Red Hat Enterprise Linux AI 1.2

Creating a custom LLM using RHEL AI

Creating files for customizing LLMs and running the end-to-end workflow

Last Updated: 2024-11-05

Red Hat Enterprise Linux AI 1.2 Creating a custom LLM using RHEL AI

Creating files for customizing LLMs and running the end-to-end workflow

Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document provides instructions on how users can create a custom LLM through SDG, training and evaluation

Table of Contents

CHAPTER 1. CUSTOMIZING YOUR TAXONOMY TREE

You can modify the taxonomy tree with knowledge data in your RHEL AI environment to create your own custom Granite Large Language Model (LLM). On RHEL AI, knowledge data sets are formatted in YAML. This YAML configuration is called a **qna.yaml** file, where "qna" stands for question and answer.

The following documentation sections describe how to create skill and knowledge sets for your taxonomy.

- Adding knowledge to your taxonomy tree

1.1. SKILLS AND KNOWLEDGE OVERVIEW

Skill and knowledge are the types of data that you can add to the taxonomy tree. You can then use these types to create a custom LLM model fine-tuned with your own data.

1.1.1. Knowledge

Knowledge for an AI model consists of data and facts. When creating knowledge sets for a model, you are providing it with additional data and information so the model can answer questions more accurately. Where skills are the information that trains an AI model on how to do something, knowledge is based on the model's ability to answer questions that involve facts, data, or references. For example, you can create a data set that includes a product's documentation and the model can learn the information provided in that documentation.

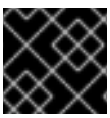
1.1.2. Skills

IMPORTANT

RHEL AI version 1.2 currently does not support customizing skills.

A skill is a capability domain that intends to train the AI model on submitted information. When you make a skill, you are teaching the model how to do a task. Skills on RHEL AI are broken into categories:

- Composition skill: Compositional skills allow AI models to perform specific tasks or functions. There are two types of composition skills:
  - Freeform compositional skills: These are performative skills that do not require additional content or information to function.
  - Grounded compositional skills: These are performative skills that require additional context. For example, you can teach the model to read a table, where the additional context is an example of the table layout.
- Foundation skills: Foundational skills are skills that involve math, reasoning, and coding.



IMPORTANT

Ensure your server is not running before you start customizing your Granite starter model.

Additional Resources

- Sample knowledge specifications

1.2. ADDING KNOWLEDGE TO YOUR TAXONOMY TREE

For RHEL AI, your knowledge data is hosted in a Git repository and in the markdown format. Skill and knowledge datasets are created using the YAML format, called a **qna.yaml** file. Each **qna.yaml** file for knowledge contains a set of key-value entries with the following keys:

- **version** - The version of the **qna.yaml** file, this is the format of the file used for SDG. The currently supported value for this parameter is 3.
- **created_by** - Your Git username or name of contributor.
- **domain** - The category of the knowledge.
- **seed_examples** - A collection of key and value entries.
  - **context** - A chunk of information from the knowledge document. The **context** field must be in markdown format. Each **qna.yaml** needs five context blocks and has a maximum token count of 500 tokens.
  - **questions_and_answers** - The parameter that holds your questions and answers.
    - **question** - Specify a question for the model. Each **qna.yaml** file needs at least three question and answer pairs per context chunk with a maximum token count of 250 tokens.
    - **answer**- Specify an answer for the model. Each **qna.yaml** file needs at least three question and answer pairs per context chunk with a maximum token count of 250 tokens.
- **document_outline** - Describe an overview of the document you're submitting. It is recomended to add details that are referenced in the context field in the **document_outline** parameter.
- **document** - The source of your knowledge data. This parameter is only required for knowledge **qna.yaml** files.
  - **repo** - The URL to your repository that holds your knowledge markdown files.
  - **commit** - The SHA of the commit in your repository with your knowledge markdown files.
  - **patterns** - Specifies the markdown file(s) in your repository.

IMPORTANT

For Red Hat Enterprise Linux AI version 1.2 general availability, your data must be in the markdown format and hosted in a Git repository.

Hosting your knowledge markdown data on your local system is currently not supported on RHEL AI version 1.2.

1.2.1. Creating a knowledge markdown file

On Red Hat Enterprise Linux AI version 1.2, you must host your knowledge documentation and data in a git repository and in markdown format. You can use the standard git workflow to create and upload files to your repository. There are various open source markdown conversion tools you can use, including:

- Pandoc: An open source conversion tool.

- Visual Studio Code with All in one extention: You can open your document in Visual Studio Code, and use the Markdown All in One extensions to convert to Markdown.
- IBM Deepsearch/Docling: Bundles PDF document conversion to JSON and markdown in a self-contained package.

Procedure

1. Select your preferred git hosting platform. You can use any platform on RHEL AI as long as it's compatible with git.
2. Convert your documents into the **.md** markdown format. You can use any markdown conversion software you want for your knowledge data.
   The following list includes guidelines for knowledge markdown files:
   - All documents must be text, images are not currently supported.
   - Remove any footnotes from your documents.
   - Tables must be in markdown format.
   - Charts and graphs are currently not supported.
3. Make a note of your file name and commit hash. This value is used in your **qna.yaml** file.
4. Create and upload a **md** file into your git repository.

1.2.1.1. Sample knowledge markdown specifications

On Red Hat Enterprise Linux AI version 1.2, your knowledge must be in the markdown format.

Example markdown

```
# Phoenix (constellation)

**Phoenix** is a minor constellation in the southern sky. Named after the mythical phoenix,
it was first depicted on a celestial atlas by Johann Bayer in his 1603 *Uranometria*. The
French explorer and astronomer Nicolas Louis de Lacaille charted the brighter stars and gave
their Bayer designations in 1756. The constellation stretches from roughly –39 degrees to –57
degrees declination, and from 23.5h to 2.5h of right ascension. The constellations Phoenix,
Grus , Pavo and Tucana, are known as the Southern Birds.

The brightest star, Alpha Phoenicis, is named Ankaa, an Arabic word meaning 'the Phoenix'.
It is an orange giant of apparent magnitude 2.4. Next is Beta Phoenicis, actually a
binary system composed of two yellow giants with a combined apparent magnitude of 3.3. Nu
Phoenicis
has a dust disk, while the constellation has ten star systems with known planets and the recently
discovered galaxy clusters El Gordo and the Phoenix Cluster—located 7.2 and 5.7 billion light years
away respectively, two of the largest objects in the visible universe. Phoenix is the radiant of
two annual meteor showers: the Phoenicids in December, and the July Phoenicids.

## History

Phoenix was the largest of the 12 constellations established by Petrus Plancius from the
observations
of Pieter Dirkszoon Keyser and Frederick de Houtman. It first appeared on a 35-cm diameter celestial
```

globe published in 1597 (or 1598) in Amsterdam by Plancius with Jodocus Hondius. The first depiction of this constellation in a celestial atlas was in Johann Bayer's*Uranometria* of 1603. De Houtman
included it in his southern star catalog the same year under the Dutch name *Den voghel Fenicx*, "The Bird Phoenix", symbolizing the phoenix of classical mythology. One name of the brightest star Alpha
Phoenicis—Ankaa—is derived from the Arabic: العنقاء, romanized: al-‘anqā’, lit. 'the phoenix', and was coined sometime after 1800 in relation to the constellation.

Celestial historian Richard Allen noted that unlike the other constellations introduced by Plancius and La
Caille, Phoenix has actual precedent in ancient astronomy, as the Arabs saw this formation as representing
young ostriches, *Al Ri'āl*, or as a griffin or eagle. In addition, the same group of stars was sometimes
imagined by the Arabs as a boat, *Al Zaurak*, on the nearby river Eridanus. He observed, "the introduction of a Phoenix into modern astronomy was, in a measure, by adoption rather than by invention."

The Chinese incorporated Phoenix's brightest star, Ankaa (Alpha Phoenicis), and stars from the adjacent
constellation Sculptor to depict *Bakui*, a net for catching birds. Phoenix and the neighboring constellation
of Grus together were seen by Julius Schiller as portraying Aaron the High Priest. These two constellations,
along with nearby Pavo and Tucana, are called the Southern Birds.

Characteristics

Phoenix is a small constellation bordered by Fornax and Sculptor to the north, Grus to the west, Tucana to the
south, touching on the corner of Hydrus to the south, and Eridanus to the east and southeast. The bright star
Achernar is nearby. The three-letter abbreviation for the constellation, as adopted by the International Astronomical
Union in 1922, is "Phe". The official constellation boundaries, as set by Belgian astronomer Eugène Delporte in
1930, are defined by a polygon of 10 segments. In the equatorial coordinate system, the right ascension
coordinates of these borders lie between 23^h 26.5^m and 02^h 25.0^m, while
the declination coordinates are between −39.31° and −57.84°. This means it remains below the horizon to anyone
living north of the 40th parallel in the Northern Hemisphere, and remains low in the sky for anyone living north
of the equator. It is most visible from locations such as Australia and South Africa during late Southern Hemisphere spring.
Most of the constellation lies within, and can be located by, forming a triangle of the bright stars Achernar,
Fomalhaut and Beta Ceti—Ankaa lies roughly in the centre of this.

1.2.2. Creating a knowledge YAML file

You can customize your taxonomy tree so a model can learn domain-specific information. Knowledge

contributions use the **qna.yaml** file to learn how to read the document that you want to teach the model. The following process displays how to create a **qna.yaml** file that teaches your LLM about the knowledge in your markdown file using the RHEL AI toolset.

Prerequisites

- You installed RHEL AI with the bootable container image.
- You installed the git CLI.
- You initialized InstructLab and can use the **ilab** CLI.
- You have root user access on your machine.

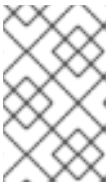
Procedure

1. Since you are hosting your knowledge files in a git repository, you need to checkout a working branch when updating your taxonomy.
2. Navigate to the taxonomy folder. RHEL AI includes a ready made taxonomy tree to interact with.
3. Navigate to the knowledge folder in the taxonomy directory.
4. Add directories and folders in the taxonomy tree where you want to add your knowledge **qna.yaml** file.

   Example file path in the taxonomy tree

   ```
   taxonomy/knowledge/technical_documents/product_customer_cases/qna.yaml
   ```

5. Using your desired text editor, create the **qna.yaml** file. Your YAML must have the **qna.yaml** title.

   NOTE

   For SDG to run properly, you must include at least five **context** chunks and three question and answer seeds per context value in the **questions_and_answers** parameter.

6. Add the necessary keys to the **qna.yaml** file and save your changes. For more information about formatting your **qna.yaml** file, see "Sample knowledge YAML specifications".

Verification

- To verify that your knowledge **qna.yaml** file is in the proper format, you can run the following command:

  ```
  $ ilab taxonomy diff
  ```

  The CLI displays if your taxonomy tree and **qna.yaml** file is valid and properly formatted. The CLI also shows you where you can fix any errors you encounter.

  Example output of valid taxonomy tree and qna.yaml file

```
knowledge/technical_documents/product_customer_cases/qna.yaml
Taxonomy in /taxonomy/ is valid :)
```

Example output of invalid taxonomy tree and qna.yaml file with errors

```
9:15 error syntax error: mapping values are not allowed here (syntax)
Reading taxonomy failed with the following error: 1 taxonomy with errors! Exiting.
```

1.2.2.1. Sample knowledge YAML specifications

Knowledge contributions use the **qna.yaml** file to learn how to read the document that you want to teach the model. On RHEL AI, the synthetic data generation (SDG) process uses your **qna.yaml** seed examples to create a large quantity of artificial data. This process makes it so the model has more data to learn from rather than exclusively relying on provided samples. The order of the question and answer pairs does not impact the SDG or training process.

For SDG to run properly, your **qna.yaml** file must include the following: * At least five **context** chunks of information from your knowledge data file. * At least three question and answer seeds per **context** value in the **questions_and_answers** parameter.

Example knowledge qna.yaml file

```
version: 3 (1)
domain: astronomy (2)
created_by: <user-name> (3)
seed_examples:
  - context: | (4)
      **Phoenix** is a minor constellation in the southern sky. Named after the mythical
      Phoenix_(mythology), it was first depicted on a celestial atlas by Johann Bayerin his 1603
      Uranometria. The French explorer and astronomer Nicolas Louis de Lacaille charted the brighter
stars
      and gave their Bayer designations in 1756. The constellation stretches from roughly –39 degrees
to –57
      degrees declination, and from 23.5h to 2.5h of right ascension. The constellations Phoenix, Grus,
      Pavo and Tucana are known as the Southern Birds.
    questions_and_answers:
      - question: | (5)
          What is the Phoenix constellation?
        answer: | (6)
          Phoenix is a minor constellation in the southern sky.
      - question: |
          Who charted the Phoenix constellation?
        answer: |
          The Phoenix constellation was charted by french explorer and
          astronomer Nicolas Louis de Lacaille.
      - question: |
          How far does the Phoenix constellation stretch?
        answer: |
          The phoenix constellation stretches from roughly –39° to –57°
          declination, and from 23.5h to 2.5h of right ascension.
  - context: |
      Phoenix was the largest of the 12 constellations established by Petrus Plancius from the
observations
```

```
    of Pieter Dirkszoon Keyser and Frederick de Houtman. It first appeared on a 35cm diameter
celestial globe
    published in 1597 (or 1598) in Amsterdam by Plancius with Jodocus Hondius. The first depiction
of this
    constellation in a celestial atlas was in Johann Bayer *Uranometria* of 1603. De Houtman
included it in
    his southern star catalog the same year under the Dutch name *Den voghel Fenicx*, "The Bird
Phoenix",
    symbolising the phoenix of classical mythology. One name of the brightest star Alpha Phoenicis—
Ankaa—is
    derived from the Arabic: العنقاء, romanized: al-‘anqā’, lit. 'the phoenix', and was coined sometime
after
    1800 in relation to the constellation.
  questions_and_answers:
    - question: |
        What is the brightest star in the Phoenix constellation
        called?
      answer: |
        Alpha Phoenicis or Ankaa is the brightest star in the Phoenix
        Constellation.
    - question: Where did the Phoenix constellation first appear?
      answer: |
        The Phoenix constellation first appeared on a 35-cm diameter
        celestial globe published in 1597 (or 1598) in Amsterdam by
        Plancius with Jodocus Hondius.
    - question: |
        What does "The Bird Phoenix" symbolize?
      answer: |
        "The Bird Phoenix" symbolizes the phoenix of classical mythology.
- context: |
    Phoenix is a small constellation bordered by Fornax and Sculptor to the north, Grus to the west,
    Tucana to the south, touching on the corner of Hydrus to the south, and Eridanus to the east and
southeast.
    The bright star Achernar is nearby. The three-letter abbreviation for the constellation, as adopted
by
    the International Astronomical Union in 1922, is "Phe". The official constellation boundaries,
    as set by Belgian astronomer Eugène Delporte in 1930, are defined by a polygon of 10
segments.
    In the equatorial coordinate system, the right ascension coordinates of these borders lie
    between 23<sup>h</sup> 26.5<sup>m</sup> and 02<sup>h</sup> 25.0<sup>m</sup>,
    while the declination coordinates are between –39.31° and –57.84°. This means it remains
    below the horizon to anyone living north of the 40th parallel in the Northern
    Hemisphere, and remains low in the sky for anyone living north of the equator. It is most
    visible from locations such as Australia and South Africa during late Southern Hemisphere spring.
    Most of the constellation lies within, and can be located by, forming a triangle of the bright
    stars Achernar, Fomalhaut and Beta Ceti—Ankaa lies roughly in the centre of this.
  questions_and_answers:
    - question: What are the characteristics of the Phoenix constellation?
      answer: |
        Phoenix is a small constellation bordered by Fornax and Sculptor to
        the north, Grus to the west, Tucana to the south, touching on the
        corner of Hydrus to the south, and Eridanus to the east and southeast.
        The bright star Achernar is nearby.
    - question: |
        When is the phoenix constellation most visible?
      answer: |
```

```
        Phoenix is most visible from locations such as Australia and
        South Africa during late Southern Hemisphere spring.
    - question: |
        What are the Phoenix Constellation boundaries?
      answer: |
        The official constellation boundaries for Phoenix, as set by Belgian
        astronomer Eugène Delporte in 1930, are defined by a polygon of 10
        segments.
  - context: |
    Ten stars have been found to have planets to date, and four planetary systems have been
    discovered with the SuperWASP project. HD 142 is a yellow giant that has an apparent
magnitude
    of 5.7, and has a planet (HD 142b) 1.36 times the mass of Jupiter which orbits every 328 days.
    HD 2039 is a yellow subgiant with an apparent magnitude of 9.0 around 330 light years away
which
    has a planet (HD 2039) six times the mass of Jupiter. WASP-18 is a star of magnitude 9.29 which
    was discovered to have a hot Jupiter-like planet (WASP-18b) taking less than a day to orbit the
star.
    The planet is suspected to be causing WASP-18 to appear older than it really is. WASP-4and
WASP-5
    are solar-type yellow stars around 1000 light years distant and of 13th magnitude, each with a
single
    planet larger than Jupiter. WASP-29 is an orange dwarf of spectral type K4V and visual magnitude
11.3,
    which has a planetary companion of similar size and mass to Saturn. The planet completes an
orbit
    every 3.9 days.
  questions_and_answers:
    - question: In the Phoenix constellation, how many stars have planets?
      answer: |
        In the Phoenix constellation, ten stars have been found to have
        planets to date, and four planetary systems have been discovered
        with the SuperWASP project.
    - question: |
        What is HD 142?
      answer: |
        HD 142 is a yellow giant that has an apparent magnitude of 5.7, and
        has a planet (HD 142 b) 1.36 times the mass of Jupiter which
        orbits every 328 days.
    - question: |
        Are WASP-4 and WASP-5 solar-type yellow stars?
      answer: |
        Yes, WASP-4 and WASP-5 are solar-type yellow stars around 1000 light
        years distant and of 13th magnitude, each with a single planet
        larger than Jupiter.
  - context: |
    The constellation does not lie on the galactic plane of the Milky Way, and there are no prominent
star
    clusters. NGC 625 is a dwarf irregular galaxy of apparent magnitude 11.0 and lying some 12.7
million
    light years distant. Only 24000 light years in diameter, it is an outlying member of the Sculptor
Group.
    NGC 625 is thought to have been involved in a collision and is experiencing a burst of active star
formation.
    NGC 37 is a lenticular galaxy of apparent magnitude 14.66. It is approximately 42 kiloparsecs
137,000
```

```
      light-years in diameter and about 12.9 billion years old. Robert's Quartet composed of the
irregular galaxy
      NGC 87, and three spiral galaxies NGC 88, NGC 89 and NGC 92 is a group of four galaxies
located around 160 million
      light-years away which are in the process of colliding and merging. They are within a circle of
radius of 1.6 arcmin,
      corresponding to about 75,000 light-years. Located in the galaxy ESO 243-49 is HLX-1, an
intermediate-mass
      black hole—the first one of its kind identified. It is thought to be a remnant of a dwarf galaxy that
was absorbed
      in a collision with ESO 243-49. Before its discovery, this class of black hole was only
hypothesized.
    questions_and_answers:
      - question: |
          Is the Phoenix Constellation part of the Milky Way?
        answer: |
          The Phoenix constellation does not lie on the galactic plane of
          the Milky Way, and there are no prominent star clusters.
      - question: |
          How many light years away is NGC 625?
        answer: |
          NGC 625 is 24000 light years in diameter and is an outlying
          member of the Sculptor Group.
      - question: |
          What is Robert's Quartet composed of?
        answer: |
          Robert's Quartet is composed of the irregular galaxy NGC 87,
          and three spiral galaxies NGC 88, NGC 89 and NGC 92.
document_outline: | (7)
  Information about the Phoenix Constellation including the
  history, characteristics, and features of the stars in the constellation.
document:
  repo: https://github.com/<profile>/<repo-name> / (8)
  commit: <commit hash> (9)
  patterns:
  - phoenix_constellation.md (10)
```

1. Specify the version of the knowledge **qna.yaml** format. Currently, the valid value is **3**.
2. Specify the subject of the document. For example, "technical_documents" or "installation_guides".
3. Specify your name or git username.
4. Specify a brief paragraph of your knowledge data. This is content that your questions and answers will be based on. The format of the context block must match the format of the markdown file.
5. Specify a question for the model. For example, "What is the latest version of the product?".
6. Specify the desired response from the model. For example, "The latest version of the product is version 1.5".
7. Specify a brief outline of the document's contents. For example, "Information about installing the product".
8. Specify the URL to the repository that holds your knowledge markdown files.

9 Specify the SHA of the commit from your git repository of your knowledge markdown files.

10 Specify the **.md** file in your git repository.

1.2.3. Optional: Creating an attributions file

You can create an **attribution.txt** file with your **qna.yaml** file to keep track of your files.

Prerequisites

- You created a **qna.yaml** with your knowledge data.
- You have root user access on your machine.

Procedures

1. Navigate to the directory where you created the **qna.yaml** file.
2. Create a text file in your preferred text editor called **attribution.txt**. The contents of this file can help keep track of the files in your taxonomy tree.

   Example attribution.txt file

   ```
   Title of work: Phoenix (constellation)
   Link to work: https://en.wikipedia.org/wiki/Phoenix_(constellation)
   Revision: https://en.wikipedia.org/w/index.php?
   title=Phoenix_(constellation)&oldid=1237187773
   License of the work: CC-BY-SA-4.0
   Creator names: Wikipedia Authors
   ```

CHAPTER 2. GENERATING A NEW DATASET WITH SDG

After customizing your taxonomy tree, you can generate a synthetic dataset using the Synthetic Data Generation (SDG) process on Red Hat Enterprise Linux AI. SDG is a process that creates an artificially generated dataset that mimics real data based on provided examples. SDG uses a YAML file containing question-and-answer pairs as input data. With these examples, SDG utilizes the **mixtral-8x7b-instruct-v0-1** LLM as a teacher model to generate similar question-and-answer pairs. In the SDG pipeline, many questions are generated and scored based on quality, where the **mixtral-8x7b-instruct-v0-1** model assesses the quality of these questions. The pipeline then selects the highest-scoring questions, generates corresponding answers, and includes these pairs in the synthetic dataset.

2.1. CREATING A SYNTHETIC DATASET USING YOUR EXAMPLES

You can use your examples and run the SDG process to create a synthetic dataset.

Prerequisites

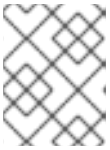
- You installed RHEL AI with the bootable container image.
- You created a custom **qna.yaml** file with knowledge data.
- You downloaded the **mixtral-8x7b-instruct-v0-1** teacher model for SDG.
- You downloaded the **skills-adapter-v3:1.2** and **knowledge-adapter-v3:1.2** LoRA layered skills and knowledge adapter.
- You have root user access on your machine.

Procedure

1. To generate a new synthetic dataset, based on your custom taxonomy with knowledge, run the following command:

```
$ ilab data generate
```

This command runs SDG with **mixtral-8x7B-instruct** as the teacher model



NOTE

You can use the **--enable-serving-output** flag when running the **ilab data generate** command to display the vLLM startup logs.

a. At the start of the SDG process, vLLM attempts to start a server.

Example output of vLLM attempting to start a server

```
Starting a temporary vLLM server at http://127.0.0.1:47825/v1
INFO 2024-08-22 17:01:09,461 instructlab.model.backends.backends:480: Waiting for
the vLLM server to start at http://127.0.0.1:47825/v1, this might take a moment...
Attempt: 1/120
INFO 2024-08-22 17:01:14,213 instructlab.model.backends.backends:480: Waiting for
the vLLM server to start at http://127.0.0.1:47825/v1, this might take a moment...
Attempt: 2/120
```

```
INFO 2024-08-22 17:01:19,142 instructlab.model.backends.backends:480: Waiting for
the vLLM server to start at http://127.0.0.1:47825/v1, this might take a moment...
Attempt: 3/120
```

b. Once vLLM connects, the SDG process starts creating synthetic data from your examples.

Example output of vLLM connecting and SDG generating

```
INFO 2024-08-22 15:16:38,933 instructlab.model.backends.backends:480: Waiting for
the vLLM server to start at http://127.0.0.1:49311/v1, this might take a moment...
Attempt: 30/120
INFO 2024-08-22 15:16:43,497 instructlab.model.backends.backends:480: Waiting for
the vLLM server to start at http://127.0.0.1:49311/v1, this might take a moment...
Attempt: 31/120
INFO 2024-08-22 15:16:45,949 instructlab.model.backends.backends:487: vLLM engine
successfully started at http://127.0.0.1:49311/v1
Generating synthetic data using '/usr/share/instructlab/sdg/pipelines/agentic' pipeline,
'/var/home/cloud-user/.cache/instructlab/models/mixtral-8x7b-instruct-v0-1' model,
'/var/home/cloud-user/.local/share/instructlab/taxonomy' taxonomy, against
http://127.0.0.1:49311/v1 server
INFO 2024-08-22 15:16:46,594 instructlab.sdg:375: Synthesizing new instructions. If you
aren't satisfied with the generated instructions, interrupt training (Ctrl-C) and try adjusting
your YAML files. Adding more examples may help.
```

2. The SDG process completes when the CLI displays the location of your new data set.

Example output of a successful SDG run

```
INFO 2024-09-15 17:12:46,548 instructlab.sdg.datamixing:200: Mixed Dataset saved to
/home/example-user/.local/share/instructlab/datasets/skills_train_msgs_2024-08-
16T16_50_11.jsonl
INFO 2024-09-15 17:12:46,549 instructlab.sdg:438: Generation took 1355.74s
```

NOTE

This process can be time consuming depending on your hardware specifications.

3. Verify the files are created by running the following command:

```
$ ls ~/.local/share/instructlab/datasets/
```

Example output

```
knowledge_recipe_2024-09-15T20_54_21.yaml                skills_recipe_2024-09-
15T20_54_21.yaml
knowledge_train_msgs_2024-09-15T20_54_21.jsonl           skills_train_msgs_2024-09-
15T20_54_21.jsonl
messages_granite-7b-lab-Q4_K_M_2024-09-15T20_54_21.jsonl  node_datasets_2024-09-
15T15_12_12/
```

IMPORTANT

Make a note of your most recent **knowledge_train_msgs.jsonl** and **skills_train_msgs.jsonl** file. You need to specify this file during multi-phase training. Each JSONL has the time stamp on the file, for example **knowledge_train_msgs_2024-08-08T20_04_28.jsonl**, use the most recent file when training.

4. Optional: You can view output of SDG by navigating to the **~/.local/share/datasets/** directory and opening the **JSONL** file.

```
$ cat ~/.local/share/datasets/<jsonl-dataset>
```

Example output of a SDG JSONL file

{"messages":[{"content":"I am, Red Hat\u00ae Instruct Model based on Granite 7B, an AI language model developed by Red Hat and IBM Research, based on the Granite-7b-base language model. My primary function is to be a chat assistant.","role":"system"},{"content":"
<|user|>\n### Deep-sky objects\n\nThe constellation does not lie on the [galactic\nplane](galactic_plane \"wikilink\") of the Milky Way, and there are no\nprominent star clusters. [NGC 625](NGC_625 \"wikilink\") is a dwarf\n[irregular galaxy](irregular_galaxy \"wikilink\") of apparent magnitude\n11.0 and lying some 12.7 million light years distant.
Only 24000 light\nyears in diameter, it is an outlying member of the [Sculptor\nGroup](Sculptor_Group \"wikilink\"). NGC 625 is thought to have been\ninvolved in a collision and is experiencing a burst of [active star\nformation](Active_galactic_nucleus \"wikilink\"). [NGC\n37](NGC_37 \"wikilink\") is a [lenticular\ngalaxy](lenticular_galaxy \"wikilink\") of apparent magnitude 14.66. It is\napproximately 42 [kiloparsecs](kiloparsecs \"wikilink\") (137,000\n[light-years](light-years \"wikilink\")) in diameter and about 12.9\nbillion years old. [Robert's Quartet](Robert's_Quartet \"wikilink\")\n(composed of the
irregular galaxy [NGC 87](NGC_87 \"wikilink\"), and three\nspiral galaxies [NGC 88](NGC_88 \"wikilink\"), [NGC 89](NGC_89 \"wikilink\")\nand [NGC 92](NGC_92 \"wikilink\")) is a group of four galaxies located\naround 160 million light-years away which are in the process of\ncolliding and merging. They are within a circle of radius of 1.6 arcmin,\ncorresponding to about 75,000 light-years. Located in the galaxy ESO\n243-49 is [HLX-1](HLX-1 \"wikilink\"), an [intermediate-mass black\nhole](intermediate-mass_black_hole \"wikilink\")\u2014the first one of its kind\nidentified. It is thought to be a remnant of a dwarf
galaxy that was\nabsorbed in a [collision](Interacting_galaxy \"wikilink\") with ESO\n243-49. Before its discovery, this class of black hole was only\nhypothesized.\n\nLying within the bounds of the constellation is the gigantic [Phoenix\ncluster](Phoenix_cluster \"wikilink\"), which is around 7.3 million light\nyears wide and 5.7 billion light years away, making it one of the most\nmassive [galaxy clusters](galaxy_cluster \"wikilink\"). It was first\ndiscovered in 2010, and the central galaxy is producing an estimated 740\nnew stars a year. Larger still is [El\nGordo](El_Gordo_(galaxy_cluster) \"wikilink\"),
or officially ACT-CL\nJ0102-4915, whose discovery was announced in 2012. Located around\n7.2 billion light years away, it is composed of two subclusters in the\nprocess of colliding, resulting in the spewing out of hot gas, seen in\nX-rays and infrared images.\n\n### Meteor showers\n\nPhoenix is the [radiant](radiant_(meteor_shower) \"wikilink\") of two\nannual [meteor showers](meteor_shower \"wikilink\"). The\n[Phoenicids](Phoenicids \"wikilink\"), also known as the December\nPhoenicids, were first observed on 3 December 1887. The shower was\nparticularly intense in December 1956, and is thought related to the\nbreakup
of the [short-period comet](short-period_comet \"wikilink\")\n[289P\/Blanpain](289P\/Blanpain \"wikilink\"). It peaks around 4\u20135 December,\nthough is not seen every year. A very minor meteor shower peaks\naround July 14 with around one meteor an hour, though meteors can be\nseen anytime from July 3 to 18; this shower is referred to as the

July\nPhoenicids.\n\nHow many light years wide is the Phoenix cluster?\n<|assistant|>\n' 'The Phoenix cluster is around 7.3 million light years wide.'","role":"pretraining"}],"metadata":" {\"sdg_document\": \"### Deep-sky objects\\n\\nThe constellation does not lie on the [galactic\\nplane](galactic_plane \\\"wikilink\\\") of the Milky Way,
and there are no\\nprominent star clusters. [NGC 625](NGC_625 \\\"wikilink\\\") is a dwarf\\n[irregular galaxy](irregular_galaxy \\\"wikilink\\\") of apparent magnitude\\n11.0 and lying some 12.7 million light years distant. Only 24000 light\\nyears in diameter, it is an outlying member of the [Sculptor\\nGroup](Sculptor_Group \\\"wikilink\\\"). NGC 625 is thought to have been\\ninvolved in a collision and is experiencing a burst of [active star\\nformation](Active_galactic_nucleus \\\"wikilink\\\"). [NGC\\n37](NGC_37 \\\"wikilink\\\") is a [lenticular\\ngalaxy](lenticular_galaxy \\\"wikilink\\\") of apparent magnitude 14.66. It is\\napproximately 42
[kiloparsecs](kiloparsecs \\\"wikilink\\\") (137,000\\n[light-years](light-years \\\"wikilink\\\")) in diameter and about 12.9\\nbillion years old. [Robert's Quartet](Robert's_Quartet \\\"wikilink\\\")\\n(composed of the irregular galaxy [NGC 87](NGC_87 \\\"wikilink\\\"), and three\\nspiral galaxies [NGC 88](NGC_88 \\\"wikilink\\\"), [NGC 89](NGC_89 \\\"wikilink\\\")\\nand [NGC 92](NGC_92 \\\"wikilink\\\")) is a group of four galaxies located\\naround 160 million light-years away which are in the process of\\ncolliding and merging. They are within a circle of radius of 1.6 arcmin,\\ncorresponding to about 75,000 light-years.
Located in the galaxy ESO\\n243-49 is [HLX-1](HLX-1 \\\"wikilink\\\"), an [intermediate-mass black\\nhole](intermediate-mass_black_hole \\\"wikilink\\\")\\u2014the first one of its kind\\nidentified. It is thought to be a remnant of a dwarf galaxy that was\\nabsorbed in a [collision](Interacting_galaxy \\\"wikilink\\\") with ESO\\n243-49. Before its discovery, this class of black hole was only\\nhypothesized.\\n\\nLying within the bounds of the constellation is the gigantic [Phoenix\\ncluster](Phoenix_cluster \\\"wikilink\\\"), which is around 7.3 million light\\nyears wide and 5.7 billion light years away, making it one of the most\\nmassive [galaxy clusters](galaxy_cluster \\\"wikilink\\\").
It was first\\ndiscovered in 2010, and the central galaxy is producing an estimated 740\\nnew stars a year. Larger still is [El\\nGordo](El_Gordo_(galaxy_cluster) \\\"wikilink\\\"), or officially ACT-CL\\nJ0102-4915, whose discovery was announced in 2012. Located around\\n7.2 billion light years away, it is composed of two subclusters in the\\nprocess of colliding, resulting in the spewing out of hot gas, seen in\\nX-rays and infrared images.\\n\\n### Meteor showers\\n\\nPhoenix is the [radiant](radiant_(meteor_shower) \\\"wikilink\\\") of two\\nannual [meteor showers](meteor_shower \\\"wikilink\\\"). The\\n[Phoenicids](Phoenicids \\\"wikilink\\\"),
also known as the December\\nPhoenicids, were first observed on 3 December 1887. The shower was\\nparticularly intense in December 1956, and is thought related to the\\nbreakup of the [short-period comet](short-period_comet \\\"wikilink\\\")\\n[289P\/Blanpain](289P\/Blanpain \\\"wikilink\\\"). It peaks around 4\\u20135 December,\\nthough is not seen every year. A very minor meteor shower peaks\\naround July 14 with around one meteor an hour, though meteors can be\\nseen anytime from July 3 to 18; this shower is referred to as the July\\nPhoenicids.\", \"domain\": \"astronomy\", \"dataset\": \"document_knowledge_qa\"}","id":"1df7c219-a062-4511-8bae-f55c88927dc1"}

CHAPTER 3. TRAINING A MODEL

RHEL AI can use your taxonomy tree and synthetic data to create a newly trained model with your domain-specific knowledge or skills using multi-phase training and evaluation. You can run the full training and evaluation process using the synthetic dataset you generated. The LAB optimized technique of multi-phase training is a type of LLM training that goes through multiple stages of training and evaluation. In these various stages, RHEL AI runs the training process and produces model checkpoints. The best checkpoint is selected for the next phase. This process creates many checkpoints and selects the best scored checkpoint. This best scored checkpoint is your newly trained LLM.

The entire process creates a newly generated model that is trained and evaluated using the synthetic data from your taxonomy tree.

3.1. TRAINING THE MODEL ON YOUR DATA

You can use Red Hat Enterprise Linux AI to train a model with your synthetically generated data. The following procedures show how to do this using the LAB multi-phase training strategy.

IMPORTANT

Red Hat Enterprise Linux AI general availability does not support training and inference serving at the same time. If you have an inference server running, you must close it before you start the training process.

Prerequisites

- You installed RHEL AI with the bootable container image.
- You downloaded the **granite-7b-starter** model.
- You created a custom **qna.yaml** file with knowledge data.
- You ran the synthetic data generation (SDG) process.
- You downloaded the **prometheus-8x7b-v2-0** judge model.
- You have root user access on your machine.

Procedure

1. You can run multi-phase training and evaluation by running the following command with the data files generated from SDG.

NOTE

You can use the **--enable-serving-output** flag with the **ilab model train** commmand to display the training logs.

```
$ ilab model train --strategy lab-multiphase \
        --phased-phase1-data ~/.local/share/instructlab/datasets/<knowledge-train-messages-jsonl-file> \
        --phased-phase2-data ~/.local/share/instructlab/datasets/<skills-train-messages-jsonl-file>
```

where

<knowledge-train-messages-file>

The location of the **knowledge_messages.jsonl** file generated during SDG. RHEL AI trains the student model **granite-7b-starter** using the data from this **.jsonl** file. Example path: **~/.local/share/instructlab/datasets/knowledge_train_msgs_2024-08-13T20_54_21.jsonl**.

<skills-train-messages-file>

The location of the **skills_messages.jsonl** file generated during SDG. RHEL AI trains the student model **granite-7b-starter** using the data from the **.jsonl** file. Example path: **~/.local/share/instructlab/datasets/skills_train_msgs_2024-08-13T20_54_21.jsonl**.

IMPORTANT

This process can be time consuming depending on your hardware specifications.

a. The first phase trains the model using the synthetic data from your knowledge contribution.

Example output of training knowledge

```
Training Phase 1/2...
TrainingArgs for current phase: TrainingArgs(model_path='/opt/app-
root/src/.cache/instructlab/models/granite-7b-starter', chat_tmpl_path='/opt/app-
root/lib64/python3.11/site-
packages/instructlab/training/chat_templates/ibm_generic_tmpl.py',
data_path='/tmp/jul19-knowledge-26k.jsonl',
ckpt_output_dir='/tmp/e2e/phase1/checkpoints', data_output_dir='/opt/app-
root/src/.local/share/instructlab/internal', max_seq_len=4096, max_batch_len=55000,
num_epochs=2, effective_batch_size=128, save_samples=0, learning_rate=2e-05,
warmup_steps=25, is_padding_free=True, random_seed=42,
checkpoint_at_epoch=True, mock_data=False, mock_data_len=0,
deepspeed_options=DeepSpeedOptions(cpu_offload_optimizer=False,
cpu_offload_optimizer_ratio=1.0, cpu_offload_optimizer_pin_memory=False,
save_samples=None), disable_flash_attn=False, lora=LoraOptions(rank=0,
alpha=32, dropout=0.1, target_modules=('q_proj', 'k_proj', 'v_proj', 'o_proj'),
quantize_data_type=<QuantizeDataType.NONE: None>))
```

b. Then, RHEL AI selects the best checkpoint to use for the next phase.

c. The next phase trains the model using the synthetic data from the skills data.

Example output of training skills

```
Training Phase 2/2...
TrainingArgs for current phase:
TrainingArgs(model_path='/tmp/e2e/phase1/checkpoints/hf_format/samples_52096',
chat_tmpl_path='/opt/app-root/lib64/python3.11/site-
packages/instructlab/training/chat_templates/ibm_generic_tmpl.py',
data_path='/usr/share/instructlab/sdg/datasets/skills.jsonl',
ckpt_output_dir='/tmp/e2e/phase2/checkpoints', data_output_dir='/opt/app-
root/src/.local/share/instructlab/internal', max_seq_len=4096, max_batch_len=55000,
num_epochs=2, effective_batch_size=3840, save_samples=0, learning_rate=2e-05,
warmup_steps=25, is_padding_free=True, random_seed=42,
```

```
checkpoint_at_epoch=True, mock_data=False, mock_data_len=0,
deepspeed_options=DeepSpeedOptions(cpu_offload_optimizer=False,
cpu_offload_optimizer_ratio=1.0, cpu_offload_optimizer_pin_memory=False,
save_samples=None), disable_flash_attn=False, lora=LoraOptions(rank=0,
alpha=32, dropout=0.1, target_modules=('q_proj', 'k_proj', 'v_proj', 'o_proj'),
quantize_data_type=<QuantizeDataType.NONE: None>))
```

d. Then, RHEL AI evaluates all of the checkpoints from phase 2 model training using the Multi-turn Benchmark (MT-Bench) and returns the best performing checkpoint as the fully trained output model.

Example output of evaluating skills

```
MT-Bench evaluation for Phase 2...
Using gpus from --gpus or evaluate config and ignoring --tensor-parallel-size
configured in serve vllm_args
INFO 2024-05-07 10:04:51,065 instructlab.model.backends.backends:437: Trying to
connect to model server at http://127.0.0.1:8000/v1
INFO 2024-05-07 10:04:53,580 instructlab.model.backends.vllm:208: vLLM starting
up on pid 79388 at http://127.0.0.1:54265/v1
INFO 2024-05-07 10:04:53,580 instructlab.model.backends.backends:450: Starting a
temporary vLLM server at http://127.0.0.1:54265/v1
INFO 2024-05-07 10:04:53,580 instructlab.model.backends.backends:465: Waiting
for the vLLM server to start at http://127.0.0.1:54265/v1, this might take a moment...
Attempt: 1/300
INFO 2024-05-07 10:04:58,003 instructlab.model.backends.backends:465: Waiting
for the vLLM server to start at http://127.0.0.1:54265/v1, this might take a moment...
Attempt: 2/300
INFO 2024-05-07 10:05:02,314 instructlab.model.backends.backends:465: Waiting
for the vLLM server to start at http://127.0.0.1:54265/v1, this might take a moment...
Attempt: 3/300
moment... Attempt: 3/300
INFO 2024-05-07 10:06:07,611 instructlab.model.backends.backends:472: vLLM
engine successfully started at http://127.0.0.1:54265/v1
```

2. After training is complete, a confirmation appears and displays your best performed checkpoint.

Example output of a complete multi-phase training run

```
Training finished! Best final checkpoint: samples_1945 with score: 6.813759384
```

Make a note of this checkpoint because the path is necessary for evaluation and serving.

Verification

- When training a model with **ilab model train**, multiple checkpoints are saved with the **samples_** prefix based on how many data points they have been trained on. These are saved to the ~/**.local**/**share**/**instructlab**/**phase**/ directory.

```
$ ls ~/.local/share/instructlab/phase/<phase1-or-phase2>/checkpoints/
```

Example output of the new models

```
samples_1711 samples_1945 samples_1456 samples_1462 samples_1903
```

3.1.1. Continuing or restarting a training run

RHEL AI allows you to continue a training run that may have failed during multi-phase training. There are a few ways a training run can fail:

- The vLLM server may not start correctly.
- A accelerator or GPU may freeze, causing training to abort.
- There may be an error in your InstructLab **config.yaml** file.

When you run multi-phase training for the first time, the initial training data gets saved into a **journalfile.yaml** file. If necessary, this metadata in the file can be used to restart a failed training.

You can also restart a training run which clears the training data by following the CLI prompts when running multi-phase training.

Prerequisites

- You ran multi-phase training with your synthetic data and that failed.

Procedure

1. Run the multi-phase training command again.

```
$ ilab model train --strategy lab-multiphase \
        --phased-phase1-data ~/.local/share/instructlab/datasets/<knowledge-train-messages-jsonl-file> \
        --phased-phase2-data ~/.local/share/instructlab/datasets/<skills-train-messages-jsonl-file>
```

   The Red Hat Enterprise Linux AI CLI reads if the **journalfile.yaml** file exists and continues the training run from that point.

2. The CLI prompts you to continue for the previous training run, or start from the beginning.

   - Type **n** in your shell to continue from your previews training run.

```
Metadata (checkpoints, the training journal) may have been saved from a previous training run.
By default, training will resume from this metadata if it exists
Alternatively, the metadata can be cleared, and training can start from scratch
Would you like to START TRAINING FROM THE BEGINNING? n
```

   - Type **y** into the terminal to restart a training run.

```
Metadata (checkpoints, the training journal) may have been saved from a previous training run.
By default, training will resume from this metadata if it exists
Alternatively, the metadata can be cleared, and training can start from scratch
Would you like to START TRAINING FROM THE BEGINNING? y
```

Restarting also clears your systems cache of previous checkpoints, journal files and other training data.

CHAPTER 4. EVALUATING THE MODEL

If you want to measure the improvements of your new model, you can compare its performance to the base model with the evaluation process. You can also chat with the model directly to qualitatively identify whether the new model has learned the knowledge you created. If you want more quantitative results of the model improvements, you can run the evaluation process in the RHEL AI CLI.

4.1. EVALUATING YOUR NEW MODEL

If you want to measure the improvements of your new model, you can compare its performance to the base model with the evaluation process. You can also chat with the model directly to qualitatively identify whether the new model has learned the knowledge you created. If you want more quantitative results of the model improvements, you can run the evaluation process in the RHEL AI CLI with the following procedure.

Prerequisites

- You installed RHEL AI with the bootable container image.
- You created a custom **qna.yaml** file with skills or knowledge.
- You ran the synthetic data generation process.
- You trained the model using the RHEL AI training process.
- You downloaded the **prometheus-8x7b-v2-0** judge model.
- You have root user access on your machine.

Procedure

1. Navigate to your working Git branch where you created your **qna.yaml** file.
2. You can now run the evaluation process on different benchmarks. Each command needs the path to the trained **samples** model to evaluate, you can access these checkpoints in your **~/.local/share/instructlab/checkpoints** folder.

   a. **MMLU_BRANCH benchmark -** If you want to measure how your knowledge contributions have impacted your model, run the **mmlu_branch** benchmark by executing the following command:

   ```
   $ ilab model evaluate --benchmark mmlu_branch
      --model ~/.local/share/instructlab/phased/phase2/checkpoints/hf_format/<checkpoint>
   \
      --tasks-dir ~/.local/share/instructlab/datasets/<node-dataset> \
      --base-model ~/.cache/instructlab/models/granite-7b-starter
   ```

   where

   <checkpoint>

   Specify the best scored checkpoint file generated during multi-phase training

   <node-dataset>

   Specify the **node_datasets** directory, in the **~/.local/share/instructlab/datasets/** directory, with the same timestamps as the.jsonl files used for training the model.

Example output

```
# KNOWLEDGE EVALUATION REPORT

## BASE MODEL (SCORE)
/home/user/.cache/instructlab/models/instructlab/granite-7b-lab/ (0.74/1.0)

## MODEL (SCORE)
/home/user/local/share/instructlab/phased/phases2/checkpoints/hf_format/samples_665(
0.78/1.0)

### IMPROVEMENTS (0.0 to 1.0):
1. tonsils: 0.74 -> 0.78 (+0.04)
```

b. **Optional: MT_BENCH_BRANCH benchmark -**If you want to measure how your skills contributions have impacted your model, run the **mt_bench_branch** benchmark by executing the following command:

```
$ ilab model evaluate \
    --benchmark mt_bench_branch \
    --model ~/.local/share/instructlab/phased/phase2/checkpoints/hf_format/<checkpoint>
\
    --judge-model ~/.cache/instructlab/models/prometheus-8x7b-v2-0 \
    --branch <worker-branch> \
    --base-branch <worker-branch>
```

where

<checkpoint>

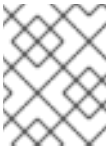
Specify the best scored checkpoint file generated during multi-phase training.

<worker-branch>

Specify the branch you used when adding data to your taxonomy tree.

<num-gpus>

Specify the number of GPUs you want to use for evaluation.



NOTE

Customizing skills is not currently supported on Red Hat Enterprise Linux AI version 1.2.

Example output

```
# SKILL EVALUATION REPORT

## BASE MODEL (SCORE)
/home/user/.cache/instructlab/models/instructlab/granite-7b-lab (5.78/10.0)

## MODEL (SCORE)
/home/user/local/share/instructlab/phased/phases2/checkpoints/hf_format/samples_665(
6.00/10.0)
```

```
### IMPROVEMENTS (0.0 to 10.0):
1. foundational_skills/reasoning/linguistics_reasoning/object_identification/qna.yaml:
4.0 -> 6.67 (+2.67)
2. foundational_skills/reasoning/theory_of_mind/qna.yaml: 3.12 -> 4.0 (+0.88)
3.
foundational_skills/reasoning/linguistics_reasoning/logical_sequence_of_words/qna.yam
: 9.33 -> 10.0 (+0.67)
4. foundational_skills/reasoning/logical_reasoning/tabular/qna.yaml: 5.67 -> 6.33
(+0.67)
5. foundational_skills/reasoning/common_sense_reasoning/qna.yaml: 1.67 -> 2.33
(+0.67)
6. foundational_skills/reasoning/logical_reasoning/causal/qna.yaml: 5.67 -> 6.0
(+0.33)
7. foundational_skills/reasoning/logical_reasoning/general/qna.yaml: 6.6 -> 6.8 (+0.2)
8. compositional_skills/writing/grounded/editing/content/qna.yaml: 6.8 -> 7.0 (+0.2)
9. compositional_skills/general/synonyms/qna.yaml: 4.5 -> 4.67 (+0.17)

### REGRESSIONS (0.0 to 10.0):
1.
foundational_skills/reasoning/unconventional_reasoning/lower_score_wins/qna.yaml:
5.67 -> 4.0 (-1.67)
2. foundational_skills/reasoning/mathematical_reasoning/qna.yaml: 7.33 -> 6.0 (-1.33)
3. foundational_skills/reasoning/temporal_reasoning/qna.yaml: 5.67 -> 4.67 (-1.0)

### NO CHANGE (0.0 to 10.0):
1. foundational_skills/reasoning/linguistics_reasoning/odd_one_out/qna.yaml (9.33)
2. compositional_skills/grounded/linguistics/inclusion/qna.yaml (6.5)
```

3. Optional: You can manually evaluate each checkpoint using the MMLU and MT_BENCH benchmarks. You can evaluate any model against the standardized set of knowledge or skills, allowing you to compare the scores of your own model against other LLMs. If you do run multi-phase training, this process is done with single-phase training.

   a. **MMLU -** If you want to see the evaluation score of your new model against a standardized set of knowledge data, set the **mmlu** benchmark by running the following command:

   ```
   $ ilab model evaluate --benchmark mmlu --model
   ~/.local/share/instructlab/phased/phase2/checkpoints/hf_format/samples_665
   ```

   where

   <checkpoint>

   Specify one of the checkpoint files generated during multi-phase training.

   Example output

   ```
   # KNOWLEDGE EVALUATION REPORT

   ## MODEL (SCORE)
   /home/user/.local/share/instructlab/phased/phase2/checkpoints/hf_format/samples_665


   ### SCORES (0.0 to 1.0):
   mmlu_abstract_algebra - 0.31
   ```

```
mmlu_anatomy - 0.46
mmlu_astronomy - 0.52
mmlu_business_ethics - 0.55
mmlu_clinical_knowledge - 0.57
mmlu_college_biology - 0.56
mmlu_college_chemistry - 0.38
mmlu_college_computer_science - 0.46
...
```

b. **MT_BENCH -** If you want to see the evaluation score of your new model against a standardized set of skills, set the **mt_bench** benchmark by running the following command:

```
$ ilab model evaluate --benchmark mt_bench --model
~/.local/share/instructlab/phased/phases2/checkpoints/hf_format/samples_665
```

where

<checkpoint>

Specify one of the checkpoint files generated during multi-phase training.

Example output

```
# SKILL EVALUATION REPORT

## MODEL (SCORE)
/home/user/local/share/instructlab/phased/phases2/checkpoints/hf_format/samples_665(
7.27/10.0)

### TURN ONE (0.0 to 10.0):
7.48

### TURN TWO (0.0 to 10.0):
7.05
```

CHAPTER 5. SERVING AND CHATTING WITH YOUR NEW MODEL

You must deploy the model to your machine by serving the model. This deploys the model and makes the model available for interacting and chatting.

5.1. SERVING THE NEW MODEL

To interact with your new model, you must activate the model in a machine through serving. The **ilab model serve** command starts a vLLM server that allows you to chat with the model.

Prerequisites

- You installed RHEL AI with the bootable container image.
- You initialized InstructLab.
- You customized your taxonomy tree, ran synthetic data generation, trained, and evaluated your new model.
- You need root user access on your machine.

Procedure

- You can serve the model by running the following command:

```
$ ilab model serve --model-path <path-to-best-performed-checkpoint>
```

where:

<path-to-best-performed-checkpoint>

Specify the full path to the checkpoint you built after training. Your new model is the best performed checkpoint with its file path displayed after training.

Example command:

```
$ ilab model serve --model-path
~/.local/share/instructlab/phased/phase2/checkpoints/hf_format/samples_1945/
```

IMPORTANT

Ensure you have a slash / at the end of your model path.

Example output of the ilab model serve command

```
$ ilab model serve --model-path
~/.local/share/instructlab/phased/phase2/checkpoints/hf_format/<checkpoint>
INFO 2024-03-02 02:21:11,352 lab.py:201 Using model /home/example-
user/.local/share/instructlab/checkpoints/hf_format/checkpoint_1945 with -1 gpu-layers
and 4096 max context size.
```

```
Starting server process
After application startup complete see http://127.0.0.1:8000/docs for API.
Press CTRL+C to shut down the server.
```

5.2. CHATTING WITH THE NEW MODEL

You can chat with your model that has been trained with your data.

Prerequisites

- You installed RHEL AI with the bootable container image.
- You initialized InstructLab.
- You customized your taxonomy tree, ran synthetic data generated, trained and evaluated your new model.
- You served your checkpoint model.
- You need root user access on your machine.

Procedure

1. Since you are serving the model in one terminal window, you must open a new terminal window to chat with the model.
2. To chat with your new model, run the following command:

```
$ ilab model chat --model <path-to-best-performed-checkpoint-file>
```

where:

<path-to-best-performed-checkpoint-file>

Specify the new model checkpoint file you built after training. Your new model is the best performed checkpoint with its file path displayed after training.

Example command:

```
$ ilab model chat --model
~/.local/share/instructlab/phased/phase2/checkpoints/hf_format/samples_1945
```

3. Example output of the InstructLab chatbot

```
$ ilab model chat
╭──────────────────────────────────────────────────────────────────────────────╮
│ system                                                                       │
╰──────────────────────────────────────────────────────────────────────────────╯
╭──────────────────────────────────────────────────────────────────────────────╮
│ Welcome to InstructLab Chat w/ CHECKPOINT_1945 (type /h for help)            │
╰──────────────────────────────────────────────────────────────────────────────╯
```

```
_____
_____
_____
_______J
>>>
[S][default]
```

Type **exit** to leave the chatbot.