

Guiding Language Models of Code with Global Context using Monitors*

Author: Lakshya et al

Presentor: Dongjae Lee

20240502

* A Agrawal, L., Kanade, A., Goyal, N., Lahiri, S., & Rajamani, S. (2023, June). Monitor-Guided Decoding of Code LMs with Static Analysis of Repository Context. *NeurIPS 2023*. Retrieved from <https://www.microsoft.com/en-us/research/publication/guiding-language-models-of-code-with-global-context-using-monitors/>

Problems in Language Models

Problems in Language Models

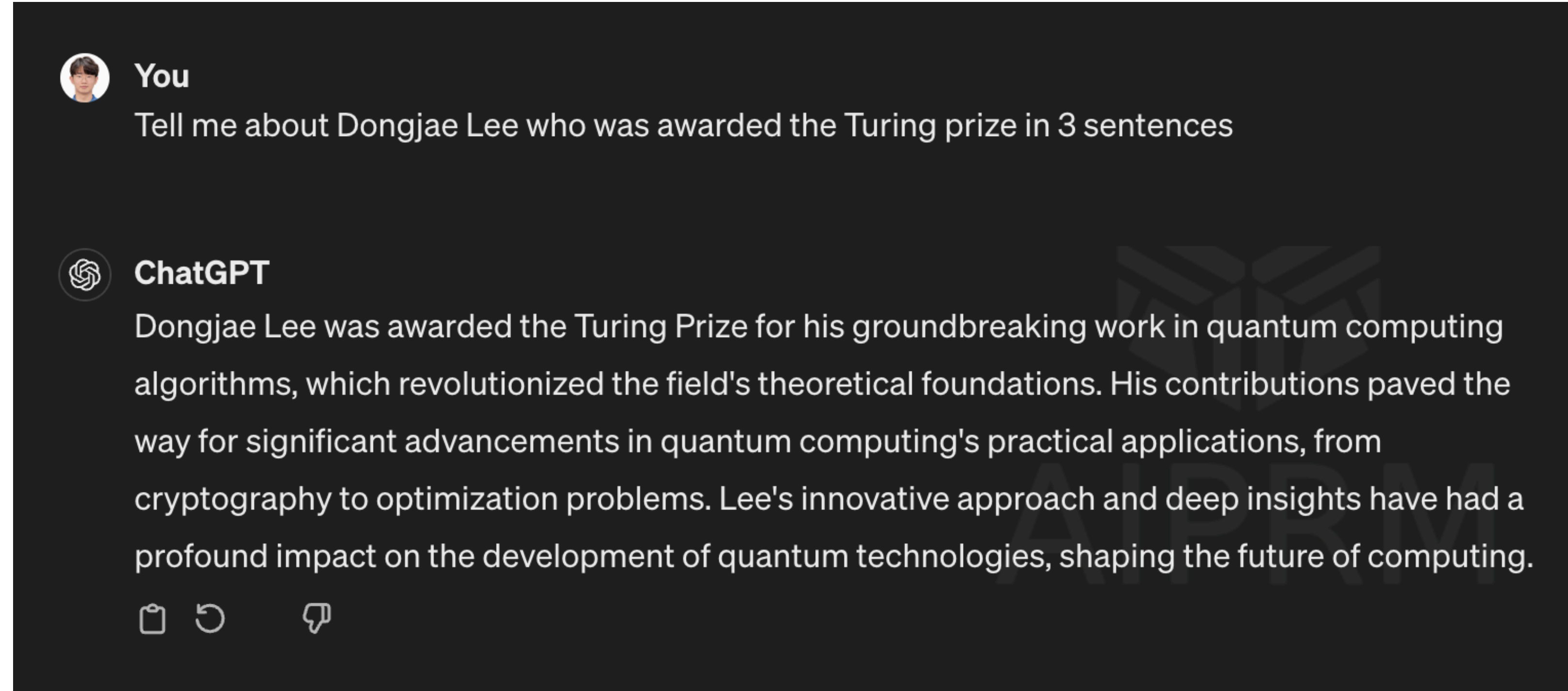
- The hallucination of LM (Language model)

Problems in Language Models

- The hallucination of LM (Language model)
 - The irrelevant answer, made up, or inconsistent with the input data

Problems in Language Models

- The hallucination of LM (Language model)
 - The irrelevant answer, made up, or inconsistent with the input data



Problems in Language Models

Problems in Language Models

- The hallucination of LM from the perspective of programming

Problems in Language Models

- The hallucination of LM from the perspective of programming
 - Syntax error

Problems in Language Models

- The hallucination of LM from the perspective of programming
 - Syntax error
 - Type error

Problems in Language Models

- The hallucination of LM from the perspective of programming
 - Syntax error
 - Type error
 - Undefined Variable

Problems in Language Models

- The hallucination of LM from the perspective of programming
 - Syntax error
 - Type error
 - Undefined Variable
 - Undefined Behavior

Problems in Language Models

- The hallucination of LM from the perspective of programming
 - Syntax error
 - Type error
 - Undefined Variable
 - Undefined Behavior
 - Memory errors (e.g. Buffer overflow)

Problems in Language Models

- The hallucination of LM from the perspective of programming
 - Syntax error
 - Type error
 - Undefined Variable
 - Undefined Behavior
 - Memory errors (e.g. Buffer overflow)
 - Semantic error (e.g. Inconsistency with given instruction)

Problems in Language Models

```
private ServerNode parseServer(string url) {  
  
    . . .  
  
    return ServerNode.Builder  
        .newServerNode()  
        .???  
  
}
```

Problems in Language Models

```
private ServerNode parseServer(string url) {  
    ...  
    return ServerNode.Builder  
        .newServerNode()  
        .host(arr[0])  
        .port(Integer.parseInt(arr[1]))  
        .build();  
}
```

Problems in Language Models

```
private ServerNode parseServer(string url) {
```

```
    . . .
```

```
    return ServerNode.Builder
```

```
        .newServerNode()
```

There is no such method in
the return object of newServerNode()

```
        .host(arr[0])
```

```
        .port(Integer.parseInt(arr[1]))
```

```
        .build();
```

```
}
```

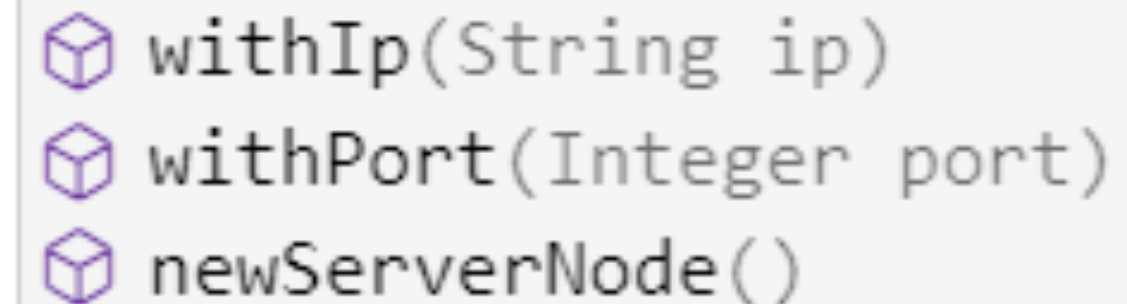

Problems in Language Models

```
private ServerNode parseServer(string url) {  
    ...  
    return ServerNode.Builder  
        .newServerNode()  
        .withIp(arr[0])  
        .withPort(Integer.parseInt(arr[1]))  
        .build();  
}
```

Problems in Language Models

```
private ServerNode parseServer(string url) {  
  
    ...  
  
    return ServerNode.Builder  
        .newServerNode()  
        .withIp(arr[0])  
        .withPort(Integer.parseInt(arr[1]))  
        .build();  
}
```

Analysis result of static analyzer



```
withIp(String ip)  
withPort(Integer port)  
newServerNode()
```

Cause of Hallucination

Cause of Hallucination

- Assume LM knows everything!!

Cause of Hallucination

- Assume LM knows everything!!
 - Concept of type

Cause of Hallucination

- Assume LM knows everything!!
 - Concept of type
 - Functionality of Library

Cause of Hallucination

- Assume LM knows everything!!
 - Concept of type
 - Functionality of Library
 - Understanding semantics

Cause of Hallucination

- Assume LM knows everything!!
 - Concept of type
 - Functionality of Library
 - Understanding semantics
 - ...

Cause of Hallucination

- Assume LM knows everything!!
 - Concept of type
 - Functionality of Library
 - Understanding semantics
 - ...

Are you sure?

Cause of Hallucination

Cause of Hallucination

- LMs are not clear on this knowledge

Cause of Hallucination

- LMs are not clear on this knowledge
- They have just a vague knowledge

Cause of Hallucination

- LMs are not clear on this knowledge
- They have just a vague knowledge
 - Because they are "probabilistic black box model"

Cause of Hallucination

- LMs are not clear on this knowledge
- They have just a vague knowledge
 - Because they are "probabilistic black box model"
 - There is no clear knowledge of the LMs

Cause of Hallucination

- LMs are not clear on this knowledge
- They have just a vague knowledge
 - Because they are "probabilistic black box model"
 - There is no clear knowledge of the LMs
- LM cannot know about your private repository's context

Cause of Hallucination

- LMs are not clear on this knowledge
- They have just a vague knowledge
 - Because they are "probabilistic black box model"
 - There is no clear knowledge of the LMs
- LM cannot know about your private repository's context
 - e.g. some Class defined in the other file in your repository

How to address the problem?

- The hallucination of LM from the perspective of programming
 - Syntax error
 - Type error
 - Undefined Variable
 - Undefined Behavior
 - Memory errors (e.g. Buffer overflow)
 - Semantic error (e.g. Inconsistency with given instruction)

How to address the problem?

- The hallucination of LM from the perspective of programming
 - Syntax error
 - Type error
 - Undefined Variable
 - Undefined Behavior
 - Memory errors (e.g. Buffer overflow)
 - Semantic error (e.g. Inconsistency with given instruction)

How to address the problem?

- The hallucination of LM from the perspective of programming
 - Syntax error **Parser**
 - Type error
 - Undefined Variable
 - Undefined Behavior
 - Memory errors (e.g. Buffer overflow)
 - Semantic error (e.g. Inconsistency with given instruction)

How to address the problem?

- The hallucination of LM from the perspective of programming
 - Syntax error
 - Type error
 - Undefined Variable
 - Undefined Behavior
 - Memory errors (e.g. Buffer overflow)
 - Semantic error (e.g. Inconsistency with given instruction)

How to address the problem?

- The hallucination of LM from the perspective of programming
 - Syntax error
 - Type error
 - Undefined Variable
 - Undefined Behavior
 - Memory errors (e.g. Buffer overflow)
 - Semantic error (e.g. Inconsistency with given instruction)

Static Analyzer

How to address the problem?

- The hallucination of LM from the perspective of programming
 - Syntax error
 - Type error
 - Undefined Variable
 - Undefined Behavior
 - Memory errors (e.g. Buffer overflow)
 - **Semantic error (e.g. Inconsistency with given instruction)**

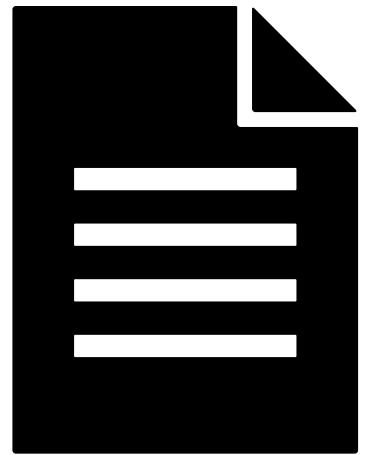
How to address the problem?

- The hallucination of LM from the perspective of programming
 - Syntax error
 - Type error
 - Undefined Variable
 - Undefined Behavior
 - Memory errors (e.g. Buffer overflow)
 - Semantic error (e.g. Inconsistency with given instruction)

Test code by LM

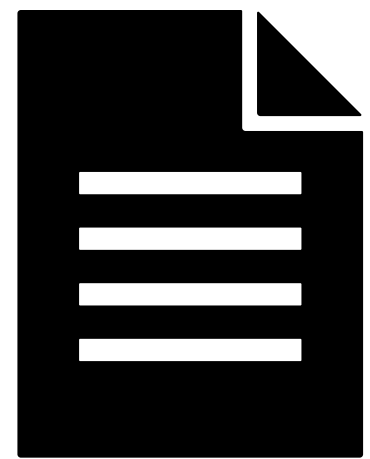
Utilize Static Analysis! (Naïve)

Utilize Static Analysis! (Naïve)



Prompt

Utilize Static Analysis! (Naïve)



Prompt



LM

Utilize Static Analysis! (Naïve)



Prompt



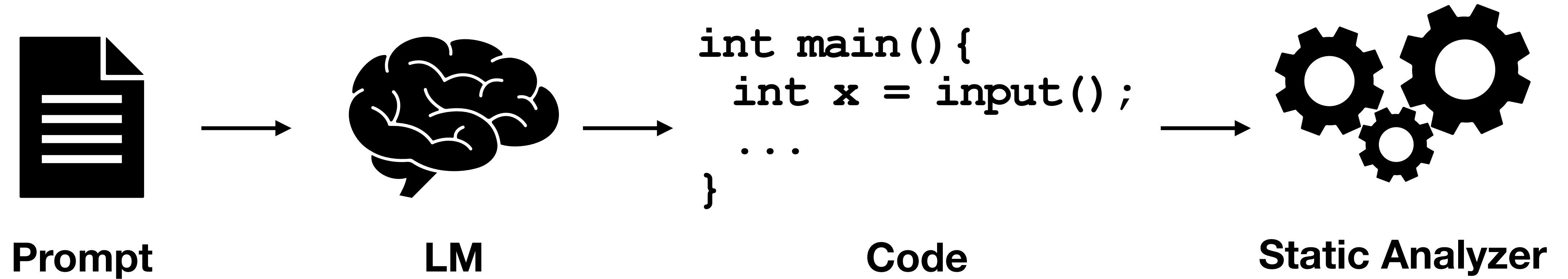
LM



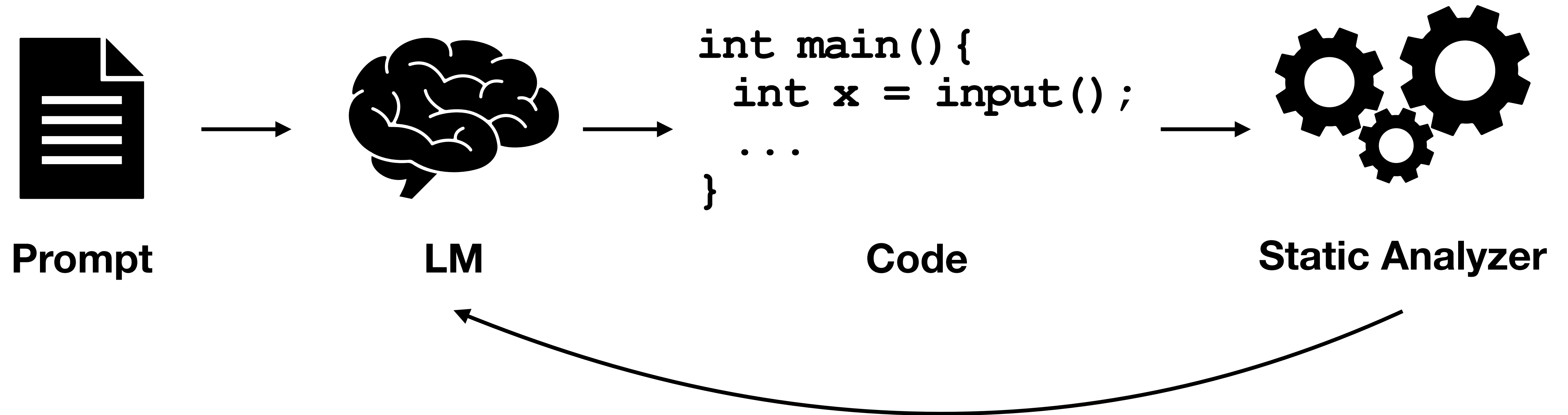
```
int main() {  
    int x = input();  
    ...  
}
```

Code

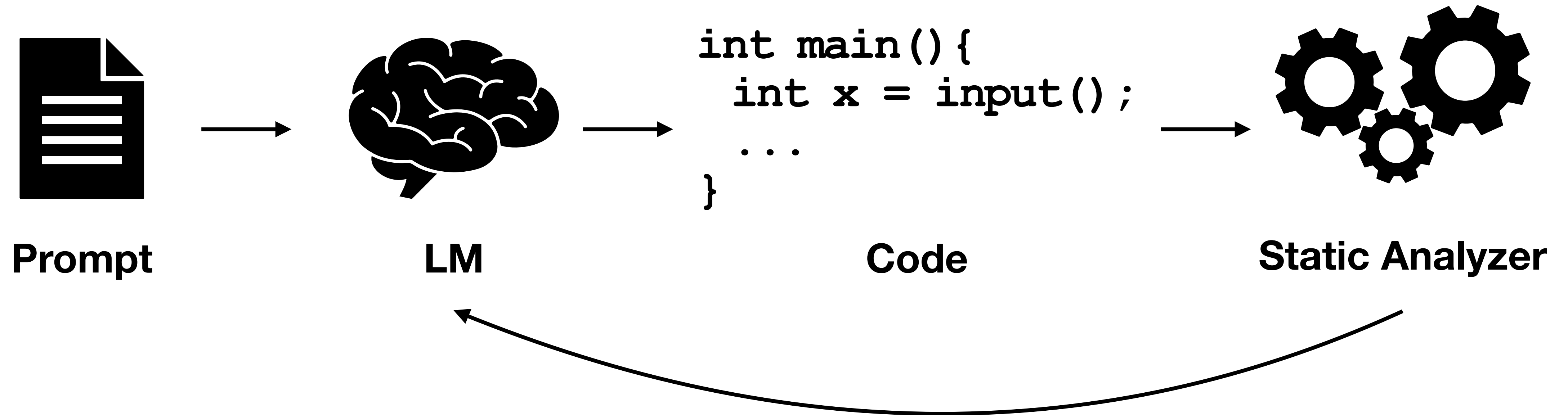
Utilize Static Analysis! (Naïve)



Utilize Static Analysis! (Naïve)

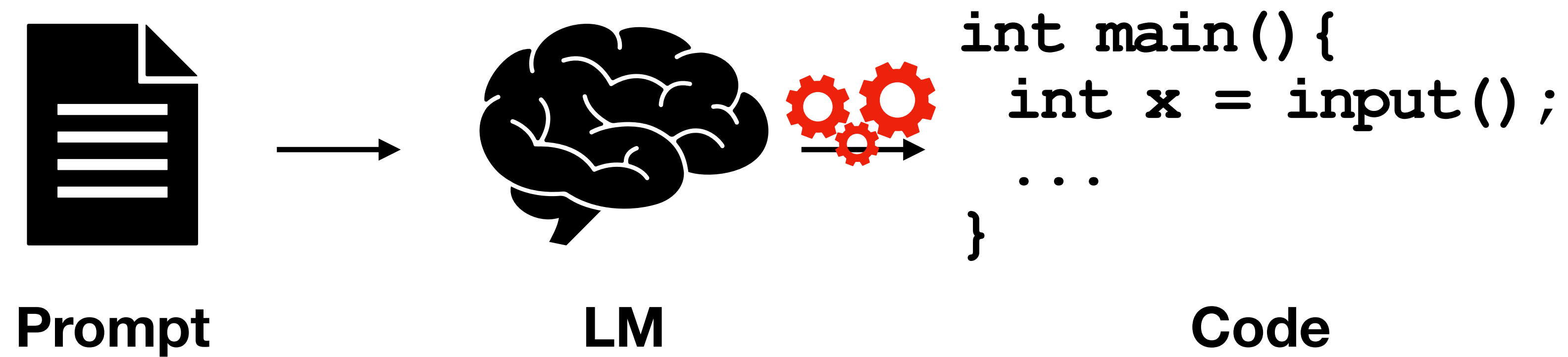


Utilize Static Analysis! (Naïve)

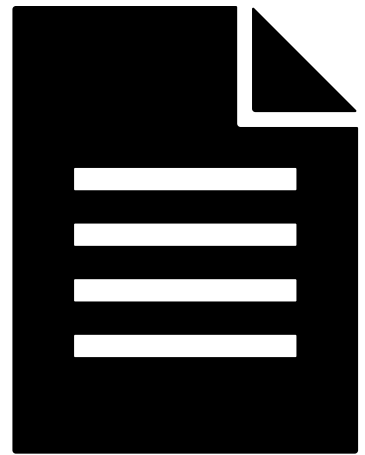


Inefficient!

Utilize Static Analysis! (Paper Version)

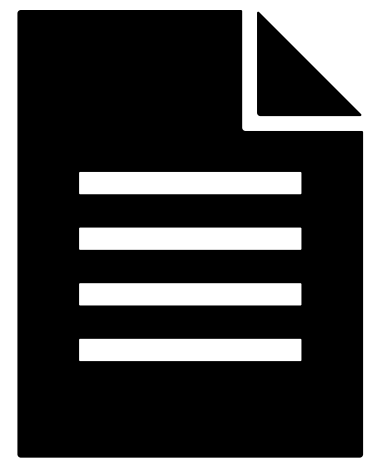


Utilize Static Analysis! (Paper Version)



Prompt

Utilize Static Analysis! (Paper Version)

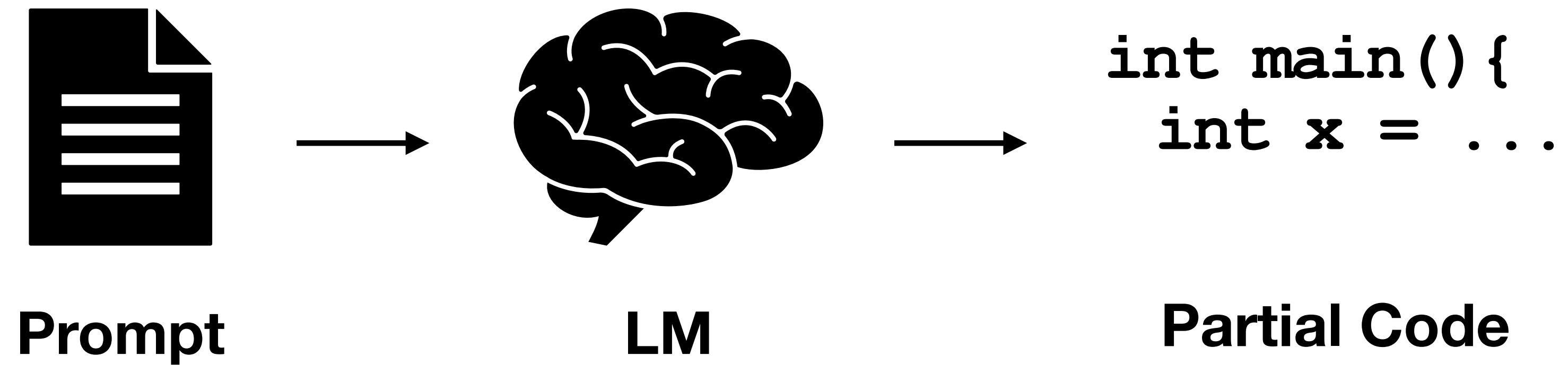


Prompt

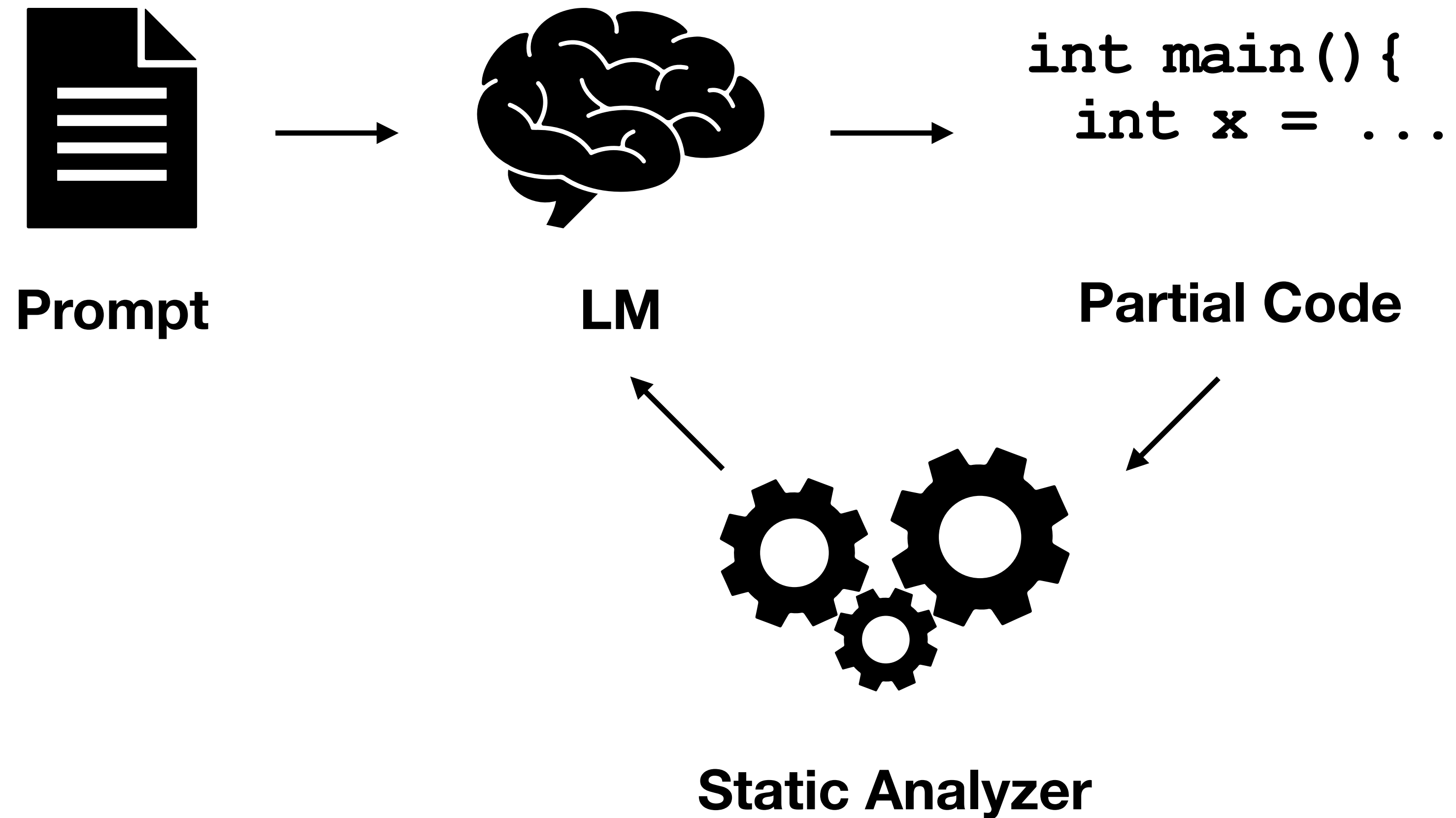


LM

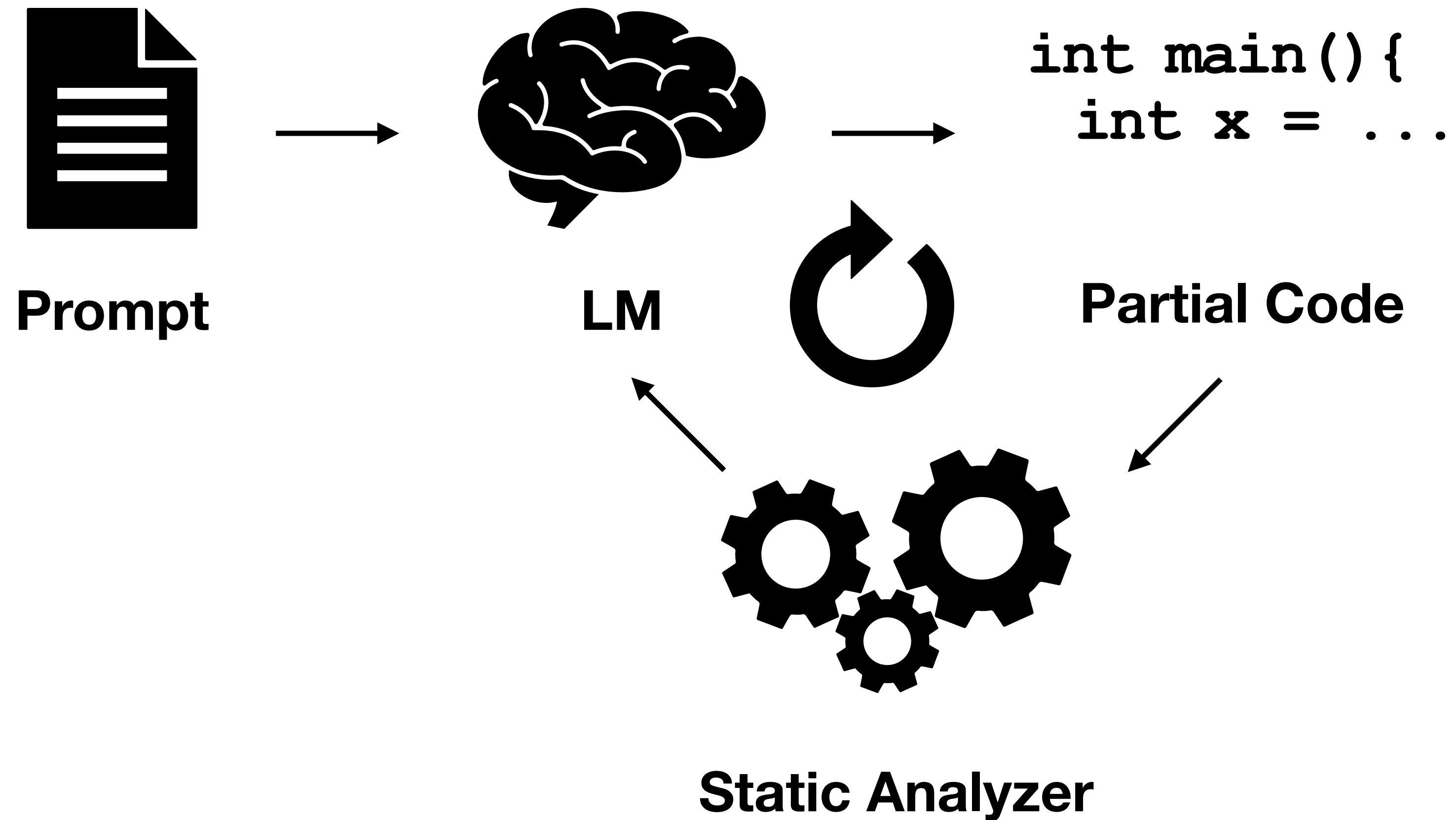
Utilize Static Analysis! (Paper Version)



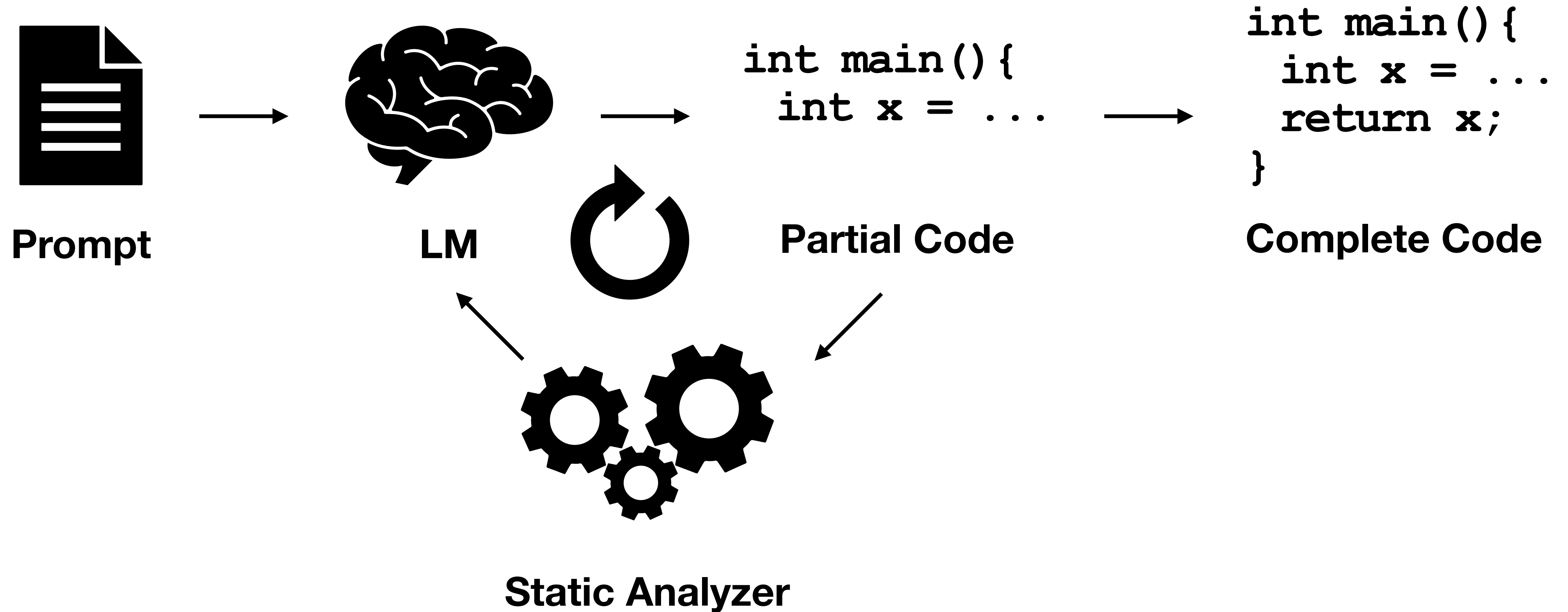
Utilize Static Analysis! (Paper Version)



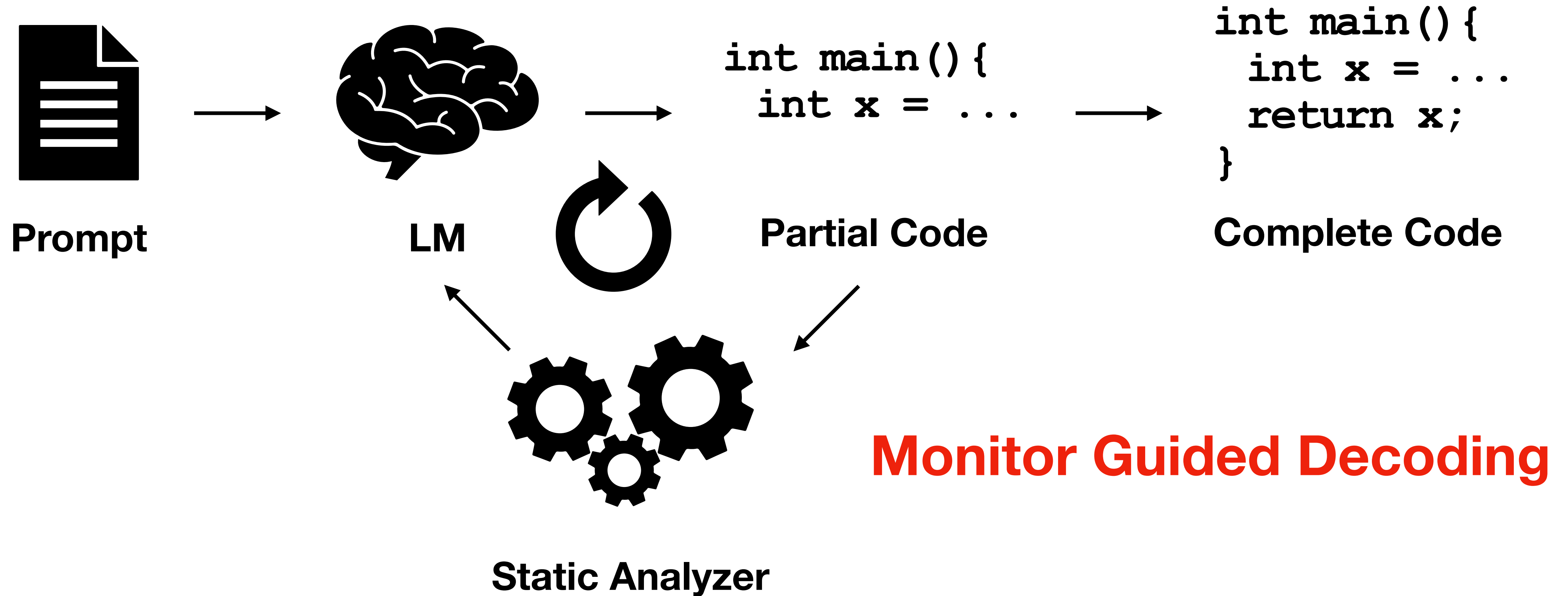
Utilize Static Analysis! (Paper Version)



Utilize Static Analysis! (Paper Version)

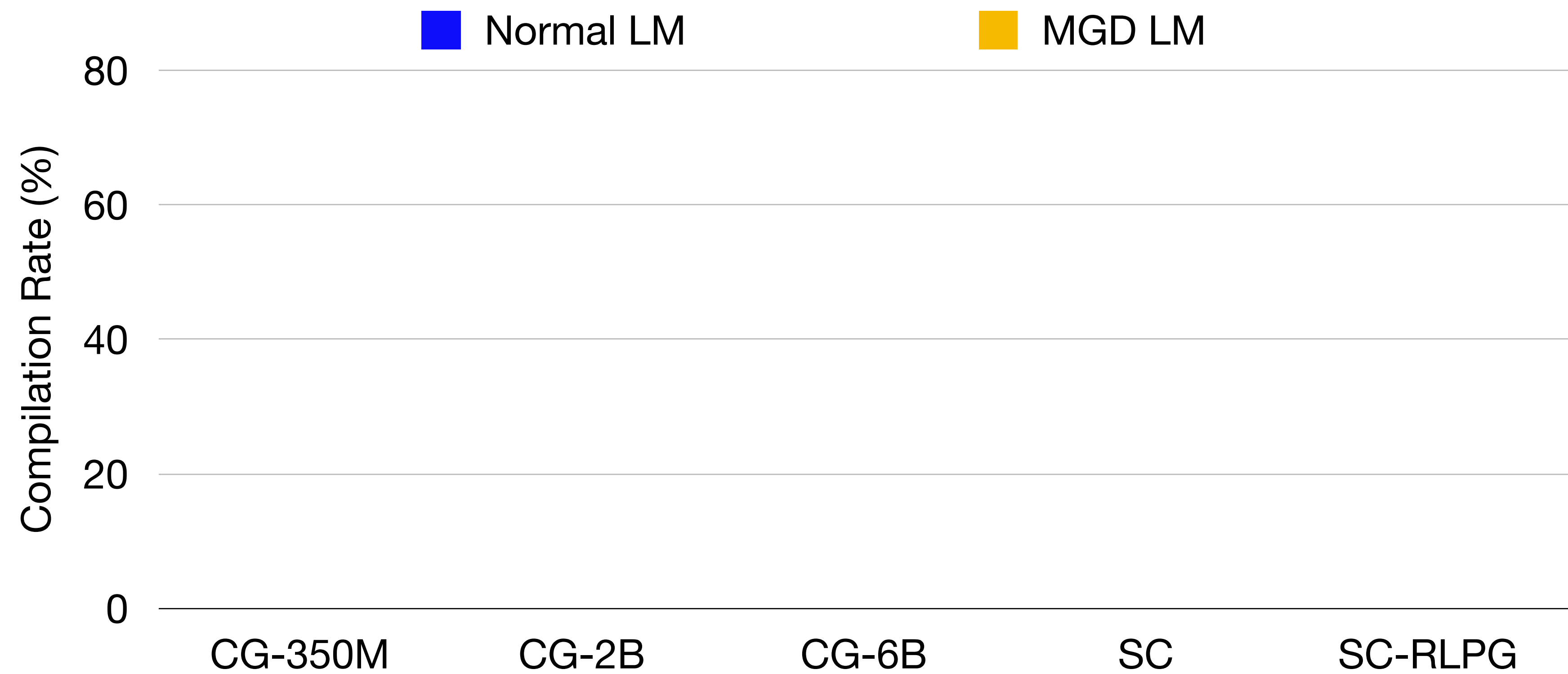


Utilize Static Analysis! (Paper Version)

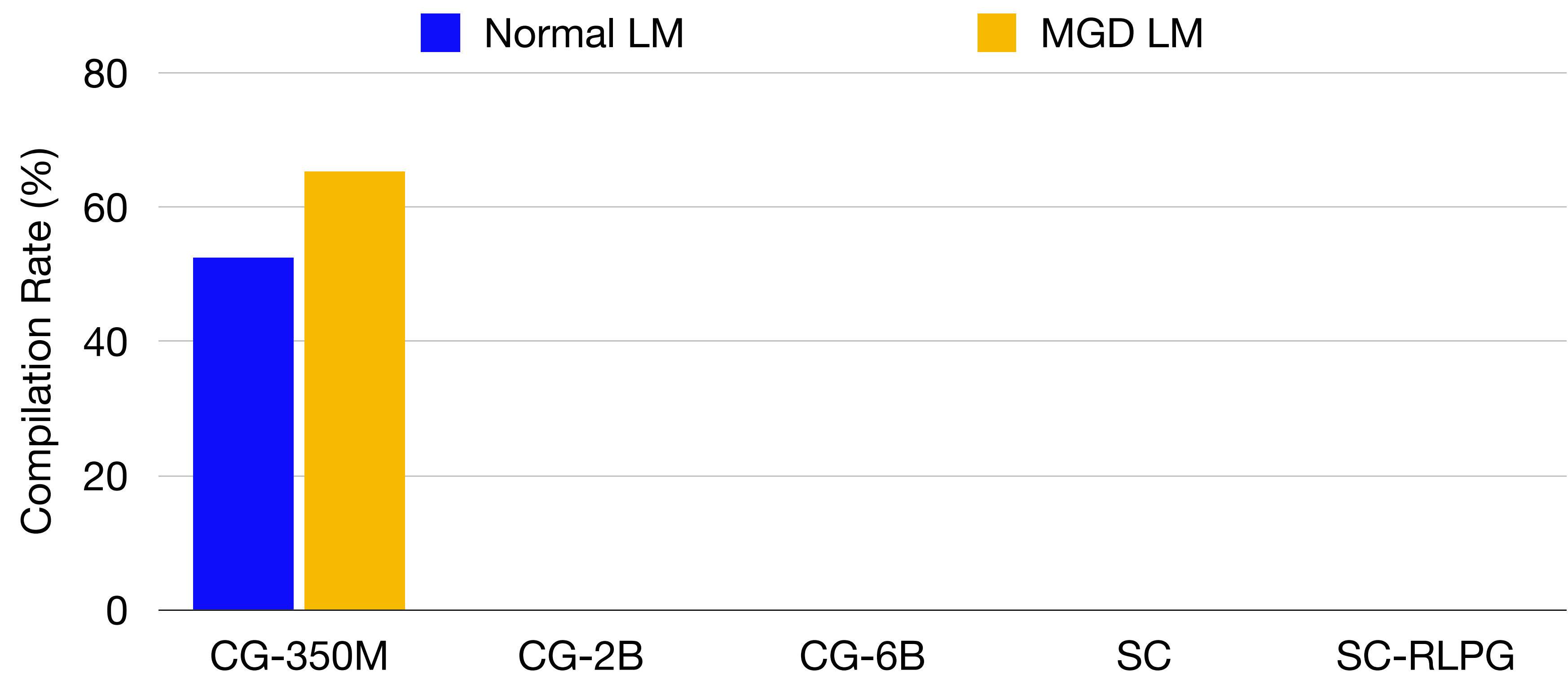


Result

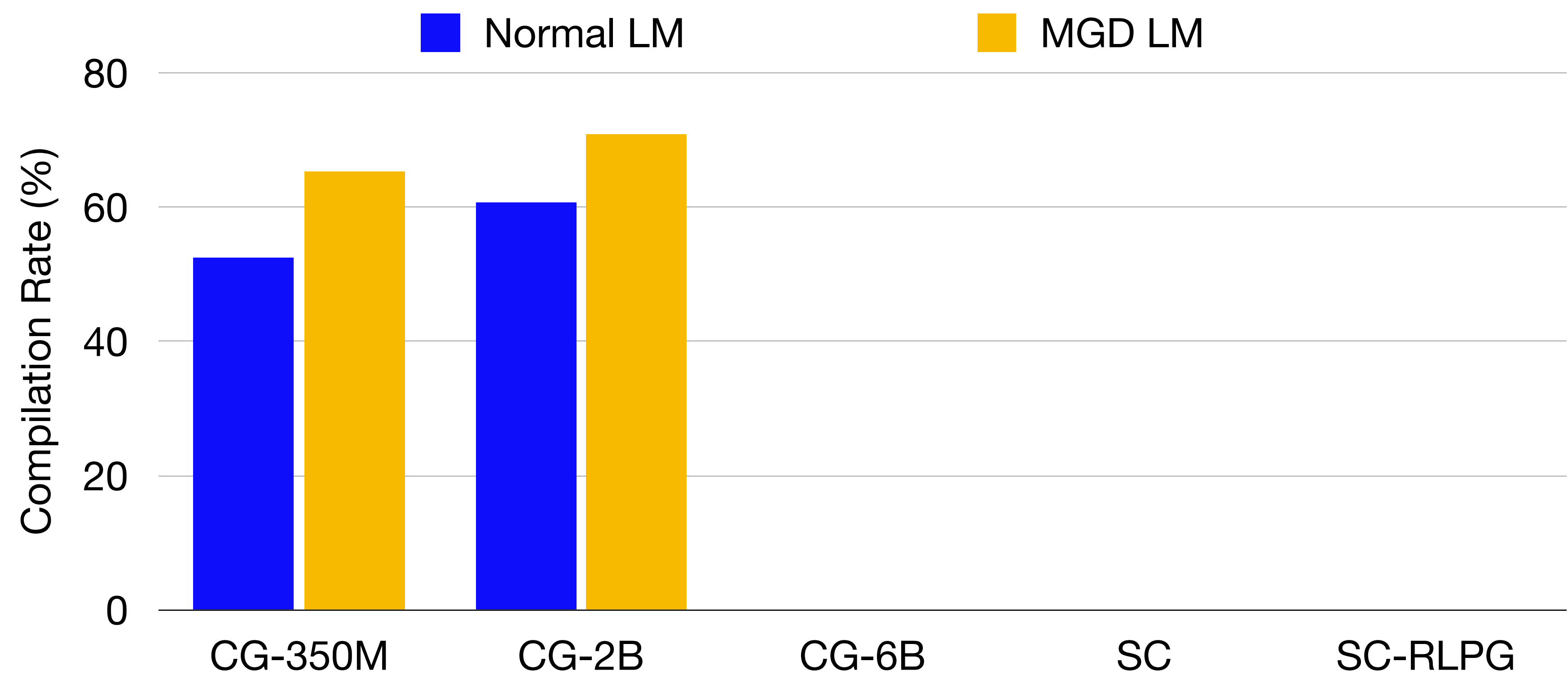
Result



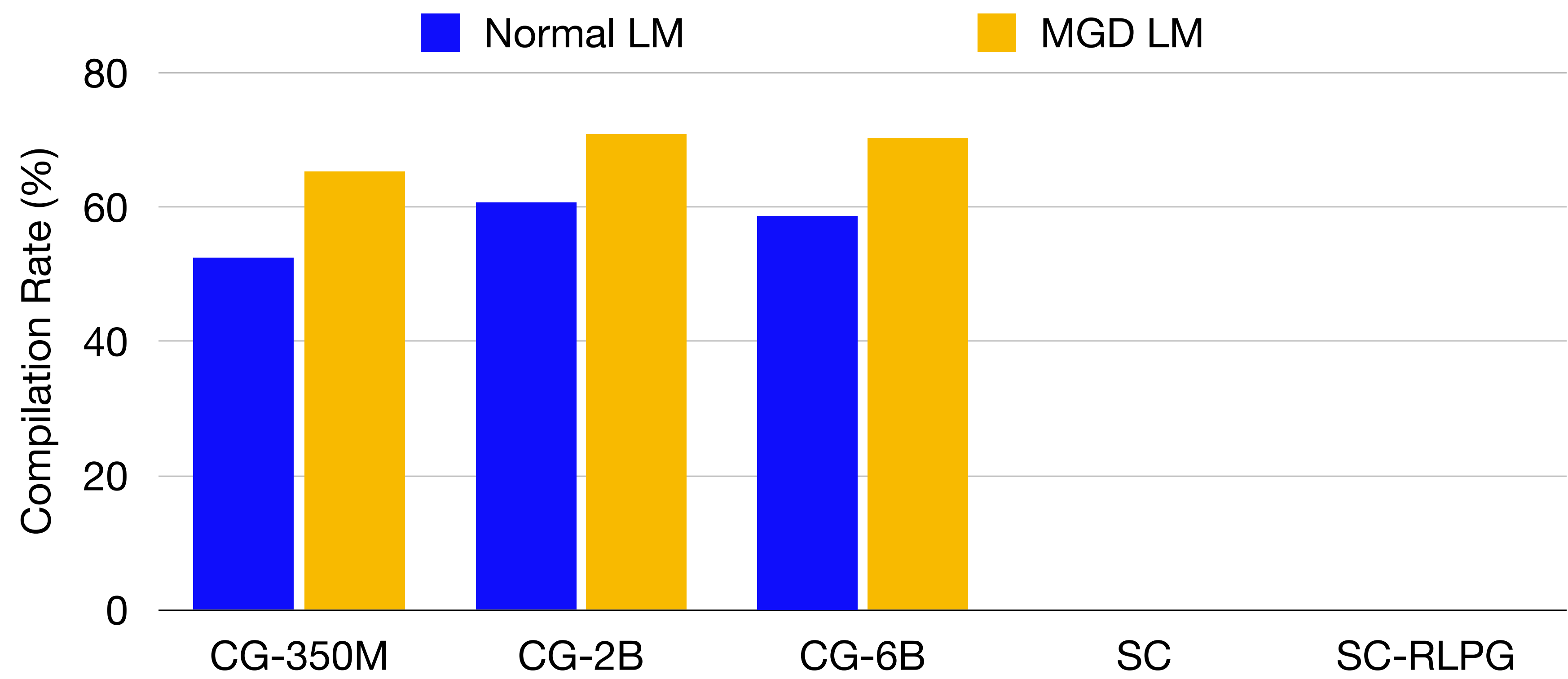
Result



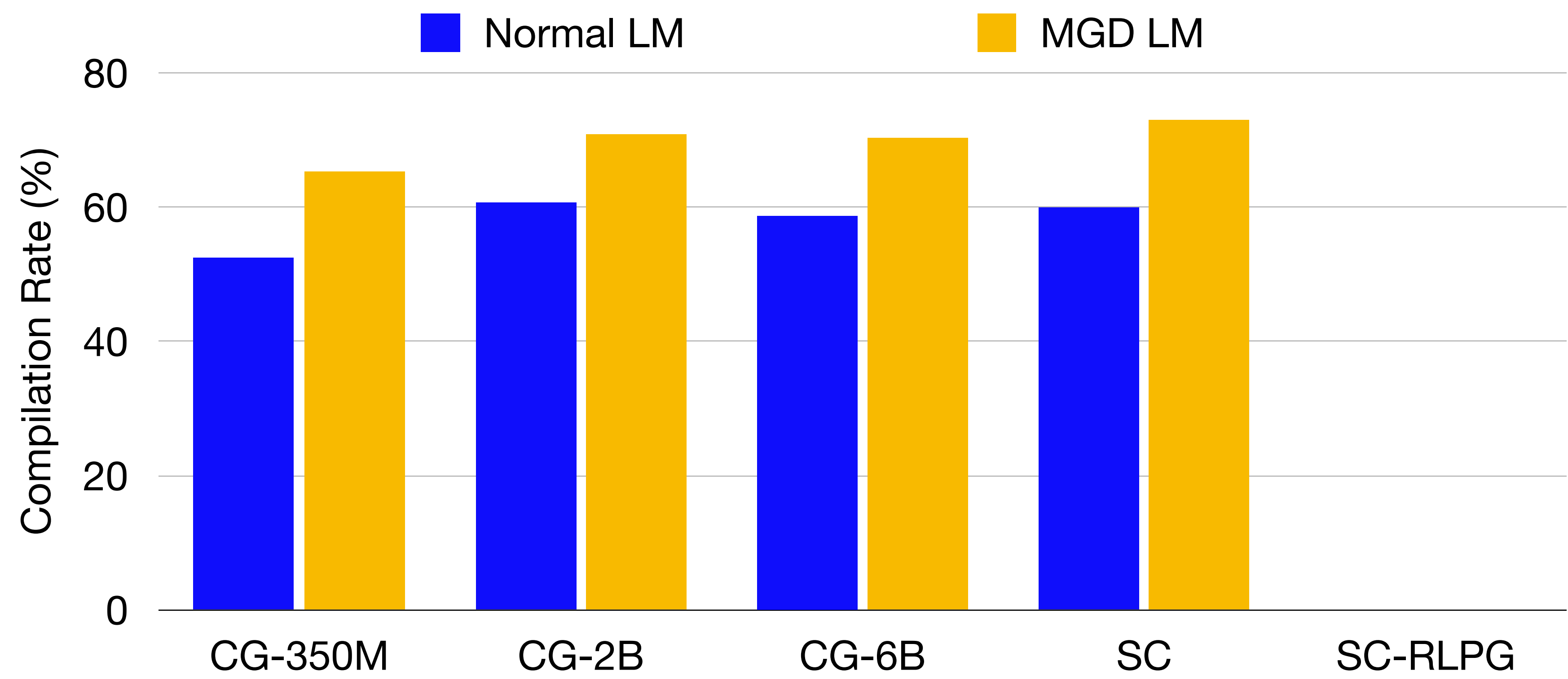
Result



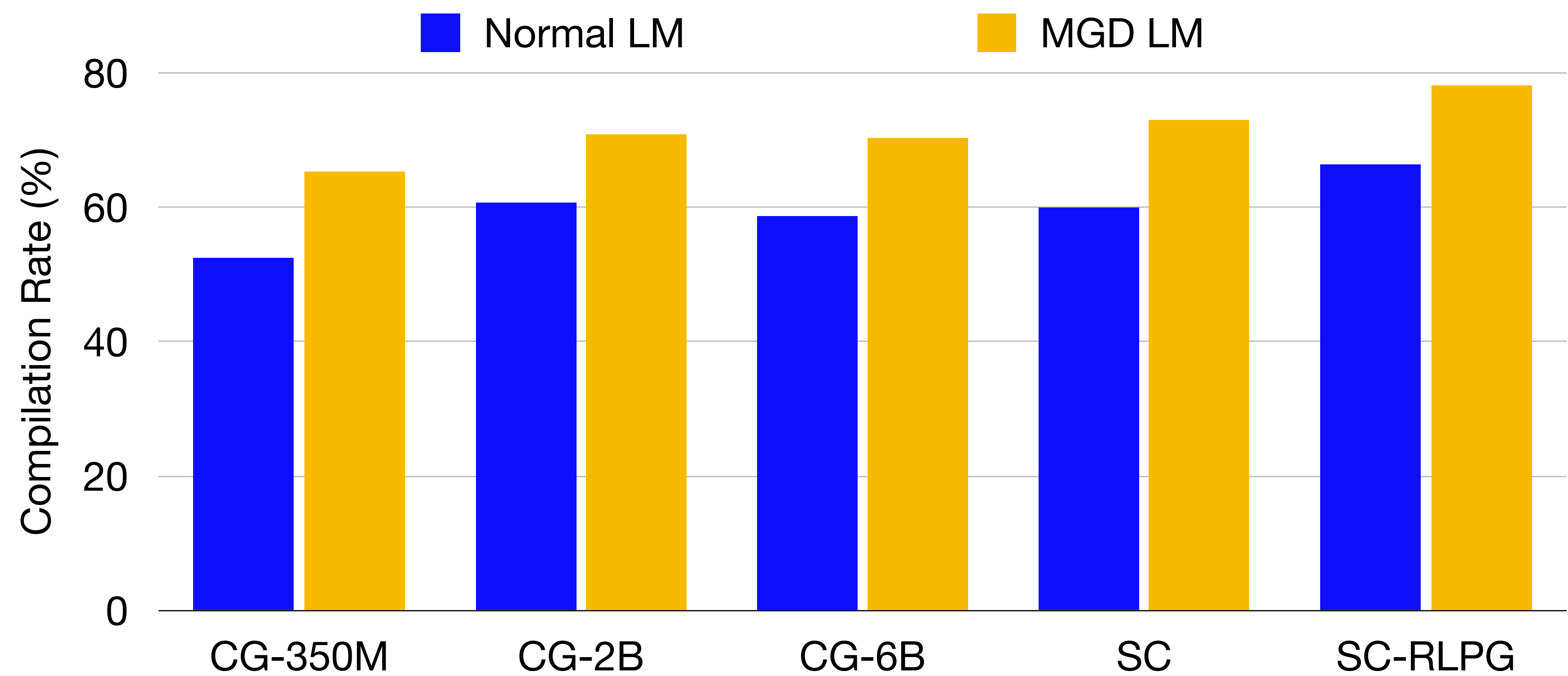
Result



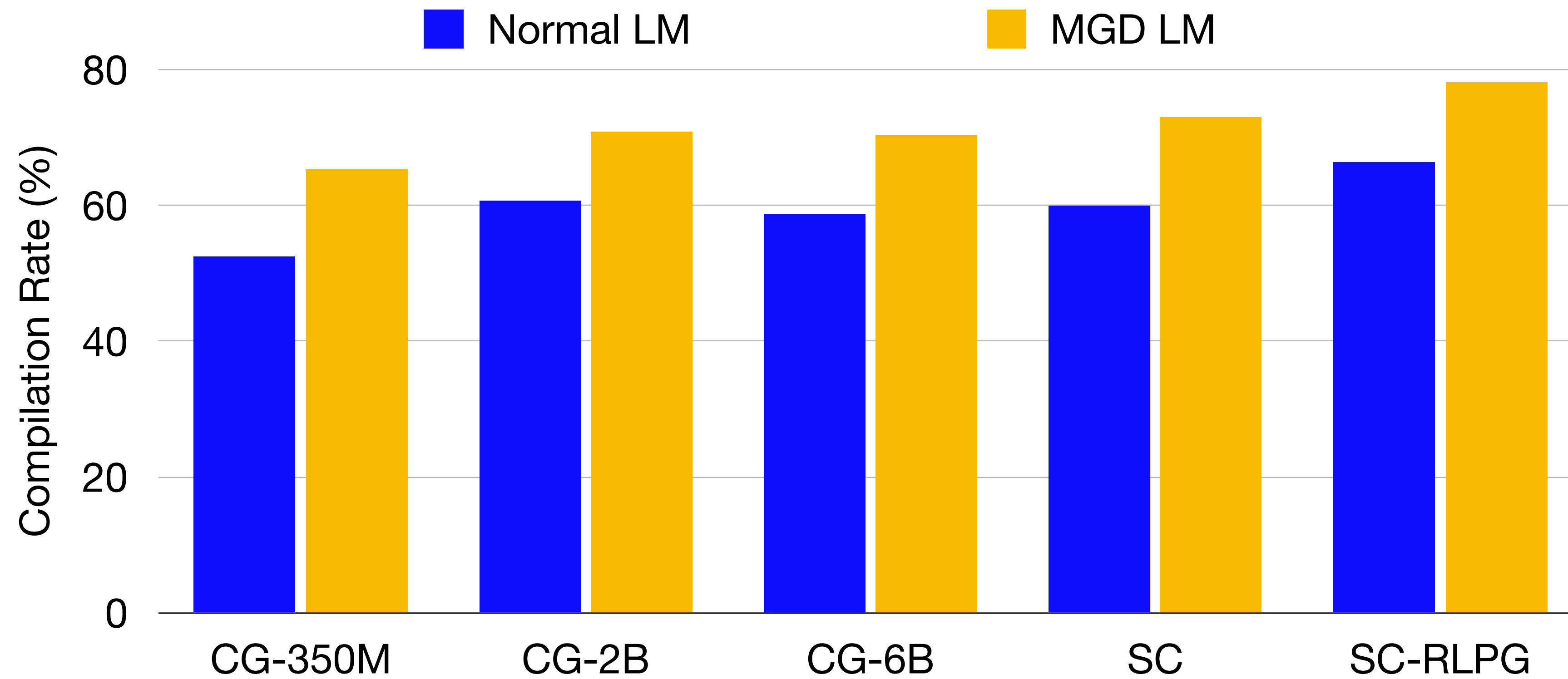
Result



Result



Result



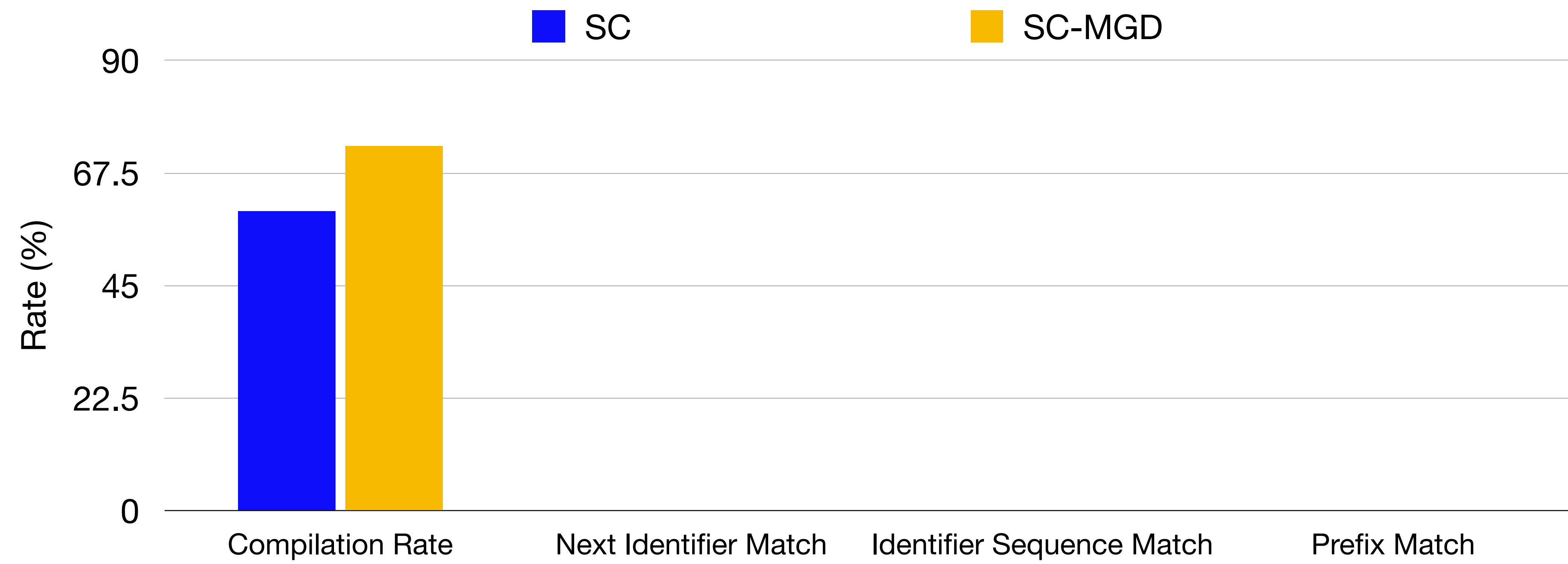
The model-free method + ensures about a 15% increased compilation rate

Result

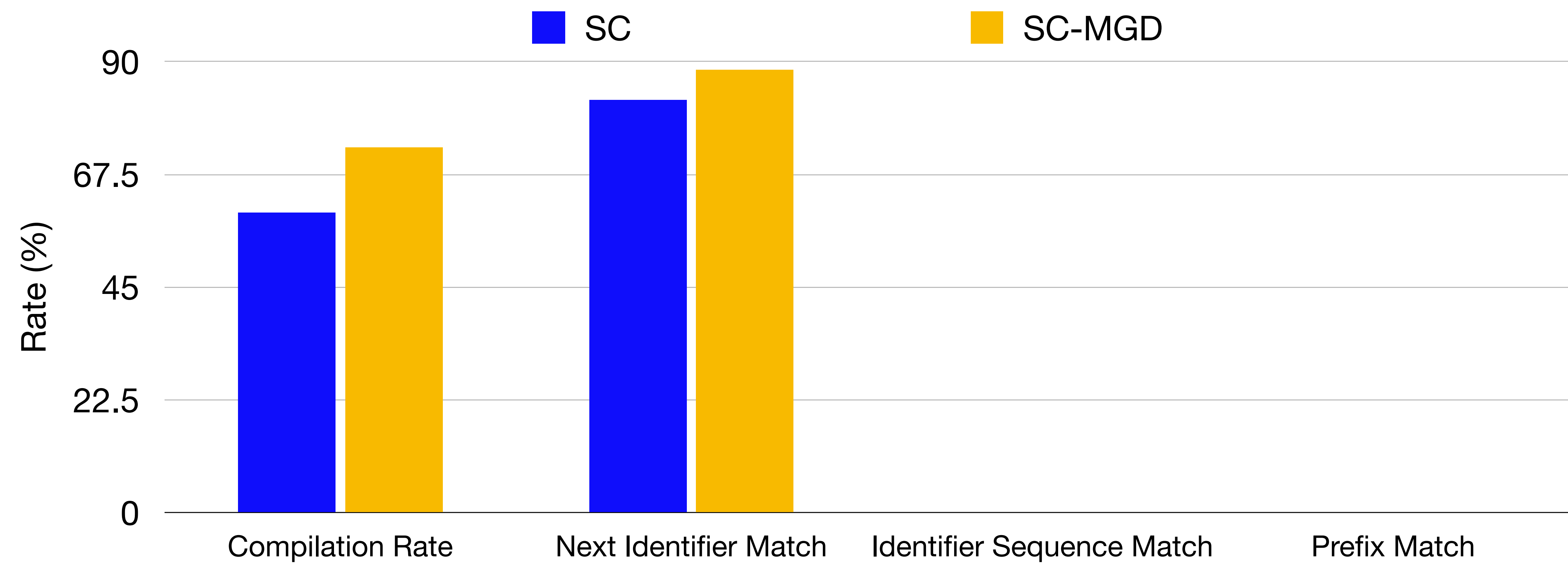
Result



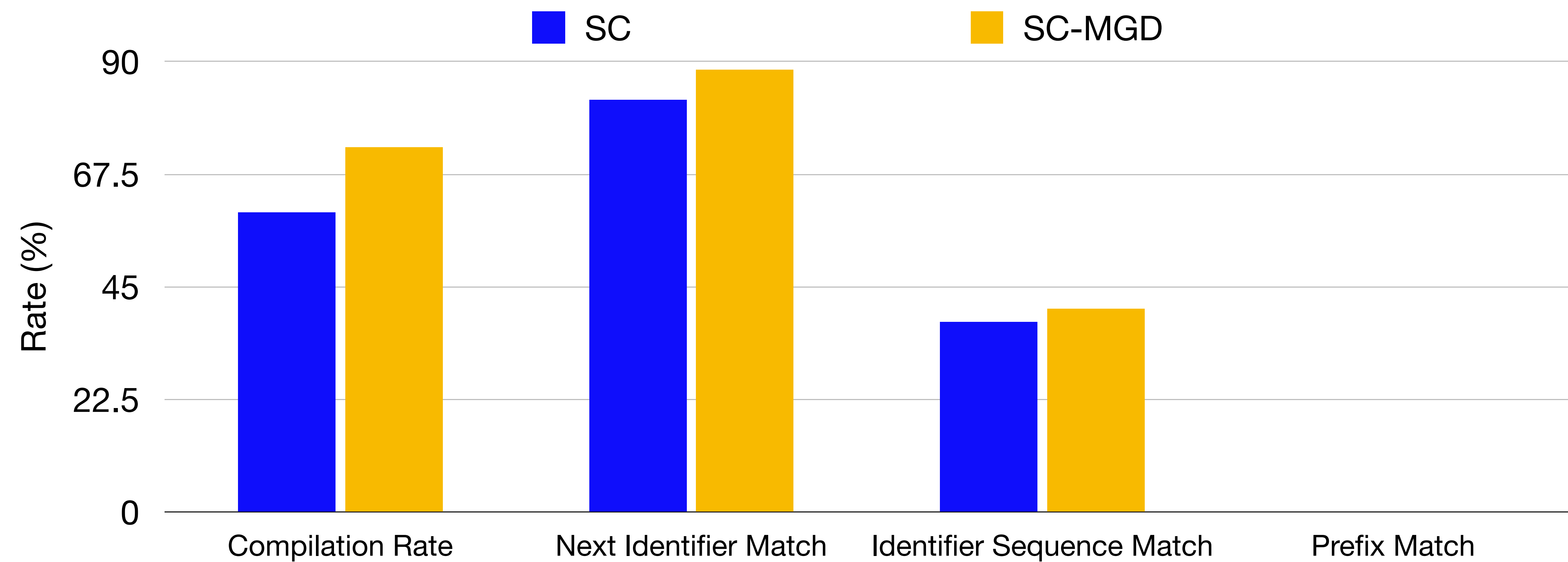
Result



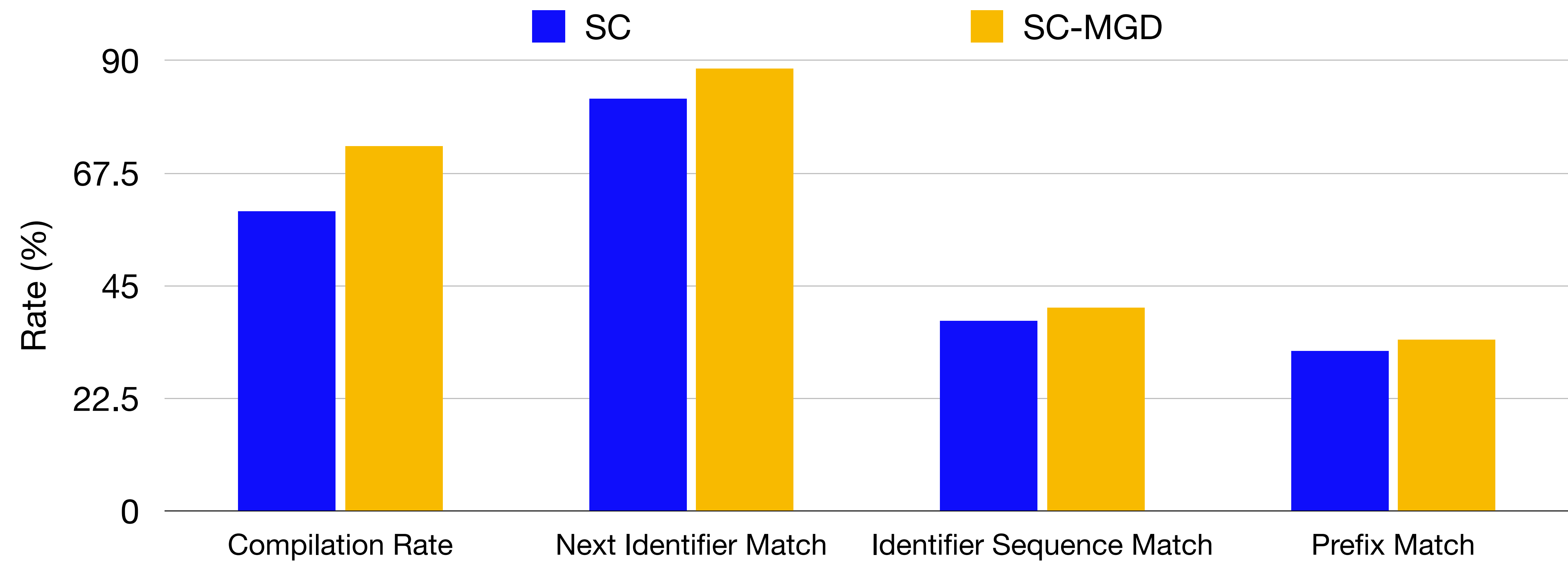
Result



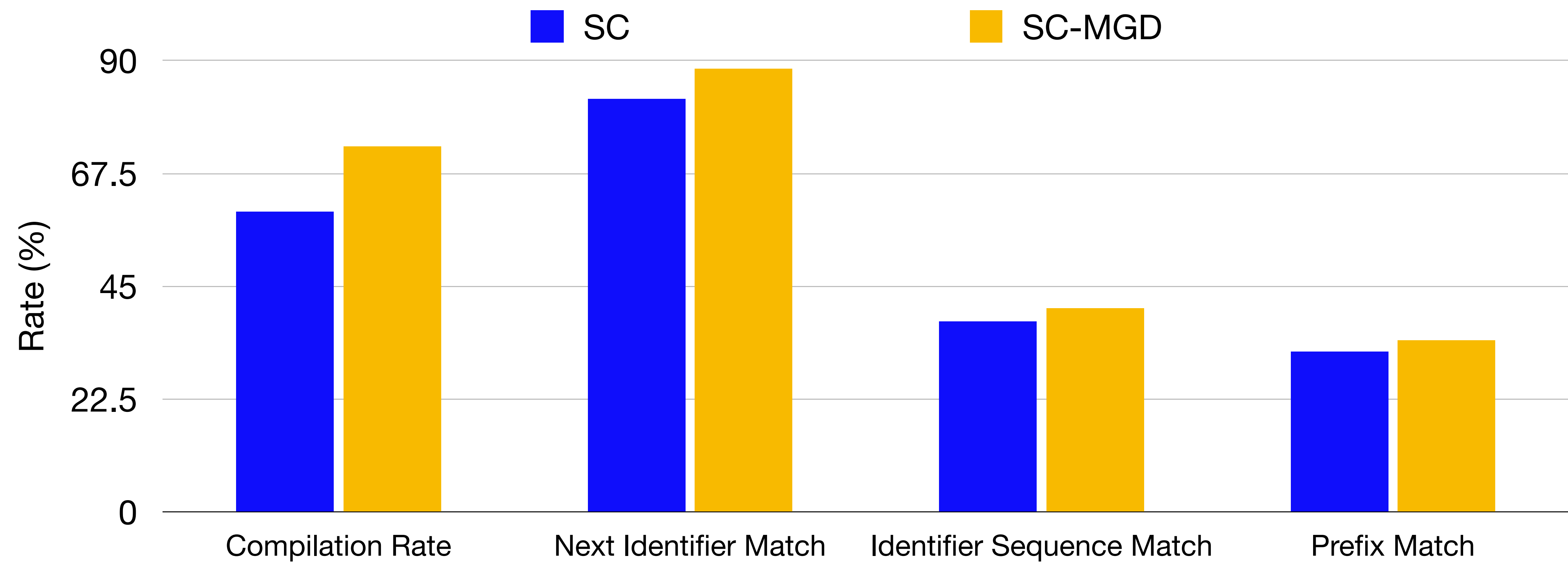
Result



Result



Result



Increased for all indicators

Technical Details

- How does a vanilla LM work?

Technical Details

- How does a vanilla LM work?

```
private SomeObj() {  
    int x = input();  
    return SomeClass()  
}
```

Partial Code

Technical Details

- How does a vanilla LM work?

```
private SomeObj() {  
    int x = input();  
    return SomeClass()  
}
```

Partial Code



LM

Technical Details

- How does a vanilla LM work?

```
private SomeObj() {  
  int x = input();  
  return SomeClass()  
}
```

Partial Code



LM



**Probability for
dictionary members**

Technical Details

- How does a vanilla LM work?

```
private SomeObj() {  
  int x = input();  
  return SomeClass()  
}
```

Partial Code



LM



**Probability for
dictionary members**

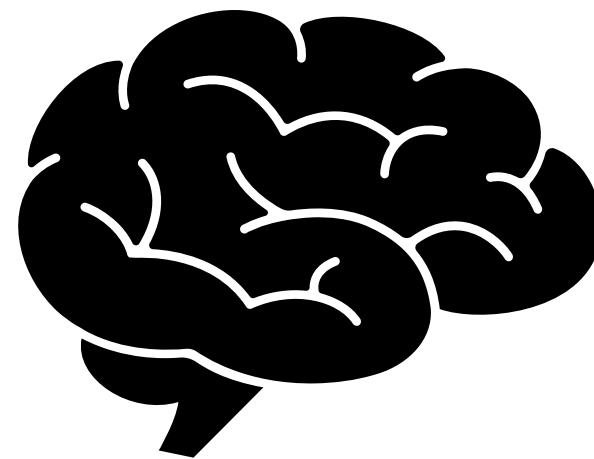
- Dictionary?

Technical Details

- How does a vanilla LM work?

```
private SomeObj() {  
    int x = input();  
    return SomeClass()  
}
```

Partial Code



LM



**Probability for
dictionary members**

- Dictionary?
 - Set of tokens LM can generate at once

Technical Details

- How does a vanilla LM work?

```
private SomeObj() {  
    int x = input();  
    return SomeClass()  
}
```

Partial Code



LM



**Probability for
dictionary members**

- Dictionary?
 - Set of tokens LM can generate at once
 - e.g. [a-z, A-Z, 0-9], def, pyt, Stri, ...

Technical Details

- How does a vanilla LM work?

```
private SomeObj() {  
    int x = input();  
    return SomeClass()  
}
```

Partial Code



LM



Probability for
dictionary members

- Dictionary?
 - Set of tokens LM can generate at once
 - e.g. [a-z, A-Z, 0-9], def, pyt, Stri, ...
 - LM can generate **only members of the dictionary**

Technical Details

Technical Details

- Monitor-Guided Decoding (MGD)

Technical Details

- Monitor-Guided Decoding (MGD)
 - Masking infeasible tokens in the dictionary

Technical Details

- Monitor-Guided Decoding (MGD)
 - Masking infeasible tokens in the dictionary
 - Monitoring target: Type-Consistency (especially dereference expression)

Technical Details

- Monitor-Guided Decoding (MGD)
 - Masking infeasible tokens in the dictionary
 - Monitoring target: Type-Consistency (especially dereference expression)

SomeObj

Technical Details

- Monitor-Guided Decoding (MGD)
 - Masking infeasible tokens in the dictionary
 - Monitoring target: Type-Consistency (especially dereference expression)

SomeObj \longrightarrow SomeObj.

Technical Details

- Monitor-Guided Decoding (MGD)
 - Masking infeasible tokens in the dictionary
 - Monitoring target: Type-Consistency (especially dereference expression)

SomeObj **Monitoring Trigger!**
→ SomeObj.

Technical Details

- Monitor-Guided Decoding (MGD)
 - Masking infeasible tokens in the dictionary
 - Monitoring target: Type-Consistency (especially dereference expression)

Monitoring Trigger!

SomeObj \longrightarrow SomeObj_ \longrightarrow Member of "SomeObj"

- setA
- getA
- findA
- ...

Technical Details

- Monitor-Guided Decoding (MGD)
 - Masking infeasible tokens in the dictionary
 - Monitoring target: Type-Consistency (especially dereference expression)

Monitoring Trigger!

SomeObj \longrightarrow SomeObj. \longrightarrow Member of "SomeObj"

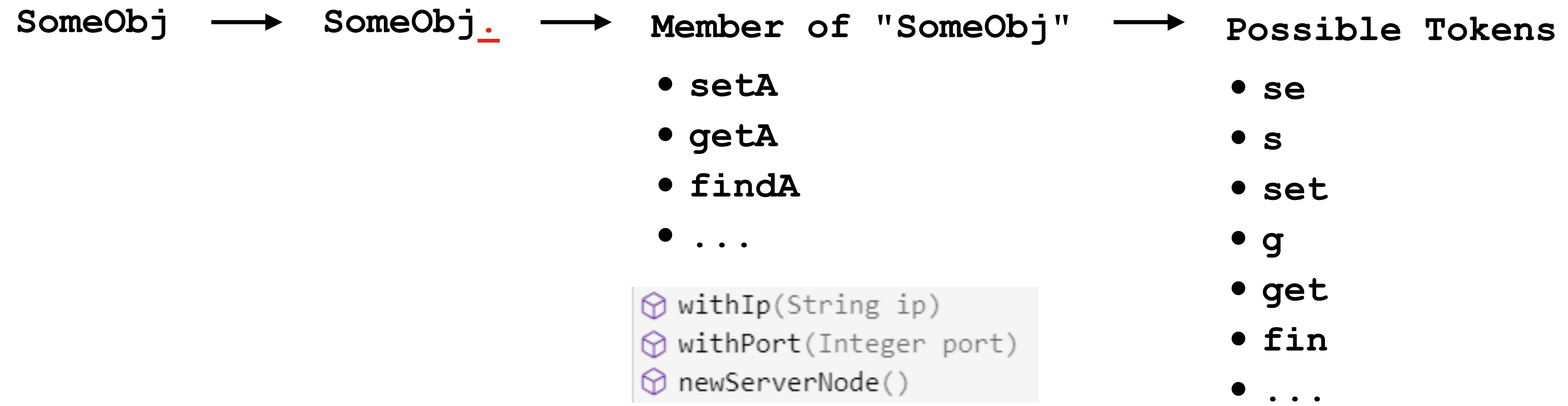
- setA
- getA
- findA
- ...

```
withIp(String ip)  
withPort(Integer port)  
newServerNode()
```

Technical Details

- Monitor-Guided Decoding (MGD)
 - Masking infeasible tokens in the dictionary
 - Monitoring target: Type-Consistency (especially dereference expression)

Monitoring Trigger!



Technical Details

Technical Details

- How to combine?

Technical Details

- How to combine?

SomeObj

Technical Details

- How to combine?

SomeObj \longrightarrow SomeObj.

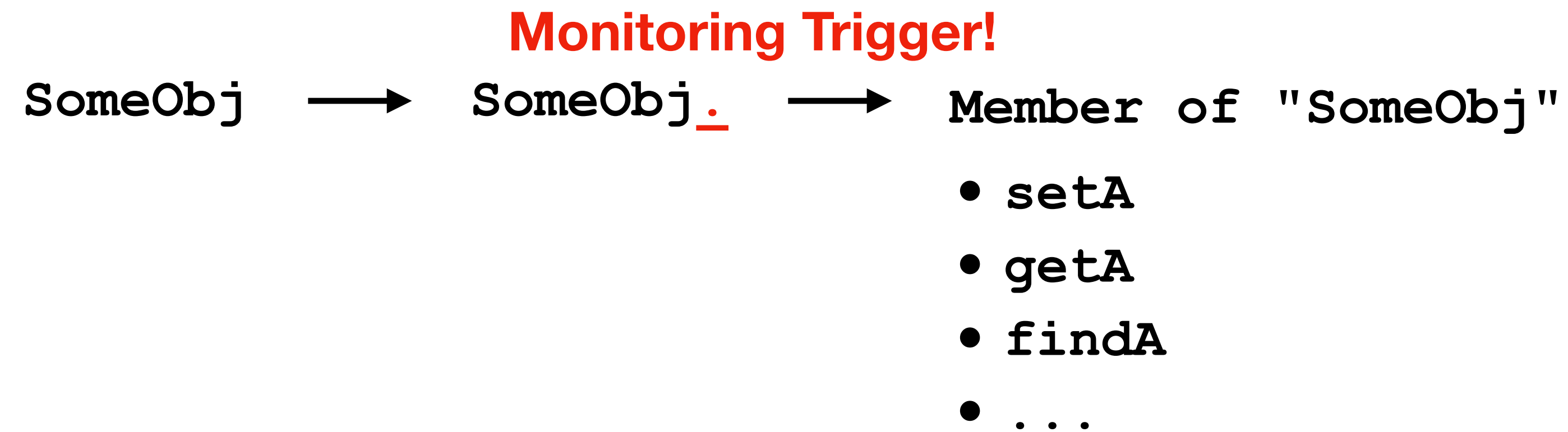
Technical Details

- How to combine?

SomeObj \longrightarrow **Monitoring Trigger!**
SomeObj.

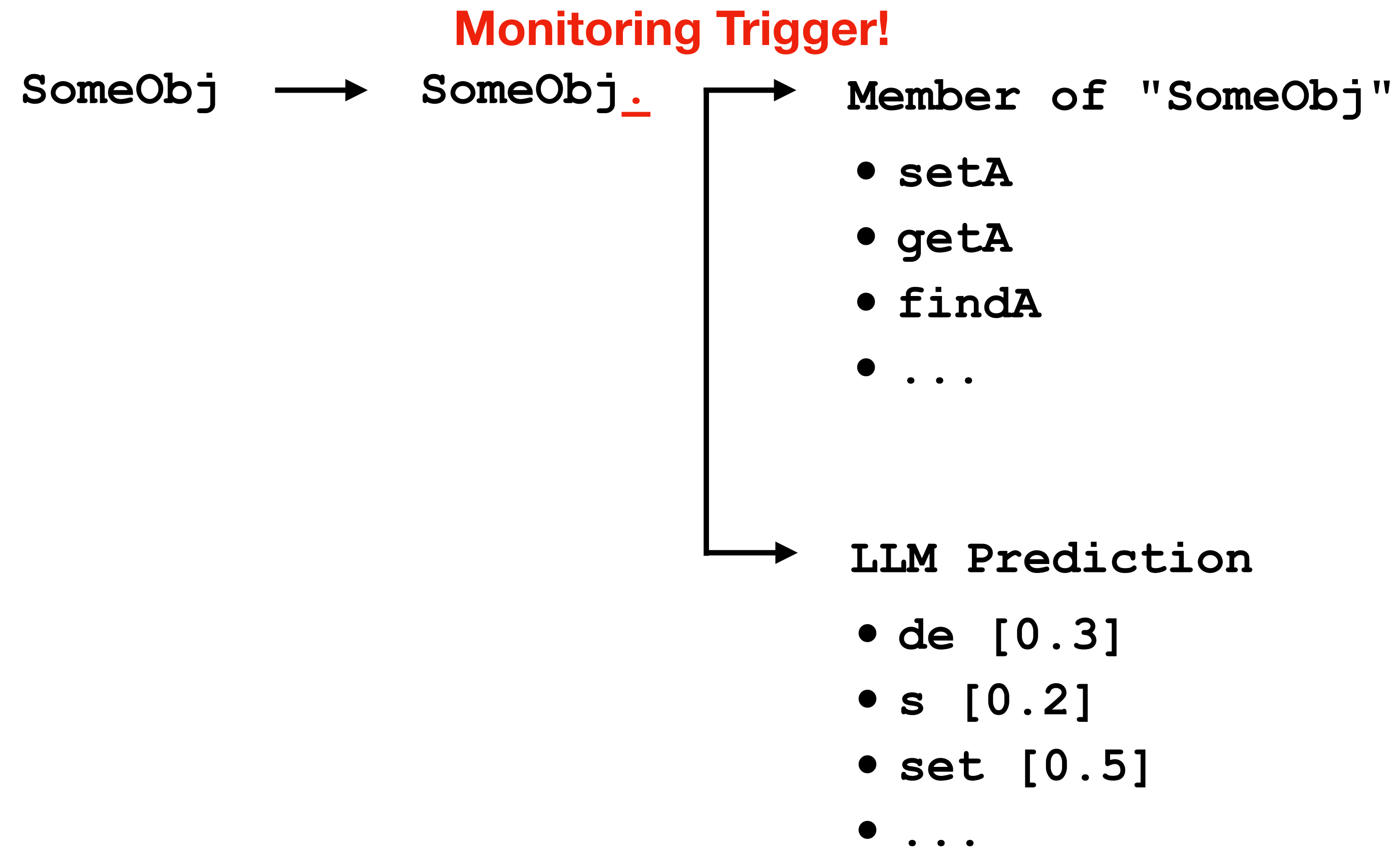
Technical Details

- How to combine?



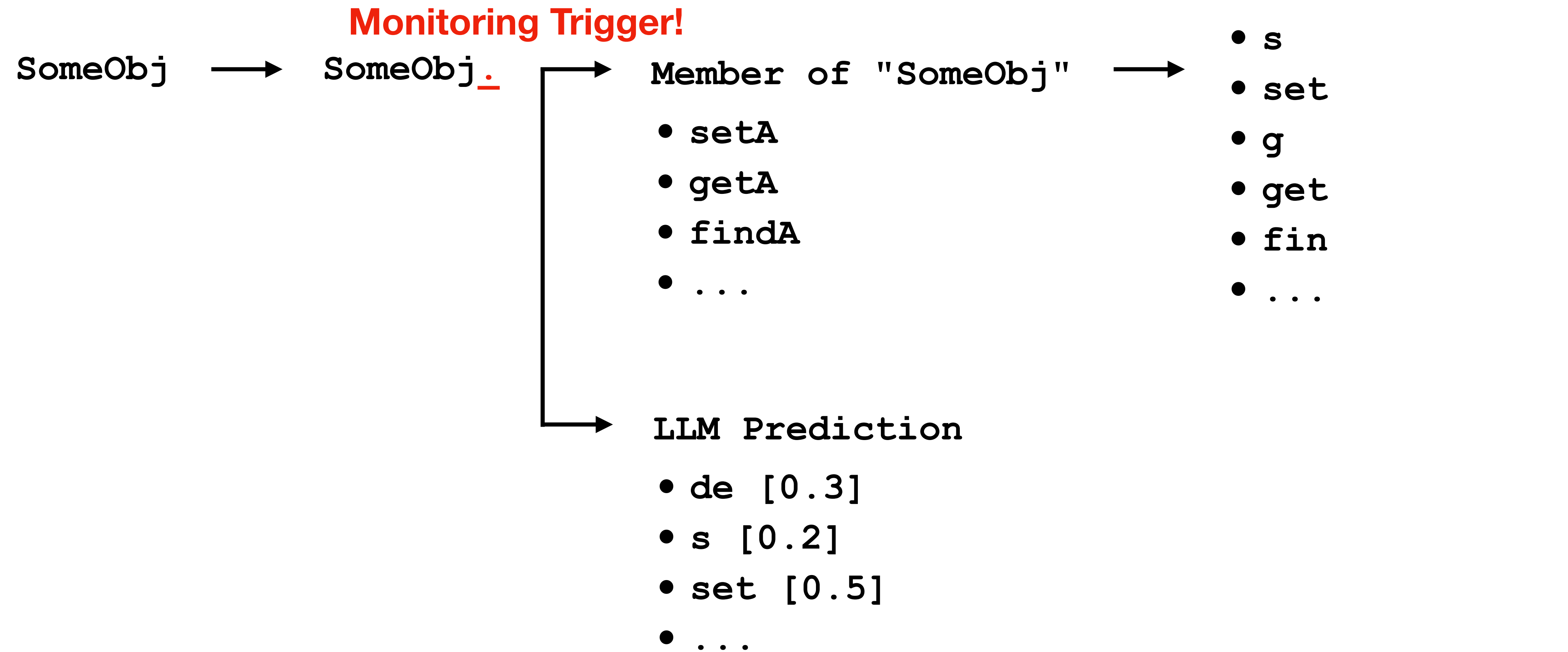
Technical Details

- How to combine?



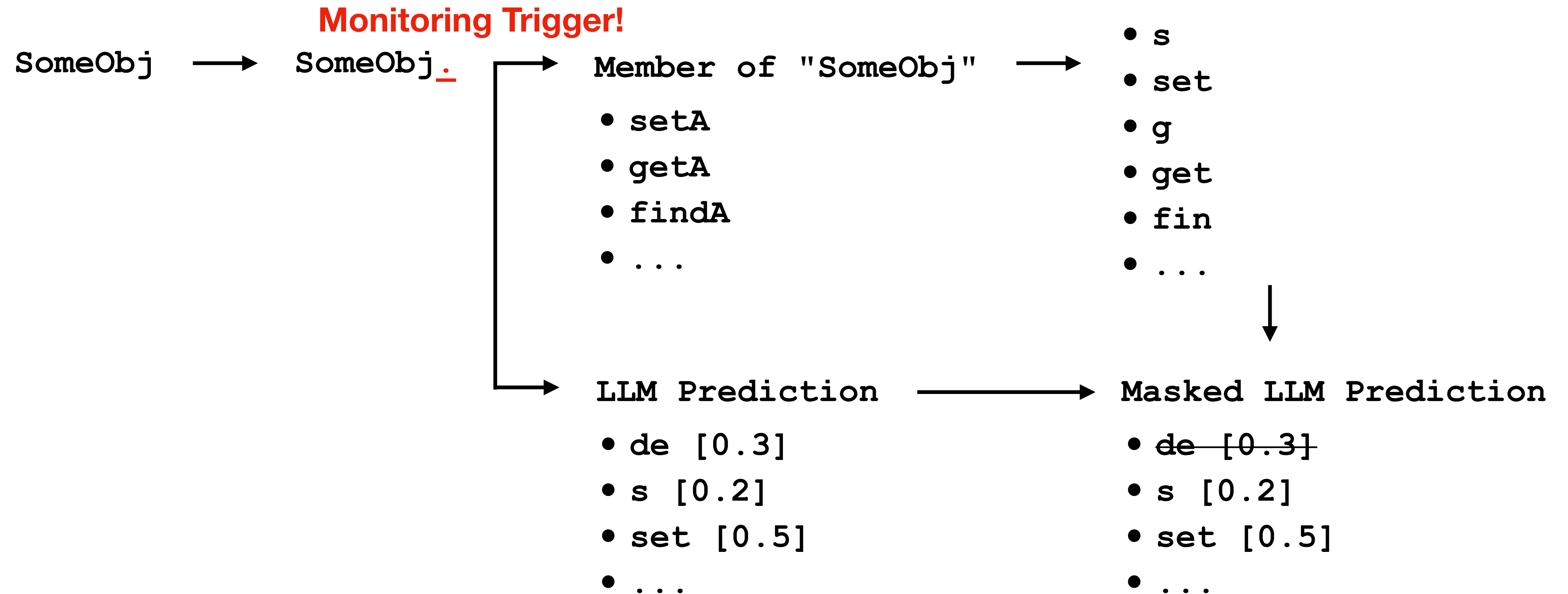
Technical Details

- How to combine?



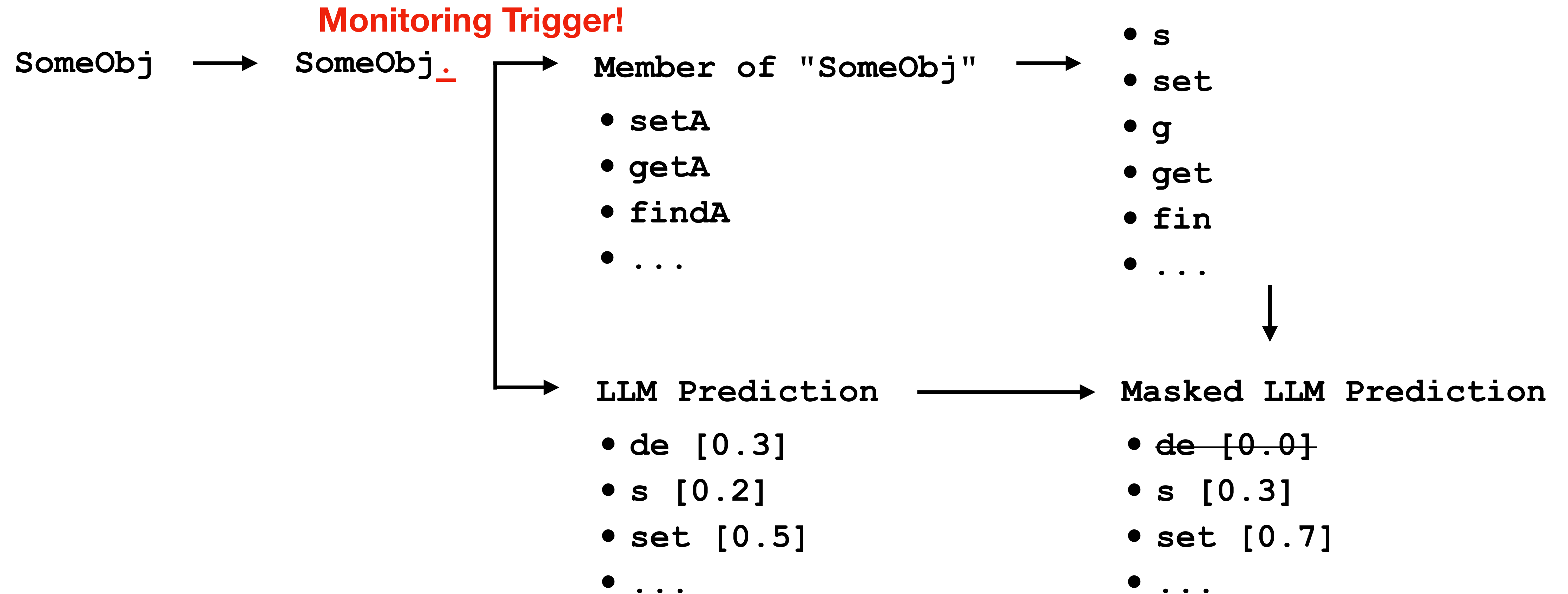
Technical Details

- How to combine?



Technical Details

- How to combine?

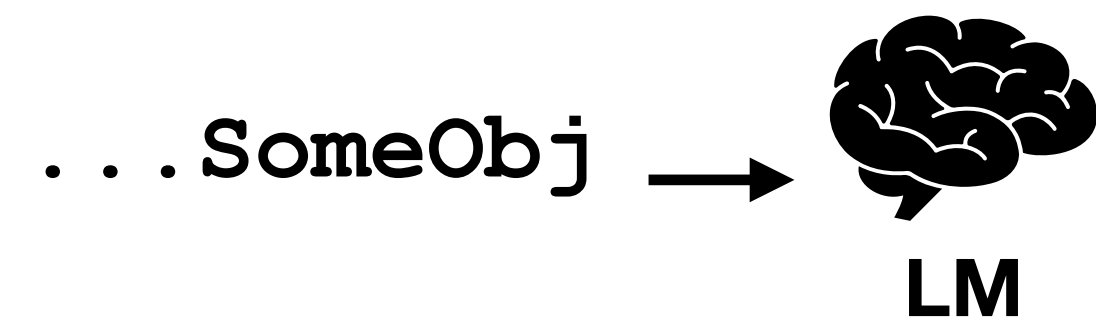


Whole Process

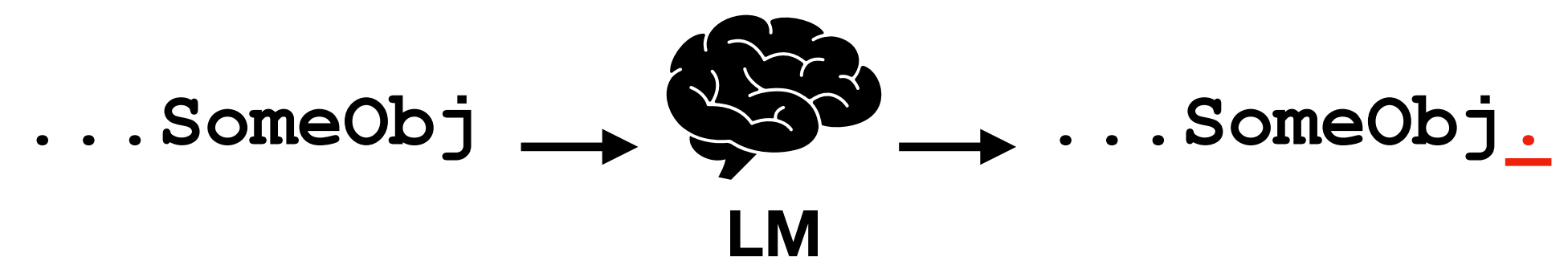
Whole Process

...SomeObj

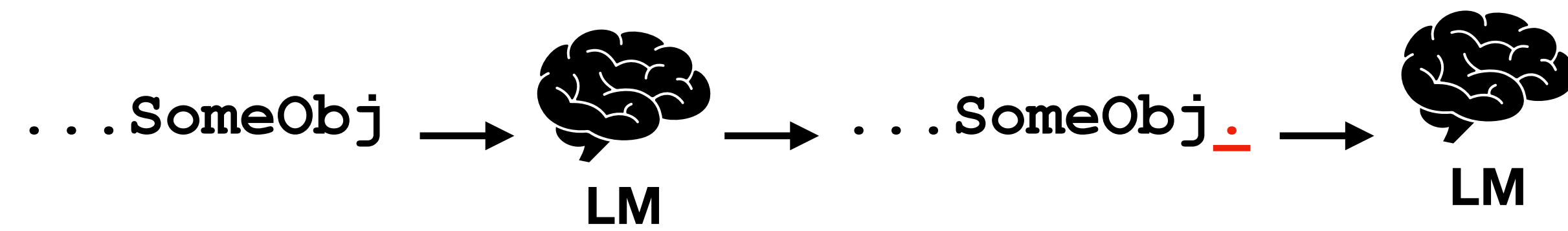
Whole Process



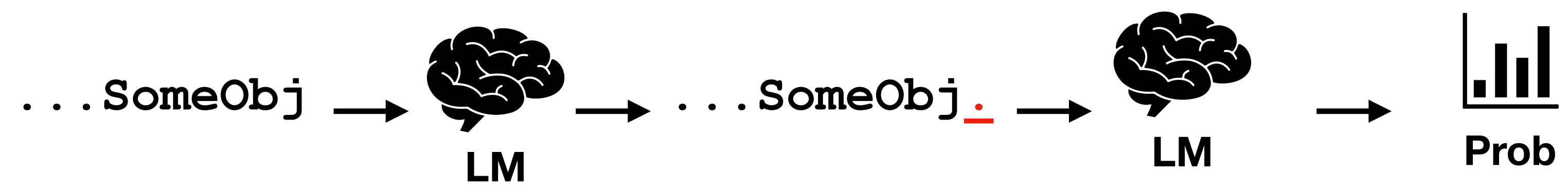
Whole Process



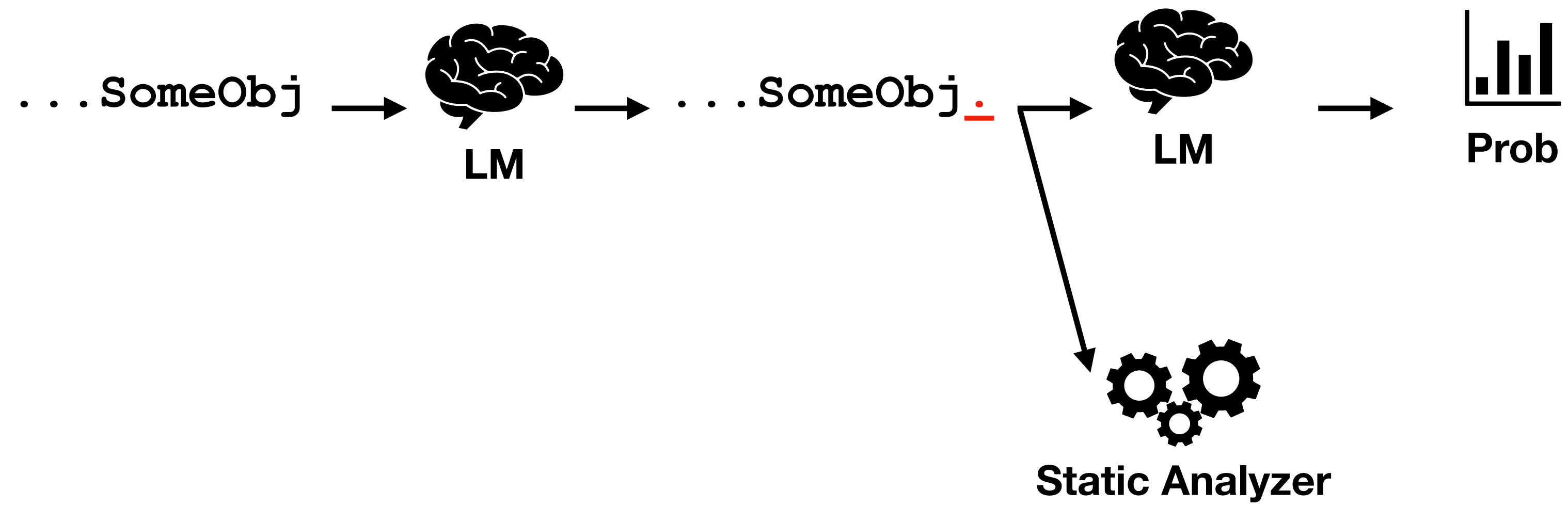
Whole Process



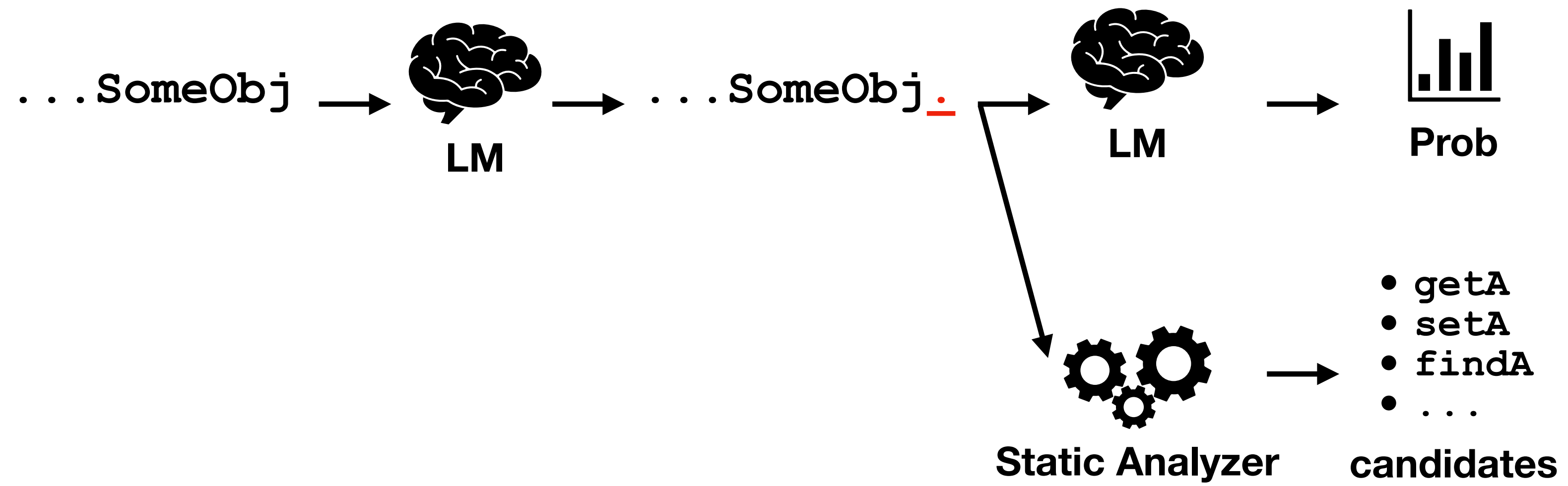
Whole Process



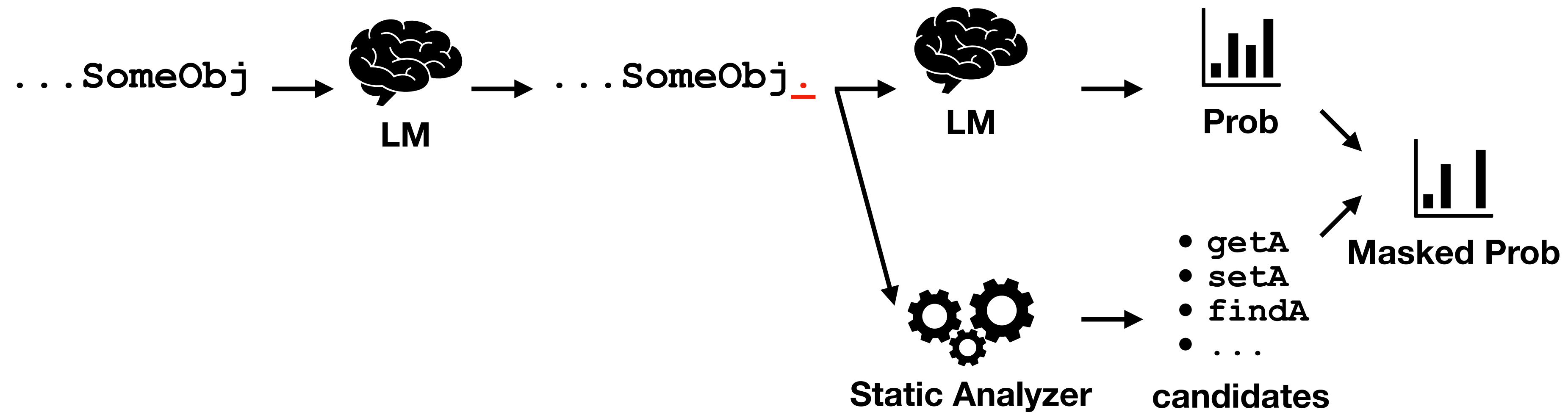
Whole Process



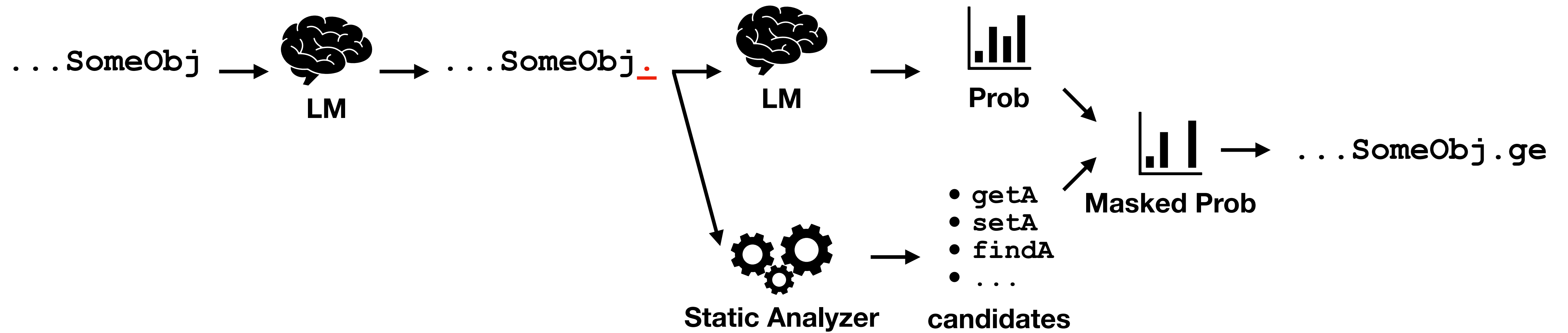
Whole Process



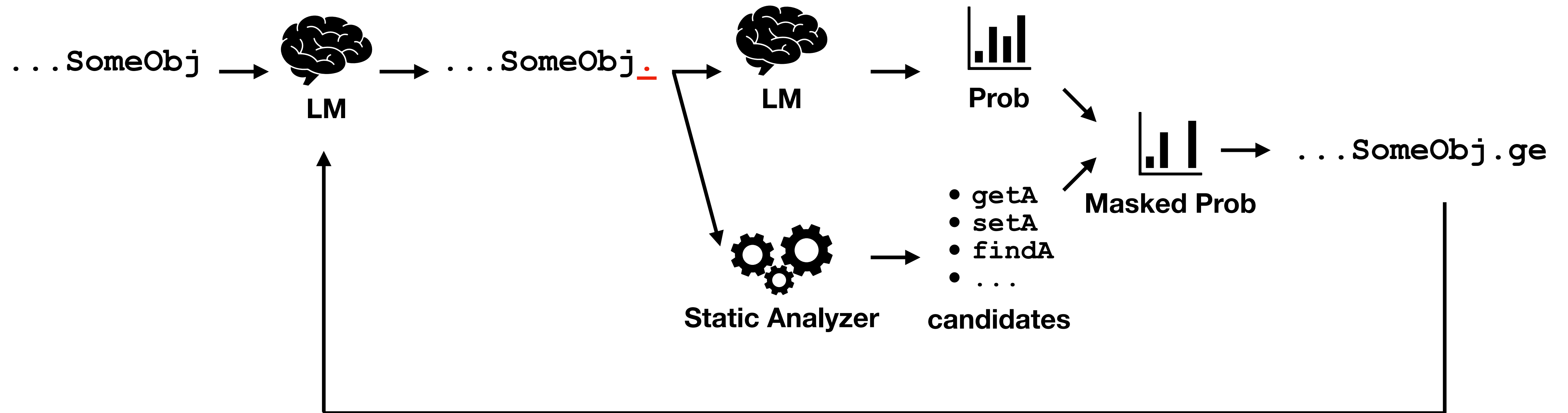
Whole Process



Whole Process



Whole Process



Evaluation

Evaluation

- Benchmark

Evaluation

- Benchmark
 - PragmaticCode (Open-source Java projects)

Evaluation

- Benchmark
 - PragmaticCode (Open-source Java projects)
 - **100** Repositories, **1420** Methods, and **10538** Dereference prompts

Evaluation

- Benchmark
 - PragmaticCode (Open-source Java projects)
 - **100** Repositories, **1420** Methods, and **10538** Dereference prompts
- Metrics

Evaluation

- Benchmark
 - PragmaticCode (Open-source Java projects)
 - **100** Repositories, **1420** Methods, and **10538** Dereference prompts
- Metrics
 - Compilation Rate (CR): Percentage of **successful compilation**

Evaluation

- Benchmark
 - PragmaticCode (Open-source Java projects)
 - **100** Repositories, **1420** Methods, and **10538** Dereference prompts
- Metrics
 - Compilation Rate (CR): Percentage of **successful compilation**
 - Next Identifier Match (NIM): Check that only **the first generated ID** is correct

Evaluation

- Benchmark
 - PragmaticCode (Open-source Java projects)
 - **100** Repositories, **1420** Methods, and **10538** Dereference prompts
- Metrics
 - Compilation Rate (CR): Percentage of **successful compilation**
 - Next Identifier Match (NIM): Check that only **the first generated ID** is correct
 - Identifier Sequence Match (ISM): Check how much **ID sequence** is correct

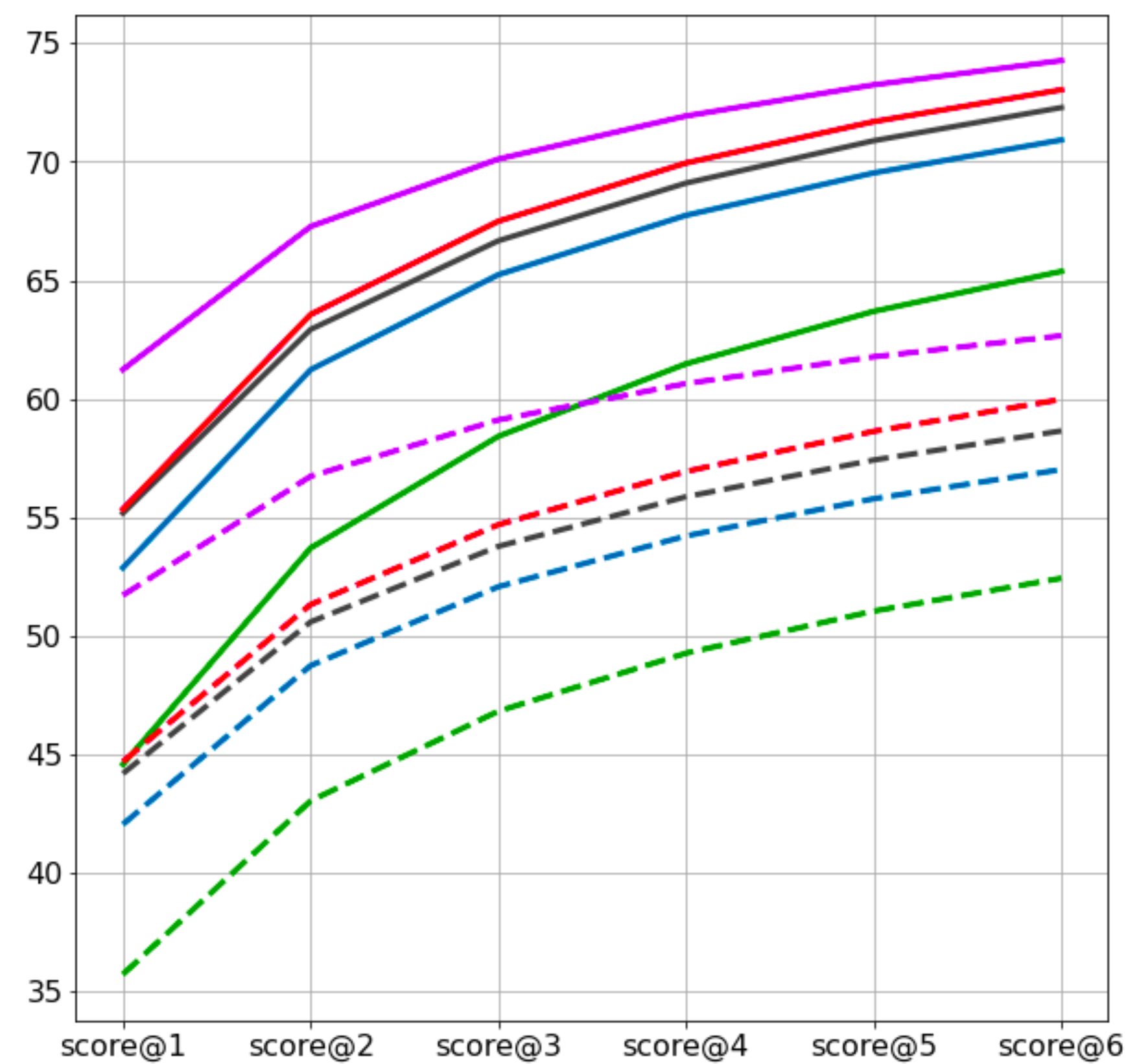
Evaluation

- Benchmark
 - PragmaticCode (Open-source Java projects)
 - **100** Repositories, **1420** Methods, and **10538** Dereference prompts
- Metrics
 - Compilation Rate (CR): Percentage of **successful compilation**
 - Next Identifier Match (NIM): Check that only **the first generated ID** is correct
 - Identifier Sequence Match (ISM): Check how much **ID sequence** is correct
 - Prefix Match (PM): Check how much **token sequence** is correct

Evaluation - Performance

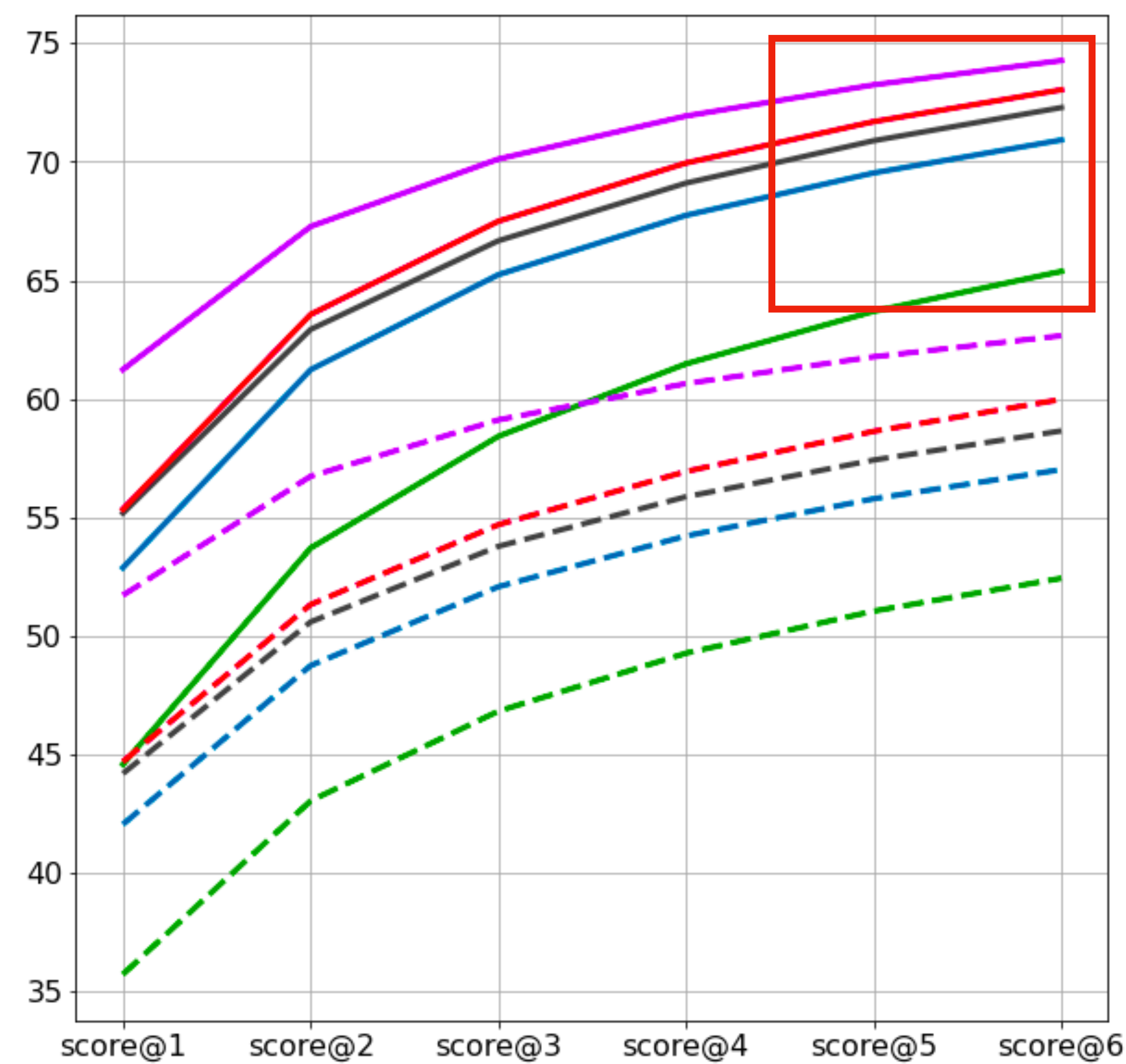
Evaluation - Performance

Compilation Rate



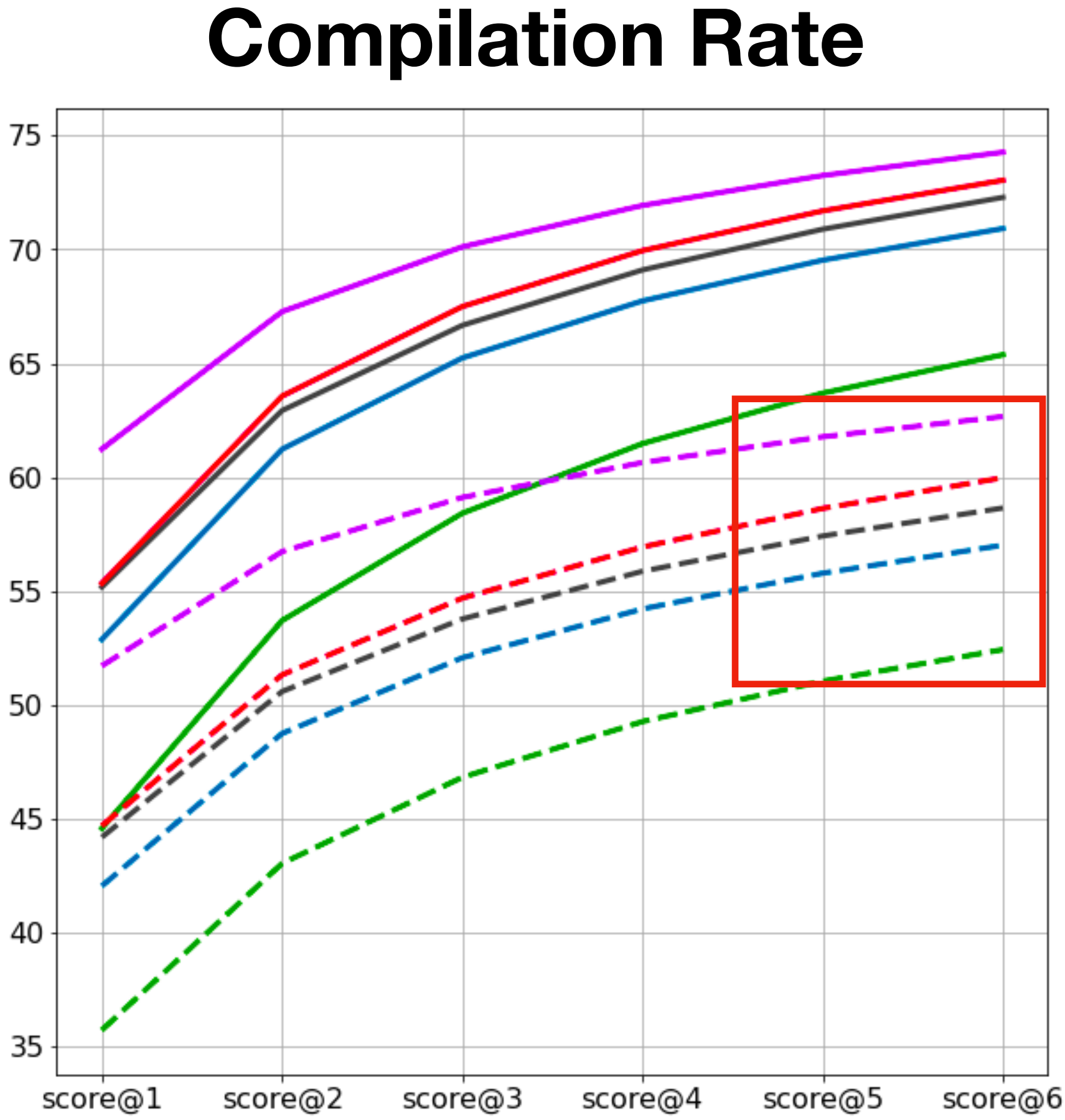
Evaluation - Performance

Compilation Rate



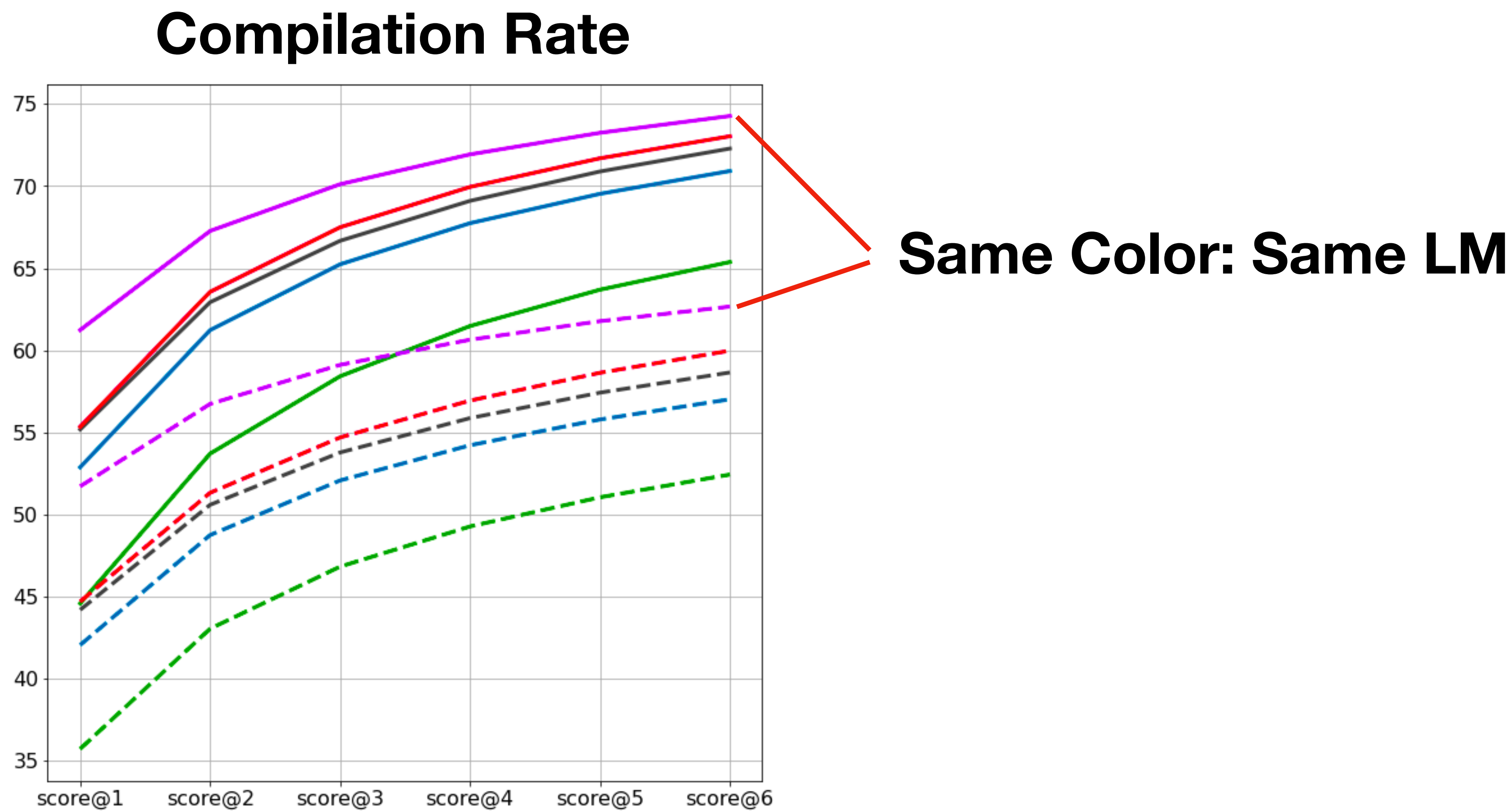
Solid lines: MGD was applied

Evaluation - Performance



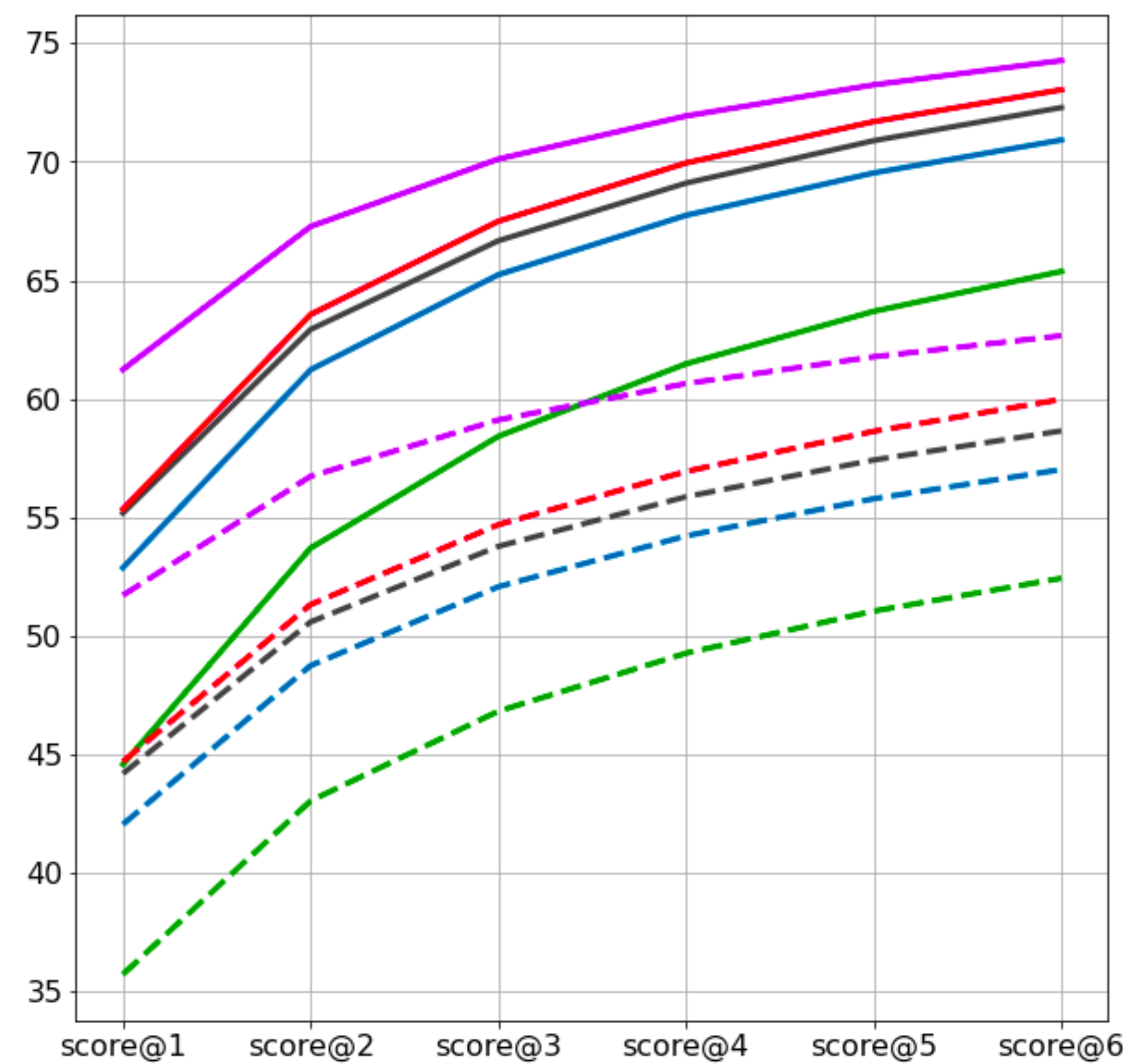
Dashed lines: Vanilla LMs

Evaluation - Performance



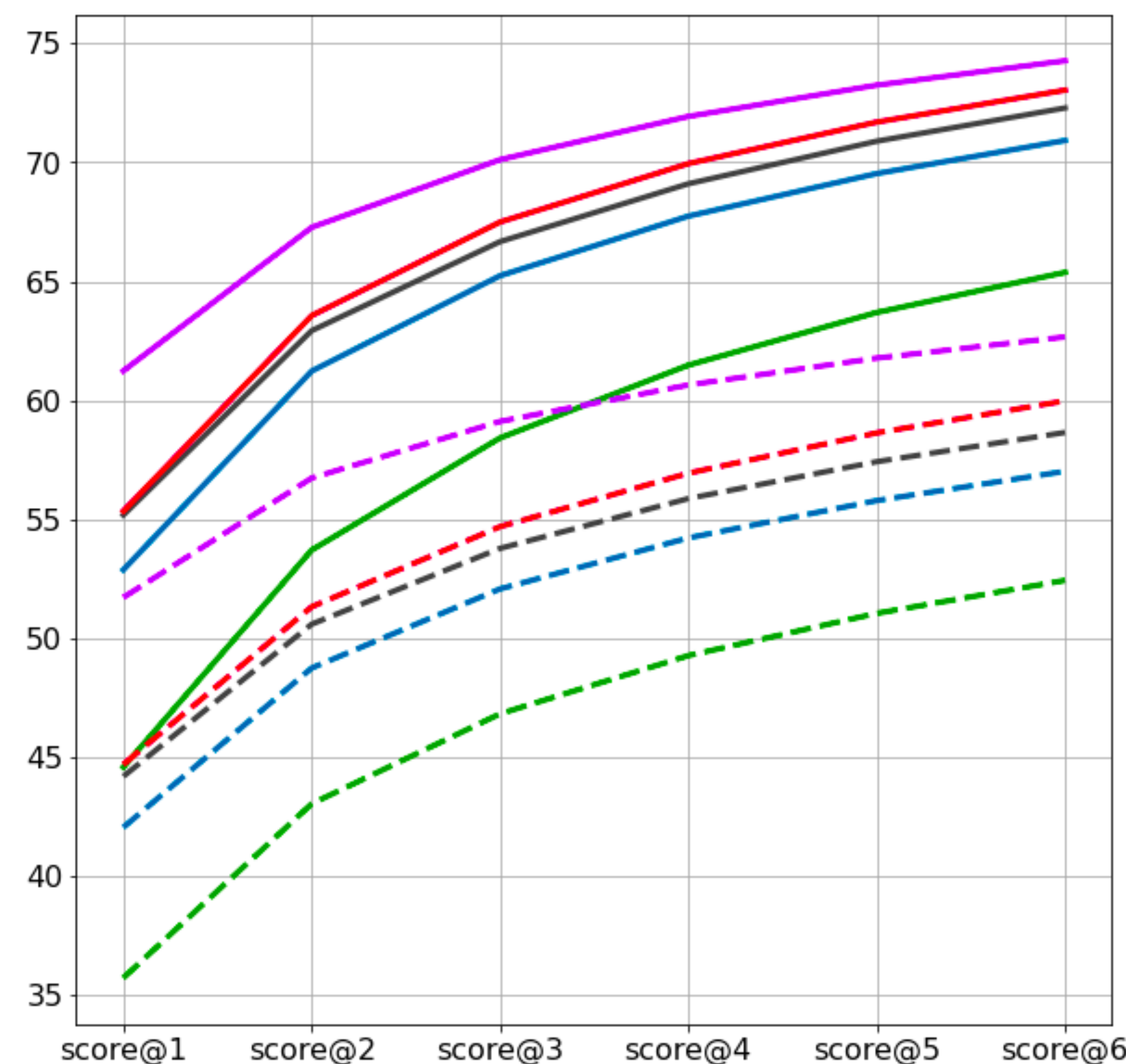
Evaluation - Performance

Compilation Rate

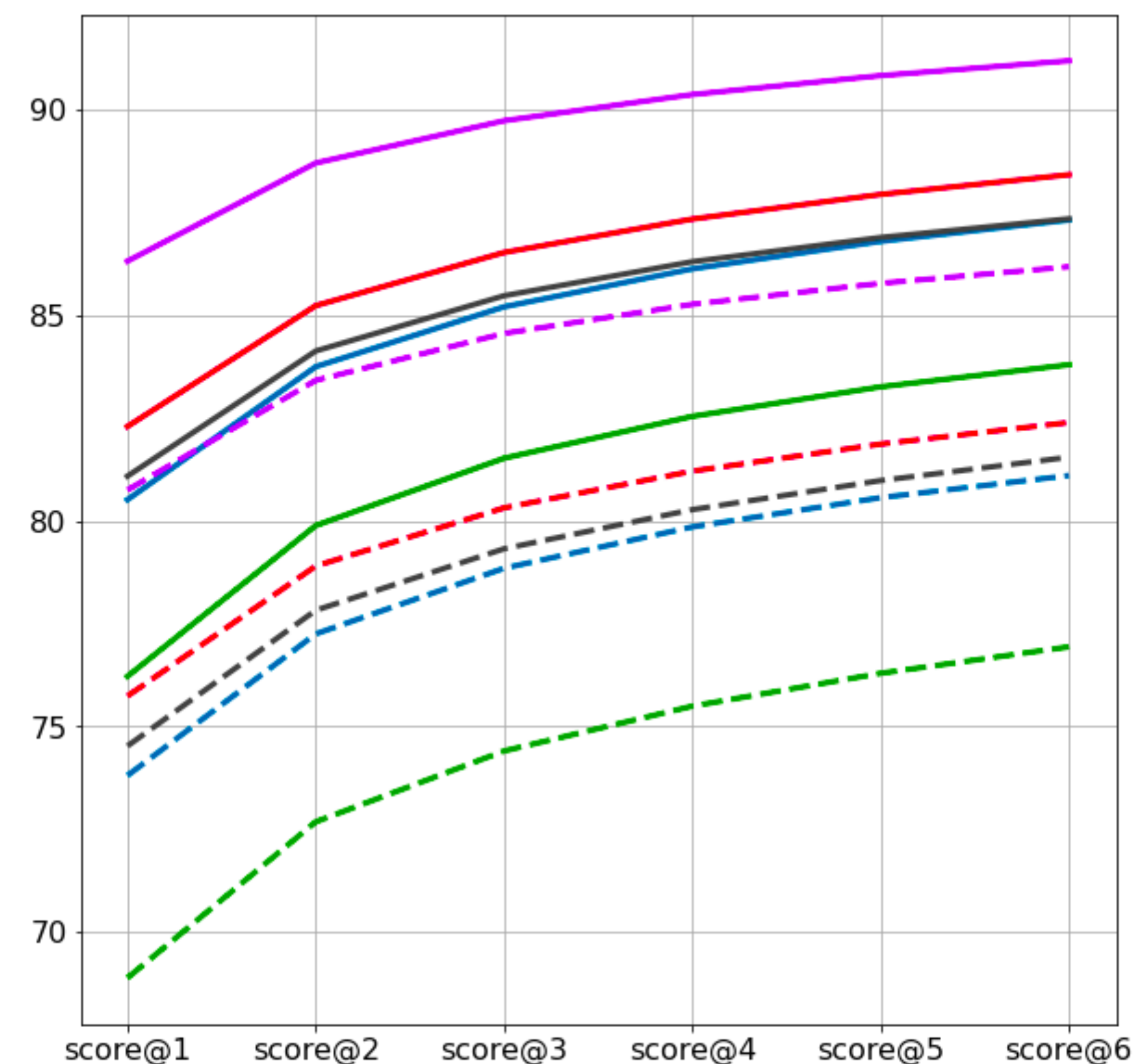


Evaluation - Performance

Compilation Rate

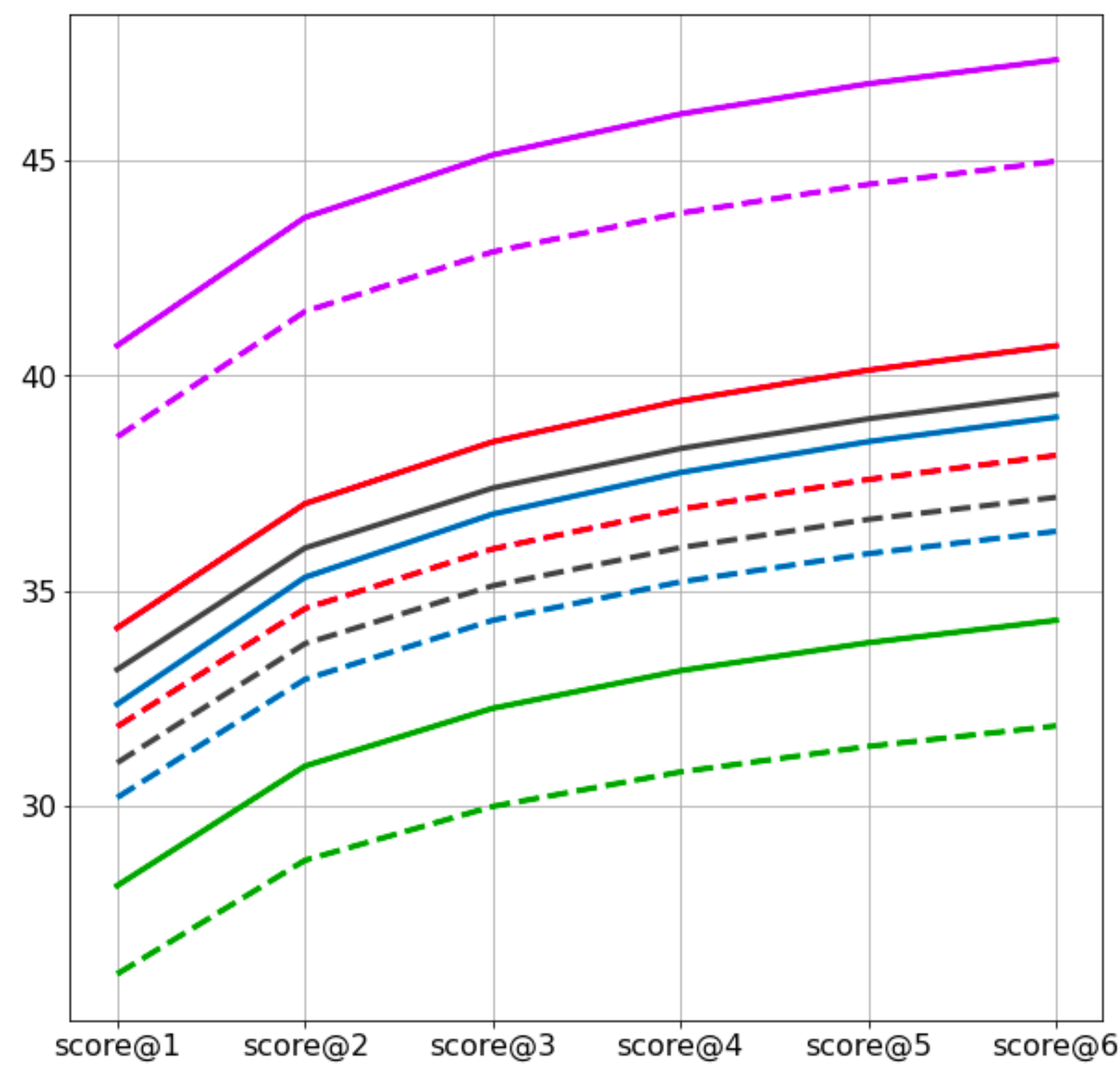


Next Identifier Match



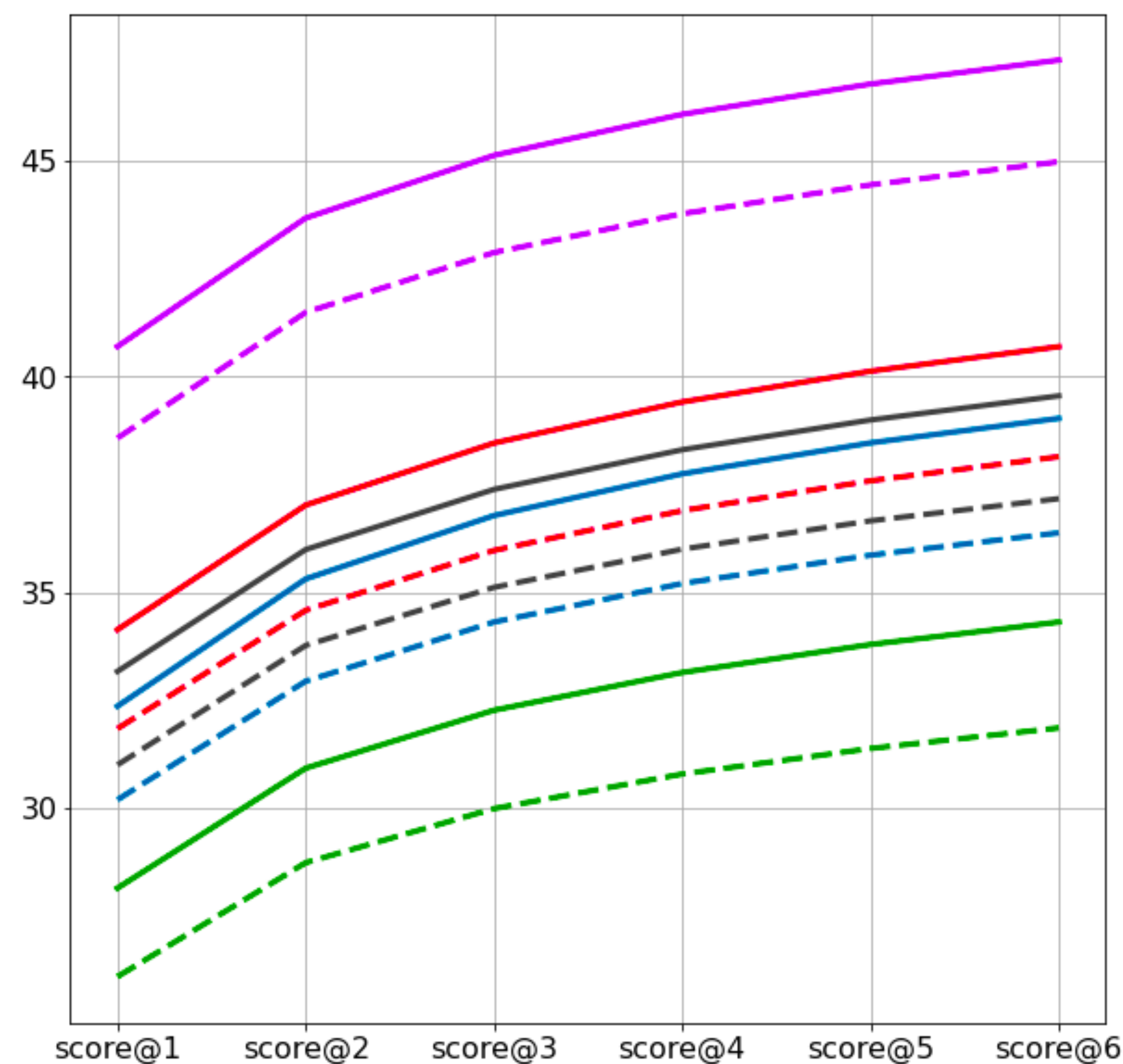
Evaluation - Performance

Next Sequence Match

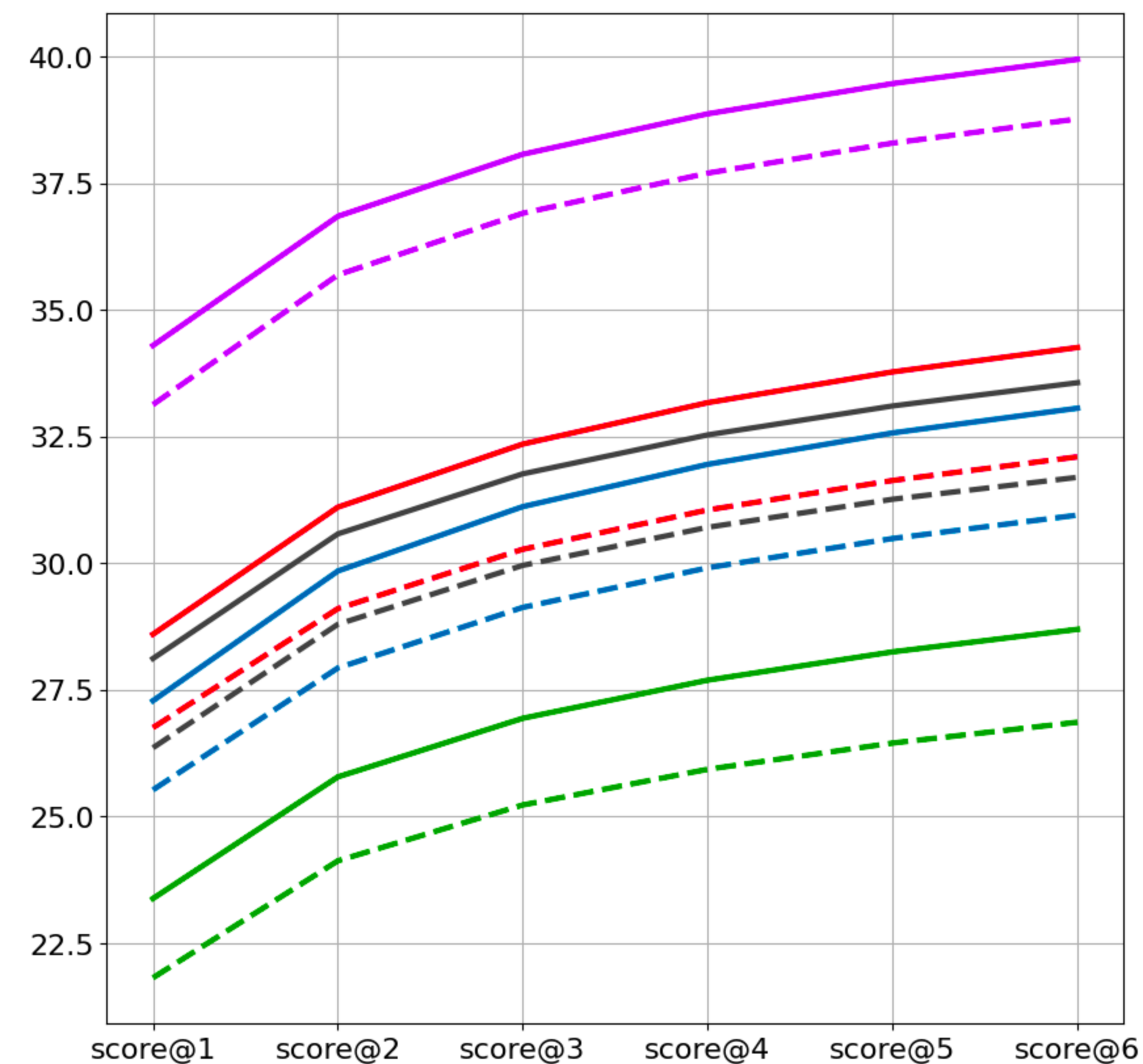


Evaluation - Performance

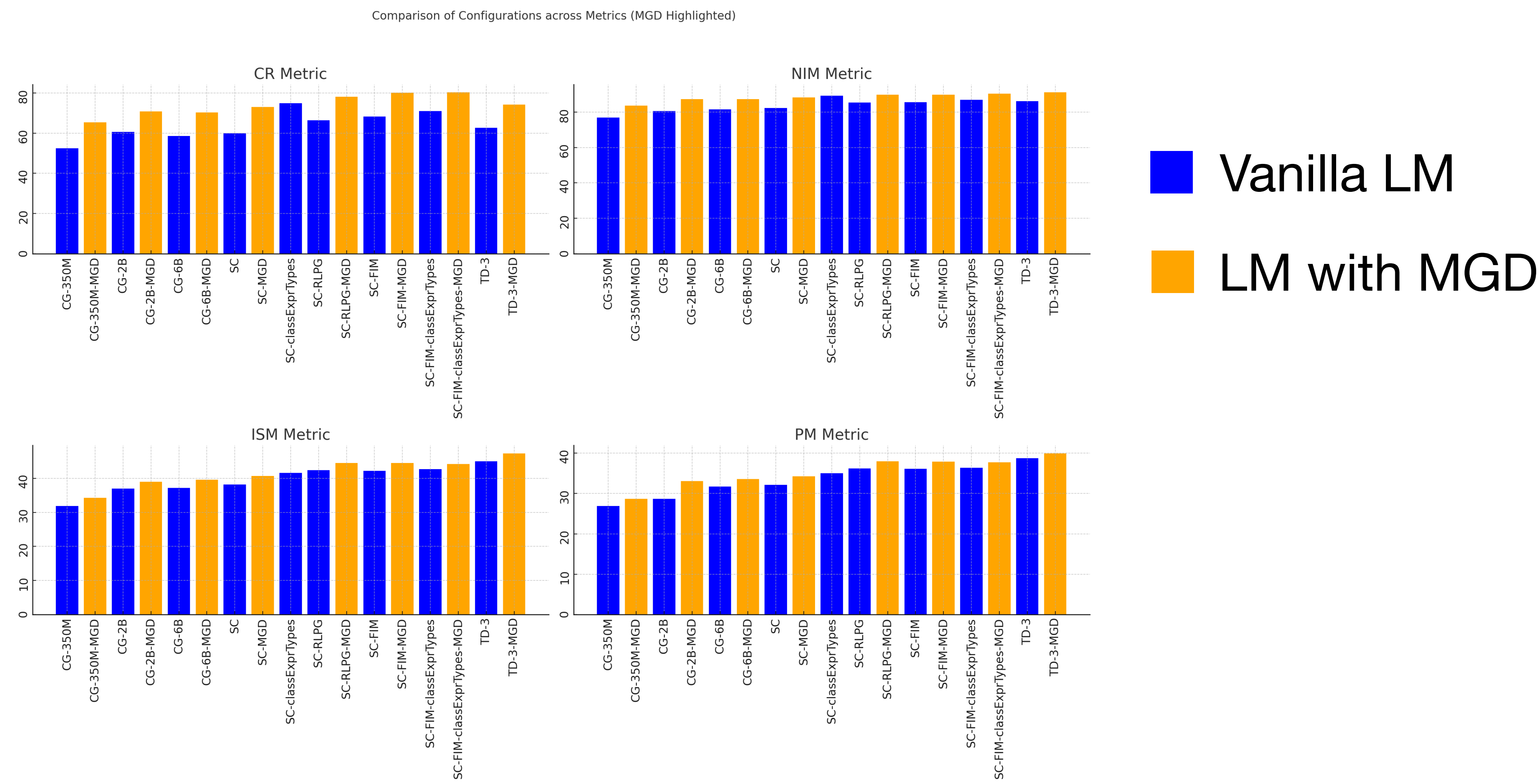
Next Sequence Match



Prefix Match



Evaluation - Performance

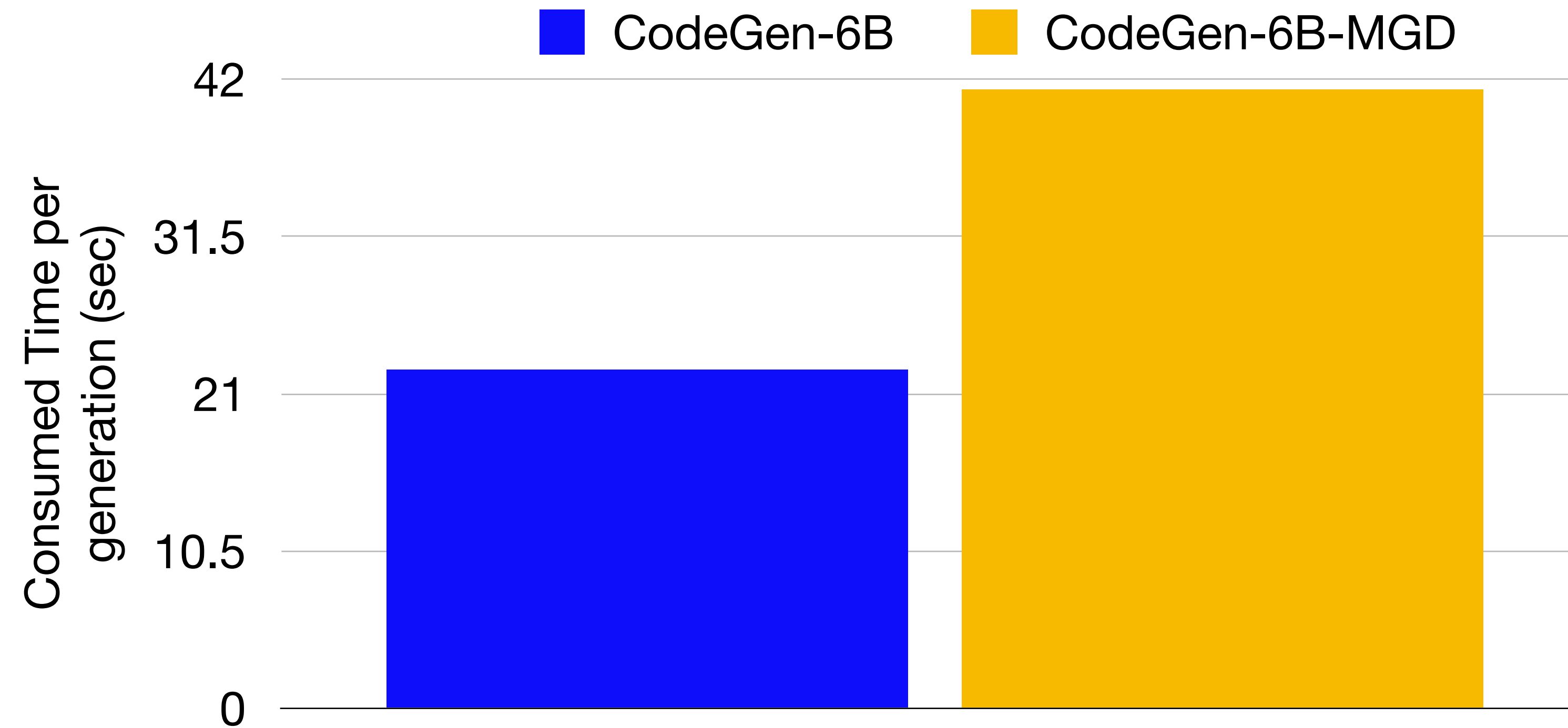


Evaluation - Overhead

- The monitoring process has lots of overhead

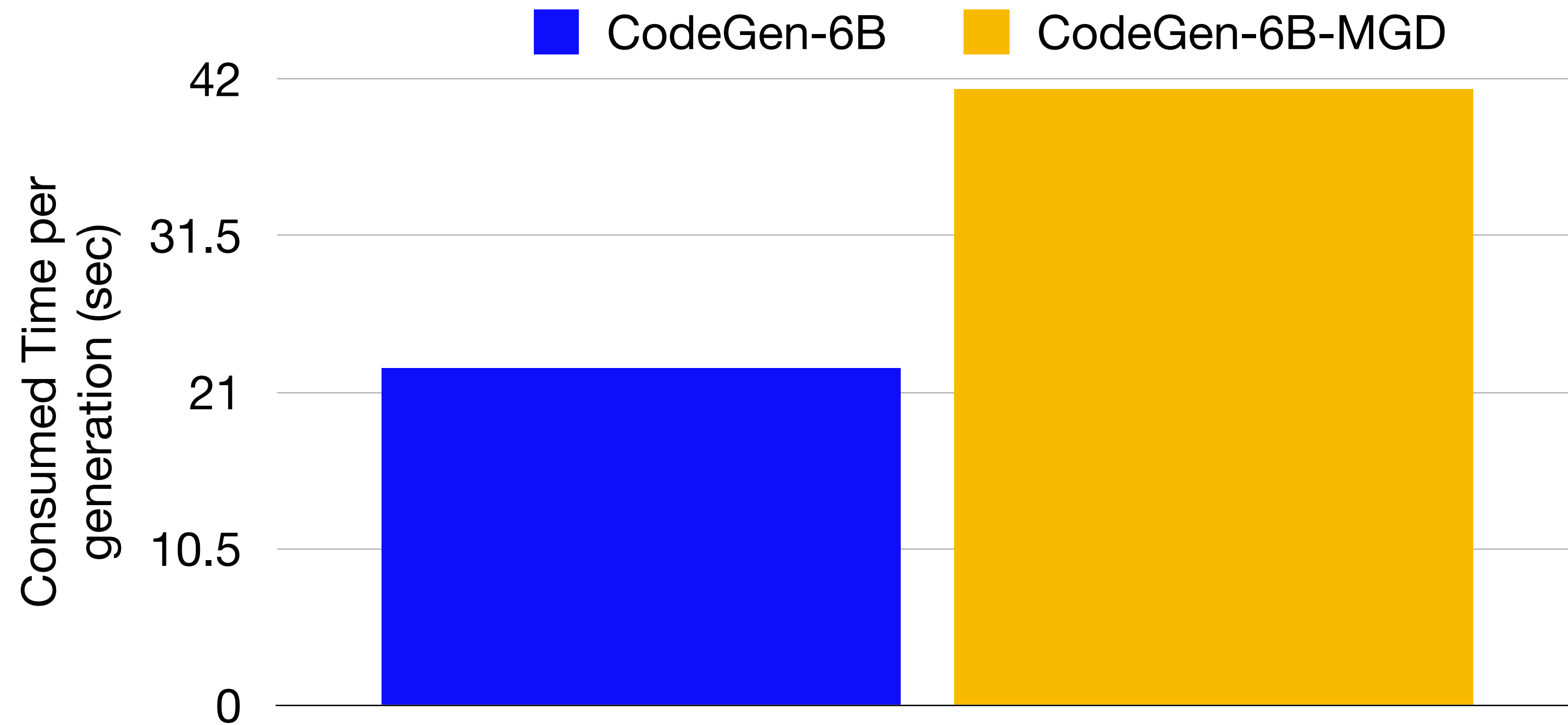
Evaluation - Overhead

- The monitoring process has lots of overhead



Evaluation - Overhead

- The monitoring process has lots of overhead



The average speed was 83.16% decreased

Generalization

Generalization

- Other Coding Scenarios

Generalization

- Other Coding Scenarios

- Valid class instantiation:

`Student std = new ???`

Generalization

- Other Coding Scenarios
 - Valid class instantiation:
`Student std = new ???`
 - `switch` over `enum`:
`switch state { case ???`

Generalization

- Other Coding Scenarios

- Valid class instantiation:

- `Student std = new ???`

- `switch` over enum:

- `switch state { case ???`

- A correct number of arguments:

- `def func1(int x, int y) ... func1(???)`

Generalization

- Other Coding Scenarios

- Valid class instantiation:

- `Student std = new ???`

- `switch` over enum:

- `switch state { case ???`

- A correct number of arguments:

- `def func1(int x, int y) ... func1(???)`

- Joint monitoring for multiple properties: # of arguments + Valid class init

Generalization

Generalization

- Complex Static Analysis

Generalization

- Complex Static Analysis
 - Typestate analysis (a.k.a. protocol analysis)

Generalization

- Complex Static Analysis
 - Typestate analysis (a.k.a. protocol analysis)
 - Analyze effective protocol for some resource

Generalization

- Complex Static Analysis
 - Typestate analysis (a.k.a. protocol analysis)
 - Analyze effective protocol for some resource
 - e.g. closing of the file must happen after the opening of the file

Generalization

- Complex Static Analysis
 - Typestate analysis (a.k.a. protocol analysis)
 - Analyze effective protocol for some resource
 - e.g. closing of the file must happen after the opening of the file

<i>TS1</i>	Android MediaPlayer	Typestate, Rust	<code>stop();</code>	<code>reset();</code>
<i>TS2</i>	GPIO Pins	Typestate, Rust	<code>set_input_pull_up();</code>	<code>set_input_high_z();</code>

My Opinion

My Opinion

- Too large overhead

My Opinion

- Too large overhead
 - No profiling

My Opinion

- Too large overhead
 - No profiling
 - How to overcome this?

My Opinion

- Too large overhead
 - No profiling
 - How to overcome this?
- Conditions for triggering monitoring are manual

My Opinion

- Too large overhead
 - No profiling
 - How to overcome this?
- Conditions for triggering monitoring are manual
 - Depending on manually defined patterns

My Opinion

- Too large overhead
 - No profiling
 - How to overcome this?
- Conditions for triggering monitoring are manual
 - Depending on manually defined patterns
 - e.g. "." dereference, class instantiation pattern, **switch** with **enum** pattern

My Opinion

- Too large overhead
 - No profiling
 - How to overcome this?
- Conditions for triggering monitoring are manual
 - Depending on manually defined patterns
 - e.g. "." dereference, class instantiation pattern, **switch** with **enum** pattern
 - How to automate this?

Summary

Summary

- This paper utilizes static analysis for the correct decoding of LMs

Summary

- This paper utilizes static analysis for the correct decoding of LMs
- Correctness of output was increased

Summary

- This paper utilizes static analysis for the correct decoding of LMs
- Correctness of output was increased
 - Compilation rate and other metrics are increased

Summary

- This paper utilizes static analysis for the correct decoding of LMs
- Correctness of output was increased
 - Compilation rate and other metrics are increased
- Requires lots of overhead

Summary

- This paper utilizes static analysis for the correct decoding of LMs
- Correctness of output was increased
 - Compilation rate and other metrics are increased
- Requires lots of overhead
 - Speed was 83.16% decreased

Summary

- This paper utilizes static analysis for the correct decoding of LMs
- Correctness of output was increased
 - Compilation rate and other metrics are increased
- Requires lots of overhead
 - Speed was 83.16% decreased
- Generalizable for other model, programming language, and static analyzer