ChinaXiv合作期刊

DATA PAPER

User Profiling for CSDN: Keyphrase Extraction, User Tagging and User Growth Value Prediction

First-place Entry for User Profiling Technology Evaluation Campaign in SMP Cup 2017

Guoliang Xing[1,2], Hao Gao[1,2], Qi Cao[1,2], Xinyu Yue[1,2], Bingbing Xu[1,2], Keting Cen[1,2] & Huawei Shen[1,2†]

[1]Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

[2]University of Chinese Academy of Sciences, Beijing 100049, China

chinaXiv:202211.00457v1

Keywords: User profiling; Keyphrase extraction; User tagging; Growth value prediction; Word embedding

Citation: G. Xing, H. Gao, Q. Cao, X. Yue, B. Xu, K. Cen & H. Shen. User profiling for CSDN: Keyphrase extraction, user tagging and user growth value prediction. Data Intelligence 1(2019), 137-159. doi: 10.1162/dint_a_00015
Received: August 27, 2018; Revised: November 29, 2018; Accepted: December 6, 2018

ABSTRACT

The Chinese Software Developer Network (CSDN) is one of the largest information technology communities and service platforms in China. This paper describes the user profiling for CSDN, an evaluation track of SMP Cup 2017. It contains three tasks: (1) user document keyphrase extraction, (2) user tagging and (3) user growth value prediction. In the first task, we treat keyphrase extraction as a classification problem and train a Gradient-Boosting-Decision-Tree model with comprehensive features. In the second task, to deal with class imbalance and capture the interdependency between classes, we propose a two-stage framework: (1) for each class, we train a binary classifier to model each class against all of the other classes independently; (2) we feed the output of the trained classifiers into a softmax classifier, tagging each user with multiple labels. In the third task, we propose a comprehensive architecture to predict user growth value. Our contributions in this paper are summarized as follows: (1) we extract various types of features to identify the key factors in user value growth; (2) we use the semi-supervised method and the stacking technique to extend labeled data sets and increase the generality of the trained model, resulting in an impressive performance in our experiments. In the competition, we achieved the first place out of 329 teams.

† Corresponding author: Huawei Shen (Email: shenhuawei@ict.ac.cn; ORCID: 0000-0002-1081-8119).

ChinaXiv合作期刊

1. INTRODUCTION

User profiling is very important for precise recommendation and personalized services in the Internet era. In this paper, we focus on three key tasks of user profiling for the Chinese Software Developer Network (CSDN), an evaluation track of SMP Cup 2017 [1]. CSDN is one of the largest information technology communities and service platforms in China. CSDN has about 50 million registered users and over 100,000 users communicate and share knowledge and information via its Web forums every day [1]. The three tasks of user profiling for CSDN include user document keypharase extraction, user tagging and user growth value prediction.

Keyphrase extraction is to automatically extract the top *k* important phrases that can represent the main idea of a document. A supervised method is used for keyphrase extraction. First, we generate 301,076 candidate phrases for the whole corpus. It covers more than 90% over the annotated keyphrases in the training set. Then we extract statistical features and semantic features of the input examples, and compare the effectiveness of these features. Finally, by incorporating both the statistical features and semantic features, we achieved the highest score among all competitors.

The second task is to label user interests. The challenge contains a variety of issues. First, users may have more than one interest. The problem can be solved by creating several independent binary classifications. However, the correlations between interests are not considered. For instance, users labeled with "machine learning" are likely to have the interests of "deep learning." Second, the labeled data are insufficient due to costly manual labeling. If there are too many parameters in the model, it may lead to overfitting. To address the above-mentioned challenges, we propose a two-phase model. To encode the high dimensional features into low dimensional features to decrease the number of parameters, the independent binary classifications are formulated and the probabilities of user labels are predicted by each classifier in the first phase. To capture the correlations between labels, the neural network with softmax loss is adopted, where the probabilities from the first phase are regarded as the input in the second phase. The labels corresponding to the highest probabilities predicted by the neural network are users' final labels.

In Task 3, we need to predict the growth value of users in CSDN during the next year based on their posts and behaviors. After formulating this task as a regression problem, we explore various types of features to incorporate the key factors driving the growth of user value. Besides, semi-regression methods and a stacking technique are introduced into our prediction architecture to overcome the lack of labeled data and enhance the generality of the prediction model. Experimental results show that our proposed architecture can effectively predict user growth value.

The rest part of this paper will introduce our work in detail for each task, and the last section presents the conclusions and the future work.

chinaXiv:202211.00457v1

ChinaXiv合作期刊

2. KEYPHRASE EXTRACTION

Keyphrases are single or multi-word expressions that can represent the main topics of a document. They are proved useful in many areas, e.g., information retrieval [2], text summarization [3], text clustering [4], text categorization [5], text indexing [6] and opinion mining [7]. Moreover, keyphrases help users grasp the main topics of a Web page [8].

2.1 Related Work

Keyphrase extraction methods can be divided into two categories, unsupervised and supervised methods. In both of them, the extraction process can be divided into two steps: candidate phrase generation and finding top *k* phrases.

Candidate phrase generation is to extract a set of phrases or words using heuristic rules. These rules should be (1) general: the rule should not depend on the training data only; (2) high quality: the generated candidate phrase set should include the keyphrases that are "labels" in training data as many as possible; (3) minimum size: rules are designed to avoid spurious instances and keep the number of candidates to a minimum. Typical heuristic rules include removing stop words [9], allowing *n*-grams that appear in Wikipedia article titles to be candidates [10], extracting noun phrases [11], and extracting *n*-gram or noun phrases that satisfy pre-defined lexico-syntactic patterns [12]. In this paper, we generate candidate phrases with a frequency threshold rule.

Extracting top *k* keyphrases is to find the most important phrases from a document. The typical unsupervised methods of this approach can be categorized into ranking-based methods such as TF-IDF and TextRank [13] and clustering-based methods. TF-IDF, short for term frequency–inverse document frequency, is a numerical statistic that can reflect how important a phrase is to a document in a collection or corpus. The TextRank algorithm is an extension of the PageRank algorithm, a graph-based ranking method which was initially used to rank the importance of Web pages [14]. The idea of TextRank is that the importance of a candidate in a certain document depends on its relation with other candidate phrases in this document, which means, the more other candidate phrases a candidate is related with, the more important this candidate is; the more important other candidates are related to a candidate, the more important this candidate is. Researchers computed relatedness between candidates using co-occurrence counts [15] and semantic relatedness [16], and used TextRank to generate top *k* important keyphrases [13].

The two important issues of the supervised method for keyphrase extraction are feature design and modeling. Berend and Farkas [17] invented SZTERGAK system to extract features. They classified features to five categories, i.e., standard features, phrase-level features, document-level features, corpus-level features and external knowledge-based features. These features perform well on their tasks, but there are still some other features to discover. For example, the semantic embedding features for the phrases and documents.

Keyphrase extraction is treated as a binary classification task. For the supervised method, the goal of modeling is to train a classifier to determine whether a candidate phrase is a keyphrase, while the annotated

chinaXiv:202211.00457v1

ChinaXiv合作期刊

keyphrases in a document and other non-keyphrase candidates are used to generate positive and negative examples. So far, researchers have tried different learning algorithms to train the binary classifier. Frank et al. used naïve Bayes to train the classifier [18], and Turney tried using decision trees in 1999 [19]. Some other learning methods such as maximum entropy [20], multi-layer perceptron [21] and support vector machines [22] are also explored. Gradient boosting is a machine learning technique used for regression and classification problems, and eXtreme Gradient Boosting (XGBoost) is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable [23]. In the keyphrase extraction task, we choose XGBoost tool, and the inputs are features of <*doc, phrase*> pair, where the *doc* represents one document, and the *phrase* is a candidate phrase extracted from this document.

2.2 Candidate Phrase Generation

In keyphrase extraction task of SMP Cup 2017, the organizers provided a total of 1,022 documents together with five annotated keyphrases for each document. These annotated keyphrases were used for positive examples, and the other candidate phrases were taken as negative examples.

2.2.1 Procedure

Existing methods for candidate phrase generation, as introduced in Section 2.1, work well for English documents, but are not suitable for Chinese documents due to challenges in Chinese word segmentation. Moreover, the formats of the annotated keyphrases in training document set are quite diverse. Specifically, they can be a single Chinese character, a Chinese phrase with multiple characters (mostly with 2 to 7 characters), an English word, an English phrase with multiple words, a phrase consisting of both English and Chinese, and even an English phrase containing special characters, like c++, node.js, and utf-8. The high diversity of keyphrase format greatly increases the difficulty of candidate phrase generation.

To combat the aforementioned challenges, we adopt a heuristic procedure to automatically generate the candidate phrases (Figure 1):

1). Take the total 1 million documents as a corpus.
2). Extract the Chinese content of each document in the corpus.
3). Use seg-for-search-engine mode in Jieba to obtain the segmentation results of the Chinese phrases.
4). Extract Chinese phrases with 2 to 7 characters.
5). Use phrase frequency to filter phrases extracted in step 4.
6). Traverse the whole corpus and extract English phrases or combination phrases, which consist of both English words and Chinese characters generated in step 5, and order them by frequency to generate non-Chinese format candidates.
7). Combine the candidate phrases extracted in steps 5 and 6.

chinaXiv:202211.00457v1

ChinaXiv合作期刊

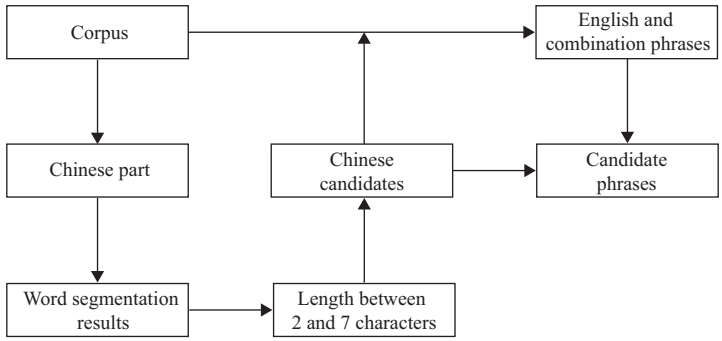


Figure 1. Candidate phrase generation procedure.

chinaXiv:202211.00457v1

Since a higher proportion of keyphrases in training set and the documents in this corpus are Chinese, we deal with Chinese phrases and English phrases separately in step 2. For Chinese phrases, the difficulty is how to balance the number of candidate phrases and the coverage of annotated keyphrases in the training set. At the very beginning, we simply use an open source library named Jieba for word segmentation. The shortcoming of this method is that only a small proportion of annotated keyphrases are included in the candidate phrases. Then we extracted every possible phrase with a length between 2 and 7 characters, which leads to explosive growing number of candidate phrases. There is a heuristic rule that the minimum unit of a Chinese phrase is a word instead of a character, so finally we use seg-for-search-engine mode in Jieba to extract every possible word and its position in the document. Then we construct every possible phrase with a length between 2 and 7 characters (step 3 and step 4). In step 5, we calculate the phrase frequency in the whole corpus, and use phrases with high frequency as the Chinese candidate phrases. Then, we traverse the corpus again to extract the phrases only in English and combination phrases. In the final stage, we merge all candidate phrases generated in step 5 and step 6 as our final candidate phrases.

2.2.2 Results

In total, we extract 301,076 phrases as the candidate phrases. Note the phrases that include special characters are not included, for there are a great number of such phrases in the whole corpus. As the amount of documents we need to deal is small, we extract phrases with special characters for each document. Finally, we generate 239,405 training examples from 1,022 documents in the training set and 233,580 validation examples from 1,022 documents in the validation set. We analyze the relation between candidate phrases in training examples and validation examples. The results are shown in Table 1. Our candidate phrases cover more than 90% of the annotated keyphrases. The difference between row 1 and row 2 comes from the noise in data, as there are some annotated keyphrases which do not exist in that document. The examples in “not in candidate phrases” of row 2 and row 3 are those phrases with special characters.

ChinaXiv合作期刊

Table 1. Candidate phrases.

	In candidate phrases	Not in candidate phrases
Annotated keyphrases	4,711	399
Positive training examples	4,454	14
Negative training examples	229,986	4,951
Validation examples	233,580	4,983

2.3 Feature Engineering

2.3.1 Statistical Features

Similar to Berend and Farkas' work [17], we divide the statistical features into four categories, features at the phrase level, at the document level and at the corpus level and external knowledge-based features.

Phrase-level features. In this task, we generate 31-dimensional features at the phrase level in terms of phrase length and formats: Chinese or English phrases, phrases containing special characters, the location of the special characters, a noun or a phrase containing a noun, the number of Chinese characters, letters or special characters it has, a subset of other candidate phrases (if phrase *A* is a subset of phrase *B*, then we call *B* is a *super-phrase* of *A*), etc.

Document-level features. They are calculated based on the candidate phrases and the document the phrases belong to. The intuitive features are calculated directly between a phrase and the document, like phrase frequency, whether a phrase appears in the title, and the location features. Also, the features of the document and the sentence in which a candidate phrase appears are also included in our model. Another type of important document-level features is the relation between a candidate phrase and some generated results from a document. For example, we analyze top 10 keywords, top 10 keyphrases and top 3 key sentences of a document using TextRank, and calculate whether a candidate phrase is included in the results. Finally, we calculate 27-dimensional document-level features in total.

Corpus-level features. We extract totally 4-dimensional features. They are phrase frequency in the whole corpus, the TF-IDF value, IDF value, and the total number of titles in which a candidate phrase appears.

External knowledge-based features. We use two external knowledge sources, the Wikipedia entry data and the Baidu Encyclopedia entry data. Specifically, we extract 4-dimensional features: whether a candidate phrase is a Wikipedia entry or is a Baidu Encyclopedia entry, whether a candidate phrase has a *super-phrase* that is a Wikipedia entry or a Baidu Encyclopedia entry.

2.3.2 Semantic Features

Semantic features refer to the word embedding features. In this task, the semantic features we extract include the embedding vectors of each candidate phrase, statistical features of the embedding vector and some distance features between a candidate phrase and the document it belongs to. The main procedures are summarized as follows.

chinaXiv:202211.00457v1

ChinaXiv合作期刊

1). Generate the embedding vector for every word in the corpus. There are two frequently used open source libraries, the word2vec and the GloVe. While the GloVe focuses on the global information, the word2vec focuses on the local information. The dimension is also an important hyper-parameter in word embedding. In this task, we set the dimension to be 128 and 256. Finally, we generate four types of embedding vectors for each word in the corpus.
2). Generate the embedding vector for phrases and document. We represent the embedding vectors of a candidate phrase or a document simply by summing the embedded values of words it is composed of. Note that the embedding vectors for candidate phrases and a document could be normalized.
3). Calculate the distance features and the distribution features. Since a document's title often contains its main idea, we use the embedding vectors of the title instead of the document to calculate the distance features of a candidate phrase, i.e., the distance between the title of the document and the candidate phrase. There are eight types of distance features we generate, i.e., the cosine similarity, the Cityblock distance, the Jaccard coefficient, the Canberra distance, the Euclidean distance, the Minkowski distance, the Bray-Curtis distance and the word mover's distance [24]. The distribution features can reflect the distributions of the embedding vectors of a candidate phrase in a title. We calculate the skewness and the kurtosis value as the distribution features.

2.4 Experimental Results

We finally extract 66-dimensional statistical features and 400-dimensional semantic features. We compare the effect of both the statistical features and the semantic features by using XGBoost. The parameters in the model are listed as follows: eta = 0.03, max_depth = 8, subsample = 0.8, colsample_bytree = 1.0, alpha = 0, lambda = 1 and gamma = 0. The performances on the validation set for different types of features are illustrated in Table 2.

Table 2. Performance of different types of features.

Features	Score
Semantic features	0.533
Statistical features	0.631
Both	0.676

Table 2 shows that the statistical features perform better than the semantic features. When combining the two types of features, we find that the performance is significantly enhanced, which means that the semantic features contain more information than the statistical features. Finally, we use the combined features as our model input, and our score is the best both in the validation set and in the test set.

3. USER TAGGING

It is challenging to label users' interests in our data set due to two main reasons: (1) users may have several interests, and (2) the labeled users are insufficient. To tackle these challenges, we propose a two-phase model to assign labels to users.

chinaXiv:202211.00457v1

ChinaXiv合作期刊

chinaXiv:202211.00457v1

3.1 Related Work

Boutell et al. [25] proposed binary relevance which independently trains one binary classifier for each label. For unseen instance x, the labels that x has are those that their probabilities are beyond the threshold. However, the correlations between labels are not considered in the model. Read et al. [26] leveraged the correlations and proposed a classier chains model. The basic idea is to form a chain of binary classifiers, where the prior result will be the feature in the next model. Unfortunately, when the cardinality of label space is large, this chain method is not scalable. Another heuristic way is to learn the neural network with softmax loss, and the labels with the top r highest probabilities are assigned to users. However, when the training set is small, it is likely that the model is overfitting with high-dimensional features.

3.2 Problem Formulation

Let $D_{train} = \{(U_1, L_1), (U_2, L_2), \dots, (U_N, L_N)\}$ denote the training set containing $|D_{train}| = N$ users, and $D_{test} = \{(U_1, L_1), (U_2, L_2), \dots, (U_M, L_M)\}$ denote the test set containing $|D_{test}| = M$ users, where $U_i = (U_{i1}, U_{i2}, \dots, U_{id})$ denotes ds features of user U_i, and $L_i = (L_{i1}, L_{i2}, \dots, L_{ir})$ denotes r labels for each user. Let $LS = \{ls_1, ls_2, \dots, ls_k\}$ denote label space which contains k labels and for (U_i, L_i), $L_i \subseteq LS$. Table 3 summarizes the notations used in this paper. For simplicity, we denote both D_{train} and D_{test} as D. Given this, we define the input and output of our problems as follows:

Input: the input of our problem consists of a set of N users in D_{train} with corresponding r labels.

Output: Assign r labels from LS to M user in D_{test}.

Table 3. Notations.

Symbol	Description
D_{train}	The training set
D_{test}	The test set
N, M	The number of users in training set and test set
$U_i = (U_{i1}, U_{i2}, \dots, U_{id})$	The d dimensions of user U_i
$L_i = (L_{i1}, L_{i2}, \dots, L_{ir})$	The r labels that U_i has
$LS = \{ls_1, ls_2, \dots, ls_k\}$	The label space that contains k labels

3.3 Two-Phase Model

In Section 3.2, we formulate the problem of labeling users as a multi-label classification problem, where our goal is to tag each user with labels from given label space. Compared with the traditional multi-label classification problem, where the number of labels assigned to each sample is not necessarily the same, our task requires us to assign the same number of labels to each user. In this section, we take into account both insufficient samples and the correlation among labels.

ChinaXiv合作期刊

Figure 2 illustrates the framework of our two-phase model, where the first phase is concerned with independent binary classifications (dark-color rectangle) and the second phase a neural network (light-color rectangle). In the first phase, we formulate each label as a binary classification problem. After training, we obtain k classifiers. In the second phase, we train a neural network, where the input values are the probabilities predicted by the first phase, and the softmax is the loss function. Finally, r labels with the highest probabilities are chosen to be a user's interests.

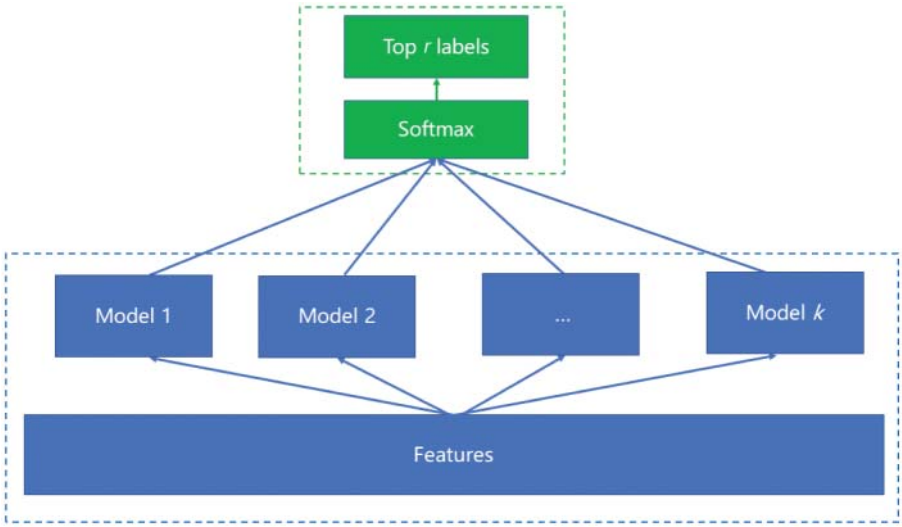


Figure 2. The framework of a two-phase model.

3.3.1 Independent Binary Classifications

We first transfer the training set and test set as follows: for ls_i in label space LS, if $ls_i \in L_j$ for (U_j, L_j) in D, $\{(U_j,1)\}$ is a training sample for label ls_i; otherwise $\{(U_j,0)\}$ is a training sample for label ls_i. Denoting with $\hat{D}_i$ the set of training samples for label ls_i, we have

$$\hat{D}_i = \{(U_j, \phi(ls_i, L_j))\} \tag{1}$$

$$\phi(ls_i, L_j) = \begin{cases} 1, & ls_i \in L_j \\ 0, & ls_i \notin L_j \end{cases}. \tag{2}$$

Then we train k binary classifications with $\hat{D}_i$ independently. In addition, we leverage bagging strategy for learning from $\hat{D}_i$ to reduce the generalization errors.

Bagging strategy. We divide $\hat{D}_i$ into c folds randomly, and the continuous c-1 folds are for training, the rest 1 fold for validation. Finally, we obtain c classifiers for $\hat{D}_i$.

chinaXiv:202211.00457v1

ChinaXiv合作期刊

chinaXiv:202211.00457v1

After training, $k * c$ classifiers are learned, where k is the number of labels in label space, and c is the quantity of classifiers of each label. When predicting unlabeled users in D_{test}, the features of users are fed to c classifiers for each label in label space and the mean of the probabilities predicted by c classifiers is used as the input which is fed to the second phase.

3.3.2 Softmax Classification

The labels which correspond to the highest r probabilities predicted by the independent binary classifications can be assigned to users heuristically. However, there are two main weaknesses. First, the probabilities of different labels are not comparable, e.g., the label with higher probability may not be the users' label as the probabilities are not predicted by the same model. Second, the relevance of labels is not considered, which is a strong prior that the first phase does not leverage.

Here we introduce the model of the second phase, which is a neural network with softmax loss to resolve the above weaknesses. Figure 3 illustrates the three-layer architecture of the neural network of our model. The input layer contains r units having r probabilities predicted by the independent binary classifiers, and the output layer contains r units. In the hidden layer, we leverage dropout to avoid overfitting. The softmax loss is defined as below:

$$loss = \frac{1}{n}\sum f(P_i, L_i), \tag{3}$$

where P_i is the probability of user i's labels predicted by the previous classification, and L_i is the ground truth of user i, and

$$f(P_i, L_i) = -\sum_{j=1}^{k} L_{ij} log P_{ij}. \tag{4}$$

Similarly, we adopt bagging strategy to decrease the loss in generalization performance. When predicting the unlabeled users, the labels corresponding to the highest probabilities are the users' labels.

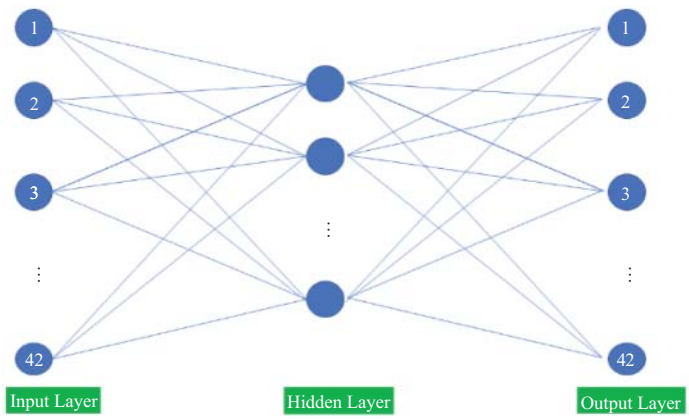


Figure 3. The architecture of the neural network.

ChinaXiv合作期刊

3.4 Experiments Results

3.4.1 Data Collection

The problem is that given the blog content, the users' behaviors, the structure relationships and chatting relationships between users in CSDN, we need to tag each user with three labels from the label space of 42 labels, e.g., machine learning and Web development. Table 4 lists the statistics of the data collection.

Table 4. The statistics of data sets.

Item	Volume
#blogs	1,000,000
#record of posting	1,000,000
#record of browsing	3,536,444
#record of commenting	182,273
#record of voting up	95,668
#record of voting down	9,326
#record of favorite	104,723
#structure relationship	667,037
#chatting relationship	46,572
#users in training set	1,055
#users in test set	1,055

Note: "Voting up" means users click a "like" button, while "voting-down" means users click a "dislike" button.

3.4.2 Feature Definition

It turns out that content features are the most helpful features, of which the basic idea is that the users with similar interests tend to post (browse, comment on, add favorites, vote down and vote up) the similar blogs.

Doc2Vec [27]. Doc2vec is an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, and the docs are embedded into distributed vectors. We believe the users associated with similar blogs tend to have the similar interests. The features of users are defined as the representations of blogs in six types of behaviors (browse, post, comment, add favorites, vote down and vote up).

In our experiments, blogs are represented by the vector of 200 dimensions. Finally, the users are represented by vectors of 1,200 dimensions (200 dimensions – six types of behaviors).

3.4.3 Evaluation Metrics and Baselines

To evaluate our proposed model, the following performance metric is considered:

$$score = \frac{1}{M}\sum_{i=1}^{M}\frac{\left|L_i \cap L_i^*\right|}{\left|L_i\right|}, \tag{5}$$

chinaXiv:202211.00457v1

ChinaXiv合作期刊

where M is the cardinality of the data set, L_i is the predicted labels and L_i^* is the ground truth. For each user, the score is 1, 2/3 or 1/3, if the 3, 2, or 1 label(s) are predicted correctly. The higher the score, the better performance we achieve.

We compare the following methods for user labeling:

Top *r* labels (TL). This method tags each user with the most r popular labels in the training set. In our data set, $r = 3$.

Independent binary classifications (IBC). This is the first phase in our model, which tags the unlabeled users with the labels corresponding to the highest r probabilities predicted by independent binary classifications.

Softmax classification (SC). This is the second phase in our model, where the inputs are the features defined in Section 3.4.2 instead of probabilities, and the neural network with softmax loss is leveraged. Finally, the top r labels are assigned to each user.

3.4.4 Performance Analysis

We perform the experiments on the data set described in Section 3.4.1, and the results are based on the test set with 1,055 users.

Table 5 shows the proposed method achieves better performance than other methods. Since IBC does not consider the correlations between labels, the probabilities predicted by the independent classifiers are not comparable, while the correlations are captured in the second phase of our model. The parameters in SC are far more than the number of training set, causing overfittings of the training data.

Table 5. Comparison of scores of methods.

Method	TL	IBC	SC	Our method
Score	0.3754	0.45592	0.4436	**0.46319**

Note: TL: top r labels, IBC: independent binary classifications, and SC: Softmax classification.

4. USER GROWTH VALUE PREDICTION

In this task, we need to predict the growth value of users for the next year based on their posts and behaviors in the past year. The growth value for each user has been regularized in [0,1], where 0 represents the loss of a user.

4.1 Architecture

The task of user growth value prediction can be easily formalized as a typical regression task in machine learning, i.e., given that feature x_i contains the statistics of behaviors and posts of user i, predict his/her

chinaXiv:202211.00457v1

ChinaXiv合作期刊

future growth value y_i. The whole architecture to solve this task is shown in Figure 4, which consists of features layer, primary model and model ensemble.

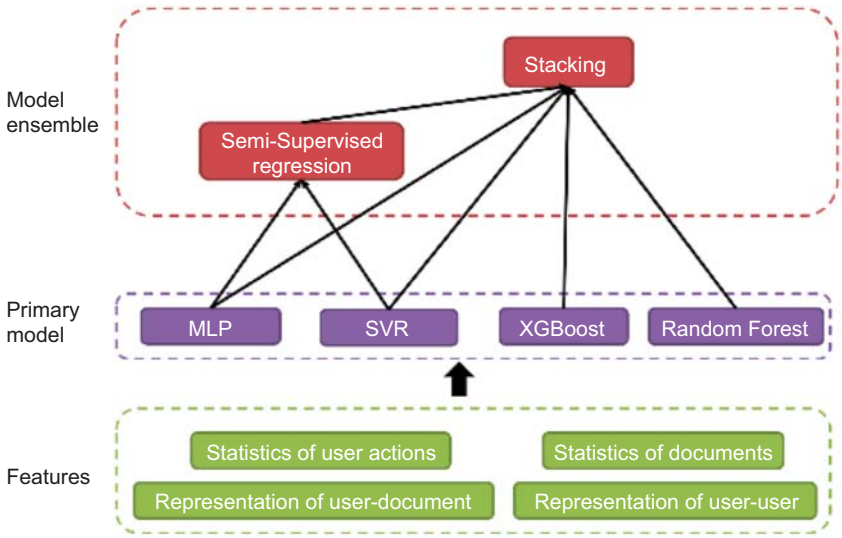


Figure 4. Architecture of user growth value prediction.

4.2 Features

4.2.1 Statistics of User Behaviors

Frequency of behavior. We count the number of behaviors triggered by a user during a certain period, i.e., the number of posts the user published, viewed, commented on, collected, voted up and voted down, the number of private messages the user received and the number of private messages the user sent to others. This type of features reflect how active a user is during a certain period of time, e.g., one quarter, which has also been used in customer lifetime value prediction [28].

Change of behavior frequency. We calculate the change of behavior frequency. In particular, we subtract the frequency of posts by the user during the last two quarters from the frequency of posts by the user during the last one quarter.

The last active time. We find the last time of a certain behavior of a user, such as writing posts, commenting on posts, viewing posts and so on. These features also reflect the activities of the user.

The proportion of a certain type of behavior. We calculate the proportion of a certain type of behavior over all behaviors during a certain period, i.e., one quarter. These features characterize the preference of a user, e.g., prefer writing articles to browsing others' articles.

chinaXiv:202211.00457v1

ChinaXiv合作期刊

Number of retweeted posts. We notice that the posts created by a user and the posts retweeted by others have quite different influences, and thus should be treated differently when studying user activities and their posts' quality. We distinguish the original posts and retweeted posts based on the MinHash algorithm [29]. The main assumption is that if we have two similar posts, we regard the oldest one as the original post and the others as retweeted posts. Specifically, we calculate a 64-byte hash value for each post. It would be costly if we consider all pairs of posts. Consequently, we use the first 16-byte of hash value as key. We only compare the hash value when the keys of two posts are the same. We count the number of original posts published by the user during one quarter as a feature. This feature can reflect the quality and creativity of the user to a certain extent.

User points in the past year. We calculate the point of a user in the past year according to the CSDN user point rule published online. This feature can also reflect the quality of the user.

chinaXiv:202211.00457v1

4.2.2 Statistics of Documents

Length of documents. We calculate the average length and variance of documents written by a user. These features reflect the characteristics of documents written by the user.

Feedback received by documents. We count the number of a user's behaviors, such as browsing, commenting, voting up or voting down and feedback the user received for his or her posts during a certain period, e.g., one quarter. This type of features can reflect the quality of documents written by the user.

4.2.3 Representation of User-Document

We also want to extract some useful information from the user-document interaction matrix. We construct six types of interaction matrices, i.e., user-document browse matrix, user-document comment matrix, user-document vote-up matrix, user-document vote-down matrix, use-document collection matrix, user-document all behaviors matrix. The element in a matrix represents the frequency of a user's certain behavior during the past year. We decompose these matrices with non-negative matrix factorization (NMF) [30] and use the user representation as features. These features reflect the similarity of user behaviors in a transformed space.

4.2.4 Representation of User-User

Matrix factorization. Similar to representation of user-document, we construct two types of user-user matrices, i.e., matrix of following relations and matrix of sending private messages. We also use NMF to decompose these two matrices (i.e., each row corresponds to a user vector) and use the user vectors as features.

PageRank. We construct a user-user interaction network to reflect the influence between users. If user A browses an article written by user B, then there exists an edge likely from A to B. We run the PageRank algorithm [14] on this network to calculate the importance of each user.

ChinaXiv合作期刊

4.3 Primary Model

We adopt four primary models with significant difference, i.e., Multi-layer Perception (MLP), Support Vector Regression (SVR), XGBoost and Random Forest to predict user growth value. For each primary model, we employ the sampling strategy to keep a model's generality (Figure 5). Specifically, after sampling k features for K times and sampling n samples for N times, we obtain $N * K$ different training data sets for each primary model, resulting in $N * K$ different trained models. When performing the final prediction, we combine all the $N * K$ results before bagging them with the median value. Next, we give a simple instruction of each primary model and their performance, where the evaluation score is defined in Equation (6):

$$score = 1 - \frac{1}{N}\sum_{i=1}^{N}\begin{cases} 0, & v_i = 0, v_i^* = 0 \\ \left|v_i - v_i^*\right| \Big/ max\left(v_i, v_i^*\right), & otherwise \end{cases}, \tag{6}$$

where v_i is the true user growth value, v_i^* is the predicted user growth value by models and N is the number of users to be predicted.

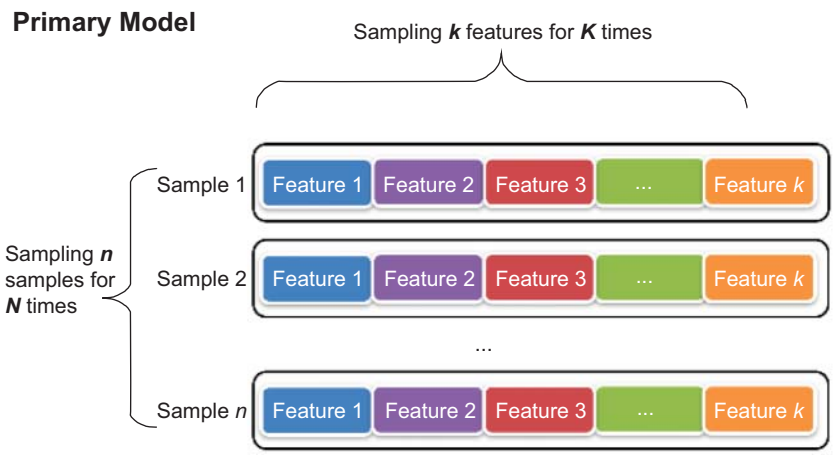


Figure 5. Bagging of primary model.

4.3.1 Multi-Layer Perception (MLP)

We use the framework of tensorflow [31] to perform this model, which can easily modify the optimized loss to be the same with the evaluation score. Since we only have 1,000 samples, which can easily cause overfitting, our MLP model only contains the input layer and output layer. The performance of this primary model achieves a prediction score of 0.736. To ensure the effectiveness of extracted features, we calculate the variance of each feature weight among $N * K$ trained models, and filter out the features with higher variance of model weight (average weight/variance >30%). We assume that the features with higher variance of model weight are not truly effective features. We finally preserve 41 features in this primary model and the final prediction score reaches 0.752 (Table 6), which indicates a significant improvement compared with the one without feature filtering.

chinaXiv:202211.00457v1

ChinaXiv合作期刊

Table 6. Performance of models.

Model	Score
Random Forest	0.737
XGBoost	0.750
MLP	0.752
SVR	0.754
Semi-supervised regression-MLP	0.753
Semi-supervised regression-SVR	0.755
Stacking	0.764

4.3.2 Support Vector Regression (SVR)

Since the evaluation score does not have the second order derivative, the next three primary models cannot optimize toward the evaluation score. We adopt an approximated loss function, i.e.,

$$loss_i = (log(v_i) - log(v_i^*))^2, \tag{7}$$

where v_i is the true user growth value, v_i^* is the predicted user growth value. SVR [32] calculate the loss only when

$$|\log(v_i) - \log(v_i^*)| > \varepsilon, \tag{8}$$

where ε is a super parameter that needs to be manually adjusted. We use grid search to find the best parameter for this primary model, resulting in the prediction score = 0.754 (Table 6). This primary model achieves similar prediction performance with the MLP model.

4.3.3 XGBoost

XGBoost [23] is an optimized variant of Gradient Boosting Decision Tree, which is much faster than the traditional one. We also use grid search to find the best super parameters, which reach a score of 0.750.

4.3.4 Random Forest

Random Forest regression model is composed of many decision trees and uses the average result of each tree as the final output [33]. Similarly, we use grid search to find the best super parameters, and reach a prediction score of 0.737. This primary model's performance is not so competitive, which may be due to the complexity of this model, causing great difficulty when tuning parameters.

4.4 Model Ensemble

Since we only have 1,000 samples with labels, this also poses a great challenge to the learning results of our model. We propose two ways to overcome this challenge: use semi-supervised regression to extend the samples with labels, and adopt the stack algorithm to ensure the generality of our model.

chinaXiv:202211.00457v1

ChinaXiv合作期刊

chinaXiv:202211.00457v1

4.4.1 Semi-Supervised Regression

According to the proposed idea of co-training for the regression problem [34,35], we choose the two best models, i.e., MLP and SVR, to extend the samples with labels. The specific description of the proposed semi-supervised regression algorithm is shown in Algorithm 1. When running this algorithm, we set the maximum iterations *T* equal to 500, and the maximum number of samples *K* that the cache pool can hold equals 200. However, it is quite slow to run the semi-supervised regression, averagely adding 10 samples with labels after one day. Because of time constraints, we extend only the data set to 1,066 samples with labels. After extending the data set, both the performance of MLP and SVR have been slightly improved, whose evaluation score has been increased by 0.001, respectively (Table 6).

Algorithm 1. Semi-supervised regression algorithm.

Input:
MLP_label: Samples with label for MLP MLP_unlabel: Samples without label for MLP
SVR_label: Samples with label for SVR SVR_unlabel: Samples without label for SVR
T: Maximum iterations K: Maximum number of samples that the cache pool can hold
Process:

```
Repeat for T iterations:
    #train MLP with MLP_label
    MLP = Train_model (MLP_label)
    #predict the label of SVR_unlabel with MLP
    SVR_unlabel_pre = Predict (MLP, SVR_unlabel)
    #select K samples from SVR_unlabel_pre as the cache pool for SVR
    SVR_unlabel_cache = Sampling (SVR_unlabel_pre, K)
    # train SVR with SVR_label
    SVR = Train_ model (SVR_label)
    #predict the label of MLP_unlabel with SVR
    MLP_unlabel_pre = Predict (SVR, MLP_unlabel)
    # select K samples from MLP_unlabel_pre as the cache pool for MLP
    MLP_unlabel_cache = Sampling (MLP_unlabel_pre, K)
    #select sample from cache pool for MLP to extend the samples with label for MLP
    for sample_i in MLP_unlabel_cache:
        new_MLP =Train_model (sample_i + MLP_label)
        Loss_i = model_loss (sample_i + MLP_label)
    selected_MLP_i = arg min (Loss_i)
    #select sample from cache pool for SVR to extend the samples with label for SVR
    for sample_i in SVR _unlabel_cache:
        new_SVR =Train_model (sample_i + SVR_label)
        Loss_i = model_loss (sample_i + SVR_label)
    selected_SVR_i = arg min (Loss_i)
```

4.4.2 Stacking

Stacking [36] is a type of ensemble learning, which can combine primary models to generate more robust results. Here we adopt the above-mentioned four single models, i.e., MLP, SVR, Random Forest, XGBoost, and Semi-Supervised Regression-MLP and Semi-Supervised Regression-SVR which have been extended to

ChinaXiv合作期刊

the training set, as our primary models. We train another SVR based on these models' output as features. The final model after stacking achieves the best performance, reaching a prediction score which equals 0.764.

5. CONCLUSIONS AND FUTURE WORK

This paper describes our work for User Profiling Technology Evaluation Campaign in SMP Cup 2017. In the keyphrase extraction task, we treat keyphrase extraction as a classification problem and use the XGBoost model to predict the top three keyphrases. By this method, we finally obtain the best score among all the participants. In the task of user tagging, two challenges are tackled in our two-phase model. In the first phase, which is independent binary classifications, we encode the high dimensional features into the low dimensional representations in a supervised way. Due to insufficient labelled data, our model has fewer parameters than traditional softmax-classifer model and avoids overfitting to some extent. In the second phase, neural network with softmax as loss function assigns the most likely labels to users. The second phase models the correlations between labels implicitly, and makes the probabilities predicted by the independent binary classifiers comparable. In the task of user growth value prediction, we overcome the lack of training data sets via introducing a semi-supervised regression model and the stacking technique. Experiments demonstrate the effectiveness of our framework for user growth value prediction with an accuracy of 0.764.

In the future, we will further improve the performance on user profiling. For keyphrase extraction, more state-of-the-art models modeling the representations of documents can be considered to extract more features, e.g., LSTM, hierarchical attention models and so on. For user tagging, we can further explicitly exploit the structures between each class of users, e.g., the taxonomy of labels, to improve the performance of multi-label classification. For user growth value prediction, methods based on point process can be employed to capture the characteristics of user growth value with time going by.

AUTHOR CONTRIBUTIONS

All of the authors contributed equally to the work. H. Shen (shenhuawei@ict.ac.cn) is the coordinator of the research group, and designs the whole framework for user profiling. G. Xing (xingguoliang@ict.ac.cn) and X. Yue (yuexinyu@ict.ac.cn) are the co-contributors for the task of keyphrase extraction. H. Gao (gaohao@ict.ac.cn) and B. Xu (xubingbing@ict.ac.cn) are the co-contributors for the task of user tagging. Q. Cao (caoqi@ict.ac.cn) and K. Cen (cenketing@ict.ac.cn) are the co-contributors for the task of user growth value prediction. All authors contributed to writing and revising this manuscript.

ACKNOWLEDGEMENTS

The work is supported by the National Natural Science Foundation of China (NSFC) (No. 61472400, No. 91746301 and No. 61802371). H. Shen is also funded by K.C. Wong Education Foundation and the Youth Innovation Promotion Association of the Chinese Academy of Sciences.

chinaXiv:202211.00457v1

ChinaXiv合作期刊

chinaXiv:202211.00457v1

REFERENCES

[1] SMP CUP 2017. Available at: https://www.biendata.com/competition/smpcup2017.

[2] O. Medelyan, E. Frank, & I.H. Witten. Human-competitive tagging using automatic keyphrase extraction. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, 2009, pp. 1318–1327. doi: 10.3115/1699648.1699678.

[3] M. Dredze, H.M. Wallach, D. Puller, & F. Pereira. Generating summary keywords for emails using topics. In: Proceedings of the 13th International Conference on Intelligent User Interfaces, 2008, pp. 199–206. doi: 10.1145/1378773.1378800.

[4] K.M. Hammouda, D.N. Matute, & M.S. Kamel. CorePhrase: Keyphrase extraction for document clustering. In: Proceedings of the 4th International Conference on Machine Learning and Data Mining in Pattern Recognition, 2005, pp. 265–274. doi: 10.1007/11510888_26.

[5] A. Hulth, & B.B. Megyesi. A study on automatically extracted keywords in text categorization. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, 2006, pp. 537–544. doi: 10.3115/1220175.1220243.

[6] C. Gutwin, G. Paynter, I. Witten, C. Nevill-Manning, & E. Frank. Improving browsing in digital libraries with keyphrase indexes. Decision Support Systems 27(1-2)(1999), 81–104. doi: 10.1016/S0167-9236(99)00038-X.

[7] G. Berend. Opinion expression mining by exploiting keyphrase extraction. In: Proceedings of the 5th International Joint Conference on Natural Language Processing, 2011, pp. 1162–1170.

[8] M. Chen, J.-T. Sun, H.-J. Zeng, & K.-Y. Lam. A practical system of keyphrase extraction for web pages. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, 2005, pp. 277–278. doi: 10.1145/1099554.1099625.

[9] Z. Liu, P. Li, Y. Zheng, & M. Sun. Clustering to find exemplar terms for keyphrase extraction. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, 2009, pp. 257–266. doi: 10.3115/1699510.1699544.

[10] K. Barker, & N. Cornacchia. Using noun phrase heads to extract document keyphrases. In: Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence, 2000, pp. 40–52. doi: 10.1007/3-540-45486-1_4.

[11] Y.-F. B. Wu, Q. Li, R.S. Bot, & X. Chen. Domain-specific keyphrase extraction. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, 2005, pp. 283–284. doi: 10.1145/1099554.1099628.

[12] C.Q. Nguyen, & T.T. Phan. An ontology-based approach for key phrase extraction. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing: Short Papers, 2009, pp. 181–184. doi: 10.3115/1667583.1667639.

[13] R. Mihalcea, & P. Tarau. Text rank: Bringing order into texts. In: Conference on Empirical Methods in Natural Language Processing, 2004, pp. 404–411. doi: 10.3115/1219044.1219064.

[14] S. Brin, & L. Page. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems 30(1–7)(1998), 107-117. doi: 10.1016/S0169-7552(98)00110-X.

[15] Y. Matsuo, & M. Ishizuka. Keyword extraction from a single document using word cooccurrence statistical information. International Journal on Artificial Intelligence Tools 13(1) 2004, 157–169. doi: 10.1142/S0218213004001466.

[16] M. Grineva, M. Grinev, & D. Lizorkin. Extracting key terms from noisy and multitheme documents. In: Proceedings of the 18th International Conference on World Wide Web, 2009, pp. 661–670. doi: 10.1145/1526709.1526798.

[17] G. Berend, & R. Farkas. SZTERGAK: Feature engineering for keyphrase extraction. In: Proceedings of the 5th International Workshop on Semantic Evaluation, 2010, pp. 186–189. Available at: http://anthology.aclweb.org/S/S10/S10-1040.pdf.

ChinaXiv合作期刊

[18] E. Frank, G.W. Paynter, I.H. Witten, C. Gutwin, & C.G. Nevill-Manning. Domain-specific keyphrase extraction. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence, 1999, pp. 668–673. doi: 10.1145/1099554.1099628.
[19] P. Turney. Learning to extract keyphrases from text. National Research Council Canada, Institute for Information Technology, Technical Report ERB-1057. Available at: https://arxiv.org/ftp/cs/papers/0212/0212013.pdf.
[20] S.N. Kim, & M.-Y. Kan. Re-examining automatic keyphrase extraction approaches in scientific articles. In: Proceedings of the ACL-IJCNLP Workshop on Multiword Expressions, 2009, pp. 9–16. doi: 10.3115/1698239.1698242.
[21] P. Lopez, & L. Romary. HUMB: Automatic key term extraction from scientific articles in GROBID. In: Proceedings of the 5th International Workshop on Semantic Evaluation, 2010, pp. 248–251. Available at: http://aclweb.org/anthology/S/S10/S10-1055.pdf.
[22] X. Jiang, Y. Hu, & H. Li. A ranking approach to keyphrase extraction. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2009, pp. 756–757. doi: 10.1145/1571941.1572113.
[23] T. Chen, & C. Guestrin. XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794. doi: 10.1145/2939672.2939785.
[24] M.J. Kusner, Y. Sun, N.I. Kolkin, & K.Q. Weinberger. From word embeddings to document distances. In: Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 957–966. Available at: http://jmlr.org/proceedings/papers/v37/kusnerb15.pdf.
[25] M.R. Boutell, J. Luo, X. Shen, & C.M. Brown. Learning multi-label scene classification. Pattern Recognition 37(9)(2004), 1757–1771. doi: 10.1016/j.patcog.2004.03.009.
[26] J. Read, B. Pfahringer, G. Holmes, & E. Frank. Classifier chains for multi-label classification. In: W. Buntine, M. Grobelnik, & J. Shawe-Taylor (eds.) Machine Learning and Knowledge Discovery in Databases. Berlin: Springer, 2009, pp. 254–269. doi: 10.1007/978-3-642-04174-7_17.
[27] Q.V. Le, & T. Mikolov. Distributed representations of sentences and documents. In: Proceedings of the 31st International Conference on Machine Learning, 2014, pp. 1188–1196. Available at: https://dl.acm.org/citation.cfm?id=3045025.
[28] B.P. Chamberlain, A. Cardoso, C.H. Liu, R. Pagliari, & M.P. Deisenroth. Customer life time value prediction using embeddings. arXiv preprint. arXiv:1703.02596, 2017.
[29] M.S. Charikar. Similarity estimation techniques from rounding algorithms. In: Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 380–388. doi: 10.1145/509907.509965.
[30] D.D. Lee, & H.S. Seung. Learning the parts of objects by non-negative matrix factorization. Nature 401(6755)(1999), 788-791. doi: 10.1038/44565.
[31] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, S. Ghemawat, … & X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. Available at: https://www.researchgate.net/publication/301839500_TensorFlow_Large-Scale_Machine_Learning_on_Heterogeneous_Distributed_Systems.
[32] H. Drucker, C.J.C. Burges, L. Kaufman, A.J. Smola, & V. Vapnik. Support vector regression machines. In: M.C. Mozer, M.I. Jordan, & T. Petsche (eds.) Advances in Neural Information Processing Systems, 1997, pp. 155–161. Available at: http://papers.nips.cc/book/advances-in-neural-information-processing-systems-9-1996.
[33] L. Breiman. Random Forests. Machine Learning 45(1)(2001), 5–32. doi: 10.1023/A:1010933404324.
[34] Z.H. Zhou, & M. Li. Semi-supervised regression with co-training. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence, 2005, 908–913. doi: 10.1109/ICSENG.2005.72.
[35] Z.H. Zhou, & M. Li. Semisupervised regression with cotraining-style algorithms. IEEE Transactions on Knowledge and Data Engineering 19(11)(2007), 1479–1493. doi: 10.1109/TKDE.2007.190644.
[36] L. Breiman. Stacked regressions. Machine Learning 24(1)(1996), 49–64. doi: 10.1007/BF00117832.

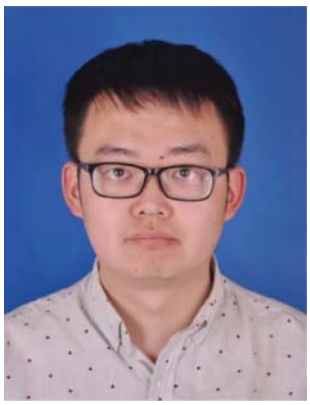
chinaXiv:202211.00457v1

ChinaXiv合作期刊

chinaXiv:202211.00457v1

AUTHOR BIOGRAPHY

Guoliang Xing received his Master's degree from Institute of Computing Technology, Chinese Academy of Sciences (CAS) in 2013. He is currently working at Baidu Online Network Technology (Beijing) Co., Ltd. His research interests include data mining and topic detection.



Hao Gao is currently a PhD student at the Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences (CAS). He received his Bachelor's degree from Northwestern Polytechnical University in 2015. His main research interests include social computing and Web data mining.

Qi Cao is currently a PhD student at the Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences (CAS). She received her Bachelor's degree from Renmin University of China in 2015. Her research interests include social media computing, influence modeling and information diffusion prediction.

ChinaXiv合作期刊

chinaXiv:202211.00457v1

Xinyu Yue received his Master's degree from Institute of Computing Technology, Chinese Academy of Sciences (CAS) in 2018. Xinyu Yue is currently working at Bytedance Technology Co., Ltd. His research interests include recommendation system and data mining.

Bingbing Xu is currently a PhD student at the Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences (CAS). She received her Bachelor's degree from Harbin Institute of Technology in 2016. Her research interests include geometric deep learning, graph-based data mining and graph convolution.

Keting Cen is currently a PhD student at the Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences (CAS). He received his Bachelor's degree from Harbin Institute of Technology in 2016. His research interests include network embedding, semi-supervised learning on graph and graph convolutional networks.

ChinaXiv合作期刊

Huawei Shen is a Professor at the Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences (CAS). He received his PhD degree in Computer Science from Institute of Computing Technology, CAS. His research interests include social network analysis, social media analytics, network data mining and scientometrics. He has published over 100 research papers in prestigious international journals and conferences.

chinaXiv:202211.00457v1