

本科期末报告

（主修专业）

**Cocoon Breaker**

茧房终结者

姓名: 廖梓强 22920212204414

曹丹颖 22920212204347

邓明宇 22920212204365

薛椀如 22920212205270

学院: 信息学院

专业: 数字媒体技术

年级: 2021级

课程名: 人工智能导论 任课教师: 姚俊峰（教授）

二〇二四年六月十六日

**Abstract**

In today’s digital age, personalized content curation and recommenda- tion algorithms have led to a common problem known as the “Information Cocoon”. This phenomenon occurs when individuals become increasingly isolated within their own information bubbles. To address this issue, we have developed a project called “Cocoon Breaker” that utilizes advanced AI technologies.The goal of our project is to help users break out of their information cocoons by gathering and organizing diverse information. By doing so, we aim to provide users with a more comprehensive understanding of various topics and social phenomena. Our system consists of several key components, including text embedding, sentiment analysis, video processing, and fact verification. These components are integrated into a user-friendly platform.

Our system offers detailed video summaries, event analysis, and the presentation of diverse viewpoints. These features aim to enhance users’ in- formation literacy and promote social cohesion. We have designed the system with a modular architecture, allowing for easy integration and asynchronous processing. We have also developed effective deployment strategies to ensure smooth operation. Through extensive evaluation, we have demonstrated the system’s effectiveness in improving user access to diverse perspectives. We believe that our project has the potential to foster an informed and har- monious society. In the future, we plan to optimize the analysis workflow, expand support for multiple platforms, and incorporate interactive features to further enhance user experience and engagement.

**Keywords:** Information Cocoons, Artificial Intelligence, Large Language Model, Deep Learning

**Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| [**1 Introduction**](#_bookmark0) | | | **1** |
| [1.1 Background](#_bookmark1) . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | | 1 |
| [1.1.1 Concept of Information Cocoons](#_bookmark2) . . . . . . . . . . . . . | | | 1 |
| [1.1.2 Social Context of Information Cocoons](#_bookmark3) . . . . . . . . . | | | 2 |
| [1.1.3 Dangers of Information Cocoons](#_bookmark4) . . . . . . . . . . . . . | | | 2 |
| [1.1.4 Conclusion](#_bookmark5) . . . . . . . . . . . . . . . . . . . . . . . . . | | | 3 |
| [1.2 User Survey](#_bookmark6) . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | | 3 |
| [1.3 Analysis of Similar Products](#_bookmark7) . . . . . . . . . . . . . . . . . . . | | | 5 |
| [1.4 Project Description](#_bookmark8) . . . . . . . . . . . . . . . . . . . . . . . . | | | 8 |
| [1.4.1 Project Goals](#_bookmark9) . . . . . . . . . . . . . . . . . . . . . . . | | | 8 |
| [1.4.2 Main Features](#_bookmark10) . . . . . . . . . . . . . . . . . . . . . . . | | | 9 |
| [1.4.3 Project Module Design](#_bookmark11) . . . . . . . . . . . . . . . . . . | | | 10 |
| [**2 Related Works**](#_bookmark12) | | | **12** |
| [2.1 Text Embedding](#_bookmark13) . . . . . . . . . . . . . . . . . . . . . . . . . | | | 12 |
| [2.1.1 BERT: A Revolutionary Approach](#_bookmark14) . . . . . . . . . . . | | | 12 |
|  | [2.1.2](#_bookmark15) | [BERT Variants and Multilingual Capabilities](#_bookmark15) . . . . . | 12 |
|  | [2.1.3](#_bookmark16) | [Applications and Benefits](#_bookmark16) . . . . . . . . . . . . . . . . | 13 |
|  | [2.1.4](#_bookmark17) | [Clustering Method](#_bookmark17) . . . . . . . . . . . . . . . . . . . . | 14 |
| [2.2](#_bookmark18) | [LLMs](#_bookmark18) | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 15 |
|  | [2.2.1](#_bookmark19) | [Two Major Tasks](#_bookmark19) . . . . . . . . . . . . . . . . . . . . . | 15 |
|  | [2.2.2](#_bookmark20) | [Traditional Methods](#_bookmark20) . . . . . . . . . . . . . . . . . . . | 16 |

* + 1. [Large Model Approaches](#_bookmark21) 17
    2. [Principles of Large Models](#_bookmark22) 20
    3. [Common Large Models Comparison](#_bookmark23) 21
    4. [Prompt Engineering](#_bookmark24) 22
  1. [Kity Minder](#_bookmark25) 24
  2. [Sentiment Analysis](#_bookmark26) 25
     1. [Introduction to Sentiment Analysis](#_bookmark27) 25
     2. [Multilingual Sentiment Analysis Model](#_bookmark28) 26

1. [Architecture Design](#_bookmark29) 27
   1. [Overall Architecture](#_bookmark30) 29
   2. [Web Plugin](#_bookmark32) 31
   3. [Frontend and Backend Design](#_bookmark33) 32
      1. [Frontend Design](#_bookmark34) 32
      2. [Backend Design](#_bookmark35) 36
   4. [Video Processing](#_bookmark36) 39
      1. [Video Information Retrieval](#_bookmark37) 39
      2. [Video Text Extraction](#_bookmark38) 42
      3. [Video Comment and Danmu Crawling](#_bookmark39) 45
   5. [Online Search](#_bookmark40) 48
   6. [GPT Analysis](#_bookmark41) 50
      1. [Video Summary](#_bookmark42) 50
      2. [Extracting Search Keywords](#_bookmark43) 52
      3. [Event Analysis](#_bookmark44) 52
      4. [Fact Verification](#_bookmark45) 54
   7. [Mind Map Generation](#_bookmark46) 56
   8. [Opinion Analysis](#_bookmark47) 59
      1. [Sentiment Analysis](#_bookmark48) 59
      2. [Clustering Analysis](#_bookmark49) 61
2. [Evaluation](#_bookmark50) 63
   1. [Stability Experiment](#_bookmark51) 63
   2. [Effectiveness Evaluation](#_bookmark52) 65
   3. [Duration Evaluation](#_bookmark53) 66
3. [Future Outlook](#_bookmark54) 68
   1. [Model Cost Issue](#_bookmark55) 68
   2. [User Account](#_bookmark56) 69
   3. [Optimization of Analysis Workflow](#_bookmark57) 70
   4. [Multi-Platform Support](#_bookmark58) 71
   5. [Hot Topic Recommendations](#_bookmark59) 73
   6. [Interactive System](#_bookmark60) 74
4. [Conclusion](#_bookmark61) 77

**Chapter 1: Introduction**

* 1. **Background**

In the current digital era, the prevalence of personalized content cura- tion and recommendation algorithms has led to the emergence of the “in- formation cocoon” phenomenon. This concept, first introduced by Harvard Professor Cass Sunstein, refers to the situation where individuals are increas- ingly isolated within their own informational niches due to selective exposure to content that aligns with their pre-existing beliefs and preferences[[1](#_bookmark62)]. As the internet has evolved, especially with the advent of platforms like Tik- Tok, Toutiao, and Zhihu, the issue of information cocoons has become more pronounced. These platforms utilize sophisticated algorithms to create de- tailed user profiles and deliver highly personalized content, which, over time, confines users to a narrow circle of information[[2](#_bookmark63)].

* + 1. **Concept of Information Cocoons**

The term “information cocoon” was coined by Sunstein to describe the process by which individuals, through selective exposure to informa- tion sources that align with their interests, gradually become encased in a metaphorical cocoon of homogeneous information[[1](#_bookmark62)][[2](#_bookmark63)]. This process is fa- cilitated by the use of recommendation algorithms that track user behavior and preferences, subsequently curating content that is likely to keep the user engaged. Over time, this leads to a narrowing of the information landscape, where users are repeatedly exposed to similar viewpoints and content, effec-

tively isolating them from contrasting information[[3](#_bookmark64)].

* + 1. **Social Context of Information Cocoons**

In today’s interconnected world, the internet serves as a vast and com- plex information dissemination platform. It functions not only as a tool for personal information processing but also as a medium for interpersonal, group, and organizational communication[[3](#_bookmark64)]. However, the rise of social me- dia and personalized content platforms has significantly changed how in- formation is consumed. The shift towards algorithm-driven content deliv- ery has led to a scenario where users are bombarded with homogeneous in- formation, reinforcing their existing views and limiting exposure to diverse perspectives[[4](#_bookmark65)]. This phenomenon is often referred to as the “information co- coon”, where individuals are trapped in a self-created bubble of information that reflects their personal biases and preferences[[1](#_bookmark62)].

* + 1. **Dangers of Information Cocoons**

Information cocoons pose several significant risks to both individuals and society at large. Firstly, they exacerbate the phenomenon of group po- larization, where users immersed in homogeneous information environments become more extreme in their views[[4](#_bookmark65)]. This is particularly evident in online communities, social networks, and discussion forums, where like-minded in- dividuals reinforce each other’s opinions, leading to more radicalized group behavior[[4](#_bookmark65)][[5](#_bookmark66)]. Secondly, the prevalence of information cocoons undermines social cohesion by reducing the instances of shared experiences and common knowledge, which are essential for societal bonds[[3](#_bookmark64)]. As people increasingly rely on personalized content, their interactions with diverse viewpoints di- minish, leading to a fragmented society with diminished social stickiness[[3](#_bookmark64)][[5](#_bookmark66)].

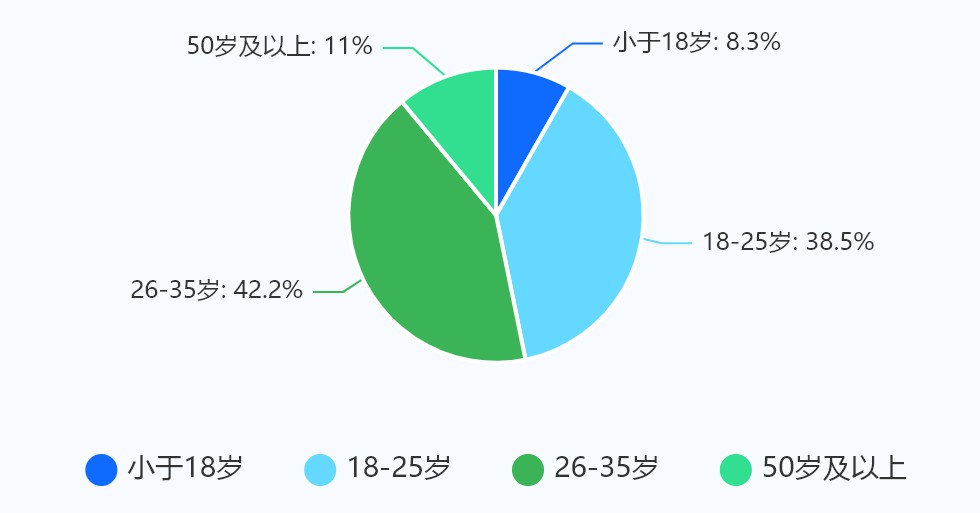
* + 1. **Conclusion**

The information cocoon phenomenon represents a significant challenge in the digital age, requiring a multifaceted approach to address its implica- tions. By understanding the mechanisms behind information cocoons and their impact on society, strategies can be developed to mitigate their neg- ative effects. These strategies include enhancing algorithmic transparency, promoting media literacy, and encouraging exposure to diverse viewpoints. Addressing the issue of information cocoons is crucial for fostering an in- formed and cohesive society.

Therefore, with a sense of social responsibility, we have decided to use technological means to assist users in breaking out of their comfort zones at a lower cost, gaining access to more information beyond their informa- tion cocoons. We will utilize AI technology to aggregate and organize this information, helping users form a more comprehensive understanding of var- ious matters and social phenomena. We believe that our system will play a positive role in promoting social harmony and stability.

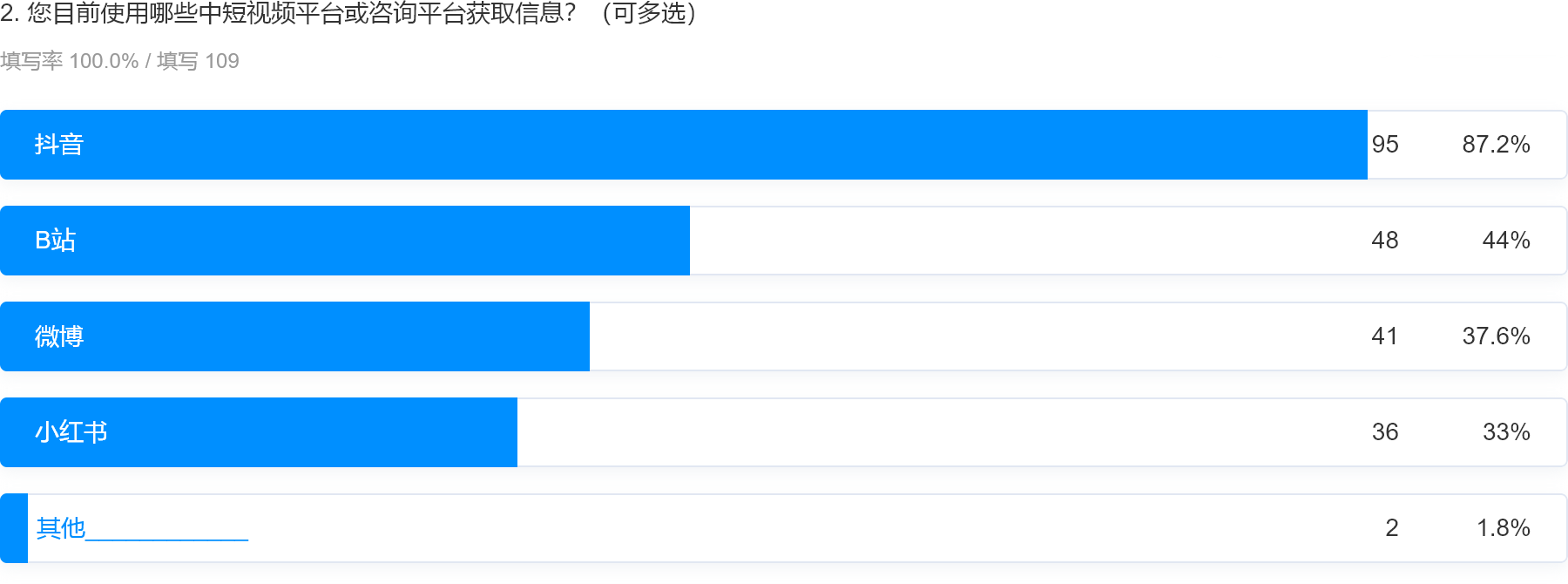
* 1. **User Survey**

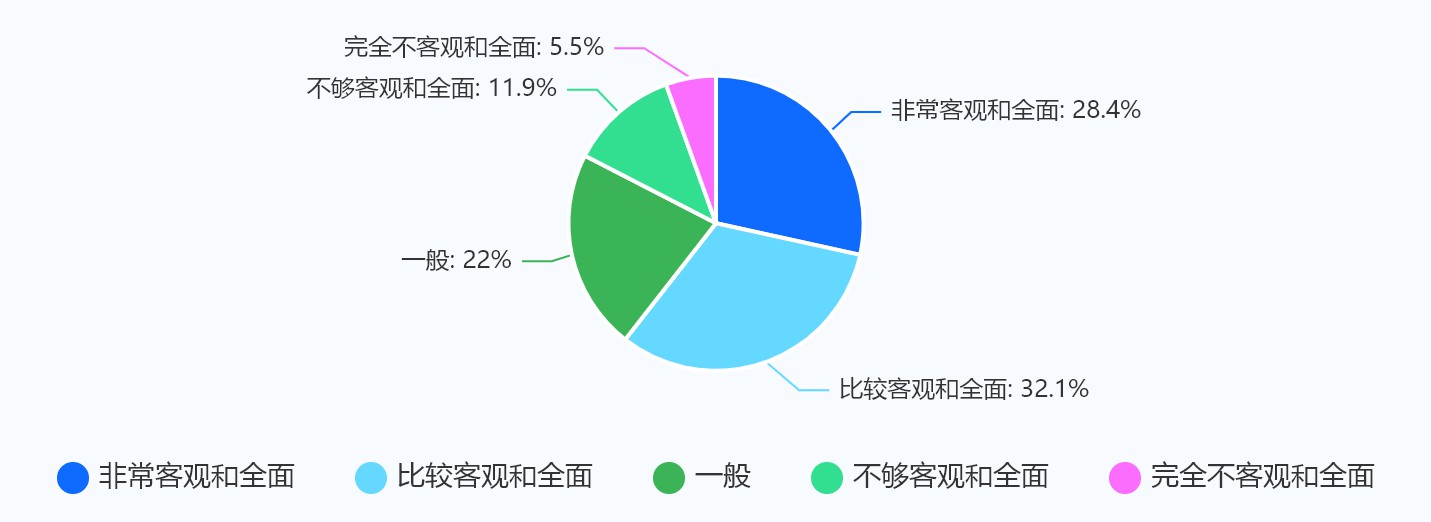
We conducted a survey of 110 users, with the following distribution of ages:



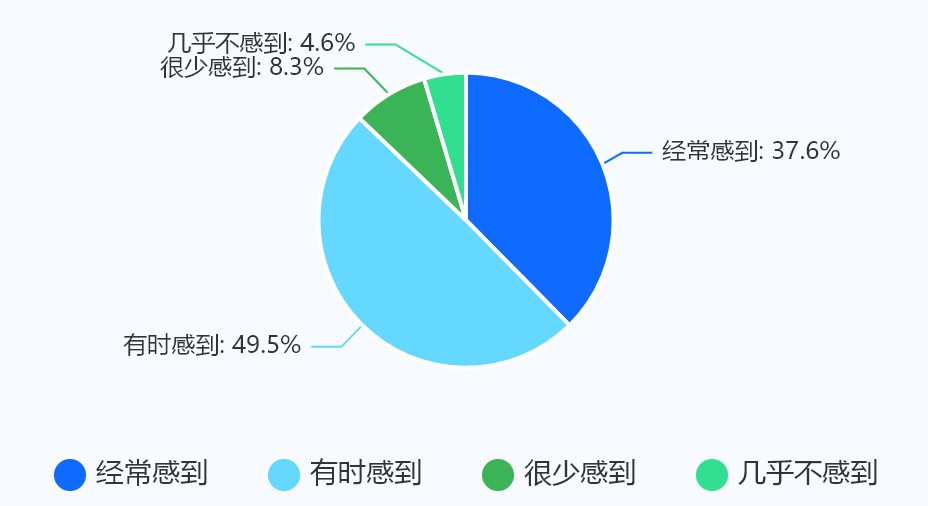
The survey results indicate that the primary sources of information for

most Chinese users are TikTok, Bilibili, Weibo, and Xiaohongshu.



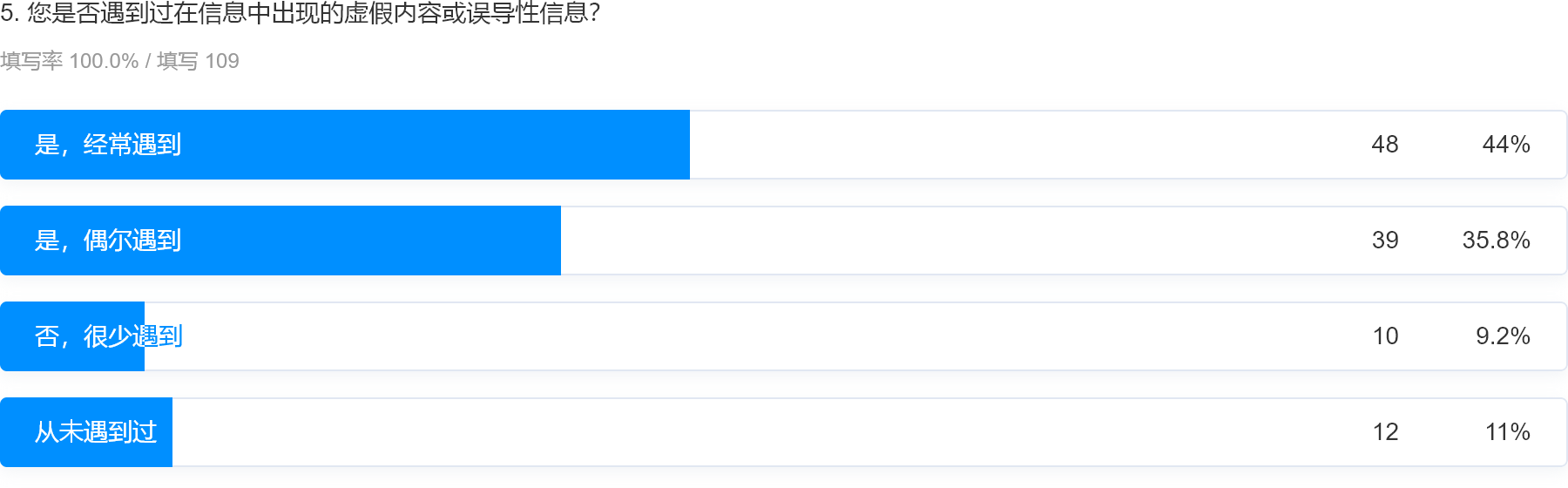
Although most users believe that modern information is sufficiently com- prehensive, there are still some users who perceive that certain information may contain subjective views from the publishers, leading to incorrect dis- semination of information.

From the survey results, it is evident that the majority of users often feel overwhelmed by information overload when browsing the internet.

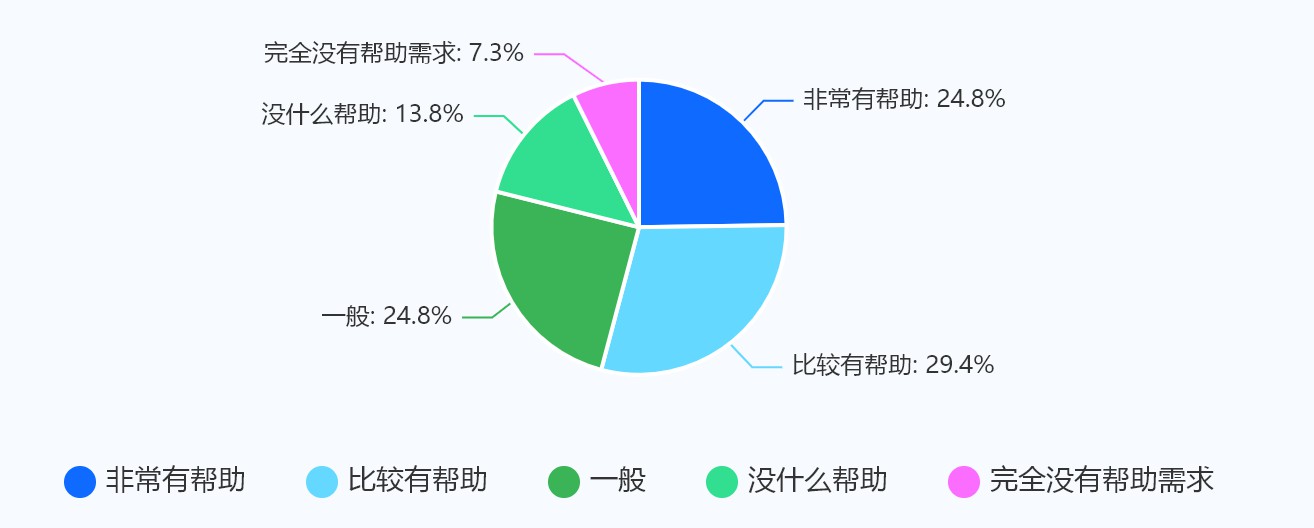


The research findings indicate that the majority of users have encoun-

tered false or misleading information in the content they consume.



Overall, the survey results indicate that the “Cocoon Breaker” project is helpful to a certain extent for the majority of users.



* 1. **Analysis of Similar Products**

In the current market, several platforms and products aim to address the issue of information cocoons and provide diverse content perspectives. Below are the major competitors and an analysis of their functionalities:

? **Braver**: Braver is a browser extension that displays multiple news view- points to help users break out of information cocoons. Its main advan- tages include being simple and easy to use, as it operates directly within the browser, and offering multiple perspectives for comparison. How- ever, its functionality is limited, primarily focusing on news and lacking in-depth analysis and customization options.

? **Ground News**: Ground News provides multi-source news comparisons, showcasing how different media outlets report on the same event. The

platform’s strengths lie in its ability to offer multi-source news com- parisons, helping users understand different viewpoints, and its user- friendly interface. On the downside, Ground News primarily focuses on news, with less coverage of other types of information, and lacks interactive features.

? **Newsela**: Newsela is an educational platform that provides multi- source news content tailored to different reading levels. It is particu- larly beneficial for educational environments, helping students improve reading skills and media literacy. However, Newsela mainly targets the education market, making it of limited use for general users and not ideal for everyday information retrieval.

? **AllSides**: AllSides classifies news by political bias, presenting multiple perspectives on the same issue. This helps users understand different political standpoints and offers clear categorization. Nonetheless, All- Sides focuses mainly on political news, with limited coverage of other information types, and has limited interactive features.

? **FlipFeed**: FlipFeed is a social media tool that allows users to view feeds from people with opposing viewpoints. It increases exposure to different perspectives and promotes strong social interaction. However, it relies on social media platforms, offering limited coverage, and lacks in-depth analytical functions.

Compared to the above competitors, our project offers unique advan- tages in the following areas:

? **Breaking Information Cocoons**: Our project, “Cocoon Breaker”, is not limited to news alone. It encompasses various types of information,

including social media, news, and videos. By leveraging AI technology, we aggregate information from multiple platforms to provide diverse content perspectives. This encourages users to engage with different viewpoints and break out of their information cocoons.

? **Identifying Information Authenticity**: We introduce an informa- tion authenticity verification feature that assists users in discerning the truthfulness of information. Utilizing AI algorithms and fact-checking databases, our system automatically detects and flags potential misin- formation, helping users navigate through the vast amount of informa- tion available more critically.

? **Information Organization and Mind Map Generation**: Our project offers advanced features for information organization and mind map generation. By employing AI technology, we organize related informa- tion and generate intuitive mind maps. This helps users manage and understand complex information more effectively, facilitating a deeper comprehension and application of the knowledge they acquire.

Our project, “Cocoon Breaker”, not only aids users in breaking out of information cocoons but also includes features for identifying information au- thenticity and organizing information. These components make our project a comprehensive tool for enhancing information literacy and promoting a more informed and cohesive society. By harnessing advanced technologies and innovative approaches, we strive to provide a diverse, transparent, and secure information environment for users.

* 1. **Project Description**
     1. **Project Goals**

**Break Information Cocoons:** This project aims to use technologi- cal means to help users access more diverse information beyond their usual sources and perspectives. Most people currently face a paradox: on one hand, they receive a large amount of current news and thus require more comprehensive information to aid their judgment; on the other hand, most of them do not possess professional investigative skills and have limited time to spend on fact-checking. Therefore, an unbiased, automated program can provide the necessary service in this regard.

**Improve Information Literacy:** Information literacy is an essential skill for modern citizens. It not only includes the ability to identify and obtain useful information but also encompasses evaluating the authenticity and reliability of information. Enhancing information literacy can help the public better cope with false information and misleading media reports, en- abling them to make informed decisions. It also reduces the likelihood of the public being swayed or deceived. Overall, improving information literacy will significantly enhance the public’s ability to navigate a complex information environment.

**Promote Social Harmony:** Social harmony relies on effective commu- nication and mutual understanding between different groups. By promoting cross-cultural exchange and diversity education, misunderstandings and prej- udices can be reduced, fostering coexistence and cooperation. For example, in diverse communities, organizing cultural exchange activities and dialogue platforms can help individuals from different backgrounds share experiences and perspectives, building mutual trust. Additionally, media and educa- tional institutions can play a crucial role by spreading values of inclusion

and respect, reducing social division and antagonism. In summary, promot- ing social harmony not only contributes to social stability but also enhances overall happiness and social cohesion.

* + 1. **Main Features**

**Video Summarization:** Video content is rich and intuitive, but its linear nature often makes it difficult to quickly search and review. Convert- ing video content into text can help users quickly browse and understand the main information of the video, making it easier for them to refer back to and think critically about the content as needed. This textual summary can include the main points, key data, and important conclusions of the video, presented in a clear and concise manner, allowing users to access the necessary information without repeatedly watching the video. Additionally, text summaries can utilize keyword search functions to help users quickly find relevant content within a large number of videos, improving information utilization efficiency. This approach is particularly useful for fields such as education and research, which require repeated in-depth analysis and verifi- cation.

**Fact Verification and Event Analysis:** In an era of information overload, the accuracy of data, events, and scientific conclusions mentioned in videos is crucial. By verifying the facts mentioned in videos, users can be ensured they are not misled. This process includes verifying statistical data cited in the video, conducting timeline analyses of mentioned historical events, and fact-checking scientific conclusions. Furthermore, the program will search the internet for related information, providing a comprehensive analysis report that outlines the complete development of events. This not only helps users understand the authenticity of the video content but also of- fers more comprehensive background information, enhancing users’ judgment

and information literacy.

**Presentation of Diverse Opinions:** In the process of information dissemination and communication, presenting diverse perspectives can ef- fectively promote understanding and interaction between different opinion groups. By collecting and analyzing different viewpoints from multiple com- monly used platforms, users can be provided with a multi-faceted understand- ing of the topic, preventing them from being confined to a single perspective. The presentation of diverse perspectives helps reduce the information bubble effect, promoting social inclusion and consensus building. Additionally, this approach can strengthen communication and interaction between different opinion groups, reducing misunderstandings and antagonism, and fostering social harmony.

* + 1. **Project Module Design**

|  |  |
| --- | --- |
| **Module** | **Description** |
| GPT Event Analysis | Obtain a query question, call Google API to search for information, crawl search results, and return textual event analysis and mindmap code. |
| Video Information Retrieval | Get Bilibili’s BV number as input, automatically download video audio and convert it to text, while also retrieving description, title, tags, etc. |
| Video Content Summary | Summarize video content, mark important content timestamps, and assist users in extracting and organizing video information. |
| Search Keyword Summary | After converting video to text, verify factual information, organize events, and search for keywords. |
| Social Media Comment Crawling  + Analysis | Retrieve related posts and comments from major social platforms and analyze them, returning the analysis results as HTML elements. |
| Frontend Design | Use HTML, CSS, JavaScript, etc., to build a beautiful and interactive frontend page. |
| Django Backend Deployment | Combine all modules into a complete project and deploy it, responding to HTTP requests from the frontend and returning new HTML file addresses. |

Table 1.1: Project Module Design

**Chapter 2: Related Works**

* 1. **Text Embedding**

Word embedding is essential in NLP, converting text into numerical vec- tors for machine learning models. Traditional methods like one-hot encoding and word2vec advanced the field by providing dense word vectors, captur- ing semantic relationships to some extent. However, these static embeddings struggle with polysemy and contextual variations [[6](#_bookmark67)].

* + 1. **BERT: A Revolutionary Approach**

Bidirectional Encoder Representations from Transformers (BERT) revo- lutionized word embeddings by generating dynamic embeddings, considering context from both directions [[7](#_bookmark68)]. This allows BERT to capture nuanced word meanings in various contexts, improving NLP task performance.

BERT’s transformer architecture uses self-attention mechanisms for par- allel processing and deep bidirectional understanding. Pre-trained on large corpora with masked language modeling (MLM) and next sentence predic- tion (NSP), BERT learns bidirectional context and sentence relationships [[7](#_bookmark68)].

* + 1. **BERT Variants and Multilingual Capabilities**

A notable BERT variant is the bert-base-multilingual-cased model, ex- tending BERT’s capabilities to multiple languages. Pre-trained on Wikipedia pages in 104 languages, it effectively processes multilingual text.

The bert-base-multilingual-cased shares the original BERT-base archi- tecture with 12 layers, 768 hidden units, and 12 attention heads, totaling 110 million parameters. Its training data includes text from 104 languages, and it preserves case information, aiding languages with case distinctions.

Here is the comparison of BERT and its multilingual variant highlights their differences and advantages:

|  |  |  |
| --- | --- | --- |
| **Aspect** | **BERT (Original)** | **bert-base- multilingual-cased** |
| **Architecture** | 12 layers, 768 hidden units, 12 attention heads | 12 layers, 768 hidden units, 12 attention heads |
| **Pre-training Data** | English Wikipedia, BooksCorpus | Wikipedia in 104 languages |
| **Case Sensitivity** | Uncased | Cased |
| **Number of Parameters** | 110 million | 110 million |
| **Contextual Embeddings** | Yes | Yes |
| **Multilingual Support** | No | Yes |

Table 2.1: Comparison between BERT (Original) and bert-base-multilingual-cased

* + 1. **Applications and Benefits**

The bert-base-multilingual-cased model excels in multilingual NLP tasks, offering consistent performance across languages. It benefits applications like machine translation, cross-lingual information retrieval, and multilingual sentiment analysis. BERT’s dynamic embeddings capture contextual word meanings, enhancing tasks requiring deep sentence semantics understanding. The bert-base-multilingual-cased model advances NLP by combining BERT’s dynamic embeddings with multilingual processing. It is ideal for applications needing robust multilingual understanding, providing a valuable

tool for researchers and practitioners.

* + 1. **Clustering Method**

After completing the embedding vectors for the text, the next task is to classify these vectors effectively. To achieve this goal, we can employ various established methods, among which K-means clustering is widely used and popular.

K-means clustering is a well-known unsupervised learning algorithm. Its core idea is to partition the data points into K clusters, such that data points within each cluster are as similar as possible, while data points in different clusters are as dissimilar as possible. The algorithm iteratively optimizes the cluster center positions and the cluster assignments of data points to minimize the within-cluster variance.

In the clustering process, the choice of distance metric is crucial, as different metrics can significantly impact the clustering results. Common distance metrics include Euclidean distance and cosine distance.

**Euclidean Distance**

Euclidean distance measures the straight-line distance between two points and is suitable for cases where vector dimensions and measurement standards are consistent. The calculation formula is as follows:

*d*(**x***,* **y**) = ,

,uΣ*n*

*i*=1

(*xi − yi*)2

where **x** and **y** are two n-dimensional vectors, and *xi* and *yi* are the *i*-th components of vectors **x** and **y**, respectively.

**Cosine Distance**

Cosine distance measures the cosine of the angle between two vectors and is suitable for clustering data in high-dimensional spaces. The calculation formula is as follows:

*d*cos(**x***,* **y**) = 1 *−*

**x** *·* **y**

where **x** *·* **y** represents the dot product of vectors **x** and **y**, and  **x ** and



**x y**

 **y ** are the norms (i.e., lengths) of vectors **x** and **y**, respectively.

In this project, since all embedding vectors have the same dimensions and measurement standards, we choose to use Euclidean distance for cluster- ing. The calculation of Euclidean distance is straightforward and intuitive, effectively reflecting the actual spatial distance between vectors. By using Euclidean distance, we can more accurately cluster similar text vectors to- gether, thus achieving a reasonable classification of text data. This not only helps improve the accuracy of clustering but also provides a solid foundation for subsequent analysis and processing.

* 1. **LLMs**

Large models refer to machine learning models with massive param- eters and complex computational structures. These models are typically constructed using deep neural networks, consisting of billions to trillions of parameters. The design goal of large models is to enhance model expres- siveness and predictive performance, enabling them to handle more complex tasks and data. Large models find wide applications across various fields including natural language processing, computer vision, speech recognition, and recommendation systems. By training on massive datasets, large models learn intricate patterns and features, exhibiting strong generalization capa- bilities to make accurate predictions on unseen data.

* + 1. **Two Major Tasks**

Text generation tasks are crucial in natural language processing (NLP), aiming to generate coherent natural language text continuously from given in-

puts (prompts, contexts, or other forms of initial information). The primary objective is to create meaningful, grammatically correct text relevant to the given context. Applications include dialogue systems, machine translation, content creation, and story generation.

Text summarization tasks are another important NLP task, aiming to extract key information from a given long text and generate a concise sum- mary. The core goal is to preserve the main content and important details of the original text while significantly reducing its length, facilitating quick understanding and information retrieval. Applications include news sum- maries, academic paper abstracts, product review summaries, and business report summaries.

There are two types of text summarization tasks. The first is extractive summarization, which directly extracts important sentences or paragraphs from the original text and arranges them in a certain order. This method usually preserves the semantics and style of the original text. The second type is abstractive summarization, which uses language generation models to generate entirely new sentences and paragraphs based on the original content. This approach allows more flexible restructuring of information but requires ensuring the accuracy and coherence of the generated content.

* + 1. **Traditional Methods**

Traditional methods for text generation tasks include rule-based systems and statistical language models.

* + - 1. Rule-based systems rely on predefined rules and templates, which are simple but lack flexibility and creativity.
      2. Statistical language models, such as n-gram models, predict the next word based on the probability of occurrence of the previous n words. These methods are effective for short text generation but can suffer

from grammar and logic errors.

Traditional methods for text summarization tasks include extractive summarization and abstractive generative summarization.

1. Extractive summarization methods select important sentences from the original text using techniques like TF-IDF, PageRank algorithms, or other statistical features to assess sentence importance.
2. Generative summarization methods based on rules and templates often produce summaries that lack fluency and naturalness.

## Large Model Approaches

Large model approaches for text generation tasks include neural net- work models, particularly recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and Transformer models. Transformer archi- tectures introduce self-attention mechanisms that effectively capture long- distance word dependencies, significantly improving the quality of text gen- eration. Representative models include the GPT series (such as GPT-3, GPT-4).

Neural network language models leverage the powerful modeling capa- bilities of neural networks to generate language models trained on large-scale corpora. They capture complex context relationships and long-term depen- dencies, generating high-quality text. There are four main types:

* + - 1. Recurrent Neural Networks (RNN): Suitable for modeling sequential data in language with a loop structure that processes each element of the input sequence and maintains context information in hidden layer states.
      2. Long Short-Term Memory Networks (LSTM): A specialized RNN that addresses gradient vanishing and exploding problems by introducing

memory units and gating mechanisms (input gate, forget gate, output gate) to control information flow.

* + - 1. Gated Recurrent Units (GRU): A simplified version of LSTM that re- tains most advantages but with lower computational complexity, con- trolling information flow through update and reset gates.
      2. Attention Mechanism: Introducing attention allows models to dynam- ically select important context information, calculating the importance of each position in the input sequence and weighting the sum accord- ingly. This significantly improves performance, especially in handling long sequences.

Transformer Model is a neural network architecture based on attention mechanisms that overcome the limitations of RNNs and LSTMs, becoming the mainstream method for text generation tasks. Key features include self- attention mechanisms, multi-head attention, and positional encoding:

1. Self-Attention Mechanism: Calculates the relevance of each position in the input sequence to others, generating a weight matrix to capture global context information. Its strong parallel computing capability is suitable for processing large-scale data.
2. Multi-Head Attention: Uses multiple independent attention heads to capture different feature space information, then concatenates and lin- early transforms the results, enhancing the model’s representation ca- pability to simultaneously focus on different context features.
3. Positional Encoding: Since Transformer models lack inherent sequence information, positional encoding explicitly introduces positional infor- mation. It preserves sequence order information, enhancing the model’s ability to process sequence data.

Large model approaches for text summarization tasks include extractive and abstractive generative summarization methods:

1. Extractive Summarization: Selects important sentences, phrases, or words directly from the original text to generate summaries without creating new text. Techniques include:
   1. Feature-based methods: Using specific text features to identify and select key sentences or phrases.
   2. Graph-based methods: Representing text as a graph, where nodes represent sentences or phrases and edges represent their relation- ships.
   3. Machine learning methods: Using supervised or unsupervised learn- ing techniques to predict which parts of the text are most suitable for summarization.
2. Abstractive Generative Summarization: Involves generating new text to express summaries rather than extracting content directly from the original text. This method relies on deep learning techniques and large models, such as Transformer architectures:
   1. Seq2Seq models: Use encoder-decoder structures to encode input text into fixed-length vector representations and then decode to generate summary text.
   2. Transformer models: Utilize self-attention and multi-head attention mechanisms to capture long-term dependencies and global informa- tion in text. They can be fine-tuned after pre-training to achieve higher-quality abstract summaries.
   3. Pre-trained language models: Such as the GPT series and BERT se- ries, learn language representations through large-scale self-supervised

pre-training and can be further adjusted through fine-tuning for specific summarization requirements.

## Principles of Large Models

The basic technological principles of large models include deep neural networks, activation functions, loss functions, optimization algorithms, reg- ularization, and model structures:

* + - 1. **Deep Neural Networks (DNN)**: Models composed of multiple layers of neurons mapping inputs to outputs through weight matrices. Depth indicates the number of layers, enabling the network to learn complex nonlinear relationships and features.
      2. **Activation Functions**: Introduce nonlinear transformations allowing neural networks to learn and represent complex nonlinear mapping re- lationships. Common functions include sigmoid, tanh, and ReLU (Rec- tified Linear Unit), each with different performances and advantages in various scenarios.
      3. **Loss Functions**: Measure the difference between model predictions and true labels. Optimization algorithms minimize the loss function dur- ing training to adjust model parameters for more accurate output pre- dictions. Common loss functions include Mean Squared Error (MSE), Cross-Entropy Loss, chosen based on task requirements.
      4. **Optimization Algorithms**: Update model parameters to reduce the value of the loss function. Common algorithms include Gradient De- scent, Stochastic Gradient Descent (SGD), Adam optimizer, computing gradients of the loss function and updating parameters in the opposite direction.
      5. **Regularization**: Techniques prevent overfitting to training data. Meth- ods like L1 regularization, L2 regularization add penalty terms to con- strain model parameter size, enhancing generalization ability.
      6. **Model Structures**: Refers to the arrangement and connection of lay- ers in neural networks. Common structures include Fully Connected Networks (FCNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and more complex variants like Transformers. Choice depends on task characteristics and data structure.

These basic principles together form the core of large models. By con- tinuously optimizing and adjusting these technologies, deep learning models adapted for different complex tasks achieve outstanding performance across various application domains.

## Common Large Models Comparison

Currently, internationally recognized large models have various types, each with different strengths and characteristics in their respective fields and tasks:

? **GPT (Generative Pre-trained Transformer)**: Uses Transformer architecture, focused on generative tasks. Generates text through au- toregressive methods, such as dialogue generation, article summariza- tion, etc. GPT series models have made significant advances in genera- tive and language understanding capabilities.

? **T5 (Text-To-Text Transfer Transformer)**: Introduces a ”text-to- text” framework, unifying multiple NLP tasks’ forms. Achieves multi- task learning and transfer learning through standardized task descrip- tions and pre-training methods. Performs well in various NLP tasks like translation, text classification, etc.

? **BERT (Bidirectional Encoder Representations from Transform- ers)**: Uses Transformer architecture, enhancing context understanding

with bidirectional encoders. Suitable for various downstream tasks of pre-trained models, such as question answering, sentiment analysis, etc. BERT is considered a breakthrough in the NLP field.



Figure 2.1: Comparison Table for Comprehensive Capability Evaluation of Large Models

For our project, we selected the GPT-4o model due to its strong capa- bilities in logical reasoning, powerful information extraction and inference, efficient text analysis, sentiment analysis, and stance classification.

## Prompt Engineering

Prompt Engineering is a technique aimed at designing and optimizing input prompts to guide large-scale language models (such as GPT-3, GPT- 4) to generate desired outputs effectively. In natural language processing (NLP), this technique plays a pivotal role, particularly in scenarios with limited training data or specific task requirements. Prompts are textual inputs provided to language models to instruct them on generating specific types of outputs.

The goal of Prompt Engineering is to enhance output quality and ac- curacy by refining the prompts. It finds broad applications across various domains including text generation, question answering systems, text sum- marization, translation, and code generation.

The key strategies for effective prompt engineering are as followings:

* + - 1. **Clear Task Definition**: Prompts should clearly and specifically de- scribe the task. For instance, instructing the model to generate a sum- mary on a given topic.
      2. **Contextual Information**: Providing adequate contextual information within prompts helps the model better understand the task require- ments. For example, offering background information before posing questions.
      3. **Example-Based Prompts**: Using examples or templates as prompts aids the model in producing outputs similar to provided examples. For instance, presenting several question-answer pairs and expecting the model to generate similar responses.
      4. **Structured Formats**: Using structured formats such as lists or headers assists in generating well-organized outputs. For example, instructing the model to generate a list containing multiple items.
      5. **Iterative Optimization**: Continuous refinement of prompts through iterative testing and adjustments based on model output observations. This iterative process helps in optimizing prompts to achieve optimal results across various tasks and models.

Prompt Engineering leverages language models to generate high-quality outputs by guiding them effectively through well-designed prompts. By refin- ing and optimizing prompts, users can steer models to produce text content

that meets specific requirements. In practice, ongoing testing and optimiza- tion of prompts are crucial to adapt to different task demands and model capabilities.

# Kity Minder

KityMinder is a sophisticated online mind mapping tool developed by Baidu’s FEX team, designed with developers in mind. Built on the powerful Kity vector graphics library, KityMinder offers extensive capabilities for data visualization and basic editing through its core module, kityminder-core.

KityMinder-core provides a robust foundation for developing custom mind mapping solutions. It supports various formats for data import and export, including JSON, text, Markdown, SVG, and PNG, making it highly versatile for integration into different applications. This flexibility allows developers to tailor the tool to specific needs, whether for personal projects or enterprise-level applications.

The KityMinder Editor enhances the core’s functionality by incorporat- ing AngularJS, providing a rich user interface and efficient input mechanisms such as hotbox components. This setup facilitates rapid development and a seamless user experience, making it easier to implement complex features and improve overall usability.

Developers can leverage KityMinder’s comprehensive API and detailed

documentation to integrate and extend its features effortlessly. The tool supports major browsers, including Chrome, Firefox, Safari, and IE 10+, en- suring broad compatibility and a consistent experience across different plat- forms.

For those looking to contribute or customize further, KityMinder’s

open-source nature encourages community involvement and collaboration. The project is hosted on GitHub, where developers can fork the repository,

contribute to its development, and share improvements with the broader community.

# Sentiment Analysis

## Introduction to Sentiment Analysis

Text sentiment recognition, also known as sentiment analysis, is a sub- field of Natural Language Processing (NLP) that aims to identify and extract subjective information from text data, determining whether the expressed opinions are positive, negative, or neutral. This task is crucial for various applications, such as analyzing customer feedback, monitoring social media, and enhancing decision-making processes for businesses and governments.

Sentiment analysis involves multiple steps, including data preprocessing, feature extraction, and classification, to handle the unstructured nature of text data and the complexity of human language [[8](#_bookmark69)][[9](#_bookmark70)]. One of the primary challenges in sentiment analysis is accurately interpreting the context and nuances of language, such as sarcasm, negation, and domain-specific termi- nology, which can significantly alter the sentiment conveyed [[9](#_bookmark70)].

Techniques for sentiment analysis can be broadly categorized into ma- chine learning approaches, which rely on labeled datasets and algorithms to learn patterns, and lexicon-based approaches, which use predefined dictio- naries of sentiment-laden words [[8](#_bookmark69)]. Recent advancements have also seen the integration of deep learning methods to improve the performance and accuracy of sentiment analysis systems [[9](#_bookmark70)].

Despite these advancements, there remain significant challenges, such as handling low-resource languages and code-mixed data, which require ongoing research and innovation to address effectively [[8](#_bookmark69)][[9](#_bookmark70)].

## Multilingual Sentiment Analysis Model

The bert-base-multilingual-uncased-sentiment model by NLP Town is a fine-tuned BERT model for sentiment analysis on product reviews in six languages: English, Dutch, German, French, Spanish, and Italian. It predicts the sentiment of a review on a scale from 1 to 5 stars. The model has been trained on a substantial dataset of product reviews and has demonstrated high accuracy, particularly in distinguishing reviews with star ratings within one star of the human reviewer.

**Chapter 3: Architecture Design**

In the development of any large-scale system, the overall architecture serves as its backbone, determining the system’s functionality, stability, per- formance, and ease of maintenance. A well-designed architecture not only ensures efficient operation but also saves significant time and effort in future expansions and maintenance.

Cocoon Breaker is a complex web application with a Python-based back- end. Its design aims to provide users with a concise and aesthetically pleasing user interface, while also ensuring that developers find the codebase easy to read, maintain, and extend through convenient function libraries.

? **User Interface**: The user interface of Cocoon Breaker is designed to be simple and intuitive, aiming to deliver an excellent user experience. Navigation and functional operations are straightforward, minimizing the learning curve and operational complexity for users.

? **For Developer**: Django is a high-level Python web framework known for its “batteries-included” philosophy, offering features like authenti- cation, admin interface, template engine, ORM, and more out of the box. This simplifies web application development significantly, align- ing with Django’s principles of “rapid development” and “DRY (Don’t Repeat Yourself).” In contrast, Flask is a lightweight framework that provides flexibility, requiring developers to integrate various extensions themselves, making it ideal for small projects or highly customizable ap- plications. Ruby on Rails, a popular Ruby framework, emphasizes rapid

development and convention over configuration. Laravel, a leading PHP framework, is praised for its elegant syntax and robust features, though it relies on PHP-specific characteristics. Django’s strengths include a comprehensive set of built-in features, reducing developer workload, and a large, active community offering extensive third-party plugins and documentation. Django also excels in security, with built-in protections against common vulnerabilities like SQL injection, XSS, and CSRF. Python’s advantages in backend development are clear. The language is easy to use and highly readable, making development more efficient. Python’s simple syntax allows developers to focus on problem-solving rather than complex code. Additionally, Python’s rich libraries and frameworks cater to various needs, from web development to data anal- ysis and AI. With modular design and extensive third-party support, Python applications can easily expand their functionalities, including user-friendly machine learning frameworks.

Given the complexity and interdependence of Cocoon Breaker’s func- tionalities, we implemented an asynchronous design to enhance system re- sponsiveness and user experience. This approach not only ensures the cor- rectness of operations but also significantly reduces user wait times. For instance, while handling user requests, the backend can perform multiple operations concurrently rather than sequentially. This parallel processing greatly improves system performance.

To ensure data safety and stability when multiple users access the system simultaneously, Cocoon Breaker uses the browser’s session ID to achieve user data isolation. This design effectively prevents data and processing conflicts, ensuring stable operation even under high concurrency conditions.

Cocoon Breaker is deployed on a server and has undergone rigorous testing and verification to ensure stable and efficient operation in real-world

usage. Through a series of validation steps, we confirmed that all system functionalities work correctly and meet user requirements.

# Overall Architecture

Our program leverages a diverse array of technologies that seamlessly integrate to create a robust and comprehensive workflow, each playing a crucial role in ensuring the system’s overall efficiency and effectiveness. The table below provides a detailed summary of these technologies and their specific applications within our workflow:

|  |  |  |
| --- | --- | --- |
| **Task** | **Technology** | **Usage** |
| **Backend** | Django | Local/Server Deployment |
| **Online Search** | Google API | API Service |
| **Video Info Scraping** | Bilibili API | API Service |
| **Web Scraping** | BeautifulSoup | – |
| **Large Language Model (LLM)** | GPT-4o | API Service |
| **Sentiment Analysis** | bert-base-multilingual- uncased-sentiment | Local Deployment |
| **Embedding** | bert-base-multilingual- cased | Local Deployment |

Table 3.1: Summary of Technologies Used in Our Program

Specifically, we use Django as our backend framework. Django’s ef- ficiency and flexibility enable rapid development and deployment of appli- cations. We deploy Django both locally and on servers to ensure system stability and scalability. For online search functionality, we use the Google API. The Google API provides powerful search capabilities and rich data sources; by invoking its API service, we can quickly obtain the necessary information and integrate it into our workflow.

To obtain video information, we use the Bilibili API. By calling the API

service provided by Bilibili, we can retrieve and process a large amount of video data. This is crucial for analyzing video content and extracting key information. For web scraping, we chose BeautifulSoup, a simple and easy- to-use Python library suitable for handling HTML and XML files. Beauti- fulSoup allows us to parse web content and extract valuable information.

We have integrated GPT-4o as the large language model (LLM) in our system. By invoking its API service, we can perform natural language pro- cessing tasks such as text generation and question answering. This makes our system perform exceptionally well in handling complex texts. For sen- timent analysis, we use the bert-base-multilingual-uncased-sentiment model. This model can identify sentiments in multiple languages and is deployed locally to ensure efficient and secure analysis. For embedding tasks, we use the bert-base-multilingual-cased model. This model generates high-quality embeddings for texts in multiple languages and can quickly process large amounts of text data when deployed locally.

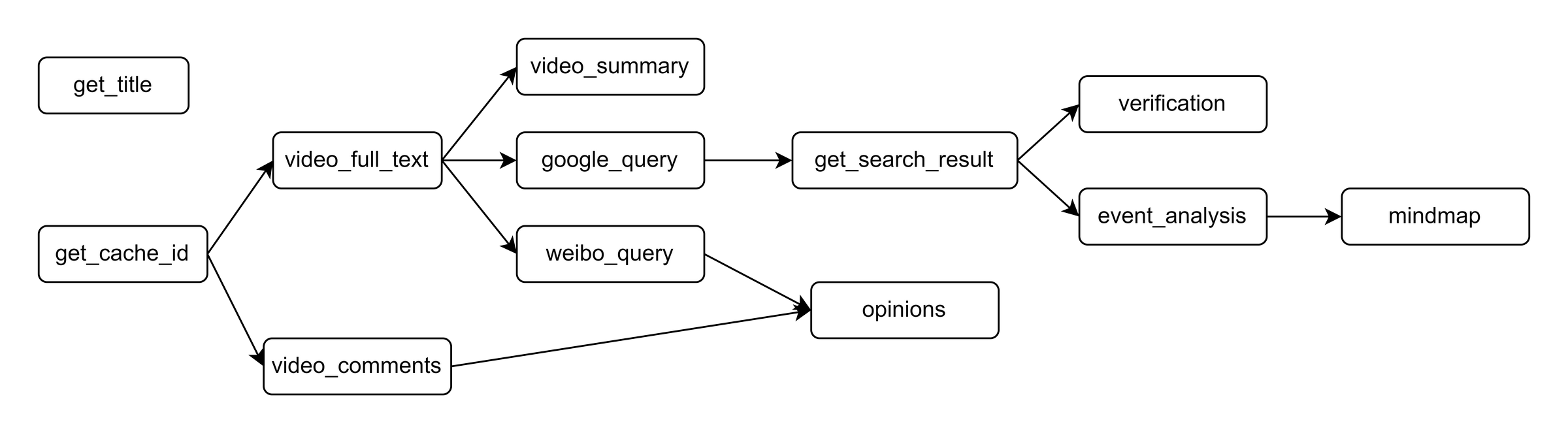


Figure 3.1: Directed Acyclic Graph (DAG) of Task Modules

Additionally, in our program design, we particularly focus on using asyn- chronous processing methods. By organizing the various subtasks into a directed acyclic graph (DAG), as shown in Figure [3.1](#_bookmark31), we can clearly ana- lyze their dependencies. In practice, for example in online search, we use asynchronous methods to concurrently scrape a large number of web pages, significantly improving efficiency. Asynchronous processing also reduces sys-

tem response time, making the entire workflow more smooth and efficient.

# Web Plugin

**Bilibili Analyzer** is a browser extension designed specifically for Google Chrome-based browsers. It aims to help users analyze the Bilibili video they are currently viewing and automatically navigate to a related report page. Once installed and enabled, users will see a **Bilibili Analyzer** icon in the browser toolbar. Clicking this icon will open a small window with a promi- nent “Start Analysis” button. This simple and clear design ensures a conve- nient and efficient user experience.

Figure 3.2: Screenshot of the plugin’s user interface

When the user clicks the “Start Analysis” button, the extension first calls the chrome.tabs.query API to get the URL of the current active tab and uses a regular expression to validate that URL, ensuring it matches the format of a Bilibili video page. Specifically, the regular expression matches the URL’s structure to identify a legitimate video page. If the current page is a valid Bilibili video page, the extension extracts the video’s BV number, which is the unique identifier for Bilibili videos. The BV number is a crucial parameter used to uniquely identify each video on the Bilibili platform. By extracting the BV number, the extension can accurately locate the specific video content.

Next, the extension constructs a new report page URL, which includes the extracted BV number and points to a pre-configured analysis report server address. This server address is used to generate and display a detailed

analysis report of the video. The extension then uses the chrome.tabs.create API to open this report page in a new tab, allowing the user to directly view the detailed analysis report of the video.

If the extension detects that the current page is not a valid Bilibili video page, it will pop up a message box to inform the user that the current page is invalid and suggest using the extension on a proper Bilibili video page. This design not only enhances the user experience but also ensures that the analysis function is only run on appropriate pages, avoiding unnecessary operations and confusion.

The development of the **Bilibili Analyzer** extension follows the Chrome Extension **Manifest V3** specification. **Manifest V3** provides a clear struc- ture and security guarantees for the functionality and permissions of exten- sions. **Manifest V3** introduces a series of new features and improvements aimed at enhancing the security, privacy protection, and performance of ex- tensions. The extension primarily relies on the chrome.tabs.query API to

get the URL of the current active tab and uses the chrome.tabs.create

API to open the report page in a new tab. Through these APIs, the exten- sion achieves efficient management and operation of browser tabs, ensuring a smooth user experience during use.

# Frontend and Backend Design

## Frontend Design

The frontend interface serves as the bridge for user interaction with the application, directly impacting user experience, brand image, and the success of the application. To enhance the usability of our project, we have designed the frontend interface with a focus on visual appeal and ease of use.



Figure 3.3: Application Main Interface1

The background image features a high-tech, predominantly blue and black color scheme. Blinking blue lights symbolize the rapid transmission of information, suggesting breaking through information barriers, aligning with the theme of “Info Cocon Breaker.” The project’s theme is prominently displayed in the center with the largest font size. An input box allows users to paste the link of the video they wish to analyze. Above the input box, the prompt “Currently revising video:” and the text in the input box “Please copy the video link here” guide users on the required action. Below the input box is a “Start Analysis” button, which turns yellow when hovered over, indicating its clickable nature.



Figure 3.4: Application Main Interface2

Upon clicking the “Start Analysis” button, it disappears, signaling to users that a video is currently being analyzed and preventing them from initi- ating another analysis. Below the input box, a progress bar and the message “Analysis in progress...” inform users about the ongoing system activity. The progress bar displays the current status of the video analysis, providing real-time feedback on the progress of the task, reducing user anxiety and uncertainty, and keeping them informed of the system’s operations. Once the progress bar completes loading, the page redirects to the analysis results interface.

Figure 3.5: Application Main Interface3

At the top right of the page is the project’s name, clickable to return to the main interface. Positioned slightly below on the left is the title of the currently analyzed video, clickable to return to the original video inter- face. The analysis results page comprises four modules: “Video Summary,” “Fact Finding,” “Timeline Overview,” and “Multiple Perspectives.” Each module is divided into two sections: the left section features an index direc- tory, while the right section displays detailed content for each module of the analysis results. Structured analysis results transmitted from the backend to the frontend automatically generate an index displayed on the left, distin- guishing headings with font sizes and bold text to maintain clarity and visual appeal, making it easier to extract key points from lengthy text. Clicking on the title in the left index section automatically navigates to the correspond- ing module’s display on the right. The “Timeline Overview” and “Multiple Perspectives” modules also feature relevant result images. The entire in- terface is clean, elegant, with clear functionality, and aesthetically pleasing, demonstrating thoughtful design.



Figure 3.6: interface of video summary

## Backend Design

Our backend system is implemented using the Django framework, which has been extensively detailed in the previous sections outlining its many ad- vantages. The backend design adheres to two key principles: asynchronous operation and modularity. Modularity forms the foundation of the entire sys- tem design. To achieve this principle, we have divided the backend system into 11 basic modules based on the characteristics of the tasks and practical considerations. These modules are functionally independent, with each mod- ule implemented by a separate class, ensuring high cohesion and low coupling of the code.

Additionally, these modules have sequential dependencies in data pro- cessing. Therefore, we have organized the tasks based on figure [3.1](#_bookmark31) and determined an asynchronous design scheme. Data transmission is handled through Django’s built-in caching mechanism, with session IDs used to en- sure isolation between different users’ data. The asynchronous design not only improves the system’s execution efficiency but also significantly reduces resource usage and response time, thereby enhancing the user experience.

During the collaborative development process, each class was developed and tested independently, and the workflow functioned correctly upon the final merge. This design approach allows for the addition of new functional modules or the optimization and expansion of existing ones without affect- ing the overall system, thus ensuring the long-term stable development of the system. In summary, the principles of asynchronous operation and mod- ularity complement each other, making our backend system not only highly performant but also exceptionally extensible and flexible.

Below is an introduction to these 11 modules, with more detailed de- scriptions provided in the subsequent sections:

? **GoogleQuery**: The GoogleQuery class is designed to read text from

the cache and generate a query suitable for Google search using Ope- nAI’s GPT model. The input to this class is a cache key, and the out- put is the generated Google query terms. By reading the complete text stored in the cache, the class uses the GPT model to extract keywords from the text and stores the resulting query back into the cache.

? **EventVeri**: The EventVeri class is used to verify the authenticity of events. The input is a cache key, and the output is a factual analysis report of the event. This class reads the full text and search results from the cache and uses OpenAI’s GPT model to generate a detailed factual analysis report, which is then returned as the output.

? **EventAnalyzer**: The EventAnalyzer class is responsible for analyzing events and generating an HTML-formatted analysis report. The input to this class is a cache key, and the output is an HTML-formatted event analysis report. By reading the complete text and search results from the cache, the class uses the GPT model to generate a detailed event analysis report, which is then stored back into the cache.

? **AnalysisMindmap**: The AnalysisMindmap class converts event anal- ysis reports into mind map format. The input is a cache key, and the output is the HTML content of the mind map. This class reads the event analysis report from the cache and uses OpenAI’s GPT model to generate a Markdown-formatted mind map, which is then converted to HTML format and returned.

? **WeiboQuery**: The WeiboQuery class is designed to generate concise queries suitable for Weibo hot searches. The input is a cache key, and the output is the query terms formatted for Weibo. By reading the complete text from the cache, the class uses the GPT model to generate concise Weibo hot search queries and stores the resulting queries back

into the cache.

? **VideoSummary**: The VideoSummary class processes video text data. The input is a cache key, and the output is an HTML-formatted sum- mary of the video content. By using the cache key, this class reads the full text of the video from the cache and employs OpenAI’s GPT model to generate a detailed summary of the video content. The summarized content is presented in a clear HTML format for easy viewing and use.

? **VideoInfo**: The VideoInfo class retrieves information about Bilibili videos. The input parameters are bv (the BV number of the Bilibili video) and cache key, and the output is the video’s title. This class uses the BV number to request video page data from Bilibili’s API, extract- ing the video’s title, description, and tags, and downloading the audio file. The audio file is then converted to WAV format for subsequent processing and analysis.

? **VideoFullText**: The VideoFullText class is designed to transcribe the audio file of a video into complete text. The inputs to this class are bv (the BV number of a Bilibili video) and cache key, and the output is the full text of the video. By invoking the iFlytek API, the class uploads and merges audio file segments to obtain the transcribed text result, which is then stored in the cache.

? **VideoComments**: The VideoComments class is used to retrieve com- ments and danmu (bullet screen comments) data from Bilibili videos. The inputs are bv (the BV number of a Bilibili video) and cache key, and the output is a CSV file containing the comments and danmu data. By using the BV number, the class fetches the video’s CID from Bili- bili’s API, then requests the danmu XML data and comments JSON data. After parsing, the data is stored as a CSV file and the result is

saved to the cache.

? **SearchResult**: The SearchResult class performs Google searches and retrieves the page content of the search results. The input is cache key, and the output is the text content of the search results. By reading the Google search query terms from the cache using the cache key, the class utilizes the Google Search API to obtain search results. It then uses multithreading to concurrently request the content of each search result page, extracts the text content, aggregates it, and stores the result back into the cache.

? **VideoAnalyzer**: The VideoAnalyzer class analyzes video content and generates an HTML-formatted report. The input is cache key, and the output is an HTML-formatted video analysis report. By reading the complete text of the video and the search results from the cache, the class uses OpenAI’s GPT model to generate a detailed video analysis report, which is then stored in the cache.

# Video Processing

## Video Information Retrieval

To obtain video title information from Bilibili, we need to access the webpage or API through request headers, simulating real user behavior to avoid being identified as bots or crawlers by the server, thereby acquiring more accurate and comprehensive data.

**Request Headers**

* + - 1. **User-Agent**

? **Function:** The User-Agent header tells the server information about the client’s software (e.g., browser type, operating system,

etc.).

? **Reason:** Many websites determine whether a request is from a browser or a crawler based on the User-Agent header. If the User- Agent is not set correctly, the server might return different content or block access altogether.

* + - 1. **Referer**

? **Function:** The Referer header contains the URL of the page that initiated the current request.

? **Reason:** Some websites check the Referer header to ensure the request comes from a page within the site, rather than directly through an API or script. Setting the Referer header can increase the credibility of the request and reduce the likelihood of being denied.

* + - 1. **Cookie**

? **Function:** The Cookie header contains the user’s session informa- tion and other state information.

? **Reason:** Many websites use cookies to store the user’s login status, preferences, etc. By setting the correct cookies, we can simulate a logged-in state, allowing access to user-specific data or restricted content.

**Retrieve a Single Video Title**

To retrieve the video title, first construct the video page URL using the BV number with the code:

[url=f’https://www.bilibili.com/video/{self.bv}’](url%20=%20f'https://www.bilibili.com/video/%7Bself.bv%7D%27)

Next, send an HTTP GET request to this URL using:

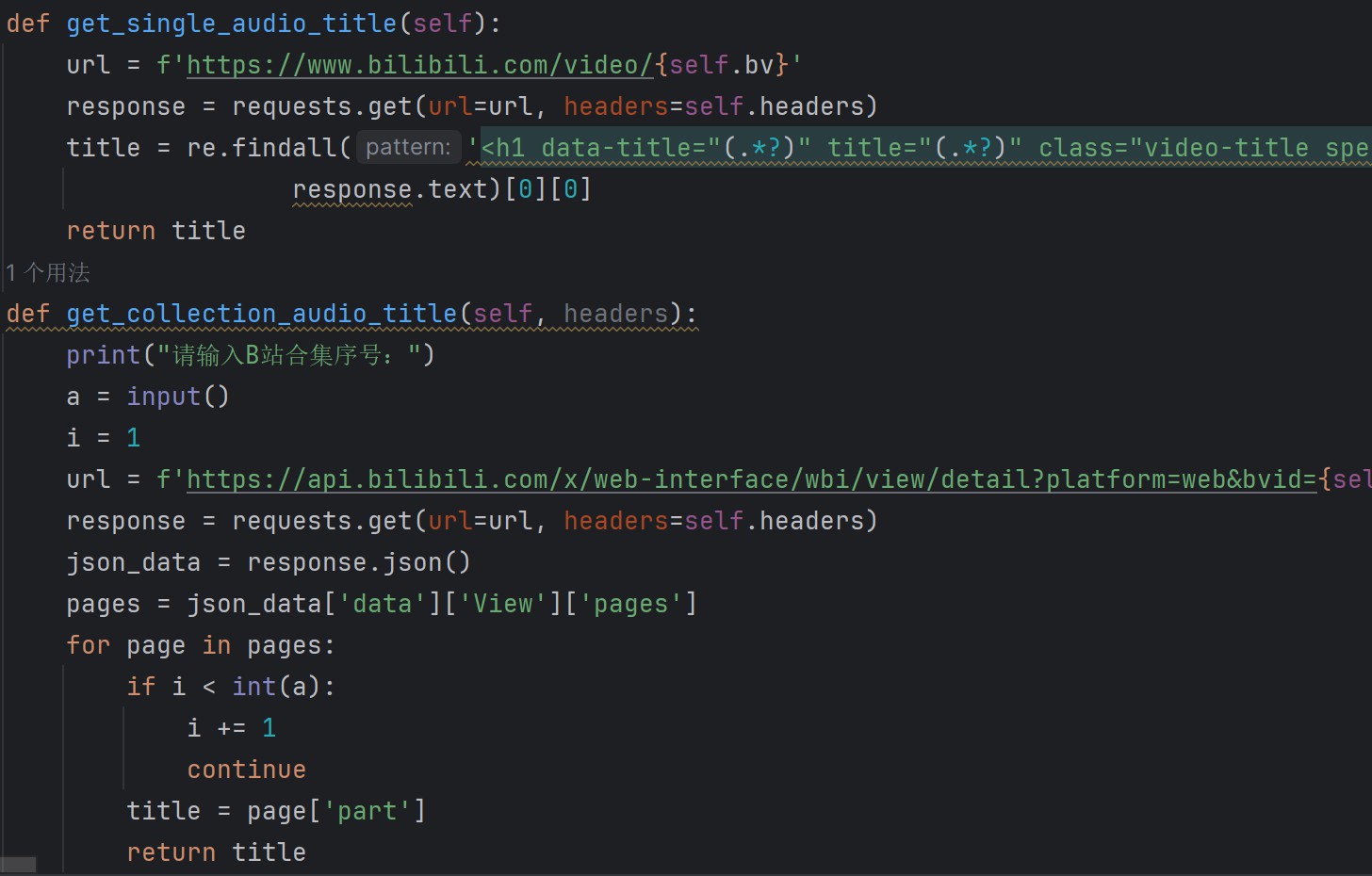


Figure 3.7: Core code of retrieving the title from video

[response=requests.get(url=url,headers=self.headers)](response%20%3D%20requests.get(url%3Durl%2C%20headers%3Dself.headers))

Then, parse the HTML response to extract the video title by using the regular expression:

<h1 data-title="(.\*)" title="(.\*)"

class="video-title special-text-indent" data-v-5120f6b9>

The extracted title can be obtained with:

title=re.findall(’<h1data-title="(.\*)"title="(.\*)"class= "video-titlespecial-text-indent"data-v-5120f6b9>’,response. text)[0][0]

**Retrieve Titles of Videos in a Collection**

To scrape a specific video from the collection, start by prompting the user to enter the index of the video they want to scrape. Then, construct the API request URL based on the BVID and other necessary parameters. Next, send an HTTP GET request to the constructed URL. Upon receiving the response, use:

<response.json()>

to parse the response content into a JSON object. Finally, extract the video title from the JSON object with the line:

title=response.json()[’data’][’View’][’pages’][int(a)-1] [’part’]

## Video Text Extraction

The main functionality of the code includes downloading the audio file of a Bilibili video, using iFlytek Spark to convert the audio to text, and saving the result as a .txt file. First, the video title is obtained and cleaned of any illegal characters, similar to the code for obtaining video title information mentioned earlier. This process includes the cleaning of illegal characters.

**Retrieve the Audio File of a Single Bilibili Video**

To get the audio file of a single Bilibili video, start by calling the get single audio title() method to get the cleaned video title. Construct the video page URL and use the requests library to send a GET request to retrieve the page content. Use the regular expression re.findall() to extract video play information from the page and parse the extracted JSON data into a dictionary format (json data). Extract the audio URL from json data, typically found in the dash field’s audio array’s first element’s baseUrl field. Next, construct the target file path (target path), usually saved in the video folder, with the file name using the video title. Use the convert audio to wav() method to download the audio file, convert it to

.wav format, and save it to target path. If the video folder does not exist,

create it before saving the file.

**Retrieve Audio Files of a Bilibili Video Collection**

To retrieve audio files from a Bilibili video collection, start by getting the video title and CID using the get collection audio title() method, which returns the cleaned video title (cleaned title), video CID (cid), and some other unused variables.

Next, construct the API request URL using the video’s BV number and the retrieved CID. This URL is typically used to obtain detailed playback information of the video. For example, the URL might look like

[https://api.bilibili.com/x/player/wbi/playurl?avid=571565015&](https://api.bilibili.com/x/player/wbi/playurl?avid=571565015&bvid=\%7Bself.bv\%7D&cid=\%7Bcid\%7D) [bvid=\{self.bv\}&cid=\{cid\}](https://api.bilibili.com/x/player/wbi/playurl?avid=571565015&bvid=\%7Bself.bv\%7D&cid=\%7Bcid\%7D)

After that, extract the audio URL from link data, usually found in the dash field’s audio array’s first element’s baseUrl field

audio\_url=link\_data[’data’][’dash’][’audio’][0][’baseUrl’

]

Finally, construct the target file path (target path), typically saved in the video folder with the file name derived from the video title. Ensure the video folder exists by using os.makedirs(’video’, exist ok=True). Call the convert audio to wav() method to download the audio file, convert it to .wav format, and save it to target path. If the video folder does not exist, create it before saving the file.

**Preparing for Audio-to-Text Conversion**

The code defines various API interfaces and file chunk sizes, setting necessary parameters such as application ID and key during initialization. To generate parameters needed for sending requests to the iFlytek Spark API, the code performs several steps:



Figure 3.8: Core code of retrieving audio from video

* + - 1. **Generate Timestamp:** A timestamp is generated to ensure the re- quest is current.
      2. **Generate MD5 Signature:** An MD5 signature is created for data integrity.
      3. **Generate HMAC Signature:** An HMAC signature is produced for secure authentication.

Next, the code retrieves the file length and file name. Based on different APIs, it generates the corresponding parameter dictionary for the following interfaces:

? Preparation Interface

? Upload Interface

? Merge Interface

? Progress and Result Interface

**Handling HTTP POST Requests**

To send an HTTP POST request to the specified iFlytek Spark API and handle the response, the code performs the following steps: it sends the request, handles the response, and checks the response result to ensure the request was successful.

**Uploading Audio Files**

To upload an audio file to iFlytek’s Speech Recognition Service, the file is divided into chunks to improve success rate and efficiency, with each chunk sized at 100MB. The process involves opening the audio file in binary mode, reading the file content by the specified chunk size, obtaining the ID of the next chunk, and calling the API interface to upload each chunk until the entire file is uploaded.

**Merge Audio Files**

Merge all uploaded chunks into a complete audio file on the server by calling the API interface to notify the server to merge all chunks into one file, which is essential for subsequent speech recognition processes.

**Get Conversion Progress and Result**

Regularly query the server for task progress until the recognition task is complete. Periodically call the progress query interface to check the task status. Once the task is complete, call the result retrieval interface to obtain the recognition result and save it as a text file.

## Video Comment and Danmu Crawling

The main functionality of the code includes initializing functions to ob- tain Bilibili video’s bullet comment (danmaku) data and comment data. It

then saves this data to a CSV file and caches the retrieved information for later use.

**Obtain Danmaku XML Data**

To obtain the danmaku XML data, the process begins by fetching the video’s cid through the Bilibili API using the BV number. This is achieved by constructing a request URL formatted with the cid obtained from the get cid from bv() method. Following this, an HTTP GET request is sent to retrieve the data. The response encoding is set to UTF-8 to ensure accu- rate handling of Chinese and special characters. Upon a successful request, the response text, which contains the danmaku XML data, is returned.

**Parse Danmaku XML Data**

First, parse the XML string into an XML tree to facilitate structured access to the data. Next, traverse all danmaku nodes within the XML tree to systematically extract the send time, sender ID, and danmaku content. Each extracted danmaku entry is formatted appropriately and then added to a list for further processing or storage. This systematic extraction ensures that all relevant danmaku data is captured efficiently and accurately for subsequent analysis or usage.

**Entry Point for Obtaining Danmaku Data**

The entry point for obtaining danmaku data involves calling the methods of get cid from bv(), get danmaku xml(), and parse danmaku xml() sequentially to retrieve the video’s CID, fetch the danmaku XML data, and parse it to extract relevant danmaku information. Subsequently, the parsed danmaku data is saved into a CSV file and cached for future use. Relevant code as follows:



**Obtain the Video’s OID**

First, we need to get the video’s OID. This is done by requesting the Bilibili API using the BV number to get the video’s AV number (OID).

**Obtain Comment Data for a Specified Bilibili Video**

To obtain comment for a specified Bilibili video, call get oid from bv method to get the video’s OID. Then, construct the API request URL and send the request.

Initialize a comment data list and loop to get all comment data.

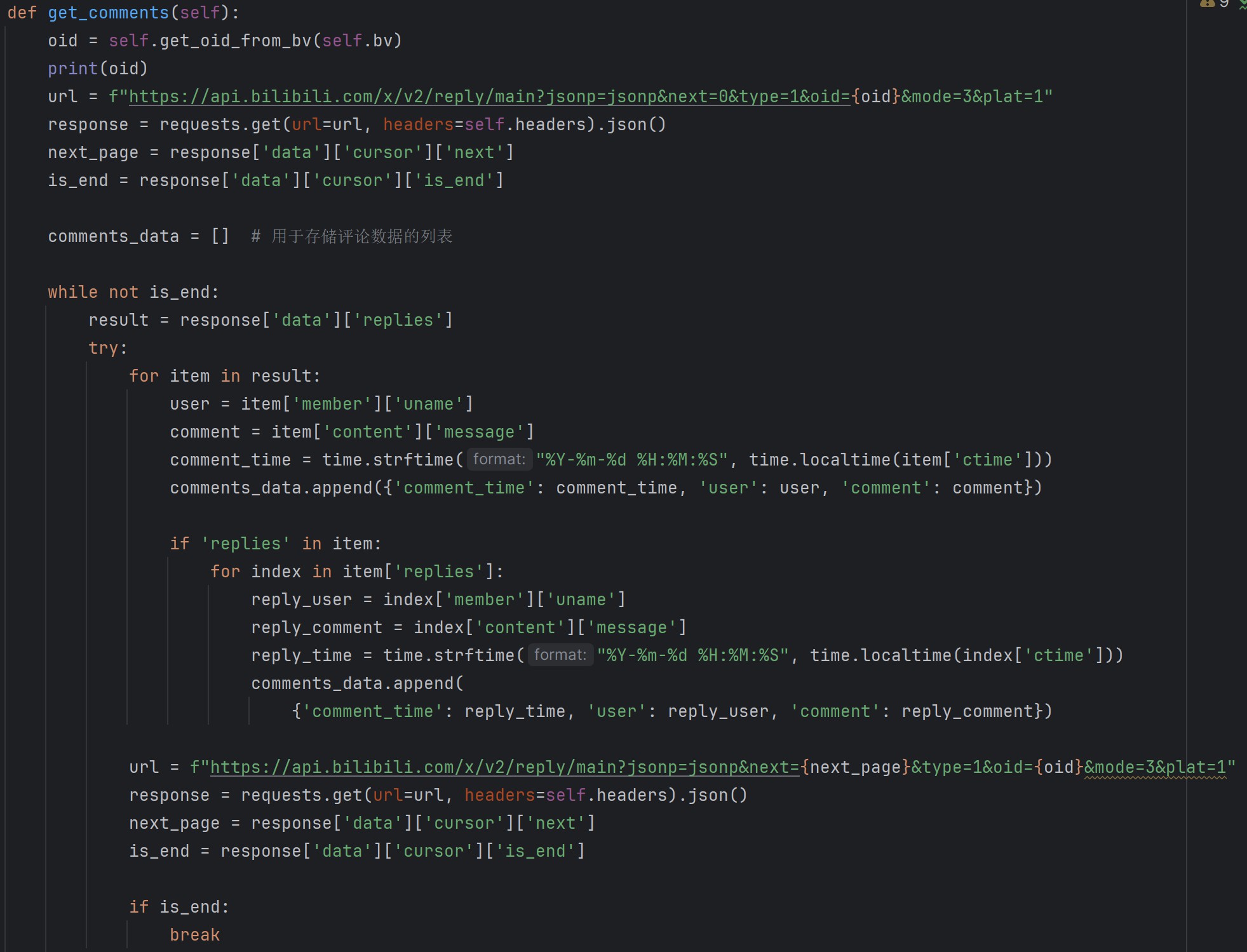
? In each loop iteration: Extract the comment list result from the current page’s data.

? Use a try block to iterate through each comment item in the com- ment list, extract the comment content and time, and store them in the comments data list.

? If a comment item contains replies, iterate through the reply list, extract the reply content and time, and store them in the comments data list.

? Construct the API request URL for the next page, send the request, and update next page and is end.

? If a TypeError exception occurs during processing, print the error mes- sage and stop the loop.

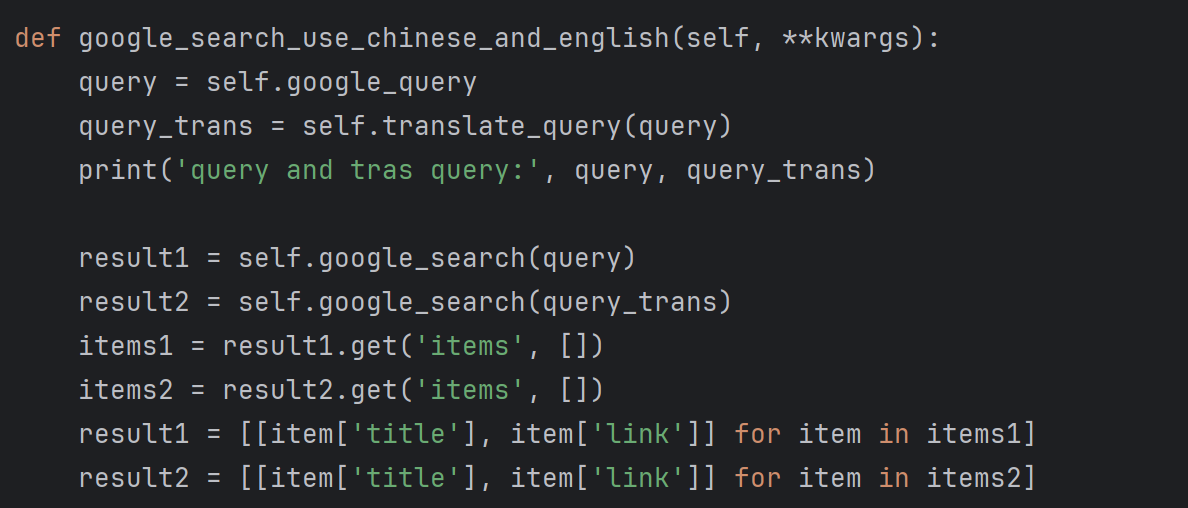
Finally, save the comment data as a CSV file and cache it.Relevant code as follows:

# Online Search

In our project, we have developed a Python class named SearchResult, which integrates multiple technologies and open-source libraries to perform bilingual Google searches, text translation, and web content extraction. This class achieves the following core functionalities:

**Bilingual Google Search Functionality**

Firstly, we utilize the Google Custom Search API to implement bilingual search capabilities. Below is the relevant code snippet and explanation:



This code demonstrates how the method is invoked to handle simulta- neous Chinese and English queries, utilizing the google search method to retrieve search results and merge them.

**AI Translation Support**

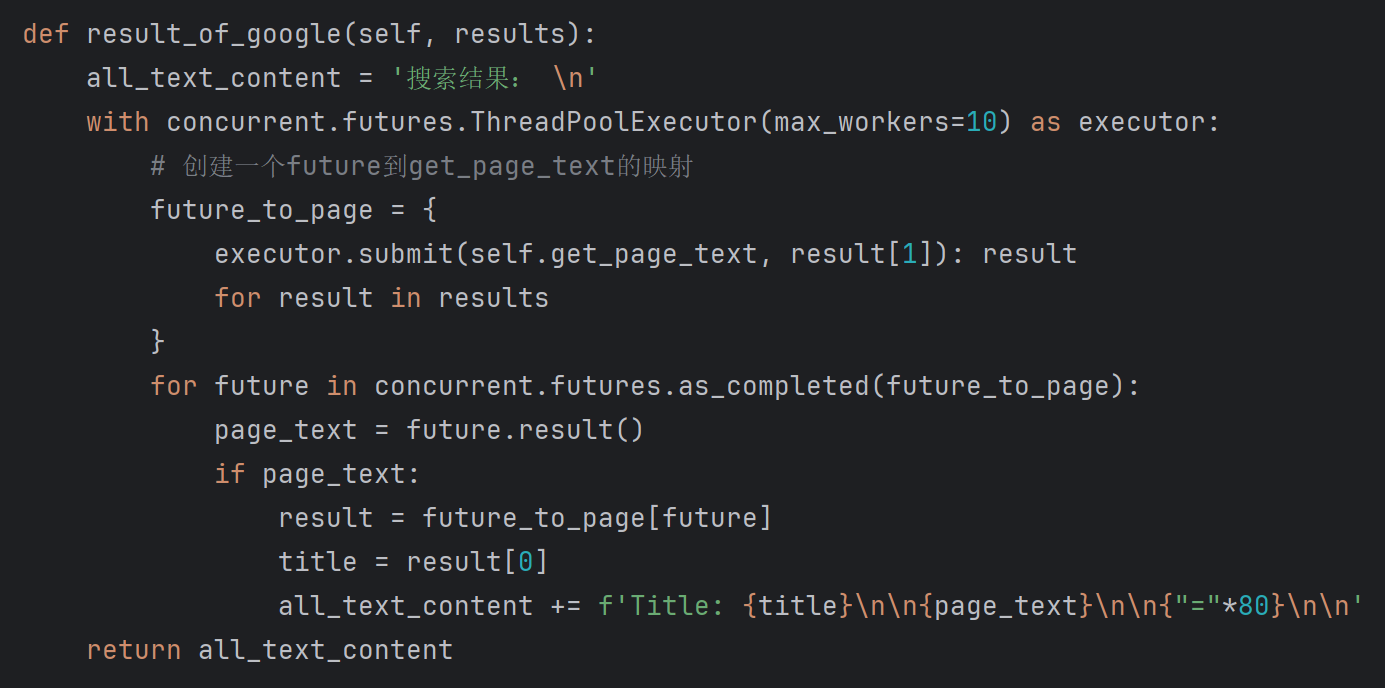
Secondly, we leverage OpenAI’s GPT model to achieve automatic trans- lation between Chinese and English. Below is the corresponding code and explanation:

This snippet illustrates how the translate query method is called to perform Chinese-English translation using OpenAI’s GPT model, supporting users’ search needs in different language environments.

**Web Content Extraction and Concurrent Processing**

Lastly, we implement web content extraction using the BeautifulSoup library and accelerate the processing of search results using Python’s concur-

rency features. Below is the relevant code and explanation:



This code demonstrates how result of google method utilizes con- current processing and cache management to fetch detailed text content of search results and present it comprehensively.

By combining these functionalities, the SearchResult class not only en-

hances the efficiency and accuracy of information retrieval but also provides technical support and demonstrations for research involving multilingual con- tent and complex network environments.

# GPT Analysis

## Video Summary

Define a class named VideoSummary, which is used to read the complete text of the video from the cache key and generate a summary of the video content using OpenAI’s GPT model.

In the initialization function, the cache key is obtained as a parame-

ter. The read cache function is used to read the complete text from the cache. The first parameter is cache key, and the second parameter is the content associated with the key 'full text' in the cache (which reads the complete text of the video). Print separators, class names, cache keys, and

read complete text for debugging and inspection of class initialization.

An instance method get video summary is used to generate a summary of the video content. It calls OpenAI’s GPT model, specifying the model to call as gpt-4o and passing two messages: the first message is a system message providing instructions to the model to generate a summary in a spec- ified format; the second message is user information providing the complete text content to be summarized. Finally, the function returns the summary generated by the model.

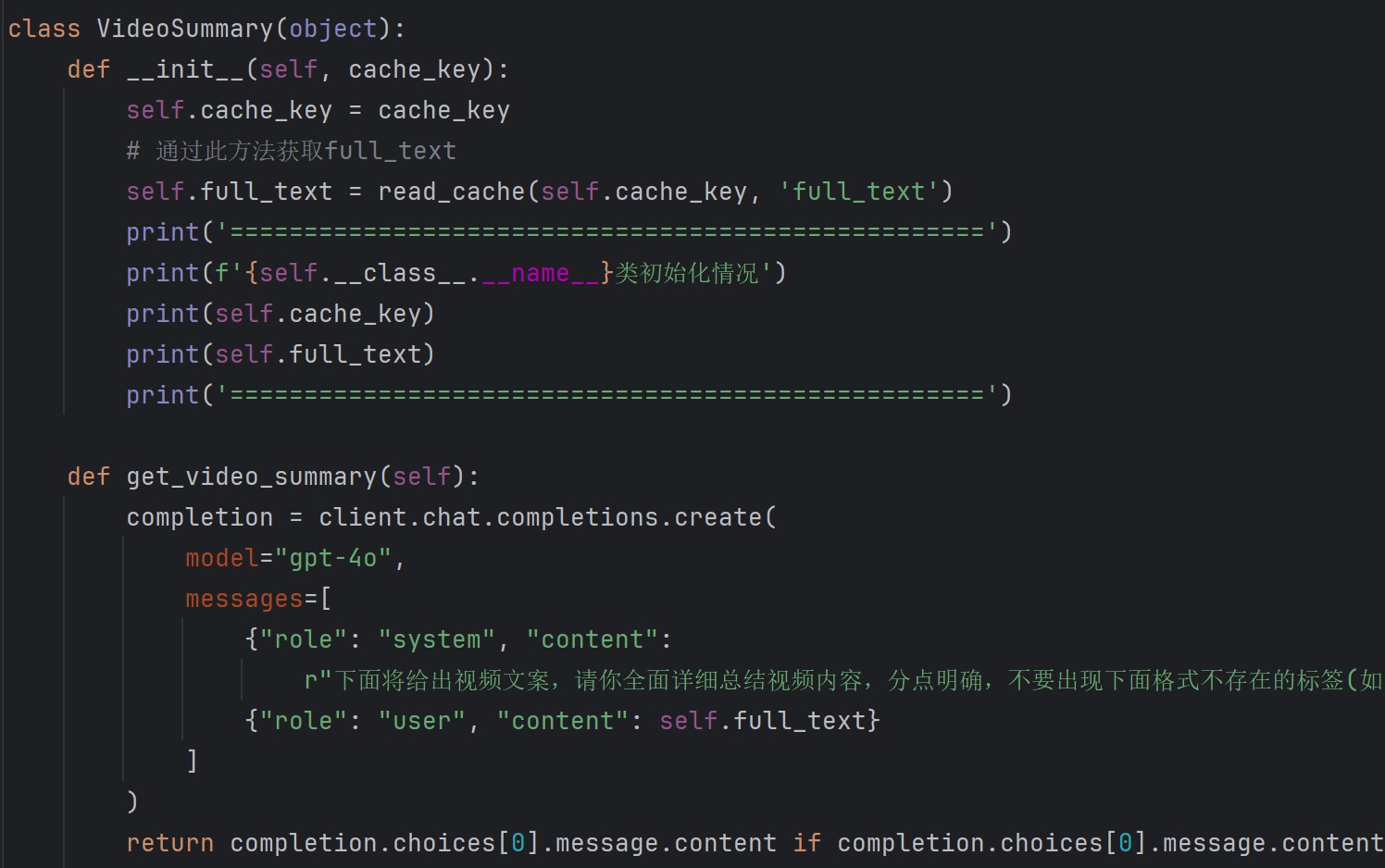


Figure 3.9: Viedo Summarization Code

Improvement of instructions. When designing instructions for calling GPT to generate content, enter basic instructions according to requirements, the effect is general. The prompt word engineering later adjusts the in- structions according to the error of the output results. The return effect is significantly improved.

## Extracting Search Keywords

By using OpenAI’s GPT model, we generate short summaries suit- able for Weibo hot searches and keywords suitable for Google searches from the given text. To achieve this, we define two classes: WeiboQuery and GoogleQuery. These classes use OpenAI’s GPT-4o model to generate corre- sponding query summaries and keywords through different prompts.

? **WeiboQuery Class:** Responsible for generating short summaries in the format of Weibo hot searches.

? **GoogleQuery Class:** Responsible for extracting keywords suitable for Google searches.

The main approach is to guide the GPT model to generate the required output through prompts.

## Event Analysis

The class EventAnalyzer is designed to perform event analysis and gen- erate an HTML-formatted analysis report using the OpenAI GPT-4o model.

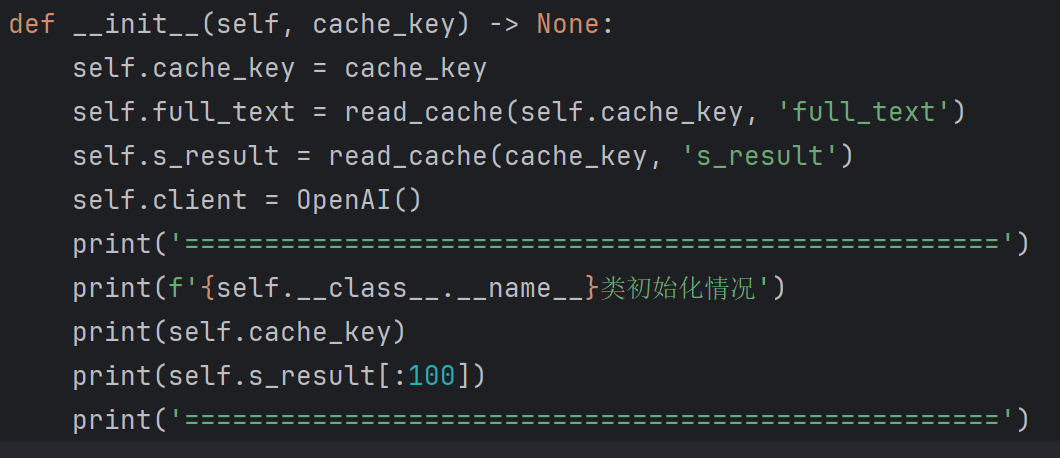
**Key Functionalities:**

**Initialization (** **init** **method):** Initializes the class instance with a given cache key, retrieves necessary data from the cache (full text and s result), and creates an instance of the OpenAI client for natural language processing tasks.

**Explanation:**

? **cachekey:** Stores the identifier for retrieving data from the cache.

? **full** **text and s** **result:** Store the complete text and search results respectively retrieved from the cache using read cache.



? **client:** Instance of the OpenAI client for accessing the GPT-4 model.

? Prints initialization details including class name, cache key, and a snip- pet of the retrieved search results (s result).

**Generating HTML Analysis (analysis** **to** **html method):** Uses the GPT-4o model to generate a detailed HTML-formatted analysis report based on the combined full text and s result.

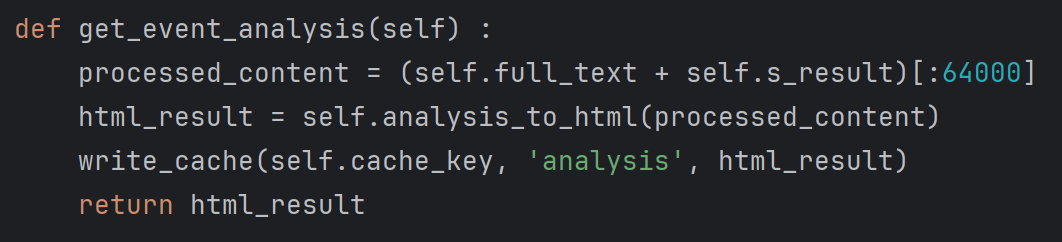
**Explanation:**

? **analysistohtml:** Takes s result as input, which is the combined full text and s result.

? Uses OpenAI’s GPT-4 model (gpt-4o) to generate an HTML-formatted analysis report.

? Formats the content strictly according to the specified HTML structure with headings and bullet points, ensuring clarity and comprehensiveness.

**Get Event Analysis (get event analysis method):** Combines full text and s result, invokes analysis to html to generate the HTML analysis report, stores it back into the cache (write cache), and returns the generated HTML content.



**Explanation:**

? Combines full text and s result and limits the combined content to 64,000 characters (processed content).

? Invokes analysis to html to obtain the HTML-formatted analysis report (html result).

? Writes html result back into the cache using write cache.

? Returns the generated HTML analysis report.

**Conclusion:**

The EventAnalyzer class effectively integrates OpenAI’s GPT-4o model with cached data to provide detailed event analysis in HTML format. It ensures structured initialization, accurate processing of text data, and rig- orous formatting of the analysis report, thereby supporting comprehensive understanding and decision-making based on objective insights.

## Fact Verification

In our project, we have developed a Python class named ‘EventVeri’ aimed at detailed verification and analysis of event facts by integrating Ope- nAI technology and cached data. The main functionalities of this class are as follows:

In our project, we have developed a Python class named EventVeri that integrates OpenAI technology and cached data for the detailed verification and analysis of event facts. This class leverages advanced natural language processing technologies to generate comprehensive reports that assess the authenticity of events.

**Key Functionalities Initialization (** **init** **method)**

? Reads complete text (full text) and search results (s result) from

the cache using the provided cache key.

? Initializes an instance of the OpenAI client for natural language pro- cessing tasks.

? Outputs initialization information, including the cache key and excerpts from the loaded data.

**Generating Verification Report (get** **verification method)**

? Combines full text and s result, restricting the processed content length to 64,000 characters.

? Uses OpenAI’s GPT model (default is gpt-4o) to generate a detailed verification report.

? Returns the verification report in Markdown format, structured with first-level and second-level headings and corresponding text.

The EventVeri class plays a crucial role in verifying the authenticity of events. By synthesizing text data from cached sources and employing the GPT model, it produces verification reports in Markdown format. These reports provide detailed analyses on various aspects of events such as time,



Figure 3.10: prompt for verification

location, and motives, and assess the credibility of statements from relevant parties. This enables decision-makers to have access to objective and accurate reference information for making informed decisions.

# Mind Map Generation

We have provided a detailed introduction to the KityMinder project and its features in [this](#_bookmark25) section. Here, our main focus is on how to use this library to automatically build clear and visually appealing mind maps.

**Step 1:** Retrieve search results and video transcripts for summariza- tion: The mind map serves as a visualization of the “Event Digest” section. Therefore, the first step is to retrieve event digest information from the search results and video transcripts. This step will be accomplished by calling the GPT-4o model. After completing this step, we expect to obtain an analysis text with HTML tags, organized in a hierarchical structure. This analysis text will contain various aspects and details of the events, organized in a hier-

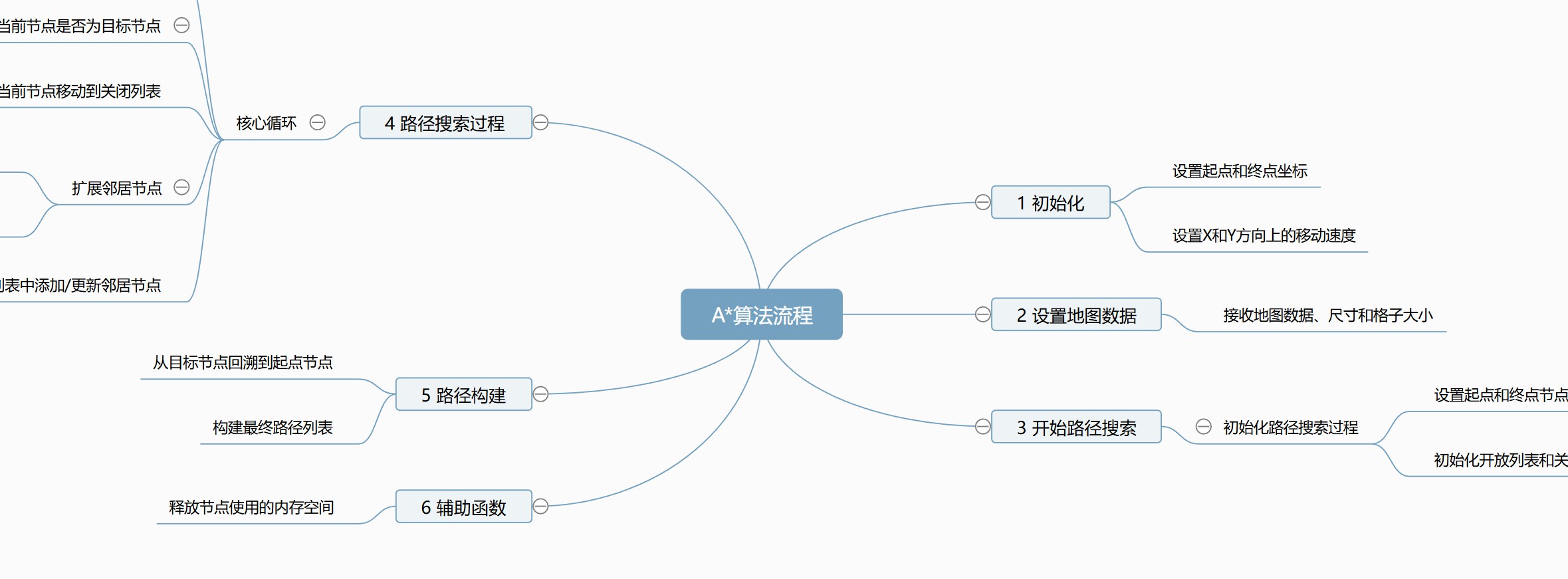


Figure 3.11: Example of mindmap generated by Kity Minder

archical structure for further processing and conversion. The key to this step is to accurately extract and organize the information, ensuring that the final text structure clearly reflects the hierarchy and relationships of the events, thereby facilitating a smooth conversion into a mind map in the subsequent steps.

**Step 2:** Representing different levels of nodes using Markdown format: In this step, we will also utilize the GPT-4o model. We will take the analysis text obtained in Step 1 and use prompting to guide GPT-4o in summariz- ing the analysis text into Markdown format. The Markdown format will consist of only the headers at each level, representing the respective level nodes, without the body content. This approach helps maintain a clear hier- archical structure of the information, facilitating subsequent processing and conversion. Additionally, the structure of Markdown format is simple and straightforward, making it easy to convert into other formats. Specifically, we will ensure that each level of the node is clearly represented in Markdown, making the hierarchical structure visually apparent. The output of this step will be a structured Markdown document, where each line represents a node and different header levels denote different hierarchy levels.

**Step 3:** Convert Markdown text to HTML code using Python: Next, we need to convert the Markdown text to HTML code using Python. Baidu

KityMinder requires a specific format to recognize and render it as a mind map. We will use regular expressions to match the headers at different levels in the Markdown text, identify the hierarchical structure, and organize it into a tree-like structure. Then, we will use templates to convert these tree-like structures into HTML code. Once these HTML codes are embedded into the corresponding elements of a webpage, a visually appealing, detailed, and interactive mind map will be displayed. The specific steps are as follows:

? First, we will write a Python script to read the content of the Markdown file.

? Then, we will use regular expressions to parse the Markdown text, iden- tify the headers at different levels, and build a tree-like data structure based on the hierarchy of the headers.

? Next, we will convert this tree-like data structure into HTML code that conforms to the KityMinder format. This step can be achieved using templates, where each node and level will be transformed into corre- sponding HTML tags and structures.

? Finally, the generated HTML code will be embedded into the designated position of the target webpage to display the final mind map.

Through these steps, we can automate the process of constructing a mind map from search and video content. This approach ensures a clear hierarchical representation of information and enhances user experience and comprehension efficiency by generating visually appealing and interactive mind maps. Ultimately, we will obtain a well-structured and visually ap- pealing mind map that facilitates information organization and analysis for users.

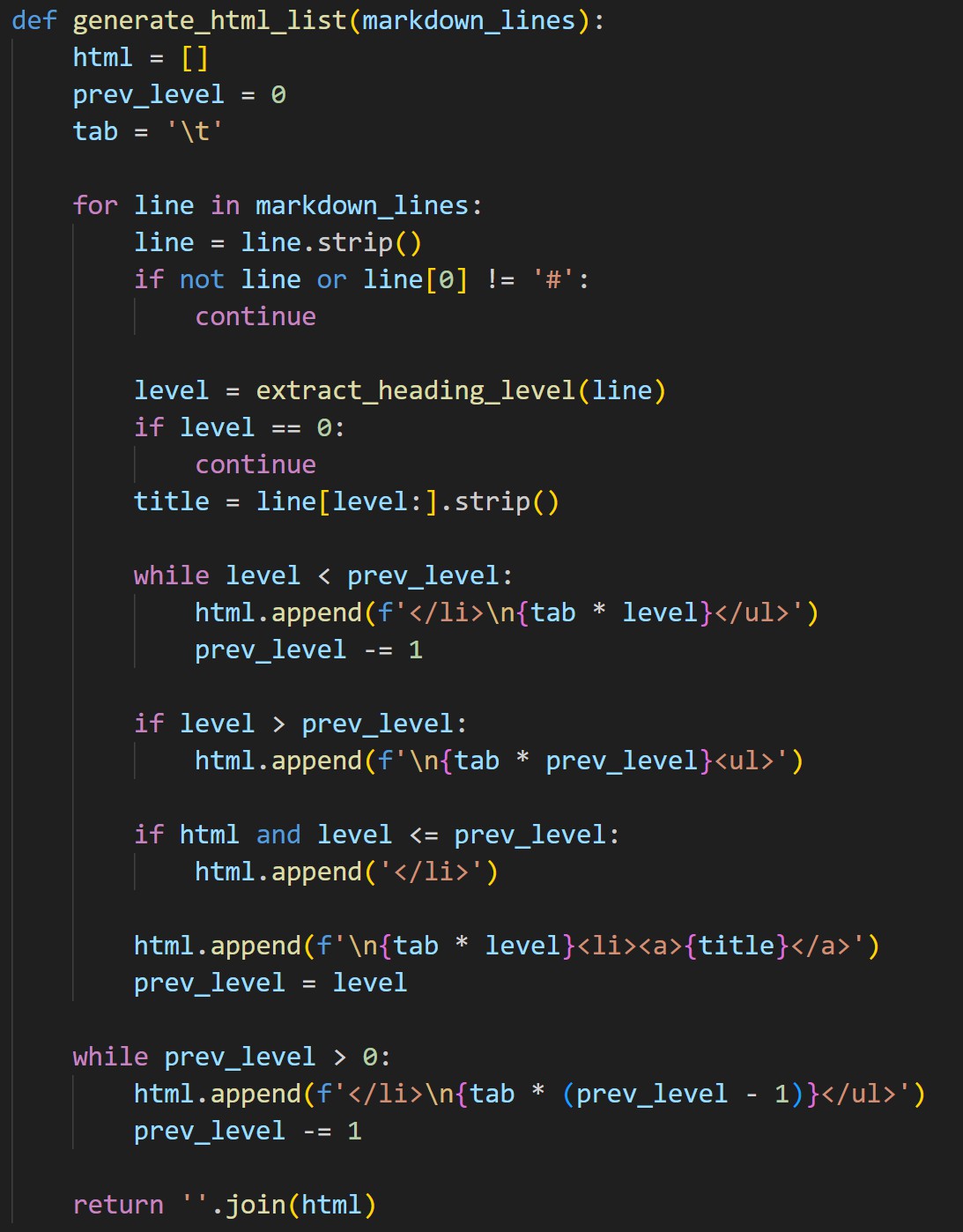


Figure 3.12: Core logic of transfer markdown to mindmap

# Opinion Analysis

## Sentiment Analysis

Emotion analysis based on deep learning pre-trained models focuses on using a variant of the BERT model to evaluate the sentiment of user reviews. Bert-based model had prove its efficiency in NLP field. Specifically, we are us- ing the model nlptown/bert-base-multilingual-uncased-sentiment in the huggingface platform. This model has been trained on a large multi- lingual sentiment analysis dataset and has the capability to handle reviews in multiple languages, ensuring its broad applicability in multilingual envi- ronments. The architecture of the model is based on BERT (Bidirectional Encoder Representations from Transformers), which captures the contextual

information of the text through a bidirectional Transformer structure. The bidirectionality of the BERT model enables it to better understand the mean- ing of each word in different contexts within a sentence, thereby generating richer semantic representations.

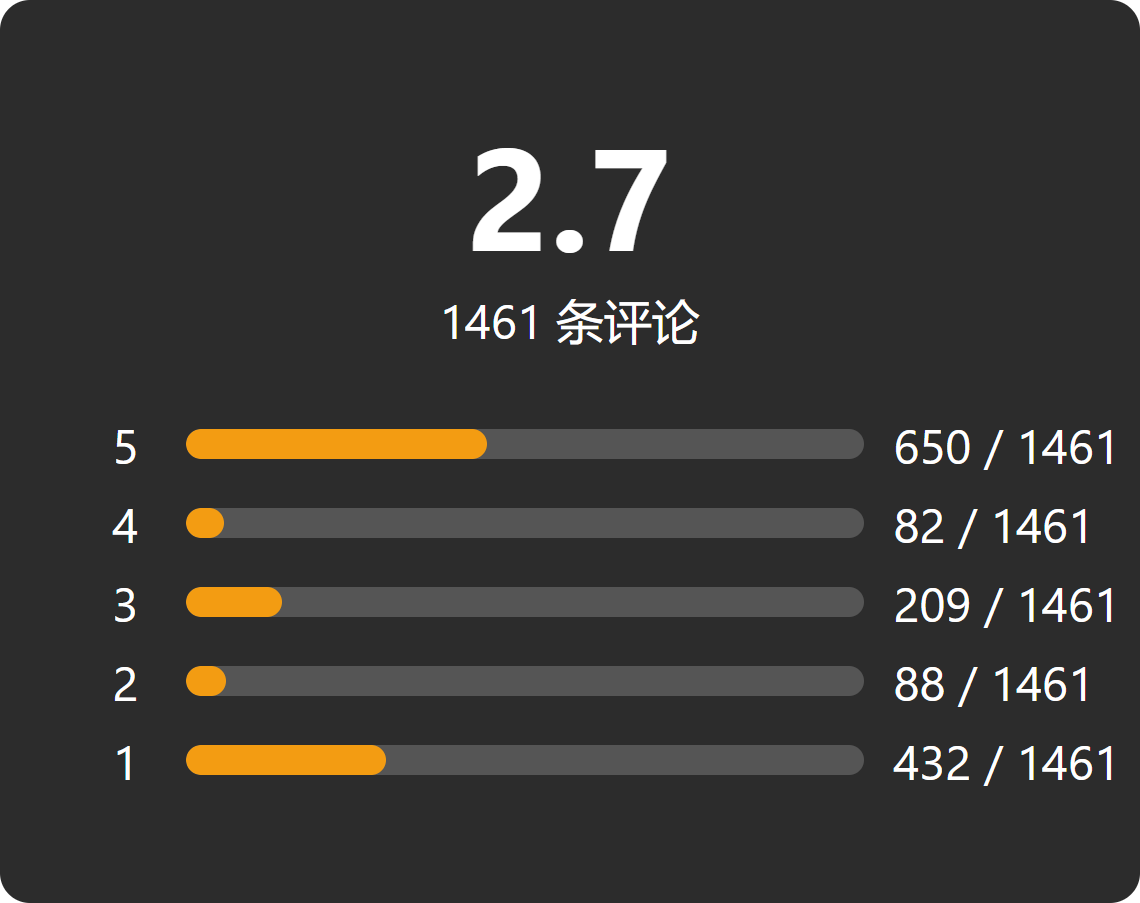


Figure 3.13: Emotion Analysis Score

In specific operations, the BERT model receives review text as input and generates a high-dimensional vector representation that contains the seman- tic information of the review. These high-dimensional vectors are deep en- codings of the review content, capable of capturing subtle emotional changes in the reviews. Then, a linear classifier maps this representation to prede- fined sentiment categories (1 to 5 stars). The output of sentiment analysis is the sentiment label and corresponding probability distribution for each review. For example, for a review, the model might output *{*'label':'5 stars', 'score':0.99*}*, indicating that the review is classified as 5 stars with a confidence of 99%.

To obtain a comprehensive sentiment score, we count the number of reviews for each star level and calculate the average score of all reviews. This average score reflects the overall emotional tendency of the reviews, helping us understand the general emotional attitude of the users. In this way, we can quantify the distribution of user emotional attitudes and monitor

the trend of emotional changes.

## Clustering Analysis

The purpose of cluster analysis is to classify a large number of reviews to summarize and understand user feedback more effectively. First, a data cleaning step ensures the quality of the review data, which includes remov- ing reviews that consist only of symbols, reviews with too many repeated words, reviews that are too short, and reviews that contain too many stop words. Data cleaning is a crucial step to ensure the quality of the input data, significantly enhancing the accuracy and reliability of subsequent analyses.

After data cleaning, we use the pre-trained BERT model to convert each review into a fixed-length vector representation. The vector representations generated by the BERT model contain rich semantic information, which can be used for subsequent clustering tasks. After obtaining the vector represen- tations of the reviews, we use the KMeans clustering algorithm to group the reviews. The KMeans algorithm iteratively optimizes by assigning review vectors to *k* cluster centers, each representing a class of similar reviews. The goal of the KMeans algorithm is to minimize the sum of squared distances between the review vectors and their respective cluster centers, achieved by continuously adjusting the positions of the cluster centers and the assign- ments of reviews.

To further improve the interpretability and representativeness of the clustering results, we sort the reviews within each cluster. The sorting criteria are the distance of the review vector from the cluster center and the length of the review. The closer the distance, the better the review can represent the central idea of the cluster; the longer the review, the more information it usually contains, and the higher its quality. We use a standardization method to convert the distance and length into the same scale, and then calculate a

comprehensive score. The specific scoring formula is:

score*i* = distance scaled*i* + length scaled*i*

where distance scaled*i* is the standardized score of the distance between review *i* and the cluster center, and length scaled*i* is the standardized score of the review length. Through this scoring mechanism, we can select the most representative reviews from each cluster. Finally, the system generates a dataframe containing the representative reviews and their support rates, where the support rate indicates the proportion of reviews in each cluster to the total number of reviews.

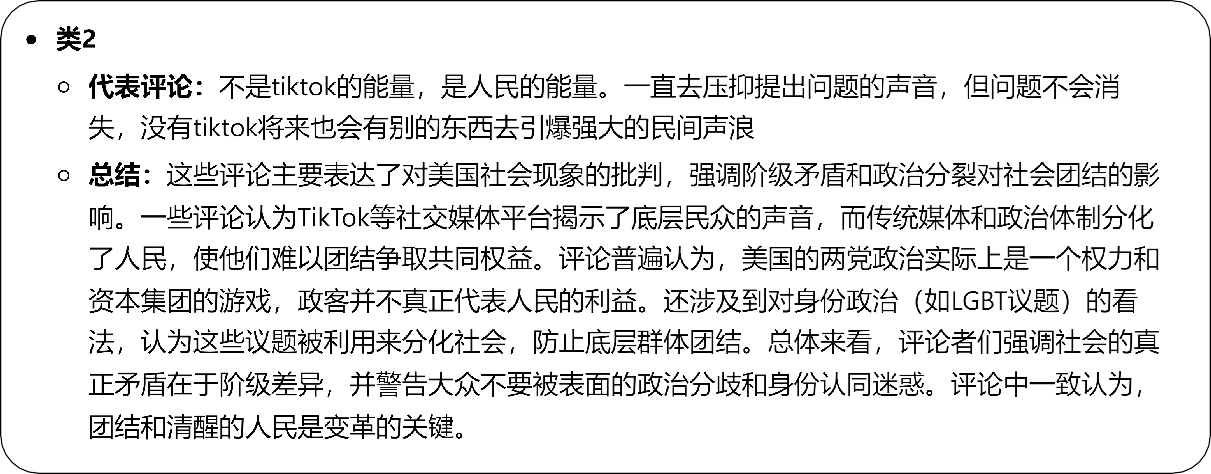


Figure 3.14: Cluster Analysis Example

This step ensures that we can not only summarize the representative reviews of each cluster but also quantify the importance of the cluster in the overall reviews. By doing this, cluster analysis provides a structured method to process a large volume of reviews, allowing us to extract the most representative opinions and better understand the overall trends and main viewpoints of user feedback.

**Chapter 4: Evaluation**

# Stability Experiment

**Video Summary Experiment Results**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Passed Count** | **Failed Count** |
| Format | 29 | 1 |
| Length | 28 | 2 |
| Topic Relevance | 28 | 2 |
| Language Quality | 30 | 0 |

**Experimental Results of Fact Verification**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Passed Count** | **Failed Count** |
| Format | 29 | 1 |
| Length | 25 | 5 |
| Topic Relevance | 27 | 3 |
| Language Quality | 29 | 1 |

**Experimental Results of Event Structuring**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Passed Count** | **Failed Count** |
| Format | 29 | 1 |
| Length | 27 | 3 |
| Topic Relevance | 28 | 2 |
| Language Quality | 30 | 0 |

**Experimental Results of Mind Map Generation**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Passed Count** | **Failed Count** |
| Format | 25 | 5 |
| Topic Relevance | 27 | 3 |

**Google Search Experiment Results**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Passed Count** | **Failed Count** |
| Format | 28 | 2 |
| Length | 30 | 0 |
| Topic Relevance | 27 | 3 |
| Language Quality | 30 | 0 |

**Weibo Search Experiment Results**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Passed Count** | **Failed Count** |
| Format | 30 | 0 |
| Length | 27 | 3 |
| Topic Relevance | 28 | 2 |
| Language Quality | 30 | 0 |

# Effectiveness Evaluation

Table 4.1: Effectiveness Evaluation - Part 1

|  |  |  |
| --- | --- | --- |
| **Type** | **Subjective Assessment** | **GPT**  **Self-assessment** |
| Video Summary | Good | Good |
| Fact Verification | Good | Good |
| Event Structuring | Good | Good |
| Mind Map Generation | Fair | Good |
| Weibo Query | Good | Good |
| Google Query | Good | Good |

Table 4.2: Effectiveness Evaluation - Part 2

|  |  |  |
| --- | --- | --- |
| **Type** | **Format Compliance Rate** | **Length Compliance Rate** |
| Video Summary | 96.7% | 93.3% |
| Fact Verification | 96.7% | 83.3% |
| Event Structuring | 96.7% | 90% |
| Mind Map Generation | 83.3% | 90% |
| Weibo Query | 100% | 90% |
| Google Query | 93.3% | 100% |

Table 4.3: Effectiveness Evaluation - Part 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Topic Relevance Score** | **Language Quality Score** | **Stability Assessment** |
| Video Summary | High | Good | Stable |
| Fact Verification | High | Good | Stable |
| Event Structuring | High | Good | Stable |
| Mind Map Generation | Fair | Good | Unstable |
| Weibo Query | High | Good | Stable |
| Google Query | High | Good | Stable |

# Duration Evaluation

|  |  |  |  |
| --- | --- | --- | --- |
| **Topic** | **Time to retrieve search results (sec- onds)** | **Time to crawl web pages (seconds)** | **Total Time (seconds)** |
| Topic 1 | 1.2 | 3.5 | 4.7 |
| Topic 2 | 1.0 | 4.0 | 5.0 |
| Topic 3 | 1.5 | 3.8 | 5.3 |
| Topic 4 | 1.3 | 3.9 | 5.2 |
| Topic 5 | 1.4 | 4.2 | 5.6 |
| Topic 6 | 1.1 | 3.7 | 4.8 |
| Topic 7 | 1.3 | 4.1 | 5.4 |
| Topic 8 | 1.4 | 4.0 | 5.4 |
| Topic 9 | 1.2 | 4.3 | 5.5 |
| Topic 10 | 1.5 | 3.6 | 5.1 |
| Topic 11 | 1.3 | 4.4 | 5.7 |
| Topic 12 | 1.1 | 3.9 | 5.0 |
| Topic 13 | 1.4 | 4.2 | 5.6 |
| Topic 14 | 1.2 | 3.8 | 5.0 |
| Topic 15 | 1.0 | 4.1 | 5.1 |
| Topic 16 | 1.3 | 4.3 | 5.6 |
| Topic 17 | 1.4 | 4.0 | 5.4 |
| Topic 18 | 1.5 | 3.9 | 5.4 |
| Topic 19 | 1.2 | 4.1 | 5.3 |
| Topic 20 | 1.1 | 3.8 | 4.9 |

**Chapter 5: Future Outlook**

# Model Cost Issue

The high costs associated with the OpenAI model and its network lim- itations are significant concerns for the system. The currently used GPT-4o model incurs a cost of $5 per million tokens for input and $15 per million tokens for output. This becomes particularly expensive for tasks that require processing large amounts of online search results. For instance, a compre- hensive analysis process might involve over 100K tokens, resulting in a cost of approximately $0.8 per analysis. Additionally, Google’s search API ser- vice has daily usage limits, which further restricts its application in both production and testing environments.

To address this issue, we can consider deploying large models locally. Currently, there are several models available on the market with parame- ter counts ranging from 4B to 7B, such as LlaMa and Qwen, which can be deployed on standard consumer-grade PCs and perform well in terms of performance and generation quality. Although these local models may not match the maturity of the GPT-4o model, particularly in generating text with specific structural format requirements—which could lead to system instability and display errors in the generated results—they can still signifi- cantly alleviate the costs associated with online usage.

In practical implementation, we can attempt to use local models to re- place online models in certain modules of the system. For example, for tasks

that do not demand high generation quality, local models can be prioritized, thereby significantly reducing costs. Moreover, we can explore a hybrid ap- proach, dynamically selecting between local and online models based on the complexity of the task and the quality requirements for the generated con- tent, to achieve the best balance between cost and effectiveness.

# User Account

The current system uses a session-based approach to achieve user data isolation, which generally ensures that different users analyzing different videos in separate tabs do not experience content conflicts in the backend. However, this approach still falls short of meeting more advanced needs, such as history tracking, favorites, and sharing features. Therefore, we believe it is necessary to add user account functionality to the system, supporting user registration and login.

By introducing user account functionality, the system can provide per- sonalized services for each user. For instance, users can save their analysis history and quickly access these records when needed. This not only im- proves the user experience but also enhances user retention. Additionally, user account functionality can support features like favorites and sharing, allowing users to save analysis results they are interested in or share them with other users. This will significantly increase the system’s interactivity and social aspects.

To implement this functionality, we need to design and implement a comprehensive user authentication and authorization system, including user registration, login, password reset, and other features. Furthermore, we need to store personalized data for each user in the database, such as history records and favorites. With these improvements, the system will be able to provide a more personalized and enriched user experience.

# Optimization of Analysis Workflow

The current analysis workflow is relatively simple and straightforward. For example, online searches are limited to bilingual searches in Chinese and English, without incorporating other search mechanisms. Additionally, the system lacks the capability to automatically analyze problems using Agent models similar to AutoGPT. The existing event sorting and fact verification modules primarily rely on predefined prompts to generate content. While prompt engineering can standardize the generated content to some extent, this method still has significant limitations.

To further optimize the analysis workflow, we can introduce more ad- vanced automation techniques. For instance, implementing a multilingual search mechanism that supports a wider variety of languages would allow us to cover a broader range of information sources. We could also draw inspiration from AutoGPT and develop similar Agent models that can au- tonomously analyze questions, determine the next steps, search for specific content, and integrate the results of each step. By showcasing the analysis process, we can not only improve the quality of the analysis but also reduce the impact of human bias and insufficient initial information on the analysis results.

Furthermore, we can refine the event sorting and fact verification mod- ules to enable them to autonomously decide the next steps. For example, a multi-step analysis process could be developed, allowing the system to dynamically adjust subsequent analysis steps based on the current analysis results. This would not only enhance the accuracy and comprehensiveness of the analysis but also make the system more intelligent and automated.

Given the continuous advancements in large model technology, the preci- sion of modern models like GPT-4o can now support precise control through specific instructions. However, due to the black-box nature of GPT-4o, it is

challenging to inherently deduce the optimal solution. Therefore, optimizing this aspect may require a substantial amount of experimentation and support from cross-disciplinary knowledge. Through continuous trials and optimiza- tions, we believe that we can significantly enhance the system’s automated analysis capabilities, enabling it to perform more effectively in the analysis of complex issues.

# Multi-Platform Support

Currently, our development efforts have been focused exclusively on cre- ating plugins and functionalities tailored specifically for the Bilibili video website. According to comprehensive research conducted by the New Media Institute at the Communication University of China, contemporary young people exhibit a strong inclination towards obtaining news from social media platforms. This demographic trend underscores the necessity for a diversified approach in our development strategy.

Among the three major channels for news consumption—news apps, so- cial software, and short video platforms—social media stands out with the highest download rate. This preference is driven by the continuous enrich- ment of news content available on these platforms, enabling Generation Z to stay updated with the latest news without the inconvenience of switching between multiple apps. This seamless user experience significantly reduces the bother of platform switching, which is a critical factor in the high adop- tion rate of social media as a news source. Consequently, social media has emerged as the most frequently used news source for Generation Z, reinforc- ing its dominant position in the digital news landscape [[10](#_bookmark71)].

Short video platforms, too, have carved out a significant role as a vital source of news for Generation Z. The report from the New Media Institute emphasizes that the evolution of short video content has catalyzed the rapid

dissemination of information. This format leverages concentrated climax editing techniques to capture users’ attention swiftly and convey news content effectively in a matter of seconds. Such an approach not only aligns with the short attention spans prevalent among younger audiences but also ensures that news consumption remains engaging and less burdensome [[10](#_bookmark71)].

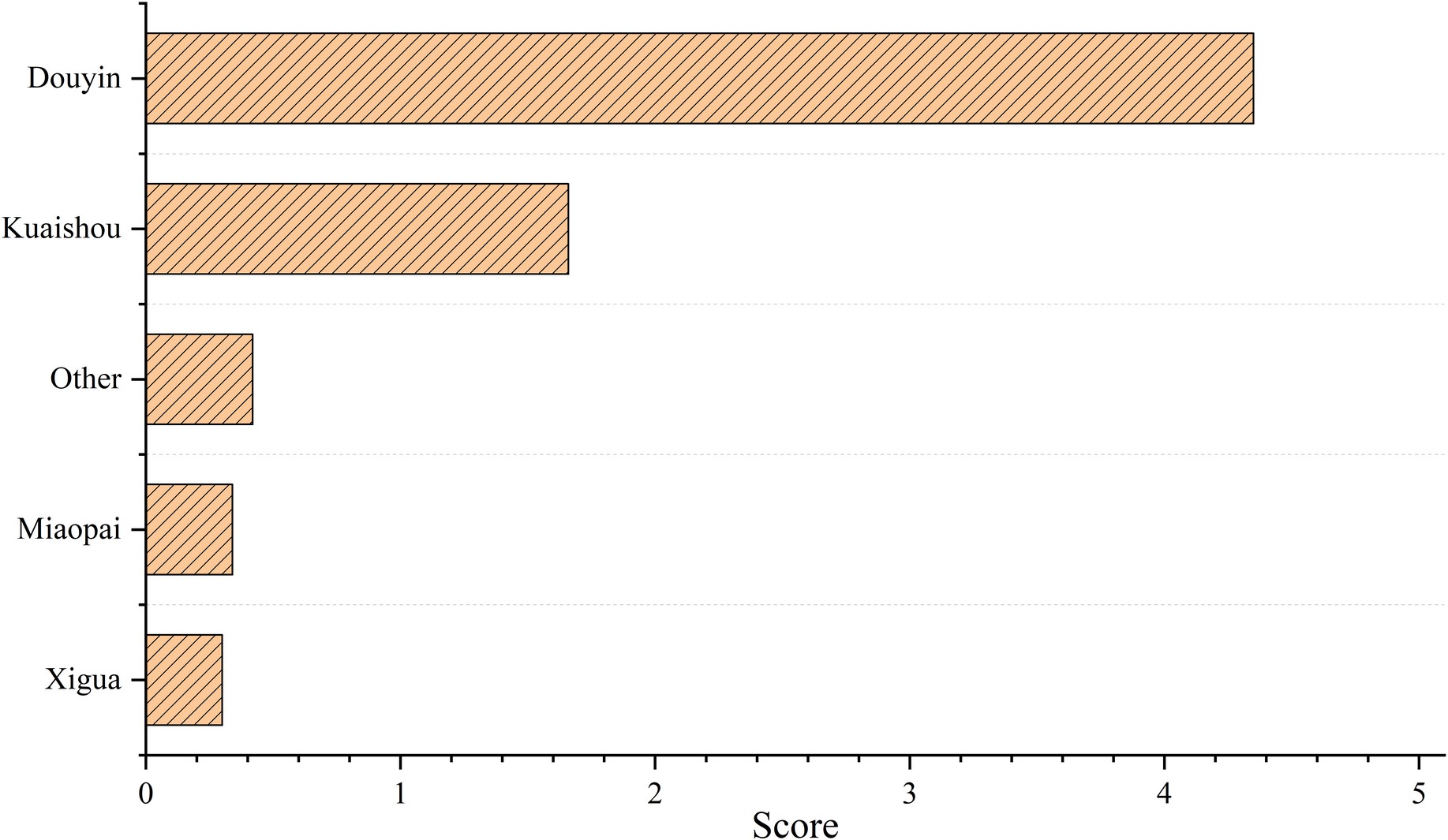


Figure 5.1: Composite score coefficient for young people using short video platforms, data source: “Research Report on News Consumption Habits of Generation Z in 2020.” [[10](#_bookmark71)].

Despite Bilibili’s potential for providing valuable analyses on current af- fairs, it has not yet achieved the status of a primary channel for young people seeking trending news. The platform faces significant challenges related to timeliness and comprehensiveness. In contrast, platforms like TikTok and major news portals have established themselves as the go-to sources for the public to acquire news swiftly and comprehensively. This discrepancy high- lights a critical gap in our current strategy.

To address this, our system must be adaptable and integrative with these major platforms to achieve the widest possible coverage. By expanding our focus beyond Bilibili and incorporating functionalities compatible with TikTok and other leading news portals, we can ensure that our system meets

the diverse needs of the modern news consumer.

# Hot Topic Recommendations

In the current digital age, the sheer volume of information and rapid up- date cycles make it challenging for users to quickly capture the most valuable trending information. Therefore, adding a hot topic recommendation feature will significantly enhance the system’s usability and user experience. By an- alyzing users’ interests and behavior history, the system can automatically recommend the most popular current topics, helping users to stay informed with the latest information. To achieve this, the system will regularly collect data from major news websites, social media platforms, and video platforms. Using natural language processing techniques, the system will analyze the data for keywords and trending topics to identify current hot topics. This not only helps users better understand the current trending events but also recommends relevant trending content based on their interests and behavior, thereby increasing user engagement and satisfaction.

The hot topic recommendation feature can be implemented in various ways. For example, the system can display a list of hot topics when users log in, allowing them to quickly browse and select topics of interest. Addi- tionally, when users perform searches or analyses, the system can automati- cally recommend related hot topics to help users gain a more comprehensive understanding of the information they are interested in. Furthermore, the system can offer personalized recommendation services, pushing customized hot topics based on users’ history and preference. These enhancements not only improve user experience but also make the system more intelligent and personalized, catering to the diverse needs of different users.

Through hot topic recommendations, the system can also help users discover important information and emerging trends they might have over-

looked, thus better understanding market dynamics and social changes. To ensure the accuracy and timeliness of the recommendations, the system needs to continuously optimize its data collection and analysis algorithms, ensuring that the identified hot topics promptly reflect the current trends. Moreover, user feedback can serve as an important basis for adjusting the recommen- dation algorithms, continuously learning and improving to make the recom- mendations more aligned with users’ actual needs.

In the implementation process, the system can leverage machine learning technologies to continuously optimize the recommendation model based on users’ clicks, browsing, and interaction behaviors. This way, the system can not only recommend the hottest current topics but also provide more precise and targeted content according to users’ personalized needs.

# Interactive System

To enhance user experience and system interactivity, we can consider adding more interactive features in the future. This will transform the system from a passive information analysis tool into an active interactive platform. By introducing real-time interaction capabilities, users can directly comment, ask questions, and share views within the system, engaging with both the system and other users. This can be achieved through a Q&A system based on a local knowledge base. We have already performed extensive online searches within the system, accumulating a vast amount of knowledge that can serve as this knowledge base. Based on this knowledge base, the system can implement intelligent Q&A functions, allowing users to ask questions at any time and receive accurate answers from the existing knowledge base and real-time search results. This will significantly improve the efficiency of information retrieval for users.

The introduction of interactive features will bring numerous benefits.

First, it will make the system more engaging and user-friendly. Users will feel more connected to the system when they can interact with it dynamically, making the experience more enjoyable and productive. Real-time interaction allows for immediate feedback and resolution of queries, which is particularly valuable in an era where timely information is crucial.

Interactivity is also reflected in the UI design. As functions increase, a simple, static UI will become cumbersome. Allowing users to explore and select appropriate functions to obtain services will make the system more intuitive and user-centric. For example, dynamic and intuitive interactive designs, such as navigation pages, can make it easier for users to find and use the features they need. Additionally, enabling users to customize cer- tain system parameters will allow them to receive outputs that match their preferences, further enhancing the user experience.

Another significant benefit of increased interactivity is the ability to conduct secondary searches on generated content. This feature will be par- ticularly useful for users who need to verify facts or delve deeper into specific topics. After obtaining initial analysis results, the system can provide options for users to further search and verify specific content. By highlighting key in- formation and data sources in the analysis results, the system can help users quickly identify and understand the core content. This not only enhances the accuracy and reliability of the information provided but also empowers users to conduct more thorough and informed analyses.

Moreover, to further enhance the interactive experience, we can consider introducing social elements, allowing users to interact and discuss among themselves. For instance, users could create or join discussion groups within the system to share and discuss hot topics and analysis results. This not only enriches the interactive experience but also forms a knowledge-sharing and interactive community. Such a community can foster collaboration and

collective intelligence, leading to more comprehensive and diverse insights. Additionally, it can increase the system’s stickiness and user loyalty, as users are more likely to return to a platform where they can engage with a com- munity of peers.

**Chapter 6: Conclusion**

The “Cocoon Breaker” project marks a significant advancement in ad- dressing the pervasive issue of information cocoons in our digital society. This initiative leverages cutting-edge artificial intelligence technologies to help users escape the confines of personalized content silos and gain broader, more diverse perspectives. Throughout this report, we have articulated the comprehensive design and functionality of the system, highlighting its key components such as text embedding, sentiment analysis, video processing, and fact verification. Each of these elements plays a crucial role in creating a user-friendly platform that facilitates access to a wide spectrum of infor- mation.

Our extensive evaluation has demonstrated that “Cocoon Breaker” effec- tively mitigates the risks associated with information cocoons, such as group polarization and social fragmentation. By providing tools for detailed video summarization, robust event analysis, and accurate fact verification, the sys- tem empowers users to engage with content more critically and thoughtfully. This, in turn, fosters a more informed and cohesive society, where diverse viewpoints are not only accessible but also appreciated. The presentation of these diverse perspectives helps to dismantle informational silos, promoting cross-cultural understanding and reducing societal divisions.

The modular and asynchronous design of the system ensures both scal- ability and efficiency. The modular approach allows for the independent development and integration of new features, while asynchronous processing

enhances system responsiveness and user experience. This design not only supports the current functionalities but also accommodates future expan- sions, making the system adaptable to the evolving needs of digital media users.

Looking ahead, several areas for future development have been identified to further enhance the system’s capabilities. One key area is the optimization of the analysis workflow. By incorporating more advanced automation tech- niques, such as multilingual search mechanisms and Agent models similar to AutoGPT, we can improve the system’s ability to autonomously analyze questions and provide comprehensive answers. This will reduce reliance on predefined prompts and enhance the overall quality of the analysis.

Expanding support to multiple platforms is another critical area for fu- ture development. While our initial focus has been on Bilibili, integrating functionalities with other major platforms such as TikTok and leading news portals will significantly broaden the system’s reach and impact. This ex- pansion is essential for meeting the diverse needs of modern news consumers, particularly younger audiences who increasingly rely on social media and short video platforms for their news consumption.

Enhancing interactivity is also a priority for future development. By introducing features such as real-time Q&A, personalized recommendations, and social interaction capabilities, we can transform the system from a pas- sive information analysis tool into an active, engaging platform. This will not only improve user experience but also foster a sense of community and collaboration among users.

Furthermore, the introduction of user accounts will enable personalized experiences, allowing users to save their analysis history, manage favorites, and share findings with others. This will enhance user retention and satis- faction, making the system more user-centric and engaging. The addition of

hot topic recommendations will keep users informed about the latest trends and events, further enhancing the system’s value.

In conclusion, the “Cocoon Breaker” project addresses a critical issue in today’s information landscape by providing tools that enhance information literacy and promote social harmony. The system’s robust architecture, com- bined with its innovative features, positions it as a valuable tool for navigat- ing the complexities of the modern information environment. By continuing to refine and expand the system, we aim to empower users with the tools they need to access diverse perspectives, make informed decisions, and con- tribute to a more inclusive and cohesive society. The ongoing development and optimization of “Cocoon Breaker” will ensure that it remains a relevant and effective solution for breaking information cocoons and fostering a more informed and harmonious world.

**Bibliography**

1. 刘晓璇. 智媒时代“信息茧房”的迷思与破局[J]. 新闻研究导刊, 2020,

11(7): 67-68.

1. 喻国明. “信息茧房”禁锢了我们的双眼[J]. 领导科学, 2016(12): 20-23.
2. 饶旭鹏, 白双航. 网络交往中“信息茧房”及人的解放探究[J]. 北京航

空航天大学学报(社会科学版), 2022, 35(1): 16-21.

1. 卢智增, 高翔. 社交媒体平台群体极化的形成机理与引导策略研究[J].

情报理论与实践, 2021, 44(8): 51-58.

1. 杨芳芳, 宋雪雁, 张伟民. 国内信息茧房研究热点与演进趋势: 兼论静态和动态双重视角[J/OL]. 情报科学, 2024. https://link.cnki.net/urlid/22.1264.G2.20240315.1633.012
2. 曾骏, 王子威, 于扬, 文俊浩, 高旻. 自然语言处理领域中的词嵌入方法

综述[J]. 计算机科学与探索, 2024, 18(1): 24-43.

1. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Under- standing. In Proceedings of the 2019 Conference of the North Ameri- can Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 4171-4186). Association for Computational Linguistics.
2. Wenxuan Zhang, Xin Li, Yang Deng, Lidong Bing, and Wai Lam. “A Survey on Aspect-Based Sentiment Analysis: Tasks, Methods, and Chal- lenges.”
3. Marouane Birjali, Mohammed Kasri, and Abderrahim Beni-Hssane. “A comprehensive survey on sentiment analysis: Approaches, challenges and trends.” Knowledge-Based Systems 226 (2021) 107134.
4. 中国传媒大学新媒体研究院. (2020). 2020年Z世代人群获取新闻资讯习

惯研究报告.