were achieved either by single recommenders or by the best static ensembles. When the best results are obtained by single recommenders emphasises the fact that no hybridisation is required for that combination (like in HL1 and HL3 for the temporal split, and HL1 and HL2 for the random split). In the other case, when the best results are achieved by the best static ensembles, it may restrict the usefulness of our approach, although our proposed dynamic ensembles significantly outperform the baseline static ensembles for some predictors such as TimeSimple and ItemSimple. We have to recall that the best static ensembles are in fact optimised using the test set, which is clearly not a fair comparison. The results of the perfect correlation ensembles in the random split are always better than those obtained by the performance predictors, confirming that predictors with stronger correlations should obtain better performance results when used for dynamic ensembles.

7.3.3 Dynamic recommender ensembles on social data

In the third experiment we exploit the social information available in the CAMRa dataset to combine collaborative and social filtering recommenders using social-based performance predictors. Table 7.13 shows the recommender combinations selected based on the correlations obtained in Section 6.5.4. Here, we present 4 ensembles where the two social filtering recommenders, Personal and PureSocial, are combined with two collaborative filtering recommenders, pLSA and kNN. We saw in Section 6.5.4 that most of the social-based predictors obtained higher correlations with the social filtering recommenders, and lower or negligible correlations with the collaborative filtering recommenders, at least for the social version of the dataset (Table 6.16). The situation for the collaborative-social version was not so clear, but for the sake of coherence, we use the same set of ensembles in both versions of the dataset.

As we mentioned in Section 6.5.4, due to the lack of coverage of the social filtering recommenders, the only methodology that provides sensible results is the AR methodology. In this section we present the results obtained using this methodology on the two available versions of the CAMRa dataset: social and collaborative-social.

Table 7.14 shows the results obtained on the social version of the CAMRa dataset. We see that only for one out of the four recommender combinations, the dynamic ensembles consistently outperform the baseline static ensemble. However, it is interesting to note that the best value is always achieved by the perfect correlation ensemble, which means that further improvements could be possible if we were able to find predictors with stronger correlations.

In the collaborative-social version of the dataset (Table 7.15) the results are similar, except that now for HS2, the best result is obtained by the best static ensemble. Moreover, a larger number of dynamic ensembles outperform the baseline static ensemble HS3, whereas at least one dynamic ensemble outperforms the baseline HS1, which is a better result than the one shown in the previous Table 7.14. We hypothesise this is because on this version of the dataset the individual recommenders display a more similar performance to each other (compare the differences between R1 and R2 in Table 7.14 and Table 7.15).

Furthermore, some of the correlations obtained for the CAMRa collaborative

	HS1	HS2	HS3	HS4
R1 (λ=1.0)	0.1066	0.1066	0.1072	0.1072
R2 (λ=0.0)	0.1007	0.0226	0.1007	0.0226
Baseline (λ=0.5)	0.1509	0.1142	0.1599	0.1219
Best static (best λ)	0.1524 (0.4)	0.1200 (0.7)	0.1632 (0.3)	0.1219 (0.5)
Perfect correlation	0.1608	0.1188	0.1640	0.1237
PC-OM	0.1202	0.1164	0.1254	0.1199
PC-FW	0.1189	0.1143	0.1263	0.1219
AvgNeighDeg-OM	0.1489▽▽	**0.1195**▽△	0.1599▽▽	0.1131▼▼
BetCentrality-OM	0.1443▼▼	0.1132▼▽	0.1487▼▼	0.1114▼▼
ClustCoeff-OM	0.1465▼▽	0.1123▼▽	0.1483▼▼	0.1108▼▼
Degree-OM	0.1472▽▽	**0.1154**▽△	**0.1614**▽△	0.1107▼▼
EgoCompSize-OM	0.1461▽▽	**0.1158**▽△	0.1596▽▽	0.1140▼▼
HITS-OM	0.1485▽▽	**0.1200**▲	0.1467▼▼	0.1134▼▼
PageRank-OM	0.1471▼▽	**0.1167**▽△	0.1579▽▽	0.1123▼▼
TwoHopNeigh-OM	0.1478▽▽	**0.1171**▽△	0.1585▼▽	0.1118▼▼
AvgNeighDeg-FW	**0.1518**▽△	**0.1191**▽▲	**0.1623**▽△	0.1204▼▼
BetCentrality-FW	0.1491▼▽	**0.1180**▽▲	0.1577▼▽	0.1213▼▼
ClustCoeff-FW	0.1500▽▽	**0.1182**▽▲	0.1566▼▼	0.1189▼▼
Degree-FW	0.1489▽▽	**0.1191**▽▲	**0.1627**▽△	0.1208▼▼
EgoCompSize-FW	0.1489▽▽	**0.1193**▽▲	**0.1618**▽△	0.1210▼▼
HITS-FW	0.1482▼▼	**0.1195**▽▲	0.1564▼▼	0.1202▼▼
PageRank-FW	0.1491▼▽	**0.1186**▽▲	**0.1610**▽△	0.1211▼▼
TwoHopNeigh-FW	0.1500▽▽	**0.1195**▽▲	**0.1619**▽△	0.1211▼▼

TaTable 7.15. Dynamic ensemble performance values (P@10) using the AR methodologyith socwith social-based user predictors (CAMRa, collaborative dataset).

dataset are more discriminative between the combined recommenders, in the sense that, for instance, the correlations between the two-hop neighbourhood predictor and the Personal recommender were -0.123 and -0.121 in the social and collaborative-social datasets, respectively. However, the correlations between the two-hop neighbourhood predictor and kNN were 0.004 and 0.130, that is, in the second dataset the relative distance in correlation between these two recommenders is larger, according to the correlation with respect to the predictor. This change in the correlations may explain the fact that in Table 7.15 some of the dynamic ensembles outperform the perfect correlation ensemble, which does not take the relative correlation into account with respect to each individual recommender, as noted in 7.3.1.

In general, **the HITS predictor obtains the best results among the dynamic ensembles for some of the tested combinations. Other predictors such as the betweenness centrality and the ego components size produce more competitive ensembles in the social version of the dataset**, whereas the degree and the average neighbour degree preditors provide better results for more than one combination in the CAMRa collaborative dataset.

7.3.4 Discussion

The analysis of the results presented in this chapter shows that ensembles can indeed benefit from a dynamic weighting of their recommenders. In particular, we have seen that when these weights come from performance predictors, which previously had shown significant correlation with the performance of individual recommenders, the resulting dynamic ensemble tends to outperform static combinations of the recommenders. In this context, in order to obtain successful hybridisations, we have to take several variables into account, which correspond to three stages proposed in our framework: the correlation between the predictor and the combined recommenders, the relative performance of such recommenders, the strategy to normalise the predictor's values, and the weight distribution among recommenders.

The relative performance of the recommenders has proven to be decisive, since in some cases, hybridisation does not make sense to begin with, when the difference in performance between the recommenders is significant and systematic, and thus, dynamic ensembles cannot obtain the best performance result, although they may outperform static ensembles. Performance prediction normalisation and weight distribution, on the other hand, do make a difference in the results. Although no explicit results are presented in this work regarding different normalisation approaches, previously conducted experiments showed us that score normalisation produce worse results than rank normalisation. Finally, the weight distribution strategy is not as critical as other stages of our framework, but helps to obtain much better results, specifically, when the one minus strategy (OM) is used.

The obtained results have also shown that more complex formalisations and probability models do not necessarily lead to better results, with respect to the adaptation and definition of the user and item clarity performance predictors. In this adaptation, various configurations were available, and we experimented with further extensions of different language models for the same clarity model, using rating and log-based information. Additionally, several graph-based metrics were tested, where the concept of the user's strength in a social network is modelled in different ways.

We find that different formulations for the user-based performance clarity predictor consistently obtain the best results in different situations for rating-based preference information. We also experimented with item-based predictors, and found that the UserItem, URItem, and RatUser predictors were noticeably better than the rest of the formulations. When log-based information is exploited, the ItemTime and TimeSimple predictors obtained better results than other predictors not based on the clarity concept, such as the Autocorrelation function. Moreover, regarding the social-based ensembles, the HITS, two-hop neighbourhood, and average neighbour degree approaches clearly outperform the ensemble weighted by the rest of the predictors and, in most of the cases, also outperform the baseline static ensemble.

These results are, in general, consistent with the correlation values between the predictors' output values and the recommenders' performance values. Figure 7.2 shows a summary of the results presented in this and previous chapters, where the difference in correlation is plotted against the gain (or loss) in performance with respect to the baseline. For this figure, the best and worst dynamic ensembles were

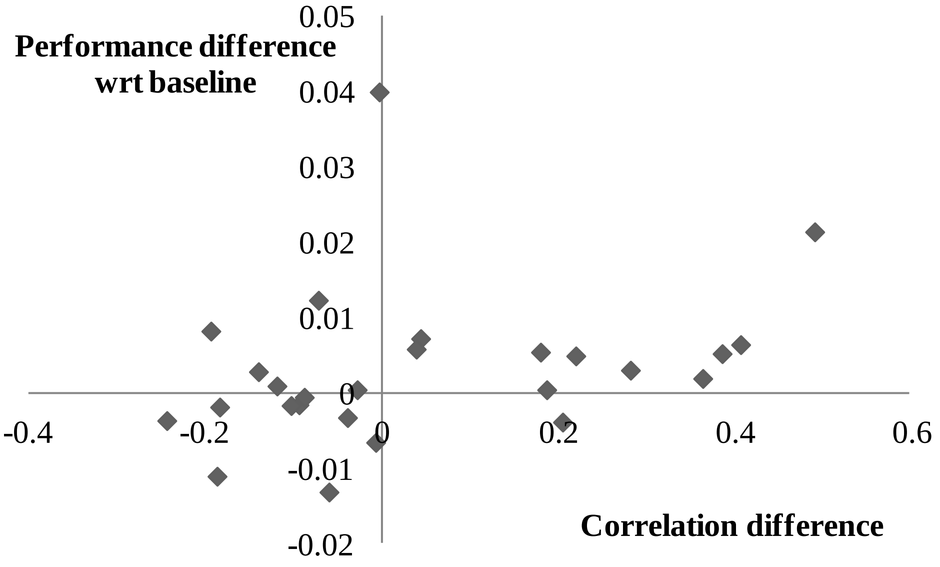

Figure 7.2. For each best and worst dynamic ensemble in Table 7.2, Table 7.11 and Table 7.15, this graph plots the difference in correlation between each predictor and a recommender against the difference in performance between the ensemble and the baseline.

selected from Table 7.2, Table 7.11 and Table 7.15. In the figure we may observe the trend that the larger the difference in correlation, the better the improvement over the baseline, which is in concordance with the requirement that both correlations should not be very similar. These results provide some insights in order to understand which features may help configure well performing dynamic recommender ensembles, where performance predictors have emerged as a clear useful characteristic.

7.4 Conclusions

In this chapter we have explored how the performance of a recommender ensemble can be improved by dynamically assigning the weights of its recommenders, by analysing the performance correlation between the values of a performance predictor and the performance of an individual recommender. In this way, we have proposed a dynamic hybrid framework that let decide when and how dynamic hybridisation should be done.

Drawing from the performance predictors proposed in the previous chapter, we have conducted several experiments in order to assess whether recommender ensembles can benefit from dynamic weights according to such predictors. The results obtained in our experiments indicate that a strong correlation with performance tends to correspond with enhancements in ensembles by using the predictor for weight adjustment. The dynamic ensembles usually outperformed the baseline static ensemble for different recommender combinations, supporting their effectiveness in different situations.

In future work we aim to evaluate our framework with more than two recommenders in an ensemble, and more than one performance predictor, eventually, one for each recommender. We also plan to test different normalisation strategies of the predictor's values, where several assumptions about the ideal weight distribution can be verified, such as whether the user's rating distribution or the recommender's output are beneficial for the final performance of the ensemble. Moreover, Machine Learning approaches could also be used to learn the best weights in a user (or item) basis. Despite being more time consuming, these techniques may also achieve good results in terms of performance of the dynamic ensemble, although they are usually more prone to overfit the learned weights.

Chapter 8

Neighbour selection and weighting in user-based collaborative filtering

User-based recommender systems suggest interesting items to a user relying on similar-minded people called neighbours. The selection and weighting of the input from these neighbours characterise different variants of the approach. Thus, for instance, while standard user-based collaborative filtering strategies select neighbours based on user similarities, trust-aware recommendation algorithms rely on other aspects indicative of user trustworthiness and reliability.

In this chapter we restate the user-based recommendation problem, generalising it in terms of performance prediction techniques. We investigate how to adopt this generalisation to define a unified framework where we conduct an objective analysis of the effectiveness (predictive power) of neighbour scoring functions. We evaluate our approach with several state-of-the-art and novel neighbour scoring functions on two publicly available datasets. The notion of performance takes here a different nuance from previous chapters. More precisely, we consider the notion of neighbour performance, for which we propose several measures and new predictors. In an empirical comparison involving four neighbour quality metrics and thirteen performance predictors, we find a strong predictive power for some of the predictors with respect to certain metrics. This result is then validated by checking the final performance of recommendation strategies where predictors are used for selecting and/or weighting user neighbours. As a result, we are able to anticipate which predictors will perform better in neighbour scoring powered versions of a user-based collaborative filtering algorithm.

In Sections 8.1 and 8.2 we present a unified formulation and the proposed framework for neighbour selection and weighting in user-based recommendation, and in Section 8.3 we describe how the different neighbour scoring functions proposed in the literature fit into the framework. Finally, in Section 8.4 we present an experimental evaluation of the framework, and in Section 8.5 we provide conclusions.

8.1 Problem statement

We focus on user-based collaborative filtering algorithms, one type of memory-based approaches that explicitly seek people – commonly called **neighbours** – having preferences (and/or other characteristics of interest) in common with the target user, and use such preferences to predict item ratings for the user. User-based algorithms are built on the principle that a particular user's rating records are not equally useful to all other users as input to provide them with item suggestions (Herlocker et al., 2002). Therefore, as stated in Chapter 2, central aspects to these algorithms are a) how to identify which neighbours form the best basis to generate item recommendations for the target user, and b) how to properly make use of the information provided by them. Once the target user's neighbours are selected, the more similar a neighbour is to the user, the more her preferences are taken into account as input to produce recommendations.

A common user-based recommendation approach consists of predicting the relevance of an item for the target user by a linear combination of her neighbours' ratings, which are weighted by the similarity between the target user and her neighbours, as presented in Equation (2.3). For the sake of clarity, and since we shall later elaborate from it, we reproduce here the above equation:

$$\tilde{r}(u,i) = \bar{r}(u) + C \sum_{v \in N_k(u,i)} sim(u,v)\big(r(v,i) - \bar{r}(v)\big) \tag{8.1}$$

User similarity has been the central criterion for neighbour selection in most of the user-based collaborative filtering approaches (Desrosiers and Karypis, 2011). Nonetheless, recently it has been suggested that additional factors could have a valuable role to play on this point. For instance, two users with a high similarity value may no longer be reliable predictors for each other at some point because of a divergence of tastes over time (O'Donovan and Smyth, 2005). Thus, in the context of user-based collaborative filtering, more complex methods have been proposed in order to effectively select and weight useful neighbours (O'Donovan and Smyth, 2005; Desrosiers and Karypis, 2011). In this context a particularly relevant dimension relates the above additional factors with the general concept of trust (trustworthiness, reputation) on a user's contribution to the computation of recommendations. Hence, a number of trust-aware recommender systems have been proposed in the last decade (Hwang and Chen, 2007; O'Donovan and Smyth, 2005; Golbeck, 2009).

Most of these systems focus on the improvement of accuracy metrics, such as the Mean Average Error, by defining different heuristic trust functions, which, in most cases, are applied either as additional weighting factors in the neighbourhood-based formulation, or as a component of the neighbour selection criteria. The way trust is measured is considerably diverse in the literature. In fact, the notion of trust has embraced a wide scope of neighbour aspects, spanning from personal trust on the neighbour's faithfulness, to trust on her competence, confidence in the correctness of the input data, or the effectiveness of the recommendation resulting from the neighbour's data. More specifically, in trust-aware recommender systems, a trust model is defined and, typically, introduced into the Resnick's equation (Equation (8.1)) either as an additional weight or as a filter for the potential user's neighbours. Moreover, depending on the nature of their input, different types of trust-aware recommendation approaches can be distinguished: rating-based approaches, and social-based approaches (using a trust network).

One of the first works that proposed *rating-based trust metrics* between users is (O'Donovan and Smyth, 2005). In that work O'Donovan and Smyth propose to modify how the "recommendation partners" (neighbours) are weighted and selected in the user-based collaborative filtering formula. They argue that the trustworthiness of a particular neighbour should be taken into account in the computed recommendation score by looking at how reliable her past recommendations were. Trust values are computed by measuring the amount of correct recommendations in which a user has participated as a neighbour, and then they are used for weighting the influence (along with computing the similarity), and selecting the target user's neighbours. Weng et al. (2006) propose an asymmetric trust metric based on the expectation of other users' competence in providing recommendations to reduce the uncertainty in predicting new ratings. The metric is used in the standard collaborative filtering formula instead of the similarity value. Two additional metrics are defined in (Kwon et al., 2009) based on the similarity between the ratings of a neighbour and the ratings from the community. Finally, Hwang and Chen (2007) define two trust metrics (local and global) by averaging the prediction error of co-rated items between a user and a potential neighbour.

Social-based trust metrics make use of explicit trust networks of users, built upon friendship relations (Massa and Avesani, 2004; Massa and Bhattacharjee, 2004) and explicit trust scores between individuals in a system (Ma et al., 2009; Walter et al., 2009). These metrics and, to some extent, their inherent meanings, are different with respect to rating-based metrics. Nonetheless, Ziegler and Lausen (2004) conduct a thorough analysis that shows empirical correlations between trust and user similarities, suggesting that users tend to create social connections with people who have similar preferences. Once such a correlation is proved, techniques based on social-based trust can be applicable. Golbeck and Hendler (2006) propose a metric called TidalTrust to infer trust relationships by using recursive search. Inferred trust values are used for every user who has rated a particular item in order to select only those users with high trust values. Then, a weighted average between past ratings and inferred trust values provides the predicted ratings. Massa and Avesani (2007b) experiment with local (MoleTrust) and global (PageRank) trust metrics, showing that trust-based recommenders are very valuable for cold start users.

The research presented here seeks to provide an algorithmic generalisation for a significant variety of notions, computational definitions, and roles of trust in neighbour selection. Specifically, we aim to provide a theoretical framework for neighbour selection and weighting in which trust metrics can be defined and evaluated in terms of improvements on a final recommender's performance. We cast the rating prediction task – typically based, as described above, on the aggregation of neighbour preferences – into a framework for dynamic combination of inputs, from a performance prediction perspective, borrowing from the methodology for this area in the Information Retrieval field. The application of this perspective is not trivial, and requires a definition of what the performance of a neighbour means in this context. Hence, restated the problem in these terms, we propose to adapt and exploit techniques and methodologies developed in Information Retrieval for predicting query performance; in our case the target user's neighbours are equivalent to the queries, and our goal is to predict which of these neighbours will perform better for the target user.

Furthermore, since our framework provides an objective measure of the neighbour scoring function efficiency, we would be able to obtain a better understanding of the whole recommendation process. For instance, if the results obtained when a particular function is introduced in a recommender are not consistent with the (already observed) objective performance measures, it would mean that the chosen strategy is not the most appropriate, suggesting to experiment with further strategies, providing such a function has already shown some predictive power.

Therefore, the main contribution of our framework is that it provides a formal setting for the evaluation of neighbour selection and weighting functions, while, at the same time, enables to discriminate whether recommendation performance improvements are achieved by the neighbour scoring functions, or by the way these functions are used in the recommendation computation. Besides, our framework provides an unification of state-of-the-art trust-based recommendation approaches, where trust metrics are casted as neighbour performance predictors. As a result, in this chapter, we shall propose four neighbour quality metrics and thirteen performance predictors, defined upon a specific neighbour (user-based), a neighbour and the current user (user-user), or a neighbour and the current item (user-item). We shall generalise the different strategies proposed in the literature to introduce trust into collaborative filtering. Moreover, thanks to the proposed formulation, we will define and evaluate new strategies.

8.2 A performance prediction framework for neighbour scoring

8.2.1 Unifying neighbour selection and weighting in user-based Recommender Systems

From the observation that most of the methods for neighbour selection and weighting are elaborated upon the standard Resnick's scheme (Equation (8.1)), we propose a unified formulation as follows. Let us suppose, for the sake of generality, that we have a neighbour scoring function $s(u, v, i)$ that may depend on the target user u, a neighbour v, and a target item i. This function outputs a higher value whenever the user, neighbour, item, or a combination of them, is more trustworthy (in the case of trust models), or is expected to perform better as a neighbour according to the information available in the system, such as other ratings and external information, like a social network. Using this function we generalise Equation (8.1) to:

$$\tilde{r}(u,i) = \bar{r}(u) + C \sum_{v \in f^{neigh}(u,i;k;s)} f^{agg}\big(s(u,v,i), sim(u,v)\big)\big(r(v,i) - \bar{r}(v)\big) \quad (8.2)$$

where the function f^{neigh} denotes the selection of the set of neighbours, and f^{agg} is an aggregation function combining the output of s and the user similarity into a single weight value. In this way, we integrate the neighbour scoring function s into the Resnick's formula in order to: a) select the neighbours to be considered, instead of or in addition to the most similar users (via function f^{neigh}), and b) provide a general weighting scheme by introducing an aggregation function f^{agg} between the actual neighbour score and the similarity between the target user and her neighbours. Note that it is not required that s is bounded, since a constant C would normalise the output rating value. The function s is thus a core component in the generalisation of the user-based collaborative filtering techniques. It may embody similarity in itself (in such case f^{agg} may just return its first input argument), but sim and f^{agg} are left to simplify the connection with the original similarity-only formulation, and to suit particular cases where s applies other principles distinct to similarity.

The aggregation function f^{agg} can take different definitions, some examples of which can be found in the literature. For instance, O'Donovan and Smyth (2005) initially propose to use the arithmetic mean of the neighbour score (x) and the similarity (y; henceforth denoted as f_1^{agg}), and end up using the harmonic mean (f_2^{agg}) because of its better robustness to large differences in the inputs. In (Bellogín and Castells, 2010), on the other hand, we use the product function (f_3^{agg}). Moreover, Hwang and Chen (2007) propose to directly use the neighbour score as the weight given to neighbours, that is, they use the projection function $f_4^{agg}(x,y) = x$. Obviously, the original Resnick's formulation can be expressed as the symmetric projection function $f_0^{agg}(x,y) = y$.

The neighbourhood selection embodied in function f^{neigh} also generalises Resnick's approach – the latter corresponds to the particular case $f_0^{neigh}(u,i;k;s) = N_k(u,i)$, where the neighbour scoring function is ignored, and only similarity is used. The general form admits different instantiations. In (Golbeck and Hendler, 2006) only the users with the highest trust values are selected as neighbours. In (O'Donovan and Smyth, 2005), on the other hand, those users whose trust values exceed a certain threshold are taken into consideration. This threshold is empirically defined as the mean across all the obtained values for each pair of users. The latter strategy can be formulated as follows:

$$f_1^{neigh}(u,i;k;s) = \{v \in N_k(u,i): s(u,v,i) > \tau\};\ \tau = \frac{1}{|\{(u,v,i)\}|} \sum_{(u,v,i)} s(u,v,i)$$

There are, nonetheless, some considerations to take into account when using specific combinations of neighbour weighting and neighbour selection functions. First, if f_4^{agg} is used together with f_0^{neigh} – only considering the most similar users in the neighbourhood –, then less reliable users (with low f_4^{agg}) who are very similar to the current user would be penalised, and more reliable neighbours but less similar to the current user are ignored, since they do not belong to the neighbourhood. Second, when using f_0^{agg} together with f_1^{neigh}, neighbours are weighted by their similarities with the target user. These similarities, however, could be very low, and thus, non-similar but reliable neighbours would be penalised. Finally, if f_4^{agg} is used with f_1^{neigh}, the similarity weight will not be considered at any point in the recommendation process.

Some of these configurations may deserve further investigation, and are considered in Section 8.4, along with other combinations not listed here.

8.2.2 Neighbour selection and weighting as a performance prediction problem

Neighbour scoring and selection can be seen as a task of predicting the effectiveness of neighbours as input for collaborative recommendations. In this section we elaborate and adapt the performance prediction framework presented in Chapter 5 to the problem of neighbour selection and weighting.

The same as performance prediction in Information Retrieval, which has been used to optimise rank aggregation (Yom-Tov et al., 2005a), in our proposed framework each user's neighbour can be considered as a retrieval subsystem (or criterion) whose output is combined to form a final system's output (the recommendations) to the user.

For user-based collaborative filtering algorithms, the estimation $\tilde{r}(u,i)$ of the preference of the target user u for a particular item i can be formulated as an aggregation function of the ratings of some other users $\hat{V}$:

$$\tilde{r}(u,i) \propto \text{aggr}_{v\in\hat{V}}\big(sim(u,v);\ r(v,i); \bar{r}(u); \bar{r}(v)\big) \tag{8.3}$$

where $\hat{V}$ denotes the selected neighbours for a particular user u according to function f^{neigh} (see Equation (8.2)). As observed in (Adomavicius and Tuzhilin, 2005), different aggregation functions can be defined, but the most typical one is the weighted average function presented in the previous section.

In the previous function the term $\tilde{r}(u,i)$ can be seen as a retrieval function that aggregates the outputs of several utility subfunctions $r(v,i) - \bar{r}(v)$, each corresponding to a recommendation obtained from a neighbour of the target user. The combination of utility values is defined as a linear combination (translated by $\bar{r}(u)$) of the neighbours' ratings, weighted by their similarity $sim(u,v)$ with the target user. Hence, the computation of utility values in user-based filtering is equivalent to a typical rank aggregation model of Information Retrieval, where the aggregated results may be enhanced by predicting the performance of the combined recommendation outputs. In fact, the similarity value can be seen as a prediction of how useful a neighbour's advice is expected to be for the target user, which has proved to be a quite effective approach. The question is whether other performance factors beyond user similarity can be considered in a way that further enhancements can be drawn, as research on user trust awareness has attempted to prove in the last years.

The Information Retrieval performance prediction view provides a methodological approach, which we propose to adapt to the neighbour selection problem. The approach provides a principled path to drive the formulation, development and evaluation of effective neighbour selection and weighting techniques, as we shall see. In the proposed view, the selection/weighting problem is expressed as an issue of neighbour performance, as an additional factor (besides user similarity) to automatically tune the neighbours' contribution to the recommendations, according to the expected goodness of their advice. As summarised in Section 5.1, there are three core concepts in the performance prediction problem as addressed in the Information Retrieval literature: performance predictor, retrieval quality assessment, and predictor quality assessment. Since we are dealing with the prediction of which users may perform better as neighbours, the above three concepts can respectively be translated into *neighbour performance predictor*, *neighbour quality*, and *neighbour predictor quality*. For the sake of simplicity, let us assume we can define a performance predictor as a function that receives as input a user profile u (in general, it could receive other users or items as well), the set of items $\mathcal{I}_u$ rated by that user, and the collection S of ratings and items (or any other user preference and item description information) available in the system. Then, following the notation given used in Chapter 5, we define a neighbour performance prediction function as:

$$\hat{\mu}(u) \leftarrow \gamma(u, \mathcal{I}_u, S). \tag{8.4}$$

The function γ can be defined in different ways, for instance, by taking into account the rating distribution of each user, the number of ratings available in the system, and the (implicit or explicit) relations made by that user with the rest of the community. Essentially, the neighbour performance predictor is intended to estimate the true neighbour quality metric, denoted as $\mu(u)$, which is typically measured using groundtruth information about whether the neighbour's influence is positive. The application of this perspective is not trivial, and requires, in particular, a definition of what the performance of a neighbour means in this context – where no standard metric for neighbour performance is yet available in the literature.

Once the estimated neighbour performance prediction values $\hat{\mu}(u_n)$ are computed for all users, the quality of the prediction can be measured as presented in Section 5.4.2, that is, either by measuring the correlation between the estimations and the real values $\mu(u_n)$, or by using classification accuracy metrics such as the F-measure. Since in this case we are interested in providing a ranking of users, this relates more with the traditional query performance task, and not with query difficulty (see Section 5.4.1), where the latter metrics are used. In other words, the neighbour predictor quality metric is defined as the following correlation:

$$q(\gamma) = \mathrm{corr}([\hat{\mu}(u_1), \cdots, \hat{\mu}(u_n)], [\mu(u_1), \cdots, \mu(u_n)]). \tag{8.5}$$

Similarly to the situation in Information Retrieval, this correlation provides an assessment of the prediction accuracy (Carmel and Yom-Tov, 2010); the higher its (absolute) value, the higher the predictive power of γ. Moreover, the sign of $q(\gamma)$ represents whether the two involved variables – neighbour prediction and neighbour quality – are directly or inversely correlated.

Besides validating any proposed predictor by checking the correlation between predicted outcomes and objective metrics, we may further test the effectiveness of the defined predictors by introducing and testing a dynamic variant of user-based collaborative filtering. In this variant, the weights of neighbours are dynamically adjusted based on their expected effectiveness, along with the decision of which users belong to each neighbourhood, as in the general formulation presented in Equation (8.2). We propose to define the neighbour scoring function $s(u, v, i)$ based on the values computed from each neighbour performance predictors.

Hence, the basic idea of the framework presented here is to formally treat the neighbour selection and weighting in memory-based recommendation as a performance prediction problem. The performance prediction framework provides a principle basis to analyse whether the predictors are capturing some valuable, measurable characteristic known to be useful for prediction, independently from their latter use in a recommendation strategy. Furthermore, if a neighbour scoring function with strong predictive power is introduced into the recommendation process and the performance is not improved, then, new ways of introducing such predictor into the rating estimation should be tested (either for selection or weighting), since we have some confidence that this function captures interesting user's characteristics, valuable for recommendation.

8.3 Neighbour quality metrics and performance predictors

The performance prediction research methodology requires a means to compare the predicted performance with the observed performance. This comparison is typically conducted in terms of some one-dimensional functional values, where the performance is assessed by some specific metric and the prediction can be translated to a certain numeric value. This value quantifies the expected degree of effectiveness, providing, thus, a relative magnitude.

Whereas in the context of performance prediction in IR, standard metrics of system effectiveness in response to a query are used for this purpose, in the case of predicting the performance of a neighbour for recommendation we would require to use metrics that measure how effective a neighbour is. In this section we propose several neighbour quality metrics and performance predictors which we shall evaluate in Section 8.4.

8.3.1 Neighbour quality metrics

The purpose of effectiveness predictors in our framework is to assess how useful specific neighbour profiles are as a basis for predicting ratings for the target user. Each predictor has to be contrasted to a measure of how "good" the neighbour's contribution is to the global community of users in the system. In contrast with query performance prediction, where a well established array of metrics are used to quantify query performance, to the best of our knowledge, in the literature there is not an equivalent function for neighbours used in user-based collaborative filtering. We therefore need to introduce and propose some sound candidate metrics.

Ideally, in the proposed framework, a quality metric should take the same arguments as the predictor, and thus, if we have, for instance, a user-item predictor, we should also be able to define a quality metric that depends on users and items. In general, we shall focus on user-based predictors, but it would be possible to explore item-based alternatives. Furthermore, we shall consider metrics taking neighbours as single input, independently from which neighbourhood is involved (i.e., independently from the target user), and which item is recommended. At the end of this section, nonetheless, we shall introduce a neighbour quality metric suitable for the user-user scenario, where both the target user and neighbour are taken into account.

Now, we propose three different neighbour quality metrics. The first two metrics had a different intended use by their authors, but we found they could be useful to evaluate how good a user is as a neighbour. The third metric was proposed by us in (Bellogín and Castells, 2010), where the problem of neighbour performance was explicitly addressed.

Rafter et al. (2009) propose two metrics in order to examine whether the neighbours have any influence in the recommendation accuracy. Both metrics are based on the comparison between true ratings and a neighbour's estimation of the ratings, as a way to measure the direction of the neighbour estimation and the average absolute magnitude of the shift produced by this estimation. Thus, the larger the neighbour's influence, the better her performance, according to our definition of a "good" neighbour. In this context we use those metrics as follows:

$$\mu_1 = \mu(v) = \frac{1}{|T_v|}\sum_{i \in T_v} \frac{1}{|N_k^{-1}(v;i)|} \sum_{w \in N_k^{-1}(v;i)} |r(w,i) - r(v,i)|$$

$$\mu_2 = \mu(v) = \frac{1}{|T_u|}\sum_{i \in T_v} \frac{1}{|N_k^{-1}(u;i)|} \sum_{w \in N_k^{-1}(v;i)} \delta\big(\big[sgn\big(r(w,i) - \bar{r}(v)\big) = sgn\big(r(v,i) - \bar{r}(v)\big)\big]; 1\big)$$

where δ is a binary function whose output is 1 if its arguments are true, and 0 otherwise. Metric μ_1 represents the **absolute error deviation** of a particular user, and μ_2 is the **sign of error deviation**. Note that $N_k^{-1}(v;i)$ denotes an inverse neighbourhood, which represents those users for whom v is a neighbour, and T_v denotes the items rated by user v in the test set. We can observe how each of these metrics represents a different method to measure how accurate the user v is as a neighbour.

In (Bellogín and Castells, 2010) we proposed a metric named **neighbour goodness**, which is defined as the difference in performance of the recommender system when including vs. excluding the user (i.e., her ratings) from the dataset. For instance, based on the mean average error standard metric, neighbour goodness can be instantiated as:

$$\mu_3 = \mu(v) = \frac{1}{\left|R_{\mathcal{U}\setminus\{v\}}\right|} \sum_{w \in \mathcal{U}\setminus\{v\}} \left[CE_{\mathcal{U}\setminus\{v\}}(w) - CE_{\mathcal{U}}(w)\right]$$

$$CE_X(v) = \sum_{i \in \mathcal{I}, r(v,i) \neq \emptyset} |\tilde{r}_X(v,i) - r(v,i)|$$

where $\tilde{r}_X(v,i)$ represents the predicted rating computed using only the data in X. This metric quantifies how much a user affects (contributes to or detracts from) the total amount of mean average error of the system, since it is computed in the same way as that metric, but leaving out the user of interest – in the first term, the user is completely omitted; in the second term, the user is only involved as a neighbour. In this way we measure how a user contributes to the rest of users, or put informally, how better or worse the "world" is in the sense of how well recommendations work with and without the user. Hence, if the error increases when the user is removed from the dataset, it is considered as a good neighbour.

Based on the same idea of the previous metric, we propose a user-user quality metric that measures how one particular user affects to the error of another user when acting as her neighbour:

$$\mu_4 = \mu(u, v) = CE_{\mathcal{U}\setminus\{v\}}(u) - CE_{\mathcal{U}}(u)$$

We call this metric **user-neighbour goodness**. It quantifies the difference in user u's error when neighbour v is not in the system against the error when such neighbour is present, that is, it measures how much each neighbour contributes to reduce the error of a particular user.

8.3.2 Neighbour performance predictors

Having formulated neighbour selection in memory-based recommendation as a task of neighbour effectiveness prediction, and having proposed effectiveness metrics to compare against, the core of an approach to this problem is the definition of effectiveness predictors. For this purpose, similarity functions and trust models such as those mentioned in Section 8.1 can be directly used, since in trust-aware recommendation, trust metrics aim at measuring how reliable a neighbour is when introduced in the recommendation process (O'Donovan and Smyth, 2005). Interestingly, some of them only depend on one user (**global trust metrics**), and others depend on a user and an item or another user (**local trust metrics**). Furthermore, other authors have proposed different indicators for selecting good neighbours, mainly based on the overlap between the user and her neighbour, without considering the concept of trust.

We thus distinguish three types of neighbour performance predictors: **user predictors** – equivalent to the global trust metrics –, **user-item predictors**, and **user-user predictors** – equivalent to the local trust metrics. Note that, although trust metrics could now be interpreted as neighbour performance predictors, the proposed performance prediction framework let us to provide an inherent value to these metrics (identified as performance predictors), independently from whether they improve a recommender's performance when used for selecting or weighting in the specific collaborative filtering algorithm. This is due to the fact that it is possible to empirically check the quality of the prediction by analysing their correlation with respect to the neighbour performance metric, prior to the integration in any collaborative filtering method. Thus, each predictor would obtain an explicit score that represents its predictive power, related to our *a priori* confidence on whether such predictor is capturing the neighbour's reliability or trustworthiness.

In the following we propose an array of neighbour effectiveness prediction methods, by adapting and integrating trust functions from the literature into our framework, and we also propose novel prediction functions.

User Predictors

User predictors are performance predictors that only depend on the target neighbour. When that neighbour is predicted to perform well, her assigned weight in the user-based collaborative filtering formulation is high.

One of the first user trust metrics proposed in the literature is the **profile-level trust** (O'Donovan and Smyth, 2005), which is defined as the percentage of correct recommendations in which a user has participated as a neighbour. If we denote the set of recommendations in which a user has been involved as

$$\text{RecSet}(u) = \{(v, i) \colon u \in N_k(v; i)\},$$

then the predictor is defined as follows:

$$\gamma_1(u, v, i) = \gamma(v) = \frac{|\text{CorrectSet}(v)|}{|\text{RecSet}(v)|},$$

where the definition of correct recommendations depends on a threshold ϵ:

$$\text{CorrectSet}(u) = \{(c_k, i_k) \in \text{RecSet}(u) \colon \text{Correct}(i_k, u, c_k; 1)\}$$
$$\text{Correct}(i, u, v; \lambda) = \delta(|r(u, i) - r(v, i)| \leq \epsilon; \lambda),$$

$\delta(a; b)$ being a binary function like before whose output is a value b if the predicate a is true, and 0 otherwise. That is, the recommendations considered as correct are those in which the user was involved as a neighbour, and her ratings were close (up to a distance of ϵ) to the actual ratings.

A similar trust metric, called **expertise trust**, is presented in (Kwon et al., 2009), where the concept of 'correct recommendation' is also used. In that work Kwon and colleagues introduce a compensation value for situations in which few raters are available. Specifically, the correct recommendation function only outputs a value of 1 when there are enough raters for a particular item (more than 10 in the paper). Otherwise, an attenuation factor is introduced by dividing the number of raters by 10, in the same way as significance weighting is introduced in Pearson's correlation in (Herlocker et al., 2002). More formally, the predictor is defined as:

$$\gamma_2(u, v, i) = \gamma(v) = \frac{1}{\sum_{j \in I_v} \sum_{w \in \mathcal{U}_i} 1} \sum_{j \in \mathcal{I}_v} \sum_{w \in \mathcal{U}_i} \text{Correct}\big(j, v, w; \lambda(j)\big)$$

where $\lambda(j)$ is 1 when item j has more than 10 raters, and $\mathcal{U}_i$ denotes the users who rated item i. In the same paper the authors propose another trust metric called **trustworthiness**, which is equivalent to the absolute value of the similarity between the target user's ratings and the average ratings given by the community (denoted as $\bar{R}$). The authors introduce the significance weighting factor β as in (Herlocker et al., 2002), in a way that $\beta(v)$ is 1 when user v has more than 50 ratings; otherwise, β is computed as the user's ratings divided by 50. Once the β factor is computed, the predictor is defined as follows:

$$\gamma_3(u,v,i) = \gamma(v) = \beta(v) \times \left| \frac{\sum_{j\in\mathcal{I}_v}\big(r(v,j)-\bar{r}(v)\big)\big(\bar{r}(j)-\bar{R}\big)}{\sqrt{\sum_{j\in\mathcal{I}_v}\big(r(v,j)-\bar{r}(v)\big)^2 \sum_{j\in\mathcal{I}_v}\big(\bar{r}(j)-\bar{R}\big)^2}} \right|$$

Hwang and Chen (2007) present a global trust metric, which we call **global trust deviation**, defined as an average of local (user-to-user) trust deviations. This metric makes use of the predicted rating for a user–item pair by using only one user as neighbour:

$$\tilde{r}(u,i) \sim \tilde{r}(u,i;v) = \bar{r}(u) + \big(r(v,i) - \bar{r}(v)\big)$$

where user v is the considered neighbour. The predictor is then computed by averaging the prediction error of co-rated items between each user, and normalising the error according to the rating range R_r (e.g. in a typical 1 to 5 rating scale, $R_r = 4$):

$$\gamma_4(u,v,i) = \gamma(v) = \frac{1}{|N_k(v)|}\sum_{w\in N(v)}\left(\frac{1}{|\mathcal{I}_v \cap \mathcal{I}_w|}\sum_{j\in\mathcal{I}_v\cap\mathcal{I}_w}\left[1 - \frac{|\tilde{r}(v,j;w) - r(v,j)|}{R_r}\right]\right).$$

Finally, a performance predictor inspired by the clarity score defined for query performance (Cronen-Townsend et al., 2002) was proposed in (Bellogín and Castells, 2010), considering its adaptation to predict neighbour performance in collaborative filtering. In the same way query clarity captures the lack of ambiguity in a query, **user clarity** is expected to capture the lack of ambiguity in a user's preferences. Thus, the amount of uncertainty involved in a user's profile is assumed to be a good predictor of her performance; and the larger the following value, the lower the uncertainty and the higher the expected performance:

$$\gamma_5(u,v,i) = \gamma(v) = KLD(v \,\|\mathcal{U} \setminus \{u\}) = \sum_{w\in\mathcal{U}\setminus\{v\}} p(w|v)\, log_2 \frac{p(w|v)}{p(w)}$$

The probabilistic models defined in that work are based on smoothing estimations and conditional probabilities over users and items. Specifically, a uniform distribution is assumed for users and items, whereas the user-user probability is defined by an expansion through items as follows:

$$p(v|u) = \sum_{i \in \mathcal{I}_u} p(v|i) p(i|u).$$

Conditional probabilities are linearly smoothed with the user's probabilities and the maximum likelihood estimators, which finally depend on the rating given by the user towards an item; i.e., $p_{ml}(i|u) \propto r(u,i)$.

It is interesting to note that this predictor (and the probability model in which is grounded) does not correspond with any of the adaptations of the clarity score proposed in Chapter 6, since relations between users are not considered in any of the rating-based probability models presented.

In addition to the integration of the above methods in the role of neighbour effectiveness predictors in our framework, we propose two novel predictors based on well known quantities measured over the probability models of (Bellogín and Castells, 2010): the entropy and the mutual information. Entropy, as an information-theoretic magnitude, measures the uncertainty associated with a probability distribution (Cover and Thomas, 1991). Borrowing the definition of user entropy from Chapter 6, we hypothesise that the uncertainty in the system's knowledge about a user's preferences may be a relevant signal in the effectiveness of a user as a potential neighbour, which could be captured by the entropy of the item distribution as follows:

$$\gamma_7(u,v,i) = \gamma(v) = -H(\mathcal{I}_v) = \sum_{j \in \mathcal{I}_v} p(j|v) \, log_2 \, p(j|v).$$

Note that uncertainty, measured in this way, can be due to the system's knowledge about the user's tastes, or may come from the user herself (e.g. some users may have strong preferences, while others may be more undecided), and both causes may similarly affect the neighbour effectiveness. In either case the predictor can be interpreted as the lack of ambiguity in a user profile.

The second information-theoretic magnitude we propose to use over the probability models presented above is the mutual information. To be precise, the mutual information is a quantity computed between two random variables that measure the mutual dependence of the variables, or, in other terms, the reduction in uncertainty about one variable provided some knowledge about the other (Cover and Thomas, 1991). Here, we propose to adapt this concept, and compute the **mutual information** between the neighbour and the rest of the community in order to assess the uncertainty involved in the neighbour's preferences. For this purpose, instead of computing the mutual information over all the events in the sample space for both variables (users), we fix one of them (for the current neighbour), and move along the other dimension:

$$\gamma_6(u,v,i) = \gamma(v) = MI(v; \mathcal{U} \setminus \{u\}) = \sum_{w \in \mathcal{U} \setminus \{u\}} p(w|v) \, log_2 \frac{p(w|v)}{p(v)p(w)}.$$

User-Item Predictors

User-item predictors consist of performance predictors that depend on a user-item pair. More specifically, they are defined upon the active neighbour and the target item. This type of predictor is more difficult to apply because of its higher vulnerability to data sparsity. In a bi-dimensional user-item input space less observations can be associated to each input data point, whereby the confidence on the predictor outcome is lower, as it can be biased to outliers or unusual users or items.

A local trust metric based on the target user and item is proposed in (O'Donovan and Smyth, 2005). This metric is called **item-level trust**, and aims to discriminate reliable neighbours depending on the current item, since the same user may be more trustworthy for predicting ratings for certain items than for others. The formulation of this predictor can be seen as a particularisation of γ_1, but constraining the recommendation set only to the pairs in which the current item is involved:

$$\gamma_8(u,v,i) = \gamma(v,i) = \frac{|\{(c_k, i_k) \in \text{CorrectSet}(v) : i_k = i\}|}{|\{(c_k, i_k) \in \text{RecSet}(v) : i_k = i\}|}.$$

User-User Predictors

The user-user predictors take as inputs two users: the active user and the current neighbour. User-user predictors based on local trust metrics have been studied further than user-item predictors in the literature, since the former are able to represent how much a user can be trusted by another, and let for different interpretations of the relation between users. These metrics have been often researched in the scope of social networks, and the users' explicit links in this context (Ziegler and Lausen, 2004; Massa and Avesani, 2007a), along with several trust metrics based on ratings, as we shall show below. In this way, although social-based metrics could be smoothly integrated in our framework, here we focus on a complementary view on trust where predictors are defined based on ratings. We leave other type of predictors as future work.

A first simple neighbour reliability criterion one may consider is the amount of common experience with the target user, that is, the amount of information upon which the two users can be compared. If we define "user experience" as the set of items the user has interacted with, we may define a predictor embodying this principle as:

$$\gamma_9(u,v,i) = \gamma(u,v) = |\mathcal{I}_u \cap \mathcal{I}_v|.$$

We shall refer to this predictor as **user overlap**. This predictor will serve as a basis for subsequent predictors, since most of them will depend on the items rated by both users. For instance, it has a clear use in assessing the reliability of the inter-user similarity assessments, which has been applied in the literature under a more practical, ad-hoc manner. Specifically, Herlocker et al. (2002) proposed the introduction of a weight on the similarity function, where the latter is devalued when it has been based on a small number of co-rated items. We may formulate **Herlocker's significance weighting** predictor as follows:

$$\gamma_{10}(u,v,i) = \gamma(u,v) = \frac{|\mathcal{I}_u \cap \mathcal{I}_v|}{n_H} \text{ if } |\mathcal{I}_u \cap \mathcal{I}_v| < n_H; 1 \text{ otherwise,}$$

where n_H is the minimum number of co-rated items that two users should have in common in order to avoid similarity penalisation. A value of $n_H = 50$ was proved empirically to work effectively.

A variation of the previous scheme was proposed in (McLaughlin and Herlocker, 2004), to which we shall refer as **McLaughlin's significance weighting**:

$$\gamma_{11}(u,v,i) = \gamma(u,v) = \frac{\max(|\mathcal{I}_u \cap \mathcal{I}_v|, n_{Mc})}{n_{Mc}}.$$

This predictor is aimed to be equivalent to the Herlocker's significance weighting (γ_{10}) formulation when $n_{Mc} = n_H$. However, we note that γ_{10} and γ_{11} represent different concepts, and are not fully equivalent. For instance, as noted in (Ma et al., 2007), γ_{11} may return values larger than 1 when $|\mathcal{I}_u \cap \mathcal{I}_v| > n_{Mc}$, while γ_{10}, by definition, always returns a value in the $(0,1]$ interval.

Alternatively, the following variant can be drawn from (Ma et al., 2007), which is just a more compact reformulation of γ_{10}:

$$\gamma_{12}(u,v,i) = \gamma(u,v) = \frac{\min(|\mathcal{I}_u \cap \mathcal{I}_v|, n_M)}{n_M}.$$

A more elaborated predictor was proposed in (Weng et al., 2006). The rationale behind such predictor is to consider two situations depending whether or not user u takes into account the recommendation made by neighbour v. In this sense trustworthiness is defined as the reduction in the proportion of incorrect predictions of going from the latter situation to the former. The definition of this predictor, denoted as **user's trustworthiness**, is the following:

$$\gamma_{13}(u,v,i) = \gamma(u,v) = \frac{1}{|R|^2 - \sum_x n(u,v;x,\cdot)^2}\left[|R|\sum_x\sum_y \frac{n(u,v;x,y)^2}{n(u,v;\cdot,y)} - \sum_x n(u,v;x,\cdot)^2\right]$$

In this formulation $|R|$ represents the number of allowed rating values in the system (e.g. in a 1 to 5 rating scale, $|R| = 5$), the function $n(u,v;x,y)$ represents the number of co-rated items on which v's ratings have the value y while u's ratings are x, that is, $n(u,v;x,y) = |\{(u,\cdot,x)\} \cap \{(v,\cdot,y)\}|$ when each rating tuple is represented as (a,b,c), given a user a, an item b, and a rating value c. In the same way, $n(u,v;x,\cdot) = \sum_y n(u,v;x,y)$ represents all the co-rated items between u and v

rated with any rating value by user v, and, analogously, $n(u, v; \cdot, y) = \sum_x n(u, v; x, y)$. In this case, the assumed hypothesis is that trust is one's expectation of other's competence in reducing its uncertainty in predicting new ratings.

Finally, a user-user predictor can be defined based on the global trust deviation predictor defined above (γ_4). In fact, Hwang and Chen (2007) define **trust deviation** by ignoring the average along users as follows:

$$\gamma_{14}(u, v, i) = \gamma(u, v) = \frac{1}{|\mathcal{I}_u \cap \mathcal{I}_v|} \sum_{j \in \mathcal{I}_u \cap \mathcal{I}_v} \left[1 - \frac{|\tilde{r}(u, j; v) - r(u, j)|}{R_r} \right]$$

This predictor identifies effective neighbours mainly based on how many trustworthy (understood as "accurate") recommendations a user has received from another.

8.4 Experimental results

In this section we report experiments in which the proposed neighbour effectiveness prediction framework is tested. First, we check the existing correlations between the user-based predictors defined in Section 8.3.2 and the neighbour performance metrics proposed in Section 8.3.1, as a direct test of their predictive power. For the user-item predictors we cannot analyse their correlation because we have no neighbour performance metric depending on both the target user and an item available.

Moreover, we test the usefulness of the predictors to enhance the final performance of memory-based algorithms, by using the predictors' values in the selection and weighting of neighbours, that is, by taking the predictors as the scoring function in Equation (8.2).

Our experiments were conducted on two versions of the MovieLens dataset, namely the 100K and 1M versions, described in Section 3.4.1 and Appendix A.1. For the user-based collaborative filtering method, we used Pearson's correlation as the similarity measure between users, and a varying neighbourhood size (k), which is a parameter with respect to which the results were examined.

8.4.1 Correlation analysis

We analyse the correlation between neighbour quality metrics and neighbour performance predictors in terms of the Pearson and Spearman's correlation metrics. Correlation provides a measure of the predictive power of the neighbour effectiveness prediction approaches: the higher the (absolute) correlation value, the better the predictor estimates the positive neighbour effect on the recommendation accuracy. The sign of the correlation coefficient represents whether the two involved variables – neighbour quality metric and neighbour performance predictor – are directly or inversely correlated.

	Absolute error deviation μ_1 (-)	Neighbour goodness μ_3 (+)	Sign of error μ_2 (+)
Clarity	-0.21	+0.17	+0.14
Entropy	-0.18	+0.18	+0.12
Expertise	-0.62	+0.03	+0.25
Global Trust Deviation	-0.35	-0.01	+0.08
Mutual Information	-0.20	+0.17	+0.12
Profile Level Trust	+0.62	-0.04*	-0.24
Trustworthiness	-0.21	+0.03	+0.20

Table 8.1. Pearson's correlation between the proposed neighbour quality metrics and neighbour performance predictors in the MovieLens 100K dataset. Next to the metric name, an indication about the sign of the metric – direct(+) or inverse(-) – is included. Not significant values for a p-value of 0.05 are denoted with an asterisk (*).

	Absolute error deviation μ_1 (-)	Neighbour goodness μ_3 (+)	Sign of error μ_2 (+)
Clarity	-0.30	+0.16	+0.21
Entropy	-0.22	+0.17	+0.15
Expertise	-0.65	+0.02	+0.30
Global trust deviation	-0.38	-0.03	+0.11
Mutual Information	-0.25	+0.16	+0.17
Profile Level Trust	+0.65	-0.02	-0.30
Trustworthiness	-0.24	+0.03	+0.25

Table 8.2. Spearman's correlation between quality metrics and performance predictors in the MovieLens 100K dataset.

Table 8.1 and Table 8.2 show the correlation values obtained on the MovieLens 100K dataset for the user-based predictors. We associate a sign to each quality metric indicating whether the metric is direct (denoted as '+') or inverse (denoted with '-'), according to the expected sign of the correlation with the predictor, i.e., a metric is direct if the higher its value, the better the true neighbour performance. We can observe that the Spearman's correlation values are consistent, but slightly higher than Pearson's, thus evidencing a non-linear relationship between the quality metrics and the performance predictors.

The absolute error deviation (μ_1) metric presents higher values when the neighbour's prediction is less accurate, being thus an inverse neighbour metric. The other two metrics, sign of error (μ_2) and neighbour goodness (μ_3), are, by definition, direct neighbour metrics, since the former indicates how many times a recommendation from the neighbour has been made in the right direction, whereas the latter represents the change in error between excluding a particular user in the neighbourhood or including her, and thus, the larger this error, the "better" neighbour this user.

	Absolute error deviation μ_1 (-)	Neighbour goodness μ_3 (+)	Sign of error μ_2 (+)
Clarity	-0.14	+0.40	+0.02
Entropy	-0.07	+0.39	-0.08
Expertise	-0.95	-0.06	+0.70
Global Trust Deviation	-0.55	-0.24	+0.36
Mutual Information	-0.17	+0.30	+0.13
Profile Level Trust	+0.83	+0.04	-0.55
Trustworthiness	-0.27	+0.03	+0.36

Table 8.3. Pearson's correlation between quality metrics and performance predictors in the MovieLens 1M dataset. All the values are significant for a p-value of 0.05.

	Absolute error deviation μ_1 (-)	Neighbour goodness μ_3 (+)	Sign of error μ_2 (+)
Clarity	-0.16	+0.35	+0.04
Entropy	-0.03	+0.37	-0.10
Expertise	-0.94	-0.09	+0.69
Global trust deviation	-0.54	-0.25	+0.39
Mutual information	-0.16	+0.31	+0.04
Profile level trust	+0.94	+0.09	-0.69
Trustworthiness	-0.25	+0.02	+0.37

Table 8.4. Spearman's correlation between quality metrics and predictors in the MovieLens 1M dataset.

We can observe in Table 8.1 that, except for some of the predictors that obtain very low absolute values (< 0.10), the four quality metrics are consistent with each other. This consistency is evidenced by the way the predictors correlate with the different metrics: some of the predictors obtain the correct correlations in every situation, that is, positive correlation with direct metrics and negative correlation with the inverse metric (like the clarity predictor), while other predictors obtain opposite values for all the metrics, that is, positive correlations with the inverse metric and negative correlations with direct metrics (such as the profile level trust predictor).

Also in Table 8.1 and Table 8.2 we see that each metric captures a different notion of neighbour quality because they show different correlation values with respect to the predictors. In this way, although consistent correlation results are obtained for direct and inverse metrics, each of them is actually detecting a different nuance of how a neighbour should behave in order to perform well.

Table 8.3 and Table 8.4 show the correlation values obtained on the Movie-Lens 1M dataset. We can observe that the trend in correlation is very similar to the behavior observed on the 100K dataset, and thus, similar conclusions can be drawn from it. There are, however, some changes in the absolute values of the correlation scores for some combinations of performance predictor and quality metric. For instance, the clarity predictor and the neighbour goodness metric obtain larger values in this dataset, while the correlation between entropy and absolute error deviation is smaller.

It is important to note that the number of points used to compute the correlation values is different in the two datasets; there are less than 1,000 points in MovieLens 100K (with 943 users), and more than 6,000 points in MovieLens 1M dataset. This difference affects the significance of the correlation results, as already described in Section 5.4.2, where we observed how the confidence test for a Pearson's (and Spearman's) correlation depends on the size of the sample, and thus, the significance of a correlation value may change for different sample sizes.

In our experiments, for MovieLens 100K, the correlations are significant for a p-value of 0.05 when $r > 0.05$, and in the 1M dataset when $r > 0.02$. Hence, in Table 8.1, there is only one non-significant correlation value (denoted with an asterisk), whereas in Table 8.3, all the results are statistically significant.

Analysing in more detail the reported results for both datasets, we observe that the profile level trust predictor consistently obtains direct correlation values with inverse metrics, whereas inverse correlation values are obtained with direct metrics. This predictor seems to give higher scores to neighbours with larger deviations in their accuracy error, which would result on bad performance prediction because these values are not in the same direction than the performance metrics. The expertise and global trust deviation predictor obtain strong inverse correlations with the absolute error deviation metric, although their correlations with respect to the neighbour goodness metric are negligible, especially for the first predictor, in both datasets. At the other end of the spectrum, the clarity, entropy, and mutual information predictors obtain strong correlation values with the neighbour goodness, and moderate correlations with the rest of metrics, which make these predictors good candidates for successful neighbour performance predictors. Finally, the trustworthiness predictor obtains a significant amount of correlation with respect to the absolute error deviation and sign of error metrics, although its correlation with respect to the neighbour goodness is very low. This predictor thus seems to be useful on estimating how accurate the neighbour may be in terms of the error in a user basis, but probably not as a global metric.

Table 8.5 shows the correlations obtained for user-user neighbour predictors and the proposed user-neighbour clarity metric. Due to the high dimensionality of the vectors involved in this computation, we have considered only those users that have at least one item in common. Despite this fact, correlations are almost negligible, except for the McLaughlin's significance weighting predictor and the Spearman's coefficient, which evidences a non-linear relation between this predictor and the metric. In the next section we shall show that this function is one of the best performing predictors among the evaluated neighbour scoring functions. This result confirms the usefulness of the proposed neighbour performance metric since it is able to discrimi-

	Movielens 100K		Movielens 1M	
	Pearson	Spearman	Pearson	Spearman
Herlocker	0.02	0.03	0.01	0.02
McLaughlin	0.01	0.12	0.01	0.11
Trust Deviation	0.01	0.01	0.01	0.01
User Overlap	0.02	0.03	0.02	0.02
User's Trustworthiness	-0.02	-0.02	-0.01	-0.01

Table 8.5. Correlation between the user-neighbour goodness and user-user predictors in the two datasets evaluated.

nate which neighbour performance predictors are able to capture interesting properties between the user and her neighbours.

In summary, we have observed that most of the performance predictors agree with respect to the different performance metrics, and in general, the correlations computed between neighbour quality metrics and neighbour performance predictors are statistically significant.

8.4.2 Performance analysis

The results reported in the previous section show that some of the studied predictors have the ability to capture neighbour performance, and because of that we hypothesise that they could be used to improve the accuracy of a recommendation model. This hypothesis, nonetheless, has to be checked since the metric against which we measure the neighbour goodness is not the same as the final recommendation performance metric we aim to optimise. With the experiments we report next we aim to confirm the usefulness of the proposed predictors, the validity of the proposed metrics as useful references to assess the power of the predictive methods, and the usefulness of the overall framework as a unified approach to enhance neighbourhood-based collaborative filtering.

In order to achieve this we test the integration of the neighbour predictors into a neighbour selection and weighting scheme for user-based collaborative filtering, as described in Section 8.2.1. Besides testing the effectiveness of the predictors, this experiment provides for observing to what extent the correlations obtained in the previous section correspond with improvements in the final performance of those predictors.

We provide recommendation accuracy and precision results on the MovieLens 1M dataset. Those obtained on the MovieLens 100K dataset are not reported here since they had similar trends. Figure 8.1 and Figure 8.2 show the Root Mean Square Error (RMSE) of the Resnick's collaborative filtering adaptation proposed in Equation (8.2) when used for different neighbour selection and weighting approaches. The curves at the top of the figures represent the values obtained when neighbour per-

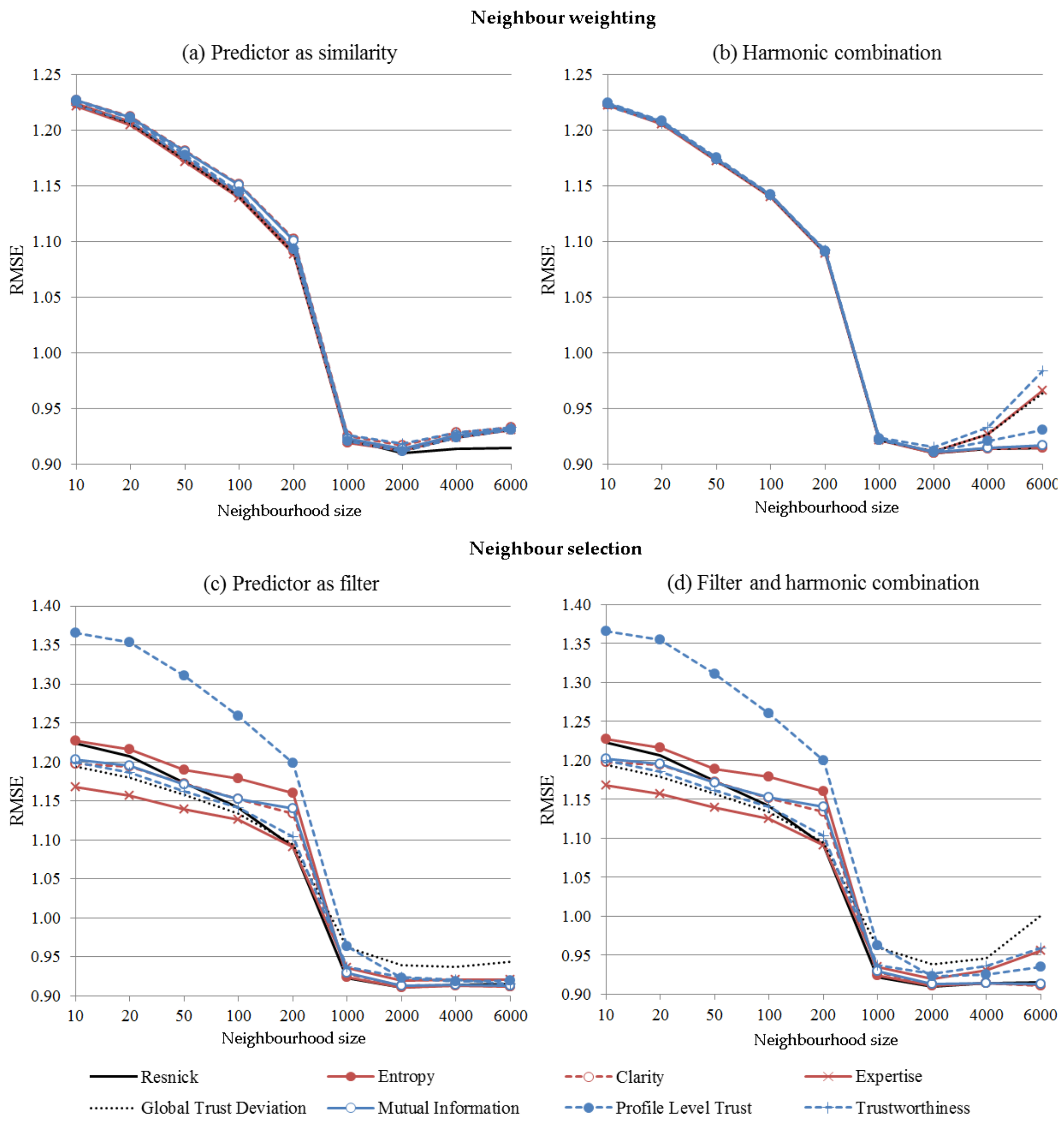

Figure 8.1. Performance comparison for user-based predictors and different neighbourhood sizes.

formance predictors are used for neighbour weighting, that is, when the standard neighbour selection strategy is used ($f^{neigh} = f_0^{neigh}$ in Equation (8.2)). Note that since the lines represent errors, the lower these values, the better the performance. Besides, Figure 8.3 presents the results found with the precision at 10 (P@10) ranking metric of a subset of the proposed methods, where in this case the higher the values, the better the performance.

A different aggregation function is used in each approach, depending on whether the harmonic mean between the predictor score and the similarity value (function $f^{agg} = f_2^{agg}$, on the right), or the projection function ($f^{agg} = f_4^{agg}$, on the left) are used, in the latter case in order to ignore the similarity. The curves at the bottom

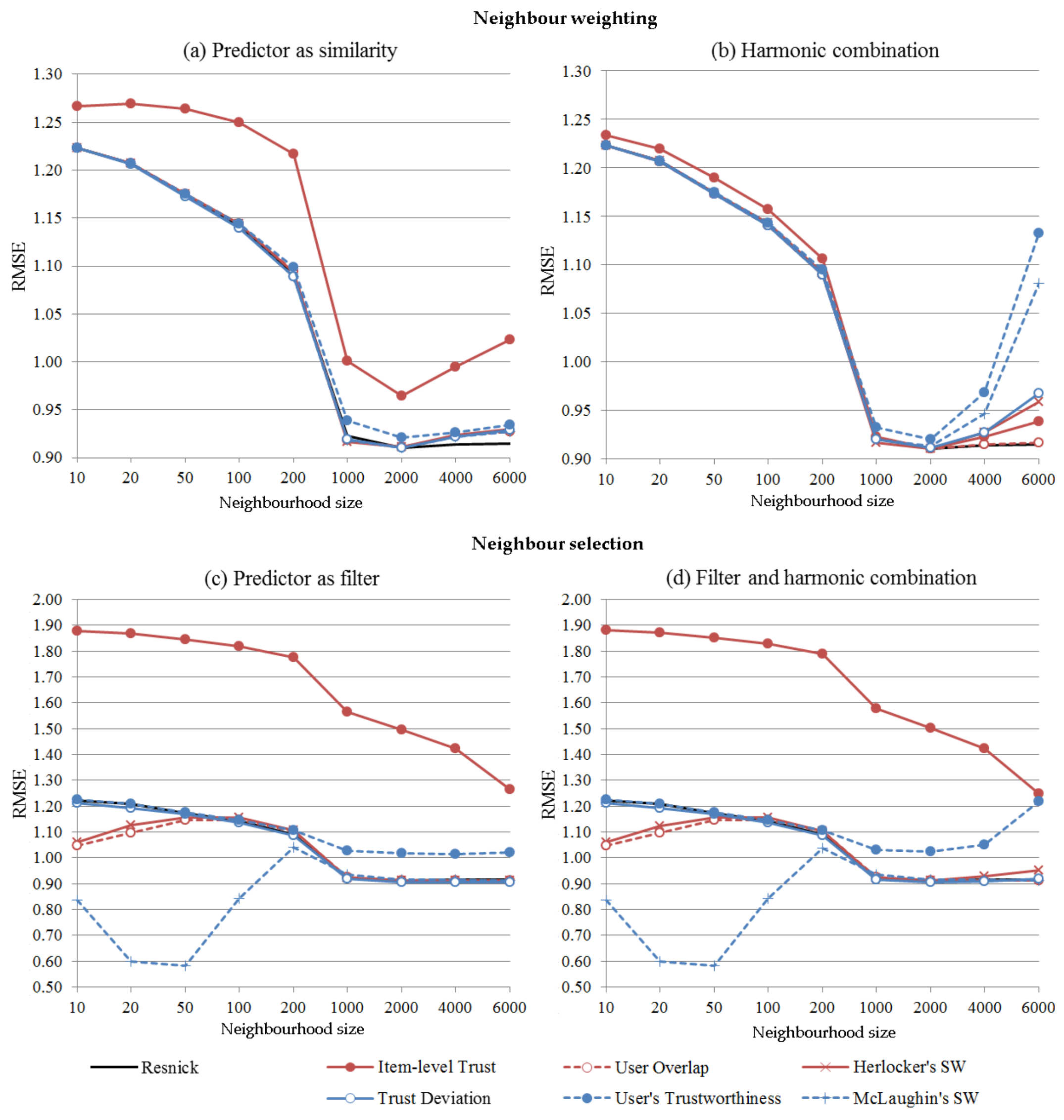

Figure 8.2. Performance comparison using user-item and user-user predictors for different neighbourhood sizes.

of the figures show the neighbour selection approach ($f^{neigh} = f_1^{neigh}$ in Equation (8.2)) along with the same neighbour weighting functions described above (i.e., f_2^{agg} on the right and f_4^{agg} on the left). The rest of the aggregation functions, such as average (f_1^{agg}) and product (f_3^{agg}), were also evaluated for neighbour selection and weighting, but provided results equivalent to those of the harmonic mean. For this reason, they have been omitted in the figures to avoid cluttering them. We believe this equivalence may be due to the normalisation factor included in the collaborative filtering formulation, since it would cancel out the weights obtained by the harmonic, average, and product functions in the same way.

	RMSE
Resnick	1.174
Clarity	1.181
Entropy	1.175
Expertise	1.171
Global Trust Deviation	1.173
Mutual Information	1.180
Profile Level Trust	1.177
Trustworthiness	1.175

	RMSE
Resnick	1.174
Herlocker	1.175
Item-level Trust	1.264
McLaughlin	1.174
Trust Deviation	1.173
User Overlap	1.175
User's Trustworthiness	1.175

Table 8.6. Detail of the accuracy of baseline vs. recommendation using neighbour weighting; here, performance predictors are used as similarity scores (50 neighbours).

	RMSE
Resnick	1.174
Clarity	1.172
Entropy	1.189
Expertise	1.139
Global Trust Deviation	1.158
Mutual Information	1.171
Profile Level Trust	1.310
Trustworthiness	1.162

	RMSE
Resnick	1.174
Herlocker	1.156
Item-level Trust	1.843
McLaughlin	0.581
Trust Deviation	1.168
User Overlap	1.146
User's Trustworthiness	1.174

Table 8.7. Detail of the accuracy of baseline vs recommendation using neighbour selection; here, performance predictors are used for filtering (50 neighbours).

Figure 8.1 shows the accuracy results when only user-based neighbour predictors are evaluated. We observe that, independently from the neighbourhood size, using performance predictors as similarity scores does not lead to large differences with respect to the baseline. These results are compatible with those presented in (Weng et al., 2006), where the improvement in RMSE is not very high (ΔMAE < 0.05 in that work). For the sake of clarity, in Table 8.6 and Table 8.7 we show the error values for a horizontal cut of the left curves; specifically, when the neighbourhood size is 50. We can observe that some predictors do improve Resnick's accuracy. Regarding the use of the harmonic mean as aggregation function (curves on the right), similar results are obtained except for very large neighbourhood sizes, for which some of the performance predictors produce worse results than the baseline, probably due to the amount of noise created by considering too many neighbours.

The curves at the bottom of the figures represent the accuracy results for neighbour selection strategies. In this case some of the predictors lead to worse performance than the baseline, particularly the profile level trust (γ_1). This situation is consistent with the correlations observed in the previous section, since this predictor obtained inverse correlations with the different metrics, i.e., direct correlation values

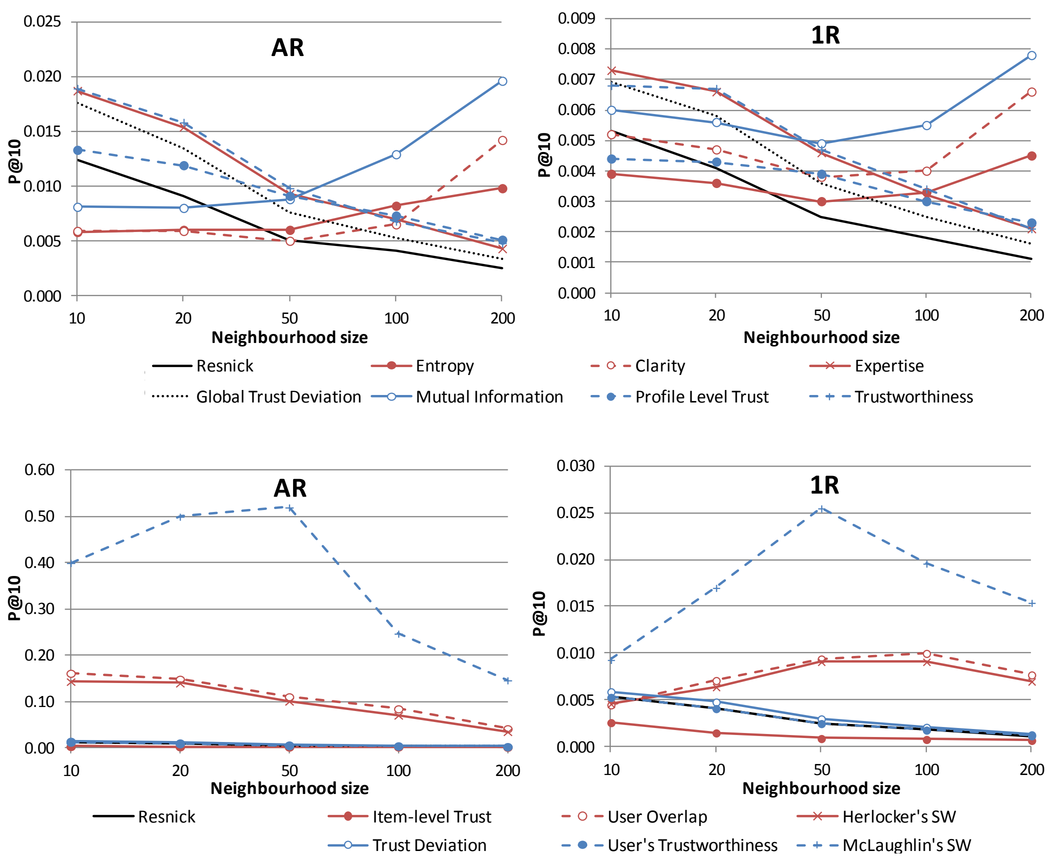

Figure 8.3. Performance comparison using ranking-based metrics for both user and user-user neighbour predictors using the AR and 1R evaluation methodologies.

with inverse metrics, and inverse values with direct metrics. Moreover, as predicted by the correlation analysis, trustworthiness (γ_3), mutual information (γ_6), and clarity (γ_5) result in some of the best performing recommenders (with strong correlations), as shown in the figures and in Table 8.7, along with expertise (γ_2) and global trust deviation (γ_4), which obtained more moderated correlation values.

In Figure 8.2 we can see how user-item and user-user neighbour predictors affect the performance of collaborative filtering recommenders. The curves in the top show that most of the predictors obtain a similar performance to that of the baseline, except for the item-level trust (γ_8), the performance of which is much worse than Resnick's. Table 8.6 shows the specific error values for these recommenders. It is interesting to note that the performance of this predictor is drastically improved when using the harmonic mean as the aggregation function (shown on the right side of the figure). Similarly to user-based neighbour predictors (Figure 8.1), some of the user-item and user-user predictors decrease their accuracy with large neighbourhoods; in this case, user's trustworthiness (γ_{13}) and McLaughlin's significance weighting (γ_{12}) are the more representative examples.

A different conclusion results when neighbour selection is analysed (curves at the bottom). Two of the predictors are characterised by a much better (McLaughlin's significance weighting, γ_{12}) or worse (item-level trust, γ_8) final performance, independently from the weighting aggregation function. Table 8.7 shows the specific error values obtained for each of these predictors. It is interesting how the McLaughlin's predictor, despite its inability to boost good neighbours (see top figures), seems to be very useful for neighbour selection. This effect, nonetheless, is attenuated when the neighbourhood increases, since in that situation, selection methods have to deal with too many users in each neighbourhood. We believe the reason why this predictor is very good for neighbour selection is because it gives higher scores to those neighbours that have more items in common with the target user, and thus the confidence in the computation of the similarity values between the neighbour and the target user is higher. It is worth noting that, to the best of our knowledge, this function has never been used for neighbour selection, since its original motivation was to penalise the similarity value whenever it has been based on a small number of co-rated items. However, by plugging this function into our framework, and measuring its predictive power for user-neighbour performance, a novel application naturally emerges and provides very good results.

Finally, in Figure 8.3 we can observe that a similar trend is found with P@10 for both user-based predictors (top curves), and user-item and user-user predictors (bottom curves). In the figure we only present the results of the neighbour selection and weighting approaches for less than 200 neighbours, since the results of the rest of the approaches and neighbourhoods are very similar. It is worth noting that the two methodologies evaluated – AR and 1R – agree on the order of the best and worst performing dynamic approaches, although as already observed in the previous chapter, the absolute performance values obtained with each methodology may be very different – e.g. the maximum P@10 value with 1R is 0.1, which is reached by several recommendation methods with the AR methodology. More interestingly, these results show consistency between the performance of some dynamic approaches using error- and ranking-based metrics, since the best and worst predictors according to RMSE and P@10 are the same; McLaughlin's significance weighting and item-level trust, respectively. Moreover, the entropy and clarity user-based predictors show worse performance in small neighbourhoods, but outperform the baseline significantly in larger neighbourhoods, something different to what we observed in the previous experiment with error-based metrics.

In summary, we have been able to validate both the proposed user-user neighbour performance metrics, and the different evaluated user-user neighbour performance predictors. We have obtained positive results when this type of predictors has been introduced and compared against the baseline in the different aggregation strategies and configurations, and these results are consistent with the correlations obtained between the predictors and the performance metrics. In particular, McLaughlin's significance weighting obtains an improvement up to 55% in both accuracy (i.e., error decrease) and precision (i.e., precision improvement) when this predictor is used to select the neighbours which will further contribute to the rating prediction. Besides, the (Spearman's) correlation for this predictor is positive and strong, in contrast to the values obtained for the rest of user-user predictors, which did not improve the accuracy of the baseline. In this context, a possible drawback of the conducted analysis is that we have not been able to define neighbour performance metrics based on user-item pairs, and thus the user-item neighbour performance predictors are out of the scope of the developed correlation analysis. Nevertheless, the obtained results showed that the only user-item neighbour performance predictor defined here – the item-level trust – is not able to outperform the baseline recommender. We believe this fact, which is in contradiction with what was reported in (O'Donovan and Smyth, 2005), may be caused by the different variables taking place in our evaluation, such as the dataset (MovieLens 1M instead of MovieLens 100K), the neighbourhood size (not specified in the original paper), and the several aggregation functions and combinations used across our experiments.

8.4.3 Discussion

The reported experiment results provide empiric evidence of the usefulness of the proposed framework, and the specific proposed predictors, as an effective approach to enhance the accuracy of memory-based collaborative filtering. As described in the preceding sections, the methodology comprises two steps, one in which the predictive power of neighbour predictors is assessed, and one in which the predictors are introduced in the collaborative filtering scheme to enhance the effectiveness of the latter. Our experiments confirm a strong correlation for some of the predictors – both user predictors and user-user predictors –, and this has been found to correspond with final accuracy enhancements in the recommendation strategy: the predictors that obtain strong direct correlations with the performance metrics are the best performing dynamic strategies; the profile level trust predictor, which obtains inverse correlation values with respect to the neighbour performance metrics, is the worst performing dynamic strategy.

In light of these results, it could be further investigated whether the actual correlation values between neighbour performance predictors and neighbour performance metrics could be used to infer how each predictor should be incorporated into a memory-based collaborative filtering method as a neighbour scoring function, since there is no obvious link between the ranking of the best performing scoring functions and the strength of their corresponding correlations. As a starting point, only the sign of the correlation could be considered, using either the raw neighbour predictor score (for positive correlations) or its inverse (for negative values). Then, this rationale could be further elaborated and evaluated in order to check whether the performance improvements are consistent.

Research on finding functions with strong correlation power with respect to neighbour performance metrics could be an interesting area by itself, since it could have different final applications. We have experimented here with variations in neighbour selection and weighting for user-based collaborative filtering, but those predictors (functions) could also be used, for instance, for active learning (Elahi, 2011), or for providing more meaningful explanations (Marx et al., 2010), depending or based on the predicted performance of a particular user's neighbours.

8.5 Conclusions

We have shown in this chapter that performance prediction does not only serve to aggregate entire recommender systems, but also to aggregate subcomponents of recommender algorithms – in this case, neighbour related terms in collaborative filtering. We propose a theoretical framework for neighbour selection and weighting in user-based recommender systems, which is based on a performance prediction approach drawn from the query performance methodology of the Information Retrieval field. By viewing the neighbourhood-based collaborative filtering rating prediction task as a case of dynamic output aggregation, our approach places user-based collaborative filtering in a more general frame, linking to the principles underlying the formation of ensemble recommenders, and rank aggregation in Information Retrieval. By doing so, it is possible to draw concepts and techniques from these areas, and vice versa. Our study thus provides a comparison of different state-of-the-art rating-based trust metrics and other neighbour scoring techniques, interpreted as neighbour performance predictors, and evaluated under this new angle. The framework lets an objective analysis of the predictive power of several neighbour scoring functions, integrating different notions of neighbour performance into a unified view. Thus, the proposed methodology discriminates which neighbour scoring functions are more effective in predicting the goodness of a neighbour, and thus identifies which weighting functions are more effective in a user-based collaborative filtering algorithm.

Drawing from different state-of-the-art neighbour scoring functions – cast as user, user-user, and user-item neighbour performance predictors –, we have reported several experiments in order to, first, check the predictive power of these functions, and second, validate them by comparing the final performance of neighbour-scoring powered memory-based strategies with that of the standard collaborative filtering algorithm. We also evaluate different ways to introduce these functions in the rating prediction formulation, namely for neighbour weighting, neighbour selection, and combinations thereof. In this context, methods where neighbour scoring functions were integrated outperform the baseline for different values of neighbourhood size and predictor type.

We have also proposed several neighbour performance metrics that capture different notions of neighbour quality. The evaluated performance predictors show consistent correlations with respect to these metrics, and some of them present particularly strong correlations. Interestingly, a correspondence is confirmed between the correlation analysis and the final performance results, in the sense that the correlation values obtained between neighbour performance predictors and neighbour performance metrics anticipate which predictors will perform better when introduced in a memory-based collaborative filtering algorithm.

This research opens up the possibility to several research lines for the integration of other types of predictors and trust metrics into our framework. For instance, performance predictors defined upon social data, such as those defined in Chapter 6 based on user's trust network, could be smoothly integrated into our framework and analysed in the future. Furthermore, alternative neighbour performance metrics may be defined to check the predictive power of user-user and user-item predictors. These metrics may help better understand which characteristics of the neighbour performance such predictors are capturing, although based on a smaller amount of information since in rating-based systems users only rate items once. In particular, our framework would allow for different interpretations of the user's performance, by modelling different neighbour performance metrics, which may be oriented to accuracy (using error metrics as in this chapter), ranking precision, or even alternative metrics such as diversity, coverage and serendipity (Shani and Gunawardana, 2011). Additionally, other predictors based on item information could be defined similar to those proposed in (Weng et al., 2006; Ma et al., 2007), and easily incorporated into our framework using item-based algorithms instead of user-based.