Master Project Market · HS 2017

Nathan Labhart
Academic Coordinator
Master Project: Rules

• The Master Project is a **group project** with two or more members. → Chance of denial for individual projects: 99%

• The Master Project can only be started **after** the Master Basismodul has been completed successfully (only for Major). → Best time: During semester break. Max. 1 year to complete.

• The Master Project must be done **with an IfI professor**.

• You will get **18 credit points**. → Submit a **final report** that concludes your work.
Master Project Market: Procedure

• Groups at IfI prepared projects for you and published them online: http://www.ifi.uzh.ch/en/teaching/studiengaenge/msc/msc-proj.html

• Projects are presented at the Market → ask representatives

• To form groups, go to OLAT http://tiny.uzh.ch/yi → use discussion boards

• Once a group is complete, hand in the application form.
Master Projects
Generating Smart Diffs Through Deep Learning
Carol Alexandru <alexandru@ifi.uzh.ch> | Jürgen Cito <cito@ifi.uzh.ch>

Goal of the Master Project
The outcome of this project should be a taxonomy of change types for different programming languages based on empirical evidence contained in open source repositories as well as the implementation of a change classifier generator.

The main tasks of the project are:
● Find a suitable unsupervised machine learning approach to identify common change types for different programming languages.
● Extract 1000s of projects from GitHub and apply the learning approach to cluster the change types.
● Implement a generator that will recognize these change types for arbitrary programming languages.

Contacts
Carol Alexandru <alexandru@ifi.uzh.ch>
Jürgen Cito <cito@ifi.uzh.ch>

Introduction
While plain-text diffs are a straightforward way of keeping track of changes in a software project, they are poorly suited for understanding those changes. Different semantic changes might be mixed together in a single diff and it is difficult to further process diffs using automated tools.

Approaches like ChangeDistiller extract changes between two revisions based on abstract syntax trees (ASTs) instead of plain text source code. This allows them to recognize semantic changes, like whether specific elements (if conditions, classes, methods, etc.) have been added, removed, modified or even moved to other locations in the source code.

However, there are two existing problems with this idea:
● The change types identified by tools such as ChangeDistiller have been manually crafted by researchers and can seem arbitrary.
● Tools like this need to be implemented separately for each specific programming language.

Since platforms such as GitHub contain the entire histories of millions of software projects, it should be possible to:
● Cluster different types of changes using unsupervised machine learning for any programming language.
● Generate a classifier that can recognize these change types instead of implementing it manually.

Can we use deep learning to synthesize change types from large-scale examples?

GitHub

Extract Raw Changes

Unsupervised learning

Cluster change types (Unsupervised Learning)

Learn program $f_{\text{ProgLang}}$ to classify changes on new code
MSc Project: Financial Crises and Network Structure

Motivation
• Banks create a complex network of interactions.
• Shocks spread through the network and cause financial crises.
• *Portfolio Compression* = Eliminate cycles in the network
  • Idea: Reduce interactions → reduce spread
  • Required by law

Research Question
• Does portfolio compression actually make financial networks safer?

Approach
• Large-scale Monte Carlo Simulations
• Optional: Theoretical guarantees

Prerequisites
• *Required* knowledge:
  • Good programming skills (Python or Java or …)
  • Basic graph theory (e.g., flows on networks)
• *Helpful* knowledge:
  • Solid background in mathematics (linear algebra), reading and writing proofs
  • Scientific computing / numerics
• Participants: 2-3

Supervisors: Prof. Sven Seuken, Steffen Schudenzucker.
Contact: schudenzucker@ifi.uzh.ch
Computation and Economics Research Group. [www.ifi.uzh.ch/ce](http://www.ifi.uzh.ch/ce)
Design and Implementation of a Crypto-Currency

- Blockchains (BC)
- Crypto-Currencies (CC)

Tasks:
- Design & Implementation of a PoSp-based BC and CC
- Design & Implementation of an Android-based wallet
- Test and Evaluation

Goals:
- Secure, Reliable, Fast and Energy Efficient CC

Sina Rafati, Bruno Rodrigues, Burkard Stiller
Communication Systems Group CSG,
Department of Informatics IfI
University of Zürich UZH
rafati|rodrigues|stiller@ifi.uzh.ch
Implementation of FFT Operator in Apache Flink

Muhammad Saad
Database Technology Group (DBTG)
Stream Processing

- Processing of *data in motion*
- Computing on data directly as it is produced or received.
- The application logic, analytics, and queries exist continuously, and data flows through them continuously.
- High throughput: high-velocity, high-volume data can be processed with minimal latency.

Apache Flink

- Apache Flink is a powerful, mature, open source stream processing framework

What you will do?

- Fast Fourier Transform (FFT) Query Operator in Apache Flink
- With Aggregate and Group Clause

```
SELECT ...
FROM FFT(Image [RANGE 1 DAY])
    AREA RA, DECL, Freq, VIS
ORDER BY RA, DECL, Freq
AS R
WHERE ...
```

Sample FFT Query
Development:
- Language: JAVA
- # of Students: 2

Contact:
- Email: saad@ifi.uzh.ch
- Office: BIN 2.E.10
UZH Open Research Data with DDIS
Community-Oriented Semantic Data Management

Using Wikidata’s Infrastructure at UZH
- Machine-readable data
- Wiki-style collaboration
- Editors and Roles
- Qualifiers, Sources

- Querying Knowledge Base (SPARQL)
- Timelines of Discoveries

Contributing to Wikidata (CCO)
- Citations
- Images
- Research Facts
- Author Networks
- Topic Evolution

Query Service  Histropeda  http://www.wikidata.org  wikicite  Scholia  PLAZI
Supporting the feedback communication between end-users and developers for Android apps

- **Fact:** Feedback is a **valuable information** for software evolution.

- **But:** Current feedback solutions often do not allow
  - Feedback senders to comment on feedback sent by others,
  - Feedback receivers to ask questions for clarification,
  - Feedback senders to receive information about the feedback status (e.g., issue solved).

- **Goal:** You develop solutions that
  - Support the discussion and negotiation among feedback senders and between feedback senders and receivers,
  - Inform the feedback senders about the current status of their requests,
  - Encourage end-users to take part in the feedback communication activities (e.g., by using gamification elements).

Norbert Seyff (seyff@ifi.uzh.ch)
Crossplatform FlexiSketch
A flexible editor for diagram sketching, runs on Android + iOS + Mac + Windows

Created with the 2D game framework Corona SDK and the scripting language Lua

Includes multi-user support

Where we want to go
Where we are
➢ Make it pretty
➢ Integration of text documents

 ✓ Done

Find videos and more on www.flexisketch.org
Contact: Dustin Wüest, wueest@ifi.uzh.ch
Test Case Coverage Visualization

• Different coverage metrics (line, statement, branch, path) → how well-tested is your app?
• Part of CI environment
• Detached from source code and given in percent (e.g., 80% of classes)
• No information about how well-tested coherent/coupled parts are
• No support for exploring the parts that need more testing

Hard Facts:
• # of Students: 2
• Contact: Giovanni Grano (grano@ifi.uzh.ch) and Christoph Laaber (laaber@ifi.uzh.ch)

1. State of the Art of Software Visualization

2. Design Map-like Visualization

3. Cluster Software through Static Analysis

4. Test Case Coverage Visualization

5. CI/IDE Integration & Source Code Navigation (jump to file, zoom, etc.)
Runtime Code Injection

Background
Experimentation is about testing new functionality on a small fraction of the user base in production environments (e.g., 1% of users)

Goal
Extract code changes directly in the IDE and inject them into the running application without recompiling and redeploying it

Test these code changes immediately in production on a small fraction of the user base and provide feedback within the IDE (e.g., visualize metrics)

Contact Gerald Schermann: schermann@ifi.uzh.ch


2 - 4 students
BenGen: Automatic Performance Test Suite Generation

**PROBLEM**

- Hard to write -> requires knowledge of (dynamic) compiler internals/optimizations
- No clear understanding, no best practices
- No (to few) standard libraries -> Java has JMH since v1.7
- Hardly anyone writes benchmarks -> in 2015 ~ 30 Github Projects w/ at least 1 JMH commit

**FACTS**

- # of Students: 2
- For Java and/or Go
- Also available as Master Thesis
- Contact: Christoph Laaber (laaber@ifi.uzh.ch)
Research direction to Support Code Reuse

Once a new version of a software library is released

1) Someone can be interested to import this library
   Question:
   - When a new version of the imported library is released can we use the new version of this library?

   Answer:
   - Yes if the API used by our project are not deprecated

   **Solution:** Mining Software Repositories to build a recommender that suggest whether it is a valid choice to update the version of the used library
Perf-CoRe: Performance Code Review

Are Performance Bugs detected during Code Review?
- Code Reviews Mining (GerritHub)
- Performance Reviews Validation (Profiler)

Does Code Review help developers in fixing Performance Bugs?
- Extraction of Versions affected by Performance Bugs (discovered during Code Review)
- Code Performance metrics BEFORE and AFTER the Code Review

How can we support Developers during code review to fix Performance Bugs?
- Code Review Augmentation with dynamic Performance metrics/information

Contact:
vassallo@ifi.uzh.ch
laaber@ifi.uzh.ch

Master Project Description available at: http://www.ifi.uzh.ch/en/seal/teaching/master/projects/Perf-CoRe.html
Annostand
Understanding how people create and use annotations in reading research articles

People highlight and scribble while they read. These annotations are found to aid in interpretation, understanding, and remembering. But too many highlights can be distracting when the reader re-read or find important information from the material. Can the computer help readers separate important annotations from spurious ones?

In this project, you will:
• Analyze types and purpose of annotations from a corpus of annotated paper and videos of annotations
• Design and conduct an eye tracking study to investigate creation and usage of annotations
• Implement and evaluate a software to classify and summarize annotations

Suitable for 2-3 students (as a master project)
Supervisor: Prof. Dr. Chat Wacharamanotham (Interaction Design)

Info: zpac.ch/projects
Through social networks, individuals participate in the diffusion of today’s news. Yet many share news without fully reading them.

In this project, you will
1) develop a social RSS reader app that:
✓ allows users to like, share and save their news within the app,
✓ links the app with the user’s facebook account and shows others that like, share and save these news,
✓ tracks the user’s interactions with the app,
2) conduct a small scale study to test users’ news consumption behavior within this app.

Suitable for 2-3 students (as a master project)

Supervisors:
Prof. Dr. Chat Wacharamanotham (Interaction Design)
Prof. Dr. Anne Scherer and Andrea Bublitz (Quantitative Marketing)

Info: zpac.ch/projects
A system for providing personalized feedback for a large number of study participants

Ambulatory assessment is a research method that allows for assessing psychological processes like emotions in the context of daily life. Study participants often want to get personalized feedback about their results, which is not possible in most cases due to a lack of automatized options.

In this project you will
- get an insight in ambulatory assessment studies of elderly at the UZH University Research Priority Program “Dynamics of Healthy Aging”
- develop a html-based flexible tool for researchers who want to give personalized feedback to study participants

Suitable for 2 students (as a master project)

Supervisors:
Prof. Dr. Chat Wacharamanotham (Interaction Design)
Dr. Andrea B. Horn (Psychology)
WISE Web-based Interdisciplinary Symptom Evaluation

- Evaluating the existing WISE questionnaire system
- Designing & implementing the summary report page
- Designing & implementing an Android mobile app or Android wear smartwatch app for capturing pain over time

Suitable for 3–4 students (as a master project)

Supervisors:
Prof. Dr. Chat Wacharamanottham (IFI),
PD Dr. Dominik Ettlin, and
lic. phil. Beat Steiger (Dental Medicine)

Info: zpac.ch/projects
Description

• Tool for sustainability assessment
  • Identify challenges and values
  • Institutional focus: UZH level
• 10 categories with each at least 3 indicators
• Rating system for indicators

Requirements

• Intuitive UI
• Upload of data to CMS
• Different user roles
• Different modes (expert/light)
• Dynamic/extendable content
• Currently common browsers
• Visualization (e.g. Spider web diagram)
• Export data as file/image

Advisors:
Prof. Dr. Lorenz Hilty
Dr. Clemens Mader
Department of Informatics – Informatics and Sustainability Research Group
• Start Date: ASAP
• Finish Date: 6 months (or less)
• Technologies not determined yet
  • Frameworks: E.g. Django (Python), JavaScript (Angular, Vue), HTML
  • SQL-Database: MySQL,
  • Host: External, e.g. hostpoint.ch

Questions? Interested? Send me a email: juerg.bargetze@uzh.ch
Visualization Toolkit Evaluation

• Evaluate several different visualization toolkits for their usability and deployability on a computer graphics cluster and display wall
  ‣ VisIt and ParaView/VTK

• Evaluate and demonstrate the feasibility of integrating custom graphics rendering or data visualization into the visualization toolkit’s output
  ‣ Combine toolkit display with separately generated image data

mthoeny@ifi.uzh.ch
ReviewNG: The Next Generation code review tool

Project vision
Devise the next code review tool, to really support developers in this hard and time consuming task

Current code review tools are primitive!
They do not support developers with the most important tasks of a review:
- understanding the code under review
- detecting the most problematic parts, quickly
- providing high quality comments

What are you going to learn and do in this project?
- learn about the problems of code review and elaborate new ideas to solve them
- learn a new exciting programming language and IDE — Smalltalk Pharo
- develop a full fledged code review tool, with the Pharo community, implementing your ideas
- design and conduct an experiment to validate your work with real-world developers

Suitable for 2-4 students (as a master project)
Supervisor: Prof. Dr. Alberto Bacchelli (Empirical Software Engineering)
Project vision
Expand CodeCity (the most famous software visualization tool) with information about connection between artifacts

CodeCity is a great software visualization tool, but it is not complete yet!

CodeCity cannot show information about:
- connections between classes
- clusters of artifacts that are related
- dynamic behavior

What are you going to learn and do in this project?
- learn about Code City and its implementation
- learn about information visualization techniques to overlay clustering information on maps
- develop an extension to an existing version of CodeCity, or create your own version
- design and conduct an experiment to validate your work with real-world developers

Suitable for 2-4 students (as a master project)
Supervisor: Prof. Dr. Alberto Bacchelli (Empirical Software Engineering)
What was your source code comment or email about?

**Project vision**

Devise an online service to automatically classify code comments and development emails, to help software developers

**Development Emails and Code Comments** are hard to parse!

They are made of many different languages and have several types of meanings:
- natural language is interleaved with code, patches, stack traces
- there is a lot of noise to be removed

**What are you going to learn and do in this project?**
- learn about information retrieval and advanced parsing techniques
- learn about machine learning algorithms for text classification and how to mix them with parsing
- develop an online service that developers and researchers can use to automatically classify thousands of emails or code comments in seconds
- design and conduct an experiment to validate your work with real-world developers

Suitable for 2-4 students (as a master project)

Supervisor: Prof. Dr. Alberto Bacchelli (Empirical Software Engineering)
Using computer interaction tracking to estimate where people look while reading research articles and software code

Eye and mouse have similar movement patterns when reading results from web search

How much does this similarity exist in reading research articles and code review?

In this project, you will:
• learn about eye and computer interactions tracking
• extend applications (in Java and Javascript) to track computer interactions and include eye movement data stream
• design and conduct a lab study and analyze data

Figure: Navalpakkam et al. (2013)

Suitable for 2-4 students (as a master project)

Supervisors:
Prof. Dr. Alberto Bacchelli (Empirical Software Engineering)
Prof. Dr. Chat Wacharamanotham (Interaction Design)

Info: zpac.ch/projects
Existing Task Planning Tools

Tasks are characterised by urgency, importance, tags, sub tasks, ...

Happy Planner

Adds emotions to tasks

Contains features to tackle annoying tasks

Contact:  Manuela Züger (zueger@ifi.uzh.ch)
Sebastian Proksch (proksch@ifi.uzh.ch)
Giovanni Grano (grano@ifi.uzh.ch)
Recommender System for Libraries Update of Client Projects

Sebastiano Panichella

Software Ecosystem:
“Groups of *software projects* that are developed and *co-evolve* in the same environment…”

Popular examples: Eclipse plug-ins, the R ecosystem, Linux distributions, Apache projects

Main Characteristics: “*share code, depend on one another, reuse the same code, are built on similar technologies*…”
Problem when Managing Software Ecosystems…

Manage dependencies in large ecosystems is an intricate task…

Releases “ignored” by client projects

69%

Client projects tend to upgrade their dependencies when substantial changes in the projects they depend on are released, including bug-fixes.

Client projects tend to postpone an upgrade when public services are deleted from the projects they depend on.
Goal of the Master Project
Software Ecosystems…

Long-term Goal

Develop a smart upgrade management system able to support the developer in the decision of whether to (i) perform the upgrade immediately, (ii) postpone it, or (iii) even avoid to perform it.

Number of required students for the project: 2-3
Interested in a project? Talk to representatives and form groups!

http://tiny.uzh.ch/yi

Good luck with your Master Project 😊