Machine Learning-Driven User Profiling: Keyphrase Extraction, User Tagging and Growth Value Prediction (Postprint)

Authors: Xing, Guoliang, Gao, Hao, Cao, Qi, Yue, Xinyu, Xu, Bingbing, Cen, Keting, Shen, Huawei, Shen, Huawei

Date: 2022-11-27T00:00:00+00:00

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.

Full Text

Preamble

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

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 our work on user profiling for CSDN, an evaluation track of SMP Cup 2017, which contains three tasks: (1) user document keyphrase extraction, (2) user tagging, and (3) user growth value prediction.

For 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 address class imbalance and capture interdependencies between classes, we propose a two-stage framework: (1) for each class, we train a binary classifier to model each class against all others independently; (2) we feed the outputs of these trained classifiers into a softmax classifier to tag each user with multiple labels.

In the third task, we propose a comprehensive architecture to predict user growth value. Our contributions are summarized as follows: (1) we extract various types of features to identify key factors in user value growth; (2) we employ semi-supervised methods and stacking techniques to extend labeled datasets and increase the generality of the trained model, resulting in impressive experimental performance. In the competition, we achieved first place out of 329 teams.

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

---

1. INTRODUCTION

User profiling is essential for precise recommendation and personalized services in the Internet era. This paper focuses on three key tasks of user profiling for the Chinese Software Developer Network (CSDN), an evaluation track of SMP Cup 2017 [**?**]. CSDN is one of China' s largest information technology communities and service platforms, with approximately 50 million registered users and over 100,000 users communicating and sharing knowledge daily via its Web forums [**?**]. The three user profiling tasks for CSDN include user document keyphrase extraction, user tagging, and user growth value prediction.

Keyphrase extraction aims to automatically extract the top k important phrases that represent a document' s main idea. We employ a supervised method for this task. First, we generate 301,076 candidate phrases for the entire corpus, covering more than 90% of annotated keyphrases in the training set. Then we extract statistical and semantic features from input examples and compare their effectiveness. Finally, by incorporating both feature types, we achieve the highest score among all competitors.

The second task involves labeling user interests, which presents several challenges. First, users may have multiple interests, which could be addressed by creating several independent binary classifications. However, this approach fails to consider correlations between interests. For instance, users labeled with "machine learning" are likely to also be interested in "deep learning." Second, labeled data are insufficient due to costly manual annotation, and models with too many parameters may overfit. To address these challenges, we propose a two-phase model. In the first phase, we formulate independent binary classifications to encode high-dimensional features into low-dimensional representations, with each classifier predicting user label probabilities. In the second phase, we adopt a neural network with softmax loss to capture label correlations, using the first phase' s probabilities as input. The labels corresponding to the highest probabilities predicted by the neural network become the user' s final labels.

In Task 3, we predict user growth value in CSDN for the next year based on their posts and behaviors. After formulating this as a regression problem, we explore various feature types to incorporate key factors driving user value growth. Additionally, we introduce semi-supervised methods and stacking techniques into our prediction architecture to overcome labeled data scarcity and enhance model generality. Experimental results demonstrate that our proposed architecture effectively predicts user growth value.

The remainder of this paper details our work for each task, with the final section presenting conclusions and future work.

2. KEYPHRASE EXTRACTION

Keyphrases are single or multi-word expressions that represent a document' s main topics. They have proven useful in many areas, including information retrieval [**?**], text summarization [**?**], text clustering [**?**], text categorization [**?**], text indexing [**?**], and opinion mining [**?**]. Moreover, keyphrases help users quickly grasp the main topics of a Web page [**?**].

2.1 Related Work

Keyphrase extraction methods can be divided into unsupervised and supervised approaches. Both involve two steps: candidate phrase generation and selecting the top k phrases.

Candidate phrase generation extracts a set of phrases or words using heuristic rules. These rules should be: (1) general—not dependent solely on training data; (2) high quality—the generated candidate set should include as many "label" keyphrases from training data as possible; (3) minimal size—rules should avoid spurious instances and keep candidate numbers manageable. Typical heuristic rules include removing stop words [**?**], allowing n-grams appearing in Wikipedia article titles as candidates [**?**], extracting noun phrases [**?**], and extracting n-grams or noun phrases satisfying pre-defined lexico-syntactic patterns [**?**]. In this paper, we generate candidate phrases using a frequency threshold rule.

Extracting top k keyphrases involves finding the most important phrases from a document. Typical unsupervised methods include ranking-based approaches like TF-IDF and TextRank [**?**], and clustering-based methods. TF-IDF reflects phrase importance to a document within a collection. TextRank extends the PageRank algorithm [**?**], a graph-based ranking method originally used for Web pages. TextRank' s premise is that a candidate' s importance depends on its relationships with other candidates—more related candidates increase importance, and connections to important candidates further boost importance. Researchers have computed relatedness using co-occurrence counts [**?**] and semantic relatedness [**?**], applying TextRank to generate top k keyphrases [**?**].

For supervised methods, feature design and modeling are crucial. Berend and Farkas [**?**] developed the SZTERGAK system to extract features, categorizing them into standard, phrase-level, document-level, corpus-level, and external knowledge-based features. While effective, additional features remain to be discovered, such as semantic embedding features for phrases and documents.

Keyphrase extraction is treated as a binary classification task. The modeling goal is to train a classifier determining whether a candidate phrase is a keyphrase, using annotated keyphrases as positive examples and other candidates as negative examples. Researchers have experimented with various learning algorithms, including naïve Bayes [**?**], decision trees [**?**], maximum entropy [**?**], multi-layer perceptron [**?**], and support vector machines [**?**]. Gradient boosting is a machine learning technique for regression and classification, and eXtreme Gradient

Boosting (XGBoost) is an optimized distributed gradient boosting library designed for efficiency, flexibility, and portability [**?**]. For our keyphrase extraction task, we choose XGBoost, with inputs being features of <doc, phrase> pairs, where doc represents a document and phrase is a candidate extracted from it.

2.2 Candidate Phrase Generation

In the SMP Cup 2017 keyphrase extraction task, organizers provided 1,022 documents with five annotated keyphrases each. These annotated keyphrases served as positive examples, while other candidates were negative examples.

2.2.1 Procedure Existing candidate phrase generation methods work well for English documents but are unsuitable for Chinese documents due to Chinese word segmentation challenges. Moreover, annotated keyphrase formats in the training set are highly diverse: single Chinese characters, Chinese phrases (mostly 2-7 characters), English words, English phrases, mixed English-Chinese phrases, and even English phrases with special characters like c++, node.js, and utf-8. This diversity significantly increases candidate generation difficulty.

To address these challenges, we adopt a heuristic procedure to automatically generate candidate phrases [Figure 1: see original paper]:

1. Use the total 1 million documents as a corpus.
2. Extract Chinese content from each document.
3. Use Jieba' s seg-for-search-engine mode to obtain Chinese phrase segmentation results.
4. Extract Chinese phrases with 2-7 characters.
5. Filter phrases from step 4 using phrase frequency.
6. Traverse the corpus to extract English phrases or combination phrases (English words + Chinese characters from step 5), ordering by frequency to generate non-Chinese format candidates.
7. Combine candidate phrases from steps 5 and 6.

Since most keyphrases in the training set and corpus are Chinese, we handle Chinese and English phrases separately in step 2. For Chinese phrases, the challenge is balancing candidate phrase quantity with training set annotated keyphrase coverage. Initially, using Jieba alone included only a small proportion of annotated keyphrases. Extracting all possible 2-7 character phrases caused explosive growth. A heuristic rule states that a Chinese phrase' s minimum unit is a word, not a character. Therefore, we use Jieba' s seg-for-search-engine mode to extract all possible words and their positions, then construct all possible 2-7 character phrases (steps 3-4). In step 5, we calculate phrase frequency across the corpus and use high-frequency phrases as Chinese candidates. We then traverse the corpus again to extract English-only and combination phrases. Finally, we merge all candidates from steps 5 and 6.

2.2.2 Results We extract 301,076 candidate phrases total. Phrases with special characters are excluded due to their abundance in the corpus. Since our document volume is small, we extract special-character phrases per document. We generate 239,405 training examples from 1,022 training documents and 233,580 validation examples from 1,022 validation documents. Table 1 shows the relationship between candidate phrases in training and validation examples. Our candidates cover over 90% of annotated keyphrases. Differences between rows 1 and 2 stem from data noise—some annotated keyphrases don' t exist in their documents. "Not in candidate phrases" examples in rows 2 and 3 are phrases with special characters.

2.3 Features

2.3.1 Statistical Features Following Berend and Farkas [**?**], we categorize statistical features into four groups: phrase-level, document-level, corpus-level, and external knowledge-based features.

Phrase-level features. We generate 31-dimensional features based on phrase length and format: Chinese/English phrases, phrases containing special characters, special character locations, noun presence, counts of Chinese characters, letters, or special characters, subset relationships with other candidates (if phrase A is a subset of B, B is A' s super-phrase), etc.

Document-level features. These are calculated from candidate phrases and their documents. Intuitive features include phrase frequency, title appearance, and location features. We also include features of the document and sentence containing a candidate. Another important type involves relationships between candidates and generated document results. For example, we analyze top 10 keywords, top 10 keyphrases, and top 3 key sentences using TextRank, calculating whether a candidate appears in these results. We compute 27-dimensional document-level features total.

Corpus-level features. We extract four features: phrase frequency in the entire corpus, TF-IDF value, IDF value, and total titles containing the phrase.

External knowledge-based features. We use Wikipedia and Baidu Encyclopedia entry data, extracting four features: whether a candidate is a Wikipedia/Baidu Encyclopedia entry, and whether it has a super-phrase that is such an entry.

2.3.2 Semantic Features Semantic features refer to word embedding features. In this task, we extract candidate phrase embedding vectors, embedding vector statistics, and distance features between candidates and their documents. The main procedures are:

1. Generate embedding vectors for every corpus word. We use two open-source libraries: word2vec and GloVe. While GloVe focuses on global

information, word2vec emphasizes local information. Dimension is an important hyper-parameter; we set it to 128 and 256, generating four embedding vector types per word.

2. Generate phrase and document embedding vectors. We represent candidate phrase and document vectors by summing their constituent word embeddings. These vectors can be normalized.

3. Calculate distance and distribution features. Since document titles often contain main ideas, we use title embedding vectors (instead of full document vectors) to calculate candidate distance features: cosine similarity, Cityblock distance, Jaccard coefficient, Canberra distance, Euclidean distance, Minkowski distance, Bray-Curtis distance, and word mover' s distance [**?**]. Distribution features reflect embedding vector distributions in titles; we calculate skewness and kurtosis as distribution features.

2.4 Experimental Results

We extract 66-dimensional statistical features and 400-dimensional semantic features. Comparing both feature types using XGBoost with parameters eta = 0.03, max_{depth} = 8, subsample = 0.8, colsample_{bytree} = 1.0, alpha = 0, lambda = 1, and gamma = 0, we obtain validation set performance shown in Table 2 .

Table 2 shows statistical features outperform semantic features. Combining both significantly enhances performance, indicating semantic features contain complementary information. Using combined features as model input yields the best scores on both validation and test sets.

---

3. USER TAGGING

Labeling user interests in our dataset is challenging for two main reasons: (1) users may have multiple interests, and (2) labeled users are insufficient. To address these, we propose a two-phase model for user labeling.

3.1 Related Work

Boutell et al. [**?**] proposed binary relevance, which independently trains one binary classifier per label. For unseen instance x, labels with probabilities beyond the threshold are assigned. However, this ignores label correlations. Read et al. [**?**] leveraged correlations through classifier chains, forming a chain of binary classifiers where prior results become features for subsequent models. Unfortunately, this method doesn't scale well with large label spaces. Another approach uses neural networks with softmax loss, assigning the top r probability labels to users. However, with small training sets, high-dimensional features likely cause overfitting.

3.2 Problem Formulation

Let D_{train} = {(U_1, L_1), (U_2, L_2), ···, (U_N, L_N)} denote the training set containing |D_{train}| = N users, and D_{test} = {(U_1, L_1), (U_2, L_2), ···, (U_M, L_M)} denote the test set containing |D_{test}| = M users, where U_i = (U_{i1}, U_{i2}, ···, U_{id}) represents d features of user U_i, and L_i = (L_{i1}, L_{i2}, ···, L_{ir}) denotes r labels per user. Let LS = {ls_1, ls_2, ···, ls_k} denote the label space containing k labels, where for (U_i, L_i), L_i LS. Table 3 summarizes the notations. For simplicity, we denote both D_{train} and D_{test} as D. We define our problem' s input and output as:

Input: A set of N users in D_{train} with corresponding r labels.

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

3.3 Two-Phase Model

Section 3.2 formulates user labeling as a multi-label classification problem, where we tag each user from a given label space. Unlike traditional multi-label classification where sample label counts vary, our task requires assigning the same number of labels per user. This section addresses both insufficient samples and label correlations.

Figure 2 [Figure 2: see original paper] illustrates our two-phase model framework. The first phase involves independent binary classifications (dark rectangles), and the second phase a neural network (light rectangles). In phase one, we formulate each label as a binary classification problem, obtaining k classifiers after training. In phase two, we train a neural network using phase one' s predicted probabilities as input, with softmax as the loss function. Finally, the r labels with highest probabilities are selected as user interests.

3.3.1 Independent Binary Classifications We transform the training and test sets as follows: for ls_i in label space LS, if ls_i 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. Denoting D_i as training samples for label ls_i, we train k binary classifications independently on D_i. We leverage bagging to reduce generalization errors.

Bagging strategy. We randomly divide D_i into c folds, using continuous c-1 folds for training and the remaining 1 fold for validation, yielding c classifiers for D*i. After training, we learn k × c classifiers, where k is the label space size and c is classifiers per label. When predicting unlabeled users in D*{test}, user features are fed to c classifiers per label, and the mean of predicted probabilities becomes input for phase two.

3.3.2 Softmax Classification Labels corresponding to the highest r probabilities from independent binary classifications could be assigned heuristically, but this has two weaknesses. First, probabilities from different labels aren' t

comparable—a higher probability may not indicate a true user label since models differ. Second, label relevance (a strong prior) isn' t considered.

Our phase two model addresses these weaknesses with a neural network using softmax loss. Figure 3 [Figure 3: see original paper] shows the three-layer architecture. The input layer contains r units with probabilities from independent binary classifiers; the output layer contains r units. The hidden layer uses dropout to avoid overfitting. Softmax loss is defined as:

f(P, L) = ···

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

f(P, L) = L × log(P)

We also adopt bagging to reduce generalization loss. When predicting unlabeled users, labels with highest probabilities become the user' s labels.

3.4 Experiments

3.4.1 Data Collection Given blog content, user behaviors, and structural/chatting relationships between CSDN users, we must tag each user with three labels from a 42-label space (e.g., machine learning, Web development). Table 4 summarizes the data collection statistics.

Table 4. Statistics of data sets.

Metric	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 Content features prove most helpful, based on the intuition that users with similar interests tend to interact with similar blogs through posting, browsing, commenting, favoriting, voting down, and voting up.

Doc2Vec [27]. Doc2vec is an unsupervised algorithm learning fixed-length feature representations from variable-length text pieces, embedding documents into distributed vectors. We believe users associated with similar blogs have similar interests. User features are defined as blog representations across six behavior types (browse, post, comment, add favorites, vote down, vote up). In our experiments, blogs are represented by 200-dimensional vectors, yielding 1,200-dimensional user vectors (200 dimensions × six behavior types).

3.4.3 Evaluation Metrics and Baselines We evaluate our model using the following metric:

score = ···

where M is dataset cardinality, L_i* are predicted labels and L_i are ground truth. For each user, the score is 1, 2/3, or 1/3 if 3, 2, or 1 label(s) are predicted correctly. Higher scores indicate better performance.

We compare the following user labeling methods:

- **Top r labels (TL):** Tags each user with the r most popular labels in the training set (r = 3 in our dataset).
- **Independent binary classifications (IBC):** Our model' s first phase, tagging unlabeled users with labels corresponding to the highest r probabilities from independent binary classifications.
- **Softmax classification (SC):** Our model' s second phase, using features from Section 3.4.2 (instead of probabilities) with a softmax neural network, assigning top r labels to each user.

3.4.4 Performance Analysis Experiments on the Section 3.4.1 dataset use a test set of 1,055 users. Table 5 shows our method outperforms others. Since IBC doesn't consider label correlations, probabilities from independent classifiers aren' t comparable, while our model' s second phase captures these correlations. SC has far more parameters than training samples, causing overfitting.

Table 5. Comparison of method scores.

Method	Score
Our method	*value*
TL	*value*
IBC	*value*
SC	*value*

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

4. USER GROWTH VALUE PREDICTION

This task requires predicting user growth value for the next year based on past posts and behaviors. Each user' s growth value is normalized to [0,1], where 0 represents user loss.

4.1 Architecture

User growth value prediction can be formalized as a typical regression task: given feature x_i containing statistics of user i' s behaviors and posts, predict future growth value y_i. Figure 4 [Figure 4: see original paper] shows our architecture, consisting of a features layer, primary model, and model ensemble.

4.2 Features

4.2.1 Statistics of User Behaviors Frequency of behavior. We count behaviors triggered by a user during a period (e.g., one quarter): number of posts published, viewed, commented on, collected, voted up/down, private messages received/sent. These features reflect user activity and have been used in customer lifetime value prediction [**?**].

Change of behavior frequency. We calculate behavior frequency changes, particularly subtracting last two quarters' post frequency from the last quarter' s frequency.

The last active time. We identify the most recent time for each behavior type (writing posts, commenting, viewing). These features also reflect user activity.

Proportion of behavior types. We calculate each behavior type's proportion over all behaviors during a period (e.g., one quarter). These features characterize user preferences (e.g., preferring writing over browsing).

Number of retweeted posts. We distinguish original posts from retweeted posts using the MinHash algorithm [**?**], assuming similar posts make the oldest original and others retweeted. We calculate a 64-byte hash per post, using the first 16 bytes as a key to avoid comparing all pairs. We count original posts published per quarter as a feature reflecting user quality and creativity.

User points in the past year. We calculate user points based on CSDN' s published point rules, reflecting user quality.

4.2.2 Statistics of Documents Length of documents. We calculate average length and variance of documents written by a user, reflecting document characteristics.

Feedback received by documents. We count behaviors (browsing, commenting, voting up/down) and feedback received for a user' s posts during a period (e.g., one quarter), reflecting document quality.

4.2.3 Representation of User-Document We extract information from user-document interaction matrices, constructing six types: browse, comment, vote-up, vote-down, collection, and all-behaviors matrices. Elements represent behavior frequency during the past year. We decompose these matrices using non-negative matrix factorization (NMF) [**?**] and use user representations as features, reflecting behavioral similarity in a transformed space.

4.2.4 Representation of User-User Matrix factorization. Similar to user-document representation, we construct following-relation and private-message matrices, decomposing them with NMF and using user vectors as features.

PageRank. We construct a user-user interaction network reflecting influence: if user A browses user B' s article, an edge likely exists from A to B. We run PageRank [**?**] on this network to calculate each user' s importance.

4.3 Primary Model

We adopt four significantly different primary models: Multi-layer Perceptron (MLP), Support Vector Regression (SVR), XGBoost, and Random Forest. For each, we employ sampling to maintain generality (Figure 5 [Figure 5: see original paper]). After sampling k features K times and n samples N times, we obtain N $\times$ K different training datasets per model, yielding N $\times$ K trained models. Final predictions combine all N $\times$ K results using median bagging. Below we describe each model and its performance, with evaluation score defined in Equation (6):

score = $\cdots$

where v_i is true user growth value, v_i* is predicted value, and N is the number of users to predict.

4.3.1 Multi-Layer Perceptron (MLP) Using TensorFlow [**?**], we easily modify the optimized loss to match the evaluation score. With only 1,000 samples risking overfitting, our MLP contains only input and output layers, achieving a prediction score of 0.736. To ensure feature effectiveness, we calculate each feature' s weight variance across N $\times$ K trained models, filtering out high-variance features (average weight/variance $>$ 30%), assuming these are ineffective. Preserving 41 features improves the final score to 0.752 (Table 6), a significant improvement.

4.3.2 Support Vector Regression (SVR) Since the evaluation score lacks a second-order derivative, the remaining three models cannot optimize directly toward it. We adopt an approximated loss function:

log(v_i) - log(v_i*)

where v_i is true growth value and v_i* is predicted value. SVR [**?**] calculates loss only when |v_i - v_i*| $>$, with as a manually adjusted hyper-parameter.

Grid search yields optimal parameters and a prediction score of 0.754 (Table 6), similar to MLP performance.

4.3.3 XGBoost XGBoost [**?**] is an optimized Gradient Boosting Decision Tree variant, much faster than traditional versions. Grid search finds optimal hyper-parameters, reaching a score of 0.750.

4.3.4 Random Forest Random Forest regression combines many decision trees, averaging their results as final output [**?**]. Grid search yields optimal hyper-parameters and a prediction score of 0.737. This model' s less competitive performance may stem from its complexity, making parameter tuning difficult.

4.4 Model Ensemble

With only 1,000 labeled samples, we face significant learning challenges. We propose two solutions: semi-supervised regression to extend labeled samples, and stacking to ensure model generality.

4.4.1 Semi-Supervised Regression Following co-training principles for regression [**?**, **?**], we select the two best models (MLP and SVR) to extend labeled samples. Algorithm 1 describes our semi-supervised regression approach. We set maximum iterations T = 500 and cache pool capacity K = 200. However, semi-supervised regression is slow, averaging 10 added labeled samples per day. Due to time constraints, we extend the dataset to only 1,066 labeled samples. Both MLP and SVR performance improve slightly, with evaluation scores increasing by 0.001 each (Table 6).

4.4.2 Stacking Stacking [**?**] is an ensemble learning method combining primary models for more robust results. We use four single models (MLP, SVR, Random Forest, XGBoost) plus semi-supervised regression versions of MLP and SVR as primary models. We train another SVR on these models' outputs as features. The final stacked model achieves the best performance, reaching a prediction score of 0.764.

---

5. CONCLUSIONS AND FUTURE WORK

This paper describes our work for the User Profiling Technology Evaluation Campaign in SMP Cup 2017. For keyphrase extraction, we treat it as a classification problem and use XGBoost to predict top three keyphrases, achieving the best participant score. For user tagging, our two-phase model addresses two challenges: phase one' s independent binary classifications encode high-dimensional features into low-dimensional representations supervisedly, with fewer parameters than traditional softmax classifiers to avoid overfitting given insufficient labeled data; phase two' s neural network with softmax loss assigns most likely

labels, implicitly modeling label correlations and making independent classifier probabilities comparable. For user growth value prediction, we overcome training data scarcity through semi-supervised regression and stacking. Experiments demonstrate our framework' s effectiveness, achieving 0.764 accuracy.

Future work will further improve user profiling performance. For keyphrase extraction, we will consider state-of-the-art document representation models (e.g., LSTM, hierarchical attention) to extract more features. For user tagging, we will explicitly exploit class structures (e.g., label taxonomy) to improve multi-label classification. For user growth value prediction, point process methods can capture user growth value characteristics over time.

AUTHOR CONTRIBUTIONS

All authors contributed equally. H. Shen (shenhuawei@ict.ac.cn) coordinates the research group and designed the user profiling framework. G. Xing (xingguoliang@ict.ac.cn) and X. Yue (yuexinyu@ict.ac.cn) co-contributed to keyphrase extraction. H. Gao (gaohao@ict.ac.cn) and B. Xu (xubingbing@ict.ac.cn) co-contributed to user tagging. Q. Cao (caoqi@ict.ac.cn) and K. Cen (cenketing@ict.ac.cn) co-contributed to user growth value prediction. All authors contributed to writing and revising this manuscript.

ACKNOWLEDGEMENTS

This 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.

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/S0218

[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/

[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.

[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.

[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/

[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-

[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.

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.

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

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.

Note: Figure translations are in progress. See original paper for figures.