EE447 PROJECT REPORT

RECOMMENDATION AND DEEP USER PORTRAIT BASED ON MOBILE PERSONALIZATION

Jing Hongyi(517030910317) Pang Lianyu(517030910322)
He Wendi(517030910315) Fan Tingyu(517030910312)

June,19th 2020

Abstract

This part is written by Jing Honyi and Fan Tingyu

Nowadays, there are a lot of algorithms of recommendation. Most of them have an immense label library, the recommendation system collects the behavior of users, and map these behaviors to some labels in the library. Everytime when user's behavior is collected, the weight of related labels will be enhanced. In this way, we can get a user profile which describes the interests of users.

There is a certain difference between the projects we do and the traditional recommendation system. The traditional recommendation system will pay attention to the user's preferences and scoring for different products, which is a quantitative method, but there is no scoring condition in our recommendation system, so we prefer the relationship between text processing and natural semantics.

There is a big problem with this approach, which is the cost of constructing such an immense label library. In most cases, like Zhihu and Douban, these labels are collected manually, which takes a lot of time and manpower. The maintenance costs are high, too, because the maintainer of the website needs to add new words to the library at any time. That's where our ideas come from, we want to design a recommendation system which doesn't need such a reluctant label library.

We propose a recommendation system based on clustering algorithm and word vector, these two methods are all unsupervised. We also design a network to demonstrate our recommendation system.

1 User Profiling

This part is written by Fan Tingyu

The generation of user profile is divided into 3 processes: TF-IDF matrix, clustering and word vector. The generation process of user profile is totally unsupervised, which means it doesn't need any manually labeled data, or any form of label libraries.

1.1 TF-IDF

TF-IDF, short for term frequencyinverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling.

1.1.1 Term frequency

In the case of the term frequency tf(t,d), the simplest choice is to use the raw count of a term in a document, i.e., the number of times that term t occurs in document d.

1.1.2 Inverse document frequency

The inverse document frequency is a measure of how much information the word provides, i.e., if it's common or rare across all

documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient).

$idf(t, D) = log \frac{N}{|\{d \in D : t \in d\}|}$

where N is the total number of documents in the corpus $N = |D|$, $|\{d \in D : t \in d\}|$ is the number of documents where the term t appears. In practice, t is usually not in the corpus, which leads to a division-by-zero. Therefore, we usually add $|\{d \in D : t \in d\}|$ by 1

1.2 Clustering

For each article A which is read or commented by a user, We can use TF-IDF to transform it into matrix form, i.e. $A' = TF - IDF(A)$, then store A' in a clustering pool P. We can use K-means algorithm to cluster the pool. The clustering result is some central points which can represent the interest of user. We can select some words in these central pools with high TF-IDF value.

There are 2 problems with this approach. As user behavior continues to be recorded, the size of clustering pool will increase, time spent on EM algorithm will increase. The second is that we didn't take user's long term interest or short term interest into consideration, every article in the clustering pool is equally regarded.

1.2.1 Improvement

We can limit the size of pool, everytime when an article is included, an old one will be excluded. Also, we allocate a weight to each article, to reflect the time that the article stays in the pool. Here we use temperature coefficient $t = \alpha^n$ Where n is the index of the article in the pool, α is a constant $\in \{0, 1\}$. α larger means the system pays more attention to user's long-term interest, α lower means the system pays more attention to user's short-term interest.

Also, we need to update the optimization goal of K-means algorithm. The original goal of K-means algorithm is to minimize:

$\sum_{C_i} \sum_{x \in C_i} \|x - \mu_i\|$

The optimization goal of Weighted K-means is to optimize:

$\sum_{C_i} \sum_{x \in C_i} f(x) \|x - \mu_i\|$

Where $f(x)$ is the temperature coefficient we've mentioned above.

The updated k-means algorithm will be:

E step: $\mu_i = \frac{\sum_{x_j \in C_i} f(x_j) x_j}{N}$

M step: $z_j = argmin \|x_j - \mu_i\|$

2 Word Embedding

This part is written by Jing Hongyi

In natural language processing, we need to quantify the similarity of Chinese words, and transform the corresponding user profiles into a unified word vector form. In short, word embedding technology transforms words into dense vectors, and for similar words, the corresponding word vectors are also similar.In natural language processing tasks, we need to consider how words are represented in computers. Generally, there are two representations: one hot representation and distribution representation.

2.1 One-hot Representation

Traditional natural semantic processing methods based on rules or statistics regard a word as an atomic symbol.It is called one hot representation. One hot representation represents each word as a long vector. The dimension of this vector is thesaurus size. Only one dimension in the vector has a value of 1, and the other dimension is 0,and this dimension represents the current word.

2.2 Distribution Representation

Distributed representation is to transform words into a distributed representation, also known as word vector. Distributed representation represents a word as a continuous dense

vector of fixed length,which makes the concept of "distance" exist between words, which is very helpful for many natural language processing tasks.

2.3 Word Embedding Production

By counting the co-occurrence times of words in a window with a specified size in advance, the number of co occurrence words around the word is taken as the vector of the current word. Specifically, we define word representation by constructing a co-occurrence matrix from a large number of corpus texts. For example, the corpus is as follows:

- I like deep learning.
- I like NLP.
- I enjoy flying.

Then the co-occurrence matrix is as follows:

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Figure 1: Co-occurrence matrix

To some extent, the word vector defined by the matrix alleviates the problem of one hot vector similarity of 0, but it does not solve the problem of data sparsity and dimension disaster.

Since the discrete word vector based on co-occurrence matrix has the problem of high dimension and sparsity, a natural solution is to reduce the dimension of the original word vector, so as to get a dense continuous word vector. For the matrix in Figure 1, SVD decomposition is carried out to obtain the matrix orthogonal matrix U, and the normalized matrix U is as follows:

I	0.24	0.2	0.10	0.38	-0.18	-0.18	-0.42	-0.06
like	0.20	0.823	-0.17	0.31	0.18	-0.23	0.13	0.14
enjoy	0.37	0.64	0.16	0.00	-0.58	0.64	0.00	-0.31
deep	0.36	0.38	0.35	-0.07	0.45	0.08	0.55	-0.47
learning	0.40	0.52	-0.50	-0.43	0.35	0.16	-0.47	-0.40
NLP	0.35	0.35	-0.22	-0.19	0.13	0.49	0.21	0.66
flying	0.41	0.42	-0.40	-0.38	-0.51	-0.43	0.42	-0.12
.	0.38	0.58	0.59	-0.62	-0.03	-0.23	-0.26	0.24

Figure 2: SVD result's U matrix

SVD obtains dense matrix of word, which has many good properties: words with similar semantics are similar in vector space, and even can reflect the linear relationship between words to a certain extent.In our project, we use the pre trained Zhihu language model to better analyze the Zhihu user behavior text.

3 Collaborative Filtering

This part is written by Jing Hongyi and He Wendi

Collaborative filtering algorithm is a well-known and commonly used recommendation algorithm. It is based on the mining of the user's historical behavior data to find the user's preference bias, and predict the product that the user may like to recommend. That is to say, common functions such as "guess what you like" and "people who buy this product also like". Its main implementation consists of:

- Recommend to you according to the people you are alike
- Recommend similar items to you according to your favorite items

Therefore, the common collaborative filtering algorithms can be divided into two types: user based collaborative filtering and item based collaborative filtering. The characteristics can be summarized as "people gather by category, and objects are divided by group", based on which prediction and recommendation can be made.

3.1 User-based

There are two main problems to be solved in the implementation of user based collaborative filtering algorithm. One is how to find people with similar interests, that is, to calculate the similarity of data.To calculate the similarity, we need to choose different similarity calculation methods according to different data characteristics. There are several common calculation methods.Because our user portraits are word vector forms, we mainly use the second method:

(1)Jaccard similarity coefficient:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

(1)Angle cosine,two n-dimensional sample points A $(X_{11}, X_{12}, ..., X_{1n})$ and B $(X_{21}, X_{22}, ..., X_{2n})$:

$$cos(\theta) = \frac{\sum_{k=1}^{n} x_{1k} x_{2k}}{\sqrt{\sum_{k=1}^{n} x_{1k}^2} \sqrt{\sum_{k=1}^{n} x_{2k}^2}} \tag{2}$$

When we are looking for people with the same hobbies, we may find many people, for example, hundreds of people like commodity a, but there may be dozens of people who like commodity B at the same time with you. Their similarity is higher. We usually set a number k, and the K person with the highest similarity is called the nearest K user as the recommended source group body.

We extract a fixed number of keywords from the user's profile we obtained before, and then convert them into word vectors with fixed dimensions. Then we compare the user's profile word vectors with those of other users to obtain the five users with the highest cosine similarity. Then we process the articles concerned by these users and compare them with the user's profile word vectors 20 articles with the highest similarity were recommended.This is the collaborative recommendation algorithm based on users. The summary steps are as follows:

1. To calculate the similarity of other users, you can use the reverse query table to remove some users
2. Find K users similar to your mouth based on similarity
3. Among the items that neighbors like, the recommendation of each item is calculated according to the similarity with you
4. Recommend items based on similarity

3.2 Item-based Collaborative Filtering

Item-based Collaborative Filtering and User-based Collaborative Filtering are two types of collaborative filtering recommendation algorithm which share some similarity and differences between each other.

Since the comparison between user and user could have limited intersection due to the scarcity of the information the user graph reveals, which may probably result in failure of efficient recommendation, so it is helpful to examine and analysis each items of the users.

The following will show how we apply Item-based Collaborative Filtering to make recommendation for users and the improvement of combining the Item-based Collaborative Filtering algorithm with user graph.

One of the major differences for Item-based Collaborative Filtering is that it mainly focuses on each item, instead of each user in User-based Collaborative Filtering. So, we need to compare the similarity between item and item, or in other words articles and articles, instead of the users and users.

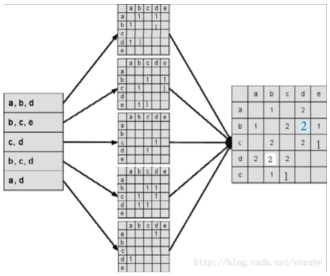
Specifically, we focus on all the articles of the target user compare them with the rest of the articles of other users. For each two articles, we analysis the keywords recorded in the file all questions info.txt which is obtained by previous crawlers and clustering method. Then, we apply the word vectors method to transform each key word to a vector.

For each two word-vectors, representing a keyword of one of the article of the target user and a keyword of one of the article if the other users, we can calculate their matching similarity by computing their Pearson Correlation Coefficient shown in the following formula.

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_i)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_j)^2}}.$$

If the value of the Pearson Correlation Coefficient for the two vectors exceed a certain threshold, then this pair of two keywords will be recognized as a close matching and we will add the value of the Pearson Correlation Coefficient to the score of the two articles contributing to the similarity of the two articles. After comparing each possible pair of keywords of the two articles, we can obtain a matrix of the score representing the similarity between each pair of articles.

As for each article of the target user, we can easily find the closest matching articles according to the results of similarity values. Thus, the articles of the other users with top five highest similarity values will be recommended to the target user.



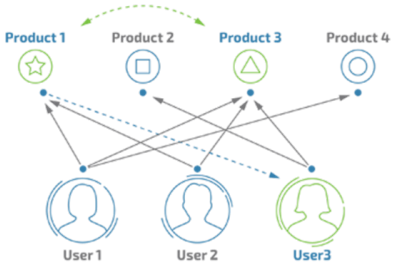
3.3 Improvement - Combination of User Graph and Item-based Collaborative Filtering

Since the Item-based Collaborative Filtering needs to compute and compare a large scale of values, it is probably a little bit time-consuming and inconvenient as such algorithm has to focus on each pair of articles instead of users. Then, we come up with a simple idea that can help to make some improvement.

That is, we can make comparison between the user graph of the target user and the articles of the rest of the users, instead of directly compare each pair of articles. Such method not only reduces the time complexity by such replacing, but also remains the information of different articles of the other users preserving both accuracy and efficiency.

Specifically, we focus on the user graph, a set of key words that represents the overall image of a user, of the target user. Then, for each articles of the rest of the users, we can simply compare the keywords in the user graph of the target user and the keywords of each articles of the other users.

And, we use the similar method including word-vector transformation, Pearson Correlation Coefficient computing, threshold matching, etc. Then, we can easily find the closest matching articles to the target users according to the results of similarity values sorted in ascending sequence as the recommendation articles for the target user.



4 Latent Semantic Analysis

This part is written by Pang Lianyu

Latent semantic analysis (LSA) is a technique in natural language processing, in particular distributional semantics, of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms.

LSA assumes that words that are close in meaning will occur in similar pieces of text (the distributional hypothesis). A matrix containing word counts per document (rows represent unique words and columns represent each document) is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD)

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

Figure 3: A: term-document matrix

is used to reduce the number of rows while preserving the similarity structure among columns.

Documents are then compared by taking the cosine of the angle between the two vectors (or the dot product between the normalizations of the two vectors) formed by any two columns. Values close to 1 represent very similar documents while values close to 0 represent very dissimilar documents.

4.1 Process

Assume that we have 6 documents $d_1, d_2, ..., d_6$. Then we can get the term-document matrix A from these documents.

In this matrix, each row represents a term/word, and each column represents a document. The value in i-th row and j-th column means that the i-th word appears $v_{i,j}^A$ times in the j-th document.

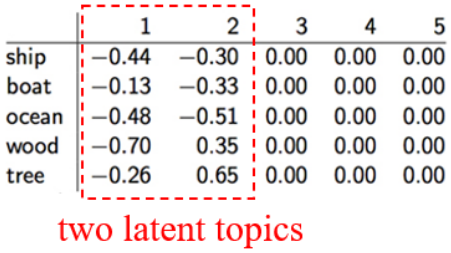
Then we do a SVD decomposition on A and we get three matrix U,D,V.

$$A = UDV^T$$

U is a term-topic matrix. Each row in U represents a term, and each column represents a topic. The value in i-th row and j-th column means the proportion of the i-th word in the j-t topic.

D is a topic-weight matrix, and it's a diagonal matrix. Each row in D represents a topic, and each column also represents a topic. The value v_i^D means the proportion of the i-th topic in all documents.

V^T is a topic-document matrix. Each row in V^T represents a topic, and each column



	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

Figure 4: U: term-topic matrix

	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

Figure 5: D: term-weight matrix

represents a document. The value in i-th row and j-th column means the proportion of the i-th topic in the j-t document.

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

Figure 6: V^T:topic-document matrix

5 About our work

The specific division of labor of team members is:

- Jing Hongyi:Word Embedding,User-based Collaborative Filtering
- Fan Tingyu:Article Clustering,User Profile
- He Wendi:Item-based Collaborative Filtering and improved version

- Pang Lianyu:Word Embedding,Latent Semantic Analysis

All the codes and resources can be got from:
https://github.com/JingHongyi/EE447_FinalProject