CFTop: Using collaborative filtering to recommend Github topics

ABSTRACT

Collaborative filtering is a well-founded technique widely used in the recommendation system domain. During recent years, a plethora of approaches have been developed to provide the users with relevant items. Considering the open-source software (OSS) domain, GitHub has gained the head role in storing, analyzing and maintaining a huge number of repositories. To represent the stored projects in an effective manner, in 2017 GitHub introduced the possibility to classify them employing topics. However, such labeling activity should be carefully conducted to avoid negative effects on project popularity. In this paper, we present CFTop, a recommender system to assist open source software developers in selecting suitable topics for the repositories. CFTop exploits a collaborative filtering technique to recommend topics to developers by relying on the set of initial ones, 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, combining the two approaches improves the overall 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, 12 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 [9]. The platform recently introduced *topics*[1] to foster the popularity and promote information discovery about available 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[2] already faced this problem by using a machine learning approach to recommend relevant topics given a README file of a repository [13]. However, this too is able to recommend only *featured* topics, a curated list of them provided by Github[3].

We propose CFTop, a recommendation system that extends the set of recommended items to non-featured topics by exploiting collaborative filtering, a broadly used technique in the recommendation system domain [30]. Given an initial set of topics coming from a GitHub project, we encode the relevant information in a graph-based structure. In such a way, we are able to represent the mutual relationships between repositories and topics. From this, a project-topic matrix is created following the common user-item structure used in existing collaborative filtering applications. Then, we compute a similarity function based on featured vectors to recommend the most similar topics.

We evaluate the CFTop's prediction performances by variating different parameters as well as comparing it with the MNB network approach. As the direct comparison is not possible due to the approaches' internal construction, we used a well-defined set of metrics used in the literature to evaluate both of the approaches considering different datasets. Furthermore, we combined the two approaches to investigate the potential benefits of this union.

The work gives the following contributions:

- Considering GitHub topics as product to recommend, we improve repositories' popularity by suggesting a 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, accuracy, and catalog coverage;
- 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 presents the MNB network approach with a motivating example to describe the faced problem. In Section 3, we present our approach and evaluate it in Section 4. Section 5 discusses relevant findings and related works are summarized in Section 7. Finally, we conclude the paper and discuss possible future works in Section 8 with possible future works.

2 MOTIVATION AND BACKGROUND

Extracting knowledge from a developed system can provide potential benefits for searching, browsing, and discovering them among a huge set of software systems. GitHub is one of the most used developing service that includes version control systems (i.e., git)

[1] https://help.github.com/en/github/administering-a-repository/classifying-your-repository-with-topics

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

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

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

plus social and collaborative features (e.g., bug tracking, contribution requests, task management, and wikis). At the moment we are writing this contribution, GitHub counts more than 40 million users and over 100 million repositories. Because of this huge among of data, the availability of reusable projects might be compromised if they cannot be suitably discovered. In recent years, GitHub introduced a topic mechanism to explore repositories in a particular subject area, learn more about that subject, and find projects to contribute to. GitHub continuously monitors the stored repository and assigned topics to organize the former with respect to the list of assigned topics. Moreover, those data are periodically analysed to extract the most popular and active topics (i.e., *featured topics*[4]). Thus, users can monitor the community's trend by consulting such a public list. In the beginning, this activity was entirely done by humans (i.e., project contributors) that label the repository according to their knowledge, feeling and belief. Literature is plenty of several approaches that mine and exploit available data to analyze repositories. Nevertheless, few of them cope with the topic recommendation task, which can be crucial in the project's development initial phase.

Figure 1 shows an example repository with related topics. By this simple snapshot, a GitHub user can figure out that the *apache spark*[5] project makes usage of several programming languages such as *java, r,* and *python* to analyze *big-data* and databases by exploiting common techniques used in this domain i.e., *sql and jdbc.*

As mentioned before, the MNB network 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. By relying on the multinomial probability distribution, the approach is able to extract relevant information from the README file and suggest a set of topics. Table 1 shows an example of the MNB network's outcomes given the list of the actual repository topics.

Actual Topics	Predicted topics
python,blender-scripts, spaceship, procedural-generation, game-development, 3d	shell, terminal, **3d**, opengl, **python**

Table 1: Example of the MNB network outcomes. Juri ►*change the table with apache/spark data*◄

Even though the MNB network works in practice, it suffers from some limitations. First, the underlying model can recommend only featured topics that represent only a small set of all possible terms that can be restricted because of antonymous term(e.g., programming languages). In this way, the MNB network doesn't express all the concepts covered by a GitHub repository. As shown in the table, only two of the predicted topics matched with the real ones. The second major limitation is the underlying structure needed for the training phase. To deliver relevant items, the MNB network requires a *balanced* dataset, i.e., each topic must have a similar number of README files. This scenario is difficult to meet in reality as the topics' heterogeneity is extremely high. Furthermore, the GitHub platform is regularly updated with new projects and, consequently, with new topics. Thus, the training phase must take place several times to avoid outdated recommendations.

[4]https://github.com/topics
[5]https://github.com/apache/sparck

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 [30] 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 [19].

Similarly to the work presented in [23], our proposed technique follows the assumption that *"if users agree about the quality or relevance of some items, then they will likely agree about other items"* [30]. 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".*

To this end, the architecture of CFTop is shown in Fig. 2, 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. 2, 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, 23, 34], 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.

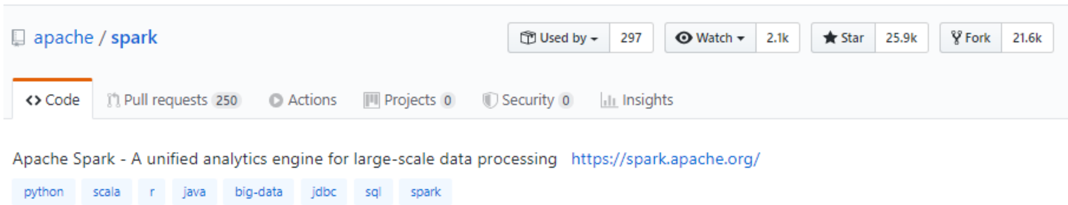


Figure 1: Example of a GitHub repository and its topics.

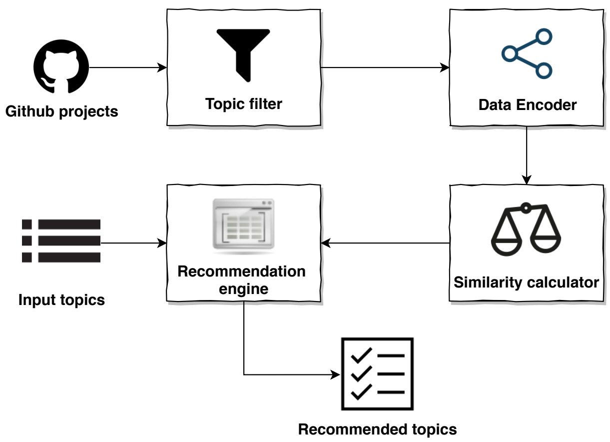


Figure 2: Overview of the CFTop Architecture.

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 [14], this preprocessing can improve the final results, thus we decide to apply it as a first step.

To this end, we develop tailored Python scripts that apply this filter to the initial dataset. As a GitHub user can manually specify the topic list for his repository, a lot of them can contain infrequent or improper terms i.e., the name of the author, duplicated values, terms that rarely appear to name a few. On one hand, imposing such preprocessing reduces the repositories to analyze as well as topics to recommend. On the other hand, we improve the overall quality of recommendation by pruning "bad" terms. This pruning phase is computed offline and doesn't affect the time required for the recommendation process.

3.2 Data Encoder

Considering traditional recommender systems for online services, we can identify three main components, namely *users*, *items*, and *ratings* [29],[24]. 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 [24]. Moving to our domain, users are substitute by projects as well as topics are the possible items to recommend. The resulting *repo-topic ratings matrix* represents possible relationships between these two elements i.e., project may include various topics. Before encoding topics in the user-item ratings matrix, raw topics are pre-processed removing possible syntactical duplicates terms (e.g., *document* and *documents*). More specifically, stemming, lemmatization, and stop words removal Natural Language Processing (NLP) techniques have been applied on the mined topics.

We can denote *project-topic 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 $L = \{t_1 = machine\text{-}learning;\ t_2 = javascript;\ t_3 = database;\ t_4 = web;\ t_5 = algorithm,\ t_6 = algorithms\}$ where $p_1 \ni t_1, t_2, t_6$; $p_2 \ni t_1, t_3$; $p_3 \ni t_1, t_3, t_4, t_5$; $p_4 \ni t_1, t_2, t_4, t_5$. After the NLP normalization steps, the topics t_5 and t_6 collapse on the same terms that will named as t_{5_6}. Then, the Table 2 are generated to represent inclusions between repositories and topics.

3.3 Similarity Calculator

Juri ▶ *The* `Recommendation Engine` *of CFTop works by relying on the mentioned repo-topic 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*

	t_1	t_2	t_3	t_4	t_{5_6}
p_1	1	1	0	0	1
p_2	1	0	1	0	0
p_3	1	0	1	1	1
p_4	1	1	0	0	1

Table 2: The *repo-topic ratings matrix* explanatory example

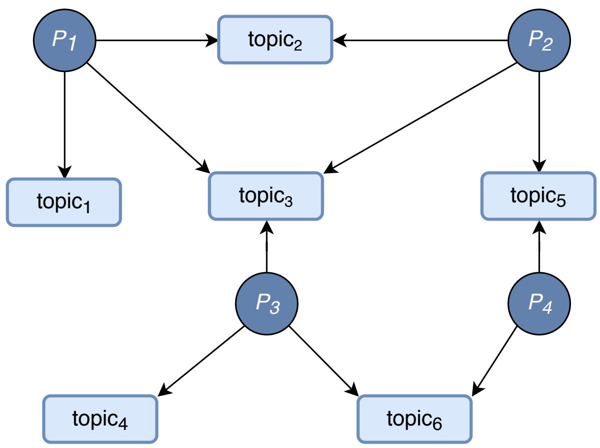


Figure 3: Graph representation for projects and topics

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. ◀

Juri ▶ *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. 3. We adopted the approach in [21],[22] to compute the similarities among OSS graph nodes. It relies on techniques successfully exploited by many studies to do the same task [12],[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. 3, 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 [12]. 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:

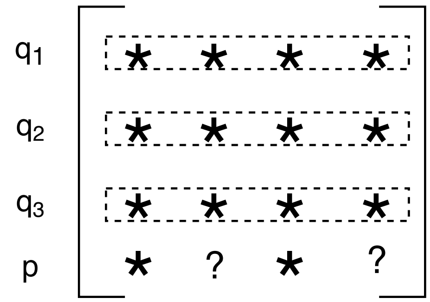


Figure 4: Computation of missing ratings using the user-based collaborative-filtering technique [34].

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

where n is the cardinality of the set of topics that p and q share in common [12]. 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

Juri ▶ *Rephrase this section*◀ The representation using a user-item ratings matrix allows for the computation of missing scores [3],[24]. Depending on the availability of data, there are two main techniques to compute the unknown ratings, namely *content-based* [26] and *collaborative-filtering* [20] 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* [34] and *item-based* [29] 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 [10].

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. 4, 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.

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 [19, 34].

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. 4 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., $p \ni l$ [24]:

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

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, and Section 4.3 respectively. Finally, Section 4.4 describes the research questions.

4.1 Dataset Extraction

To evaluate the approach, we reuse the same dataset employed for the MNB network available here [31]. 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 starting from the original one 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 3 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	15,743	9.9	3.9
Dt_5	3,884	1,989	8.4	16.5
Dt_{10}	2,897	964	8.0	24.1
Dt_{15}	2,273	634	7.8	28.1
Dt_{20}	1,806	456	7 .7	30.5

Table 3: 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. 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 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 6 shows the comparison among all the examined datasets. As can see, filtering repositories by the t value helps to smooth the distribution. On one hand, the Dataset Dt_5 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=5 and t=20 respectively. On the other hand, the slope depicted in Dataset Dt_{20} is higher than the original dataset. The positive trend is confirmed by observing the distribution of the

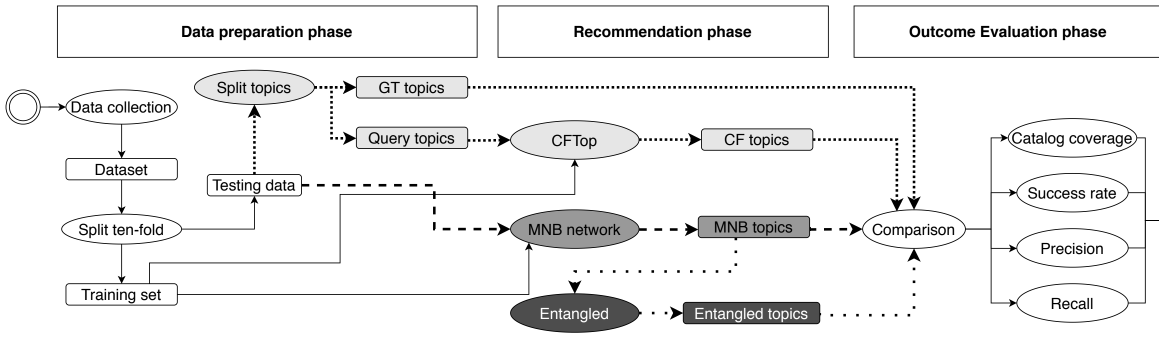
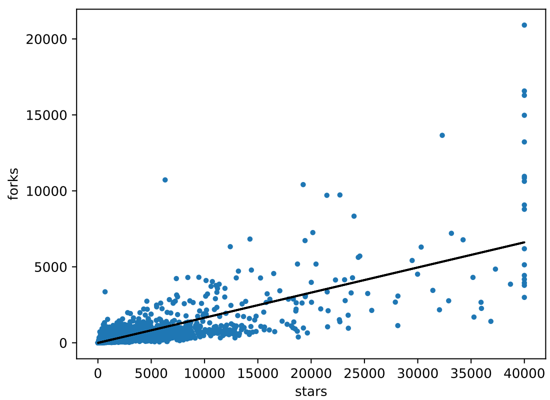
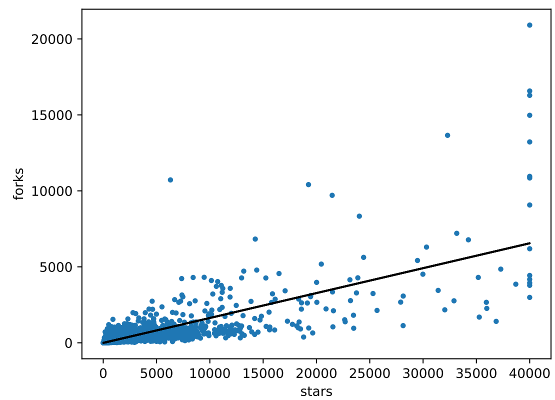


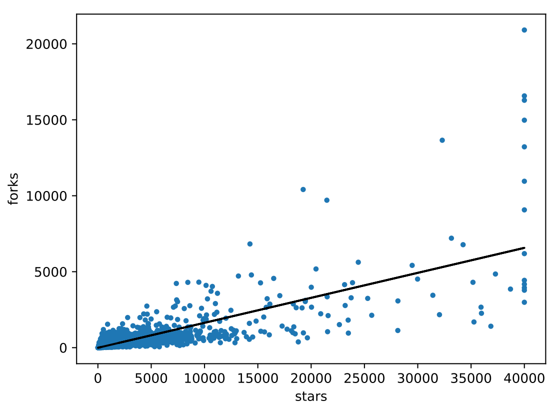
Figure 5: Evaluation Process.



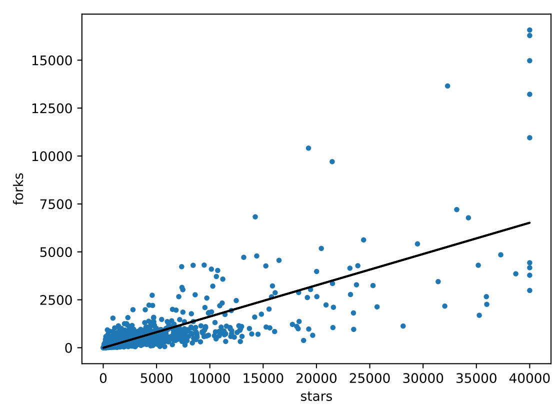
(a) Dataset Dt_5



(b) Dataset Dt_{10}



(c) Dataset Dt_{15}



(d) Dataset Dt_{20}

Figure 6: Quality analisis of the examined datasets.

other datasets i.e., Dataset Dt_{10} and Dt_{15}. In particular, we are able to remove repositories with less number of stars i.e., from 5,000 to 10,000. From this study, we observe a correlation between the number of stars and the topics' frequency. In other words, most frequent topics appear in the high-ranked repositories and this finding affects the quality of the recommendations.

4.2 Evaluation process

The *ten-fold cross-validation* methodology [17] 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. Figure 5 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 during `Data collection` step. This dataset is used to evaluate CFTop, MNB network, and the combination of two. The dataset is then split into training and testing sets (i.e., `Split ten-fold` activity). Due to the different nature of the the recommender systems (i.e., MNB network requires `README` files as input and training data, whereas CFTop uses a set of assigned topics as input and for training the recommendation system), the testing and training data are specifically cooked for both approaches). The `Split topic` activity resembles a real development process where a developer has already included some topics in his repository (i.e.,`Query topics`) 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 topics`.

■ **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). In this respect, both `Testing data` and `Training set` boxes are simplified to provides the needed data (i.e., README file and assigned topics) to the different recommender systems.

■ **Outcome Evaluation phase** evaluate the recommendation results with those stored as ground-truth data to compute the quality metrics (i.e.,`Success rate`, `Precision`, `Recall`, and `Catalog coverage`) during the `Comparison` activity.

It is worth noting that we can't directly compare directly CFTop and MNB network approaches because they rely on different input data (i.e., CFTop requires an initial set of assigned topics for suggesting new ones, whereasMNB network uses the information mined from the README files to recommend the expected topics).

4.3 Metrics' definition

There are several metrics available to evaluate a ranked list of recommended items [24]. In the scope of this paper, *success rate accuracy*, and *catalog coverage* have been used to study the systems' performance as already proposed in et al. [27]

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 neighbour 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 $rt \in REC(r)$ 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 $p \in P$ [32]:

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

Accuracy. Accuracy is considered as one of the most preferred *quality indicators* for Information Retrieval applications [28]. 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 [21],[12],[11]:

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

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

Catalog coverage. This metric is particularly suitable to measure the performance predictions of recommendation systems that suggest a list of items [16]. Given the set of projects U_p, we compare the number of recommended topics with the global number of the available ones i.e., $REC_N(p)$ and T respectively. Reversely from the previous two metrics, the Catalog Coverage measures the suitability of the delivered topics considering all the possible set of values. From the evaluation point of view, it is interesting to assess the impact of N value on the coverage stability, meaning what values of N impacts on the overall prediction performances.

$$coverage@N = \frac{\left|\cup_{p \in P} REC_N(p)\right|}{|T|} \quad (8)$$

4.4 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.
- **RQ$_2$**: *Is the entangled approach able to improve the MNB network'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 two different experiments. Section 5.1 discusses the CFTop results by variating different parameters. The results obtained with the entangled approach (i.e., the combination of CFTop and MNB network approaches) are investigate in Section 5.2 .

5.1 CFTop evaluation

RQ$_1$: *Which collaborative filtering configuration brings the best performance to CFTop?*

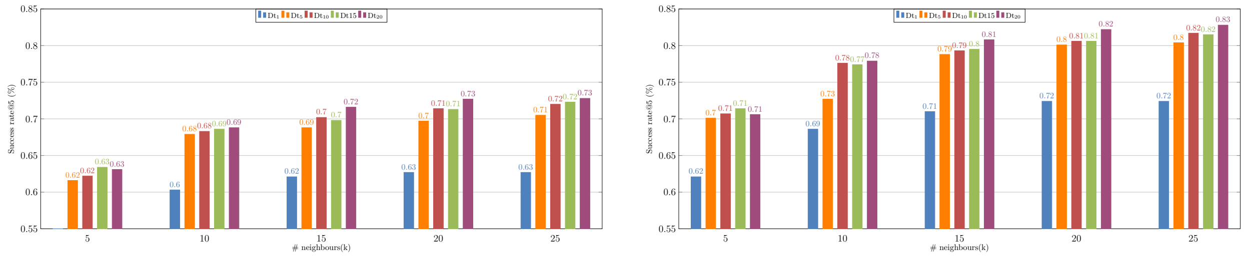
To find the best configuration in terms of prediction performances, we experiment with different CFTop configuration by variating the available parameters i.e., number of neighbors k, the recommended topic cut-off value N, and the involved dataset.

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}. The bar charts in Fig. 7(a) and 7(b) show the average success rates of all ten folds of CFTop.Both figures depict the results of CFTop applied on the different datasets defined in Section 4.1 i.e., Dt_1, Dt_5, Dt_{10}, Dt_{15}, and Dt_{20}. In particular, Fig. 7(a) and Fig. 7(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 neighbours 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. 8. 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 [12], the figure shows that a higher value of t reaches better accuracy for all values of N.

As defined in Section 4.3, the coverage metric is the percent of recommended topic in the training data that the model is able to recommend on a test set. In our experiment, we are measuring the catalog coverage value among all the possible topic (i.e., 15,743 topics). For each dataset (i.e., Dt_1, Dt_5, Dt_{10}, Dt_{15}, Dt_{20}), Table 4 reports the average coverage value for all ten rounds. The average catalog coverage of all folds decreases from 9.306% (Dt_1) to 1.805% (Dt_{20}) because there are no training data to recommend infrequent topics. Having a higher value of t has a negative impacts on the global catalog coverage value, because too many training data are discarded due to the topic frequency cut-off t. Differently from the discussed metric outcomes, this experiment shows how an higher values of topic frequency cut-off negatively impacts on the catalog coverage metric.

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. 9 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 with k = 25 and t = 20 as configuration settings. 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.



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

Figure 7: Success rate with 5 and 10 input topics.

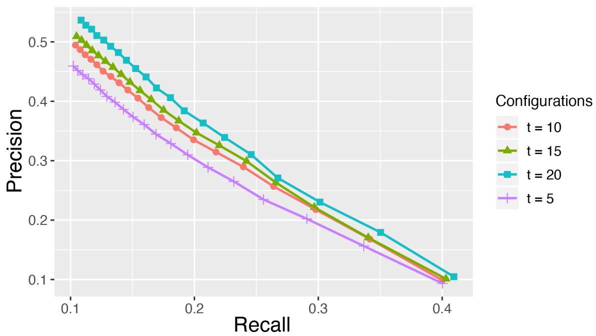


Figure 8: Evaluation of the different configuration.

N	Catalog Coverage				
	Dt_1	Dt_5	Dt_{10}	Dt_{15}	Dt_{20}
2	2.313	1.433	1.075	0.886	0.715
4	3.925	2.362	1.753	1.440	1.143
6	5.494	3.232	2.346	1.858	1.478
8	7.075	4.035	2.835	2.185	1.737
10	8.720	4.788	3.239	2.458	1.920
12	10.385	5.472	3.615	2.702	2.082
14	12.073	6.120	3.934	2.915	2.223
16	13.872	6.729	4.216	3.088	2.339
18	15.753	7.252	4.475	3.244	2.442
20	17.746	7.770	4.699	3.369	2.521
AVG	**9.306**	**4.737**	**3.111**	**2.339**	**1.805**

Table 4: Coverage values for the proposed datasets (i.e., Dt_1, Dt_5, Dt_{10}, Dt_{15}, Dt_{20})

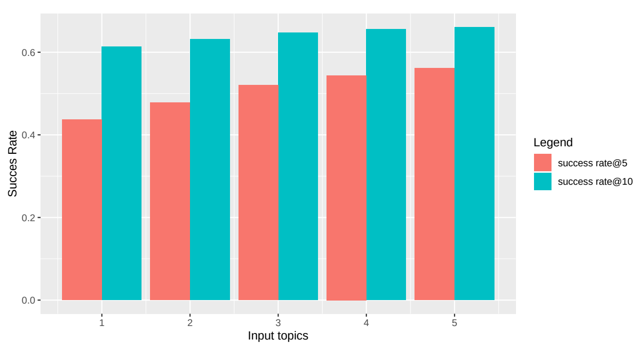


Figure 9: Evaluation of the different input topics.

The quality evaluation demonstrates that CFTop achieves better performance in terms of accuracy and success rate by increasing the number of considered neighbours *k* and filtered data *t*. However, the conducted experiment shows that an higher values of topic frequency cut-off *t* negatively impact on the global catalog coverage. For this reason, the *t* values should be careful selected during the filtering phase to obtain balanced results in term of accuracy, success rate, and global catalog coverage. However, the precision and recall values are still low, suggesting that bias lives in the users topic.

5.2 Entangled evaluation

RQ$_2$: *Is the entangled approach able to improve the MNB network's overall performance?*

Due to the internal construction of the MNB network, the direct comparison of the two approaches can bring biased results. Thus, we combined the two approaches to investigate potential improvements. We create this *entagled* configuration by feeding CFTop with the 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. We conduct our experiment on two of the proposed datasets, i.e., Dt_{20}, and Dt_1 that are the detasets with lower and higher topic to be recommended respectively. For experiment purposes, we variate the number of recommendation items as well as the number of input topics i.e., *N* and *Tin* values respectively. From the previous assessment, we figured out that the number of inputs leading the best results is Tin=5. Thus, we compare the outcomes considering the minimum number of input topics provided by the MNB network, i.e., Tin=2. The results demonstrate that the MNB network gains notable improvement by means of the entangled configuration in terms of the mentioned metrics i.e., accuracy, success rate, and catalog coverage. We witness that CFTop outperforms the MNB network by augmenting the number of recommended items.

The results in Table 6 demonstrate that the MNB network gains notable improvement by means of the entangled configuration in terms of the mentioned metrics i.e., accuracy, success rate, and catalog coverage. Table 5 also confirms this trend by performing the same experiment with D_1 dataset. We witness that CFTop outperforms the MNB network by augmenting the number of recommended items. For both datasets i.e., D_1 and D_{20}, after K=8 the accuracy and success rate overcomes the MNB network results considering the CFTop's best configuration even though the overall accuracy trend is decreasing. This happens because enlarging the set of recommended items impacts negatively on the precision values. Reversely, the success rate rises up to 0.855 and 0.531 with D_1 and D_{20} respectively. As witnessed for the accuracy value, the MNB network records better results until a certain threshold of output items. This degradation in performance is due to the internal probabilistic model used by the approach.

Although the examined metrics are useful to analyze the overall performances, the catalog coverage can evaluate properly the capability to recommend a *list* of items instead of a single one. Looking at the results, we can observe a substantial increase after 8 output items. As expected, the coverage dramatically increases with a larger number of outcomes for both of the considered approaches. Nevertheless, the positive gap of the entangled configuration is greater than the MNB network value. Considering the Out=20, the maximum value reached by the MNB network is 39.636 while the best configuration in the entangled experiment reaches a coverage of 58.725.

These findings can be explained by considering the nature of the considered topics. As said before, the MNB network can predict only featured topics as training the entire set of GitHub topics is not possible due to the computation issues. Reversely, CFTop covers a larger set of topics by enabling the described collaborative filtering technique. In this way, the *entangled* is capable of suggesting both featured and not featured topics to the final user and enlarging the possible set of outcomes.

The entangled approach success in improving the prediction performances. By variating both the input and output number of topics, the accuracy and success rate experienced an enhancement even though the former reached low values. The MNB network lacks in catalog coverage, as clearly demonstrated by the higher value of the entangled experiment.

6 THREATS TO VALIDITY

This section discusses the threats that may affect the results of the evaluation. We also list the countermeasures taken to minimize these issues.

The *internal validity* could be compromised by the dataset features i.e., the number of projects for each topic, the number of available outcomes. We tackle this issue by variating the aforementioned parameters to build datasets with different characteristics. In this way, several scenarios are used to evaluate CFTop's overall performances.

External validity concerns the rationale behind the selection of the GitHub repository used in the assessment. As stated in the related section, we download randomly repositories by imposing the quality filter on the stars. Nevertheless, some repositories could be tagged with topics that can affect the quality of the graph computed in the data extraction phase. To be concrete, a user can label its repository using terms that are not enough descriptive i.e., using infrequent or duplicated terms in the topic list. To deal with this issue, we apply the topic filter as stated in Section 3.1 to reduce the possible noise during the graph building.

Threats to *construction validity* concerns the choice of the MNB network as the baseline in the conducted experiment. First of all, the availability of the replication package allows a more comprehensive evaluation rather than other approaches. As we claimed before, the two approaches are strongly different from the construction point of view including the recommendation engine and data extraction components. To make the comparison as fair as possible, we run the MNB network on the same datasets by adapting the overall structure for the ten folder validation.

7 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 [15]. 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 [25], 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. [18] 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

	Recall			Precision			Success rate			Catalog coverage		
N	MNB	Tin=5	Tin=2	MNB	Tin=5	Tin=2	MNB	Tin=5	Tin=2	MNB	Tin=5	Tin=2
2	0.026	0.017	0.017	0.137	0.077	0.077	0.240	0.148	0.148	0.175	0.411	0.411
4	0.042	0.035	0.039	0.123	0.081	0.090	0.383	0.287	0.283	0.314	0.617	0.802
6	0.049	0.064	0.055	0.104	0.098	0.085	0.441	0.449	0.358	0.398	0.919	1.119
8	0.055	0.085	0.065	0.093	0.098	0.075	0.499	0.531	0.397	0.474	1.272	1.462
10	0.061	0.100	0.074	0.087	0.092	0.069	0.550	0.572	0.439	0.557	1.626	1.783
12	0.066	0.111	0.082	0.081	0.086	0.064	0.584	0.603	0.467	0.620	2.008	2.100
14	0.067	0.120	0.090	0.074	0.080	0.060	0.601	0.624	0.493	0.658	2.431	2.425
16	0.068	0.128	0.097	0.067	0.074	0.056	0.616	0.643	0.514	0.687	2.825	2.775
18	0.069	0.136	0.103	0.062	0.070	0.053	0.632	0.660	0.531	0.718	3.169	3.163
20	0.073	0.143	0.107	0.060	0.066	0.050	0.662	0.675	0.545	0.768	3.544	3.575

Table 5: Results for the entangled approach for the Dataset Dt_1.

	Recall			Precision			Success rate			Catalog coverage		
N	MNB	Tin=5	Tin=2	MNB	Tin=5	Tin=2	MNB	Tin=5	Tin=2	MNB	Tin=5	Tin=2
2	0.035	0.031	0.031	0.206	0.118	0.118	0.363	0.217	0.217	9.068	8.593	8.593
4	0.075	0.063	0.088	0.221	0.119	0.166	0.600	0.389	0.466	19.405	15.340	15.912
6	0.094	0.121	0.119	0.187	0.153	0.149	0.635	0.601	0.549	24.682	22.131	21.780
8	0.106	0.171	0.142	0.159	0.162	0.133	0.680	0.704	0.599	27.967	29.296	27.428
10	0.116	0.204	0.163	0.140	0.156	0.123	0.701	0.754	0.644	30.719	35.296	32.967
12	0.124	0.230	0.181	0.124	0.146	0.114	0.719	0.788	0.681	32.786	40.659	38.373
14	0.130	0.254	0.201	0.111	0.138	0,109	0.733	0.808	0.706	34.308	45.912	43.098
16	0.135	0.274	0.215	0.101	0.131	0.102	0.745	0.829	0.722	35.742	50.505	47.582
18	0.143	0.290	0.227	0.095	0.123	0.096	0.759	0.840	0.736	37.644	54.615	51.318
20	0.150	0.306	0.241	0.090	0.117	0.092	0.772	0.855	0.756	39.636	58.725	54.923

Table 6: Results for the entangled approach for the Dataset Dt_{20}.

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.

REPERSP tool [33] aims to recommend GitHub projects by exploiting users' behavior. As the first step, the tool computes the similarities between projects using the TF-IDF weighting scheme to obtain the content similarity matrix. Additionally, REPERSP captures the developer's behavior by considering his activity on GitHub i.e., create, star, and fork actions over projects. A different value is assigned for each type of action to create a user-project matrix. Finally, the tool combines the two similarity matrixes to deliver the recommended projects. To assess the quality of the work, REPERSP is compared with the traditional collaborative filtering techniques i.e., user-based and item-based. The study is conducted over two groups with different users, projects, and purposes. The results show that the proposed tool outperforms the mentioned techniques in terms of accuracy, precision, and recall.

8 CONCLUSIONS AND FUTURE WORK

GitHub is nowadays the most popular platform to handle and maintain OSS projects. Topics have been introduced in 2017 to promote the project's visibility on the platform. Although a couple of works face the problem, there are additional challenges to be faced. In this work, we have presented CFTop, a collaborative filtering based recommender system to suggest GitHub topics. By representing repositories and related topics in a graph format, we built a user-item matrix and apply a syntactic-based similarity function to predict missing topics. To assess the prediction performances, we compared CFTop with a well-founded work based on an ML technique in terms of success rate and accuracy. The results show that CFTop outperforms the opponent with a relevant improvement of the mentioned metrics. Furthermore, we combined the two approaches in an *entangled* evaluation to explore possible enhancements. We figured out that CFTop gained a significant boost in prediction performances by employing the MNB network outcomes as input topics. Nevertheless, the accuracy didn't reach higher values in all the experimental settings. To our best knowledge, it depends on the similarity function used in the recommendation engine as well as on the heterogeneity of the dataset. Thus, we are planning to extend CFTop by adding different degrees of similarity i.e., semantic analysis on topics, README encoding to name a few. Moreover, we can enlarge the evaluation by considering other common metrics in the collaborative filtering domain such as sales diversity and novelty. These augmentations will be considered as possible future work.

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] V. Cosentino, J. L. C. Izquierdo, and J. Cabot. 2016. Findings from GitHub: Methods, Datasets and Limitations. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. 137–141.

[10] 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

[11] 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

[12] 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

[13] 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. In *Proceedings of the Evaluation and Assessment in Software Engineering* (Trondheim, Norway) *(EASE '20)*. Association for Computing Machinery, New York, NY, USA, 71–80. https://doi.org/10.1145/3383219.3383227

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

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

[16] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. 2010. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems - RecSys '10*. ACM Press, Barcelona, Spain, 257. https://doi.org/10.1145/1864708.1864761

[17] 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.

[18] 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

[19] 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

[20] 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

[21] 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

[22] 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

[23] Phuong T. Nguyen, Juri [Di Rocco], Davide [Di Ruscio], and Massimiliano [Di Penta]. 2020. CrossRec: Supporting software developers by recommending third-party libraries. *Journal of Systems and Software* 161 (2020), 110460. https://doi.org/10.1016/j.jss.2019.110460

[24] 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

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

[26] 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

[27] 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

[28] 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

[29] 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

[30] 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

[31] 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/

[32] 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

[33] Wenyuan Xu, Xiaobing Sun, Jiajun Hu, and Bin Li. 2017. REPERSP: Recommending Personalized Software Projects on GitHub. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 648–652. https://doi.org/10.1109/ICSME.2017.20

[34] 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