An Article Recommendation System Based On Collaborative Topic Modeling

Dang Fan[*]
School of Software
Tsinghua University
dangf09@gmail.com

Wang Shuhao[†]
School of Software
Tsinghua University
shudiwsh2009@gmail.com

Ding Peng[‡]
School of Software
Tsinghua University
dingpeng09@gmail.com

ABSTRACT
The aim of this report is to describe the algorithm and the implementation of an article recommendation system, which combines the traditional collaborative filtering and probabilistic topic modeling. This system produces latent vectors for both users and articles, which can be used to form recommendations easily. Besides, it can recommend new articles to users without training.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms
Experimentation, Algorithms

Keywords
recommendation, collaborative filtering, topic modeling

1. INTRODUCTION
With the rapid development of computers and the Internet, data is becoming unprecedentedly rich, based on which recommendation systems have been focused on for years. According to wikipedia[2], three approaches to design a recommendation system are commonly used, which are collaborative filtering, content-based filtering and a hybrid approach.

Collaborative filtering(CF) can be implemented by many different algorithms, for instance, user-based Nearest Neighbor algorithm[1], and matrix factorization[2]. However, CF suffers from three problems, cold start, scalability, and sparity.[2]

[*]Student ID: 2009013215
[†]Student ID: 2009013229
[‡]Student ID: 2009013219

Content-based filtering is another common approach. A few algorithms can be used to build a profile of an item (like an article). For example, topic modeling provides a representation of the articles in terms of latent themes discovered from the collection.

In our work, we implement an article recommendation system according to Wang's result[5]. In this implementation, we combine both the collaborative filtering and the content-based filtering in a probabilistic model.

This paper is organized as follows. In section 2, we describe the collaborative topic model we used. In section 3, we focus on the detail of implementing this system. Section 4 contains the evaluation of our implementation and the conclusion (including the limitation).

2. COLLABROTIVE TOPIC MODELING
2.1 Model
In this model[5], we assume I users and J articles. The rating variable $r_{ij} \in \{0, 1\}$ denotes whether user i includes article j in his or her library.

We represent users and articles in the latent topic space of demension K — user i is represented by a latent vector $u_i \in \mathbb{R}^K$ and article j by a latent vector $v_j \in \mathbb{R}^K$. Then we form the prediction of whether user i will like article j with the inner product between their latent representations,

$$\hat{r}_{ij} = u_i^T v_j. \tag{1}$$

Then we combine the traditional collabrotive filtering — matrix factorization[4] and the simplest topic model — latent Dirichlet allocation (LDA)[3].

Assume that there are K topics $\beta = \beta_{1:K}$. Then we generate users and articles as follows,

1. For each user i, draw user latent vector $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_K)$, where λ_u is a regularization parameter.
2. For each article j,

   (a) Draw topic distributions $\theta_j \sim \text{Dirichlet}(\alpha)$.

(b) Draw article latent offset $\epsilon_j \sim \mathcal{N}(0, \lambda_v^{-1} I_K)$ and set the article latent vector as $v_j = \epsilon_j + \theta_j$, where λ_v is a regularization parameter.

(c) For each word w_{jn},

i. Draw topic assignment $z_{jn} \sim \text{Mult}(\theta)$.

ii. Draw word $w_{jw} \sim \text{Mult}(\beta_{z_{jn}})$.

3. For each user-article pair (i, j), draw the rating

$$r_{ij} \sim \mathcal{N}(u_i^T v_j, c_{ij}^{-1}),$$

where c_{ij} is the precision parameter for r_{ij}.

Here, the precision parameter c_{ij} serves as a confidence parameter for rating r_{ij}. Thus, we let

$$c_{ij} = \begin{cases} a, & \text{if } r_{ij} = 1, \\ b, & \text{if } r_{ij} = 0, \end{cases}$$

where a and b are tuning parameters satifying $a > b > 0$.

Note that we call this the collaborative topic regression (CTR) model.

2.2 Learning the parameters

In this section, we assume that topic parameter β is known after being processed using LDA. Then we focus on how to compute the parameters u_i, v_j and θ_j. According to Wang's result[5], an EM-style algorithm can be used to learn the maximum a posteriori estimates.

First, we compute the complete log likelihood of $U, V, \theta_{1:J}$, and R given λ_u, λ_v and β,

$$\begin{aligned}\mathcal{L} = &-\frac{\lambda_u}{2}\Sigma_i u_i^T u_i - \frac{\lambda_v}{2}\Sigma_j (v_j - \theta_j)^T (v_j - \theta_j) \\ &+ \Sigma_j \Sigma_n \log(\Sigma_k \theta_{jk} \beta_{k,w_{jn}}) - \Sigma_{i,j} \frac{c_{ij}}{2}(r_{ij} - u_i^T v_j)^2.\end{aligned} \quad (2)$$

To maximize this function, we iteratively optimize the collaborative filtering variables $\{u_i, v_j\}$ and the topic distribution θ_j. In order to do this, we firstly use the following calculation to optimize u_i, v_j

$$\begin{aligned} u_i &\leftarrow (V C_i V^T + \lambda_u I_K)^{-1} V C_i R_i \\ v_j &\leftarrow (U C_j U^T + \lambda_v I_K)^{-1} (U C_j R_j + \lambda_v \theta_j) \end{aligned} \quad (3)$$

where C_i is a diagonal matrix with $c_{ij}, j = 1, \cdots, J$ as its diagonal elements and $R_i = (r_{ij})_{j=1}^J$ for user i. For article j, C_j and R_j are similarly defined.

As Wang's result[5] suggests, θ_j cannot be optimized analytically. In his paper, he uses projection gradient, which is able to be found in his paper.

3. IMPLEMENTATION

The implementation of this system can be separated into four parts — preprocessing, LDA processing, CTR processing, and recommending.

3.1 Preprocessing

In this stage, we preprocess each article to generate the input file for the sencond stage.

For each article, we concatenate its title and abstract. Using Word Vector Tool[1], we remove stop words, stem the rest words, and generate tf-idf vectors. We sum up these vectors and choose the top 8000 words as the vocabulary. After that, we reuse Word Vector Tool to get the term counts vector for each article.

The preprocessor is located at the package `cn.edu.tsinghua.AR.preprocessor`.

3.2 LDA Processing

LDA[3] was first presented by David Blei, Andrew Ng, and Michael Jordan. On Blei's homepage[2], a Java-implemented LDA-J project (under GNU GPL v2) from Gregor Heinrich[3] is provided.

We use this package to do LDA processing with some modification required to fit our whole system. The LDA code is located at the package `org.knowceans.lda` and `org.knowceans.util`

After this stage, the parameter matrice β and θ are produced.

3.3 CTR Processing

The author Wang of this paper, according to which we implement the system, released partial code of this algorithm written in C on his homepage[4].

The partial CTR code provided by Wang is under GNU GPL v2. It uses GNU Scientific Library to do calculations on matrice and vectors, which is unavailable in Java. Thus we totally reimplement the CTR algorithm in Java based on his C code. We use JAMA[5] to do calculations on matrice.

The CTR code is located at the package `cn.edu.tsinghua.AR.ctr`. It simply follows the learning strategies described in section 2.2.

After this stage, the latent vectors of users and articles are produced.

3.4 Recommendation

As we described in section 2, we form the prediction with the inner product between the latent vector of user and article in Eq. 1.

We compute each prediction in the candidate library for each user, and output the top 5 result as our recommendations.

The code is located at the package `cn.edu.tsinghua.AR.system`. The program will generate `result.txt` as the required output file.

[1]`http://sourceforge.net/projects/wvtool/`
[2]`http://www.cs.princeton.edu/~blei/`
[3]`http://www.arbylon.net/projects/`
[4]`http://www.cs.princeton.edu/chongw/`
[5]`http://math.nist.gov/javanumerics/jama/`

4. PERFORMANCE AND CONCLUSION

4.1 Evaluation

To evaluate the result, we use the training data provided by TA as the library. Then we randomly choose at least 5 articles collected by each user as the actual training data, and still choose 250 candidate articles in total.

We evaluate our result with two metrics.

The first metric is accuracy. We compute the average accuracy by

$$\text{Average Accuracy} = \frac{\text{articles in the libarary}}{\text{articles recommended}} \quad (4)$$

The second metric is Average Precision at 5(AP@5)[6].

For LDA, we vary $K \in \{25, 50, 75, 100, 125, 150, 175, 200\}$, and set $\alpha = 50/K$.

For CTR, we set $a = 1, b = 0.01, \lambda_u = 0.01, \lambda_v = 100$, and the same K in LDA.

4.2 Comparison

Table 1 demonstrates the two metrics we mentioned above and the running time on a 2.4Ghz CPU computer.

Table 1: The performance of AR

K	Accuracy	AP@5	Running Time
25	2.9831	0.7435	40min
50	3.4597	0.8296	54min
75	3.6293	0.8545	1.8h
100	3.7348	0.8717	3.5h
125	3.7709	0.8794	4.3h
150	3.8080	0.8812	7.0h
175	3.8258	0.8873	9.8h
200	3.8427	0.8881	13.2h

Figure 1,2 and 3 also show the performance.

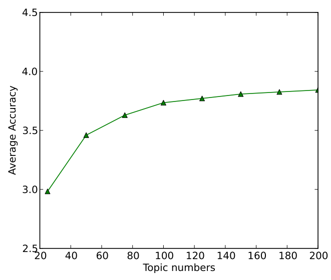

Figure 1: Accuracy

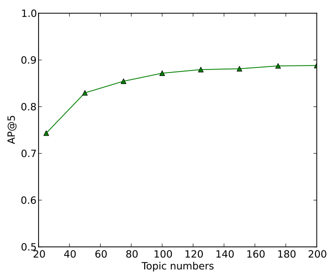

Figure 2: AP@5

From the comparison above, we find that the higher K is, the more precise it is. However, the running time also increases fast. In this case, we think that the result of $K = 100$ is fair.

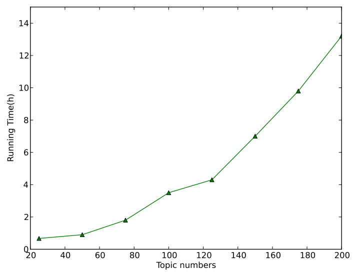

Figure 3: AP@5

4.3 Conclusion

Based on the evaluation, we are able to conclude that the collaborative modeling performs well.

Based on the latent vector and LDA, recommending new articles is also available. LDA can generate inference of a new article, thus we could simply compute the inner product to give a prediction.

However, there are several limitations in this system,

1. The running time is very long.
2. When rating changes, the system has to learn new parameters again.

5. ACKNOWLEDGMENTS

Thanks to Mr Jin for being such good guider for us. Thanks to TA, Mr Ding, for answering our questions.

6. REFERENCES

[1] Collaborative filtering. `http://en.wikipedia.org/wiki/Collaborative_filtering`, 2012.

[2] Recommender system. `http://en.wikipedia.org/wiki/Recommender_system`, 2012.

[3] Julia Hockenmaier, D Blei, A Ng, and M Jordan Latent. Lecture 6: Topic models (latent dirichlet allocation). *Machine Learning*, pages 3–8, 2009.

[4] Y Koren, R Bell, and C Volinsky. Matrix factorization techniques for recommender systems, 2009.

[5] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 448–456, New York, NY, USA, 2011. ACM.

[6] Mu Zhu. Recall, precision and average precision. 2004.