Chapter 6

Performance prediction in recommender systems

In this chapter, we state and address the recommendation performance prediction problem, proposing and evaluating different prediction schemes. After laying out a formal frame for the problem, we start by researching the adaptation of principles and prediction techniques that have been proposed and developed in ad-hoc Information Retrieval. More specifically, we draw from the notion of query clarity as a basis for finding suitable performance predictors that provide a well grounded theoretical formalisation. In analogy to query clarity, we hypothesise that the amount of uncertainty involved in user and item data (reflecting ambiguity in user's tastes and item popularity patterns) may also correlate with the accuracy of the system's recommendations. This uncertainty can be captured as the clarity of users and/or the clarity of items by an adaptation of the query clarity formulation. This adaptation, however, is not straightforward, as we shall describe. Besides the approaches elaborating on the notion of clarity, we propose new predictors based on theories and models from Information Theory and Social Graph Theory.

In Section 6.1 we formulate the research problem we aim to address. Next, in Sections 6.2, 6.3, and 6.4 we propose several performance predictors for recommender systems, some of them based on the clarity score, information theoretical related concepts – such as entropy –, and graph-based metrics. The proposed predictors are defined upon three different spaces, namely ratings, logs, and social networks. Moreover, we also provide specific correlations of the described predictors in Section 6.5 in order to show their predictive power under different conditions along with a discussion of the results. Finally, in Section 6.6 we provide some conclusions.

6.1 Research problem

Performance prediction finds a special motivation in recommender systems. Contrary to query-based information retrieval, as far as the initiative relies on the system, a performance prediction approach may provide a basis to decide producing recommendations or holding them back, depending on the expected level of performance on a per case basis, delivering only the sufficiently reliable cases. On the other hand, recommenders based on a single algorithm are not competitive in practice, and real applications heavily rely on hybridisations and ensembles of algorithms.

The capability to foresee which algorithm can perform better in different circumstances can therefore be envisioned as a good approach to enhance the performance of the combination of algorithms by dynamically adjusting the reliance on each subsystem. Furthermore, it is well-known in the recommender systems field that the performance of individual recommendation methods is highly sensitive to different conditions, such as data sparsity, quality and reliability, which are subject to an ample dynamic variability in real settings. Hence, being able to estimate in advance which recommenders are likely to provide the best output in a particular situation opens up an important window for performance enhancement. Alternatively, estimating which users of a system are likely to receive worse recommendations allows for modifications in the recommendation algorithms to predict this situation, and react in advance.

The problem of performance prediction has been however barely addressed in the Recommender Systems field. The issue has been nonetheless mentioned in the literature – evidencing the relevance of the problem – and is in some way often implicitly addressed by means of ad hoc heuristic tweaks such as significance weighting in nearest neighbour recommenders (Herlocker et al., 1999) and confidence scores (Wang et al., 2008a), along with additional computations (mainly normalisations) which are introduced into the recommendation methods aimed to better estimate the predicted ratings.

In the recommendation context, the problem of performance prediction can be stated as follows. We define a performance predictor as a function that takes a certain input, and returns a real value that correlates with some utility dimension of a recommender system. This is an instantiation of the problem presented in Section 5.1 but in the recommendation setting. For such purpose, we first specify more precisely what the input space of predictors consists of, and how the predictor input and output relate to the data involved in recommendation. Thus, a utility predictor handles the following information:

Input variables

- The specific configuration of the recommender system. For instance, for a nearest neighbour recommender input parameters could be the neighbour map (that assigns a set of neighbours to each user) and a user similarity metric.
- Any input of the recommender, such as the active user and the active item.
- Background/context information: any known user, item, and user-item interaction data, such as user ratings, user features, item features, social network information, data timestamps, etc. We have to note that, even though the predictor will generally use this type information, we consider it as implicit input and do not include it explicitly in our notation to avoid making it needlessly cumbersome.

Output variable

- A value in $\mathbb{R}$.

A predictor is thus a function $\gamma: R \times \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}$ (R being the set of all recommenders) that estimates the performance of the system, possibly using additional information available in the background. A predictor can be independent from some of these inputs, which would be then omitted in the previous notation. For instance, in this chapter we shall present predictors of the form $\gamma: \mathcal{U} \rightarrow \mathbb{R}$ and $\gamma: \mathcal{I} \rightarrow \mathbb{R}$. Additionally, a predictor may assume a specific parameterised recommender algorithm family (e.g. nearest neighbour collaborative filtering), and needs some element of its configuration as input. It may also happen that a predictor does not make any assumption on the recommender – it does not depend on it – but still the predictor works well only for certain types of recommenders. It would be syntactically possible and correct to apply the predictor with other recommenders, although it may work badly. In general, what it means for a predictor to work "well" may depend on the application, but we generally assume it can be evaluated in terms of its correlation to some utility dimension of recommendations, such as an accuracy metric (RMSE, precision, nDCG) or alternative metrics such as novelty, diversity, etc.

If a recommender system can be decomposed into its internal configuration, then a predictor can directly take as input the components of the recommender configuration. For instance, neighbourhood-based collaborative filtering recommenders can be represented in $R \equiv \mathcal{E} \times \mathcal{N} \times \mathcal{S}$, where $\mathcal{E}: \mathbb{R}^k \times \mathbb{R}^k \rightarrow \mathbb{R}^+$ is a preference estimation function (based on k similarity values between the target user and her neighbours, and k neighbours' ratings on the target item), $\mathcal{N}: \mathcal{U} \rightarrow P(\mathcal{U})$ is a neighbourhood assignment map, and $\mathcal{S}$ is a similarity metric. Upon such a model, we would have $\gamma: \mathcal{E} \times \mathcal{N} \times \mathcal{S} \times \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}$.

We may also constrain some inputs to a relevant condition they should meet. For instance, we could limit ourselves to a neighbourhood map that considers a user v as a candidate neighbour. In that case, this map can be essentially represented by v, and then we would have $\gamma: \mathcal{E} \times \mathcal{U} \times \mathcal{S} \times \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}$ (note that the first $\mathcal{U}$ in the Cartesian product stands for neighbour users, and the second $\mathcal{U}$ for target users).

It is important to note that when the predictor takes as input some of the inputs of the recommender, namely the active user and/or the active item, then the predictor's correlation with the recommender's utility must be measured on a per-input basis. For instance, if the predictor just takes users as input arguments, it should correlate with the average utility per user.

Moreover, predictors can also be used to enhance hybrid recommenders by favouring strategies that are predicted to produce better results. This can be done by relating activation switches in the recommenders to predictor values, so that one recommender or the others are activated or favoured depending on the predictor's estimation.

The way in which these activation switches are related to predictors is typically application-dependent. For instance, in ensemble recommenders consisting of a unique (Boolean) selection among a set of recommenders, the selection/discarding of recommenders can be a binary function of a predictor for each recommender. If the ensemble consists of a linear combination of recommenders, the weights in the combination can also be a function of the predictors. In neighbourhood-based collaborative filtering, activation switches can be the weights of neighbours in the prediction of user ratings. Indeed, relating predictor values to activation switches is a non-trivial problem and generally requires some research on itself.

Based on all the above mentioned issues, the general research problem we address consists of a) finding effective predictors of recommendation utility, and b) identifying and testing useful applications for the found predictors. In the reminder of this chapter we propose different predictors of recommendation utility using different types of input, namely ratings, logs, and social information. In Chapters 7 and 8 we shall exploit and evaluate such predictors in two applications: dynamic hybrid recommendation, and dynamic neighbour weighting in collaborative filtering.

6.2 Clarity for preference data: adaptations of query clarity

In this thesis, we propose different adaptations for the concept of query clarity to recommender systems. First, we deal with the definition of user clarity when rating-based preference data is available, where alternative ground models are proposed, depending on which random variables want to be considered in the computation of the user clarity. Then, we define the concept of user clarity for log-based preference data. Additionally, for ratings we also define the concept of item clarity.

Now we propose a fairly general adaptation of query clarity, which may be instantiated into different schemes, depending on the input spaces considered. At an abstract level, we consider an adaptation that equates users in the recommendation domain to queries in the search domain, as the corresponding available representations of user needs in the respective domains. This adaptation results in the following formulation for **user clarity**:

$$\text{clarity}(u) = \sum_{x \in \mathcal{X}} p(x|u) \log_2 \frac{p(x|u)}{p(x)} \tag{6.1}$$

As we can observe, the clarity formulation strongly depends on a "vocabulary" space $\mathcal{X}$, which further constrains the user-conditioned model (or user model for short) $p(x|u)$, and the background probability $p(x)$. In ad-hoc information retrieval, this space is typically the space of words, and the query language model is a probability distribution over words (Cronen-Townsend et al., 2002). In recommender systems, however, we may have different interpretations, and thus, different formulations for such a probabilistic framework, as we shall show. In all cases, we will need to model and estimate two probability distributions: first, the probability that some event (depending on the current probability space $\mathcal{X}$) is generated by the user language model (user model); and second, the prior probability of generating that event (background model).

Under this formulation, user clarity is in fact the difference (Kullback-Leibler divergence) between a user model and a background model. The use of user and background distributions as a basis to predict recommendation performance lies on the hypothesis that a user probability model being close to the background (or collection) model is a sign of ambiguity or vagueness in the evidence of user needs, since the generative probabilities for a particular user are difficult to single out from the model of the collection as a whole. In Information Retrieval, this fact is interpreted as a query for which the relevant documents are a mix of articles about different topics (Cronen-Townsend et al., 2002).

As an additional step, we generalise the adaptation stated in Equation (6.1) to allow for different reference probability models parameterised by a generic variable θ.

$$\text{clarity}(u) = \mathbb{E}_\theta \left[\sum_{x \in \mathcal{X}} p(x|u,\theta) \log_2 \frac{p(x|u,\theta)}{p(x|\theta)} \right] \tag{6.2}$$

This generalisation will allow for the development of further varieties of the clarity scheme, and simplifies to Equation (6.1) whenever we implicitly consider a fixed θ, as we shall see next. Equivalently, the variable θ may be integrated in both user and background models by exploiting a multidimensional vocabulary space:

$$\text{clarity}(u) = \sum_{x \in \mathcal{X}, \theta \in \Theta} p(x, \theta|u) \log_2 \frac{p(x, \theta|u)}{p(x, \theta)} \tag{6.2b}$$

It is easy to see that Equations (6.2) and (6.2b) are fully equivalent, and thus allow two interpretations for the same magnitude.

As stated in (Cronen-Townsend et al., 2002), language models capture statistical aspects of the generation of language. Therefore, if we use different vocabularies, we may capture different aspects of the user. The probabilistic relations between the variables involved in Equation (6.2) also depend on the nature of the data, and the different possible generative models induced by the recorded observations of user-item interactions (the input to a recommender system). In this thesis we consider two types of interaction data records: users-rating-items (where the atomic event is a user rating an item with a value), and users "consuming" items (a user accesses an item at some time instant). The first type fits a dataset such as MovieLens and CAMRa, and the second fits well Last.fm data – the datasets on which we shall test the methods to be developed here. Across these two types, in our research we explore mainly three vocabulary spaces for $\mathcal{X}$: ratings, items, and time. Each of the vocabulary spaces induces its own user-specific interpretation, as we shall see. As for the optional contextual parameter θ, we shall consider here only the space of items ranging over the set of items – thus fully leveraging the triadic nature of the user-item-rating and user-item-time spaces. The scheme is however open to the exploration of further possibilities, as is the vocabulary space itself, beyond the options researched here.

In the following sections we thus explore several alternatives for rating-based and log-based data spaces (and their induced generative models).

6.2.1 Rating-based clarity

As just mentioned, in the rating space, we consider a set of user-item-rating tuples, where each user-item pair appears in a unique tuple (i.e., users only rate items once). We consider two possible vocabulary spaces: items and ratings, and two context alternatives: items (which make only sense in the rating vocabulary) and none. The resulting clarity schemes are summarised in Table 6.1, and have each their own interpretation.

The rating-based clarity model captures how differently a user uses rating values (regardless of the items the values are assigned to) with respect to the rest of users in the community. The item-based clarity takes into account which items have been rated by a user, and therefore, whether she rates (regardless of the rating value) the most rated items in the system or not. Finally, the item-and-rating-based clarity computes how likely a user would rate each item with some particular rating value, and compares that likelihood with the probability that the item is rated with some particular rating value. In this sense, the item-based user model makes the assumption that some items are more likely to be generated for some users than for others de-

User clarity	Vocabulary $\mathcal{X}$ / Context θ	User model	Background model	Formulation
Rating-based	Ratings / None	$p(r\|u)$	$p_c(r)$	$\sum_r p(r\|u) \log_2 \frac{p(r\|u)}{p(r)}$
Item-based	Items / None	$p(i\|u)$	$p_c(i)$	$\sum_i p(i\|u) \log_2 \frac{p(i\|u)}{p(i)}$
Item-and-rating-based	Ratings / Items	$p(r\|u,i)$	$p_{ml}(r\|i)$	$\sum_{r,i} p(i) p(r\|u,i) \log_2 \frac{p(r\|u,i)}{p(r\|i)}$

Table 6.1. Three possible user clarity formulations, depending on the interpretation of the vocabulary and context spaces.

pending on their previous preferences. The rating-based model, on the other hand, captures the likelihood of a particular rating value being assigned by a user, which is an event not as sparse as the previous one, with a larger number of observations. Finally, the item-and-rating-based model is a combination of the two previous models into a unified model incorporating items and ratings. As we mentioned before, this could be made more explicit by considering the user model $p(r,i|u)$ in the Equation (6.2b), which would be equivalent to this model under some indepence assumptions, i.e., when $p(r,i|u) = p(r|u,i)p(i)$.

Ground models for user clarity

We ground the different clarity measures defined in the previous section upon a rating-oriented probabilistic model very similar to the approaches taken in (Hofmann, 2004) and (Wang et al., 2008a). The sample space for the model is the set $\mathcal{U} \times \mathcal{I} \times \mathcal{R}$, where $\mathcal{U}$ stands for the set of all users, $\mathcal{I}$ is the set of all items, and $\mathcal{R}$ is the set of all possible rating values. Hence, an observation in this sample space consists of a user assigning a rating to an item. We consider three natural random variables in this space: the user, the item, and the rating value, involved in a rating assignment by a user to an item. This gives meaning to the distributions expressed in the different versions of clarity as defined in the previous section. For instance, $p(r|i)$ represents the probability that a specific item i is rated with a value r – by a random user –, $p(i)$ is the probability that an item is rated – with any value by any user –, and so on.

The probability distributions upon which the proposed clarity models are defined can use different estimation approaches, depending on the independence assumptions one would consider, and the amount of involved information. Background models are estimated using relative frequency estimators, that is:

$$p_c(r) = \frac{|\{(u,i) \in \mathcal{U} \times \mathcal{I} | r(u,i) = r\}|}{|\{(u,i) \in \mathcal{U} \times \mathcal{I} | r(u,i) \neq \emptyset\}|} \tag{6.3}$$

$$p_c(i) = \frac{|\{u \in \mathcal{U} | r(u,i) \neq \emptyset\}|}{|\{(u,j) \in \mathcal{U} \times \mathcal{I} | r(u,j) \neq \emptyset\}|}$$

$$p_{ml}(r|i) = \frac{|\{u \in \mathcal{U} | r(u,i) = r\}|}{|\{u \in \mathcal{U} | r(u,i) \neq \emptyset\}|}$$

$$p_{ml}(r|u) = \frac{|\{i \in \mathcal{I} | r(u,i) = r\}|}{|\{i \in \mathcal{I} | r(u,i) \neq \emptyset\}|}$$

These are maximum likelihood estimations in agreement with the meaning of the random variables as defined above. Starting from these estimations, user models can be reduced to the above terms by means of different probabilistic expansions and Bayesian reformulations, which we define next for the three models introduced in the previous section.

Item based model. The $p(i|u)$ model can be simply expanded through marginalisation over ratings, but under two different assumptions: the item generated by the model only depends on the rating value, independently from the user or, on the contrary, depends on both the user and the rating. These alternatives lead to the following developments, respectively:

$$p_R(i|u) = \sum_{r \in \mathcal{R}} p_{ml}(i|r) p_{ml}(r|u) \tag{6.4}$$

$$p_{UR}(i|u) = \sum_{r \in \mathcal{R}} p(i|u,r) p_{ml}(r|u) \tag{6.5}$$

Rating based model. This model assumes that the rating value generated by the probability model depends on both the user and the item at hand. For this model, we sum over all possible items in the following way:

$$p(r|u) = \sum_{r(u,i)=r} p(r|u,i) p(i|u) \tag{6.6}$$

where the $p(i|u)$ term can be developed as in the item-based model above. The term $p(r|u,i)$ requires further development, which we define in the next model.

Item-and-rating based model. Three different models can be derived depending on how the Bayes' rule is applied. In these models, item probability is assumed to be uniform and thus it can be ignored in the computation of the expectation in Equation (6.2). In the same way as proposed in (Wang et al., 2008a), three relevance models can be defined, namely a user-based, an item-based, and a unified relevance model:

$$p_U(r|u,i) = \frac{p(u|r,i) p_{ml}(r|i)}{\sum_{r \in \mathcal{R}} p(u|r,i) p_{ml}(r|i)} \tag{6.7}$$

$$p_I(r|u,i) = \frac{p(i|u,r)p_{ml}(r|u)}{\sum_{r\in\mathcal{R}} p(i|u,r)p_{ml}(r|u)} \tag{6.8}$$

$$p_{UI}(r|u,i) = \frac{p(u,i|r)p_c(r)}{\sum_{r\in\mathcal{R}} p(u,i|r)p_c(r)} \tag{6.9}$$

The first derivation induces a user-based relevance model because it measures by $p(u|r,i)$ how probable it is that a user rates item i with a value r. The item-based relevance model is factorised proportional to an item-based probability, i.e., $p_I(r|u,i) \propto p(i|u,r)$. Finally, in the unified relevance model, we have $p_{UI}(r|u,i) \propto p(u,i|r)$. These estimations correspond respectively with the Equations 20a, 20b, and 21 from (Wang et al., 2008a); to make the thesis self-contained and facilitate the comparison between the different probability models, we present now these equations from (Wang et al., 2008a):

$$p(u|r,i) = \frac{1}{|S(r,i)|} \sum_{v\in S(r,i)} \frac{1}{h_u^{|\mathcal{I}|}} K\left(\frac{\boldsymbol{u}-\boldsymbol{v}}{h_u}\right) \tag{6.10}$$

$$p(i|u,r) = \frac{1}{|S(r,u)|} \sum_{j\in S(r,u)} \frac{1}{h_i^{|\mathcal{U}|}} K\left(\frac{\boldsymbol{i}-\boldsymbol{j}}{h_i}\right) \tag{6.11}$$

$$p(u,i|r) = \frac{1}{|S(r)|} \sum_{(v,j)\in S(r)} \frac{1}{h_u^{|\mathcal{I}|}} K\left(\frac{\boldsymbol{u}-\boldsymbol{v}}{h_u}\right) \frac{1}{h_i^{|\mathcal{U}|}} K\left(\frac{\boldsymbol{i}-\boldsymbol{j}}{h_i}\right) \tag{6.12}$$

where $K(\cdot)$ is a Parzen Kernel function (Duda et al., 2001). In this formulation, $\boldsymbol{u}$ denotes the user u represented as a vector by her ratings in the space of items. Unrated items can be filled with the average rating value or with other constant value, such as 0 or the average rating in the community. Respectively, $\boldsymbol{i}$ represents the item i in the user space. h_u and h_i are the bandwidth window parameter for the user and item vector, respectively; $S(\cdot)$ denotes the set of observed samples where event $(\cdot)$ has happened. For example, $S(r,i)$ denotes the set of observed samples with event $(R = r, I = i)$. More specifically:

$$S(r,i) = \{u \in \mathcal{U} | r(u,i) = r\} \tag{6.13}$$

$$S(r,u) = \{i \in \mathcal{I} | r(u,i) = r\} \tag{6.14}$$

$$S(r) = \{(u,i) \in \mathcal{U} \times \mathcal{I} | r(u,i) = r\} \tag{6.15}$$

In the experiments, we used a Gaussian Kernel function, i.e., $K(x) = e^{-x^2/2}/\sqrt{2\pi}$, and $h_i = h_u = 0.9$ as suggested in (Wang et al., 2008a).

User clarity name	User dependent model	Background model
RatUser	$p_U(r\|u,i); p_{UR}(i\|u)$	$p_c(r)$
RatItem	$p_I(r\|u,i); p_{UR}(i\|u)$	$p_c(r)$
ItemSimple	$p_R(i\|u)$	$p_c(i)$
ItemUser	$p_{UR}(i\|u)$	$p_c(i)$
IRUser	$p_U(r\|u,i)$	$p_{ml}(r\|i)$
IRItem	$p_I(r\|u,i)$	$p_{ml}(r\|i)$
IRUserItem	$p_{UI}(r\|u,i)$	$p_{ml}(r\|i)$

Table 6.2. Different user clarity models implemented.

Finally, different combinations of distribution formulations and estimations result in a fair array of alternatives. Among them, we focus on a subset that is shown in Table 6.2, which provide the most interesting combinations, in terms of experimental efficiency, of user and background distributions for each clarity model. These alternatives are further analysed in detail below (with examples) and in Section 6.5.1 where correlations obtained by each model are presented.

Qualitative observation

In order to illustrate the proposed prediction framework and give an intuitive idea of what user characteristics the predictors are capturing, we show the relevant aspects of specific users that result in clearly different predictor values, in a similar way to the examples provided in (Cronen-Townsend et al., 2002) for query clarity. We compare three user clarity models out of the seven models presented in Table 6.2: one for each formulation included in Table 6.1. In order to avoid distracting biases on the clarity scores that a too different number of ratings between users might cause, we have selected pairs of users with a similar number of ratings. This effect would be equivalent to that found in Information Retrieval between the query length and its clarity for some datasets (Hauff, 2010).

Table 6.3 shows the details of two sample users on which we will illustrate the effect of the predictors. As we may see in the table, u_2 has a higher clarity value than u_1 for the three models analysed. That is, according to our theory, u_2 is less "ambiguous" than u_1. Figure 6.1 shows the clarity contribution in a term-by-term basis for one of the item-and-rating-based clarity models – where, in this case, terms are equivalent to a pair (rating, item) – as analysed in (Cronen-Townsend et al., 2002). In the figure, we plot $p(r|u,i)\log_2(p(r|u,i)/p(r|i))$ for the different terms in the collection, sorted in descending order of contribution to the user model, i.e.,

User	Number of ratings	ItemUser clarity	RatItem clarity	IRUserItem clarity
u_1	51	216.015	28.605	6.853
u_2	52	243.325	43.629	13.551

Table 6.3. Two example users, showing the number of ratings they have entered, and their performance prediction values for three user clarity models.

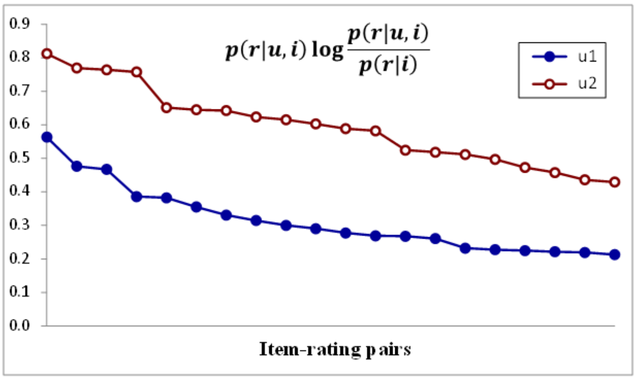


Figure 6.1. Term contributions for each user, ordered by their corresponding contribution to the user language model. IRUserItem clarity model.

$p(r|u,i)$, for each user. For the sake of clarity, only the top 20 contributions are plotted. We may see how the user with the smaller clarity value receives lower contribution values than the other user. This observation is somewhat straightforward since the clarity value, as presented in Equation (6.1), is simply the sum of all these contributions, over the set of terms conforming the vocabulary. In fact, the figures are analogous for the rest of the models, since one user always obtains higher clarity value than the other.

Let us now analyse more detailed aspects in the statistical behaviour of the users that explain their difference in clarity. The IRUserItem clarity model captures how differently a user rates an item with respect to the community. Take for instance the top item-rating pairs for users 1 and 2 in the above graphic. The top pair for u_2 is (4, “McHale’s Navy”). This means that the probability of u_2 rating this movie with 4 is much higher than the background probability (considering the whole user community) of this rating for this movie. Indeed, we may see that u_2 rated this movie with a 3, whereas the community mode rating is 1 – quite farther away from 4. This is the trend in a clear user. On the other extreme of the displayed values, the bottom term in the figure for u_1 is (2, “Donnie Brasco”), which is rated by this user with a 5, and the community mode rating for this item is 4, thus showing a very similar trend between both. This is the characteristic trend of a non-clear user.

Furthermore, if we compare the background model with the user model, we obtain more insights about how our models are discriminating distinctive from mainstream behaviour. This is depicted in Figure 6.2. In this situation, we select those terms which maximise the difference between the user and background models. Then, for this subset of the terms, we sort the vocabulary with respect to its collection probability, and then we plot the user probability model for each of the terms in the vocabulary.

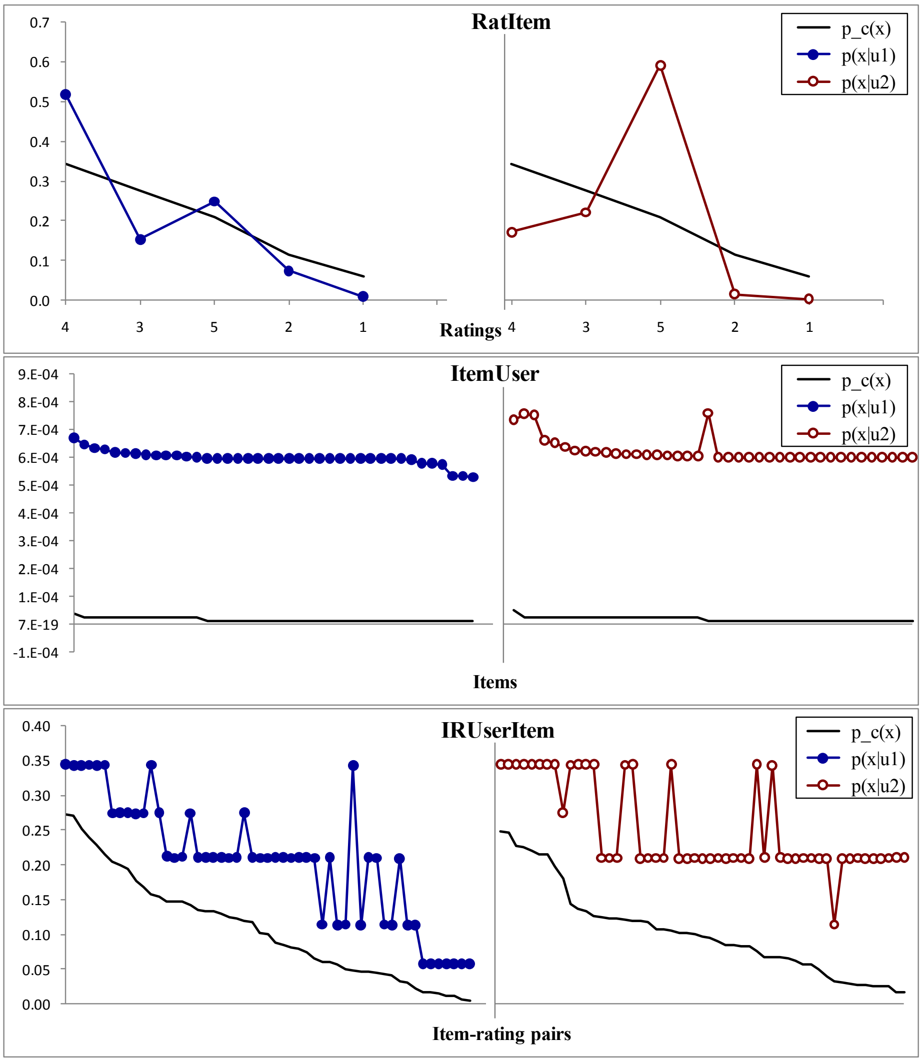


Figure 6.2. User language model sorted by collection probability.

These figures show how the most ambiguous user obtains a similar distribution to that of the background model, while the distribution of the less ambiguous user is more different. In the rating-based model this effect is clear, since the likelihood of not so popular rating values (i.e., a '5') is larger for u_2 than for u_1, and at the same time, the most popular rating value (a '4') is much more likely for u_1. The figure about the ItemUser model is less clear in this aspect, although two big spikes are observed for u_1 with respect to the collection distribution, which correspond with two unpopular movies: 'Waiting for Guffman' and 'Cry, the beloved country', both with a very low collection probability. Finally, the figure about the IRUserItem model successfully shows how u_2 has more spikes than u_1, indicating a clear divergence from the background model; in fact, u_1's distribution partially mimics that of the collection. In summary, the different models proposed are able to successfully sepa-

Item clarity	Vocabulary $\mathcal{X}$ / Context θ	Item model	Background model	Formulation
Rating-based	Ratings / None	$p(r\|i)$	$p_c(r)$	$\sum_r p(r\|i) \log_2 \frac{p(r\|i)}{p(r)}$
User-based	Users / None	$p(u\|i)$	$p_c(u)$	$\sum_u p(u\|i) \log_2 \frac{p(u\|i)}{p(u)}$
User-and-rating-based	Ratings / Users	$p(r\|u,i)$	$p_{ml}(r\|u)$	$\sum_{r,u} p(u) p(r\|i,u) \log_2 \frac{p(r\|i,u)}{p(r\|u)}$

Table 6.4. Three possible item clarity formulations, depending on the interpretation of the vocabulary and context spaces.

rate information concerning the user and that from the collection, in order to infer whether a user is different or similar from the collection as a whole.

Item clarity

Alternatively to user-based predictors, we can also consider item-based predictors, where the performance prediction is made on an item-basis. Item predictors can be defined analogously as those defined previously for users, the equation for **item clarity** being as follows:

$$\text{clarity}(i) = \mathbb{E}_\theta \left[\sum_{x \in \mathcal{X}} p(x|i,\theta) \log_2 \frac{p(x|i,\theta)}{p(x|\theta)} \right] \tag{6.16}$$

The formulation of the item predictors we propose is basically equivalent to the user-based scheme but swapping users and items. That is, we have the three formulations presented in Table 6.4 where the vocabulary now may be either ratings or users, and the context variable is the user space. Based on these three formulations, and on derivations analogous to those presented before, we propose the seven item predictors defined in Table 6.5 which are further evaluated in Section 6.5.2.

In some of the instantiations of the item clarity predictor, we may observe that there are item probability models statistically equivalent to some of the user probability models, such as the $p_U(r|i,u)$ and $p_U(r|u,i)$. For this reason, we now only spec-

Item clarity name	Item dependent model	Background model
RatItem	$p_I(r\|i,u); p_{IR}(u\|i)$	$p_c(r)$
RatUser	$p_U(r\|i,u); p_{IR}(u\|i)$	$p_c(r)$
UserSimple	$p_R(u\|i)$	$p_c(u)$
UserItem	$p_{IR}(u\|i)$	$p_c(u)$
URItem	$p_I(r\|i,u)$	$p_{ml}(r\|u)$
URUser	$p_U(r\|i,u)$	$p_{ml}(r\|u)$
URItemUser	$p_{IU}(r\|i,u)$	$p_{ml}(r\|u)$

Table 6.5. Different item clarity models implemented.

ify those probability models which have not been defined before, for the rest of estimations see Equation (6.3):

$$p_R(u|i) = \sum_{r \in \mathcal{R}} p_{ml}(u|r) p_{ml}(r|i) \tag{6.17}$$

$$p_{IR}(u|i) = \sum_{r \in \mathcal{R}} p(u|i,r) p_{ml}(r|i) \tag{6.18}$$

$$p_{IU}(r|i,u) = p_{UI}(r|i,u) \tag{6.19}$$

6.2.2 Log-based clarity

In this section we adapt some of the previous models proposed for user clarity when the preference data come in the form of user-item interaction logs. Log data has a particularity we aim to exploit: the number of times a user consumes (purchased, listened, browsed, etc.) an item may be higher than one, in contrast with rating-based preferences, where the relation between a user and an item is summarised as a unique value, the rating. Moreover, the timestamp of the interactions has a stronger meaning in the implicit approach, as it informs of the very instant the user decided to use the item, rather than the time when the user decided to reflect on her quality of experience with the item (rating time). Specialised recommendation algorithms have been proposed in the literature that exploit such features in order to obtain better recommendations (Xiang et al., 2010; Lee et al., 2008). Additional alternatives for the definition of the vocabulary may be proposed, but we shall focus on these two: log co-occurrences and timestamps.

Specifically, based on Equation (6.2) and the three instantiations of $\mathcal{X}$ and θ shown in Table 6.1, in principle only an instantiation analogous to the second one ($\mathcal{X} = \mathcal{I}$, no context – to which we shall refer as frequency-based clarity) makes sense here, as there is no rating space. However, it is possible to consider an additional space, which leads to structurally similar instantiations by taking time as the $\mathcal{X}$ vocabulary. The similarity is only syntactic, as the meaning and implications of the resulting magnitude, to which we shall refer as time-based clarity, are quite different from rating-based clarity – in other words, ratings and time are quite different dimensions –, as we shall describe later below.

Frequency-based clarity

As mentioned above, we may define the following instantiation of the Equation (6.2) based on frequencies as follows:

$$\text{frequency-based clarity}(u) = \sum_i p(i|u) \log_2 \frac{p(i|u)}{p(i)} \tag{6.20}$$

where now the estimations of the user and background models are computed using directly the frequencies of the co-occurrences of some particular user-item interaction in the data:

$$p(i) = \frac{\mathrm{freq}(i)}{\sum_{j \in \mathcal{I}} \mathrm{freq}(j)}$$
$$p(i|u) = \frac{\mathrm{freq}(i,u)}{\sum_{j \in \mathcal{I}_u} \mathrm{freq}(j,u)} \tag{6.21}$$

An alternative to such estimations is to use transformations from implicit log-based to explicit ratings, such as the one proposed in (Celma, 2008). In that approach, any of the predictors based on ratings proposed in the previous section could be applied, since these transformations give the additional vocabulary space of ratings that was absent in principle in log data.

Time-based clarity

As introduced earlier, the second dimension susceptible to be exploited when log-based preference data are available is time. The time dimension is being paid increasing attention in Information Retrieval, where, for instance, it has been integrated into language models as a means to capture some temporal information needs from the user (Berberich et al., 2010), and the temporal query dynamics are being increasingly considered in the field (Kulkarni et al., 2011). In fact, temporal query features have also been used for query performance prediction, showing low or moderate correlation with query performance by themselves, although higher correlation is obtained when such features are combined with query clarity (Diaz, 2007; Diaz and Jones, 2004).

Furthermore, time has an inherent place in recommendation: recommender systems take as input (potentially long) histories of user interaction with items (Lathia, 2010; Zimdars et al., 2001; Burke, 2010). Time is an essential dimension in making sense of the data, and in explaining, analysing and interpreting the motivations behind the actions of users recorded over time. We propose to bring these ideas to recommender systems, in particular, to adapt the temporal features studied by Díaz and colleagues on a recommender system dataset. More specifically, we use the temporal Kullback-Leibler divergence described in (Diaz and Jones, 2004) as a starting point, which we generalise and elaborate upong by considering the instantiation of Equation (6.2) for a time-based space $\mathcal{X}$, and the space of items as a possible contextual dimension, as presented in Table 6.6. In the following, we define the specific instantiations of the temporal clarity formulations presented in this table.

User clarity	Vocabulary $\mathcal{X}$ / Context θ	User model	Background model	Formulation
Time-based	Time / None	$p(t\|u)$	$p(t)$	$\sum_t p(t\|u)\log_2\frac{p(t\|u)}{p(t)}$
Item-and-time-based	Time / Items	$p(t\|u,i)$	$p(t\|i)$	$\sum_{t,i} p(i)p(t\|u,i)\log_2\frac{p(t\|u,i)}{p(t\|i)}$

Table 6.6. Two temporal user clarity formulations, depending on the interpretation of the vocabulary space.

Time based model. We denote as **TimeSimple clarity** the most direct adaptation for temporal clarity, which does not use any further extension over other dimensions. It simply computes $p(t|u)$ using smoothing (see below) and $p_c(t)$ from the collection frequencies.

Item-and-time based model. Like in the previous section, we develop conditional probabilities into sums with respect to a third variable: the items rated by the user. Here, we define two temporal clarity predictors depending on the distribution assumed for the items in the summation. If the distribution is uniform we denote such predictor as **ItemTime clarity** and $p(i) = 1/|\mathcal{I}|$. If, on the other hand, we also want to incorporate the popularity of the item for – which we have more data in this context and makes more sense than in rating data, since there the interaction between a user and an item is binary –, we include the prior item probability as $p(i) = p_c(i)$, which can be estimated considering the frequency by which i is accessed based on the interaction log.

The probabilities presented above are estimated as follows:

$$\begin{aligned}
p_c(t) &= \frac{|\{(v,j,t) \in \mathcal{L} | v \in \mathcal{U}, j \in \mathcal{I}\}|}{|\mathcal{L}|} \\
p_c(i) &= \frac{|\{(v,i,s) \in \mathcal{L} | v \in \mathcal{U}, s \in \mathcal{S}\}|}{|\mathcal{L}|} \\
p_{ml}(t|i) &= \frac{|\{(v,i,t) \in \mathcal{L} | v \in \mathcal{U}\}|}{|\{(v,i,s) \in \mathcal{L} | v \in \mathcal{U}, s \in \mathcal{S}\}|} \\
p_{ml}(t|u) &= \frac{|\{(u,j,t) \in \mathcal{L} | j \in \mathcal{I}\}|}{|\{(u,j,s) \in \mathcal{L} | j \in \mathcal{I}, s \in \mathcal{S}\}|} \\
p_{ml}(t|u,i) &= \frac{|\{(u,i,t) \in \mathcal{L}\}|}{|\{(u,i,s) \in \mathcal{L} | s \in \mathcal{S}\}|}
\end{aligned} \tag{6.22}$$

Note that the variable t in (u,i,t) in the above expressions denotes a timestamp in the discretised time segment (e.g. day, week) represented by t. Furthermore, these are simple estimations of the distributions; hence, it is also possible to introduce non-parametric estimations or additional expansions through similar users or items (Wang et al., 2006a; Wang et al., 2008a). Moreover, distributions can also be modeled by other statistical theories or hypothesis (such as Bayesian inversion), and distribution fitting/modelling from time series theory could also be studied (Diaz and Jones, 2004; Wang et al., 2008b).

In particular, we have smoothed these estimations using Jelinek-Mercer as follows:

$$\begin{aligned} p(t|i) &= \lambda\, p_{ml}(t|i) + (1-\lambda)\, p_c(t) \\ p(t|u) &= \lambda\, p_{ml}(t|u) + (1-\lambda)\, p_c(t) \\ p(t|u,i) &= \lambda\, p_{ml}(t|u,i) + (1-\lambda)\, p_c(t) \end{aligned} \tag{6.23}$$

6.3 Predictors based on social topology

Social information is widespread nowadays. As we surveyed in Chapter 2, recommender systems that use social information are proliferating in the research literature, as well as in the recommender system industry, because of the effectiveness they are being found to have. It seems therefore sensible to consider social information as a potentially useful input for predicting the performance of recommendation. The motivation for this approach is obvious when applied to social recommender systems, though we will also explore its potential properties in relation to non-explicitly social recommendation, in order to study whether social topologies may have an indirect effect on the results of the algorithms for different users.

With this goal in mind, we explore the use of graph-based measures as indicators of the user strength in the social network, which may in turn correlate with the ease or difficulty of users as recommendation targets. Graph-based measures developed from link-analysis theory are straightforward to interpret where they are often used to understand the structure of communities within a population (De Choudhury et al., 2010; Albert and Barabási, 2002). As a basis for user performance prediction they may thus bring an advantage in terms of explaining the predictions.

More specifically, the utilised indicators of the user strength in the network are based on the following vertex measures computed over the social network for each user, where a user is represented as a node in the graph, and the user's friends correspond with the node's neighbours:

- **Average neighbour degree**: mean number of friends of each user's friend (Kossinets and Watts, 2006).
- **Betweenness centrality**: indicator of whether a user can reach others on relative short paths (Freeman, 1977).
- **Clustering coefficient**: probability that the user's friends are friends themselves (Watts and Strogatz, 1998).

- **Degree**: number of the user's friends in the social network (Milgram, 1967).
- **Ego components size**: number of connected components remaining when the user and her friends are removed (Newman, 2003).
- **HITS**: Kleinberg, 1999) defines two complementary measures which assign recursively a weight to each vertex (user) depending on the topology of the network. In this way, they define hubs and authorities: a vertex is a hub when it links to authoritative vertices, and is an authority when it links to hub vertices. Since the social network used here (see Appendix A.1.3) is undirected, hub and authority scores are redundant and we only report one, denoted as **HITS**.
- **PageRank** score: well-known measure of connectivity relevance within a social network based on a random walk over the vertices, where a probability ($\alpha = 0.2$ in our experiments) of jumping to any other vertex is introduced (Brin and Page, 1998).
- **Two-hop neighbourhood** size: count of all the user's friends plus all the user's friend's friends (De Choudhury et al., 2010).

6.4 Other approaches

As a reference for comparison, we shall also test further predictors besides the ones proposed in the thesis, directly drawn from the literature, and not necessarily based on probabilistic formalisations, but following more loose formalisations, or heuristic approaches. As a further sanity check, we shall also examine obvious and simple functions (such as the amount of activity of a user), as a reference for the justification of elaborate approaches as proposed. Next, we present these predictors which are evaluated and compared in Section 6.5.

6.4.1 Using rating-based preference data

A fairly simple user predictor against which we would like to compare more elaborate functions is the **count** predictor, namely the number of items a user has rated at some specific moment. This predictor, as we shall see later, can be defined in the training set and in the test set, and although its rationale is the same, the output has different implications. Whereas in training this predictor is measuring how much information a recommender knows about some specific user, in test this value would be related to the amount of relevance used to obtain the performance metric. Furthermore, as observed in Chapter 4, the amount of relevance would be different depending on the evaluation methodology considered. However, we have to note that, due to statistical effects, the training count (profile size in training) and test count (profile size in test) would probably be related if the training/test split is performed randomly.

$$\text{count}(u) = |\mathcal{I}_u| = |\{(u,\cdot,\cdot)\}| \tag{6.24}$$

Two additional heuristic predictors can be defined by looking at user statistics such as the **mean** and the **standard deviation** of the user's ratings. It seems plausible that such predictors would not be equally powerful for any type of recommender: it would depend on whether these statistics are used by the recommender. For instance, one might have the intuition that the higher the standard deviation, the lower the recommendation performance as one may figure out uniform user ratings to be a somewhat easier target.

$$\text{mean}(u) = \frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} r(u,i) \tag{6.25}$$

$$\text{std}(u) = \sqrt{\frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} \big(r(u,i) - \text{mean}(u)\big)^2} \tag{6.26}$$

Alternatively to these heuristic predictors, we have also experimented with a predictor defined upon the past observed recommender's performance. In this way, this predictor – denoted as **training performance** from now on – use a validation set (as a subset of the original training set) to evaluate the performance of each user with respect to a specific recommender; then, this value is the one returned by the predictor at test time. This approach is inspired in the Machine Learning techniques which aim to learn a feature (in this case, the user's performance) by using some training information. For this predictor, this training information is the performance computed on the validation set.

Additionally, we propose to measure the **entropy** of the user's preferences as a quantification of the uncertainty associated with a probability distribution (Cover and Thomas, 1991). We may therefore assess the uncertainty involved in the system's knowledge about a user's preferences by the entropy of the item distribution (the probability to choose an item) given the information in the user profile, using the ground models presented in Section 6.2.1. Hence, we define this predictor as follows:

$$\text{entropy}(u) = \sum_{i \in \mathcal{I}_u} p(i|u) \log_2 p(i|u) \tag{6.27}$$

Alternative measures from Information Theory could be used to define user-based predictors, like Information Gain (Bellogín, 2009), but we leave them out of this analysis because its application to Recommender Systems is neither clear nor principled and their predictive results are not optimal. Furthermore, other measures already proposed in the literature such as inverse user frequency (Breese et al., 1998) and the analogous inverse item frequency (Bellogín, 2009), and other manipulations of the same concept, are also ignored here because they are simply transformations of the previously presented count predictor. Finally, the concept of power users (Lathia et al., 2008) could also be used as a proxy for well-performing users, but preliminary results have not shown strong predictive power.

6.4.2 Using log-based preference data

As we have observed in the previous section, recommendation performance usually has obvious predictors, obvious in the sense that they do not involve any interesting finding or insightful kind of analysis, or anything to learn from. We include in our analysis some of these obvious predictors, framed as baseline performance predictors that basically count how many interactions a user has had with the system. In this sense, these predictors are slightly different to the ones presented in the previous section, namely because in log-based datasets repetitions of items are allowed in a user's profile. In order to account for this difference, apart from **count**, **mean**, and **standard deviation** predictors, we propose to normalise the count predictor by the number of items consumed by each user, that is, we define the **average count** predictor as follows:

$$\text{average count}(u) = \frac{|\{(u,j,s) \in \mathcal{L} | j \in \mathcal{I}, s \in \mathcal{S}\}|}{|\{j \in \mathcal{L} \colon (u,j,s) \in \mathcal{L}, s \in \mathcal{S}\}|} \tag{6.28}$$

We also test more elaborate predictors based on the temporal dimension, such as the ones defined in (Diaz and Jones, 2004). First-order autocorrelation (or temporal self-correlation) can be considered with a reinterpretation of the random variables. Specifically, this predictor, in contrast with the temporal Kullback-Leibler divergence where the similarity with the temporal background model is assessed, captures the structure of the query time series. For instance, a uniform distribution would have an autocorrelation value of 0, whereas a query time series with strong inter-day (or whatever segment size is used to build the discrete time series) dependency will obtain a high autocorrelation value.

Thus, we define the **autocorrelation** user predictor as follows:

$$\text{autocorrelation}(u) = \frac{\sum_{t=1}^{T}(p(t|u) - 1/\mathrm{T})(p(t+1|u) - 1/\mathrm{T})}{\sum_{t=1}^{T}(p(t|u) - 1/\mathrm{T})^2} \tag{6.29}$$

where T is the total number of time units in the time interval. We can observe how this predictor captures the similarity between two consecutive observations.

Extensions of this predictor could use the probabilities defined in Section 6.2.2, like $p(t|u,i)$, instead of $p(t|u)$. Similarly, other predictors proposed by Díaz and Jones in (Diaz and Jones, 2004) and (Jones and Diaz, 2007) such as the kurtosis or the burst model could be adapted to recommender systems, but we leave such extensions for future work.

6.5 Experimental results

In this section we provide correlation results where all the predictors – heuristic, social, and clarity-based – are compared against each other using an array of recommendation methods and evaluation methodologies.

6.5.1 User predictors using rating-based preference data

In this section we compare the correlations obtained for the clarity-based predictors defined in Section 6.2.1, the user entropy defined in Equation (6.27), and the baselines presented in Equations (6.24), (6.25), and (6.26) using the MovieLens 1M dataset. The λ parameter for the language model smoothing was not optimised for this task and a default value of 0.6 was used in all the models as originally used in (Cronen-Townsend et al., 2002). Here, we focus on Pearson's correlation and P@10. Additional results are reported in Appendix A.4.1.

Table 6.7 shows the correlation values when the AR methodology is used. We can observe fairly high correlation values for recommenders pLSA, ItemPop, TFL2, and kNN, comparable to results in the query performance literature. A slightly lower correlation is found for TFL1, whereas no correlation is found for CB and IB. These results are consistent when other performance metrics are used such as nDCG, and at different cutoff points. Spearman's correlation yields similar values. Here we also include the count predictor in test, which is obviously not a predictor in strict sense,

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2	**Median**	**Mean**
Count (training)	0.135	0.164	0.042	0.512	0.424	0.442	0.198	0.644	0.311	0.320
Count (test)	0.135	0.172	0.042	0.520	0.431	0.452	0.200	0.647	0.316	0.325
Training performance	0.024	0.176	0.258	0.429	0.296	0.357	0.215	0.485	0.277	0.280
Mean	0.019	0.067	-0.002	0.015	0.022	0.108	0.026	-0.018	0.021	0.030
Standard deviation	0.008	0.008	0.011	-0.029	-0.031	-0.032	0.011	-0.051	-0.011	-0.013
ItemSimple Clarity	0.149	0.191	0.046	0.549	0.453	0.489	0.222	0.683	0.338	0.348
ItemUser Clarity	0.134	0.166	0.048	0.493	0.416	0.428	0.215	0.634	0.316	0.317
RatUser Clarity	0.135	0.160	0.048	0.514	0.442	0.435	0.214	0.651	0.325	0.325
RatItem Clarity	0.127	0.159	0.039	0.475	0.402	0.405	0.203	0.611	0.303	0.303
IRUser Clarity	0.128	0.157	0.027	0.486	0.382	0.408	0.182	0.599	0.282	0.296
IRItem Clarity	0.122	0.165	0.034	0.446	0.352	0.386	0.188	0.551	0.270	0.281
IRUserItem Clarity	0.128	0.158	0.033	0.479	0.379	0.403	0.193	0.594	0.286	0.296
Entropy	0.121	0.168	0.025	0.492	0.389	0.483	0.140	0.589	0.279	0.301
Median	0.128	0.162	0.037	0.489	0.396	0.418	0.196	0.605		
Mean	0.112	0.145	0.033	0.413	0.338	0.367	0.166	0.511		

Table 6.7. Pearson's correlation between rating-based user predictors and P@10 for different recommenders using the AR methodology (MovieLens dataset).

since it uses a different input than the other predictors, but we include it in our analysis as a further reference to check behaviours.

As mentioned in Chapter 5, the standard procedure in Information Retrieval for this kind of evaluation is to compute correlations between the predictor(s) and one retrieval model (like in (Cronen-Townsend et al., 2002) and (Hauff et al., 2008a)) or an average of several methods (Mothe and Tanguy, 2005). This approach may hide the correlation effect for some recommenders, as we may observe from the median and mean correlation values included in the table, which are still very large despite the fact that two of the recommenders analysed have much lower correlations. Nonetheless, these aggregated values, i.e., the mean and the median, provide competitive correlation values when compared with those in the literature.

The difference in correlation for CB and IB recommenders may be explained considering two factors: the actual recommender performance and the input sources used by the recommender. With regards to the first factor, as presented in Table 6.8, the IB algorithm performs poorly (in terms of the considered ranking quality metrics, such as precision and nDCG) in comparison to the rest of recommenders. It seems natural that a good predictor for a well performing algorithm (specifically, pLSA is the best performing recommender in this context) would hardly correlate at the same time with a poorly performing one.

This does not explain however the somewhat lower correlation with the content-based recommender, which has better performance than TFL1. The input information that this recommender and the predictors take in are very different: the latter compute probability distributions based on ratings given by users to items, while the former uses content features from items, such as directors and genres. Furthermore, the CB recommender is not coherent with the inherent probabilistic models described by the predictors, since the events modeled by each of them are different: CB would be related to the likelihood that an item is described by the same features as those items preferred by the user, whereas predictors are related to the probability that an item is rated by a user. Moreover, the predictors' ground models coherently fit in the standard collaborative framework (Wang et al., 2008a), which reinforces the suitability of the user performance predictors presented herein, at least for collaborative filtering recommenders.

It is worth noting to this respect that most clarity-based query performance predic-

Recommender	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
AR methodology	0.0025	0.0163	0.0001	0.0897	0.0307	0.1454	0.0024	0.0696
1R methodology	0.0099	0.0221	0.0074	0.0649	0.0437	0.0836	0.0221	0.0690
U1R methodology	0.0100	0.0223	0.0068	0.0406	0.0381	0.0718	0.0294	0.0524
P1R methodology	0.0101	0.0197	0.0208	0.0282	0.0265	0.0604	0.0203	0.0348

Table 6.8. Summary of recommender performance using different evaluation methodologies (evaluation metric is P@10 with the MovieLens dataset).

tion methods in Information Retrieval study their predictive power on language modelling retrieval systems (Cronen-Townsend et al., 2002; Hauff et al., 2008a; Zhou and Croft, 2007) or similar approaches (He and Ounis, 2004). This suggests that a well performing predictor should be defined upon common spaces, models, and estimation techniques as the retrieval system the performance of which is meant to be predicted.

Finally, the correlation values found by the training performance predictors, although sometimes strong, are not as high as those of the baselines predictors – such as training count – in most situations, in particular, they are always lower except for the IB and TFL1 recommenders. This highlights the importance of having a more general model for predicting the performance of a user, since these predictors in fact depend considerably on the properties of the validation (and test) partition of the data, such as the amount of sparsity, type of items evaluated and so on.

Unbiased performance prediction

In Chapter 4 we already demonstrated that some methodologies may be biased towards more popular items or sparsity constraints. We can observe in the previous table that trivial predictors such as count (either in training or in test) obtain significant (and positive) correlation, no matter the recommender. We argue whether this is because these predictors are really capturing an interesting effect or the evaluation methodology is prone to such effect. In order to overcome this problem, now we present the same correlation analysis but with the different methodologies presented in Chapter 4.

In Table 6.9 we show results with the methodology 1R. Here we can observe that most of the correlation values are lower than in the previous case; interestingly, the correlation with the Random recommender now is almost 0 for every predictor (and in particular, for the training and test profile size). This is evidence that per-

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
Count (training)	0.061	-0.038	0.092	0.258	0.108	0.303	0.086	0.394
Count (test)	0.063	-0.033	0.091	0.266	0.115	0.312	0.089	0.398
Training performance	0.012	0.332	0.168	0.272	0.266	0.133	0.303	0.240
Mean	0.036	0.082	-0.029	0.028	0.111	0.117	0.145	0.031
Standard deviation	-0.010	0.006	0.051	-0.060	-0.116	-0.080	-0.040	-0.114
ItemSimple Clarity	0.066	-0.033	0.094	0.265	0.115	0.322	0.105	0.409
ItemUser Clarity	0.059	-0.038	0.087	0.236	0.100	0.287	0.096	0.375
RatUser Clarity	0.057	-0.054	0.083	0.245	0.130	0.285	0.086	0.372
RatItem Clarity	0.057	-0.044	0.069	0.225	0.110	0.268	0.094	0.352
IRUser Clarity	0.056	-0.020	0.053	0.250	0.069	0.280	0.077	0.364
IRItem Clarity	0.051	-0.010	0.058	0.205	0.029	0.235	0.074	0.310
IRUserItem Clarity	0.056	-0.020	0.052	0.242	0.066	0.273	0.081	0.357
Entropy	0.091	0.021	0.144	0.354	0.169	0.460	0.114	0.543

Table 6.9. Pearson's correlation between rating-based user predictors and P@10 for different recommenders using the 1R methodology (MovieLens dataset).

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
Count (training)	0.048	-0.012	0.237	0.162	0.115	0.140	0.022	0.235
Count (test)	0.049	-0.001	0.226	0.135	0.110	0.137	0.036	0.213
Mean	0.023	0.051	-0.035	0.009	0.108	0.075	0.155	-0.006
Standard deviation	0.015	0.032	0.023	-0.047	-0.098	-0.038	-0.061	-0.049
ItemSimple Clarity	0.055	-0.005	0.241	0.166	0.128	0.153	0.042	0.241
ItemUser Clarity	0.046	-0.009	0.232	0.142	0.109	0.133	0.028	0.216
RatUser Clarity	0.045	-0.028	0.234	0.155	0.137	0.130	0.022	0.225
RatItem Clarity	0.043	-0.025	0.212	0.136	0.119	0.117	0.033	0.203
IRUser Clarity	0.044	0.002	0.180	0.153	0.069	0.134	0.029	0.210
IRItem Clarity	0.036	0.011	0.178	0.114	0.035	0.108	0.014	0.173
IRUserItem Clarity	0.042	0.003	0.178	0.147	0.065	0.130	0.028	0.203
Entropy	0.078	0.044	0.278	0.227	0.169	0.249	0.073	0.321

Table 6.10. Pearson's correlation between rating-based user predictors and P@10 for different recommenders using the U1R methodology (MovieLens dataset).

formance results using the AR methodology are higher for users with more items in their test, independently from the recommendation algorithm complexity (see correlations with Random recommender in Table 6.7). In the same way, the U1R (Table 6.10) and P1R (Table 6.11) methodologies also obtain negligible correlation values for the Random recommender, which confirms the suitability of these methodologies for our purposes. We also have to note that we have not applied the training performance predictor in these methodologies because their restrictions do not let to replicate the same conditions in a validation split. Furthermore, as stated in Chapter 4, both approaches aim to remove the bias towards more popular items. Here, we can observe how the correlation with respect to the ItemPop recommender is comparable to that with the Random recommender with the P1R methodology, confirming again the ability of this methodology to produce unbiased results (at least, with respect to popular items).

The main difference in the results obtained between these three methodologies (1R, U1R, and P1R) seems to be more at the recommender level rather than at the

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
Count (training)	0.073	-0.005	0.253	0.088	0.103	0.160	-0.001	0.307
Count (test)	0.076	0.000	0.253	0.093	0.108	0.168	0.003	0.308
Mean	0.034	0.073	-0.033	0.008	0.110	0.085	0.188	-0.026
Standard deviation	-0.010	0.009	0.014	-0.058	-0.104	-0.044	-0.061	-0.051
ItemSimple Clarity	0.078	0.000	0.254	0.084	0.111	0.169	0.019	0.313
ItemUser Clarity	0.072	-0.001	0.249	0.075	0.101	0.156	0.005	0.303
RatUser Clarity	0.071	-0.016	0.252	0.086	0.128	0.148	0.003	0.297
RatItem Clarity	0.067	-0.011	0.234	0.077	0.113	0.138	0.016	0.288
IRUser Clarity	0.066	0.002	0.200	0.086	0.066	0.147	0.006	0.274
IRItem Clarity	0.059	0.010	0.192	0.061	0.037	0.123	-0.006	0.242
IRUserItem Clarity	0.066	0.003	0.200	0.082	0.065	0.145	0.006	0.272
Entropy	0.092	0.038	0.286	0.133	0.128	0.266	0.039	0.379

Table 6.11. Pearson's correlation between rating-based user predictors and P@10 for different recommenders using the P1R methodology (MovieLens dataset).

predictor level, in the sense that the trend in predictor effectiveness is similar for each methodology but the correlations obtained for each recommender vary dramatically from one methodology to another. For instance, IB recommender obtains near zero correlations with 1R but higher (significative) values for U1R and P1R; a similar situation occurs with the TFL2 recommender, where the correlations are lower for the U1R methodology and higher for 1R and P1R. Note that the training and test sets are the same for all the methodologies except for U1R, which means that the performance predictors are entirely new for that methodology. Thus, a priori it would not be clear that such an agreement between the different methodologies should appear at the predictor level unless they are really capturing the same nuance about the user, no matter the evaluation methodology used.

It is worth noting that the correlation values of these three methodologies have been found after a careful examination of the available data, where two different trends emerged: one where the performance values were more or less uniformly distributed in the interval [0,0.1] – recall that 0.1 is the maximum value for the metric P@10 with the 1R methodology, since there is only one relevant item – ; and a second one where a fixed value was obtained. This second trend, against which our predictors shown no correlation at all (since the performance had a zero standard deviation, and thus the correlation was impossible to calculate) is able to degrade the correlation coefficient almost to negligible values, mainly because it accounts for half of the number of points. This problem with correlation coefficients, and with Pearson's correlation in particular, is well known in the literature of performance prediction (Hauff, 2010; Pérez Iglesias, 2012). For this reason, we have divided the performance values and computed two correlations in order to account for these two trends: the values with respect to the first trend are those presented in the previous tables, whereas the correlation with respect to the second trend was not computable because the variable had a zero standard deviation.

In summary, there seems to be no clear winner among the set of performance predictors proposed. The predictive power of each of them is clearly influenced by the actual recommender its performance aims to be predicted and the evaluation methodology in use. Nonetheless, **the proposed predictors usually obtain higher correlation values than baseline predictors** such as the mean or the standard deviation, evidencing their predictive power **independently from the evaluation methodology**. Surprisingly, the ItemSimple clarity predictor obtains very good results in most of the situations, although more complex predictors like IRUser or IRUserItem clarity obtain stronger correlations for some recommenders.

6.5.2 Item predictors using rating-based preference data

In the same way we have assessed the predictive power of user predictors, we now aim to estimate the predictive power of item predictors. However, the true perform-

	u_1	u_2	u_3
	i_1 0.8	i_2 0.6	i_3 0.9
	$*i_2$ 0.7	$*i_3$ 0.5	$*i_4$ 0.6
	$*i_3$ 0.6	$*i_4$ 0.4	$*i_1$ 0.5
	i_4 0.5	i_1 0.3	i_2 0.1
P@2	0.5	0.5	0.5

i_1	i_2	i_3	i_4
$*u_1$ 0.8	u_1 0.7	u_3 0.9	$*u_3$ 0.6
$*u_3$ 0.5	u_2 0.6	$*u_1$ 0.6	u_1 0.5
u_2 0.3	u_3 0.1	$*u_2$ 0.5	$*u_2$ 0.4
1.0	0.0	0.5	0.5

Table 6.12. Procedure to obtain ranking for items from user rankings generated by a standard recommender. * denotes a relevant item, and the numbers are the score predicted by the recommendation method.

ance value for an item is not straightforward to compute, since the process has to produce unbiased results in the space of items (as described in Chapter 4) but with the characteristic that the item dimension is not the main input of the recommendation process, and thus, sone new approach has to be put in place.

There are basically two possibilities for computing the true performance on an item: either starting from the results obtained using a standard procedure (obtain a ranking for each user by recommending items to users), then transposing users and items (generating, thus, user rankings for each item) and computing the per-ranking performance as usual; or transpose the original rating matrix in order to effectively "recommend users" for each item. This would implicitly imply a transposition of the recommendation task, which may also make sense: find the most suitable users to recommend an item – this would be the scenario, e.g. in advertisement targeting when a new product is released on the market. Here, we use the former approach since the latter does not produce consistent results in our experiments, probably because the recommendation problem is not completely symmetric and, thus, this method is not able to properly capture the recommender's performance for each item. On the other hand, non-personalised recommenders (such as recommendation by item popularity) cannot be applied in the symmetric formulation: since the same item ranking is built for all users, the user ranking for an item would be a global tie on all users. Table 6.12 shows an example of how we may transpose users and items from an item ranking for three users. We show that the precision for all the users is the same, whereas for the items is completely diverse, ranging from zero to perfect precision.

In our experiments, we have tested the different methodologies already presented along with a modified version of the U1R evaluation methodology (user-uniform U1R, or uuU1R). The rationale for the uuU1R design goes as follows: in the U1R methodology we force the same number of ratings (or, equivalently, users) for the items in the test set, however, users are freely assigned to each item. Now, when we transpose users and items this situation may produce a new problem, since there

Predictor	Random	CB	ItemPop	kNN	pLSA
Count (training)	0.414	0.060	-0.151	-0.021	-0.269
Count (test)	-------	-------	-------	-------	-------
Mean	0.602	0.125	0.096	0.040	-0.038
Standard deviation	-0.313	0.025	-0.006	-0.003	0.075
UserSimple Clarity	0.467	0.080	-0.120	-0.015	-0.240
UserItem Clarity	0.419	0.064	-0.145	-0.018	-0.261
RatItem Clarity	0.440	0.075	-0.127	-0.015	-0.230
RatUser Clarity	0.451	0.085	-0.103	-0.004	-0.201
URItem Clarity	0.396	0.053	-0.174	-0.026	-0.289
URUser Clarity	0.408	0.072	-0.132	-0.004	-0.243
URItemUser Clarity	0.409	0.061	-0.161	-0.021	-0.277
Entropy	0.381	-0.001	-0.216	-0.055	-0.442

Table 6.13. Pearson's correlation for rating-based item predictors and precision using the uuU1R methodology (MovieLens dataset).

could be users assigned to more items which would bias the ranking's performance towards items contained in the test set of heavy raters. Therefore, if we impose a uniform distribution also on the user's dimension, this bias should decrease. We refer to the reader to Appendix A.3 for more details.

However, despite these efforts, we have not found a reliable methodology to evaluate the item performance. We present in Table 6.13 the results using the uuU1R methodology and the predictors defined in Table 6.5 for the precision metric. Recall that, since we transpose users and items from the generated rankings, to obtain a similar measure of P@10 we only use the top 10 items from each original ranking and then compute precision over the whole ranking for each item. We may observe in the table that the correlations with the Random recommender are very strong, questioning the validity of such results. Besides, the entropy predictor obtains stronger correlation than clarity-based in this case, and most of them (except for URItem) show little difference to training count. Note that it is not possible to compute a correlation with the test count predictor since that predictor has a constant value with zero standard deviation (see Equation (5.11) for more details on Pearson's correlation) since every item has the same number of ratings in the test set in the uuU1R methodology.

As a conclusion, we have found that **a proper evaluation of item performance is not obvious**, mainly because the task of suggesting users to items is not completely symmetric with respect to the standard task of recommendation. We have devised different methodologies to estimate the recommendatoin performance of an item, however the difficulty lies mainly in forming consistent lists of "recommended" users for items, a difficulty which is not conceptual (ranking target users to whom an item may be recommended does make sense as a task in many scenarios), but technical (obtaining balanced result lists that allow for undistorted performance measurements).

6.5.3 User predictors using log-based preference data

In this section we analyse the correlation obtained between the predictors defined in Sections 6.2.2 and 6.4.2 and five recommenders using the 1R methodology on two versions of the Last.fm dataset – one where a temporal partition is performed and another where the partition is randomly made (more details about the splits in Appendix A.1.2). No smoothing was used in the language models since preliminary tests obtained better results with lower values of λ. Besides, for comparison purposes, we also include one of the clarity models proposed for rating-based preference data using the transformation proposed in Section 6.2.2 to use such predictors with log data along with the frequency-based clarity proposed in Equation (6.20). Like in the previous section, Pearson's correlation with the P@10 evaluation metric is reported; for additional metrics, see Appendix A.4.2.

First, we can observe in Table 6.14 (temporal split) that ItemPriorTime clarity obtains strong correlation values, especially for the ItemPop and kNN recommenders. It is interesting to compare the correlations between this predictor and the ItemTime clarity, which are much lower. This is probably because the ItemPriorTime clarity predictor, as opposed to ItemTime clarity, incorporates a component that measures the item popularity, i.e., $p(i)$. The TimeSimple and the frequency-based clarity predictors, on the other hand, obtain strong correlation but negative values for all the recommenders except the ItemPop for the TimeSimple predictor. Furthermore, the ItemSimple clarity (a predictor based on explicit information) obtains negligible correlations except for the ItemPop and kNN recommenders.

Table 6.15, on the other hand, shows the results when a random split is used. We have to note that such split does not preserve the temporal continuity of the user's preferences, and thus, any recommender or technique which makes use of temporal features is not guaranteed to succeed. Here, we can observe that TimeSimple predictor obtains strong correlations for all the recommenders except for the Random

Predictor	Random	CB	ItemPop	kNN	pLSA
Average Count	0.027	0.138	0.069	-0.013	0.191
Count	0.046	0.118	-0.058	0.131	0.139
Mean	-0.079	-0.361	0.054	-0.110	-0.376
Standard deviation	-0.050	-0.158	0.082	-0.132	-0.177
Autocorrelation	0.004	0.139	-0.066	-0.105	0.100
TimeSimple Clarity	-0.091	-0.342	0.093	-0.317	-0.354
ItemTime Clarity	0.037	0.078	0.038	0.258	0.064
ItemPriorTime Clarity	0.057	0.154	0.189	0.307	0.154
Frequency-based Clarity	-0.049	-0.410	-0.221	-0.291	-0.376
ItemSimple Clarity	0.027	0.047	-0.107	0.221	0.029

Table 6.14. Pearson's correlation between log-based predictors and P@10 for different recommenders using 1R methodology (Last.fm temporal dataset).

Predictor	Random	CB	ItemPop	kNN	pLSA
Average Count	-0.023	-0.068	-0.170	-0.018	-0.087
Count	-0.012	-0.236	-0.242	-0.086	-0.198
Mean	0.036	0.182	0.100	0.047	0.118
Standard deviation	-0.009	0.089	0.079	0.092	0.082
Autocorrelation	0.045	-0.069	-0.089	-0.012	-0.055
TimeSimple Clarity	0.031	0.274	0.314	0.169	0.240
ItemTime Clarity	0.021	-0.145	0.004	0.025	-0.053
ItemPriorTime Clarity	0.011	-0.057	0.176	0.145	0.083
Frequency-based Clarity	0.025	0.018	-0.287	-0.182	-0.220
ItemSimple Clarity	0.020	-0.247	-0.163	-0.068	-0.186

Table 6.15. Pearson's correlation between log-based predictors and P@10 for different recommenders using 1R methodology (Last.fm five-fold dataset).

technique. Like before, ItemPriorTime has a high correlation with the ItemPop recommender. In contrast with the previous situation, the ItemSimple clarity obtains strong but negative correlations for the personalised recommenders. Besides, the frequency-based clarity has negative correlations for all the recommenders except CB, a consistent situation with the results obtained with the temporal split.

Hence, we may conclude that **log-based and time-aware predictors successfully predict the performance of the recommendation algorithms**, although in some situations the sign of the prediction is negative. Moreover, frequency-based, ItemSimple, and TimeSimple clarity obtain consistently strong correlations both in a temporal split and in a random split of the data, evidencing their predictive power.

6.5.4 User predictors using social-based preference data

In this section we study the correlation between the predictors described in Section 6.3 and several recommenders using the two versions of the CAMRa dataset: social and collaborative. In this case, we also consider social filtering recommenders in order to analyse whether these predictors are sensitive to the source of information used by the recommender, and thus, whether they obtain stronger correlations with social filtering recommenders. Besides, one clarity-based predictor (ItemSimple) and the baseline rating predictors presented in Section 6.4.1 are also included in the analysis for comparison purposes. Additionally, for the HITS and PageRank graph metrics in this experiment we use the implementation developed in the JUNG library (O'Madadhain et al., 2003).

Table 6.16 shows correlation values obtained when using the AR methodology in the social version of the dataset. Here, we can observe that most of the correlation values obtained for the social predictors are negative, representing that the lower the predictor output, the better the performance, which may seem a little counter-intuitive, at least for the social filtering recommenders (Personal and PureSocial).

Among the social-based predictors, degree and two-hop neighbourhood size obtain better correlations than the rest.

A similar situation is presented in Table 6.17, where the collaborative-social version of the dataset is used. Again, most of the correlations with the social-based predictors are negative, and degree and two-hop neighbourhood size obtain higher correlations (in absolute value). Interestingly, in this situation strong correlations are obtained with the user-based recommender (kNN), in particular with degree and the average neighbour degree predictors. Nonetheless, these correlations are lower than those obtained for the ItemSimple predictor with the collaborative filtering recommenders. At the same time, this predictor always obtains worse correlations (in absolute value) than the social-based predictors for the social filtering recommenders, as expected.

Additionally, note that the number of points used in the correlation computation is different in each version of the dataset, namely: in the collaborative-social version

Predictor	Random	ItemPop	kNN	pLSA	Personal	PureSocial
Count (training)	0.032	0.122	0.113	0.031	0.062	0.111
Count (test)	0.158	0.252	0.382	0.167	0.235	0.174
Mean	-0.066	0.033	-0.012	0.023	-0.057	-0.051
Standard deviation	0.034	0.054	-0.020	0.115	0.128	0.183
Avg neighbour degree	-0.062	-0.089	-0.013	0.011	-0.074	-0.106
Betweenness centrality	-0.031	-0.016	0.027	-0.038	-0.012	-0.079
Clustering coefficient	0.049	-0.084	-0.023	0.048	-0.027	-0.035
Degree	-0.038	-0.046	0.015	-0.059	-0.147	-0.133
Ego components size	-0.058	0.005	0.004	-0.046	-0.056	-0.020
HITS	-0.021	-0.043	0.005	0.061	0.038	0.000
PageRank	-0.022	-0.025	-0.023	-0.039	-0.102	-0.037
Two-hop neighbourhood	-0.080	-0.082	0.004	-0.054	-0.123	-0.136
ItemSimple Clarity	0.030	0.157	0.130	0.050	0.072	0.126

Table 6.16. Pearson's correlation between social-based predictors and P@10 for different recommenders using <u>AR methodology</u> (CAMRa Social).

Predictor	Random	ItemPop	kNN	pLSA	Personal	PureSocial
Count (training)	0.012	0.098	0.203	0.107	0.058	0.111
Count (test)	0.096	0.207	0.389	0.179	0.232	0.170
Mean	-0.067	0.000	-0.126	-0.024	-0.051	-0.050
Standard deviation	0.082	0.014	-0.029	0.016	0.129	0.182
Avg neighbour degree	0.071	-0.008	0.152	0.046	-0.073	-0.104
Betweenness centrality	-0.007	-0.008	0.010	-0.005	-0.012	-0.078
Clustering coefficient	0.006	-0.022	0.152	0.076	-0.032	-0.035
Degree	0.032	0.018	0.164	0.006	-0.143	-0.134
Ego components size	0.026	0.044	0.133	0.002	-0.053	-0.022
HITS	-0.011	-0.034	-0.001	0.061	0.038	0.001
PageRank	-0.002	0.021	0.118	0.014	-0.099	-0.040
Two-hop neighbourhood	0.059	-0.015	0.130	0.012	-0.121	-0.135
ItemSimple Clarity	0.010	0.120	0.211	0.129	0.070	0.126

Table 6.17. Pearson's correlation between social-based predictors and P@10 for different recommenders using <u>AR methodology</u> (CAMRa Collaborative).

the number of users contained in the test set is twice the number available in the social version (see Appendix A.1.3), which means that significant correlations can be achieved with lower values (as described in Chapter 5).

In the results described above, we can observe how, like in the previous sections, the size of the user profile in test (predictor count in test) obtains significant correlations. This trend, however, is almost neutralised in the collaborative-social dataset with respect to the Random recommender. Thus, as before, we would attempt to use the 1R methodology with each dataset in order to obtain unbiased correlations towards users with more ratings in test. However, due to the lack of coverage of Personal and PureSocial recommenders, this methodology do not obtain sensible results (for instance, the value of precision at 10 is almost invariably 0.10, that is, the maximum possible value when only one relevant document – as assumed in the 1R methodology – is retrieved in the top 10, mainly because the recommender is not able to retrieve most of the not relevant items). This lack of coverage is natural for these recommenders since they can only suggest items rated by users in the active user's social network (see Appendix A.2 for details on the implementation of the algorithms).

In conclusion, most of the social performance predictors proposed obtain significant correlations, however, **correlations with the social filtering methods are not so strong as we would expect**. Nonetheless, the **ItemSimple clarity does obtain significant correlations** with respect to most of the recommenders, highlighting the importance and validity of this predictor even when the main input of some recommenders (social network) is so different to that of the predictor (ratings).

6.5.5 Discussion

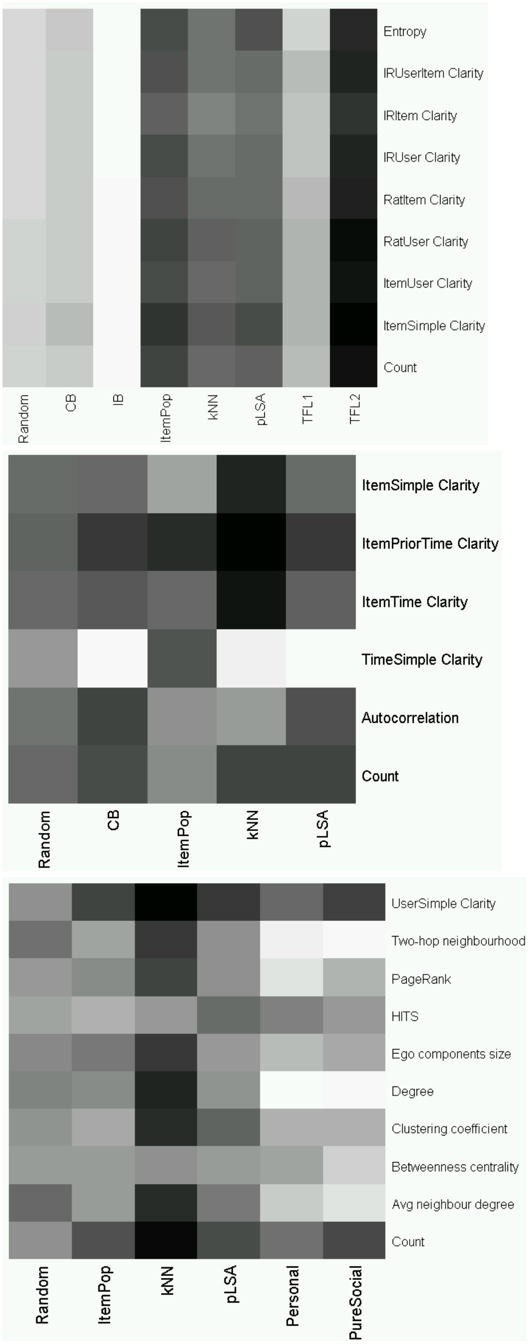
The reported experiments confirm that it is possible to predict a recommender's performance and obtain strong correlations in this regard. The results show that, in general, the proposed predictors (mostly based on Kullback-Leibler divergences over different language models and other concepts from Social Graphs and Information Theory such as entropy) obtain significant correlations in the three spaces considered: ratings, logs, and social networks. More importantly, these correlations are stronger than those obtained by more simple predictors, such as the profile size of a user, the standard deviation of her ratings, and the user's performance using a validation split. Specifically, for each recommendation input considered we have observed the following:

- Clarity-based predictors are very powerful for rating-based preferences, in particular, the ItemSimple, IRUser, and IRUserItem clarity predictors obtain strong correlations for most of the recommendation methods.

- The use of the item space as a contextual variable shows strong correlation values when the AR methodology is used, but these correlations decrease when we use unbiased methodologies, which may indicate that this new dimension is in fact capturing the item popularity and, thus, when the popularity bias is neutralised such predictors show less predictive power. We find a similar situation with the item clarity and the user space used as the contextual dimension.
- Temporal and log-based versions of the clarity predictor show higher prediction power than the rest of predictors.
- Social-based predictors are not the ones with the strongest correlation regarding the social filtering recommenders in this experiment, but the correlation found is significative and they could serve as a complement to other predictors based on a different input such as the rating-based.
- The ItemSimple clarity predictor consistently obtains strong correlation values in most of the datasets where we have analysed it. This is an evidence of the theoretical power of the user clarity to capture the uncertainty in user's tastes, even when the recommender's input is different (social filtering recommenders) or when we apply some transformation to the data (frequency-based clarity with transformation from implicit to explicit).
- As described in the Appendix A.4, most of the correlations presented in this chapter are stable when other evaluation metrics and correlation coefficients are used.

In the Recommender Systems field there are, however, additional problems due to subtle differences with respect to the common settings and experimental assumptions in Information Retrieval. Since we aim to predict the performance of a recommender, we have to be sure that we are using an unbiased performance metric, and its subsequent evaluation methodology. As we analysed in Chapter 4 there are at least two biases in the evaluation of recommender systems which may distort the results: data sparsity and item popularity. Thus, in this chapter we have computed correlations between the output of the predictors and the evaluation metrics using different evaluation methodologies, in order to analyse how sensitive the different proposed predictors are to these biases. Interestingly, although the correlations may change drastically when different evaluation methodologies are considered, most of the performance predictors still obtain good correlations. In particular, this result evidences that our proposed predictor are not so prone to the analysed biases like other simple predictors.

Finally, in Figure 6.3 we summarise the correlations found for the proposed predictors in each dimension – ratings, logs, and social. We have selected the most representative evaluation methodology (AR for rating and social data, and 1R for log



Correlation between rating-based user predictors and recommenders using AR methodology in MovieLens (Table 6.7).

Correlation between log-based user predictors and recommenders using 1R methodology in Last.fm (Table 6.14).

Correlation between social-based user predictors and recommenders using AR methodology in CAMRa collaborative (Table 6.17).

Figure 6.3. Heatmap of the correlation values between a subset of predictors and recommenders, using the most representative methodologies for the three considered spaces.

data) and a subset of the evaluated predictors and recommenders from each experiment, where the same information presented in Table 6.7, Table 6.14, and Table 6.17 (except for the average and median correlation values) is depicted in a more visual form. In particular, we may observe that predictors in MovieLens seem to be more redundant since the correlations are too similar.

From the figure we may also observe that in Last.fm and CAMRa datasets such redundancy is much lower and the predictors are quite different. Moreover, the first column and row (from the bottom) represent the recommender and predictor baselines, which serve as references from where the correlations should be analysed. In the three cases we can observe that most of the predictors obtain larger (darker) values than the count predictor. In the first case (rating-based predictors), however, it is clear that the correlation depends more on the recommender and less on the actual predictor.

6.6 Conclusions

We have proposed adaptations of query performance techniques from ad-hoc Information Retrieval to define performance predictors in Recommender Systems. Taking inspiration in the query predictor known as query clarity, we have defined and elaborated in the Recommender Systems domain several predictive models according to different formulations and assumptions. Furthermore, we propose performance predictors from theories and models of Information Theory, Social Graph Theory, and Information Retrieval based on three types of preference data: rating-based, log-based, and social-based.

We find several effective schemes with a high predictive power for recommender systems performance. We have proposed different ways for the adaptation of the query clarity predictor to recommender systems depending on the equivalences between the involved spaces. The clarity formulation is powerful because of its theoretical soundness, which is suitable to different domain-oriented adaptations. Hence, for rating-based preferences we use different expansions which take into account the rating values and the items rated by the user. For log-based preferences we exploit the co-occurrences of the items in the user profile and, more importantly, the temporal dimension, which allows for more principled functions such as the temporal Kullback-Leibler divergence or the user's autocorrelation. Finally, for social-based preferences we exploit the user's social network and different graph metrics are used apart from the user clarity based on the ratings. The results, as summarised in the previous section, are in general positive and provide evidences that the proposed functions are able to indeed predict the performance of user or items in recommender systems.

Furthermore, by analysising the behaviour of trivial predictors (such as the count of ratings in training and test) we have been able to uncover noisy biases or sensitivity to irrelevant variables in the way performance is measured. Irrelevant and uninteresting in the sense that it is not clear that the variations due to these variables really reflect actual differences in quality. As a result, we have used unbiased evaluation methodologies where non trivial predictors still obtain positive results with respect to performance correlation.

As a side-effect, our study introduces an interesting revision of the gray sheep user concept. A simplistic interpretation of the gray sheep intuition would suggest that users with a too unusual behavior are a difficult target for recommendations. It appears however in our study that, on the contrary, users who somewhat distinguish themselves from the main trends in the community are easier to give well-performing recommendations. This suggests that perhaps the right characterisation of a gray sheep user might be one who has scarce overlap with other users. On the other hand, the fact that a clear user distinguishes herself from the aggregate trends does not mean that she does not have a sufficiently strong neighbourhood of similar users. In particular, this seems to indicate that users who follow mainstream trends are more difficult to be suggested successful items by a recommender system (at least, by a personalised one). In Information Retrieval, one can observe a similar trend: more ambiguous (mixture of topics) queries perform worse than higher-coherence queries (Cronen-Townsend et al., 2002).

In the future we plan to explore further performance predictors. Specifically, we are interested in incorporating explicit recommender dependence into the predictors, so as to better exploit the information managed by the recommender, allowing to the predictor a smoother adaptation to the recommender performance, and increasing the final correlation between them. Additionally, we are also interested in exploring alternative item-based predictors apart from those defined in this chapter, and, eventually, using other information sources such as log-based preference data and even the social network of the users who rated a particular item.