CHI ’20, April 25–30, 2020, Honolulu, HI, USA.
http://dx.doi.org/10.1145/3313831.3376218

ArguLens: Anatomy of Community Opinions On Usability Issues Using Argumentation Models

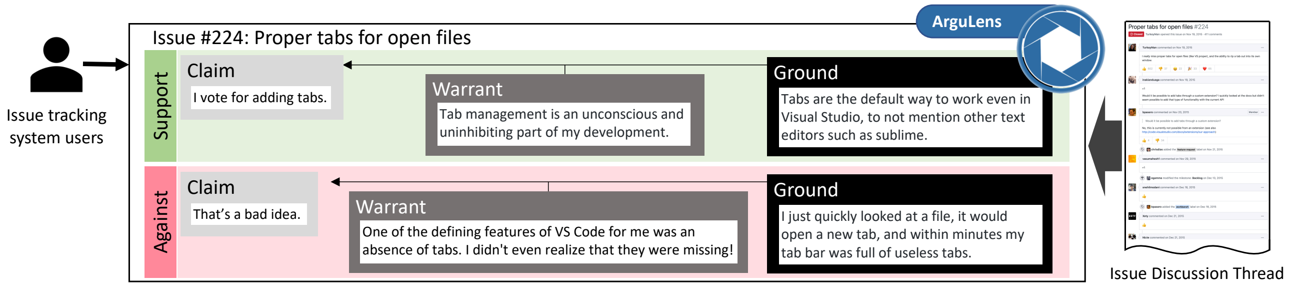
Wenting Wang
McGill University
Montréal, Canada
wenting.wang@mail.mcgill.ca

Deeksha Arya
McGill University
Montréal, Canada
deeksha.arya@mail.mcgill.ca

Nicole Novielli
University of Bari
Bari, Italy
nicole.novielli@uniba.it

Jinghui Cheng
Polytechnique Montréal
Montréal, Canada
jinghui.cheng@polymtl.ca

Jin L.C. Guo
McGill University
Montréal, Canada
jguo@cs.mcgill.ca


Issue tracking system users
ArguLens
Issue #224: Proper tabs for open files
Support
Against
Claim
Warrant
Ground
Issue Discussion Thread

Claim: I vote for adding tabs.

Warrant: Tab management is an unconscious and uninhibiting part of my development.

Ground: Tabs are the default way to work even in Visual Studio, to not mention other text editors such as sublime.

Claim: That’s a bad idea.

Warrant: One of the defining features of VS Code for me was an absence of tabs. I didn't even realize that they were missing!

Ground: I just quickly looked at a file, it would open a new tab, and within minutes my tab bar was full of useless tabs.

Figure 1. ArgueLens analyzes lengthy discussions about usability on issue tracking systems for open source projects, anatomizing them into argumentative components (claim, ground, and warrant) and standpoints (support or against). Such a conceptual framework and the underlying argument extraction technique enable tools for supporting open source community members to understand and consolidate diverse opinions on usability issues.

ABSTRACT

In open-source software (OSS), the design of usability is often influenced by the discussions among community members on platforms such as issue tracking systems (ITSs). However, digesting the rich information embedded in issue discussions can be a major challenge due to the vast number and diversity of the comments. We propose and evaluate ArguLens, a conceptual framework and automated technique leveraging an argumentation model to support effective understanding and consolidation of community opinions in ITSs. Through content analysis, we anatomized highly discussed usability issues from a large, active OSS project, into their argumentation components and standpoints. We then experimented with supervised machine learning techniques for automated argument extraction. Finally, through a study with experienced ITS users, we show that the information provided by ArguLens supported the digestion of usability-related opinions and facilitated the review of lengthy issues. ArguLens provides the direction of designing valuable tools for high-level reasoning and effective discussion about usability.

Author Keywords

Open source software; usability; online communities; issue discussion analysis; argumentation analysis.

CCS Concepts

•**Software and its engineering** → **Software usability;** •**Human-centered computing** → *Open source software; Computer supported cooperative work;*

INTRODUCTION

Usability is a well-known, yet important concept that focuses on creating a system that is easy, efficient, error-preventing, and pleasant to be used [44]. In community-driven software development environments, such as that of the modern Open Source Software (OSS) applications, usability considerations are usually determined by the collective concerns of the community itself [3], which, in turn, comprises a heterogeneous group of participants [18].

In the recent years, OSS applications vastly broadened their scope, going beyond the tools for “technology adventurers” while embracing a wide range of applications for diverse tasks

ACM ISBN 978-1-4503-6708-0/20/04 ...$15.00.
http://dx.doi.org/10.1145/3313831.3376218

and user groups. OSS development has since evolved beyond the "scratch-your-own-itch" model, to focus on thethesatisfaction and involvement of end users [58]. Collection of the community feedback in OSS development is largely done by leveraging modern software engineering tools such as *Issue tracking systems* (ITSs). ITSs allow OSS community members to create, discuss about, and manage the status of various system and project-related tasks, enhancements, problems, and questions, including those related to usability [64]. Previous research demonstrated how the discussions on ITSs provide rich information to diverse community participants [4]. Particularly, usability issues are important venues to capture and record discussions about user experiences, opinions, desires, and justifications [17]. However, because of the sheer amount of comments posted daily to ITSs, as well as the varied perspectives of different community members, contributors of OSS application projects face a major challenge in digesting the rich information embedded in ITSs in order to determine the actual user needs and consolidate the diverse feedback [10].

In this paper, we directly target this challenge through ArguLens (see Figure 1), a conceptual framework and machine learning-based technique leveraging an argumentation model to extract structured information from complex usability discussions. We formulate our research questions as follows:

RQ1: How do the open source software communities argue about usability issues in issue tracking systems?

RQ2: How effective are machine learning models in extracting arguments and their structure in usability issue discussions?

RQ3: To what extent can argumentation-enhanced representations of usability issue discussions support practitioners in understanding and consolidating community opinions and needs?

To answer our research questions and construct ArguLens, we first conducted a content analysis on arguments in usability issue discussion threads of the Visual Studio Code GitHub project[1], using an adapted version of the Toulmin's argumentation model [59]. Then we experimented with multiple supervised classifiers using different feature sets to extract argumentative information from the discussions. Finally, we performed a user study with participants who have experiences in using issue tracking systems to understand their perception of the argumentation-annotated usability issue discussion threads.

Our work produced the following outcomes and contributions. First, our content analysis of usability issue discussions resulted in an annotated corpus containing 5123 quotes (i.e. sentences or self-contained phrases) that serves as the foundation for ArguLens. Results indicated that contents in usability discussions broaden beyond the original post topic. In addition, we found that highly-disagreed arguments usually reside in the middle of the issue thread and thus can be easily overlooked. Second, we demonstrate that we can train supervised classifiers for automated argument identification, thus enabling argumentation-enhanced issue representation in ArguLens. Specifically, we observed that both Linear Support Vector Machine (SVM) and Naive Bayes classifiers provided satisfactory results using textual features based on term frequency-inverse document frequency (TF-IDF). Finally, our user study provided evidence that practitioners preferred representations of issue discussions supported by ArguLens. Removing non-argumentative contents, separating arguments by their topics, and providing information about argument components were appreciated by the participants when they focused on understanding the community's opinions in issue discussions. Our results also called for the needs of designing visual and interactive representations of the argument anatomy supported by ArguLens. In sum, ArguLens provides the direction of designing valuable tools and techniques for reasoning and discussing usability issues in a community-driven environment.

BACKGROUND AND RELATED WORK

Our work is situated in the literature on (1) usability of open source projects, (2) argumentation theory and their applications to usability, and (3) automated argument mining.

Open Source Usability

Due to the distinctive characteristics of OSS projects, addressing their usability is particularly challenging. The usability of OSS applications is known to be negatively affected by factors such as developers' scarce awareness of users' needs, insufficient expertise, and an excessive emphasis on features and complexity rather than ease-of-use in OSS projects [42]. Along with the recent development of software engineering techniques and tools, as well as the increased awareness of usability in the software industry, OSS usability had become a growing research direction carrying important practical implications. The *Issue Tracking Systems* (ITS) represent a common and fundamental tool for gathering the community members' needs and feedback about usability-related issues, as identified in several empirical studies about OSS usability practices [6, 43, 33]. The OSS community members (including users, developers, and other stakeholders) use ITSs to collaboratively raise, discuss, negotiate, and address usability-related topics.

Although ITSs are useful in providing a common platform to engage the community, they suffer from several limitations when used for discussing usability issues in practice. For example, there is currently limited theoretical and practical support in classifying usability defects and handling the multifaceted usability discussions [60, 64]. Current tools also do not accommodate the different communication styles adopted in diverse OSS communities [17]. Moreover, community members engaged in ITS usability discussions usually experience an information overload, which hinders their effective participation and contribution [10]. Because of these limitations, getting involved in OSS usability discussions is still a daunting affair, deterring OSS developers as well as other crucial stakeholders such as end users and user experience experts [7, 48, 34]. In this paper, we tackle this challenging problem of supporting OSS usability through the creation and evaluation of ArguLens to support OSS stakeholders in understanding and consolidating the diverse community opinions.

Argumentation Theory and Its Application to Usability

The study of argumentation has a long history due to the prevalent importance of persuasive speaking [32]. Many researchers studied the structure of arguments, thus producing rhetoric and

[1] https://github.com/microsoft/vscode

argumentation theory [59, 26]. Others applied these models to assist tasks in areas such as knowledge representation, legal reasoning, and negotiation [5, 12, 13, 21, 40].

Macewan et al's defined the term "argumentation" to emphasize its progressive property: "argumentation is the process of proving or disproving a proposition. Its purpose is to induce a new belief, to establish truth or combat error in the mind of another" [39]. We adopt this definition of *argumentation* and additionally use the term *argument* to refer to a concrete instance (e.g. a paragraph or a series of persuasive statements on a certain topic) produced via argumentation.

Collectively, researchers have proposed many argumentation models for different purposes [11]. In order to understand how each argument towards usability unfolds in issue discussions, we adopt one of the most influential models, i.e., Toulmin's model of argumentation. Toulmin describes arguments using the metaphor of an organism, suggesting that an argument should have both a gross, "anatomical" structure and a finer, "physiological" one [59]. To study the pattern of arguments, he identified six fundamental argumentative components:

Claim: Conclusions or viewpoints one tries to convince others to agree with.
Ground: Facts used to play as a foundation for the claim.
Warrant: General or hypothetical statements, acting as bridges between data and claim.
Qualifier: Indicating the strength granted by the warrant to support its claim.
Rebuttal: Conditions in which the warrant is not applicable. As a result of a rebuttal, the consequential conclusion can be overturned.
Backing: Demonstrating the validity of warrant.

Toulmin further incorporated their inter-relationships to form the layout of arguments, as shown in Figure 2.

Researchers have adopted Toulmin's argumentation theory for usability evaluations. For example, Nogaard et al. proposed that usability feedback can be viewed as an argument for a series of usability problems [45]. By applying concepts from Toulmin's model and Aristotle's modes of persuasion, they provided guidelines about how to create persuasive usability feedbacks. Usability problems and solutions are also typically consolidated with a group of stakeholders involved in the design process [38]. A study with ten novice usability evaluators illustrated that their ability to defend their arguments during negotiation impacts feature/fix prioritization, which may further affect the design decisions [38]. Therefore, it is important to analyze argumentation in usability-related discussions. Our study contributes to this body of literature by adopting and adjusting Toulmin's model of argumentation to examine arguments contained in OSS usability issue discussion threads.

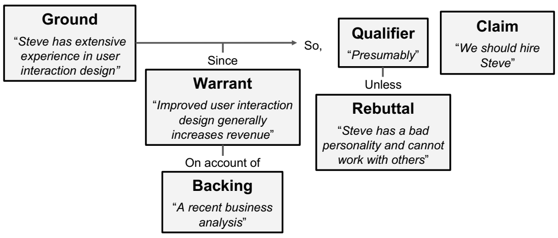

Ground
"Steve has extensive experience in user interaction design"
Since
So,
Warrant
"Improved user interaction design generally increases revenue"
On account of
Backing
"A recent business analysis"
Qualifier
"Presumably"
Unless
Rebuttal
"Steve has a bad personality and cannot work with others"
Claim
"We should hire Steve"

Figure 2. Relationship among the six components of the Toulmin's model of argumentation [59] with examples.

Argument Mining

Argument mining is the task of identifying argumentative contents and components in natural language texts [52]. Detecting argument components can help the readers consolidate ideas and support the argumentation authors to create a persuasive reasoning [45]. Arguments are extensively studied in formal written communications, such as legal documents and scientific publications [41, 24, 55, 56]. Sentences in these documents are usually well-structured, which can facilitate the task of argument extraction. Comparing to formal sources, arguments taking place on online platforms are often less structured, vague, implicit, or simply poorly worded [14]. Several recent studies have focused on addressing these significant challenges in user-generated online texts. For example, Abbott et al. identified that contextual and dialogic features help in recognizing disagreement in informal political arguments [1]. Florou et al. experimented using verbal tense and mood as features for the task of argument extraction of Web contents [25]. Some researchers also worked on argument extraction from news, blogs and social media [30, 52].

Recently, Habernal et al. adapted Toulmin's argumentation model to Web discourse. Using an annotated dataset of 340 documents, they investigated the suitability of machine learning in automated identification of argument components. Their findings suggested that argumentation mining in user-generated Web discourse is a challenging but feasible task [31]. In this paper, we investigate the effectiveness of supervised machine learning techniques to detect arguments, along with argumentative components, in usability issue discussions.

ARGUMENTATION ANALYSIS

To build the foundation for ArguLens and answer RQ1 (*How do the OSS communities argue about usability?*), we conducted a content analysis based on an argumentation model.

Methods

We carried out three main steps for argumentation analysis: (1) selecting OSS projects and issues, (2) adapting Toulmin's argumentation model, and (3) conducting the content analysis.

Project and Issue Selection

We focused our study on analyzing issue thread discussions for the Visual Studio Code[2] project (VS Code), a source-code editor supported by Microsoft, published under the permissive MIT license. The reasons we selected this project are as follows. First, the application has a sophisticated graphical user interface which allows our study to focus on usability for end users. Second, the project is under active development and involves a large, heterogeneous community. Since the creation of the repository, an average of 353 issues are created each week. There are 906 recognized contributors, who made wide-ranging contributions including developing features, discussing issues, fixing bugs, conducting code reviews,

[2] https://github.com/microsoft/vscode

and updating project documentation and websites. Lastly, the authors of this study are all experienced using this application. This ensures us to conduct a comprehensive and accurate analysis of the issue discussions in the project.

The issue selection was conducted in July 2018. Using GitHub REST API [29], we first ordered the closed issue threads by the number of comments. We then examined the title and the issue description for usability-relevance until five issues were selected. We only included closed issues in the analysis because they allow us to understand the complete flow of discussion until the issue is resolved. Table 1 summarizes information about each issue we gathered.

IssueID	Issue Title	# of Comments
224	Proper tabs for open files	411
396	Add support for opening multiple project folders in same window	380
4865	Enhanced Scrollbar (add minimap)	105
9388	Provide a setting so that only double click opens a file in the editor or expands a folder	87
14909	Support a grid layout for editors	191

Table 1. Information about issues in our corpus.

Argumentation Model Adaption

In this study, we lay emphasis on examining the internal structure (i.e. anatomy) of usability-related arguments. While we acknowledge that some recent studies on argument mining (e.g. [22, 56]) apply a simple claim-premise model, we decided to adopt a full model of argumentation (particularly Toulmin's model [59], described in related work). Our choice is grounded on our extensive experience working with OSS projects. Indeed, we believe that it is crucial for OSS contributors to identify (1) facts about the system (which can be captured by "ground" in the Toulmin model), and (2) general opinions about why a fact is relevant (corresponding to "warrant"). While simply distinguishing between 'claim' and 'premise' may result in a better accuracy in automated classification, we believe that such distinction is of limited usefulness in supporting the practical needs of OSS contributors that aim at understanding and prioritizing usability issues.

Toulmin's model focuses on documents containing well-formed explicit arguments. To adapt it to the informal arguments made in usability issue discussions, we started by directly using Toulmin's model to code two issues. Based on our observations, we made the following adjustments to better reflect the characteristics of ITS discussions.

First, in our initial analysis, we found that the boundary of *Warrant* and *Backing* is extremely difficult to draw in issue discussions. To avoid ambiguity, we merged those two components and annotated any statements acting as the bridge between user requests and concrete facts with *Warrant*. Second, to adapt to the short, informal style of arguments in issue discussions, we removed *Rebuttal* as an argument component and instead identified the standpoints of any comment in the discussion as *Support* or *Against* with respect to the original claim. Finally, similar to Habernal et al. [31], we excluded *Qualifier* because of its low representation in our issue discussion data. These changes resulted in the *Components* coding schema used in the content analysis (see Figure 3).

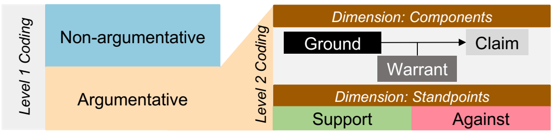


Figure 3. Summary of the two levels of codes (including those in the two dimensions in the Level 2 coding) used in the content analysis process.

Content Analysis

We performed the content analysis iteratively on two levels as illustrated in Figure 3. Upon data collection, we split the comments in each issue into individual sentences. In Level 1 coding, we coded each sentence as either *argumentative*, which denotes that the sentence had a persuasive connotation, or *non-argumentative*, which include all non-persuasive sentences. In Level 2 coding, we identified the characteristics of all the argumentative sentences based on two independent dimensions: (1) the argument *components* based on the modified Toulmin's model (i.e. *Claim*, *Warrant*, or *Ground*) and (2) the *standpoint* of the author (i.e. *Support* or *Against*). During the coding process, if one sentence contains multiple argumentation components (e.g. claim + warrant), we manually split the sentence into segments to assign each segment a single, unambiguous code. We refer these sentences and self-contained segments as *quotes*. Thus, all argumentative quotes were annotated using exactly one code from each of the aforementioned dimensions.

Furthermore, because an extended issue discussion usually contains multiple topics [4, 17], we allowed coding multiple arguments in one issue thread to capture the diverse concerns people raised. We define an *argument* as all the quotes that discuss the same claim, which may agree or disagree with the claim. A new argument can be brought up by making a claim about a new topic. Then, all discussions around this new claim are considered to be part of the new argument. We gave each argument a distinctive ID based on the chronological order of their first appearance in the issue discussion thread.

The first author conducted the initial coding for model refinement, with the support of two other authors when ambiguities or questions raised. Once the initial coding was finished, three researchers conducted six two-hour-long meetings and reviewed about 60% of the quotes, including the most representative and questionable examples. The discussion was iterative, through which the team reached an agreement on all the discussed quotes and created a detailed codebook (see https://github.com/HCDLab/ArguLens). The first author finally used the codebook to re-code the non-discussed quotes.

To further assess the quality of our coding schema, we evaluated inter-rater reliability on a random sample of 200 non-discussed quotes. Four authors other than the first author each independently labeled 50 quotes using the codebook. The Coehn's κ [19] against the label provided by the first author is 0.85 for Level 1 coding, 0.79 for argumentation components, and 0.80 for standpoints, indicating a "substantial" to "almost perfect" agreement [61].

Results

The content analysis generated a total of 5123 quotes from the five issue threads with a median of 621 quotes per issue

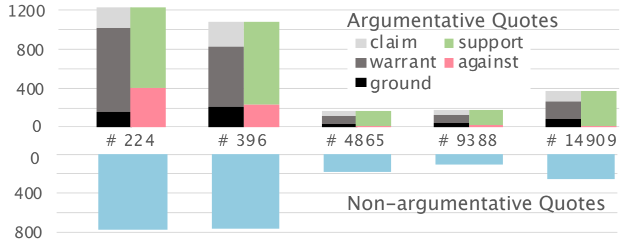


Figure 4. Distribution of quotes coded with each of the adapted codes.

($IQR = 1849 - 319.5$). Figure 4 shows the number of quotes annotated with each code for each issue.

Level 1 Coding

The first-level coding focused on identifying the argumentative comments in the issue threads and included two codes:

Argumentative (3034 quotes): Argumentative comments are made with the purpose of persuasion and fulfill a role of argument component (i.e., claim, ground, or warrant) in an issue discussion. These comments are the focus of our study and are further distinguished in Level 2 coding. As an example, Ivalexa made the following comment in issue 9388 to request implementing a desired behavior: "*Is it possible to add feature in order to force vscode to open file only by double clicking on it, single click should only select a file in explorer?*"

Non-argumentative (2089 quotes): This category includes any comments that are not persuasive in nature. We identified 11 concrete sub-categories of non-argumentative quotes such as questions (e.g., *"Do you expect this behavior ... everywhere else ... or you just want it for the explorer?"*), workaround suggestions (e.g., *"when I tried that earlier today it opened the file that received focus in the explorer tree immediately"*), progress inquiries (e.g., *"Any news on this?"*), expressions of appreciation (e.g., *"That's music to my ears!!"*), *"+1"* or *"-1"* alone in a sentence, etc. Please refer to the codebook in the auxiliary material for a complete list of non-argumentative sub-categories with examples.

Level 2 Coding

In this level, each argumentative quote is examined along two dimensions. The first dimension considers argument *components* and contains the following three codes:

Component - Claim (675 quotes): These quotes represent specific requests from the issue discussion participant. In our data about software usability in OSS, these quotes focused on describing users' expected behavior, proposing a new design, giving design suggestions, and expressing opinion towards a proposed design, etc. For example, in issue 4865, a participant provided the following design suggestion: "*I would just like you to consider showing selections and search results highlighted in the minimap as a feature.*"

Component - Ground (541 quotes): These quotes illustrate facts about the current system or about a competitor' system reported by the discussion participant. There are two observed sub-themes under the *Ground* code: (1) Report on the behavior of an existing system; for example: "*ctrl+tab between working files appears to work in most-recent order, as opposed to tabs which usually operate in the order they appear or are arranged. (I've seen some editors support both).*" (2) Description on how they usually interact with an existing system; for example: "*The first thing I ever do with VS Code is swap its close file and close active editor keyboard shortcuts, so that ctrl+W actually closes a file when I press it, rather than still leaving it in working files and cluttering up ctrl+tab.*".

Component - Warrant (1818 quotes): This group of quotes is focused on connecting a *Claim* and a *Ground* with justifications such as potential use cases if the feature were implemented, the importance of following basic design principles, and personal feelings from using the existing systems. Most frequently, a *Warrant* manifests use cases to demonstrate how the proposed changes (a *Claim*) would fix the users' self-reported problem (i.e., a *Ground*). For example, the following comment from issue 396 includes a description of a potential use case for the feature of "opening multiple project folders in the same window": "*As a Go developer, I find this feature extremely useful... I need to be able to quickly navigate to [many third-party libraries and projects] and read that code.*"

The second dimension of Level 2 coding models the *standpoint* of the discussion participant and includes the following codes:

Standpoint - Support (2347 quotes): This code includes comments showing agreement with the standpoint of the original argument. For example, in issue 224 an OSS participant supported adding proper tabs, commenting: "*Regardless of whether one can get by / used to 'ctrl+tab' as an alternative, new VS code users will probably be put off by the lack of tabs.*"

Standpoint - Against (687 quotes) This group of comments shows disagreement with respect to a particular argument. For example, opposing the argument given in the previous quote, another participant commented: "*I do not think tabs are a good way to show the list of open files unless you manage these things actively and close them.*"

A Synthesized Example

The following segment of discussion from issue 224 illustrates how our proposed framework can be applied in the context of OSS usability issues. In this example, the assigned codes from both the argument *components* and the *standpoint* dimensions are shown in pairs in square brackets at the end of each quote.

TURKEYMAN: I really miss proper tabs for open files (like VS proper), and the ability to rip a tab out into its own window. [*warrant, support*]

NICTE: +1 [*non-argumentative*]. Tabs are the default way to work even in Visual Studio, to not mention other text editors such as sublime [*ground, support*].

NVIVO: I vote for adding tabs too. [*claim, support*]

RUSSBAZ: To be honest, one of the defining features of VS Code for me was an absence of tabs. [*warrant, against*] I didn't even realise (at first) that they were missing! [*warrant, against*] It was a very interesting point of view. [*warrant, against*]

FELIXFBECKER: . . . Back when I used Atom, I hated it that [*warrant, against*] every time I just quickly looked at a file, it would open a new tab, and within minutes my tab bar was full of useless tabs. [*ground, against*] . . .

Multiple Arguments

Extensive discussion of an issue can start from debating the needs for a requested feature and develop into considering design alternatives, more fine-grained design details, and opinions towards competitors' solutions. The following example (issue 224) shows a case where an OSS participant explicitly leads the discussion to the design of *tabs* after a long discussion around the need for tabs: "*It's the top requested feature with thousands of votes. ... This conversation is pointless and needs to move on to "How exactly should tabs work?"*

We identified a median of 24.0 ($IQR = 49.0 - 15.5$) arguments per issue among the five issue discussion threads. The median number of discussion quotes per argument is 5.0 ($IQR = 11.5 - 2.0$). Over all argumentative quotes in an issue thread, the median percentage of quotes belonging to arguments other than the original argument addressed in the issue post is 54.5% ($IQR = 62.1\% - 32.4\%$).

Discussion

The content analysis results unveil several key characteristics of the arguments in usability issue discussions. First, non-argumentative comments constitute a large portion of the discussion. These comments usually include contents that are less relevant to the community's needs. Simply hiding them could help developers retrieve useful information and, therefore, would be a big step forward towards addressing information overload in a community-driven environment. Second, *warrant* represents the majority component in the argumentative comments, followed by *claim* then *ground*. Furthermore, *warrant*s usually include speculated use cases and personal opinions. These evidence echos with the previous observations that OSS discussions about usability are usually influenced by the personal opinions and experiences [6, 17]. Third, while all the issues were initially created for requesting specific changes in the existing interaction design, the discussion did not bound to the topic of the original post. Conversely, their content develops and broadens to include many related topics.

To better understand the characteristics of multiple arguments, we analyzed the *Against* to *Support* ratio among all arguments in the issue discussion threads to see how disagreed or controversial they were. We found that although the arguments from the original posts were the mostly-discussed in all five issues, they are not necessarily a highly-disagreed issue (their average *Against-Support* ratio is 0.115, comparing to the overall average of 0.325). We also observed that the highly-disagreed arguments (i.e. *Against-Support* ratio closer to 1) usually reside in the middle of an issue thread, and are not necessarily heavily-discussed. One potential explanation is that these arguments may be hard to recognize in a lengthy thread. The current issue discussion platforms only display an issue thread in chronological order; this may cause the multiple requests and their corresponding discussion in the middle of the discussion threads to be easily overlooked.

MACHINE LEARNING-BASED ARGUMENT EXTRACTION

Annotated issue discussion threads for argumentation are essential for building intelligent tools supporting high-level reasoning tasks for the issue reporter, triager, assignees, and various stakeholders, including argument navigation, retrieval and trade-off analysis towards usability. However, manually labeling the comments is time-consuming, especially when the discussions become lengthy and the issue repository gets large. In this section, we explore the potential of using automated techniques in ArguLens to reduce the annotation effort and address RQ2: *How effective are machine learning models in extracting arguments and their structure?*

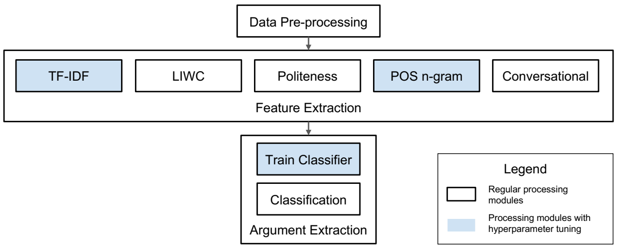


Figure 5. Major modules of the supervised argument extraction process.

We used the 5123 quotes manually labeled in the previous section as the gold standard. While it would be ideal to adopt unsupervised methods to eliminate any manual effort on annotation, previous studies showed the limited performance on text-based unsupervised methods (e.g. topic modeling and clustering) and their sensitivity to parameter setting [2, 4, 16]. Therefore, we experimented with supervised learning techniques to address the argument extraction task.

Methods

Similar to the two-level coding analysis, the complete argument extraction is comprised of two-layer classification steps. Given one sentence from the issue discussion, the first layer classifies the sentence into either *Argumentative* or *Non-argumentative*. In the second layer, two separate classifiers label each argumentative sentence with topics from both dimensions of argumentation components (*Claim, Warrant or Ground*) and standpoint (*Support or Against*). The complete process includes three main steps: data pre-processing, feature extraction, and argument extraction (see Figure 5).

Data Pre-processing

We first cleaned up the dataset by removing the quotes that contain only non-alphabet characters and replaced the special contents with uniquely assigned tokens (see Table 2). We then tokenized and lowercased each quote into individual words. Additionally, when extracting TF-IDF features and POS tag features (see the next section), we excluded punctuation marks and common contractions, and performed lemmatization on each word to obtain its dictionary form; for the other features, such as Politeness, we skipped this step because the information such as punctuation can be a strong indicator. We explain all the features and how we obtained them in the next section.

Feature Extraction

We extracted two major types of features, i.e. Textual Features and Conversational Features, to train our supervised classifiers. Previous work on issue content analysis and argumentation mining [4, 31] has demonstrated the effectiveness of using those features on related tasks with similar datasets.

Special Token	Content Replaced
CODE_BLOCK	multi-line source code blocks
CODE_SEGMENT	inline source code snippets
QUOTE	quotations made in previous comments
URL	reference links to external resources
SCREEN_NAME	mentions to GitHub users
VERSION_NUM	version numbers
PLUS_ONE	token + followed by numbers (i.e. +1)
MINUS_ONE	token - followed by numbers (i.e. -1)
ISSUE_REFERENCE	token # followed by numbers (i.e. #224)

Table 2. Special tokens used during pre-precessing.

Textual Features are extracted from the textual content of individual quotes. The fundamental assumption is that people potentially describe the same type of content in the argument using similar expressions, i.e., the words they use and the underlying social and physiological meanings they convey [57]. The concrete textual features used in our method include:

1. *Term Frequency-Inverse Document Frequency (TF-IDF):* a numerical measure that indicates how important a token is to a document in a corpus [51]. Each word in corpus after data pre-processing acts as one feature weighted using the frequencies of words multiplied by their inverse document-frequency. We added n-grams as additional features to represent *n* token sequences.
2. *Linguistic Inquiry and Word Count (LIWC):* a series of features obtained from LIWC, a text analysis program that captures a wide range of linguistic and psychological characteristics [46]. Previous studies have shown its effectiveness in various text analysis tasks, including detecting emotions, opinions, and bias on blogs and social media [57, 63].
3. *Politeness:* an overall measure ranging between 0 and 1 to reflect the key components of politeness theory, such as indirection, deference, personalization, and modality [20].
4. *Part-of-Speech n-gram (POS n-gram):* a sequence of *n* POS tags. POS tags indicate the part of speech, such as nouns, verbs, adjectives, etc. POS n-grams capture the structure of the sentence and therefore can potentially deduce subjective information comparing to simple text n-gram [8].

Conversational Features emphasize the characteristics of each quote within the discourse. It has been applied to processing software issues for issue thread summarization and information type detection [49, 4]. Similar to Arya et. al. [4], our conversational features include five groups: (1) *Participant features* describe the role that the comment author plays in the project (i.e. owner, collaborator, member, or other) and in the current issue thread (i.e. issue author or not); (2) *Length features* depict the absolute length of a quote and its relative length with respect to other quotes in the issue comment and in the thread; (3) *Structural Features* describe the location of the quote in the whole discussion threads; (4) *Temporal Features* describe the time when the comment is made in relation to the immediately previous and next comment, as well as the whole discussion thread; and (5) *Code Feature* indicates if the current comment contains code snippets.

Argument Extraction

Using the features extracted from issue quotes as input, we trained the machine learning classifiers for extracting arguments and their anatomy in the discussion quotes. Our candidate classifiers are Support Vector Machine (SVM) and Naive Bayes (NB) models. SVMs are well suited to deal with learning tasks where the number of features is large with respect to the number of training instances [35, 37]. In particular, we applied LinearSVM classifier with the class weight balancing technique to automatically adjust weights based on the class frequencies in the sample data. LinearSVM is computationally much more efficient than SVMs with non-linear kernels and yields state-of-the-art performance for high-dimensional sparse datasets, a typical scenario in text classification tasks like ours [36]. Performing a substantial feature selection would avoid dealing with such high dimensional input spaces. However, in supervised learning for text classification tasks, very few features are actually irrelevant, and feature selection often may cause a significant loss of information [35]. Thus, we did not perform any feature selection. We also considered Naive Bayes model because it tends to achieve good performance when the available training set is relatively small [37]. Specifically, we used the Complement Naive Bayes (CNB), which is an adaption of the standard Multinominal Naive Bayes (MNB) algorithm and is particularly suited for imbalanced data [50].

Experiment Design

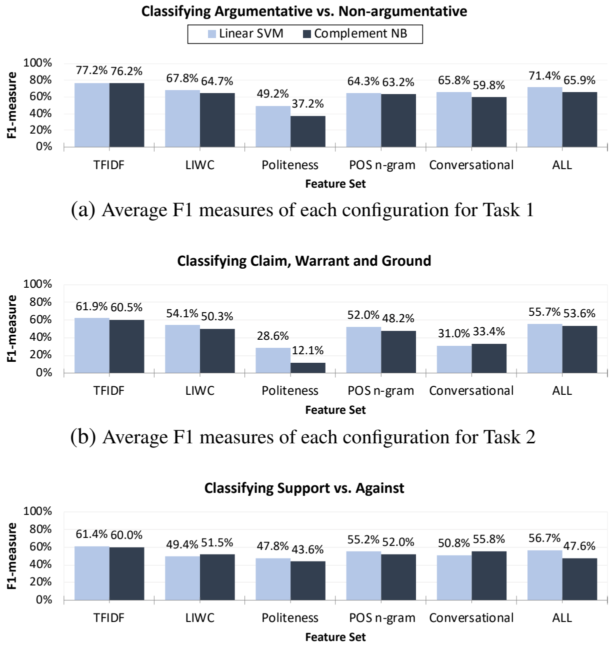
We focused on three classification tasks: (Task 1) classifying Argumentative vs. Non-argumentative quotes, (Task 2) classifying the Claim, Warrant, and Ground components of argumentative quotes, and (Task 3) classifying Support vs. Against standpoints of argumentative quotes. We experimented building classifiers considering two independent variables:

(1) The machine learning model to adopt with two levels: LinearSVM and Complement NB.
(2) Feature set to use as input for the machine learning models with six levels: TFIDF, LIWC, Politeness, POS n-gram, Conversational features, and concatenating all five aforementioned feature sets together.

The combination of the two independent variables resulted in 12 configurations. To evaluate the performance of all configurations, we adopted the standard nested stratified 5-fold cross-validation [27], which contains two layers. The inner layer is used to tune the hyperparameters for certain modules in our techniques (modules that need hyperparameter tuning are shown with a blue shade in Figure 5); e.g., for Linear SVM, the tuned hyperparameter is the cost parameter C, which indicates the penalty of misclassification on the training set. The outer layer is then used to evaluate the performance of the models on unseen data. The performance of the models is evaluated using average Precision, Recall, and F1-Measure from the outer layer; these metrics are commonly used to evaluate classification tasks [47]. For a certain class in question, Precision measures how accurate the classifiers' predictions on the class are correct; Recall measures how effective the classifier is in predicting the correct labels for all instances from that class; F1-Score is the harmonic mean of the Precision and Recall. We used python and the Scikit Learn library to implement the experiments and calculate all metrics.

Results and Discussion

Figure 6 summarizes the average F1-scores of the 12 configurations for the three classification tasks. Our results indicate



(a) Average F1 measures of each configuration for Task 1

(b) Average F1 measures of each configuration for Task 2

(c) Average F1 measures of each configuration for Task 3

Figure 6. Model Performance for each classification task.

Label to Classify	Precision	Recall	F1	Support
Argumentative	0.80	0.85	0.82	608
Non-argumentative	0.76	0.68	0.72	418
Task 1 Average/Total	0.78	0.77	0.77	1025
Baseline	0.29	0.50	0.37	1025
Claim	0.63	0.52	0.57	135
Warrant	0.74	0.83	0.78	394
Ground	0.56	0.46	0.50	109
Task 2 Average/Total	0.64	0.60	0.62	608
Baseline	0.21	0.33	0.25	608
Support	0.83	0.84	0.83	470
Against	0.41	0.40	0.40	138
Task 3 Average/Total	0.62	0.62	0.61	608
Baseline	0.39	0.50	0.44	608

Table 3. Detailed measurements for three classification tasks with configuration of Linear SVM and TF-IDF features, as well as the baseline (always predicting the majority class) performance. Support refers to the number of instances in the testing set for that label.

that using TF-IDF features yielded the highest F1-score. One possible explanation is that our annotated corpus includes a large portion of domain-specific words. TF-IDF features excelled other feature sets at capturing domain knowledge from the text, which might be strong indicators for their argumentation roles. This result also has positive practical implications because TF-IDF features are relatively easy to obtain compared to other feature sets. On the other hand, the selection of different classifiers was not as critical as the feature sets used. In fact, given the same set of features, we observed a negligible difference in performance between Linear SVM and Complement Naive Bayes classifiers.

The best configuration for all three classification tasks was obtained using Linear SVM with TF-IDF features, achieving an average F1-score of 77.22%, 61.92%, and 61.42% for the three tasks respectively. We report the Precision, Recall and F1-score for all three tasks in Table 3. As a baseline, we also report the performance of a trivial classifier always predicting the majority class (i.e., Argumentative, Warrant, and Support, for the three tasks, respectively). This is a consolidated practice in machine learning for text categorization as the baseline performance gives an indication of the inherent difficulty of the classification task itself [54, 62].

For all tasks, we observed substantial improvements over the baseline classifier. Specifically, the first level classification, i.e. Argumentative versus Non-argumentative, achieved the best result. Notably, the Precision and Recall for the Argumentative class are both more than 80%. Such strong performance demonstrates the possibility of designing tools that leverage supervised classification models to remove a large number of irrelevant content for the issue readers. In terms of the argument components and standpoints, our classifiers performed better for classes that contain more data instances, i.e. Support, and Warrant. However, its lukewarm performance on Against, Claim, and Ground suggests that the user might need to accumulate more annotated training data to achieve a better result. While not perfect, our techniques can be used in a semi-automated way to suggest the underlying argument anatomy in the issue discussion, or be plugged into tools for empirical studies involving mining overall trends in ITSs.

As regard to the classification approach, we would like to remind the reader that our goal in this work is not to advance the latest NLP techniques for argument classification. Rather, we aim at demonstrating the potential of anatomizing usability issue discussions using an argumentation framework, thus paving the road for renovating how people can collaboratively address open source usability issues. We are aware of recent advancements in NLP research leveraging word or sentence embedding and neural architectures for text categorization. However, the practical value of such approaches has been questioned in many software engineering areas because of the difficulty to acquire a large amount of training data and to disambiguate domain-specific terms [23, 28]. Meanwhile, we hope that our method can serve as a strong baseline to attract more researchers to apply advanced argumentation-mining techniques on this challenging and important problem.

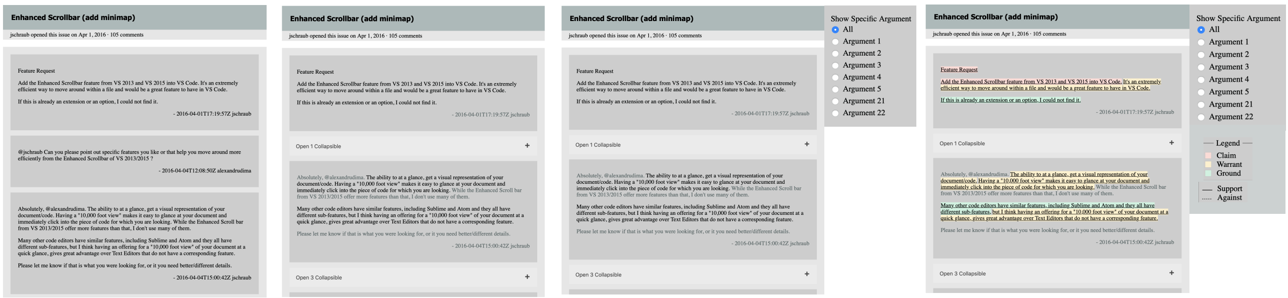
PRACTITIONER EVALUATION

To better understand the ArguLens approach from the perspectives of experienced OSS practitioners (RQ3), we performed a within-subject user study with eight frequent users of GitHub Issues. This study is approved by the Institutional Review Boards of Polytechnique Montreal and McGill University.

Methods

We recruited eight participants (three females) from personal contacts. All participants had experience with GitHub and its *Issues* feature, having contributed to at least one project on the platform. They all had used VSCode as a text or code editor; some used it on a daily basis at the time of the study.

Each user study took about 1.5 hours to complete. During the study, we first conducted a semi-structured interview with each participant to collect demographic information. We then asked



Enhanced Scrollbar (add minimap)

jschraub opened this issue on Apr 1, 2016 · 105 comments

Feature Request

Add the Enhanced Scrollbar feature from VS 2013 and VS 2015 into VS Code. It's an extremely efficient way to move around within a file and would be a great feature to have in VS Code.

If this is already an extension or an option, I could not find it.

- 2016-04-01T17:19:57Z jschraub

@jschraub Can you please point out specific features you like or that help you move around more efficiently from the Enhanced Scrollbar of VS 2013/2015 ?

- 2016-04-04T12:08:50Z alexandrudima

Absolutely, @alexandrudima. The ability to at a glance, get a visual representation of your document/code. Having a "10,000 foot view" makes it easy to glance at your document and immediately click into the piece of code for which you are looking. While the Enhanced Scroll bar from VS 2013/2015 offer more features than that, I don't use many of them.

Many other code editors have similar features, including Sublime and Atom and they all have different sub-features, but I think having an offering for a "10,000 foot view" of your document at a quick glance, gives great advantage over Text Editors that do not have a corresponding feature.

Please let me know if that is what you were looking for, or it you need better/different details.

- 2016-04-04T15:00:42Z jschraub

(1) Raw Condition

Enhanced Scrollbar (add minimap)

jschraub opened this issue on Apr 1, 2016 · 105 comments

Feature Request

Add the Enhanced Scrollbar feature from VS 2013 and VS 2015 into VS Code. It's an extremely efficient way to move around within a file and would be a great feature to have in VS Code.

If this is already an extension or an option, I could not find it.

- 2016-04-01T17:19:57Z jschraub

Open 1 Collapsible +

Absolutely, @alexandrudima. The ability to at a glance, get a visual representation of your document/code. Having a "10,000 foot view" makes it easy to glance at your document and immediately click into the piece of code for which you are looking. While the Enhanced Scroll bar from VS 2013/2015 offer more features than that, I don't use many of them.

Many other code editors have similar features, including Sublime and Atom and they all have different sub-features, but I think having an offering for a "10,000 foot view" of your document at a quick glance, gives great advantage over Text Editors that do not have a corresponding feature.

Please let me know if that is what you were looking for, or it you need better/different details.

- 2016-04-04T15:00:42Z jschraub

Open 3 Collapsible +

(2) Argumentative-Only Condition

Enhanced Scrollbar (add minimap)

jschraub opened this issue on Apr 1, 2016 · 105 comments

Feature Request

Add the Enhanced Scrollbar feature from VS 2013 and VS 2015 into VS Code. It's an extremely efficient way to move around within a file and would be a great feature to have in VS Code.

If this is already an extension or an option, I could not find it.

- 2016-04-01T17:19:57Z jschraub

Open 1 Collapsible +

Absolutely, @alexandrudima. The ability to at a glance, get a visual representation of your document/code. Having a "10,000 foot view" makes it easy to glance at your document and immediately click into the piece of code for which you are looking. While the Enhanced Scroll bar from VS 2013/2015 offer more features than that, I don't use many of them.

Many other code editors have similar features, including Sublime and Atom and they all have different sub-features, but I think having an offering for a "10,000 foot view" of your document at a quick glance, gives great advantage over Text Editors that do not have a corresponding feature.

Please let me know if that is what you were looking for, or it you need better/different details.

- 2016-04-04T15:00:42Z jschraub

Open 3 Collapsible +

Show Specific Argument
● All
○ Argument 1
○ Argument 2
○ Argument 3
○ Argument 4
○ Argument 5
○ Argument 21
○ Argument 22

(3) Separated Arguments Condition

Enhanced Scrollbar (add minimap)

jschraub opened this issue on Apr 1, 2016 · 105 comments

Feature Request

Add the Enhanced Scrollbar feature from VS 2013 and VS 2015 into VS Code. It's an extremely efficient way to move around within a file and would be a great feature to have in VS Code.

If this is already an extension or an option, I could not find it.

- 2016-04-01T17:19:57Z jschraub

Open 1 Collapsible +

Absolutely, @alexandrudima. The ability to at a glance, get a visual representation of your document/code. Having a "10,000 foot view" makes it easy to glance at your document and immediately click into the piece of code for which you are looking. While the Enhanced Scroll bar from VS 2013/2015 offer more features than that, I don't use many of them.

Many other code editors have similar features, including Sublime and Atom and they all have different sub-features, but I think having an offering for a "10,000 foot view" of your document at a quick glance, gives great advantage over Text Editors that do not have a corresponding feature.

Please let me know if that is what you were looking for, or it you need better/different details.

- 2016-04-04T15:00:42Z jschraub

Open 3 Collapsible +

Show Specific Argument
● All
○ Argument 1
○ Argument 2
○ Argument 3
○ Argument 4
○ Argument 5
○ Argument 21
○ Argument 22

— Legend —
🟥 Claim
🟨 Warrant
🟩 Ground

—— Support
..... Against

(4) Decomposed Arguments Condition

Figure 7. Stimuli used in the user studies with OSS practitioners (using issue 4865 as an example).

participants to review four issue discussion threads that we analyzed in the previous steps; to make the review a manageable task within the context and the time frame of the user study, we trimmed each issue discussion to only include the first 50 comments. We developed web-based stimuli (see Figure 7) to represent the issues in four increasingly complex conditions. Consistently with our intention of exploring practitioners' perspectives about the argument anatomy itself, we focused on presenting an interface with minimal interaction design. All but the first condition leveraged the ArguLens technique and all annotations about arguments, argumentation components, and standpoints were based on the manual content analysis results. The four conditions are:

1. **Raw**: Participants see the original discussion comments with no processing.
2. **Argument-Only**: Participants see the discussion comments with non-argumentative sentences greyed out or put into collapsed sections (if the entire comment is non-argumentative).
3. **Separated Arguments**: Participants see the argumentative comments (same as Condition 2) and a list of radio buttons as a sidebar indicating different arguments in the thread. The participants can further navigate among the arguments by clicking on the radio buttons, making all other arguments greyed out or put into collapsed sections.
4. **Decomposed Arguments**: Participants see the same content and have the same argument filtering ability as Condition 3. However, all argumentative sentences they see are color-coded and highlighted according to their argumentation components and standpoints.

We followed a within-subject experimental design; the order of the conditions and the condition-issue assignments were counterbalanced using Zeelenberg and Pecher's approach [65]. At the beginning of each issue review, participants were provided with a one-page description about the concepts related to the corresponding condition and could ask any clarification questions. While reviewing each issue, the participants were asked to (1) describe the main topics discussed, (2) identify the potential positive and negative impacts on users when the issue is fixed, and (3) decide whether the issue should be fixed. We also encouraged the participants to think-aloud during these tasks. At the end of each issue review, the participants completed a System Usability Scale (SUS) questionnaire [15] and a Subjective Mental Effort Question (SMEQ) [53]. The SUS questionnaire is commonly used to quantify the perceived usability of a system; higher SUS scores (ranging from 0 to 100) indicate better usability; a meta review of 2,324 studies indicated a mean SUS score of 70 across various systems [9]. SMEQ includes one rating question (ranging from 0 to 150) for perceived mental effort; lower scores indicate a lower mental effort. Once all issues were reviewed, we asked the participants to select the most and the least useful representation and to provide rationales for their choice.

Results and Discussion

All but one participant completed the four tasks; the remaining participant (P2) did not evaluate the *Separated Arguments* representation due to time constraints. The average SUS and SMEQ scores are summarized in Table 4. Given the qualitative nature of the study, in the rest of this section we focus on reporting the themes emerged from participants' feedback.

	Raw	Arg-only	Sep. Args.	Dec. Args.
SUS	78.21	84.38	87.86	71.88
SMEQ	24.50	18.13	17.86	34.25

Table 4. Average SUS and SMEQ scores for the four conditions.

Our results indicated that designing the representation of the anatomy of arguments is an important topic. Among our eight participants, seven appreciated having the possibility to hide or gray out the non-argumentative comments for the purpose of determining the community opinions when reviewing the issues. For example, P6 mentioned: "*It [hiding the non-argumentative sentences] already helps a lot to quickly get the idea about the issue and what people are thinking about it.*" P5 also mentioned that simply highlighting the argumentative comments and sentences helped "*differentiate things that you want to put more emphasis on.*". Only one (P3) preferred the *Raw* representation, because it presents an interface familiar to the participant and includes all contextual information; P3 commented: "*I don't like having just the arguments. I would like to read a little bit more about the background.*"

Four participants considered the *Separated Arguments* representation as the most useful. They thought that separating arguments by their topics helped them to be more focused when considering the diverse opinions. For example, P5 mentioned "*I think that feature [being able to quickly switch among the arguments] alone is really great. ... it's really helpful to be* *specific.*" P1 also thought that having separated arguments is "*useful to summarize long issues and get main points quickly.*"

Two participants found the *Decomposed Arguments* the most useful. They both used the argument components and standpoints extensively while reviewing the issues to determine the main topics and the potential impacts to users. For example, P8 said: *"The first question [main topic of the issue] was probably the claim here ... the support and against could determine the other two [positive and negative impacts]... and probably, if I count how many support and against I have in the whole issue, I can say that it's a good idea to implement it or not."*

Although the *Decomposed Arguments* provides useful supports for issue review, four participants disliked this representation because of its conceptual and visual complexity. For example, P3 explained: "*I can say that having three colors in here and underline and everything... like, it makes reading hard.*" Participants also indicated that understanding this representation requires some learning. For example, P6 said: "*The colors are not very intuitive in a way, but it may be that I'm just not used to it.*" P2 also commented: "*I think this would be very powerful ... if someone has been using this for a longer time.*"

GENERAL DISCUSSION

ArguLens has the potential to support a wide range of applications and at the same time has limitations that can be addressed in the future. We discuss these topics here.

Potential Applications

ArgueLens can be directly used for tools that visualize the argument anatomy of community-generated usability discussions. Our user study showed that such anatomy helps the participants focus on the most relevant information when reviewing usability issues. Effective interaction and visual design leveraging ArgueLens would then provide the much-needed support for OSS stakeholders (e.g. developers, designers, and end users) to better comprehend and consolidate community opinions about the usability issues of the system.

Moreover, ArgueLens introduces an effective, yet straightforward framework of usability argumentation. Tools that encourage or reinforce such framework can support OSS communities to move towards a shared convention and a common language of discussing usability-related topics, which in turn supports community engagement in such challenging issues.

Leveraging ArguLens, new automated or semi-automated techniques can also be developed to support analytic activities on OSS usability issues at the project level. For example, discussion summary and prioritization tools of usability issues can be developed based on ArguLens. Such efforts would then further bolster OSS developers' empathy towards users, thus encouraging them to better incorporate the end users' voices.

Limitations and Future Work

Due to the effort required for the task of manual coding, we performed an in-depth analysis on only five usability issues from one OSS project. This might have introduced a threat to external validity that we mitigated by focusing on the most heavily discussed issues that involved diverse community members of an active OSS project. In this way, we were able to address the complex situations where support for understanding the community opinions is most needed. Nevertheless, the ArguLens framework and the performance of the machine learning techniques can be affected by the unique characteristics of the analyzed issues and the project. Future work extending this analysis to other usability issues and projects would allow establishing its external validity. Additionally, our work focused on the usability issues because of their increasing importance in OSS applications and the overwhelming challenges involved to address them. Although we speculate that ArguLens can be applicable to other types of OSS issues prone to extended community discussions (e.g. privacy or security), as well as other types of community-driven platforms, these applications require future work.

Further, although our machine learning solutions performed well in certain tasks (e.g. discriminating Argumentative and Non-argumentative sentences), the performance was less desirable in others. Particularly, its performance is sensitive to the amount of the training data, indicating that extended manual labeling effort is still necessary to achieve its utility in practice. Furthermore, given the current precision and recall achieved by the argument extraction tasks, human-in-the-loop method is necessary to support the ArguLens technique. Future work might also explore combining supervised and unsupervised approaches to further reduce the manual effort.

Finally, while our user study focused on a typical application scenario of reviewing a usability issue to determine its potential impact on the community, the study certainly does not cover all the scenarios potentially supported by ArguLens. Moreover, the stimuli used in our user study provided only a basic presentation of the embedded information and did not focus on offering a sophisticated visual and interaction design. Thus, exploring an effective design of tools that leverage the ArguLens techniques is an important future work.

CONCLUSION

We proposed and evaluated ArguLens, a conceptual framework and machine learning-based technique that leverages an argumentation model to support comprehension and consolidation of OSS community's opinions on usability issues. We created a corpus of 5123 quotes, annotated with an argumentation framework as the foundation for ArguLens. Leveraging supervised machine learning on this corpus, we evaluated various argument extraction techniques; our best performing classifier achieved satisfying results, paving the way to the implementation of semi-automated approaches and tools. Our study with experienced ITS users revealed that ArguLens can help practitioners get focused and retrieve the relevant information when reviewing issue discussions. In sum, as an important step forward, ArguLens can accelerate building effective methods and tools that aim to improve OSS usability.

ACKNOWLEDGEMENT

We thank our participants in the user study for their thoughtful efforts. This work was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) [RGPIN-2018-04470, RGPIN-2019-05403].

REFERENCES

[1] Rob Abbott, Marilyn Walker, Pranav Anand, Jean E. Fox Tree, Robeson Bowmani, and Joseph King. 2011. How Can You Say Such Things?!?: Recognizing Disagreement in Informal Political Argument. In *Proceedings of the Workshop on Languages in Social Media (LSM '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 2–11. http://dl.acm.org/citation.cfm?id=2021109.2021111

[2] Amritanshu Agrawal, Wei Fu, and Tim Menzies. 2018. What is wrong with topic modeling? And how to fix it using search-based software engineering. *Information and Software Technology* 98 (2018), 74–88.

[3] Morten Andreasen, Henrik Nielsen, Simon Schrøder, and Jan Stage. 2006. Usability in open source software development: opinions and practice. *Information technology and control* 35, 3A (2006), 303–312. DOI: http://dx.doi.org/10.5755/j01.itc.35.3.11776

[4] Deeksha Arya, Wenting Wang, Jin L. C. Guo, and Jinghui Cheng. 2019. Analysis and Detection of Information Types of Open Source Software Issue Discussions. In *Proceedings of the 41st International Conference on Software Engineering (ICSE '19)*. IEEE Press, Piscataway, NJ, USA, 454–464. DOI: http://dx.doi.org/10.1109/ICSE.2019.00058

[5] Katie Atkinson, Trevor Bench-Capon, and Peter McBurney. 2006. Computational representation of practical argument. *Synthese* 152, 2 (2006), 157–206.

[6] Paula M. Bach and John M. Carroll. 2009. FLOSS UX Design: An Analysis of User Experience Design in Firefox and OpenOffice.org. In *Open Source Ecosystems: Diverse Communities Interacting*, Cornelia Boldyreff, Kevin Crowston, Björn Lundell, and Anthony I. Wasserman (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 237–250.

[7] Paula M Bach, Robert DeLine, and John M Carroll. 2009. Designers wanted: Participation and the User Experience in Open Source Software Development. In *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*. ACM Press, New York, New York, USA, 985. DOI: http://dx.doi.org/10.1145/1518701.1518852

[8] Akshat Bakliwal, Piyush Arora, Ankit Patil, and Vasudeva Varma. 2011. Towards Enhanced Opinion Classification using NLP Techniques. In *Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2011)*. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, 101–107.

[9] Aaron Bangor, Philip T. Kortum, and James T. Miller. 2008. An Empirical Evaluation of the System Usability Scale. *International Journal of Human-Computer Interaction* 24, 6 (2008), 574–594. DOI: http://dx.doi.org/10.1080/10447310802205776

[10] Olga Baysal, Reid Holmes, and Michael W. Godfrey. 2014. No Issue Left Behind: Reducing Information Overload in Issue Tracking. In *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014)*. ACM, New York, NY, USA, 666–677. DOI: http://dx.doi.org/10.1145/2635868.2635887

[11] Jamal Bentahar, Bernard Moulin, and Micheline Bélanger. 2010. A taxonomy of argumentation models used for knowledge representation. *Artificial Intelligence Review* 33, 3 (2010), 211–259.

[12] Jamal Bentahar, Bernard Moulin, and Brahim Chaib-draa. 2004. Commitment and Argument Network: A New Formalism for Agent Communication. In *Advances in Agent Communication*, Frank Dignum (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 146–165.

[13] Jamal Bentahar, Bernard Moulin, John-Jules Ch. Meyer, and Brahim Chaib-draa. 2005. A Computational Model for Conversation Policies for Agent Communication. In *Computational Logic in Multi-Agent Systems*, João Leite and Paolo Torroni (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 178–195.

[14] Filip Boltužić and Jan Šnajder. 2014. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*. Association for Computational Linguistics, Baltimore, Maryland, 49–58.

[15] John Brooke and others. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[16] Tse-Hsun Chen, Stephen W Thomas, and Ahmed E Hassan. 2016. A survey on the use of topic models when mining software repositories. *Empirical Software Engineering* 21, 5 (2016), 1843–1919.

[17] Jinghui Cheng and Jin L.C. Guo. 2018. How Do the Open Source Communities Address Usability and UX Issues?: An Exploratory Study. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems (CHI EA '18)*. ACM, New York, NY, USA, Article LBW523, 6 pages. DOI: http://dx.doi.org/10.1145/3170427.3188467

[18] Jinghui Cheng and Jin L. C. Guo. 2019. Activity-based Analysis of Open Source Software Contributors: Roles and Dynamics. In *Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '19)*. IEEE Press, Piscataway, NJ, USA, 11–18. DOI: http://dx.doi.org/10.1109/CHASE.2019.00011

[19] Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20, 1 (1960), 37–46. DOI: http://dx.doi.org/10.1177/001316446002000104

[20] Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of the 51st Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Sofia, Bulgaria, 250–259. https://www.aclweb.org/anthology/P13-1025

[21] Phan Minh Dung. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence* 77, 2 (1995), 321–357.

[22] Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural End-to-End Learning for Computational Argumentation Mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 11–22. DOI: http://dx.doi.org/10.18653/v1/P17-1002

[23] S. Fakhoury, V. Arnaoudova, C. Noiseux, F. Khomh, and G. Antoniol. 2018. Keep it simple: Is deep learning good for linguistic smell detection?. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 602–611. DOI: http://dx.doi.org/10.1109/SANER.2018.8330265

[24] Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, 987–996. https://www.aclweb.org/anthology/P11-1099

[25] Eirini Florou, Stasinos Konstantopoulos, Antonis Koukourikos, and Pythagoras Karampiperis. 2013. Argument extraction for supporting public policy formulation. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*. Association for Computational Linguistics, Sofia, Bulgaria, 49–54. https://www.aclweb.org/anthology/W13-2707

[26] Kathleen Freeman. 1993. *Toward Formalizing Dialectical Argumentation*. Ph.D. Dissertation. University of Oregon, Eugene, OR, USA. UMI Order No. GAX94-05172.

[27] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2009. *The elements of statistical learning* (2nd ed.). Springer series in statistics, New York, USA.

[28] Wei Fu and Tim Menzies. 2017. Easy over Hard: A Case Study on Deep Learning. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017)*. Association for Computing Machinery, New York, NY, USA, 49âĂŞ60. DOI:http://dx.doi.org/10.1145/3106237.3106256

[29] GitHub. 2018. GitHub REST API. https://developer.github.com/v3/. (2018). Accessed: February 2019.

[30] Theodosis Goudas, Christos Louizos, Georgios Petasis, and Vangelis Karkaletsis. 2014. Argument Extraction from News, Blogs, and Social Media. In *Artificial Intelligence: Methods and Applications*, Aristidis Likas, Konstantinos Blekas, and Dimitris Kalles (Eds.). Springer International Publishing, Cham, 287–299.

[31] Ivan Habernal and Iryna Gurevych. 2017. Argumentation mining in user-generated web discourse. *Computational Linguistics* 43, 1 (2017), 125–179.

[32] James A Herrick. 2017. *The history and theory of rhetoric: An introduction*. Routledge, New York.

[33] Netta Iivari. 2011. Participatory design in OSS development: interpretive case studies in company and community OSS development contexts. *Behaviour & Information Technology* 30, 3 (may 2011), 309–323. DOI:http://dx.doi.org/10.1080/0144929X.2010.503351

[34] Netta Iivari. 2013. 'Configuring the User and the Designer' – A Critical Inquiry on Usability Work in the Company Open Source Software Development Context. In *Nordic Contributions in IS Research*, Margunn Aanestad and Tone Bratteteig (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–17.

[35] Thorsten Joachims. 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the 10th European Conference on Machine Learning (ECML'98)*. Springer-Verlag, Berlin, Heidelberg, 137–142. DOI: http://dx.doi.org/10.1007/BFb0026683

[36] Thorsten Joachims. 2006. Training Linear SVMs in Linear Time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*. ACM, New York, NY, USA, 217–226. DOI: http://dx.doi.org/10.1145/1150402.1150429

[37] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. 2007. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering* 160 (2007), 3–24.

[38] Effie Lai-Chong Law and Ebba Thora Hvannberg. 2008. Consolidating Usability Problems with Novice Evaluators. In *Proceedings of the 5th Nordic Conference on Human-computer Interaction: Building Bridges (NordiCHI '08)*. ACM, New York, NY, USA, 495–498. DOI:http://dx.doi.org/10.1145/1463160.1463228

[39] Elias J MacEwan. 1898. *The essentials of argumentation*. DC Heath & Company, Boston, USA.

[40] Jim D Mackenzie. 1979. Question-begging in non-cumulative systems. *Journal of philosophical logic* 8, 1 (1979), 117–133.

[41] Raquel Mochales and Marie-Francine Moens. 2011. Argumentation mining. *Artificial Intelligence and Law* 19, 1 (2011), 1–22.

[42] David Nichols and Michael Twidale. 2003. The Usability of Open Source Software. *First Monday* 8, 1 (2003). DOI:http://dx.doi.org/10.5210/fm.v8i1.1018

[43] David Nichols and Michael Twidale. 2006. Usability processes in open source projects. *Software Process: Improvement and Practice* 11, 2 (mar 2006), 149–162. DOI:http://dx.doi.org/10.1002/spip.256

[44] Jakob Nielsen. 1993. *Usability Engineering*. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA.

[45] Mie Nørgaard and Rune T. Høegh. 2008. Evaluating Usability: Using Models of Argumentation to Improve Persuasiveness of Usability Feedback. In *Proceedings of the 7th ACM Conference on Designing Interactive Systems (DIS '08)*. ACM, New York, NY, USA, 212–221. DOI: http://dx.doi.org/10.1145/1394445.1394468

[46] James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates* 71, 2001 (2001), 2001.

[47] David Martin Powers. 2011. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies* 2, 1 (2011).

[48] Mikko Rajanen and Netta Iivari. 2015. Power, Empowerment and Open Source Usability. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 3413–3422. DOI: http://dx.doi.org/10.1145/2702123.2702441

[49] Sarah Rastkar, Gail C. Murphy, and Gabriel Murray. 2010. Summarizing Software Artifacts: A Case Study of Bug Reports. In *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1 (ICSE '10)*. ACM, New York, NY, USA, 505–514. DOI: http://dx.doi.org/10.1145/1806799.1806872

[50] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. 2003. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning (ICML'03)*. AAAI Press, Washington, DC, USA, 616–623. http://dl.acm.org/citation.cfm?id=3041838.3041916

[51] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24, 5 (1988), 513–523.

[52] Christos Sardianos, Ioannis Manousos Katakis, Georgios Petasis, and Vangelis Karkaletsis. 2015. Argument Extraction from News. In *Proceedings of the 2nd Workshop on Argumentation Mining*. Association for Computational Linguistics, Denver, CO, 56–66. DOI: http://dx.doi.org/10.3115/v1/W15-0508

[53] Jeff Sauro and Joseph S. Dumas. 2009. Comparison of Three One-question, Post-task Usability Questionnaires. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1599–1608. DOI: http://dx.doi.org/10.1145/1518701.1518946

[54] Fabrizio Sebastiani. 2002. Machine Learning in Automated Text Categorization. *ACM Comput. Surv.* 34, 1 (March 2002), 1–47. DOI: http://dx.doi.org/10.1145/505282.505283

[55] Christian Stab and Iryna Gurevych. 2014. Annotating Argument Components and Relations in Persuasive Essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, 1501–1510. https://www.aclweb.org/anthology/C14-1142

[56] Christian Stab and Iryna Gurevych. 2017. Parsing Argumentation Structures in Persuasive Essays. *Computational Linguistics* 43, 3 (Sept. 2017), 619–659. DOI:http://dx.doi.org/10.1162/COLI_a_00295

[57] Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of language and social psychology* 29, 1 (2010), 24–54.

[58] Michael Terry, Matthew Kay, and Ben Lafreniere. 2010. Perceptions and practices of usability in the free/open source software (FoSS) community. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, New York, New York, USA, 999–1008. DOI: http://dx.doi.org/10.1145/1753326.1753476

[59] Stephen E. Toulmin. 2003. *The Uses of Argument* (2 ed.). Cambridge University Press, Cambridge, UK. DOI: http://dx.doi.org/10.1017/CBO9780511840005

[60] M.B. Twidale and D.M. Nichols. 2005. Exploring Usability Discussions in Open Source Development. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. IEEE, Big Island, HI, USA, 198c–198c. DOI: http://dx.doi.org/10.1109/HICSS.2005.266

[61] Anthony J. Viera and Joanne M. Garrett. 2005. Understanding interobserver agreement: the kappa statistic. *Family Medicine* 37.5 (2005), 360–363.

[62] Yiming Yang. 1999. An Evaluation of Statistical Approaches to Text Categorization. *Inf. Retr.* 1, 1-2 (May 1999), 69–90. DOI: http://dx.doi.org/10.1023/A:1009982220290

[63] Tae Yano, Philip Resnik, and Noah A. Smith. 2010. Shedding (a Thousand Points of) Light on Biased Language. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk (CSLDAMT '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 152–158. http://dl.acm.org/citation.cfm?id=1866696.1866719

[64] Nor Shahida Mohamad Yusop, John Grundy, and Rajesh Vasa. 2017. Reporting Usability Defects: A Systematic Literature Review. *IEEE Transactions on Software Engineering* 43, 9 (sep 2017), 848–867. DOI: http://dx.doi.org/10.1109/TSE.2016.2638427

[65] René Zeelenberg and Diane Pecher. 2015. A method for simultaneously counterbalancing condition order and assignment of stimulus materials to conditions. *Behavior Research Methods* 47, 1 (01 Mar 2015), 127–133. DOI: http://dx.doi.org/10.3758/s13428-014-0476-9