Reccomendation System : Collaborative filtering item-based approach

Giovanni Buscemi

Università degli studi di Milano

1 Introduction

In the digital age, with the arrival of large on-line store, it has become increasingly difficult to find products that meet one's needs. Reccomendation systems have been shown to provide important information about user behavior, offering ways to suggest items to users that they are likely to enjoy. In fact, all the on-line stores and platforms have their own reccomandation system that try to improve user experience. For example, Netflix reccomends film based on item-similarity, while Youtube reccomends video based on an hybrid approach. Despite the rise of deep learning, collaborative filters have remained popular due to their trade-off between complexity and computational accuracy. There are two main way to realize a Collaborative filter : user and item based. In this paper, we focus on item-based similarity approach, which has proven to be better of the user-based approach, because it's simpler, it doesn't require to be updated often. Additionally, there is a significant distinction between users and items; while two users may have similar tastes in one aspect, they could have different preferences in another. However, item similarity is easier to detect due to the inherent properties of the items. Therefore, this paper presents an implementation of collaborative filtering based on item similarity that leverages parallel computation provided by Spark and the MapReduce paradigm. The objective is to develop an algorithm that scales well with data volume and can extract useful information from users.

2 Algorithm and method

Collaborative-Filtering item-based system focus on the relationship between items. The relationship is modeled by choosing a notion of similarity. Different measure of similarity are available, but in this approach the focus is on Cosine Distance between items. Basically cosine similarity it's the dot product of the vectors divided by the product of their length and this measure focus on the angle between vectors and it's defined as:

$$\text{Cosine Similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum A_i B_i}{\sqrt{\sum (A_i)^2}\sqrt{\sum (B_i)^2}}. \tag{1}$$

In collaborative filters, unlike the content-based one, the similarity is not based on item's proprieties but on user's rating. So this measure the distance of two

item by user's rating to the same item. The proposed algorithm aim to calculate user's vote for unrated businesses, in order to reccomend item with high similarity to the business that user has rated. Before explaining the algorithm, we need to do some consideration. For first, utility and similarity matrix are sparse, so we don't store into matrix, but we use a structure like the triple method. In addiction, the algorithm aim to be scalable with the use of Spark. Spark offers distributed computing capabilities, allowing us to process large datasets efficiently across multiple nodes in a cluster. By leveraging Spark's RDD (Resilient Distributed Dataset) abstraction, we can perform parallelized computations on our data, making it suitable for handling the computational demands of collaborative filtering algorithms. The algorithm is divided into 3 phases :

1. **Create Utility Matrix** : From the review json file, user_id and business_id are mapped into unique integer value. Trough a map function, we create an RDD containing key-values pair in the format : (User,(Business,Score)).
2. **Compute Similarity Matrix**: Before calculating the similarity matrix, the utility matrix is scaled by subtracting the average of the users' ratings for each rating. This step is useful to normalize the data and account for differences in rating scales among users. To compute the similarity between items, we focus on the columns of the utility matrix. To achieve this, we perform a matrix transposition, representing the data in a way where the rows correspond to the items and the columns correspond to the users. Subsequently, we compute the cosine similarity between all item combinations, considering only the lower half of the matrix, since the similarity matrix is symmetric, so we compute only the lower half part of the matrix. This allows us to reduce computational cost but does not reduce the storage space, because the other part of the matrix is still memorized, but anyways the similarity matrix contains only non-zero values and does not include the leading diagonal (since the similarity of an item with itself is always 1).
3. **Blank Value Predictions** : This is the most important step. Basically here, we iterate on utility matrix by user and for each user we attempt to predict every blank value . For each blank value, we identify the k values with the highest similarity within the corresponding row of the similarity matrix. These values are expected to represent the most similar items to the item we are considering. If the user has rated some of these items, we compute the prediction of the user u rating for the item i using the formula:

$$\text{Rating}(u,i) = \frac{\sum_{j \in \text{Similarity}} \text{Similarity}(i,j) \times \text{Rating}(u,j)}{\sum_{j \in \text{Similarity}} \text{Similarity}(i,j)} + \text{AvgRating}(u) \tag{2}$$

This value is not computed if there's no similarity or if user has no ratings.Another consideration of the algorithm is its vulnerability to the "cold start" problem. This occurs when there is a new user, a new item, or an inactive user. This can lead to various issues. For instance, a new item may have no ratings by users, resulting in no correspondence in the similarity matrix. Similarly, a new user with no ratings may not receive any good recommendations. The "cold start" problem will be addressed in the Experiment section.

3 Experiment

Dataset and preprocessing The dataset is defined by 3 attribute : user_iid,business_id, scores. In order to have a good representation of the relation between user and item, the preprocessing goal is to eliminate non informative user and business. User and business are non informative if they have only few review.

Computing The computation of the algorithm starts by setting the SparkSession, so we can use parallel computing by using cluster computing or thread computing if spark is executed in local. Data are manupulated through RDD, which can be distributed along the worker node. Here we choose the fundamental parameter for the algorithm that are : k,T. k determines the number of the similar item to find in the sorted row of the similarity matrix. T is a threshold that indicates the minimum number of user-made reviews or reviews received by a business.

Cost of the algorithm In order to analyze the scalability of the algorithm, we need to do some consideration about the cost. Considering n as the number of review, u as the number of user and b as the number of business, the algorithm for calculating the Maxtric containing the prediction is the most expensive in term of computational complexity. Considering a double loop iteration to calculate every blank value, the sorting operation on the similarity row, $O(u \times b^2 logb)$. About the Space complexity of the algorithm, the space needed to memorize the Utility prediction matrix is $O(u)$ by considering that the prediction matrix is not supposed to be sparse. Here we are not considering k because $k \ll u, n$

4 Experimental results

Different experiment are conducted. If the start is a random subsamle of the dataset, without preprocessing, the Utility matrix is very sparse, and the 'Cold start problem' is evident. In fact, the result contain a lot of blank row, that means that there's no similar item to reccomend for a given user. Also the Similarity Matrix is really sparse without preprocessing, because there's not enough information to calculate similarity between items. If T, the threshold, is large enough, there's no blank value and the Prediction matrix contain suggestion for every user. With the augment of the dimension of the data, the algorithm is not executable in local. That means that the CPU architecture is not enough for the dimension of the data and the complexity of the algorithm. Algorithm could be optimized in the calculation of the blank values in different way. One way is to use a LSH to reduce the cost in memory of the similarity matrix and the computational cost of calculating the similarity matrix and sort the row to extract the similar business.

5 Declaration

"I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study."