AI Wargaming

Release 0.4

Peter Asjes, Henry Frye, Caleb Koutrakos, Will Robinson, and Bol

Feb 18, 2025

CONTENTS:

AI Wargaming is a undergraduate capstone project part of a larger initiative to modernize wargaming using AI.

> **Note**
>
> This project is under active development.

CHAPTER

ONE

USAGE

1.1 Installation

1.1.1 Download the Code

The code for this project can be downloaded via GitHub.

```
$ git clone https://github.com/sideoffryes/AI_Wargaming_Capstone.git
```

1.1.2 Python Management

To use this project, the are several prerequisites that are necessary. The easiest way to manage these dependencies is using pip and a virtual environment.

If you do not have python already, download and install a release of python 3 for your platform.

To ensure that pip is available on your system, follow these instructions for your platform.

All of the required python packages can be easily installed via the provided configuration files and setup script. There are separate files for GPU and CPU dependencies.

Setup Script

```
$ ./setup.sh
```

The setup script will ask if you would like to install the CPU or GPU configuratio and install the appropriate configuration.

Once the setup is complete, make sure to activate the virtual environment.

```
$ source .venv/bin/activate
```

1.1.3 Hugging Face

The project accesses the Llama 3.1, 3.2, and 3.3 families from Meta via Hugging Face. Running this project requires a Hugging Face account and access to those families.

1. Visit the page for the model on Hugging Face. For example, Llama-3.2 (1B)
2. Create a free account and login.
3. Return to the Llama webpage (if not already there).
4. You should see a Community License Agreement at the top. Click the "Expand to review" button:
5. If you agree with the terms, fill out the form

6. Check email later.

Once you have received access to the models, visit your tokens page and click "Create new token". Choose the "Read" token type at the very top. Then click "Create token". Copy the generated string.

In the terminal, run the following command and paste in your access token when prompted:

```
(.venv) $ huggingface-cli login
```

1.2 Running the Project

Before attempting to run any of the scripts, make sure that you have the correct Conda environment activated.

```
(.venv) $ conda activate capstone_gpu
```

All of the code that runs the webserver and actually generated the documents can be found inside of the *capstone* directory. *app.py* is the webserver and *docgen.py* is the script that accesses the LLM to generate documents. Running the entire project can be accomplished with the following:

```
(.venv) $ cd capstone
(.venv) $ python3 app.py
```

The webserver can be reached from your browser by using one of the ip addresses printed out in the terminal when the server is created.

1.3 Interacting with the Webserver in the Browser

The form presented to you when the website is first loaded can be used to generate a document. Use the *selection options* dropdown menu to select the type of document that you would like to create. You can specify your requirements and any additional specifications in the *additional parameters* textbox.

Depending on the size of the model used to generate the document, the server may load for a few minutes before the final output is produced.

1.4 Generating the Docs

The repository is shipped with a precompiled PDF version of the documentation for the entire project for both users and developers.

The HTML documentation that can be viewed from the browser when running the webserver can be created by cding into the docs directory and using the make file.

```
(.venv) $ cd docs
(.venv) $ make html
```

The generated documentation will appear in the docs/build/html directory.

CHAPTER

TWO

API

2.1 docGen

Functions

`gen`(model_num, type_num, prompt[, save])	Generates a specificed document using a specified LLM and returns the result.
`load_examples`(type)	Returns real life examples of the requested document type.
`save_response`(response, prompt, model_name, ...)	Writes the response and information about how it was generated to the disk.
`select_doc`(num)	Translates between the numeric representation of a document type and its full name.
`select_model`(num)	Translates between the numeric representation of a model to the full string of its name.

2.2 app

Functions

createAcc()
docs(filename)
handle_indexPost()
handle_loginPost()
handle_registerPost()
home()
index()
login()
logout()
my_artifacts()
userprofile()

Classes

GeneratedArtifact(**kwargs)
Profile(**kwargs)

2.3 pdfToText

Functions

pdf_to_text(pdf_path, output_txt)

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX