HSE UNIVERSITY · EST 1992 ·

Faculty of
Computer Science

Data Science and
Business Analytics

Moscow
2025

Chinese Media Analysis / Анализ китайских новостей

Team Software Project
Completed by: Markin Alexander Andreevich | *DSBA-231*, Sidorova Alika Igorevna | *CDS-232*, Dzhanobilov Tehron Sherozovich | *AMI-2312*
Supervisor: Kolesnichenko Elena Yurievna, Associate Professor @ *HSE Faculty of Computer Science, Big Data and Information Retrieval School*
Co-supervisor: Rybalko Dmitry Mikhailovich,
Product Architect for ML services @ *Yandex*

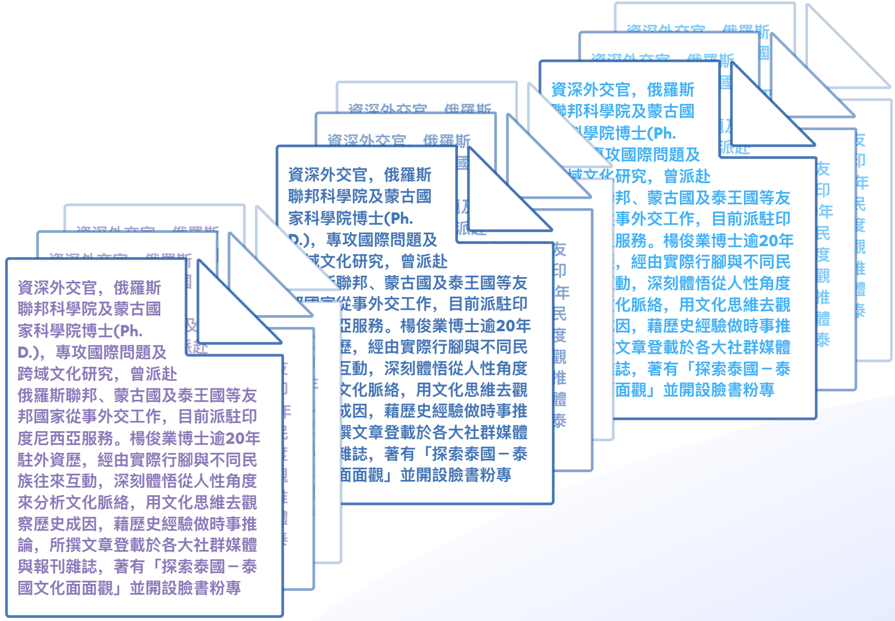
你
好！如果你正在读这篇文
章，非常感谢！我们为这个项目付出了
很长时间的努力！它占据了我们第二年相当长
一段时间的生活，我们为此感到非常自豪。
作为高等经济学院的一名女学生，我要特别感谢我的队友、
莫斯科国立大学的亚历山大·巴格罗夫和叶戈尔·杰尼索夫，以及来
自 *Yandex* 和俄罗斯科学院东方研究所的导师和管理人员。
如果您还在阅读这篇文章，我们也想感谢您对我们项目的关注。我们很高
兴您对此项目如此感兴趣，并随时准备解答任何相关问题。
PS：以上句子可能存在翻译错误。

Subject matter

Over the past few decades, the presence of a certain **trend** in **media consumption** has become increasingly obvious:

There are **more news sources** than ever being read and interpreted worldwide, each with their own **biases** (both intentional and unintentional).

This makes it **ever so hard** for people to **properly parse information** about what’s happening around them.


資深外交官，俄羅斯
聯邦科學院及蒙古國
家科學院博士(Ph.
D.)，專攻國際問題及
跨域文化研究，曾派赴
俄羅斯聯邦、蒙古國及泰王國等友
邦國家從事外交工作，目前派駐印
度尼西亞服務。楊俊業博士逾20年
駐外資歷，經由實際行腳與不同民
族往來互動，深刻體悟從人性角度
來分析文化脈絡，用文化思維去觀
察歷史成因，藉歷史經驗做時事推
論，所撰文章登載於各大社群媒體
與報刊雜誌，著有「探索泰國－泰
國文化面面觀」並開設臉書粉專

Subject matter

Chinese news sources, in particular, get **pushed away** from the sociopolitical limelight more than others in Europe, in no small part due to the **linguistic and cultural barriers**.

At the same time, China plays a **significant role in international relations**, which raises the importance of properly **researching** not only the external, but also the **internal relations** of its citizens.

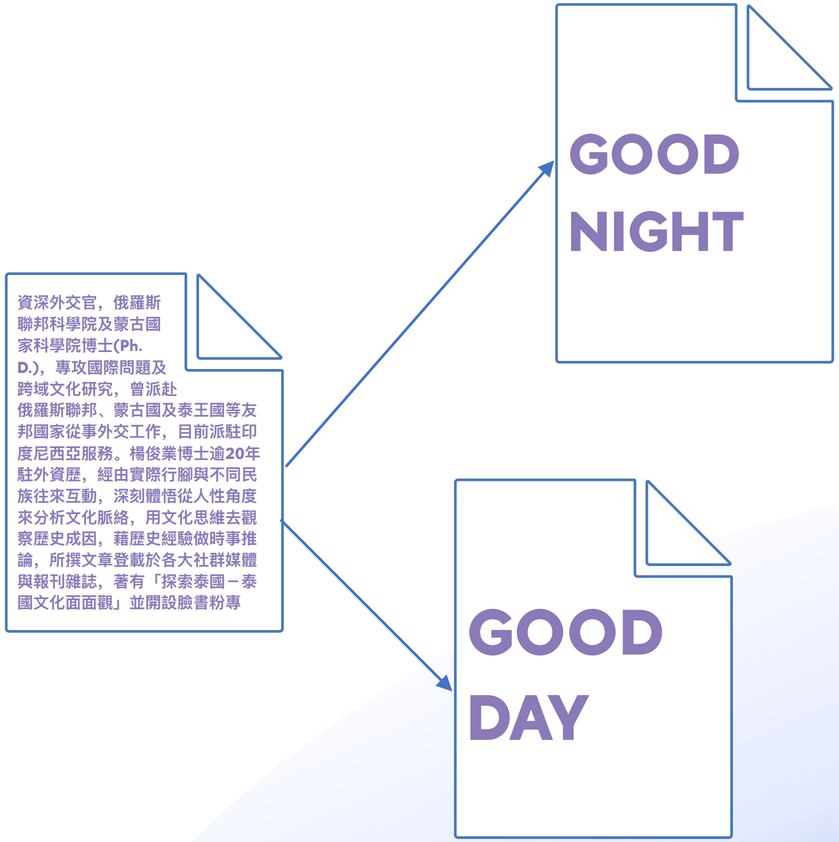
Given that mass media has now become, in some ways, the **cornerstone** of societal perceptions of the world, it's clear that research should properly utilise **tools for analysing** said media.

資深外交官，俄羅斯
聯邦科學院及蒙古國
家科學院博士(Ph.
D.)，專攻國際問題及
跨域文化研究，曾派赴
俄羅斯聯邦、蒙古國及泰王國等友
邦國家從事外交工作，目前派駐印
度尼西亞服務。楊俊業博士逾20年
駐外資歷，經由實際行腳與不同民
族往來互動，深刻體悟從人性角度
來分析文化脈絡，用文化思維去觀
察歷史成因，藉歷史經驗做時事推
論，所撰文章登載於各大社群媒體
與報刊雜誌，著有「探索泰國－泰
國文化面面觀」並開設臉書粉專

Subject matter

Currently, however, there are multiple **roadblocks** which halt most serious attempts at **wide-range research**:

- **Hands-on** analysis of each article **doesn't scale well** in the current social climate
- Most **automatic translation** tools transcribe Chinese names in their own ways which can lead to **conflicting readings** of the same name
- Websites which host Chinese news articles **rarely** have **open APIs** or **standardised layouts**, making it highly complex to analyse vast amounts of data



資深外交官，俄羅斯聯邦科學院及蒙古國家科學院博士(Ph. D.)，專攻國際問題及跨域文化研究，曾派赴俄羅斯聯邦、蒙古國及泰王國等友邦國家從事外交工作，目前派駐印度尼西亞服務。楊俊業博士逾20年駐外資歷，經由實際行腳與不同民族往來互動，深刻體悟從人性角度來分析文化脈絡，用文化思維去觀察歷史成因，藉歷史經驗做時事推論，所撰文章登載於各大社群媒體與報刊雜誌，著有「探索泰國－泰國文化面面觀」並開設臉書粉專

Importance of Innovation

- Scientists focused on **research** of **China** and its relations with other countries currently have to **rely on** tools built with **Western media** and **languages** in mind
- There are **few competent** tools for **accumulating**, **processing** and **analysing** Chinese news data
- There is **increased** interest around China due to its key **geopolitical role**

As it currently stands, LLM-assisted research tools are not as omnipresent in Oriental research as in other fields

Initial pitch

In coordination with *Yandex*, the *Institute of Oriental Studies of the Russian Academy of Sciences* reached out to students with the intention of **developing a tool** which would **solve** the aforementioned **problems in research**.

Upon completion, it **would allow** the Institute's research team, as well as anyone interested in the tool, to **increase** their **efficiency** and **accuracy** in **researching Chinese media**.

Core Objective & Goals

Core Objective:

Develop an **online service** for **fetching**, **processing** and **analysing** Chinese media using **RAG**.

Upon completion, the service should be launched on the **IOS RAS website**.

Goals:

● **Data collection**

Collect *HTML* article data from Chinese websites using existing and new solutions

● **Data processing and storage**

Retrieve article texts from *HTML* files and translate them into English, then store the resulting files in a database

● **RAG workflows**

Develop multiple implementations of the *RAG* workflow for further assessment by the Institute

● **Containerisation**

Separate the codebase into containers to accommodate for scaleability and easier deployment

Comparison of Alternatives

Initially, we had to **analyse alternative solutions**.

For this, a **matrix analysis** approach was chosen. Services were assessed on a number of factors we deemed important, such as:

- Is it **free** to use**?**
- How well does it **perform** with the **Chinese language**?
- Does it **provide** article **texts**?
- Are its **analysis** features **competent**?

Comparison of Alternatives - Matrix View

Factors \ **Alternatives**	**Google News API**	**Aylien News API**	**NewsAPI.org**	**IBM Watson Discovery**	**Dataminr**
Cost	*Relatively high*	Adequate	Adequate	*High licensing + resource costs*	*High*
Chinese language performance	*Poor due to lack of indexed websites*	*Poor*	*Poor due to lack of Chinese sources*	Depends on the model	*Poor due to a focus on English sources*
Provides article texts?	*Only basic metadate*	Yes	*Only headlines and abstracts*	Yes	*No, operates on accumulated data*
AI/Analysis features	*Only keyword/ region/topic filtering*	NLP, Sentiment analysis, etc.	*Only filtering*	A variety of analysis features	A variety of best-in-class features

Comparison of Alternatives

As a result of this comparison, we have outlined a number of key problems which have to be avoided in our solution:

- **High** setup & maintenance **costs**
- **Poor** Chinese **language performance**
- **Incomplete** data **storage**

This demonstrates the uniqueness of our solution, as well as the need for it.

Functional Requirements

Based on previous analysis, we have arrived at a set of functional requirements for the app:

- Fetch HTML data from websites
- Parse HTML files into plaintext article texts
- Translate articles into English
- Embed and store articles for Embedding-based RAG
- Build a graph and detect communities for Graph RAG
- Implement a user account/history system
- Properly containerise the app
- Construct an intuitive frontend

Functional Requirements

These requirements can be logically separated into thematic groups:

○ **Fetch HTML data from websites**	**Data Processing**
○ **Parse HTML files into plaintext article texts**	
○ **Translate articles into English**	
○ **Embed and store articles for Embedding-based RAG**	**Embedding-based RAG**
○ **Build a graph and detect communities for Graph RAG**	**Graph RAG**
○ **Implement a user account/history system**	**User System**
○ **Properly containerise the app**	**Containerisation**
○ **Construct an intuitive frontend**	**Frontend design**

Data Processing

Chinese-English Translator:

We assessed multiple available LLM models on the basis of Chinese language performance, pricing and fine-tuning availability, finally arriving on the choice of *Llama 3.1*, accessed via *Yandex Foundation Models.*

Website Parser:

We ended up using *BeautifulSoup4* in conjunction with our self-made tool.

Website Content Loader:

Due to most websites using *JavaScript*, we opted for the Python library *Selenium*, as it's the most feature complete and regularly updatable option.

Embedding-based RAG

Embedder:

By far, the most easily available and cheap approach in our case was to use *YandexGPTEmbeddings* via *LangChain* - both for the articles and the queries.

Database:

As for the vector database, *ChromaDB* was chosen due to its versatility and open-source nature, as well as ease of user

YandexGPT

Chroma

Assistant-based RAG

Since the assistant-based RAG was created for the sole purpose of testing Yandex's "Assistant with Search Index" functionality, it utilises the assistant available via *Yandex Cloud ML SDK.*

YandexGPT

Graph RAG

Database:

Neo4j was chosen as the database due to its adequate pricing and the abundance of data analysis and visualisation features.

Named Entity Recognition:

For NER, two separate approaches were used together: a function from the *spaCy* Python library for algorithmic NER and the *YandexGPTPro* model available via *LangChain* for LLM-assisted NER.

Community Detection:

Community detection was performed using the *Leiden* algorithm, specifically its implementation in the *GraphDataScience* package for *Neo4j*.

Community Summarisation:

In order to summarise information within each community, *YandexGPTPro* was once again used via *LangChain*.

User System

As the user system would not include any vectors or graphs, we decided to opt-in for the "industry standard" - *PostgreSQL*.

Frontend

Frontend was built on the *Streamlit* library for Python due to its ease of use and built-in optimisations for a variety of different device forms.

Containerisation

Just like with the user database, we chose to opt for an industry standard - *Docker*.

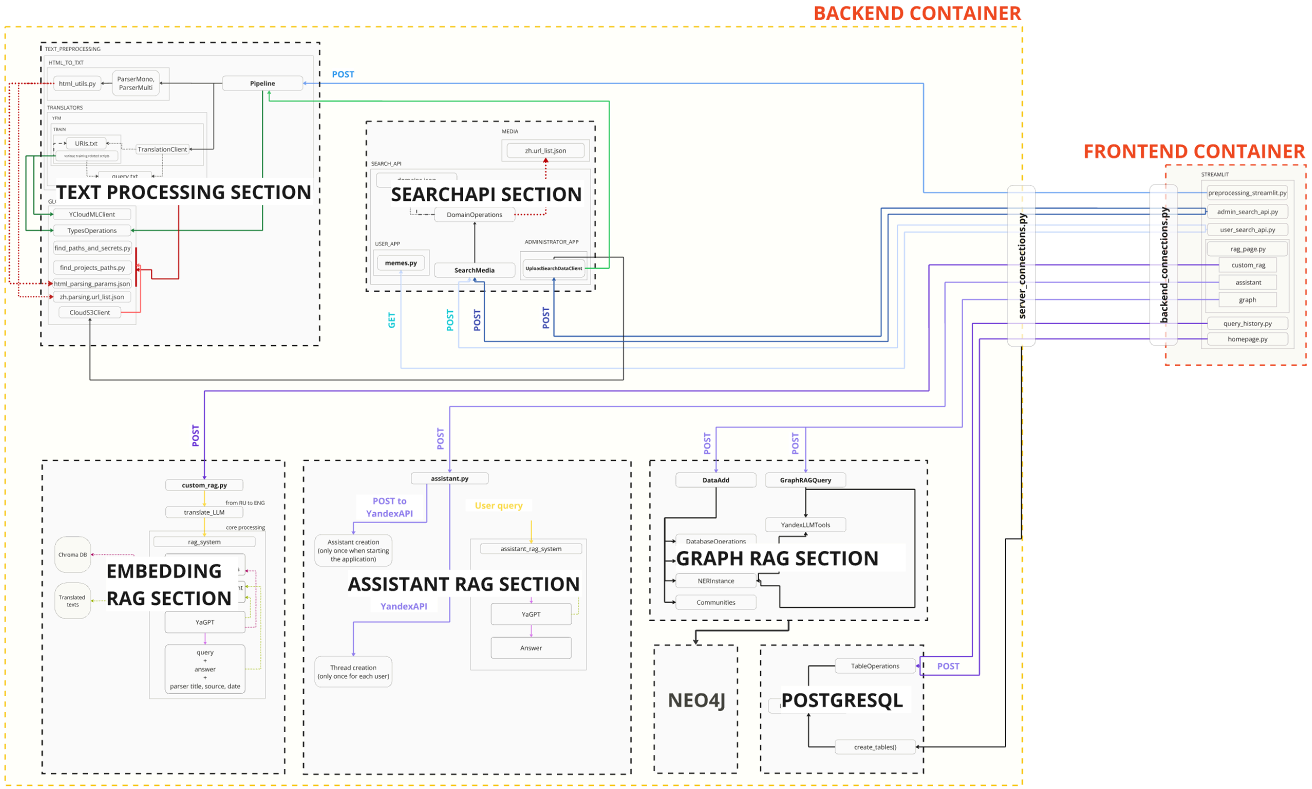
Program Architecture

Our program is separated into two main blocks:

- Backend, accessible via a *FastAPI* server
- Frontend, which runs on *Selenium*

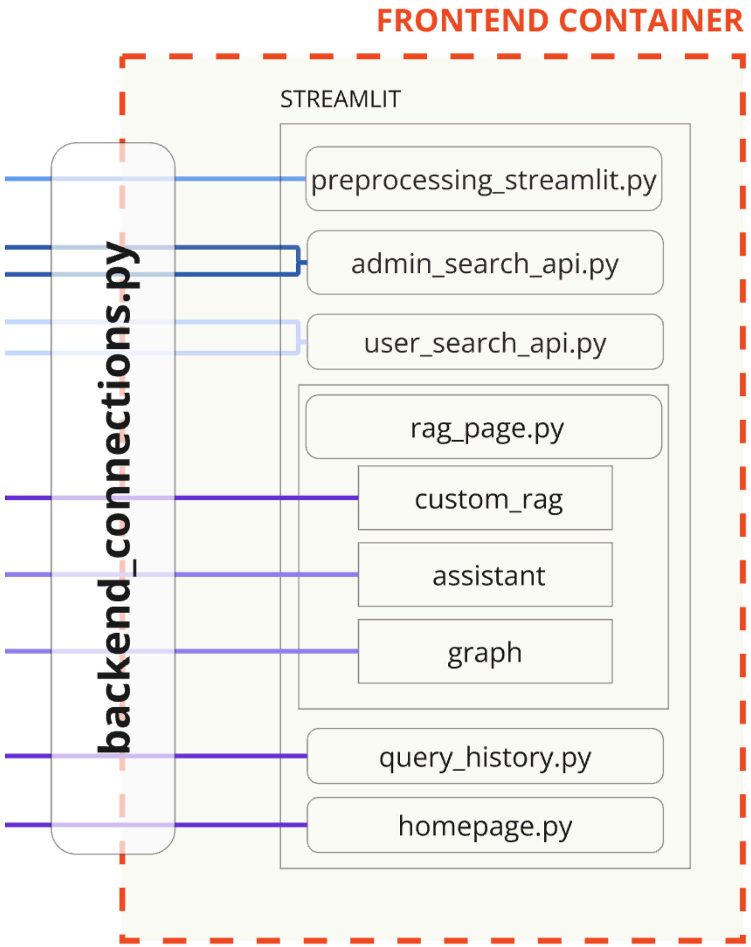
Additionally, there are three other services which the backend connects to:

- the *Neo4j* container
- the *PostgreSQL* container
- Shared data volume for some file-based operations between the frontend and the backend



Program Architecture: Frontend

The frontend consists of multiple *Streamlit* UI files for a variety of views accessible to users, all of which send user requests to the backend via the *backend_connections* file

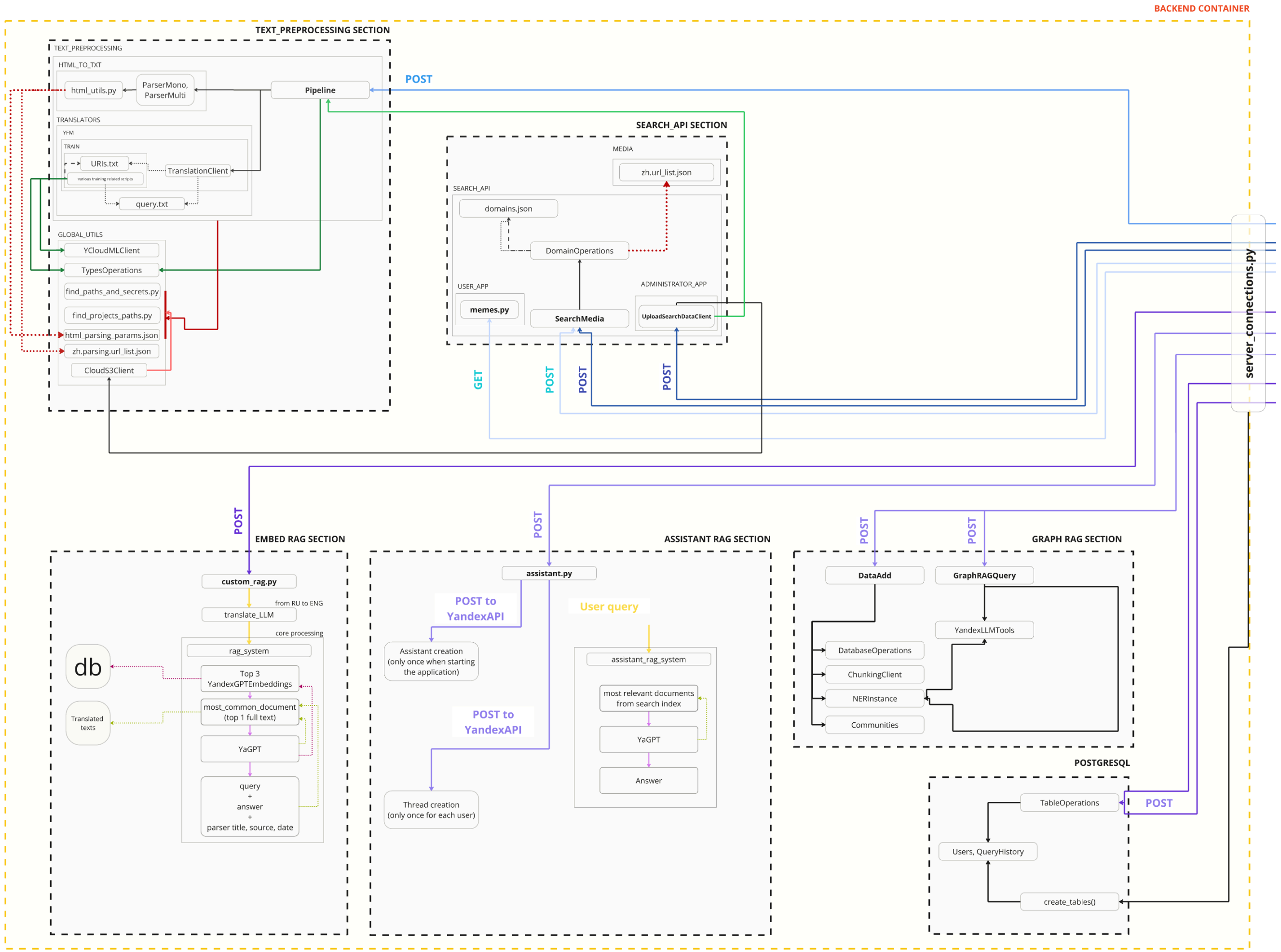


Program Architecture: Backend

The backend consists of multiple large sub-modules, each with their own group of classes.

These modules include:

- the Text Processing module
- the Search API module
- the User Database module
- the Graph RAG module
- the Assistant RAG module
- the Embedding RAG module



Implementation: Text Processing

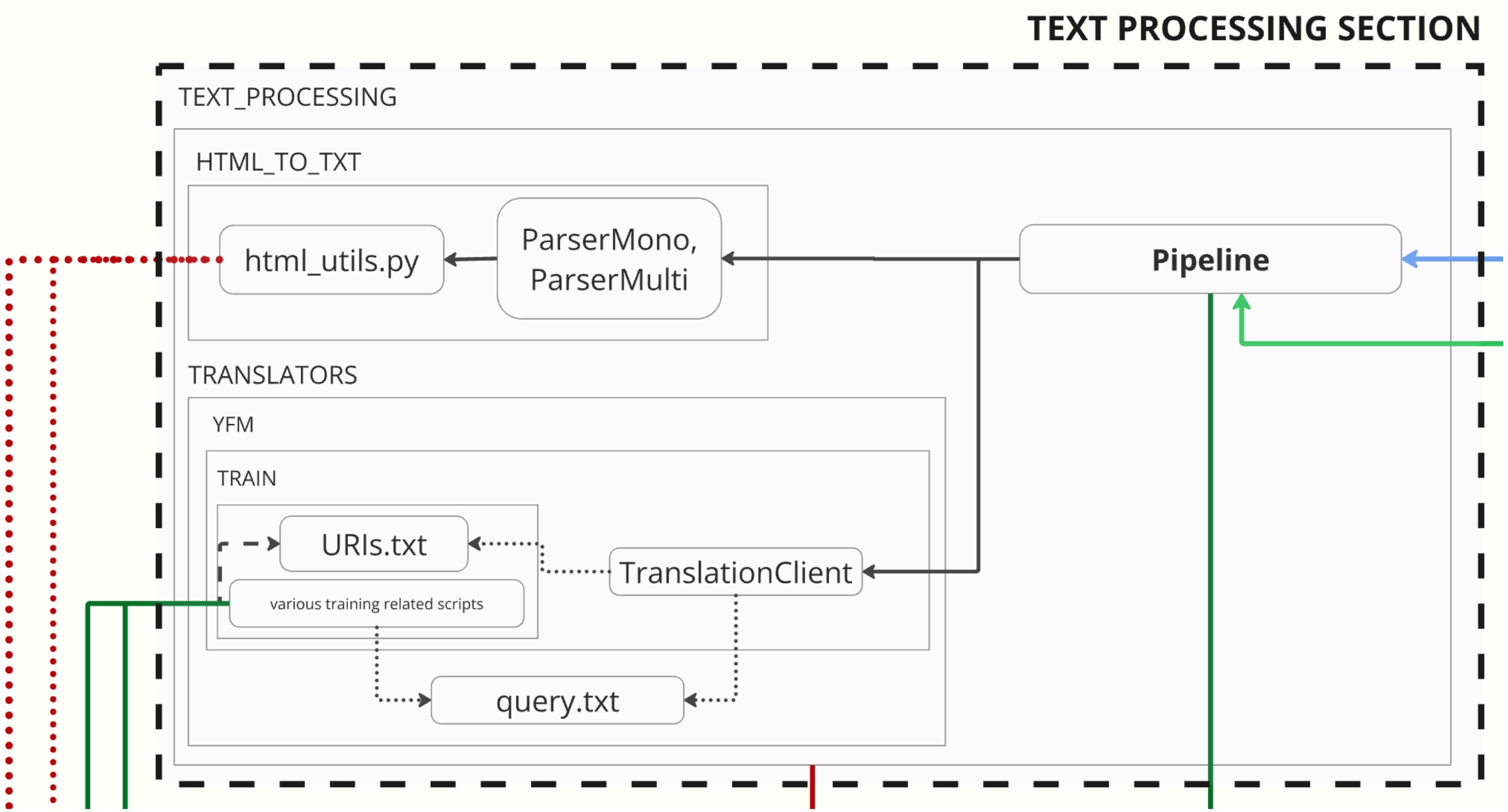
The Text Processing module consists of a parser and a translator.

The parser functionality is enabled by the **ParserMono** and **ParserMulti** classes. They utilise *BeautifulSoup 4* and aim to algorithmically parse any given *HTML* file. The latter of the two parses *warc.gz* files with *HTML* data.

Translation is enabled via the **TranslationClient** class, which checks the local storage for the latest fine-tuned translation model and runs a given text along with a system prompt through it.

The products of the two classes are combined in the **Pipeline** class, the methods of which consecutively create **Parser_** and **TranslationClient** objects, later saving the output to shared storage and returning the paths to the created files.

Implementation: Text Processing



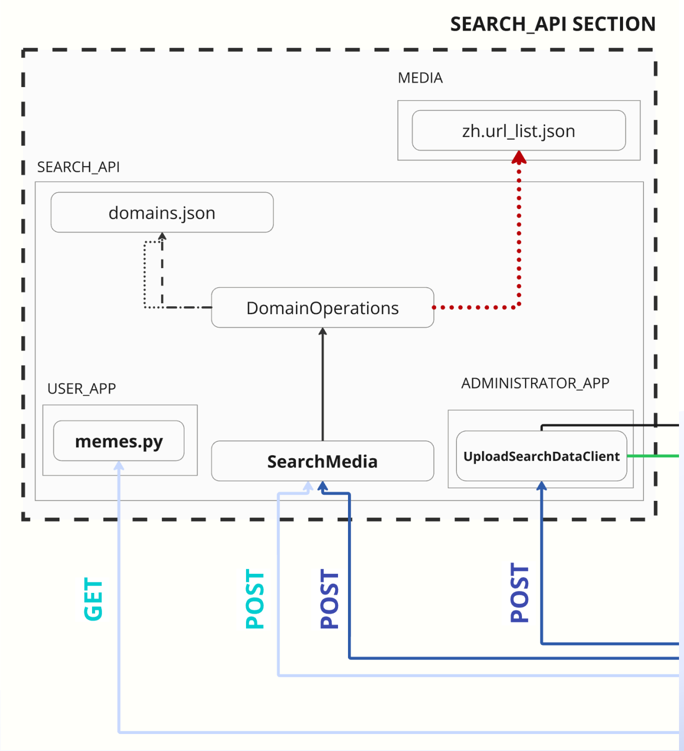
Implementation: SearchAPI

The SearchAPI section also houses three classes:

DomainOperations works with a local list of Chinese domains, regularly updated by researchers and allows one to select domains with publicly available search pages.

SearchMedia is responsible for going through these domains and finding articles relevant to a given user query using *Selenium* for loading the pages and *BeautifulSoup 4* for locating the links to search results.

Finally, **UploadSearchDataClient** provides a connection to the **Pipeline** class for processing the data fetched by **SearchMedia's** *.search_api*() method



Implementation: GraphRAG

GraphRAG happens to be the most dense of the sections, with a significant chunk of the code consisting of *Cypher* queries for interaction with *Neo4j*.

The two main classes are **DataAdd** and **GraphRAGQuery**, housing methods for adding article data and constructing a reply to a user query respectively.

DataAdd initialises objects of **DatabaseOperations**, **ChunkingClient**, **NERInstance** and **Communities**.

DatabaseOperation consists of basic operations for adding *Document*, *Chunk*, *Entity* nodes and *Relation* edges to the graph.

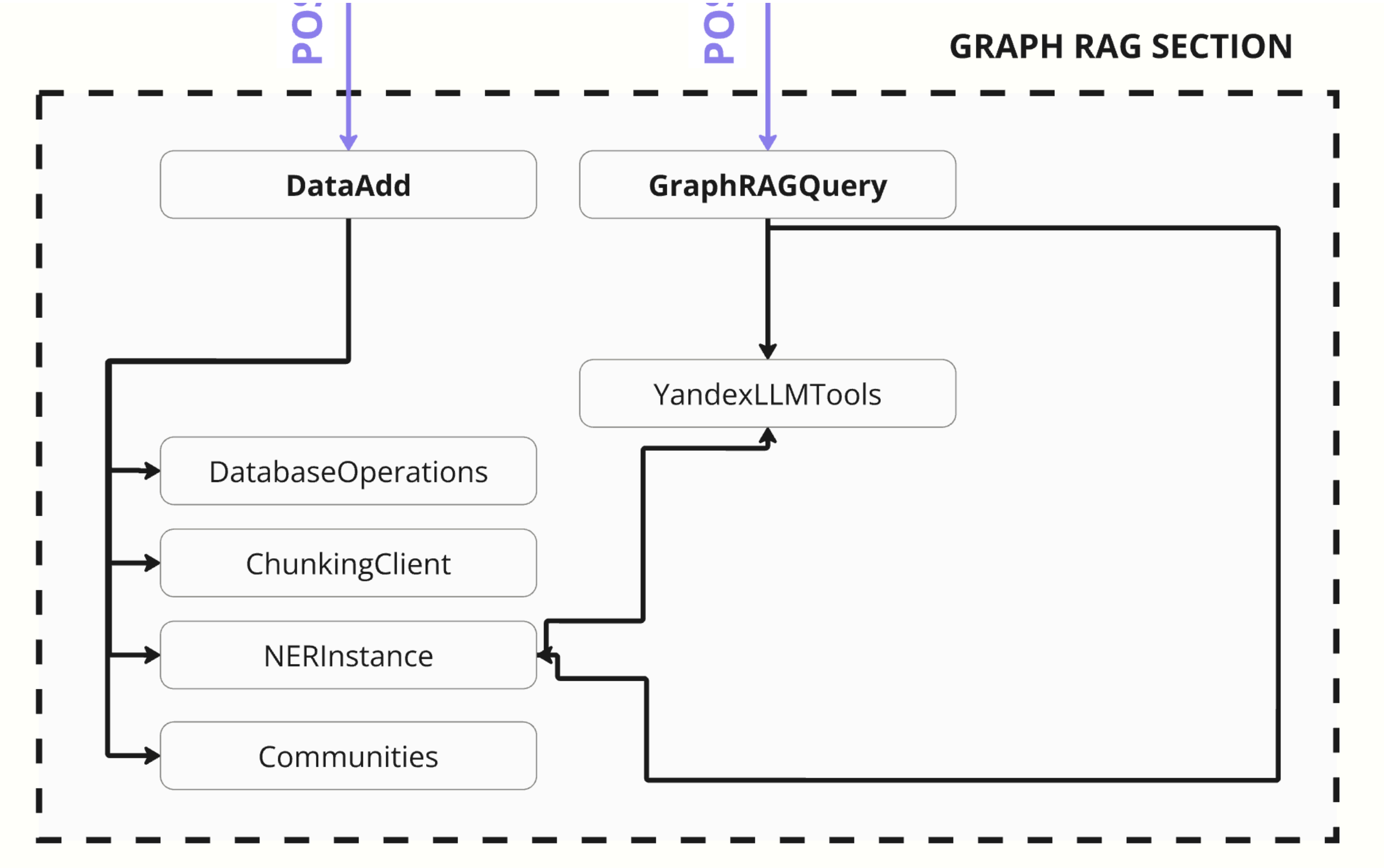
ChunkingClient takes a file as input and separates it into multiple chunks

NERInstance houses methods for Entity Recognition: *.spacy_entity_extraction_basic_list*() extracts entities using *spaCy's* default algorithm, after which *.yandex_chain_entity_extraction_using_list*() and *.ER_lists_of_dicts_retrieval*() use *YandexGPTPro* to extract more entities along with information about them and their relations to each other

Communities uses the *Leiden* algorithm for detecting communities, adds *Community* nodes to the graph and populates them with summaries about the entities related to the community and their relationships

Finally, **YandexLLMTools** initiates a *LangChain* endpoint for *YandexGPTPro*, which is used by other classes in this section

Implementation: GraphRAG



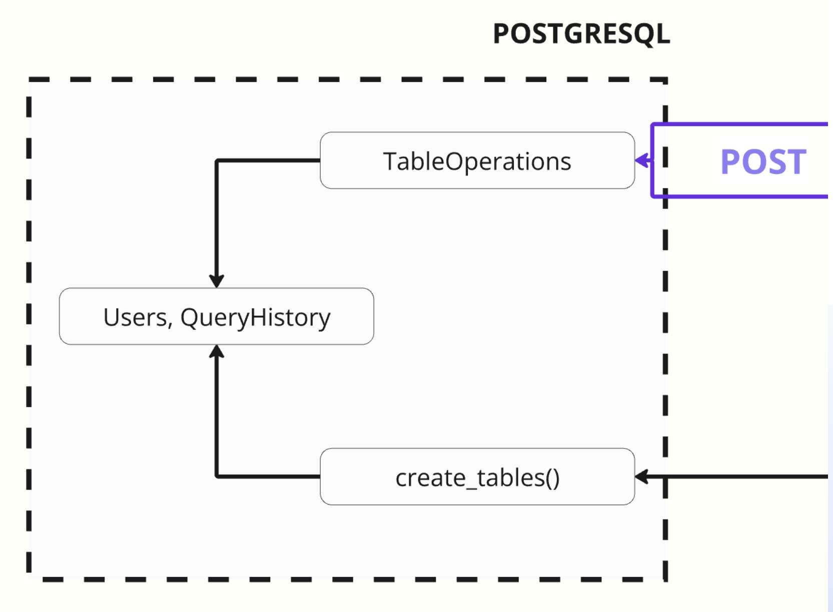
Implementation: User Database

This section is quite straightforward.

There are two *ORM* classes created with the help of the *SQLAlchemy* library - one for each of the tables (**Users**, **QueryHistory**) in the database, as well as a **RoleEnum** class fro enumerating user roles.

There is also a **TableOperations** class, housing all the operations on the tables:

- Creating and removing a user
- Checking user credentials
- Fetching a user's query history, adding records to and removing records from it

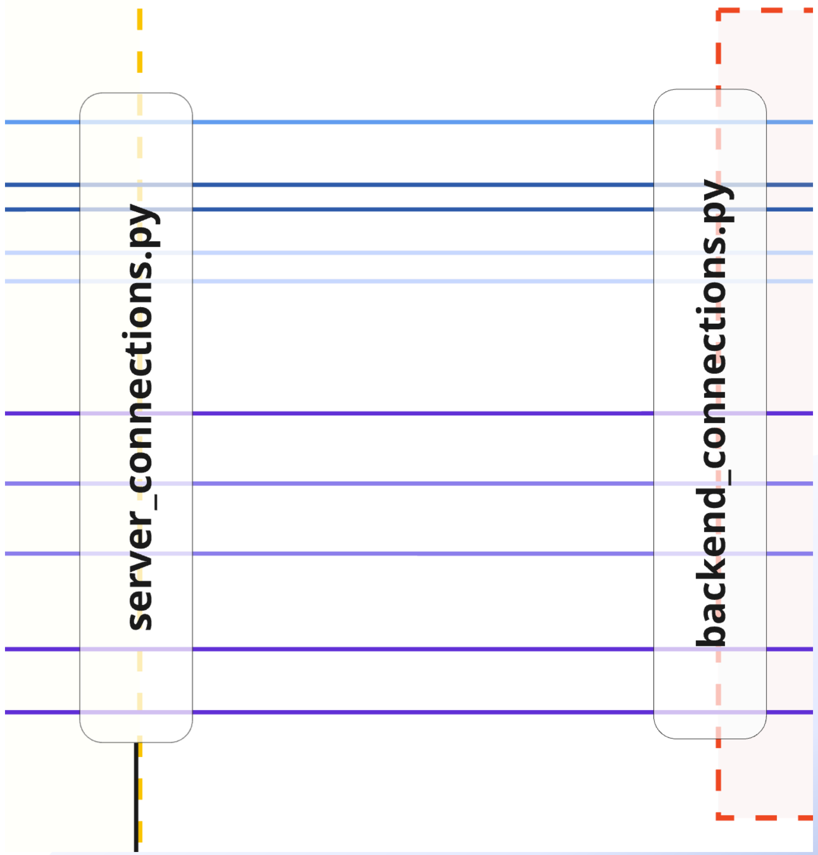


Implementation: Frontend–Backend Communication

Frontend-Backend communication is dependent on two core files: *backend_connections.py* in the Frontend container and *server_connections.py* in the Backend container.

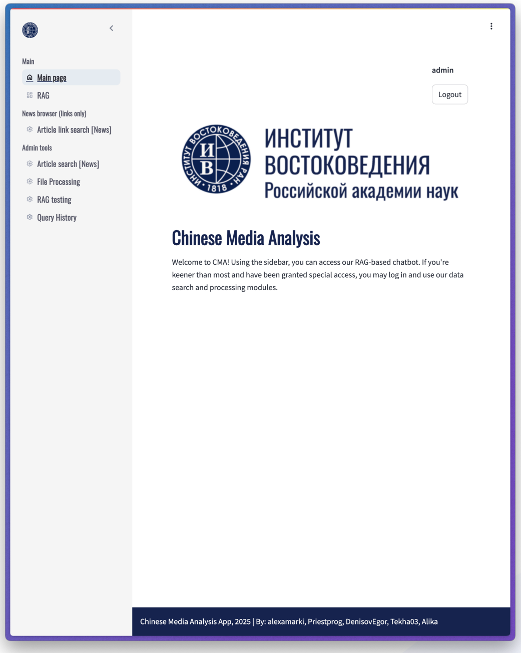
Both of these contain functions related to calls which might be made by a user or the system at some point.

The backend side of the controller also houses *PyDantic* classes for each possible request format, which are then used by the *FastAPI* methods to parse *HTTP* requests from the frontend.



Implementation: Frontend Design

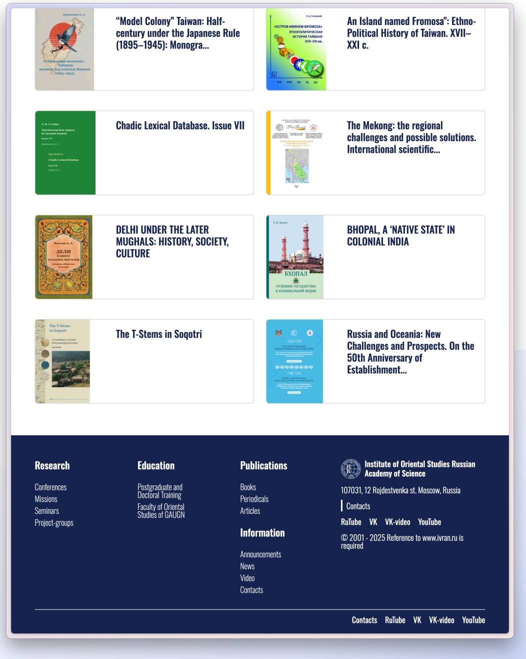
Seeing as the application was meant to exist on the **IOS RAS** website from the very beginning, a few modifications were implemented using *CSS* and the *streamlit_extras* library, bringing the design ever so **closer to** the one found on the **Institute's** other pages



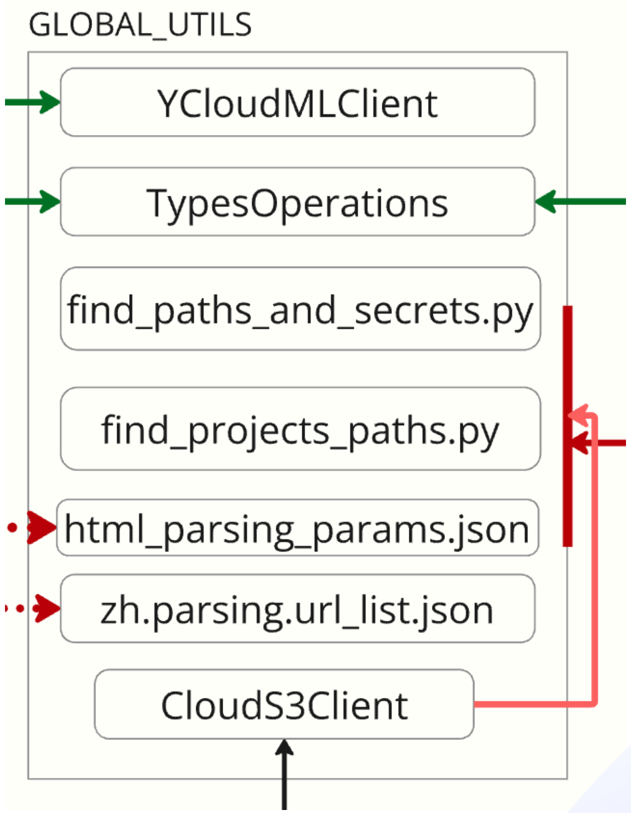
Chinese Media Analysis

Welcome to CMA! Using the sidebar, you can access our RAG-based chatbot. If you're keener than most and have been granted special access, you may log in and use our data search and processing modules.

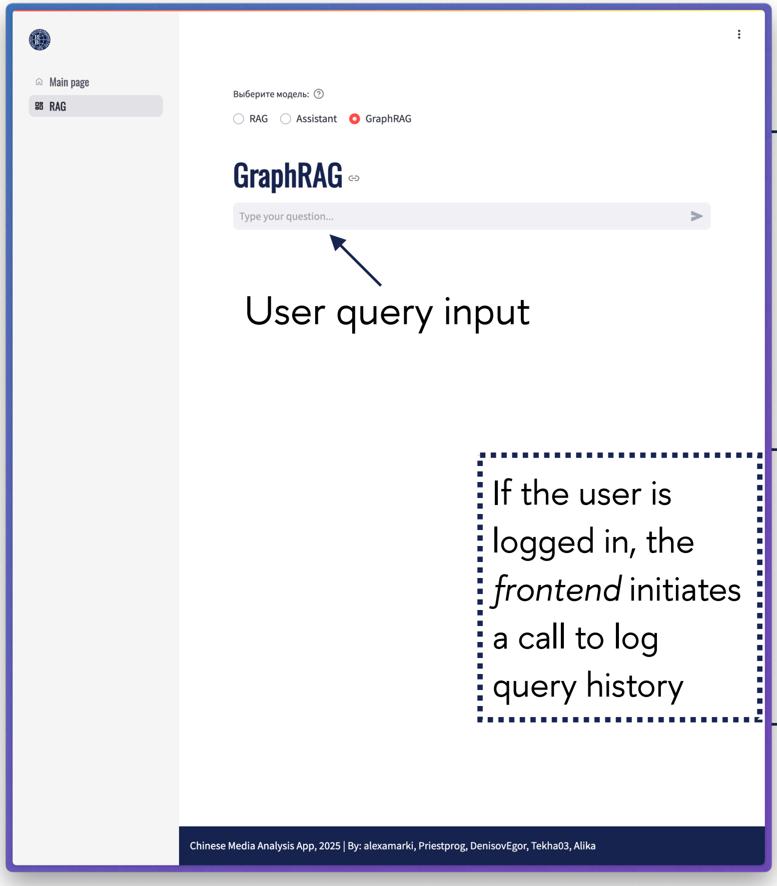
Chinese Media Analysis App, 2025 | By: alexamarki, Priestprog, DenisovEgor, Tekha03, Alika


"Model Colony" Taiwan: Half-century under the Japanese Rule (1895–1945): Monogra...
An Island named Fromosa": Ethno-Political History of Taiwan. XVII–XXI c.
Chadic Lexical Database. Issue VII
The Mekong: the regional challenges and possible solutions. International scientific...
DELHI UNDER THE LATER MUGHALS: HISTORY, SOCIETY, CULTURE
BHOPAL, A 'NATIVE STATE' IN COLONIAL INDIA
The T-Stems in Soqotri
Russia and Oceania: New Challenges and Prospects. On the 50th Anniversary of Establishment...
Research
Conferences
Missions
Seminars
Project-groups
Education
Postgraduate and Doctoral Training
Faculty of Oriental Studies of GAUGN
Publications
Books
Periodicals
Articles
Information
Announcements
News
Video
Contacts
Institute of Oriental Studies Russian Academy of Science
107031, 12 Rojdestvenka st. Moscow, Russia
Contacts
RuTube
VK
VK-video
YouTube
© 2001 - 2025 Reference to www.ivran.ru is required
Contacts
RuTube
VK
VK-video
YouTube

Implementation: Helper classes



Interactions: GraphRAG Query



Main page

RAG

Выберите модель:

○ RAG ○ Assistant ● GraphRAG

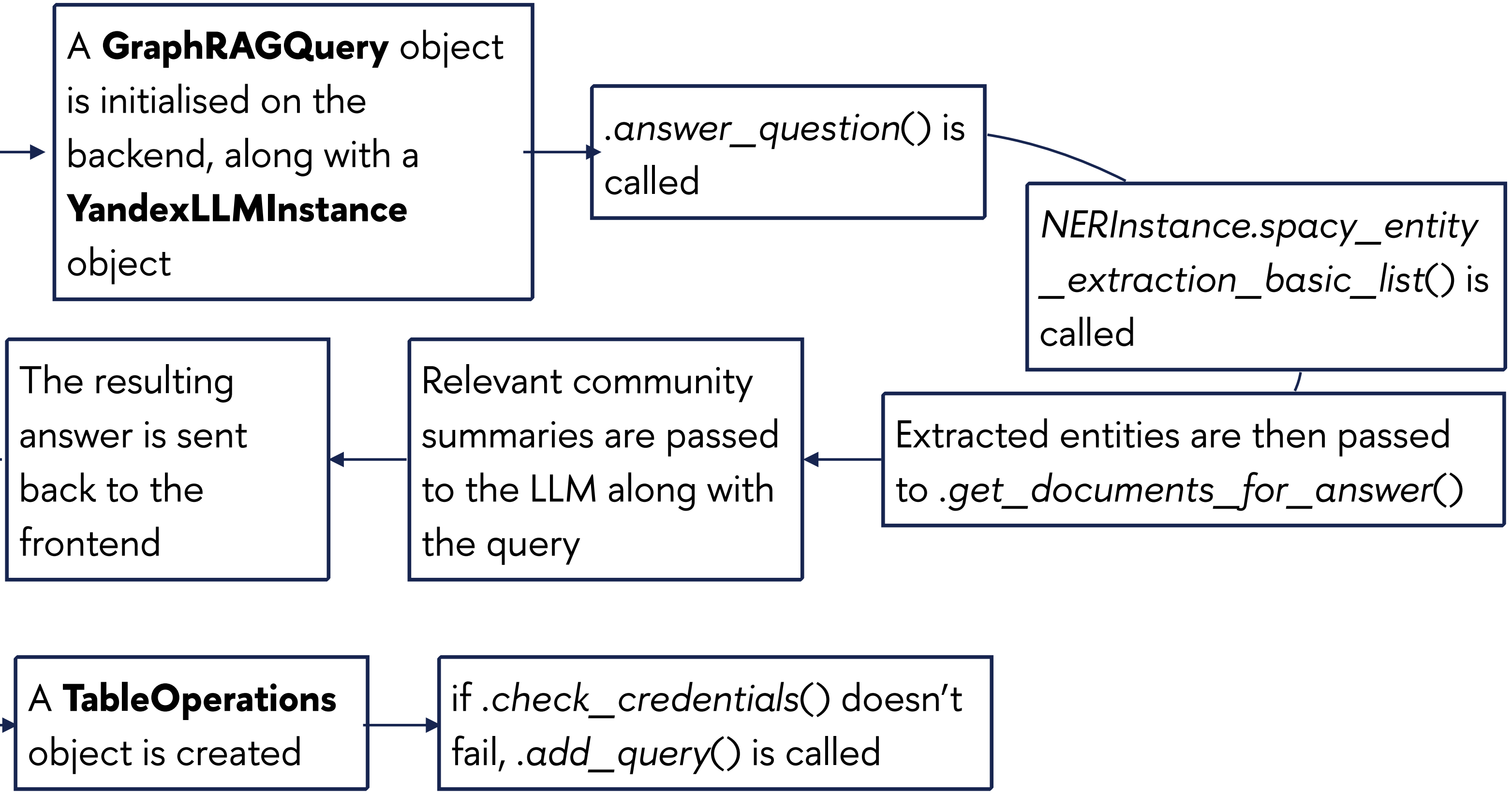
GraphRAG

Type your question...

User query input

If the user is logged in, the *frontend* initiates a call to log query history

Chinese Media Analysis App, 2025 | By: alexamarki, Priestprog, DenisovEgor, Tekha03, Alika



A **GraphRAGQuery** object is initialised on the backend, along with a **YandexLLMInstance** object

.answer_question() is called

NERInstance.spacy_entity_extraction_basic_list() is called

Extracted entities are then passed to *.get_documents_for_answer*()

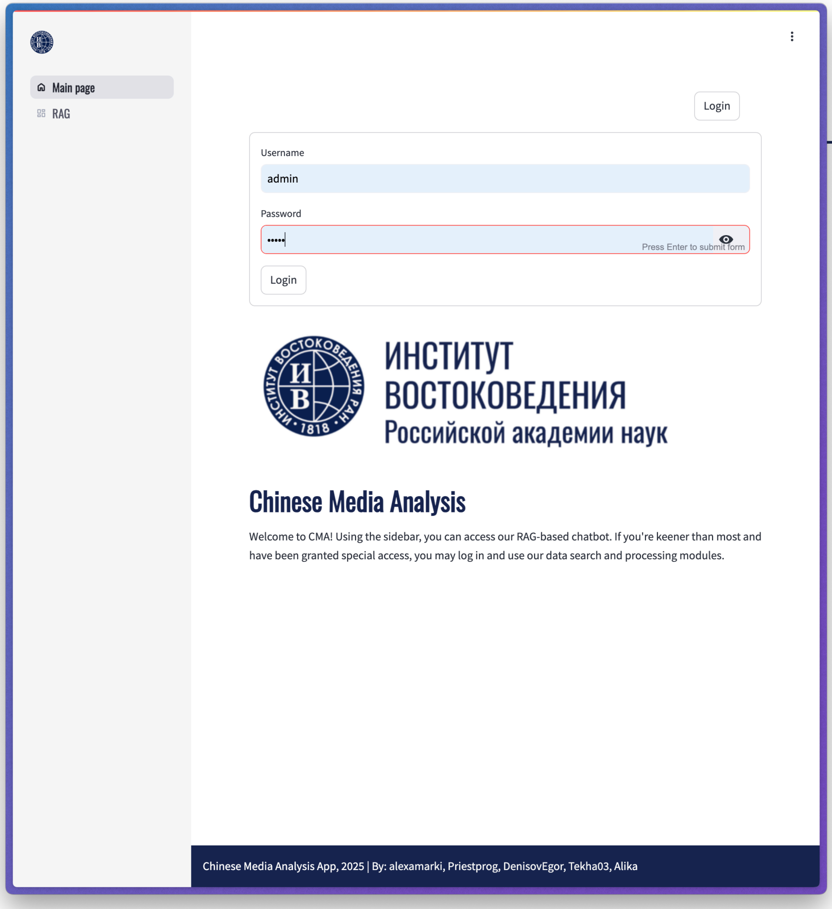
Relevant community summaries are passed to the LLM along with the query

The resulting answer is sent back to the frontend

A **TableOperations** object is created

if *.check_credentials*() doesn't fail, *.add_query*() is called

Interactions: User Login



Username
admin

Password
•••••
Press Enter to submit form

ИНСТИТУТ ВОСТОКОВЕДЕНИЯ
Российской академии наук

Chinese Media Analysis

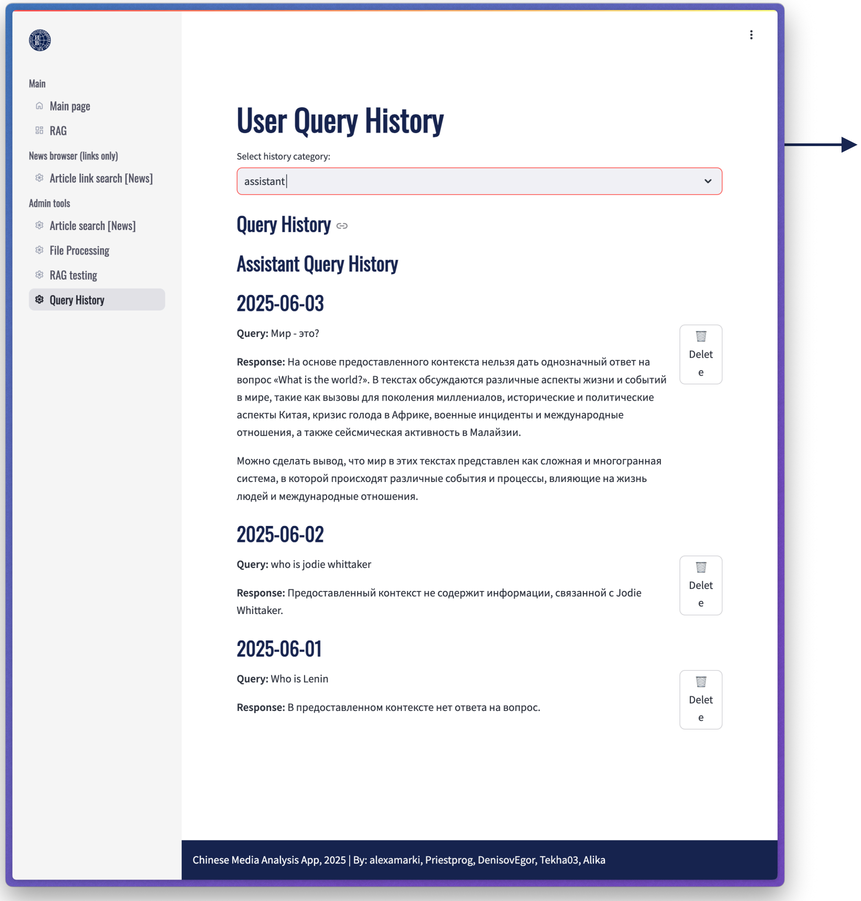
Welcome to CMA! Using the sidebar, you can access our RAG-based chatbot. If you're keener than most and have been granted special access, you may log in and use our data search and processing modules.

Chinese Media Analysis App, 2025 | By: alexamarki, Priestprog, DenisovEgor, Tekha03, Alika

A **TableOperations** object is created

if .*check_credentials*() doesn't fail, a success code is returned to the frontend

Interactions: User History



User Query History

Select history category:

assistant

Query History

Assistant Query History

2025-06-03

Query: Мир - это?

Response: На основе предоставленного контекста нельзя дать однозначный ответ на вопрос «What is the world?». В текстах обсуждаются различные аспекты жизни и событий в мире, такие как вызовы для поколения миллениалов, исторические и политические аспекты Китая, кризис голода в Африке, военные инциденты и международные отношения, а также сейсмическая активность в Малайзии.

Можно сделать вывод, что мир в этих текстах представлен как сложная и многогранная система, в которой происходят различные события и процессы, влияющие на жизнь людей и международные отношения.

2025-06-02

Query: who is jodie whittaker

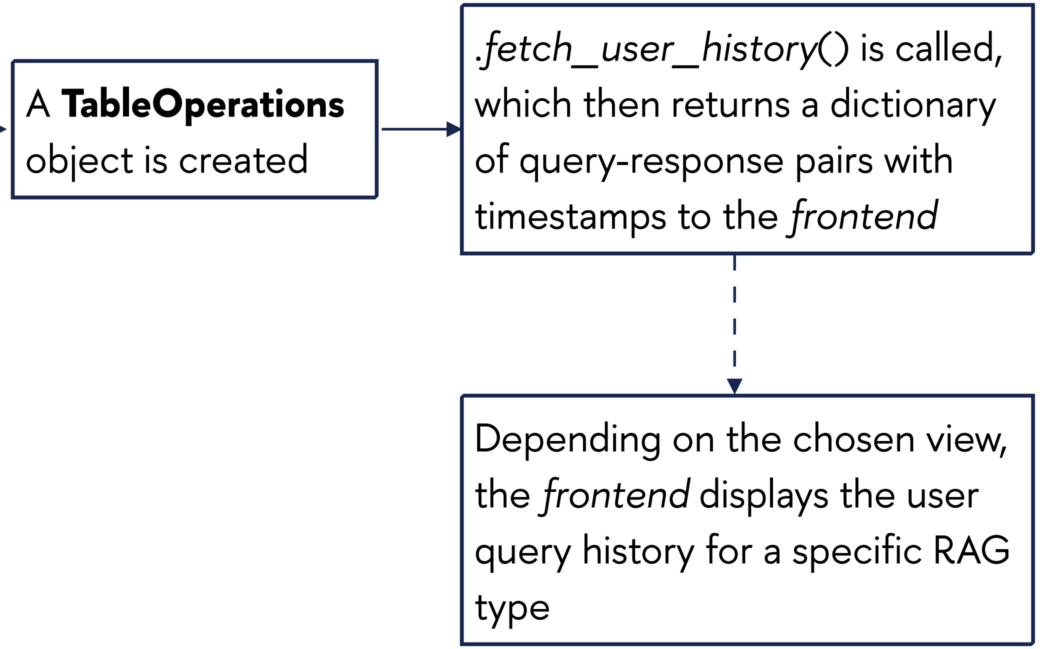
Response: Предоставленный контекст не содержит информации, связанной с Jodie Whittaker.

2025-06-01

Query: Who is Lenin

Response: В предоставленном контексте нет ответа на вопрос.

Chinese Media Analysis App, 2025 | By: alexamarki, Priestprog, DenisovEgor, Tekha03, Alika

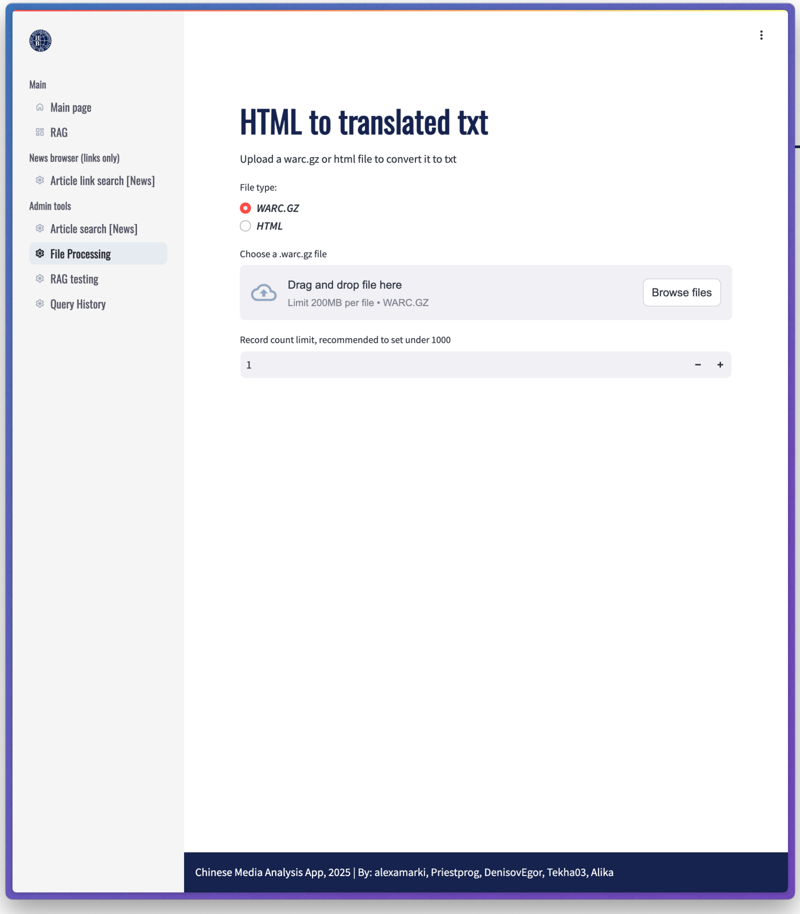


A **TableOperations** object is created

.fetch_user_history() is called, which then returns a dictionary of query-response pairs with timestamps to the *frontend*

Depending on the chosen view, the *frontend* displays the user query history for a specific RAG type

Interactions: Text Processing + Upload to DB



HTML to translated txt

Upload a warc.gz or html file to convert it to txt

File type:

● *WARC.GZ*

○ *HTML*

Choose a .warc.gz file

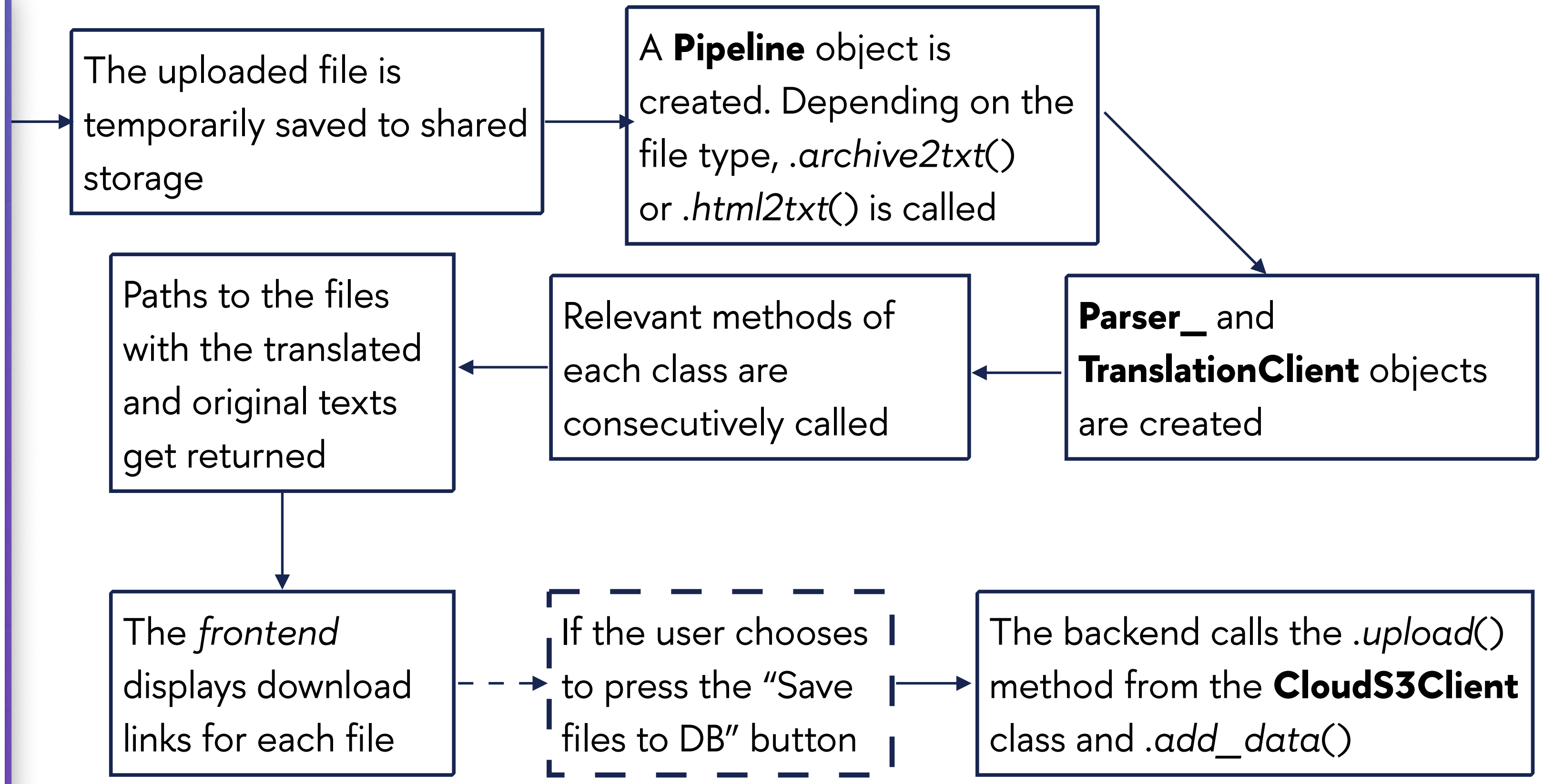
Drag and drop file here

Limit 200MB per file • WARC.GZ

Record count limit, recommended to set under 1000

1

Chinese Media Analysis App, 2025 | By: alexamarki, Priestprog, DenisovEgor, Tekha03, Alika



The uploaded file is temporarily saved to shared storage

A **Pipeline** object is created. Depending on the file type, .*archive2txt*() or .*html2txt*() is called

Parser_ and **TranslationClient** objects are created

Relevant methods of each class are consecutively called

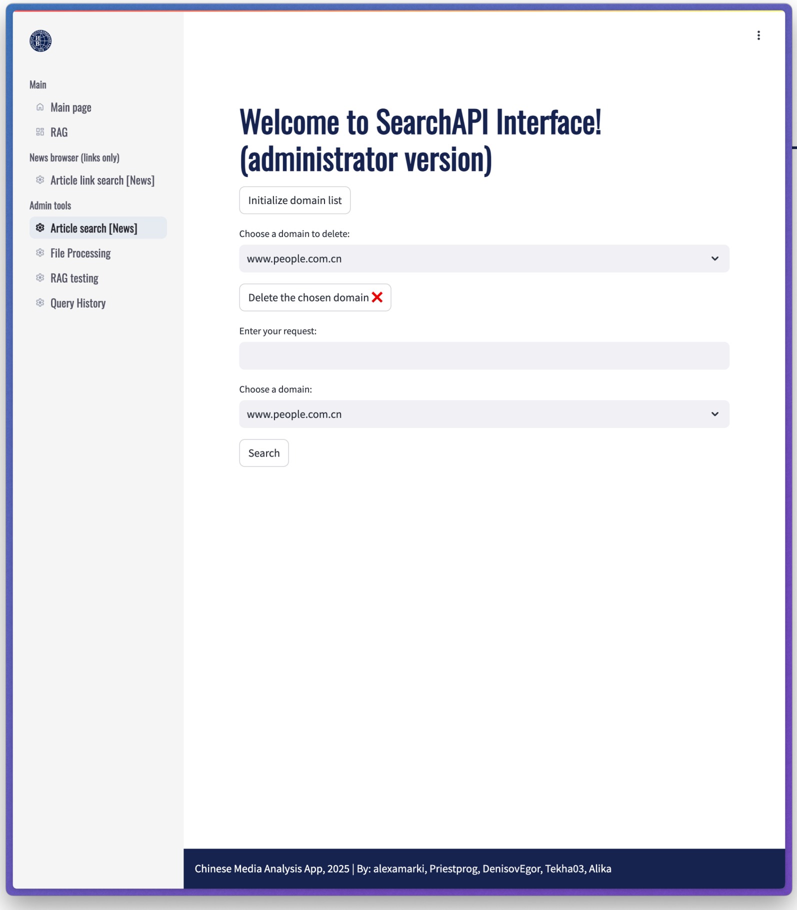
Paths to the files with the translated and original texts get returned

The *frontend* displays download links for each file

If the user chooses to press the “Save files to DB” button

The backend calls the .*upload*() method from the **CloudS3Client** class and .*add_data*()

Interactions: SearchAPI + Upload to DB



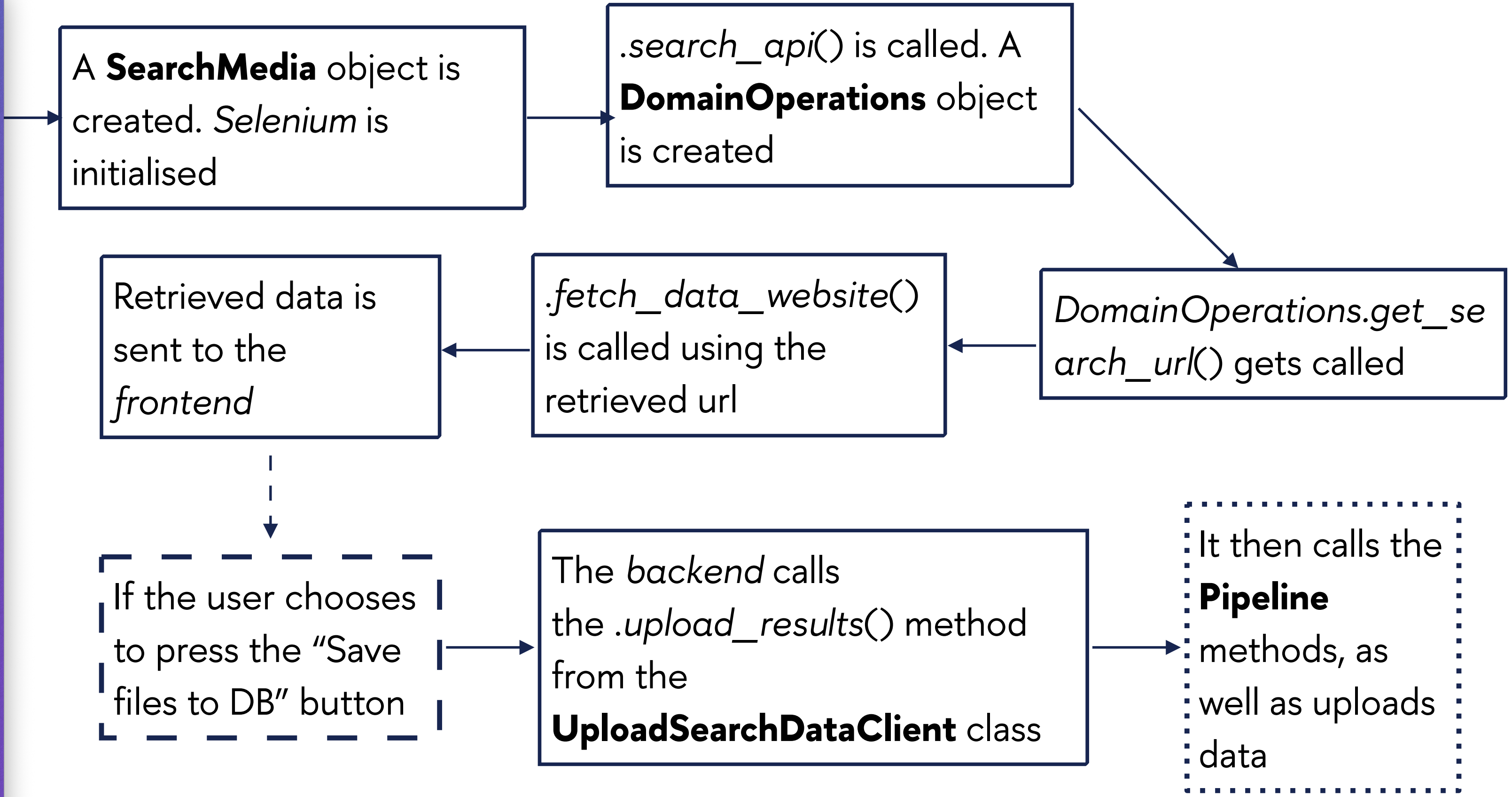
Welcome to SearchAPI Interface! (administrator version)

Choose a domain to delete: www.people.com.cn

Enter your request:

Choose a domain: www.people.com.cn

Chinese Media Analysis App, 2025 | By: alexamarki, Priestprog, DenisovEgor, Tekha03, Alika



A **SearchMedia** object is created. *Selenium* is initialised

.search_api() is called. A **DomainOperations** object is created

DomainOperations.get_search_url() gets called

.fetch_data_website() is called using the retrieved url

Retrieved data is sent to the *frontend*

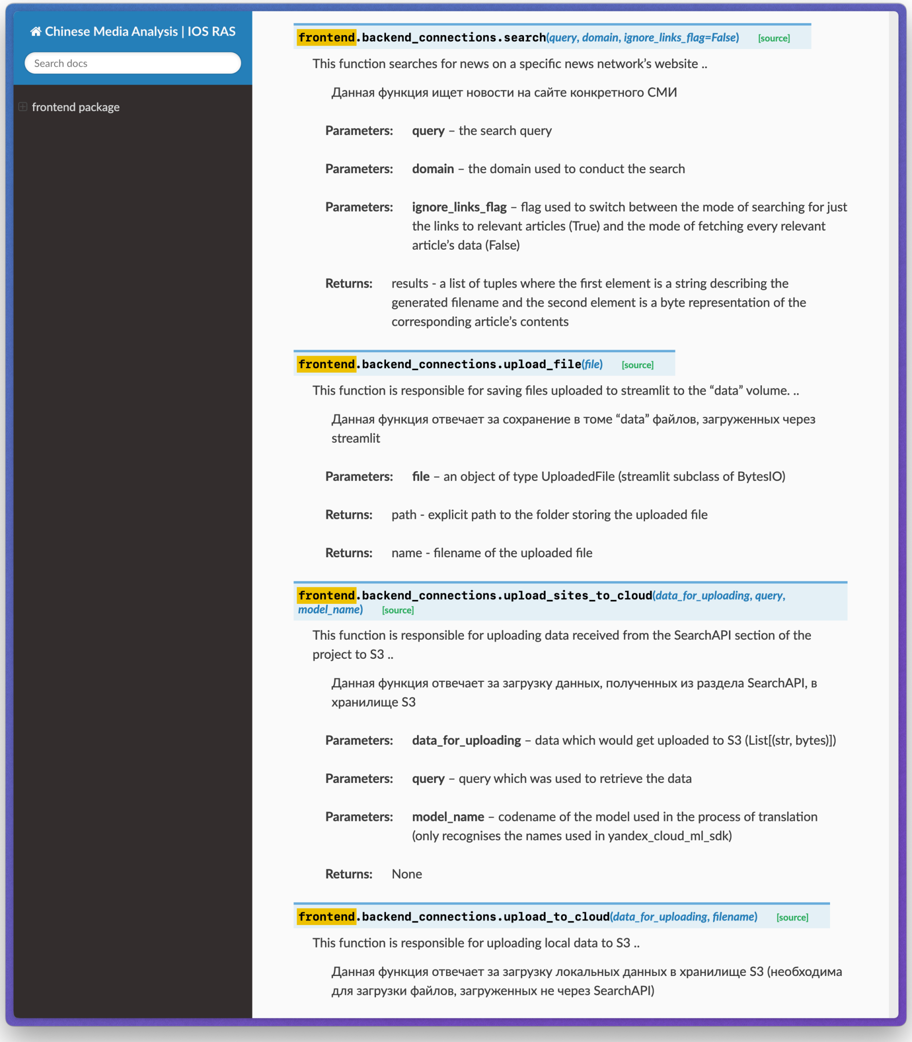
If the user chooses to press the "Save files to DB" button

The *backend* calls the *.upload_results*() method from the **UploadSearchDataClient** class

It then calls the **Pipeline** methods, as well as uploads data

Documentation

There is also (as of now) a private set of HTML files with **documentation** encompassing the whole project. It was built using **Sphinx**.



frontend.backend_connections.search(*query, domain, ignore_links_flag=False*) [source]

This function searches for news on a specific news network's website ..

Данная функция ищет новости на сайте конкретного СМИ

Parameters: **query** – the search query

Parameters: **domain** – the domain used to conduct the search

Parameters: **ignore_links_flag** – flag used to switch between the mode of searching for just the links to relevant articles (True) and the mode of fetching every relevant article's data (False)

Returns: results - a list of tuples where the first element is a string describing the generated filename and the second element is a byte representation of the corresponding article's contents

frontend.backend_connections.upload_file(*file*) [source]

This function is responsible for saving files uploaded to streamlit to the "data" volume. ..

Данная функция отвечает за сохранение в томе "data" файлов, загруженных через streamlit

Parameters: **file** – an object of type UploadedFile (streamlit subclass of BytesIO)

Returns: path - explicit path to the folder storing the uploaded file

Returns: name - filename of the uploaded file

frontend.backend_connections.upload_sites_to_cloud(*data_for_uploading, query, model_name*) [source]

This function is responsible for uploading data received from the SearchAPI section of the project to S3 ..

Данная функция отвечает за загрузку данных, полученных из раздела SearchAPI, в хранилище S3

Parameters: **data_for_uploading** – data which would get uploaded to S3 (List[(str, bytes)])

Parameters: **query** – query which was used to retrieve the data

Parameters: **model_name** – codename of the model used in the process of translation (only recognises the names used in yandex_cloud_ml_sdk)

Returns: None

frontend.backend_connections.upload_to_cloud(*data_for_uploading, filename*) [source]

This function is responsible for uploading local data to S3 ..

Данная функция отвечает за загрузку локальных данных в хранилище S3 (необходима для загрузки файлов, загруженных не через SearchAPI)

Conclusion

Our team has implemented:

- A **parsing pipeline** for files with news articles
- A module for directly **sending search queries** to Chinese news websites
- **Three** different **RAG** implementations for researchers
- A relatively **easily scalable project**

We expect the service to **launch** on the IOS RAS website in **late 2025**. For now, it's available on a separate domain.

After the last 6 months, everything now functions correctly and works together in unison, providing for an outstanding user experience

Future Prospects

- Consider the possibility of using another model for translation capabilities
- Assess if switching to a model which could directly work with Chinese text while keeping the same answer quality would make a meaningful difference
- Assess the quality of the 3 RAG implementations in the project in relation to each other
- Deploy the service on the IOS RAS website

Sources

[1] About this guide | Google developer documentation style guide. url: https://developers.google.com/style.

[2] BAAI. url: https://www.baai.ac.cn/.

[3] Beautiful Soup Documentation | Beautiful Soup 4.13.0 documentation — crummy.com. url:

https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (visited on Feb. 4, 2025).

[4] Chroma. url: https://www.trychroma.com/.

[5] Claude. url: https://claude.ai.

[6] codecs — Codec registry and base classes — docs.python.org. url: https://docs.python.org/3/library/codecs.html (visited on Feb. 4, 2025).

[7] Docker: Acceleterated Container Application Development. url: https : / /www.docker.com/.

Sources

[8] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. "From local to global: A graph rag approach to query-focused summarization". In: arXiv preprint arXiv:2404.16130 (2024).

[9] EntityRecognizer· spaCy API Documentation.url:https://spacy.io/api/entityrecognizer.

[10] FastAPI. url: https://fastapi.tiangolo.com/.

[11] Full featured documentation deployment platform. url: https://about.readthedocs.com/?ref=app.readthedocs.org.

[12] GPT-4. url: https://openai.com/index/gpt-4/.

[13] Industry Leading, Open-Source AI | Llama by Meta. url: https://www.llama.com/.

Sources

[14] Institute of Oriental Studies at the Russian Academy of Sciences — ivran.ru. url: https://ivran.ru/en (visited on Feb. 4, 2025).

[15] Introduction - Cypher Manual. url: https://neo4j.com/docs/cypher-manual/current/introduction/.

[16] LangChain. url: https://www.langchain.com/.

[17] Angela M. Lee and Hsiang Iris Chyi. "The Rise of Online News Aggregators: Consumption and Competition". In: International Journal on Media Management 17.1 (2015), pp. 3–24.

44[18] Leiden - Neo4j Graph Data Science. url: https : / / neo4j . com / docs / graph - data - science/current/algorithms/leiden/.

Sources

[19] Seth C. Lewis, Rodrigo Zamith, and Alfred Hermida. “Content Analysis in an Era of Big Data: A Hybrid Approach to Computational and Manual Methods”. In: Journal of Broadcasting Electronic Media 57.1 (2013), pp. 34–52.

[20] Mistral AI: Frontier AI LLMs, assistants, agents, services. url: https://mistral.ai/.

[21] Models and Pricing | DeepSeek API Docs — api-docs.deepseek.com. url: https://api-docs.deepseek.com/quick_start/pricing (visited on Feb. 4, 2025).

[22] Neo4j Graph Database Analytics | Graph Database Management. url: https://neo4j.com/.

[23] OpenAI - Pricing — yandex.cloud. url: https://openai.com/api/pricing/ (visited on Feb. 4, 2025).

Sources

[24] PostgreSQL: The world’s most advanced open source database.url:https://www.postgresql.org/.

[25] Tadej Praprotnik. “Digitalization and new media landscape”. In: Peer-reviewed academic journal Innovative Issues and Approaches in Social Sciences (2016), pp. 541–1855.

[26] Ragas — docs.ragas.io. url: https://docs.ragas.io/en/stable/#frequently-asked-questions (visited on Feb. 4, 2025).

[27] reStructuredText — Sphinx documentation. url: https://www.sphinx-doc.org/en/master/usage/restructuredtext/index.html.

[28] Selenium. url: https://www.selenium.dev/.

[29] Sphinx — Sphinx documentation. url: https://www.sphinx-doc.org/en/master/.

Sources

[30] SQLAlchemy - The Database Toolkit for Python. url: https://www.sqlalchemy.org/.

[31] st.Page - Streamlit Docs. url: https://docs.streamlit.io/develop/api-reference/navigation/st.page.

[32] Streamlit Docs — docs.streamlit.io. url: https://docs.streamlit.io/ (visited on Feb. 4, 2025).

[33] Text vectorization models | Yandex Cloud - Documentation. url: https://yandex.cloud/en/docs/foundation-models/concepts/embeddings.

[34] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. “From Louvain to Leiden: guaranteeing well-connected communities”. In: Scientific reports 9.1 (2019), pp. 1–12.

45[35] Welcome to Pydantic - Pydantic. url: https://docs.pydantic.dev/latest/.

Sources

[36] Yandex Cloud Documentation | Cloud Glossary | S3 — yandex.cloud. url: https : / / yandex.cloud/en-ru/docs/glossary/s3 (visited on Feb. 4, 2025).

[37] Yandex Cloud Documentation | Yandex Foundation Models | Yandex Foundation Models pricing policy — yandex.cloud. url: https://yandex.cloud/en-ru/docs/foundation-models/pricing (visited on Feb. 4, 2025).

[38] Yandex Foundation Models — yandex.cloud. url: https : / / yandex . cloud / en - ru / services/foundation-models (visited on Feb. 4, 2025).

[39] Yandex Search API — yandex.cloud. url: https : / / yandex . cloud / en - ru / services / search-api (visited on Feb. 4, 2025).

[40] YandexGPT 5 - the new generation of Yandex LLM. url: https://ya.ru/ai/gpt.

HSE UNIVERSITY • EST 1992 •