

LLM Code Generation and Test Synthesis

Multi-Agentic Orchestration for Partial Order Estimation of Code Edits

Lee Zong Xun

School of Computing
National University of Singapore

August 28, 2025

Overview

1. Introduction
2. Methodology
3. Approach
4. Evaluation
5. Experimental Results
6. Conclusion

Motivation

Before 2022: Manual Coding

- Developers wrote every line of code manually. Fundamentally human-driven.
- **Tools:** Syntax highlighting, debugging, auto indentation.

In 2022: Revolutionary AI Copilots

- **GitHub Copilot** leveraged **OpenAI's Codex** to introduce AI-assisted coding.
- Developers could:
 - Auto-complete entire functions.
 - Receive explanations for code snippets.
 - Reduce boilerplate coding.

Motivation

Fast forward to 2024...and today

- Unlike Copilots, AI Agents work autonomously.
- They perform **multi-step problem-solving**, including:
 - Identifying issues.
 - Writing and refactoring code.
 - Running tests and debugging.
- Examples:
 - **ManusAI** - Automates entire workflows.
 - **Deep Research** by Google and OpenAI

Modern AI-IDEs

- CursorAI
- Windsurf
- TraeAI

Result? Productivity saw significant improvements, with some companies experiencing up to 10x efficiency gains.

Modern AI-IDEs

AI-IDEs generate high-entropy edits across multiple files at once. This is in contrast to earlier versions where they could only generate predictions locally and one at a time.

- Developers pass control to AI agents to build applications, and complete tasks.
- Inclusive (human-in-the-loop) design enforces that the user should always be in control (or at least be aware of the actions of the AI agents).

Challenges

Can we evaluate the coherence of the sequence of edits that the model suggests?

- No benchmark on what it means to generate logical and coherent edit suggestions.
- Compromises mental flow.
 - Developers need to manually adjust AI-generated code.
 - Need to context switch frequently.

Objective

To enable the evaluation of LLMs in next edit suggestions, we first need to recover the original commit process.

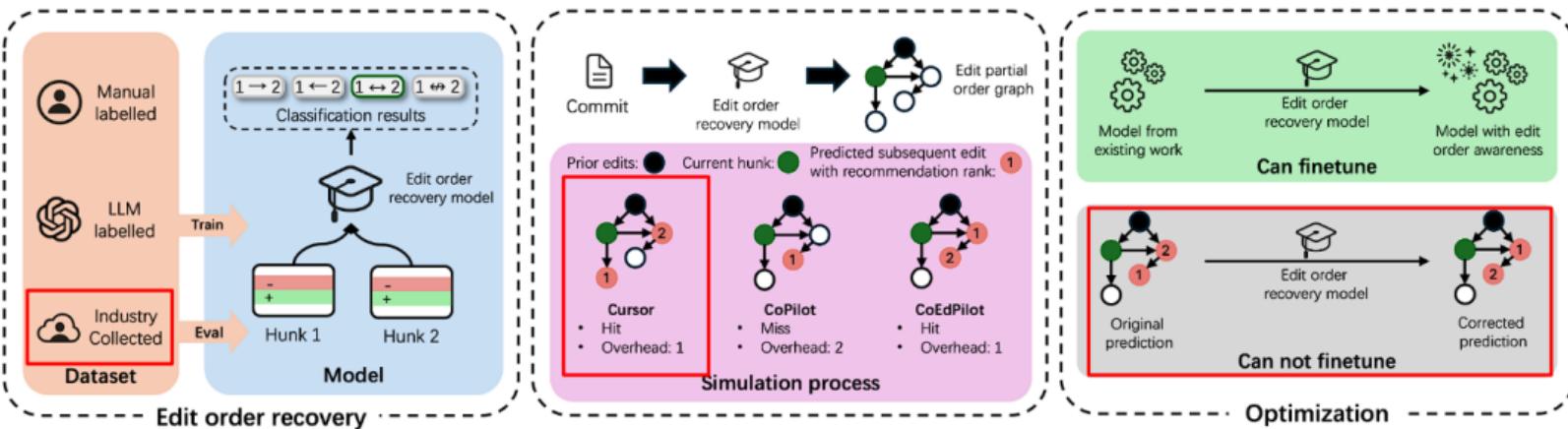


Figure: Research Plan

Problem Formulation

Objective: Determine the partial ordering of code edits in a set \mathbf{H} .

- Given two edits $h_i, h_j \in \mathbf{H}$, define their sequential relationship:

$$f(h_i, h_j) = \begin{cases} h_i \rightarrow h_j, & h_i \text{ must occur before } h_j \\ h_i \leftarrow h_j, & h_j \text{ must occur before } h_i \\ h_i \leftrightarrow h_j, & \text{no strict order} \\ h_i \times h_j, & \text{independent edits} \end{cases}$$

Construct a Directed Acyclic Graph (DAG) to model dependencies among edits.

Partial Order Reduction (POR)

Challenge: The number of possible orderings grows exponentially.

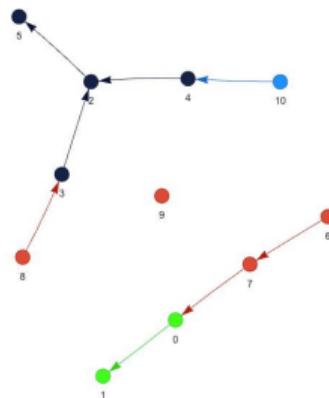


Figure: Partial order reduction reduces the search space from 13,860 to 36.

Solution: Apply POR to prune unnecessary interleavings of independent edits while preserving essential dependencies.

Key Components of POR

- **Independence:** Edits are independent if their execution order does not affect the system state.

$$t_1(s) \xrightarrow{t_2} s' \iff t_2(s) \xrightarrow{t_1} s' \text{ (Commutativity)}$$

and execution of one transition does not disable the other (Non-interference):

$$t_1 \text{ is enabled in state } s \implies t_2 \text{ is also enabled in } t_1(s).$$

- **Dependency:** If two edits influence each other, their order must be preserved.
- **Equivalence Classes:** Equivalent execution paths are merged to reduce complexity.
- **Reduction Criterion:** Every equivalence class of executions contains at least one representative path that is explored.

$$\forall t \in T, \exists t' \in T_{\text{red}}(s) : t \sim t'.$$

Dependency Analysis

Key Observations:

- A single edit region may contain multiple edit semantics, which may require multiple rounds of parsing before full processing.
- The order of edits should be arranged based on their dependencies to ensure correct sequencing.
- Independent edits can be applied concurrently with no regards to consequences.
- Edits within the same scope are more likely to occur sequentially due to their local dependencies and proximity.
- Inter-file dependencies are often mediated through shared components (e.g., imports, global variables, or shared libraries).

Observation 1

Commit 2e7635b

andrewgiessel authored and soumith committed on May 3, 2017

Add flexible bilinear upsampling aspect ratio redux (#1317)

```
diff --git a/torch/nn/_functions/thnn/upsampling.py b/torch/nn/_functions/thnn/upsampling.py
--- a/torch/nn/_functions/thnn/upsampling.py
+++ b/torch/nn/_functions/thnn/upsampling.py
@@ -8,4,6 +8,8 @@ from torch._thnn import type2backend
 5
 6 3   from . import _all_functions
 7 4   from ...modules.utils import _pair
 8 5   + from ...functional import _check_bilinear_2d_scale_factor
 9 6
10 7
11 8 class _UpsamplingBase(Function):
12 9     def __init__(self, size=None, scale_factor=None):
13 10         super(_UpsamplingBase, self).__init__()
14 11         if size is None and scale_factor is None:
15 12             raise ValueError('either size or scale_factor should be defined')
16 13         if scale_factor is not None and not isinstance(scale_factor, Integral):
17 14             raise ValueError('scale_factor must be of integer type')
18 15         if size is not None and not isinstance(size, tuple):
19 16             size = (size, size)
20 17         + if scale_factor is not None and not isinstance(scale_factor, (Integral, tuple)):
21 18             raise ValueError('scale_factor must be of integer type or tuple of integer types')
22 19         self.size = size
23 20         self.scale_factor = scale_factor
24 21
25 22     2   class UpsamplingBilinear2d(_UpsamplingBase):
26 23         def __init__(self, size=None, scale_factor=None):
27 24             super(UpsamplingBilinear2d, self).__init__(size, scale_factor)
28 25             if self.scale_factor is not None and not isinstance(self.scale_factor, Integral):
29 26                 raise ValueError('scale_factor must be of integer type for nearest neighbor sampling')
30 27             self.size = _pair(self.size) if self.size is not None else None
31 28
32 29     2   class UpsamplingNearest2d(_UpsamplingBase):
33 30         def forward(self, input):
34 31             assert input.dim() == 4
```

Update subclass, in this
upsampling alg, scale_factor
must be int

Update superclass, allow scale_factor to be
int or tuple (for different upsampling alg)

5

18

2

Update subclass, in this upsampling alg, scale_factor must be tuple type, exemplified by _check_bilinear_2d_scale_factor()

```
diff --git a/torch/nn/_functions/thnn/upsampling.py b/torch/nn/_functions/thnn/upsampling.py
--- a/torch/nn/_functions/thnn/upsampling.py
+++ b/torch/nn/_functions/thnn/upsampling.py
@@ -65,73 +65,130 @@ class UpsamplingBilinear2d(_UpsamplingBase):
    def __init__(self, size=None, scale_factor=None):
        super(UpsamplingBilinear2d, self).__init__(size, scale_factor)
        if self.scale_factor is not None:
            self.scale_factor = _check_bilinear_2d_scale_factor(self.scale_factor)
        self.size = _pair(self.size) if self.size is not None else None
    def forward(self, input):
        assert input.dim() == 4
@@ -78,86 +86,104 @@ if self.scale_factor is not None:
    if self.scale_factor is not None:
        if self.scale_factor is not None:
            self.output_size = (
                input.size(2) * self.scale_factor,
                input.size(3) * self.scale_factor,
                input.size(2) * self.scale_factor[0],
                input.size(3) * self.scale_factor[1],
            )
        else:
            self.output_size = self.size
@@ -106,6 +106,6 @@ + 00 -106,6 +122,9 00 def backward(self, grad_output):
    )
    return grad_input
@@ -122,2 +122,2 @@ + 106,122 + def _setstate__(self, state):
    self._dict_.update(state)
    self.scale_factor = _tuple(self.scale_factor)
@@ -128,2 +128,2 @@ + 127,128 + _all_functions.append(UpsamplingNearest2d)
    _all_functions.append(UpsamplingBilinear2d)
@@ -130,2 +130,2 @@ + 130,130 + _all_functions.append(UpsamplingBilinear2d)
```

4

8

9

10

Figure: One edit region may contain multiple edit semantic, that may be visited twice

Observation 2

```
    torch/nn/functional.py
```

```
*** -0,-1,-2,-3,-4,-5,-6,-7 ***  
1 1     """Functional interface"""  
2 2  
3 3 + from numbers import Integral  
4 4 +  
5 5 import torch  
6 6 from . import _functions  
7 7 from .modules import utils  
8 8  
9 9 @@ -610,11 +612,23 @@ def upsample_bilinear2d(input, size=None, scale_factor=None):  
10 10     Args:  
11 11         input (Variable): input  
12 12         size (int or Tuple[int, int]): output spatial size.  
13 13 -     scale_factor (int): multiplier for spatial size. Has to be an integer.  
14 14 +     scale_factor (int or Tuple[int, int]): multiplier for spatial size  
15 15 +         ...  
16 16  
17 17         return _functions.thnn.UpsamplingBilinear2d(size, scale_factor)(input)  
18 18  
19 19 + def _check_bilinear_2d_scale_factor(scale_factor):  
20 20 +     scale_factor = _pair(scale_factor)  
21 21     try:  
22 22         assert unscale_factor ==  
23 23         assert all(isinstance(s, Integral) and s >= 1 for s in scale_factor)  
24 24     except AssertionError as e:  
25 25         raise ValueError('scale_factor must be a non-negative integer, '\br/>26 26         'or a tuple of non-negative integers for bilinear upsamplings, but got: '\br/>27 27         '{}'.format(scale_factor))  
28 28     return scale_factor  
29 29 +  
30 30 +  
31 31 +  
32 32 def pad(input, pad, mode='constant', value=0):  
33 33     """Pads tensor.  
34 34
```

Update superclass class `_UpsamplingBase(Module)`; allow `scale_factor` to be `int` or `tuple` (for different upsampling alg), `torch/nn/modules` is API for user

```
    torch/nn/modules/upampling.py
```

```
*** -11,-12,-13 ***  
11 11     super(_UpsamplingBase, self).__init__()  
12 12     if size is None and scale_factor is None:  
13 13         raise ValueError('either size or scale_factor should be defined')  
14 14 -     if scale_factor is not None and not isinstance(scale_factor, Integral):  
15 15 -         raise ValueError('scale_factor must be of integer type')  
16 16 -     self.size = _pair(size)  
17 17 +     if scale_factor is not None and not isinstance(scale_factor, (Integral, tuple)): 11  
18 18 +         raise ValueError('scale_factor must be of integer type or tuple of integer types')  
19 19 +     self.size = size  
20 20 +     self.scale_factor = scale_factor  
21 21  
22 22     def __repr__(self):  
23 23  
24 24 @@ -65,-6 +65,12 @@ class UpsamplingNearest2d(_UpsamplingBase): Update subclass  
25 25     Args:  
26 26         input (Variable): input  
27 27         size (int or Tuple[int, int]): output spatial size.  
28 28 -     scale_factor (int): multiplier for spatial size. Has to be an integer.  
29 29 +     scale_factor (int or Tuple[int, int]): multiplier for spatial size  
30 30 +         ...  
31 31 +  
32 32 +     def __init__(self, size=None, scale_factor=None):  
33 33 +         super(UpsamplingNearest2d, self).__init__(size, scale_factor)  
34 34     if self.scale_factor is not None and not isinstance(self.scale_factor, Integral):  
35 35         raise ValueError('scale_factor must be of integer type for nearest neighbor sampling')  
36 36     self.size = _pair(self.size) if self.size is not None else None  
37 37  
38 38     def forward(self, input):  
39 39         return F.upsample_nearest(input, self.size, self.scale_factor)  
40 40  
41 41 @@ -110,-5 +116,12 @@ class UpsamplingBilinear2d(_UpsamplingBase): Update subclass  
42 42     Args:  
43 43         input (Variable): input  
44 44         size (int or Tuple[int, int]): output spatial size.  
45 45 -     scale_factor (int): multiplier for spatial size. Has to be an integer.  
46 46 +     def __init__(self, size=None, scale_factor=None):  
47 47 +         super(UpsamplingBilinear2d, self).__init__(size, scale_factor)  
48 48     if self.scale_factor is not None:  
49 49         self.scale_factor = f_.check_bilinear_2d_scale_factor(self.scale_factor)  
50 50     self.size = _pair(self.size) if self.size is not None else None  
51 51  
52 52     def forward(self, input):  
53 53         return F.upsample_bilinear(input, self.size, self.scale_factor)
```

Figure: Arrange edit order by their dependency

Observation 3

Add ability to get a list of supported pipeline tasks (#14732)

codesue authored on Dec 13, 2021 (Verified)

```
diff --git a/src/transformers/pipelines/_init__.py b/src/transformers/pipelines/_init__.py
--- a/src/transformers/pipelines/_init__.py
+++ b/src/transformers/pipelines/_init__.py
@@ -20,7 +20,7 @@ # See the License for the specific language governing permissions and
@@ -21,21 +21,21 @@ # limitations under the License.
@@ -22,22 +22,22 @@ import warnings
@@ -23,23 +23,23 @@ - from typing import TYPE_CHECKING, Any, Dict, Optional, Tuple, Union
@@ -24,24 +24,24 @@ + from typing import TYPE_CHECKING, Any, Dict, List, Optional, Tuple, Union
@@ -25,25 +25,25 @@ from ..configuration_utils import PretrainedConfig
@@ -26,26 +26,26 @@ from ..feature_extraction_utils import PreTrainedFeatureExtractor
@@ -27,27 +27,27 @@ 
@@ -28,28 +28,28 @@ 00 -20,7 +20,7 @@ 00
@@ -29,29 +29,29 @@ 252,252 -->
@@ -30,29 +30,29 @@ 253,253 Extract get supported task as a new function
@@ -31,29 +31,29 @@ 254,254
@@ -32,29 +32,29 @@ 255,255 + def get_supported_tasks() -> List[str]:
@@ -33,29 +33,29 @@ 256,256 + """
@@ -34,29 +34,29 @@ 257,257 + Returns a list of supported task strings.
@@ -35,29 +35,29 @@ 258,258 +
@@ -36,29 +36,29 @@ 259,259 +     supported_tasks = list(SUPPORTED_TASKS.keys()) + list(TASK_ALIASES.keys())
@@ -37,29 +37,29 @@ 260,260 +     supported_tasks.sort()
@@ -38,29 +38,29 @@ 261,261 +     return supported_tasks
@@ -39,29 +39,29 @@ 262,262 +
@@ -40,29 +40,29 @@ 263,263 +
@@ -41,29 +41,29 @@ 264,264     def get_task(model: str, use_auth_token: Optional[str] = None) -> str:
@@ -42,29 +42,29 @@ 265,265     tmp = io.BytesIO()
@@ -43,29 +43,29 @@ 266,266     headers = {}
@@ -44,29 +44,29 @@ 267,267 
@@ -45,29 +45,29 @@ 268,268 00 -320,9 +320,7 @@ 00 def check_task(task: str) -> Tuple[Dict, Any]:
@@ -46,29 +46,29 @@ 269,269     return targeted_task, (tokens[1], tokens[3])
@@ -47,29 +47,29 @@ 270,270     raise KeyError(f"Invalid translation task {task}, use 'translation_XX_to_YY' format")
@@ -48,29 +48,29 @@ 271,271 
@@ -49,29 +49,29 @@ 272,272 -     raise KeyError(
@@ -50,29 +50,29 @@ 273,273 -         f"Unknown task {task}, available tasks are {list(SUPPORTED_TASKS.keys()) +"
@@ -51,29 +51,29 @@ 274,274 -         ['translation_XX_to_YY']}"
@@ -52,29 +52,29 @@ 275,275 -     )
@@ -53,29 +53,29 @@ 276,276 +     raise KeyError(f"Unknown task {task}, available tasks are {get_supported_tasks() +"
@@ -54,29 +54,29 @@ 277,277 +         ['translation_XX_to_YY']}")"
@@ -55,29 +55,29 @@ 278,278 
@@ -56,29 +56,29 @@ 279,279 326 333
```

The diagram illustrates parallel edit compositions between two code files. Red arrows point from changes in `src/transformers/commands/serving.py` to `src/transformers/commands/run.py`, indicating that both files were edited simultaneously or in parallel.

```
diff --git a/src/transformers/commands/serving.py b/src/transformers/commands/serving.py
--- a/src/transformers/commands/serving.py
+++ b/src/transformers/commands/serving.py
@@ -15,7 +15,7 @@ 00 -15,7 +15,7 @@ 00
@@ -16,16 +16,16 @@ 15,16 from argparse import ArgumentParser, Namespace
@@ -17,17 +17,17 @@ 16,17 from typing import Any, List, Optional
@@ -18,18 +18,18 @@ 17,18 - from ..pipelines import SUPPORTED_TASKS, TASK_ALIASES, Pipeline, pipeline
@@ -19,19 +19,19 @@ 18,19 + from ..pipelines import Pipeline, get_supported_tasks, pipeline
@@ -20,20 +20,20 @@ 19,20 from ..utils import logging
@@ -21,21 +21,21 @@ 20,21 from . import BaseTransformersCLICommand
@@ -22,22 +22,22 @@ 
@@ -23,23 +23,23 @@ 00 -184,7 +184,7 @@ 00 def register_subcommand(parser: ArgumentParser):
@@ -24,24 +24,24 @@ 104,104 serve_parser.add_argument(
@@ -25,25 +25,25 @@ 105,105     "--task",
@@ -26,26 +26,26 @@ 106,106     type=str,
@@ -27,27 +27,27 @@ 107,107 -     choices=list(SUPPORTED_TASKS.keys()) + list(TASK_ALIASES.keys()),
@@ -28,28 +28,28 @@ 108,108 +     choices=get_supported_tasks(),
@@ -29,29 +29,29 @@ 109,109     help="The task to run the pipeline on",
@@ -30,29 +30,29 @@ 110,110     )
@@ -31,29 +31,29 @@ 111,111     serve_parser.add_argument("--host", type=str, default="localhost", help="Interface the server will
@@ -32,29 +32,29 @@ 112,112     listen on."
@@ -33,29 +33,29 @@ 113,113 Parallel
@@ -34,29 +34,29 @@ 114,114
@@ -35,29 +35,29 @@ 115,115
@@ -36,29 +36,29 @@ 116,116
@@ -37,29 +37,29 @@ 117,117 - from ..pipelines import SUPPORTED_TASKS, TASK_ALIASES, Pipeline, PipelineDataFormat, pipeline
@@ -38,29 +38,29 @@ 118,118 + from ..pipelines import Pipeline, PipelineDataFormat, get_supported_tasks, pipeline
@@ -39,29 +39,29 @@ 119,119 from ..utils import logging
@@ -40,29 +40,29 @@ 120,120 from . import BaseTransformersCLICommand
@@ -41,29 +41,29 @@ 
@@ -42,29 +42,29 @@ 121,121 00 -63,9 +63,7 @@ 00 def __init__(self, nlp: Pipeline, reader: PipelineDataFormat):
@@ -43,29 +43,29 @@ 122,122     staticmethod
@@ -44,29 +44,29 @@ 123,123     def register_subcommand(parser: ArgumentParser):
@@ -45,29 +45,29 @@ 124,124     run_parser = parser.add_parser("run", help="Run a pipeline through the CLI")
@@ -46,29 +46,29 @@ 125,125     run_parser.add_argument(
@@ -47,29 +47,29 @@ 126,126     "--task", choices=list(SUPPORTED_TASKS.keys()) + list(TASK_ALIASES.keys()), help="Task to run"
@@ -48,29 +48,29 @@ 127,127 -     )
@@ -49,29 +49,29 @@ 128,128 +     run_parser.add_argument("--task", choices=get_supported_tasks(), help="Task to run")
@@ -50,29 +50,29 @@ 129,129     run_parser.add_argument("--input", type=str, help="Path to the file to use for inference")
@@ -51,29 +51,29 @@ 130,130     run_parser.add_argument("--output", type=str, help="Path to the file that will be used post to write
@@ -52,29 +52,29 @@ 131,131     results")
@@ -53,29 +53,29 @@ 132,132     run_parser.add_argument("--model", type=str, help="Name or path to the model to instantiate.")
@@ -54,29 +54,29 @@ 133,133
```

Figure: Parallelism between similar edit compositions

Observation 4

feat(rnd): Add staticOutput field on block API (#7802)

majdyz authored on Aug 16 · 12 / 16 · Verified

```
diff --git a/rnd/autogpt_server/autogpt_server/data/block.py b/rnd/autogpt_server/autogpt_server/data/block.py
--- a/rnd/autogpt_server/autogpt_server/data/block.py
+++ b/rnd/autogpt_server/autogpt_server/data/block.py
@@ -127,6 +127,7 @@ class Block(BlockData):
    test_output: BlockData | list[BlockData] | None = None,
    test_mock: dict[str, Any] | None = None,
    disabled: bool = False,
+   static_output: bool = False,
@@ -130,7 +130,13 @@ class Block(BlockData):
    ): ...
    """
    Initialize the block with the given schema.
@@ -131,3 +131,13 @@ class Block(BlockData):
    test_output: The list or single expected output if the test_input is run.
    test_mock: function names on the block implementation to mock on test run.
    disabled: If the block is disabled, it will not be available for execution.
+   static_output: Whether the output links of the block are static by default.
@@ -143,13 +143,13 @@ class Block(BlockData):
    test_output: ...
    test_mock: ...
    disabled: ...
+   static_output: ...
@@ -146,13 +146,13 @@ class Block(BlockData):
    test_output: ...
    test_mock: ...
    disabled: ...
+   static_output: ...
@@ -149,13 +149,13 @@ class Block(BlockData):
    self.id = id
    self.input_schema = input_schema
@@ -154,13 +154,13 @@ class Block(BlockData):
    self.categories = categories or set()
    self.contributors = contributors or set()
    self.disabled = disabled
+   self.static_output = static_output
@@ -161,13 +161,13 @@ class Block(BlockData):
    @abstractmethod
    def runself, input_data: BlockSchemaInputType) >>> BlockOutput:
@@ -180,7 +180,7 @@ class Block(BlockData):
    "outputSchema": self.output_schema.jsonschema(),
    "description": self.description,
    "categories": {category.dict() for category in self.categories},
-   "contributors": [contributor.dict() for contributor in self.contributors],
+   "contributors": [
+       contributor.model_dump() for contributor in self.contributors
+   ],
-   "staticOutput": self.static_output,
+   "staticOutput": self.static_output,
@@ -184,13 +184,13 @@ class Block(BlockData):
    }
}
```

4

Same intention within under same block
typically follows sequential order

2

```
ValueBlock is a sub-class of Block,
syntactical propagation
"""
This block allows you to provide a constant value as a block, in a stateless manner.
The common use-case is simply pass the 'input' data, it will 'output' the same data.
But this will not retain the state, once it is executed, the output is consumed.
"""
To retain the state, you can feed the 'output' to the 'data' input, so that the data
is retained in the block for the next execution. You can then trigger the block by
feeding the 'input' pin with any data, and the block will produce value of 'data'.
Ex:
<constant_data> <any_trigger>
||   ||
====> 'data'      'input'
||   // ValueBlock
||   ||
===== 'output'
The block output will be static, the output can be consumed multiple times.
"""
class Input(BlockSchema):
    def __init__(self):
        super().__init__(
            id="1f0f065e9-80e8-4358-9d82-8dc91f622ba9",
            description="This block forwards the 'input' pin to 'output' pin."
        )
        "If the 'data' is provided, it will prioritize forwarding 'data' "
        "over 'input'. By connecting the 'output' pin to 'data' pin, "
        "you can retain a constant value for the next executions.",
        "This block output will be static, the output can be consumed many times.",
        categories={BlockCategory.BASIC},
        input_schema=ValueBlock.Input,
        output_schema=ValueBlock.Output,
        "output", "Hello, World!", # No data provided, so trigger is returned
        ("output", "Existing Data"), # Data is provided, so data is returned.
    ),
    static_output=True,
)
}
```

5

Happen in sequential order

6

7

Figure: Edits in the same block are more likely to happen in sequential order

Observation 5

The figure shows four code snippets from different files:

- 1. introduce into utils file**: A screenshot of `torch/_logging/_internal.py` at line 1280. It shows the introduction of a new function `dtrace_structured`. A red arrow points from this line to the second screenshot.
- 2. import into __init__.py**: A screenshot of `torch/_logging/__init__.py` at line 12. It shows the import of `dtrace_structured` and other functions like `get_structured_logging_overhead` and `LazyString`.
- 3. Import into file**: A screenshot of `torch/_experimental/symbolic_shapes.py` at line 60. It shows the import of `dtrace_structured` from the utils file.
- 4. use in file**: A screenshot of `torch/_experimental/symbolic_shapes.py` at line 6000. It shows the usage of `dtrace_structured` within a function `_log_preamble`.

The code snippets illustrate how a new function is first introduced in one file, then imported into another, and finally used within a third file, demonstrating the flow of dependencies between components.

Figure: Inter-file dependencies are often mediated through shared component

Types of Dependencies

Syntactic Arise from structural relationships within the AST.

Example: Edits affecting function definitions, imports, etc.

Semantic Derived from the *behavior* of the code.

Example: Dependencies based on control flow and scope.

Proximity Inherent to the *position* of the code.

Dependency Derivation

Goal: Estimate the dependencies between edits $h_i, h_j \in \mathbf{H}$ using multiple approaches.

- **Heuristic:** Based on observed patterns and logical rules.
- **Static Analysis:** Uses AST, LSP, and code structure.
- **Agentic:** Dependency inference using multiple worker agents.

Heuristic Approach

Key Idea: Establish edit order based on common patterns.

1. **Edit Priority:** Deletions $h_i \in \mathbf{H}_{\text{del}}$ precede additions $h_i \in \mathbf{H}_{\text{add}}$.
2. **Top-Down Order:** Edits within a single file are applied from top to bottom, yielding an order based on line numbers $\text{line}(h_i) < \text{line}(h_j) \implies h_i \rightarrow h_j$.
3. **Import Order:** If h_i involves a module import and h_j references it, we establish $h_i \rightarrow h_j$ based on the dependency.
4. **Lint Errors:** Procedurally sample across a set of error-free edits $E = \{e_1, e_2, \dots, e_n\}$ to iteratively construct a valid program $P = \{p_1, p_2, \dots, p_m\}$
5. **Identical Code:** Merge each equivalence class $[e] = \{e_i \in \mathcal{E} \mid e_i \sim e\}$ into a single representative edit to minimize redundancy and reduce conflicts.

Example: Edit Priority

Rule: Deletions precede additions.

```
# Original Code
def compute(a, b):
    return a + b

# Edit 1: Delete function
- def compute(a, b):
-     return a + b

# Edit 2: Add a new implementation
+ def compute(a, b):
+     return a * b  # Multiplication instead of addition
```

Dependency: The deletion must occur before the addition to prevent conflicts.

Static Analysis Approach

Key Idea: Analyze source code structure to derive dependencies.

1. **AST Generation:** Extract syntax structure.
2. **Edit Position & Identifier Extraction:** Track function/variable modifications.
3. **LSP Dependency Tracing:** Use cross-references for inter-file dependencies.
4. **Code Clone Detection:** Identify similar edits.

Example: Static Analysis

Original Code:

```
def add(a, b):  
    return a + b
```

AST Representation:

```
FunctionDef  
    arguments  
    Return  
        BinOp  
            Name (a)  
            Add  
            Name (b)
```

Edit position, and identifiers extracted

```
Function: add, Line: 2  
Identifier: result, Line: 3, Column: 4  
Identifier: a, Line: 3, Column: 13  
Identifier: b, Line: 3, Column: 17  
Identifier: result, Line: 4, Column: 11
```

LSP: If the function signature is modified,
all references to 'add()' must be updated.

Agentic Approach

Key Idea: Use a language model to infer edit relationships.

- **Context Vector:** Capture relevant usage patterns.

$$\mathbf{c}(h_i) = \{\text{usage}(h_i), \text{dependencies}(h_i)\}$$

- **Dependency Scoring:** Rank edit relationships.
- **Multi-Agent Framework¹:** Assign specialized agents to different tasks.

Models Used: GPT-4, Gemini-Flash, DeepSeek v3, Llama.

¹We will be using the Chain-of-Agents framework

Why Context Selection Matters?

Challenge: Limited context window in LLMs prevents full repository input.

Solution: A systematic process to select the most relevant code snippets and examples.

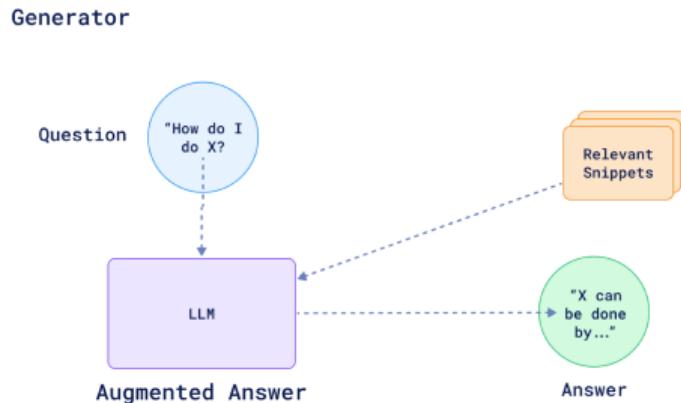


Figure: Retrieval Augmented Generation (RAG) for prompt construction – Input Reduction.

Input Space

Components:

- **Repository (R)**: Contains files $F = \{f_1, f_2, \dots, f_n\}$.
- **External Vector Database (V)**: Stores examples $E = \{e_1, e_2, \dots, e_m\}$.

Goal: Extract only the most relevant snippets for an efficient prompt.

Relevance Selection Algorithm

How do we pick relevant examples?

Define a function $\text{rel}(x, q)$ that evaluates the relevance of snippet x to query q :

$$E_{\text{rel}} = \{e \in E \mid \text{rel}(e, q) > \tau_e\}$$

where τ_e are predefined relevance thresholds.

Methods Tested:

- Mean Squared Error (MSE)
- Mean Absolute Error (MAE)
- Cosine Similarity with vector embeddings

Vector Representation

How do we represent edit relationships?

Each edit pair (h_i, h_j) is mapped to a d -dimensional feature vector:

$$\phi(h_i, h_j) = [f_1(h_i, h_j), f_2(h_i, h_j), \dots, f_d(h_i, h_j)]$$

Why not code embeddings?

- Code embeddings prioritize syntactic similarity.
- We need **semantic and structural relevance**.

Feature Definitions

Each feature function $f_k(h_i, h_j)$ captures a relationship:

- **File Similarity:** Do both edits occur in the same file?

$$f_1(h_i, h_j) = \mathbb{1}[\text{file}(h_i) = \text{file}(h_j)]$$

- **Edit Type Matching:** Do they modify the same type of entity?

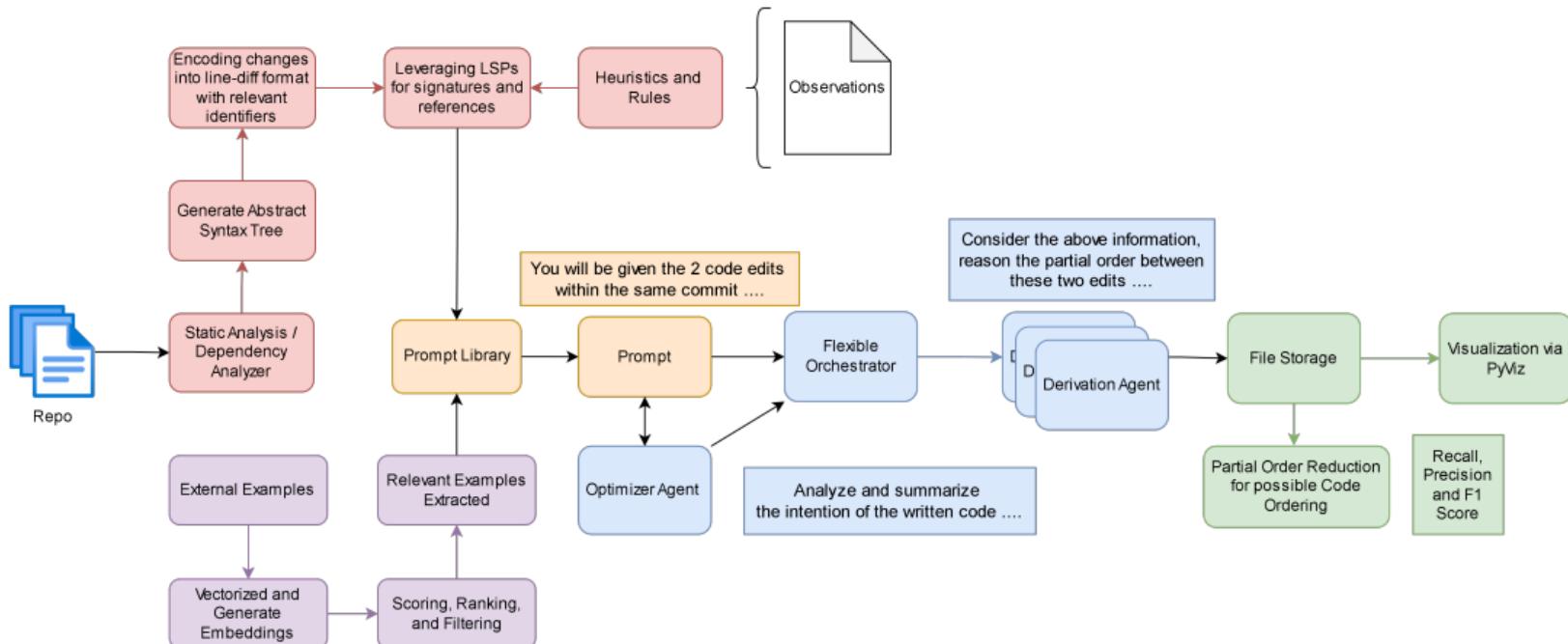
$$f_2(h_i, h_j) = \mathbb{1}[\text{type}(h_i) = \text{type}(h_j)]$$

- **Lexical Similarity:** Measures textual closeness.

$$f_6(h_i, h_j) = \frac{\text{edit-distance}(h_i, h_j)}{\max(|h_i|, |h_j|)}$$

System Overview

How does the system work?



Encoding Code Changes

```
--- src/transformers/commands/run.py
+++ src/transformers/commands/run.py
```

```
@@61,3 61,1 @@ class RunCommand(BaseTransformersCLICommand): # Line Numbers
>>> def register_subcommand(parser: ArgumentParser) # Logical Path
```

```
@staticmethod
def register_subcommand(parser: ArgumentParser):
    run_parser = parser.add_parser("run", help="Run a pipeline through the CLI")
-    run_parser.add_argument(
-        "--task", choices=list(SUPPORTED_TASKS.keys()) + list(TTASK_ALIASES.keys()), help="Task
→ to run"
-    )
+    run_parser.add_argument("--task", choices=get_supported_tasks(), help="Task to run")
```

Annotations:

- **File Path:** src/transformers/commands/run.py (Highlighted in red).
- **Line Numbers:** @@61,3 61,1@@ (Highlighted in blue).
- **Logical Path:** class RunCommand and function register_subcommand() (Highlighted in blue).

Multi-Agent System Overview

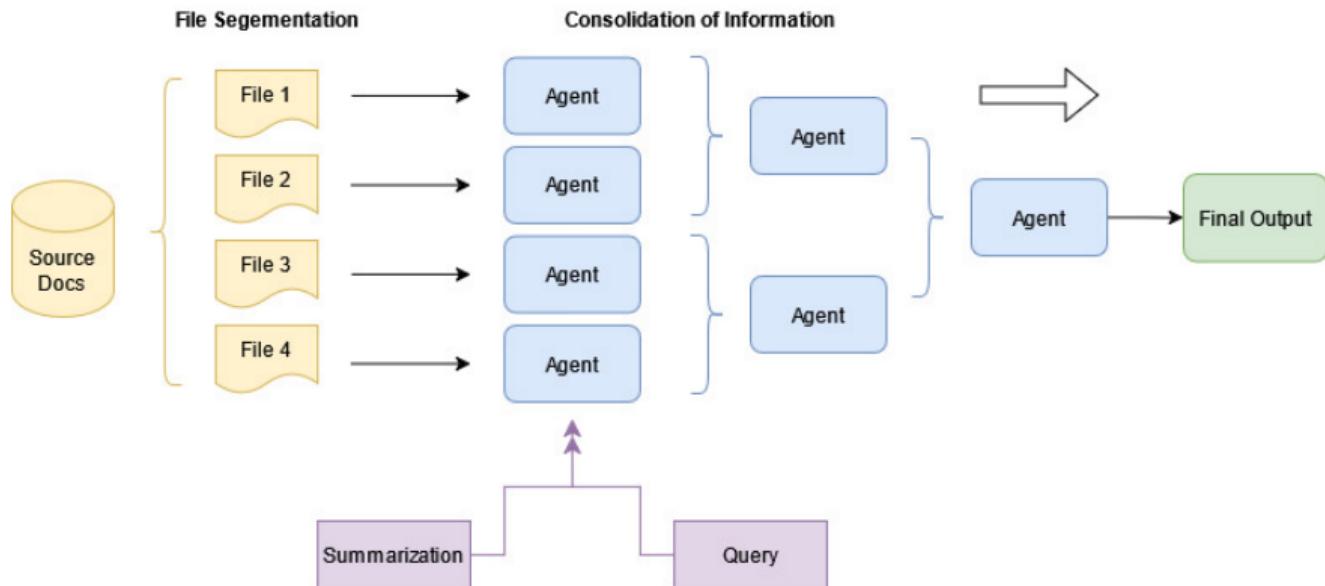


Figure: Agentic Workflow

Consensus Mechanism

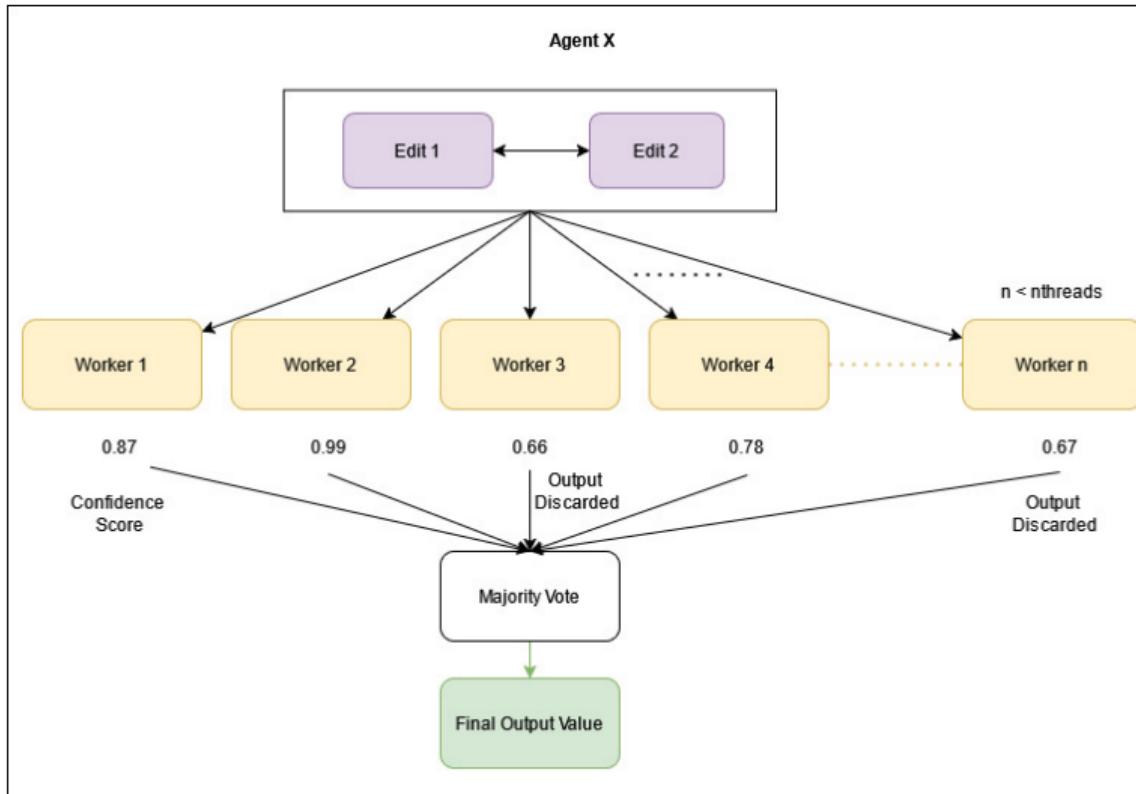
Why is consensus needed?

- Each agent has multiple workers.
- Different workers may suggest conflicting edit orders.
- We need a structured way to merge results.

Methods Used:

- **Voting-Based Resolution** - Majority vote for edit order.
- **Confidence-Weighted Merging** - Prioritize high-confidence relations.
- **Cycle Detection** - Ensure edit sequences remain valid.

Voting Example for Conflict Resolution



Dataset Construction

- 50 commits sourced from popular Python repositories on GitHub.
- Chosen to reflect diverse real-world development tasks.

Attribute	Value
Total Commits	50
Total Edits	364
Total Partial Orders	289
Avg. Edits per Commit	7.28
Avg. Lines Deleted per Edit	1.03
Avg. Lines Inserted per Edit	2.17

Sampling Strategy

To create a representative dataset, each commit C_i is scored by:

- **Edit Type Distribution:** Feature additions, bug fixes, refactoring, etc.
- **File Coverage:** Preference for multi-file commits.
- **Structural Impact:** Significant logic-altering edits prioritized.

Commits selected if their score $S(C_i)$ meets threshold θ :

$$S(C_i) \geq \theta$$

Ensures balanced diversity and relevance.

Reconstructing Edit Sequences

Objective

Reconstruct developer intent via edit sequences:

- Pairwise dependencies form Directed Acyclic Graphs (DAGs).
- Edits ordered respecting identified dependencies (partial orders).

Intuition

glances/core/glances_client_browser.py

```
0 from glances.core.glances_passwordlist import  
GlancesPasswordList as GlancesPassword  
  
1 from glances.core.glances_passwordlist import  
GlancesPassword
```



glances/core/glances_passwordlist.py

```
2 from glances.core.glances_password import  
GlancesPassword  
  
3 class GlancesPassword(object):  
class GlancesPasswordList(GlancesPassword):  
  
4     def __init__(self, config=None, args=None):  
        super(GlancesPassword, self).__init__()
```



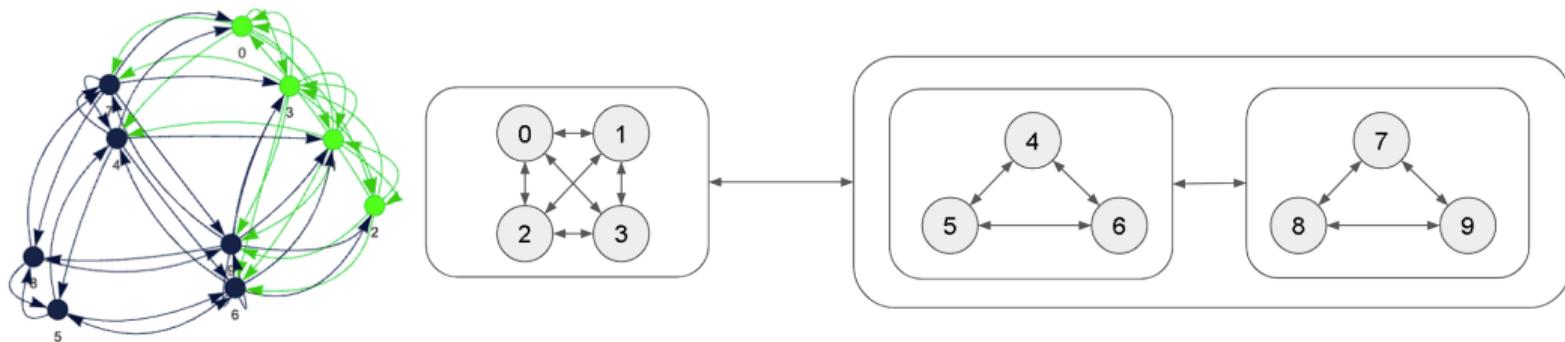
- Suppose we have visited $1 \rightarrow 2 \rightarrow 3$
- Existing mental flow $2 \rightarrow 3$: positional and dependency
- Candidate mental flow: $3 \rightarrow 4$: positional and dependency
- Candidate mental flow: $3 \rightarrow 0$: dependency
-

$$P(3 \rightarrow 4 | 1 \rightarrow 2 \rightarrow 3) = \frac{Sim(2 \rightarrow 3, 3 \rightarrow 4)}{\sum_i Sim(2 \rightarrow 3, 3 \rightarrow i)} > P(3 \rightarrow 0 | 1 \rightarrow 2 \rightarrow 3)$$

Constructing Equivalence Classes

Two edits should be in an equivalence class if they:

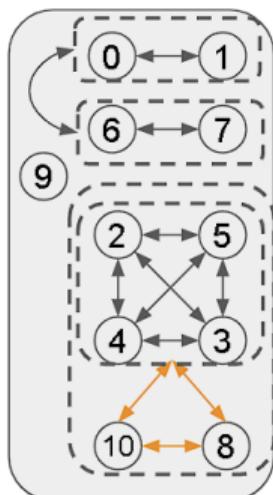
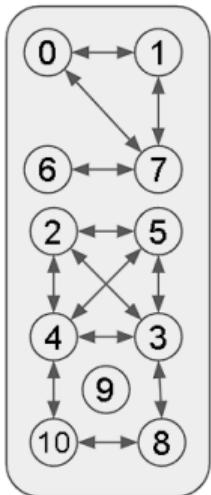
- Are clones of one another in the same/different file
- Share same context
 - Within the same scope and logical path
 - Has bi-directional dependency
 - Import-use / def-use relationship



Hierarchical Graph Statistics

- Average number of levels 1.76
- Average edge reduction rate from converting to hierarchical graph: 16.99%
- Average number of nodes in each equivalence class: 2.74

26 edges



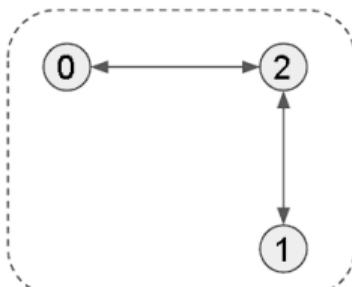
- 2 levels
- 24 edges
- Avg. 3.6 node in cluster

Does it make sense?

- **Input:** Hierarchical edit partial order graph, Ground-truth edit sequence
- **Recall:** How many ground-truth edit orders do not violate the hierarchical graph?
- **Derived restrictions:**
 - The subsequent edit node cannot be an upper-level node unless all nodes at the current level have been visited.
(e.g., $2 \rightarrow 8$ is forbidden when $3/4/5$ is unvisited)
 - The subsequent edit node must be connected to the current node, unless all other nodes at the current level have been visited.
(e.g., $0 \rightarrow 9$ is allowed while $0/1/6/7$ is visited)
- **Results:** 96.41% (25 commits)

Failed Cases

- The partial order graph has false negative edges: missing edge $0 \leftrightarrow 1$



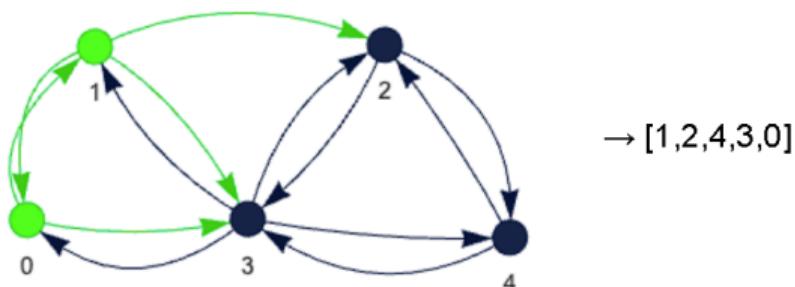
```
41 41     library_paths: Dict[str, List[str]] = {}
42 42     ldpaths: List[str] = []
43 43     ccver: Tuple[int, int]
44 44 +    vcs_rev: str = ''
45 45
46 46     # glfw stuff
47 47     all_headers: List[str] = []
48 48     @@ -52,11 +53,13 @@
49 49
50 50
51 51
52 52
53 53
54 54     def __init__(
55 55         self, cc: List[str] = [], cppflags: List[str] = [], cflags: List[str] = [], ldflags: List[str] = [],
56 56         library_paths: Dict[str, List[str]] = {}, ldpaths: Optional[List[str]] = None, ccver: Tuple[int, int] =
57 57         (0, 0),
58 58         vcs_rev: str = ''
59 59     ):
60 60         self.cc, self.cppflags, self.cflags, self.ldflags, self.library_paths = cc, cppflags, cflags, ldflags,
61 61         self.ldpaths = ldpaths or []
62 62         self.ccver = ccver
63 63         self.vcs_rev = vcs_rev
64 64
65 65     def copy(self) -> 'Env':
```

Constrained Ordering Algorithm

Use Depth-First Traversal with Derived Restrictions. Define $\Pi(S)$ to be an ordered set containing the equivalence classes in the correct order. Then, the constrained group ordering is:

$$\mathcal{P} = \bigcup_{\pi \in \Pi(S)} \text{Flatten}(\pi)$$

Output: A Sequence of Edit Order



Evaluation Metrics

Given predicted sequence P and ground truth \mathbf{C} :

- **Perfect Match:** Exact matching sequence.
- **Minimum Swaps:** Minimal swaps needed to reach valid ground-truth order.
- **Edit and Kendall Tau Distances:** Measures sequence alignment.

Evaluation metrics reported:

- Precision, Recall, F1 Score
- Swap counts and distances

Results Summary

Metric	Score
Precision	0.84
Recall	0.92
F1 Score	0.87
Avg Swaps	2.2 (Median: 2, Max: 6)
Avg Edit Distance	6
Avg Kendall Tau Distance	10
Perfect Matches	82%

Table: Performance Metrics

False Positives

- Tend to treat similar keywords as dependency

```
430 431
431 -     return Env(cc, cppflags, cflags, ldflags, library_paths, ccver=ccver, ldpaths=ldpaths)
432 +     return Env(cc, cppflags, cflags, ldflags, library_paths, ccver=ccver, ldpaths=ldpaths, vcs_rev=vcs_rev)
432 433
910 913         cmd = [go, 'build', '-v']
911 -     ld_flags = [f"-X 'kitty.VCSRevision={get_vcs_rev_define()}'"]
914 +     vcs_rev = args.vcs_rev or get_vcs_rev()
915 +     ld_flags = [f"-X 'kitty.VCSRevision={vcs_rev}'"]
912 916     if for_freeze:
```

False Positives

- Contain logic jump
 - import json → to_json → json*

1	1	from functools import partial
2	2	+ import json
2	3	
832	845	table_meta["select_data_query"],
833	-	table_meta["training_options"],
846	+	to_json(table_meta["training_options"]),
834	847	table_meta["label"],

False Negatives

- Existing methods: only when 2 edit hunks within same file, or share dependency, will be sent to lsm.
- Here 2 edit hunks share a dependency that cannot be identified by LSP, hence no partial order detected.

```
glances/core/glances_main.py
@@ -197,10 +197,14 @@ def parse_arg(self):
197 197         # By default Help is hidden
198 198         args.help_tag = False
199 199
200 +     # Display Rx and Tx, not the sum for the network
201 +     args.network_sum = False
202 +     args.network_cumul = False
203 +
200 204     # !!! Change global variables regarding to user args
```

```
glances/outputs/glances_curses.py
@@ -217,11 +217,11 @@ def __catchKey(self):
217 217         # 's' > Show/hide sensors stats (Linux-only)
218 218         self.args.disable_sensors = not self.args.disable_sensors
219 219         elif self.pressedkey == ord('t'):
220 -             # 't' > View network traffic as combination
221 -             self.network_stats_combined = not self.network_stats_combined
220 +             # 't' > View network traffic as sum Rx+Tx
221 +             self.args.network_sum = not self.args.network_sum
222 222         elif self.pressedkey == ord('u'):
```

Limitations of the Approach

Despite promising results, several limitations exist:

1. Static and Heuristic-Based Analysis

- Does not fully capture implicit dependencies from runtime or dynamic behaviors.

2. Metric Limitations

- Potential underestimation of generalization due to incomplete ground-truth coverage.

3. Dataset Coverage:

- May not generalize fully across different programming languages or paradigms.

Summary of Contributions

- Formalized a framework for reconstructing developers' code edit sequences.
- Developed a multi-agent orchestration system to infer logical ordering of edits.
- Demonstrated the practicality and effectiveness using real-world data from GitHub.

Thank You!

Q&A