Study Notes for Software Engineering II Exam

Administrative Information

- **Lecturer:** Shaowei Wang
- **Contact:** shaowei@cs.umanitoba.ca
- **Final Exam Schedule:**
  - Date: December 13, 6 pm
  - URL for schedule: https://umanitoba.ca/registrar/final-exams/final-exam-schedule
- **Exam Format:**
  - Multi-choice + problem-solving
  - In-person
  - Open book (class notes, textbooks, slides)
  - Digital copy of the test from UMlearn, submit to UMlearn
  - **No internet searches or generative AI tools**

Key Topics Covered

What to Mine in Data-driven SE

- **Sources to Mine:**
  - Source Code
  - Issue Reports
  - Stack Overflow (Q&A Pairs)
  - Pull Requests
  - Traces/Logs
  - Mobile Apps Marketplace

Artifacts in SE

Issue Reports

- **Content may include:**
  - Description of the issue
  - Steps to reproduce
  - Severity level
  - System parts affected
  - Failure traces
- **Repositories:**
  - GitHub
  - Jira
  - Bugzilla

Applications in Software Engineering

Bug Localization

- **Objective:** Identify bugs in source code from a bug report.
- **Challenges:** Large programs with thousands of files.

Modeling for Bug Localization

- **Techniques:**
  - Model similarity between bug reports and files/methods using:
    - Information retrieval techniques (tfidf weighting)
    - Machine learning models (Word2vec, Doc2vec, BERT)
  - Model bug proneness of files based on history.
  - Leverage similarities of previous bug reports.

Developer Recommendations

- Based on:
  - Activity profiles (ownership, review, resolve history)
  - Expertise (topic models)

Topic Modeling in SE

- **Applications:**
  - Dimensionality reduction from words to topics.
  - Data exploration & linking documents/queries with shared topics.
- **Models:**
  - LDA (Latent Dirichlet Allocation)

Stack Overflow (S.O.) in SE Research

- **Mining SE Crowd Knowledge:**
  - Explore posts, code snippets, votes, tags on S.O.
  - Identify challenges in developing deep learning applications.
  - Empirical study on code reuse from S.O. posts.

Code Reuse from S.O.

- **Code Origin:** Determine if code in GitHub projects originates from S.O. posts.
- **Study Structure:** Part I - Exploratory Study. Part II - Survey.
- **Findings:**
  - Not all code is copied exactly; modifications are frequent.
  - Reuse includes non-accepted answers.
  - Developers choose answers based on needs like simplicity, correctness, efficiency.

Thesaurus & Term Semantics

- **Need for a Thesaurus:** Software-specific terms and morphological forms.
- **Word Embeddings:**
  - Skip-gram model

    - FastText model
    - Use cleaned S.O. data to learn word embeddings.

Building Vocabulary

- Contrast term frequency in software-specific corpus vs. general corpus to identify domain-specific terms.

Semantically Related Terms

- Use cosine similarity and string edit distance to extract and discriminate synonyms and abbreviations.

Exam Preparation Tips

- Review all slides and key concepts mentioned above.
- Understand techniques for bug localization, including the usage of VSM, tf-idf, and different machine learning models.
- Study the role of S.O. in software engineering research and how developers utilize S.O. content.
- Familiarize yourself with topic modeling, term semantics, and the approach to building a software-specific thesaurus.

Class Notes

Slide: 1

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

Slide: 2

Title: Administrative item
Other Placeholder:

- 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

Other Placeholder:

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

Slide: 4

Title: Applications
Other Placeholder:

- Leveraging source code
  - Code recommendation
  - Code completion
  - …

Slide: 5

Other Placeholder:

- 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
Other Placeholder:

- 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
Other Placeholder:

- Various kinds of issue repositories
  - GitHub
  - Jira
  - Bugzilla
  - Etc.

Slide: 8

Title: Applications
Other Placeholder:

- Bug localization – given a bug report, identify the bugs in the source code.

Slide: 9

Title: Bug localization: Introduction
Other Placeholder:

- 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
Other Placeholder:

- 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
  - 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
Other Placeholder:

- Information retrieval techniques to calculate the similarity between reports and files
- Vector space model – tfidf weighting

Slide: 12

Other Placeholder:

- 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
  Title: Pre-processing code

Slide: 13

Other Placeholder:

- 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
Other Placeholder:

- Represent documents and queries as vectors
- Compute the similarity between the vectors
- Cosine similarity is normally used:
  - ![Cosine Similarity Equation]

Slide: 15

Other Placeholder:
Title: Weighting on different components

- log
- Stack trace

Slide: 16

Title: Model the similarity of bug reports and files
Other Placeholder:

- 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
Other Placeholder:

- 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
Other Placeholder:

- Observation: a file that occurs a bug recently is more likely to happen bugs again.
    - Predicting Faults from Cached History, by Kim et al.

Slide: 19

Title: Capture frequency and time of a file being committed
Other Placeholder:

- Time window
- Commits in k days
- Target file
- Number of days that have elapsed between a commit c and the input bug report.

Slide: 20

Title: Bug Localization: Technique
Other Placeholder:

Slide: 21

Title: Model the similarity of previous bug reports
Other Placeholder:

- 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

Slide: 24

Title: Hybrid technique

Slide: 25

Title: Other applications
Other Placeholder:

- Recommend developer to fix bugs
- Predict the severity of bugs
- Etc.

Slide: 26

Title: Recommend developer to fix bugs: Introduction

Slide: 27

Other Placeholder:

- 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
Other Placeholder:

- The fix

er of the similar bugs

- Reviewer of the similar bugs

Slide: 29

Other Placeholder:

- 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
Other Placeholder:

- Model a document as a probability distribution of topics
- A topic is a probability distribution of words
  - Most simple and common: LDA

Slide: 31

Other Placeholder:
Title: Usage of Topic Modeling

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

Slide: 32

**- Developer

- Comments
- Fixed bugs
- Developed project/component/files
- Tags**

Slide: 33

Title: Leverage expertise
Other Placeholder:

- Model expertise of developers using a topic model
- Model the required expertise of a bug report using a 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
Other Placeholder:

- Mining big SE crowd knowledge data
    - Link to Papers

Slide: 37

Slide: 38

Slide: 39

References:

- Zhang T, Gao C, Ma L, Lyu M, Kim M. An empirical study of common challenges in developing deep learning applications. In2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE) 2019 Oct 28 (pp. 104-115). IEEE.

Slide: 40

Title: How do developers utilize source code from Stack Overflow?
Other Placeholder:

- Empirical Software Engineering Journal(EMSE) 2018
    - Yuhao Wu et al.

Slide: 41

Other Placeholder:

- 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

- 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
Other Placeholder:

- 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

Other Placeholder:

- 21%
- 32%
- 22%
- 13%
- 35%
  Title: Only 21% of the code is copied exactly

Slide: 46

Other Placeholder:

- 43%
- 26%
- 1%
- 30%
  Title: 26% of the source code was reused from non-accepted answers

Slide: 47

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

- 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

Other Placeholder:

- 35%
- 24%
- 13%

- 12%
- 10%
- 7%
  Title: Most suggestions are on code quality

Slide: 49

Title: Suggestions: improving code quality (35% of all categories)
Other Placeholder:

- Integrated validator (42.2%)
- Outdated code (29.7%)
- Answer quality (17.2%)
- Code review (10.9%)
- “An inbuilt environment for as many languages/environments as possible.”
- “Make date important in marking outdated code, and deprecate those snippets via the community”
- “Better support for answers that are good, but out of date.”
- “In-browser code review and commenting similar to that provided by commercial code review tools.”

Slide: 50

References:

- 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
Other Placeholder:

- Mining big SE crowd knowledge data
  - Link to Papers

Slide: 52

Title: We need a thesaurus to contain software-specific terms and commonly-used morphological