Study Notes for Software Engineering II

Data-Driven Software Engineering

- **Big Data in Software Engineering:**
    - Large amounts of artifacts generated during software development.
    - Increased data from software archives in open-source projects.

Software Data Types

- **Source Code:**
    - Found on platforms like GitHub.
    - Analyzed via automatic parsing and understanding, modeled using program analysis tools.
    - Challenges: Various languages, complexities, scales, systems, programming styles.

Applications of Source Code Analysis

- Code recommendation.
- Clone detection.
- Code completion.

CodeHow: Code Search Enhancement

- Combines API recommendation with source code search.
- Utilizes API documentation to identify relevant APIs for a query.

Techniques for Code Analysis

- **Vector Space Model (VSM):** Measures similarity between a query and API docs using TF-IDF (Term Frequency-Inverse Document Frequency).
- **API-oriented search engines:** Extends queries with relevant APIs for code snippet retrieval.
- **Text Preprocessing:**
    - Tokenization: Segments text into words.
    - Stop-word Removal: Eliminates common but low-value words.
    - Stemming: Reduces words to their root form.

Code Preprocessing

- Regarded as NLP, uses static analysis, parsing, identifier extraction, split, and expansion.

Document Representation

- **Bag-of-Words Model:** Ignores word order, focuses on word frequency.
- **Vector Embedding:** Determines word similarity and defines relations and properties.
- **BERT & CodeBERT:**
    - Pre-trained models for NL-PL applications.
    - Contextual word embeddings.

Clone Detection

- **Code Clone:** Refers to similar or identical code fragments.
- **Types of Clones:**
    - Type-1: Exact copies excluding whitespace and comments.
    - Type-2: Structurally identical with variable names variations.
    - Type-3: Syntactically similar with possible statement alterations.
    - Type-4: Different syntax but similar semantics.

Code Completion

- **Statistical Language Models:** Predicts next word based on previous n words.
- **Deep Learning Models:** GPT series, CodeBERT, focus on predicting the next word with all previous context.

Issue Reports

- Bug and feature requests encountered by users, containing descriptions, reproduction steps, severity, affected parts, and failure traces.
- Repositories such as GitHub, Jira, Bugzilla.

Bug Localization Techniques

- **Modeling Similarity of Bug Reports:**
    - Uses information retrieval techniques like VSM with tfidf.
    - Considers the bug proneness of files based on history.
    - Relates current reports to previous ones for similarity.

Developer Recommendation for Issue Fixing

- Leverages activity profile and expertise (commit, ownership, review history).
- Topic modeling to match developer expertise with bug report topics.

Topic Modeling

- **Latent Dirichlet Allocation (LDA):**
    - Document represented as a mixture of topics.
    - Topics represented as a mixture of words.
    - Used for dimensionality reduction, data exploration, and linking related terms.

References & Acknowledgements

- Credits to Dr. David Lo, Dr. Lingxiao Jiang, Dr. Jiawei Han, Dr. Emily Fox, and Carlos Guestrin.
- Research papers and tools mentioned in slides.

Source Code Leveraging

- Applications:
    - Code recommendation

  - Code completion

Artifact: Issue Reports

- Components:
  - Description, steps, severity, affected parts, failure traces

Bug Localization

- Introduction:
  - Locating bug-related files in large programs
- Techniques:
  - Similarity modeling
  - Bug proneness modeling
  - History-based bug prediction
- Similarity Modeling:
  - Vector space model, tf-idf weighting
- Bug Proneness:
  - Recent bug occurrences predict future bugs

Model & Techniques

- Model:
  - Similarity of bug reports and files/methods
  - Previous bug report similarity
- Techniques:
  - Capture file frequency, commit time
  - Avoiding duplicate bug reports
- Hybrid Techniques

Applications & Recommendations

- Applications:
  - Recommending developers to fix bugs
  - Predicting bug severity

Leveraging Expertise

- Techniques:
  - Leverage activities profile
  - Leverage expertise using topic models
- Matching developers with bug reports based on expertise

Stack Overflow Data Mining

- Data Types:
  - Text, code snippets, votes, tags
- Leveraging Crowd Knowledge:
  - Mining SE crowd knowledge data

Source Code Reuse from Stack Overflow

- Code Reuse:
    - Example snippet usage in GitHub projects
- Identifying Reused Code:
    - Determining reused code snippets from SO

Large-Scale Survey & Study

- Exploratory Study:
    - Analyzing code reuse, modification frequency
- Survey Findings:
    - % of reused code, difficulties faced

Challenges & Solutions

- Challenges Faced:
    - Code quality, outdated code
- Proposed Solutions:
    - Integrated validator, answer quality improvement

WordNet in SE

- Importance:
    - Mitigating data sparsity issues
- Application:
    - Identifying software-specific terms
    - Leverage Stack Overflow to identify word regularities

Approach & Data Collection

- Data Sources:
    - Stack Overflow (specific corpus)
    - Wikipedia (domain agnostic corpus)
- Pre-processing:
    - Text cleaning, tokenization, phrase detection
- Building Vocabulary:
    - Identifying software-specific terms in the corpus

Class Notes

Slide: 1

Title: Software Engineering II Data-driven Software Engineering -2
SubTitle: Lecturer: Shaowei Wang
Email: shaowei@cs.umanitoba.ca

Slide: 2

Title: Administrative item

- Sprint 4's deadline is postponed to Dec 1st (Friday)
- Final project deliverable is due on Dec 15th
- Final exam Schedule: Dec 13, 6 pm. Final Exam Schedule
- Scope: Lecture 1 - 12 (including fuzz testing, excluding data-driven SE)
- Question type: multi-choice + problem-solving
- Examination format: In-person. You will be able to download a digital copy of the test (word file) from UMlearn and submit it to UMLearn
- Guideline: The exam is open book. You are allowed to access class notes, textbooks, and slides. However, no search online and the use of generative AI (e.g., ChatGPT) is not allowed

Slide: 3

Source Code

- Issue Report
- Question Answering Pairs: Stack Overflow
- Pull Requests
- Traces/Logs
- What to mine?
- Mobile apps marketplace
- ...

Slide: 4

Leveraging source code

- Code recommendation
- Code completion
- ...
  Title: Applications

Slide: 5

Source Code

- Issue Report
- Question Answering Pairs: Stack Overflow
- Pull Requests
- Traces/Logs
- What to mine?
- Mobile apps marketplace
- ...

Slide: 6

Title: Artifact: Issue Reports
People report bugs and feature requests that they encounter in the field
Information possibly in issue reports:

- Description of the issue
- Steps to reproduce the issue
- Severity level (priority?)
- Parts of the system affected by the issue
- Failure traces

Slide: 7

Title: Artifact: Issue Reports
Various kinds of issue repositories

- GitHub
- Jira
- Bugzilla
- Etc.

Slide: 8

Title: Applications
Bug localization – given a bug report, identify the bugs in the source code.

Slide: 9

Title: Bug localization: Introduction
Program is often large with thousands of files.
Given a bug report, how to locate files responsible for the bug?
A (semi) automated solution is needed.

Slide: 10

Title: Bug Localization: Technique
Model the similarity of bug reports and files
Model the bug proneness of files
Number of bugs in a file (based on its history)
Model the similarity of previous bug reports
Wang S, Lo D. Version history, similar report, and structure: Putting them together for improved bug localization. In Proceedings of the 22nd International Conference on Program Comprehension 2014 Jun 2 (pp. 53-63).

Slide: 11

Title: Model the similarity of bug reports and files
Information retrieval techniques to calculate the similarity between reports and files
Vector space model – tfidf weighting

Slide: 12

Title: Pre-processing code
Vector_q [1,0.4, 0.2,…]
Vector_f1 [0.2,0.4, 0.6,…]
Vector_f2 [0.2,0.4, 0.6,…]
Vector_f2 [0.2,0.4, 0.6,…]
Text pre-processing

Slide: 13

Vector_q [1,0.4, 0.2,…]
Vector_f1 [0.2,0.4, 0.6,…]
Vector_f2 [0.2,0.4, 0.6,…]
Vector_f2 [0.2,0.4, 0.6,…]
Text pre-processing
Similarity
Title: Pre-processing code

Slide: 14

Title: VSM: Retrieval
Represent documents and queries as vectors
Compute the similarity between the vectors
Cosine similarity is normally used:

Return top-k most similar documents
qi is the tf-idf weight of term i in the query
di is the tf-idf weight of term i in the document

Slide: 15

Title: Weighting on different components
log
Stack trace

Slide: 16

Title: Model the similarity of bug reports and files
Information retrieval techniques to calculate the similarity between reports and files
Frequency based
Word counts
Term frequency – inverse document frequency weighting
Machine learning model-based
Word2vector
Doc2vector
BERT

Slide: 17

Title: Bug Localization: Technique
Model the similarity of bug reports and files/methods
Model the bug proneness of files
Number of bugs in a file (based on its history)
Model the similarity of previous bug reports

Slide: 18

Title: Model the bug proneness of files
Observation: a file that occurs a bug recently is more likely to happen bugs again.
Similar bugs usually happen in burst and not in isolation.
Predicting Faults from Cached History, by Kim et al.

Slide: 19

Title: Capture frequency and time of a file being committed
Time window
Commits in k days
Target file
number of days that has elapsed between a commit c and the input bug report.

Slide: 20

Title: Bug Localization: Technique
Model the similarity of bug reports and files/methods
Model the bug proneness of files
Number of bugs in a file (based on its history)
Model the similarity of previous bug reports

Slide: 21

Title: Model the similarity of previous bug reports
Observation: similar bugs are likely to happen in similar files
Check out bug report before submitting a new bug report, to avoid duplicate.

Slide: 22

Slide: 23

Title: Model the similarity of previous bug reports

Slide: 24

Title: Hybrid technique

Slide: 25

Title: Other applications
Recommend developer to fix bugs

Predict the severity of bugs
Etc.

Slide: 26

Title: Recommend developer to fix bugs: Introduction

Slide: 27

Leverage activities profile
Ownership, review, resolve in the bug track system
Leverage expertise
Topic model
Title: Recommend developer to fix bugs: Technique

Slide: 28

Title: Find similar bug reports
The fixer of the similar bugs
Reviewer of the similar bugs

Slide: 29

Leverage activities profile
Review, commit, ownership, resolve in the bug track system
Leverage expertise
Topic model
Title: Recommend developer to fix bugs: Technique

Slide: 30

Title: Topic Modeling: Black-Box View
Model a document as a probability distribution of topics
A topic is a probability distribution of words
Most simple and common :LDA

Slide: 31

Title: Usage of Topic Modeling
Dimension

ality reduction: words -> topics
Help with data exploration.
Able to link a document and a query
Do not have share any words
Share related words of the same topics

Slide: 32

(Gp:) Developer
comments
Fixed bugs
Developed project/component/files
Tags
Title: Leverage expertise

Slide: 33

Title: Leverage expertise
Model expertise of developers using topic model
Match developers and bug reports based on their expertise
Topic matching
Calculate similarity

Slide: 34

Slide: 35

Title: Various types of data on Stack Overflow
Text description
Code snippets
votes
Tags

Slide: 36

Title: Mining & leveraging big SE crowd knowledge data
Mining big SE crowd knowledge data
Stack Exchange Academic Papers

Slide: 37 to Slide: 39

Slide: 40

Title: How do developers utilize source code from Stack Overflow?
Empirical Software Engineering Journal(EMSE) 2018
Yuhao Wu et al.

Slide: 41

```
function generateID() {
    return "avalon" +
        Math.random().toString(36).substring(2, 15) +
        Math.random().toString(36).substring(2, 15)
}
```

Title: Developers reuse source code from S.O. posts
Source code in a GitHub project
An answer on Stack Overflow

Slide: 42

Title: How we can know a code snippet was reused from SO

Slide: 43

Title: Structure of this study

- Part I: Exploratory Study
- Part II: Survey
  How do developers reuse source code?
  How do you think S.O. can be improved?
  Responses

Slide: 44

Title: Part I: Exploratory study
searchcode.com
“stack_overflow”
4,878 source files
Filter
289 files
Analyze code reuse
RQ1: modification frequency and reason
RQ2: origin of code reuse

Slide: 45

Only 21% of the code are copied exactly
32% of the reused source code snippets require additional modification (C2+C3)
Developers adopt answers based on different needs: simplicity, correctness, efficiency, etc.
Title: 26% of the source code was reused from non-accepted answers

Slide: 46

43%
26%
1%
30%
An advanced tagging system might be helpful
Title: 26% of the source code was reused from non-accepted answers

Slide: 47

Title: Part II: Large-scale survey on 400+ participants
Experience

Type of project participated
Difficulties in reusing source code
Opinions on OSS licenses
Usefulness of proposed advanced tagging system
Any other suggestions
Demography (7)
Barriers (10)
Suggestions (2)
Survey Questions

Slide: 48

Title: Most suggestions are on code quality
35%
24%
13%
12%
10%

Slide: 49

Title: Suggestions: improving code quality (35% of all categories)
Integrated validator (42.2%)
Outdated code (29.7%)
Answer quality (17.2%)
Code review (10.9%)

Slide: 50

Beyer, S., Macho, C., Di Penta, M. and Pinzger, M., 2020. What kind of questions do developers ask on Stack Overflow? A comparison of automated approaches to classify posts into question categories. Empirical Software Engineering, 25(3), pp.2258-2301.
Yang, X.L., Lo, D., Xia, X., Wan, Z.Y. and Sun, J.L., 2016. What security questions do developers ask? a large-scale study of stack overflow posts. Journal of Computer Science and Technology, 31(5), pp.910-924.
Ahmed, S. and Bagherzadeh, M., 2018, October. What do concurrency developers ask about? a large-scale study using stack overflow. In Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (pp. 1-10).
Islam, M.J., Nguyen, H.A., Pan, R. and Rajan, H., 2019. What do developers ask about ml libraries? a large-scale study using stack overflow. arXiv preprint arXiv:1906.11940.
Barua, A., Thomas, S.W. and Hassan, A.E., 2014. What are developers talking about? an analysis of topics and trends in stack overflow. Empirical Software Engineering, 19(3), pp.619-654.
Title: Understanding the challenges and problems that developers face

Slide: 51

Title: Mining & leveraging big SE crowd knowledge data
Leveraging big SE crowd knowledge data
Stack Exchange Academic Papers

Slide: 52

Title: We need a thesaurus to contain software-specific terms and commonly-used morphological forms
We have different morphological forms (abbreviations, synonyms and misspellings)

Slide: 53

Title: We need a thesaurus to contain software-specific terms and commonly-used morphological forms
We have existing general thesaurus for morphological forms
Wordnet (https://wordnet.princeton.edu/)
Same word in SE and in other domain may have different meaning

Slide: 54

Title: Why we care about this?
Identify various morphological forms of the same concept may also mitigate data sparsity problems when applying NLP techniques for various applications -> reduce the vocabulary size
Information retrieval
Data mining

Slide: 55

Title: Leverage Stack Overflow to train a model to identify the relational regularities
SEthesaurus: WordNet in Software Engineering IEEE TRANSACTIONS ON SOFTWARE ENGINEERING (TSE) 2015. Xiang Chen, Chunyang Chen, Dun Zhang, and Zhenchang Xing
Underlying intuition is that words with similar meaning would appear in the similar context.

Slide: 56

Title: Overview of the approach
Title: Data collection

Slide: 57

Title: Data collection
Software specific corpus

- Stack Overflow
- 15,930,617 questions and 24,676,333 answers
  Domain agnostic corpus
- Wikipedia
- 8,556,773 articles

Slide: 58

Title: Overview of the approach
Title: Pre-processing

Slide: 59

Title: Pre-processing
Text cleaning
Tokenization
Phrase Detection

Slide: 60

Title: Overview of the approach
Title: Building software-specific vocabulary

Slide: 61

Title: Building software-specific vocabulary
Identify software-specific terms by contrasting the frequency of a term in the software-specific corpus compared with its frequency in the general corpus