CFTop: Using collaborative filtering to recommend Github topics

ABSTRACT

Collaborative filtering is a well-founded technique spreadly used in the recommendation system domain. During recent years, a plethora of approaches has been developed to provide the users with relevant items. Considering the open-source software (OSS) domain, GitHub has become a precious service for storing and managing software source code. To represent the stored projects in an effective manner, in 2017 GitHub introduced the possibility to classify them employing topics. However, assigning wrong topics to a given repository can compromise the possibility of helping other developers reach it and eventually contribute to its development. In this paper, we present CFTop, a recommender system to assist open source software developers in selecting suitable topics to the repositories. CFTop exploits a collaborative filtering technique to recommend libraries to developers by relying on the set of initial topics, which are currently included in the project being. To assess the quality of the approach, we exploit a recent work in this domain and validate both of them using different metrics. The results show that CFTop outperforms it in all the examined aspects. More interesting, the chain of the two approaches lead an improvement of the prediction performances.

CCS CONCEPTS

• **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

datasets, collaborative filtering, topic recommender

ACM Reference Format:

. 2018. CFTop: Using collaborative filtering to recommend Github topics. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In recent years, the developer community heavily exploits open source repositories during their daily activities. GitHub has become one of the most popular platforms that aggregate these projects and support the development activity in a collaborative fashion. The platform recently introduced *topics* to foster the popularity and promote information discovery about popular projects. They are a set of terms used to characterize projects by summarizing their features. Thus, the topic labeling activity can compromise the popularity and reachability of a project if it is not properly addressed. A recent work already faced this problem by using a machine learning approach to recommend relevant topics given a README file of a repository [?]. However, this tool[1] is able to recommend only *featured* topics, a curated list of them provided by Github[2].

In this work, we extend the set of recommended items to non-featured topics by exploiting collaborative filtering, a widely spread technique in the recommendation system domain [28]. Given an initial set of topics coming from a GitHub project, we use repository-topic matrixes to suggest relevant topics. The work gives the following contributions:

- Considering the GitHub projects as products, we suggest relevant topics to the project given an initial list of them;
- We assess the quality of the work employing a well-defined set of metrics commonly used in the recommendation system domain i.e., success rate and accuracy;
- Considering a well-founded approach, we improve it by providing an extended set of possible topics

The rest of the work is structured as follows. Section 2 shows the issues and the potential challenges in the domain. In Section 3, we present our approach and evaluate it in Section 4. We present the results of the assessment in Section 5 and we discuss the findings. Section 6 summarizes relevant works in the field and we conclude the paper in Section 7 with possible future works.

2 BACKGROUND

Manually assigning topics can be an error-prone activity that can lead to wrongly specified tags. Over the last years, several attempts have been made to *classify* GitHub projects by automatically inferring appropriate topics. In the context of data mining, *classification* is one of the critical operations that are used to dig deep into available data for gaining knowledge and for identifying repetitive patterns [?].

With the aim of contributing the resolution of the problem of recommending GitHub topics, in the next section we propose to use item-based collaborative filtering to recommend relevant topics. The challenges that we had to cope with for evaluating its performance are mainly the following ones:

▹ *Dataset definition:* the creation of the datasets to be used for evaluating the approach being proposed and comparing it with some baseline is a daunting task: repositories might be moved, heavily changed or even deleted during the initial creation. Thus, the crawling activity can be negatively affected by these continuous

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

[1]For the sake of presentation, we refer to this work as MNB network throughout the paper

[2]https://github.com/topics

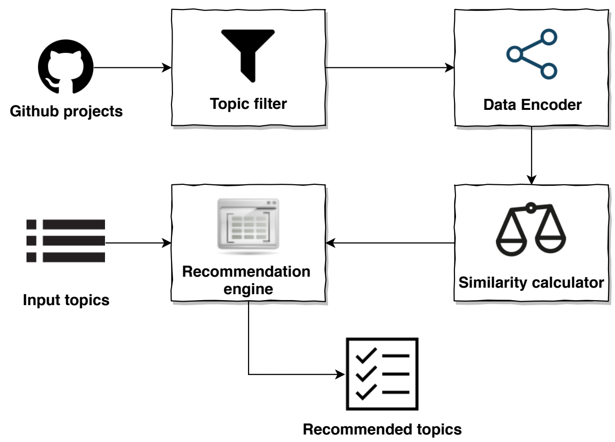


Figure 1: Overview of the CFTop Architecture.

changes and lead to lack of data, and poor topic coverage. GHTorrent[3] tries to mitigate this issue by offering daily dumps of the repositories' metadata. However, this kind of data might not be enough or even appropriate (e.g., source code is not available in GHTorrent dumps) to properly classify an entire repository. Even considering directly GitHub data can be difficult: GitHub limits the total number of requests per hour to 5,000 for authenticated users and 60 for unauthorized requests. Considering all these constraints, building a suitable dataset represents a real challenge to be managed carefully.

▹ *Topics distribution:* although tags can be assigned only by the owners of GitHub repositories, users can potentially wrongly specify topics or introduce information overload by inserting too many elements. Thus, creating a reliable ground truth to assess the classification performance of the proposed approach represents another relevant difficulty.

3 PROPOSED APPROACH

In this section, we describe CFTop that provides developers with relevant topics for GitHub repositories. More specifically, CFTop is a *recommender system* [3] that encodes the relationships among different topics by means of a graph and utilizes a collaborative filtering technique [28] to recommend GitHub topics. Such a technique has been used mostly in the e-commerce domain to exploit the relationships among users and products to predict the missing ratings of recommended items [16]. The technique follows the assumption that *"if users agree about the quality or relevance of some items, then they will likely agree about other items"* [28]. Under the same premise, our tool aims to solve the problem of the reachability of a GitHub repository given a set of topics. Instead of recommending goods or services to customers, we recommend a set of topics using an analogous mechanism: *"if a user tags his project with some topics, then similar projects will probably contain common topics."*

[3]http://ghtorrent.org/

To this end, the architecture of CFTop is shown in Fig. 1, and consists of the software components supporting the following activities:

- *Representing the relationships* among projects and topics retrieved from existing repositories;
- *Computing similarities* to find projects, which are similar to that under development; and
- *Recommending topics* to projects using a collaborative-filtering technique.

In a typical usage scenario of CFTop, we assume that a developer is creating a new GitHub repository, in which she has already included some topics to improve its reachability. As shown in Fig. 1, the developer interacts with the system by demanding for recommendations. Such a request contains a list of topics that are already included in the project the developer is working on. As a preprocessing phase, we apply a *Topic filter* according to their frequencies i.e., the measured occurrences over all repositories in the initial dataset. The `Graph Encoder` represents the mentioned repositories in the graph format. This is a preparatory phase for the next steps of the recommendation process. The `Similarity Calculator` module computes similarities among topics to discover similar ones to recommend. The `Recommendation Engine` implements a *collaborative-filtering* technique [3],[31], it selects top-k similar topics, and performs computation to generate a ranked list of *top-N* topics. Finally, the final list of topics is sent back to the developer.

The aforementioned components are singularly described in the next sections.

3.1 Topic filter

As a preprocessing, we filter the initial set of topics using their frequencies counted on the entire GitHub dataset. We remove irrelevant topics to reduce the noise in the prediction phase. Through the *cut-off* value, we progressively increase the frequency threshold to evaluate possible impacts on overall performances. As stated in [12], this preprocessing can improve the final results, thus we decide to apply it as a first step.

3.2 Data Encoder

Considering traditional recommender systems for online services, we can identify three main components, namely *users*, *items*, and *ratings* [27],[22]. All mutual relationships among system components are encoded in a *user-item ratings matrix*. Specifically, in the matrix a user is represented by a row, an item is represented by a column and each cell in the matrix corresponds to a rating given by a user for an item [22]. Moving to our domain, users are substitute by projects as well as topics are the possible items to recommend. The analogous user-item ratings matrix represents possible relationships between these two elements i.e., project may include various topics. We can denote *project-library inclusion* relationships as $\ni$. In this matrix, each row represents a project and each column represents a topic. A cell in the matrix is set to 1 if the topic in the column is included in the project specified by the row, it is set to 0 otherwise. For the sake of clarity and conformance, we still denote this as a user-item ratings matrix throughout this paper.

For explanatory purposes, we consider a set of four projects $P = \{p_1, p_2, p_3, p_4\}$ together with a set of topics $L = \{topic_1$=*machine-learning*; $topic_2$=*javascript*; topiclib$_3$=database; $topic_4$=*web*; $topic_5$=*algorithm*$\}$. By extracting the list of defined topics of the projects in P, we discovered the following inclusions: $p_1 \ni topic_1, topic_2$; $p_2 \ni topic_1, topic_3$; $p_3 \ni topic_1, topic_3, topic_4, topic_5$; $p_4 \ni topic_1, topic_2, topic_4, topic_5$. Accordingly, the user-item ratings matrix built to model the occurrence of the topic is depicted in Fig. **??**.

3.3 Similarity Calculator

The `Recommendation Engine` of CFTop works by relying on the mentioned user-item ratings matrix. To provide inputs for this module, the first task of CFTop is to apply a similarity function on its input data to find the most similar topics to a given initial set. Computing properly this similarity score affects the quality of recommendation outcomes.

Nonetheless, computing similarities among topics could be a daunting task. GitHub allows any repository owner to add, change, or delete the list of topics that describe his project []. This impacts on the stability of the topics, as they can change rapidly over time. In addition, a developer can freely specify the entire set of topics. This makes the similarity computation more complicated, as some topics couldn't have a semantic link with the others. Moreover, we can miss some key relationships depending on the similarity function employed by the calculator. For example, a purely sintactic-based similarity function assign a lower score to the topic pair 3d-graphics even though these two terms are strongly bounded in their meaning.

We assume that a representation model that addresses mutual relationships among GitHub repositories and their topics is profitable to proposed similarity computation. To this end, we derive a *graph-based* model to represent this kind of relationships and eventually to calculate similarities. In the context of mining OSS repositories, the graph model is a convenient approach since it allows for flexible data integration and numerous computation techniques. By applying this representation, we are able to transform the set of projects and topics into a directed graph as in Fig. 2. We adopted the approach in [19],[20] to compute the similarities among OSS graph nodes. It relies on techniques successfully exploited by many studies to do the same task [11],[8]. Among other relationships, two nodes are deemed to be similar if they point to the same node with the same edge. By looking at the graph in Fig. 2, we can notice that p_1 and p_2 shares two nodes, namely $topic_2$ and $topic_3$. From the graph, we can also learn additional information about the topics themselves. For example, $topic_3$ seems a very popular term since is pointed by three different projects. In the meanwhile, $topic_1$ and $topic_4$ are used only by one project at once, p_1 and p_3 respectively.

Using this metric, the similarity between two project nodes p and q in an OSS graph is computed by considering their feature sets [11]. Given that p has a set of neighbor nodes $(topic_1, topic_2, .., topic_l)$, the features of p are represented by a vector $\overrightarrow{\phi} = (\phi_1, \phi_2, .., \phi_l)$, with ϕ_i being the weight of node $topic_i$. It is computed as the *term-frequency inverse document frequency* value as follows:

$$\phi_i = f_{topic_i} \times log(\frac{|P|}{a_{topic_i}}) \quad (1)$$

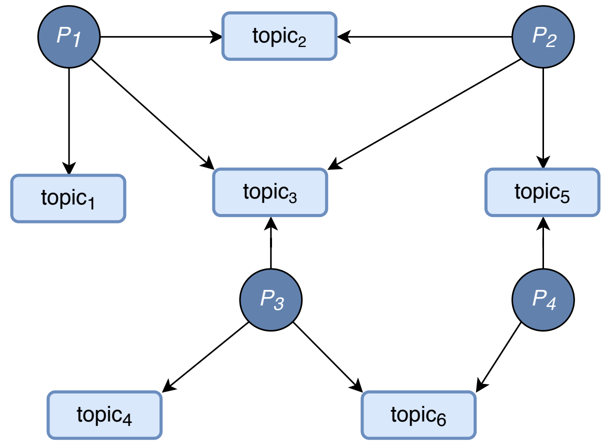


Figure 2: Graph representation for projects and topics

where f_{topic_i} is the number of occurrence of $topic_i$ with respect to p, it can be either 0 and 1 since there is a maximum of one $topic_i$ connected to p by the edge *includes*; $|P|$ is the total number of considered projects; a_{topic_i} is the number of projects connecting to $topic_i$ via the edge *includes*. Eventually, the similarity between p and q with their corresponding feature vectors $\overrightarrow{\phi} = \{\phi_i\}_{i=1,..,l}$ and $\overrightarrow{\omega} = \{\omega_j\}_{j=1,..,m}$ is computed as given below:

$$sim(p, q) = \frac{\sum_{t=1}^{n} \phi_t \times \omega_t}{\sqrt{\sum_{t=1}^{n} (\phi_t)^2} \times \sqrt{\sum_{t=1}^{n} (\omega_t)^2}} \quad (2)$$

where n is the cardinality of the set of topics that p and q share in common [11]. Intuitively, p and q are characterized by using vectors in an n-dimensional space, and Eq. 2 measures the cosine of the angle between the two vectors.

3.4 Recommendation engine

The representation using a user-item ratings matrix allows for the computation of missing scores [3],[22]. Depending on the availability of data, there are two main techniques to compute the unknown ratings, namely *content-based* [24] and *collaborative-filtering* [18] recommendation techniques. Focusing on the latter, this technique computes the ratings by taking into account the set of items rated by similar customers. There are two main types of collaborative-filtering recommendation: *user-based* [31] and *item-based* [27] techniques. As their names suggest, the user-based technique computes missing ratings by considering the ratings collected from similar users. Instead, the item-based technique performs the same task by using the similarities among items [9].

In the context of CFTop, the term *rating* describes the appearance of a topic in a project and the employed collaborative filtering techniques aim to find additional similar topics. The project that needs prediction for topic suggestion is called the *active project*. By the matrix in Fig. 3, p is the active project and an asterisk (∗) represents a known rating, either 0 or 1, whereas a question mark (?) represents an unknown rating and needs to be predicted.

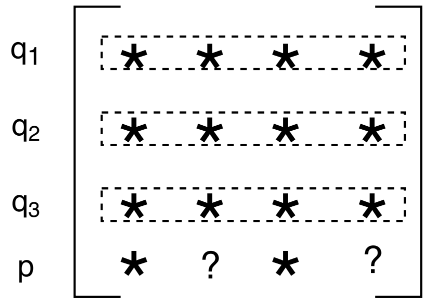


Figure 3: Computation of missing ratings using the user-based collaborative-filtering technique [31].

We can employ the proposed engine into two different ways *(i)* as a stand-alone given an initial set of topics or *(ii)* using MNB network results to enable the collaborative filtering based recommendation. Consider the mutual relationships between a project and its topics represented in a graph data structure, we exploit the user-based collaborative-filtering technique to enable the topic recommendation process [16, 31].

Given an active project p, the inclusion of libraries in p can be deduced from projects that are similar to p. The process is summarized as follows:

- Compute the similarities between the active project and all projects in the collection;
- Select *top-k* most similar projects; and
- Predict ratings by means of those collected from the most similar projects.

The rectangles in Fig. 3 imply that the row-wise relationships between the active project p and the similar projects q_1, q_2, q_3 are exploited to compute the missing ratings for p. The following formula is used to predict if p should include l, i.e., $p \ni l$ [22]:

$$r_{p,l} = \overline{r_p} + \frac{\sum_{q \in topsim(p)} (r_{q,l} - \overline{r_q}) \cdot sim(p,q)}{\sum_{q \in topsim(p)} sim(p,q)} \quad (3)$$

where $\overline{r_p}$ and $\overline{r_q}$ are the mean of the ratings of p and q, respectively; q belongs to the set of *top-k* most similar projects to p, denoted as $topsim(p)$; $sim(p,q)$ is the similarity between the active project and a similar project q, and it is computed using Equation 2.

3.5 Entanglement with MNB network

So far, we have described CFTop as a stand-alone recommender system by detailing all the involved components in the process. To highlight its flexibility in a different context, we *entangle* our tool with the MNB network using it as a black box. As mentioned before, this recent work using the README file of a repository to predict featured topics. It involves all the standard techniques employed in the ML domain i.e., textual engineering, feature extraction, and training phase. Given a README file, the approach computes vectors using the TF-IDF weighting scheme to extract features. Then, the model is trained to retrieve the most probable featured topics according to the multinomial distribution with the Naive Bayesian assumption. The outcomes are evaluated using the ten folder validation process. In the landscape of our work, we consider the set of featured topics predicted by the MNB model as the input of CFTop. The aim of this kind of analysis is to evaluate CFTop capability using a well-founded technique in the literature.

4 EVALUATION

In this section, we report how CFTop has been evaluated, having the *goal* of evaluating the performance of the proposed approach. In Section 4.1, the dataset involved in our evaluation has been presented. We describe the evaluation methodology and metrics in Section 4.2. Finally, Section 4.3 describes the research questions.

4.1 Dataset Extraction

To evaluate the approach, we reuse the same dataset employed for the MNB network available here [29]. The GitHub query language [2] allows the fetching of relevant repository metadata including name, owner, and list of topics to mention a few. Thus, we *randomly* collected a dataset consisting of 6,258 repositories that use 15757 topics by means of the GitHub API [1]. We employ the GitHub star voting mechanism as a popularity measure to avoid including unpopular, unmaintained and toy projects [6]. As claimed in several works[5, 7], a high number of stars means the attention of the community for that project. So, we impose the following filter during the query execution:

$$Qf = \text{"}is : featured\ topic : t\ stars : 100..80000\ topics :>= 2\text{"} \quad (4)$$

to consider only GitHub repositories having a number of stars between 100 and 80,000, and tagged with at least two topics. The boolean qualifier *is:featured* is used in the MNB network work to group repositories given a certain featured topic (please refers to https://github.com/topics for the complete list of featured topics). As CFTop is able to retrieve both featured and not-featured topics, this filter doesn't affect the quality of the collected data. To investigate the CFTop prediction performances, we populated five different datasets by variating the topic frequency cut-off value t i.e., the maximum frequency of the topic distribution (it will be better described in Section 4.2). In this way, we remove the infrequent elements from the dataset to analyze the impacts on the recommendation phase as well as on the composition of the dataset. Table 1 summarizes the datasets' features with t = 1, 5, 10, 15, 20.

Dataset	No. of repos	No. of topics	Avg topics for repo	Avg freq. for topic
Dt_1	6,253.0	15,743.0	9.9	4.0
Dt_5	3,884.0	1,989.0	8.0	17.0
Dt_{10}	2,897.0	964.0	8.0	24.0
Dt_{15}	2,273.0	634.0	7.7	28.0
Dt_{20}	1,806.0	456.0	7 .7	30.0

Table 1: Datasets' description.

As we can see in the next section, removing the infrequent topics improves the overall quality of the considered datasets. Similarly to the other collaborative filtering approaches, the overall prediction performance strongly depends on the dataset. As we will demonstrate in the next section, the collaborative filtering provides better prediction performance when there are enough data (i.e., topics) in the training set to resemble the repository behaviour. Juri ►*Check this sentence, 5 is a magic number*◄ . After infrequent topics is removed, the repository that consist of less then 5 topics are filter out from the dataset because they contain very few information to enable the

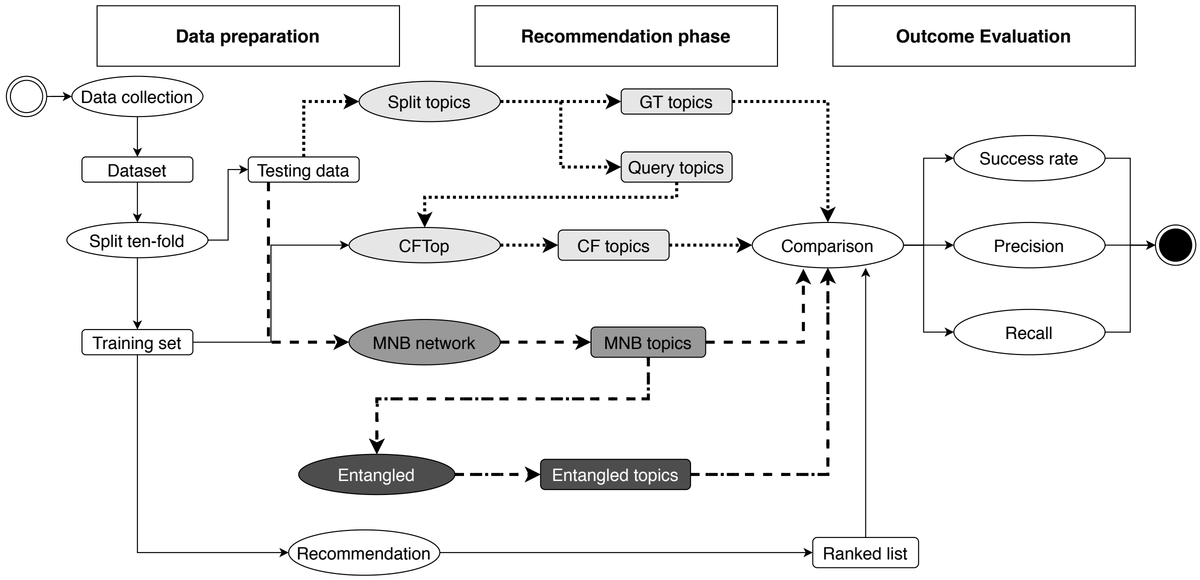


Figure 4: Evaluation Process.

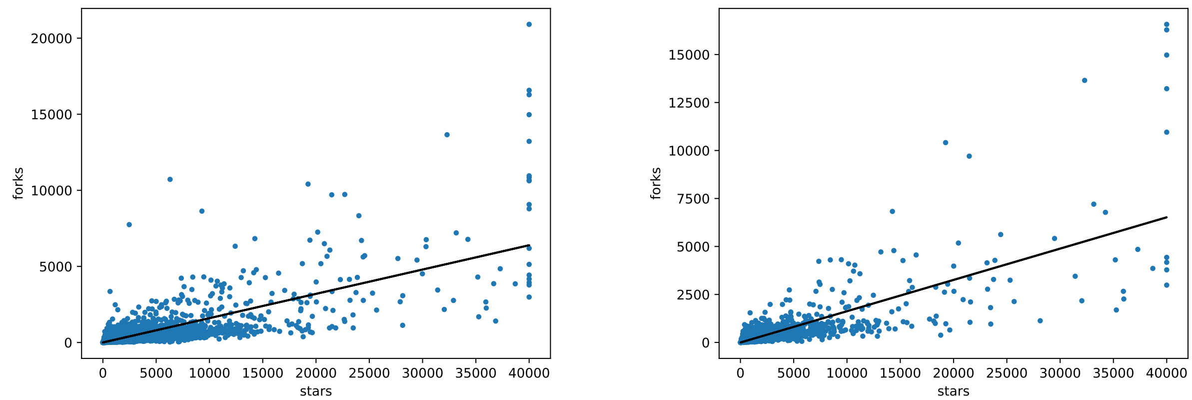
collaborative filtering prediction. In particular, we remove around 2,300 repositories by increasing the cut-off value from 1 to 5. It means that the excluded repositories in Dataset Dt_5 are tagged with topics that rarely appear in the considered repositories. This finding is strengthened by the number of topics, which dramatically decreases to 1,989. The other datasets confirm this trend even though the delta of removed repositories goes down at each filtering step. Thus, we stop at t=20 and consider Dataset Dt_{20} as the best one according to our metrics. Additionally, we observe that repositories are tagged by 9.9 and 7.7 topics on average for $t = 1$ and $t = 20$ respectively. This demonstrates that a huge number of topics doesn't help the discoverability of a project.

Furthermore, we evaluate the quality of the OSS project belonging to the examined dataset. As mentioned before, the GitHub community assesses this aspect by mainly using forks and stars. Thus, we collect this data for each dataset using the same Github API library employed for the crawling. Figure 5 shows the comparison between Dataset Dt_1 and Dt_{20}. Due to space issues, we omitted the other datasets although they confirm the depicted distribution. As can see, filtering repositories by the t value helps to smooth the distribution. On one hand, the Dataset Dt_1 contains more repositories with a high forks number rather than the ultimate dataset i.e., it reaches around 20,000 forks against 15,000 with t=1 and t=20 respectively. On the other hand, the slope depicted in Dataset Dt_{20} is higher than the original dataset. This demonstrates that the trend of the distribution is more uniform by applying the filtering process. Although the process removes some high-ranked repositories, the final dataset mitigates issues.

4.2 Evaluation methodology and Metrics' Definition

Figure 4 depicts the evaluation process consists of three consecutive phases, i.e., *Data Preparation*, *Recommendation*, and *Outcome Evaluation*. *Data Preparation* phase collects repositories that match the requirements defined in previous section from GitHub. This dataset is used to evaluate CFTop, MNB network, and the combination of two. The dataset is then split into training and testing sets. The *Recommendation* phase follows three different flows, according to the required input and produced output of the three mentioned approaches. In particular, the common operations are in white while the three different evaluation flows are represented in a grayscale fashion (i.e., light grey, grey and dark grey boxes are related to CFTop, MNB network, and entangled approaches evaluation respectively). To enable CFTop, we extract a portion of topics from a given testing project i.e., the ground-truth part (it is defined as $GT(p)$ in the following). The left part is used as a query to produce recommendations (see the dotted line flow). As the MNB network uses the README file of a repository to predict a set of topics, this doesn't require any topic as input. Thus, the approach encodes the document relevant information in vectors using the TF-IDF weighting scheme. Then, to feed the network that delivers a set of topics (see the bold line). Finally, the entangled approach uses CFTop as the recommendation engine which is fed by the MNB network suggested topics (see dashed line flow). All the results are assessed in the *Outcome Evaluation* phase, which compares the recommendation results with those stored as ground-truth data to compute the quality metrics.

The *ten-fold cross-validation* methodology [14] has been used to assess the performance of CFTop, MNB network and combined approach where every time 9 folds are used for training and the remaining one for testing. For each testing project p, we randomly delete half topics and save it as ground truth ($GT(p)$) . The ground truth data will be used to validate the recommendation outcomes. The remaining half topics are used as query topics to the CFTop.



(a) Dataset Dt_1 (b) Dataset Dt_{20}

Figure 5: Quality analisis of the examined datasets.

The `Split topic phase` resembles a real development process where a developer has already included some topics in his repository and waits for recommendations i.e., additional topics to be incorporated. CFTop recommender system is expected to provide her with the other half, i.e., *GT(p)*.

Juri ►*Rephrase*◄ There are several metrics available to evaluate a ranked list of recommended items [22]. In the scope of this paper, *success rate* and *accuracy* have been used to study the systems' performance as already proposed by Robillard et al. [25] and other studies [30],[21]. The metrics considered during the outcome evaluation follows this notation:

- t is the frequency cut-off value of input topics (i.e., all topics that occur less than t times are removed from the dataset)
- $|t_{in}|$ is the size of topics that CFTop takes as input;
- N is the cut-off value for the recommended ranked list of topic;
- k is the number of neighbor projects exploited for the recommendation process;
- For a testing project r, a half of its topics are extracted and used as the ground-truth data named as *GT(r)*;
- $REC(r)$ is the *top-N* topics recommended to a repository r. It is a ranked list in descending order of real scores;
- If a recommended topic $t \in REC(r)$ for a testing project r is found in the ground truth of r (i.e., *GT(r)*), hereafter we call this as a topic *match*

If $REC_N(p)$ is the set of top-N items and $match_N(p)$ is the set of items in the *top-N* list that match with those in the ground-truth data, then the metrics are defined as follows.

Success rate@N. Given a set of testing projects P, this metric measures the rate at which a recommender system returns at least a topic match among *top-N* items for every project $p \in P$ [30]:

$$success\ rate@N = \frac{count_{p\in P}(|match_N(p)| > 0)}{|P|} \tag{5}$$

Juri ►*Check if it will be used*◄ **Success rate$_M$@N.** Given a set of testing projects P, this metric measures the rate at which a recommender system returns at least M topics match among *top-N* items for every project $p \in P$ [30]:

$$success\ rate_M@N = \frac{count_{p\in P}(|match_N(p)| >= M)}{|P|} \tag{6}$$

where the function *count()* counts the number of times that the boolean expression specified in its parameter is *true*.

Accuracy. Accuracy is considered as one of the most preferred *quality indicators* for Information Retrieval applications [26]. However, *success rate@N* does not reflect how accurate the outcome of a recommender system is. For instance, given only one testing project, there is no difference between a system that returns 1 topic match out of 5 and another system that returns all 5 topic matches, since *success rate@5* is 100% for both cases (see Eq. (6)). Thus, given a list of *top-N* libraries, *precision@N* and *recall@N* are utilized to measure the *accuracy* of the recommendation results. *precision@N* is the ratio of the *top-N* recommended topics belonging to the ground-truth dataset, whereas *recall@N* is the ratio of the ground-truth topics appearing in the N recommended items [19],[11],[10]:

$$precision@N = \frac{|match_N(p)|}{N} \tag{7}$$

$$recall@N = \frac{|match_N(p)|}{|GT(p)|} \tag{8}$$

4.3 Research Questions

By performing the evaluation, we aim at addressing the following research questions:

- **RQ$_1$**: *Which collaborative filtering configuration brings the best performance to CFTop?* To answer this question, we investigate different configurations to find the best one i.e., we variate the number of input topics T, the number of neighbours N and the considered number of outcomes N.

- $\mathbf{RQ}_2$: *How CFTop behave with respect the MNB network in terms of prediction performance?* Because of CFTop and MNB network are completely different in term of input data, we are interested in comparing them by considering many factors that can impact on the performance.
- $\mathbf{RQ}_3$: *Is the entangled approach able to improve CFTop's overall performance?* From an empirical point of view, it is relevant to analyze the combination of the two approaches and measure its performances.

We study the experimental results in the next section by referring to these research questions.

5 RESULTS

This section discusses the findings of the qualitative assessment. To address the formulated research questions, we perform three different experiments. Section 5.1 discusses the CFTop results by variating different parameters. We measure the predict performances of the MNB network in Section 5.2. Finally, Section 5.3 investigates the results obtained with the entangled approach i.e., the combination of the two previous approaches.

5.1 CFTop evaluation

$\mathbf{RQ}_1$: *Which collaborative filtering configuration brings the best performance to CFTop?* To find the best configuration in terms of prediction performances, we experiment with different CFTop configuration by variating the available parameters i.e., number of neighbors *k*, the recommended topic cut-off value *N*, and the topic frequency cut-off *t*.

As we are relying on a collaborative filtering technique, the number of output topics, the number of neighbours, and the data preprocessing play an important role in the assessment. Thus, we variate the recommended list of topics *N* for 5 and 10, and the number of neighbours *k* i.e., $N = \{5, 10, 15, 20, 25\}$. Moreover, we use different topic frequency cut-off *t* to remove very infrequent topics from the dataset. The bar charts in Fig. 6(a) and 6(b) show the average success rates of all ten folds of CFTop, divided by the different topic frequency cut-off *t*. In particular, Fig. 6(a) and Fig. 6(b) shows the success rate considering the first 5 and 10 recommended topics respectively. The horizontal axes shows the success rate outcomes for different size of neighbours *N*. Overall, it is evident that infrequent topics negatively affect both success rate values. At the first glance we can see that the success rate of CFTop with all topics is much lower than others *t* cut-off. The success rate assessment exhibits an average improvement of 10% in all of the possible configurations obtained by variating *N* and *k* values. In particular, the success rate archives better results by setting higher values of *k*. Nevertheless, increasing the number of neighbors gives remarkable benefits only until a certain threshold. Given $k = 5$, the success rate@5 passes from 63% to 69% if we consider k=10. This positive delta decreases by augmenting the number of neighbours until it reaches a stable success rate. Thus, we can consider $k = 25$ as the maximum value capable of improving prediction performances. This trend is further confirmed by introducing more topics in the initial set. We also demonstrate that the topic filtering preprocessing fosters this enhancement and noise removal is a critical step of the entire process.

This is also confirmed by the precision and recall curves depicted in Fig. 7. The line graph depicts the precision and recall curves on average for all 10 rounds by considering *N* value ranges from 1 to 20 and *t*. So, each dot in a curve corresponds to a specific value of *N*. These outcomes have been obtained by keeping 25 as the number of neighbours *k* because we have already discussed that higher values of neighbours reach better prediction performances. Overall, the precision and recall values rise when the *t* cut-off grows. Given that better prediction performance appears near to the upper right corner [11], the figure shows that a higher value of *t* reaches better accuracy for all values of *N*.

In the methodology described in Section 4.2, for each repository *r*, the evaluation outcomes consider the half part of real topics as input and remaining ones as ground truth data *GT(r)*. Because of we are also interested to understand how the number of input topics impacts on prediction performance, Fig. 8 shows the average success rate of all ten folds by choosing different number of input topics. Varying $|t_{in}|$ means changing the length of input topics that enable the CFTop collaborative filtering recommender. In this picture we report the average success of all folds values for the best configuration settings (i.e., $k = 25$, $t = 20$ and) . The success rate values exhibits an improvements when the size of input topic rises. This behaviour demonstrate that CFTop computes better similar repositories as neighbours when it has a higher number of topic as input. This is due to the similarity function that has been involved in the computation of first *k* neighbours. Because the average number of topics for each considered repository is 9.896 we can consider $|t_{in}| = 5$ as the maximum value capable of improving prediction performances.

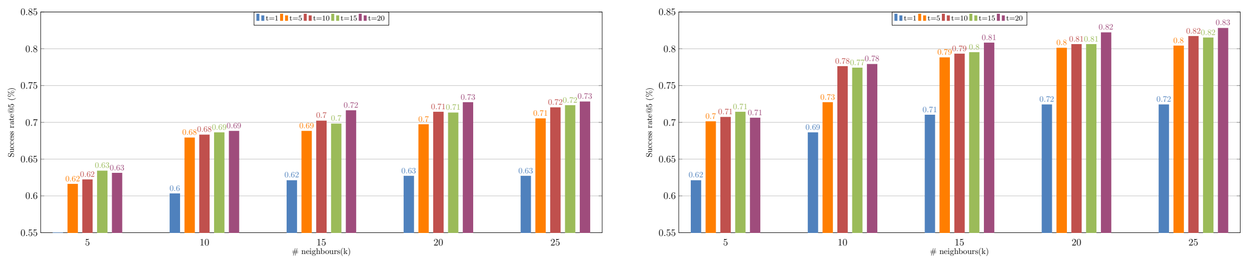
> The quality evaluation demonstrates that CFTop achieves better results by increasing the input data. The number of neighbors and the topic filters contribute to this improvement. However, the precision and recall values are still low, suggesting that bias lives in the users topic.

5.2 MNB network evaluation

$\mathbf{RQ}_2$: *How CFTop behave with respect the MNB network in terms of prediction performance?*

Due to the lack of a baseline, we investigate the prediction performances of the MNB network to compare its outcomes with CFTop. Reversely from the original paper, we apply the MNB network and we compared the outcomes whit respect to all topics (includin non featured ones) leaving the underlying structure untouched. This is necessary to undertake a fair comparison with CFTop. Table 2 shows the evaluation results in terms of the three aforementioned metrics.

We evaluate both approaches by variating the number of recommended topics up to 20. For the sake of the presentation, we report half of the data as we aim to show the overall trend. As we can see, CFTop outperforms the MNB network considering all the metrics. In particular, the success rate grows according to the number of input for both of the approaches. Although the MNB network reaches the same values of CFTop with 20 input topics, the latter starts from an initial success rate value of 55%. This statement holds for all metrics considered in the comparison. A significant achievement is



(a) Success rate@5 (b) Success rate@10

Figure 6: Success rate with 5 and 10 input topics.

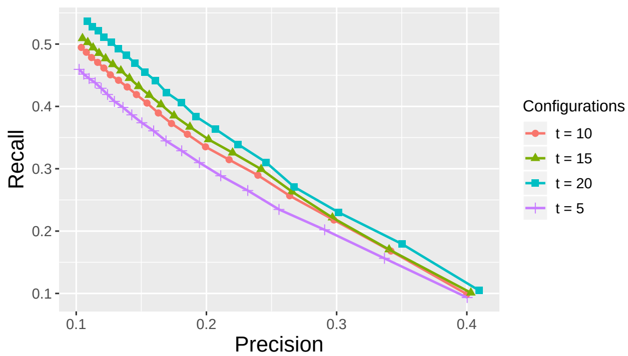


Figure 7: Evaluation of the different configuration.

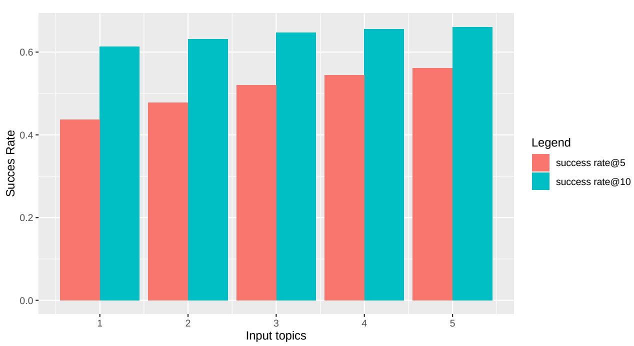


Figure 8: Evaluation of the different input topics.

given by the recall value which is the almost triplicated on average using CFTop as the recommendation engine. For some input, the MNB network slightly outperforms CFTop even though they are meaningless compared to the other findings. This gap is explained by the MNB network model features. In this comparison, we have added the not featured topics to the possible set of outputs [4]. Consequently, the accuracy of the model is compromised by these new possible outcomes that the MNB network is not able to provide. This impacts especially on the recall values, as proved by the experiment.

[4]Due to the space issues, we cannot explain in detail the MNB network internal construction. Thus, the interested reader can find more information in the related work

	Success rate		Precision		Recall	
N	**MNB**	**CFTop**	**MNB**	**CFTop**	**MNB**	**CFTop**
2	0.220	0.554	0.117	0.350	0.031	0.179
4	0.392	0.682	0.119	0.267	0.063	0.271
6	0.538	0.754	0.122	0.224	0.096	0.339
8	0.648	0.803	0.119	0.192	0.125	0.384
10	0.711	0.828	0.112	0.169	0.147	0.422
12	0.765	0.851	0.112	0.153	0.177	0.455
14	0.815	0.863	0.119	0.139	0.220	0.482
16	0.853	0.879	0.112	0.127	0.258	0.503
18	0.874	0.886	0.122	0.117	0.290	0.521
20	0.891	0.892	0.121	0.117	0.320	0.537
Average	**0.651**	**0.785**	**0.120**	**0.194**	**0.165**	**0.397**

Table 2: Comparison of the two approaches.

The aim of this comparison is to prove the soundness of CFTop as a recommendation algorithm where the possible outcomes are heterogeneous i.e., featured topics are shuffled with not featured ones. However, the accuracy is very low compared with the success rate. This could be affected by the similarity function embedded in the recommendation engine.

From the evaluation, we can claim that CFTop outperforms the MNB network. This result is lead by the construction differences between the two approaches, even though the MNB network performances are negatively affected by the introduction of the not featured topics. This demonstrates the rightness of CFTop in a miscellaneous environment.

5.3 Entangled evaluation

RQ$_3$: *Is the entangled approach able to improve CFTop's overall performance?*

As a further experiment, we combined the two approaches to investigate potential improvements. We create this *entagled* configuration by feeding CFTop with the first top-N results of the MNB network. This simulates the exact use case of the collaborative filtering approach, in which the developer is represented by the MNB network. Table 3 summarizes the results of this experiment by comparing CFTop and the entangled approach. As it can be seen there, CFTop gains notable improvement by means of this configuration. For comparison purposes, we enable the entangled approach after the top-5 recommended items given by the MNB network. We witness that all the measured metrics gain a relevant improvement.

In particular, the success rate duplicates its value with only one additional input topic. Moreover, it reaches the maximum value with top-8 topics. CFTop leads to improvement of the MNB network accuracy, by increasing of 10% the precision and recall values on average.

These findings can be explained by considering the nature of the input topics. In the first CFTop evaluation, we randomly elicited a sub-set of topics form the repository to obtain the ground truth topics. Reversely, the entangled approach employs directly the results coming from the MNB network which represent a more curated list of inputs. Thus, selecting proper topics to enable the collaborative filtering algorithm improves the overall performance prediction even though the average accuracy is still low. Additionally, CFTop covers also not featured topics that represent the weakness of the MNB network. In such a way, we combined successfully two different approaches to enlarge the possible set of outcomes.

The entangled approach success in the improvement of the prediction performances. Success rate achieves the best results after the addition of a few topics. Furthermore, the accuracy benefits from this strategy although we measure lower values than expected.

6 RELATED WORK

This section discusses relevant work in this domain.

Immediately after GitHub platform introduces topics, they present Repo-Topix, an automatic approach to suggest them [13]. Such a tool relies on parsing the README files and the textual content of a repository to enable the standard NLP techniques. Then, they filter this initial set of topics by exploiting the TF-IDF scheme and a regression model to exclude "bad" topics. As the final step, Repo-Topix computes a custom version of Jaccard Distance to discover additional similar topics. A rough evaluation based on the n-gram ROUGE-1 metrics has been conducted by counting the number of overlapping units between the recommended topics and the repository description. Nevertheless, a replication package with the complete dataset and the source code is not available for further investigation

In [23], the author proposes a collaborative topic regression (CTR) model to excerpt topics from an initial GitHub repository. The final aim is to recommend other similar projects given the input one. Given a pair of user-repository, the approach uses a Gaussian model to compute matrix factorization and extract the latent vectors given a pre-computed matrix rating. Additionally, a probabilistic topic modeling is applied to find topics from the repositories by analyzing high frequent terms. The approach is evaluated by conducting five-fold cross-validation on a dataset composed of 120,867 repositories. Such evaluation considers the pairs user-repository that have at least 3 watches.

Lia et al. [15] propose a user-oriented portrait model to recommend a set of labels for GitHub projects. An initial set of labels is obtained by computing the LDA algorithm on the textual elements of a repository i.e., issues, commits, and pull requests. Then, the approach exploits a project familiarity technique that relies on the user's behavior considering the different repositories operation. Such a strategy enables the collaborative filtering technique that exploits two kinds of similarity i.e., attribute and social similarity. The former takes into account the personal user information such as the company, the geographical information and the time when the account has been created. The latter computes the similarity scores considering the proportion of items contributed by the user. The approach is evaluated by considering 80 different users with an average of 1894 different behaviors for each one. By considering the first two months of activity in 2016 as a test set, the assessment shows that the approach improves the performances in terms of precision, recall, and success rate

A model-based fuzzy C-means for collaborative filtering (MFCCF) has been proposed in [4] with the aim of recommending relevant human resources during the GitHub project development. Similarly to our approach, the proposed model encodes relevant information about repositories in a graph structure and excerpt from it the sparsetest sub-graph. This phase is preparatory to enable the fuzzy C-means clustering technique. Using the computed sparse sub-graph as the center of the cluster, the model can handle the sparsity issue that normally arises in the CF domain. Then, MFCCF computes the Pearson Correlation for each pair user-item belonging to a cluster and retrieves the top-N results. The evaluation is performed using the GHTorrent dump to collect the necessary information. Using ten projects as the testing dataset, the results of the MFCCF are compared with the ones chosen by HR company managers. The results demonstrate the effectiveness of the approach with an accuracy of 80% on average.

7 CONCLUSIONS AND FUTURE WORK

conclusion

REFERENCES

[1] 2019. PyGithub/PyGithub. https://github.com/PyGithub/PyGithub original-date: 2012-02-25T12:53:47Z.

[2] 2019. Understanding the search syntax - GitHub Help. https://help.github.com/en/github/searching-for-information-on-github/understanding-the-search-syntax

[3] Charu Aggarwal. 2016. *Neighborhood-Based Collaborative Filtering.* Springer International Publishing, Cham, 29–70. https://doi.org/10.1007/978-3-319-29659-3_2

[4] Shohreh Ajoudanian and Maryam Nooraei Abadeh. 2019. Recommending human resources to project leaders using a collaborative filtering-based recommender system: Case study of gitHub. *IET Software* 13, 5 (2019), 379–385. https://doi.org/10.1049/iet-sen.2018.5261 Conference Name: IET Software.

[5] Hudson Borges, Andre Hora, and Marco Tulio Valente. 2016. Predicting the Popularity of GitHub Repositories. *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering - PROMISE 2016* (2016), 1–10. https://doi.org/10.1145/2972958.2972966 arXiv: 1607.04342.

[6] Hudson Borges and Marco Tulio Valente. 2018. What's in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform. *Journal of Systems and Software* 146 (Dec. 2018), 112–129. https://doi.org/10.1016/j.jss.2018.09.016 arXiv: 1811.07643.

[7] Hudson Borges, Marco Tulio Valente, Andre Hora, and Jailton Coelho. 2017. On the Popularity of GitHub Applications: A Preliminary Note. *arXiv:1507.00604 [cs]* (March 2017). http://arxiv.org/abs/1507.00604 arXiv: 1507.00604.

[8] Cristian E. Briguez, Maximiliano C.D. Budán, Cristhian A.D. Deagustini, Ana G. Maguitman, Marcela Capobianco, and Guillermo R. Simari. 2014. Argument-based mixed recommenders and their application to movie suggestion. *Expert Systems with Applications* 41, 14 (2014), 6467 – 6482. https://doi.org/10.1016/j.eswa.2014.03.046

[9] Paolo Cremonesi, Roberto Turrin, Eugenio Lentini, and Matteo Matteucci. 2008. An Evaluation Methodology for Collaborative Recommender Systems. In *Proceedings of the 2008 International Conference on Automated Solutions for Cross Media Content and Multi-channel Distribution (AXMEDIS '08).* IEEE Computer Society, Washington, DC, USA, 224–231. https://doi.org/10.1109/AXMEDIS.2008.13

	RECALL			PRECISION			SUCCESSRATE			CATALOG COVERAGE		
k	MNB	Tin=5	Tin=2	MNB	Tin=5	Tin=2	MNB	Tin=5	Tin=2	MNB	Tin=5	Tin=2
1	0,018	0,018	0,018	0,138	0,138	0,138	0,136	0,136	0,136	10.769	4.835	4.835
2	0,031	0,031	0,031	0,118	0,118	0,118	0,217	0,217	0,217	15.604	8.593	8.593
3	0,047	0,047	0,060	0,118	0,118	0,151	0,301	0,301	0,367	19.780	12.483	12.571
4	0,063	0,063	0,088	0,119	0,119	0,166	0,389	0,389	0,466	23.956	15.340	15.912
5	0,081	0,081	0,104	0,124	0,124	0,157	0,476	0,476	0,508	25.494	18.307	19.054
6	0,096	0,121	0,119	0,122	0,153	0,149	0,538	0,601	0,549	27.032	22.131	21.780
7	0,112	0,149	0,131	0,121	0,161	0,140	0,605	0,668	0,574	28.791	25.912	24.681
8	0,125	0,171	0,142	0,119	0,162	0,133	0,648	0,704	0,599	29.010	29.296	27.428
9	0,135	0,189	0,153	0,115	0,160	0,128	0,678	0,734	0,623	29.230	32.417	30.307
10	0,147	0,204	0,163	0,112	0,156	0,123	0,711	0,754	0,644	29.230	35.296	32.967
12	0,177	0,230	0,181	0,112	0,146	0,114	0,762	0,788	0,681	29.230	40.659	38.373
14	0,220	0,254	0,201	0,119	0,138	0,109	0,815	0,808	0,706	29.230	45.912	43.098
16	0,258	0,274	0,215	0,122	0,131	0,102	0,853	0,829	0,722	29.230	50.505	47.582
18	0,290	0,290	0,227	0,122	0,123	0,096	0,874	0,840	0,736	29.450	54.615	51.318
20	0,320	0,306	0,241	0,121	0,117	0,092	0,891	0,855	0,756	29.450	58.725	54.923

Table 3: Results for the entangled approach.

[10] Jesse Davis and Mark Goadrich. 2006. The Relationship Between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning* (Pittsburgh, Pennsylvania, USA) *(ICML '06)*. ACM, New York, NY, USA, 233–240. https://doi.org/10.1145/1143844.1143874

[11] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, Davide Romito, and Markus Zanker. 2012. Linked Open Data to Support Content-based Recommender Systems. In *Proceedings of the 8th International Conference on Semantic Systems* (Graz, Austria) *(I-SEMANTICS '12)*. ACM, New York, NY, USA, 1–8. https://doi.org/10.1145/2362499.2362501

[12] Ganesan. 2019. Topic Suggestions for Millions of Repositories - The GitHub Blog. https://github.blog/2017-07-31-topics/

[13] Kavita Ganesan. 2017. Topic Suggestions for Millions of Repositories - The GitHub Blog. https://github.blog/2017-07-31-topics/

[14] Ron Kohavi et al. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, Vol. 14. Montreal, Canada, 1137–1145.

[15] Zhifang Liao, Tianhui Song, Yan Wang, Xiaoping Fan, and Yan Zhang. 2018. User personalized label set extraction algorithm based on LDA and collaborative filtering in open source software community. In *2018 International Conference on Computer, Information and Telecommunication Systems (CITS)*. 1–5. https://doi.org/10.1109/CITS.2018.8440167

[16] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.Com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing* 7, 1 (Jan. 2003), 76–80. https://doi.org/10.1109/MIC.2003.1167344

[17] Collin McMillan, Mark Grechanik, and Denys Poshyvanyk. 2012. Detecting Similar Software Applications. In *Proceedings of the 34th International Conference on Software Engineering* (Zurich, Switzerland) *(ICSE '12)*. IEEE Press, Piscataway, NJ, USA, 364–374. http://dl.acm.org/citation.cfm?id=2337223.2337267

[18] Catarina Miranda and Alípio M. Jorge. 2008. Incremental Collaborative Filtering for Binary Ratings. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01 (WI-IAT '08)*. IEEE Computer Society, Washington, DC, USA, 389–392. https://doi.org/10.1109/WIIAT.2008.263

[19] Phuong T. Nguyen, Juri Di Rocco, Davide Di Ruscio, Lina Ochoa, Thomas Degueule, and Massimiliano Di Penta. 2019. FOCUS: A Recommender System for Mining API Function Calls and Usage Patterns. In *Proceedings of the 41st International Conference on Software Engineering* (Montreal, Quebec, Canada) *(ICSE '19)*. IEEE Press, Piscataway, NJ, USA, 1050–1060. https://doi.org/10.1109/ICSE.2019.00109

[20] P. T. Nguyen, J. Di Rocco, R. Rubei, and D. Di Ruscio. 2018. CrossSim: Exploiting Mutual Relationships to Detect Similar OSS Projects. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 388–395. https://doi.org/10.1109/SEAA.2018.00069

[21] Phuong T. Nguyen, Paolo Tomeo, Tommaso Di Noia, and Eugenio Di Sciascio. 2015. An Evaluation of SimRank and Personalized PageRank to Build a Recommender System for the Web of Data. In *Proceedings of the 24th International Conference on World Wide Web* (Florence, Italy) *(WWW '15 Companion)*. ACM, New York, NY, USA, 1477–1482. https://doi.org/10.1145/2740908.2742141

[22] Tommaso Di Noia and Vito Claudio Ostuni. 2015. Recommender Systems and Linked Open Data. In *Reasoning Web. Web Logic Rules - 11th International Summer School 2015, Berlin, Germany, July 31 - August 4, 2015, Tutorial Lectures*. 88–113. https://doi.org/10.1007/978-3-319-21768-0_4

[23] Naoki Orii. 2013. Modeling for Recommending GitHub Repositories.

[24] Michael J. Pazzani and Daniel Billsus. 2007. *Content-Based Recommendation Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 325–341. https://doi.org/10.1007/978-3-540-72079-9_10

[25] Martin Robillard, Robert Walker, and Thomas Zimmermann. 2010. Recommendation Systems for Software Engineering. *IEEE Softw.* 27, 4 (July 2010), 80–86. https://doi.org/10.1109/MS.2009.161

[26] Tefko Saracevic. 1995. Evaluation of Evaluation in Information Retrieval. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Seattle, Washington, USA) *(SIGIR '95)*. ACM, New York, NY, USA, 138–146. https://doi.org/10.1145/215206.215351

[27] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (Hong Kong, Hong Kong) *(WWW '01)*. ACM, New York, NY, USA, 285–295. https://doi.org/10.1145/371920.372071

[28] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. The Adaptive Web. Springer-Verlag, Berlin, Heidelberg, Chapter Collaborative Filtering Recommender Systems, 291–324. http://dl.acm.org/citation.cfm?id=1768197.1768208

[29] Claudio Di Sipio, Riccardo Rubei, Davide Di Ruscio, and Phuong T. Nguyen. 2020. A Multinomial Naïve Bayesian (MNB) network to automatically recommend topics for GitHub repositories - Online appendix. https://github.com/MDEGroup/MNB_TopicRecommendation/

[30] Ferdian Thung, David Lo, and Julia Lawall. 2013. Automated library recommendation. In *2013 20th Working Conference on Reverse Engineering (WCRE)*. 182–191. https://doi.org/10.1109/WCRE.2013.6671293

[31] Zhi-Dan Zhao and Ming-sheng Shang. 2010. User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop. In *Proceedings of the 2010 Third International Conference on Knowledge Discovery and Data Mining (WKDD '10)*. IEEE Computer Society, Washington, DC, USA, 478–481. https://doi.org/10.1109/WKDD.2010.54