DEV1351C
Harnessing Multimodal Data for Enhanced Splunk Analytics - Lab Exercises

Overview

This lab provides hands-on training in using Splunk, Python, and Ollama, which can run large language models (LLMs) locally.
Key learning points include:

- Setup and Configuration: installation of Ollama and LLM models, adding the LLM Parser Splunk Add-on, configuring the Add-on and environment.
- Python Scripting and LLMs: Participants will learn about Splunks Python SDK and leveraging LLMs for image-to-text conversions and explore the basics of LLMs.
- Image Processing: Techniques for extracting meaningful information from images, converting images to text using LLMs, and integrating this process within Splunk.
- Analysis Techniques: Utilize Splunk streaming commands in SPL to analyze data using LLMs.
- Real-World Scenarios: Exercises based on real-life cases, according to Buttercup.

Supplemental Materials: All participants will have access to a GitHub repository containing a custom Splunk Add-on app and sample files for practice in the labs, enhancing the learning experience. It is assumed that each participle has Splunk Enterprise 9 running locally they can modify.

Lab Exercise 1 - Getting to Know Ollama

Description

In this hands-on lab, participants will install and configure Ollama, an application for running large language models. This exercise includes:

- *Installation: Guidance on downloading and installing Ollama from Ollama.com, available for macOS, Linux, and Windows.*
- *Introduction to Ollama usage: Learn to navigate Ollama's commands, such as pull, list, and run, which are essential for managing models.*
- *Running a Model: Participants will pull the latest llama3 model, verify its installation, and execute it to engage with the model interactively.*
- *Interactive Prompts: Explore how to input multiline prompts and retrieve responses, enhancing understanding of dynamic model interaction.*
- *Troubleshooting and Feedback: Troubleshooting tips for common issues and a mechanism for feedback to facilitate continuous learning and support.*

This lab is designed to give participants a thorough introduction to using Ollama, empowering them with the skills to integrate and leverage large language models in their projects.

Steps

1. Download and install Ollama from the following location. https://ollama.com/ Ollama is available for macOS, Linux and Windows.
2. Open a command prompt or terminal.
    a. On Windows, search for “cmd” or “Command Prompt” in the Start menu.
    b. On macOS, you unzip, then move it to the applications folder.
    c. On macOS or Linux, open “Terminal”.
3. Check Installation
    a. Type the command
    `ollama --version`

b. Press Enter. This command should return the version number of Ollama installed on your system, confirming that it is installed correctly.

4. Review Usage - See Appendix for usage
   a. Type the command
      `ollama`
   b. Press Enter. This command should return the usage information of Ollama.
5. Download the “llama3” model
   a. Type the command
      `ollama pull llama3`

      Note: Pulling a model with the ollama pull command may take some time, depending on your internet speed and the size of the model.
   b. This command downloads and adds the llama3 model to your Ollama installation.
   c. This file is over 4.7GB and may take a few mins
6. List the installed model.
   a. Type the command
      `ollama list`
   b. This command should show which models are installed.
7. Try running a model
   a. Type the command
      `ollama run llama3`
   b. You should see a prompt line “>>>”
   c. Type the following at the prompt
      `/?`
   d. You should see the available commands
   e. Type the following prompt
      `What is Splunk?`
   f. Type the following prompt
      `“””`
      `What is`
      `Splunk?`
      `“””`
   g. Type the following prompt to exit ollama.
   h. `/bye`

> Note: We will need the llava-llama3 LLM model for a later lab. Lets go ahead and start the download now. Ollama supports multitasking, so while it's downloading we can proceed in a new terminal.

8. Download the "llava-llama3" model
    a. Type the command
    `ollama pull llava-llama3`
    b. This command downloads and adds the llama3 model to your Ollama installation.
    c. This file is over 4.1GB and may take a few mins

Summary

In this engaging lab, participants learned to install and configure Ollama, an application for running large language models. They navigated essential commands like pull, list, and run, simplifying model management. A key activity was pulling and interacting with the llama3 model, including handling multiline prompts to enhance their understanding of dynamic interactions. Additionally, participants received troubleshooting tips and initiated the download of the llava-llama3 model for future use.

Lab Exercise 2 - Creating a Prompt

Description

This lab teaches participants to craft effective prompts using Ollama. It covers four main areas:

- *Clarity is Key: Learn the impact of detail in prompts through practical examples.*
- *Understand the User Needs: Explore how specifying output formats, like JSON, affects responses.*
- *Refining the Prompt: Refine prompts to include all necessary details for comprehensive responses.*
- *Optimizing Future Prompts: Develop skills to generate optimized prompts for future use.*

Participants will gain proficiency in prompt engineering, crucial for leveraging LLMs in various applications.

Steps

1. Start Ollama
    a. Type the command `ollama run llama3`
2. Compare the following prompts
    a. At the prompt type `Provide information about Buttercup.`
    b. This prompt may reply with information about Buttercup, Splunks mascot. Most likely not.
    c. Update the prompt to mention your talking about Splunks Mascot `Buttercup is a horse mascot for Splunk. Provide information about Buttercup.`
3. Let's improve this prompt even more.

```
Buttercup is a horse mascot for Splunk. Provide
information about Buttercup. Detail some
characteristics about Buttercup.
```

   a. This prompt will provide a more detailed result.

4. Next let's provide a prompt that will create a properly formatted result.

```
Buttercup is a horse mascot for Splunk. Provide
information about Buttercup. Detail some
characteristics about Buttercup.  Output the results
in JSON format for easy integration.
```

   a. This prompt should contain a JSON object.

5. Now, lets ask it to reformat our prompt and include a JSON Template

```
Help me create a prompt that will respond with the
previous results with a JSON template for proper
formatting, including placeholders for JSON values.
```

   a. This will refactor our existing prompt and give a JSON Template for future use

> Note: Your results may be slightly different and OK. Review the prompt and continue.

6. Now we can run this LLM optimized prompt ensuring consistent results.

```
“””
Create a JSON response for Buttercup, the horse mascot
of Splunk. The JSON should include the following
characteristics:
* Name (string): [Buttercup]
* Species (string): [Horse]
* Personality (array of strings):
    + [Friendly]
    + [Approachable]
    + [Enthusiastic]
* Appearance (object):
    + Color (string): [Yellow]
    + Mane (string): [White]
```


```
    + Tail (string): [White]
* Symbolism (array of strings):
    + [Joy of working with data]
    + [Importance of having fun while doing it]

**JSON Template:**

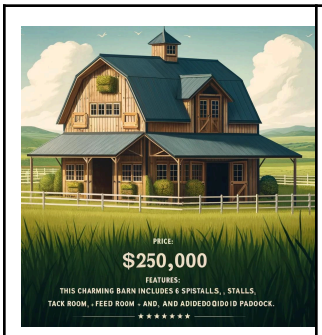
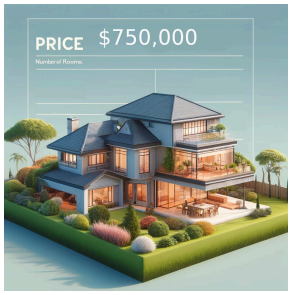
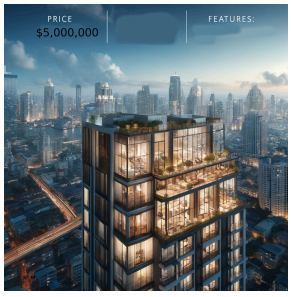
```json
{
  "name": "${Buttercup}",
  "species": "${Horse}",
  "personality": [
    "${Friendly}",
    "${Approachable}",
    "${Enthusiastic}"
  ],
  "appearance": {
    "color": "${Yellow}",
    "mane": "${White}",
    "tail": "${White}"
  },
  "symbolism": [
    "${Joy of working with data}",
    "${Importance of having fun while doing it}"
  ]
}
```

Placeholders:

* `${Buttercup}`: Replace with the actual value for Buttercup's name.
* `${Horse}`: Replace with the actual value for the species (which is Horse, in this case).
```

```
* `${Friendly}`, `${Approachable}`, and `${Enthusiastic}`: Replace with the actual values for personality traits.
* `${Yellow}`, `${White}`: Replace with the actual values for appearance color and mane/tail.
* `${Joy of working with data}` and `${Importance of having fun while doing it}`: Replace with the actual values for symbolism.
"""
```

7. Close out Ollama.
    a. Run the following command to stop Ollama
    `/bye`

Summary

In this lab, participants learned to create effective prompts for LLMs, focusing on clarity, understanding user needs, refining details, and optimizing future prompts. They practiced starting Ollama, comparing different prompts for the mascot Buttercup, refining these prompts for detail and format, and ultimately creating a JSON template for consistent and comprehensive responses. This lab equips participants with essential prompt engineering skills to leverage large language models effectively in various applications.

Lab Exercise 3 - Can it Splunk?

Description

This lab demonstrates how Large Language Models (LLMs) can be utilized to streamline and enhance various Splunk functions, making administration tasks more efficient.

- *Writing Regular Expressions: Participants will use LLMs to create regular expressions for parsing and extracting data from complex log files.*
- *Writing SPL Queries: Participants will learn how to use LLMs to generate complex SPL queries for specific analytical needs, such as monitoring network traffic or user activities.*

By the end of this lab, participants will be proficient in leveraging LLMs to enhance their Splunk environment, reducing the complexity and time spent on routine tasks.

Sample Data

2022-07-15 10:42:03 INFO User called from 321-456-7890 regarding account issues.

2022-07-15 10:45:21 ERROR Connection attempt from 800.555.1234 was blocked.

2022-07-15 10:49:58 DEBUG Incoming call at 2121234567 recorded in the system."system."issues.

FIX - Convert buttercup theme and image (ocr) with regex

Steps

1. Type the command
   `ollama run llama3`
2. At the prompt type
   ```
   """
   Write a regular expression that will match on the
   phone number. It should match all three versions of
   ```


```
the phone number (321-456-7890, 800.555.1234,
2121234567).

Sample data:
"2022-07-15 10:42:03 INFO User called from
321-456-7890 regarding account issues.
2022-07-15 10:45:21 ERROR Connection attempt from
800.555.1234 was blocked.
2022-07-15 10:49:58 DEBUG Incoming call at 2121234567
recorded in the system."system."issues."
"""
```

3. Open the website regex101 at https://regex101.com/
    a. Copy the “Sample Data” above into the “TEST STRING” window
    b. Copy the response from step 2 into the “REGULAR EXPRESSION” window
    c. Verify that each of the phone numbers are matched.

> Note: If not all phone numbers are matched. Follow up with an additional prompt like:
> ”It didn't match on the phone number 2121234567. Try again.”

4. Close out Ollama.
    a. Run the following command to stop Ollama
    `/bye`

Lab Exercise 4 - A picture is worth a thousand words.

Description

This lab teaches participants to use the LLava: Large Language and Vision Assistant multimodal modal running in Ollama for processing and describing images, refining prompts to return results in JSON format. Key activities include:.

- *Download and run the "llava-llama3" Large Language and Vision Assistant model.*
- *Process sample data images.*
- *Refine prompts to generate detailed JSON responses for image descriptions and specific details.*

By the end of this lab, participants will be proficient with processing images and refining prompts to generate detailed JSON responses.

Sample Images

	
buttercup_id.jpg	weather_map.jpg

Steps

1. Download the Splunk app from https://github.com/rbarber68/DEV1351C
    a. Click on the “<> Code” button, select “Download ZIP”
    b. Extract the zip into temp folder
2. Move the llm_parser directory into the Splunk app folder
    a. The Splunk app folder is located on your local machine ($SPLUNK_HOME/etc/apps/)
    b. You can copy the folder using the following
    ```
    cd {temp_folder}/DEV1351C-main
    cp -R llm_parser $SPLUNK_HOME/etc/apps/
    ```
3. Download and Install Splunk SDK Python from https://github.com/splunk/splunk-sdk-python
    a. Click on the “<> Code” button, select “Download ZIP”
    b. Extract the zip into temp folder
4. Move the splunklib directory into the llm_parser lib folder
    a. The llm_parser lib folder is located on your local machine ($SPLUNK_HOME/etc/apps/llm_parser/lib/)
    b. You can copy the folder using the following
    ```
    cd {temp_folder}/splunk-sdk-python-master
    cp -R splunklib $SPLUNK_HOME/etc/apps/llm_parser/lib
    ```
5. You should restart Splunk
    ```
    $SPLUNK_HOME/bin/splunk restart
    ```
6. Change directory into the image folder
    a. The images are located into the llm_parser/appserver/static/images folder
    ```
    Cd $SPLUNK_HOME/etc/apps/llm_parser/appserver/static/image
    ```
7. Type the command
    ```
    ollama run llava-llama3
    ```
8. Let's look at sampleimage
    a. At the prompt type
    ```
    ./buttercup_id.jpg
    ```
9. The response should mention the following, “Added Image ‘./buttercup_id.jpg”, then provide a description of the image.
10. Lets refine our prompt to return JSON.
    ```
    ./buttercup_id.jpg  What are the details in this image? Only reply with information in the image. I only need a name and address.
    ```

```
Your response should be in JSON
```

11. The response should mention the following, “Added Image ‘./buttercup_id.jpg”, then a contain a json with Name and Address
12. Let's refine the prompt one more time.

    ```
    rewrite the previous prompt and include a JSON
    template with placeholders.
    ```

13. Review the revised template and edit to create

    ```
    ./buttercup_id.jpg
    Please provide the name and address of the person in
    the image of an ID card.
    The JSON template with placeholders is as follows:

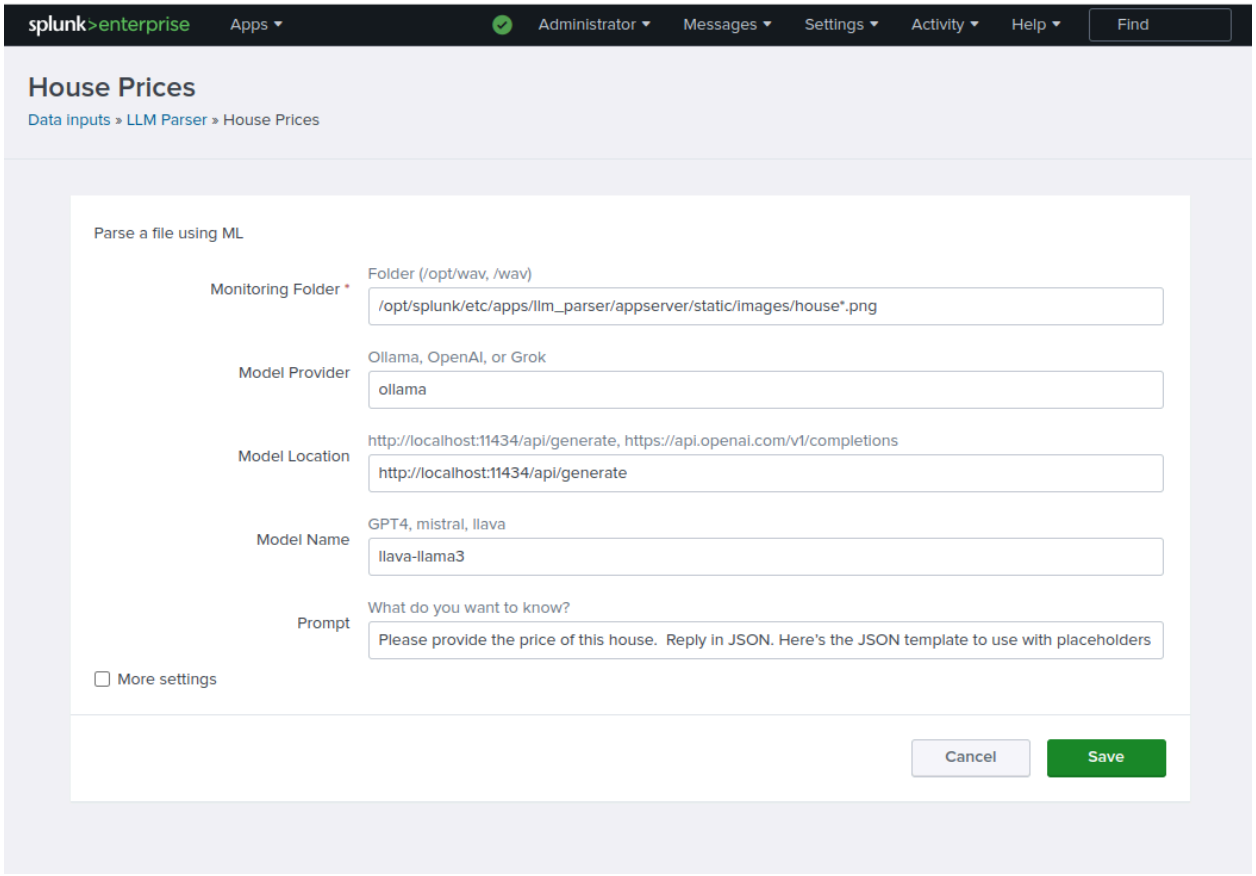
    ```json
    {
    "Name": "[NAME]",
    "Address": "[ADDRESS]"
    }
    ```
    Please fill in the placeholders with the appropriate
    values for each field.
    ```

14. Let's look at another image, at the prompt type

    ```
    ./city_map.jpg
    ```

    a. The response should mention the following, “Added Image ‘./city_map.jpg”, then provide a description of the image.
15. Lets refine our prompt to find out how many states are in the map.

    ```
    ./city_map.jpg How many states are in the map?
    ```

    a. It should reply with ~8
16. Lets refine our prompt to find out how many states are in the map.

    ```
    ./city_map.jpg What are the state names?
    ```

    a. It should reply with list of all the state names.
17. Lets refine our prompt to find out the state names and their temperatures in the map and reply in JSON.

    ```
    ./city_map.jpg  What is the temperature for each of
    the 8 states?  Reply in json format.  Make sure to
    include all 8 states?
    ```

    a. It should reply with a list of all the state names and temps.

18. Lets ask it to review our prompt and create a JSON template.

```
rewrite the previous prompt to make it better and
include a JSON template for the output with
placeholders for each state and temp.
```

19. Let's review our final refined prompt.
    a. Review the refined prompt and make changes if necessary.

```
./city_map.jpg
Please provide the temperature for each of the 8
states: Breezeburg, Sunnyville, Unawtown, Rainytown,
Stanmargus, Cloudyale, Breezburgh and
Sunville.

Here's the JSON template with placeholders:
```json
{
    "Breezeburg": "[Temperature]",
    "Sunnyville": "[Temperature]",
    "Unawtown": "[Temperature]",
    "Rainytown": "[Temperature]",
    "Stanmargus": "[Temperature]",
    "Cloudyale": "[Temperature]",
    "Breezburgh": "[Temperature]",
    "Sunville": "[Temperature]"
}
```
Replace the `[Temperature]` placeholders with the
respective temperatures for each city.
```

20. The response should mention the following, “Added Image ‘./city_map.jpg”, then provide details in json on the cities and temperatures.

Summary

In this lab, participants learned to use the LLava (Large Language and Vision Assistant) multimodal model running in Ollama for processing and describing images, refining prompts to return results in JSON format. Key activities included downloading the LLM Parser, Splunk Python SDK, and running the llava-llama3 model, processing sample images (buttercup_id.jpg, and city_map.jpg), and refining prompts to generate detailed JSON responses for image descriptions and specific details. By the end of the lab, participants will be proficient in processing images and crafting prompts to produce structured JSON outputs.

Lab Exercise 5 - What's happening?

Description

This lab demonstrates how we can leverage Splunk's Modular Inputs to call and pass data to LLMs. This provides a way to automate processing of multimodal inputs.

- *Walk through installing the LLM Parser Splunk Add-on.*
- *Configure the LLM Parse app to process sample data.*
- *Review processed data and evaluate the data*

By the end of this lab, participants will be proficient in leveraging LLMs to enhance their Splunk environment, reducing the complexity and time spent on routine tasks.

Sample Images

			
house_barn.jpg	house_suburbs.jpg	house_city.jpg	house_stable.jpg

Steps

1. Let's review the sample images to create a good prompt
2. Type the command
   `ollama run llava-llama3`
3. At the prompt type

```
./house_suburbs.png
```

4. The response should mention the following, “Added Image ‘./house_suburbs.png”, then provide a description of the image.
5. Lets refine our prompt to return JSON.

```
./house_suburbs.png How much is this house? Reply in JSON format.
```

6. Lets refine our prompt to return JSON.

```
rewrite the previous prompt to make it better and include a JSON template for the output with placeholders for the price.
```

7. The final prompt should look something like the following.

```
./house_suburbs.png Please provide the price of this house.  Reply in JSON.
Here’s the JSON template to use with placeholders

```json
{
  "price": "{price}"
}
```
Replace the {price} with the price of the house.
```

8. Now lets create a Data Input
   a. Go into Splunk, click on “Settings”, in the top right
   b. Under DATA, click on “Data Inputs”
   c. Scroll thru the list of “Local inputs”, click on “LLM Parser”.
   d. Click on the “New” Green Button in the top right.
9. Enter the following:

```
Name:House Prices
Monitoring Folder:
$SPLUNK_HOME/etc/apps/llm_parser/appserver/static/images/house*.png
Model Provider:      ollama
Model Location:      http://localhost:11434/api/generate
Model Name:          llava-llama3
Prompt:              Please provide the price of this house. Reply in JSON.
```

```
Here's the JSON template to use with placeholders

```json
{
  "price": "{price}"
}
```
Replace the {price} with the price of the house.
```

a. Click "Next"
b. Click "Start Searching" button



10. From a Splunk search run the following
    `index=main sourcetype=llm_parser`
11. You should see events for each of the 4 images with price information.

Summary

This lab focused on leveraging Splunk's Modular Inputs to call and pass data to Large Language Models (LLMs) for automated processing of multimodal inputs. Participants installed the LLM Parser Splunk Add-on, configured it to process sample images, and reviewed the processed data. The lab included creating prompts for image data, refining them to return JSON responses, and setting up a Data Input in Splunk to monitor a folder of images and retrieve price information. By the end, participants were proficient in using LLMs to enhance their Splunk environment, simplifying complex tasks and reducing time spent on routine processes.

Lab Exercise 6 - SPL Integration

This lab demonstrates how we can leverage Splunks streaming commands to call LLMs using SPL. We will look at how SPL can pass arguments to LLMs and return results. This provides a way to interactively call LLMs from SPL.

- *Review how to pass arguments to the LLM*
- *Review returned values from the LLM.*
- *Walk thru the llm_prompt python code.*

By the end of this lab, participants will be proficient in leveraging LLMs to enhance their Splunk environment, reducing the complexity and time spent on routine tasks.

Steps

1. This lab assumes that you've installed the LLM Parser Splunk Add-on from Lab 5.
2. Verify the LLM Parser app is installed.
    a. Click on "Apps" dropdown, on the top left of a Search
    b. Verify the app "LLM Parser" is in the list of apps
3. In a new search window type the following

```
| llmprompt model=llama3 prompt="what is Splunk's mascot Buttercup favorite food?"
```

    a. The result should be a single line with a response.
    b. Review the response
4. Review the parameters
    a. Model: The name of the ollama model you wish to call "llama3"
    b. Prompt: The prompt info to run.
5. Next will wll look at SPL Streaming commands that will allow us to pipe results into the LLM for processing.
6. In a new search window type the following

a. First, lets verify we have atleast one event in that will view. Adjust the time range to find atleast one event.

```
index=_internal log_level
| head 1
```

b. Now lets translate after fields into other languages using the llmstream SPL streaming command and return the results.

```
index=_internal log_level
| head 1
| llmstream evalfieldname=log_level prompt="translate
the following into Imaginary horse Language.  Only
reply with horse speak" model=llama3
fieldname=log_level_horse
| llmstream evalfieldname=log_level prompt="translate
the following into Spanish.  Only reply with the
Spanish translation" model=llama3
fieldname=log_level_spanish
```

Summary

In this Lab we focused on integrating Splunk with Large Language Models (LLMs) using Splunk's streaming commands, enabling interactive LLM calls directly from SPL. Participants learned how to pass arguments to LLMs, handle returned results, and walk through the Python code. By the end of the lab, participants will be proficient in utilizing LLMs to enhance their Splunk environment, streamlining complex and routine tasks.

Appendix

Ollama usage info

```
$ ollama
Usage:
  ollama [flags]
  ollama [command]

Available Commands:
  serve       Start ollama
  create      Create a model from a Modelfile
  show        Show information for a model
  run         Run a model
  pull        Pull a model from a registry
  push        Push a model to a registry
  list        List models
  ps          List running models
  cp          Copy a model
  rm          Remove a model
  help        Help about any command

Flags:
  -h, --help      help for ollama
  -v, --version   Show version information

Use "ollama [command] --help" for more information about a command.
```

Summary

In this lab, we delved into … We discovered how to …. These exercises actualize the activities our security customers conduct in their real-world environments, giving us …

Happy Splunking!