A Parameter-free Algorithm for an Optimized Tag Recommendation List Size

Modou Gueye
Université Cheikh Anta Diop
BP. 64432, Fann
Dakar, Sénégal
gmodou@ucad.sn
modou.gueye@enst.fr

Talel Abdessalem
Institut Telecom - Telecom ParisTech
46, rue Barrault 75013
Paris, France
talel.abdessalem@enst.fr

Hubert Naacke
LIP6 - UPMC Sorbonne Universités
4 place Jussieu 75005
Paris, France
hubert.naacke@lip6.fr

ABSTRACT

Tag recommendation is a major aspect of collaborative tagging systems. It aims to recommend suitable tags to a user for tagging an item. One of its main challenges is the effectiveness of its recommendations. Existing works focus on techniques for retrieving the most relevant tags to give beforehand, with a fixed number of tags in each recommended list. In this paper, we try to optimize the number of recommended tags in order to improve the efficiency of the recommendations. We propose a parameter-free algorithm for determining the optimal size of the recommended list. Thus we introduced some relevance measures to find the most relevant sublist from a given list of recommended tags. More precisely, we improve the quality of our recommendations by discarding some unsuitable tags and thus adjusting the list size.

Our solution is an add-on one, which can be implemented on top of many kinds of tag recommenders. The experiments we did on five datasets, using four categories of tag recommenders, demonstrate the efficiency of our technique. For instance, the algorithm we propose outperforms the results of the task 2 of the ECML PKDD Discovery Challenge 2009 [1]. By using the same tag recommender than the winners of the contest, we reach a F1 measure of 0.366 while the latter got 0.356. Thus, our solution yields significant improvements on the lists obtained from the tag recommenders.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Filtering

General Terms

Algorithms, Experimentation

1. http ://www.kde.cs.uni-kassel.de/ws/dc09/

RecSys'14, October 6–10, 2014, Foster City, Silicon Valley, CA, USA.

http://dx.doi.org/10.1145/2645710.2645727.

Keywords

Tag Recommender Systems, Quality of recommendations, Optimization

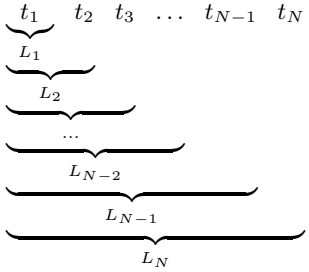
1. INTRODUCTION

Collaborative tagging is the practice of allowing users to annotate content. Users can organize, and search content with annotations so called tags. Nowadays the growth in popularity of social media sites has made the area of recommender systems for social tagging an active and growing topic of research [6].

Tag recommenders aim to suggest the most suitable tags to a user when tagging an item. They are a salient part of the web 2.0 where applications are user-centered. One of their main challenges is the effectiveness of their recommendations. People generally focus on techniques that enable retrieving the best suitable tags to give beforehand, with a fixed number of tags at each recommendation.

We follow another direction in order to improve tag recommendation accuracy. We aim to dynamically adjust the number of tags to recommend. In other words, consider $L_N = \{t_1, t_2, \ldots, t_N\}$ the list of N tags to be recommended to a user, the goal is to substitute L_N by one of its sublists that is more accurate, and provide a better quality of recommendation.

We consider all the sublists of L_N in increasing size (i.e., the prefixes : L_1, L_2, $\ldots L_{N-1}$ and L_N) so as to keep the tag order as illustrated below [2].



We introduce a relevance measure for the recommended lists $Rel(L_N|u,i)$, which estimates the probability that a user u will use all the recommended tags in L_N for the tagging of an item i. Based on this measure, we compute the best list (the one having the optimal size, *bls*) that will be finally recommended to the user. We define the optimal size as

2. $L_N = L_{N-1} \cup \{t_N\} = L_{N-2} \cup \{t_{N-1}, t_N\} \ldots$

follows :

$$bls = \max(s \mid s \in \mathcal{S}) \quad (1)$$

with

$$\mathcal{S} = \left\{ s \;\middle|\; s \leq N \wedge \forall n \leq N,\ Rel(L_n|u,i) \leq Rel(L_s|u,i) \right\}$$

Existing approaches use linear combinations to compute the global average number of tags per post, the one related to a user and/or the one specific to an item [7, 8, 9]. These combinations are then used to infer a fixed list size. Such approaches need some calibrations which are generally difficult to set, and they lack of dynamicity which limits their accuracy.

The algorithm we propose here enables adjusting dynamically the size of the recommended list of tags, and then increases the accuracy of the recommendations. It is a parameter-free algorithm that adjusts the list of recommended tags by discarding those which are estimated irrelevant. Our method looks like an add-on filter on top of a tag recommender. It estimates the sublist which gives the best accuracy. We present in the paper two relevance measures and the algorithm that we use to retrieve the optimal sublist from a given tag recommendation list.

To evaluate the efficiency of our approach we implement it on top of four tag recommenders, from different approaches. One of our candidates is the pairwise interaction tensor factorization model (PITF) of Rendle and Schmidt-Thieme which won the task 2 of the ECML PKDD Discovery Challenge 2009 [9]. It is still considered in the literature as one of the best tag recommenders. We took also the well-known tripartite graph-based algorithm FolkRank [4] and the « Most Popular Tag » recommendation approach [6]. The last candidate is a network-based tag recommender we developed [3], and which computes the list of tags based on the opinions of the users' neighborhood and their tagging posts.

The experiments we did on five datasets demonstrate the efficiency of the optimization technique we propose in this paper.

The remainder of this paper is organized as follows. In Section 2 we present some preliminaries and describe briefly the four tag recommender candidates. Section 3 details the related work and presents our approach for finding the best size of a tag recommendation list to keep. In Section 4, we present our experiments and Section 5 concludes the paper.

2. PRELIMINARIES

A folksonomy is a system of classification that allows its users creating and managing tags to annotate and categorize content. It is related to the event of social tagging systems. A folksonomy $\mathbb{F}$ can be defined as a collection of a set of users U, a set of tags T, a set of items I, and a ternary relation between them $S \subseteq U \times I \times T$ as $\mathbb{F} := (U, I, T, S)$. A tagging triple $(u, i, t) \in S$ means that user u has tagged an item i with the tag t. A user can tag an item with one or more distinctive tags from T. We assume that a user can tag an item with a given tag at most once.

The interest of a tag t for a given user u to annotate an item i is generally estimated by a score $score(t|u,i)$. The purpose of a tag recommender is to compute the top-K highest scoring tags for a post (u,i) which represents its recommendations.

$$Top(u, i, K) = \underset{t \in T}{\overset{K}{argmax}}\ score(t|u,i) \quad (2)$$

For convenience, let us introduce some definitions and properties of a folksonomy :

$$\begin{array}{lcl} T(u) & \equiv & \{t \in T \mid \exists i \in I : (u,i,t) \in S\} \\ T(i) & \equiv & \{t \in T \mid \exists u \in U : (u,i,t) \in S\} \\ T(u,i) & \equiv & \{t \in T \mid (u,i,t) \in S\} \\ I(u) & \equiv & \{i \in I \mid \exists t \in T : (u,i,t) \in S\} \\ I(u,t) & \equiv & \{i \in I \mid (u,i,t) \in S\} \\ U(i) & \equiv & \{u \in U \mid \exists t \in T : (u,i,t) \in S\} \\ U(i,t) & \equiv & \{u \in U \mid (u,i,t) \in S\} \end{array}$$

In the following we describe how our four tag recommender candidates model the scores associated to the tags.

2.1 Factor Models for Tag Recommendation

Factorization models are known to be among the best performing models. They are a very successful class of models for recommender systems where they outperform the other approaches.

We chose the pairwise interaction tensor factorization model (PITF) of Rendle and Schmidt-Thieme in our experimentation due to its efficiency [9]. Indeed, it took the first place of the ECML PKDD Discovery Challenge 2009 for graph-based tag recommendation.

PITF proposes to infer pairwise ranking constraints from the set of tagging triples S. It captures the interactions between users and tags as well as between items and tags. Its model equation is given by :

$$score(t|u,i) = \sum_f \hat{U}_{u,f} \cdot \hat{T}^U_{t,f} + \sum_f \hat{I}_{i,f} \cdot \hat{T}^I_{t,f} \quad (3)$$

Where $\hat{U}$, $\hat{I}$, $\hat{T}^U$ and $\hat{T}^I$ are feature matrices which capture the latent interactions.

The main assumption of PITF is that within a post $T(u,i)$, a tag t can be preferred over another tag t' iff the tagging triple $(u,i,t) \in S$ (i.e., has been observed) and not (u,i,t'). PITF models these preferences in Equation 3 such as the score of a tag which is more preferred than another one is greater.

2.2 FolkRank - A Topic-Specific Ranking

FolkRank is a tripartite graph-based tag recommender designed in the spirit of PageRank [4, 5]. It assumes that a tag becomes important when it is used by important users or to tag important items. It also takes the same principle for users and items. Therefore FolkRank represents a folksonomy $\mathbb{F}$ as a graph where the vertices are mutually reinforcing each other by spreading their weights.

Let $G_{\mathbb{F}} = (V, E)$ be this graph. Its vertices are the users, items and tags (i.e. $V = U \cup I \cup T$) and the edges defined between them such as if a tagging triple $(u,i,t) \in S$ then $\left\{\{u,i\},\{u,t\},\{i,t\}\right\} \subset E$.

Let v_i a vertex of this graph, i.e. $v_i \in V$. We denote by $\mathcal{N}(v_i)$ the set of neighbors of v_i and by $w(v_i, v_j)$ the weight of the edge between the vertices v_i and v_j. The weight of an edge is equal to the number of times its two vertices appear together in the tagging triples in S. From that, the degree of a vertex v_i is defined as follows :

$$w(v_i) = \sum_{v_j \in \mathcal{N}(v_i)} w(v_i, v_j) \quad (4)$$

FolkRank ranks the vertices according to their importances that it computes as follows :

$$PR(v_i) = \lambda \sum_{v_j \in \mathcal{N}(v_i)} \frac{w(v_i, v_j)}{w(v_j)} \cdot PR(v_j) + (1 - \lambda) \cdot p(v_i) \quad (5)$$

where $PR(v_i)$ is the PageRank value and $p(v_i)$ the preference value of the vertex v_i. Hence a straightforward idea for tag recommendation is to set a high preference to the considered user and item (item to be tagged by the user), and then compute ranking values using PageRank as in Equation 5. The parameter λ determines the influence of $p(v_i)$. Its value is between 0 and 1.

To recommend some tags for a user u and an item i, FolkRank uses two random surfer models on the graph, $s^{(0)}$ and $s^{(1)}$, to infer the importance of each vertex of the graph. The first surfer, $s^{(0)}$, set the same preference value to all the vertices ($p(v) = 1, \forall v \in V$) while the second, $s^{(1)}$, set their preference values to 0 except for the user u and the item i for which $p(u) = 1$ and $p(i) = 1$.

After running the two surfers, the difference $s := s^{(1)} - s^{(0)}$ is computed. Then the tags are ranked according to their importance value s, and the first ones are recommended to user u.

2.3 Recommending the Most Popular Tags

The rational of Most Popular Tags' Recommenders (MPTR) is that when a tag t is popular (i.e., frequently used) for an item i and/or by a user u, that tag t may be a relevant recommendation for u aiming to tag i. Hence, MPTR model these popularities in their scoring models. A well-known MPTR model – also known as most popular ρ-mix – consists in adding the tag popularities (i.e. the ones of the user with those of the item) after normalizing and weighting them as follows :

$$score(t|u, i) = \rho \cdot \frac{|U(i, t)|}{|U(i)|} + (1 - \rho) \cdot \frac{|I(u, t)|}{|I(u)|} \quad (6)$$

The parameter ρ in Equation 6 allows to tune the relative importance of the item or user-related tag popularity with respect to the second. When $\rho = 1$ we keep only the tags which are most specific to the item. On the other hand, when we set it to 0, we consider only the user's popular tags. By default we fix ρ to 0.5 in our experimentation. Thus, we consider the user tags as important as those associated to the item. The advantage of MPTR is that they are fast to compute, while giving good predictions [6].

2.4 Trust and popularity-based Recommender

One weakness of MPTR (which entirely relies on popularity measures) is that they are not able to decide between tags with close popularities. Furthermore, some particular users can have their own vocabulary (i.e., tags) and the popular tags of the item may not be relevant for them. Thus, having reliable opinions about the tags from some trusted neighbors, in addition to the popularities of tags, may be a great asset to make better recommendations.
FasTag [3] uses such an approach. It models the relevance score of a tag t for a user u and an item i (i.e. $score(t|u, i)$) as a popularity-dependent component, based a user's trust in his neighbors in the network. Let us denote by $score^{MPTR}(t|u, i)$ the scoring model of MPTR in Equation 6, the scoring one of FasTag is defined as

$$score(t|u, i) = score^{MPTR}(t|u, i) \cdot \Big(1 + \eta(t|u, i)\Big) \quad (7)$$

where $\eta(t|u, i)$ represents the opinion of the user's neighbors about tag t. It is a normalized sum of the user's trust values associated to his neighbors who already tagged this item with t. The rationale of this *score* function is to estimate a relative popularity of a tag depending on the trust neighbors of the user : i.e., the more a user trusts a neighbor, the more this neighbor's opinion contributes in the user recommendations. The sum $1 + \eta(t|u, i)$ enables taking into account the isolated users (when $\eta(t|u, i) = 0$).

We chose FasTag as a candidate for network-based tag recommenders, since it is fast and efficient as shown in [3]. Its scoring model is not only based on the trust associated to the direct neighbors of the user, it also considers trust propagation, following a natural interpretation that trust is, at some extent, transitive. See [10, 11] for more details on trust propagation models.

3. ADJUSTED RECOMMENDATION LIST SIZE

In this section, we present two ways to choose the best recommendation list size. The first one is based on existing works employing linear combination techniques, and the second one is our proposal to optimize dynamically the size of the recommended list.

3.1 Linear combination models

To choose the best list size (bls) of tags to recommend, usual approaches use some linear combinations of the global average number of tags per post, the one related to a user and/or the one specific to an item [9, 8, 7]. Therefore, we take as baseline the following general linear combination model

$$bls = \min\left(K, \lfloor \lambda + (\beta_G \cdot \mu_G) + (\beta_u \cdot \mu_u) + (\beta_i \cdot \mu_i) \rfloor\right) \quad (8)$$

where K stands for the maximum number of tags to recommend, i.e. the maximal list size ; μ_G the global average number of tags per post ; μ_u the average number of tags per user and per post, and μ_i the average number of tags per item and per post. The rest stands for parameters which allow us to make a lot of possible combinations. For instance, if we set the parameter β_u to 1 and all the other parameters to zero, we obtain as a list size the average number of tags per user and post. We denote in the following the linear combination method by *LC_bls*.

For our experiments, we apply a grid search in order to find optimal parameters to keep. We test $K \times 1,000$ combinations of these parameters each time a top-K query is asked (i.e., each time we look for a list of K tags maximum). For instance, we make 10,000 combinations for the top-10 query and 5,000 for the top-5 one. We vary the parameter λ from 0 to $K-1$, each time by a step of 1. And using nested loops, we vary each of the other parameters from 0 to 1 by a step of 0.1. Thus, we test enough combinations with different values of parameters. At the end, we keep the combination that gives the best result (in terms of reached F1-measure).

3.2 The blsC algorithm

blsC denotes the algorithm we propose to find the best list size. Let $Rel(t|u, i)$ be the relevance of a tag t, according to a user u, for the tagging of an item i. And $L_N =$

$\{t_1, t_2, \ldots, t_N\}$ is an ordered list of N tags. We define the relevance of the list L_N, for both a user u and an item i, as follows :

$$Rel(L_N|u,i) = \omega(L_N|u,i) \cdot \frac{\sum_{t \in L_N} Rel(t|u,i)}{N} \quad (9)$$

In this formula, $\omega(L_N|u,i)$ stands as a weight for the adjustment of the list relevance. It allows us to promote longer lists than the others. We will give its definition shortly in Subsection 3.2.2. $Rel(t|u,i)$ is the relevance of a tag t for a user u and an item i. Intuitively, it measures the probability that user u will tag the item i with the tag t.

Let Max be a maximal list size and $Rel(L_{Max}|u,i)$ the relevance of this list (L_{Max}). We look for the best list size starting from Max down to 1. At each step, we compute the relevance of the current list and update the best list size as shown in Algorithm 1. In case we obtain the same relevance

Algorithm 1: *blsC* : Best list size Computation

Input: L_{Max} /* Initial recommended tags list*/
Output: *bls* /* Suggested number of tags to keep */

```
1 bls ← Max                          /* bls : Best list size */
2 blR ← Rel(L_Max|u,i)               /* blR : Best list relevance */
3 for N = (Max − 1) to 1 do
4     if Rel(L_N|u,i) > blR then
5         bls ← N
6         blR ← Rel(L_N|u,i)
7     end
8 end
9 return bls
```

for two different lists, we choose the longest one.

To compute the relevance of a tag, we propose two solutions. In the first one, we distinguish the known tags from the others and assign them different relevance values. In the second solution, we link the relevance values of the tags to some statistics we obtain from the available data (our training sets).

3.2.1 Simple relevance measure

Making a distinction between the known tags (those already used by the user and associated to the item) from the others may be useful to determine the relevance of a recommended list of tags. Our intuition is that the tags already linked to the user u and also to the item i are more relevant to recommend than the others.

Let $P_N = L_N \cap T(u) \cap T(i)$ be the sublist of L_N containing the known tags. We assign a unique high relevance value, Rel_{max}, to the tags in P_N and an unique low one, Rel_{min}, to the other tags. By fixing the weight of each list to one, we can rewrite the Equation 9 as follows :

$$Rel(L_N|u,i) = \frac{\sum_{t_j \in P_N} Rel_{max} + \sum_{t_j \in \{L_N \setminus P_N\}} Rel_{min}}{N} \quad (10)$$

After simplification it becomes :

$$Rel(L_N|u,i) = \frac{\left(|P_N| \cdot Rel_{max} + (N - |P_N|) \cdot Rel_{min}\right)}{N} \quad (11)$$

From Equation 11, one can see that the relevance of a recommendation list depends mainly on the ratio between $|P_N|$ and N. The relevance is an increasing function, having its inputs in the interval $[0, N]$. It reaches its maximum when $|P_N| = N$. Thus, we can simply consider the relevance of a recommendation list as follows :

$$Rel(L_N|u,i) = \frac{|P_N|}{N} \quad (12)$$

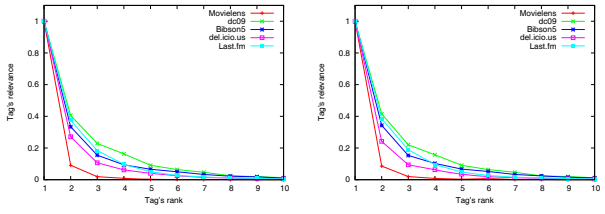
With this formula, we can easily adjust the list size in order to obtain the best relevance value. $Rel(L_N|u,i)$ can be seen as a density measure of known tags in the recommended list. Then, the *blsC* is just seeking for the best density.

3.2.2 Refining the relevance measure

Among the weaknesses of the relevance measure given in Equation 12 we can mention the fact that it fails to give a sublist when all the tags in the recommendation list are known (in P_N). We propose a relevance measure which faces the drawbacks of the previous one by taking into account the popularity of the tags for both the user and the item.

$$Rel(t|u,i) = \frac{|U(i,t)|}{|U(i)|} \times \frac{|I(u,t)|}{|I(u)|} \quad (13)$$

Naturally, we expect that the relevance of a tag decreases according to its rank in a recommended list (high relevance for the first tag and low relevances for the last ones). This intuition is confirmed by our experimentations, on all the datasets we used and for all the tag recommenders we tested, as shown in Figure 1. We limit our tests to the top-10 lists of recommended tags.



(a) MPTR (b) FasTag

(c) PITF (d) FolkRank

Figure 1: Relative relevance of a tag according to its position in a recommended list and the relevance of the first tag

From these observation, we see the need to use weighted means when evaluating the relevance of recommended lists. Indeed, since a recommendation list is ordered, the first tag is generally more relevant than the others (the second one and so on). Thus, using the average of the relevances of the tags contained in a list may still lead to a singleton, i.e. L_1 as the best list.

To take into account this natural decrease of the relevances of tags according to their positions, we define the weight

$\omega(L_N|u,i)$ of a recommendation list L_N as follows :

$$\omega(L_N|u,i) = \frac{N}{\sqrt{\frac{1}{2}(N+Max)}} \quad (14)$$

This weight function estimates the importance of a list size compared to the maximal possible list size. It allows us to penalize short recommendation lists while promoting long lists. Therefore, from Equation 9 we derive a new list relevance measure :

$$Rel(L_N|u,i) = \frac{\sum_{t \in L_N} Rel(t|u,i)}{\sqrt{\frac{1}{2}(N+Max)}} \quad (15)$$

As one can see, we do not compute the mean of the relevance of tags but a relative list relevance according to the greatest possible list size. The comparison of the relevance of all the sublist is done as described in Algorithm 1. We just compute the most relevant list size from Max down to 1 by using this new relevance measure given in Equation 15. We denote this second proposal *blsC_v2*. The latter can overcome the weak points of *blsC*. Indeed, even if all the tags in a recommendation list are in P_N, *blsC_v2* does not rely on their presence but only on their relevance. Thus, it is able to decide when *blsC* fails.

4. EXPERIMENTATION

We demonstrate in this section the effectiveness of our proposals. We led a set of experiments with four tag recommender candidates on five publicly available datasets. In the next two subsections, we shortly describe these datasets and the evaluation measures and methodology we used. Then we present the results we obtained.

4.1 Datasets

We chose five datasets from four online systems : delicious [3], Movielens [4], Last.fm [5], and BibSonomy [6].

We took the ones of delicious, movielens, and last.fm from *HetRec 2011* [2] and the two other ones from Bibsonomy : a post-core at level 5 and a one at level 2 [1, 6]. We call them respectively *Bibson*5 and *dc*09.

*dc*09 is the one of the task 2 of ECML PKDD Discovery Challenge 2009 [7]. This task was especially intended for methods relying on the graph structure of the training data only. The user, item, and tags of each post in the test data are all contained in the training data, a post-core at level 2. Let us remind that a post-core at level p is a subset of a folksonomy with the property that *each user, tag and item occur at least p times* in the tagging triples of S. Table 1 presents some details of these datasets.

4.2 Evaluation Measures and Methodology

To evaluate our solution, we used a variant of the leave-one-out hold-out estimation called LeavePostOut [6]. In all datasets except *dc*09, we picked randomly for each user u, one item i that he had tagged before. Thus, we created a test set and a training one. The task of our recommender was then to predict the tags the user assigned to the item.

3. http://www.delicious.com
4. http://www.grouplens.org
5. http://www.lastfm.com
6. http://www.bibsonomy.org
7. http://www.kde.cs.uni-kassel.de/ws/dc09/

Table 1: Characteristics of the datasets

dataset	$\|U\|$	$\|I\|$	$\|T\|$	$\|T(u,i)\|$
dc09	1,185	22,389	13,276	64,406
Last.fm	1,892	17,632	11,946	71,065
delicious	1,867	69,226	53,388	104,799
Movielens	2,113	10,197	13,222	27,713
Bibson5	116	361	412	2,526

Dataset	% of users with at most #tags per post				
	1	2	3	4	5
Bibson5	5.17	21.55	50.0	69.82	81.89
dc09	6.49	28.27	52.82	73.58	85.23
delicious	2.89	19.86	41.32	61.72	76.17
Last.fm	12.57	46.32	66.50	78.65	86.47
Movielens	**65.37**	**92.57**	**97.63**	**98.95**	**99.76**

Dataset	% of items with at most #tags per post				
	1	2	3	4	5
Bibson5	0.27	9.418	32.40	61.21	80.60
dc09	11.46	25.45	45.47	64.47	78.83
delicious	11.10	27.08	46.61	63.72	76.40
Last.fm	19.77	51.87	79.50	90.03	94.08
Movielens	**41.76**	**81.41**	**92.72**	**96.39**	**97.86**

On each dataset, we ran the choosed tag recommenders. We successively obtained from them a top-*1*, then a top-*2* and so on, up to a top-*10*. Then, we applied our algorithms *LC_bls*, *blsC* and *blsC_v2* on the tag recommendation lists in order to get better sublists. The performance evaluation was based on the the F1-measure.

Let us notice that for all the experiments, we fixed the parameter ρ of MPTR and FasTag to 0.5 (see Equation 6). Similarly, we set the parameter λ of FolkRank to 0.7 (see Equation 5) as in [5]. For PITF we used the software [8] and the parameters given by the winners of the task 2 of the ECML PKDD Discovery Challenge 2009, and we did not rely on assembling factor models as they did in [8]. We only computed one model with 64 factors as the dimensionality and a regularization value of $5 \cdot 10^{-5}$. We ran the learning phase for $2,000$ iterations.

4.3 Results

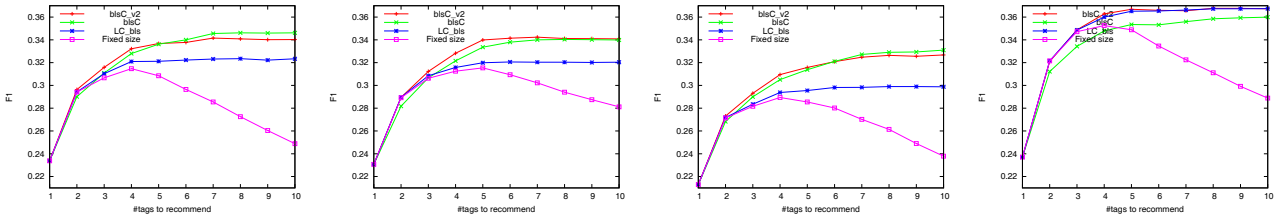
We present in this part the results of our experimentation. We aim to point out that our proposal outstrips the methods based on linear combinations.

4.3.1 *Effectiveness of our proposals*

As we said, we show here the effectiveness of our proposals. We implemented them on top of the four tag recommenders. Figures 2 to 6 show the F1 quality of the original recommendation lists given by each tag recommender and the ones of each optimization method (i.e., *blsC*, *blsC_v2* and *LC_bls*). The x-axis of each of these figures gives the original number of tags to recommend before length optimization.

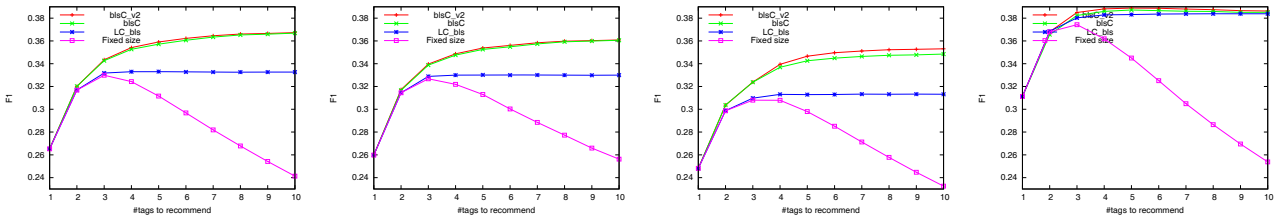
In almost all these figures, we see that the quality of the adjusted lists is increasing while the one of those with fixed sizes decreases when they exceed a certain size (specific to

8. http://bit.ly/1qL6NeF



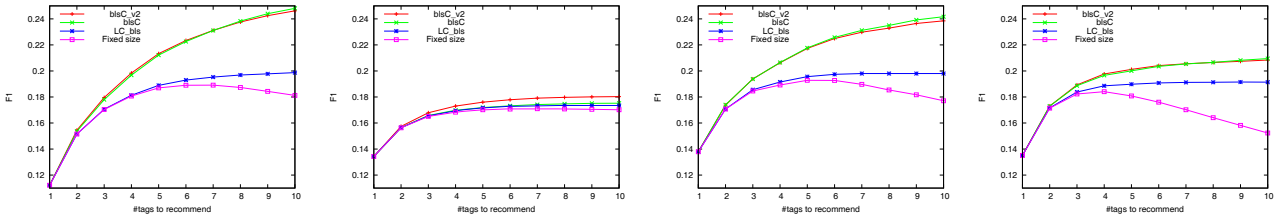
(a) MPTR (b) FasTag (c) FolkRank (d) PITF

Figure 2: Quality increases versus recommendation list sizes on *dc*09



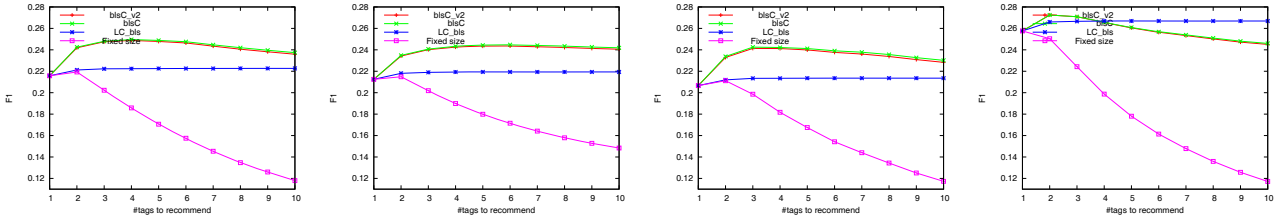
(a) MPTR (b) FasTag (c) FolkRank (d) PITF

Figure 3: Quality increases versus recommendation list sizes on *Last.fm*



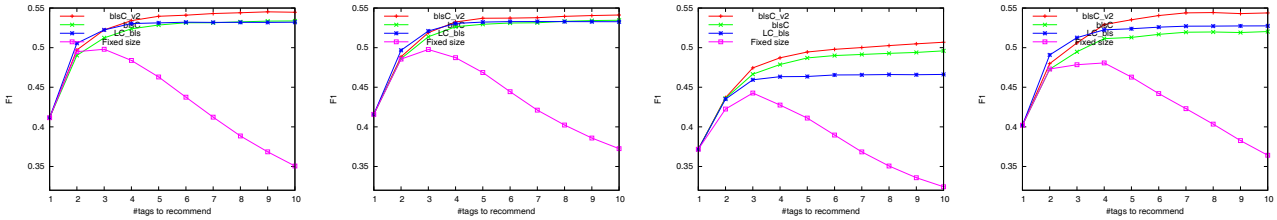
(a) MPTR (b) FasTag (c) FolkRank (d) PITF

Figure 4: Quality increases versus recommendation list sizes on *delicious*



(a) MPTR (b) FasTag (c) FolkRank (d) PITF

Figure 5: Quality increases versus recommendation list sizes on *Movielens*



(a) MPTR (b) FasTag (c) FolkRank (d) PITF

Figure 6: Quality increases versus recommendation list sizes on *Bibson*5

each dataset). This demonstrates the importance of giving optimal recommendation list size.

Second, in the most cases *blsC_v2* outperforms *blsC* and linear combinations. For instance, it outstrips the results of the task 2 of the ECML PKDD Discovery Challenge 2009. With the same tag recommender than the winners, we reach an F1 measure of 0.366 while they got 0.356. They used a linear combination to adjust the size of the recommended lists (see [8]). *blsC_v2* yields 3% of improvement over their F1 score on the dataset of the challenge.

Tables 2 and 3 compare the percentage of cases where each of the optimization methods gives the best contribution. Linear combinations overpass our proposals in only 9.19% of cases (see 2). On all the rest, *blsC_v2* dominates with more than 60% of cases, and *blsC* follows with 30%. Linear combinations are mainly better on Movielens dataset, which was the worst dataset of our five ones. In this dataset around 65% of the users never used more than one tag in their posts, which is poor for accurate recommendations. Similarly 41.76% of the items have never been tagged with more that one tag (see Table 1). This explains why our methods which use the popularity of the tags fail to give better results than linear combinations.

Table 2: Comparison of the three methods

Method	% of cases with the best size
blsC_v2	**60.34**
blsC	30.45
LC_bls	9.19

Table 3: Comparison by pairs

Method	% of cases with the best size
blsC_v2	**90.85**
LC_bls	9.14

Method	% of cases with the best size
blsC	**76.96**
LC_bls	23.03

Method	% of cases with the best size
blsC_v2	**65.34**
blsC	34.09

4.3.2 Average optimal list size

Table 4 presents the average optimal list size given by the methods *LC_bls*, *blsC* and *blsC_v2*. In addition to the fact that *blsC_v2* leads to the best size for 90.85% of cases compared to *LC_bls* (see Table 3), we see it gives longer lists than the latter in 80% of cases.

5. CONCLUSION

We presented a new proposal that improves the accuracy of the obtained recommendations from a tag recommender system. Our solution optimizes the size of the recommended list in order to obtain a better recommendation quality. The experimentation we did shows the effectiveness of our approach. Furthermore, our approach suits well in the context of recommendation diversity. Since it shortens the number of relevant tags to recommend, it frees some space that can be used for the diversification purposes.

Table 4: Average optimal list length with 10 tags at maximum

Dataset	FasTag		
	LC_bls	*blsC*	*blsC_v2*
bibsonomy	3.28	3.73	**4.91**
movielens	1.96	6.51	**6.56**
delicious	**7.17**	2.92	3.08
lastfm	**5.94**	4.22	4.72
dc09	4.56	3.93	**4.86**

Dataset	MPTR		
	LC_bls	*blsC*	*blsC_v2*
bibsonomy	3.21	3.85	**5.00**
movielens	1.90	8.23	**8.28**
delicious	7.23	7.12	**7.35**
lastfm	**5.53**	4.81	5.34
dc09	4.56	4.57	**5.51**

Dataset	FolkRank		
	LC_bls	*blsC*	*blsC_v2*
bibsonomy	3.49	3.70	**4.94**
movielens	2.37	8.23	**8.29**
delicious	6.75	7.18	**7.51**
lastfm	**5.91**	4.76	5.28
dc09	4.41	4.44	**5.46**

Dataset	PITF		
	LC_bls	*blsC*	*blsC_v2*
bibsonomy	3.50	6.89	**7.70**
movielens	1.67	9.37	**9.41**
delicious	5.25	9.36	**9.47**
lastfm	6.08	7.51	**7.97**
dc09	4.53	6.35	**7.38**

6. REFERENCES

[1] D. Benz, A. Hotho, R. Jäschke, B. Krause, F. Mitzlaff, C. Schmitz, and G. Stumme. The social bookmark and publication management system BibSonomy. *The VLDB Journal*, 2010.

[2] I. Cantador, P. Brusilovsky, and T. Kuflik. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, 2011.

[3] M. Gueye, T. Abdessalem, and H. Naacke. An efficient trust- and popularity-based tag recommender. `http://www.infres.enst.fr/~mogueye/fastag.php`, 2013.

[4] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Folkrank : A ranking algorithm for folksonomies. In *Fachgruppe Information Retrieval (FGIR)*, 2006.

[5] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In *Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, 2007.

[6] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in social bookmarking systems. *AI Commun.*, 2008.

[7] L. Marinho, C. Preisach, and L. Schmidt-Thieme. Relational classification for personalized tag recommendation. In *ECML PKDD Discovery Challenge*, 2009.

[8] S. Rendle and L. Schmidt-Thieme. Factor models for tag recommendation in bibsonomy. In *ECML PKDD Discovery Challenge*, Bled, Slovenia, 2009.

[9] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Int. Conf. on Web search and data mining (WSDM)*, 2010.

[10] M. Tahajod, A. Iranmehr, and N. Khozooyi. Trust management for semantic web. In *Computer and Electrical Engineering (ICCEE)*, 2009.

[11] C.-N. Ziegler and G. Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 2005.