# Are Large Language Models Good at Generating Software Specifications? Yes, but not Quite.

Danning Xie[1], Byungwoo Yoo[2], **Nan Jiang[1]**, Mijung Kim[2], Lin Tan[1], Xiangyu Zhang[1], Judy S Lee[3]

## Motivation

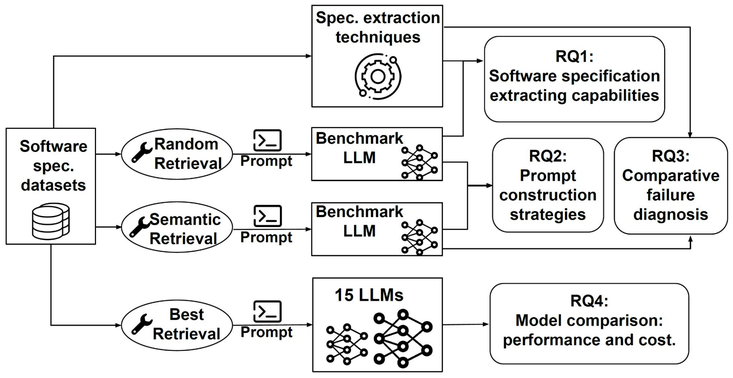
- Software specifications are essential for ensuring the reliability of software systems.
- Existing approaches on specifications extraction (from comments or documents) are domain-specific and semi-automatic.

**Function signature:** `isNullOrEmpty(java.lang.String string)`
**Javadoc comment:** `@return` true if the string is null or is an empty string
**Specification extracted by Jdoctor:**
`string==null || string.isEmpty() -> methodResultID==true`

Are LLMs effective in generating software specifications from documentation or comments?

What are the strengths and weaknesses of LLMs for software specification generation compared to traditional approaches?

## Study Overview




***Studied datasets and techniques:***

- **Jdoctor**: translates Javadoc comments (`@param, @returns, @throws`) into specifications
- **DocTer:** extracts DL-specific constraints (e.g., tensor shapes) from API documentation.

***Benchmark model* — Starcoder**

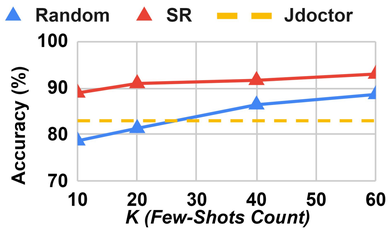
- 15.5 B, open-source, long input support (8,192 tokens)

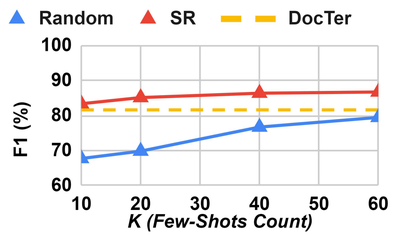
### Few-Shot Learning

Signature: <$x_1$ - signature>
Javadoc comment: <$x_1$ - comment>
Specification: <$y_1$>
...
Signature: <$x_K$ - signature>
Javadoc comment: <$x_K$ - comment>
Specification: <$y_K$>
Signature: <$x_{target}$ - signature>
Javadoc comment: <$x_{target}$ - comment>
Specification:

***Random Retrieval***: Randomly selecting K samples as the few-shots.
***Semantic Retrieval (SR):*** Applying a RoBERTa model as the semantic retrieval model to select the most semantically similar ***K*** samples as the few-shots.

## RQ1 & 2: Specification Extraction Capability




Starcoder, with 10–60 of randomly selected examples, achieves comparable results with the SOTA specification extraction tools.

*Semantic retrieval (SR) strategy* further improves Starcoder's performance to **outperforming SOTA approaches**.

## RQ4: Model Comparison

"-" denotes experiments skipped due to token limits.

| Approach/ Model (+SR) | | #param | open-source? | Overall Accuracy (%) K=10 | 20 | 40 | 60 | Cost ($) |
|---|---|---|---|---|---|---|---|---|
| **Jdoctor** | | | ✓ | 83.0 | | | | |
| **StarCoder** | | 15.5B | ✓ | **88.9** | **91.0** | **91.7** | **93.0** | 0 |
| **GPT-3** | davinci | 175B* | ✗ | **92.9** | **93.5** | **94.4** | **95.6** | 163.8 |
| | curie | Unknown | ✗ | 54.3 | 66.4 | - | - | 3.9 |
| **GPT-3.5** | turbo | Unknown | ✗ | **89.3** | **87.9** | **87.4** | **84.4** | 16.9 |
| **BLOOM** | | 176B | ✓ | **86.8** | - | - | - | 0 |
| **CodeGen (Multi)** | | 16B | ✓ | **86.4** | **88.4** | - | - | 0 |
| | | 6B | ✓ | **86.0** | **88.4** | - | - | 0 |
| | | 2B | ✓ | 82.8 | **87.4** | - | - | 0 |
| | | 350M | ✓ | 68.7 | 78.5 | - | - | 0 |
| **CodeGen2** | | 16B | ✓ | **86.5** | **89.0** | - | - | 0 |
| | | 7B | ✓ | **83.5** | **88.3** | - | - | 0 |
| | | 3.7B | ✓ | 70.0 | 80.4 | - | - | 0 |
| | | 1B | ✓ | 75.7 | 81.7 | - | - | 0 |
| **Incoder** | | 6B | ✓ | 52.7 | 61.6 | - | - | 0 |
| | | 1B | ✓ | 54.2 | 62.9 | - | - | 0 |

**Most LLMs achieve better or comparable performance** as custom-built traditional specification extraction techniques.

**StarCoder**, an open-sourced model, is the *most competitive* model for extracting specifications, with its *high performance, $0 cost, and long prompt support*, facilitating its adaptability and customization.

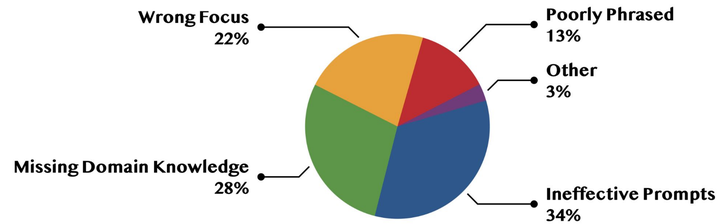
StarCoder's strong performance makes GPT3 Davinci less desirable given its size and cost. CodeGen and CodeGen2 are reasonable open-source alternatives.

## RQ3: Failure Root Cause Analysis

Manually sample and examine failing cases of both LLM and the baseline approaches to identify their ***unique failure root causes*** .

### Large Language Models

We identify the root causes of the LLM by manually fixing them.




***Ineffective prompts:*** The examples selected in the prompts are not good enough. Fixed by manually selecting more relevant examples, or altering the order of examples.

***Missing domain knowledge:*** LLM is lack of context while some traditional methods are search-based.

```
+ Relevant functions:                                   Additional domain knowledge
+   searchForDanger(int range,float threat)
+   ...
  ... (K examples)                                      prompt
  Signature: isInDanger(int range,float threat)
  Javadoc comment: @return True if a threat was found.
- Condition: this.getTile().isInDanger(range,threat) → methodResultID==true
+ Condition: this.searchForDanger(range,threat) → methodResultID==true
```

***Wrong focus:***

```
values: 1-D or higher numeric `Tensor`.
values: 1-D or higher `numeric` `Tensor`.
```

***Poorly Phrased:*** the original documents or comments are poorly written, ambiguous, or hard to understand even for humans.

***Other:*** "contradictory document" and "unclear"

### Baseline Approaches

- **Missing rule** (78%)
- **Incomplete Semantic Comprehension** (13.5%)
- **Incorrect Rule** (8.5%)

1 PURDUE UNIVERSITY
2 UNIST 3 IBM

Preprint