Computers in Industry 65 (2014) 279–290

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Cloud based real-time collaborative filtering for item–item recommendations

Rafael Pereira [a], Hélio Lopes [a,*], Karin Breitman [a,c], Vicente Mundim [b], Wandenberg Peixoto [b]

[a] Department of Informatics, Pontifical Catholic University of Rio de Janeiro, Rua Marquês de São Vicente 225, Gávea, Rio de Janeiro 22.453-900, Brazil
[b] WebMidia, Globo.com, Brazil
[c] EMC², Brazil

ARTICLE INFO

Article history:
Received 22 January 2013
Received in revised form 10 September 2013
Accepted 6 November 2013
Available online 2 December 2013

Keywords:
Distributed architecture
Cloud computing
System architecture
Service orientation
Collaborative filtering

ABSTRACT

We describe a large scale implementation of a video recommendation system in use by the largest media group in Latin America. Taking advantage of existing recommendation system techniques, the proposed architecture goes beyond the state of the art by making use of a commercial cloud computing platform to provide scalability, reduce costs and, more importantly, response times. We discuss the implementation in detail, in particular the design of cloud based features. We also provide a comprehensive generalization of the architecture that allows its application in other settings.

1. Introduction

The ability to offer relevant suggestions to users is becoming extremely important for web applications, particularly those whose business models are dependent on audience ratings, e.g., content commercialization and product sales [1]. Moreover, because effective recommendations play such an important role in user experience, users frequently resort to recommendations as a means for content discovery. This is the case with Amazon [2,3] and Netflix [4–6].

Recommendation systems [7,8] have been recognized as an important tool on web science and e-commerce applications [9,10]. They usually combine user profiles and product information to generate recommendations. In other words, a recommendation system is composed of contextual data, which is the information that the system has before it starts the recommendation process, input data, which is the information received for the recommendation process, and an algorithm that uses the context and input data to model recommendations.

Collaborative filtering [11–14] is a recommendation technique that resorts to the user–item interaction history to find relationships between them. One example of such a relationship is computing the similarity between two items, such as videos [15], both viewed by the same group of users. In this case, no contextual information about the items is considered, which means that the recommendation algorithm does not have any information on which are the items and what are their types or characteristics. According to Herlocker et al. [16], collaborative filtering is considered the finest technique of choice for recommendation algorithms. However, there are several challenges associated with collaborative filtering engines [10]: these algorithms must have sufficient performance to manipulate large sparse datasets, scale as the numbers of users and items grow, and retrieve relevant recommendations within a reasonable timeframe.

Thus, increasing the numbers of users and items poses great challenges for this type of system. One of these challenges is to improve the quality of recommendations made to users, since good recommendations increase user fidelity. A further challenge is the appropriate scaling of the system to retain its effectiveness. These two challenges tend to be conflicting; to obtain fast recommendations the model should be simple to use, but this reduces the overall quality of the recommendations. Nevertheless, an efficient recommendation engine must take both aspects into consideration.

Additionally, there are specific scenarios that require the real-time processing of input data to produce relevant recommendations.

* Corresponding author. Tel.: +55 21999945339; fax: +55 2135271500.
E-mail addresses: rpereira@inf.puc-rio.br (R. Pereira), lopes@inf.puc-rio.br, hcvlopes@gmail.com (H. Lopes), karin@inf.puc-rio.br (K. Breitman), vicente.mundim@corp.globo.com (V. Mundim), wandenberg@corp.globo.com (W. Peixoto).

0166-3615/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.compind.2013.11.005

This is the case in breaking news, where the content of interest is very volatile, and may become obsolete in a manner of minutes after being posted [17,18]. In cases such as this there are two main challenges: one is of producing relevant recommendations in real-time, and the other is to deal with scalability. This scenario adds an element of complexity to the recommendation models used by Amazon and Netflix, for example, which deal with less volatile items.

An ideal recommendation engine is capable of both scaling up, as the number of items and/or users grows, and producing a relevant recommendation in as small a time as possible (almost real-time).

One approach for scaling up the recommendation engine is through input data sampling [19]. With this alternative it is possible to produce recommendations within a reasonable amount of time, using available, but often limited, computational resources. This process, of course, tends to reduce the accuracy of the recommendation, which may not be relevant to the user in question. Therefore, it is highly desirable that all available data is used to produce the best recommendation possible [2].

The ability to contract computing resources on demand, from a cloud computing vendor, lifts the resource availability limitation, and thus enables the development of a high quality solution, where all available input data is used to produce recommendations, while at the same time, allowing for the system to be scaled up (or down) to adapt to fluctuations in demand [20–23]. It is important to note, however, that cloud computing environments provide the necessary resources to guarantee that all computation can be done. What is not guaranteed is that the computation can be done time effectively, i.e., producing real-time recommendations irrespective of the numbers of items and users in question. This is a challenge and the major motivation of this work.

1.1. Contributions

The main contribution of the paper is a real-life implementation that combines state-of the art recommendation techniques to emerging cloud computing technology in order to provide a scalable, cost, and time efficient solution of a real-time online recommendation system. The proposed system architecture provides a solution for computing online item similarity, without having to make use of either model simplification or input data sampling [2]. Our proposal makes use of on-demand cloud computing resources to scale up and down, and adapt to variations in the number of items as well as users, thereby meeting the needs of large content portals, i.e., those dealing with several million hits on a daily basis and trading with catalogs with millions of different items. To validate this architecture, we implemented and tested it in a real production scenario of video recommendations on the Globo.com portal. Globo.com is the Internet branch of Globo Organizations, the largest media group in Latin America entertainment industry, and the leader in the broadcasting media segment for Brazilian Internet audiences. The results from this implementation show that it is possible to greatly reduce recommendation times with low financial costs.

1.2. Paper outline

In the next section, we introduce the key concepts, proposed by related work, to generate item–item recommendations through collaborative filtering, and the challenges associated with this process. Section 3 presents the proposed architecture and describes how the issues associated with such process are addressed. Section 4 discusses how the proposed architecture was deployed in the cloud in a real production environment, and the results obtained with those tests are discussed in Section 5. Section 6 compares our work with related ones. Finally, Section 7 presents the conclusions, discussing further works.

2. Item–item recommendation

It is common in collaborative filtering systems to store user profiles as vectors of items and ratings [8]. These vectors tend to increase continually as users interact with the system. Some systems take into account temporal dynamics to discount the standard deviations in user interests over time [24,25]. User feedback may be binary, e.g., liked or not liked, or valued according to preference. Netflix [4–6], MovieLens [26], and Amazon [3] are among those that adopt the latter approach.

The two most common ways of implementing a solution for collaborative filtering are the use of neighborhood algorithms or graphs, and latent factorization models [11].

Neighborhood algorithms focus on extracting the relationship between users or items to build a model based on a graph capable of describing these adjacencies.

- Item–item methods build a neighborhood graph with vertexes connecting similar items.
- User–user methods build a neighborhood graph with vertexes connecting users with similar preferences.

The idea behind an item–item recommendation engine is, for any given item, finding a set of items that is most similar to the item in question. Similarity is measured using a combination of input data, generally structured in a bi-dimensional matrix with the first dimension representing users and the second items [8]. The ultimate goals of an item–item recommendation system are to predict how users would rate an item that is not yet rated, and to recommend items from the collection.

With respect to user ratings, recommendation engines may use different types of feedback. Ideally, explicit feedback is preferred, with users explicitly indicating their preferences. However, in most situations, such information is not available. In these cases, it is necessary to infer user preferences through implicit feedback [27], which indirectly represents a user's preferences based on his/her behavior. Implicit feedback is obtained from user navigation history, the list of products bought, and even from the mouse path on specific screens.

After obtaining user feedback, for example, information that a user accessed a particular item, the input can be stored in a square matrix representing users and items, where, in this case, each element denotes whether an item was accessed by a user. Thus, considering each item separately, we have a multidimensional vector with each dimension representing a user. Consequently, as the number of distinct users in the system increases, the number of dimensions in the vector also increases. Typically, large e-commerce or content portals have tens or even hundreds of millions of unique visitors per month, which means that their item vectors will have a similar number of dimensions.

Therefore, the problem of obtaining the similarity between two specific items can be translated into one of obtaining the similarity between the two multidimensional vectors representing the items. One of the possible approaches for doing this is by calculating the cosine between these vectors [8]. Let us suppose that two items, say I_1 and I_2 in $\mathbb{R}^M$, where M is the number of users, then the similarity between these two items are calculated as follow:

$$Similarity(I_1, I_2) = \cos(\angle(I_1, I_2)) = \frac{\langle I_1, I_2 \rangle}{||I_1|| \cdot ||I_2||}, \quad (1)$$

where $\angle$, $\langle \cdot, \cdot \rangle$ and $|| \cdot ||$ represent, respectively, the angle between the two vectors, the usual inner product in $\mathbb{R}^M$ and the Euclidian vector norm.

Having calculated the similarity between a specific item and all others, one can then obtain the items that are most similar to one