MLOps - Dyula to French Machine Translation Deployment

1 Objective

This document details the deployment of a machine learning model using Highwind. The primary goal is to demonstrate proficiency in MLOps deployment, with the machine learning algorithm being a Dyula-to-French translation model.

2 Deployment

The deployement involves using Docker, Kserve and Highwind. The deployment files used in the deployment (Dockerfile, docker-compose.yml, input.json, main.py, model_card, and serve-requirements) were cloned from theGitHub Repository under the translate-dyu-fr-hugging-face repository.

2.1 Deployment Commands

The model used was a trained on Google Colab which was then exported and downloaded locally. The file was then copy in the saved files directory.

To build the Docker container and give it a tag, use the following command:

```
docker build −t local/highwind−examples/dyu−fr−inference:latest .
```

After building the Kserve predictor image, the model inference was tested locally by building the Docker container and giving it a tag.

```
docker compose up −d
docker compose logs
```

The following command is used to send a payload to the model to test its response. The script reads a JSON file, sends it as a request to a model's inference endpoint, and processes the response by converting it into a PowerShell object. Finally, it outputs the response back in JSON format, ensuring it handles nested data correctly with the specified depth.

```
$json = Get−Content −Raw −Path ./input.json
$response = Invoke−WebRequest −Uri http://localhost:8080/v2/models/model/
    infer −Method Post −ContentType 'application/json' −Body ([System.Text.
    Encoding]::UTF8.GetBytes($json))
$responseObject = $response.Content | ConvertFrom−Json
$responseObject | ConvertTo−Json −Depth 10
```

2.2 Deployment on Highwind

An asset on the Highwind platform was created, specifying the Name of the Docker Image Repository, Asset Type, and Docker Image Type. To push to the created asset, the instructions shown in Figure 1 were used.
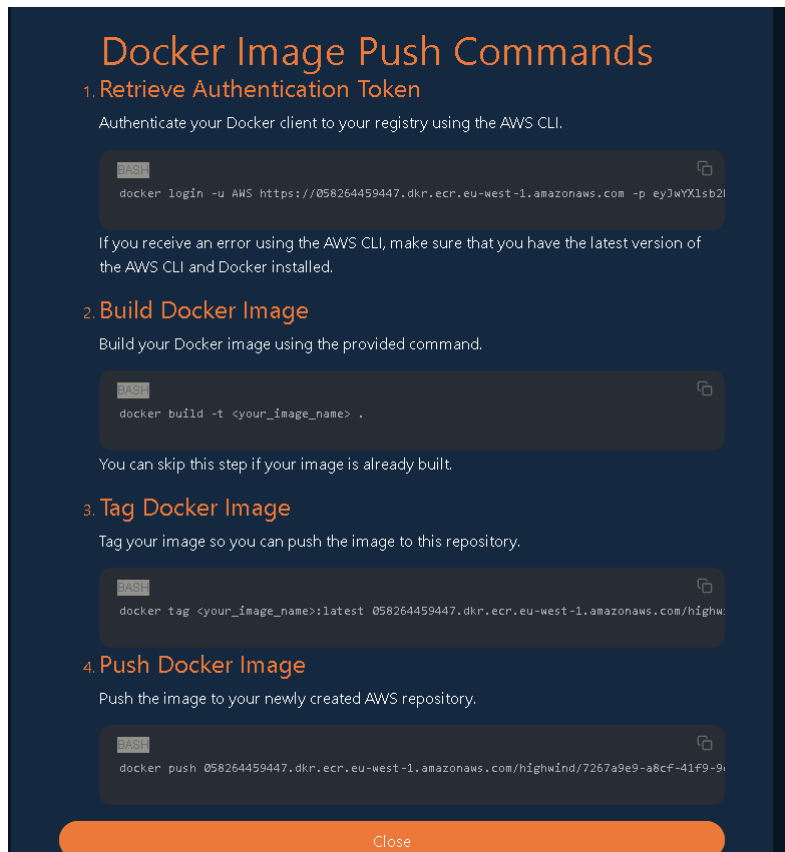
Figure 1: Commands Required to Deploy to Highwind Asset

Authenticating my Docker client to the registry using AWS CLI proved challenging on Windows. The following error occurred when trying to log in.

```
Error saving credentials: error storing credentials — err: exit status 1,
    out: "The stub received bad data."
```

The possible solutions found in the Highwind Documentation's FAQ section did not resolve the issue. Method 1 caused the following error when executed, as seen in Figure 1. This seemed to crash Docker, and resetting to factory settings would restore the configuration file to its original state. Method 2 involved re-entering the deleted line after removing it on Docker startup.
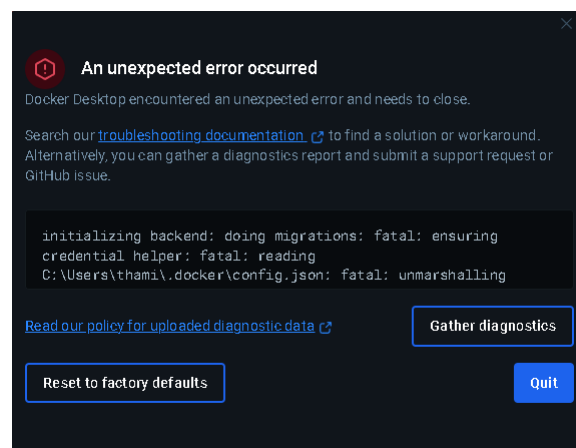

Figure 2: Your image caption here

The solution to this issue is an extension of Method 2. By following a suggestion from StackOverflow, removing the docker-credential-wincred.exe from the resource/bin directory, along with deleting the

credsStore line, seemed to resolve the problem.

Following the rest if the commands the Docker image was tagged then pushed into HighFlow asset created earlier.

A use case with the name "Dyi-to=Fre" was created with input json is as follows

fontspec Arial

```
{
    "inputs": [
        {
            "name": "input-0",
            "shape": [1],
            "datatype": "BYTES",
            "parameters": null,
            "data": [i t g   bi cogod ]
        }
    ]
}
```

The auto generated inference schema from this gave errors when trying to compute an inference on the model. The inference schema was edited to the following:
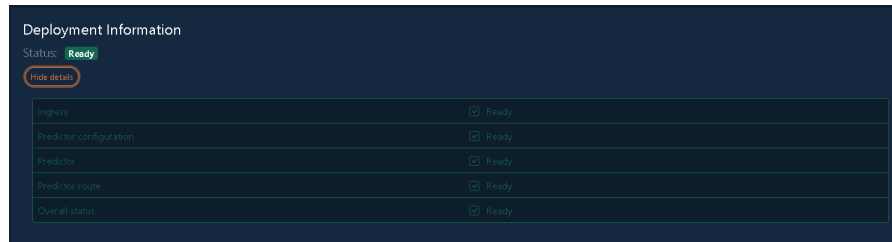
```
{
  "$schema": "http://json-schema.org/draft-07/schema",
  "title": "Generated schema for Root",
  "type": "object",
  "properties": {
    "inputs": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "name": {
            "type": "string"
          },
          "shape": {
            "type": "array",
            "items": {
              "type": "number"
            }
          },
          "datatype": {
            "type": "string"
          },
          "data": {
            "type": "array",
            "items": {
              "type": "string"
            }
          }
        },
        "required": [
          "name",
          "shape",
          "datatype",
          "data"
        ]
      }
    }
  },
  "required": [
```

```
    "inputs"
  ]
}
```

The deployment information gave the following output in Figure 3

Figure 3: Deployment Information

Running compute on the input schema gave the following output:

```
{
  "model_name": "bo6r9fph",
  "model_version": null,
  "id": "2351884e-57bd-4b3c-a5c8-d1427302f3a1",
  "parameters": null,
  "outputs": [
    {
      "name": "output-0",
      "shape": [
        1
      ],
      "datatype": "STR",
      "parameters": null,
      "data": [
        "tu m int resses"
      ]
    }
  ]
}
```


Figure 4: Input JSON

Figure 5: Output JSON

Figure 6: Input and output of Compute on Use Case

4

3   Implemented Machine Learning Algorithm

The machine learning algorithm used was the Dyula-to-French translation model. The overall approach was similar to the examples repository, with a few differences in hyperparameters. The batch size was adjusted to 32 to better utilize the 15GB T4 GPU on Google Colab. Additionally, the number of epochs was increased to 40. There were issues obtaining the Hugging Face dataset from data354/Koumankan_mt_dyu_fr, so the workaround was to download the dataset from the Koumankan4Dyula: Parallel Dyula-French Dataset for Machine Learning asset on Highwind. The total training time (wall time) was 50 minutes and 47 seconds. The dataset used did not include a test set, meaning the model was not evaluated using a separate test dataset. However, an attempt to test the model with an example was made using Google Translate, as shown in Figure 7. As seen, the model is prone to providing incorrect translations.
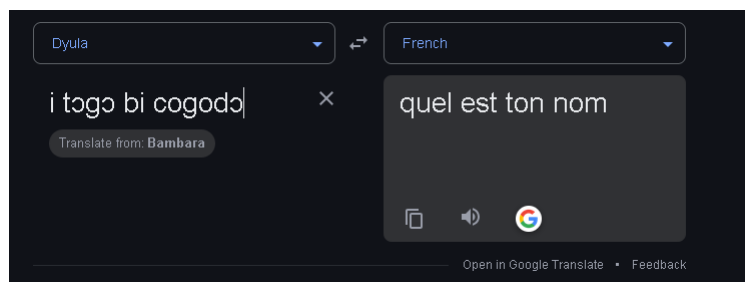

Figure 7: Google Translation prompt

4   Code Repository

4.1   Repository Structure

```
deployement/          # Deployement Scripts
notebooks/            # Highwind deployment scripts
saved_model/          # Output of the training model
Documentation/        # Contains this file in pdf
README.md
```

4.2   Future Work

Having a test dataset is crucial for properly verifying the model's performance. Additionally, the dataset used for training is relatively small compared to other translation datasets, such as the English-Hindi dataset, which contains 1,659,083 rows, whereas the given dataset is significantly smaller.

The training process is not automated and requires manually transferring files from Google Drive to local storage and then to the deployment folder. Automating the pipeline from model training to Highwind deployment should be considered to streamline the workflow.

Exploring alternative models, such as different transformers, could enhance the translation system. While the current approach was based on a specific example, expanding the search to include other models may improve performance.

5   Submission Links

- Use Case Page
- Model (In Google Drive)

6   Resources Used

- Melio AI Example

- HighWind Documentation

- Stack Overflow Suggestion

- GeeksforGeeks Machine Translation with Transformer in Python