Conceptualization and Development of ML-based Recommender Systems for Software Engineering

Candidate: Claudio Di Sipio
Advisor: Prof. Davide Di Ruscio

Co-Advisors:
Dr. Juri Di Rocco
Dr. Phuong Thanh Nguyen

Università degli studi dell’Aquila / Italy
XXXV Doctoral cycle
a.a. 2021/2022
About me

- 2016-2018: Master degree at University of l’Aquila
- 2018-2019: Research fellowship at Univaq
  - API recommendation by using code cloning techniques
- 2019-2023: Ph.D. in Computer Science

Hobbies: running, play tennis, writing novels ☺
Main research topics

- **Recommender system for software engineering**
  - Categorization of OSS repositories
  - API function call recommendations

- **Model-Driven engineering**
  - Modeling assistants
  - MDE approach to design recommender systems

- **Quality aspects in recommender systems**
  - Adversarial attacks
  - Bias and fairness
Research group

Davide Di Ruscio
Associate professor

Juri Di Rocco
Assistant professor

Phuong T. Nguyen
Assistant professor

Riccardo Rubei
Post-doc researcher
Collaborations

Massimiliano Di Penta
Full professor
University of Sannio

Cezar Sas
Ph.D. student
University of Gronigen

Andrea Capiluppi
Associate professor
University of Gronigen

Leonardo Mariani
Full professor
University of Milano-Bicocca
Recommender system in a nutshell

Users

Items
Recommender systems* (RS) are complex software system that comes in handy to alleviate the burden of choice in different domains.

Recommender systems in software engineering** (RSSEs) has been defined.

---

RSSE phases as described by Robillard et. al**

---


RSSEs: Conceptual framework

**Data Preprocessing**

- Knowledge Base
  - Lookup/Store

**Mining and Analysis Tools**

- Source Code Miner
- NLP Miner
- Modeling Artifact Encoder
- Configuration Miner
- Cross project Analysis
- Encoder-Decoder Architecture

**Data sources**

- Source Code
- Natural language channels
- Configuration Scripts
- Modeling Artifacts

**Capturing Context, Producing Recommendations, Presenting Recommendations Phases**

- Developer IDE
  - query
  - recommendations
- Knowledge Base
- Data Storage
Categorizing GitHub projects
Conceptual map

Categorizing OSS projects

MNBN

HybridRec

Conceptualization and development of RSSEs
Setting the context

In 2017, GitHub introduced *topics* to help developers increase the reachability of their repositories.

Assigning wrong labels can compromise the discoverability.

As first attempt, we develop an automatic approach based on Multinomial Naive Bayesian network (MNBN).


Motivating example

Bootstrap GitHub projects and its topics
Proposed solution: HybridRec*

Evaluation metrics

\[
\text{success rate} \@ N = \frac{\text{count}_{p \in P}(\left| GT(p) \cap \bigcup_{r=1}^{N} \text{REC}_r(p) \right| > 0)}{|P|}
\]

\[
\text{precision} \@ N(p) = \frac{\sum_{r=1}^{N} \left| GT(p) \cap \text{REC}_r(p) \right|}{N}
\]

\[
\text{recall} \@ N(p) = \frac{\sum_{r=1}^{N} \left| GT(p) \cap \text{REC}_r(p) \right|}{|GT(p)|}
\]

\[
\text{Top rank} = \frac{\text{TpRank}(r)}{|R|} \times 100\
\]

\[
\text{coverage} \@ N = \frac{\left| \bigcup_{p \in P} \bigcup_{r=1}^{N} \text{REC}_r(p) \right|}{|I|}
\]
Evaluation process

Input data preparation

- Data extraction
  - Dataset
  - Split ten-fold

Split labels
- Query labels
  - GT labels

Testing data
- Training data

Recommendation production

- ST
- CF
  - ST labels
  - CF labels

HybridRec labels
- HybridRec

Outcome evaluation

Comparison
- Success rate
  - Top rank
  - Precision
  - Recall
## Results

<table>
<thead>
<tr>
<th></th>
<th>D₁</th>
<th>D₂</th>
<th>D₃</th>
</tr>
</thead>
<tbody>
<tr>
<td># of artifacts</td>
<td>11,694</td>
<td>6,253</td>
<td>5,620</td>
</tr>
<tr>
<td># of topics</td>
<td>19,337</td>
<td>455</td>
<td>6,442</td>
</tr>
<tr>
<td>Avg. number of topics</td>
<td>8.24</td>
<td>6.70</td>
<td>8.60</td>
</tr>
<tr>
<td>Avg. frequency of topics</td>
<td>16.13</td>
<td>42.10</td>
<td>29.90</td>
</tr>
</tbody>
</table>

![Success Rate Chart](chart.png)
HybridRec: Contributions and limitations

Contributions
- It works also for MVN projects
- Combining different approaches increases the performances
- It can cover all the types of GitHub topics

Limitations
- No comparison with a baseline
- Not able to analyze the source code
Assisting modelers in specifying models and metamodels
Conceptual map

Categorizing OSS projects

Conceptualization and development of RSSEs

Assisting modelers in models specification

MNBN

HybridRec

MemoRec

MORGAN
Setting the context

- MDE employs models as first-class artifacts to facilitate the overall development lifecycle*

- Recently, intelligent modeling assistants (IMAs) have been proposed to automatize their specification**

- Selecting a proper encoding scheme and suitable data is still open challenges


Motivating example

Partial metamodel

Complete metamodel
Proposed solution: MORGAN*

Evaluation process
Results

Models results

Metamodels results
MORGAN: Contributions and limitations

Contributions

- It was the first approach using graph kernel similarity to support modeling activities
- It can support both metamodel and model completion

Limitations

- Scalability issue when large datasets are considered
- No user evaluation
Automating the design and the deployment of RSSEs
Conceptual map

Conceptualization and development of RSSEs

- Categorizing OSS projects
- Assisting modelers in models specification
- Automating the development of RSSEs
- Challenges and lessons learned

- MNBN
- HybridRec
- LEV4REC
- MemoRec
- MORGAN
Lessons learned from developed RSSE

Requirements

#R1: Importance of a clear requirement definition process

#R2: Users skepticism

Development

#D1: Selecting the right representation can be of paramount importance

#D2: Do not pretend to immediately find the optimal solution (move on iteratively)

#D3: Start first with the techniques you already know and move on from there

Evaluation

#E1: Candidate baselines might not be reusable

#E2: User studies are cumbersome, and they can take a long time to be conducted and completed

#E3: In certain contexts, the k-fold cross-validation technique is a good alternative to user studies
Setting the context

Several frameworks have been proposed to automatize the development of RS*,**, None of the existing approaches consider constraints at the design time


Motivating example
Proposed solution: LEV4REC*,**


RS Metamodel
CrossRec* use case

Feature specification

Model fine-tuning

Results: Quantitative analysis

<table>
<thead>
<tr>
<th>Avg. metrics</th>
<th>Collaborative filtering</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CROSSREC</td>
<td>LEV4REC</td>
</tr>
<tr>
<td>Precision</td>
<td>0.623</td>
<td>0.634</td>
</tr>
<tr>
<td>Recall</td>
<td>0.622</td>
<td>0.552</td>
</tr>
<tr>
<td>F1</td>
<td>0.623</td>
<td>0.590</td>
</tr>
</tbody>
</table>
Qualitative evaluation: Focus group*

<table>
<thead>
<tr>
<th>Participant</th>
<th>Recommender Systems</th>
<th>Low-code Engineering</th>
<th>Software Product Lines</th>
<th>Model-driven Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Expert</td>
<td>Outsider</td>
<td>Outsider</td>
<td>Outsider</td>
</tr>
<tr>
<td>P2</td>
<td>Good Knowledge</td>
<td>Good Knowledge</td>
<td>Good Knowledge</td>
<td>Good Knowledge</td>
</tr>
<tr>
<td>P3</td>
<td>Expert</td>
<td>Expert</td>
<td>Expert</td>
<td>Expert</td>
</tr>
<tr>
<td>P4</td>
<td>Familiar</td>
<td>Good Knowledge</td>
<td>Familiar</td>
<td>Expert</td>
</tr>
<tr>
<td>P5</td>
<td>Expert</td>
<td>Expert</td>
<td>Expert</td>
<td>Expert</td>
</tr>
</tbody>
</table>

**Pro**

- Decent degree of automation
- Generalizability

**Cons**

- Usability, i.e., improving the user interface
- Extensibility, i.e., providing dedicated extension point

---

LEV4REC demo

LEV4REC Feature model

Algorithm Component
- Collaborative filtering
- Context-based
- Classification
- Mining Algorithms

Dataset component
- Data structure
  - Textual Data
  - Graphs
  - Matrix
  - ARFF
  - Tree

Dataset type
- Unsupervised dataset
- Supervised dataset

Preprocessing Data Component
- NLP
- Feature Scaling
- Dimensionality Reduction
LEVI4REC: Conclusion

Contributions

- An MDE-based environment to specify recommender systems
- It can cover additional languages apart from Python, e.g., Java, C++
- Each component can be extended by modifying the two models

Limitations

- User interface needs improvement
- Provide dedicated extension point
- Only a subset of features has been implemented
Developed RSSEs

Categorizing Github

MNBN

HybridRec

Modeling assistance

MemoRec

MORGAN

<table>
<thead>
<tr>
<th>Repository</th>
<th>Real topics</th>
<th>Featured topics</th>
<th>Recommended topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1studmuffin/SpaceshipGenerator</td>
<td>python, blender-scripts, spaceship,</td>
<td>python, 3d</td>
<td>shell, terminal, 3d,</td>
</tr>
<tr>
<td></td>
<td>procedural-generation, game-</td>
<td></td>
<td>opengl, python</td>
</tr>
<tr>
<td></td>
<td>development, 3d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xAX/go-algorithms</td>
<td>golang, algorithm, data-structures,</td>
<td>algorithm, data-structures,</td>
<td>data-structures, algorithm, twitter, library, go</td>
</tr>
<tr>
<td></td>
<td>go, sort, tree-structure</td>
<td>go</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Context</th>
<th>Recommended item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow</td>
<td>attribute finalizeFlow:EBigInteger attribute eventPatternId:EString reference initialState:initState reference finalStates:FinalState</td>
</tr>
<tr>
<td>Step</td>
<td>metaclass StepAlternative:EObject metaclass Automaton metaclass FSM metaclass mFSM</td>
</tr>
</tbody>
</table>
Automating the design of RSSEs

#1 Conceiving a series of RSSEs to assist developers in specific tasks

![Diagram of RSSE design process]

```
RModel myRModel {  
  dataset : unsupervisedDataset datasetName {  
    1  
  }  
  evaluation : Evaluation eval {  
    2  
  }  
  recommendationSystem : WebService presentation_layer {  
    3  
  }  
  recommendationSystem : Evaluation eval {  
    4  
  }  
  recommendationSystem : Filtering eval {  
    5  
  }  
  generator SUMMARIZE filteringMLalgorithm ITEM_BASED {  
    6  
  }  
}
```

<table>
<thead>
<tr>
<th>Avg. metrics</th>
<th>Collaborative filtering</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CROSSREC</td>
<td>LEVIREC</td>
</tr>
<tr>
<td>Precision</td>
<td>0.623</td>
<td>0.634</td>
</tr>
<tr>
<td>Recall</td>
<td>0.622</td>
<td>0.552</td>
</tr>
<tr>
<td>F1</td>
<td>0.623</td>
<td>0.590</td>
</tr>
</tbody>
</table>

Generate Code
Future work in RSSEs

- Embody user feedback in an agnostic way*

- RSSEs for IoT application domain**

- Investigate qualitative aspects in RSSEs, e.g., fairness***, trustworthiness, explainability

---


**Di Rocco J. and Di Sipio C., ResyDuo: Combining data models and CF-based recommender systems to develop Arduino projects, 5th International Workshop on Multi-Paradigm Modeling for Cyber-Physical Systems (MPM4CPS’23)