Generative AI Programming Assistant

Danny Dig

Boulder

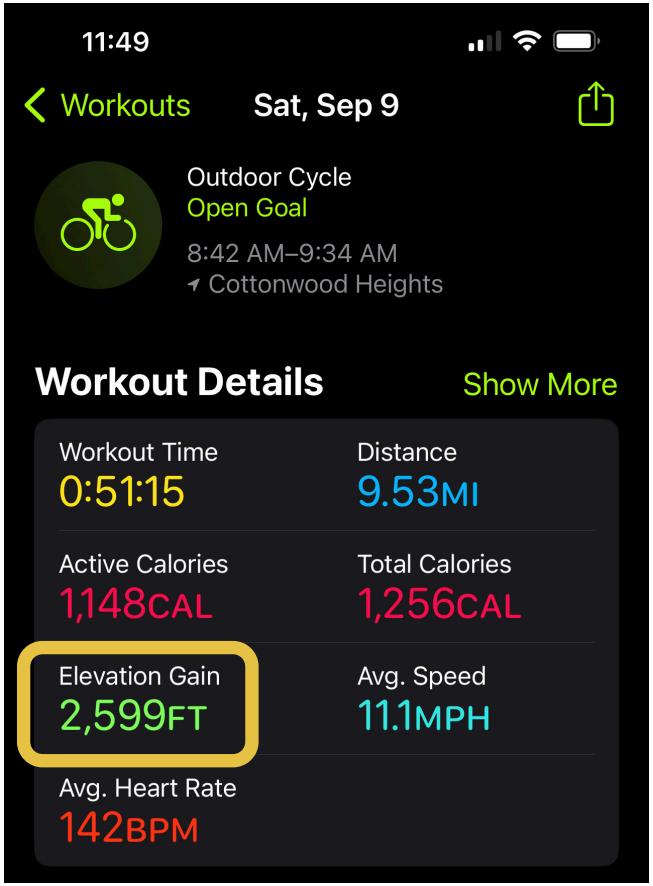
danny.dig@colorado.edu

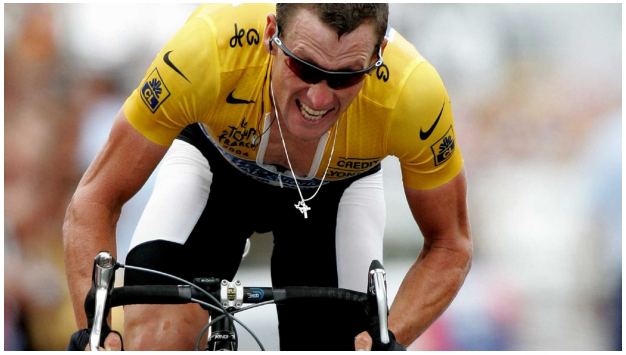
JET BRAINS Research

CENTER ON Pervasive Personalized Intelligence

Together We Go Farther

E-Assistant augments our capacity





85 Nm, 500 Wh

CENTER ON Pervasive Personalized Intelligence

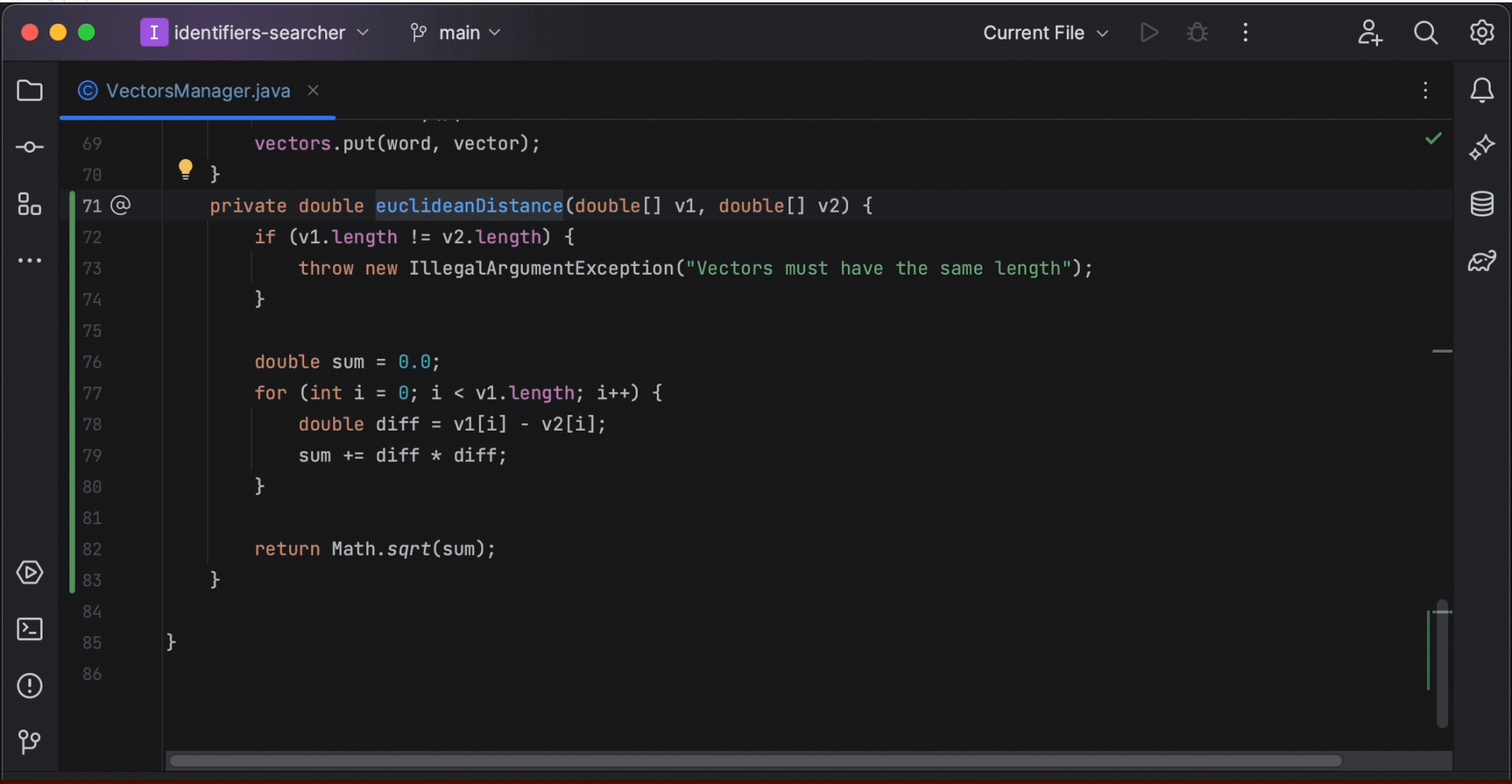
9 hardest things programmers have to do

- Writing tests
- Writing documentation
- Implementing functionality you disagree with …
- Working with someone else's code
- Dealing with other people …
- Estimating time to complete a task
- Explaining what I do (or don't do)
- Naming things

source: Elizabeth Churchill, UX Director at Google

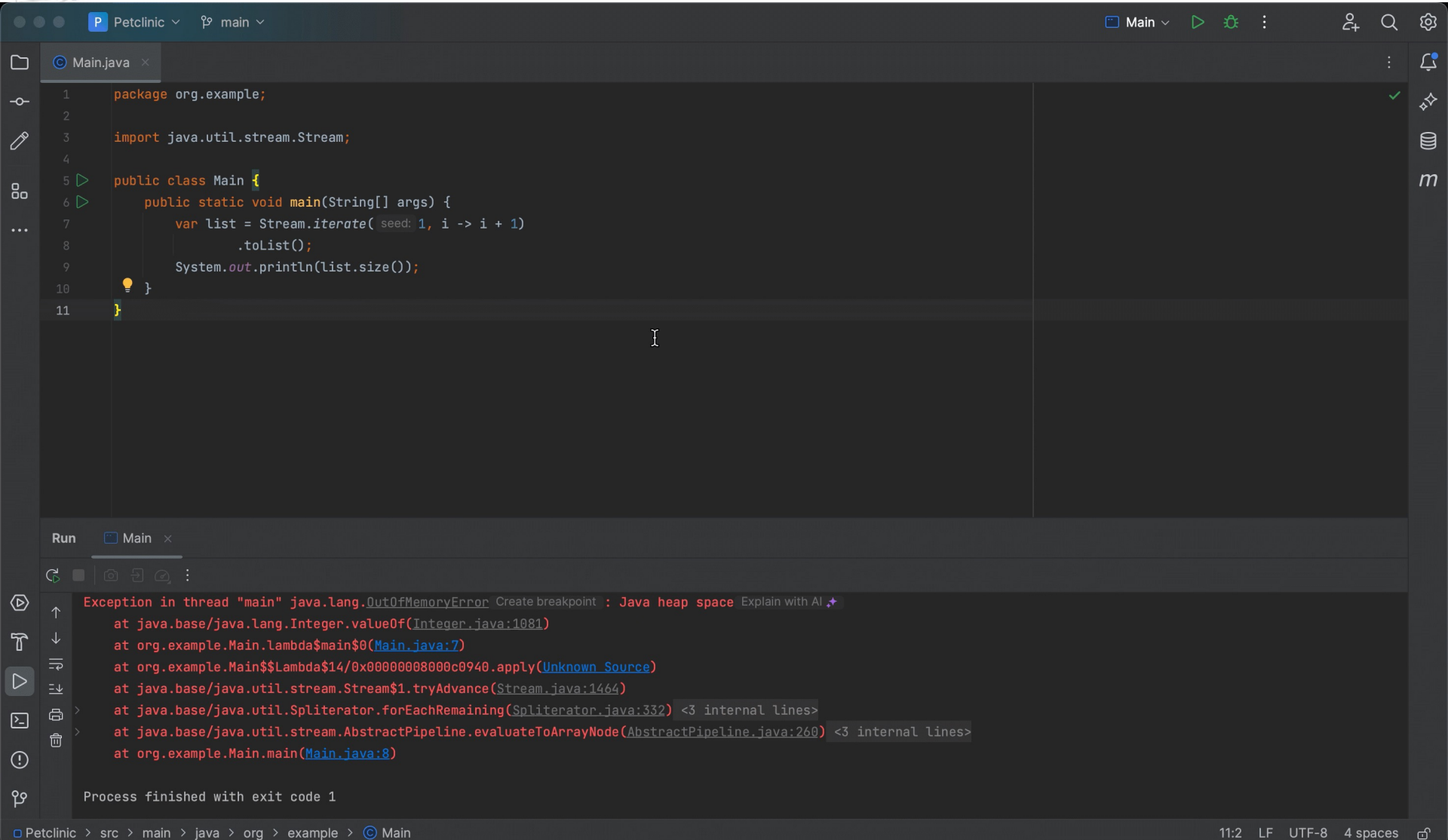
Generative AI explains code

https://blog.jetbrains.com/idea/tag/ai-assistant/



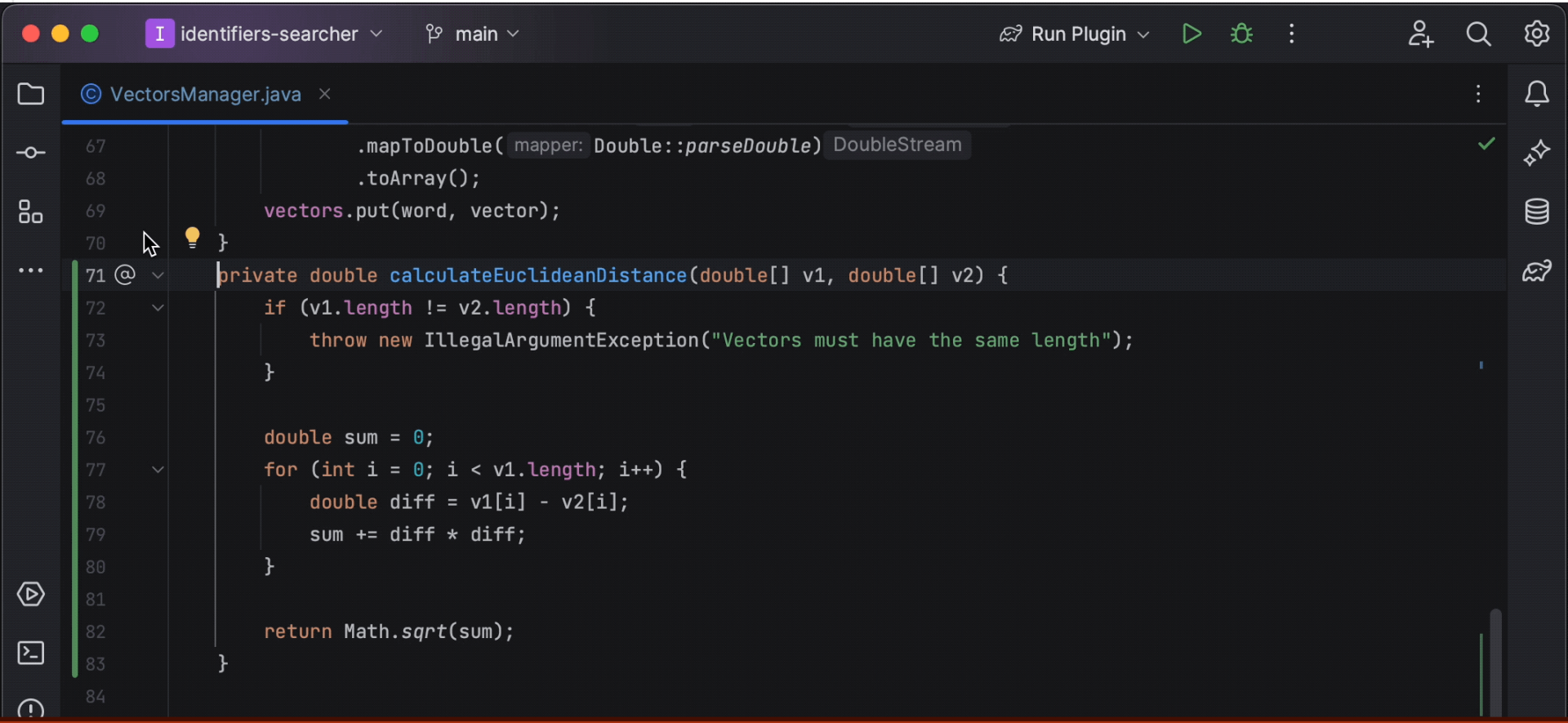
Explain code, bugs and offer fixes, summarize recent code changes, perform code reviews

Generative AI explains Runtime Errors



Generative AI generates documentation

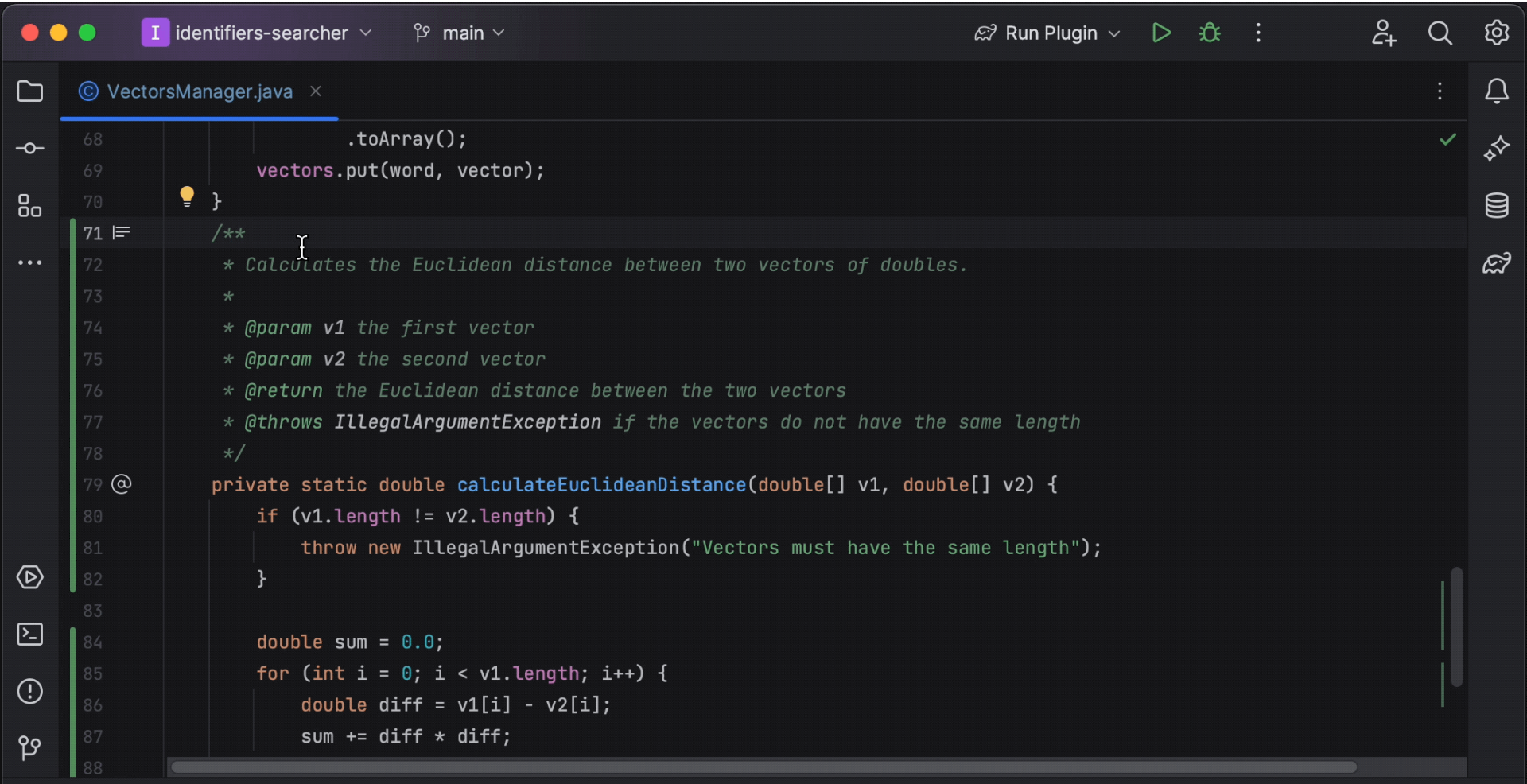
Programmers often don’t document their code, so AI generates documentation



Generate: documentation, code, test cases, configuration files, comments, commit messages, CLI commands, suggestions to improve code

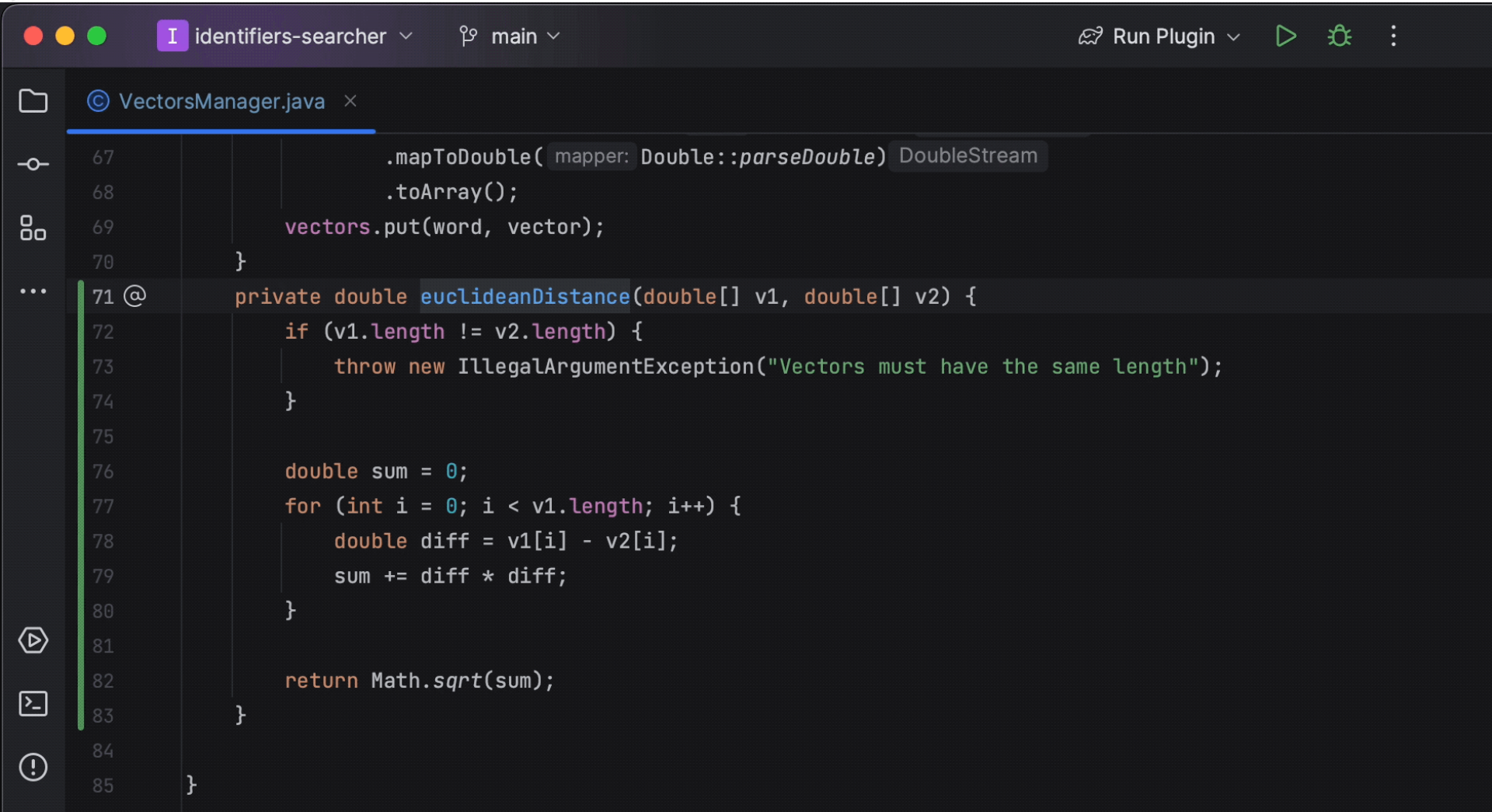
Generative AI generates commit messages

Programmers don’t write good commit messages when they push their changes into code repositories



Generative AI name suggestions

Programmers don’t spend enough time giving intention revealing names to code entities, though it is crucial for code readability



Addressing top challenge with Generative AI

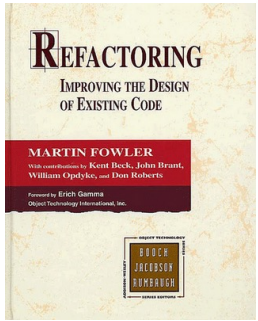
Top concerns of surveyed developers are security, safety, and trustworthiness of AI Generative tools

Combine creativity of Generative AI with

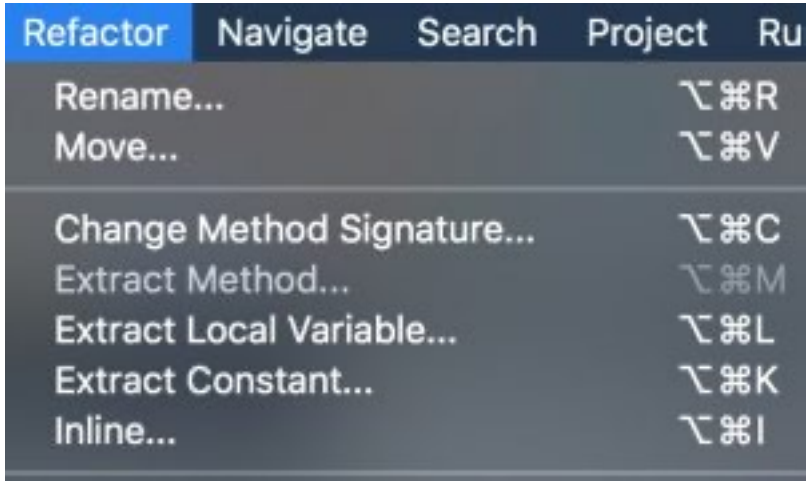
Safety analysis of Programming Environments

My Refactoring Story

"A change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behaviour" – M. Fowler [1999]



Top-level menu in all modern IDEs



In 2000, I created the first open-source refactoring tool

In 2004, I joined the team

In 2023-24 doing a Sabbatical at JetBrains

JET BRAINS Research

To go fast, go alone. To go far, go with others.

CENTER ON Pervasive Personalized Intelligence

ExtractMethod-Assist

LLM + IDE: Together We Go Further

Dorin Pomian, Malinda Dilhara, Zarina Kurbatova, Abhiram Bellur, Egor Bogomolov, Andrey Sokolov, Timofey Bryksin, Danny Dig

CENTER ON
Pervasive Personalized Intelligence

ExtractMethod in IntelliJ today

- Top-5 most used refactorings, IntelliJ was the first Java IDE to support

- Solution to remove Long Method, it improves:

  readability, reuse, testing, reduce technical debt

ExtractMethod in IntelliJ today

- ✗ Semi-automated
- ✗ No recommendations

SerializerUtil.java | MethodExtractor.kt

25 23

```
83
84      //  Write myMethods
85      int methodCount = 0;
86      for (JvmMethod jvmMethod : jvmClass.getMethods()) methodCount++;
87      DataInputOutputUtil.writeINT(out, methodCount);
88      for (JvmMethod jvmMethod : jvmClass.getMethods()) {
89        writeJvmMethod(jvmMethod, out);
90      }
91
92      //  Write AnnotationTargets
93      int elemTypeCount = 0;
94      for (ElemType elemType : jvmClass.getAnnotationTargets()) elemTypeCount++;
95      DataInputOutputUtil.writeINT(out, elemTypeCount);
96      for (ElemType elemType : jvmClass.getAnnotationTargets()) {
97        writeElemType(elemType, out);
98      }
99
100     if (jvmClass.getRetentionPolicy() != null) {
101       out.writeUTF(jvmClass.getRetentionPolicy().name());
102     }
103     else {
104       out.writeUTF( s: "");
105     }
106   }
```

Extract Method Research

- Many research tools
  - JDeodorant
  - JExtract
  - LiveREF
  - REMS
  - GEMS
  - SEMI
- Optimize software quality metrics (complexity, coupling, cohesion)
- Generate refactorings that do not align with developers' preferences

Large Language Models (LLMs) and Refactoring

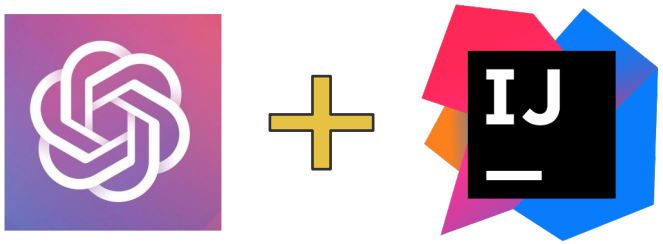
Corpus of 2,849 real-world extract methods:

- LLMs are creative and prolific: average 20 suggestions per method

- 63% of suggestions are hallucinations:
  - 46% are invalid: non-compiling or semantically incorrect
  - 17% are not useful (e.g. one liners, or entire method body)

Key Idea

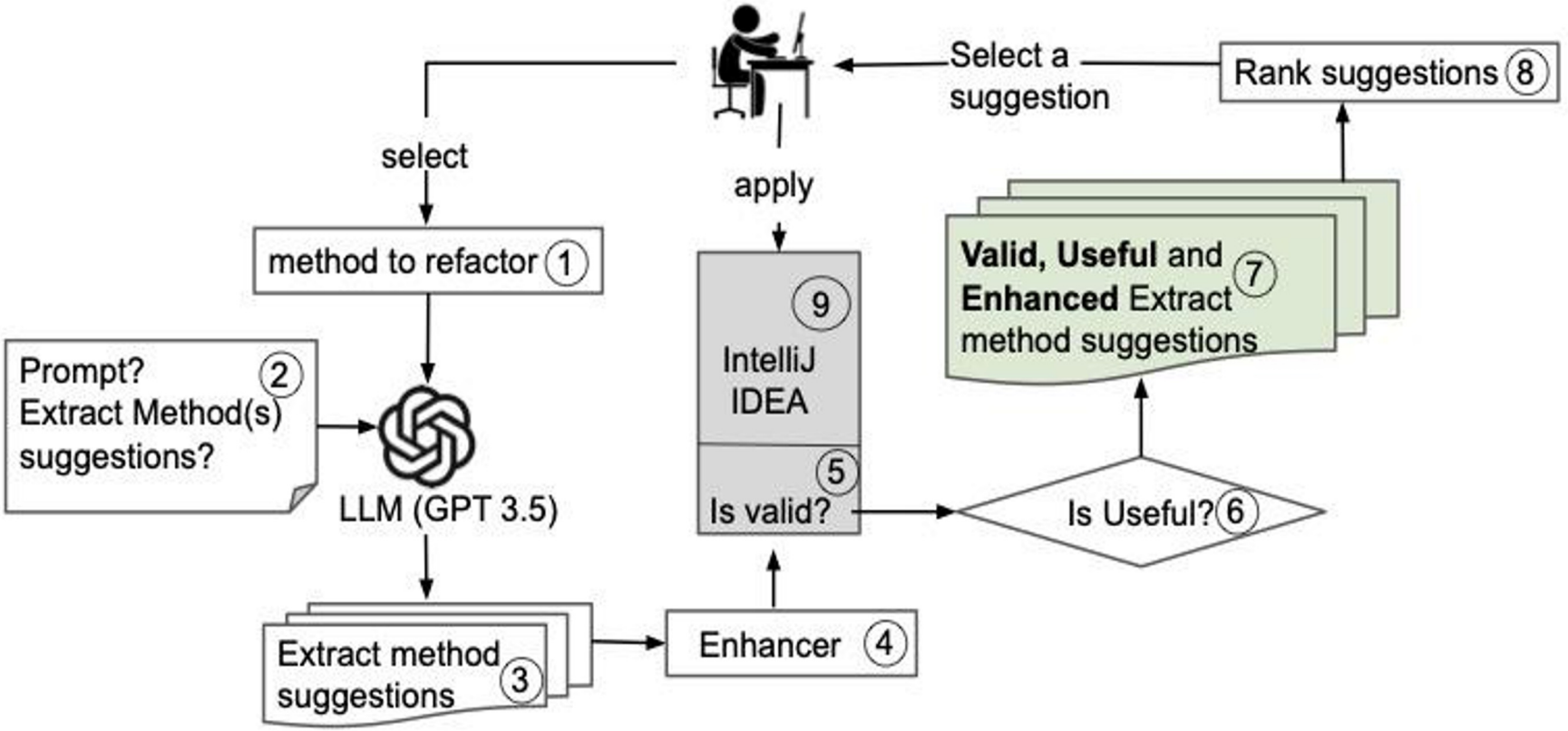
Combine creative capabilities of LLMs + full power of static analysis in IntelliJ to apply refactorings safely



Use static analysis techniques to filter, further enhance, an rank LLM-provided suggestions


Java™

Under the Hood



Empirical Evaluation

RQ1: How effective are LLMs for refactoring?

A: WOW effect, but with a high rate of hallucinations

RQ2: What are best practices for using LLMs for refactoring?

A: sensitivity analysis of LLM hyper-parameters, ablation study for our components

RQ3: Effectiveness of EM-Assist

1. Synthetic corpus: 122 Extract Methods
   - Used by other tools (JDeodorant, JExtract, LiveREF, REMS, GEMS, SEMI)
   - EM-Assist has 61% recall
   - Next best tools 54% recall (ML-based), and 52% (static analysis)

2. Extended corpus: 2,840 Extract Methods from real world
   - EM-Assist has 42% recall
   - Compared to 6.5% recall rate reported by its peers (i.e., *LiveREF*)

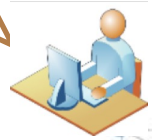
RQ4: Usefulness and Usability

Usefulness of suggestions: fire-house survey with 16 experienced developers
- 81% positive rating for recommendations

Usability study with 18 developers: 94% gave a positive rating

”Thank you for interesting suggestions! Hope to see this in production.”

”These suggestions made me look at this code with new eyes, and I will refactor it.”



Lessons Learned

Prompt engineering: do what LLM suggests, not what they do
- remove hallucinations automatically, execute with the IDE
- precise prompt for higher quality suggestions
- few-shot learning worked best

Taming LLM non-determinism:
- re-prompting not a waste
- newly-designed ranking to match LLM workflow

Sweet spot: temperatures and number of iterations

Effectiveness is slightly better on synthetic benchmarks, but dramatic improvements (6.4x) on real-world large datasets

Research Preview: Programming By Example

Learn coding best practices from open-source ML Python repos and transplant these into other code

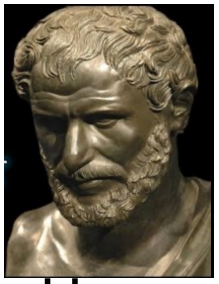
Cannot apply these to new sites unless the code is exactly the same

Use LLM to generate many examples, validate and apply suggestions to new locations

25x improvements over previous state of the art

We submitted hundreds of patches to famous open-source projects (e.g., TensorFlow), they accepted 83%

Executive Summary



Change is the only guaranteed constant

Generative AI + Programming Environments + Human

" … spot on with what I hate to do as a programmer, happy if LLMs could help with that" – NEC software engineer

"…Game changer! Enter the code assistant, the great equalizer. It's the ace up the sleeve for OEMs" --- Lenovo CTO

Our responsibility as a research community for trustworthy GenAI

PPI Center: industry-university-government partnership

- iterative process, present progress, get feedback, improve, reenter

PPICenter.org

danny.dig@colorado.edu

CENTER ON Pervasive Personalized Intelligence