# Generating Chemistry Simulations using Large Language Models.

Master project by Pieter Jacobs
Internal supervisor: Matthia Sabatelli
External supervisor: Robert Pollice

# Introduction

## Situation

- Large Language Models (LLMs) have demonstrated significant advancements in code synthesis, with specialized models such as OpenAI Codex becoming increasingly prevalent.

- Aside from automating the task of programming, LLMs enable interactivity. Users can ask questions about previously written code and provide feedback to the model to obtain better responses.

- Due to this, they have been widely used to assist in writing code.

# Introduction

## Trigger

- Researchers across various domains can enhance their productivity by leveraging LLMs for programming tasks, allowing them to allocate more time and energy to their core research activities.

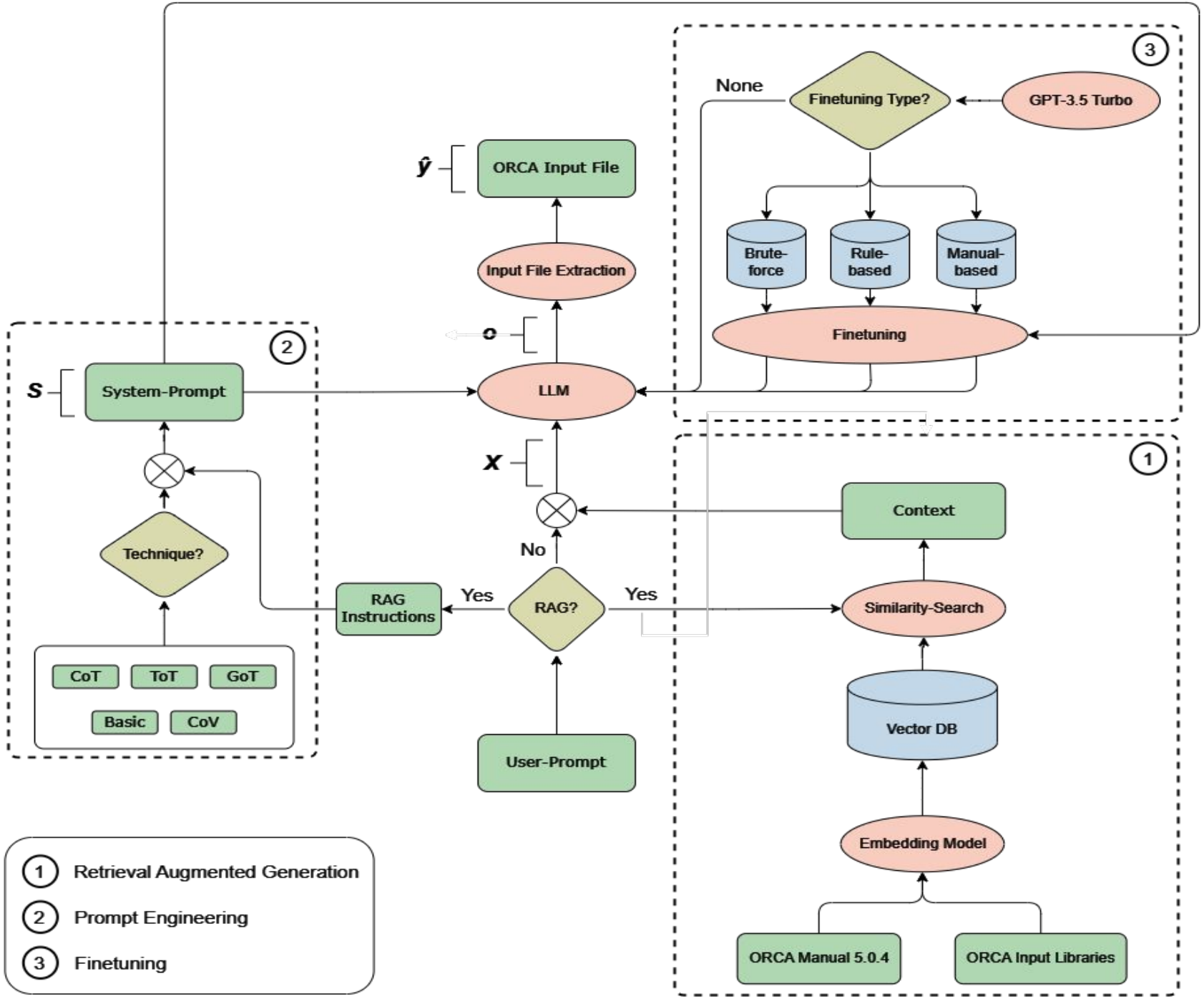
- They often need to learn and use Domain Specific Languages (DSLs) to perform their research. For example, in the field of quantum chemistry, DSLs are used to write code for generating the necessary simulations.

- However, LLMs currently perform well only for large general-purpose programming languages and not for Domain Specific Languages (DSLs) due to limited exposure during training.

# Introduction

## Question

*How do prompt-engineering techniques, retrieval-augmented generation, and finetuning impact the ability of a large language model to generate accurate and functional input files for ORCA chemistry simulations?*


3
None
Finetuning Type?
GPT-3.5 Turbo
Brute-force
Rule-based
Manual-based
Finetuning
ŷ
ORCA Input File
Input File Extraction
c
2
S
System-Prompt
LLM
X
1
Context
Technique?
No
RAG Instructions
Yes
RAG?
Yes
Similarity-Search
CoT
ToT
GoT
Basic
CoV
Vector DB
User-Prompt
Embedding Model
ORCA Manual 5.0.4
ORCA Input Libraries
1 Retrieval Augmented Generation
2 Prompt Engineering
3 Finetuning

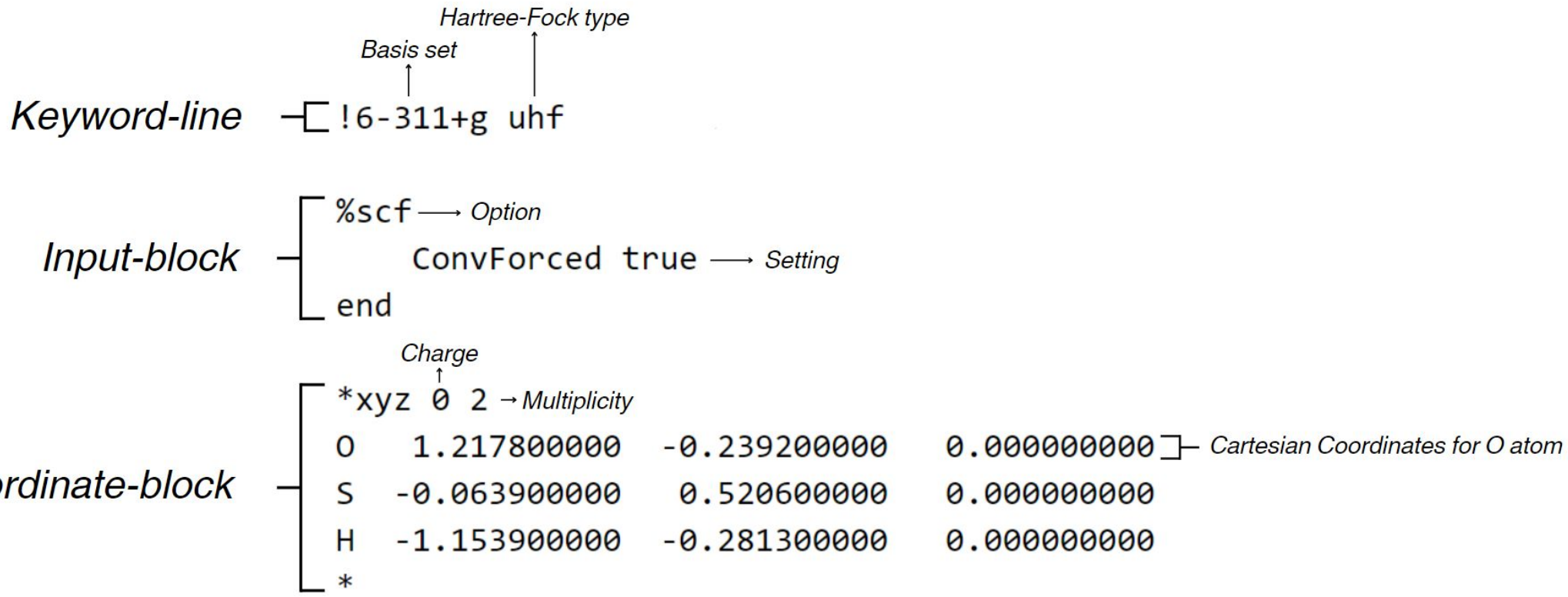
# Methods: Finetuning

- Finetuning is a form of transfer learning where the LLM is further trained on a curated dataset for supervised learning.

- In our study, we use finetuning to expose the model to more ORCA input files. In our dataset
  - ***X*** = Prompts describing a desired quantum chemistry simulation
  - ***y*** = Corresponding ORCA input files

- However, our initial problem becomes apparent: there is little data available for DSLs! We were only able to extract ~600 input files.

# ORCA Input File



# Methods: Input File Generation

## Brute-force

1. Randomly combine keywords for the keyword line.
2. Randomly choose an option for an input-block.
3. Add random settings to this block.
4. If the file runs successfully, save it.

## Manual-based

1. Extract all keyword lines and (full) input blocks from the ORCA manual.
2. Combine them randomly to create an input file.
3. If the file runs successfully, we save it.

```
! BP86 def2-SVP def2/J
%scf
maxiter 100
end
%tddft
nroots 10
end
```

# Methods: Prompt Creation

## Rule-based

1. We choose a input file category, like a single point Hartree Fock calculation.
2. Based on this, we use predefined rules to decide what keywords and input blocks get added.
3. If the file runs successfully (likely!), we save it.

```
! BP86 def2-SVP def2/J
%scf
maxiter 100
end
%tddft
nroots 10
end
```

# Methods: Prompt Creation

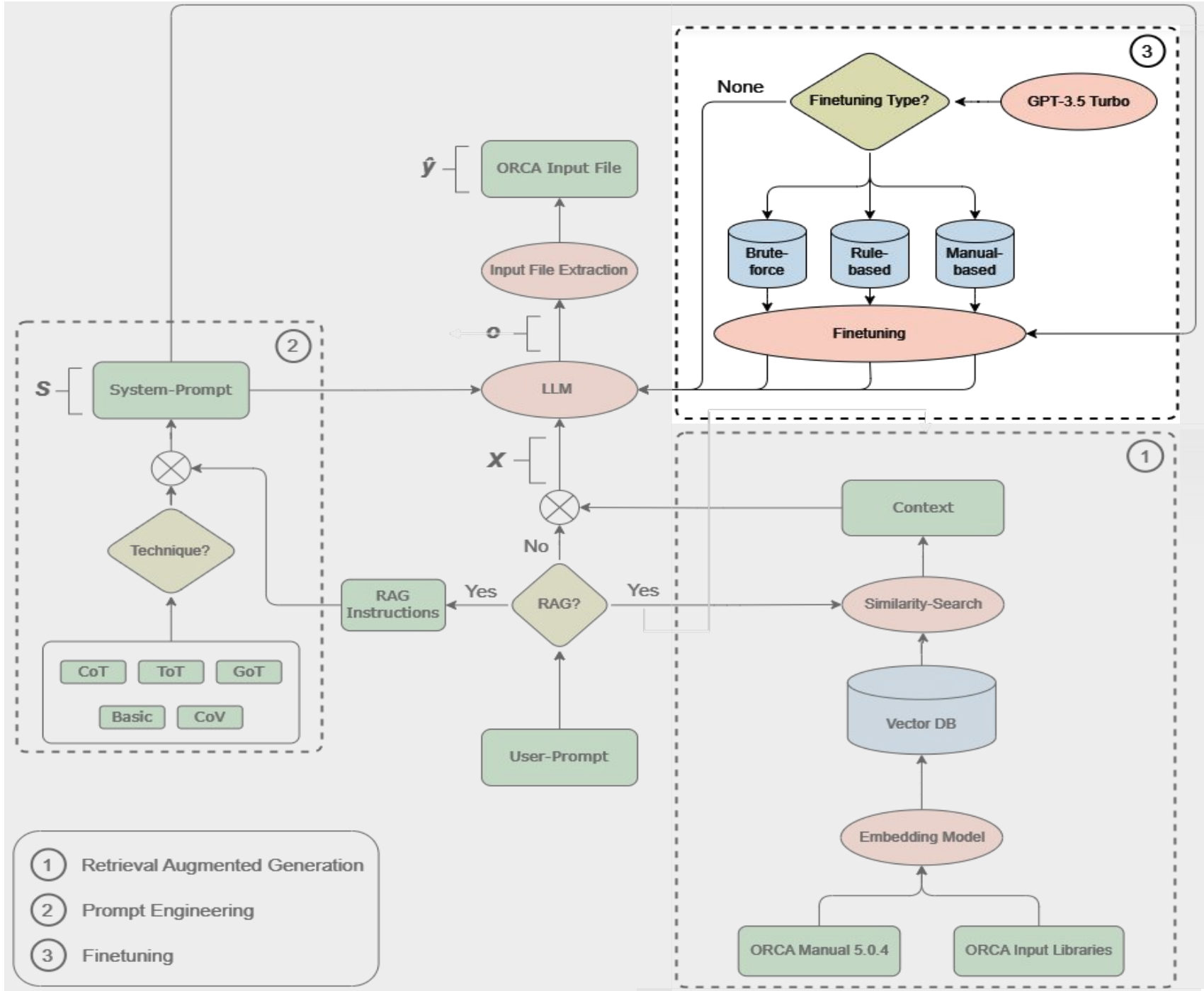
1. Extract all ORCA documentation, to get the description of keywords and options.
2. Given an input file, we map each word of the input file to a description, with separators that indicate what part of the input file it belongs.
3. We use this crude description and feed it to an LLM, to make it a more comprehensible and user-like prompt.

# Methods: Prompt Creation Example

*Perform a Fock-Space CCPNO-CCSD calculation with the Pople 6-31G basis set and its modifications on the thioformaldehyde molecule. Use tight SCF convergence criteria. Utilize spin unrestricted SCF and produce UHF natural orbitals. Additionally, control single reference correlation methods by setting nroots to 9, locrandom to 0, and useqros to true.*

# Methods: Finetuning

- The test and validation set consist of real-world data.
- There are three training datasets, based on the described generation methodologies.
- Finetuning thus ends up providing us, given a chosen LLM architecture, with three different models.


3
None
Finetuning Type?
GPT-3.5 Turbo
Brute-force
Rule-based
Manual-based
Finetuning
ŷ
ORCA Input File
Input File Extraction
o
2
S
System-Prompt
LLM
X
1
Context
Technique?
No
RAG Instructions
Yes
RAG?
Yes
Similarity-Search
CoT
ToT
GoT
Basic
CoV
Vector DB
User-Prompt
Embedding Model
ORCA Manual 5.0.4
ORCA Input Libraries
1 Retrieval Augmented Generation
2 Prompt Engineering
3 Finetuning

# Methods: Prompt Engineering

- Leveraging the LLM's pre-trained understanding of language lies in crafting effective ways to prompt it.

- Small changes to prompts, like in phrasing or the amount of examples included in it, can provide significant changes in its output.

- *“There are many tasks at which our models may not initially appear to perform well, but results can be improved with the right prompts”* - OpenAI

- Therefore, we employ various prompt engineering methodologies to try and improve ORCA input file synthesis.

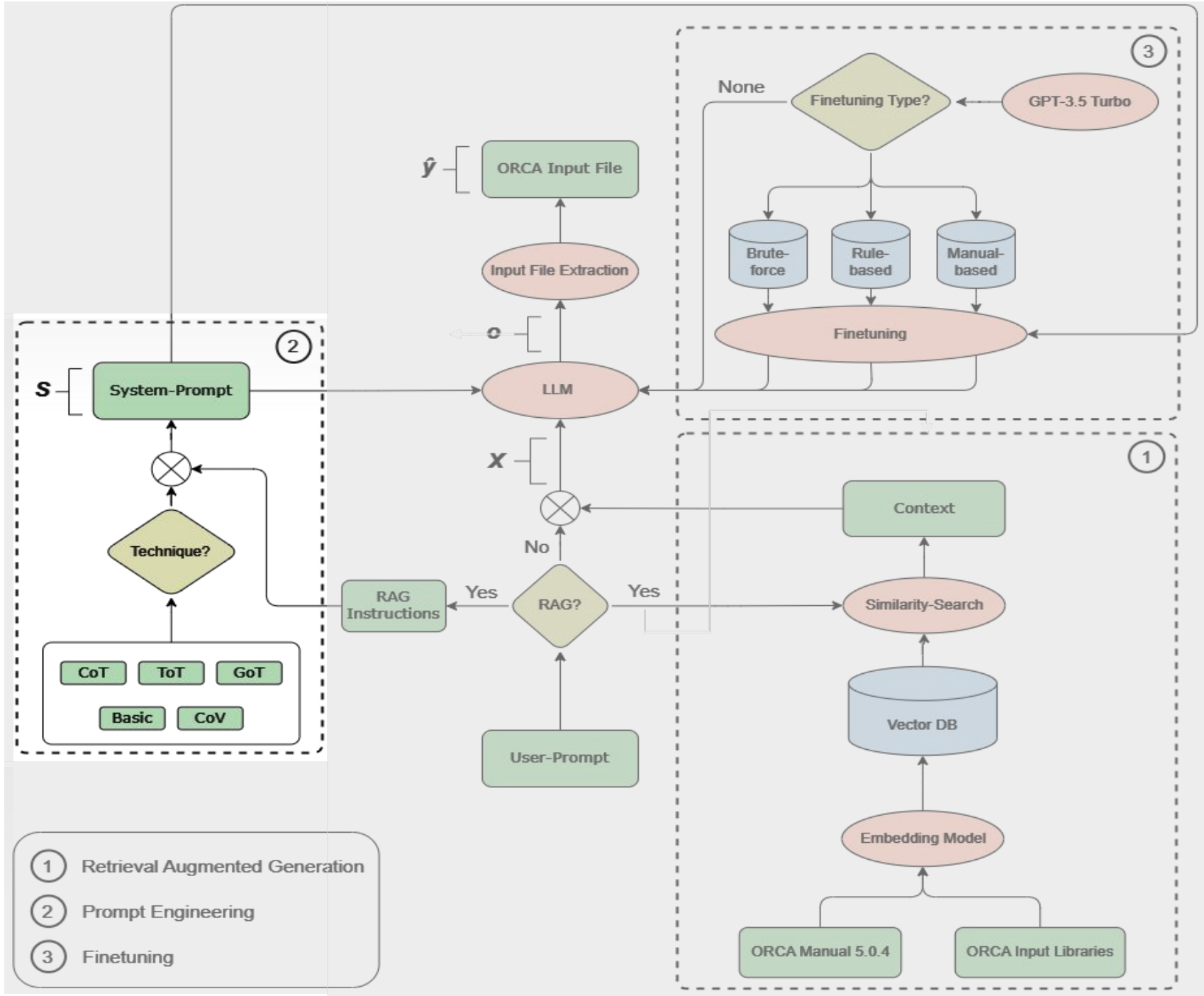
# Methods: Prompt Engineering Example

## Chain of Thought

***In making your input file, use step-by-step reasoning and write your reasoning down:***
***Step 1: What is the user trying to do with the input file? How is he/she trying to achieve this?***
***Step 2: Identify what ORCA keywords should be used? like for instance the basis set and a possible density functional.***
***Step 3: What are the advanced settings should be used, if there are any. What are the ORCA input block settings needed to get to these settings?***
***Step 4: What is the selfies of the molecule the user wants to use?***
***Step 5: What is the final input file?***
***Generate the input file by following these steps and ensure each step is explained.***

## Chain of Verification

***Start by creating an initial version of the input file based on the user's description. Then verify it with these steps:***
***Step 1: Ensure the simulation type (e.g., geometry optimization, frequency calculation) matches the user's goal.***
***Step 2: Check that the selected methods, density functionals, and basis sets are appropriate for the simulation.***
***Step 3: Confirm that any advanced settings and input block options are correctly specified and relevant.***
***Step 4: Conduct a comprehensive final review of the entire input file to ensure all components are coherent and correctly formatted.***
***Using this approach, generate the ORCA input file and verify each step to ensure the highest accuracy and alignment with the user's goals.***

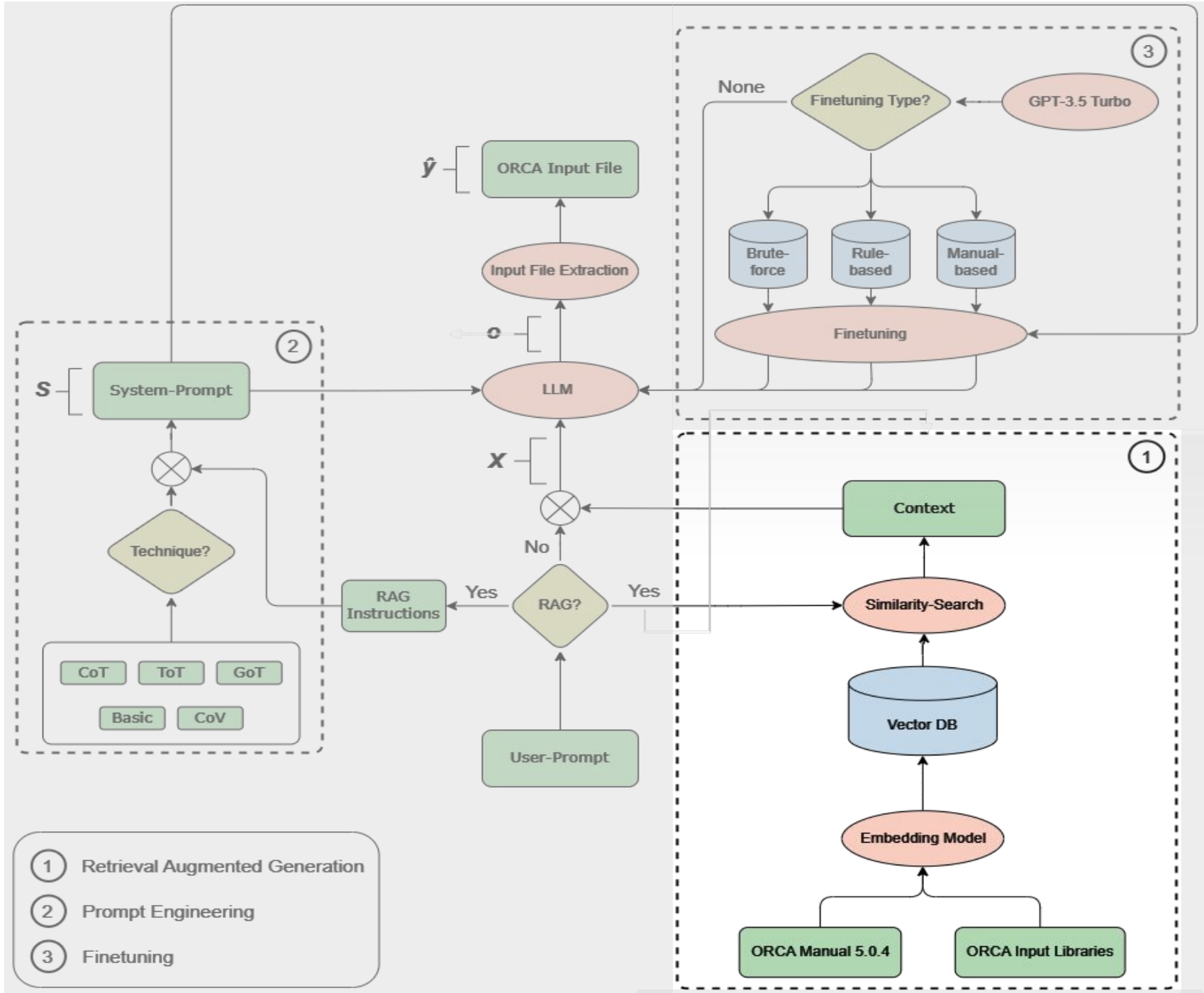

3
None
Finetuning Type?
GPT-3.5 Turbo
ŷ
ORCA Input File
Input File Extraction
Brute-force
Rule-based
Manual-based
Finetuning
o
2
S
System-Prompt
LLM
1
X
Context
Technique?
No
RAG Instructions
Yes
RAG?
Yes
Similarity-Search
CoT
ToT
GoT
Basic
CoV
Vector DB
User-Prompt
Embedding Model
ORCA Manual 5.0.4
ORCA Input Libraries
1 Retrieval Augmented Generation
2 Prompt Engineering
3 Finetuning

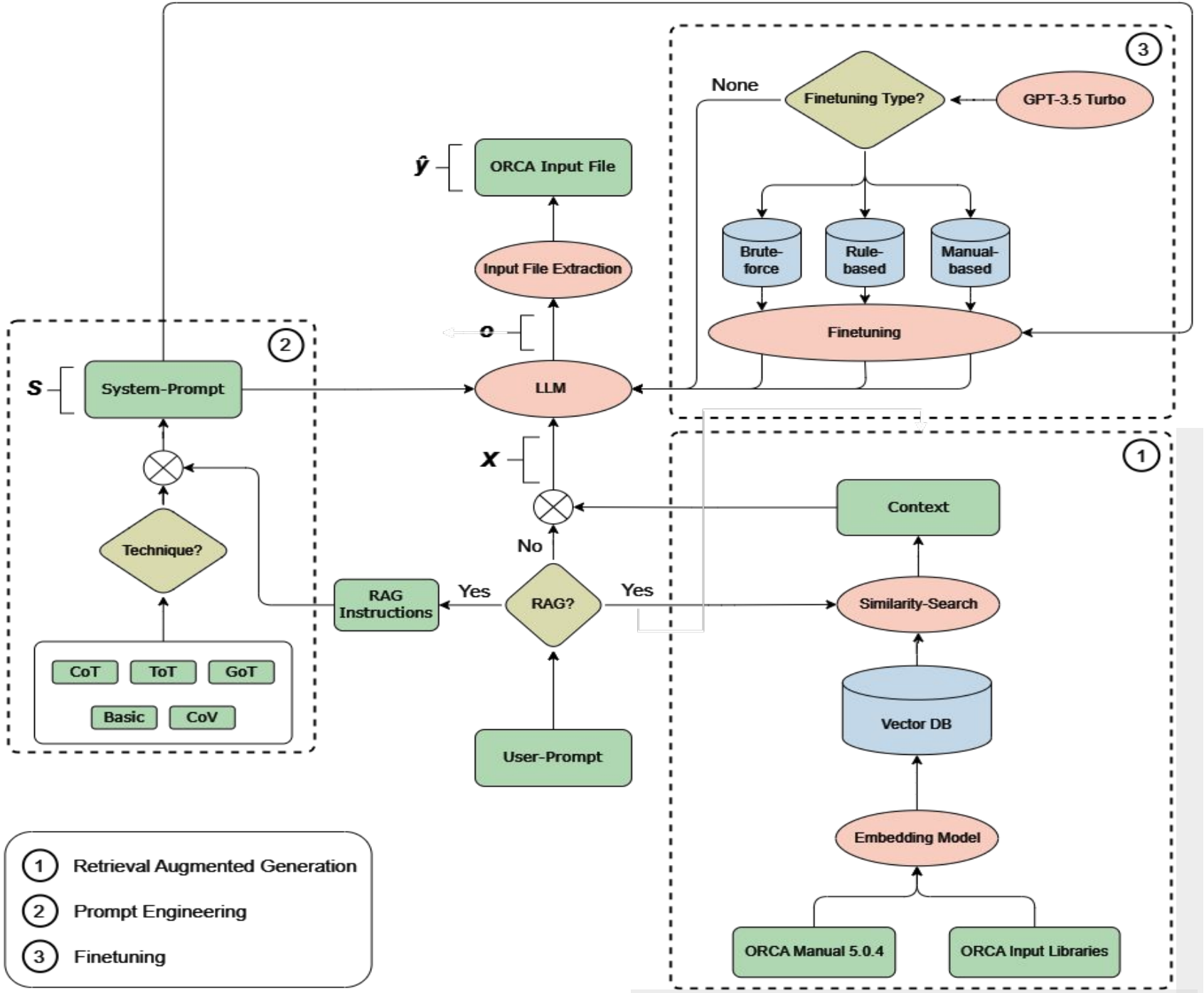
# Methods: Retrieval Augmented Generation

- In context of LLMs, Retrieval-Augmented Generation (RAG) involves fetching relevant text from documents or other data to enhance the LLM's responses by injecting the prompt with this information.

- We utilize the ORCA manual and a scraped website detailing various ORCA input files as a database, generating embeddings from all individual pages of these documents.

- Given a prompt, we use similarity search on these embeddings and retrieve the most similar text out of our database and add it to the prompt.


3
None
Finetuning Type?
GPT-3.5 Turbo
Brute-force
Rule-based
Manual-based
Finetuning
ŷ
ORCA Input File
Input File Extraction
o
2
S
System-Prompt
LLM
X
1
Context
Technique?
No
RAG Instructions
Yes
RAG?
Yes
Similarity-Search
CoT
ToT
GoT
Basic
CoV
Vector DB
User-Prompt
Embedding Model
ORCA Manual 5.0.4
ORCA Input Libraries
1 Retrieval Augmented Generation
2 Prompt Engineering
3 Finetuning


3
None
Finetuning Type?
GPT-3.5 Turbo
Brute-force
Rule-based
Manual-based
Finetuning
ŷ
ORCA Input File
Input File Extraction
c
2
S
System-Prompt
LLM
X
1
Context
Technique?
No
RAG Instructions
Yes
RAG?
Yes
Similarity-Search
CoT
ToT
GoT
Basic
CoV
Vector DB
User-Prompt
Embedding Model
ORCA Manual 5.0.4
ORCA Input Libraries
1 Retrieval Augmented Generation
2 Prompt Engineering
3 Finetuning

# Experiments

- Test GPT-3.5 Turbo with all possible combinations of the three methods. For a given prompt engineering tactic, this results in 16 different models to compare.

- We evaluate these models on the code they generated using BLEU- and F1-score, and whether the generated input files can be ran successfully.

- This should provide insight into how the different methodologies help an LLM generate code for a DSL like ORCA.

# Results: GPT-3.5 Turbo

| COT | RAG | Runnable (%) | $F1_{total}$ | $F1_{avg}$ | BLEU Score |
|---|---|---|---|---|---|
| ✗ | ✗ | 3.401 | 0.217 | 0.182 | 10.121 |
| ✗ | ✓ | 11.905 | **0.242** | **0.194** | 8.467 |
| ✓ | ✗ | 3.741 | 0.229 | 0.193 | **10.288** |
| ✓ | ✓ | **14.626** | 0.229 | 0.188 | 8.026 |

# Results: Finetuned Models

| COT | RAG | Finetuning Dataset | Runnable (%) | $F1_{total}$ | $F1_{avg}$ | BLEU Score |
|---|---|---|---|---|---|---|
| ✗ | ✗ | Rule-based | 11.225 | 0.333 | 0.237 | 8.962 |
| ✗ | ✓ | Rule-based | 10.884 | 0.282 | 0.206 | 7.311 |
| ✓ | ✗ | Rule-based | 19.048 | 0.361 | 0.254 | 9.418 |
| ✓ | ✓ | Rule-based | 15.646 | 0.298 | 0.205 | 7.504 |
| ✗ | ✗ | Manual-based | 10.544 | 0.390 | 0.261 | 12.293 |
| ✗ | ✓ | Manual-based | 14.626 | 0.311 | 0.228 | 7.887 |
| ✓ | ✗ | **Manual-based** | 21.769 | **0.426** | **0.286** | **12.916** |
| ✓ | ✓ | Manual-based | 15.986 | 0.330 | 0.223 | 8.300 |
| ✗ | ✗ | Brute-force | 15.306 | 0.382 | 0.267 | 10.234 |
| ✗ | ✓ | Brute-force | 25.122 | 0.221 | 0.177 | 10.963 |
| ✓ | ✗ | Brute-force | 17.006 | 0.384 | 0.264 | 17.114 |
| ✓ | ✓ | Brute-force | **26.191** | 0.282 | 0.205 | 7.312 |