

NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science
Bachelor's Programme "Data Science and Business Analytics"

Software Team Project Report on the Topic:
Chinese Media Analysis

Submitted by the Students:

group #БПАД231, 2nd year of study	Markin Alexander Andreevich
group #БКНАД232, 2nd year of study	Sidorova Alik Igorevna
group #БПМИ2312, 2nd year of study	Dzhanobilov Tehron Sherozovich

Approved by the Project Supervisor:

Kolesnichenko Yelena Yurievna
Associate professor
HSE Faculty of Computer Science, Big Data and Information Retrieval School

Co-supervisor:

Rybalko Dmitriy Mikhailovich
Product Architect for ML services
Yandex LLC, Yandex.Cloud

Contents

Annotation	5
1 Introduction	7
1.1 Overview	7
1.2 Objectives	7
2 Analysis of Existing Solutions	10
2.1 Solutions	10
2.1.1 Google News API	10
2.1.2 Aylien News API	10
2.1.3 NewsAPI.org	10
2.1.4 IBM Watson Discovery	11
2.1.5 Dataminr	11
2.2 Conclusion	11
3 Work Organization	13
3.1 Organization	<i>A. Sidorova</i> 13
4 Data Retrieval	13
4.1 Approach	<i>T. Dzhanobilov</i> 13
4.2 Implementation	<i>T. Dzhanobilov</i> 13
4.2.1 Storage	13
4.2.2 SearchAPI (Our version)	<i>A. Markin and T. Dzhanobilov</i> 14
4.2.3 Data Export	<i>T. Dzhanobilov and A. Markin</i> 14
4.3 Interface	<i>T. Dzhanobilov and A. Sidorova</i> 15
4.4 Optimization	<i>A. Sidorova</i> 16
4.5 Testing	<i>A. Sidorova</i> 16
4.6 Conclusion	<i>T. Dzhanobilov</i> 16
5 Comparative LLM Analysis	17
5.1 Research Methodology	<i>A. Sidorova</i> 17
5.2 Comparative Analysis	<i>A. Sidorova</i> 18
5.2.1 Entity Recognition	18
5.2.2 Sentiment Analysis	18

5.2.3	Summary Generation	18
5.3	Future Prospects <i>A. Sidorova</i>	19
6	Data processing	20
6.1	HTML Parsing <i>A. Markin</i>	20
6.1.1	Approach	20
6.1.2	Implementation	21
6.1.3	Input types	22
6.1.4	Debugging	23
6.2	Translation <i>A. Markin</i>	23
6.2.1	Approach	23
6.2.2	Prompt engineering	24
6.2.3	Fine-tuning	25
6.2.4	Structure	26
6.2.5	Debugging	26
6.3	Combined processing pipeline <i>A. Markin</i>	27
6.3.1	Implementation	27
6.3.2	Debugging	27
7	Retrieval-augmented generation	28
7.1	Embedding-based RAG	28
7.2	AI Assistant-based RAG	29
7.3	GraphRAG <i>A. Markin</i>	29
7.3.1	Approach	29
7.3.2	Neo4j database setup, operations	29
7.3.3	Named entity recognition	29
7.3.4	Communities via Leiden, community summaries	30
7.3.5	Adding data	31
7.3.6	User query handler	32
7.4	Streamlit	32
8	User accounts	33
8.1	Approach <i>A. Markin</i>	33
8.2	Table setup	33
8.3	User operations	34

9	Utilities & Secrets	34
10	Frontend	34
10.1	Approach	<i>A. Markin</i> 34
10.2	Implementation	34
10.3	Design	35
11	Docker & FAST API	35
11.1	Approach	<i>A. Markin</i> 35
11.2	Container separation	36
11.3	Controllers	36
12	Documentation	37
12.1	Docstring	<i>A. Markin</i> 37
12.2	Sphinx	37
13	Logging	38
14	Current user experience	38
14.1	Homepage	<i>A. Markin</i> 38
14.2	RAG	38
14.3	Query history	39
14.4	File processing	39
14.5	Article search	39
15	Future prospects	43
16	Conclusion	43
	References	44
A	Glossary	47
B	Translation Prompt	48
C	Entity Recognition Prompt	49

Annotation

The main goal of this project was to develop an online service that would allow users interested in events directly or indirectly related to China to receive in-depth information based on sources whose target audience consists of native speakers.

We organised a call chain which fetches relevant Chinese news articles from news websites using Selenium, parses and translates their contents into English using a LoRA fine-tuned Llama¹ model, and saves the information from said articles into databases used by our implementations of the RAG method.

The data is converted them into LLM embeddings for use in our basic implementation of the RAG method, while GraphRAG requires the data to go through the processes of Named Entity Recognition and community detection via Leiden's method.

In addition, we have developed several internal tools to facilitate the work on the project, one of them being a RAG quality assessment tool.

As of now, the service is publicly deployed and it will be available via the website of the Institute of Oriental Studies of the Russian Academy of Sciences later this year.

This report consists of 51 pages.

Аннотация

Нашей целью было создание онлайн-сервиса, позволяющего пользователям, заинтересованным в событиях, напрямую или косвенно связанных с Китаем, получить более подробную информацию, основанную на источниках, целевой аудиторией которых являются носители языка.

В рамках проекта мы организовали цепочку вызовов, находящую релевантные новости на китайском языке на сайтах СМИ с использованием Selenium, обрабатывющую и переводящую тексты статей на английский с использованием дообученной модели Llama¹ и, наконец, корректно размещающую информацию из статей в базы данных, используемые нашим проектом для реализации метода RAG.

Данные конвертируются в LLM-эмбеддинги для дальнейшего использования в нашей базовой имплементации метода RAG, а для реализации аналога GraphRAG проходят через процесс анализа сущностей и определения сообществ сущностей путём метода Лейдена.

Помимо этого мы разработали несколько внутренних инструментов для облегчения

¹Meta, the developer of Llama, is considered an extremist organisation in Russia

работы над проектом, включая инструмент для оценки качества RAG.

На данный момент сервис уже доступен публично, и в будущем он станет доступен на официальном сайте Института Востоковедения РАН.

Данный отчет состоит из 51 страницы.

Keywords

LLM, Yandex Cloud, machine translation, RAG, IOS RAS, China, news articles, analysis, online assistant, NER, Leiden, GraphRAG

1 Introduction

1.1 Overview

Over the last quarter of a century, the media landscape has undergone massive changes (as seen in [25]), making it virtually impossible to continuously manually² analyse, compare and contrast news articles from multiple sources. Consequently, the market has now been introduced to a variety of news aggregators [17]. Additionally, the rise of search engines enabled users to learn about the sources of funding and political biases of each media outlet disseminating a particular news story.

However, it is worth noting that as of now, there are almost no services that use all of this information together in order to provide a brief digest of a particular event. We believe this to be a significant omission, as this kind of service would help both the researchers who wish to get the general idea of a particular event before diving deeper into the topic, as well as the ordinary users interested in the latest developments in the world. Of course, there are some LLM-based products able to crawl websites and gather relevant information on what's happening around the world, such as ChatGPT, yet that is not their main goal.

Within the framework of this project, we have been developing a more narrowly focused variation of the service mentioned above, specialising in news related in one way or another to China and its immediate neighbours. Our ultimate goal was to have created a publicly available online assistant which will respond to user queries based on data regularly collected from an extensive catalogue of relevant news, providing crucial context on a variety of world events.

1.2 Objectives

The following objectives were set at the beginning of the project:

1. Repeatedly fetch Chinese news using Yandex SearchAPI (Website: [39]) and other possible methods
2. Download relevant articles in HTML form and upload them to the cloud
3. Parse HTML files to get the article in text form
4. Translate the articles from Chinese to English³

²For more information on manual/automated analysis, refer to [19]

³English was chosen for ease of translation – machine translation usually performs better between English and other languages than between Russian and other languages.

5. Develop multiple RAG⁴ workflows using article data
6. Assess RAG quality using **ragas** (Website: [26])
7. Combine items 1-6 into a complete service
8. Create a simple frontend for the service
9. Separate the project files into clearly defined **Docker** (Website: [7]) containers for easier deploy
10. Launch the service on the website (Website: [14]) of the IOS RAS (hereinafter referred to as "*the Institute*")

At the time of writing this final report, our team has successfully almost all of the above items, except for item 10, as it requires additional approval from inside the Institute. However, the project *is* currently available on a public IP address.

In other words, we have obtained the following results over the course of the last 6 months:

Main:

- Successful collection of HTML files with Chinese news articles via our local module called SearchAPI
- Proper parsing and translation of articles into English
- Development of three RAG implementation based on article data:
 - * A custom RAG made by *Alexander Bagrov*
 - * A Yandex Assistant-based RAG by *Egor Denisov*
 - * A GraphRAG implementation by *Alexander Markin*
- Rearrangment of several parts of the codebase into logically sound modules
- Creation and testing of relevant **Docker** containers
- Development of a REST API service for container interaction

Auxiliary:

- Addition of a user authentication and chat history system
- Development of a data collection service for fine-tuning
- Creation of **streamlit** interfaces for debugging items 1-4 from the objective list

⁴Since this project involves the development of several RAG implementations, each team member will have worked on item 5 by the time of the defence

- Implementation of a basic framework for RAG quality assessment
- Creation of a diagram outlining interactions between the project’s various components

Our team consists of **Alexander Markin** (*HSE DSBA*), **Tehron Dzhanobilov** (*HSE AMIS*), **Alika Sidorova** (*HSE CDS*), as well as two *MSU* students – **Alexander Bagrov** and **Egor Denisov**, whose contributions will be mentioned briefly.

T. Dzhanobilov and **A. Sidorova** have been responsible for items 1-2 and 7-8, **A. Markin** – for items 3-5 and 7-9, **A. Bagrov** and **E. Denisov** – for items 5-9.

2 Analysis of Existing Solutions

2.1 Solutions

2.1.1 Google News API

Allows retrieving aggregated news from multiple sources, filtering them by keywords, regions, and topics.

Drawbacks:

- Limited number of free requests, expensive commercial usage.
- Poor performance with Chinese sources – not all websites are indexed.
- Cannot deeply analyze content, only provides basic metadata (title, description).

Avoiding drawbacks:

Use direct website parsing instead of relying on aggregators.

2.1.2 Aylien News API

Analyzes news articles by determining sentiment, key entities, and topic classification.

Drawbacks:

- Weak support for the Chinese language.
- Closed API that does not provide full transparency of algorithm operations.
- Often makes mistakes in sentiment analysis due to language nuances.

Avoiding drawbacks:

If we end up requiring sentiment analysis, custom algorithms tailored for the Chinese lexicon should be sought out..

2.1.3 NewsAPI.org

Allows retrieving fresh news from different sources and offers a simple REST API.

Drawbacks:

- Does not provide full article texts, only headlines and short descriptions.
- Chinese sources are almost absent.

- No built-in content analysis, only search and filtering.

Avoiding drawbacks:

Use full parsing methods and NLP processing of content.

2.1.4 IBM Watson Discovery

An advanced tool for news analysis with AI-powered features.

Drawbacks:

- High licensing costs, requires significant resources.
- Complex setup, not always clear how to adapt it to a specific task.
- May incorrectly interpret Chinese texts due to model-specific limitations.

Avoiding drawbacks:

Use open models with the ability to fine-tune on specific data.

2.1.5 Dataminr

An AI system that analyzes news and social media in real time.

Drawbacks:

- Primarily focused on English-language sources.
- Requires a subscription and does not provide full API access.
- High probability of false positives when analyzing events.

Avoiding drawbacks:

Avoid sources prone to falsified news, such as social media, and use models optimized for the Chinese language.

2.2 Conclusion

Almost all existing solutions are either expensive or perform poorly with Chinese news. The main ways to outperform them are the following:

- Using comprehensive text analysis rather than just fetching the headlines.
- Introducing support for the Chinese language, considering its specific features.

- Ensure flexibility in model configuration and data filtering.
- Avoid setting limitations on the number of sources and news articles.

3 Work Organization

3.1 Organization

For the most part, Yandex.Tracker (website: [40]) was used for setting up workflows within the team for the purpose of effective task management. This led to better organization and a relatively fair workload distribution, letting team members track the current status of tasks, and ensure clearer coordination of project participants' actions. The most convenient ways to conduct teamwork were analysed, suitable tools for process management were selected, and recommendations for their use were developed.

Another notable fact is that throughout the duration of the project, weekly synchronisation calls were held with the team and representatives from Yandex and the Institute, motivating each team member to do their best the whole time..

4 Data Retrieval

4.1 Approach

At this stage, the main task at hand was the technical implementation of the data collection process, setting up APIs, and interacting with cloud storage, which was a crucial part of the entire system and required a detailed approach. We started by studying potential sources of Chinese news and selecting the optimal ways to collect them automatically.

For this, various tools were explored, such as Yandex SearchAPI, and approaches for storing collected data in the cloud were developed. Initially, we also analyzed existing methods for automatic news article retrieval, considered different structuring approaches, and selected relevant materials. This process required a deep study of available technologies and their comparative analysis to choose the best solution.

4.2 Implementation

4.2.1 Storage

One of the first stages was the development of a mechanism for uploading news to Object Storage S3 (website: [36]), ensuring convenient data access for further processing.

As part of this task, available cloud solutions were analyzed, their functionality was tested, and the most suitable storage system was selected while considering the specifics of working

with Chinese texts. Data security issues were thoroughly examined and potential limitations on processing large amounts of information were analyzed, which helped avoid performance problems and ensure system stability.

4.2.2 SearchAPI (Our version)

Once the data collection process was successfully deployed, the focus shifted to implementing news search capabilities and extracting XML and HTML files of relevant articles. This stage included testing a variety of query strategies, analyzing their effectiveness, and selecting the most suitable solution.

In order to obtain data from foreign news websites, a system, which collected a list of search pages for Chinese news websites and subsequently received data from the top-20 search results, was developed. The core dependency of this system in particular ended up being `selenium` (Website: [28]). In the future, `selenium` could be substituted with a lightweight browser emulator, yet as of now, it seems to be the best option available.

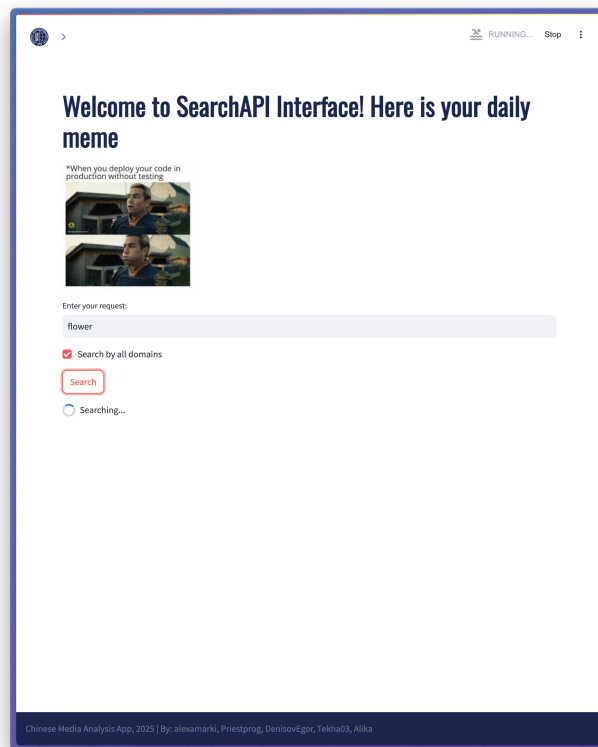
Gradually, a working prototype which allowed one to retrieve data from Chinese websites and store it in the cloud for further analysis was developed. We experimented with various API parameters, analyzed the results, and optimized the data retrieval process to minimize information loss. This stage was crucial as it significantly improved the quality of collected data.

4.2.3 Data Export

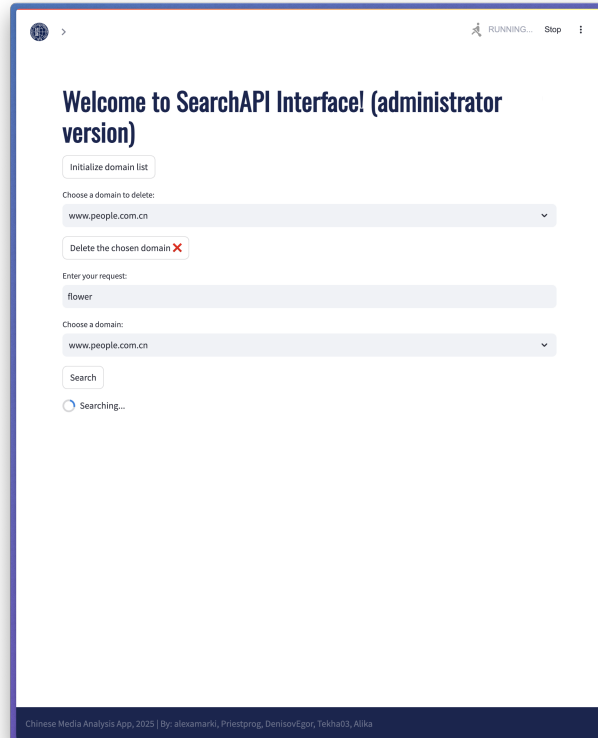
Next, automatic data export capabilities were implemented and processes were set up for delivering the data to users. The system could now process search queries and provide relevant results in a convenient format.

We also worked on improving the accuracy of the mechanism by eliminating potential errors and optimizing the system’s interaction with cloud storage. Logging mechanisms were introduced which allowed one to track and analyze possible system failures and promptly resolve errors. This ensured higher system stability and simplified the task execution monitoring process.

4.3 Interface



(a) User Interface



(b) Administrator Interface

Figure 4.1: Interface of `streamlit` applications for data collection via SearchAPI

One of the key stages of development was creating a user-friendly and intuitive `streamlit` interface, which allowed users to interact with the system without delving into technical details (Figure 4.1).

We designed the interface structure, analyzed user scenarios, and adapted the best UI design practices to the project’s needs. We studied the `streamlit` library, its documentation, and usage examples (website: [32]), which helped us develop the interface architecture.

Additionally, various visualization options were tested in order to make the interface as convenient and understandable as possible. This stage was critically important as it enabled us to create a comprehensive tool that significantly simplified access to the system’s functionality, making news data processing more user-friendly.

4.4 Optimization

We also worked on optimizing news search by testing different text preprocessing methods. Automatic keyword extraction algorithms were explored and data clustering capabilities were analyzed.

Ensuring the correct operation of algorithms for the Chinese language, which required adapting existing solutions to the specifics of logographic writing, ended up being a significant challenge.

4.5 Testing

At the next stage, we actively participated in testing the developed system prototype, verifying the correctness of data processing and analyzing potential errors.

During the tests, some peculiarities in processing Chinese characters that required additional refinement were identified. We also tested various search system configurations to ensure proper operation.

4.6 Conclusion

In the final stages of working on this part of the project, we developed a system for searching news by specific domains and added an administrative interface that allows changing the list of sources. This significantly simplified the data management process and made the system more convenient for end users.

In addition, we conducted final testing, eliminated identified deficiencies, and prepared the system for further development. We also formulated recommendations for further improvement,

including the integration of new models to improve the accuracy of news analysis. Further development of this part of the project involves the implementation of additional functions aimed at improving data analysis and increasing the efficiency of the system, which will make it even more useful and convenient for users.

5 Comparative LLM Analysis

We also conducted a comparative analysis of analogous models by testing several popular LLMs on key project tasks: entity recognition, sentiment analysis of news, and summary generation.

5.1 Research Methodology

The following models were selected for analysis based on their potential integration into the system:

- **YandexGPT** – one of the most accessible and easily integrable models with Russian language support (Website: [\[41\]](#))
- **Mistral-7B** – a powerful open-source model demonstrating high text processing quality (Website: [\[20\]](#))
- **Llama-2-13B** – a model from Meta⁵ with advanced natural language processing capabilities (Website: [\[13\]](#))
- **ChatGPT (GPT-4)** – one of the most accurate and flexible models for text analysis (Website: [\[12\]](#))
- **Claude 2** – a promising model from Anthropic optimized for processing long contexts (Website: [\[5\]](#))

Each of these models was assessed on several criteria: accuracy in extracting entities, generating short summaries, determining the emotional coloring of news, and the ability to work with the Chinese language.

⁵Meta, the developer of Llama, is considered an extremist organisation in Russia

5.2 Comparative Analysis

5.2.1 Entity Recognition

The task of extracting key entities yielded varying results across the models:

- **Best Performance:** GPT-4 and Llama-2-13B. These models successfully identified company names, politicians' names, and even specific geographical objects.
- **Moderate Performance:** Mistral-7B – performed reasonably well but occasionally confused names with company titles.
- **Worst Performance:** YandexGPT – identified key objects but frequently made transliteration errors.

Conclusion: The identification of entities is best handled by powerful models with extensive pre-trained datasets.

5.2.2 Sentiment Analysis

Testing the models on text sentiment detection produced notable results:

- **Best Performance:** GPT-4 and Claude 2 – accurately determined the sentiment of news, distinguishing neutral statements from positive and negative ones.
- **Moderate Performance:** Llama-2-13B – effectively recognized emotional context but sometimes provided overly generalized assessments.
- **Worst Performance:** YandexGPT – exhibited a high error rate, especially in complex news texts.

Conclusion: Sentiment analysis requires sophisticated contextual models, though even these occasionally make errors.

5.2.3 Summary Generation

- **Highest Quality:** GPT-4 – generated concise and informative summaries without losing critical information.
- **Acceptable Performance:** Claude 2 and Llama-2-13B – generally conveyed meaning well but sometimes omitted key points.

- **Worst Performance:** YandexGPT and Mistral-7B – either excessively condensed the text or lost essential details.

Conclusion: Summary generation is best performed using powerful commercial LLMs.

5.3 Future Prospects

Table 5.1: Final model competency matrix

Task	GPT-4	Claude 2	Llama-2-13B	Mistral-7B	YandexGPT
Entity Recognition	Excellent	Good	Excellent	Moderate	Good
Sentiment Analysis	Excellent	Excellent	Good	Moderate	Good
Summary Generation	Excellent	Good	Good	Moderate	Excellent

Analysis of this table (Table 5.1) has led to several conclusions:

1. YandexGPT

Despite being optimized for the Russian context, YandexGPT may perform less effectively when analyzing Chinese news compared to more specialized models designed for multitasking and multilingual support. During testing, the model showed decent results in some areas, such as entity extraction and text analysis, but additional fine-tuning was needed for more accurate work with Chinese news.

- **Recommendation:** Retain YandexGPT for tasks involving news where it’s important to consider the specific characteristics of the Russian and Chinese contexts. The model is quite suitable for tasks that require localisation and high accuracy in working with texts, including Russian ones.

2. Mistral-7B

The Mistral-7B model demonstrated good results in tasks related to entity extraction and text analysis, including work with the Chinese language. It is suitable for multilingual projects, as it offers decent support for various languages, including Chinese.

- **Recommendation:** Consider Mistral-7B, as the model handles multitasking effectively, including the analysis of Chinese news, and has sufficient power for working with data.

3. Llama-2-13B

Llama-2-13B showed good results in tasks involving summarization and identifying emotional tone. It works flexibly with different languages and is suitable for analyzing texts that require generalization and information synthesis.

- **Recommendation:** Retain Llama-2-13B, as it is suitable for creating concise summaries and analyzing the tone of Chinese news, which is an important aspect of our project.

4. ChatGPT (GPT-4)

ChatGPT is a powerful model that delivers excellent results in summarization, entity extraction, and tone detection. It performs well with the Chinese language and is also suited for more complex multitask operations.

- **Recommendation:** Consider GPT-4, as it is highly versatile and suitable for complex tasks involving the analysis of Chinese news. It will be indispensable for multitasking operations, including deep news analysis.

5. Claude 2

Claude 2 is one of the most advanced models for creating brief and accurate summaries. It showed good results in tone analysis, although it slightly lagged behind other models such as Mistral-7B in entity extraction tasks.

- **Recommendation:** Consider Claude 2, as it is suitable for tasks related to emotion analysis and the creation of concise summaries of Chinese news. This is especially important for ensuring accuracy and brevity of information.

Thus:

- GPT-4 – the best option, though potentially costly.
- Claude 2 and Llama-2-13B – strong alternatives, particularly for processing long texts. Our team opted for Llama, albeit a different model, which was available via Yandex Foundation Models - Llama 3.1.
- Mistral-7B and YandexGPT – currently unsuitable for high-quality analytical tasks.

Further testing should be conducted on an extended dataset to assess model performance in real-time news processing.

6 Data processing

6.1 HTML Parsing

6.1.1 Approach

Once the program had successfully retrieved the HTML files, it had to tackle the problem of parsing them. Ideally, it should only output the text relevant to the article, as any other data

present in the file is of no use to this service.

To achieve this, two approaches were initially considered:

1. Using the `BeautifulSoup` 4 Python library (Website: [3])
2. Using a large language model (LLM)

However, the latter of the two quickly proved ineffective: most of the files in the system exceeded the token limit on closed-source models and took too long to process on open-source models, even when run locally. Moreover, since our service must call an LLM for its machine translation capabilities every time it needs to translate an article, the latter approach would have effectively doubled the number of API calls and more than doubled the costs of running the service.

Therefore, it was decided that `BeautifulSoup` 4 would be used for basic parsing. Of course, since each website has its own structure, this method would not completely isolate the contents of each article, but it would remove all HTML code and most of the non-article text.

6.1.2 Implementation

Implementing parsing ended up being relatively easy due to `BeautifulSoup` 4's `get_text()` ([3]) function, which returns all the human-readable text from the HTML file. However, multiple additional measures were taken in order to further parse the text:

- Prior to calling `.get_text()`, BS's `.decompose()` ([3]) is used in order to get rid of sections of the file where the class, id or name clearly indicates that they belong to something other than a news article (e.g., footer, advertisement or copyright)
- Repeated sequences of multiple whitespace characters are replaced with a single instance of each of the given characters (e.g., multiple consecutive spaces are converted into one)
- The title of the original web page gets added to the beginning of the document. Usually the title of the article itself and the source are located there, so this ensures they will be present in the output

Since there might be some problems related to the encoding used in the page title, two different ways of determining it were implemented in the code:

1. Reading the web page header using `BeautifulSoup` 4 (Website: [3])
2. Using the `codecs` library to guess the encoding based on the contents (Website: [6])

The latter of the two approaches is only invoked if the former fails, making sure there is always a determined encoding for the text.

In the end, the parsing module comprised various helper functions and two classes: `ParserMono` and `ParserMulti`, the former of which contained methods mostly pertaining to HTML parsing, while the latter is derived from the former and included methods for warc.gz parsing.

6.1.3 Input types

As mentioned above, this parser accepts two different types of files as input:

- an HTML file with a single article
- a warc.gz archive containing a large number of HTML files


In most cases, only the files belonging to the first type would be passed to the parser. However, support for the second type is still important, as it offers administrators the ability to provide a large amount of context for the RAG model while making fewer manual calls to the backend.

6.1.4 Debugging


WARC.GZ to TXT

Upload a warc.gz file to convert it to txt

Choose a .warc.gz file

 Drag and drop file here
Limit 200MB per file • WARC.GZ

Browse files

 www2.hkej.com-2016-12.warc.gz 2.8MB ×

Record count limit, recommended to set under 1000
5 - +

☐ Display all the individual .txt files (not recommended for archives over 5MB in size due to SEVERELY ADVERSE consequences)

Uploaded file: www2.hkej.com-2016-12.warc.gz

Processing the file...

File processed successfully!

Generated txt files:

- w2t_www2.hkej.com-2016-12.warc.gz.zip

Download w2t_www2.hkej.com-2016-12.warc.gz.zip

Figure 6.1: Interface for the parsing `streamlit` app

In addition, a basic `streamlit`-based interface was created for the parser, which can be seen in Figure 6.1. By default, parsed versions of the files are saved to the user's home directory in the folder `.chinese_media/streamlit`, but they can also be downloaded to the `Downloads` folder through the interface.

Since `streamlit`-based websites dynamically update after each interaction, the interface proved valuable for debugging purposes earlier in development, as developers were able to select a file and rerun the parsing process every time they tweaked the code.

6.2 Translation

6.2.1 Approach

In order to translate articles from Chinese to English, a model which would act as a translator had to be chosen. The most crucial factors in making this decision were:

- Model availability

- Adequate request pricing, comparable to average market costs (about \$2.20/1 million tokens)⁶
- Accuracy in translation from Chinese to English
- Possibility of fine-tuning

Given the above factors, the selection process proceeded as follows:

1. The first model to be considered was **YandexGPT Pro**, access to which was kindly granted by *Yandex*, the company at the request of which this project is being undertaken. While this model fit most of the criteria, it was not the most accurate among its competitors in terms of Chinese to English translation due to the dataset on which it was originally trained.
2. **WuDao 2.0**, developed by the *Beijing Academy of Artificial Intelligence* (Website: [2]), was the second candidate. In stark contrast to **YandexGPT**, this model was extremely good at translating between the aforementioned languages. However, it soon became apparent that only its previous version, **WuDao 1.0**, which is inferior to some of its current competitors, was available to the public. Moreover, the best-performing version of 2.0 would seemingly be too costly to run because of its sheer size.
3. Finally, the last two candidates to be considered were the **DeepSeek v3**⁷ and **Llama 3.1**⁸ models. Based on some rudimentary testing, both models fit all of the criteria from above, yet the latter of the two became available through the Yandex Foundation Models (website: [38]) system on the 9th of December, 2024, which made it the most appealing model for the project.

Based on this assessment and the analysis from Chapter 5, it was decided that **Llama 3.1**⁸ would be used for translation.

6.2.2 Prompt engineering

Apart from the translation itself, it was necessary for the model to do the following:

- Ensure that all text unrelated to the article itself is eliminated

⁶This judgement was made based on the pricing lists available on the OpenAI website [23], the DeepSeek website [21] and the Yandex Foundation Models website [37]

⁷A few months prior to the time of writing this report, *DeepSeek* has introduced a new, improved model, **R1**, comparable in response quality to **ChatGPT o1**. We may consider it as part of an alternative to **Llama 3.1**.

⁸Meta, the developer of **Llama**, is considered an extremist organisation in Russia

- Keep the names of people, places and institutions in the language of origin in brackets after their transliterations (Ignoring the fact that there are multiple ways to transliterate the same name would lead to problems later on).
- Separately specify the title of the article, its source and the date of publication, as they provide valuable context for the news story.

Given these requirements, it was obvious that simply asking the model to translate the text would not suffice - the prompt would most certainly have to be significantly more nuanced.

Through trial and error, a prompt which performed relatively well on news-related data was created (Appendix B), yet there were still inconsistencies in the structure of the model's output that made it clear that fine-tuning should be strongly considered.

6.2.3 Fine-tuning

The version of Llama 3.1⁹ available through *Yandex Foundation Models* ([38]) supported LoRA fine-tuning, which ended up being noticeably beneficial for the quality of the answers given by the model. In order to initiate the tuning, a `.jsonl` dataset had to be prepared, containing in each line:

System message - the context for the model, containing rules for its behaviour

User message - the request for translation, consisting of the original Chinese text

Assistant message - the English translation itself

Since manually collecting a sizeable dataset would be too time-consuming, it was decided that the process of collecting the data and tuning on it would have to be performed semi-automatically. For this purpose, a supplementary service was created:

First, the service collects a few articles from each of the archives passed to it as arguments, parses them and translates them into English using the base model with the prompt featured in Appendix B. Having completed this action, it saves two `.txt` files for each article - one containing the parsed text and the other containing the translated text - into the folder named `training_data_v*`, where `*` is the lowest integer number for which this folder doesn't initially exist.

Following this, a human goes through the outputs and tweaks the files so that all of the outputs fit the rules established earlier.

⁹Meta, the developer of Llama, is considered an extremist organisation in Russia

Finally, the last automated part of the service is launched. It finds the newest folder of the type `training_data_v*`, fetches the files from it and uses their contents to create the dataset `training_dataset_*.jsonl`. Then, it uploads the dataset to Yandex Cloud, runs LoRA fine-tuning on it, and adds the URI of the model to `uris.txt` along with its serial number, concluding the process.

6.2.4 Structure

Translation-related functions are located in the `TranslationClient` class, which is in turn derived from a base class for any connection to the Yandex Foundation Models - `YandexMLClient`. As for fine-tuning, its pipeline got separated into a separate folder:
`/backend/.../yandex_foundation_models/train`.


6.2.5 Debugging

Llama-lite translator

Text input type:

- ☐ *Insert text*
☒ *Upload .txt file*

Choose a .txt file with the text

 Drag and drop file here
Limit 200MB per file • TXT

Browse files

 txt_www.taiwannews.com.tw-2016-12.warc.gz_record_1.txt 1.1KB ×

Process Input

Translation:

Title: Dumping Eggs and Spreading Ashes: CATS Union (興航工會) Brings 500 Members to the Streets

Source: Taiwan English News

Date: December 22, 2016

The CATS Union (興航工會) today called on over 500 CATS employees to throw eggs and spread ashes in front of the Produced Industries Building (國產實業大樓) to protest. CATS Union advisor Lin Jia-wei said that the march will reach the Presidential Office (總統府), and if

Figure 6.2: Interface for the translation `streamlit` app

Similarly to [6.1.4](#), a `streamlit` interface for testing translation quality was created, which can be seen in [Figure 6.2](#)

7 Retrieval-augmented generation

Retrieval-augmented generation is an approach used applied to AI assistant systems which helps the system retrieve relevant information from existing sources based on a user's query, instead of generating it seemingly out of nowhere. While most models can easily answer general questions like "Where is the Great Wall located?" due to having been trained on a large dataset which definitely included a sizeable amount of mentions of the tourist destination, the likelihood of the model correctly answering the question "What were the exchange rates in Russia on the 9th of March, 2006?" is much lower. However, given access to a database with exchange rate data over the last 20 years, an LLM with RAG would be able to find the data and add it to its context.

Since the purpose of our project is to let users research a wide variety of information using Chinese data, most of the potential queries would only be answerable if such a system were to be a part of the project.

Currently, the project houses three different implementations of a RAG system along with a counterpart evaluation system. This is by design, as the Institute expressed a desire for multiple systems they could compare with each other in terms of efficiency and truthfulness.

The *MSU* part of the team was responsible for developing the embedding-based RAG, as well as the version which utilises Yandex AI Assistant, while **Alexander** from the *HSE* part of the team implemented a Graph-based RAG system inspired by the Microsoft Research GraphRAG study ([8]).

In this chapter, *MSU* contributions will be mentioned briefly, as they're still required for one to get a clear vision of the whole project. GraphRAG, however, will be described in detail.

7.1 Embedding-based RAG

The first RAG implementation to be developed by our team utilised YandexGPT Embeddings ([33]) via LangChain ([16]) when converting text into vector embeddings. This operation was used both on the data and any user input. Embedded data got stored in a ChromaDB ([4]) database, following which embedded user input was compared with every other vector in the database. The vectors which ended up being the closest to the ones from the user query were then added to a prompt passed along to YandexGPT, which then finally provided an answer to the query.

7.2 AI Assistant-based RAG

This implementation was quite easy to get up and running, as Yandex’s AI assistant is able to automatically create a search index from files passed along to it. While it generally performs adequately on the data our team is working with, it is noticeably less customisable than the other two solutions given in this paper.

7.3 GraphRAG

7.3.1 Approach

During testing of the two former solutions, it was noticed on multiple occasions that files relevant to a user query were sometimes ignored by the system for no apparent reason. Although it got partially solved by offering the end user more control over the model’s parameters, this observation resulted in further inquiries about various RAG solutions which could be utilised more efficiently.

Following an intense period of research, it was decided that the third variation of RAG in this project should be based on GraphRAG (as described in [8]).

7.3.2 Neo4j database setup, operations

Neo4j ([22]) was used as the main database for this part of the project due to its unique intersection of public availability, accessibility via a wide variety of programming languages (in case parts of the project ever have to get rewritten in a different language) and the existence of a REST API-based interface for direct interaction with the data.

Due to this choice, a significant portion of the code in this section consisted of queries in Cypher ([15]), an SQL-like language Neo4j uses.

By default, Neo4j automatically creates a database upon its first launch, so only methods for interacting with the graph had to be developed. That is exactly how the class `DatabaseOperations` came to be, containing functions for adding *Document*, *Chunk* and *Entity* objects, as well as establishing relationships between them. Additionally, this class includes a `.del_all()` method, in case there’s a real need to delete the whole database.

7.3.3 Named entity recognition

In order to properly utilise the `DatabaseOperations` class, there needed to be entities which could be added to the graph. Now, at this point in the development process, there were

three different options:

- Detect entities strictly by using prompt engineering and calling an LLM
- Detect entities using `spaCy`'s natural language processing algorithm ([9])
- Combine the two approaches from above

Exclusively using prompt engineering could lead to wide fluxuations in the answer between identical calls, while using only `spaCy` ([9]) would lead to a loss of important, meaningful information about the entities themselves and their relationships.

This led to the creation of the `NERInstance` class. It combines the 3 steps of our entity recognition pipeline into one class:

- 1 Using `spaCy`, extract entities from the original text
- 2 Using the list of extracted entities, a predefined list of entity types and a slightly altered (Appendix C) prompt from the GraphRAG research paper ([8]), extract additional information about said entities and their relationships
- 3 Using regular expressions, parse the LLM's output and return two lists - of dictionaries with extracted entities and of dictionaries with the entities' relationships

7.3.4 Communities via Leiden, community summaries

A crucial piece was still missing, as working with a set of at least thousands of interconnected, yet ungrouped nodes is, at best, exceedingly complicated. Hence, just like in the GraphRAG study, community detection had to be performed. For this, we, too, have decided to use the *Leiden algorithm* ([34]) - specifically, its implementation available via the `GraphDataScience` plugin for `Neo4j` ([18]). The main algorithm is as follows:

- 1 Assign every node in the graph its own community
- 2 Choose any node and compare the value of the modularity function depending on which community the node is moved to
- 3 The permutation which leads to the highest modularity should be chosen, and each of the nodes which neighbour the recently moved node should be considered
- 4 Once every node in the graph has been processed, "refine" the partition by running steps 1-3 within each of the communities

- 5 Aggregate data by combining each of the subcommunities into single nodes within the communities
- 6 Iterate over steps 1-5 until every community has only one node each

The modularity function used within `gds.Leiden` specifically is:

$$Q = \sum_{ij} \left(A_{ij} - \gamma \frac{k_i k_j}{2m} \right) \delta(c_i, c_j),$$

where A_{ij} represents the edge weight between nodes i and j ; k_i is the sum of weights of edges attached to node i ; m is the graph-wide sum of edges; $\delta(c_i, c_j)$ acts as a boolean function which returns 1 if i and j reside in the same community; and γ is a linear resolution parameter which is set to 1 by default in GDS.

One could possibly point out a significant drawback of this algorithm - none of the nodes can belong to more than one community, which may lead to lost data later down the road. However, in practice, this rarely resulted in any actual data loss.

With this algorithm in mind, the `Communities` class was developed (inheriting methods and parameters from `DatabaseOperations`). This class contained a function which ran the Leiden algorithm on the whole graph and supplied relevant labels to each of the Entity nodes; a function which utilised those labels to create Community nodes connected to each of the relevant entities; and several other functions.

The most important of these was `Communities.generate_summaries()`. It went through each of the communities present on the graph, collected information on nodes within the community and their relations to other nodes, and then passed all of the information along to an LLM, which then provided a summary for each community node.

7.3.5 Adding data

Of course, the data from our database still had to be added to Neo4j one way or another, which is why the `ChunkingClient` and `AddData` classes were created. `ChunkingClient`, as the name suggests, helped split a text document into chunks for easier LLM processing, while `AddData` combined all of the above classes into two functions:

- `.add_data()`, which adds every entity and relationship from a given file to the graph
- `.redo_communities_and_summaries()`, which makes several calls to the `Community` methods, resulting in recalculated communities and new summaries for each of them

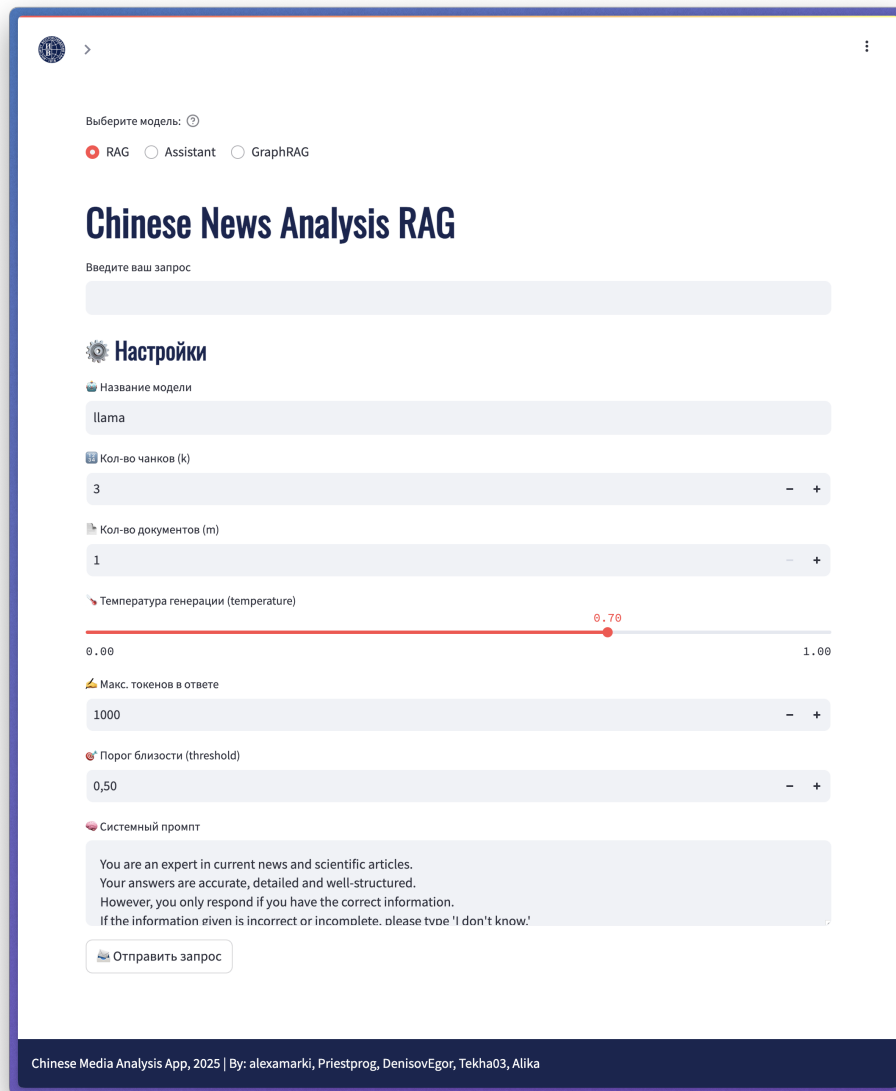


Figure 7.1: RAG Interface

7.3.6 User query handler

Finally, the system needed a way to leverage all of its data in order to provide a user with an informed answer. Due to this, the `GraphRAGQuery` class was created. Every query passed through its methods has to first go through `spacy`'s NER function, after which the community summaries of each of the communities these entities belong to get added to the LLM as context for the query itself.

7.4 Streamlit

We have also developed a `streamlit` interface (Figure 7.1) which offered users the ability to switch between different RAG implementations and send queries to each of the three.

id	username	role	email	password_hash	created_at
----	----------	------	-------	---------------	------------

Table 8.1: The Users table

id	user_id	rag_type	query_text	response_text	query_time
----	---------	----------	------------	---------------	------------

Table 8.2: The Query History table

8 User accounts

8.1 Approach

As the main purpose of this project is, in one way or another, to function like a search assistant for Chinese news articles, it only made sense that some users would wish to save their query history. Additionally, while there exists functionality for adding new, relevant data to the databases via the UI, giving access to such pages to all users could result in some major data skew.

This is why a decision was made to add a "Log in" widget to the website and a history page for logged in users. From the backend, this was powered by a PostgreSQL ([24]) database, which will be discussed below.

Additionally, users were separated into role groups, which determined how much access to the service the user had:

- Users: get access to RAG & History sections of the website
- Admins: get access to everything Users have + SearchAPI and text processing (to help populate the database with relevant information)
- Superusers: in addition to Admin access, they get account creation and deletion capabilities, as well as access to certain high-workload database operations, such as redoing community summaries.

8.2 Table setup

With the help of SQLAlchemy ([30]), two ORM models for the tables were created: **Users** (Table 8.1) and **QueryHistory** (Table 8.2)

A `.create_tables()` function was also developed, which would create these tables and add the superuser account to the Users table.

8.3 User operations

In order to facilitate interaction with the tables, a `TableOperations` class was created. It included methods for creating/deleting a user account, checking one's login credentials, as well as checking a user's query history and adding/removing records from the History table

9 Utilities & Secrets

Due to a high amount of shared processes across the project and the existence of secrets and certain locations for storing data, a set of utility classes, functions and files was created.

This includes `CloudS3Client`, the base class every class interacting with upload to S3 Storage is derived from; `YCloudMLClient`, the base class any method operating on Yandex datasets or using *Yandex Foundation Models* is derived from, as well as `TypesOperations` - the class whose methods get applied whenever something has to be done with multiple files of the same filetype. Additionally, there exist two files storing data loaded from the `.env` file, as well as resolved paths both within the backend and on an external volume.

10 Frontend

As of right now, the website itself runs on the `streamlit` (website: [\[32\]](#)) framework.

10.1 Approach

This specific framework was chosen due to its relative ease of use compared to other web frameworks - this project did not have any necessary aspect which `streamlit` would not allow to implement, so the choice ended up being justified. Most of the pages on the frontend have already been outlined previously, which is why this chapter will only discuss the process of combining all of them into one whole app.

10.2 Implementation

`Streamlit`'s multipage functionality was used to reach this exact goal - several `streamlit.Page()` ([\[31\]](#)) objects were initialised and the sidebar of the website got populated differently, depending on the user's authentication and role. Even if the user managed to reach a page which did not show up in their sidebar, they would not be allowed to view any content on the page, ensuring the enforcement of permission rules.

With the use of `streamlit`'s localisation capabilities, a language switch was also added, allowing the user to switch between Russian and English as languages of the interface.

10.3 Design

While `streamlit` offers a user-friendly default look, it did not feel quite right for this project. In pursuit of inspiration, we accessed the Institute's new website ([14]) and decided on applying the same colour and font palette to the app using `CSS` in order to make it feel right at home on the website (Figure 10.1).

Additionally, a footer with information about the contributors and a logo with a link to the Institute's website were added, making the app feel complete.



Figure 10.1: Changes in the interface after applying `CSS`

11 Docker & FAST API

11.1 Approach

In order to make the project easy to deploy, it had to be separated into smaller services - containers. In order to make sense of how exactly that should be done, we needed to sketch a relatively detailed scheme (Figure 11.1) of the project. Having done that, we managed to obtain clarity in regards to the arrangement of folders within the codebase, rearrange multiple files and realise which requests would be sent using each of the submodules.

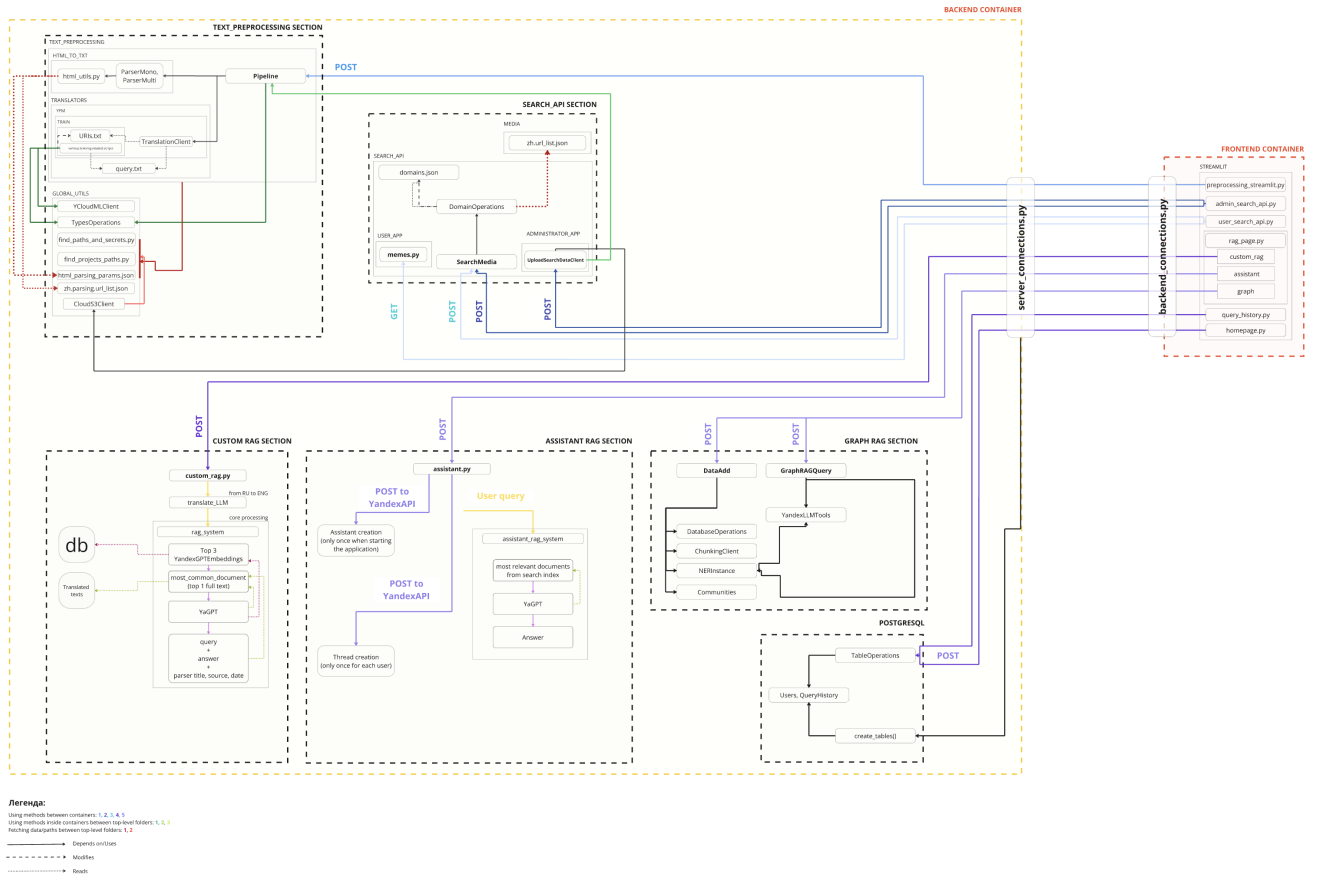


Figure 11.1: Project Scheme

11.2 Container separation

Having arrived at the decision of separating the project into two containers - backend and frontend, we created two `Dockerfile` and `requirements.txt` files - one for each container. We also made a `docker.compose` file, which additionally initialised a shared volume (for exchanging data between the frontend and the backend) and `PostgreSQL` and `Neo4j` containers in order to accommodate our database needs. Finally, a bash script called `start.sh` was created to ensure a 1-command setup experience of the project right out of the box.

11.3 Controllers

Throughout the project's move to Docker, it was clear that the backend and the frontend would not be able to communicate the same way as before, which is why a `FastAPI` ([10]) server was set up on the backend. Thanks to our project scheme, it was easy to figure out which services had to be added to the server. All of these services were added to a file called `server_connection.py`, where they accepted frontend requests using predefined `Pydantic` ([35]) request schemas. From the frontend side, the functions sending `POST` and `GET` requests to these services were also separated into a file - `backend_connections.py`.

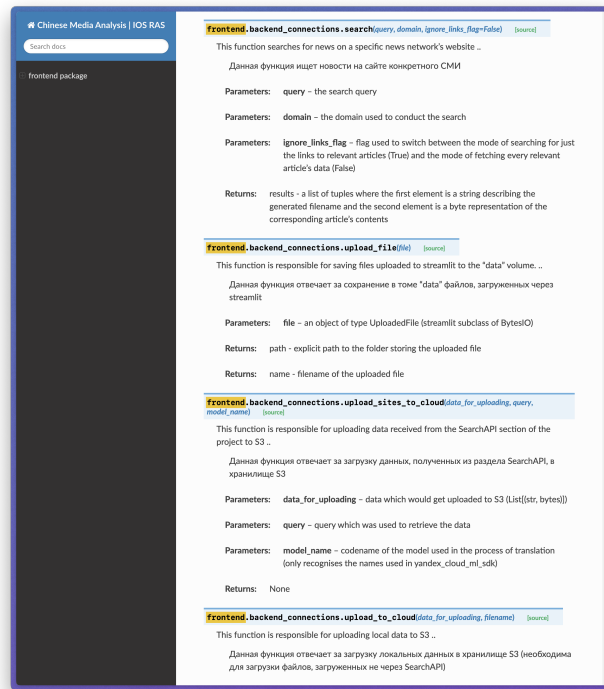


Figure 12.1: Documentation

12 Documentation

12.1 Docstring

Early on, it was decided that every function in every file should have at least basic pythonic documentation in case any other developers come along and start working on the project. Therefore, by the end, every function and every class had docstrings abiding to the *Sphinx ReStructuredText documentation guidelines* ([27]). Additionally, drawing inspiration from *Google's documentation guidelines* ([1]), every file now had a docstring at the top, outlining its general purpose in the project.

12.2 Sphinx

With the use of **Sphinx** ([29]), a prevalent documentation creation tool (used by a variety of projects, such as *ReadTheDocs* ([11])), we managed to create an html-based view of our documentation (Figure 12.1), making it a lot easier and more comfortable to navigate. It should also be noted that this documentation is available in English and Russian (albeit, the latter isn't completely translated yet - there is still some work to do).



Figure 14.1: Homepage

13 Logging

Due to problems with debugging which arose slowly, but surely over the development process, logging was implemented in each of the `.py` files. This helped immensely when trying to root out the causes of the bugs which appeared during the most recent feature additions, cutting the time spent by several orders of magnitude.

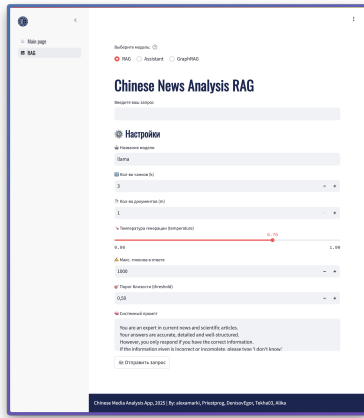
14 Current user experience

14.1 Homepage

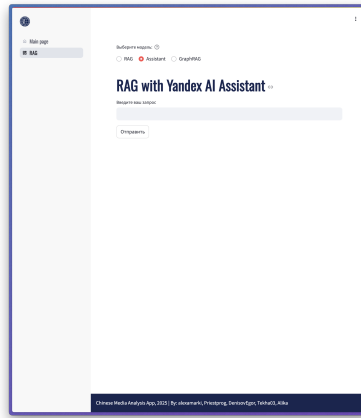
Upon opening the page, the user is greeted with the welcome screen (Figure 14.1), from where they can either log in or switch to the RAG page

14.2 RAG

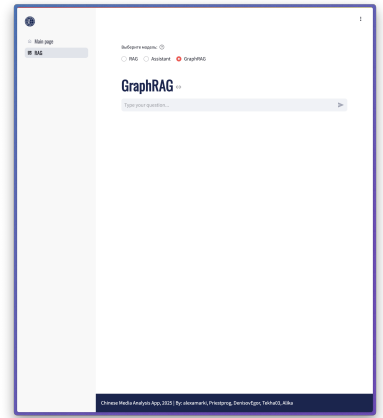
The RAG page (Figure 14.2) features a radio-style RAG implementation picker. On each of the subpages, the interface is mostly similar. The user only has to write their question, press send and await a response from the system.



(a) Custom RAG



(b) Assistant RAG



(c) Graph RAG

Figure 14.2: RAG Interface

14.3 Query history

Provided the user is logged in, they may visit this page (Figure 14.3) to look through their query history for each of the RAG implementations and delete what is no longer needed.

14.4 File processing

On the file processing page (Figure 14.4), one has the ability to process a file and press "Save to Cloud" in order to add the uploaded data to the databases responsible for the RAG systems.

14.5 Article search

Finally, on the article search page, the user can choose a specific Chinese media website they wish to obtain relevant data from. Additionally, the user can modify the domain list, removing unwanted entries.

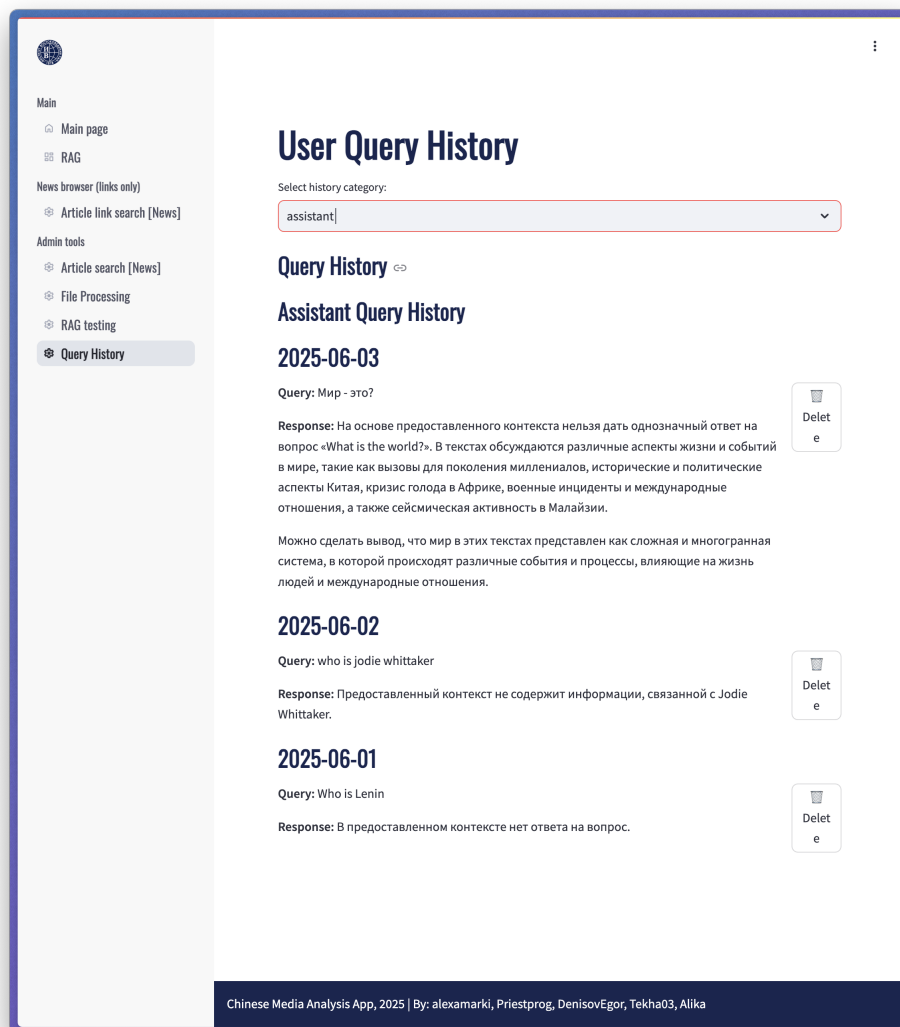


Figure 14.3: Query history for Assistant RAG

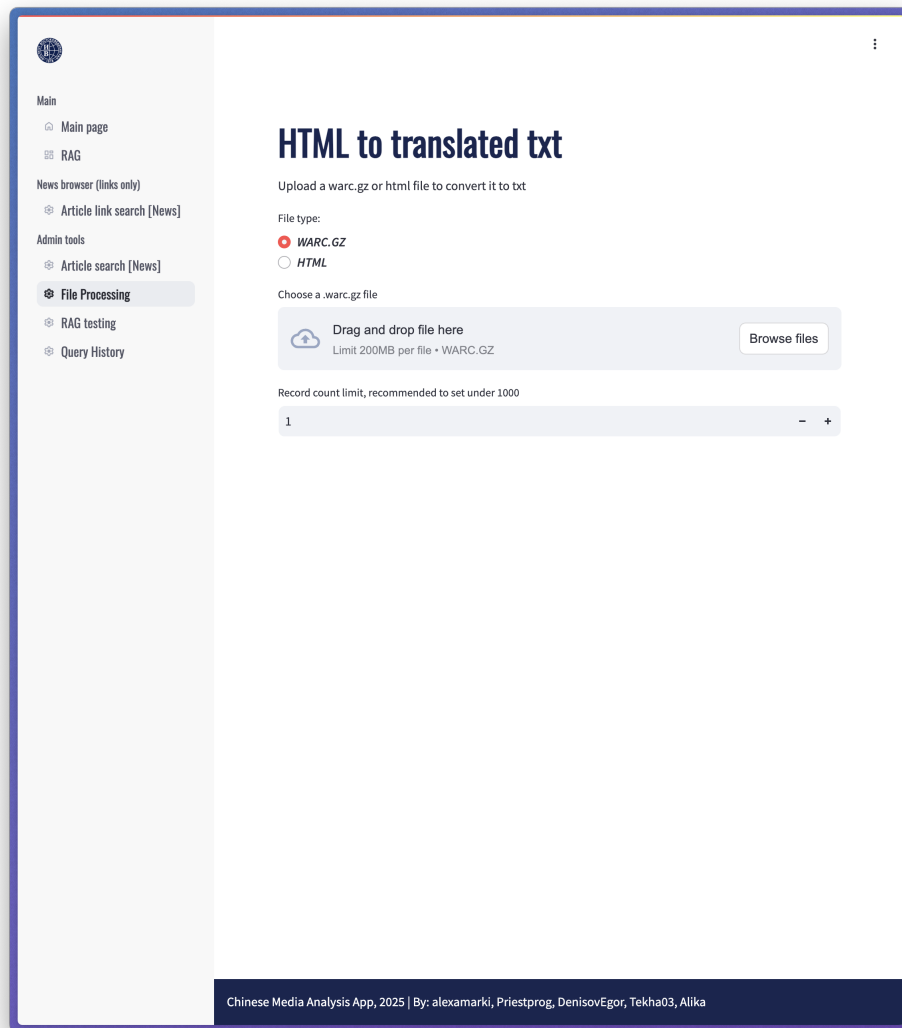


Figure 14.4: Processing interface

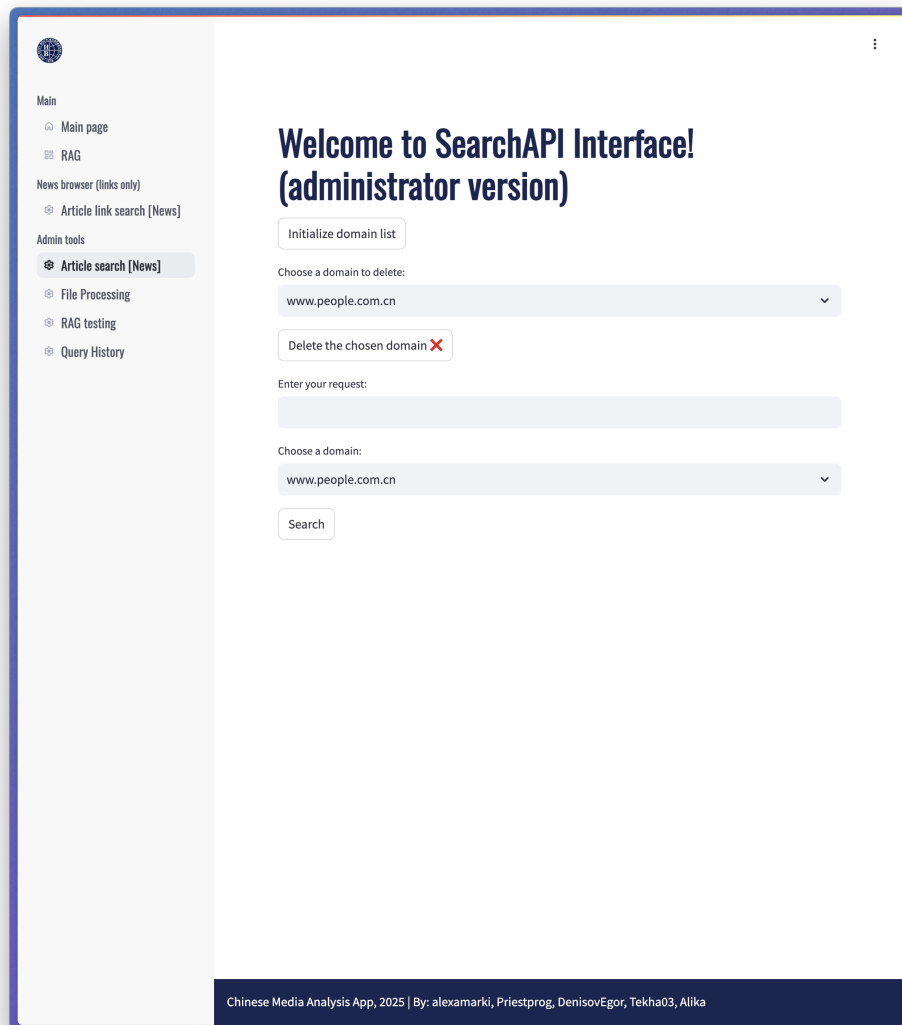


Figure 14.5: Article search

15 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

16 Conclusion

Overall, over the last 6 months, our team has achieved notable results. Having integrated all the components into one rigid system, we finally managed to achieve stability in the project's performance. Frontend, backend, databases, the shared volume and external services, now all interact with each other correctly, working on users' requests in unison, ensuring the system's reliability and efficiency.

Finally, logging was introduced, making it much easier to monitor the service and root out any problems before it is too late.

We expect the website with the service to launch on the Institution's website during 2025, although that still cannot be said for certain. As of now, it's already publicly available, albeit on a separate IP.

References

- [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: Accelerated Container Application Development*. URL: <https://www.docker.com/>.
- [8] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitan, 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/>.
- [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.

- [18] *Leiden - Neo4j Graph Data Science*. URL: <https://neo4j.com/docs/graph-data-science/current/algorithms/leiden/>.
- [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).
- [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/>.
- [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.

- [35] *Welcome to Pydantic - Pydantic*. URL: <https://docs.pydantic.dev/latest/>.
- [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] *Yandex Tracker* — *yandex.cloud*. URL: <https://yandex.cloud/en-ru/services/tracker> (visited on Feb. 4, 2025).
- [41] *YandexGPT 5 - the new generation of Yandex LLM*. URL: <https://ya.ru/ai/gpt>.

A Glossary

BeautifulSoup 4 - a Python library for scraping/parsing web pages, HTML files and XML files

Debugging - a process of finding and fixing possible errors/problems in code

Deploy - the process of publishing a product/its new version in the environment accessible by the end user

Entity Recognition - the task of entity classification in an unstructured text

Fine-tuning - the process of training a pre-trained neural network on new data. Most often, it is used in order to modify the output structure

Flask - a Python framework for creating web apps

LLM (Large Language Model) - a type of machine learning model with the purpose of performing natural language processing tasks.

LLM Embedding - conversion of LLM tokens into vectors

LLM Tokenisation - conversion of a text into smaller words/subwords/characters

LoRA (Low-Rank Adaptation) - an optimised variation of fine-tuning, taking up less storage and requiring less data

Machine Translation - computer-assisted translation. In this paper, the term refers specifically to LLM-assisted translation

ORM (Object-Relational Mapping) - an approach to converting data between an object-oriented programming language and a relational database, such as PostgreSQL

Parsing - analysing and converting data of one format into another

Pipeline - a set of various code elements connected in a sequential manner (after the former element stops running, the latter starts, often using the former's output as a part of its input)

Prompt Engineering - the process of optimising instructions sent to an LLM in order to get the best possible results

RAG (Retrieval-Augmented Generation) - an approach to text generation which grants LLMs the ability to base their answers on specific predefined data

REST (REpresentational State Transfer) - an approach to web service development which assumes interaction between the frontend and the backend via HTTP

S3 (Simple Storage Service) - a web service providing object storage capabilities

Sentiment Analysis - the task of classifying a piece of text by its emotional tone (*positive, negative, neutral*)

Streamlit - a Python framework for interactive data apps

Summary Generation - the task of summarising a text

Token Limit - the limit on the input or output size in terms of tokens. In this paper, the term is only used in reference to the input limit rather than the output limit

Yandex Cloud - a cloud platform providing storage, development and machine learning tools. Owned by *Yandex LLC*.

Yandex SearchAPI - a Yandex Cloud service allowing users to index and search website contents

Yandex Tracker - a Yandex Cloud project management service

B Translation Prompt

You are a translation tool specializing in translating text from Chinese to English. Your task is to translate text in such a way that original hieroglyphic Chinese place names, personal names, and institution names are preserved in parentheses next to their English translation;

You SHOULD:

1. All text elements must be translated into English as accurately as possible.
2. English translations of *Chinese* place names (cities, streets, natural landmarks, etc.), personal names, and institution names (schools, universities, companies, etc.) should always be followed by their original Chinese name in hieroglyphics in parentheses.
3. If the text contains abbreviations or acronyms related to *Chinese* institutions, their translation should be supplemented with the original Chinese version - in hieroglyphics.
4. Ensure the sentence structure remains logical and readable for English speakers.
5. The translation should always begin with the article name, the source and the publication date.

You SHOULD:

1. NOT Return any text other than the translation of the article
2. NOT Translate or output sections related to "navigation", "similar/other articles", "contact info", "advertisements"
3. NOT insert full translations of a sentence/section after said section
4. NOT summarise text, since the translations have to be as complete as possible

5. NOT translate text that is already in English
6. NOT include any sections of the original text which seem like a repetition of text from the website or an ad

C Entity Recognition Prompt

Core Objective

Given a text document which is a snippet from a news article, a list of entities and a list of types, link each entity with its proper type and identify all relationships among these entities.

Steps

1. Find all the entities in the text. For each entity, extract the following information:

entity_name: Name of the entity, capitalised
entity_name_zh: A Chinese name of the entity (if found in brackets next to the entity, otherwise NONE)
entity_type: one of the following types: [entity_types]
entity_description: A comprehensive description of the entity's attributes and activities

Format each entity as (entitytuple_delimiter<entity_name>tuple_delimiter<entity_name_zh>tuple_delimiter<entity_type>
tuple_delimiter<entity_description>)

Note: if there are multiple entries of the same entity in the text (translation to a certain language, variable spelling, abbreviation [i.e., US vs United States]), combine them into one entity.

2. From the entities identified in step 1, identify all pairs of (source_entity, target_entity) that are *clearly related* to each other.

For each pair of related entities, extract the following information:

source_entity: name of the source entity, as identified in step 1
target_entity: name of the target entity, as identified in step 1
relationship_name: a short name explaining the relationship type
relationship_description: explanation as to why you think the source entity and the target entity are related to each other
relationship_strength: an integer score between 1 to 10, indicating strength of

the relationship between the source entity and target entity

Format each relationship as (relationshiptuple_delimiter<source_entity>

tuple_delimiter<target_entity>

tuple_delimiter<relationship_name>

tuple_delimiter<relationship_description>

tuple_delimiter<relationship_strength>)

3. Return output as a single list of all the entities and relationships identified in steps 1 and 2. Use record_delimiter as the list delimiter.

4. If you have to translate into language, just translate the descriptions, nothing else!

5. When finished, output completion_delimiter.

#

Example (DO NOT USE THIS IN YOUR ANSWER)

#

Entities: TechGlobal's, , Vision Holdings

Entity_types: ORGANIZATION

Text:

TechGlobal's () stock skyrocketed in its opening day on the Global Exchange Thursday. But IPO experts warn that the semiconductor corporation's debut on the public markets isn't indicative of how other newly listed companies may perform.

TechGlobal, a formerly public company, was taken private by Vision Holdings in 2014. The well-established chip designer says it powers 85% of premium smartphones.

#

Output:

(entitytuple_delimiterTECHGLOBALtuple_delimiterttuple_delimiterORGANIZATION

tuple_delimiterTechGlobal is a stock now listed on the Global Exchange which powers 85% of premium smartphones)

(entitytuple_delimiterVISION HOLDINGStuple_delimiterNONEtuple_delimiterORGANIZATION

tuple_delimiterVision Holdings is a firm that previously owned TechGlobal)

(relationshiptuple_delimiterTECHGLOBALtuple_delimiterVISION HOLDINGS

tuple_delimiterFORMER_OWNERtuple_delimiterVision Holdings formerly owned TechGlobal from 2014

until presenttuple_delimiter5)

completion_delimiter

```
#  
#  
# Now, the data you will be operating on:  
  
entities:  entities  
entity_types:  entity_types  
text:  text  
  
output:
```