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, 8 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 projects' features. Their availability triggers different applications and studies to improve the literature, including the automated cataloging of GitHub repositories [?]. 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 [].

In this work, we propose to extend the set of recommended items to non-featured topics by exploiting collaborative filtering, a widely spread technique in the recommendation system domain [23]. 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., sales diversity, novelty, and accuracy;
- Considering a well-founded approach, we extend 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 [?].

In [?] the authors present an approach based on *topic modeling* techniques to create categories of GitHub projects. Manual interventions are needed to refine initial sets of categories, which are identified by an LDA-GA technique, that combines two algorithms: Latent Dirichlet Allocation (LDA) and Genetic Algorithm (GA) [?]. The approach proposed in [?] is unsupervised, meaning that the categories of the catalogue being identified are not known ex-ante.

In a GitHub blog post [?] the author presents *repo-topix*, a tool to recommend topics for GitHub repositories. Such a tool combines NLP standard techniques to find an initial set of topics, by parsing the README files and the textual content of a repository e.g., the

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

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

repository's description. Then, they weight the results with the TF-IDF scheme and remove "bad" topics using a regression model. Using this refined list, repo-topix computes a custom version of Jaccard Distance to identify additional similar topics. To assess the quality of the framework, they made a rough evaluation based on ROUGE-1 metrics, an n-gram overlap metric that counts the number of overlapping units between the suggested topics and the repository description. Unfortunately, in [?] the author discusses an approximation of the repo-topix accuracy, without providing the reader with the complete dataset that was used and the source code of the developed tool.

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 changes and lead to lack of data, and poor topic coverage. GHTorrent[2] 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 [23] 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 [12]. 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"* [23]. 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."*

[2]http://ghtorrent.org/

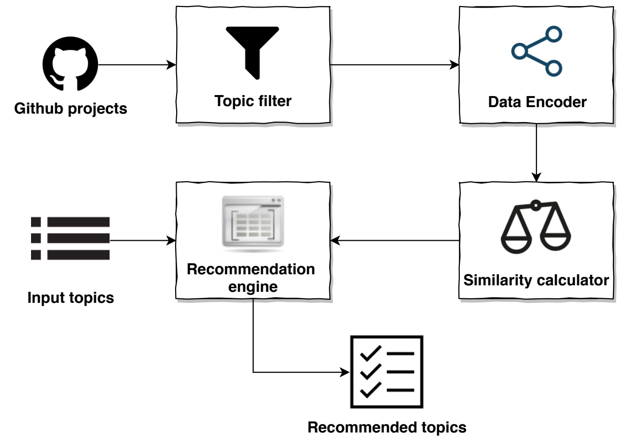

Figure 1: Overview of the CFTop Architecture.

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],[26], 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 [11], 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* [22],[18]. 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 [18]. Moving to our domain, users are substitute by projects as well as topics are the possible items to recommend. The analogus 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 shown in Fig. **??** into a directed graph as in Fig. 2. We adopted our proposed CrossSim approach [15],[16] to compute the

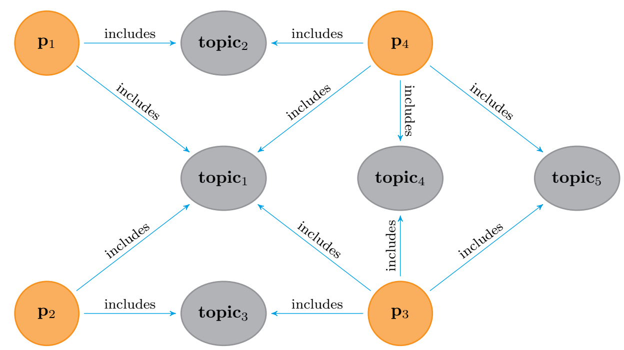

Figure 2: Graph representation for projects and libraries.

similarities among OSS graph nodes. It relies on techniques successfully exploited by many studies to do the same task [10],[7]. 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_3 and p_4 are highly similar since they both point to three nodes $topic_1, topic_4, topic_5$. This reflects what also suggested in a previous work by McMillan et al. [13], i.e., similar projects implement common pieces of functionality by using a shared set of libraries.

Using this metric, the similarity between two project nodes p and q in an OSS graph is computed by considering their feature sets [10]. 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:

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

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 [10]. 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.

The representation using a user-item ratings matrix allows for the computation of missing scores [3],[18]. Depending on the availability of data, there are two main techniques to compute the unknown ratings, namely *content-based* [19] and *collaborative-filtering* [14] 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

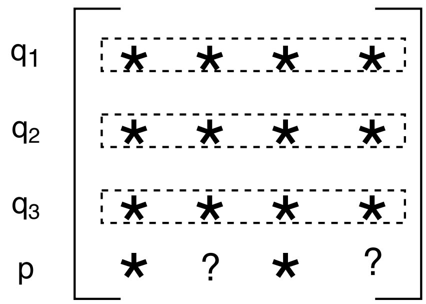

Figure 3: Computation of missing ratings using the user-based collaborative-filtering technique [26].

collaborative-filtering recommendation: *user-based* [26] and *item-based* [22] 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 [8].

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.

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 [12, 26]. 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., $p \ni l$ [18]:

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

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.

4 EVALUATION

This section describes the planning of our evaluation, having the *goal* of evaluating the performance of the proposed approach. In Section 4.1, we introduce the dataset exploited in our evaluation. The evaluation methodology and metrics are presented in Section 4.2. Finally, Section 4.3 describes the research questions.

The evaluation process is depicted in Fig. 4 and it consists of three consecutive phases, i.e., *Data Preparation*, *Recommendation*, and *Outcome Evaluation*. We start with the *Data Preparation* phase by creating a dataset from GitHub projects. 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. To enable CFTop, we extract a portion of topics from a given testing project i.e., the ground-truth part. The left part is used as a query to produce recommendations. 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. Finally, the entangled approach uses CFTop as the recommendation engine which is fed by the MNB network suggested topics. 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.

4.1 Dataset Extraction

To evaluate the approach, we reuse the same dataset employed for the MNB network available here [24]. 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]. To overcome the request limit during the crawling activity, we employ the GitHub star voting mechanism as a popularity measure [5]. As claimed in several works[4, 6], a high number of stars means the attention of the community for that project. So, we impose the following filter during the query execution:

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

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. As CFTop is able to retrieve both featured and not-featured topics, this filter doesn't affect the quality of the collected data.

4.2 Metrics' Definition

To assess the performance of CFTop, we applied ten-fold cross-validation, considering every time 9 folds (each one contains 625 projects) for training and the remaining one for testing. For every testing project p, half of its topics are *randomly* taken out and saved as ground truth data, let us call them *GT(p)*, which will be used to validate the recommendation outcomes. The other half is used as testing topics or query, which are called ***te***, and serve as input for `Similarity Computation` and `Recommendation` components. The splitting simulates a real development process where a developer has already included some topics in his repository, i.e., ***te*** and waits for recommendations i.e., additional topics to be incorporated. A

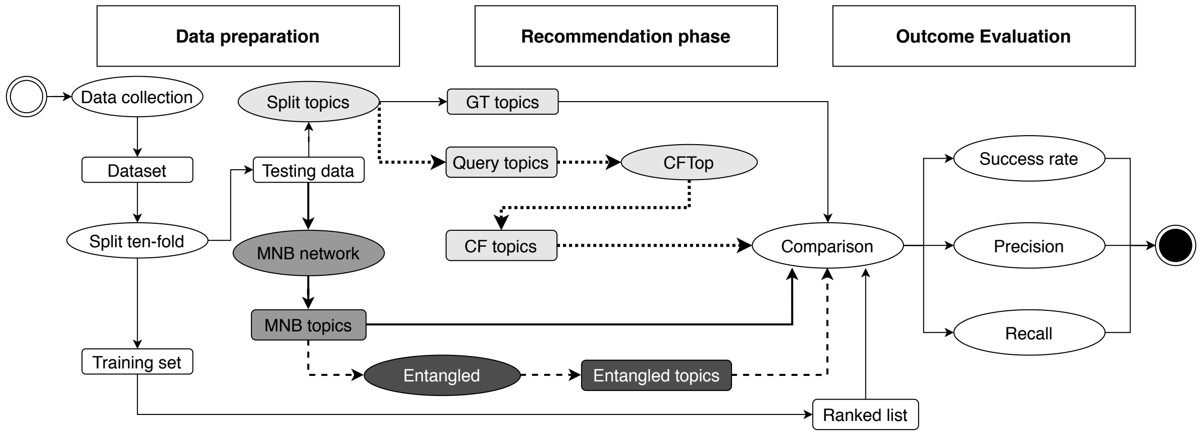

Figure 4: Evaluation Process.

recommender system is expected to provide her with the other half, i.e., *GT(p)*.

There are several metrics available to evaluate a ranked list of recommended items [18]. 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. [20] and other studies [25],[17]. The metrics considered during the outcome evaluation follows this notation:

- N is the cut-off value for the ranked topic list;
- k is the number of neighbor projects exploited for the recommendation process;
- For a testing project p, a half of its topics are extracted and used as the ground-truth data named as *GT(p)*;
- $REC(p)$ is the *top-N* topics recommended to p. It is a ranked list in descending order of real scores;
- If a recommended topic $t \in REC(p)$ for a testing project p is found in the ground truth of p (i.e., *GT(p)*), 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 $p \in P$ [25]:

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

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 [21]. 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. (5)). 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 [15],[10],[9]:

$$precision@N = \frac{|match_N(p)|}{N} \tag{6}$$

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

4.3 Research Questions

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

- **RQ$_1$**: *How CFTop parameter settings impact on the prediction performance?* To answer this question, we investigate different configurations to find the best one. In particular, we variate the number of input topics and the number of neighbors
- **RQ$_2$**: *How CFTop behave with respect the MNB network in terms of prediction performance?* As the two approaches have a completely different internal structure, we are interested in investigating the reasons behind such quality variation
- **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.

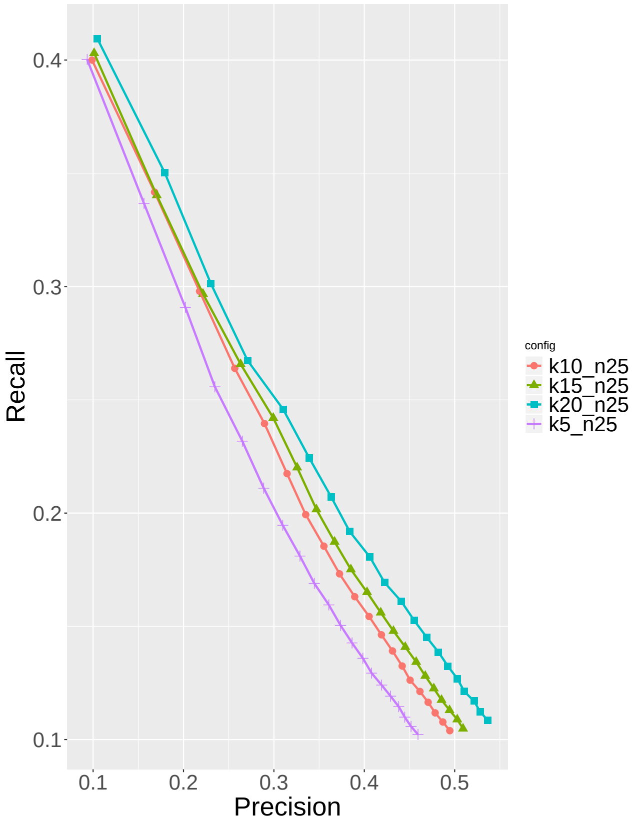

Figure 5: Evaluation of the different configuration.

5.1 CFTop evaluation

RQ$_1$: *How CFTop parameter settings impact on the prediction performance?* 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 and cut-off value. The former refers to the number of graph nodes used in the recommendation engine. The latter is used to select the input topics based on their frequencies. Given an initial set of topics, we filter them with the cut-off value to reduce the noise in the original dataset. Then, the recommendation phase is enabled by variating the number of parameters. According to Section 4.2, N is the cut-off value and k is the number of neighbors of the graph. We evaluate different configuration by setting N=1,5,10,15,20 and k=5,10,15,20,25. Figure 5 shows the results in terms of precision and recall.

As we are relying on a collaborative filtering technique, the number of input topics plays an important role in the assessment. Thus, we variate this additional parameter and compute the success rate for 5 and 10 input topics. Figure 6 and 7 show the outcome of this comparison.

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 passes from 63% to 69% if we consider k=10. This positive delta decreases by augmenting the number of neighbors 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.

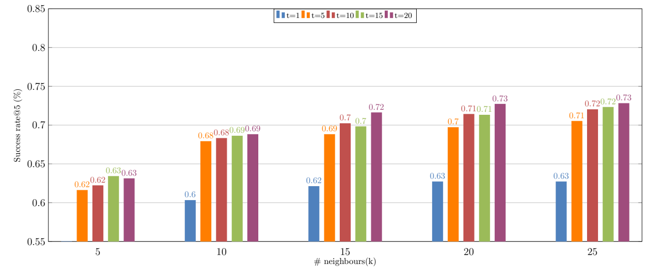

Figure 6: Success rate with 5 input topics.

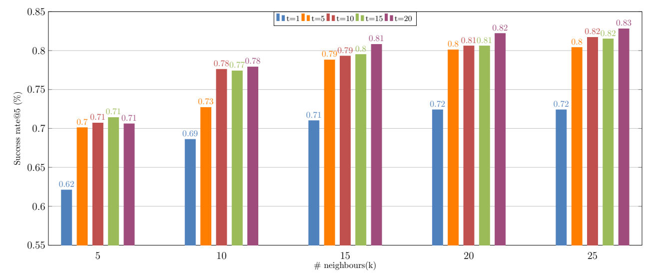

Figure 7: Success rate with 10 input topics.

> 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 values are still low, suggesting that bias lives in the users topic

5.2 MNB network evaluation

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 to not featured topics leaving the underlying structure untouched. This is necessary to undertake a fair comparison with CFTop. Table 1 shows the evaluation results in terms of the three aforementioned metrics.

We evaluate both approaches by variating the number of input 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

	MNB network			CFTop		
No. of input	**Success rate**	**Precision**	**Recall**	**Success rate**	**Precision**	**Recall**
2	0.220	0.117	0.031	0.554	0.350	0.179
4	0.392	0.119	0.063	0.682	0.267	0.271
6	0.538	0.122	0.096	0.754	0.224	0.339
8	0.648	0.119	0.125	0.803	0.192	0.384
10	0.711	0.112	0.147	0.828	0.169	0.422
12	0.765	0.112	0.177	0.851	0.153	0.455
14	0.815	0.119	0.220	0.863	0.139	0.482
16	0.853	0.112	0.258	0.879	0.127	0.503
18	0.874	0.122	0.290	0.886	0.117	0.521
20	0.891	0.121	0.320	0.892	0.117	0.537
Average values	**0.651**	**0.120**	**0.165**	**0.785**	**0.194**	**0.397**

Table 1: Comparison of the two approaches.

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 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 [3]. 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. 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 2 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.

	CFTop			Entangled approach		
No. of input	**Success rate**	**Precision**	**Recall**	**Success rate**	**Precision**	**Recall**
1	0.409	0.409	0.105	0.138	0.221	0.029
2	0.554	0.350	0.179	0.220	0.198	0.053
3	0.632	0.301	0.230	0.304	0.192	0.077
4	0.682	0.267	0.271	0.393	0.186	0.099
5	0.728	0.246	0.310	0.479	0.183	0.122
6	0.754	0.224	0.339	0.983	0.278	0.225
7	0.778	0.207	0.363	0.999	0.340	0.322
8	0.803	0.192	0.384	1	0.371	0.40
10	0.828	0.169	0.422	1	0.382	0.511
15	0.872	0.132	0.493	1	0.322	0.636
20	0.892	0.117	0.537	1	0.266	0.696
Average values	**0.785**	**0.194**	**0.397**	**0.826**	**0.296**	**0.433**

Table 2: Results for the entangled approach.

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 both *(i)* approaches based on collaborative filtering techniques in recommending activity and *(ii)* works that mine GitHub projects.

6.1 Recommends item by means of collaborative filtering

Amazon [12] proposes an item-to-item recommendation system to suggest relevant products to the final user.

6.2 Recommending OSS using GitHub topics

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] Due to the space issues, we cannot explain in detail the MNB network internal construction. Thus, the interested reader can find more information about this in our previous work

[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] 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.

[5] 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.

[6] 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.

[7] 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

[8] 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

[9] 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

[10] 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

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

[12] 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

[13] 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

[14] 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

[15] 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

[16] 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

[17] 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

[18] 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

[19] 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

[20] 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

[21] 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

[22] 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

[23] 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

[24] 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/

[25] 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

[26] 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