1 Introduction

The ability to offer relevant suggestions to users is becoming extremely relevant 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, 5, 6].

Recommender 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, 12, 13, 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, here 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. 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 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, 21, 22, 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, and was detailed described in a published Computers in Industry journal paper [70].

Other important aspect is related with users' feedbacks. In the same scenario of breaking news, or even sports and some entertainment shows, the explicit user feedback about an item may be imprecise, which could leverage to poor recommendations. This happens because the user feedback is subjective, and every different user could have a different interpretation of which aspects should be taken into account when rating content. To avoid this, an implicit feedback method can be used, in order to objectively evaluate user interest over a specific item. Find the best implicit feedback model is also challenging, since each type of content could have a different model that maximizes recommendation efficiency. This work is also detailed described in our published workshop paper from ACM Recommender Systems [71].

1.1. Goals

The main goals of this thesis are:

- provide a solution for computing item similarity, without having to make use of either model simplification or input data sampling;
- make 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; and
- allow for the stream processing of user feedbacks, in real-time, continuously updating item similarity graph in order to always have up-to-date recommendations, which is crucial where content interest is ephemeral, such as breaking news.

This Thesis also proposes a new implicit feedback model for short-lived videos, for example the ones related to sports and news, which makes usage of the proposed architecture to retrieve real-time recommendations of such content.

1.2. Main Contributions

This thesis proposes an architecture for collaborative filtering that introduces the following contributions:

- identify and characterize the class of problems of large scale collaborative filtering for item-item recommendations, specially in scenarios where content interest is very volatile;
- a new architecture that can tackle scalability issues of performing collaborative filtering considering dozens of millions of objects;
- a new architecture that can continuously process feedbacks and updates similarity graphs in order to have real-time recommendations;
- a new implicit feedback model for videos which users interest is very ephemeral;
- a new collaborative filtering algorithm that allows the parallel and distributed processing of tasks and to be able to make use of the elasticity provided by the cloud computing platforms, adjusting the amount of available resources according to demand; and
- show the feasibility of proposed architecture and algorithm in a real-life implementation scenario;

Those contributions were also published in a journal paper from Computers in Industry [70] and in a workshop paper from ACM Recommender Systems [71].

1.3. Related Work

The use of recommendation algorithms is becoming a trend in todays Internet businesses, playing an important role in user experience and product commercialization, and most recommendation algorithms start by finding a set of customers whose purchased and rated items overlap the users purchased and rated items [28].

One of the most successful practical implementation of recommendation algorithms is from Amazon, and their approach is described by Linden et. al. [2]. This work is particularly interesting because of the scale of users and items used in the recommendation model. Linden et al. state that Amazon.com has more than 29 million customers and several million catalog items, which is similar to the numbers that our work is focused. However, their work relies to offline computation, reduction or sampling to scale a collaborative filtering algorithm for item-item recommendations: Traditional collaborative filtering does little or no offline computation, and its online computation scales with the number of customers and catalog items. The algorithm is impractical on large data sets, unless it uses dimensionality reduction, sampling, or partitioning all of which reduce recommendation quality. This thesis fills this gap describing a practical implementation of an online collaborative filtering for large dataset, using Cloud Computing.

Davidson et al. [57] also discussed the challenges to work with large datasets, describing the YouTube video recommendation solution, which chooses a batch-oriented pre-computation approach instead an online calculation of recommendations. According Davidson et al., this has the advantages of allowing the recommendation generation stage access to large amounts of data with ample amounts of CPU resources while at the same time allowing the serving of the pre-generated recommendations to be extremely low latency. However, Davidson et al. highlights that the most significant downside of this approach is the delay between generating and serving a particular recommendation data set.

Kim et al. [67] propose a new collaborative tagging technique that produces good improvement on recommendation quality. To do so, the user has to tag each viewed item. This solution is not suitable for a real-time application on the scale of our experiment because of the idea to add a new dimension on the user-item matrix, which would be the tag dimension, increasing significantly the computation complexity to perform recommendations for large datasets. The architecture present in this thesis chose to provide a solution for a practical industrial large problem.

Koren [68] introduces a new neighborhood model with an improved accuracy on par with recent latent factor models. At the same time, the model retains fundamental advantages of neighborhood models. The paper describes that past

models, such as Linden et al. [2], were limited by the need to compute all pairwise similarities between items or users, which grow quadratically with input size. According to Koren, this limitation vastly complicates adopting user similarity models, due to the typical large number of users. Their proposed model solves these limitations by factoring the neighborhood model, while our approach solves Lindens limitation by a rational and efficient usage of Cloud Computing. Furthermore, Koren did not present experimental results and benchmark using very large datasets, with dozens of millions of users and items.

Yang et al. [69] address data sparsity and difficulty in scalability of traditional collaborative filtering approaches by using incremental update and local link prediction. However, there is a lack of experimental results and benchmark using very large datasets, since the presented experiment is limited to less than 20000 objects.

1.4.Outline

This thesis is organized as follows. Next chapter presents background information about recommender systems, and cloud computing, highlighting important concepts required for the comprehension of the proposed work, such as collaborative filtering, user feedback and cloud computing models and platforms. Chapter 3 discusses the challenges associated with large scale collaborative filtering, and point some works in this area developed by large video services such as YouTube and Netflix.

Chapter 4 presents the proposed architecture to tackle the main challenges of large scale collaborative filtering, through the usage of cloud computing platforms. It also gives details of architecture implementation, and how real-time processing is feasible even with large datasets. Chapter 5 presents another contribution of this thesis, the model to extract implicit feedback that represents the user interest in a specific video. Chapter 6 highlights the implementation of the architecture in a real environment, and presents performance results of such implementation. Finally, Chapter 7 presents related work with the conclusions in Chapter 8.