Deployment Documentation

1. User Guide

1.1 Overview

This is the user interface of the intelligent log system designed in this paper, which includes five functions: upload, extraction, report, analyse and answer, as shown in the following figure.

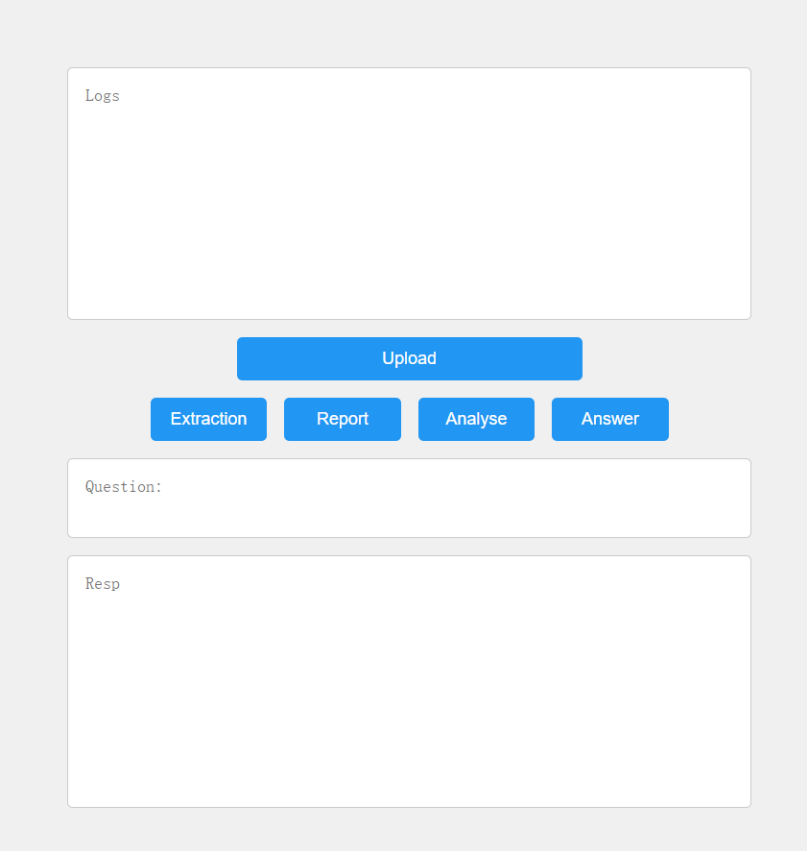

Fig.1 ILS User Interface

When using our intelligent logging system, the first button the user uses is: Upload, which is used to upload and preprocess the log content to be analyzed. When uploading, the input log is limited to 512 tokens, and the system will automatically truncate the input over the character limit. Preprocessing is used to filter the input content, which has been described in detail in the data preprocessing stage. Here's an example of the input log content we used to demonstrate the functionality:

```
[INFO][server.go:45] 2023/05/11 00:04:58 bind: 127.0.0.1:6399, start listening...
[INFO][server.go:77] 2023/05/11 00:05:21 accept link
[INFO][server.go:58] 2023/05/11 00:08:24 get exit signal
[INFO][server.go:62] 2023/05/11 00:08:24 shutting down...
[ERROR][server.go:72] 2023/05/11 00:08:24 listener accept err: accept tcp 127.0.0.1:6399: use of closed network
connection
[INFO][server.go:108] 2023/05/11 00:08:24 handler shutting down...
[INFO][server.go:74] 2023/05/11 00:08:24 connection closed: 127.0.0.1:60042
[INFO][server.go:82] 2023/05/11 00:08:24 disconnect link
[INFO][server.go:45] 2023/05/11 00:12:41 bind: 127.0.0.1:6399, start listening...
[INFO][server.go:77] 2023/05/11 00:12:59 accept link
[INFO][server.go:74] 2023/05/11 00:13:06 connection closed: 127.0.0.1:60287
[INFO][server.go:82] 2023/05/11 00:13:06 disconnect link
[INFO][server.go:58] 2023/05/11 00:13:23 get exit signal
[INFO][server.go:62] 2023/05/11 00:13:23 shutting down...
[ERROR][server.go:72] 2023/05/11 00:13:23 listener accept err: accept tcp 127.0.0.1:6399: use of closed network
connection
[INFO][server.go:108] 2023/05/11 00:13:23 handler shutting down...
[INFO][server.go:45] 2023/05/11 00:13:38 bind: 127.0.0.1:6399, start listening...
[INFO][server.go:58] 2023/05/11 00:13:42 get exit signal
[INFO][server.go:62] 2023/05/11 00:13:42 shutting down...
[ERROR][server.go:72] 2023/05/11 00:13:42 listener accept err: accept tcp 127.0.0.1:6399: use of closed network
connection
[INFO][server.go:108] 2023/05/11 00:13:42 handler shutting down...
[ERROR][main.go:46] 2023/05/11 00:14:51 listen tcp 175.178.59.92:6399: bind: can't assign requested address
```

Fig.2 Input log file

1.2 Extraction

After uploading word limit and filtering processing, the following four steps of Extraction, Report, Analyse and Answer can be carried out.

Etraction part: General log error information (EEROR) scattered in a large number of normal information (INFO), manual search error information needs a lot of browsing time. So we designed the extraction function. After filtering the content, the line extraction is completed, and the ERROR line and the normal information line are extracted. The input and output are as follows:

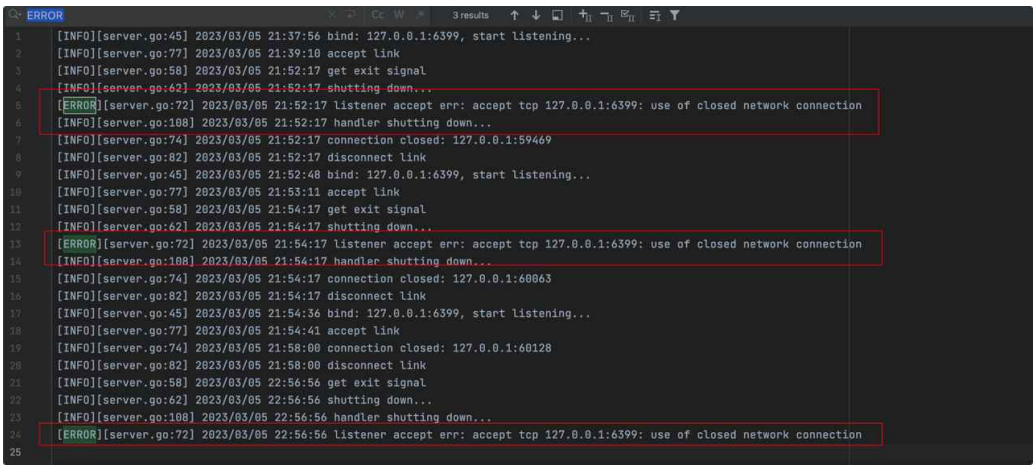

Fig.3 ERROR in log file

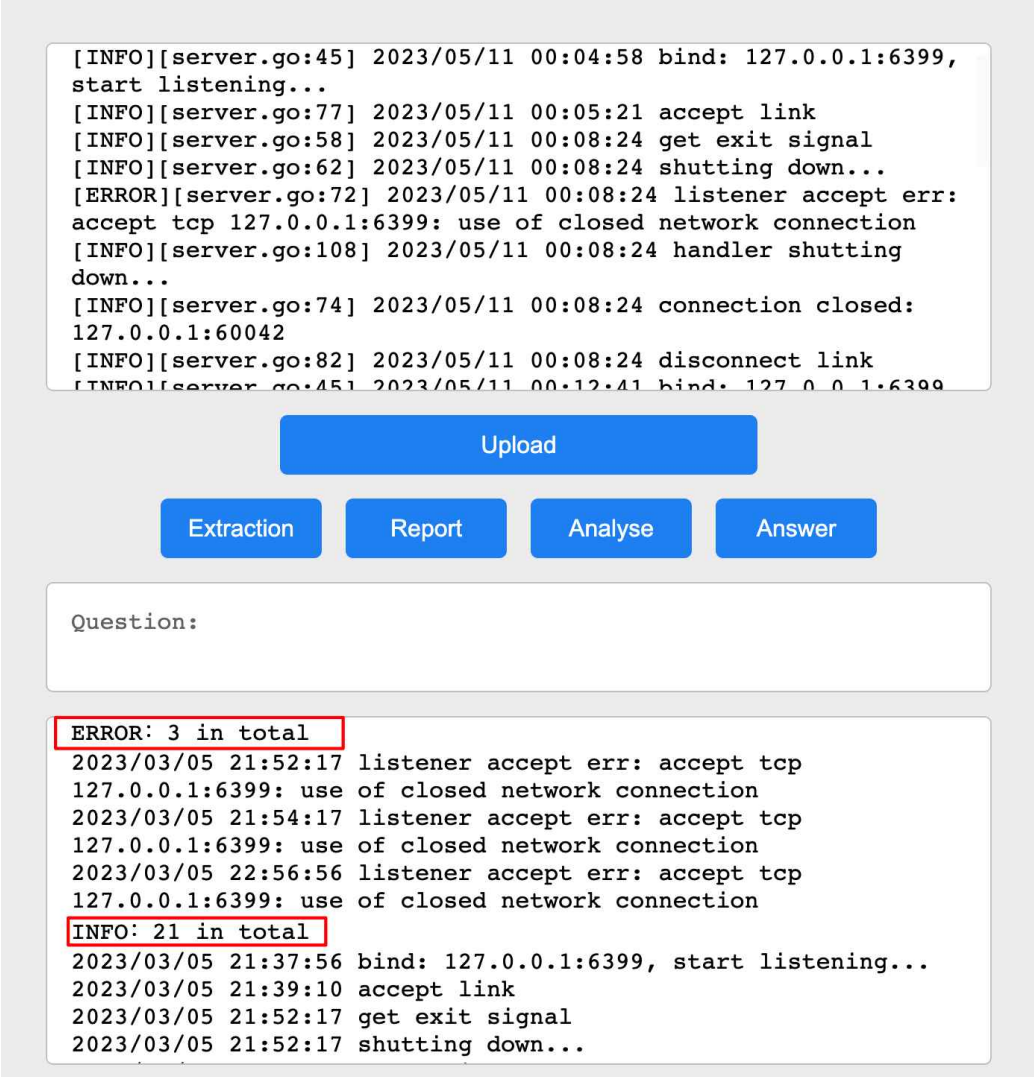

Fig.4.4 Extraction output

The output is shown in the figure above.

1.3 Analyse

Click the analyse button, and the system will automatically study all the error lines and analyze them to give possible explanations. As shown below:

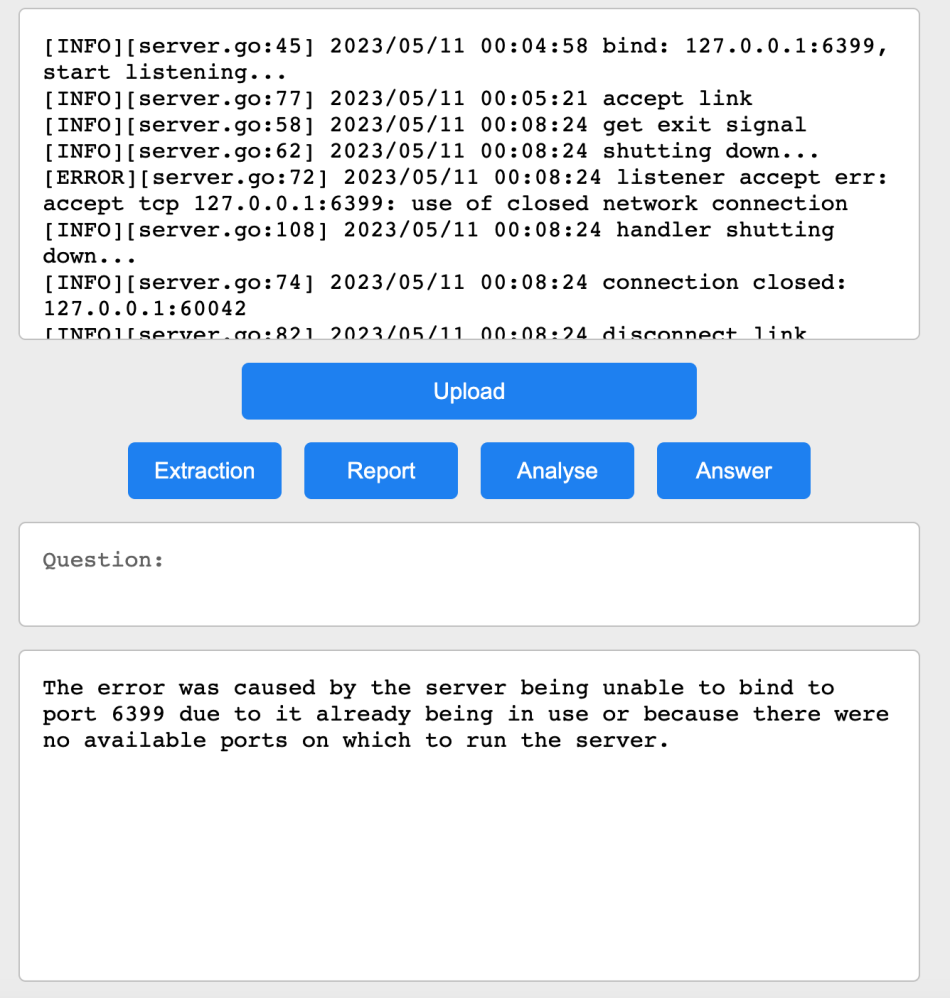

Fig.4.5 Analyse output

1.4 Report

The Report feature generates a more general text report that is emailed to developers to alert them. Since the data set of the model contains text training data related to emails, we will also consider sending the generated report content to the corresponding developers in the format of emails.

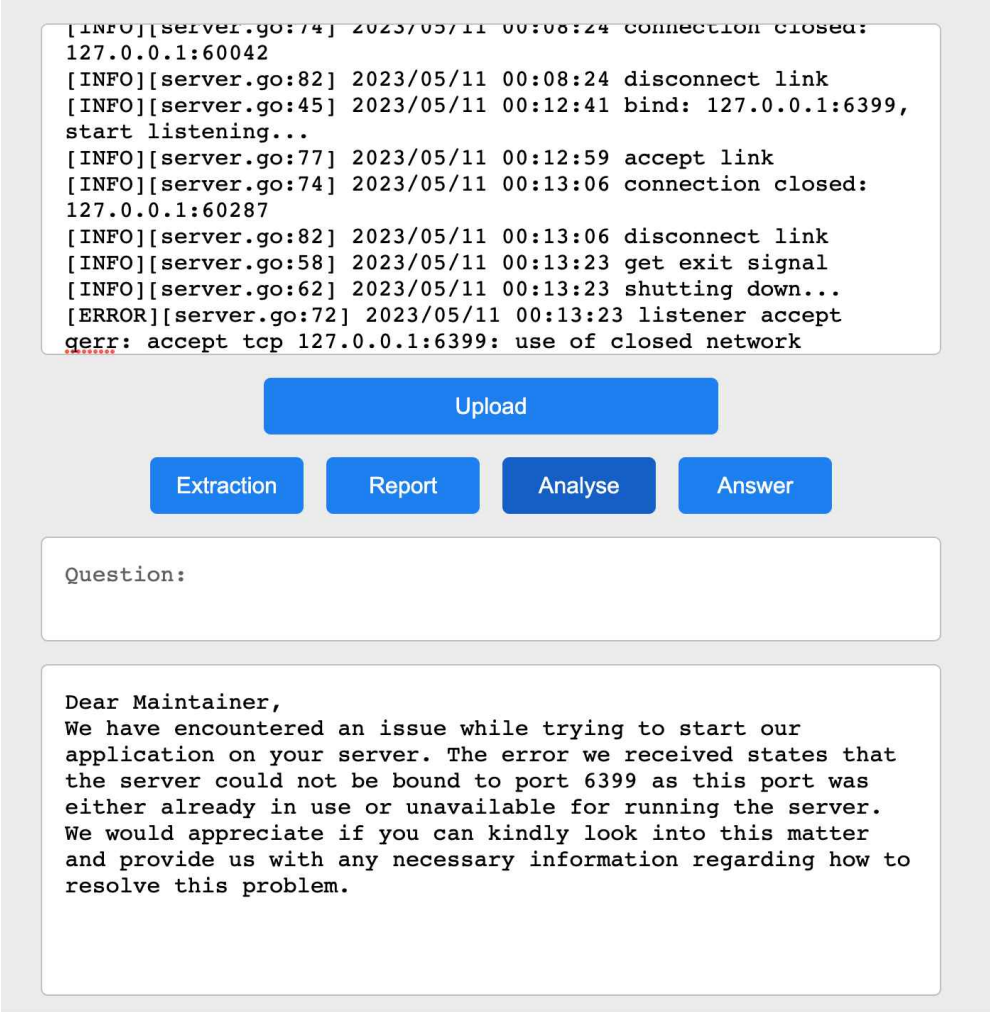

Fig.4.6 Report output

1.5 Answer

The user can ask the logging system questions, such as: What are the possible solutions to this problem? Seek system solutions and more explanations for specific problems.

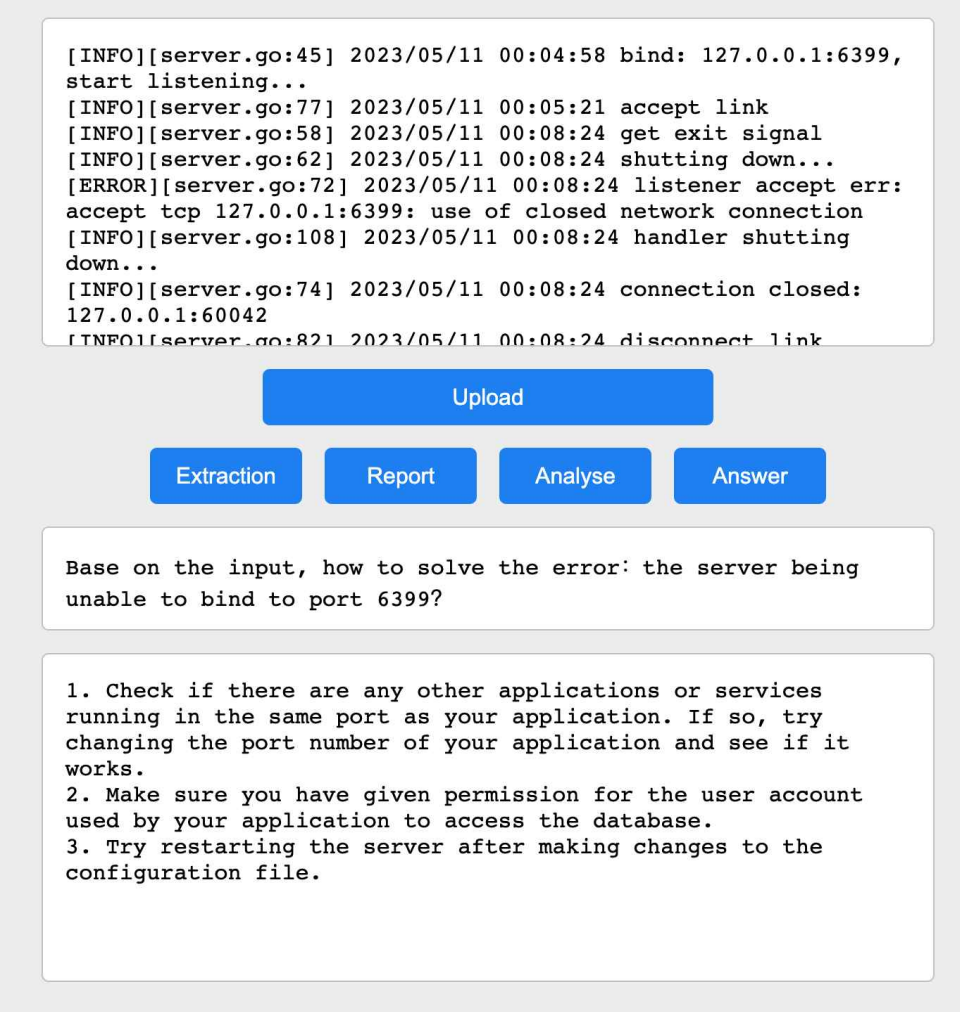

Fig.4.7 Answer output

2. Deployment

2.1 Backend

For the backend, we use jupyter notebook to run our model on colab, and we use the Python web framework Flask to build a server so that users can interact with the model by calling the server api. At the same time, in order to realize the separation of frontend and backend, we use the Ngrok component library to realize the inner network penetration, and map the colab local ip address and port to the WAN, so that the front end can call the server api in the WAN, so as to realize the various functions of the system.

To start the backend code, we need to run all the code blocks of the `EBA5004_ILS\logqa_deploy.ipynb` file.

```
!pip install -q datasets loralib sentencepiece
!pip uninstall transformers
!pip install -q git+https://github.com/zphang/transformers@c3dc391
!pip -q install git+https://github.com/huggingface/peft.git
!pip -q install bitsandbytes
!pip install flask-ngrok flask pyngrok


# import locale
# def getpreferredencoding(do_setlocale = True):
#     return "UTF-8"
# locale.getpreferredencoding = getpreferredencoding
!ngrok authtoken '2DfVh33cQ2h24Owm6UvB3tHsKzw_7tiQw7VghWLbyF6QYihC5'
!pip install -U flask-cors
```

First, run the code block that installs the necessary component libraries and set your own ngrok secret key.

```
from peft import PeftModel
from transformers import LLaMATokenizer, LLaMAForCausalLM, GenerationConfig
import textwrap


tokenizer = LLaMATokenizer.from_pretrained("decapoda-research/llama-7b-hf")

model = LLaMAForCausalLM.from_pretrained(
    "decapoda-research/llama-7b-hf",
    load_in_8bit=True,
    device_map="auto",
)
# model = PeftModel.from_pretrained(model, "samwit/alpaca7B-lora")
model = PeftModel.from_pretrained(model, "shadow006/logqa-alpaca-lora-7b")
```

Then, run the code block to import the dependencies, load the pretrained model and our fine-tuned model.

After that, we run all the core methods of the code block(Model, Extraction, Log filtering, etc.) and finally deploy the Flask server.

```
...  * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server
instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
 * Running on http://6575-35-197-29-243.ngrok-free.app
 * Traffic stats available on http://127.0.0.1:4040
```

After running, copy this address to the corresponding position of the front-end to complete docking.

Please note, when pasting to the frontend, the url `"http"` needs to be changed to `"https"`, otherwise will confront the Cross-Origin Resource Sharing (CORS) issue.

2.2 Frontend

Since our frontend is Flask-based deployment, we first need to install the flask framework in advance.

```
79
80      const base_url = "https://d66b-35-239-50-229.ngrok-free.app";
81      let uploadedText = '';
82
```

Next, navigate to `EBA5004_ILS\web\templates`, open the `index.html` file in your editor, and change the url on line 80 to match the url on your server.

```
PS E:\data\workspace\python\plp\EBA5004_ILS\system\web> flask run
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

After that, flask was run in the `EBA5004_ILS\web` directory and the front-end web page was successfully opened by clicking the url of the console.

Finally, please enjoy it!