A Multicriteria Evaluation of Hybrid Recommender Systems: On the Usefulness of Input Data Characteristics

Reinaldo Silva Fortes[1,2], Alan R. R. de Freitas[2] and Marcos André Gonçalves[1]
[1]*Computer Science Department, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil*
[2]*Computer Department, Universidade Federal de Ouro Preto, Ouro Preto, Brazil*

Keywords: Recommender Systems, Information Filtering, Hybrid Filtering, Collaborative Filtering, Content-based Filtering, Meta-feature.

Abstract: Recommender Systems (RS) may behave differently depending on the characteristics of the input data, encouraging the development of Hybrid Filtering (HF). There are few works in the literature that explicitly characterize aspects of the input data and how they can lead to better HF solutions. Such work is limited to the scope of combination of Collaborative Filtering (CF) solutions, using only rating prediction accuracy as an evaluation criterion. However, it is known that RS also need to consider other evaluation criteria, such as novelty and diversity, and that HF involving more than one approach can lead to more effective solutions. In this work, we begin to explore this under-investigated area, by evaluating different HF strategies involving CF and Content-Based (CB) approaches, using a variety of data characteristics as extra input data, as well as different evaluation criteria. We found that the use of data characteristics in HF proved to be useful when considering different evaluation criteria. This occurs in spite of the fact that the experimented methods aim at minimizing only the rating prediction errors, without considering other criteria.

1 INTRODUCTION

Recommender Systems (RS) (Jannach et al., 2010; Ricci et al., 2011) have gained significant attention over the past decades due to information overload regarding the items available to users. There exist several RS approaches proposed in the literature. The most commonly known ones are Collaborative Filtering (CF), Content-Based Filtering (CB), and Hybrid Filtering (HF). Knowing that each approach has strengths and weaknesses, HF has as core principle to combine such strengths while minimizing the weaknesses, regardless of the type of combination and approaches exploited (Burke, 2002; Burke, 2007).

In search for more effective results, some authors started to exploit alternative sources of information as additional features to obtain a more profitable combination. In particular, two works (Bao et al., 2009; Sill et al., 2009) have developed HF methods using characteristics of the input data (hereinafter *meta-feature*) as additional features. These *meta-features* are designed to measure some implicit characteristics of the input data, such as the amount of data (*e.g.*, number of ratings), temporal relations (*e.g.*, number of different dates on which a user has provided ratings) and relationships between items (*e.g.*, number of items with high similarity with a particular item).

Those solutions focus mainly on the combination of CF approaches. Perhaps even more importantly, they focus on the analysis of only one dimension of the problem: the rating prediction accuracy. However, the literature in RS has recognized that several aspects are important for recommendation besides prediction accuracy, especially those based on ranked lists, such as relevance, novelty and diversity. Despite their similarities, a comparison of these works was not found in the literature.

To summarize, despite the diverse behaviors, few RS make use of *meta-features*, exploiting a limited number of alternatives regarding the exploitation of different approaches and various facets of the problem (*i.e.*, different evaluation criteria). Thus, this work provides some novel contributions to this field by performing an extensive experiment on the use of *meta-features* in HF. We evaluate solutions considering a large number of alternatives in order to better understand the impact of the use of *meta-features* in RS in a multicriteria analysis. In particular, we try to answer the following research question: *are the meta-features really useful for HF when considering different eval-*

Fortes, R., Freitas, A. and Gonçalves, M.
A Multicriteria Evaluation of Hybrid Recommender Systems: On the Usefulness of Input Data Characteristics.
DOI: 10.5220/0006315406230633
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 2*, pages 623-633
ISBN: 978-989-758-248-6

uation criteria?

In our empirical analysis we answered the research question observing that the use of *meta-features* in HF solutions proved useful when considering different evaluation criteria. As a consequence, we can highlight the need to deepen the research in this area in future works, developing new multi-objective methods that use *meta-features* to obtain better results taking into account several objectives simultaneously. We also observed a distinct behavior for all experimented datasets when comparing the different RS approaches, confirming that the algorithms are influenced in different forms by the input data characteristics and that HF remains not omnipotent despite the improvements obtained with the use of *meta-features*.

The main contributions of this work are:

- A comprehensive analysis of the impact of *meta-features* in HF solutions, considering many alternatives and aspects of the recommendation problem.
- The definition of a set of *meta-features* which covers the main characteristics of CF and CB input data.
- The definition of an effective experimental strategy for the evaluation of methods that use *meta-features*.
- Making available the datasets used in the experiments composed of the original data and the additional processed data[1].

This paper is organized as follows. Section 2 reviews related works. Section 3 presents the formal definition of *meta-features* and the algorithms explored. Section 4 presents the resources, settings, and the experimental strategy applied. Section 5 discusses the experimental results. Finally, Section 6 concludes the paper.

2 RELATED WORK

In this section we review related works, starting with studies regarding the importance of the input data characteristics and the use of different evaluation criteria. Thereafter, we deal with works concerning the use of *meta-features* in HF solutions to improve recommendation results.

(Breese et al., 1998), (Herlocker et al., 2004), and (Gunawardana and Shani, 2009), compared the predictive accuracy of various CF methods, in distinct domains. In their results, they highlighted that the best algorithm depends on specific factors, such as application domain, the number of users, items, and ratings, and the rating value scale. (Adomavicius and Zhang, 2012) investigated the impact of rating *meta-features* on CF algorithms. The strategy applied was based on a sampling procedure, selecting samples with different metric values. They conclude that rating *meta-features* may, in fact, impact the CF approaches' accuracy, underlining that a bigger rating space size and higher data density produces a positive effect, while a larger standard deviation incurs in a negative effect.

(Herlocker et al., 2004) also observed that another important factor is the evaluation objective, considering that, in some cases, the goal is to improve the accuracy in predicting ratings. In others, it is how the system influences the users' satisfaction. (McNee et al., 2006) highlighted the importance of user-centric evaluation instead of just accuracy. Accordingly, some authors developed methods to increase the diversity of the recommendations (Ziegler et al., 2005; Zhang and Hurley, 2008) and some recent works apply optimization techniques to address multiple goals simultaneously (Ribeiro et al., 2014; Zuo et al., 2015; Geng et al., 2015; Wang et al., 2016).

The best two teams in the Netflix Prize made use of Weighted HF. Among these works, (Sill et al., 2009) proposed the Feature-Weighted Linear Staking (FWLS). They combined a variety of *meta-features* calculated from the available ratings and several purely CF algorithms using standard linear regression methods and evaluating the results according to the prediction accuracy.

(Bao et al., 2009) proposed another Weighted HF strategy using *meta-features* – the STacking Recommendation Engines with Additional Meta-features (STREAM). Although the authors report the exploration of CF and CB, the use of content is restricted to the users' and items' neighborhood definition, predicting the ratings based on CF approaches. They concluded that the use of different *meta-features* reached similar prediction accuracy as those obtained using only the number of ratings.

To summarize, the aforementioned works demonstrate that RS are somewhat influenced by *meta-features* and its use in HF was effective to produce better prediction accuracy and the importance of considering different criteria. However, all of them have limitations regarding the use of *meta-features*, the RS strategies used and the evaluation criteria applied. In this sense, we believe that we provide the most comprehensive evaluation reported so far to better understand how RS behaves regarding all these aspects.

[1]Available at http://www.decom.ufop.br/reinaldo/publications/iceis2017/.

3 META-FEATURES AND ALGORITHMS

A key issue in the present work is the definition of *meta-features* capable of characterizing the input data and the selection of algorithms that help find the answer to our research question. Thus, this section describes the *meta-features* and algorithms explored in this work in three subsections. Subsection 3.1 describes the *meta-features*. Subsection 3.2 presents the Individual algorithms used to build the input features for HF solutions. In Subsection 3.3 the HF strategies are described.

3.1 Meta-features

To define the *meta-features*, some issues must be observed: **(i)** in CF, the input is a *rating matrix* mapping the user satisfaction about items. Additionally, the date and time definition of ratings can be provided. Thus, the characteristics would be related to the number of ratings involved, the distribution of their values, and their date definition; and **(ii)** in CB, the input is a set of items' attributes that can be defined by structured and unstructured data and a set of preferred items by users. Thus, the characteristics would be related to the volume of the content available and similarity measures calculated for such contents.

We exploit a set of *meta-features* able to characterize the input data regarding all aforementioned aspects related to CF and CB approaches in a broad scope but with a minimal number of *meta-features*. Table 1 lists the *meta-features*, their types and short descriptions. Formalization and details are given below.

The computation of the *meta-features* was formalized to get a set where each value is related to an element in Equation 1. Table 2 lists the *meta-features* and their mathematical definitions for the Equation 1 computation.

$$\mathrm{MF}(\Sigma,\delta,\sigma)=\left\{\sum_{e}\left(\delta\left(\sigma_{e}\left(\mathcal{D}\right)\right)\right),\forall e\in\mathcal{D}\right\} \quad (1)$$

where: $\mathcal{D}$ is the input data; σ_e is a selection function applied on $\mathcal{D}$ for the element e, which can be a user or an item; δ is a measure calculated for the selected subset of the input data; and Σ is a function, which might be *count*, *average*, *logarithm*, etc., calculated for each element e.

COUNTSIZE aims to quantify the size of the content available. It is based on the number of *tokens* (*i.e.* processed words) extracted from the textual attribute values.

COSINE, DICE, and JACCARD aim to quantify similarities between the element e and other elements, using of the Cosine Similarity (Baeza-Yates and Ribeiro-Neto, 1999), the Dice Coefficient (Adar et al., 2009), and the Jaccard Index (Baeza-Yates and Ribeiro-Neto, 1999) as similarity measure, respectively. The Dice coefficient is originally used by (Adar et al., 2009) to compute the differences between two versions of the same *document* over time. We use it to compute the differences between the content from two different elements.

ENTROPY aims to quantify the cohesiveness of the element's content via the Entropy measure (Bendersky et al., 2011). Low values indicate a tendency for the content to cover a single topic.

LOGQTDR aims to capture a notion of the amount of ratings available via the logarithm of the number of ratings (Sill et al., 2009).

PCR (*Proportion of Common Ratings*) aims to capture a notion of the size of the neighborhood via the concept of *users in common* and *items in common*. *Users in common* are those who ranked the

Table 1: Meta-features. Types: (I) Content; (II) Number of ratings; (III) Rating values; (IV) Rating dates.

Acronym	Type	Short Description
COUNTSIZE	I	Number of indexed tokens from content.
COSINE	I	Average of Cosine Similarity calculated for pairs of users' or items' content.
DICE	I	Average of Dice calculated for pairs of users' or items' content.
ENTROPY	I	Average of Entropy calculated for individual users' or items' content.
JACCARD	I	Average of Jaccard calculated for pairs of users' or items' content.
LOGQTDR	II	Log of the number of ratings.
PCR	II	Percentage of users or items in common.
PR	II	Percentage of the number of ratings.
GINI	III	Gini Index calculated for the rating values.
PEARSON	III	Pearson's Coefficient of Variation calculated for the rating values.
PQMEAN	III	pq-mean Index calculated for the rating values.
SD	III	Standard deviation calculated for the rating values.
LOGDATER	IV	Log of the number of distinct rating dates.
PRDATER	IV	Percentage of the number of rating dates.

same items that a user ranked and *items in common* are those that were ranked by the same users who ranked an item.

PR (*Proportion of Ratings*) also aims to capture a notion of the amount of ratings available, as well as LOGQTDR. The difference is that it is not logarithmic, but a percentage.

GINI, PEARSON, PQMEAN, and SD aim to capture the rating values distribution, using the Gini Index (Hurley and Rickard, 2009), the Pearson's Coefficient of Variation, the *pq*-mean metric (Hurley and Rickard, 2009), and the Standard Deviation, respectively.

LOGDATER aims to capture a notion of time about the ratings via the logarithm of the number of distinct dates in which a user gave ratings or an item received ratings. Σ is the logarithm, δ is the number of distinct dates, and σ_e consists of the selection of the ratings' dates for the element.

PRDATER (*Proportion of Rating's Dates*) also aims to capture a notion of time, as well as LOGDATER. The difference is that it is not logarithmic, but a percentage.

We evaluate different combinations of attributes for all content *meta-features* in preliminary experiments and due to the correlation between their values we used all attributes in the final experiments produced in this work.

3.2 Individual Algorithms

The Individual algorithms used in this study were named as: **(i)** *CF*: purely CF algorithms and those using CB strategies to measure similarities, but a rating prediction is made using *CF* strategies; and **(ii)** *CB*: purely CB algorithms.

The set of *CF* algorithms explored in this work are provided by the MyMediaLite (Gantner et al., 2011), including most of the state-of-the-art techniques, such as K-Nearest Neighbors (kNN), Matrix Factorization, Singular Value Decomposition, and Asymmetric Factor Model (AFM).

A *CB* algorithm was implemented using Apache Lucene (McCandless et al., 2010). The algorithm performs the indexing of the items' content, building the user profile from their known preferred item's content, and returning a list of ranked items defined by the similarity scores from user's profile and item's content. Thus, this algorithm does not perform rating predictions.

3.3 Hybrid Strategies

There are many strategies for hybridization (Burke, 2002), but the few works that make use of *meta-features*, are focused only on the *Weighted* strategy. In a Weighted HF system, a final score is obtained from a numerical combination of a set of features. The purpose of such system is to define weights to be applied to the features that minimize the prediction error.

Three weighted hybridization strategies were explored, the descriptions and equations which describe how the features are constructed for each strategy are shown below. Let $\mathcal{S}$ be the set of individual recommender scores and M be the set of *meta-features* extracted from the input data:

- *Hybrid Recommender* (**HR**): does not make use of *meta-features*, the features are defined using only the Individual Algorithms' scores.

$$F_{\text{HR}} = \{s \mid \forall s \in \mathcal{S}\} \quad (2)$$

Table 2: Meta-features' mathematical definitions for Equation 1.

Acronym	Function Σ	Measure δ	Selection σ
COUNTSIZE	Average	Number of tokens	Content from element *e*
COSINE	Average	Cosine similarity	Content from elements
DICE	Average	Dice coefficient	Content from elements
ENTROPY	Average	Entropy measure	Content from element *e*
JACCARD	Average	Jaccard index	Content from elements
LOGQTDR	Logarithm	Number of ratings	Ratings from element *e*
PCR	Percentage	Number of elements in common	Elements in common
PR	Percentage	Number of ratings	Ratings from element *e*
GINI	The δ value	Gini index	Ratings from element *e* in ascending order
PEARSON	The δ value	Pearson's Coefficient of Variation	Ratings from element *e*
PQMEAN	The δ value	*pq*-mean	Ratings from element *e*
SD	The δ value	Standard Deviation	Ratings from element *e*
LOGDATER	Logarithm	Number of distinct dates	Ratings dates from element *e*
PRDATER	Percentage	Number of distinct dates	Ratings dates from element *e*

- *STacking Recommendation Engines with Additional Meta-features* (**STREAM**): defined by (Bao et al., 2009), the features are built by the union of scores and *meta-features*.

$$F_{\text{STREAM}} = \{m \mid \forall m \in \mathcal{M}\} \cup \{s \mid \forall s \in \mathcal{S}\} \quad (3)$$

- *Feature-Weighted Linear Stacking* (**FWLS**): defined by (Sill et al., 2009), the features are built by products of scores and *meta-features*. The idea is to capture the nonlinear relationship between scores and *meta-features* in the process of building features, enabling the use of linear learning methods to obtain good results consuming less time and resources, since nonlinear methods are expensive.

$$F_{\text{FWLS}} = \{m \cdot s \mid \forall m \in \mathcal{M} \wedge \forall s \in \mathcal{S}\} \quad (4)$$

The learning methods explored in this work to obtain the weights are: **(i) Linear**: Ridge Regression, Bayesian Ridge Regression, and Linear Support Vector Regression; **(ii) Nonlinear**: Random Forest and Gradient Boosted Regression Trees. These methods are available in Scikit-Learn (Pedregosa et al., 2012). For FWLS only the linear learning is applied.

Basically, the difference between all the three strategies is the building of features. Thus, the comparison of HR with STREAM and FWLS will be used to answer whether the *meta-features* are useful for HF. The choice to use both STREAM and FWLS strategies was motivated by the fact that they have never been compared before and can have distinct behaviors for the different datasets and *meta-features* combinations.

4 EXPERIMENTAL SETUP

A key issue in RS experimentation is the choice and production of the resources and in the definition of an efficient strategy to build significant results to make reliable conclusions. This section presents the resources and experimental design in four subsections. Subsection 4.1 presents the datasets. Subsection 4.2 describes the evaluation metrics. In Subsection 4.3 the scenarios experimented are defined. Finally, Subsection 4.4 describes the experimental strategy.

4.1 Datasets

The datasets used in the experiments are: Bookcrossing (Ziegler et al., 2005), Jester (Goldberg et al., 2001), and Movielens (1M) (Herlocker et al., 1999). Table 3 lists some characteristics of these datasets. From Bookcrossing we have used only the explicit ratings and the rating values were normalized in the range of $[0.0, 1.0]$ for all datasets to make the results comparable.

These datasets were selected because of their particular differences, such as application domains, number of elements, rating scales and sets of content. This aspects allows the evaluation and discussion of the results from diverse perspectives and make the conclusions more valuable.

4.2 Evaluation Metrics

We exploit a variety of evaluation criteria able to measure different aspects of recommendation including rating prediction errors and the quality of the list of recommended items (ranking measures). The ranking measures can be categorized in three classes: *relevance*, *novelty*, and *diversity*. All evaluation metrics explored are summarized in Table 4. See (Gunawardana and Shani, 2009; Baeza-Yates and Ribeiro-Neto, 1999; Vargas and Castells, 2011) for details.

To compute the evaluation metrics, we used the RankSys (Vargas and Castells, 2011). We implemented RMSE using resources supplied by this framework. To define the topics considered in a-NDCG and to calculate the items distance in EPD and EILD we used the books' language and category for Bookcrossing, the movie's genres for Movielens, and the jokes' text content for Jester. Parameter α for a-NDCG was set at 0.5. The items considered relevant were those which received normalized ratings at least 0.67 for Bookcrossing, 0.60 for Jester, and $0,75$ for Movielens. These values are around the upper 40%

Table 3: Datasets used in the experiments.

Name	Domain	Ratings (scale)	Users Items	Content
Bookcrossing	Books	433.671 (1 to 10)	77.805 185.973	Books' language, category, description, and editorial review (collected from Amazon).
Jester	Jokes	4.136.360 (-10 to 10)	73.421 100	Jokes' text content.
Movielens	Movies	1.000.209 (1 to 5)	6.040 3.706	Movies' title, genre, and plot (collected from TMDb or IMDb) and users' age, gender, and occupation.

of the possible original integer values in each rating scale (*e.g.* value 4 in the range 1 to 5). This strategy was chosen to provide the same scale over the original data for the relevant items definition.

4.3 Experimental Scenarios

Three different scenarios were defined to build the STREAM and FWLS features (Section 3.3): **(S1)** Using all *meta-features*; **(S2)** Selecting *meta-features* by computing the *Spearman* correlation value between pairs of *meta-features*, and whenever the absolute value is greater or equal to 0.8, the one that has the lower Coefficient of Variation is excluded; and **(S3)** Using only PR, due to the (Bao et al., 2009)'s assertion.

4.4 Experimental Strategy

The experimental strategy is a k-folded cross-validation procedure composed of the process shown in Algorithm 1. The first task (line 1) performs the selection of the k folds applying a stratified random sampling based on the number of users' ratings. Then, using the same sampling strategy, a percentage p of the data from each modeling data is selected (lines 2 to 5) to perform the algorithms tuning ($p = 0.05$ for Jester due to the number of ratings).

Next, the tuning of the Individual algorithms (line 6) is performed using the *irace* (López-Ibáñez et al., 2016). Irace defines parameter settings and calls a nested cross-validation (*i.e.*, within the own training set) procedure provided by the MyMediaLite. Irace performs the parameters' tuning for all selected tuning sub-samples simultaneously, selecting the set of parameter values which best fit for all sub-samples. For CB, preliminary experiments showed similar behavior for different configurations. Thus, to reduce time, we used a configuration with the Lucene's *DefaultSimilarity* and all content attributes.

Then, the tuned Individual algorithms scores and the *meta-features* are computed (lines 7 to 13). For each of the five folds, one fold is fixed to be used as *testing* in future task (line 18), and among the remaining folds (line 9), one is used for *validation* (line 10)

Algorithm 1: Experimental Strategy.

```
    Input: data, k=5, p=0.2, rs
    Output: evaluation metrics values
 1: samples ← kFoldSampling(data, k)
 2: tSamples ← {}
 3: for t in {1..k} do
 4:     model ← samples[{1..k}-{t}]
 5:     tSamples.add(SubSampling(model, p))
 6: indAlgs ← TuningIndividuals(tSamples)
 7: scores ← {}; meta ← {}
 8: for t in {1..k} do
 9:     for v in {1..k}-{t} do
10:         valid ← samples[v]
11:         train ← samples[{1..k}-{t, v}]
12:         scores.add(RunInd(indAlgs, valid, train))
13:         meta.add(CalcMetaFeatures(train))
14: metaSel ← SelectMetaFeatures(meta, rs)
15: hfAlgs ← TuningHF(tSamples, scores, metaSel)
16: scores ← {}
17: for t in {1..k} do
18:     test ← samples[t]
19:     model ← samples[{1..k}-{t}]
20:     meta ← CalcMetaFeatures(model, metaSel)
21:     sc ← RunInd(indAlgs, test, model)
22:     scores.add(sc)
23:     scores.add(RunHF(hfAlgs, test, sc, meta))
24: return CalcEvaluationMetrics(scores)
```

Table 4: Evaluation metrics. Types: (I) Rating prediction errors; (II) Relevance; (III) Novelty; (IV) Diversity.

Acronym	Type	Description
RMSE	I	*Root Mean Squared Error* measures differences between predicted and known ratings.
Precision	II	Measures the proportion of retrieved items that are relevant.
Recall	II	Measures the proportion of relevant items that are retrieved.
NDCG	II	*Normalized Discounted Cumulative Gain* measures the relevance of recommended items considering their position.
EPC	III	*Expected Popularity Complement* measures the novelty considering the popularity of the items, then it measures the ability to recommend items from the "long tail".
EPD	III	*Expected Profile Distance* is based on the notion of novelty for the target user, *i.e.* the distance of the recommended items to the items in the user profile.
EILD	IV	*Expected Intra-List Distance* measures diversity based on the distances between recommended items.
a-NDCG	IV	*Alpha-NDCG* is an adaptation of NDCG that takes into account the diversity of the recommended items, which is defined by the categorization of items in topics and in setting a degree of the importance for diversity.

and the others are used for *training* (line 11) in different combinations.

Next, the selection of *meta-features* (line 14) is accomplished for the running scenario *rs* as defined in Section 4.3. Then, the *Weighted* HF strategies parameters tuning is performed (line 15). The HF's features are built from the Individual algorithms' scores and *meta-features* processed before according to the definitions given in Section 3.3, considering only the tuning sub-samples. For this tuning process the randomized parameter optimization provided in the Scikit-Learn (Pedregosa et al., 2012) is used.

Finally, the last task consists of the computation of scores for the *testing* set and the analysis of results (lines 16 to 24). First, the selected *meta-features* are calculated for the modeling data (line 20). Next, the scores of the tuned Individual and HF algorithms are processed (lines 21 and 23). Finally, the evaluation metrics, described in Section 4.2, are calculated to perform the final analysis of the results (line 24). Such results are obtained by averaging each evaluation metric for each user in Top-K lists of recommendations (we explored varied values for K, and as expected, with the growth of K the statistical differences from results decrease. In Section 5 we present the $K = 5$ results). For comparison purposes, the Bootstrap Confidence Interval (Efron and Tibshirani, 1986), is computed with 95% of confidence.

5 EXPERIMENTAL RESULTS

In this section, the results are presented and discussed in order to provide an analysis of the solutions in the three datasets and to obtain an answer to the following research question: *are the meta-features really useful for hybridization when considering different evaluation criteria?*

The experimental results are presented in three subsections. Subsection 5.1 shows an overall analysis considering all experimented solutions. In Subsection 5.2 the analysis is based in the best ranked solutions concerning each solution category being compared. Finally, in Subsection 5.3 some discussions about the results are presented.

5.1 Results Overview

As discussed before (Section 4.1) the three datasets are very diverse, making it difficult to execute all Individual algorithms in some cases. As an example, for Bookcrossing the kNN methods were not executed due to the large number of users and items, and for Jester users' kNN and AFM methods were not executed due to the large number of users and ratings.

Figure 1 shows the heat maps of the evaluation metrics rankings for all solutions. For Bookcrossing we can observe many ties in all metrics except in RMSE and a great result for CB solution. This result can be explained by the number of ratings, users and items, *i.e.* there are many users and items and few ratings composing a much sparse scenario, damaging CF and HF, since their features are built with CF scores. However, the items' content is not scarce, favoring CB.

For Jester we can observe an opposite behavior. It is possible to identify best results for CF and HF solutions (with an apparent advantage for CF) and worse result for CB. Again, the behavior can be explained by the dataset characteristics. Jester has few items and many ratings, favoring CF. However, the content is limited only to the jokes text, damaging CB and HF, since their features are built with CB scores.

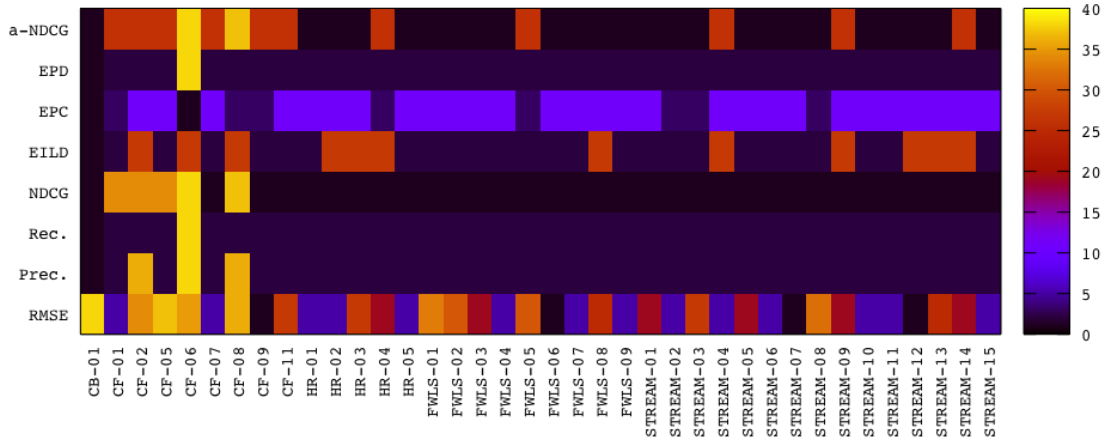
Finally, Movielens showed a different behavior from the other two datasets. HF shows the best results, despite not being better in all metrics. Again, the explanation can be done by the dataset's characteristics. Indeed, the dataset did not evidence benefits for CF or CB. The number of ratings, users, and items, and the content available seems to be balanced when compared with the other two datasets, which may favor HF.

5.2 Ranked Solutions Analysis

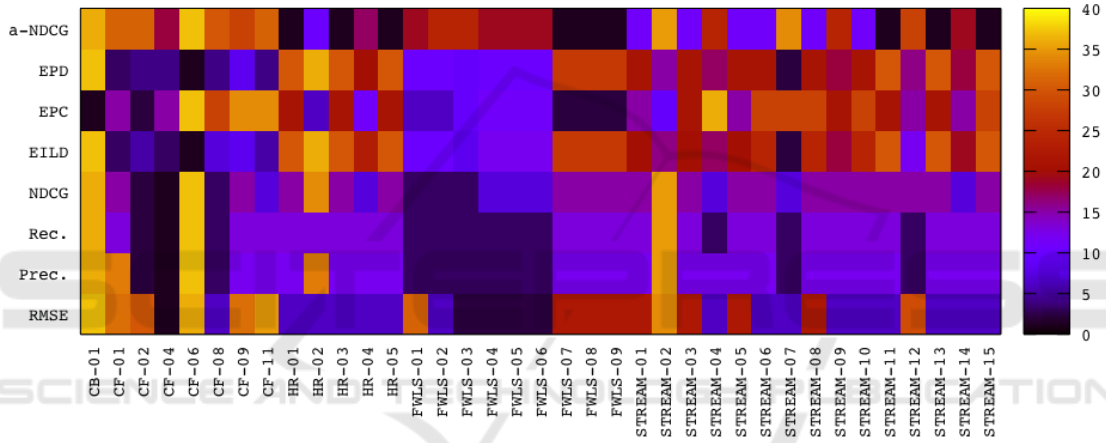
To produce a multicriteria evaluation we ranked the solutions using the sum of the Fractional rankings ("1 2.5 2.5 4") of each criterion. Table 5 shows the solutions chosen for all datasets.

For Bookcrossing, confirming the results overview, CB was best ranked, being placed first for all ranking measures. Only for NDCG and a-NDCG we can observe ties with other solutions, showing similar results when the position in the ranked list is considered. The most expressive CB results are in EPD, EILD and Precision, proving great performance for the three types of criteria (relevance, novelty, and diversity). Analyzing the other four approaches, it is observed a close contest, FWLS and HR were basically affected by the third place in EPC. Again, the dataset's characteristics explain this result, there are many users and items, few ratings, and many item's content.

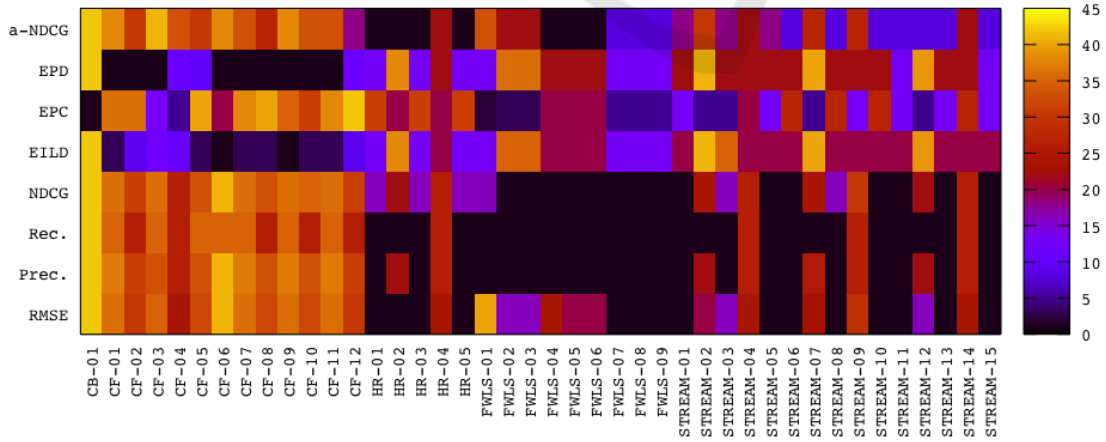
For Jester, CF was the best ranked solution but it was not the winner for all measures. CF showed better results for RMSE and relevance measures and worse results for novelty and diversity. The CF winner was



(a) Bookcrossing



(b) Jester



(c) Movielens

Figure 1: Heat maps of rankings for all solutions. We show the Standard competition raking (“1 2 2 4”) for better visualization. The x-*axis* labels shows the RS strategy and the number of the algorithm or variation.

the items' kNN. This result can be explained by the large number of ratings and low number of items, so, all items have many ratings, favoring items' kNN.

For Movielens, HF was best ranked in relation to CF and CB. FWLS was best ranked due to a more regular performance, being in the first place for RMSE and relevance measures and in the second place for novelty and diversity measures, demonstrating an intense competition.

Finally, analyzing only HF for all datasets to answer our research question, we can observe that when there are no ties, FWLS and/or STREAM are better than HR in all criteria except for a-NDCG, proving the utility of *meta-features* in hybrid RS solutions when considering different evaluation criteria. The a-NDCG metric evaluates the diversity considering the ranking position, which may indicate that the use of *meta-features* did not promote diverse items on the top of the recommendations. Considering the scenarios, S1 was best ranked for Bookcrossing and Jester, and S3 was better for Movielens, which may indicate that the *meta-feature* selection used was not useful and the (Bao et al., 2009)'s assertion is not universal. Finally, comparing FWLS and STREAM we can observe that STREAM was best ranked only for Bookcrossing, showing that FWLS really obtains good results spending less time and resources by using nonlinear methods.

5.3 Discussions

The experimental results showed distinct behaviors. For a context in which we have a sparse rating matrix and a rich content available (*i.e.* Bookcrossing)

Table 5: Best solutions from each RS strategy. For FWLS and STREAM we show the scenario in parentheses. For evaluation metrics we show the Dense raking ("1 2 2 3") of the solutions in parentheses for better visualization. The best solutions are in bold. There are no RMSE results for CB, since it did not produce rating predictions. For comparison purposes we applied a 5-folded cross-validation procedure and computed the Bootstrap Confidence Interval (Efron and Tibshirani, 1986) with 95% of confidence (Section 4.4).

Solution	RMSE	Precision	Recall	NDCG	EPC	EPD	EILD	a-NDCG
Bookcrossing								
CB-01	-	**0.2169 (1)**	**0.5930 (1)**	**0.5813 (1)**	**0,9997 (1)**	**0,5369 (1)**	**0,2153 (1)**	**0,4937 (1)**
STREAM-02(S1)	0.1551 (2)	0.1906 (2)	0.5864 (2)	**0.5825 (1)**	0,9997 (2)	0,3893 (2)	0,1594 (2)	**0,4920 (1)**
CF-07	**0.1546 (1)**	0.1900 (2)	0.5860 (2)	**0.5813 (1)**	0,9997 (2)	0,3893 (2)	0,1597 (2)	0,4898 (2)
FWLS-06(S2)	**0.1548 (1)**	0.1908 (2)	0.5866 (2)	**0.5835 (1)**	0,9997 (3)	0,3900 (2)	0,1603 (2)	**0,4935 (1)**
HR-01	0.1557 (2)	0.1904 (2)	0.5862 (2)	**0.5824 (1)**	0,9997 (3)	0,3893 (2)	0,1594 (2)	**0,4915 (1)**
Jester								
CF-04	**0.1852 (1)**	**0.5323 (1)**	**0.5920 (1)**	**0.6562 (1)**	0.3799 (4)	0.9258 (2)	0.9171 (2)	0.6326 (2)
FWLS-03(S1)	0.1906 (2)	0.5240 (2)	0.5835 (2)	0.6424 (2)	0.3818 (2)	0.9256 (3)	0.9164 (3)	0.6294 (3)
STREAM-07(S2)	0.1910 (3)	0.5236 (2)	0.5833 (2)	0.6412 (3)	0.3789 (5)	**0.9260 (1)**	**0.9179 (1)**	0.6214 (4)
HR-04	0.1909 (3)	0.5222 (3)	0.5816 (3)	0.6406 (3)	0.3805 (3)	0.9250 (4)	0.9133 (4)	**0.6340 (1)**
CB-01	-	0.4795 (4)	0.5345 (4)	0.5649 (4)	**0.4005 (1)**	0.9217 (5)	0.9032 (5)	0.5722 (5)
Movielens								
FWLS-07(S3)	**0.1799 (1)**	**0.8145 (1)**	**0.4473 (1)**	**0.8282 (1)**	0.8200 (2)	0.6921 (2)	0.6582 (2)	0.6260 (2)
STREAM-11(S3)	**0.1801 (1)**	**0.8131 (1)**	**0.4465 (1)**	**0.8267 (1)**	0.8191 (3)	0.6921 (2)	0.6579 (3)	0.6238 (2)
HR-01	**0.1802 (1)**	**0.8130 (1)**	**0.4463 (1)**	0.8265 (2)	0.8159 (4)	0.6923 (2)	0.6588 (2)	**0.6273 (1)**
CF-04	0.1897 (2)	0.7981 (2)	0.4396 (2)	0.8062 (3)	0.8198 (2)	**0.7032 (1)**	**0.6803 (1)**	0.6017 (3)
CB-01	-	0.6528 (3)	0.3859 (3)	0.6350 (4)	**0.8516 (1)**	0.6418 (3)	0.4897 (4)	0.5793 (4)

we observed advantage for CB. On the other hand, in a context in which we have a high number of ratings and poor content available (*i.e.* Jester) we observed that the best choice is to use a simple CF method. However, when the context has a more balanced number of ratings and items' content (*i.e.* Movielens), HF proved to be the best option. These results reinforce the relevance of the input data characteristics, as discussed in previous works, and the need to evaluate all possible solutions to make good choices, since HF remains not omnipotent, despite the improvements provided by *meta-features*. It is important to mention that HF can be improved with feature selection possibly changing the best choices.

Concerning the main objective of this work, which is to identify if the use of *meta-features* is really useful for hybridization when considering different evaluation criteria, we can observe that, for all datasets, the FWLS and STREAM were better ranked than the HR solution. This observation is useful to show the utility of the *meta-features* to improve the HF results for different evaluation criteria. It is worth mentioning that all hybrid methods experimented aim only to minimize the rating prediction error, without explicitly considering other criteria.

The definition of useful *meta-features* depends on the application domain and the available solutions (Bao et al., 2009). Thus, the results we found in this work are useful to draw some conclusions and to guide the application in other domains and/or resources encouraging the use of *meta-features* to improve recommendation results. We believe that the use of multi-objective methods using the *meta-features* as additional input data can produce great results, promoting intriguing research possibilities for future works.

6 CONCLUSION

In this work, we made a comprehensive analysis of the use of *meta-features* (the input data characteristics) in Hybrid Filtering Systems using different approaches and strategies over a variety of evaluation criteria. We explored Collaborative Filtering, Content-Based, and three weighted hybrid strategies. We defined a diverse set of *meta-features* used as additional input data into two of the hybrid strategies. These *meta-features* are able to capture different characteristics of the data, both in relation to the ratings provided as well as the content of the items.

Our main objective was to analyze the behavior of hybridization when using *meta-features* to improve results according to multiple evaluation criteria. In our experiments we observed that the use of *meta-features* was useful to improve some criteria without significant degradation of others as well as to highlight the importance of exploring different alternatives to achieve better results.

Many possibilities can be explored in future works. First, we can explore feature selection to improve the hybrid results. Second, we can evaluate the impact of each *meta-feature* to better understand their significance in final results. We can also explore *meta-features* extracted from individual algorithms, such as *risk sensitivity*, to further improve the hybrid results. Finally, the most relevant issue, we can explore multi-objective methods that take into account many objectives simultaneously, other than rating prediction accuracy, to obtain even more expressive results.

ACKNOWLEDGEMENTS

This work was partially funded by projects InWeb (grant MCT/CNPq 573871/2008-6), MASWeb (grant FAPEMIG/PRONEX APQ-01400-14), and TerraLab (grant CNPq 481285/2012-1), and by the authors' individual grants from CNPq and FAPEMIG.

REFERENCES

Adar, E., Teevan, J., Dumais, S. T., and Elsas, J. L. (2009). The Web Changes Everything: Understanding the Dynamics of Web Content. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 282–291, New York, NY, USA. ACM.

Adomavicius, G. and Zhang, J. (2012). Impact of Data Characteristics on Recommender Systems Performance. *ACM Trans. Manage. Inf. Syst.*, 3(1):3:1–3:17.

Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Bao, X., Bergman, L., and Thompson, R. (2009). Stacking Recommendation Engines with Additional Metafeatures. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 109–116, New York, NY, USA. ACM.

Bendersky, M., Croft, W. B., and Diao, Y. (2011). Quality-biased Ranking of Web Documents. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 95–104, New York, NY, USA. ACM.

Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*,

UAI'98, pages 43–52, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Model User-Adap Inter*, 12(4):331–370.

Burke, R. (2007). Hybrid Web Recommender Systems. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web*, number 4321 in Lecture Notes in Computer Science, pages 377–408. Springer Berlin Heidelberg.

Efron, B. and Tibshirani, R. (1986). Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. *Statist. Sci.*, 1(1):54–75.

Gantner, Z., Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2011). MyMediaLite: A Free Recommender System Library. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 305–308, New York, NY, USA. ACM.

Geng, B., Li, L., Jiao, L., Gong, M., Cai, Q., and Wu, Y. (2015). NNIA-RS: A multi-objective optimization based recommender system. *Physica A: Statistical Mechanics and its Applications*, 424:383–397.

Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Inf. Retr.*, 4(2):133–151.

Gunawardana, A. and Shani, G. (2009). A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *J. Mach. Learn. Res.*, 10:2935–2962.

Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 230–237, New York, NY, USA. ACM.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.*, 22(1):5–53.

Hurley, N. and Rickard, S. (2009). Comparing Measures of Sparsity. *IEEE Trans. Inf. Theor.*, 55(10):4723–4741.

Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender Systems: An Introduction*. Cambridge University Press, New York.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., and Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

McCandless, M., Hatcher, E., and Gospodnetic, O. (2010). *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Manning Publications Co., Greenwich, CT, USA.

McNee, S. M., Riedl, J., and Konstan, J. A. (2006). Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1097–1101, New York, NY, USA. ACM.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2012). Scikit-learn: Machine Learning in Python. *arXiv:1201.0490 [cs]*. arXiv: 1201.0490.

Ribeiro, M. T., Ziviani, N., Moura, E. S. D., Hata, I., Lacerda, A., and Veloso, A. (2014). Multiobjective Pareto-Efficient Approaches for Recommender Systems. *ACM Trans. Intell. Syst. Technol.*, 5(4):53:1–53:20.

Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to Recommender Systems Handbook. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 1–35. Springer US. DOI: 10.1007/978-0-387-85820-3_1.

Sill, J., Takacs, G., Mackey, L., and Lin, D. (2009). Feature-Weighted Linear Stacking. *arXiv:0911.0460 [cs]*. arXiv: 0911.0460.

Vargas, S. and Castells, P. (2011). Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 109–116, New York, NY, USA. ACM.

Wang, S., Gong, M., Li, H., and Yang, J. (2016). Multi-objective optimization for long tail recommendation. *Knowledge-Based Systems*, 104:145–155.

Zhang, M. and Hurley, N. (2008). Avoiding Monotony: Improving the Diversity of Recommendation Lists. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 123–130, New York, NY, USA. ACM.

Ziegler, C.-N., McNee, S. M., Konstan, J. A., and Lausen, G. (2005). Improving Recommendation Lists Through Topic Diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 22–32, New York, NY, USA. ACM.

Zuo, Y., Gong, M., Zeng, J., Ma, L., and Jiao, L. (2015). Personalized Recommendation Based on Evolutionary Multi-Objective Optimization [Research Frontier]. *IEEE Computational Intelligence Magazine*, 10(1):52–62.