Prop\_ronzonip\_IT.doc

**An empirical analysis of collaborative filtering’s algorithms**

Author: Paolo Ronzoni

Stipulation

The text of this proposal is my own, except where explicitly indicated. I give my permission for this essay to be submitted to the JISC Plagiarism Detection Service.

**Introduction**

This project test different collaborative filtering algorithms, compare the results obtained and analyze how these algorithms can be useful to improve the predictive accuracy of a collaborative recommendation system. The main data set of this project is the *Anonymous Microsoft Web Data Set* (predict areas of the web site a user visited based on data on other areas the user visited).

This is version 1.3 of this proposal dated 5 April 2013.

The author of this proposal is Paolo Ronzoni, a full time MSc IT student.

The project supervisor is Dr. David Weston.

**Technology Review**

***Recommender Systems***

A Recommendation System or Recommender System (RS) is software that provides users with personalized advice about a set of objects belonging to a specific domain (eg music, movies, books). A Recommender System uses the data, stored in its database, which relate to clients and objects, to provide purchasing advice.

The main idea of a recommender systems is to use the information about the behaviour or opinions of previous clients to predict what product the current customer will purchase or to predict which product, the current customer, will be interested in.

Nowadays, the recommender systems are widely adopted by enterprises, especially large retail firms.

***Types of recommendation systems***

Recommender systems can be divided into four different types:

* Collaborative recommendation
* Content-based recommendation
* Knowledge-based recommendation
* Hybrid recommendation approaches

The first type of recommendation systems is represented by collaborative recommendation. The main idea behind ​​collaborative recommender systems is to use information about the past behaviour of customers, or opinions of a community of users that already exists, in order to predict which products, the current user of the system might be interested in.

The second type of recommendation systems is represented by content-based recommendation. These RS use a series of discrete features of the product in order to recommend alternative products that possess similar characteristics. These systems use neither the past behaviour of customers nor a range of opinions (ratings). Rather, these systems rely on other sources of data to make correct predictions.

The third category of recommendation systems consists of knowledge-based recommendation. These RS are used in all those situations in which collaborative and content-based systems do not provide satisfactory results. That is, those cases in which the product is not purchased frequently: think, for example, buying a house or a car. In such situations, the collaborative and content-based systems do not work due to the limited number of available ratings and due to the wide interval of time between a purchase and the next: if there is a lot of time between the purchase of two similar products can also significantly change the preferences of the customer due to, for example, of changes in income or in the family composition. In such scenarios, knowledge-based systems are used. These recommendation systems use some explicit rules: the user must specify the requirements and the system tries to find the best solution. If it is not possible to find a solution, the user must change the list of requirements.

The last category of recommendation systems consists of "Hybrid recommendation approaches". The name of these recommendation systems comes from the latin word "hybrida": an object made by combining two different elements. The Hybrid recommendation systems try to mix, in a single system of recommendation, approaches and algorithms used by the previous three recommender systems that have been mentioned.

***Collaborative recommendation systems***

The collaborative recommender systems are the most common category of recommender systems. Thanks to the relative ease of implementation they are used both by the large sites of e-commerce and Internet companies of small and average size. The algorithms that use the collaborative recommendations systems are well-understood and applicable to many domains (movies, books, songs, ...). The basic assumption of these RS is that user provides, either implicitly or explicitly, a rating of the products in the catalog. The effectiveness of these algorithms is based on the idea that "Customers who had similar tastes in the past, will have similar tastes in the future"

In this project, I will limit my analysis to the collaborative filtering algorithms. This choice is dictated by a number of considerations. First, I can have access only to data that allow the analysis of collaborative recommendation systems. If I had to analyse, for example, a content-based recommender system, I should use a database containing a list of discrete characteristics of the product. The free access to these databases is precluded to me or in any case very difficult to achieve. Second, the wide range of recommender systems has generated a very large number of algorithms. Due to the limitations of time and resources, I must necessarily limit the number of algorithms that intend to analyze. I decided to focus on collaborative algorithms because the abundant literature developed on these algorithms will allow me to develop them, properly, in Java, and to test them using the metrics that have been developed in recent years (Gunawardana et al. 2009). Last but not least, the collaborative filtering algorithms are extremely popular and represent the type of algorithms more used in most recommender systems (Su et al. 2009, Herlocker et al. 2004).

**Main Recommender Systems' Algorithms**

My analysis will focus on the following methodologies adopted widely in recent years for the creation of collaborative filtering:

* User-based nearest neighbor recommendation;
* Item-based nearest neighbor recommendation;
* Probabilistic recommendation approaches;
* Slope one predictor.

**User-based nearest neighbor recommendation**

This methodology uses a database containing the ratings of the products; while, as input, uses the choices made by the current user of the system by identifying other users, called peer users or nearest neighbors, characterized by preferences similar to those that the current user has shown in the past (Jannach et al. 2011).

At the basis of this methodology there are two main assumptions:

* if the users of the system had certain tastes in the past they will have the same tastes in the future;
* the preferences of users remain stable and consistent over time.

We can express these concepts in mathematical terms.  to denote the set of users,  to denote the set of products (items) and R as a *n x m* matrix of ratings  with .

The possible rating values ​​can be defined, for example, on a numerical scale from 1 (strongly dislike) to 5 (strongly like). If a particular user *i* did not assess an item *j*, the corresponding value in the array  is empty.

A most widely used measure in practice, by the nearest neighbor recommender systems, to represent the set of similar users is the Pearson's correlation coefficient. The similarity sim (*a, b*​​) of users a and b, given a rating matrix R is defined by the following formula (where corresponds to the average rating of the user a):



The Pearson's correlation coefficient takes values ​​between +1 (strong positive correlation) to -1 (strong negative correlation) (Jannach et al. 2011).

**Item-Based nearest neighbor recommendation**

The approach "User-based nearest neighbor recommendation" is used successfully in several Internet companies, but in certain scenarios when you need to handle millions of users and you need to manage the ratings from a catalog composed of millions of products, this approach has limitations. For this reason the e-commerce sites of greater size prefer to adopt another technique called "item-based nearest neighbor recommendation" (Jannach et al. 2011).

The main idea of this approach is to make predictions using the similarity between the products rather than the similarity between users. To find similar products, there must be defined a measure of similarity. The item-based approach uses, as a standard metric, the cosine similarity. This metric measures the similarity between two *n*-dimensional vectors based on the corner between them.

The similarity between two items *a* and *b* is calculated with the following formula:



Let U be the set of all users who value the items *a* and *b*. The adjusted cosine measure is calculated in this way:



It is possible to predict the rating for the user *u* of the product *p* in this way:



**Probabilistic recommendation approaches**

To predict the rating that a given customer will assign to a certain product several techniques developed within the theory of probability can be used. One of the most well-known methods is based on Bayes classifiers (Jannach et al. 2011).

This technique uses conditional probability. To calculate this we use Bayes' theorem, which allows to calculate this posterior probability  through the class-conditional probability , more formally:



Under the assumption that the attributes (the ratings users) are conditionally independent, we can calculate the posterior probability for each value of Y with a naive Bayes classifier, d being the number of attributes in each X:



**Slope one prediction**

The original idea of the "Slope One" predictors is simple: it is based on what the creators of this technique called "differential popularity."

The main idea of this method is to find the functions in the form:



This formula is used to predict, given a pair of items, the rating of an item starting with the rating of the other item.

A series of experiments have shown that this simple predictor often outperforms a linear regression and has the advantage of having approximately half of the repressors. The slope one prediction has the further advantage of reducing storage requirements and latency (Lemire et al. 2005).

**Problem Statement**

The objective of this project can be stated as follows:

Given the four methodologies of collaborative filtering, described in the previous paragraphs, create several java algorithms to mine the "Anonymous Microsoft Web Data Set" and eventually, if time permits add other database to test the algorithms.

Compare the results obtained and analyze how these algorithms can be useful to improve the predictive accuracy of a collaborative recommendation system

**Solution Description**

There are two types of metrics to evaluate a Recommender System: measures to assess the quality and measures to assess performance. The first analyzes the ratings, the second analyzes the RS as system.

In my project, I will confine my analysis to the metrics used to evaluate the quality.

Among the measures to evaluate the quality of ratings of a RS we can distinguish the accuracy (Su et al., 2009), and the coverage (Vozalis et al 2003)

***Accuracy Evaluation Metrics of Recommendation Systems***

Accuracy: There are many techniques designed to quantify the accuracy. The first category of techniques includes the mean absolute error (MAE) and root mean squared error (RMSE), the second category includes the ROC sensitivity.

Mean Absolute Error (MAE): one of the methods to determine the accuracy (or vice versa, the classification error), is the Mean Absolute Error, which calculates the average of the absolute difference between the ratings provided by the RS and the assessment that subsequently the user expresses.

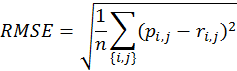


where *n* is the total number of rating of all users, is the expected rating for the user *i* on the item *j*, and is the actual rating. The lower the value of the MAE, the better the prediction of the rating.

Sometimes it is useful to normalize this value relative to the scale used in the RS specific, so you can then compare systems with minimum rating () and maximum () different. In this case one calculates the normalized MAE (NMAE):



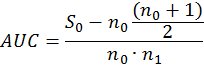
Root Mean Squared Error (RMSE):



where *n* is the total number of rating of all user,  is the expected rating for the user *i* on the item *j* and  is the actual rating.

ROC sensitivity: is a measure of the power of prediction of an RS. Operationally is given by the Area Under the ROC Curve (AUC), literally the area under the curve ROC.

In reality, the value of AUC is the area under the ROC curve only if the prediction is a problem with two possible outcomes (binary problem). In general, the ROC sensivity is calculated with the following formula:



where  and  are, respectively, the number of wrong predictions and correct predictions, and , where  is the rank of the *i* th positive outlook on the list of predictions, ordered by accuracy.

***Coverage Evaluation Metrics of Recommendation Systems***

Coverage: The coverage is a measure of the percentage of items for which the algorithm that generates the ratings can solve the problem recommendation. In some cases, for example due to the sparseness of the user-item matrix, the system is not able to produce a prediction for the rating of many items. In such situations the value of the coverage is certainly low.

Assuming that are the items for which the user has provided an evaluation and is the number of items for which the RS is able to generate a prediction of the rating, then coverage is given by the formula:



where m is the number of rows and columns of the user-item matrix (*m x m*)

**Project Execution**

***Technologies***

The project will use these software tools:

* Java

**Timeteble**

|  |  |
| --- | --- |
| Period | Activity |
| 15 June - 30 June | Review the entire literature, included in the appendix, on the collaborative filtering algorithms and the evaluation metrics. |
| 1 July - 15 July | Writing in java the algorithms related to:   * User-based nearest neighbor recommendation * Item-based nearest neighbor recommendation; |
| 16 July - 30 July | Writing in java the algorithms related to:   * Probabilistic recommendation approaches; * Slope one predictor. |
| 1 August - 15 August | Develop in Java the accuracy evaluation metrics and test the algorithms. |
| 16 August - 31 August | Develop in Java the coverage evaluation metrics and test the algorithms. |
| 1 September - 15 September | Writing the final version of the project. |

**Conclusions**

In this project will test different collaborative filtering algorithms, compare the results obtained and analyze how these algorithms can be useful to improve the predictive accuracy of a collaborative recommendation system. I will limit my analysis to the collaborative filtering algorithms. I decided to focus on collaborative algorithms because the abundant literature developed on these algorithms will allow me to develop them, properly, in Java, and to test them using the metrics that have been developed in recent years. In this project, I will confine my analysis to the metrics used to evaluate the quality of e Recommender Systems.

**References**

The following books and papers were reviewed during the compilation of this project proposal:

John S. Breese, David Heckerman, Carl Kadie, “Empirical Analysis of Predictive Algorithms for Collaborative Filtering,” in *UAI’98 Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence.*

http://research.microsoft.com/apps/pubs/default.aspx?id=69656

Robin Burke, “Knowledge-based recommender systems,” www.cs.odu.edu/~mukka/cs795sum10dm/Lecturenotes/Day6/burke-elis00.pdf

Derek Hamilton-Gibbs, Don Esslemont, and Dalton McGuinness, “Predicting the Demand for Frequently Purchased Items,”in *Marketing Bulletin*, 1992, 3, 18-23, Article 3. http://marketing-bulletin.massey.ac.nz/V3/MB\_V3\_A3\_Esslemont.pdf.

Asela Gunawardana and Guy Shani, “A Survey of Accuracy Evaluation Metrics of Recommendation Tasks,” in *Journal of Machine Learning Research* 10, 2009, 2935-2962.

http://dl.acm.org/citation.cfm?id=1755883

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl, “Evaluating Collaborative Filtering Recommender Systems,” in *ACM Transactions on Information Systems*, Vol. 22, No. 1, January 2004, Pages 5–53.

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.78.8384

Jonathan L. Herlocker, Joseph A. Konstan, and John T. Riedl, “Explaining Collaborative Filtering Recommendations,” in *CSCW '00 Proceedings of the 2000 ACM conference on Computer supported cooperative work*, Pages 241-250.

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.252

Dietmar Jannach , Markus Zanker, Alexander Felfering, Gerhard Friedrich, “Recommender Systems. An Introduction,” *Cambridge University Press,* 1st edition 2011, ISBN. 978-0521493369.

Daniel Lemire, Anna Maclachlan, “Slope One Predictors for Online Rating-Based Collaborative Filtering,” in *SIAM Data Mining (SDM'05)*, Newport Beach, California, April 21-23, 2005

http://arxiv.org/abs/cs/0702144.

Prem Melville, Raymond J. Mooney and Ramadass Nagarajan, “Content-Boosted Collaborative Filtering,” in *Proceedings of the SIGIR-2001 Workshop on Recommender Systems*, New Orleans, LA, September 2001.

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.6429

Prem Melville and Vikas Sindhwani, ”Recommender Systems,” *in Encyclopedia of Machine Learning*, Claude Sammut and Geoffrey Webb (Eds), Springer, 2010. http://www.prem-melville.com/.

Xiaoyuan Su and Taghi M. Khoshgoftaar, “Survey of Collaborative Filtering Techniques,” in *In Advances in Artificial Intelligence archive* Volume 2009; Hindawi

Publishing Corp. New York, NY, United States.

http://portal.acm.org/citation.cfm?id=1722966

Emmanouil Vozalis, Konstantinos G. Margaritis, “Analysis of Recommender Systems’ Algorithms” presented at the *Sixth Hellenic-European Conference on Computer Mathematics and its Applications* (HERCMA), Athens, Greece, 2003, pp. 732-745.

http://www.mendeley.com/research/analysis-of-recommender-systems-algorithms-1/