Repository-Level Prompt Generation for Large Language Models of Code

Université de Montréal

Mila

Google Research

Large Language Models (LLMs) of Code

- Used in code-assistants (e.g. GitHub Copilot, Bard).
- Struggle when encountered with code not seen during training.
  - Proprietary Software
  - WIP Code Project
- Finetuning on code from the local repository is often impractical
  - Black-box access to strong code LLMs.
  - Computationally expensive as well as challenging to update frequently.
- Leverage relevant context from other files in the repository (e.g. imports, parent classes, files with similar names), but only during inference.

Large Language Models (LLMs) of Code

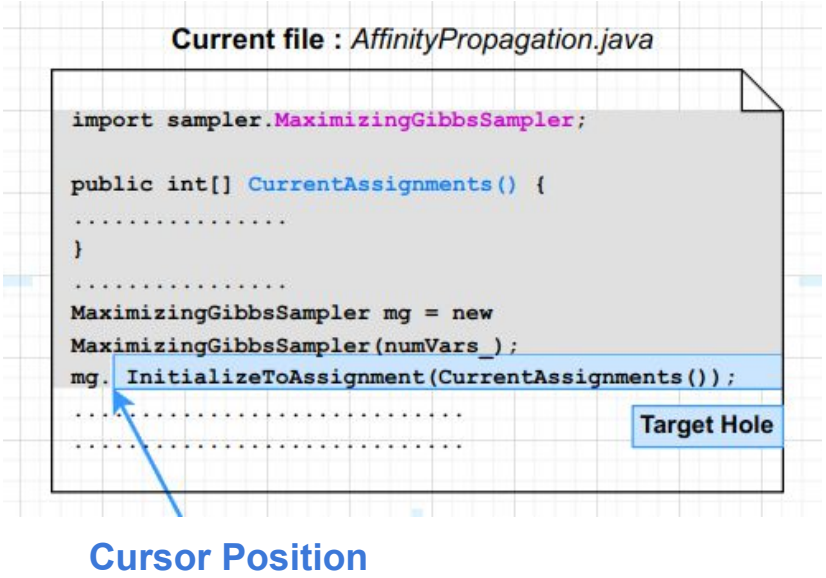
- Used in code-assistants (e.g. GitHub Copilot, Bard).
- Struggle when encountered with code not seen during training.
  - Proprietary Software
  - WIP Code Project
- Finetuning on code from the local repository is often impractical
  - Black-box access to strong code LLMs.
  - Computationally expensive as well as challenging to update frequently.
- Leveraging relevant context from other files in the repository (e.g. imports, parent classes, files with similar names), but only during inference.

Select relevant repository context in a way that doesn't require access to the weights of the LLM.

Code Completion in an IDE

Our setting* simulates editing a file in an IDE

- Objective: **Complete the line following the cursor** (***target hole***)
- There can be code after the cursor.



*As in [On-the-Fly Adaptation of Source Code Models](), *Disha Shrivastava, Hugo Larochelle, Daniel Tarlow*

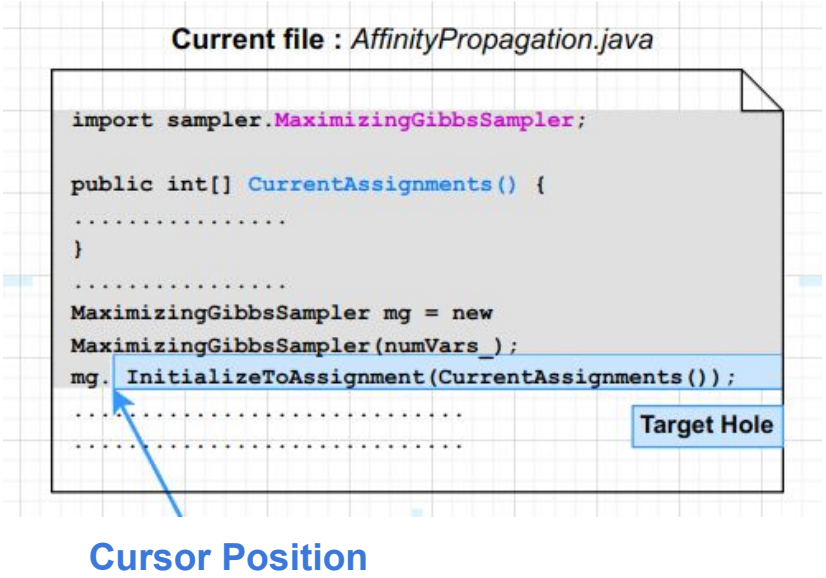
Code Completion in an IDE

Our setting* simulates editing a file in an IDE

- Objective: **Complete the line following the cursor** (***target hole***)
- There can be code after the cursor.

Vanilla Training: given a prefix of code, predict the next tokens.

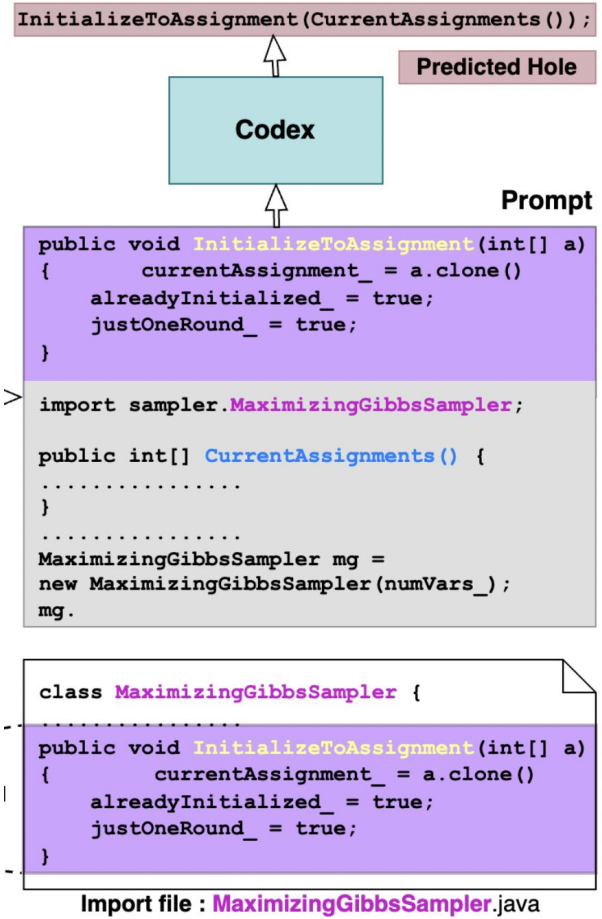
Vanilla Inference (to match the training): take **context prior to the cursor** in the current file and predict the **target hole**.



Repository Context in the Prompt

Take an LLM trained in the usual way, but use it differently during inference.

During inference, in addition to the prior context in the current file, we add relevant context from the repository in the prompt.



Repository Context in the Prompt

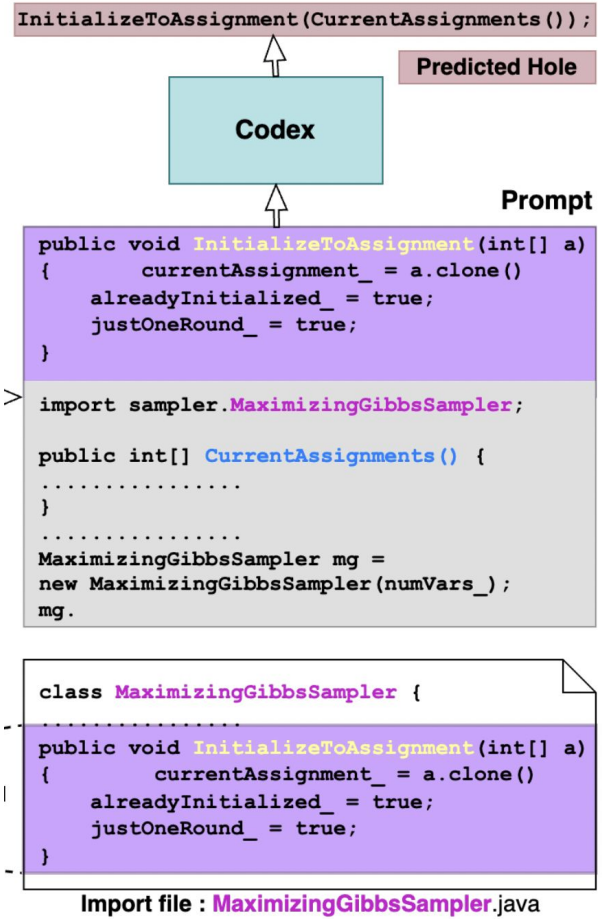
Take an LLM trained in the usual way, but use it differently during inference.

During inference, in addition to the prior context in the current file, we add relevant context from the repository in the prompt.

To select relevant context, we want a method that

- Utilizes Structure of the repository
- Utilizes Context in relevant files

Solution: Use domain knowledge to guide the selection of relevant context via a set of **prompt proposals.**



Prompt Proposals

- **Prompt Source**: where to take the context from?
- **Prompt Context Type**: what to take from the prompt source?

Prompt Proposals

- **Prompt Source**: where to take the context from?
- **Prompt Context Type**: what to take from the prompt source?

10 Prompt Sources

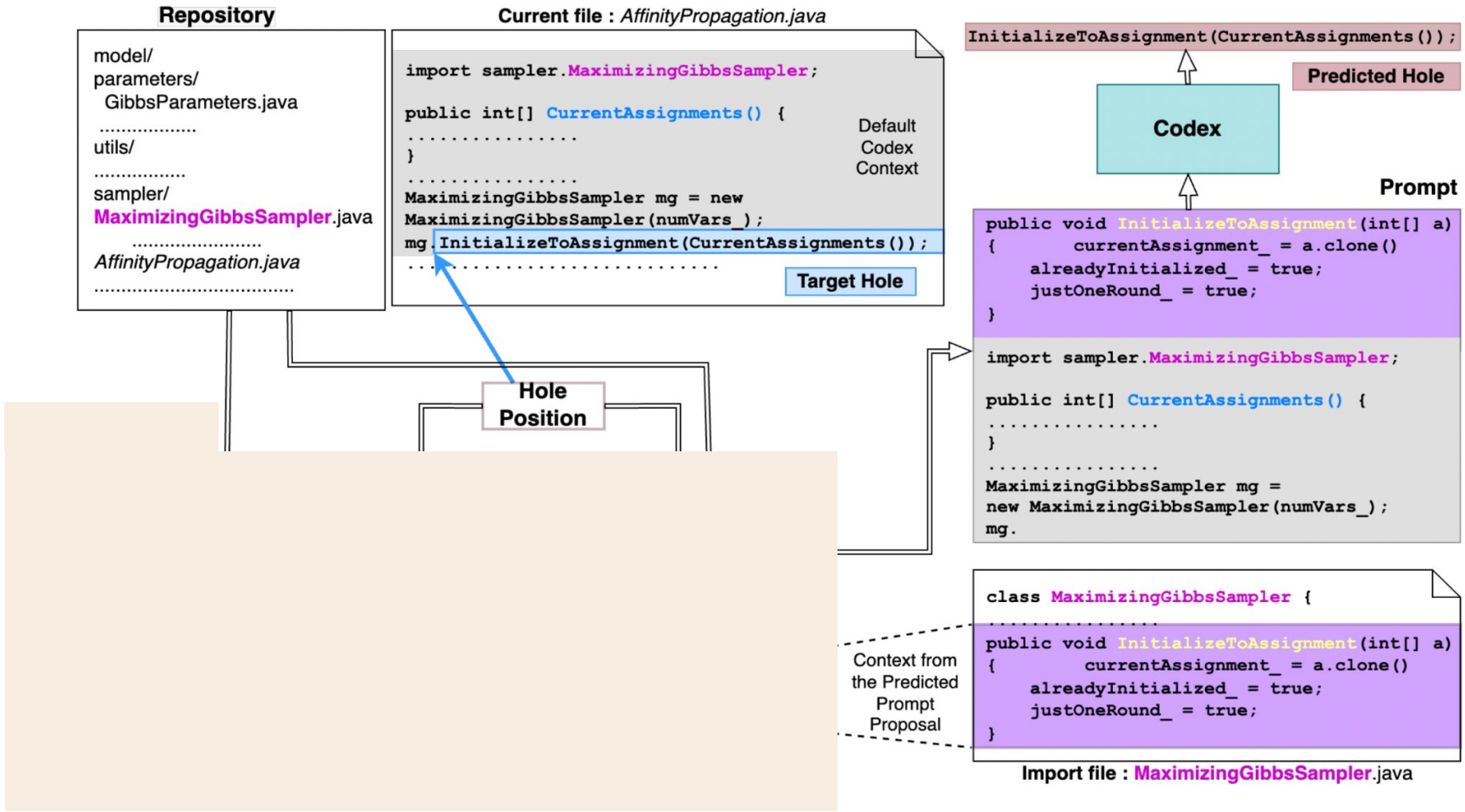
- Current file
- Parent Class file
- Sibling file
- Similar name file
- Child Class file
- Import of the above

In total, we propose a list of **63** prompt proposals

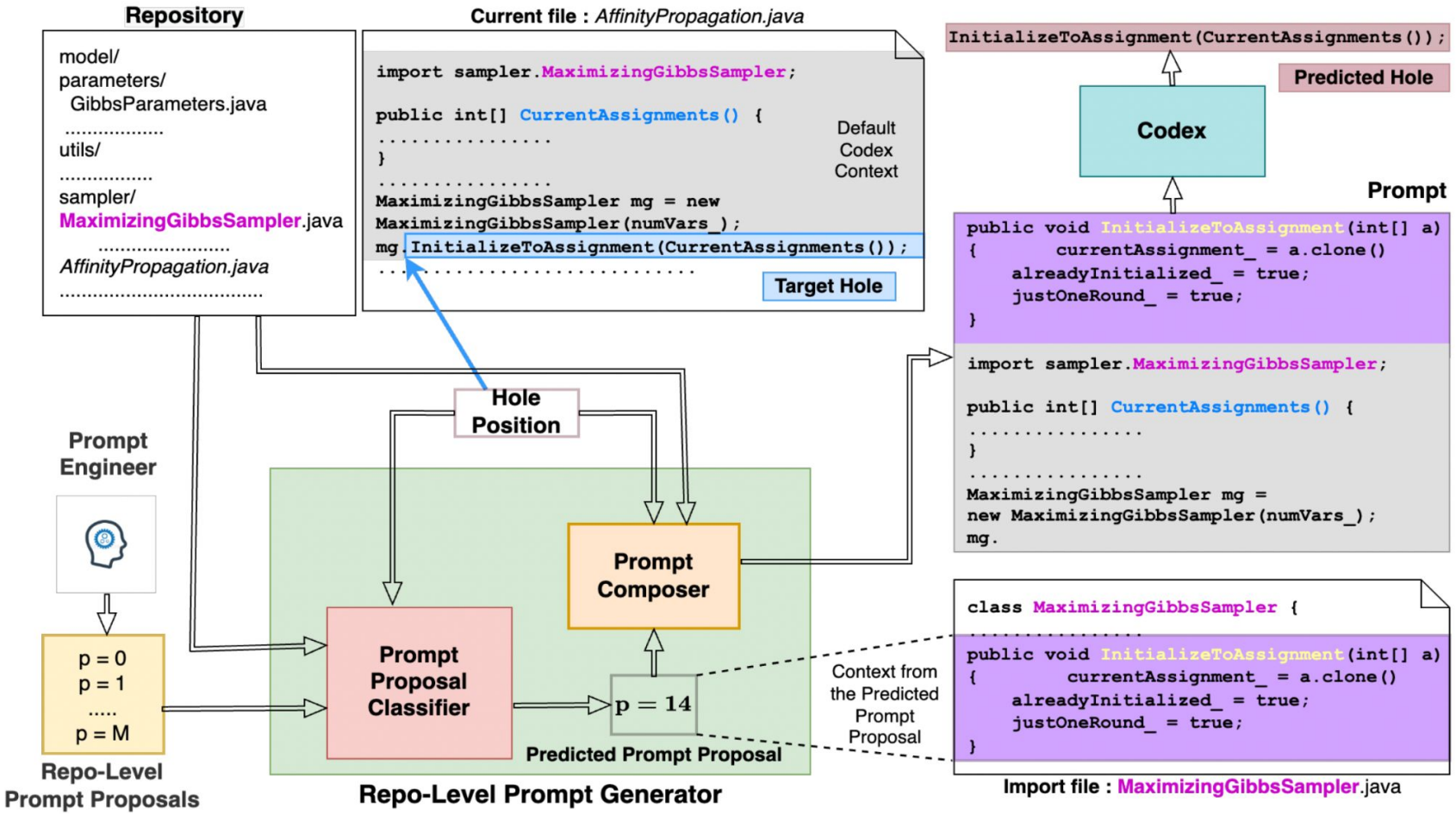
7 Prompt Context Types

- Lines after the cursor
- Identifiers
- Field declarations
- Type identifiers
- String literals
- Method names
- Method names and bodies

Repo-Level Prompt Generator (RLPG)



Repo-Level Prompt Generator (RLPG)



Prompt Proposal Classifier

- Multi-label binary classifier that *learns* to select a prompt proposal that is likely to lead to a successful prediction for the target hole.
- *Example-Specific:* different prediction conditioned on the context around the hole.
- *Success* = When inclusion of the context from the prompt proposal in the prompt leads to an accurate prediction of the hole.

Results

Table 2. Performance of the oracle relative to Codex.

Data Split	Success Rate Codex(%)	Success Rate Oracle(%)	Rel. ↑ over Codex(%)
Train	59.78	80.29	34.31
Val	62.10	79.05	27.28
Test	58.73	79.63	35.58

Including contexts from our prompt proposals during inference is quite useful even though Codex has not seen them during training.

Results

Table 2. Performance of the oracle relative to Codex.

Data Split	Success Rate Codex(%)	Success Rate Oracle(%)	Rel. ↑ over Codex(%)
Train	59.78	80.29	34.31
Val	62.10	79.05	27.28
Test	58.73	79.63	35.58

Using RLPG with prompt proposal classifier shows significant improvements.

Table 3. Success Rate (SR) of different methods on the test data when averaged across all holes.

	Method	Success Rate(%)	Rel. ↑(%)
	Codex (Chen et al., 2021)	58.73	-
	Oracle	79.63	35.58
Retrieval Baselines	Random	58.13	-1.02
	Random NN	58.98	0.43
	File-level BM25	63.14	7.51
	Identifier Usage (Random)	64.93	10.55
	Identifier Usage (NN)	64.91	10.52
Non-learned RLPG	Fixed Prompt Proposal	65.78	12.00
	RLPG-BM25	66.41	13.07
Learned RLPG	RLPG-H	68.51	16.65
	RLPG-R	67.80	15.44

Thank You! Checkout our poster :)

Paper: https://arxiv.org/abs/2206.12839

Code: https://github.com/shrivastavadisha/repo_level_prompt_generation

Correspondence: dishu.905@gmail.com