UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ADRIANO CARNIEL BENIN

A comparison of recommender systems for crowdfunding projects

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Bruno Castro da Silva

Porto Alegre
April 2018

ABSTRACT

Este documento é um exemplo de como formatar documentos para o Instituto de Informática da UFRGS usando as classes LaTeX disponibilizadas pelo UTUG. Ao mesmo tempo, pode servir de consulta para comandos mais genéricos. *O texto do resumo não deve conter mais do que 500 palavras.*

Keywords: UFRGS. recommender systems. AI. crowdfunding.

Using LaTeX to Prepare Documents at II/UFRGS

RESUMO

This document is an example on how to prepare documents at II/UFRGS using the LaTeX classes provided by the UTUG. At the same time, it may serve as a guide for general-purpose commands. *The text in the abstract should not contain more than 500 words.*

Palavras-chave: Electronic document preparation. LaTeX. ABNT. UFRGS.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

SMP	Symmetric Multi-Processor
CBF	Content-based filtering
CF	Collaborative filtering
SPMD	Single Program Multiple Data
ABNT	Associação Brasileira de Normas Técnicas

CONTENTS

1 INTRODUÇÃO

No início dos tempos, Donald E. Knuth criou o TEX. Algum tempo depois, Leslie Lamport criou o LATEX. Graças a eles, não somos obrigados a usar o Word nem o Libre-Office.

2 RECOMMENDATION ALGORITHMS

Recommendation Algorithms are widely used in the industry today to provide useful suggestions to end-users in a completely automated manner. They are ubiquitous in modern e-commerce Web sites (SCHAFER; KONSTAN; RIEDL, 2001), where new products can be recommended based on a customer's interests and preferences, and in many other fields such as movies (Netflix) and music (Spotify). Its importance can't be overstated: the effectiveness of targeted recommendations, as measured by click-through and conversion rates, far exceed those of untargeted content (LINDEN; SMITH; YORK, 2003).

The basic idea behind any recommender system is to obtain a utility function to estimate a user preferences towards an item. The meaning of this function will differ for each context; it could mean how likely a user will want to watch a specific movie or listen to a song, or the likelihood of buying a particular product. In our case, the goal is to find projects the customer is most likely to back given his backing history and other characteristics.

Recommender systems can be broadly divided into two categories: content-based methods and collaborative filtering methods (RAKESH; LEE; REDDY, 2016). The former utilizes the content features of users or items in order to recommend items to users. In collaborative filtering methods, user ratings and other data are used in order to calculate similarities between users or items, which are then ranked to show the most relevant recommendations. These two methods are sometimes combined into what is known as Hybrid Recommender Systems.

2.1 Collaborative filtering

Collaborative filtering is based on the principle that similar users will share similar interests. In the traditional approach, each customer is represented by a N-dimensional vector of items, where N stands for the number of available items, and each vector component corresponds to the user rating of the item. These ratings can be obtained explicitly - e.g. star rating or "likes" - or implicitly - e.g. a user buying an item or listening to a song can be considered a positive rating. By collecting ratings from each user, we build a R_{nxm} matrix which is the starting point for CF. For most applications, R will be extremely sparse, our job then is to try to predict the missing ratings.

CF algorithms can be further divided into Memory-Based and Model-Based techniques. Memory-Based algorithms use a dataset of user-item pairs to identify groups of similar users, which are in turn used to make predictions of preference for new items. In Model-Based methods, machine learning algorithms are used to develop a model that will be used to predict user ratings. These methods can solve some of the shortcomings of Memory-Based approaches, such as the need for large rating matrices. Finally, both approaches can be combined into Hybrid Recommenders. A overview of these techniques is depicted in figure 2.1.

2.1.1 Neighbourhood methods

Neighborhood methods take rows or columns of R and compute a similarity value between them. If the rows correspond to users and the columns to items, we can obtain the similarity between two users u and v by computing the correlation between $R(u)$ and $R(v)$. In the same fashion, similarity between items i and j can be computed by taking $R^T(i)$ and $R^T(j)$.

For User-Based CF, given users u and v one common method to compute similarity is the cosine measure given by:

$$sim(u,v) = \cos(u,v) = \frac{u \cdot v}{||u|| \cdot ||v||} = \frac{\sum_{i \in I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I_u} r_{ui}^2} \sqrt{\sum_{i \in I_v} r_{vi}^2}}$$

where I_u and I_v are the sets of items rated by user u and v respectively and $I_{uv} = I_u \cap I_v$. To get a rating prediction $\check{r}_{ui}$ given by the user u to item i, we aggregate ratings from the subset of users most similar to u. One such function could be:

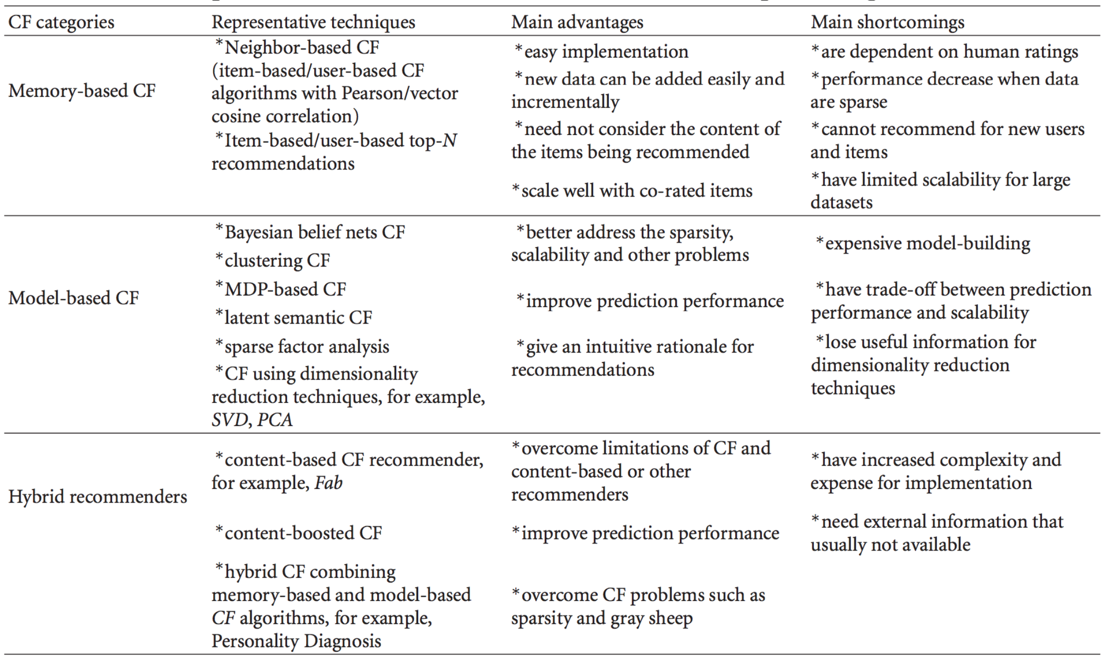
$$\check{r}_{ui} = n \sum_{u' \in U'} sim(u,u') r_{u'i}$$

where n is a normalizing factor.

2.1.2 Item-Based CF

Item-Based CF works almost exactly the same as User-Based CF. One important distinction is that item-based CF retrieves recommendations directly from the similarity matrix and does not require R to be kept in memory. A major reason for the adoption of

Figure 2.1: Overview of collaborative filtering techniques.



CF categories	Representative techniques	Main advantages	Main shortcomings
Memory-based CF	*Neighbor-based CF (item-based/user-based CF algorithms with Pearson/vector cosine correlation) *Item-based/user-based top-*N* recommendations	*easy implementation *new data can be added easily and incrementally *need not consider the content of the items being recommended *scale well with co-rated items	*are dependent on human ratings *performance decrease when data are sparse *cannot recommend for new users and items *have limited scalability for large datasets
Model-based CF	*Bayesian belief nets CF *clustering CF *MDP-based CF *latent semantic CF *sparse factor analysis *CF using dimensionality reduction techniques, for example, *SVD*, *PCA*	*better address the sparsity, scalability and other problems *improve prediction performance *give an intuitive rationale for recommendations	*expensive model-building *have trade-off between prediction performance and scalability *lose useful information for dimensionality reduction techniques
Hybrid recommenders	*content-based CF recommender, for example, *Fab* *content-boosted CF *hybrid CF combining memory-based and model-based *CF* algorithms, for example, Personality Diagnosis	*overcome limitations of CF and content-based or other recommenders *improve prediction performance *overcome CF problems such as sparsity and gray sheep	*have increased complexity and expense for implementation *need external information that usually not available

Source: (SU; KHOSHGOFTAAR, 2009)

this approach is that in most systems users are much more numerous than items, leading to a significantly reduced similarity matrix when using item-based CF (**??**). We can redefine our cosine measure for this case like so: let U_i and U_j be the set of users who rated items i and j respectively, and $U_{ij} = U_i \cap U_j$ be the set of users who rated both items i and j, the similarity is then given by:

$$sim(i,j) = \cos(i,j) = \frac{i \cdot j}{||i|| \cdot ||j||} = \frac{\sum_{u \in U_{ij}} r_{ui} r_{uj}}{\sqrt{\sum_{u \in U_i} r_{ui}^2}\sqrt{\sum_{u \in U_j} r_{uj}^2}}$$

2.1.3 Item-to-Item CF

For large databases, CF can be prohibitively computationally expensive. Its worst case performance is $O(MN)$ where M is the number of customers and N is the number of items (LINDEN; SMITH; YORK, 2003), but this problem can be generally alleviated due to the customer vector sparsity. Item-to-item CF is another method proposed by Amazon that tries to minimize these scaling issues by focusing on item instead of user similarity. Each item purchased by the customer is compared to other items in the dataset in order to calculate a similarity metric. The algorithm is shown bellow:

```
for each item in product catalog, I1 do
    for each customer C who purchased I1 do
        for item I2 purchased by customer C do
            Record that a customer purchased I1 and I2;
        end
    end
    for each item I2 do
        Compute the similarity between I1 and I2;
    end
end
```

Algorithm 1: Item-to-item CF

In either case, for this to work large amounts of user data is required. This is known as the cold start problem: new users who haven't rated many items yet will have a reduced recommendation quality. On the other hand, no information about the item itself is needed, making CF specially applicable to collections of hard-to-analyze items such as movies.

2.2 Content-based filtering

In this method a description of each item is constructed using structured data or some sort of item presentation algorithm such as Latent Dirichlet Allocation or TF-IDF. This representation is then compared to the user profile and the best-matching items are recommended. The user profile can consist of many different types of information: a history of the user's interaction with the system such as page views, searches and purchases is often used to train the model without any explicit user input. Some systems may require the user to explicitly state his interests, however it may be very hard to get users to make this effort, rendering this approach very limited in practice.

As CBF focus on item rather than user similarity, it avoids the cold start problem since little information about user preference is needed. However, since only similar items to those already rated by the user will be considered, CBF approaches tend to suffer from over-specialization (IAQUINTA et al., 2008). This is known as the serendipity problem. Another limitation of CBF algorithms is that items are required to contain enough information in order to distinguish items the user likes from items the user doesn't like. For example, a dataset of songs where only the song name is available would not be enough to make good predictions based on this content only, however this would pose no problem for collaborative filtering methods which rely solely on user similarity.

3 ABOUT CATARSE

Launched in January 2011, Catarse was the first crowdfunding platform for creative projects in Brazil. With over 7000 successfully financed projects raising R$77 millions from 480.000 people, it's currently the largest national platform of its kind. It works similarly to most crowdfunding platforms: the project owner presents his or her idea and specifies the required investment as well as the cutoff date for the project, while offering rewards for those who back it. Projects are divided into 3 main categories: all-or-nothing, flexible and recurrent. In the first type, projects are available for backing up to 60 days and the project owner only receives the raised amount if the project's goal is met, otherwise all the money is returned to its original backers. On flexible projects the owner receives the raised amount whether the goal is reached or not. Recurrent projects are subscription based and the owner can collect the money monthly. This work will only focus on the first two types of projects.

4 RECOMMENDING PROJECTS

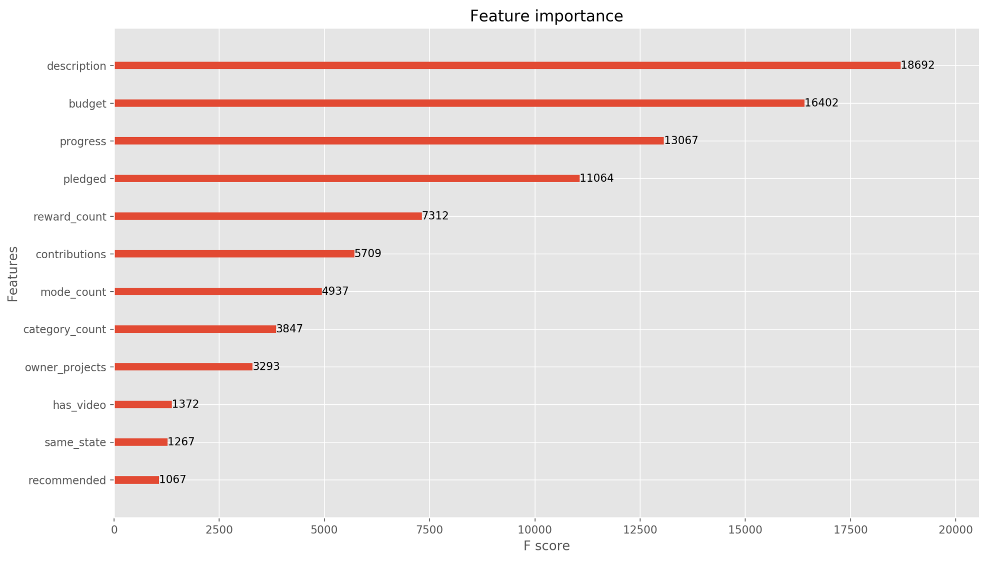
Catarse's dataset is unique in a variety of ways that makes collaborative filtering techniques hard to apply. Firstly, unlike regular e-commerce Web sites such as Amazon where products stay available for many years, crowdfunding projects have a predetermined cutoff date, after which it's no longer possible to make a pledge. It makes no sense to recommend expired projects, severely limiting our training data to online projects only. Another challenge is the fact that the majority of Catarse's users only back one project, making it very hard to get enough data for CF methods to work properly. Our only choice is then to use content-based methods, this allows us to train our model with the whole dataset to extract backer-project features, and later use this model to search for online projects with the highest backing probability for the current user.

4.1 Data preparation

The dataset D was obtained directly from Catarse's production database. Each training example consists of a project-backer pair (p, b) with 12 dimensions presented below:

- *category count*: Number of projects backed by the user b in the same category as project p
- *mode count*: Number of projects backed by the user b with the same mode (aon, flex or sub) as project p
- *same state*: True if the project location is the same as the backer's
- *recommended*: True if project p was manually recommended by an admin
- *has video*: True if the project has uploaded a video
- *budget*: Budget text length in chars
- *description*: Description text length in chars
- *pledged*: Amount pledged in the first 3 days
- *contributions*: Amount of contributions in the first 3 days
- *progress*: Percentage of goal reached in the first 3 days
- *owner projects*: Amount of projects from the same project owner as p
- *reward count*: Number of offered rewards in project p

Figure 4.1: Feature weights in gbtree



In order to trim irrelevant data, we remove canceled and draft projects as well as projects with no backers from the dataset. The final cardinality of D was 300000. For the sake of balancing our dataset, we create 300000 more negative instances by randomly combining users with projects they haven't backed. To further emphasize project quality, only successful projects are considered for the positive instances, while only failed projects are used for the negative instances. Finally, due to changes in the platform that occurred in 2015, we ignore data from earlier periods so as to maintain a consistent dataset.

5 GRADIENT BOOSTING TREES

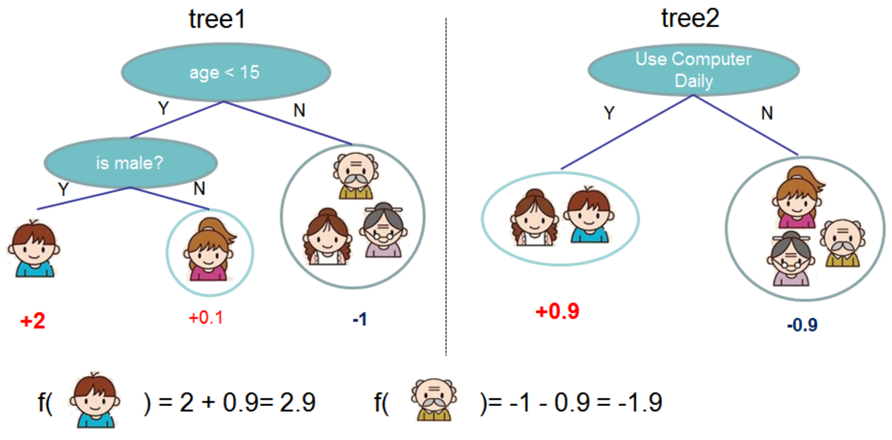
In recent years, tree boosting has become a increasingly popular method and been shown to give state-of-the-art results for many classification problems (LI,). As in any supervised learning method, our objective is to train a model to predict a target variable y_i given features x_i. Boosting methods are characterized by combining many weak learners into a strong one in a iterative fashion. In this case, our model is an ensemble of trees, more specifically a set of classification and regression trees (CART). Unlike decision trees, where the leafs contain decision values, the leafs on gradient boosting trees contain a score $s \in R$. Multiple simple trees are then constructed and the prediction of each tree is summed up to get the final score 5.1. Our model can then be defined as

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

where K is the number of trees, f is a function in the functional space $\mathcal{F}$, and $\mathcal{F}$ is the set of all possible CARTs (CHEN; GUESTRIN, 2016). Our objective function can the be written as

$$obj(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Figure 5.1: GbTree example



Source: XGBoost official site

5.1 Objective function

Our objective function is used to measure the performance of our model for each set of parameters. We define it as the sum of the training loss function L with the regularization term Ω.

$$obj(\theta) = L(\theta) + \Omega(\theta)$$

In this work we define our loss function as the logistic loss function for logistic regression.

$$L(\theta) = \sum_i [y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})]$$

And our regularization function is defined as follows:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Where w is the vector of scores on leaves and T is the number of leaves.
After defining our objective function, we optimize it in order to train our model.

6 IMPLEMENTATION

The XGBoost python library was used to create our gradient boosting tree model. Hyperparameters were tuned by running 10-fold cross validation while optimizing for negative log-likelihood. No data standardization or normalization was necessary since the base learners are trees.

REFERENCES

CHEN, T.; GUESTRIN, C. XGBoost. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16**. [S.l.]: ACM Press, 2016.

IAQUINTA, L. et al. Introducing serendipity in a content-based recommender system. In: **2008 Eighth International Conference on Hybrid Intelligent Systems**. [S.l.]: IEEE, 2008.

LI, P. Robust logitboost and adaptive base class (abc) logitboost.

LINDEN, G.; SMITH, B.; YORK, J. Amazon.com recommendations: item-to-item collaborative filtering. **IEEE Internet Computing**, Institute of Electrical and Electronics Engineers (IEEE), v. 7, n. 1, p. 76–80, jan 2003.

RAKESH, V.; LEE, W.-C.; REDDY, C. K. Probabilistic group recommendation model for crowdfunding domains. In: **Proceedings of the Ninth ACM International Conference on Web Search and Data Mining - WSDM 16**. [S.l.]: ACM Press, 2016.

SCHAFER, J. B.; KONSTAN, J. A.; RIEDL, J. E-commerce recommendation applications. In: **Applications of Data Mining to Electronic Commerce**. [S.l.]: Springer US, 2001. p. 115–153.

SU, X.; KHOSHGOFTAAR, T. M. A survey of collaborative filtering techniques. **Advances in Artificial Intelligence**, Hindawi Limited, v. 2009, p. 1–19, 2009.