Software Engineering for Artificial Intelligence

Basics and Challenges
Outline

• Intelligent Systems
• When to Use Intelligent Systems
• Challenges of Intelligent Systems
  • Good Goals
  • SE Workflow
  • Technical Debt
Let's talk about Toast
Intelligent Systems [2]

Successful Intelligent Systems have:

- **Objective**
  - Meaningful to user
  - Achievable

- **Intelligence Creation**
  - Through anything from simple heuristics to complex ML

- **Experience**
Intelligent Experience [2]

- Achieve system’s objective
- Present intelligence to users
  - Balance quality with forcefulness
  - Key actions: automate, prompt, organize and annotate
- Minimize intelligence flaws
  - Experience can avoid risky decisions
  - Experience can control the number of user interactions
  - Experience can use less forceful actions in risky situations
- Create data for system growth
  - Experience must know the interaction context, the action taken by the user, and the outcome
Intelligent Systems [2]

Successful Intelligent Systems have:

**Objective**
- Meaningful to user
- Achievable

**Implementation**

**Experience**
- Presents output to the user
- Minimizes mistakes
- Must collect feedback

**Intelligence Creation**
- Through anything from simple heuristics to complex ML

**Orchestration**
- Controlling system changes
- Keep experience in sync with intelligence quality
- Involves dealing with mistakes, controlling risks and defusing abuse
Intelligence Implementation [2]

- Intelligence Runtime: executes the intelligence and gathers the context of the interaction
- Intelligence Management: deploying new versions of the intelligence
- Monitoring and Telemetry Pipeline: what and how to observe, sample, and summarize what is going on
- Intelligence Creation Environment: intelligence creator must be able to recreate runtime context to create accurate intelligence
- Intelligence Orchestration: controlling the system, i.e., when the intelligence evolves, runs into problems
Intelligent Systems [2]

Successful Intelligent Systems have:

**Objective**
- Meaningful to user
- Achievable

**Implementation**
- Executing system
- Includes telemetry and feedback

**Experience**
- Presents output to the user
- Minimizes mistakes
- Must collect feedback

**Intelligence Creation**
- Through anything from simple heuristics to complex ML

**Orchestration**
- Controlling system changes
- Keep experience in sync with intelligence quality
- Involves dealing with mistakes, controlling risks and defusing abuse
When to Use Intelligent Systems [2]

Intelligent systems should be only used for complex problems

- Complex problems: *big, open-ended, time-changing* or *intrinsically hard*
- Requirements for intelligent systems
  - Partial solution must be viable and interesting
  - Usage data must be recordable (to improve the system)
  - Ability to influence meaningful objective
    - Objective should be *directly* and *quickly* affectable; taken actions should be *measurable* in the outcome
  - Problem must justify effort
    - Intelligence creation is cheaper than in other methods, but the overhead is very expensive
Challenges

ML challenges remain, but the SE challenges of intelligent systems are much broader

General ML Challenges [1]

- Insufficient Quantity of Training Data
- Nonrepresentative Training Data
- Poor-Quality Data
- Irrelevant Features
- Overfitting the Training Data
- Underfitting the Training Data
Challenges: Good Goals [2]

Intelligent System

Intelligence + Experience $\rightarrow$ Outcome

Targeted at right problem

Encourages correct user behavior

Achievable
Challenges: Good Goals [2]

- **Communicate desired outcome** to everyone with clear importance and understanding of success
- **Are achievable**, meaning there is an explainable approach and a likely chance of success
- **Are measurable**, optimizing for non-measurable goals is impossible

Abstract goals

<table>
<thead>
<tr>
<th>Very concrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>organizational objectives</td>
</tr>
</tbody>
</table>

**Effective goal sets tie usually goals of various types together**
SE Workflow [4]

- Case study at Microsoft: 9 stages ML workflow with big feedback loops

- Big difference to „traditional software“: Very data centric & more loops

- SE is about software code, ML is all about data for learning models
  - Software has specifications, datasets usually do not have specifications
  - Specifications change rarely, data schemas may change very frequently
  - No mature tools for data versioning and meta-data management, while for code these systems exist
- Customization and reuse of models is harder than of code
  - Even a slight variation in the usage scenario may require deep changes to the model, training data or the executing system
- Modularity in ML and strict boundaries between models are difficult
  - Models are not easily extensible
  - Models interact in non-obvious ways: model results affect others training and tuning processes; isolated development is hard
Technical Debt

- SE is all about making **qualified decisions** based on tradeoffs
- Sometimes decision are knowingly taken, which are good in the short run, but will cause more work in future: „technical debt“

Technical Debt

Sources of technical debt are ubiquitous in today‘s ML

https://xkcd.com/2054/ (accessed in 04.05.2020)
Hidden Technical Debt in ML Systems [3]

1. Model Complexity

- **Entanglement**: ML mixes many different external and internal signals; isolated improvement is impossible, wherefore changes are expensive.
- **Correction Cascades**: For reuse it is tempting to add a new tiny AI on top of an existing one, but this makes analysis and improvement much more expensive.
- **Undeclared Consumers**: Opening AI results is great for re-use, but makes overall progress much more expensive.
Hidden Technical Debt in ML Systems [3]

2. Data Dependencies

- Data dependencies cause dependency debt, which is hard to detect; code dependencies are easily traceable through static analysis.
- Unstable Data Dependencies: Some sources might vary in quality and quantity of provided data.
- Underutilized Data Dependencies: Some data sources might not really be relevant to the outcome of the intelligence, however, they still increase complexity.
Hidden Technical Debt in ML Systems [3]

3. Feedback Loops

- Intelligent systems often influence their own behavior through feedback loops
- Causes **analysis debt**: behavior after release is hard to, if it depends on the sexecution
- Direct Feedback Loops: Explicitly build in loops, e.g., for selection of future training data
- Hidden Feedback Loops: Implicit feedback loops, e.g., through reactions of users
  - Example: ML-based stock market agents: developed separately, but through shared market they influence each other and themselves
Hidden Technical Debt in ML Systems [3]

4. Others

- Anti-patterns
  - Glue Code: big support code makes the system heavy
  - Pipeline Jungles: special kind of glue code; expensive to test
  - Dead Experimental Codepaths: common source of sudden errors

- Common Smells
  - Plain-old-data type smell
  - Multiple-language smell: Increases testing complexity and makes ownership transition often harder
  - Prototype smell
Summary

• Intelligent systems connect AI and users
  • Objective, intelligence creation, implementation, experience, and orchestration
• Intelligent systems should be only used for complex problems
• Challenges include:
  • Definition of Goals
  • Differences between SE for 4ML and traditional SE methods
  • Ubiquitous sources of technical debt in ML
Literature


Discussion
Acknowledgements & License

- Material Design Icons, by Google under Apache-2.0
- Other images are either by the authors of these slides, attributed where they are used, or licensed under Pixabay or Pexels
- These slides are made available by the authors (Daniel Sokolowski, Guido Salvaneschi) under CC BY 4.0