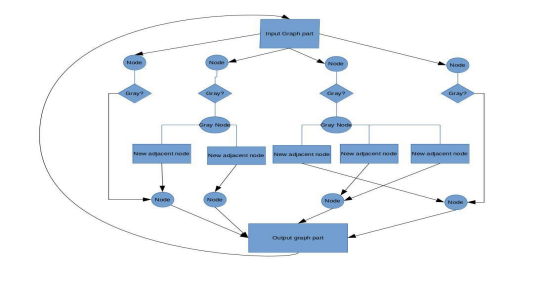
A personal recommender system based on an unweighted graph

Davide Lissoni Daniele Dellagiacoma
University of Trento

Problem & Solution

Nowadays, many different system can usefully be represented as graphs. A key feature of these systems is provided by personalized recommender systems for information retrieval and content discovery in today's informationrich environment. Usually modern **recommender systems** use complex technique to provide advices to each user. We thought about a different way to provide recommendations to users of a networks. Our program aims to visit a unweighted graph starting from a set of nodes (also called keynodes) to find out which nodes can be reached by the keynodes within a adjustable maximum distance. It has been implemented using **MapReduce** in order to handle big datasets.

We decided to implement our algorithm on a **BFS** structure. Starting from the graph given as input, map method check for each node if its color is gray (at the first iteration only the keynodes would be gray), the method creates as many new nodes as the number of neighbor node, the mapper is also in charge to set the values of the new node created, based on the parent-node's information.
The reducer job is to merge all the mapper results according to the value of the key that in our case is the node ID. The node information will be set by the reducer, choosing the best values founded in all the nodes given by the mapper.
This process will be repeated until the program has achieved the convergence or the maximum distance.
The algorithm is able to work in parallel using more reducers.



Today's systems

Often recommender systems use complex technique to provide advices to users. The advices can be based on measurements of similarity between items or users. Existing recommender systems can be categorized into two different groups: **content-based** and **collaborative filtering**

Content-based

recommends items to a specific user, similar to previous item rated highly by him/her. In this case every item is represented using a vector of features where value of a feature can be established by **TF-IDF**.

$$TF_{i,j} = \frac{n_{i,j}}{|d_j|} \qquad IDF_i = log\frac{|D|}{n_i}$$

n is the number of a term in a document **d**, **D** is the number of documents

$$(TF - IDF)_{i,j} = TF_{i,j} * IDF_i$$

Collaborative filtering

the system builds a database of preferences for items by users. A new user is matched against the database to discover neighbors, which are other users who have historically had similar taste to him/her. Items that the neighbors like are then recommended to the user, as he/she will probably also like them.

Model-based
allow the system to learn to recognize complex patterns based on the training data
Memory-based
uses user rating data to compute the similarity between users or items
Hybrid
combines the memory-based and the model-based CF algorithms

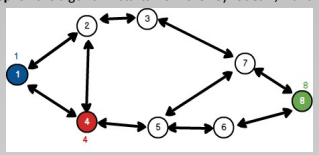
Idea

We provide an alternative way to marking suggestions in different kind of systems (i.e. social network, e-commerce, etc) where the structure of the system is described only as a graph. We used **Hadoop** MapReduce in order to handle big datasets.

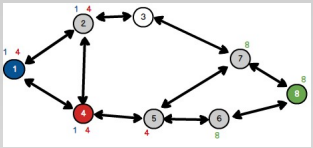
Our model needs a **graph** and a set of **keynodes** as input. Starting from the keynodes to find out which nodes can be reached by the keynodes within a adjustable maximum distance

This implementation could be a useful and simple model used to provide advices to users of a network. Keynodes may represent items or users of the network, this allows to find out items/user reachable within a certain distance. More keynodes reach a node, more that node can be considered **interesting**. The algorithm can be used in many **different ways** to find out the information whom you are looking for.

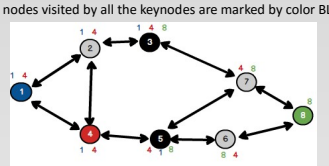
Step 0: the algorithm starts from the keynodes 1, 4 and 8



Step 1: each neighbor of the keynodes has been visited and is marked by color GRAY.



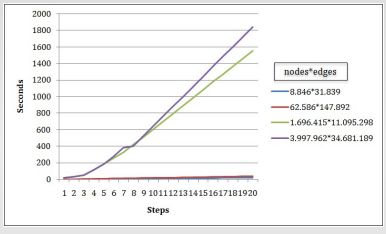
Step 2: nodes visited by all the keynodes are marked by color BLACK.



In addiction each node memorizes which keynodes has been reached it and the **minimum distance** from its closest keynode.

Result

We tested our algorithm using graphs with different number of nodes and edges.
The MapReduce algorithm is more affected by the number of edges than the number of nodes O(iterations*edges), but with a less number of nodes, the single iteration time is minor.



The **total time** for perform the entire process of the algorithm depends on the number of input keys and the file size (dimensions and complexity of the graph) given as input, but the time to perform a single iteration, depends only on the file size taken by the processed iteration, so on the number of gray nodes that it has to process. The algorithm is really performed until the number of nodes processed in a step is about **500.000**.
In a bigger scenario the numbers of keynodes and steps have to be chosen carefully for a optimal performance of the algorithm. This means that the graph have to be known and analyzed before runnning the algorithm in order to do a smart input-choice.