KNEO 300 DEVELOPER HANDBOOK V1.2

(For Developer)

May, 2024

Revision History :

version	description	date
1.0	Initial version	2024/03/07
1.1	Update FW v0.16.0 load balance API	2024/05/10
1.2	Update FW v0.16.2 load balance API	2024/08/02

Notice:

1. Kneron (Taiwan) Co., Ltd may make changes to any information in this document at any time without any prior notice. The information herein is subject to change without notice.

2. THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT ANY WARRANTY OR CONDITION OF ANY KIND, EITHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, ANY WARRANTY OR CONDITION WITH RESPECT TO MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR NON-INFRINGEMENT .KNERON DOES NOT ASSUME ANY RESPONSIBILITY AND LIABILITY FOR ITS USE NOR FOR ANY INFRINGEMENT OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE.

3. Information in this document is provided in connection with Kneron products.

4. All referenced brands, product names, service names and trademarks in this document are the property of their respective owners.

Table of contents

Table of Figures

1. Introduction to the Architecture

1.1. Standalone Chatbot structure

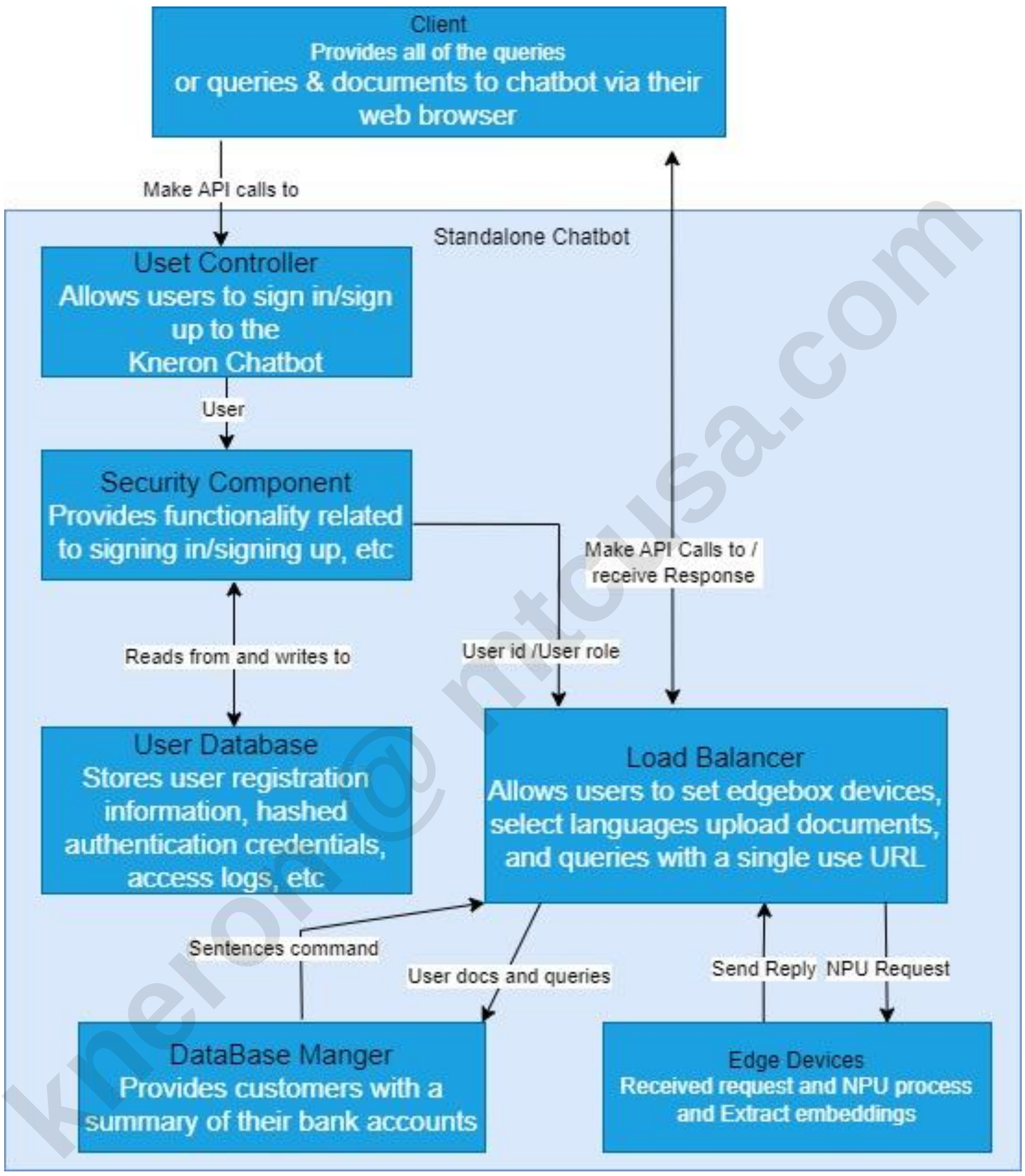

Client
Provides all of the queries
or queries & documents to chatbot via their web browser
Make API calls to
Standalone Chatbot
Uset Controller
Allows users to sign in/sign up to the Kneron Chatbot
User
Security Component
Provides functionality related to signing in/signing up, etc
Make API Calls to / receive Response
Reads from and writes to
User id /User role
User Database
Stores user registration information, hashed authentication credentials, access logs, etc
Load Balancer
Allows users to set edgebox devices, select languages upload documents, and queries with a single use URL
Sentences command
User docs and queries
Send Reply
NPU Request
DataBase Manger
Provides customers with a summary of their bank accounts
Edge Devices
Received request and NPU process and Extract embeddings

Figure 1—1 Standalone Chatbot structure.

1.1.1. Advantages:

1.1.1.1. Autonomy: They can operate independently without the need for integration with other systems or platforms.

1.1.1.2. Privacy and Security: Since they don't rely on other platforms, there's often more control over data privacy and security.

1.1.2. Disadvantages:

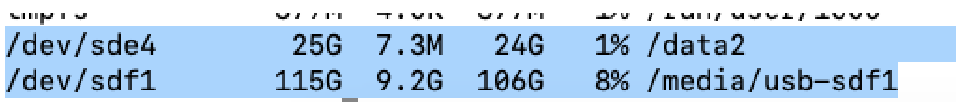
1.1.2.1. Scalability: Scaling up may limited by disk space and memory resources

1.1.3. External disk mount location as shown in the following image:

If we plug-in USB disk in edge machine, it automatically mounts in the devices

Low-speed USB: it mounts /media/usb-sdf1

High-speed USB: it mounts /data2



```
/dev/sde4         25G  7.3M   24G   1% /data2
/dev/sdf1        115G  9.2G  106G   8% /media/usb-sdf1
```

1.2. PC + Edge-Devices Chatbot structure

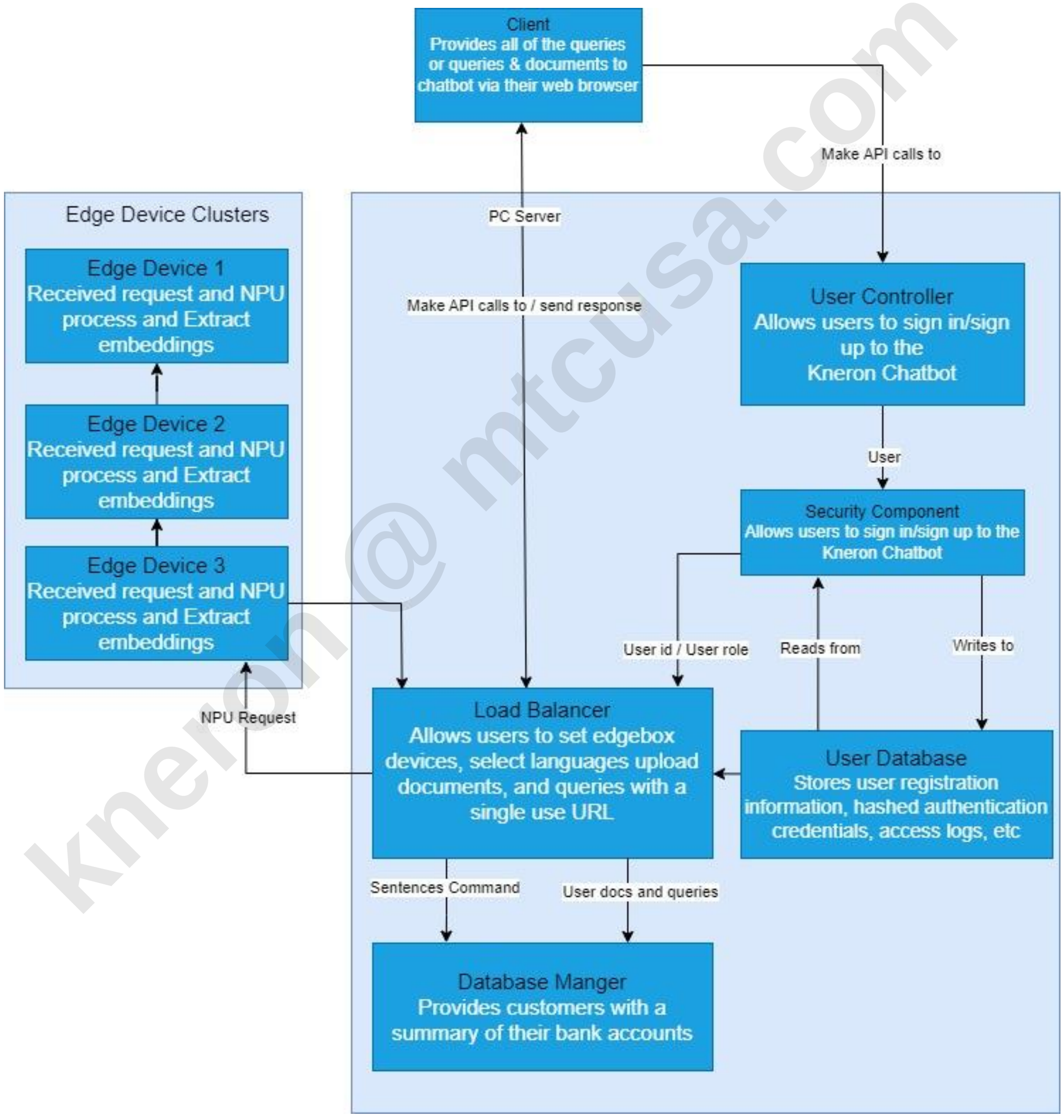

Client
Provides all of the queries or queries & documents to chatbot via their web browser
Make API calls to
Edge Device Clusters
PC Server
Edge Device 1
Received request and NPU process and Extract embeddings
Make API calls to / send response
User Controller
Allows users to sign in/sign up to the Kneron Chatbot
Edge Device 2
Received request and NPU process and Extract embeddings
User
Security Component
Allows users to sign in/sign up to the Kneron Chatbot
Edge Device 3
Received request and NPU process and Extract embeddings
User id / User role
Reads from
Writes to
NPU Request
Load Balancer
Allows users to set edgebox devices, select languages upload documents, and queries with a single use URL
User Database
Stores user registration information, hashed authentication credentials, access logs, etc
Sentences Command
User docs and queries
Database Manger
Provides customers with a summary of their bank accounts

Figure 1—2 PC + Edge-Devices Chatbot structure

1.2.1. Advantages:

- Scalability: They can support scaling up large number of users' knowledge-based documents and users' database
- Efficiency in private way: Leverage internal PC server unlimited disk and memory resources and edge-devices powerful NPU process capabilities, there is can control over data more efficiently in a private way.

1.2.2. Disadvantages:

- Control mechanism: They uses complex control mechanisms to dispatch tasks to edge-devices cluster and provides more interfaces APIs to communicate with PC server and edge-devices cluster.
- External NAS location in PC machine: For PC machine, the location is depending on PC machine's OS and mounting methods.

2. Introduction to the Python API

This chapter introduces load balancer Python API. Let's take two edge devices as examples.

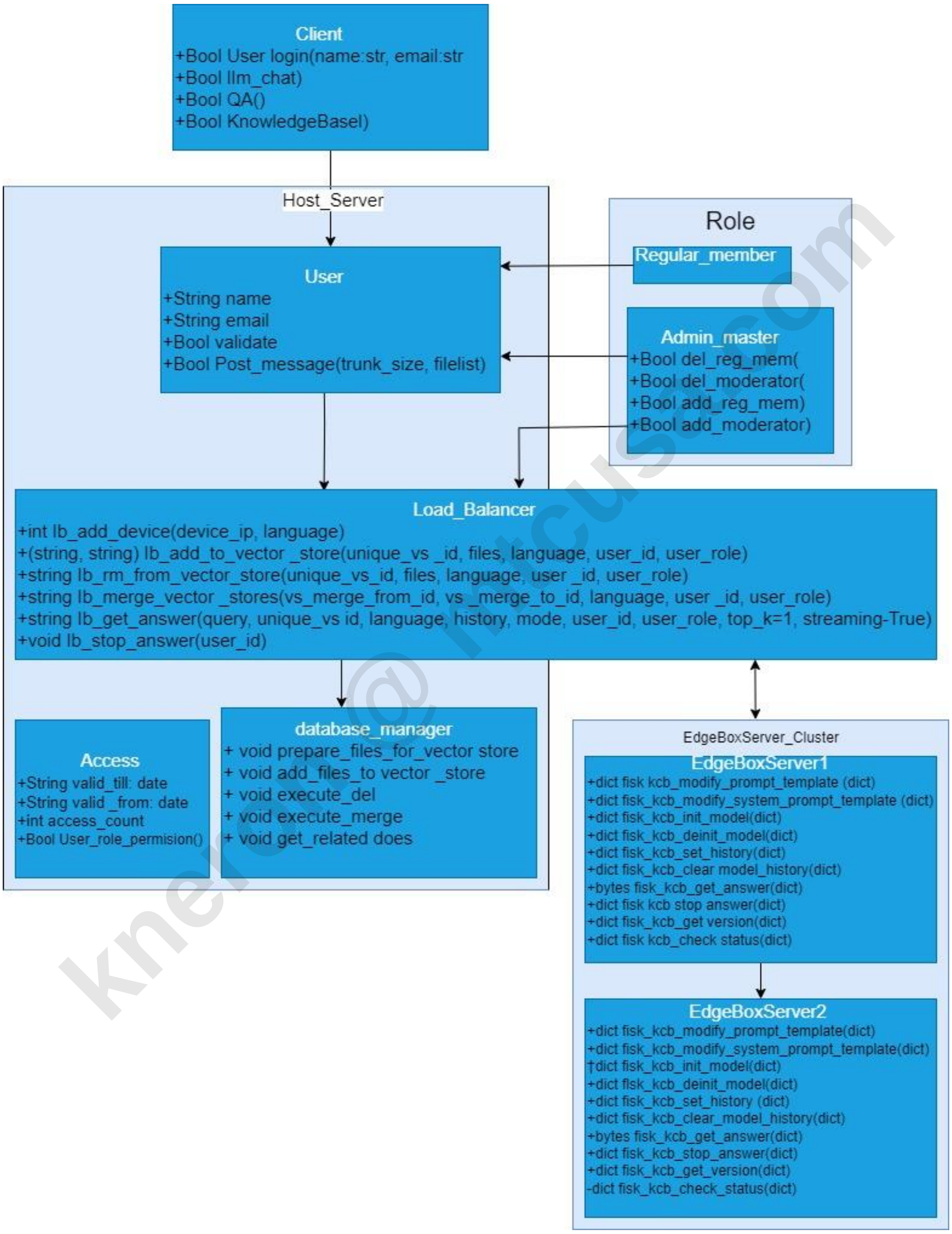

Client
+Bool User login(name:str, email:str
+Bool llm_chat)
+Bool QA()
+Bool KnowledgeBasel)
Host_Server
User
+String name
+String email
+Bool validate
+Bool Post_message(trunk_size, filelist)
Role
Regular_member
Admin_master
+Bool del_reg_mem(
+Bool del_moderator(
+Bool add_reg_mem)
+Bool add_moderator)
Load_Balancer
+int lb_add_device(device_ip, language)
+(string, string) lb_add_to_vector _store(unique_vs _id, files, language, user_id, user_role)
+string lb_rm_from_vector_store(unique_vs_id, files, language, user _id, user_role)
+string lb_merge_vector _stores(vs_merge_from_id, vs _merge_to_id, language, user _id, user_role)
+string lb_get_answer(query, unique_vs id, language, history, mode, user_id, user_role, top_k=1, streaming-True)
+void lb_stop_answer(user_id)
Access
+String valid_till: date
+String valid _from: date
+int access_count
+Bool User_role_permision()
database_manager
+ void prepare_files_for_vector store
+ void add_files_to vector _store
+ void execute_del
+ void execute_merge
+ void get_related does
EdgeBoxServer_Cluster
EdgeBoxServer1
+dict fisk kcb_modify_prompt_template (dict)
+dict fisk_kcb_modify_system_prompt_template (dict)
+dict fisk_kcb_init_model(dict)
+dict fisk_kcb_deinit_model(dict)
+dict fisk_kcb_set_history(dict)
+dict fisk_kcb_clear model_history(dict)
+bytes fisk_kcb_get_answer(dict)
+dict fisk kcb stop answer(dict)
+dict fisk_kcb_get version(dict)
+dict fisk kcb_check status(dict)
EdgeBoxServer2
+dict fisk_kcb_modify_prompt_template(dict)
+dict fisk_kcb_modify_system_prompt_template(dict)
†dict fisk_kcb_init_model(dict)
+dict flsk_kcb_deinit_model(dict)
+dict fisk_kcb_set_history (dict)
+dict fisk_kcb_clear_model_history(dict)
+bytes fisk_kcb_get_answer(dict)
+dict fisk_kcb_stop_answer(dict)
+dict fisk_kcb_get_version(dict)
-dict fisk_kcb_check_status(dict)

Figure 2—1 APIs of service

2.1. Load Balancer Python API

For more information about the Python API, see the kneron_chatbot_prod directory. README_Kneron_load_balancer_APIs.md. Currently, the following APIs are available.

Python API	Description of the function.
lb_get_version	Get the current version of the chatbot.
lb_get_language	Gets the language of the currently loaded device.
lb_get_devices	Gets the list of initialized devices.
lb_add_device	Add edge-devices from devices 'cluster.
lb_check_device_status	Check device status
lb_get_llm_params	Gets the current parameters for the LLM model
lb_update_llm_params	Set parameters for the LLM model.
lb_get_vector_store_list	Gets the list of vector stores that the user can access.
lb_rm_from_vector_store	Delete files or folders from existing document database.
lb_merge_vector_stores	Merge two knowledge bases.
lb_get_file_list	Gets the list of files for a specific vector store.
lb_upload_to_vector_store	Upload files to a new or existing vector store in different storage.
lb_get_answer	Ask question and get answer.
lb_stop_answer	Stop answering what is currently being done.
lb_clean_history	Clear the discussion on the front end
lb_set_local_prompts	Set runtime prompts.
lb_get_vector_store_saving_path	Get path info for vector stores including saving paths and index of current saving path.
lb_add_vector_store_saving_path	Add a new path to saving path.
lb_rm_vector_store_store_saving_path	Remove a path from saving path.

Description of an example of an API

Before Using those APIs, you must bring up *load_balancer_host* service. There are two ways to enable load_balancer_host service: You can start up all services, including *load_balancer_host*, with "new_launch.sh" which describes in *2.4.2 ssh* section of *KNEO 300 User Manual.* Alternatively, you can start load_balancer_host service independently by following commands:

```
$ cd /home/linaro/kneron_chatbot_prod/kneron_doc_chat
$ source ../envsetting.sh
$ python3 kneron_software/kneron_server_software/load_balancer/load_balancer_host.py
```

When the port is activated, the port will display its IP address and port number, such as: http://192.192.11.101:5002
Now, you can use python command to connect with load_balancer via API
Example:

response = requests.post(“http://192.192.11.101:5002/load_balancer/
”+"lb_add_device", json=data)

Here are the detailed API descriptions.
kneron_doc_chat/webui_edge.py

2.1.1 .lb_get_version

Data parameters:
None.

Returns:
It returns a string of version number if command is executed correctly.

Description:
To obtain current version number of chatbot.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
response = requests.get(API_URL+"lb_get_version")
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Returns:
0.16.0
```

2.1.2 .lb_get_language

Data parameters:
None.

Returns:
It returns a string of usage of language if command is executed correctly.

Description:
To obtain current language of chatbot.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
response = requests.get(API_URL+"lb_get_language")
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Returns:
```

```
Chinese
```

2.1.3 .lb_get_devices

Data parameters:

None.

Returns:

It returns a list of initialized devices if command is executed correctly.

Description:

To obtain current initialized devices which chatbot in connected to.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
response = requests.get(API_URL+"lb_get_devices")
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Returns:
{"device_list":[{"busy_status":false,"host":"192.168.100.165","init_status":true,"language":"Chi
nese","machine_status":"True"}]}
```

2.1.4. lb_add_device

Data parameters:

device_ip(string): IP address of device which will process NPU request. Example: 192.168.100.200.

language(string): language of usage. Input: English, Chinese.

Returns:

status: 0 when the initialization is failed. 1. Device is added successfully. 2 means the device was added, and the language is the same with user's selection. 3 means the device was added, and the language is different with user's selection.

Description:

Automatically try to connect with the IP of the input. If there is no response within 5 seconds, it is considered a failure. When it connects successfully, please select language first. After that, please add the devices IP address to initialize the edge devices cluster (you can add more edge devices IP address to support this service). If the initialization failed, it would return status=0. Please check if edge devices launch or not.

Example:

```
API_URL = fhttp://HOST_IP:5002/load_balancer/
data = { "device_ip": "192.168.100.165",
                "language": "English" }
response = requests.post(API_URL+"lb_add_device", json=data)
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Return:
{"status":1}
```

2.1.5 lb_check_device_status

Data parameters:

Ip(string): IP address of the device.

Return:

Status of the device including initialization status, language, machine status, and NPU status.

Description:

The API returns current status of the device. Those status are initialization, language, machine, and NPU.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = { "device_ip": "192.168.100.165" }
response = requests.post(API_URL+"lb_check_device_status", json=data)
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)


Return:
{"init_status":true,"language":"Chinese","machine_status":"True","npu_status":true}
```

2.1.6. lb_update_llm_params

Data parameters:

language(string): language of usage. Input: English, Chinese.

user_id(string): ID of user who sent the request. Example: admin

token_max_length(number): Max toke length for LLM model. Input: <= 4096 for Chinese, <= 2048 for English.

postprocess_mode(number): postprocess mode for LLM model. Input: 0 for greedy mode, 1 for top_p.

Temperature(number): parameter that controls the randomness of language model output. Higher values result in a more “confident” output, whereas lower values result in more conservative output. Input: range for [0,1.0].

top_p(number): Probability threshold for which tokens with greater probability will be selected.

Larger values generate more random output. Input: range of [0, 1.0].
repetition_penatly(number): repetition penalty for the model. Input: range of [1.0, 10.0].
log_level(number): Log level for LLM model.

Return:
status: `information about the parameter setting operation.`

Description:
An API to adjust parameters of LLM model. Chatbot will answer user’s query according to LLM model parameters. LLM model parameters can be adjust via “lb_upldate_llm_params” or “lb_get_answer”.

Example:
```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = { "user_id": "admin",
             "language": "Chinese",
             "token_max_length": "4096",
             "postprocess_mode": "1",
             "temperature": "1.0",
             "top_p": "0.4",
             "repetition_penalty": "1.1",
             "log_level": "0" }
response = requests.post(API_URL+"lb_update_llm_params", json=data)
for bytes_data in response.iter_content(chunk_size=8192):
      str_data = bytes_data.decode('utf-8')
      print(str_data)


Return:
{"status":"Params set successfully."}
```

2.1.7. lb_get_llm_params

Data parameters:
language(string): language of usage. Input: English, Chinese.
Returns:
A list of current LLM parameters.

Description:
To obtain current parameters of running LLM models such as log level, postprocess_mode, repetition_penalty, temperature, token_max_length, and top_p.

Example:
```
API_URL = fhttp://192.168.100.168:5002/load_balancer/
data = {"language": "English" }
```

```
response = requests.post(API_URL+"lb_get_llm_params", json=data)
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Return:
{"log_level":"0","postprocess_mode":"1","repetition_penalty":"1.1","temperature":"1.0","toke
n_max_length":"4096","top_p":"0.3"}
```

2.1.8. lb_get_vector_store_list

Data parameters:

language(string): language of usage. Input: English, Chinese.

user_id: ID of requestor.

user_role: permissions of requestor.

Returns:

A list of vector stores that user can access.

Description:

To obtain current available vector stores in connected device.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = { "user_id": "admin",
              "user_role": "admin",
              "language": "English" }
response = requests.post(API_URL+”lb_get_vector_store_list”, json=data)
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode(‘utf-8’)
        print(str_data)

Return:
{"vector_store_list":["admin/admin_vs","public/custom_vs","John/John_fav_vs","John/John_p
b_vs, "admin/test"]}
```

2.1.9. lb_rm_from_vector_store

Data parameters:

unique_vs_id(string): creator id/ knowlegde base id of the target knowledge base. Example: public/Installation_manual.files(string): filename list (without full address) Example: Readme.txt

language(string): language of usage. Input: English, Chinese.user_id(string): ID of the requestor. Example: John.user_role(string): permissions of the requestor. Input: admin, regular.

Returns:

status: `information about the removing operation`

Description:

Delete specified files from the specified knowledge base address or delete the entire knowledge base. Use the creator ID / knowledge base ID to point to a specific knowledge base. If you wish to delete the entire knowledge base, please set the entry in 'files' to ['Delete Whole Knowledge Base']

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = { "user_id": "admin",
           "user_role": "admin",
           "language": "English",
           "files": ["readme.txt"],
           "unique_vs_id": "admin/test"}
response = requests.post(API_URL+"lb_rm_from_vector_store", json=data)
for bytes_data in response.iter_content(chunk_size=8192):
      str_data = bytes_data.decode('utf-8')
      print(str_data)

Return:
{"status":"File readme.txt has been deleted from test."}
```

2.1.10. lb_merge_vector_stores

Data parameters:

vs_merge_from_id(string): creator id/ knowlegde base id of first knowledge base. Example: public/test1.

Vs_merge_to_id(string): creator id/ knowlegde base id of second knowledge base. Example: public/test2.

language(string): language of usage. Input: English, Chinese.

user_id(string): ID of the requestor. Example: John.

user_role(string): permissions of the requestor. Input: admin, regular.

Returns:

status: `information about the merging operation.`

Description:

The two knowledge libraries are merged. Use the creator ID / knowledge base ID to point to the specific knowledge bases.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = { "user_id": "admin",
           "user_role": "admin",
           "language": "English",
           "vs_merge_from_id": "admin/test",
            "vs_merge_to_id": "admin/test2"}
response = requests.post(API_URL+"lb_merge_vector_stores", json=data)
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Return:
{"status":"Knowledge base admin/test has been merged to admin/test2 successfully."}
```

2.1.11. lb_get_file_list

Data parameters:

unique_vs_id(string): creator id/ knowlegde base id of the target knowledge base. Example: admin/test2.

language(string): language of usage. Input: English, Chinese.

Returns:

`A list of files within desired knowledge base.`

Description:

To obtain a list of files which are added into a specific knowledge base.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = { "language": "English",
              "unique_vs_id": "admin/test2" }
response = requests.post(API_URL+"lb_get_file_list", json=data)
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Return:
{"file_list":["SDK_Readme.txt","README_AM3X.txt"]}
```

2.1.12. lb_upload_to_vector_store

Data parameters:

unique_vs_id(string): creator id/ knowlegde base id of the target knowledge base. Example: public/Installation_manual.

files(string): filename list (without full address) Example: "/mnt/sd/Readme.txt"

language(string): language of usage. Input: English, Chinese.

user_id(string): ID of the requestor. Example: John.

Returns:

formatted files: `string of loaded files Has been processed for displaying on the web.`

status: `loaded file status.`

Description:

Deposit the specified files into the designated knowledge base address. Use the creator ID / knowledge base ID to point to a specific knowledge base.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = { "language": "English",
            "unique_vs_id": "admin/test2",
            "user_id": "admin",
            "user_role": "admin"}
files = []
to_close = []
fp = open("ReadSample2.txt", "rb")
files.append(('file', ('ReadSample2.txt',fp, 'application/octet-stream')))
to_close.append(fp)
response = requests.post(API_URL+"lb_upload_to_vector_store", files=files, data=data)

for bytes_data in response.iter_content(chunk_size=8192):
     str_data = bytes_data.decode('utf-8')
     print(str_data)

Return:
{"formated_files":"<h3>Knowledge Base File list (Total Files: 3)</h3><div style='height: 200px;
overflow-y: scroll;'><ul style='list-style-type: disc;'><li>StreamEye_.txt</li><li>
ReadSample2.txt</li><li>SDK_Readme.txt</li><li>README_AM3X.txt</li></ul></div>","status":
"Files ReadSample2.txt have been uploaded to the knowledge base and are now loaded. Please
start asking questions.\n"}
```

2.1.13. lb_get_answer

Data parameters:

query(string): questions for the chatbot. Example: What is a spring?

unique_vs_id(string): creator id/ knowlegde base id of the target knowledge base. Example: public/garage.

language(string): language of usage. Input: English, Chinese.

history(array[string]): list of strings of past questions-and-answers. Example: [["start",None]].

mode(string): type of question-and-answer mode. Input: LLM, Knowledge Base Q&A, 知識庫問答.

user_id(string): ID of the requestor. Example: John.

user_role(string): permissions of the requestor. Input: admin, regular.

top_k(number): match quanity. Input: 1, 2, 3.

token_max_length(number)(optional): Max toke length for LLM model. Input: <= 4096 for Chinese, <= 2048 for English.

postprocess_mode(number)(option): postprocess mode for LLM model. Input: 0 for greedy, 1 for top p.

temperature(number)(optional): parameter that controls the randomness of language model output. Higher values result in a more “confident” output, whereas lower values result in more conservative output. Input: range for [0,1.0].

top_p(number)(optional): Probability threshold for which tokens with greater probability will be selected. Larger values generate more random output. Input: range of [0, 1.0].

repetition_penalty(number)(optional): repetition penalty for the model. Input: range of [1.0, 10.0].

log_level(number)(optional): Log level for LLM model.

Returns:

```
reply: string of the reply.
```

Description:

Use the creator ID / knowledge base ID to point to a specific knowledge base. The history of questions and answers is a list, and each item in the list is another list that contains one question and one answer. If a question/answer does not exist, set it to None. The mode can be set to LLM or 'Knowledge Base Q&A'. top_k can be set to determine the number of matching information sources. Recommendations are not more than 3. Since a real-time return function is used, please refer to the following example for processing the return information.

Optional LLM settings can affect behavior of LLM model. User can give value to all or some of settings. For setting that are not given any values, system will use default setting automatically.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = {"query": "what is a spring?",
            "unique_vs_id": "admin/test",
            "language": "English",
            "history": [["hello", "hello"],["thank you", "you are welcome"]],
            "mode": "LLM",
            "user_id": "admin",
            "user_role": "admin",
            "top_k": 1,
            "token_max_length": 2048,
            "postprocess_mode": 1,
            "temperature": 0.5,
            "top_p": 0.5,
            " repetition_penalty": 5,
            "log_level": 0 }
response = requests.post(API_URL+"lb_get_answer", json=data, stream=True)
for bytes_data in response.iter_content(chunk_size=16384):
```

```
                    str_data = bytes_data.decode('utf-8')
                    print(str_data)

Return:
A spring is a device that stores mechanical energy in the form of an elastic deformation. It is
made up of a flexible material...
```

2.1.14. lb_stop_answer

Data parameters:

  user_id(string): ID of the requestor. Example: John.

Returns:

  1: Successful operation.

Description:

  This will stop the ongoing Q&A session of the user who made the request.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = { "user_id": "John"}
response = requests.post(API_URL+"lb_stop_answer", json=data)
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Return:
{"status":1}
```

2.1.15. lb_clear_history

Data parameters:

  user_id(string): ID of the requestor. Example: John.

Returns:

  1: Successful operation.

Description:

  To clear the discussion on the front end.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = { "user_id": "John"}
response = requests.post(API_URL+"lb_clear_answer", json=data)
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Return:
{"status":1}
```

2.1.16. lb_set_local_prompts

Data parameters:

Question_prompt: question prompt for Chinese or English mode. Do not change the contents for {***}

System_prompt: system prompt for Chinese or English. Do not change the contents for between < and the next , and the contests for {***}

Returns:

Operation status, 200 for successful operation.

Description:

Question and system prompt can be reconfigured to adopt temporary needs. Use templates,json as example, You can modify prompt and submit. Do not change any word between **<xxxxx\n** and**{xxxxxxx}**, For example, ***<|im_start>system\n*** , ***{{QUESTION}}.*** New prompt is effective until system is rebooted, service is restarted or another new prompt is set.

Example:

```
data = { "question_prompt":
         {
          "Chinese": "###INSTRUCTION\n 使用最精簡的方式回答問題。
           \n\n###REQUIREMENT\n 不可捏造答案，如果無法找到答案,回答:找不到答案
           {{ CONTEXT }}",
          "English": "###INSTRUCTION\nPlease provide a neat short answer.
           \n\n###REQUIREMENT\ndo not make up answers, if can not find a answer, simply say :I
           don't know the answer.{{ CONTEXT }}”
         }
       }, "system_prompt":
         {
          "Chinese": "<|im_start|>system\n 你是個圖書管理員，看起來冷漠無情，但是其實很
             熱情<|im_end|>\n<|im_start|>user\n\n{{ QUESTION_PROMPT }}\n\n###Question：
             {{ QUESTION }}<|im_end|>\n<|im_start|>assistant\n",
          "English": "<|im_start|>system\nYou are a librarian. You have motionless look, but
               actually love to help people. <|im_end|>\n <|im_start|>user\n\n
               {{ QUESTION_PROMPT }}\n\n###question: {{ QUESTION }} <|im_end|>\n
               <|im_start|>assistant\n"
           }
          }
       response = requests.post(API_URL+"lb_set_local_prompts", json=data)
```

```
    print_result(response)

Return:
{"status":"pass"}
```

2.1.17. lb_get_vector_store_saving_path

Data parameters:

None

Returns:

List of all saving paths.

Description:

Get path info for vector stores including saving paths and index of current saving path.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
response = requests.post(API_URL+"lb_get_vector_store_saving_path")
for bytes_data in response.iter_content(chunk_size=8192):
    str_data = bytes_data.decode('utf-8')
    print(str_data)

Return:
{"paths_info":{"current_path":0,"paths":["default","/data"]}}
```

2.1.18. lb_add_vector_store_saving_path

Data parameters:

path(string): path name to be added. Example: /data

language(string): language of usage. Input: English, Chinese.

Returns:

List of added path.

Description:

Add a new path to saving paths.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = { "path": "/data",
         "language": "English" }
response = requests.post(API_URL+"lb_add_vector_store_saving_path", json=data)
for bytes_data in response.iter_content(chunk_size=8192):
    str_data = bytes_data.decode('utf-8')
    print(str_data)
```

```
Return:
{"paths":"Path added: /data"}
```

2.1.19. lb_rm_vector_store_saving_path

Data parameters:

Path(string): path name to be removed. Example: /data

Returns:

List of removed path.

Description:

Remove a path from saving path.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = { "path": "/data" }
response = requests.post(API_URL+"lb_rm_vector_store_saving_path", json=data)
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Return:
{"paths":"Path removed: /data"}
```

2.1.20. lb_set_vector_store_current_saving_path

Data parameters:

current_path(integer): index of path in the path list. Example: 0

Returns:

List of current path.

Description:

Set current saving path for vector store.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = { "current_path": 1 }
response = requests.post(API_URL+"lb_set_vector_store_current_saving_path", json=data)
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Return:
{"current_path":1,"status":"Current path has been set to: /data"}
```

2.1.21. lb_get_max_match_size

Data parameters:

None

Returns:

Value of max match size.

Description:

Get maximum match size which is supported for matched sources.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
response = requests.get(API_URL+"lb_get_max_match_size")
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Return:
{"max_match_size":3}
```

2.1.22. lb_get_model_configs

Data parameters:

None

Returns:

Current model configuration.

Description:

Retrieve model configuration details. Details description of configurations are written in 2.1.24 lb_set_model_configs.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
response = requests.get(API_URL+"lb_get_module_configs")
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Return:
{"COMPANY_QA_THRESHOLD":0.725,"EN_OVERLAP_SIZE":60,"EN_REFER_CHUNK_SIZE":400,"K
NOWLEDGE_BASE_QA_THRESHOLD":0.0,"ONLY_COMPANY_QA":false,"REPHRASE_QUERY":false
,"TITLE_THRESHOLD":0.6,"USE_COMPANY_QA":false,"USE_EXPLAIN":false,"USE_RERANK_SORTI
NG":true,"USE_SKIP_PDF_WITH_UNDECODED_UNICODE":true,"USE_SPLIT_RERANK":true,"ZH_
OVERLAP_SIZE":350,"ZH_REFER_CHUNK_SIZE":900}
```

2.1.23. lb_set_model_configs

Data parameters:

ZH_REFER_CHUNK_SIZE(integer): Split Chinese documents as chunk. 0 to 1000.
ZH_OVERLAP_SIZE(integer): Split Chinese documents as chunk with overlap size. 0 to 512.
EN_REFER_CHUNK_SIZE(integer): Split English documents as chunk. 0 to 512.
EN_OVERLAP_SIZE(integer): Split English documents as chunk with overlap size. 0 to 256.
KNOWLEDGE_BASE_QA_THRESHOLD(number): The lower the threshold, the more sources can be retrieved.
TITLE_THRESHOLD(number): The lower the threshold, the more likely your question to be matched with the filename first.
USA_COMPANY_QA(bool): Set if enable company Q&A feature.
COMPANY_QA_THRESHOLD(number): Probability threshold for which tokens with greater probability will be selected. Larger values generate more random output. range of [0,1].
ONLY_COMPANY_QA(bool): Set to true means chatbot will stop answer if question doesn't match any of company Q&A. Set to false means chatbot will try to answer question with free chat when questions doesn't match any of company Q&A.
USE_EXPLAIN(bool)(Not available in KNEO300): decide if let chatbot extract definitions if documents. Such process will slow down the upload file process significantly. Set to Ture for insurance related documents only.
REPHRASE_QUERY(bool) (Not available in KNEO300): let chatbot rephrase your question first if set to true.
USE_SPLIT_RERANK(bool): Set to True for KNEO300/330 if use splitting and reranking for matched document chunk.
USE_RERANK_SORRING(bool) Set to True for KNEO300/330 if use bce reranking when retrieve relevant documents.
USE_SKIP_PDF_WITH_UNDECODED_UNICODE(bool): Set to True if skipping pdf file with Unicode, which can not be decoded, when creating knowledge base.

Returns:

Status of operation: 200 for successful operation.

Description:

Set model configuration for documents processing and query answering. USE_EXPLAIN and REPHRASE_QUERY are not available in KNEO300. Modified configurations are effective until system is rebooted, service is restarted, or configurations are modified again.

Example:

```
API_URL = f"http://192.168.100.165:5002/load_balancer/"
data = { "COMPANY_QA_THRESHOLD":0.725,
                    "EN_OVERLAP_SIZE":60,
                    "EN_REFER_CHUNK_SIZE":400,
                    "KNOWLEDGE_BASE_QA_THRESHOLD":0.0,
                    "ONLY_COMPANY_QA":True,
                    "REPHRASE_QUERY": True,
                    "TITLE_THRESHOLD":0.6,
                    "USE_COMPANY_QA":True,
                    "USE_EXPLAIN": True,
                    "USE_RERANK_SORTING":True,
                    "USE_SKIP_PDF_WITH_UNDECODED_UNICODE":True,
                    "USE_SPLIT_RERANK":True,
                    "ZH_OVERLAP_SIZE":350,
                    "ZH_REFER_CHUNK_SIZE":900}
response = requests.post(API_URL+"lb_set_model_configs", json=data)
for bytes_data in response.iter_content(chunk_size=8192):
        str_data = bytes_data.decode('utf-8')
        print(str_data)

Return:
{"COMPANY_QA_THRESHOLD":"pass","EN_OVERLAP_SIZE":"pass","EN_REFER_CHUNK_SIZE":"p
ass","KNOWLEDGE_BASE_QA_THRESHOLD":"pass","ONLY_COMPANY_QA":"pass","REPHRASE_
QUERY":"not support
currently","TITLE_THRESHOLD":"pass","USE_COMPANY_QA":"pass","USE_EXPLAIN":"not
support
currently","USE_RERANK_SORTING":"pass","USE_SKIP_PDF_WITH_UNDECODED_UNICODE":"p
ass","USE_SPLIT_RERANK":"pass","ZH_OVERLAP_SIZE":"pass","ZH_REFER_CHUNK_SIZE":"pass"}
```

3. WEBUI Interface Introduction

3.1. Interface layout

Figure below is the Chatbot interface. Start from top-right corner is setting menu where user where user can select chatbot mode, free chat or knowledge base. In the figure, Current working mode is Free chat mode.
Globe icon next to setting menu is language selection, current support English and Chinese.
On the left side, light blue area, is query history log which show every asked questions. The "New Chat" icon is reset main dialog window as figure shows. User can modify and export query history.
Bottom-left show current user and logout icon.
Bottom-mid is dialog box where user can enter question for chatbot.

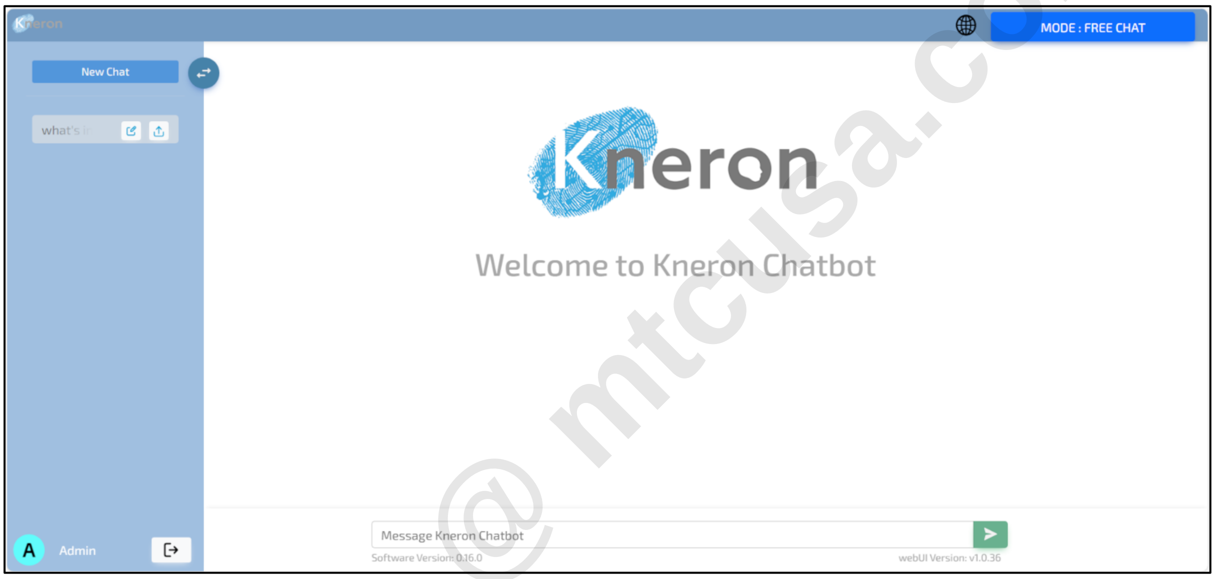


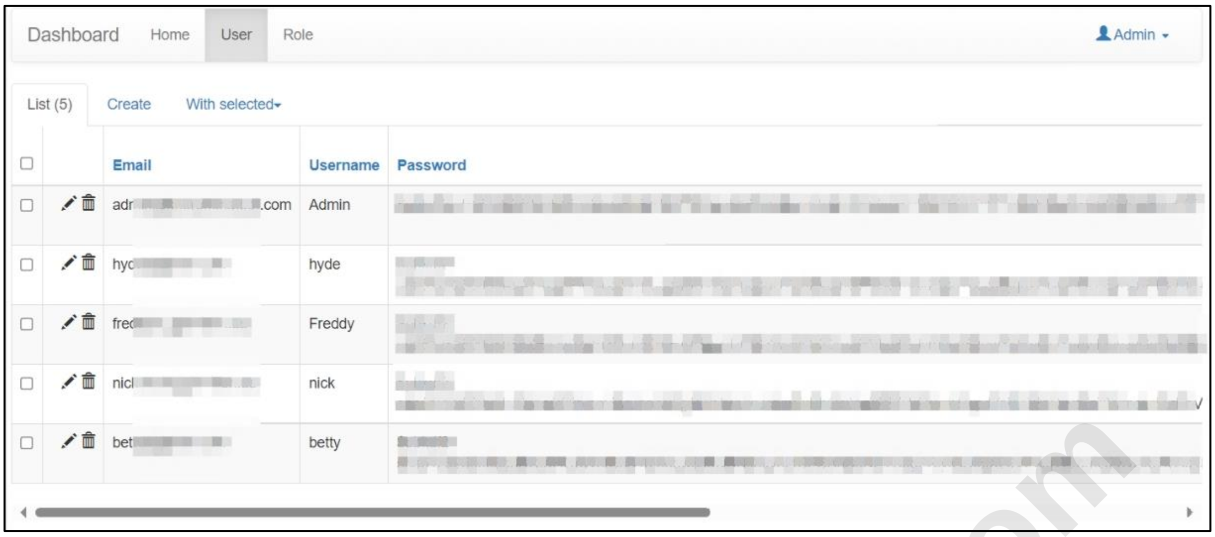
Figure 3—1 Screen shot of main interface.

3.2. Setting menu

Click top-right corner icon can open setting menu. User can add device, language, switch work mode in this menu, Admin user has more option in this menu than regular user has.

General settings are **Language**, **Host**, **Saving Path**, **LLM Params**, and **User management** in this setting menu. Language and Host are used to add device, KNEO300 can add multiple units to share query loading. Saving path and LLM Params definitions are described in APIs examples.

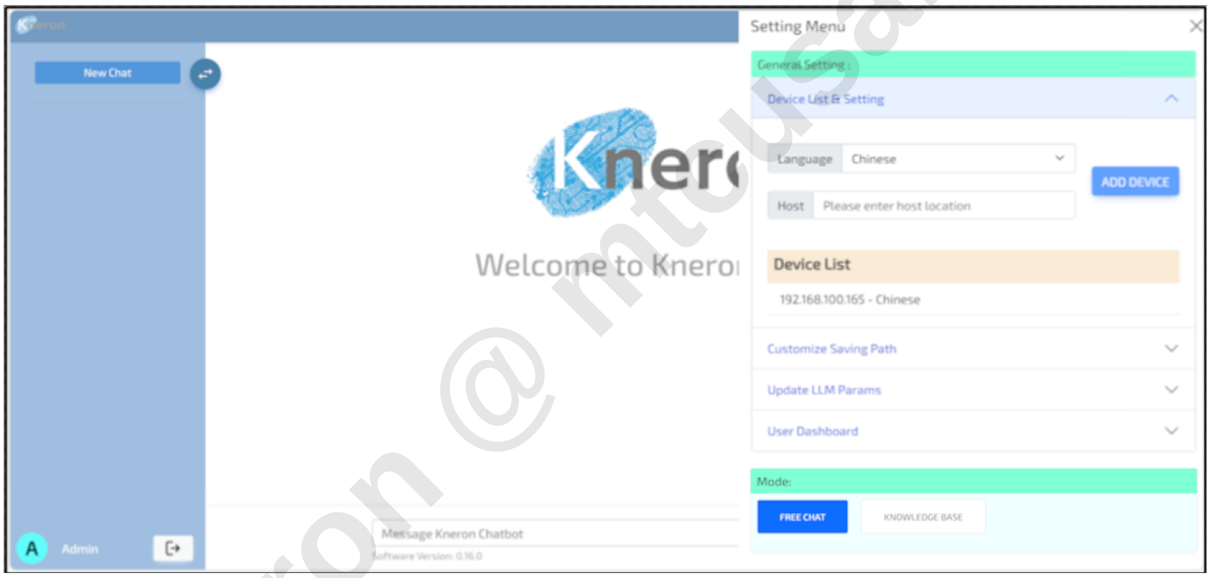
User management is another page where admin user can create/delete/modify user account and privilege.



		Email	Username	Password
☐	✎ 🗑	adr[redacted].com	Admin	[redacted]
☐	✎ 🗑	hyc[redacted]	hyde	[redacted]
☐	✎ 🗑	fre[redacted]	Freddy	[redacted]
☐	✎ 🗑	nicl[redacted]	nick	[redacted]
☐	✎ 🗑	bet[redacted]	betty	[redacted]

Figure 3—2 Screen shot of user management.

3.2.1. Free Chat: a LLM service provides general conversation.



Language: Chinese

Host: Please enter host location

Device List

192.168.100.165 - Chinese

Figure 3—3 Screen shot of Free Chat Setting menu.

3.2.2. Knowledge base mode:

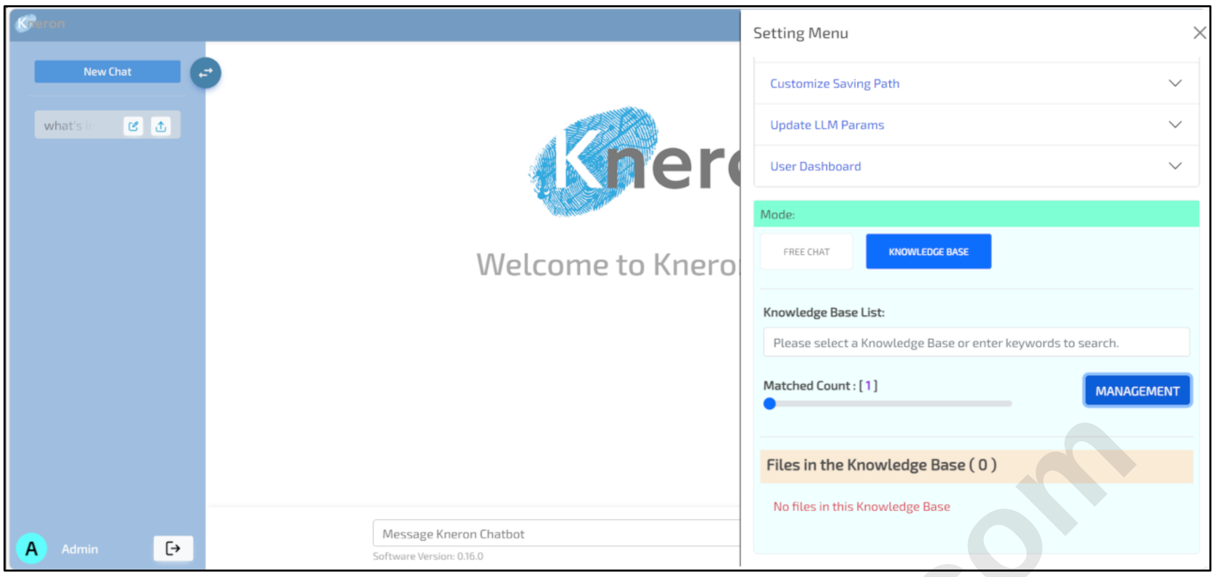


Figure 3—4 Screen shot of Knowledge Base mode setup menu.

In the knowledge base mode, users can conduct a Q&A conversation which interested knowledge based by clicking on **Knowledge Base List**. Or to create a new knowledge base by clicking on **MANAGEMENT** icon, a management menu will popup.

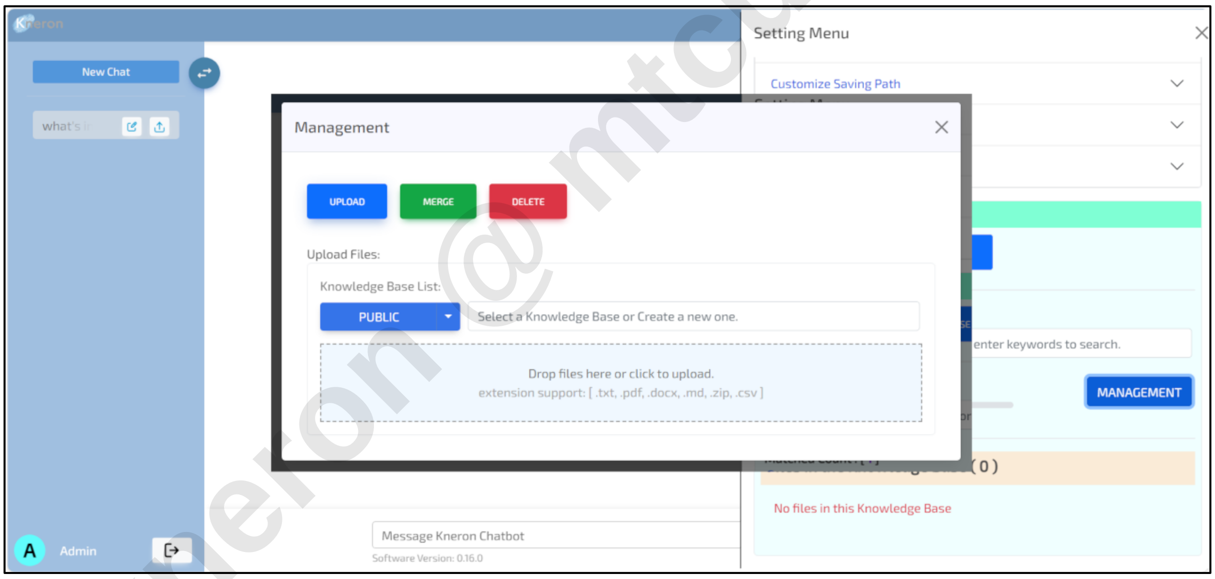

Kneron
New Chat
what's
Admin
Message Kneron Chatbot
Software Version: 0.16.0
Setting Menu
Customize Saving Path
Management
UPLOAD
MERGE
DELETE
Upload Files:
Knowledge Base List:
PUBLIC
Select a Knowledge Base or Create a new one.
Drop files here or click to upload.
extension support: [.txt, .pdf, .docx, .md, .zip, .csv]
enter keywords to search.
MANAGEMENT
(0)
No files in this Knowledge Base

Figure 3—5 Screen shot of knowledge base management set up menu.

3.2.2.1. UPLOAD file to knowledge base:

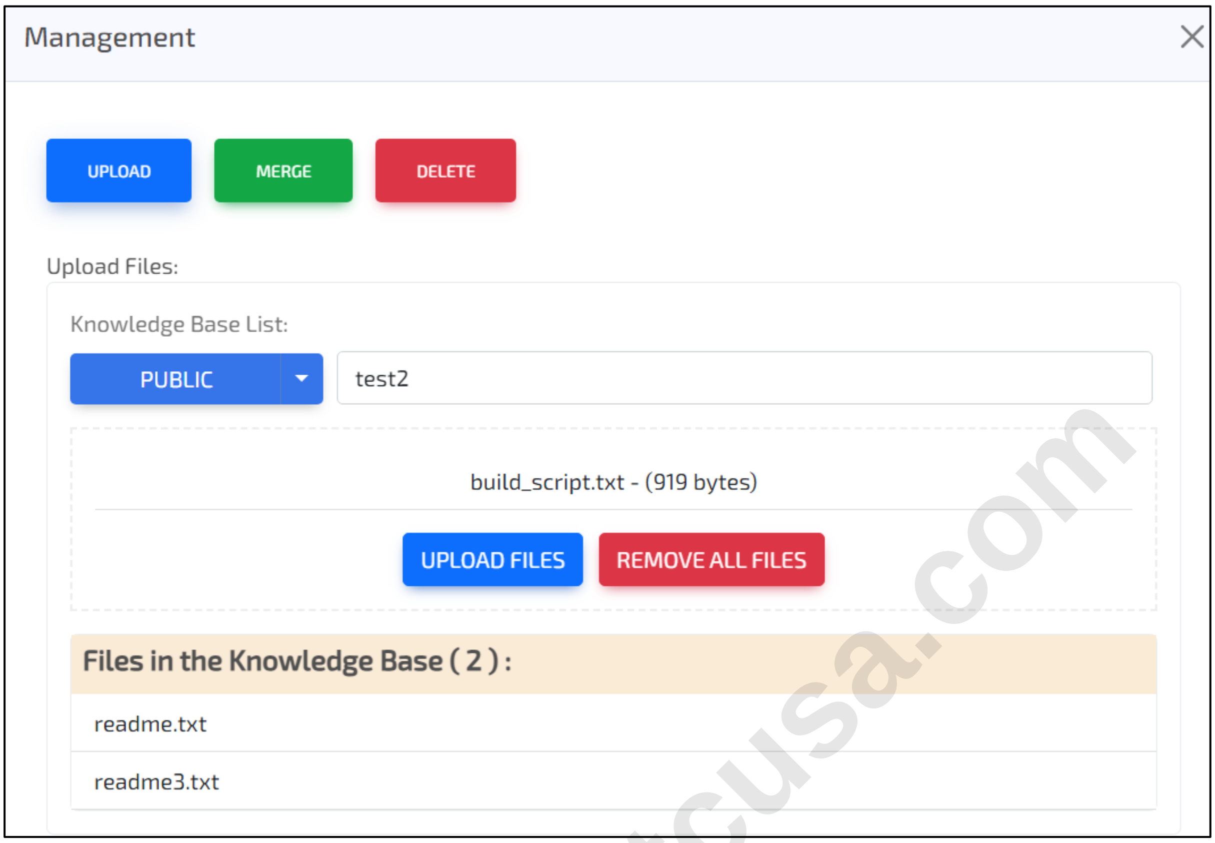


Figure 3—6 Screen shot of knowledge base upload menu.

user can upload more data to existed knowledge base or create new one by clicking on text box to select existed database or type new knowledge base name in text box. Knowledge base can be public or private, user must decide it when create a knowledge base. user can drag knowledge materials to light blue window in lower area or click it to bring up an upload window. Click UPLOAD bottom to upload materials.

3.2.2.2. MERGE knowledge Bases:

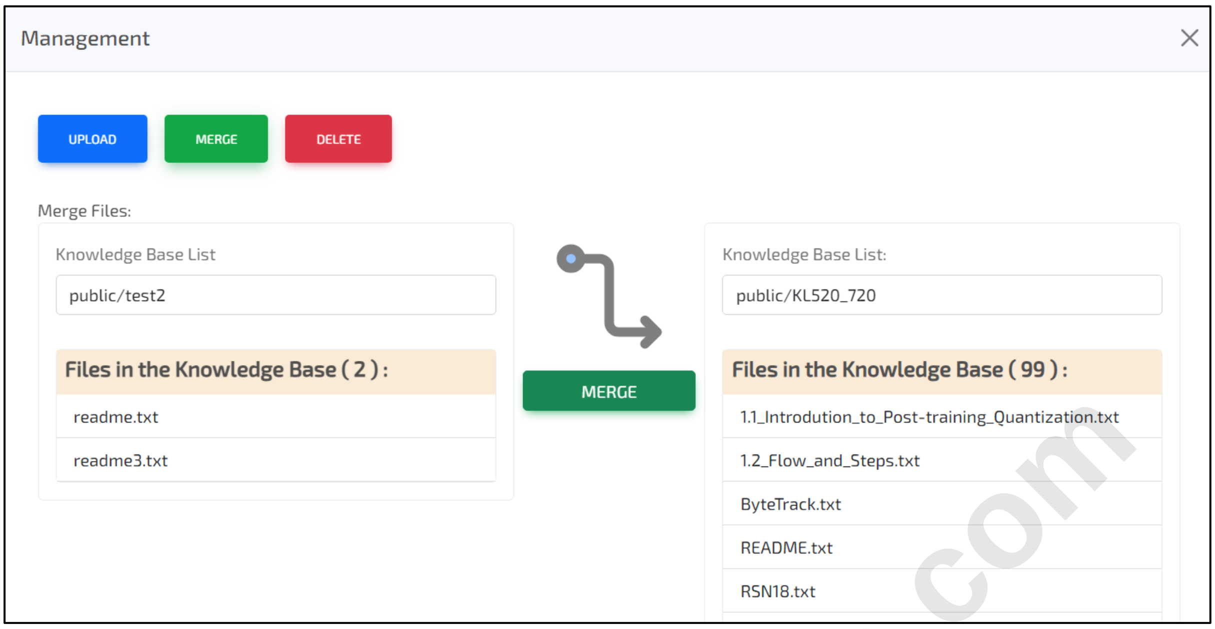


Figure 3—7 Screen shot of knowledge base merge menu.

User can merge one knowledge base to another. Select source knowledge base from left list and target knowledge base from right list. Clicking MERGE bottom in dark green will push documents from left knowledge base to right side knowledge base.

3.2.2.3. DELETE files from knowledge base or delete entire knowledge base:

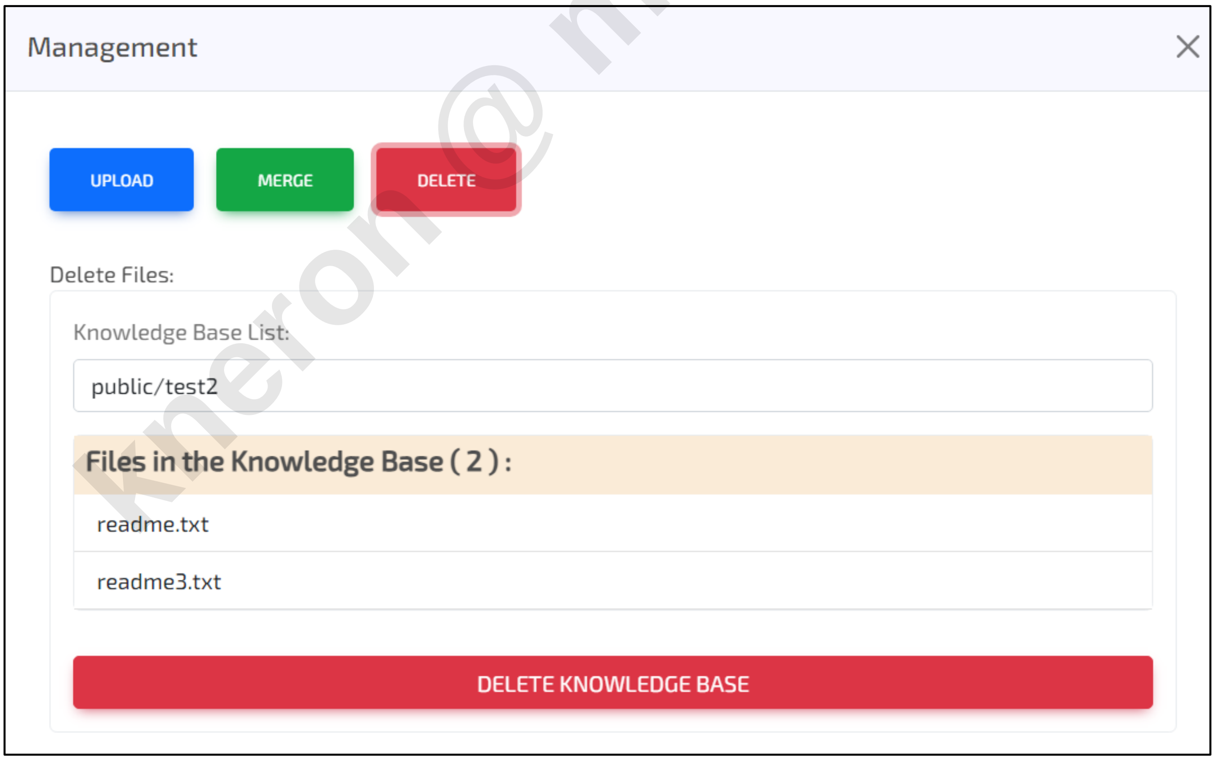


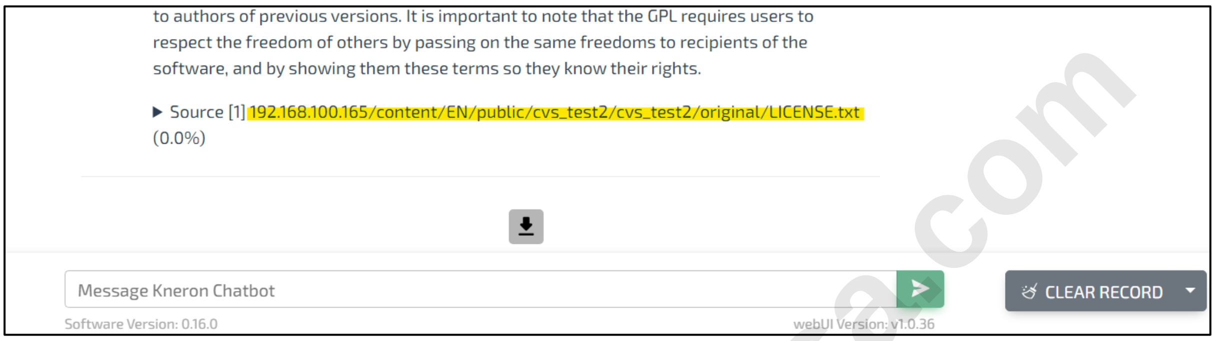
Figure 3—8 Screen shot of knowledge base delete menu.

User can select unwanted files from knowledge base list, then click RED bar (once a file is selected, Text on red bar will be "DELETE SELECED FILES"). To delete a single knowledge base, please click red bar which show “DELETE KNOWLEDGE BASE”

Prompt:

- Please keep the file name specifications in the uploaded file name and do not include special characters, such as (), $, {} , etc.
- Normally, it takes about 15 seconds to upload a file of size 25kbytes, just for reference. Upload speed will be affected by file size, type, format, and current network environment.

3.3. Download the source that related document:



to authors of previous versions. It is important to note that the GPL requires users to respect the freedom of others by passing on the same freedoms to recipients of the software, and by showing them these terms so they know their rights.

▶ Source [1] 192.168.100.165/content/EN/public/cvs_test2/cvs_test2/original/LICENSE.txt (0.0%)

Figure 3—9 Screen shot of knowledge base answer source.

In the generated answers, you can download the source's information and documents. Copy the highlighted part as shown in the image, open a new tab in the browser, paste the file path, and click Enter to view or download the source content.

4. Modify the config files in device

4.1. LLM Model parameters

The following parameters are located in path_config_en.json and path_config_zh.json of kneron_chatbot_prod/kneron_doc_chat/kneron_software/configs. Default values are set in json files.

4.1.1. Postprocess_mode and token_max_length

- Postprocess_mode: 0 for greedy, 1 for top_p. In greedy mode, the output results for the same question, asking the same question repeatedly, are the same. For top_p mode, the output results will be different if the same question is entered again.
- Token_max_length: the maximum token length for input and output of LLM model.

4.1.2. Top-p and temperature

- Top-p: It sets a threshold of probability and selects the top tokens whose cumulative probability exceeds the threshold. The model then randomly samples from this set of tokens to generate output. A larger top-p setting yields more random output. The value of top-p ranges from 0 to 1.
- Temperature: a hyperparameter that controls the randomness of language model output. a higher temperature setting causes the model to be more “confident” in its output. A lower temperature setting yields more conservative and predictable output. The value of temperature ranges from 0 to 1.

4.1.3. Repetition_penalty

- Repetition_penalty definition: the parameter for repetition penalty. The value of repetition_penalty ranges between 1.0 and infinity. 1.0 means no penalty.

4.2. Chunk Size and Overlap Size Setup

The corresponding parameters are located in model_config.py of /home/linaro/kneron_chatbot_prod/kneron_doc_chat/knowledge_base/configs.

4.2.1. Refer Chunk size:

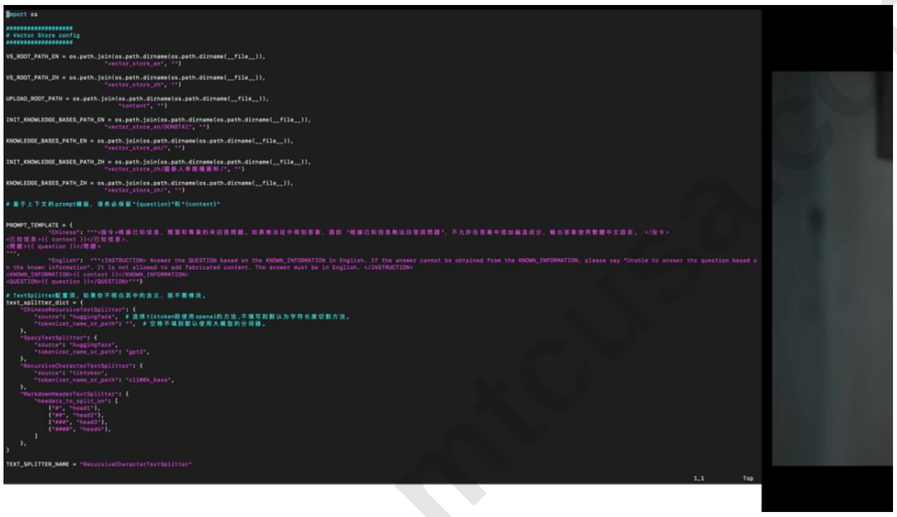
- Definition: Refers to the amount of data from the source document that is processed during Q&A. The size of each block determines the amount of data processed each time. Reasonably setting the block size can effectively manage memory usage and improve processing efficiency. Increasing the block size can provide better contextual information for some tasks but will lead to an increase in memory usage. Affect the original file query paragraph, Token output size, memory use. Each update does not affect the use of the original database, but only the newly generated database will use this setting.
- Here is the link about chunk size https://chunkviz.up.railway.app/

4.2.2. Overlap size:

- Definition: Refers to the number of words or phrases shared between two or more text segments when processing text data. When dividing text into smaller processing units, Overlap can help preserve contextual information, ensuring that important context is not lost at the edges of blocks. Adding overlap means that more computational resources are needed, but the accuracy will increase.
- Here is the link about Overlap size: https://chunkviz.up.railway.app/
- How it works:

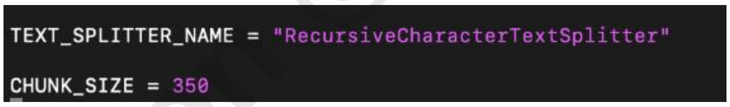
  1. Here is command to adjust chunk size and overlap:

  #cd /home/linaro/kneron_chatbot_prod/kneron_doc_chat/knowledge_base/configs/
  #vim model_config.py



```
import os

####################
# Vector Store config
####################

VS_ROOT_PATH_EN = os.path.join(os.path.dirname(os.path.dirname(__file__)),
                               "vector_store_en", "")

VS_ROOT_PATH_ZH = os.path.join(os.path.dirname(os.path.dirname(__file__)),
                               "vector_store_zh", "")

UPLOAD_ROOT_PATH = os.path.join(os.path.dirname(os.path.dirname(__file__)),
                                   "content", "")

INIT_KNOWLEDGE_BASES_PATH_EN = os.path.join(os.path.dirname(os.path.dirname(__file__)),
                               "vector_store_en/DONGTAI", "")

KNOWLEDGE_BASES_PATH_EN = os.path.join(os.path.dirname(os.path.dirname(__file__)),
                               "vector_store_en/", "")

INIT_KNOWLEDGE_BASES_PATH_ZH = os.path.join(os.path.dirname(os.path.dirname(__file__)),
                               "vector_store_zh/國泰人壽[illegible]/", "")

KNOWLEDGE_BASES_PATH_ZH = os.path.join(os.path.dirname(os.path.dirname(__file__)),
                               "vector_store_zh/", "")

# 基于上下文的prompt模版，请务必保留"{question}"和"{context}"


PROMPT_TEMPLATE = {
            "Chinese": """<指令>根據已知信息，簡潔和專業的來回答問題。如果無法從中得到答案，請說 "根據已知信息無法回答該問題"，不允許在答案中添加編造成分，輸出答案使用繁體中文語言。 </指令>
<已知信息>{{ context }}</已知信息>,
<問題>{{ question }}</問題>
""",
            "English":  """<INSTRUCTION> Answer the QUESTION based on the KNOWN_INFORMATION in English. If the answer cannot be obtained from the KNOWN_INFORMATION, please say "Unable to answer the question based o
n the known information". It is not allowed to add fabricated content. The answer must be in English. </INSTRUCTION>
<KNOWN_INFORMATION>{{ context }}</KNOWN_INFORMATION>
<QUESTION>{{ question }}</QUESTION>"""}

# TextSplitter配置项，如果你不明白其中的含义，就不要修改。
text_splitter_dict = {
    "ChineseRecursiveTextSplitter": {
        "source": "huggingface",  # 选择tiktoken则使用openai的方法,不填写则默认为字符长度切割方法。
        "tokenizer_name_or_path": "",  # 空格不填则默认使用大模型的分词器。
    },
    "SpacyTextSplitter": {
        "source": "huggingface",
        "tokenizer_name_or_path": "gpt2",
    },
    "RecursiveCharacterTextSplitter": {
        "source": "tiktoken",
        "tokenizer_name_or_path": "cl100k_base",
    },
    "MarkdownHeaderTextSplitter": {
        "headers_to_split_on": [
            ("#", "head1"),
            ("##", "head2"),
            ("###", "head3"),
            ("####", "head4"),
        ]
    },
}

TEXT_SPLITTER_NAME = "RecursiveCharacterTextSplitter"
                                                                              1,1           Top
```

  2. Enter “i” to edit the document, modify CHUNK_SIZE and OVERLAP_SIZE



```
TEXT_SPLITTER_NAME = "RecursiveCharacterTextSplitter"

CHUNK_SIZE = 350
```

  3. After that, enter “Esc”
  4. Save and leave: enter “: wq” and then click “Enter”

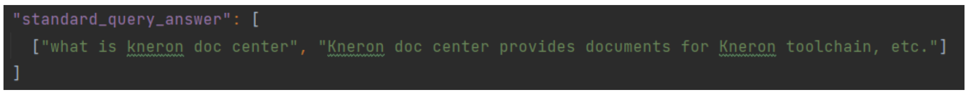
4.3. Standard questions and answers

Customer can also setup standard question and answers.

- For standard questions, chatbots will answer according to standard answers. After the user modifies the custom content, it is necessary to restart the service in the background to complete the model initialization, so that the user-defined content will be updated to the chatbot.
- The user's own definition:

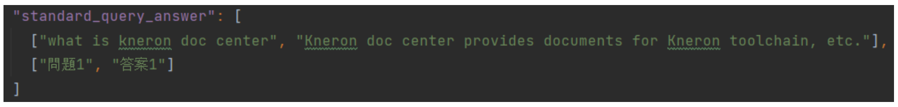
User also can modify templates.json to add custom defined standard questions and answers.

In the templates.json add self-determination standard questions and answers templates.json is located at kneron_chatbot_prod/kneron_doc_chat/kneron_software/configs directory. Users can add more standard questions and answers in the following paragraphs.



```
"standard_query_answer": [
  ["what is kneron doc center", "Kneron doc center provides documents for Kneron toolchain, etc."]
]
```

For example, to add new question1 and answer 1



```
"standard_query_answer": [
  ["what is kneron doc center", "Kneron doc center provides documents for Kneron toolchain, etc."],
  ["問題1", "答案1"]
]
```

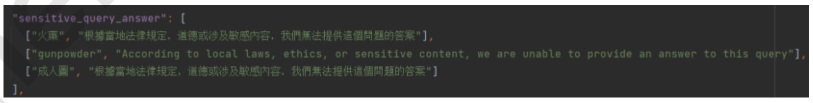
After modification, it is necessary to restart the service in the backend to complete the model initialization, so that the user-defined content will be updated to the chatbot.

4.4. Sensitive words and answers

Customer can adjust sensitive words and answers.

- For sensitive words, the chatbot will answer according to the standard answer. After the user customizes and modifies the content, it is necessary to restart the service in the backend to complete the model initialization, so that the user-defined content will be updated to the chatbot.
- The user's own definition
  Users can customize sensitive words and answers by modifying the templates.json file. After modification, the service must be restarted in the backend to complete the model initialization, so that the user-defined content will be updated to the chatbot.

  Add self-defined sensitive words and answers to the templates.json templates.json The file is located in the kneron_chatbot_prod/kneron_doc_chat/kneron_software/configs directory, and the templates.json the file name and location of this file can not be modified currently. Users can add more sensitive words and answers in the following paragraphs.



```
"sensitive_query_answer": [
  ["火藥", "根據當地法律規定，道德或涉及敏感內容，我們無法提供這個問題的答案"],
  ["gunpowder", "According to local laws, ethics, or sensitive content, we are unable to provide an answer to this query"],
  ["成人圖", "根據當地法律規定，道德或涉及敏感內容，我們無法提供這個問題的答案"]
],
```

After the user customizes and modifies the content, it is necessary to restart the service in the backend to complete the model initialization, so that the user-defined content will be updated to the chatbot.

4.5. Keyword match

Customer can set their own keywords and corresponding file.

- If keywords appear in the question, the chatbot will first search for results in the corresponding file. For example, you have a keyword “Kneron API”, and set the related file to be

"Kneron_api.txt". If your question contains the keyword, the chatbot will check if the file "Kneron_api.txt" is in the knowledge base you are using and raise the priority of results from this file. It is OK if you do not have the file in your current knowledge base.

- The user's own definition
  Users can add/change keywords and files by modifying the keyword_database.json file. The file is located in the kneron_chatbot_prod/kneron_doc_chat/knowledge_base/configs directory. Users should change the keywords in the form of json dictionary. The key is the keyword, the values is the related file (only a single string with one file name allowed).

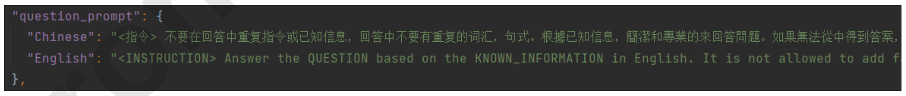
  After modification, the service must be restarted in the backend to complete the model initialization, so that the user-defined content will be updated to the chatbot.

4.6. Problem reminder and prompt

Users can modify the prompts used for Q&A: problem prompts, free-form question and answer system prompts, and system prompts used for the knowledge base.

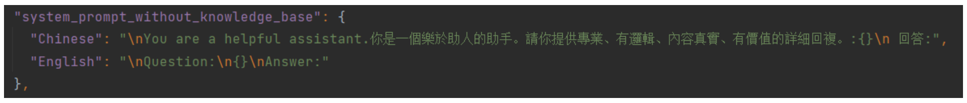
4.6.1 Problem Reminder
User can modify templates.json to adjust problem reminder

4.6.2 Prompt adjustment
- Question-prompt adjustment: in the file templates.json, which is located at kneron_chatbot_prod/kneron_doc_chat/kneron_software/configs folder, user can modify the question-prompt or system prompt to adjust the response from large language model. After modified, you need to re-launch all services to enable the new changes.



```
"question_prompt": {
  "Chinese": "<指令> 不要在回答中重复指令或已知信息，回答中不要有重复的词汇，句式。根據已知信息，簡潔和專業的來回答問題。如果無法從中得到答案，
  "English": "<INSTRUCTION> Answer the QUESTION based on the KNOWN_INFORMATION in English. It is not allowed to add f
},
```

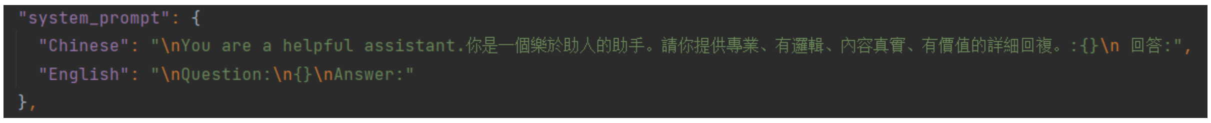
- Systematic prompting of free LLMchat: The free LLM chat is defined as chat with LLM without using knowledge base database. Users can also modify the template.json to adjust the system prompt. In the templates.json, which is located at
  i. kneron_chatbot_prod/kneron_doc_chat/kneron_software/configs folder, users can modify this in the following paragraphs.



```
"system_prompt_without_knowledge_base": {
  "Chinese": "\nYou are a helpful assistant.你是一個樂於助人的助手。請你提供專業、有邏輯、內容真實、有價值的詳細回複。:{}\n 回答:",
  "English": "\nQuestion:\n{}\nAnswer:"
},
```

○ System prompts for knowledge based Q&A:
  In the templates.json, which is located at

  kneron_chatbot_prod/kneron_doc_chat/kneron_software/configs folder, users can modify this in the following paragraphs. Users can modify this in the following paragraphs. After modified, you need to re-launch all services to enable the new changes.



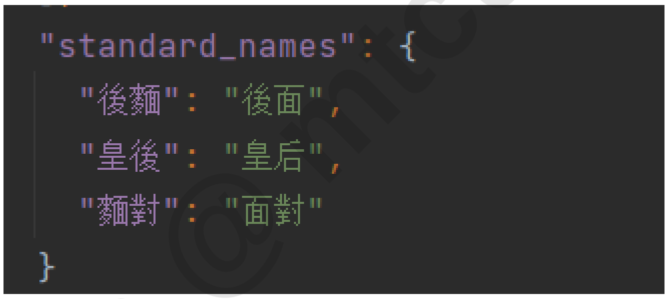
```
"system_prompt": {
  "Chinese": "\nYou are a helpful assistant.你是一個樂於助人的助手。請你提供專業、有邏輯、內容真實、有價值的詳細回覆。:{}\n 回答:",
  "English": "\nQuestion:\n{}\nAnswer:"
},
```

4.7. Standard name replacement

Users can add the placement for standard names, which means specific words will be replaced by standard names.

- Update dictionary for standard names in json
  User can modify templates.json to adjust standard names. If any key in standard_names is found in answer, the key in the answer will be replaced by standard name. There are some examples shown as below. For example, if 後麵 is found in answer, it will be replaced by 後面.

  Templates.json is located at kneron_chatbot_prod/kneron_doc_chat/kneron_software/configs folder.



```
"standard_names": {
  "後麵": "後面",
  "皇後": "皇后",
  "麵對": "面對"
}
```

4.8. Company Q&A

Customer can use their own Q&A database in our chatbot. Once the database is imported and the company Q&A config option is set to True, the chatbot will answer users' query based on given database.

4.8.1. How to process company QA database:

I. The database must be converted to a csv file before importing. The last column must be the answer, and the second-to-last column must be the question. To import the database, first copy the csv file to the edge box. Then use the python script company_qa_db_creator.py and pass the csv file path and language as arguments.

II. For example, if your file is located at /data/example.csv, you can use the following commands:

i) cd /data/kneron_chatbot_prod/kneron_doc_chat
ii) source ../envsetting.sh
iii) python3 company_qa_db_creator.pyc

/data/example.csv English

4.8.2. Config setup:

To enable the company Q&A feature, user must locate the parameters USE_COMPANY_QA and ONLY_COMPANY_QA in /home/linaro/kneron_chatbot_prod/kneron_doc_chat/knowledge_base/configs/model_config.py

When the USE_COMPANY_QA is set to **True**, the chatbot will answer users' question if it finds similarity from company database. This feature will be enabled only under knowledge base Q&A mode. However, if a question is not related to those in company database, the chatbot will decide whether continue to provide free chat answer or knowledge base Q&A answer according to ONLY_COMPANY_QA.

When USE_COMPANY_QA and ONLY_COMPANY_QA are both set to True, once the company Q&A session is finished, the chatbot will stop responding regardless of the question is answerable by company database.