The background of the slide is a detailed, close-up photograph of a complex mechanical device, likely a historical automaton or a large clockwork mechanism. It features numerous large, interlocking brass gears of various sizes, some with decorative patterns. The mechanism is mounted on a wooden base. In the lower right foreground, an open book with aged, yellowed pages lies flat on a wooden surface, possibly part of the machine's base. The lighting is warm and focused, highlighting the metallic textures and the book's pages against a dark, blurred background.

Large Language Models

LLM4SE

Riccardo Coppola

Definitions

- **AI4SE** applies augmented intelligence and machine learning techniques to support systems engineering practices More and more systems are software controlled
- **SE4AI** applies systems engineering methods to learning- based systems' design and operation

Definitions

- **LLM-Based Software Engineering (LLMSE):** integration of Large Language Models (LLMs) into software engineering. It encompasses any application where the products or processes leverage LLMs to enhance development and operational efficiency.

Definitions

- **LLM Application:** Defined as any task or activity that benefits from LLM insights. This broad definition captures the essence of LLM's versatility across various domains, offering improvements through its advanced computational capabilities.
- **LLM Consumer:** Any individual, system, or process that utilizes LLM outputs. This definition acknowledges the wide array of LLM beneficiaries, from developers and businesses to automated systems, all relying on LLM-generated intelligence.

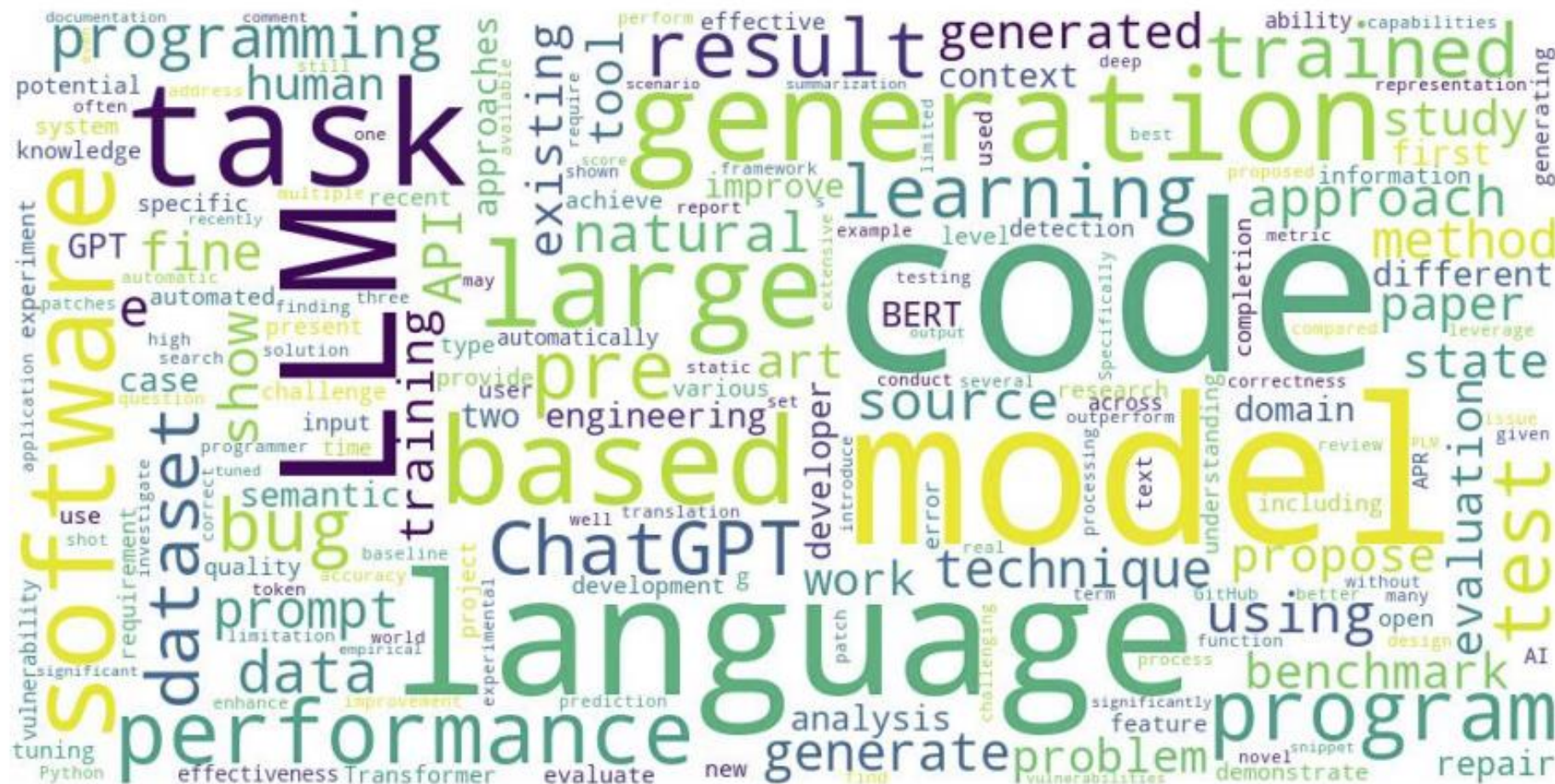
Definitions

- **Assured LLMSE:** This innovative approach guarantees the reliability of LLM outputs. Every response from an LLM, possibly after undergoing post-processing, comes with a verifiable assertion of its usefulness. Assured LLMSE sets a standard for trust and quality in the application of LLMs in software engineering.



LLM4SE

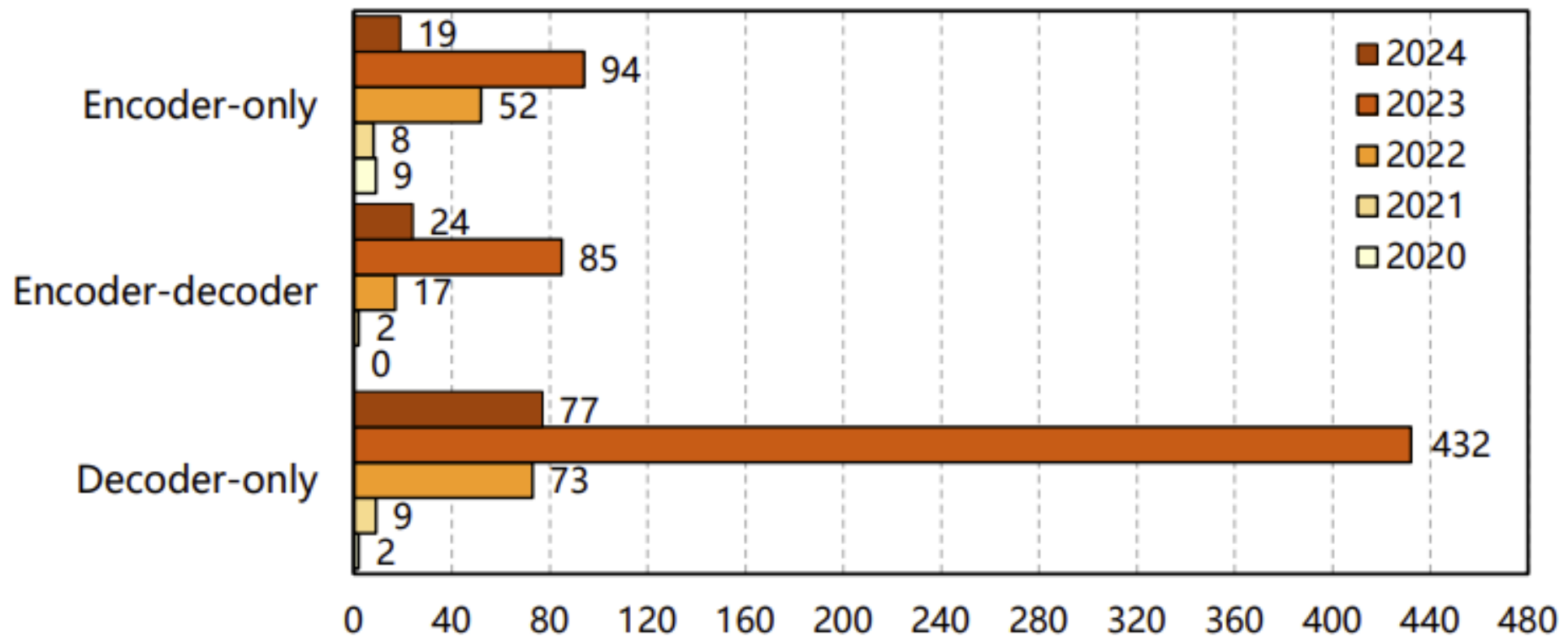
A Word Cloud



Current state of the art

- In Software Engineering literature, more than 70 different LLMs have been used for SE tasks. All three categories of LLMs (decoder-only, encoder-decoder, and encoder-only) have been used.
- Different categories of LLMs serve a specific purpose in SE tasks:
 - Encoder-only LLMs are mostly used on comprehensive understanding;
 - Encoder-decoder LLMs are mostly used for tasks requiring understanding input information followed by content generation;
 - Decoder-only LLMs are more suitable for generation tasks.
- The most widely used LLMs are with decoder-only architectures.

Current state of the art



Some Examples

Model	Type	Example of SE tasks
Encoder-only	Understanding	Code Understanding Bug localization Vulnerability detection
Encoder-Decoder	Understanding and Generation	Code summarization Code translation Program repair
Decoder-only	Generation	Code generation Code completion Test case generation

Criteria for LLM selection

- The selection of LLM for SE tasks should involve more careful consideration rather than arbitrary choice.
- Key factors guiding this selection are:
 - Model proficiency in understanding the context of the code
 - Ability to generate relevant content
 - Responsiveness to fine-tuning
 - Demonstrated performance in SE-specific benchmarks

Task-specific fine tuning

- A notable trend is the customization of LLMs for precise SE tasks.
- By fine tuning models with datasets tailored to specific functions (e.g., bug detection or code review) researchers are able to achieve marked performance improvements.

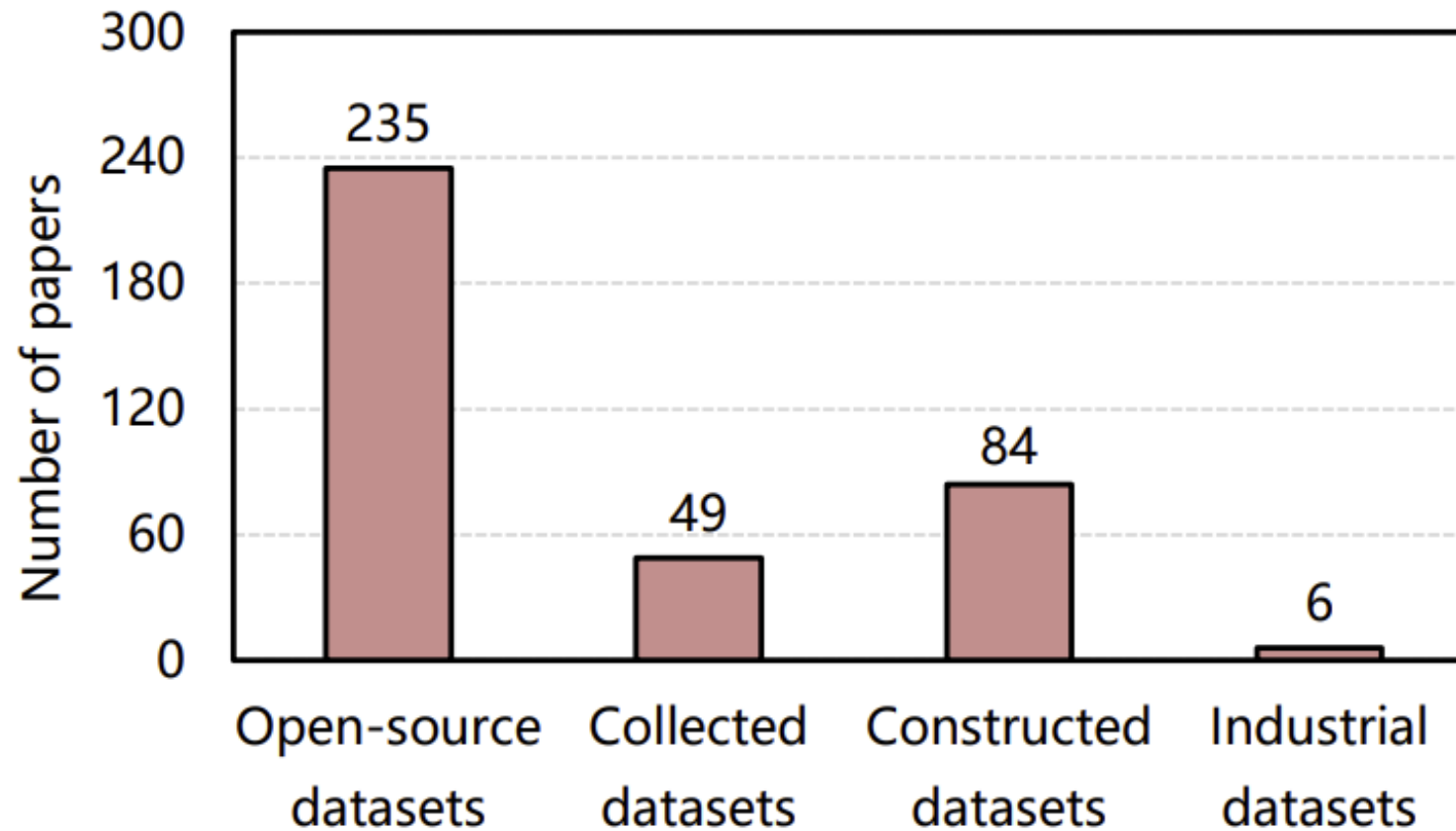


Types of Datasets

Sources for datasets

- Data determines the generalization ability, effectiveness, and performance of the models.
- Four different methods can be used for data collection:
 - **Open-source datasets:** publicly accessible collections of data that are often disseminated through open-source platforms or repositories.
 - **Collected datasets:** datasets compiled from the researchers directly from a multitude of sources, including (but not limited to) major websites, forums, blogs, and social media platforms.
 - **Constructed datasets:** specialized datasets that researchers create by modifying or augmenting collected datasets to better align with their specific research objectives.
 - **Industrial datasets:** obtained from commercial or industrial entities; often contain proprietary business data, user behaviour logs, and other sensitive information.

Sources for datasets



Sources for datasets

- Main benefits of open-source datasets:
 - Authenticity and credibility;
 - They often contain real-world data collected from various sources;
 - LLMs have recently emerged – so a lack of suitable sets does exist. Therefore, researchers often collect data from open-source repositories.

Types of data utilized in training LLM4SE

- **Text-based datasets:** datasets composed by textual (natural language) elements, or non-functioning code (snippets).
 - Among text-based datasets there are programming tasks/problems, the most frequently used of all data types. This dominance can be attributed to the diverse and challenging nature of programming problems.

Types of data utilized in training LLM4SE

- **Code-based datasets:** datasets composed only by code artefacts.
 - Among code-based datasets, the predominant ones are repository of production source code. This predominance is due to the fundamental role in SE, since source code serves as the foundation of any software project.
 - Other common data types are bugs/buggy code, patches for program repair, vulnerable source code.

Types of data utilized in training LLM4SE

- **Graph-based datasets:** can be used when representing the GUI states of an application to develop or test.
 - An example is the use of screenshot from Google Play Android to construct a graphical user interface (GUI) repository.

Types of data utilized in training LLM4SE

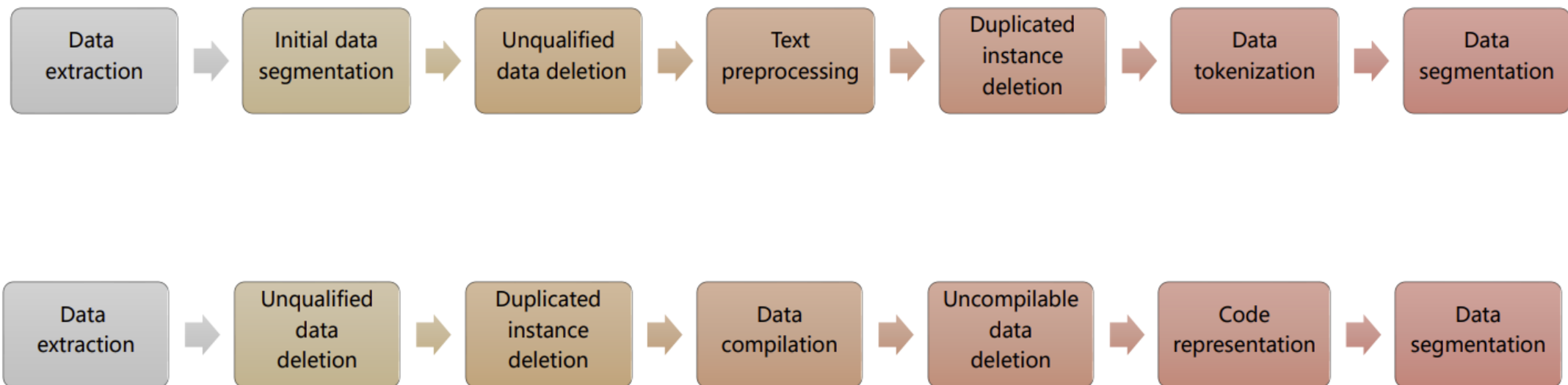
- **Software repository based-datasets:** compilations of data extracted from version control systems, such as Git repositories, containing code, documentation, and related artefacts.
 - This data includes code repository, issues and commits, and so on.
 - The data in code repositories provide information covering all aspects of the software development process (history, issue fixes, feature improvements, quality assessments...)
 - These data are valuable for studying behaviours and trends in the software development process.

Types of data utilized in training LLM4SE

- **Combined datasets:** combinations of multiple types of datasets.
 - Most common datasets are “programming tasks and test suites/cases”
 - ...or source code and comments/documentation

Data-preprocessing

- The data types influence the selection of data-preprocessing techniques



Used tuning techniques

- Many general-purpose LLMs (e.g., ChatGPT) are efficiently and directly applied to SE tasks such as code generation, code summarization, and program repair without fine tuning.
- Tuning is often needed to realize the true potential of LLMs.

Used tuning techniques

- Many studies have used BERT series models with full tuning
 - This requires a large amount of computational resources, and massive amounts of data.
 - It is also costly to train and deploy the fine-tuned models separately for each downstream task.
- Some efforts to reduce the burden:
 - In-Context Learning (ICL)
 - Parameter Efficient Fine-Tuning (PEFT)
 - Low-Rank Adaptation (LoRA)
 - Prompt tuning
 - Prefix Tuning
 - Adapter Tuning
 - Reinforcement Learning (RL)

Types of data utilized in training LLM4SE

Category	Data type		Total
Text-based datasets	Programming tasks/problems (42)	Prompts (33)	151
	SO (i.e. Stack Overflow) posts (12)	Bug reports (11)	
	Requirements documentation (9)	APIs/API documentation (8)	
	Q&A pairs (6)	Vulnerability descriptions (4)	
	Reviews (4)	Logs (3)	
	Methods (3)	Project issues (3)	
	Code comments (2)	Theorems (2)	
	Buggy text (1)	Dockerfiles (1)	
	Outage descriptions (1)	Semantic merge conflicts (1)	
	Site text (1)	Software development tasks (1)	
	User intents (1)	Software specifications (1)	
	User reviews (1)		
Code-based datasets	Source code (60)	Bugs/Buggy code (16)	103
	Vulnerable source code (8)	Patches (4)	
	Code changes (3)	Test suites/cases (3)	
	Bug-fix pairs (2)	Error code (2)	
	Error-fix pairs (1)	Flaky test cases (1)	
	Identifiers (1)	Labeled clone pairs (1)	
	Packages (1)		
Graph-based datasets	GUI Images (1)		1
Software repository -based datasets	Code repository (9)	Android apps (3)	20
	Issues and commits (3)	Pull-requests (2)	
	Industrial projects (1)	Open-source projects (1)	
	Web applications (1)		
Combined datasets	Programming tasks and test suites/cases (17)	Source code and comments (12)	55
	Programming tasks and solutions (8)	Source code and description (3)	
	Code-text pairs (2)	Source code and API usage sequences (2)	
	Source code and test suites/cases (2)	Bug report and test suites/cases (1)	
	Buggy code and comments (1)	Buggy code and solutions (1)	
	Code files and summaries (1)	Binary code and related annotations (1)	
	Failing test code and error messages (1)	Source code and Q&A pairs (1)	
	Source code, methods, and logs (1)	Vulnerable code and description (1)	

Text-based datasets: MBPP

- **Mostly Basic Python Programming**
- A benchmark of around 1000 crowd-sourced Python programming problem, designed to be solvable by entry-level programmers, covering programming fundamentals, standard library functionalities, and so on. Each problem consists of a task description, code solution, and 3 automated test cases.
- <https://huggingface.co/datasets/google-research-datasets/mbpp>

Angelica Chen, Jérémy Scheurer, Tomasz Korbak, Jon Ander Campos, Jun Shern Chan, Samuel R Bowman, Kyunghyun Cho, and Ethan Perez. 2023. Improving code generation by training with natural language feedback. arXiv preprint arXiv:2303.16749 (2023).

Text-based datasets: MBPP

Text	Code	Test List
Write a function to reverse words in a given string.	<pre>def reverse_words(s): return ' ' .join(reversed(s.split()))</pre>	<pre>["assert reverse_words(\"python program\")==(\"program python\")", "assert reverse_words(\"java language\")==(\"language java\""), "assert reverse_words(\"indian man\")==(\"man indian\")"]</pre>
Write a function to check if the given integer is a prime number.	<pre>def prime_num(num): if num >=1: for i in range(2, num//2): if (num % i) == 0: return False else: return True else: return False</pre>	<pre>["assert prime_num(13)==True", "assert prime_num(7)==True", "assert prime_num(-1010)==False"]</pre>

Text-based datasets: Bug Reports and Changesets

- Changesets can encapsulate code changes across one or multiple source code files.
- Modifications to each file are divided into *hunks* – groups of modified lines surrounded by unchanged (context lines)

	#Bugs	#Changesets	#Changeset-files	#Hunks
AspectJ	200	2,939	14,030	23,446
JDT	94	13,860	58,619	150,630
PDE	60	9,419	42,303	100,373
SWT	90	10,206	25,666	69,833
Tomcat	193	10,034	30,866	72,134
ZXing	20	843	2,846	6,165

Agnieszka Ciborowska and Kostadin Damevski. 2023. Too Few Bug Reports? Exploring Data Augmentation for Improved Changeset-based Bug Localization. arXiv preprint arXiv:2305.16430 (2023)

Text-based datasets: Bug Reports and Changesets

Bug 83699

Summary: Font reset to default after screen saver

Description: All editors and views using a StyledText widget have the font reset to default after coming back from my screen saver. [..].

This breakpoint gets hit when I return from the screen saver:

[..] StyledText(Control).updateFont(Font, Font) line: 2913

```
git diff 68f73a31d3bd23bb9be3de8de4cfa69258483b46
```

```
+++ b/[..]/org.eclipse.swt/widgets/Composite.java
```

```
[empty lines omitted]
```

```
- void updateFont (Font oldFont, Font newFont) {
```

```
+ boolean updateFont (Font oldFont, Font newFont) {
```

```
    Control [] children = _getChildren ();
```

```
    for (int i=0; i<children.length; i++) {
```

```
        Control control = children [i];
```

```
+++ b/[..]/org.eclipse.swt/widgets/Control.java
```

```
[empty lines omitted]
```

```
- void updateFont (Font oldFont, Font newFont) {
```

```
-     Font font = getFont ();
```

```
-     if (font.equals (oldFont)) setFont (newFont);
```

```
+ boolean updateFont (Font oldFont, Font newFont) {
```

```
+     boolean sameFont = getFont ().equals (oldFont);
```

```
+     if (!sameFont) setFont (newFont);
```

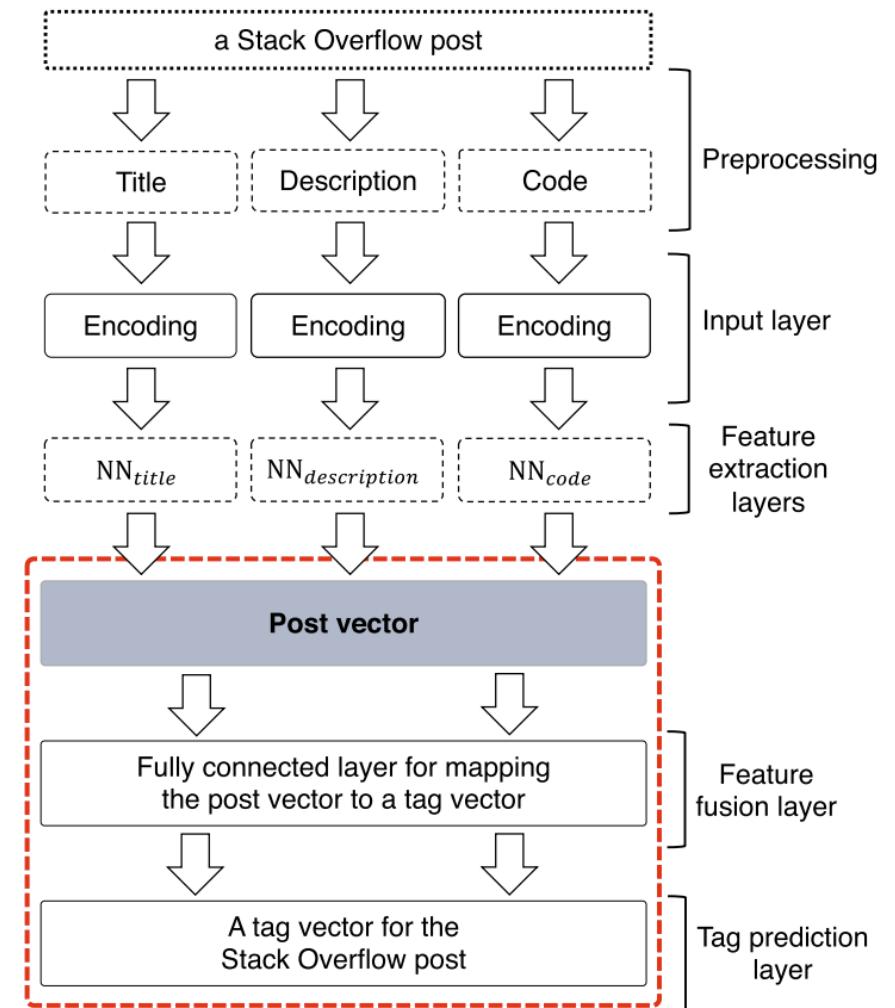
```
+     return !sameFont;
```

```
}
```

```
void updateLayout (boolean resize, boolean all) {
```

Text-based datasets: Post2Vec (Stack Overflow posts)

- Posts on stackoverflow can be separated in several different steps:
 - 1) Separation of description and code snippets from the body
 - 2) Remove HTML tags
 - 3) Tokenize title, description and code snippets
 - 4) Construct component-specific vocabularies



Text-based datasets: Post2Vec (Stack Overflow posts)

Title

How do I write JSON data to a file?

Body

I have JSON data stored in the variable `data` .

I want to write this to a text file for testing so I don't have to grab the data from the server each time.

Currently, I am trying this:

```
obj = open('data.txt', 'wb')  
obj.write(data)  
obj.close
```

And am receiving the error:

```
TypeError: must be string or buffer, not dict
```

How to fix this?

Description

Code

Description

Tags

python

json

Bowen Xu, Thong Hoang, Abhishek Sharma, Chengran Yang, Xin Xia, and David Lo. 2021. Post2Vec: Learning Distributed Representations of Stack Overflow Posts. IEEE Transactions on Software Engineering (2021), 1–1. <https://doi.org/10.1109/TSE.2021.3093761>

Code-based datasets: CodeSearchNet

- 2 million (comment, code) pairs from open source libraries. Concretely, a comment is a top-level function or method comment (e.g. docstrings in Python), and code is an entire function or method.
- Currently, the dataset contains Python, Javascript, Ruby, Go, Java, and PHP code.

php	578118
java	496688
python	457461
go	346365
javascript	138625
ruby	53279

<https://github.com/github/CodeSearchNet?tab=readme-ov-file#data>

Code-based datasets: CodeSearchNet

```
{
  'code': 'def get_vid_from_url(url):\n'
        '    """Extracts video ID from URL.\n'
        '    """\n'
        '    return match1(url, r\'youtu\\.b\n'
        '    match1(url, r\'youtube\\.com/\n'
        '    match1(url, r\'youtube\\.com/\n'
        '    match1(url, r\'youtube\\.com/\n'
        '    parse_query_param(url, 'v')\n'
        '    parse_query_param(parse_quer
```

Software repository-based datasets: DeHallucinator

- Utilization of full projects mined from GitHub as dataset to fine-tune a LLM agent.
- To create a dataset for API-related code completion, API usages are removed from the benchmark projects. The removed API calls are used as ground truth to be predicted from the model.
- For test generation, tests are removed from the projects.


Project (owner/name)	Commit	LoC	Stars*
Python projects used for code completion			
graphql-python/graphene	57cbef6	9,484	7.8k
geopy/geopy	ef48a8c	10,000	4.3k
nvbn/thefuck	ceeaeab	12,181	83.3k
aaugustin/websockets**	ba1ed7a	14,186	5k
arrow-py/arrow	74a759b	14,402	8.6k
lektor/lektor	be3c8cb	16,852	3.8k
Parsely/streamparse	aabd9d0	26,214	1.5k
Supervisor/supervisor	ca54549	29,860	8.3k
mwaskom/seaborn	f9827a3	37,367	12.1k
psf/black	ef6e079	106,005	37.6k
scikit-learn/scikit-learn	f3c6fd6	193,863	58.5k
JavaScript projects used for test generation			
node-red/node-red	29ed5b2	60	18.8k
winstonjs/winston	c63a5ad	496	22.2k
prettier/prettier	7142cf3	916	48.5k
tj/commander.js	83c3f4e	1,134	26.3k
js-sdsl/js-sdsl	055866a	1,198	0.7k
goldfire/howler.js	003b917	1,319	23.1k
websockets/ws	b73b118	1,546	21.2k
handlebars-lang/handlebars.js	8dc3d25	2,117	17.8k
petkaantonov/bluebird	df70847	3,105	20.4k
hapijs/joi	5b96852	4,149	20.7k
Unitech/pm2	a092db2	5,048	40.9k
11ty/eleventy	e71cb94	5,772	16.4k

* As of June 5, 2024


** Has been moved to python-websockets/websockets

Software repository-based datasets: DeHallucinator


- The whole codebase of (several) projects is used for fine-tuning!





















erikwrede

release: 3.4.1



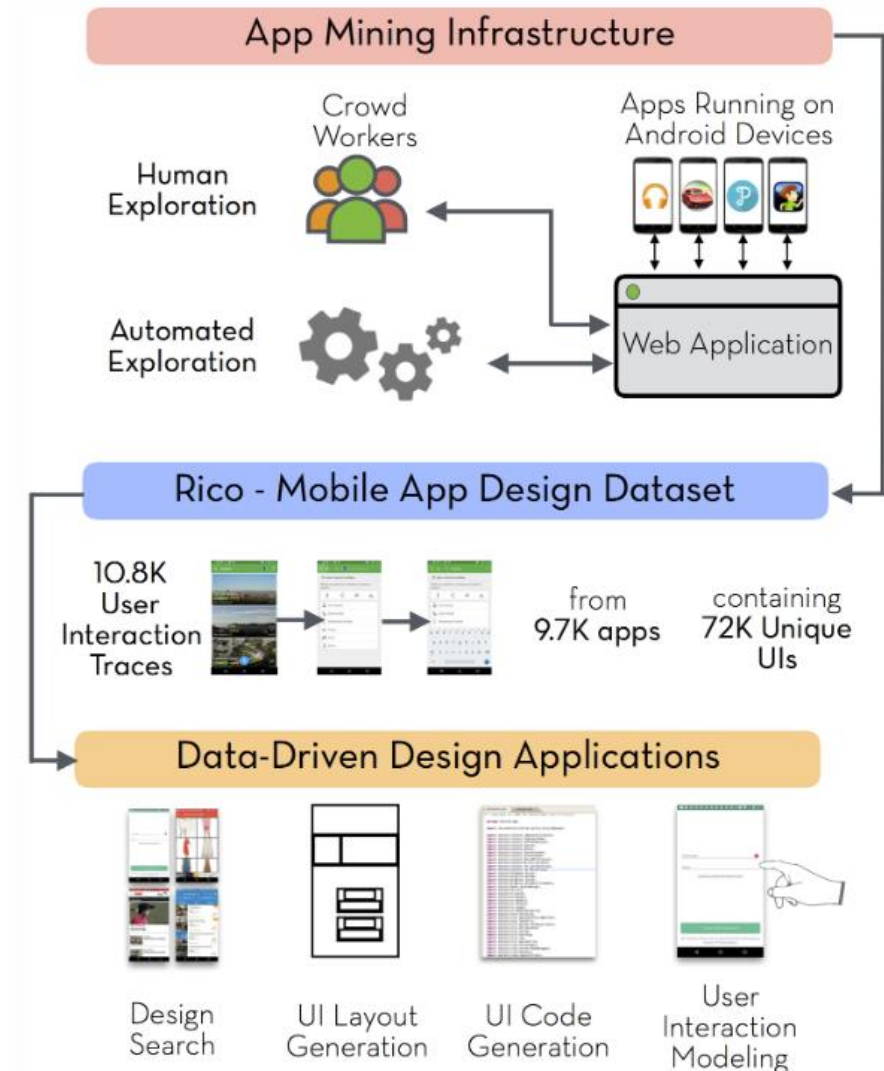
ccae736 · last week

 1,783 Commits

 .github	housekeeping: switch 3.13 to non-dev	3 weeks ago
 bin	Lint using Ruff (#1563)	5 months ago
 docs	fix: run the tests in python 3.12 and 3.13 and remove snapsh...	3 months ago
 examples	fix: run the tests in python 3.12 and 3.13 and remove snapsh...	3 months ago
 graphene	release: 3.4.1	last week
 .coveragerc	Fixed coverage	8 years ago
 .editorconfig	Add isort precommit hook & run on all files (#743)	6 years ago
 .gitignore	CI: format check using Ruff (#1557)	5 months ago
 .pre-commit-config.yaml	lint: use ruff pre commit hook (#1566)	2 months ago
 LICENSE	Updated license	8 years ago
 MANIFEST.in	Include license in manifest for source bundles	7 years ago
 Makefile	feat: Add support for custom global (Issue #1276) (#1428)	2 years ago
 README.md	chore: remove travis ci link	last year
 SECURITY.md	fix lint error in SECURITY.md (#1556)	5 months ago
 UPGRADE-v1.0.md	docs: Fix a few typos	2 years ago
 UPGRADE-v2.0.md	docs: Fix a few typos	2 years ago
 mypy.ini	Added mypy static checking	7 years ago
 setup.cfg	Lint using Ruff (#1563)	5 months ago
 setup.py	fix: use dateutil-parse for < 3.11 support (#1581)	last week
 tox.ini	fix: use dateutil-parse for < 3.11 support (#1581)	last week

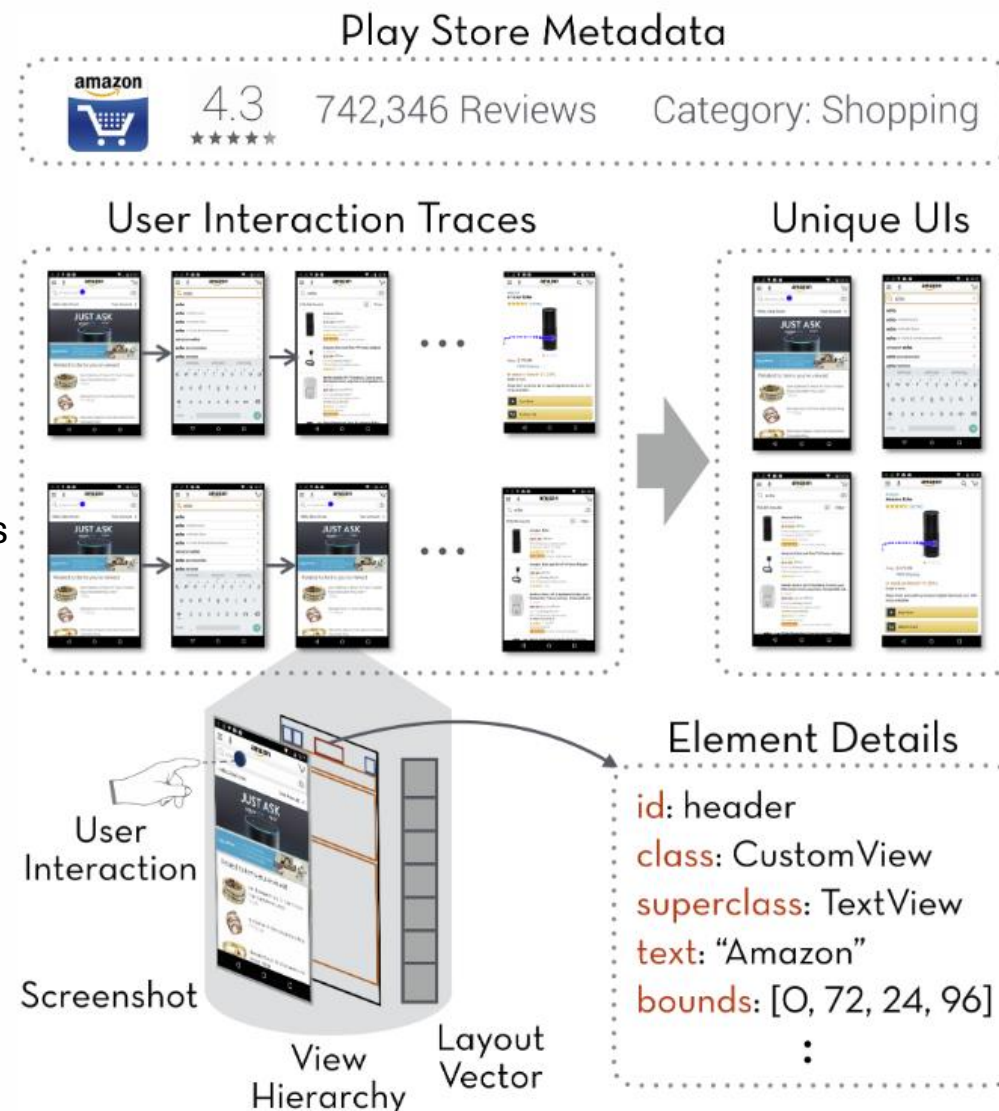
Graph-based datasets: the RICO dataset

- A dataset built by mining Android apps at runtime by programmatic and human exploration

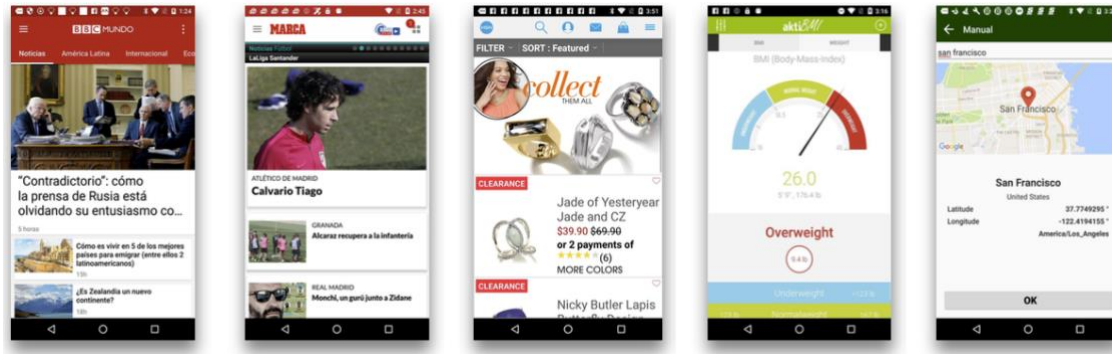


Graph-based datasets: the RICO dataset

- For each app, Rico exposes:
 - Google play store metadata (app's category, average rating, number of ratings, number of downloads)
 - A set of user interaction traces (screenshot + augmented Android view hierarchy, set of explored user interactions, set of effects in response to user interactions, learned vector representation of the UI's layout)
 - A list of all the unique UIs discovered



Graph-based datasets: the RICO dataset

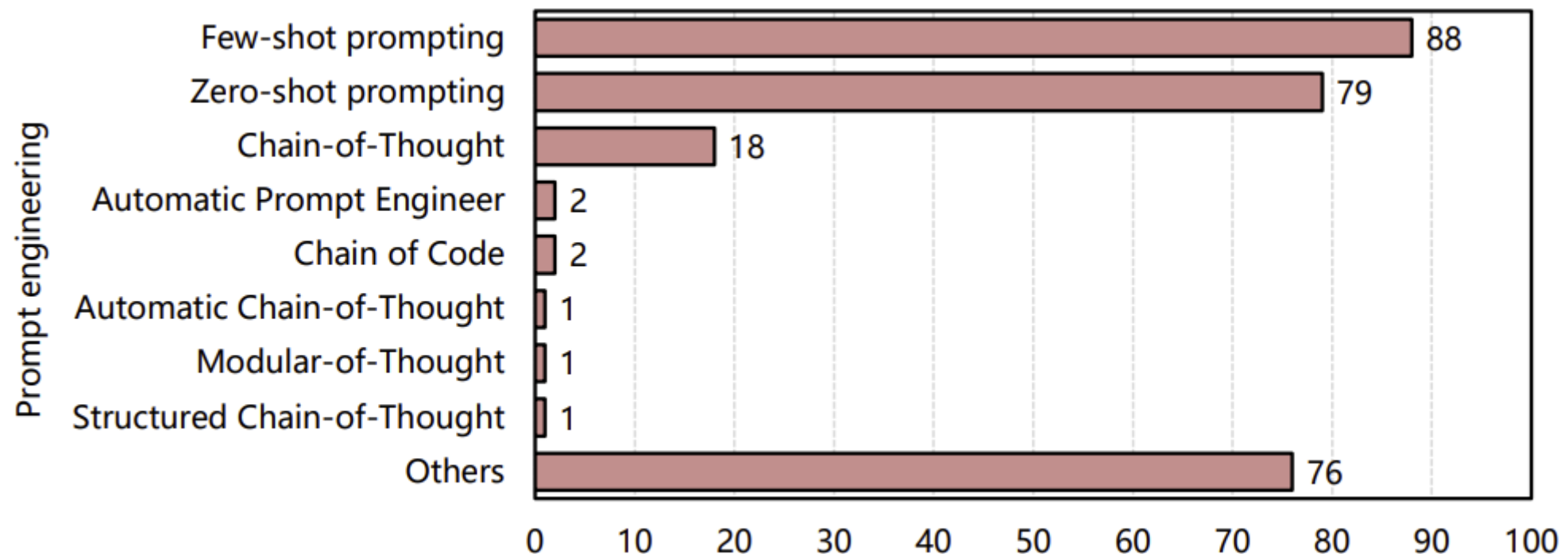


```
tivity_name": "com.funforphones.android.chicagocita/com.funforphones.android.chicagocita",
"activity": {
  "root": {
    "scrollable-horizontal": false,
    "draw": true,
    "ancestors": [
      "android.widget.FrameLayout",
      "android.view.ViewGroup",
      "android.view.View",
      "java.lang.Object"
    ],
    "clickable": false,
    "pressed": "not_pressed",
    "focusable": false,
    "long-clickable": false,
    "enabled": true,
    "bounds": [
      36,
      84,
      1404,
      2392
    ],
    "visibility": "visible",
    "content-desc": [
      null
    ],
    "rel-bounds": [
      0,
      0,
      1368,
      2308
    ],
    "focused": false,
```

Prompt Engineering

- Prompt engineering is a method of enhancing model performance by using task-specific instructions, known as prompts, without modifying the core model parameters (more on this later...)
- Several prompt engineering techniques have been efficiently applied in the LLM4SE domain.

Prompt Engineering



Evaluation Metrics to assess LLM4SE

- The evaluation metrics to use to assess LLM4SE tasks are related to the type of activity that is performed by the LLM agent.
 - For classification tasks, the most commonly used metrics are Precision and F1-Score.
 - For recommendation tasks, MRR (Mean Reciprocal Rank) is the most frequent metric.
 - For generation task, metrics like BLEU and Pass@k are used.

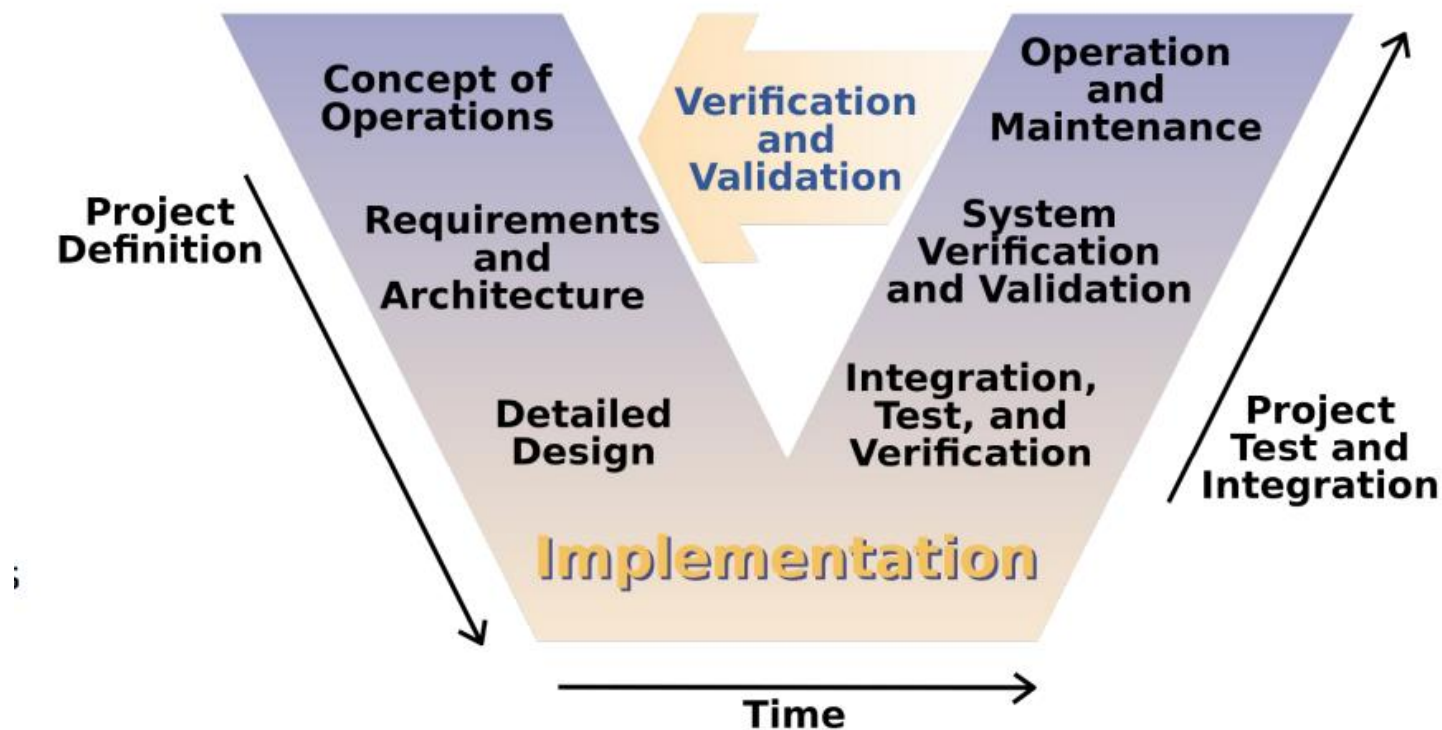
Evaluation Metrics to assess LLM4SE

Problem Type	Metric		Total
Regression	MAE (Mean Absolute Error) (1)		1
Classification	Precision (35) F1-score (33) AUC (Area Under the ROC Curve) (9) FPR (False Positive Rate) (4) MCC (Matthews Correlation Coefficient) (2)	Recall (34) Accuracy (23) ROC (Receiver Operating Characteristic) (4) FNR (False Negative Rate) (3)	147
Recommendation	MRR (Mean Reciprocal Rank) (15) MAP/MAP@k (6) Recall/Recall@k (4)	Precision/Precision@k (6) F-score/F-score@k (5) Accuracy (3)	39
Generation	BLEU/BLEU-4/BLEU-DC (62) Accuracy/Accuracy@k (39) CodeBLEU (29) Precision (18) Recall (15) MRR (Mean Reciprocal Rank) (6) ED (Edit Distance) (5) ChrF (3) CodeBERTScore (2) PP (Perplexity) (1)	Pass@k (54) EM (Exact Match) (36) ROUGE/ROUGE-L (22) METEOR (16) F1-score (15) ES (Edit Similarity) (6) MAR (Mean Average Ranking) (4) CrystalBLEU (3) MFR (Mean First Ranking) (1)	338

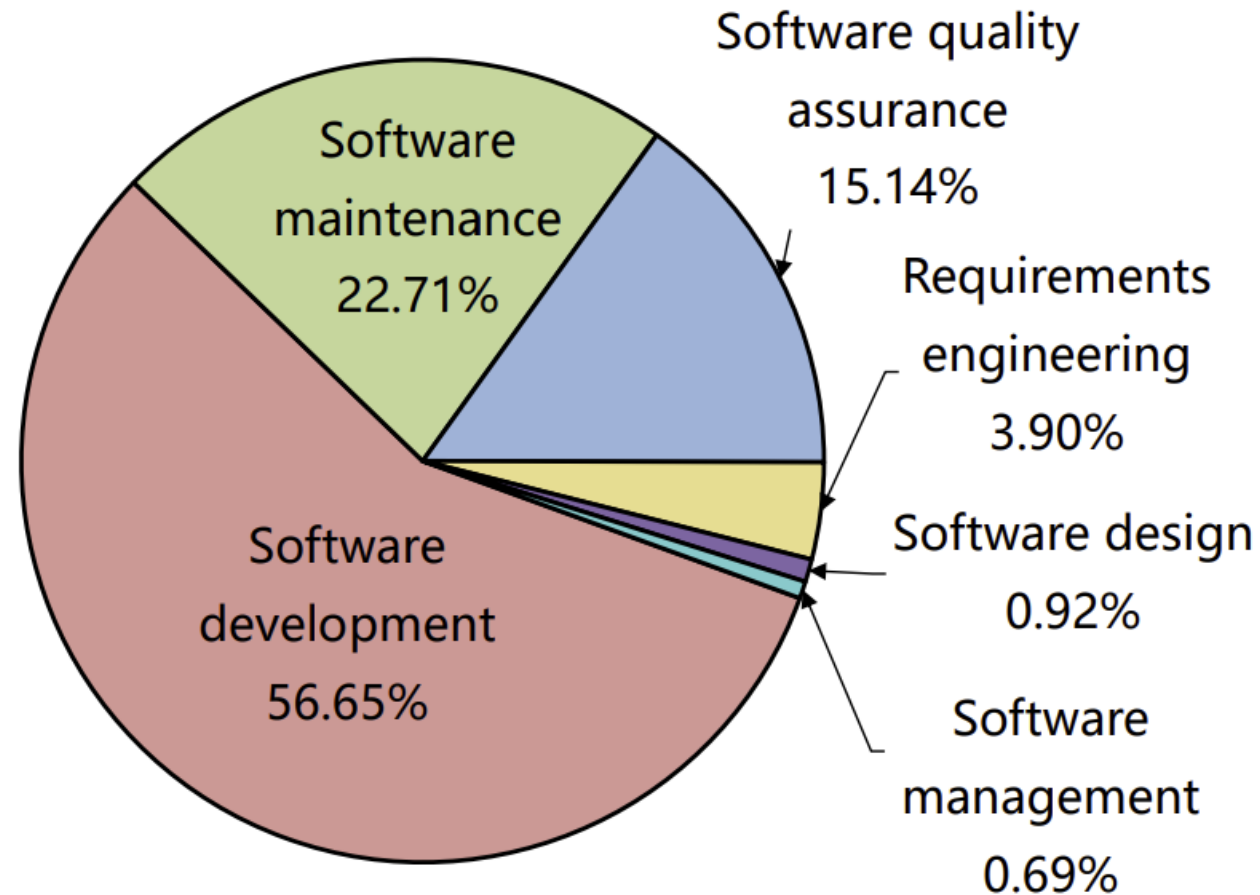


SE Tasks

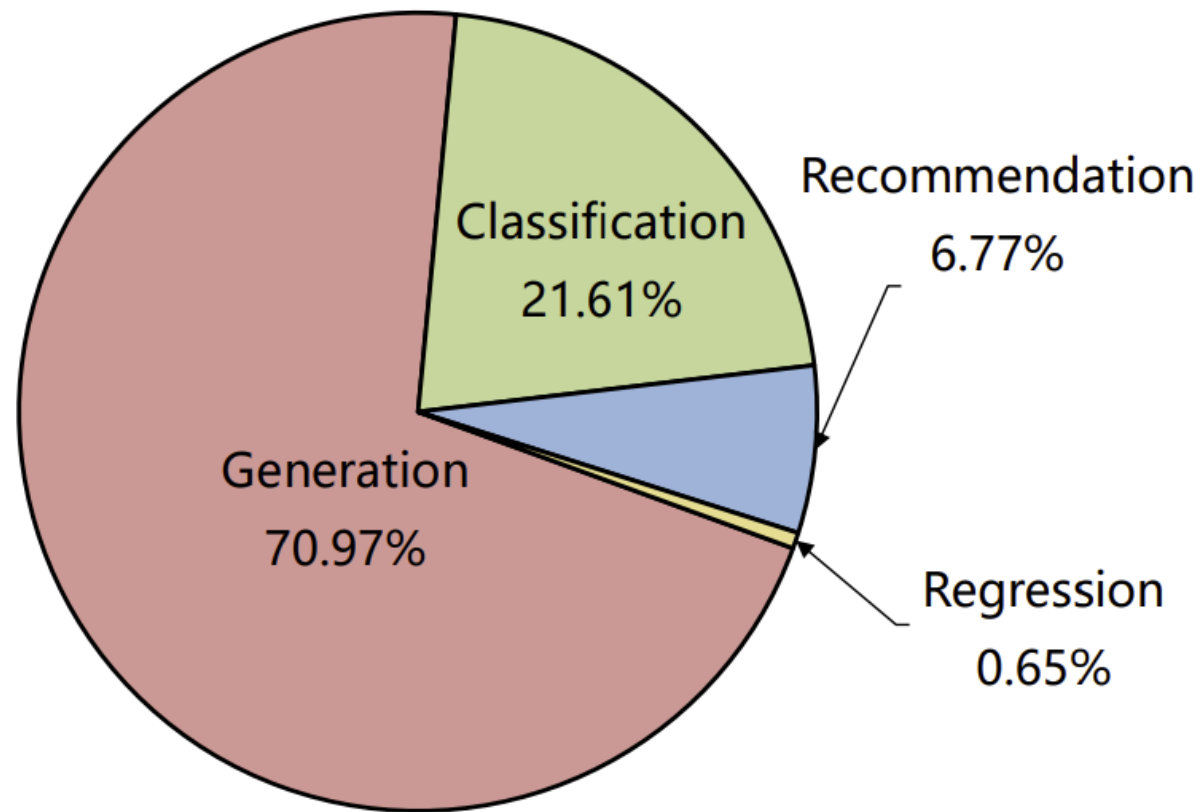
Let's consider the V-model



Distribution of LLM usages in SE Activities



Problem classification



Distribution of LLM usages in SE Activities

- The primary focus to date is to utilize LLMs to enhance coding and development processes.
- LLMs are frequently used also to aid software updates and improvements (i.e., software maintenance)

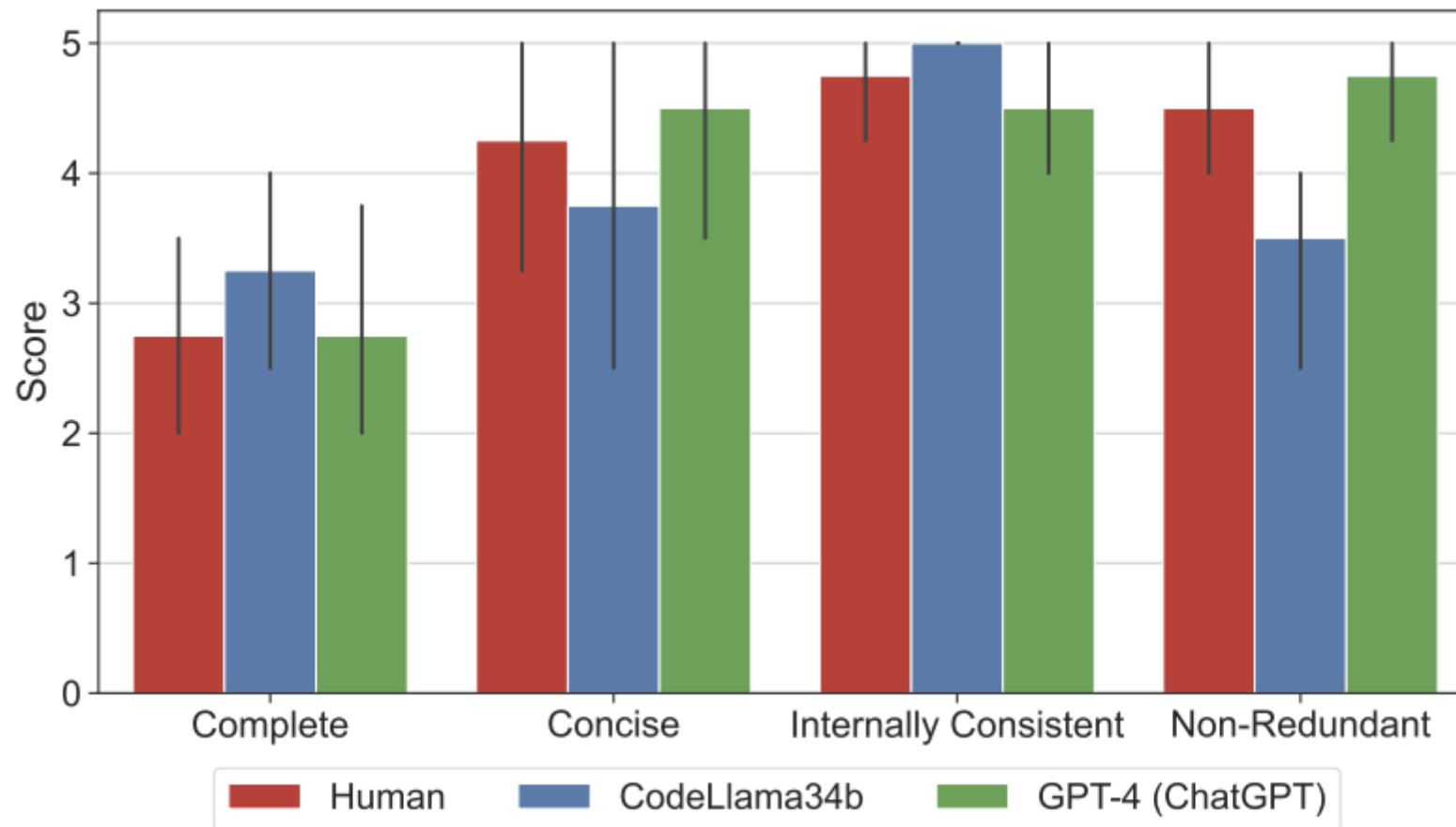
Requirements: Requirements elicitation

- Elicitation of a Software Requirements Specification (SRS) from natural language
- Generation by using pre-trained models (Code-LLAMA and GPT)
- Interpretation of the quality of the generated requirements
- After quality interpretation, the LLMs are asked again to correct the requirements.

Format for the SRS

1. Problem Background
2. Stakeholders
3. Functional Requirements
4. Performance Requirements
5. Design Constraints
6. External Interfaces
7. Security Requirements
8. Use cases for the application
 - Actor
 - Purpose
 - Event Flow
 - Special conditions
9. Glossary of terms

Requirements: Requirements elicitation



Requirements: Requirements elicitation

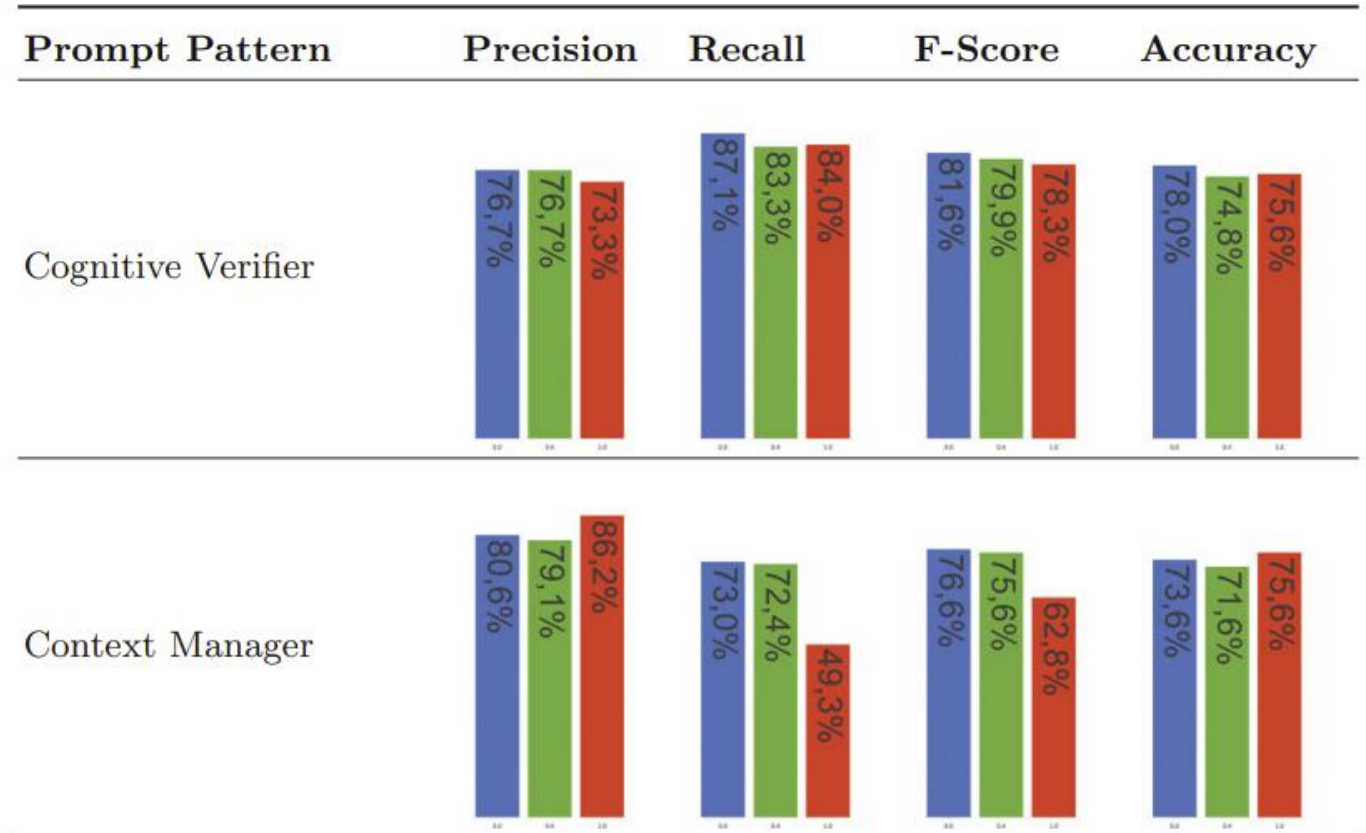
Section	Requirement Number	Human scores	GPT-4 scores	Old Requirements	New Requirements
Performance Requirements	4.1	1.75, 4.75, 4, 4.25	3, 4, 5, 3	Platform should withstand considerable amounts of traffic.	The platform must support up to 10,000 concurrent users and maintain functionality during peak usage times, such as event registration openings and result announcements.
Design Constraints	5.1	3.25, 4.75, 3.75, 3.5	3, 4, 5, 3	Develop with accessibility features like screen readers, font adjusters, etc.	The platform will include accessibility features such as screen readers compatibility, font size adjusters, high contrast mode, and keyboard navigation support.
Design Constraints	5.3	2.5, 4.75, 4.75, 3.25	3, 4, 5, 3	Easy to maintain with readable and well-documented code.	The platform's codebase shall adhere to industry-standard practices for readability, including clear naming conventions, modular design, and comprehensive documentation.
External Interfaces	6.1	4.5, 4, 4.5, 4.75	4, 5, 5, 4	Compatibility with major browsers.	The platform must be compatible with the latest two versions of major browsers including Chrome, Firefox, Safari, Edge, and Brave.
Security Requirements	7.4	4, 4.25, 4, 4.5	3, 4, 5, 3	User data management as per institute's policy.	User data shall be managed in compliance with the institute's Data Protection Policy, including provisions for data encryption, regular audits, and adherence to privacy regulations.

Requirements: Requirements classification

- Requirements demand effective classification, especially for early-stage project discernment (e.g., security-related ones).
 - Is this requirement functional or non-functional?
- Requirements are classified by using five different prompt strategies with GPT.

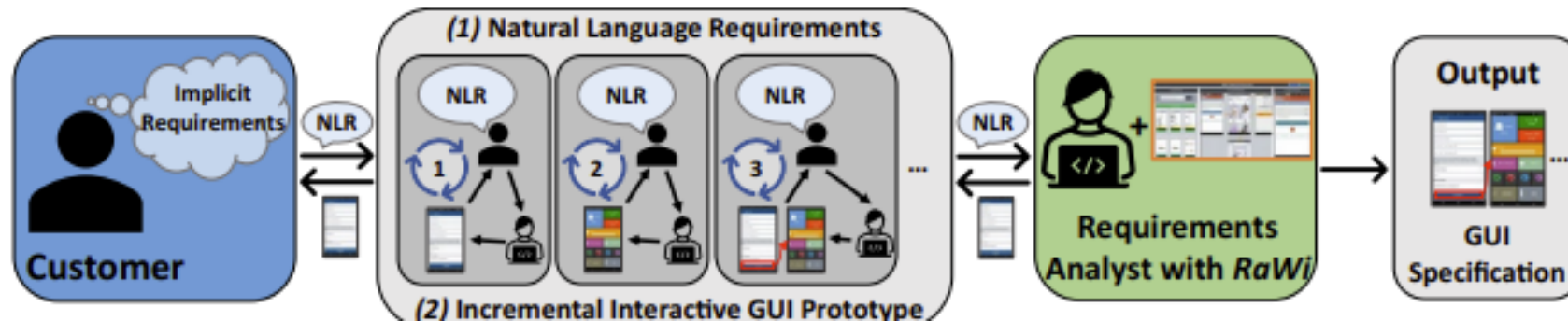
Requirements: Requirements classification

Pattern	Classification
Cognitive verifier	Classify the given list of requirements into functional (labelled as F) and non-functional requirements (labelled as NF). Ask me questions if needed to break the given task into smaller subtasks. All the outputs to the smaller subtasks must be combined before you generate the final output
Context manager	Classify the given list of requirements into functional (labelled as F) and non-functional requirements (labelled as NF). When you provide an answer, please explain the reasoning and assumptions behind your response. If possible, address any potential ambiguities or limitations in your answer, in order to provide a more complete and accurate response

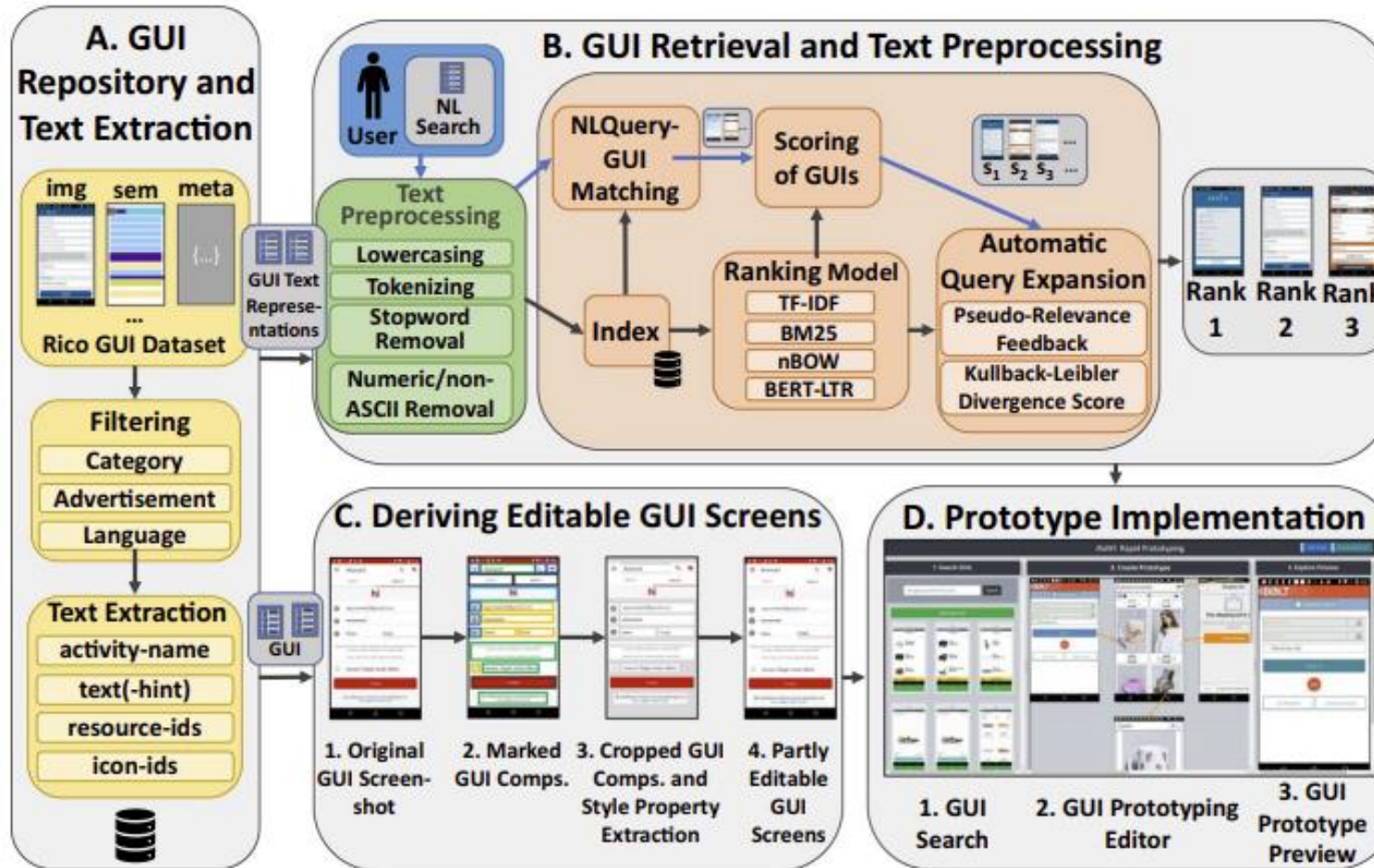


Design: GUI retrieval

- Kolthoff et al., 2022: a NL-based GUI prototyping approach to create high-fidelity GUI prototypes.



Design: GUI retrieval



Development: Code generation

- Research has demonstrated empirically that LLMs like GPT-4, BERT series, Codex, CodeGen, can play a key role in code generation.
- The models, since they are pre-trained on large scale text-data, learn rich linguistic knowledge and semantic representations that enable them to understand the meaning and structure of natural language.
- The models generate program code from natural language descriptions, enhancing code-writing efficiency and accuracy.

Development: Code generation

Requirement:

Return True is list elements are monotonically increasing or decreasing.

ChatGPT:

```
def monotonic(l: list):  
    if all(l[i] <= l[i+1] for i in range(len(l)-1)) or all(l[i]  
>= l[i+1] for i in range(len(l)-1)):  
        return True  
    else:  
        return False
```

TiP (Code sketch & Source Code):

Input: l: list

Output: True or False

```
1: for i from 1 to len(l)-1:  
2:     if l[i] > l[i-1] then  
3:         l increases  
4:     if l[i] < l[i-1] then  
5:         l decreases  
6: if both increase and decrease  
7:     return False  
8: else  
9:     return True
```

```
def monotonic(l: list):  
    increasing = False  
    decreasing = False  
    for i in range(1, len(l)):  
        if l[i] > l[i-1]:  
            increasing = True  
        if l[i] < l[i-1]:  
            decreasing = True  
    if increasing and decreasing:  
        return False  
    else:  
        return True
```

Development: Code generation

- Score is an integer ranging from 0 to 2 (bad, average, good)


Approach	Correctness	Code quality	Maintainability
GPT-2	0.077	0.332	0.291
CodeParrot	0.121	0.739	0.885
PyCodeGPT	0.205	1.137	1.255
InCoder	0.372	1.302	1.395
LLaMA	0.369	1.207	1.344
CodeGeeX	0.433	1.459	1.316
CodeGen	0.579	1.559	1.421
Codex	0.814	1.622	1.479
ChatGPT	1.119	1.653	1.552
TiP	1.342	1.839	1.803

Development: Code completion

- An assistive feature provided by many integrated development environments (IDEs) and code editors.
- The purpose is to automatically display possible code suggestions or options as the developers write code.
- Prominent models are Copilot and CodeGPT, pre-trained on extensive code datasets.

Development: Code completion

JS test.js 1 ●

JS test.js >  calculateDaysBetweenDates

```
1 function calculateDaysBetweenDates(begin, end) {  
    var beginDate = new Date(begin);  
    var endDate = new Date(end);  
    var days = Math.round((endDate - beginDate) / (1000 * 60 * 60 * 24));  
    return days;  
}
```

2

Development: Code completion

		GitHub Copilot	Tabnine	Kite	IntelliCode
Current word completion		✓	✓	✓	✓
Generating a program continuation:	Word	✓	✓	✓	✓
	Line	✓	✓	✓	
	Section of source code	✓			
API exploration			✓	✓	✓
API proposal		✓	✓	✓	✓
Naming variables		✓			
Source code generation based on natural language		✓	✓		
Learning the model on your own source code (for enterprises)		✓	✓	✓	✓

Development: Code completion

- Evaluation metrics (human judgement):
 - SUS: System Usability Scale [0, 100] to evaluate the usability of the system.
 - User Experience measurement (UEQ) [-3, 3] to evaluate the pragmatic aspects of the user experience. A questionnaire with 8 standardized questions.
 - Net Promoter Score (NPS) [0,10] to gauge the likelihood of participants recommending the intelligent assistant

	Mean	Standard Deviation
SUS	65.9	16.01
UEQ	1.1	1.37
NPS	6.81	2.57

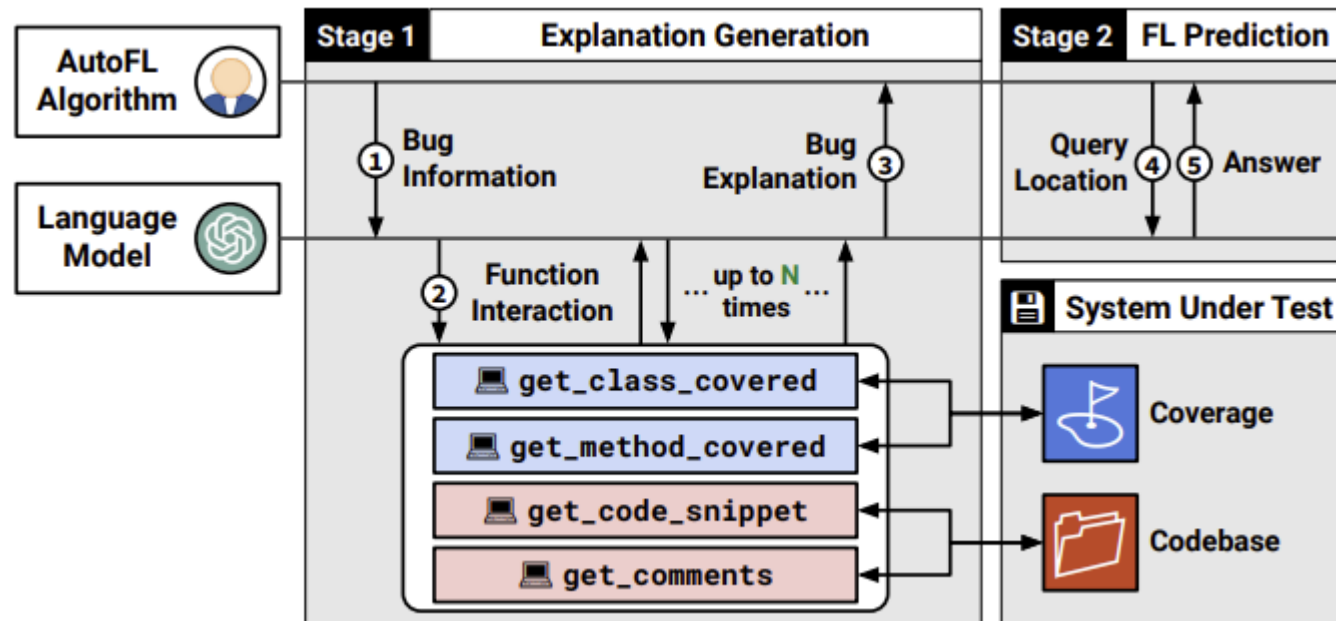
Development: Code summarization

- Code summarization is the task of understanding the code and automatically generate descriptions directly from the source code – an extended form of documentation.
- Successful code summarization facilitates the maintenance of source code, but can also be used to improve the performance of code search utilizing natural language queries.
- LLMs such as Codex, CodeBERT and T5 comprehend the functionality and logic of the code, producing easily understandable language descriptions.

Quality Assurance: Bug Localization

- Bug localization refers to the process of identifying the specific source code files, functions, or lines of code that are responsible for a reported bug or software defect.
- It typically involves analyzing bug reports or issue descriptions provided by users or testers and correlating them with the relevant portions of the source code.

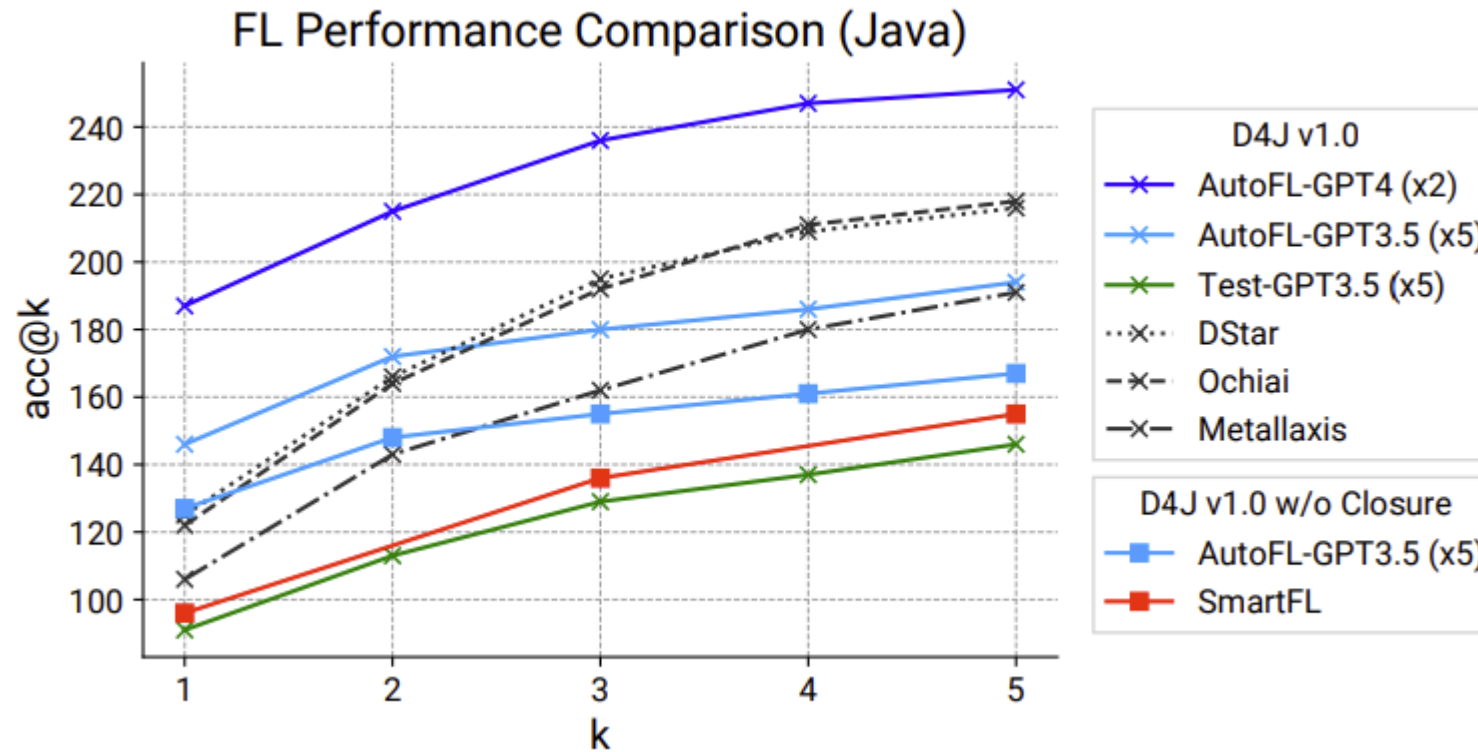
Quality Assurance: Bug Localization



Based on the available information, provide the signatures of the most likely culprit methods for

- ↪ the bug. Your answer will be processed automatically, so make sure to only answer with the
- ↪ accurate signatures of the most likely culprit (in ``ClassName.MethodName(ArgType1,`
- ↪ `ArgType2, ...)`` format), without commentary (one per line).

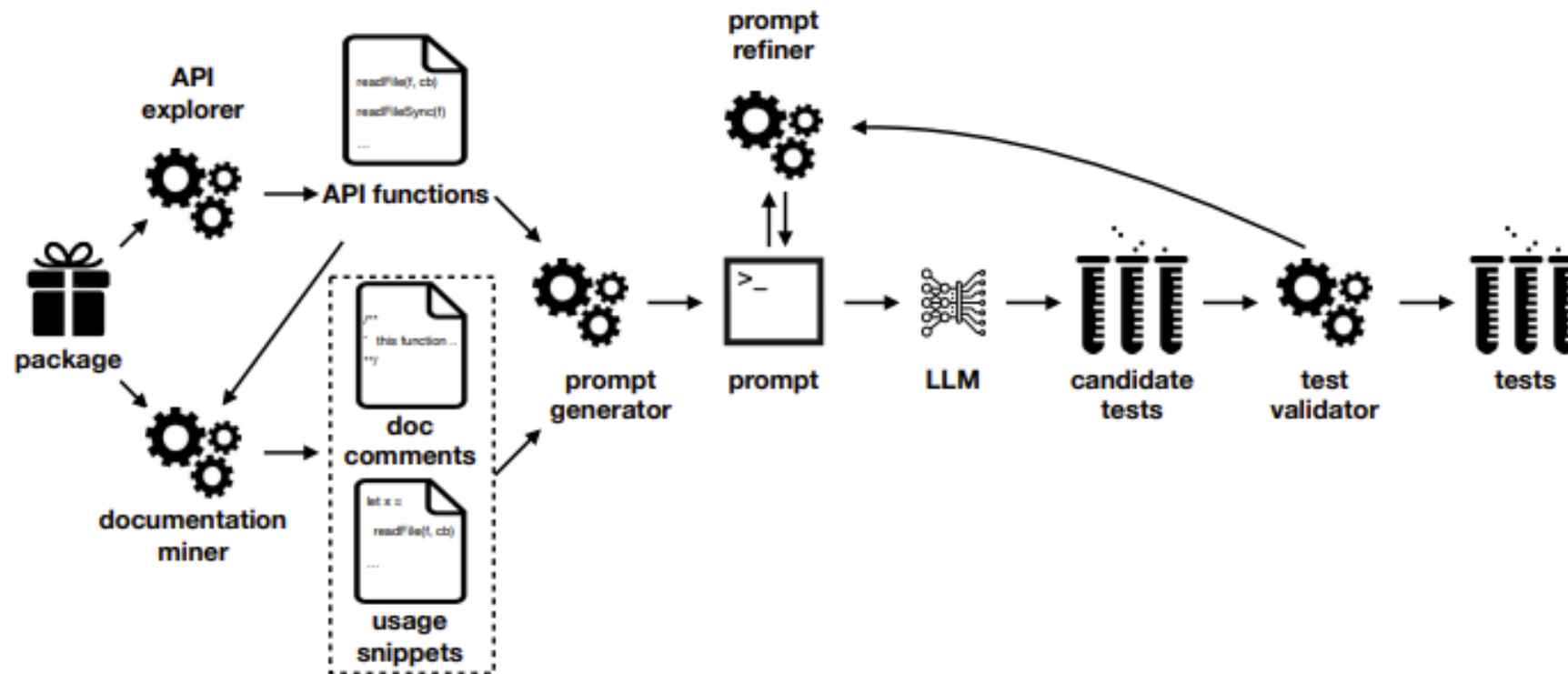
Quality Assurance: Bug Localization



Quality Assurance: Test Generation

- Test generation involves automating the process of creating test cases, to evaluate the correctness and functionality of software applications
- It encompasses different levels of testing (more on this later...)
- LLM applications in test generation offer several advantages, i.e. automatically generating diverse test cases, improve coverage, identifying defects.

Quality Assurance: Test Generation



Schäfer, Max, et al. "An empirical evaluation of using large language models for automated unit test generation." IEEE Transactions on Software Engineering (2023).

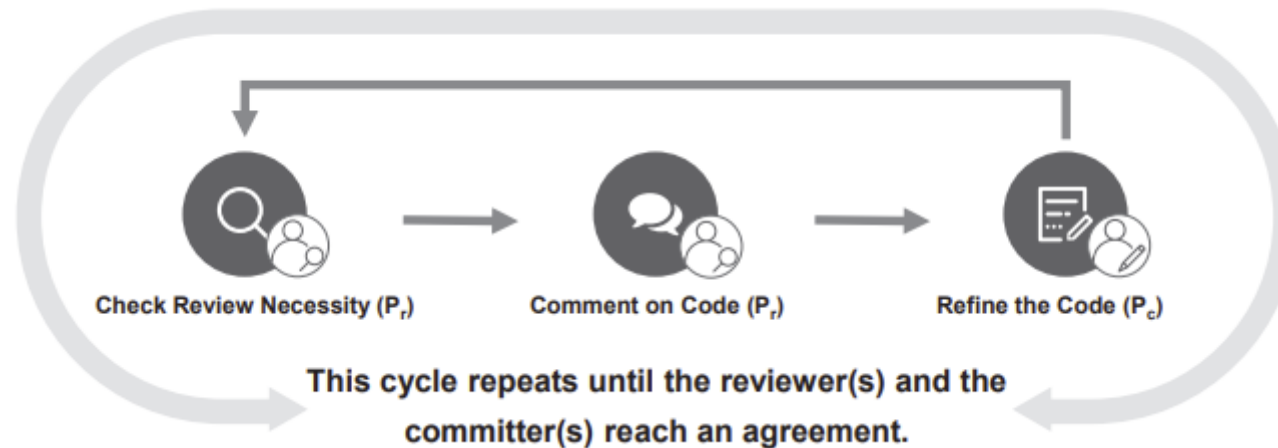
Quality Assurance: Test Generation

Project	Loading Coverage		TESTPILOT				
	Stmt Cov	Branch Cov	Total Tests	Passing Tests (%)	Stmt Cov	Branch Cov	Uniquely Contr. (%)
glob	7.0%	0.4%	68	18 (26.5%)	71.3%	66.3%	4 (22.2%)
fs-extra	16.8%	0.9%	471	277 (58.8%)	58.8%	38.9%	17 (6.1%)
graceful-fs	28.6%	9.8%	345	177 (51.4%)	49.3%	33.3%	1 (0.6%)
jsonfile	19.1%	0.0%	13	6 (48.0%)	38.3%	29.4%	0 (0.0%)
bluebird	23.7%	7.8%	370	204 (55.2%)	68.0%	50.0%	26 (12.5%)
q	22.4%	9.1%	323	186 (57.6%)	70.4%	53.7%	20 (10.5%)
rsvp	16.4%	12.6%	109	70 (64.2%)	70.1%	55.3%	6 (7.9%)
memfs	29.3%	7.2%	1037	471 (45.4%)	81.1%	58.9%	40 (8.5%)
node-dir	5.9%	0.0%	40	19 (48.1%)	64.3%	50.8%	4 (21.1%)
zip-a-folder	16.0%	0.0%	11	6 (54.5%)	84.0%	50.0%	0 (0.0%)
js-sdsl	7.9%	3.7%	409	46 (11.3%)	33.9%	24.3%	18 (39.1%)
quill-delta	8.1%	1.6%	152	33 (21.7%)	73.0%	64.3%	8 (24.2%)
complex.js	8.4%	4.6%	209	121 (58.0%)	70.2%	46.5%	10 (8.3%)
pull-stream	18.1%	0.0%	83	34 (41.0%)	69.1%	52.8%	11 (32.4%)
countries-and-timezones	4.9%	0.0%	28	13 (46.4%)	93.1%	69.1%	2 (15.4%)
simple-statistics	2.6%	0.0%	353	250 (70.9%)	87.8%	71.3%	14 (5.4%)
plural	53.8%	0.0%	13	8 (61.5%)	73.8%	59.1%	1 (12.5%)
dirty	4.7%	0.0%	70	32 (45.3%)	74.5%	65.4%	2 (6.3%)
geo-point	12.2%	0.0%	76	50 (65.8%)	87.8%	70.6%	1 (2.0%)
uneval	9.4%	0.0%	7	2 (28.6%)	68.8%	58.3%	0 (0.0%)
image-downloader	24.2%	0.0%	5	4 (80.0%)	63.6%	50.0%	0 (0.0%)
crawler-url-parser	7.2%	1.3%	14	2 (14.3%)	51.4%	35.0%	2 (100.0%)
gitlab-js	26.9%	0.6%	141	14 (9.9%)	51.7%	16.5%	7 (46.4%)
core	16.1%	0.0%	85	13 (15.3%)	78.3%	50.0%	5 (38.5%)
omnitoool	19.2%	0.6%	1033	330 (31.9%)	74.2%	55.2%	90 (27.2%)
Median	16.1%	0.4%		48.0%	70.2%	52.8%	10.5%

Maintenance: Code Reviews

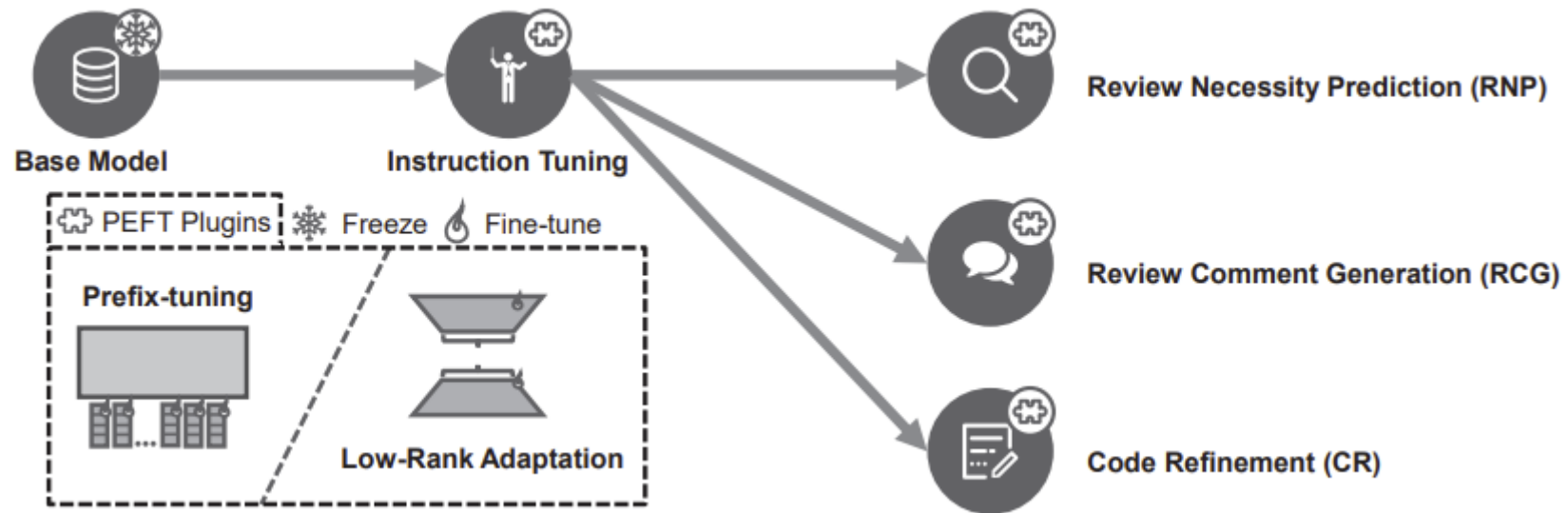
- Code review is a critical quality assurance practice used to inspect, assess and validate the quality and consistency of software code.
- Code review aims to identify potential errors, vulnerabilities and code quality issues, while also improving code maintainability, readability and scalability.
- LLMs like BERT, ChatGPT, and T5, trained on massive code repositories, possess the ability to understand and learn the semantics, structures, and contextual information of code.
- In the code review process, LLMs assist reviewers in comprehensively understanding code intent and implementation details, enabling more accurate detection of potential issues and errors.

Maintenance: Code Reviews



Lu, Junyi, et al. "LLaMA-Reviewer: Advancing code review automation with large language models through parameter-efficient fine-tuning." *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023.

Maintenance: Code Reviews



Maintenance: Code Reviews

TABLE V
RESULTS OF REVIEW COMMENT GENERATION.

Model	L.	Model Params	Trainable Params	Storage Space	BLEU-4	
					Crer.	Tuf.
Transformer-s	12	~60M	~60M	231M	–	6.94*
Transformer-b	24	~220M	~220M	850M	4.76	–
Tufano et al.	12	~60M	~60M	231M	4.39	7.39*
CodeT5	24	~220M	~220M	850M	4.83	–
CodeReviewer	24	~220M	~220M	850M	5.32	–
CommentFinder	–	–	–	~100M	3.82*	4.19*
AUGER	24	~220M	~220M	850M	–	3.03*
Ours (Prefix)	32	~6.7B	~1.2M	2.4M	5.16	4.66
Ours (LoRA)	32	~6.7B	~8.4M	16M	5.70	5.04



Challenges and opportunities

Applicability

- **Model size and deployment:** the size of LLMs has seen a marked increase over time (even if limited in novel approaches). Significant computational costs are associated with training LLMs.
- **Data dependency:** the quality, diversity, and quantity of data directly affect the performance and generalizability of the models. LLMs often require large amounts of data to capture nuances, but obtaining such data can be challenging.
- **Ambiguity in code generation:** when code intent is unclear, LLMs may struggle to produce accurate and contextually appropriate code. This can lead to syntactically correct but functionally incorrect code.

Generalizability

- Generalizability is the ability of a model to consistently and accurately perform tasks in different situations.
- The generalizability challenge is particularly evident in the SE domain, since context and semantics of code or documents vary greatly across projects, languages, or domains.

Evaluation

- Several key evaluation metrics are used in SE tasks.
- The metrics, while useful in some cases, may not fully capture all the effects and impacts of a model in a given SE task.

Interpretability, Trustworthiness, and Ethical usage

- It is often difficult to understand the decision-making process of the models, due to their black-box nature.
- LLM of code trained based on low-quality datasets can have vulnerabilities (e.g., insecure code)
- Many LLMs are not open and it is unclear what data they have been trained on, both quality and representativeness-wise.