

Identifying Emergent Leadership in Open Source Software Projects Based on Communication Styles

Yuekai Huang^{1,3}, Ye Yang^{4,*}, Junjie Wang^{1,2,3,*}, Wei Zheng⁵, Qing Wang^{1,2,3,*}

¹Laboratory for Internet Software Technologies, ²State Key Laboratory of Computer Sciences, Institute of Software Chinese Academy of Sciences, Beijing, China;

³University of Chinese Academy of Sciences, Beijing, China; *Corresponding author

⁴School of Systems and Enterprises, Stevens Institute of Technology, Hoboken, NJ, USA;

⁵School of Business, Stevens Institute of Technology, Hoboken, NJ, USA;

huangyuekai18@mails.ucas.ac.cn, yangye@gmail.com, junjie@iscas.ac.cn, wzhen11@stevens.edu, wq@iscas.ac.cn

Abstract—In open source software (OSS) communities, existing leadership indicators are dominantly measured by code contribution or community influence. Recent studies on emergent leadership shed light on additional dimensions such as intellectual stimulation in collaborative communications. This paper aims to mine communication styles and identify emergent leadership behaviors in OSS communities, using issue comments data. We start with the construction of 6 categories of leadership behaviors based on existing leadership studies. Then, we manually label leadership behaviors in 10,000 issue comments from 10 OSS projects, and extract 304 heuristic linguistic patterns which represent different types of emergent leadership behaviors in flexible and concise manners. Next, an automated algorithm is developed to merge and consolidate different pattern sets extracted from multiple projects into a final pattern ranking list, which can be applied for the automatic leadership identification. The evaluation results show that iLead can achieve a median precision of 0.82 and recall of 0.78, outperforming ten machine/deep learning baselines. We argue that emergent leadership behaviors in issue discussion should be taken into consideration to broaden existing OSS leadership viewpoints.

Index Terms—Leadership, Communication style, Linguistic pattern, Open source software

I. INTRODUCTION

Steady influx of developer resource is key to healthy and sustainable OSS communities [1], [2]. The presence of community leaders, who both steer the development direction and motivate more developers to contribute is crucial to ensure a successful outcome of an OSS project [3]. In addition, it is also reported that developer retention is highly associated with the attention and treatment they receive in the project [4]–[6]. Therefore, it is an essential function for an OSS community to effectively monitor and appropriately recognize developer contribution in timely and fair manners.

In light of the importance of having leadership figures, many studies have investigated various developer leadership factors across OSS communities. Most rely on metrics representing individuals' code contribution or community influence. Example metrics of code contribution include code size (e.g., #lines of code added/removed [7], #commits [8], [9]), code structure (e.g., code instability [8]), code quality (e.g., #problems reported [7], #buggy commits [9]), or code topics [10]. Example metrics of community influence include the number

of followers (e.g., #followers [3], [11]) and influence on other developer's motivations [12].

Most of today's OSS communities operate beneath an umbrella organization, while others are organized entirely independently, and yet others follow a strategy somewhere in between [13]. Different OSS organizational forms correspond with different leadership structure and coordination strategies, which complicates the measurement of leadership effectiveness. As detailed in Section II, we find that there is mismatch between these developers' contribution in issue discussion and their contributions measured by other dimensions such as code contribution or community followers. We conjecture that in OSS communities where development is highly autonomous, and explicit coordination and control are loose or non-existent, those who know how to communicate effectively with others will contribute more in leading, coordinating, and influencing the development. This indicates the need of a new lens in identifying and recognizing OSS leadership behaviors corresponding to more effective and efficient issue resolution.

To bridge the gap, we start with constructing six categories of emergent leadership behaviors applicable to OSS projects from existing leadership models [3], [14]–[18]. These include proposal, redirection, confirmation, inquiry, operation, and volunteer (as show in Table I). And then we propose an automated approach, named iLead, to mine communication styles and identify such emergent leadership behaviors in OSS communities, using issue comments data. The construction of iLead consists of three steps. First, we carry out the data labelling process to identify the category of leadership per issue comment. Second, we adapt the definition of linguistic patterns in previous studies [19]–[21] and manually extract the heuristic linguistic patterns to express different types of communication styles in flexible and concise manners. Third, based on the two previous steps, we develop an automatic algorithm to merge and consolidate different pattern sets extracted from multiple projects into a final pattern ranking list. When applying iLead for leadership identification, the final pattern ranking list can be used to automatically match new issue comments and identify corresponding leadership behaviors. The evaluation is conducted on 10 popular OSS projects with 10,000 comments. Results show that iLead can achieve a median precision of 0.82

and recall of 0.78, outperforming ten baselines.

To demonstrate practical usefulness, we also conduct empirical analysis and human evaluation of the identified leadership behaviors with iLead. Results motivate the need for considering the emergent leadership behaviors in issue discussion to broaden existing OSS leadership viewpoints. Practical insights on community building and leadership skill development for OSS community developers are also discussed. To summarize, the main contributions of this paper are as follows:

- The construction of 6 categories of emergent leadership behaviors in OSS projects. This is the first taxonomy of the emergent leadership behaviors in issue discussions of OSS communities.
- The development of iLead to automatically identify emergent leadership behaviors in OSS projects. To our knowledge, this is the first study on automatic mining of emergent leadership behaviors in OSS issue discussion communications.
- Publicly accessible dataset and source code¹ to facilitate the replication of our study.

II. BACKGROUND AND MOTIVATION

A. Leadership Models and Theories

It is important to note the difference between “leadership” and “leader”, in that leadership status is not necessarily based on an OSS community position or designated authority, i.e., leader status [13], [22], [23]. Instead, leaders emerge and earn their status through incremental influences and contributions to OSS communities. In the context of this study, we adopt the Yukl’s definition of leadership as “influence exerted over other people to guide, structure, and facilitate relationships in a group” [24]. Existing work on leadership theories and styles can be largely categorized into two polarized yet important leadership styles: transformational and transactional, depending on followers’ behavioral response [25]. Transformational leadership [26] requires leaders to work with followers to stimulate the enthusiasm of subordinates by creating a vision and motivation. Transactional leadership [27] is a kind of leadership behavior that pays more attention to organizational management and performance. However, both styles are more focusing on traditional organizations with clear management hierarchy and team structure, which are not directly applicable to OSS communities.

In recent years, a number of new theories characterizing leadership in virtual teams were proposed, such as shared leadership [24], [28] and emergent leadership [29]. Shared leadership [30] is a leadership mechanism existing in a team with clear hierarchy, and it is a process in which individuals in the group influence each other and work together towards the goal. Emergent leadership [29] is a spontaneous leadership mechanism as an emergent phenomenon that develops over time through group processes.

¹<https://github.com/iLeadGroup/iLead>

B. OSS Community, Project and Developers

An OSS community is an ecosystem comprising of a set of closely-related OSS projects, which draws expertise and contributions from a pool of developers [31]. OSS developers usually assume certain roles by themselves according to their personal interest, rather than being assigned a task by someone else [32]. This freedom in assuming roles/responsibilities, while significantly differing from traditional organizations, necessarily nurtures the open innovation that drives the success of OSS. As examples of such open contribution, developers may invest effort on a specific OSS project instead of others, and he/she may choose to submit code contribution, or participate in issue discussions. As examples of developer impact measures, the number of commits and the number of followers typically reflect a developer’s relative influence in the OSS community. Consequently, an OSS developer’s role in an OSS project may vary significantly based on his/her interest, availability, and expertise. Nakakoji et. al. [32] proposed to classify OSS developers’ roles into 8 types including project leader, core member, active developer, peripheral developer, bug fixer, bug reporter, reader, and passive user. The order implies the relative significance of their contribution. In addition, in measuring developer’s contribution, they usually play greater weight on the developer’s code contribution [33], [34], i.e., the number of commits, which is readily accessible from OSS platforms such as GitHub or SourceForge.

C. A Motivational Example

To explore differences among different leadership indicators, we compare three indicators, i.e., #commits, #followers, and #issue comments, across a set of contributing developers from the *atom*² OSS project. Figure 1 illustrates the comparison results of three example developers. It is clear that developer *Dev1* (id:1476) has the highest #commits and #followers among all three, followed by *Dev2* (id:7910250) then *Dev3* (ID:4525388). *Dev1* and *Dev2* will likely be considered as having greater contributions than the third one following existing leadership indicators [3], [7], [8], [11] because they have more code contributions or have more followers. However, although developer *Dev3* has less commits, he or she has actually published the most issue comments. Specifically, there are 267 comments made by *Dev3*, including the following examples:

- “Have you tried safe mode? Try –safe and see if the issue stops.”
- “What is your atom version?”
- “Duplicate of #1667”

Through these communications, the developer *Dev3* either provides alternative solution to the issue reporter, or asks clarification questions, or redirects the issue to other similar ones. These behaviors are actually representative of emergent leadership and managerial responsibilities, which are essential to facilitate collaborative discussion and issue resolution.

²<https://github.com/atom/atom>

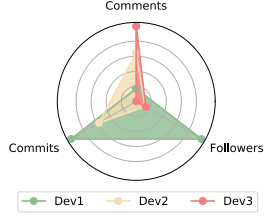


Fig. 1. Example developers in *atom*

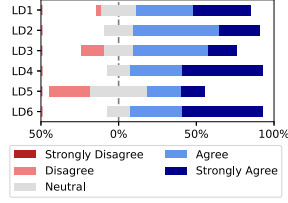


Fig. 2. Survey results

To further examine the differences of these indicators at project level, we rank all developers in *atom* by the number of comments and the number of commits, respectively, and compare the top 25% of the developers in the two rankings. Note that the number of followers is defined at the overall GitHub, therefore, we did not include it for the project level comparison here. We find that large mismatch exists between these two lists. Specifically, 59% of developers on the two rankings are different, and those appearing in both rankings correspond with different orders as well. This indicates that existing code commit metric might significantly underestimate the influences and contributions of many developers from issue discussion aspects. Besides, although the number of comments can represent the frequency of developer' participation in issue discussion to a certain extent, it misses abundant semantic details reflecting underlying leadership behaviors, as illustrated in the examples above. Therefore, this study is motivated to develop an effective approach to mine and identify such communication associated leadership, to offer a complimentary perspective to existing studies and help OSS projects to enable informed decisions such as understanding leadership dynamics and influences, recommending leadership styles for stimulating issue discussion, reflecting on leadership variation, etc.

III. CATEGORIZING LEADERSHIP BEHAVIORS

This section presents a set of six categories of "emergent leadership" behaviors in collaborative issue discussion processes in OSS communities. Unlike previous studies on OSS leadership, we take a different approach in this study, by considering leadership as an influence process, not the totality of the behaviors of a particular developer. To that end, we aim at examining leadership behavior at individual comment level, instead of at the developer level (e.g. measuring the corpus of comments by a person). Such a conceptualization of leadership has two main advantages. First, it allows us to observe comment-level leadership behavior with finer granularity. Specifically, this enables us to examine immediate impact of each leadership behavior on other developers (e.g., as demonstrated by responding comments from others) and contextual factors surrounding leadership influence process, such as stimulating downstream discussions, and/or accelerating issue resolution. Second, using these comment-level leadership behaviors would also allow us to aggregate behaviors to the individual level. As OSS communities are constantly changing, more research is needed for aggregating comment-

level metrics at the individual/team level, with respect to particular usage scenarios.

Existing leadership literature has offered many taxonomies of leadership behaviors. We build the six categories based on reviewing transformational leadership [14], Yukl's comprehensive taxonomy [17], [18], shared leadership [3], and emergent leadership [35]. The six leadership categories were adapted from 12 Yukl's comprehensive taxonomy [17], [18] according to the matching leadership functions in the OSS context, particularly for achieving the goal of efficient issue resolution. Table I summarizes these leadership categories with description and comment examples, along with the mapping with the 6 selected Yukl's categories [17], [18]. The other 6 Yukl's categories concerns more on change-oriented, hierarchy and external components of leadership, which are not included in this study due to less relevance to the OSS issue discussion context. The leadership included in this study is as follows:

LD1 (Proposal). This category corresponds to the short-term planning category from Yukl's taxonomy, and refers to the idea proposal type of issue discussion. Issue discussion is an intelligence-intensive problem solving process, and LD1 discussions usually shape potential resolution plans, and accelerate the issue resolution. We believe it is also consistent with the intellectual stimulation/inspiration category in transformational leadership [14], [36].

LD2 (Redirection). This category corresponds to the supporting category from Yukl's taxonomy, and refers to comments providing supportive information to redirect attentions to a new topic (e.g., concepts, processes, or resources) or to a more relevant information sources. Due to the open and interconnected nature of OSS communities, such topics and information sources may be either internal (e.g., a related issue within the OSS project) or external (e.g., a site outside project).

LD3 (Confirmation). This category corresponds to the recognizing category from Yukl's taxonomy, and refers to the leadership function in confirming or reiterating an idea or opinion, which facilitates the consensus processes among members, through implicit voting or shaping shared perception among developers.

LD4 (Inquiry). This category corresponds to the consulting category from Yukl's taxonomy, and refers to cases where developers ask for more clarification question about the issue being reported/discussed. Example follow-up inquiries include asking for specific project version info or issue reproduce steps. Such issue comments typically provide some sort of confirmation to former opinions, and facilitate the issue discussion threads.

LD5 (Operation) and **LD6 (Volunteer)** represent two distinct types of administrative functions during issue life cycle, corresponding to the monitoring and clarifying roles categories, respectively, from Yukl's taxonomy. LD5 refers to comments related to either closing or reopening an issue, and LD6 relates to volunteering in undertaking some wrap-up bug fixing actions, respectively.

To validate the representativeness of the 6 comment-level leadership categories, we design and conduct a survey to

TABLE I
CATEGORIES OF EMERGENT LEADERSHIP BEHAVIORS IN OSS PROJECTS

Comment Type	Description	Examples	Yuki's Mapping
LD1 (Proposal)	Propose a solution or alternative	1. Try install Bitcoin Core 0.17. 2. You could try to remove Homebrew's binutils.	Planning
LD2 (Redirection)	Guide to other topics or places to discuss	1. This is a duplicate of #17576. 2. This should be an issue in ... repo.	Supporting
LD3 (Confirmation)	Confirm an issue or opinion	1. I agree with @sipa. 2. Confirming that I got the same error on ...	Recognizing
LD4 (Inquiry)	Ask for more information	1. Where did you get the code from? 2. What version of Boost are you using?	Consulting
LD5 (Operation)	Suggest to close or reopen the issue report	1. Going to close due to lack of information. 2. It's just a workaround... plz reopen this.	Monitoring operation
LD6 (Volunteer)	Voluntary acceptance of a task	1. I would like to work on this. 2. I will open a PR fixing this soon.	Clarifying roles

gather feedback from OSS developers. The main question asked is whether they agree that these categories capture the relevant leadership behaviors which contribute to the issue discussion and resolution processes. We distribute the survey to the contributors of 10 popular projects (as shown in Table II) on GitHub, and received 27 responses. The results, as shown in Figure 2, indicates that an average of 71% of developers believe that these six categories of leadership provide a good representation of emergent leadership behaviors in OSS issue discussion communications. Furthermore, even the most controversial LD5 has no strong disagree, and the number of disagrees is only 26% (7/27). One possible reason for the disagreement on LD5 might be because developers tend to link leaders or leadership behaviors with more technical contribution, while LD5 mostly deals with process monitoring and technical focused.

IV. APPROACH

When developers explain something similar, they are likely to use recurrent textual expressions in their comments [37]. Existing studies frequently rely on such expressions to extract linguistic patterns in order to facilitate context understanding and task automation [19]–[21], [38], [39]. In this section, we present a novel approach, iLead, which employs a set of heuristic linguistic patterns to automatically identify the emergent leadership behaviors. Figure 3 illustrates the overview of the iLead approach, consisting of following four steps.

A. Manual Labelling

In this step, we will mark each issue comment with a corresponding leadership label (i.e., LD1, LD2, ..., or LD6), as introduced in Section III. Specifically, the first three authors individually label the same issue comments, then discuss and merge the results on each individual issue comment. The merged leadership labels serve as the ground truth in the later steps, i.e., pattern consolidation and evaluation.

To ensure the validity of the manual labelling outcomes, we adopt a labelling process in the existing work [40], the process includes four activities, i.e., defining a labelling schema, independently labelling by authors, group reviewing of individual labels, and building a final consensus corpus.

Specifically, the labelling schema is based on the six categories of emergent leadership behaviors, as mentioned in Section III, plus an additional category (N) for comments not containing any leadership behaviors. A group discussion is conducted to ensure that everyone involved in labelling have a common understanding of these six leadership behaviors in open source issue discussion context. Then, each comment will be independently labelled by the first three authors and produce the label of a corresponding leadership behavior, or *N* indicating no leadership behavior. The average Cohen's Kappa is 0.75 between each pair of annotators, indicating substantial agreement among the annotators. Next, a group meeting is held to review individual labelling results. When a discrepancy arises, each author involved in labelling provides rationale for his/her choice, and the group then discusses and reaches an agreement. Finally, after several rounds of discussions, agreements are reached for every comment and we get the final consensus corpus.

B. Linguistic Pattern Extraction

The second step is to manually extract linguistic patterns from the textual content of issue comment, which can be used for predicting each of the six leadership behaviors. For all comments with leadership labels (i.e., LD1 to LD6), we first extract the key sentence in the comments which determines it belonging to certain leadership behavior. Take the following LD4 (Inquiry) comment as an example “*Can you provide more information? I've just done a build of master using the 64 bit Windows instructions and it's working correctly.*” The key sentence is “*Can you provide more information?*”, since the other sentences help little in identifying the leadership behaviors. After that, the more structured part of the sentence like “*...can...you...provide...*” will be identified and the linguistic pattern will be generated based on it. In this example, a pattern containing three elements about “*can*”, “*you*” and “*provide*” will be extracted. The patterns generated from the above process is relatively large, if only concrete word/phrase elements are considered. For example, “*Can you provide more information*”, “*Could you give us the steps to reproduce*”, “*Would you mind uploading the dockerfile*” and “*Are you sure you're not running master*” all have similar

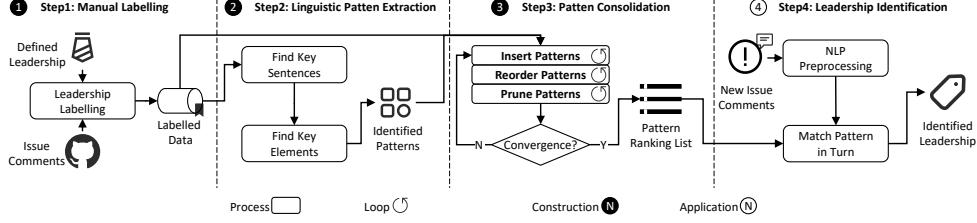


Fig. 3. iLead overview

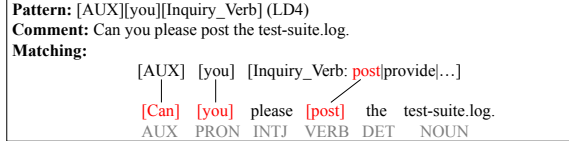


Fig. 4. Example of leadership identification with pattern

structures and belong to LD4 (Inquiry). Since the words are different, we get four patterns: “can you provide”, “could you give”, “would you mind uploading” and “are you sure”. This will lead to a huge pattern set, which is difficult to maintain.

In order to reduce the number of patterns, we follow the existing research [20], [21] by considering not only words as elements of a patterns, but also the Part-of-Speech (POS) tag of the words, and use dictionary to represent a group of words that have similar usage. For a pattern with *Dictionary*, it will be used as a template which means each word in the dictionary can be applied to the corresponding positions and formulates a specific pattern. As a result, the four LD4 patterns in the previous paragraph can be merge into “[AUX(POS)] [you(word)] [Inquiry_Verb(Dictionary)]”. This not only reduces the total number of patterns, but also enables more sophisticated pattern matching. Figure 4 shows how this pattern can be applied in leadership identification.

Two of the authors participate in the process and employ open card sorting [41] during this process. Specifically, we do not pre-define the relevant patterns and dictionaries. Each author extracts patterns independently on the collected data and organize their corresponding pattern set. Then group meetings are conducted to discuss and merge the pattern sets. After that, the merged pattern set will be maintained by the authors together and each author can add new patterns to the set or refine existing patterns in their subsequent pattern extraction. Finally, we get a set of patterns that can be used to match comment and identify leadership. For a complete list of all linguistic patterns, please refer to our project website [42], and a total of 304 patterns are extracted for the follow-up leadership identification (see details in Table III).

C. Pattern Consolidation

In this step, we develop an iterative pattern consolidation algorithm to merge patterns extracted from individual projects, monitor and tune the performance change of iLead against

the newly merged pattern set in each iteration. This step is essential due to two main reasons: (1) patterns extracted from individual projects are at the risk of pattern over-fitting; (2) in cases that multiple patterns can be simultaneously applicable, appropriate priorities need to be assigned in order to avoid complexity and ambiguity caused by overloaded patterns. To address the pattern over-fitting and overloading issues, we develop an algorithm to automate the pattern merge and consolidation process. This algorithm consists of three sequential tasks (i.e., insert, reorder, and prune), and stops when the performance of iLead converges to a satisfactory level. We will elaborate each of the three tasks next.

1) **Insert Patterns:** This step takes two input: one is a target pattern ranking list (i.e., initially an empty list), and the other is a new pattern set to be inserted into the target pattern ranking list.

Specifically, for each new pattern, the algorithm will iteratively examine the impact of merging it into the target pattern ranking list, and only accept it when it positively contributes to the prediction performance of iLead. Following an existing study [20], [21], [38], we choose the F1-Score as the performance metric. Specifically, the algorithm will probe every candidate position in the pattern ranking list for inserting a new pattern, calculate the performance of iLead in each probing, record the best performance and the corresponding optimal insertion position. As a result, a new pattern will only be inserted into the merged list if it positively contributes to the performance of iLead, and the optimal position is the one corresponds to the best performance. Otherwise, the new pattern will be discarded and will not be merged into the pattern ranking list.

2) **Reorder Patterns:** We observe that, after the previous merging process, some issue comments may be incorrectly identified according to a loosely relevant pattern with a higher priority, while a more relevant pattern is ranked lower in the list and never be applied. To address this, the algorithm reorders the ranking of patterns by examining these incorrectly labelled comments, to promote the overall performance on the whole dataset. The ranking update is conducted iteratively and dynamically in terms of each incorrectly labelled comment by current pattern ranking list.

In detail, for an incorrectly labelled comment (the comment mislabelled by the current pattern ranking list), we define the highest ranking pattern which correctly labels it as *realwinner*, while the patterns which incorrectly match it and higher than

realwinner as the *fakewinners*. If *fakewinners* list is not empty, algorithm will try the ranking update. Specifically, algorithm will insert the *realwinner* ahead of the *fakewinners* in the pattern ranking list, and examine the leadership labelling performance on the whole dataset (i.e., F1-Score). If the performance increases with this new pattern ranking list, the update is accepted and the current pattern ranking list will change to the new pattern ranking list. Then the algorithm iterates with the next incorrectly labelled comment found by the current pattern ranking list until it remains unchanged. As a result, we obtain the updated pattern ranking list which will be used next.

3) **Prune Patterns:** During this task, the algorithm will identify excessive patterns whose existence negatively contributes to the performance of iLead. Similar to the previous task, the pattern pruning is conducted iteratively and dynamically by examining each incorrectly labelled comment.

In detail, for each incorrectly labelled comment, the algorithm first detects the *realwinner* (i.e., the highestly ranked pattern which is able to correctly label the comment), as well as all *fakewinners* (i.e., the patterns which incorrectly match the comment and higher than *realwinner*). After that, the algorithm will remove the *fakewinners* in the pattern ranking list, and accept the update if the performance (i.e., F1-Score) increases. Then the algorithm iterates with the next incorrectly labelled comment found by the refined pattern ranking list until it remains unchanged.

After that, we will obtain the pattern ranking list and if the patterns from a new project need to be added, the algorithm will start executing from *Insert Patterns* again and use this list as the input (instead of the original empty list).

D. Leadership Identification

After completing the construction of pattern ranking list, we can use it to identify leadership behaviors. Specifically, for a new issue comment, iLead first preprocesses it to get the words and the Part-of-Speech (POS) tags. Then, it matches the preprocessed data with element(s) specified in a given pattern. We employ the following two distance constraints between elements, adapted from existing study [43], to promote the performance of leadership identification. The first constraint is that the latter matched element cannot appear before the prior matched element, since the pattern is extracted by traversing the words in sequence. The second constraint is that the distance between two matched elements should be no more than three, which is to avoid the mismatch of patterns since the leadership behaviors are usually expressed in a concise way. iLead will automatically match the comment with the patterns in the list in order, and following existing study [21], we choose the highest priority strategy which means the leadership is identified by the first matched pattern.

V. EXPERIMENT DESIGN

A. Research Questions

To evaluate the proposed approach, we answer the following three research questions:

- **RQ1 (Performance Convergence)** When can the generated linguistic patterns reach the state of convergence?
- **RQ2 (Performance Evaluation)** How effective is iLead in leadership identification on new projects?
- **RQ3 (Baseline Comparison)** Can iLead outperform other techniques in leadership identification?

B. Subject Projects

We collect 10 popular open source software projects from GitHub for evaluation. The projects are selected with the following criteria: (1) Popularity: with 10k+ stars and 1.5k+ forks; (2) Activeness: regularly updated within 1 month of the data collection date; (3) Diversity: representing different OSS organizational structures from various domains. A brief summary of the projects are summarized in Table II.

For the experimental projects, we use the GitHub's REST APIs to crawl the issues and related comments³. We only crawl the comments of closed issues since they represent the complete issue life-cycle. In addition, since it is our interest to compare the emergent leadership behaviors with traditional leadership indicators, we also extract two indicators at the developer level: the number of commits and the number of followers. For each project, we randomly sample a portion of issues and obtain 1,000 related comments (in Table II), and conduct manual labelling, as introduced in Section IV-A. The overview for the ground truth is shown in Table II.

C. Data Pre-processing

We observe that some developers would refer to the previous comments in their own comments. The referenced descriptions may contain another developer's leadership, and should be removed to reduce potential noise leading to incorrect matching. In this study, we implement this via removing the node used to highlight the reference descriptions in the HTML text.

In addition, we use Stanford natural language processing tool Stanza [54] to process the text in lowercase, tokenization, part of speech (POS) tagging and lemmatization, so as to obtain the data applied to the construction process of iLead. We also use Urlextract⁴ to annotate the URL contained in the comments, which is the element in certain patterns.

D. Experiment Setup

1) **Experiment for answering RQ1:** We start with the set of linguistic patterns established from one project, and then employ an iterative process as follows: 1) add another set of linguistic patterns established from a new project; 2) apply the pattern consolidation algorithm from Section IV-C to generate a pattern ranking list; 3) input the pattern ranking list to iLead for leadership identification; 4) evaluate the performance of iLead using the previously defined metrics. Throughout the iterative process, we continuously monitor the performance change of iLead, and stop the process when the performance converges (i.e., when the additional performance gain is less than 0.01). By the convergence time, we stop to incorporate

³The data is crawled between July 2020 and February 2021.

⁴<https://github.com/lipoja/URLExtract>

TABLE II
SUBJECT PROJECTS

Basic information							Leadership labels						
ID	Project	Link	Domain	Stars	#Comments	#Issues	LD1	LD2	LD3	LD4	LD5	LD6	TotalLD
P1	bitcoin	[44]	digital currency	57k	28,504	4,997	95	98	123	132	55	18	521
P2	sklearn	[45]	machine learning	47k	40,432	6,287	81	47	63	73	105	81	450
P3	ember.js	[46]	js framework	22k	32,225	5,833	69	54	132	77	89	43	464
P4	brew	[47]	package manager	29k	14,269	2,694	74	55	89	85	34	38	375
P5	atom	[48]	text editor	56k	83,004	15,621	50	255	141	42	56	6	550
P6	bokeh	[49]	visualization	15k	21,432	4,573	74	76	79	65	60	56	410
P7	efcore	[50]	entity framework	11k	55,820	11,596	87	140	47	162	72	7	515
P8	knex	[51]	SQL query	15k	10,337	2,177	166	28	87	104	56	39	480
P9	roslyn	[52]	compiler	15k	76,862	14,722	61	123	74	56	60	37	411
P10	solidity	[53]	program language	12k	11,567	2,513	44	52	84	90	46	28	344

new patterns, and finalize the pattern ranking list in iLead as the default pattern list used for further experiment.

2) **Experiment for answering RQ2:** To evaluate the performance of iLead, we use the remaining projects (projects unused in RQ1) as test sets and measure the identification performance of iLead on each test project. In addition, we will also examine the performance of iLead across the six leadership categories to obtain more comprehensive results.

3) **Experiment for answering RQ3:** We employ commonly-used machine learning based and deep learning based text identification approaches as baselines to compare with iLead. Specifically, we use Term Frequency and Inverse Document Frequency (TF-IDF) [55] to vectorize textual descriptions of comments, and use the term vectors as input for identification. For selection of machine learning classifiers, we evaluate four commonly-used machine learning models: Logistic Regression (LR) [56], Support Vector Machine (SVM) [57], Decision Tree (DT) [58], and Random Forest (RF) [59], and due to the limited space, we only show the best classifier (SVM). For the deep learning based classifiers, we utilize TextCNN [60], BiLSTM [61], CNN and RNN with Bert pre-training model [62].

In addition, we further explore whether our extracted patterns are helpful to the machine/deep learning models, following the existing studies [20], [21], [38]. In detail, we enrich the model inputs with the knowledge about pattern matching, which will be considered as a boolean vector, i.e., if a comment match with certain patterns, the corresponding position of the vector is set to 1, otherwise it is 0. We also run the above mentioned machine/deep learning models on the combined vectors, and they are short for <Model_Name-P>(e.g., TextCNN-P). In total, we have 10 baselines.

VI. RESULTS AND ANALYSIS

A. Answering RQ1. Performance Convergence

As introduced in Section V-D, We start with the pattern set extracted from one project, and iteratively add a pattern set extracted from a new project, consolidate into the pattern ranking list following the algorithm described in Section IV-C. In each iteration, we record the number of patterns added, deleted and changed, as well as the performance of iLead.

TABLE III
PERFORMANCE CHANGE WITH ITERATIONS (RQ1)

Pattern extraction results					
Projects	P1	P1-P2	P1-P3	P1-P4	P1-P5
#Patterns	198	250	287	300	304
#Add	-	55	41	13	4
#Delete	-	2	4	0	0
#Change	3	5	1	0	0
Leadership identification performance					
Precision	0.81	0.82	0.83	0.83	0.83
Recall	0.74	0.82	0.84	0.85	0.85
F1-Score	0.76	0.82	0.83	0.84	0.84

We find that the performance of iLead converges after five iterations. Table III summarizes the detail results.

Across the five iterations, we can see that the number of patterns increase from 198 to 304, and the accuracy increases from 0.81 to 0.85. We also observe that different projects share common linguistic patterns in issue comments, as seen by the smaller number of newly added patterns in each iteration. In addition, the most pattern changes caused by newly added patterns is in the second iteration, when adding the pattern set of *sklearn*, corresponding to 55 added, 2 deleted, and 4 changed patterns. Though there are new patterns added in the later two iterations as well, however, their added impact on the performance is diminishing. When the fifth project is added, only very slight changes are observed in the pattern list and the three metrics. This means that the information from the fifth project does not bring much contribution to the pattern extraction, consolidation and performance increase. We believe that the results can support the conclusion on the sufficient state of convergence.

In the remaining experiments, the set of patterns and their rankings obtained from the fifth iteration are finalized in iLead, and used to investigate other research questions.

B. Answering RQ2. Performance Evaluation

Table IV presents the performance of iLead applying to the five new projects. The results show that iLead has an average precision of 82%, and an average recall of 79% across the five projects. In addition, the results are quite close to

TABLE IV
PERFORMANCE ON UNFITTED PROJECTS (RQ2)

Projects	bokeh	efcore	knex	roslyn	solidity	AVG
Precision	0.80	0.83	0.82	0.82	0.84	0.82
Recall	0.77	0.84	0.77	0.81	0.78	0.79
F1-Score	0.77	0.83	0.79	0.81	0.80	0.80

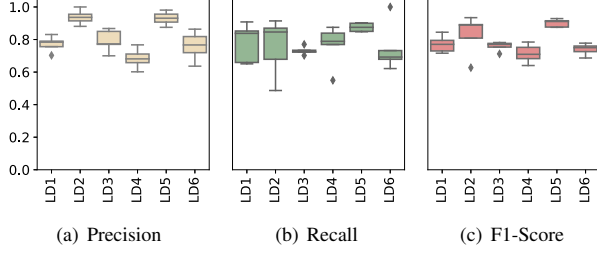


Fig. 5. Performance on six leadership categories (RQ2)

those from Table III, which indicates the performance stability of iLead, as well as the generality of the pattern set and their relative ranking in iLead. It is also noticeable that the performance is slightly higher than the inter-rater agreement level of 0.75 during the manual annotation process (Section IV-A). A possible reason is associated with the group consensus based on majority vote. More specifically, some inter-raters discrepancy involves long comments containing multiple sentences, in which two or more sentences inside a long issue comment may correspond to different leadership labels. We employed group discussion and majority vote to keep 1 dominant leadership label. This handling might lead to the performance of iLead slightly higher than the interrater's initial agreement level. Nevertheless, we observe a slight decrease in precision and recall, compared with the performance in Table III. One possible reason is that these five new projects might contain new patterns in expressing leadership behaviors, which necessitates further tuning or addition investigation to include more patterns to improve the performance.

We also examine the performance of iLead in identifying each leadership category on the five new projects, as shown in Figure 5. From the comparison, we can see that LD2 (Redirection) and LD5 (Operation) are among the categories with highest performance, meaning relatively easier to identify. These two categories both reach a median precision over 0.9, median recall over 0.8. The lowest performance is observed in LD4 (Inquiry) and LD6 (Volunteer), yet with a satisfactory median of 0.70 F1-Score.

C. Answering RQ3. Baseline Comparison

Figure 6 demonstrates the comparison of the performance of iLead with the ten baselines. We can see that iLead outperforms all the baselines in all the investigated metrics, e.g., the median F1-Score of iLead is 13% (i.e., $(0.80-0.71)/0.71$) higher than the best baseline *SVM-P*. Furthermore, we conduct Mann-Whitney U test between the performance of iLead and the baselines. Results show that iLead significantly (p-

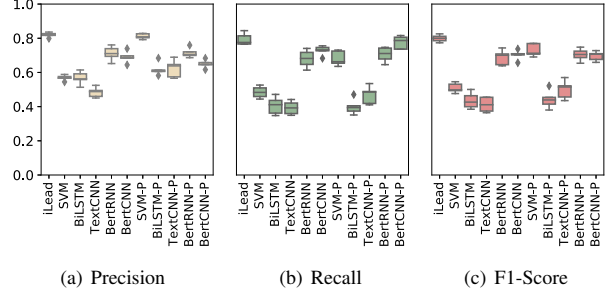


Fig. 6. Performance comparison with baselines (RQ3)

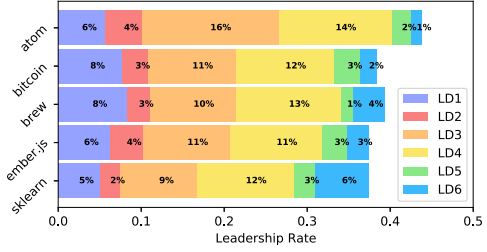


Fig. 7. Distribution of leadership

value <0.01) outperform all the baselines in F1-Score, and significantly outperform most of the baselines in precision (except *SVM-P*) and recall (except *BertCNN-P*). This implies the effectiveness of our extracted patterns as well as the pattern consolidation algorithm in iLead.

Among the baselines, the machine learning classifier with combined features, e.g., *SVM-P*, is better than its counterpart only with textual features. For example, the median F1-Score achieved by *SVM-P* is 42% (i.e., $(0.71-0.50)/0.50$) higher than its counterpart *SVM*. This implies our designed patterns are useful even when integrated into the machine learning models.

Nevertheless, they still underperform our proposed approach, because our proposed approach has well-designed mechanism to optimize the patterns and their rankings, while the machine learners consider less of the application order of the patterns. We also observe that the deep learning based approaches do not achieve a satisfactory result as anticipated. This might because training an efficient deep learning models requires a large amount of labelled data, yet the dataset utilized in our experiment is far from that amount. Besides, there are many low frequency words which distinguish a certain category of leadership behaviors from others. But the deep learning models are not good at capturing these words.

VII. REVISITING OSS LEADERSHIP LANDSCAPE

A. Distribution of Leadership Behaviors

Figure 7 illustrates the distribution of the identified emergent leadership behaviors on the five projects. It is observed that: (1) Overall, leadership-associated comments account for 39% of all comments on average (ranging from 37% to 44%); (2) The Top-3 dominant leadership categories are LD1 (Proposal),

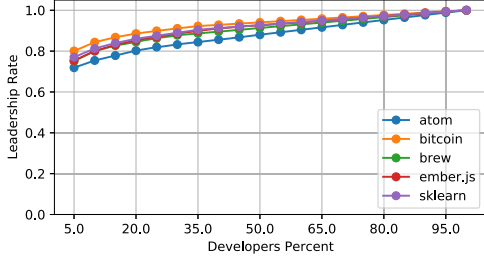


Fig. 8. Cumulative distribution of developers' leadership

LD3 (Confirmation), and LD4 (Inquiry), corresponding to an average of 7%, 11%, and 12% of all comments, respectively; and (3) The other three types, i.e., LD2 (Redirection), LD5 (Operation) and LD6 (Volunteer), are observed at less frequency, corresponding to an average of 3%, 2%, and 3% of all comments, respectively. These results provide quantitative evidences on developers leadership contribution in issue discussion, i.e., through brainstorming ideas, information inquiry, and consensus-based decision making.

Figure 8 illustrates the pareto curves of individual developers' leadership-related actions in issue discussion, across the five projects. Two outstanding results are noted: (1) On the one hand, an average of 11% (ranging from 5% to 20%) developers contribute to more than 80% of leadership behaviors; (2) On the other hand, the majority (i.e., 61%) developers (ranging from 40% to 75%) are associated with merely less than 10% of leadership behaviors. This implies the potential need for more research to study OSS leadership through this new lens, e.g., supporting OSS leadership evolution and individual/team leadership coaching.

B. Comparison among Leadership Indicators

Figure 9(a) shows the correlation between the number of code commits and the number of identified leadership behaviors across the six emergent leadership categories. In general, the correlation between the number of leadership behaviors and the number of commits is low. This again confirms that emergent leaders are not always the leading code contributors. We can observe a relatively higher correlation between these two indicators in the *brew* project. This may be because the *brew* project entrusts its daily management activities to a third-party service organization [63], and the developers focus more on the code contribution and issue discussion, i.e., resulting a higher correlation between these two indicators. Moreover, Figure 9(b) shows a much lower correlation between the number of followers and the number of leadership behaviors than that from Figure 9(a). This confirms one of frequent findings in existing leadership literature [64], [65], which refers that leaders are not depending on how much followers they have, but how many leaders they create. In OSS community, there are a variety of motivating factors for following, such as user's interest in learning, socializing, obtaining updates, or easy access to others, etc. [66].

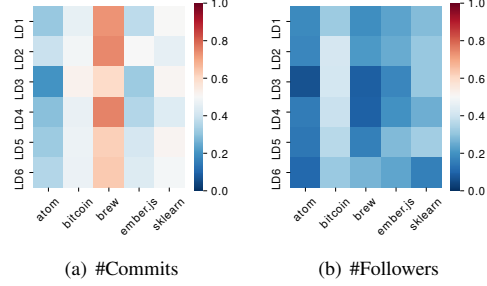


Fig. 9. Correlation between existing contribution indicators and #leadership

We think there are a couple of reasons for such low correlation. First, OSS developers may have different roles and preferences in different projects. For example, some prefer coding, while others participate more in the discussion. This suggests the need for a more dynamic and diverse lens to measure OSS leadership. Second, this reveals further opportunities for in-depth analysis to demonstrate the correlation between emergent leadership behaviors with existing leadership indicators, e.g., through data segmentation, time-lag analysis, etc. The low correlation between the identified emergent leadership behaviors and existing leadership indicators suggest that the identification of emergent leadership helps to broaden existing code contribution and followers viewpoints to embrace and recognize more comprehensive perspectives.

C. Human Evaluation

To evaluate the usefulness of iLead, we recruited 9 developers from three OSS communities (i.e., *bitcoin*, *atom*, *ember.js*) as evaluator. Specifically: (1) use iLead to identify leadership behaviors in these three projects, and then derive a ranked list of developers according to the frequency of leadership behaviors; (2) create two subsets of developers: one with the Top 20 developers (used as study group), and the other with developers ranking 21-40 (used as comparison group); (3) randomly select 10 developers from each group, and ask the human evaluators (from the same OSS project) to vote (Yes or No) for the 20 randomly selected developers, based on whether they believe the names on the list play important role(s) in driving issue resolution progress. Note that the developers are provided in random order (without ranking information) to the human evaluators to avoid introducing potential bias.

The results show that the random developers from the first subset (i.e., Top-20) account for 74% of overall votes, while those from the second subset receives 26% of overall votes. This indicates that, human evaluators' votes are mostly consistent with the leadership status identified by iLead.

VIII. DISCUSSION

A. Practical Insights for OSS Communities

Our findings suggest that leadership in OSS projects should encompass the key facets of emergent leadership behaviors in issue discussion. Based on the results from this study, we provide the following practical insights for OSS community.

First, while developers with large code submissions are naturally considered as leaders/core contributors to the project, OSS projects may not necessarily pay attention to emergent leadership of other developers, who serve important influential roles in guiding/facilitating issue resolution. In addition to recognize project leaders based on their code contribution, OSS projects should also consider developers emergent leadership behaviors in their interpersonal communications, e.g. issue comments and live chat.

Second, with support from iLead, OSS community can automate the following tasks accordingly: (1) to identify developers with the most influential emergent leadership behaviors according to the LD1 (Proposal) category; (2) to identify developers coordinating the most dominant brainstorming and exploration tasks according to the LD3 (Confirmation) and LD4 (Inquiry) categories; and (3) to identify developers facilitating the management and operation tasks such as redirecting a conversation, issue closing and volunteering, according to LD2 (Redirection), LD5 (Operation) and LD6 (Volunteer). With these knowledge, there is also great potential to appropriately automate and streamline such issue discussion and operational tasks, in order to accelerate issue resolution cycles.

B. Practical Insights for OSS Developers

We also observe several findings that may shed light to practices related to leadership skill development. First, developers will benefit from learning about different roles of emergent leadership. To emerge and to be accepted as a leader in an OSS project, a developer needs to know how to communicate effectively with others via textual messages. As the core construct of iLead, the linguistic pattern set embodies strong verbal-cues correlated with the emergent leadership. These can be used to coach newcomer developers in effectively inquiring about an issue, collaboratively confirming a candidate solution, or clearly proposing an alternative option, etc.

Second, while intuitively, it may seem that OSS leaders should play greater roles in LD1 (Proposal), since it is the most influential leadership behaviors, we actually observe something slightly different. We observe no much difference between junior and senior OSS developers in three types of leadership behaviors, i.e., LD1 (Proposal), LD4(Inquiry), and LD6(Volunteer). In addition, junior developers tend to conduct more LD3 (Confirmation), and less LD2 (Redirection) and LD5 (Operation); and senior developers conduct more LD2 (Redirection) and LD5 (Operation), and less LD3 (Confirmation). It is easy to understand that LD2 and LD5 rely on comprehensive knowledge about a specific OSS projects. Therefore, we encourage new or in-experienced developers to focus more on LD1, LD3, LD4, and LD6 when looking for opportunities to make contributions to a new OSS project.

C. Threats to Validity

The first threat concerns the generality of the proposed approach. The iLead is only evaluated using 10 popular OSS projects, which might not be representative of other projects.

However, the projects are from various domains, and the performance convergence experiment is conducted to ensure the comprehensiveness and effectiveness of the finalized patterns.

The second threat comes from the representativeness of the six categories of emergent leadership behaviors. One co-author's area of research is in leadership, and the six leadership categories are carefully identified based on reviewing and examining both traditional leadership styles and emergent leadership studies for virtual teams. In addition, the leadership survey results also show that this provides a meaningful representation of emergent leadership behaviors in OSS issue discussion communications.

The third threat is associated with the results of manual labelling. We follow an iterative process [40], and conduct the labelling by three authors independently, and common consensus are reached after several rounds of discussion.

The last threat is associated with the human evaluation since the number of participants is small. We cannot guarantee that the votes from the human evaluators are fair. To mitigate this threat, we ask the evaluators to vote on 10 randomly selected developers from two subsets, and use the total votes for comparison, to mitigate individual bias.

IX. RELATED WORK

The leadership behaviors have long been investigated in business and management domain, and have attracted increasing attention in software engineering domain in recent years. Giuri et al. [67] characterized the difference in individual and project-specific characteristics between project leaders and other project members. Bian et al. [68] explored the correlation between leader characteristics and success of OSS projects. Li et al. [3] investigated the relationship between leadership characteristics and the motivation to contribute. Jensen et al. [69] explored the collaborative efforts, the leadership and control structures in OSS community. Tan et al. and Zhang et al. [70]–[72] investigated how developers and companies collaborate and communicate with each other. Daniel et al [73]. analyze the emails to understand the communication patterns in mature community. By comparison, this paper proposes an automatic approach to identify leadership behaviors in OSS projects, with which one can conduct the leadership related investigation to facilitate the OSS development.

X. CONCLUSION

This paper explores an innovative perspective to look at OSS leadership, and proposes an automated approach, iLead, to mine communication styles and identify emergent leadership behaviors in OSS projects using issue comments data. The evaluation results demonstrate the effectiveness of the lightweight heuristic rule-based approach employed in iLead. In the future, we will further investigate the relationship between leadership and the OSS community.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China Grant No.62232016, No.62072442 and No.62272445.

REFERENCES

- [1] A. Hars and S. Ou, "Working for free? motivations for participating in open-source projects," *Int. J. Electron. Commer.*, vol. 6, no. 3, pp. 25–39, 2002. [Online]. Available: <https://doi.org/10.1080/10864415.2002.11044241>
- [2] H. Baytiyeh and J. Pfaffman, "Volunteers in wikipedia: Why the community matters," *J. Educ. Technol. Soc.*, vol. 13, no. 2, pp. 128–140, 2010. [Online]. Available: http://www.ifets.info/abstract.php?art_id=1046
- [3] Y. Li, C.-H. Tan, and H.-H. Teo, "Leadership characteristics and developers' motivation in open source software development," *Information & Management*, vol. 49, no. 5, pp. 257–267, 2012.
- [4] M. Zhou and A. Mockus, "What make long term contributors: Willingness and opportunity in OSS community," in *34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland*, M. Glinz, G. C. Murphy, and M. Pezzè, Eds. IEEE Computer Society, 2012, pp. 518–528. [Online]. Available: <https://doi.org/10.1109/ICSE.2012.6227164>
- [5] C. Jensen, S. King, and V. Kuechler, "Joining free/open source software communities: An analysis of newbies' first interactions on project mailing lists," in *44th Hawaii International Conference on Systems Science (HICSS-44 2011), Proceedings, 4-7 January 2011, Koloa, Kauai, HI, USA*. IEEE Computer Society, 2011, pp. 1–10. [Online]. Available: <https://doi.org/10.1109/HICSS.2011.264>
- [6] I. Steinmacher, I. S. Wiese, A. P. Chaves, and M. A. Gerosa, "Newcomers withdrawal in open source software projects: Analysis of hadoop common project," in *2012 Brazilian Symposium on Collaborative Systems, Sao Paulo, Brazil, October 15-18, 2012*. IEEE Computer Society, 2012, pp. 65–74. [Online]. Available: <https://doi.org/10.1109/SBSC.2012.16>
- [7] A. Mockus, R. T. Fielding, and J. Herbsleb, "A case study of open source software development: the apache server," in *Proceedings of the 22nd international conference on Software engineering*, 2000, pp. 263–272.
- [8] X. Ben, S. Beijun, and Y. Weicheng, "Mining developer contribution in open source software using visualization techniques," in *2013 Third International Conference on Intelligent System Design and Engineering Applications*. IEEE, 2013, pp. 934–937.
- [9] D. A. da Costa, U. Kulesza, E. Aranha, and R. Coelho, "Unveiling developers contributions behind code commits: an exploratory study," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, 2014, pp. 1152–1157.
- [10] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi, "Mining eclipse developer contributions via author-topic models," in *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)*. IEEE, 2007, pp. 30–30.
- [11] M. Moqri, X. Mei, L. Qiu, and S. Bandyopadhyay, "Effect of "following" on contributions to open source communities," *Journal of Management Information Systems*, vol. 35, no. 4, pp. 1188–1217, 2018.
- [12] J. A. Roberts, I.-H. Hann, and S. A. Slaughter, "Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the apache projects," *Management science*, vol. 52, no. 7, pp. 984–999, 2006.
- [13] R. Eckert, M. Stuermer, and T. Myrach, "Alone or together? inter-organizational affiliations of open source communities," *Journal of systems and software*, vol. 149, pp. 250–262, 2019.
- [14] B. M. Bass and B. J. Avolio, "Multifactor leadership questionnaire," *Western Journal of Nursing Research*, 1996.
- [15] H. Zhu, R. Kraut, and A. Kittur, "Effectiveness of shared leadership in online communities," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, 2012, pp. 407–416.
- [16] T. A. Carte, L. Chidambaram, and A. Becker, "Emergent leadership in self-managed virtual teams," *Group Decision and Negotiation*, vol. 15, no. 4, pp. 323–343, 2006.
- [17] G. Yukl, A. Gordon, and T. Taber, "A hierarchical taxonomy of leadership behavior: Integrating a half century of behavior research," *Journal of leadership & organizational studies*, vol. 9, no. 1, pp. 15–32, 2002.
- [18] G. Yukl, "Effective leadership behavior: What we know and what questions need more attention," *Academy of Management perspectives*, vol. 26, no. 4, pp. 66–85, 2012.
- [19] A. R. da Silva, "Linguistic patterns and linguistic styles for requirements specification (I): an application case with the rigorous rsl/business-level language," in *Proceedings of the 22nd European Conference on Pattern Languages of Programs, EuroPLOP 2017, Irsee, Germany, July 12-16, 2017*. ACM, 2017, pp. 22:1–22:27. [Online]. Available: <https://doi.org/10.1145/3147704.3147728>
- [20] B. Lin, F. Zampetti, G. Bavota, M. D. Penta, and M. Lanza, "Pattern-based mining of opinions in q&a websites," in *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, J. M. Atlee, T. Bultan, and J. Whittle, Eds. IEEE / ACM, 2019, pp. 548–559. [Online]. Available: <https://doi.org/10.1109/ICSE.2019.00066>
- [21] L. Shi, C. Chen, Q. Wang, S. Li, and B. W. Boehm, "Understanding feature requests by leveraging fuzzy method and linguistic analysis," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, Urbana, IL, USA, October 30 - November 03, 2017*, G. Rosu, M. D. Penta, and T. N. Nguyen, Eds. IEEE Computer Society, 2017, pp. 440–450. [Online]. Available: <https://doi.org/10.1109/ASE.2017.8115656>
- [22] P. G. Northouse, *Leadership: Theory and practice*. Sage publications, 2021.
- [23] V. J. Vanderslice, "Separating leadership from leaders: An assessment of the effect of leader and follower roles in organizations," *Human relations*, vol. 41, no. 9, pp. 677–696, 1988.
- [24] G. Yukl, *Leadership in Organizations, 9/e*. Pearson Education India, 1981.
- [25] B. M. Bass, B. J. Avolio, D. I. Jung, and Y. Berson, "Predicting unit performance by assessing transformational and transactional leadership," *Journal of applied psychology*, vol. 88, no. 2, p. 207, 2003.
- [26] J. M. Burns *et al.*, "Leadership," 1978.
- [27] E. P. Hollander, "Leadership dynamics: A transactional perspective." STATE UNIV OF NEW YORK AT BUFFALO DEPT OF PSYCHOLOGY, Tech. Rep., 1978.
- [28] J. B. Carson, P. E. Tesluk, and J. A. Marrone, "Shared leadership in teams: An investigation of antecedent conditions and performance," *Academy of management Journal*, vol. 50, no. 5, pp. 1217–1234, 2007.
- [29] Y. Yoo and M. Alavi, "Emergent leadership in virtual teams: what do emergent leaders do?" *Information and organization*, vol. 14, no. 1, pp. 27–58, 2004.
- [30] C. L. Pearce and J. A. Conger, *Shared leadership: Reframing the hows and whys of leadership*. Sage Publications, 2002.
- [31] C. Jergensen, A. Sarma, and P. Wagstrom, "The onion patch: migration in open source ecosystems," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, 2011, pp. 70–80.
- [32] K. Nakakoji, Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye, "Evolution patterns of open-source software systems and communities," in *Proceedings of the international workshop on Principles of software evolution*, 2002, pp. 76–85.
- [33] GitHub, "Github: Viewing a project's contributors," <https://docs.github.com/en/github/visualizing-repository-data-with-graphs/accessing-basic-repository-data/viewing-a-projects-contributors>, 2022.
- [34] A. Lee and J. C. Carver, "Are one-time contributors different? A comparison to core and periphery developers in FLOSS repositories," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2017, Toronto, ON, Canada, November 9-10, 2017*, A. Bener, B. Turhan, and S. Biffl, Eds. IEEE Computer Society, 2017, pp. 1–10. [Online]. Available: <https://doi.org/10.1109/ESEM.2017.7>
- [35] S. Ferebee and J. Davis, "Emergent leadership, persuasion, and trust in virtual leaderless groups," *The Exchange*, vol. 1, no. 1, 2012.
- [36] D. J. Neufeld and H. Gu, "Leadership emergence and impact on open source software project success: A comparative case study," in *Academy of Management Proceedings*, vol. 2019, no. 1. Academy of Management Briarcliff Manor, NY 10510, 2019, p. 11698.
- [37] T. Winograd, F. Flores, and F. F. Flores, *Understanding computers and cognition: A new foundation for design*. Intellect Books, 1986.
- [38] Y. Zhao, L. Xiao, P. Babvey, L. Sun, S. Wong, A. A. Martinez, and X. Wang, "Automatically identifying performance issue reports with heuristic linguistic patterns," in *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, P. Devanbu, M. B. Cohen, and T. Zimmermann, Eds. ACM, 2020, pp. 964–975. [Online]. Available: <https://doi.org/10.1145/3368089.3409674>

- [39] S. Panichella, A. D. Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in *2015 IEEE International Conference on Software Maintenance and Evolution, ICSME 2015, Bremen, Germany, September 29 - October 1, 2015*, R. Koschke, J. Krinke, and M. P. Robillard, Eds. IEEE Computer Society, 2015, pp. 281–290. [Online]. Available: <https://doi.org/10.1109/ICSM.2015.7332474>
- [40] J. Pustejovsky and A. Stubbs, *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*. "O'Reilly Media, Inc.", 2012.
- [41] G. Rugg and P. McGeorge, "The sorting techniques: a tutorial paper on card sorts, picture sorts and item sorts," *Expert Syst. J. Knowl. Eng.*, vol. 22, no. 3, pp. 94–107, 2005. [Online]. Available: <https://doi.org/10.1111/j.1468-0394.2005.00300.x>
- [42] iLead, "iLead," <https://github.com/iLeadGroup/iLead>, 2022.
- [43] D. Shen, J. Sun, Q. Yang, and Z. Chen, "Text classification improved through multigram models," in *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006*, P. S. Yu, V. J. Tsotras, E. A. Fox, and B. Liu, Eds. ACM, 2006, pp. 672–681. [Online]. Available: <https://doi.org/10.1145/1183614.1183710>
- [44] bitcoin, "Bitcoin," <https://github.com/bitcoin/bitcoin>, 2022.
- [45] scikit learn, "Scikit-learn," <https://github.com/scikit-learn/scikit-learn>, 2022.
- [46] emberjs, "Ember.js," <https://github.com/emberjs/ember.js>, 2022.
- [47] Homebrew, "Brew," <https://github.com/Homebrew/brew>, 2022.
- [48] atom, "Atom," <https://github.com/atom/atom>, 2022.
- [49] bokeh, "Bokeh," <https://github.com/bokeh/bokeh>, 2022.
- [50] dotnet, "Efc core," <https://github.com/dotnet/efcore>, 2022.
- [51] knex, "Knex," <https://github.com/knex/knex>, 2022.
- [52] dotnet, "Roslyn," <https://github.com/dotnet/roslyn>, 2022.
- [53] ethereum, "Solidity," <https://github.com/ethereum/solidity>, 2022.
- [54] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A Python natural language processing toolkit for many human languages," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020.
- [55] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008. [Online]. Available: <https://nlp.stanford.edu/IR-book/pdf/irbookprint.pdf>
- [56] J. Berkson, "Application of the logistic function to bio-assay," *Journal of the American statistical association*, vol. 39, no. 227, pp. 357–365, 1944.
- [57] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [58] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth, 1984.
- [59] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.
- [60] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, A. Moschitti, B. Pang, and W. Daelemans, Eds. ACL, 2014, pp. 1746–1751. [Online]. Available: <https://doi.org/10.3115/v1/d14-1181>
- [61] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, S. Kambhampati, Ed. IJCAI/AAAI Press, 2016, pp. 2873–2879. [Online]. Available: <http://www.ijcai.org/Abstract/16/408>
- [62] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186. [Online]. Available: <https://doi.org/10.18653/v1/n19-1423>
- [63] sfconservancy, "Software freedom conservancy: Member project services," <https://sfconservancy.org/projects/services/>, 2022.
- [64] S. Kean, E. HAYCOCK-STUART, S. Baggaley, and M. Carson, "Followers and the co-construction of leadership," *Journal of Nursing Management*, vol. 19, no. 4, pp. 507–516, 2011.
- [65] R. E. Kelley, *In praise of followers*. Harvard Business Review Case Services, 1988.
- [66] K. Blincoe, J. Sheoran, S. Goggins, E. Petakovic, and D. Damian, "Understanding the popular users: Following, affiliation influence and leadership on github," *Information and Software Technology*, vol. 70, pp. 30–39, 2016.
- [67] P. Giuri, F. Rullani, and S. Torrisi, "Explaining leadership in virtual teams: The case of open source software," *Inf. Econ. Policy*, vol. 20, no. 4, pp. 305–315, 2008.
- [68] Y. Bian, W. Mu, and J. L. Zhao, "Online leadership for open source project success: Evidence from the github blockchain projects," in *22nd Pacific Asia Conference on Information Systems, PACIS 2018, Yokohama, Japan, June 26-30, 2018*, 2018, p. 189.
- [69] C. Jensen and W. Scacchi, "Collaboration, leadership, control, and conflict negotiation and the netbeans.org open source software development community," in *38th Hawaii International Conference on System Sciences (HICSS-38 2005), CD-ROM / Abstracts Proceedings, 3-6 January 2005, Big Island, HI, USA, 2005*.
- [70] X. Tan and M. Zhou, "How to communicate when submitting patches: An empirical study of the linux kernel," *Proc. ACM Hum. Comput. Interact.*, vol. 3, no. CSCW, pp. 108:1–108:26, 2019.
- [71] X. Tan, M. Zhou, and B. Fitzgerald, "Scaling open source communities: an empirical study of the linux kernel," in *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, 2020, pp. 1222–1234. [Online]. Available: <https://doi.org/10.1145/3377811.3380920>
- [72] Y. Zhang, M. Zhou, K. Stol, J. Wu, and Z. Jin, "How do companies collaborate in open source ecosystems?: an empirical study of openstack," in *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, 2020, pp. 1196–1208.
- [73] D. Schneider, S. Spurlock, and M. Squire, "Differentiating communication styles of leaders on the linux kernel mailing list," in *Proceedings of the 12th International Symposium on Open Collaboration, OpenSym 2016, Berlin, Germany, August 17-19, 2016*, A. I. Wasserman, Ed. ACM, 2016, pp. 2:1–2:10. [Online]. Available: <https://doi.org/10.1145/2957792.2957801>