Technical Report of Project 2 for ADS

Zhao Leiyu
School of Software
Tsinghua University
Building No.1, Zijing Student Apartment
+86 18811368521
zly.george@163.com

Liu Junlin
School of Software
Tsinghua University
Building No.1, Zijing Student Apartment
+86 13552172829
frankliujunlin@163.com

ABSTRACT
In this report, we mainly present our implementation of an article recommender system including its mechanisms incorporating Content-based Method and Collaborative Filtering, and its good performance on data sets extracted from those given by teaching assistant. Before final version, we have done lots of experiments most of which turn out to be failures. These are also mentioned in the report.

Keywords
article recommendation system, content based, collaborative filtering

1. INTRODUCTION
Recommender system is a widely used kind of intelligent agents for recommending items of potential interest to certain user. Generally, recommender systems can be divided into two categories in terms of the methods they adopt, content-based (CN) and collaborative filtering (CF) [1]. In some other classification, demographic and knowledge-based ones are also independent categories [2]. According to some research, there are no effective ways to evaluate and compare the performance of existing recommender systems [3], but empirical studies reveal that at most times, pure CN and CF methods are not good enough to be satisfactory.

As a consequence, a hybrid method incorporating both CN and CF are often required to improve the accuracy of recommendation. Similarly, hybrid methods can also be divided into several categories, including weighted, switching, mixed, feature combination, feature augmentation, cascade and meta-level. Among these categories, feature augmentation is an excellent one [2].

On the basis of above observations and after times of trial and error, we finally build our recommender system using feature augmentation hybrid method with decent performance.

2. MECHANISMS OF OUR SYSTEM
2.1 Tf-Idf
The first key technique in our recommender system is Tf-Idf which is commonly used to extract feature words from articles. Tf-Idf is known as an important component of content based methods. Due to its commonness, we do not present its calculation here anymore. After feature words have been extracted from an article d_j and their Tf-Idf values calculated, it can be reorganized into a vector form:

$$(w_{1j}, w_{2j}, w_{3j} \dots w_{nj})$$

Where w_{ij} is the value of Tf-Idf of the i-th word for d_j. Since a single article contains a few of all the words, the vector could be very sparse. We could only record non-zero values to reduce spatial and temporal complexity.

Next, we can measure similarity between d_i and d_j by calculating inner product of their vectors. By doing so for any two articles, we get a matrix indicating similarities between all the articles, which is then used in collaborative filtering part.

Note that before calculating Tf-Idf values, a library of stop words is imported to eliminate the effect of those meaningless words such as I, for, to, etc.

2.2 Item-based Collaborative Filtering
According to the paper [3] as well as the patent [4][6] proposed by Amazon, it is verified that pure CF method based on items performs slightly better than the one based on users, which is also confirmed in our experiments (see Chapter 4 below). Moreover, this item-based method have other advantages:

Since this method mostly based on a similarity matrix and the value can stay relatively unchanged regardless of what method to be implemented in the recommending procedure, it can be computed off-line and then stored in the disk, which can save a lot of time in massive trial and errors.

Also, the format is closer to the output of Tf-Idf measurement, both of which are $M \times M$ matrix (where M is the number of articles in database). This two measurement of similarity can be mixed easily, thus transforming into a hybrid one.

2.3 Similarity Measures
After trials, we adopt the method proposed in *Recommender Systems Handbook* [5]. In this book, the similarity between article i and j is defined in the following formula:

$$s_{ij} = \frac{\Sigma_{u\in U(i,j)}(r_{ui} - b_{ui})(r_{uj} - b_{uj})}{\sqrt{\Sigma_{u\in U(i,j)}(r_{ui} - b_{ui})^2 \cdot \Sigma_{u\in U(i,j)}(r_{uj} - b_{uj})^2}}$$

Where r_{ui} denotes the rating of article i given by user u, b_{ui} the standard value of ui and $U(i,j)$ is the set of users who rated both i and j. In our case, unfortunately, due to the absence of rating information (all we can get from training data is whether a user like an article), $r_{ui} \equiv 1, b_{ui} \equiv 0$. Then the formula is simplified to

$$s_{ij} = \frac{|U(i,j)|}{\sqrt{|U(j)||U(i)|}}$$

and in practice, in order to eliminate the symmetry, we choose another value as denominator.

$$s_{ij} = \frac{|U(i,j)|}{|U(j)|}$$

It turns out to be better than the former formula.

2.4 Hybrid Similarity

Now, we've got two measurements of similarity between articles. Having tried several method described in [2], we adopt the feature augmentation method. Precisely, we use the method below:

$$\widetilde{s_{ij}} = s_{ij} + \left(\frac{d_{ij}}{d_{ii}}\right)^{\alpha}$$

Where d_{ij} denotes similarity between article i and j calculated by inner product of their tf-idf values. Since it is obvious that d_{ii} is the maximum of all the d_{ij}s, the ratio is used to normalize content-based measurement. Also, constant α is to balance the portion of the two, which is finally assigned 0.8 in practice.

2.5 Dynamic Learning

Given the insufficiency of collection records and the absence of users' rates, collaborative filtering performs relatively worse. To make up for this, we use dynamic learning: adding the recommended articles to corresponding user's collection and then modifying the item-item similarity matrix each time.

Dynamic modifying takes a great amount of time. Fortunately, because the s_{ij} relies on only two factors, which can be stored in the memory at an acceptable complexity. By reducing the unnecessary memory cost to the minimum, we manage to revise our algorithm.

3. PERFORMANCE AND RESULTS

3.1 Performance Test Benchmarks

Since there is no direct measure to estimate the performance of every algorithm, we use cross-check to set up two benchmarks and then test every one of our trails.

In detail, we randomly delete 2 collection records of each user from the training data and the test data consists of those deleted records together with randomly generated articles as false-candidates(98 per user). Aware of which two are the right choices, we are then able to calculate the two values as benchmark:

$$R_u = \frac{d_{c1} + d_{c2}}{d_{(1)} + d_{(2)}}$$

Where $d_{(i)}$ denotes the i-th biggest value of the candidates' recommendation index and d_c is the index of right candidate. Obviously, the more, the better. This value is actually measuring the feasibility of single-method similarity to get cascade hybrid (See [2]).

$$p_i = \frac{number\ of\ \text{right candidates located in i}}{all\ users}$$

By checking $p_1 + p_2$ we can get the exact precision of the algorithm adopted.

3.2 Results

After determining two benchmarks, we apply them to measure the massive trials. We pick out some archetypes and show the results as follows:

Table 1. Benchmarks on Different Methods

	$\overline{R_u}$	$p_1 + p_2$
Content-based Bayes	0.508	31.62%
User-based Correlation	0.591	42.75%
Item-based Original	0.570	40.89%
Item-based Revised	0.620	42.11%
CB-CF Cascade	/	40.05%
CF-CB Cascade	/	42.49%
Feature Augmentation	/	48.90%
Feature Augmentation + Dynamic Learn	/	51.03%

Note: 1. the detailed description of algorithms mentioned in the chart will be introduced in the next chapter.

2. Since the first value is used to measure the feasibility for this feature to act as a filter, which is then used for cascading hybrid, it is meaningless for hybrid method. Thus it is ignored in the hybrid algorithms.

4. OTHER TRIALS

In addition to our final implementation of the project, we have also done lots of trials and tried to apply diverse algorithms such as Bayesian Classifier. Unfortunately, they do not seem powerful and are ultimately abandoned.

4.1 Bayesian Classifier

Bayesian Classifier is a potent algorithm in the field of text categorization. It is argued in several papers that Bayesian Classifier is also effective in article recommendation in that articles could be categorized into "like" and "dislike" types for each user. However, after applying Naive Beyasian Classifier and Bernouli Modeled Bayesian Classifier to our implementation, the result does not seem to be satisfactory. A reasonable explanation is that from our training data, we are not able to know what a user dislikes. But in practice, we treat those uncollected articles as disliked ones, which is obviously simple and irrational since some of them are actually collected by users. So it is no wonder that Bayesian Classifier does not produce expected results.

Moreover, we have tried a hybrid method based on Bayesian Classifier, that is, adopt similarity between users from collaborative filtering and then enlarge one's collection by adding his similar users' collections to his with lower weight. To our disappointment again, it does not work.

4.2 User-based Collaborative Filtering

According to the simplest correlation model of user-based CF [7], the similarity between two users is measured by their correlation:

$$S_{mn} = \mathrm{corr}(R_m, R_n)$$

Specially, an article that is not collected by one is seen disliked by him in the calculating process, which may take advantage of the negative-items but has the tendency to give the candidates a relatively low recommendation index.

In practice, it behaves not bad (See figure 1). However, it cannot guarantee the precision of hybrid algorithm. So we choose to give it up.

4.3 Cascading Hybrid

As proposed in the paper [2], there are several ways for two algorithms to get combined. We choose the best two methods – cascading and feature augmentation. CB-CF cascade method pre-eliminates candidates that are unlikely to be the answer in the Content-based algorithm and subsequently use Collaborative filtering to choose 5 articles. Things are similar in the CF-CB cascade method.

In practice, we have tried to use $\overline{R_u}$ benchmark and ranking to determine elimination rules, which turns out, however, to be a failure. No matter whether CF-CB or CB-CF, no remarkable augmentation is observed.

5. CONCLUSIONS

After poring over tens of papers on recommender systems and failing in tens of experiments, ultimately we have accomplished our own one. Our final implementation seems plain, but in fact, it performs pretty well. What is more important is that all the work has been done by ourselves including tens of thousands of lines of codes. The highlight of our implementation is the "dynamic learning" part, not only because no other groups come up with it, but also for its creativeness and attractiveness in industry in recent years.

Yet our implementation is still to be improved. Other methods and algorithms such as LDA and ANN are also potentially great solutions to this problem, which we do not take into consideration.

6. REFERENCES

[1] Gedeminas, A. and Alexander, T. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, Vol.17, No.6, June 2005.

[2] Robin, B. 2007 Hybrid Web Recommender Systems. DePaul University.

[3] Joeran, B., Stefan, L., Marcel, G., Bela, G., Corinna, B. and Andreas, N. 2013. Research Paper Recommender System Evaluation: A Quantitative Literature Survey. Repsys'13 (October 12 2013), Hong Kong, China. DOI = http://dx.doi.org/10.1145/2532508.253212

[4] Linden G, Smith B, York J. Amazon. com recommendations: Item-to-item collaborative filtering[J]. Internet Computing, IEEE, 2003, 7(1): 76-80.

[5] Kantor P B, Rokach L, Ricci F, et al. Recommender systems handbook[M]. Springer, 2011.

[6] Linden G D, Jacobi J A, Benson E A. Collaborative recommendations using item-to-item similarity mappings: U.S. Patent 6,266,649[P]. 2001-7-24.

[7] Breese J S, Heckerman D, Kadie C. Empirical analysis of predictive algorithms for collaborative filtering[C]//Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1998: 43-52.