The importance of being dissimilar in Recommendation

Vito Walter Anelli⋆, Tommaso Di Noia⋆,
Eugenio Di Sciascio⋆, Azzurra Ragone•, Joseph Trotta⋆
⋆Polytechnic University of Bari
Bari - Italy
firstname.lastname@poliba.it
•Independent Researcher
azzurra.ragone@gmail.com

ABSTRACT

In recommendation scenarios, similarity measures play a fundamental role in memory-based nearest neighbors approaches. In fact, they recommend items to a user based on the similarity of either items or users in a neighborhood. In this paper, we argue that similarity between users or items, although it keeps leading importance in computing recommendations, should be paired with a value of dissimilarity (computed not just as the complement of the similarity one). We formally modeled and injected this notion in some of the most used similarity measures and evaluated our approach in a recommendation scenario showing its effectiveness with respect to accuracy and diversity results on three different datasets.

CCS CONCEPTS

• **Information systems → Similarity measures**; **Recommender systems**; *Top-k retrieval in databases*;

KEYWORDS

Dissimilarity, Asymmetric similarity, Collaborative Filtering

ACM Reference Format:
Vito Walter Anelli⋆, Tommaso Di Noia⋆, Eugenio Di Sciascio⋆, Azzurra Ragone•, Joseph Trotta⋆. 2019. The importance of being dissimilar in Recommendation. In *The 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19), April 8–12, 2019, Limassol, Cyprus.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3297280.3297360

1 INTRODUCTION

Neighborhood-based approaches have been the first family of algorithms developed for collaborative filtering recommender systems. They identify similar users or items, and they provide users with a list of items they could be interested in by exploiting the degree of similarity. Though there have been around for many years, it has been shown that neighborhood-based approaches may perform better than latent model-based methods to solve the *top-N* recommendation problem [3, 12, 16, 20]. In the *top-N* recommendation task, the focus is on providing an accurate ranked list rather than minimizing the rating prediction error. Among neighborhood-based methods the best-known are user-kNN, item-kNN and Sparse LInear Methods (SLIM) [20]. User-based and item-based schemes have proven to be effective in different settings although they use the same logic behind the scenes. In details item-kNN and SLIM (which uses an item-based scheme) have shown to outperform user-kNN to solve the *top-N* recommendation problem [10] and several algorithms have been proposed in the literature to enhance neighbors models like GLSLIM [10] and Weighted kNN-GRU4REC [15], taking advantage of personalized models and recurrent neural networks. Moreover, we also have approaches focusing on the injection of time in neighborhood models [6] and modeling similarities by directly optimizing the pair-wise preferences error [23]. All in all, under the hood, what makes neighborhood-based methods work is a similarity measure.

Several similarity measures have been proposed and used extensively such as Jaccard [13][22] and Tanimoto [11] coefficients, Cosine Vector similarity [1, 4, 7], Pearson Correlation [14], Constrained Pearson correlation [25], Adjusted Cosine similarity [24], Mean Squared Difference similarity [25], Spearman Rank Correlation [18], Frequency-Weighted Pearson Correlation [8], Target item weighted Pearson Correlation [5]. In the vast majority of cases, all these similarity measures are based on two assumptions:

(1) the correlation between i and j is the same correlation between j and i (symmetry of similarity);
(2) the correlation between two entities only captures how much they are similar to each other without taking into account their degree of dissimilarity.

To the best of our knowledge, a few works have been proposed in the past years related to asymmetric similarity, and they are mainly designed for a user-based scheme. Dissimilarity was first suggested in 1999 [28] when Varian described the value of introducing diversity into search results. An Asymmetric User Similarity has been proposed in [19] where the authors underline that a similarity measure should distinguish between a user with a rich profile and a cold user. Thus, given two users u and v, they slightly modify the Jaccard index in order to consider exclusively the number of ratings of the current user (instead of the overall number of both users). In HYBRTyco [17] the similarity proposed by Millan [19] is combined with the Sørensen index [26]. HYBRTyco [17] is a hybrid recommender system which combines matrix factorization with an asymmetric similarity model to realize a typicality-based collaborative filtering recommender system. The same approach is exploited in another asymmetric user similarity model [21] to feed a user-user similarity

SAC '19, April 8–12, 2019, Limassol, Cyprus

ACM ISBN 978-1-4503-5933-7/19/04…$15.00
https://doi.org/10.1145/3297280.3297360

matrix that is then completed using a matrix factorization algorithm. Additionally, both of them provide an extension for the latter similarity measure based on explicit numerical feedbacks (ratings). Despite previous works are focused on the user-based scheme, we already underlined that item-kNN shows excellent performance in *top-N* recommendation task. Moreover, when the number of users exceeds the number of items, as in most of the cases, item-based recommendation approaches require much less memory and time to compute the similarity weights than user-based ones, making them more scalable. Due to these reasons, both approaches have been considered in this work.

In this work, we investigate the effect on recommendation accuracy when we go beyond the above two assumptions and define (and include) the concepts of dissimilarity and asymmetry in similarity measures. In our proposal, we start from a probabilistic interpretation of similarity to define symmetric and asymmetric dissimilarities. The dissimilarity measures are then combined with traditional similarity values using additive and multiplicative strategies. The experimental evaluation shows that our approach outperforms the non-dissimilarity-aware counterparts improving the accuracy of results or diversity or both.

The rest of the paper is organized as follows: Section 2 presents the motivation behind our work and the proposed approach. Section 3 presents the evaluation protocol, metrics, datasets and performance of the method. Finally, in Section 4 concluding considerations are provided.

2 DISSIMILARITY IN RECOMMENDATION

2.1 Motivation

The main idea behind our proposal is that symmetric similarity may not be sufficient to capture subtle interactions between items. We assume that representing the similarity through traditional measures can lead to imperfect results as important information might not be properly considered. Let us consider some examples in an item-kNN scenario. Suppose we are dealing with a dataset containing rating data from the book domain on the following books:

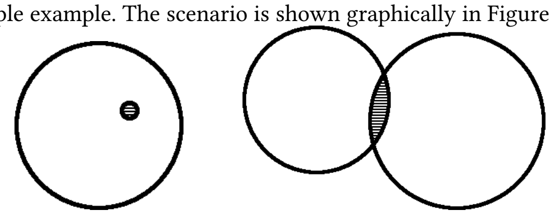
Title	Short name	Author	# Votes
A Game of Thrones	GoT	G. R. R. Martin	100
A Dance with Dragons	DwD	G. R. R. Martin	10
Shroud of Eternity	SoE	T. Goodkind	120

By looking at the previous data, we see that both *A Game of Thrones* and *A Dance with Dragons* belong to the same saga *A song of ice and fire* and they are, respectively, the first and the fifth volume. *Shroud of Eternity* is the second volume of Nicci Chronicles' saga. We may assume that all the users who rated DwD also rated GoT, i.e., $U_{DwD} \subseteq U_{GoT}$. Analogously, since the topic of the book is mostly the same, we may assume that a number of readers of SoE also voted GoT, $U_{SoE} \cap U_{GoT} \neq \emptyset$. Suppose now that we have $U_{SoE} \cap U_{GoT} = 20$ and $U_{DwD} \cap U_{GoT} = 10$. If we compute the Jaccard similarity between the pairs SoE, GoT and DwD, GoT we have

$$JS(DwD, GoT) = \frac{|U_{DwD} \cap U_{GoT}|}{|U_{DwD} \cup U_{GoT}|} = 0.1$$

$$JS(SoE, GoT) = \frac{|U_{SoE} \cap U_{GoT}|}{|U_{SoE} \cup U_{GoT}|} = 0.1$$

In our opinion, much relevant information has been lost in this simple example. The scenario is shown graphically in Figure 1.



(a) U_{DwD} is entirely contained in U_{GoT}

(b) U_{GoT}, is partially overlapped with U_{SoE}

Figure 1: Representation of the motivation example

It is clear that U_{DwD} is a proper subset of U_{GoT} and, on the contrary, there are many users in U_{SoE} that have not experienced GoT. This information is mostly lost in the computation of the similarity values even though a piece of this information is retained in the denominator of the Jaccard coefficient through the overall value of $U_{GoT} \cup U_{DwD}$ and $U_{GoT} \cup U_{SoE}$.

In order to clarify the reason why we do not consider this remaining information sufficient, let us consider a more formal description of the scenario. Many similarity measures (like Jaccard in the previous example) mainly rely on a value that denotes the similarity between two items normalized by their overall weight. This can be represented as a probability. Let $\mathcal{I}(u) = \{\langle i, r_{ui}\rangle \mid u \text{ rated } i \text{ with } r_{ui}\}$ be the user profile containing the pairs item-rating and $\mathcal{U}(i) = \{\langle u, r_{ui}\rangle \mid \langle i, r_{ui}\rangle \in \mathcal{I}(u)\}$ be the set of users that experienced i. If we consider Jaccard similarity we see it represents the following probability:

$$p^{JS}(i, j) = p(\langle u, r_{ui}\rangle \in \mathcal{U}(i) \wedge \langle u, r_{uj}\rangle \in \mathcal{U}(j)) \tag{1}$$

which can be read as the probability that u experienced both i and j. Under a probabilistic lens, we may define dissimilarity measures with two different probabilities:

- the probability that a generic user u experienced the item i but never experienced the item j
- the probability that a generic user u experienced the item j but never experienced the item i

Since we are introducing an asymmetric behavior in computing the similarity between i and i as well as between j and i, we see that the two probabilities have a different role while computing a similarity measure.

In a classical memory-based Item-kNN, $sim(i, j)$ is used to compare i to different j to find out which js are the most similar to i. In practical terms, we are interested in how much j is similar to i. If we focus on Figure 1a, we realize that DwD is more similar to GoT than the opposite situation. The reason behind this behavior is not directly related to the size of the involved sets but it depends on the probability that a user who experienced GoT did not experience DwD.

Though some interesting asymmetric similarities have been proposed in the last years [17, 21], to our knowledge, no one focused on this probability that represents a negative asymmetric dissimilarity.

2.2 Metrics

In this work, we propose a general asymmetric similarity model in which items i, j similarities are computed by taking into account the probability that users that experienced j never experienced i. The idea behind our work is preserving the core meaning of a specific similarity, applying a corrective factor encoding the dissimilarity we mentioned before.

We introduced this correction into two binary symmetric similarities: Jaccard and Sørensen index, and in the asymmetric variant to the Jaccard coefficient proposed in [19]. We tested this correction both as an additive and as a multiplicative factor.

For the sake of completeness, we reintroduce *Jaccard coefficient similarity (JS)* that, for a memory-based item-kNN model can be expressed as:

$$JS(i,j) = \frac{|\mathcal{U}(i) \cap \mathcal{U}(j)|}{|\mathcal{U}(i) \cup \mathcal{U}(j)|} \tag{2}$$

The probability that users who experienced j never experienced i can be modeled as the complementary probability of Equation (1) w.r.t. $\mathcal{U}(i)$ over $|\mathcal{U}(i) \cup \mathcal{U}(j)|$. This probability, we name *Jaccard Asymmetric Dissimilarity*(JAD) can be formulated as follows:

$$JAD(i,j) = \frac{|\mathcal{U}(j)| - |\mathcal{U}(i) \cap \mathcal{U}(j)|}{|\mathcal{U}(i) \cup \mathcal{U}(j)|} \tag{3}$$

Again, Equation (3) can be seen in terms of probability as

$$p^{JAD}(i,j) = p(\langle u, r_{uj} \rangle \in \mathcal{U}(j)) - p^{JS}(i,j)$$

We now propose to modify the original similarity by injecting the former negative correction weighted with a parameter λ that can be easily customized. The overall similarity, *Additive Adjusted Jaccard (AAJ)*, is then formulated as:

$$AAJ(i,j) = JS(i,j) - \lambda \cdot JAD(i,j) \tag{4}$$

where λ is a parameter that could depend on many factors such as the number of users and items in the dataset and the intensity of interactions between them.

Recalling that the overall formula should represent a degree of similarity between the two different items we defined the Multiplicative Adjusted Jaccard as the product of Jaccard similarity with the inverse of Jaccard Asymmetric Dissimilarity (IJAD). In other words, we use $\frac{1}{p^{JAD}(i,j)}$ as a corrective factor for $p^{JS}(i,j)$. We then define the *Multiplicative Adjusted Jaccard (MAJ)* as:

$$MAJ(i,j) = \frac{JS(i,j)}{JAD(i,j)} = JS(i,j) \cdot IJAD(i,j) \tag{5}$$

In the multiplicative variant, in order to avoid division by zero the minimum value for JAD is set to $\frac{1}{|\mathcal{U}(i) \cup \mathcal{U}(j)|}$. As mentioned in Section 2.1 a symmetric variant of Jaccard coefficient (named *Jaccard Symmetric Dissimilarity (JSD)*) can be used composing Equation (3) with the probability that a user u experienced the item i but never experienced the item j:

$$\begin{aligned} JSD(i,j) &= \frac{(|\mathcal{U}(j)| - |\mathcal{U}(i) \cap \mathcal{U}(j)|) + (|\mathcal{U}(i)| - |\mathcal{U}(i) \cap \mathcal{U}(j)|)}{|\mathcal{U}(i) \cup \mathcal{U}(j)|} \\ &= \frac{|\mathcal{U}(i)| + |\mathcal{U}(j)| - 2 \cdot |\mathcal{U}(i) \cap \mathcal{U}(j)|}{|\mathcal{U}(i) \cup \mathcal{U}(j)|} \end{aligned}$$

leading to the corresponding probability

$$p^{JSD}(i,j) = p^{JAD}(i,j) + p^{JAD}(j,i)$$

Thus the *Symmetric Additive Adjusted Jaccard (S-AAJ)* and *Symmetric Multiplicative Adjusted Jaccard (S-MAJ)* can be defined as follows:

$$S\text{-}AAJ(i,j) = JS(i,j) - \lambda \cdot JSD(i,j)$$

$$S\text{-}MAJ(i,j) = \frac{JS(i,j)}{JSD(i,j)}$$

In order to test our idea, we applied all the variants previously introduced for Jaccard similarity to two popular similarity measures: *Asymmetric Jaccard Similarity (AJS)* and *Sørensen coefficient (SOR)*. All the derived variants are represented, respectively, in Table 1 and Table 2.

Table 1: Asymmetric Jaccard considered variants.

Short name	Extended	Formula
AJS(i,j)	Asymm. Jaccard Similarity	$\frac{\lvert\mathcal{U}(i) \cap \mathcal{U}(j)\rvert}{\lvert\mathcal{U}(i)\rvert}$
AJD(i,j)	Asymm. Jaccard Dissimilarity	$\frac{\lvert\mathcal{U}(j)\rvert - \lvert\mathcal{U}(i) \cap \mathcal{U}(j)\rvert}{\lvert\mathcal{U}(i)\rvert}$
AAAJ(i,j)	Additive Adjusted Asymm. Jaccard	$AJS(i,j) - \lambda \cdot AJD(i,j)$
MAAJ(i,j)	Multiplicative Adjus. Asymm. Jaccard	$AJS(i,j) \cdot IJAD(i,j)$
S-AAAJ(i,j)	Symmetric AAAJ	$AJS(i,j) - \lambda \cdot \frac{\lvert\mathcal{U}(i)\rvert + \lvert\mathcal{U}(j)\rvert - 2 \cdot \lvert\mathcal{U}(i) \cap \mathcal{U}(j)\rvert}{\lvert\mathcal{U}(i)\rvert}$
S-MAAJ(i,j)	Symmetric MAAJ	$AJS(i,j) \cdot IJSD(i,j)$

Table 2: Sørensen similarity considered variants.

Short name	Extended	Formula
SOR(i,j)	Sørensen Similarity	$\frac{\lvert\mathcal{U}(i) \cap \mathcal{U}(j)\rvert}{\lvert\mathcal{U}(i)\rvert + \lvert\mathcal{U}(j)\rvert}$
ASD(i,j)	Asymm. Sørensen Dissimilarity	$\frac{\lvert\mathcal{U}(j)\rvert - \lvert\mathcal{U}(i) \cap \mathcal{U}(j)\rvert}{\lvert\mathcal{U}(i)\rvert + \lvert\mathcal{U}(j)\rvert}$
AAS(i,j)	Additive Adjusted Assym Sørensen	$SOR(i,j) - \lambda ASD(i,j)$
MAS(i,j)	Multiplicative Adjusted Asymm Sørensen	$SOR(i,j) \cdot IJAD(i,j)$
S-AAS(i,j)	Symmetric AAS	$SOR(i,j) - \lambda \frac{\lvert\mathcal{U}(i)\rvert + \lvert\mathcal{U}(j)\rvert - 2 \cdot \lvert\mathcal{U}(i) \cap \mathcal{U}(j)\rvert}{\lvert\mathcal{U}(i)\rvert + \lvert\mathcal{U}(j)\rvert}$
S-MAS(i,j)	Symmetric MAS	$SOR(i,j) \cdot IJSD(i,j)$

All the above metrics have been introduced having in mind an item-kNN approach but, without loss of generality, they can be applied to user-kNN model as well.

3 EXPERIMENTAL EVALUATION

The experimental evaluation has been carried out on three publicly available datasets and with different values of k and λ.

Datasets. We evaluated the effectiveness of our approach on the three datasets shown in Table 3 belonging to different domains (Music, Books, and Movies). The `Last.fm` dataset [9] corresponds to transactions with Last.fm online music system released in HetRec 2011[1]. It contains social networking, tagging, and music artist listening information from a set of 2K users. `LibraryThing` represents books ratings collected in the LibraryThing community website. It contains social networking, tagging, and rating information on a [1..10] scale. `Yahoo!Movies` (Yahoo! Webscope dataset ydata-ymovies-user-movie-ratings-content-v1_0)[2] contains movies

[1] http://ir.ii.uam.es/hetrec2011/
[2] http://research.yahoo.com/Academic_Relations

ratings generated by Yahoo! Movies up to November 2003. It provides content, demographic, rating information, and mappings to MovieLens and EachMovie datasets.

Table 3: Datasets statistics.

Dataset	#Users	#Items	#Transactions	Sparsity
Yahoo! Movies	7642	11,916	221,367	99.76%
LibraryThing	7279	37,232	2,056,487	99.24%
Last FM	1850	11,247	59,071	99.72%

Columns corresponding to #Users, #Items and #Transactions show the number of users, number of items and number of transactions, respectively, in each dataset. The last column shows the sparsity of the dataset.

Evaluation Protocol and Experimental Setting with Parameters tuning. The evaluation protocol we adopted in our experiments is *all unrated items* [27]. With this protocol, the recommendation list is computed from a candidate list given by the cartesian product between users and items minus the items each user experimented in the training set. We performed a temporal 64-16-20 hold-out split (when temporal information is available) retaining the last 20% of ratings as test set and 16% as validation set. We measured the performance by computing Precision@N (*Prec@N*) for *top-N* recommendation lists as accuracy metric. Precision has been computed on a per-user basis, and the returned results have been averaged. As Precision needs relevant items to be computed, we set the relevance threshold to 8 over 10 for LibraryThing and Yahoo!Movies, and to 0 for Last.fm since in this latter no ratings are provided but the number of user-item transactions. We measured Diversity through *catalog coverage* (aggregate diversity in *top-N* list). The *catalog coverage*, also called diversity-in-top-N (*D@N*) [2], is measured by computing the overall number of different items recommended within the complete recommendation list. It represents the propensity of a system to recommend always the same items.

Baselines. We compared our approaches in both User-kNN and Item-kNN settings. The former finds the k-nearest user neighbors based on a similarity function and then exploits them to predict a score for each user-item pair. The latter is the item-based version of the k-nearest neighbors algorithm that uses the k-nearest items to compute the predictions. For both schemes we used the validation set to find the optimal hyper-parameters. However, we are not interested in the algorithm itself but on the similarity measures that are used to compute neighbors and predictions. As baseline to compare with, we used both symmetric and asymmetric measures, namely, ***Jaccard (JS)*** and ***Sorensen (SOR)*** (for symmetric measures) and ***asymmetric Jaccard (AJS)*** and ***asymmetric Jaccard weighted with the Sorensen Index (ASOR)*** [17] (for asymmetric measures). For all the similarities that make use of λ we evaluated them varying λ in $\{0.2, 0.4, 0.6, 0.8\}$ whereas we considered a number of $Neighbors$ varying in $\{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. We ran the algorithms with all possible combinations, and we selected the best performing ones with respect to Precision@N. The best parameters for user-based and item-based schemes are represented, respectively in Table 4 and 5.

Performance of the proposed methods. Results in Table 6 show the performance of all algorithms with a user-based scheme. Concerning accuracy, it is clear that both asymmetric and symmetric multiplicative variants are the best-performing ones. MAJ and S-MAJ achieve good performance, outperforming JS. The same trend is shown between MAAJ, S-MAAJ, and AJS. The same behavior can be observed in the Sorensen algorithms block, in which MAS outperforms SOR in LibraryThing and Yahoo!Movies datasets with the only exception of Last.fm datasets. Concerning Diversity the proposed variants constantly outperform the base variants. It is worth to note that AAJ and AAAJ algorithms that can recommend much more items with a little loss of Precision. Table 7 shows the results for the item-based scheme. Performance is much different here, and we can note that the multiplicative asymmetric variants have a worse behavior. However, the Last.fm results show that the additive variants outperform the base ones. We can observe the same behavior also in the whole asymmetric Jaccard similarity block for all the three datasets. The Jaccard similarity and the Sørensen block, for LibraryThing and Yahoo!Movies show no clear champion concerning the performance of JS/SOR and S-MAJ/S-MAS that result very similar. This behavior may be due to the tuning results that are very close to each other, and this probably prevented us from selecting the best parameters.

Experimental Setting with a fixed number of neighbors. In tables 4 and 5 we see that best values of λ and k are different depending on the adopted approach. Hence, we tested also the different algorithms with a fixed number of neighbors. In other words, we checked: How would the different algorithms perform if, i.e., we fix the number of neighbors? Once again we employed the *all unrated items* evaluation protocol to evaluate the methods. We performed a temporal 80-20 hold-out split retaining the last 20% of ratings as test set using temporal information when available.

Baselines. Also in this experiment, we compared our approaches with both User-kNN and Item-kNN settings considering, for all the algorithms, the number of neighbors fixed and set to $k = 80$. For all the similarities that make use of λ we evaluated them varying λ in 0.2,0.4,0.6,0.8. In Tables 8 and 9 we show the best results we obtained[3]. The best values regarding Precision and Aggregate Diversity are highlighted in bold. We computed significance tests for precision results, and we found they are statistically significant at the 0.05 level w.r.t. their respective baselines.

Performance of the proposed methods. Results in Table 8 show that our approach always outperforms baseline variants in the User-kNN scheme. In details **additive** asymmetric similarity and **multiplicative** asymmetric similarity significantly perform better than JS, SOR and ASOR for all three dataset. Among these two variants of similarity, the multiplicative variant is the best-performing one. Quite interestingly, modifying AJS, which is asymmetric in its inner nature with our asymmetric dissimilarity factor leads to an improvement irrespective of the considered dataset. It is worth noticing that, other than the accuracy improvements, aggregate diversity also increases due to the dissimilarity injection. In details, the asymmetric additive variant achieves the best results and triples catalog coverage values for LibraryThing and Yahoo!Movies.

Table 9 shows Precision and Catalog Coverage results for an item-based scheme. Obtained results are quite interesting for many reasons. First of all, it is clear that the same similarities can lead to

[3]The complete results are publicly available at **https://github.com/sisinflab/The-importance-of-being-dissimilar-in-Recommendation**.

Table 4: Best parameters for User-kNN scheme

Precision - P@10																								
	JS	AAJ		MAJ	S-AAJ		S-MAJ	ASOR	SOR	AAS		MAS	S-AAS		S-MAS	AJS	AAAJ		MAAJ	S-AAAJ		S-MAAJ		
Datasets	**k**	λ	**k**	**k**	λ	**k**	**k**	**k**	**k**	λ	**k**	**k**	λ	**k**	**k**	**k**	λ	**k**	**k**	λ	**k**	**P@10**		
LibraryThing	50	0.2	100	50	0.2	30	100	90	50	0.2	50	50	0.2	30	100	20	0.2	100	40	0.2	50	20		
Yahoo	90	0.4	100	40	0.2	100	90	100	90	0.2	40	40	0.2	100	90	10	0.6	80	30	0.2	100	70		
Last FM	90	0.2	100	100	0.2	50	90	90	100	0.2	100	100	0.2	50	90	30	0.2	100	100	0.2	50	20		

Table 5: Best parameters for Item-kNN scheme

Precision - P@10																						
	JS	AAJ		MAJ	S-AAJ		S-MAJ	ASOR	SOR	AAS		MAS	S-AAS		S-MAS	AJS	AAAJ		MAAJ	S-AAAJ		S-MAAJ
Datasets	**k**	λ	**k**	**k**	λ	**k**	**k**	**k**	**k**	λ	**k**	**k**	λ	**k**	**k**	**k**	λ	**k**	**k**	λ	**k**	**P@10**
LibraryThing	10	0.2	20	20	0.4	10	20	70	10	0.2	20	20	0.4	10	20	60	0.2	10	10	0.2	10	60
Yahoo	10	0.2	20	10	0.2	10	10	10	10	0.2	10	10	0.2	10	10	20	0.2	20	10	0.2	10	10
Last FM	10	0.2	20	30	0.2	20	10	10	10	0.2	30	30	0.2	20	10	10	0.2	40	20	0.2	70	10

Table 6: Comparison in terms of Precision and Aggregate Diversity for User-kNN scheme with best parameters

Precision - P@10																
Datasets	**JS**	**AAJ**	**MAJ**	**S-AAJ**	**S-MAJ**	**ASOR**	**SOR**	**AAS**	**MAS**	**S-AAS**	**S-MAS**	**AJS**	**AAAJ**	**MAAJ**	**S-AAAJ**	**S-MAAJ**
LibraryThing	0.03025	0.04005	**0.04675**	0.00416	0.03102	0.03565	0.03031	0.04687	**0.04687**	0.00416	0.03088	0.03768	0.03871	**0.04670**	0.00429	0.03796
Yahoo	0.04165	0.05196	**0.06284**	0.03892	0.04149	0.04261	0.04151	0.06354	**0.06354**	0.03917	0.04170	0.05465	0.04234	**0.06176**	0.03705	0.05207
Last FM	0.02704	0.02163	0.02532	0.00601	**0.02773**	**0.02927**	**0.02747**	0.02532	0.02532	0.00609	0.02695	0.02901	0.02120	0.03039	0.00592	**0.03107**

Aggregate Diversity - D@10																
Datasets	**JS**	**AAJ**	**MAJ**	**S-AAJ**	**S-MAJ**	**ASOR**	**SOR**	**AAS**	**MAS**	**S-AAS**	**S-MAS**	**AJS**	**AAAJ**	**MAAJ**	**S-AAAJ**	**S-MAAJ**
LibraryThing	2735	**7330**	3093	3341	1953	2687	2660	2999	2999	**3337**	1879	2519	**7443**	3037	3337	2834
Yahoo	698	2087	1126	**2181**	796	882	662	1074	1074	**2179**	732	974	1903	1147	**2177**	580
Last FM	1136	**1665**	1485	1156	1203	1283	1097	1446	**1446**	1154	1145	764	**1748**	889	1163	1025

Table 7: Comparison in terms of Precision and Aggregate Diversity for Item-kNN scheme with best parameters

Precision - P@10																
Datasets	**JS**	**AAJ**	**MAJ**	**S-AAJ**	**S-MAJ**	**ASOR**	**SOR**	**AAS**	**MAS**	**S-AAS**	**S-MAS**	**AJS**	**AAAJ**	**MAAJ**	**S-AAAJ**	**S-MAAJ**
LibraryThing	**0.08180**	0.07449	0.03189	0.06482	0.08048	0.07835	**0.07949**	0.02462	0.02462	0.06111	0.07894	0.04556	**0.08748**	0.02256	0.06293	0.06097
Yahoo	0.05105	0.04976	0.00272	0.04966	**0.05187**	0.05013	0.05043	0.00196	0.00196	0.04915	**0.05141**	0.01598	**0.05031**	0.00175	0.04842	0.02437
Last FM	0.02146	**0.02549**	0.00326	0.02489	0.02120	0.02052	0.02069	0.00275	0.00275	**0.02403**	0.02077	0.00876	**0.02910**	0.00240	0.02506	0.01373

Aggregate Diversity - D@10																
Datasets	**JS**	**AAJ**	**MAJ**	**S-AAJ**	**S-MAJ**	**ASOR**	**SOR**	**AAS**	**MAS**	**S-AAS**	**S-MAS**	**AJS**	**AAAJ**	**MAAJ**	**S-AAAJ**	**S-MAAJ**
LibraryThing	11945	12338	**22097**	11654	11551	10521	11774	21365	**21365**	11447	11450	**18604**	12899	13042	13604	16018
Yahoo	3262	**5159**	4644	3466	3091	3539	3340	4742	**4742**	4351	3725	**6864**	3946	3645	4514	5565
Last FM	2867	**4394**	3884	2920	2763	3389	2917	3648	**3654**	3446	3341	**5150**	3680	3078	3429	4447

very different results depending on the adopted scheme. In particular, asymmetric Jaccard (AJS) performs very badly for the item-kNN algorithm. Under the dissimilarities perspective, we have the same behavior, and the multiplicative approach performs badly. Quite surprisingly, the additive version can always outperform the base variants. This suggests that adopting an additive strategy for item-kNN may lead to better results. This may be due to the wide number of items pairs without any common user. Focusing on additive symmetric and asymmetric similarities we can note that aggregate diversity results reflect the same improvements observed in accuracy values. The only case that appears to behave differently is AAAJ that registered a catalog coverage lower than AJS. This happens as we considered the best performing λ for precision. In case of $\lambda \in \{0.4, 0.6\}$ we obtain aggregate diversity values of $16,205$ and $18,071$, respectively, with precision results constantly higher than AJS (0.09374 and 0.08820). We may observe another interesting pattern on the `Yahoo!Movies` row: the symmetric version outperforms the asymmetric one. This could be due to some datasetś characteristic. By looking at the data in Table 3 we see that the ratio of the number of items to the number of users is much higher in `LibraryThing` and `Last.fm` (≈ 5 and ≈ 6) with respect to `Yahoo!Movies` (≈ 1.5). This suggests that the more the ratio is, the more is convenient to adopt an asymmetric scheme. However, this consideration needs to be further investigated.

4 CONCLUSION AND FUTURE WORK

In this work, we propose a method to improve the performance of neighborhood-based models, by capturing subtle interactions between users and items, which cannot be appreciated using a traditional similarity measure. We defined a dissimilarity measure, that can be used combined with traditional user-based and item-based schemes. The proposed approach takes into account the single asymmetric components, leading to an improvement in both precision and aggregate diversity results. We performed a comparative experimental evaluation using three well-known datasets, varying the tuning parameter λ and k. Experiments show that our approach outperforms competing algorithms, denoting the usefulness of incorporating symmetric and asymmetric dissimilarity in neighborhood-based models. We are currently working on an extension of our idea that takes into account also user ratings and not just set-based measures. As a further extension, we are also interested in making the approach even more personalized by weighting dissimilarity with user-centered values of λ.

REFERENCES

[1] S Abiteboul, P Buneman, D Suciu, J Allan, R Papka, V Lavrenko, A Blum, T Mitchell, A Bonifati, S Ceri, et al. 1995. News Weeder: Learning to Filter Netnews. In *Proceedings of the 12th International Conference of Machine Learning (ICMLâĂŽ95)*.

[2] Gediminas Adomavicius and YoungOk Kwon. 2012. Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques. *IEEE Trans. Knowl. Data Eng.* 24, 5 (2012), 896–911.

[3] Fabio Aiolli. 2013. A Preliminary Study on a Recommender System for the Million Songs Dataset Challenge. In *Proceedings of the 4th Italian Information Retrieval*

Table 8: Comparison in terms of Precision and Aggregate Diversity for User-kNN scheme

Precision - P@10

	JS	AAJ		MAJ	S-AAJ		S-MAJ	ASOR	SOR	AAS		MAS	S-AAS		S-MAS	AJS	AAAJ		MAAJ	S-AAAJ		S-MAAJ
Datasets	**P@10**	λ	**P@10**	**P@10**	λ	**P@10**	**P@10**	**P@10**	**P@10**	λ	**P@10**	**P@10**	λ	**P@10**	**P@10**	**P@10**	λ	**P@10**	**P@10**	λ	**P@10**	**P@10**
LibraryThing	0.0363	0.2	0.0582	**0.0627**	0.4	0.0364	0.0010	0.0394	0.0363	0.2	0.0586	**0.0629**	0.2	0.0405	0.0364	0.0364	0.2	0.0558	**0.0603**	0.2	0.0057	0.0375
Yahoo	0.0437	0.4	0.0561	**0.0676**	0.2	0.0403	0.0433	0.0442	0.0438	0.4	0.0573	**0.0687**	0.2	0.0568	0.0435	0.0607	0.6	0.0529	**0.0664**	0.2	0.0378	0.0529
Last FM	0.0164	0.4	0.0242	**0.0257**	0.2	0.0037	0.0160	0.0215	0.0166	0.4	0.0241	0.0253	0.2	**0.0275**	0.0164	0.0248	0.2	0.0228	**0.0323**	0.2	0.0037	0.0242

Aggregate Diversity - D@10

	JS	AAJ		MAJ	S-AAJ		S-MAJ	ASOR	SOR	AAS		MAS	S-AAS		S-MAS	AJS	AAAJ		MAAJ	S-AAAJ		S-MAAJ
Datasets	**D@10**	λ	**D@10**	**D@10**	λ	**D@10**	**D@10**	**D@10**	**D@10**	λ	**D@10**	**D@10**	λ	**D@10**	**D@10**	**D@10**	λ	**D@10**	**D@10**	λ	**D@10**	**D@10**
LibraryThing	2136	0.2	**7367**	2406	0.4	2246	873	2819	2083	0.2	**7330**	2292	0.2	5528	2171	1299	0.2	**7501**	2060	0.2	3331	1486
Yahoo	734	0.4	2070	835	0.2	**2187**	825	936	695	0.4	**2048**	786	0.2	1114	784	451	0.6	1773	717	0.2	**2175**	587
Last FM	1449	0.4	1345	**1654**	0.2	1203	1460	1433	1324	0.4	1324	**1625**	0.2	1410	1420	774	0.2	**1665**	1077	0.2	1207	941

Table 9: Comparison in terms of Precision and Aggregate Diversity for Item-kNN scheme

Precision - P@10

	JS	AAJ		MAJ	S-AAJ		S-MAJ	ASOR	SOR	AAS		MAS	S-AAS		S-MAS	AJS	AAAJ		MAAJ	S-AAAJ		S-MAAJ
Datasets	**P@10**	λ	**P@10**	**P@10**	λ	**P@10**	**P@10**	**P@10**	**P@10**	λ	**P@10**	**P@10**	λ	**P@10**	**P@10**	**P@10**	λ	**P@10**	**P@10**	λ	**P@10**	**P@10**
LibraryThing	0.0869	0.2	**0.0949**	0.0451	0.4	0.0822	0.0944	**0.1019**	0.0815	0.2	**0.0914**	0.0374	0.4	0.0826	0.0901	0.0598	0.2	**0.1011**	0.0303	0.4	0.0830	0.0792
Yahoo	0.0331	0.2	0.0510	0.0016	0.2	**0.0535**	0.0373	0.0447	0.0297	0.2	0.0504	0.0014	0.2	**0.0531**	0.0352	0.0046	0.4	0.0509	0.0012	0.2	**0.0527**	0.0083
Last FM	0.0141	0.2	**0.0248**	0.0036	0.2	0.0230	0.0158	0.0127	0.0124	0.4	0.0195	0.0031	0.2	**0.0230**	0.0155	0.0036	0.2	**0.0237**	0.0032	0.2	0.0229	0.0068

Aggregate Diversity - D@10

	JS	AAJ		MAJ	S-AAJ		S-MAJ	ASOR	SOR	AAS		MAS	S-AAS		S-MAS	AJS	AAAJ		MAAJ	S-AAAJ		S-MAAJ
Datasets	**D@10**	λ	**D@10**	**D@10**	λ	**D@10**	**D@10**	**D@10**	**D@10**	λ	**D@10**	**D@10**	λ	**D@10**	**D@10**	**D@10**	λ	**D@10**	**D@10**	λ	**D@10**	**D@10**
LibraryThing	9745	0.2	12004	**17510**	0.4	12304	10249	10399	9556	0.2	11727	**16557**	0.4	13306	9998	**17361**	0.2	11221	11132	0.4	12315	14819
Yahoo	3103	0.2	3718	3463	0.2	**4315**	2541	3346	3184	0.2	**3935**	3397	0.2	3541	2475	2948	0.4	**4092**	2334	0.2	3756	2295
Last FM	3508	0.2	**3538**	3486	0.2	3462	2911	**3779**	3265	0.4	305	3175	0.2	3509	2887	2992	0.2	**3935**	2644	0.2	3692	3457

Workshop, Pisa, Italy, January 16-17, 2013. 73–83. http://ceur-ws.org/Vol-964/paper12.pdf

[4] Marko Balabanovic and Yoav Shoham. 1997. Content-Based, Collaborative Recommendation. *Commun. ACM* 40, 3 (1997), 66–72. https://doi.org/10.1145/245108.245124

[5] Linas Baltrunas and Francesco Ricci. 2009. Item Weighting Techniques for Collaborative Filtering. In *Knowledge Discovery Enhanced with Semantic and Social Information.* 109–126. https://doi.org/10.1007/978-3-642-01891-6_7

[6] Alejandro Bellogín and Pablo Sánchez. 2017. Revisiting Neighbourhood-Based Recommenders For Temporal Scenarios. In *Proceedings of the 1st Workshop on Temporal Reasoning in Recommender Systems co-located with 11th International Conference on Recommender Systems (RecSys 2017), Como, Italy, August 27-31, 2017.* 40–44. http://ceur-ws.org/Vol-1922/paper8.pdf

[7] Daniel Billsus and Michael J. Pazzani. 2000. User Modeling for Adaptive News Access. *User Model. User-Adapt. Interact.* 10, 2-3 (2000), 147–180. https://doi.org/10.1023/A:1026501525781

[8] John S. Breese, David Heckerman, and Carl Myers Kadie. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998.* 43–52. https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=231&proceeding_id=14

[9] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011). In *Proceedings of the 5th ACM conference on Recommender systems (RecSys 2011).* ACM, New York, NY, USA.

[10] Evangelia Christakopoulou and George Karypis. 2016. Local Item-Item Models For Top-N Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016.* 67–74. https://doi.org/10.1145/2959100.2959185

[11] Edjalma Q. da Silva, Celso G. Camilo-Junior, Luiz Mario L. Pascoal, and Thierson Couto Rosa. 2016. An evolutionary approach for combining results of recommender systems techniques based on collaborative filtering. *Expert Syst. Appl.* 53 (2016), 204–218. https://doi.org/10.1016/j.eswa.2015.12.050

[12] Mukund Deshpande and George Karypis. 2004. Item-based top-*N* recommendation algorithms. *ACM Trans. Inf. Syst.* 22, 1 (2004), 143–177. https://doi.org/10.1145/963770.963776

[13] Wei Dong, Charikar Moses, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web.* ACM, 577–586.

[14] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. 2002. An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms. *Inf. Retr.* 5, 4 (2002), 287–310. https://doi.org/10.1023/A:1020443909834

[15] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017.* 306–310. https://doi.org/10.1145/3109859.3109872

[16] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: factored item similarity models for top-N recommender systems. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013.* 659–667. https://doi.org/10.1145/2487575.2487589

[17] Rahul Katarya and Om Prakash Verma. 2016. Effectivecollaborative movie recommender system using asymmetric user similarity and matrix factorization. In *Computing, Communication and Automation (ICCCA), 2016 International Conference on.* IEEE, 71–75.

[18] MG Kendall and JD Gibbons. 1990. Rank correlation methods, trans. *JD Gibbons (5th edn ed.). Edward Arnold: London* (1990).

[19] Marta Millan, Maria Trujillo, and Edward Ortiz. 2007. A collaborative recommender system based on asymmetric user similarity. In *International Conference on Intelligent Data Engineering and Automated Learning.* Springer, 663–672.

[20] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011.* 497–506. https://doi.org/10.1109/ICDM.2011.134

[21] Parivash Pirasteh, Dosam Hwang, and Jason J Jung. 2015. Exploiting matrix factorization to asymmetric user similarities in recommendation systems. *Knowledge-Based Systems* 83 (2015), 51–57.

[22] Ali Mustafa Qamar, Eric Gaussier, Jean-Pierre Chevallet, and Joo Hwee Lim. 2008. Similarity learning for nearest neighbor classification. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on.* IEEE, 983–988.

[23] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009.* 452–461. https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=1630&proceeding_id=25

[24] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001.* 285–295.

[25] Upendra Shardanand and Pattie Maes. 1995. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Human Factors in Computing Systems, CHI '95 Conference Proceedings, Denver, Colorado, USA, May 7-11, 1995.* 210–217. https://doi.org/10.1145/223904.223931

[26] T. Sørensen. 1948. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. *Biol. Skr.* 5 (1948), 1–34.

[27] Harald Steck. 2013. Evaluation of recommendations: rating-prediction and ranking. In *RecSys.* 213–220.

[28] Hal R. Varian. 1999. Economics and Search. *SIGIR Forum* 33, 1 (1999), 1–5. https://doi.org/10.1145/331403.331404