Statement on Research

Preetha Chatterjee

My research interests are primarily in software engineering (SE), with an emphasis on improving software engineers tools and environments through data mining, text analysis and machine learning. I am especially interested in mining software repositories to build or improve tools for software engineers and analyzing the availability and quality of information in developer communications. I am also broadly interested in empirical software engineering.

My research takes a significant step to positively impact new research directions on mining previously unexplored resources in SE, and have been published in top venues including ICSE [1, 4] and MSR [3, 5, 6]. I build tools based on evidence from quantitative and qualitative empirical studies, and adapting state-of-the-art techniques from the fields of Natural Language Processing and Machine Learning. I support open science, and have contributed openly available scripts/datasets for reuse by the SE community [3]. In this statement, I present an overview of my PhD research, future research directions, and my experience in industry-academia collaboration in conducting research.

PhD Research Overview

Integrated development environments today include sophisticated program modeling and analyses behind the scenes to support the developer in navigating, understanding, and modifying their code. While much can be learned from the results of static and dynamic analysis of their source code, developers also look to others for advice and learning. As software development teams are more globally distributed and the open source community has grown, developers rely increasingly on written documents for help they might have previously obtained through in-person conversations. My research activities cover (a) conducting empirical studies to analyze the information in written documents of software archives, and (b) designing techniques to mine useful information from the software archives which could be used in building/improving software maintenance and evolution tools.

Empirical Analysis of Software Artifacts

a) Learning about Code Snippet Characteristics in Software Artifacts [8]: Large corpora of software-related artifacts (e.g., blogs, bug reports, emails) offer the unique opportunity to learn from developers' discussion about code snippets. I conducted an empirical study of 12 types of artifacts to investigate: 1) characteristics of the embedded code snippets, 2) kinds of information available across all artifacts, and their frequency and distribution of availability, and 3) textual cues that indicate code-related information, and how the cues differ across artifacts. The analysis shows that, occasionally individual words are adequate cues for a given kind of information (e.g., name of a programming language). More often, a cue requires a phrase or sequence of words (e.g., indication of code clarity: 'see how much cleaner it is!'). Since these phrases are different across information types, automatically detecting different code-related information embedded in the text would require natural language processing and machine learning. The results provide preliminary indications to the software artifact mining research community regarding the document types that would serve as the most fruitful sources for specific kinds of information.

b) Studying Developer Focus on Question and Answer (Q&A) Forums [7]: Although popular Q&A forums such as Stack Overflow serve as a good knowledge resource, the abundance of information can cause developers to spend considerable time in identifying relevant answers and suitable fixes. I conducted an exploratory study to understand how novice software engineers direct their efforts and what kinds of information they focus on within a Stack Overflow post. I qualitatively analyzed the software engineers' perceptions and annotations from a survey involving 400 Stack Overflow posts related to errors and exceptions in Java and C++. The results indicate that software engineers pay attention to only 27% of code and 15-21% of text in a post to understand and determine how to apply the relevant information to their context. The results also discern the kinds of information (e.g., cause of error) prominent in that focus. These results can be leveraged to improve the Q&A forum interface, guide tools for mining forums, and potentially improve granularity of traceability mappings involving forum posts.

c) Understanding the Potential Usefulness and Challenges of Mining Information from Developer Chats [5]: Popular chat platforms such as Slack host public chat communities that focus on specific software development topics such as Python or Ruby-on-Rails. I conducted an exploratory study into the potential usefulness and challenges of mining developer Q&A format chat conversations for supporting software maintenance and evolution tools. The findings of this study indicate significant prevalence of useful information (e.g., API mentions) in

developer chats, which could potentially be used for helping developers and in building mining-based software maintenance tools. The lack of inbuilt formal Q&A mechanisms on chats result in potential mining challenges, such as automatically identifying an accepted answer. Part of this challenge stems from the free-form style of chats, in which a question may be followed by a series of clarification or follow-up questions, and contextual clues must be followed to determine which question is ultimately being answered. We also observed that developers use chats to share opinions on best practices, APIs, etc. Stack Overflow explicitly forbids the use of opinions. The availability of such valuable information in chats may lead to new mining opportunities for software tools.

Mining Source Code Descriptions from Research Articles

Digital libraries of computer science research articles can be a rich source for code examples that are used to motivate or explain particular concepts or issues. I designed a technique to automatically identify natural language descriptions of code segments embedded within articles. Extracting these natural language descriptions alongside code will enable new advances in areas including code-based search, automatic code comment generation, and documentation generation. Mining code descriptions from research articles presents challenges beyond those faced in mining from unstructured documents such as forums, bug reports, and emails. Code segments in research articles are sometimes embedded within the text, but often separated as figures that are located in a different section or different page. Because research articles often contain multiple code segments, it is necessary to associate each code segment with its corresponding natural-language description. I advised two undergraduate mentees to design and evaluate a set of heuristics that address these challenges. We created an automatic tool that produces XML-based representations with markups to associate identified code segments with their corresponding descriptions. Our work takes a step towards unleashing the potential to mine the vast number of computing articles in digital libraries for code segments and extract descriptive information about functionality and properties of segments [6].

Mining Information from Developer Chat Conversations Towards Building Software Maintenance Tools

The emerging trend of increased participation in developer chats (e.g., Slack, IRC) and evidence from my previous empirical studies [5, 8] motivated me to develop techniques to extract useful knowledge available in developers' chat communication channels. Specifically, i) I created and published an openly available dataset of software-related chat conversations, ii) designed approaches towards automatically analyzing the quality of information in chats using supervised machine learning techniques and natural language analysis, and iii) developed automatic techniques to extract opinion-based questions and answers from chats using deep learning architectures.

- Different from many sources of software development-related communication, the information on chat forums is shared in an unstructured, informal, and asynchronous manner. There is no predefined delineation of conversation in chats; multiple questions are discussed and answered in parallel by different participants. Therefore, a technique is required to separate, or disentangle, the conversations for analysis by researchers or automatic mining tools. We customized an existing supervised machine-learning based disentanglement algorithm, and published a publicly available dataset of software-related Q&A chat conversations, curated for two years from three open Slack programming communities [3]. Our dataset consists of 39k conversations, 400k messages, contributed by 12k users. This dataset could potentially facilitate further research on developing software support and maintenance tools, training and designing chatbots for software development activities, etc.

- Assessing the quality of information in a mining source is crucial towards building effective software engineering tools. Since most chat platforms do not contain built-in quality indicators (e.g., accepted answers, vote counts), I developed an approach for automatically determining the quality of developer conversations [2]. First, a data-driven approach was adopted to investigate conversation quality. We conducted a study where 60 software engineers were recruited to judge the quality of 400 conversations. The analysis of the study led to a set of properties that are generally observable about conversations of varying quality. I trained multiple machine learning-based classifiers (e.g., Stochastic Gradient Boosted Trees, Sequential Neural Networks) with a total of 32 features that are closely related to the observed properties of varying quality conversations. Evaluation of 2000 developer conversations indicated that my approach can achieve a precision of 0.82, and recall of 0.90.

This research could advance the field of information mining towards building software maintenance tools by using high-quality information from software development conversations.

- Recognizing the increasing capabilities of conversational artificial intelligence, researchers are working towards building virtual assistants that support SE tasks (e.g., bug repair, pair programming). I explored the feasibility of using developer chats as a resource towards building opinion-providing virtual assistants for software engineers. Sophisticated virtual assistants often integrate specialized instances including dialog management, knowledge retrieval, opinion-mining, and question-answering. Towards that end, I developed an approach to automatically *E*xtract *O*pinion-based Q&A from *Chats* (ChatEO) [4]. ChatEO has two modules: (1) automatic identification of opinion-asking questions (questions that ask for opinions from other chat participants) using a linguistic pattern-based technique, and (2) extraction of participants' answers to opinion-asking questions using a sequence-to-sequence deep learning-based model. ChatEO opinion-asking question identification significantly outperformed existing sentiment analysis and pattern-based techniques. ChatEO answer extraction showed improvement over existing answer sequence labeling technique designed for non-SE artifacts. The better performance in answer extraction is attributed to capturing the context of discussion in chats through Bidirectional Long Short Term Memory (BiLSTM), and using a customized word embedding model. This model does not require complex feature engineering, and could be used to extract answers to other types of questions in SE chats. Beyond reducing developers effort on information gathering, ChatEO could help in increasing developer productivity, improving code efficiency, and building better recommendation systems.

Future Research

Short-term Research Directions:

Improving Developer Communication Quality. With developers' and software tools' heavy reliance on written developer communications, and the open social nature of these sources, the quality of the information being shared is an important concern. Specifically, developers are often concerned with judging the reliability and credibility of information sources. I am interested in developing automated approaches to assess and improve the quality of information shared on developer communication channels with respect to lack of detail, poor software properties, obsolescence, contradicting information, and duplication.

Developing Feedback Mechanisms. To assist software engineers in their daily tasks, my goal is to design and actuate improvements in the developer communication mediums through feedback mechanisms. One possible research approach is to develop bots, i.e. software applications that run automated tasks. Conversational bots act directly on natural language commands and requests, often integrating services that provide a natural language processing backend, such as Google's Api.AI, Facebook's Wit.AI, or IBM Watson. Chatbots could monitor a conversation and participate in various ways, including by posting messages or displaying buttons preconfigured with specific responses. Building such virtual assistants (e.g., chatbots, voice assistants) could improve code quality and developer productivity, and thus will be of immense interest to researchers and industry practitioners.

Personalizing Feedback Mechanisms. Communicating by posing or answering questions on developer forums has a strong personal component, and each participant carries a significant amount of context to a conversation. This context is often a burden to convey in each individual conversation. For example, in asking programming-related questions, the version of the programming language is often a relevant context that needs to be communicated as part of the posed question, as well as the answer. Not communicating this context can lead to additional conversational overhead, which can invalidate initial responses, or result in the exchange of invalid information. For the purpose of personalization of feedback, I plan to extract context from recent development activity (e.g., git commits, IDE interaction) and/or from prior communication (e.g., natural language text).

Long-term Research Directions:

To build better software development and maintenance tools, machine learning and data mining techniques are used to mine knowledge from various developer artifacts (e.g., source code, bug reports, emails, tutorials, developer socio-technical online network). This is known as "software analytics" and is one of the promising research directions in software engineering. The mined knowledge can be used for supporting decisions and

generating insights, i.e., findings, conclusions, and evaluations of software systems and their implementation, composition, behavior, quality, evolution, etc. I plan to continue my research in the field of software analytics; specifically to explore a variety of developer artifacts at a large scale (big software data analysis), extending data analytics solutions to transform the plethora of information available in software artifacts into actionable nuggets of knowledge and tools, useful for both software engineers and researchers. Machine/deep learning provide new tools for addressing software engineering research challenges, and software engineering challenges motivate new machine learning research. I will be particularly interested in conducting research in the intersection of software engineering and machine learning. I am also interested in investigating the principles and practices of Human Computer Interaction (HCI) with a focus on building better tools for software developers and thus improve programmer experience and productivity.

Research Collaborations

I have collaborated and co-authored papers with researchers from both academia (Virginia Commonwealth University) and industry (ABB Corporate Research). I worked on DARPA's Mining and Understanding Software Enclaves (MUSE) project, with the goal of mining different developer communication sources to extract tagged code snippets. I assisted my advisor Lori Pollock and her Co-PI in writing an NSF grant proposal titled "Automatically Enhancing Quality of Social Communication Channels to Support Software Developers and Improve Tool Reliability". This grant was accepted, and a major part of my dissertation is supported by this NSF grant. I plan to continue to expand my research collaborations with industry to find practical problems for my research, evaluate my research in an industrial context, transfer my research into industry, and obtain financial support for my research. At the same time, I also plan to seek collaborations with researchers from academia, and apply for financial support available for research through university, industry and government programs (e.g., NSF CAREER, CCF-SHF).

References

[1] **P. Chatterjee**. Extracting Archival-Quality Information from Software-Related Chats. In *Proceedings of the 42nd International Conference on Software Engineering*, ICSE '20, New York, NY, USA, 2020. Association for Computing Machinery.

[2] **P. Chatterjee**, K. Damevski, N.A. Kraft, and L. Pollock. Automatically Identifying the Quality of Developer Chats for Post Hoc Use. In *Transactions on Software Engineering and Methodology (TOSEM) (In Submission)*, TOSEM '20, 2020.

[3] **P. Chatterjee**, K. Damevski, N.A. Kraft, and L. Pollock. Software-related Slack Chats with Disentangled Conversations. In *Proceedings of the 17th International Conference on Mining Software Repositories (MSR)*, May 2020.

[4] **P. Chatterjee**, K. Damevski, and L. Pollock. Automatic Extraction of Opinion-based Q&A from Online Developer Chats. In *Proceedings of the 43rd International Conference on Software Engineering*, ICSE '21, 2021.

[5] **P. Chatterjee**, K. Damevski, L. Pollock, V. Augustine, and N.A. Kraft. Exploratory Study of Slack Q&A Chats as a Mining Source for Software Engineering Tools. In *Proceedings of the 16th International Conference on Mining Software Repositories (MSR)*, May 2019.

[6] **P. Chatterjee**, B. Gause, H. Hedinger, and L. Pollock. Extracting Code Segments and Their Descriptions from Research Articles. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 91–101, May 2017.

[7] **P. Chatterjee**, M. Kong, and L. Pollock. Finding Help with Programming Errors: An Exploratory Study of Novice Software Engineers' Focus in Stack Overflow Posts. In *Journal of Systems and Software (JSS)*, JSS '20, 2020.

[8] **P. Chatterjee**, M. A. Nishi, K. Damevski, V. Augustine, L. Pollock, and N. A. Kraft. What Information About Code Snippets is Available in Different Software-related Documents? An Exploratory Study. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 382–386, Feb 2017.