



Data Mining from Education-Related Search Suggested Text on the Web

Jincheng Zhang^{1*}, Thada Jantakoon¹, Rukthin Laoha¹ and Potsirin Limpinan²

¹*Faculty of Science and Technology, Rajabhat Maha Sarakham University, Maha Sarakham, Thailand*

²*Faculty of Information Technology, Rajabhat Maha Sarakham University, Maha Sarakham, Thailand*

*Corresponding author; E-mail: zjc1639834588@gmail.com

Received: 03 September 2023 /Revised: 09 May 2024 /Accepted: 15 May 2024

Abstract

An increasing number of industries are implementing digital technology to increase efficacy due to its accelerated development. Undoubtedly, the education sector is not an exception. In addition to conducting data mining and analysis of Internet search-recommended text for the first time, this research developed and proposed the world's first algorithm and technology to obtain such text automatically and rapidly. By performing data mining on education-related search suggestion texts on the Internet, this study aims to extract educationally beneficial information. This study collects education-related search recommendation texts from YouTube using the platform as an example. Literature review and theoretical analysis; algorithm design and optimization; system design and implementation; support for applications; construction and annotation of data sets; empirical research and experimental validation; interdisciplinary research and application are some research methods employed. These are the outcomes of this research: Valuable insights were extracted from an analysis of over 50,000 lines of collected data, including popular keywords and the emotional proclivities of individuals. Thus, supporting the instructional and decision-making practices of individuals.

Keywords: Crawler, Data mining, Education, Algorithm, ICT for education

Introduction

The influence of digital technology has evidently permeated every aspect of existence, including the education sector. The use of digital technology in education is growing¹. By integrating digital technology into the classroom, instructors can enhance the caliber of instruction and the learning outcomes of their pupils². Utilizing a Python crawler, we developed and proposed in this paper a novel algorithm for obtaining search suggestion text from the Internet. We then applied this newly developed algorithm to the obtained education-related data. Eight codes were employed to examine the data from various levels or perspectives.

1. Digital transformation in education

Digital technology has become an integral and crucial component of education due to its growing utilization in the field. YouTube is the largest video website globally and the second-largest search engine in the world. This project will use YouTube as a case study to collect data for research purposes.

2. Application of data mining technology in education

Data mining has been utilized extensively in the analysis of educational data³⁻⁴. Data mining in education has broad uses in enhancing students' engagement in learning, providing tailored education, forecasting student achievement, and

predicting dropout rates⁵⁻⁸. Data mining tools can help get a deeper insight into students and their learning environment. Data mining can assist educators in creating more effective teaching strategies and enhancing students' learning outcomes.

3. Research purpose and importance

This study introduces a novel method that enables individuals to rapidly access a substantial volume of search recommendation texts on the Internet, addressing the challenge of data acquisition. We utilized the new technology to gather over 50,000 education-related search recommendation texts from YouTube. We utilize 8 pieces of code to examine data from several levels or perspectives, including classification, identifying popular search terms, doing sentiment analysis, and detecting unusual content.

4. Research framework and structure

The study will be segmented into seven segments to thoroughly elucidate the study aims. The initial Part elucidates the research backdrop and its relevance, establishing the basis for comprehending the study context. The second part will examine pertinent theories and studies, and undertake a thorough investigation of the implementation of educational data analysis theory and data mining in the education sector. The third part will provide a thorough explanation of the

research methodologies employed. The fourth part will discuss the algorithms and methods employed in data gathering, the process of data analysis and mining, and the outcomes of data analysis and mining across several levels or aspects employing 8 distinct codes. The fifth part elaborates on the CRISP-DM data mining approach that was utilized. The sixth part is an overview of data analysis and data mining. Part seven will present the study's results and suggest directions for further research and practical applications in education.

5. Scope

This research has a comprehensive and detailed scope, primarily encompassing the following aspects:

5.1. Use the Python programming language to write a web crawler program to obtain education-related search suggestion text data from the YouTube platform.

5.2. Apply machine learning and data mining technology to conduct in-depth analysis of education search term data and identify current hot topics, keywords and trends in education.

5.3. Analyze research results and explore application scenarios such as the development direction of the education field, optimization of teaching content, and formulation of educational policies.

6. Expected return

Through this research, we expect to achieve the following benefits:

6.1. Provides an educational research method based on big data analysis, enriching the tools and means of educational research.

6.2. Invented and proposed a new algorithm to obtain search suggestion text on the web.

6.3. Deeply explore the potential of search recommended texts on the Internet for data analysis or data mining, providing educators and decision-makers with more accurate and comprehensive data support.

6.4. Provide data support and reference basis for the optimal allocation of educational resources, the formulation of educational policies and the promotion of educational innovation.

6.5. Promote digital transformation in the education field and promote the globalization and popularization of education.

7. Beneficiary

Identify groups who will benefit from the research, including researchers, policymakers, educators, and technicians.

Who will benefit from this research:

Beneficiaries of this study include but are not limited to:

7.1. Educational researchers and scholars can use this research method to conduct deeper and more comprehensive research in the field of education. It provides

educational researchers and scholars with an educational research method based on big data analysis. And we invented and proposed a new algorithm to obtain search suggestion text on the Internet. Educational researchers and scholars can use this new algorithm to obtain the data they want in any field.

7.2. Educational managers and policymakers can use the results of this study to optimize the allocation of educational resources and guide the formulation of educational policies. For example, from the results of sentiment analysis of data, data are collected from different time periods to analyze changes in people's emotions about education. Not only can it collect data related to education, but you can also obtain relevant data on a certain subject by changing the search terms used to obtain search suggestions (such as changing educate to math, etc.), thereby knowing what people think about a certain subject. Emotions and emotional changes.

7.3. Education practitioners and educational technicians can use the findings of this study to guide teaching practice and improve teaching effects. For example, you can find content that people want to learn and are interested in from popular search words, and use it to optimize your teaching plans.

8. Related theories and research

We delved into the theoretical framework and previous related research relevant to this study. Through a review of relevant theories and research, we can better understand the background, motivation, and theoretical basis of the current study.

8.1 Theoretical framework

There has been no technology or algorithm that can automatically and quickly collect search recommendation text on the Internet, and there has been no previous research on data analysis or data mining of search recommendation text on the Internet. This study pioneered the invention and proposed the use of Python crawlers to automatically and quickly collect search recommendation texts on the Internet, and used the collected data for data analysis and data mining.

8.2 Key concepts and definitions

Web crawler technology and data mining technology both have a long history.⁹⁻¹⁰ People use web crawler technology and data mining technology widely in all walks of life.¹¹ Web crawling technology is an automated program used to browse the Internet and collect information.¹² Data mining is the process of discovering and extracting useful information and knowledge from large amounts of data.¹³⁻¹⁴ To conduct data mining, you first need to have a large amount of

data¹⁵. How to obtain a large amount of data? In this study we use crawler technology to obtain search recommendation texts on the web through a new algorithm we invented and proposed.

9. Summarize

There is a lot of data on the Internet, but it is not easy to find the valuable data you want.¹⁶ Through the methods and algorithms we invented and proposed to obtain search recommendation text on the Internet., we can automatically and quickly collect search recommendation texts on the Internet, and this technology can be used in various fields, such as collecting data from different industries, such as collecting entertainment, education, politics, Data from various different industries such as medical care are then used for data analysis or data mining. In this study, we use "educate" as the main keyword to collect search recommendation text to collect search recommendation text on YouTube websites related to "educate". For example, if we want to collect medical-related data, we can use "medical" as the main keyword to collect search recommendation text to collect search recommendation text on the YouTube website related to "medical". Of course, we can also use this method to collect search recommendation text from other websites, such as Google, Yahoo, etc. We can also

change to different countries and regions or browser languages in the Python crawler code to collect search recommendation text in different countries and regions or regions without language, search recommendation text in different countries and regions or regions without language will vary. When we collect data in this study, the country is set to the United States and the language is set to English in the code.

Material and method

1. Literature review and theoretical analysis

In this study, we systematically reviewed the existing research results and theories and discussed their applications, advantages and disadvantages, and future development directions in specific fields. Through a literature review, this paper has an in-depth understanding of the application of crawler technology in data collection, the role of data mining technology in information discovery and knowledge extraction, and the transformation and impact of information and communication technology on education. These theoretical analyses provide important theoretical support and guidance for our research.

2. Algorithm design and optimization

In the course of the study, we designed a new algorithm for obtaining education-related data to obtain the text of search

suggestions on the web. The algorithm can efficiently obtain a large amount of education-related data from platforms such as YouTube, and extract education-related knowledge through data mining technology.

3. System design and implementation

We have designed and implemented a complete system based on crawler and data mining technology, including data collection, storage, cleaning, analysis, and display functions. The system can automatically collect education-related data from the Internet, process and analyze the data, and ultimately present valuable information and insights to users. The design and implementation of this system are an integral part of the research work, and they provide technical support for the in-depth analysis of the current educational dynamics. We invent and propose a new algorithm for obtaining text from web search suggestions.

4. Application examples

We select specific application scenarios, use crawlers and data mining techniques to solve practical education problems and analyze the effectiveness and practicability of the method. Through application cases, the feasibility and effectiveness of our research methods and technologies in practical scenarios are verified, and the practical application in the field of education is provided with reference and reference.

5. Construction and annotation of datasets

We built a text dataset of YouTube search suggestions in education and cleaned the data, tagged it, and preprocessed it. The dataset provides a reliable database for subsequent data mining tasks, and ensures the reliability and reproducibility of the research results.

6. Empirical research and experimental verification

Through experiments and empirical analysis, the effectiveness and reliability of crawler and data mining methods in this scenario are verified. The experimental results were analyzed and discussed, and case analysis and conclusions were put forward, which provided important reference and guidance for subsequent research and practice.

7. Interdisciplinary research and application

We combine crawling and data mining techniques with knowledge of educational disciplines to explore the possibilities and application prospects of interdisciplinary research. Through interdisciplinary research, we are able to gain a more comprehensive understanding of the needs and challenges in the field of education, and provide new ideas and methods for educational reform and innovation. This interdisciplinary research exploration will advance the development and progress of the field of education.

8. Algorithm flowchart

We use an algorithm flowchart to make it easier for people to understand our algorithms.

9. Collecting, Analyzing, and Mining Data

9.1 Data collection

A brief description of the techniques used in this study: The principle of obtaining data is shown in the figure below:

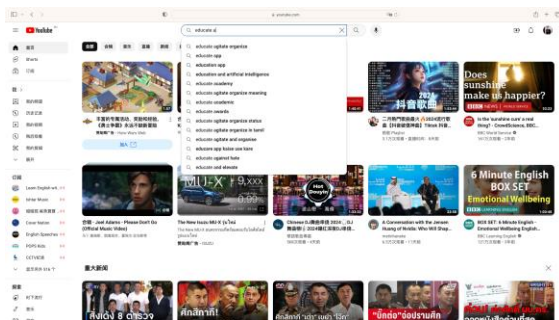


Figure 1. Search Suggested Text on YouTube.

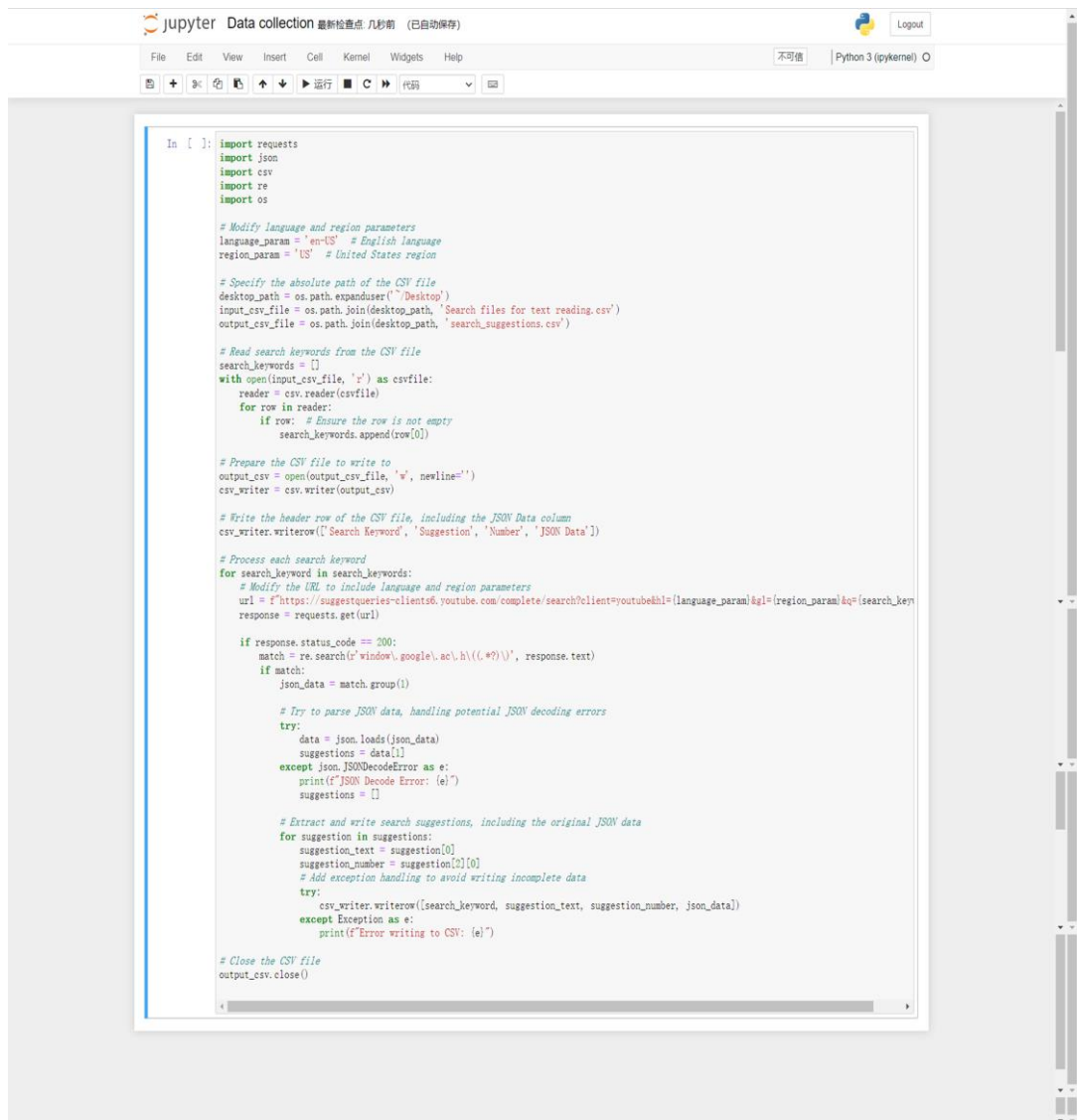
When a user enters "educate a" in the YouTube search bar, some search suggestion text will be displayed below the YouTube search bar. As shown in the picture: educate agitate organize, educate app, education app, etc. And use the 10,000 words provided by MIT.csv (<https://www.mit.edu/~ecprice/wordlist.10000>) (Of course you can also choose other word lists, such as larger word lists.) to replace the

"a" in "educate a" in the picture. " word to collect new data.

Table 1. Part of the content in the "Search files for text reading.csv" file

Search term
educate a
educate aa
educate aaa
educate aaron
educate ab
educate abandoned
educate abc
educate aberdeen
educate abilities
educate ability

There are 10,000 such phrases. Because there are 10,000 words in the file "10,000 words provided by MIT.csv". In this way, more than 50,000 search suggestion texts such as educate agitate organize, educate app, education app, etc. were collected. Use the following Python crawler code to automatically and quickly collect such search suggestion text.



```
In [ ]: import requests
import json
import csv
import re
import os

# Modify language and region parameters
language_param = 'en-US' # English language
region_param = 'US' # United States region

# Specify the absolute path of the CSV file
desktop_path = os.path.expanduser("~/Desktop")
input_csv_file = os.path.join(desktop_path, 'Search files for text reading.csv')
output_csv_file = os.path.join(desktop_path, 'search_suggestions.csv')

# Read search keywords from the CSV file
search_keywords = []
with open(input_csv_file, 'r') as csvfile:
    reader = csv.reader(csvfile)
    for row in reader:
        if row: # Ensure the row is not empty
            search_keywords.append(row[0])

# Prepare the CSV file to write to
output_csv = open(output_csv_file, 'w', newline='')
csv_writer = csv.writer(output_csv)

# Write the header row of the CSV file, including the JSON Data column
csv_writer.writerow(['Search Keyword', 'Suggestion', 'Number', 'JSON Data'])

# Process each search keyword
for search_keyword in search_keywords:
    # Modify the URL to include language and region parameters
    url = f"https://suggestqueries-clients6.youtube.com/complete/search?client=youtu&hl={language_param}&gl={region_param}&q={search_key}"
    response = requests.get(url)

    if response.status_code == 200:
        match = re.search(r'window\google\ac\h\((.*?)\)', response.text)
        if match:
            json_data = match.group(1)

            # Try to parse JSON data, handling potential JSON decoding errors
            try:
                data = json.loads(json_data)
                suggestions = data[1]
            except json.JSONDecodeError as e:
                print(f"JSON Decode Error: {e}")
                suggestions = []

            # Extract and write search suggestions, including the original JSON data
            for suggestion in suggestions:
                suggestion_text = suggestion[0]
                suggestion_number = suggestion[2][0]
                # Add exception handling to avoid writing incomplete data
                try:
                    csv_writer.writerow([search_keyword, suggestion_text, suggestion_number, json_data])
                except Exception as e:
                    print(f"Error writing to CSV: {e}")

# Close the CSV file
output_csv.close()
```

Figure 2. Python code used to obtain search suggested text on the YouTube website. The 1st picture

While crawling: This code reads the data in "Search files for text reading.csv". For details, please see Table 1. Part of the content

in the "Search files for text reading.csv" file. Then the generated output file: "search_suggestions.csv", the content of this file is as follows:

Table 2. YouTube search suggested text data obtained through a crawler

Search Keyword	Suggestion	Number	JSON Data
educate a	educate alabama	512	["educate a",["educate alabama",0,[512]],["educate agitate organize",0,[512]],["educate and celebrate",0,[512]],["educate app",0,[512]],["educate ai",0,[512]],["educate antonyms",0,[512]],["educate america act",0,[512]],["educate academy",0,[512]],["educate a child",0,[512]],["educate about or on",0,[512]],["educate anagram",0,[512]],["educate agitate organize wikipedia",0,[512]],["educate a woman you educate a nation",0,[512]],["educate adny",0,[512]]],{"k":1,"q":"uBLP_gtJHpxRMi9yglRwfD-dvuM"}]

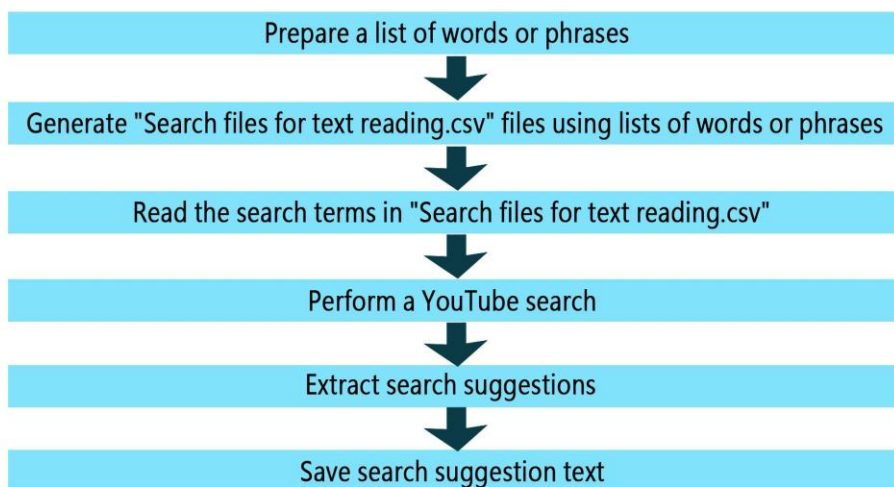


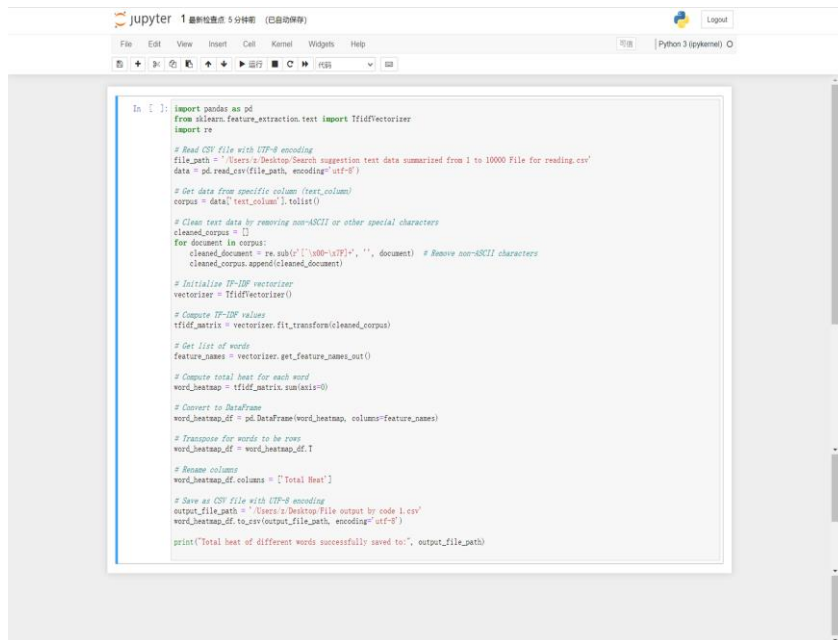
Figure 3. Algorithm flowchart for data collection

Data analysis: When we use a Python crawler to obtain search suggestion text such as educate agitate organize, educate app,

education app, etc., we use the following 8 codes to analyze the collected data from different levels or aspects.

9.2 1st code used for data analysis

Code description:



```
In [ ]: import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import re

# Read CSV file with UTF-8 encoding
file_path = 'C:/Users/1/Desktop/Search suggestion test data summarized from 1 to 10000 File for reading.csv'
data = pd.read_csv(file_path, encoding='utf-8')

# Get data from specific column (text_column)
corpus = data['text_column'].tolist()

# Clean text data by removing non-ASCII or other special characters
cleaned_corpus = []
for document in corpus:
    cleaned_document = re.sub(r'[^\x00-\x7F]+', '', document) # Remove non-ASCII characters
    cleaned_corpus.append(cleaned_document)

# Initialize TF-IDF vectorizer
vectorizer = TfidfVectorizer()

# Compute TF-IDF values
tfidf_matrix = vectorizer.fit_transform(cleaned_corpus)

# Get list of words
feature_names = vectorizer.get_feature_names_out()

# Compute total heat for each word
word_heatmap = tfidf_matrix.sum(axis=0)

# Convert to DataFrame
word_heatmap_df = pd.DataFrame(word_heatmap, columns=feature_names)

# Transpose for words to be rows
word_heatmap_df = word_heatmap_df.T

# Rename columns
word_heatmap_df.columns = ['Total Heat']

# Save as CSV file with UTF-8 encoding
output_file_path = 'C:/Users/1/Desktop/File output by code 1.csv'
word_heatmap_df.to_csv(output_file_path, encoding='utf-8')

print("Total heat of different words successfully saved to:", output_file_path)
```

Figure 4. 1st code

Analysis results: Use this code¹⁷. A total of 14878 words and corresponding "Total Heat" values were output. This is the first 10

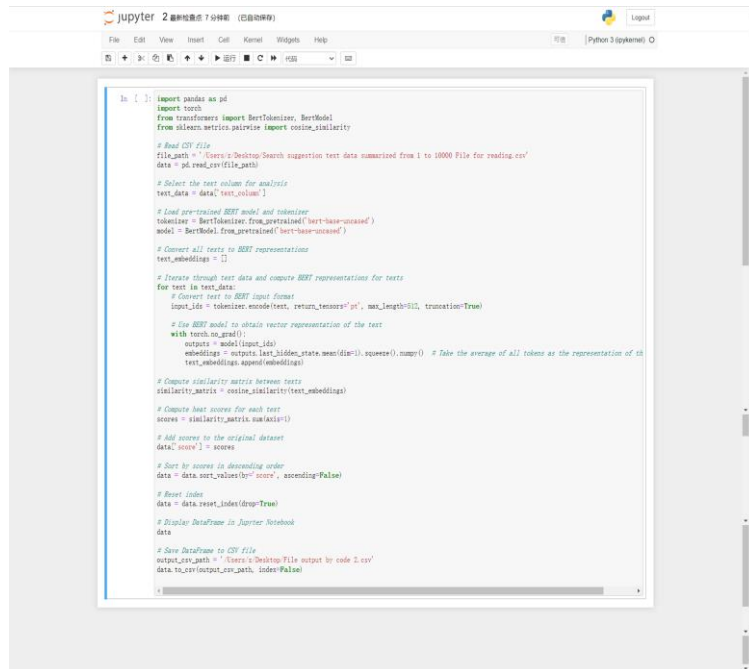
data with the largest Total Heat value output by this code.

Table 3. Part of the data output by 1st code.

Word	Total Heat
education	4263.189847
educate	2749.069812
in	1176.302716
is	1082.611081
what	984.2769601
of	665.8920437
meaning	570.5850557
fnaf	567.9055954
educating	564.8674301
for	538.6324134

Taking the seventh word "meaning" as an example, we can predict that people may want to know the meaning of education. This can help teachers and others explain the meaning of the courses they are learning to students in the course, thereby improving students' enthusiasm for learning

9.3 Second code for data analysis



```

In [ ]: import pandas as pd
import sklearn
from transformers import BertTokenizer, BertModel
from sklearn.metrics.pairwise import cosine_similarity

# Read CSV file
file_path = 'Users\\Desktop\\Search suggestion test data summarized from 1 to 10000 File for reading.csv'
data = pd.read_csv(file_path)

# Select the text column for analysis
text_data = data['text_column']

# Load pre-trained BERT model and tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')

# Convert all texts to BERT representations
text_embeddings = []

# Iterate through text data and compute BERT representations for texts
for text in text_data:
    # Convert text to BERT input format
    input_ids = tokenizer.tokenize(text, return_tensors='pt', max_length=512, truncation=True)

    # Use BERT model to obtain vector representation of the text
    with torch.no_grad():
        outputs = model(input_ids)
        embeddings = outputs.last_hidden_state.mean(dim=-1).squeeze().numpy() # Take the average of all tokens as the representation of the text
        text_embeddings.append(embeddings)

# Compute similarity matrix between texts
similarity_matrix = cosine_similarity(text_embeddings)

# Compute best scores for each text
scores = similarity_matrix.sum(axis=-1)

# Add scores to the original dataset
data['score'] = scores

# Sort by scores in descending order
data = data.sort_values(by='score', ascending=False)

# Reset index
data = data.reset_index(drop=True)

# Display DataFrame in Jupyter Notebook
data

# Save DataFrame to CSV file
output_csv_path = 'Users\\Desktop\\File output by code 2.csv'
data.to_csv(output_csv_path, index=False)

```

Figure 5. Second code

Analysis results: Use this code¹⁸. In total, the code outputs more than 50,000 lines

of data. This is the top 10 data output by this code, including the same rows.

Table 4. Part of the data output by the second code

Text column	Score
education promotion apple	38917.7
education promotion apple	38917.7
education clarity	38880.16
education teaching quotes	38879.61
education teaching quotes	38879.61
education ideas	38879.44
education ideas	38879.44
education ideas	38879.44
education ideas	38879.44
education innovations	38870.42



Taking the No. 1 “education promotion apple” as an example, we can predict that Apple Inc.’s promotion activities in the field of education are very successful and have great influence.

9.4 3rd code used for data analysis

```

In [ ]: import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.naive_bayes import MultinomialNB

        # Read CSV file
        file_path = 'D:/Users/z/Desktop/Search suggestion text data summarized from 1 to 10000 File for reading.csv'
        data = pd.read_csv(file_path)

        # Select the text column for analysis
        text_data = data['text_column']

        # Split the dataset into training and testing sets
        X_train, X_test = train_test_split(text_data, test_size=0.2, random_state=42)

        # Use TF-IDF to vectorize the text data
        tfidf_vectorizer = TfidfVectorizer(max_features=1000) # Set the maximum number of features
        X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
        X_test_tfidf = tfidf_vectorizer.transform(X_test)

        # Use Naive Bayes model for classification
        nb_classifier = MultinomialNB()
        nb_classifier.fit(X_train_tfidf, X_train) # Here, the text itself is used as the label for training

        # Get the predicted probabilities for the test set
        test_probabilities = nb_classifier.predict_proba(X_test_tfidf)

        # Rank the texts in the test set based on predicted probabilities
        ranked_tests = sorted(zip(X_test, test_probabilities[:, 1]), key=lambda x: x[1], reverse=True)

        # Create a DataFrame to save the ranking results
        ranked_df = pd.DataFrame(ranked_tests, columns=['Text', 'Probability'])

        # Save the ranking results to a CSV file
        ranked_df.to_csv('File output by code 3.csv', index=False)

```

Figure 6. 3rd code

Analysis results: Use this code¹⁹. The code outputs a total of more than 10,000 lines

of data. This is the top 10 data with the largest Probability value output by this code.

Table 5. Part of the data output by 3rd code

Text	Probability
cookies making course	3.79E-05
course navigation instructions	3.79E-05
fisheries course subjects	3.79E-05
cookies making course	3.79E-05
erase course	3.79E-05
education refresher course	3.55E-05
education aide course	3.55E-05
education supervision course	3.55E-05
education trading course	3.55E-05
embedded course meaning	3.39E-05



Taking the No. 1 “cookies making course” as an example, we can predict that many people want to learn how to make cookies through the “cookies making course”.

9.5 4th code used for data analysis

```

In [ ]: import pandas as pd
from gensim.corpora import Dictionary
from gensim.models import LdaModel
from gensim.utils import simple_preprocess
from nltk.corpus import stopwords

# Download stopwords
stop_words = stopwords.words('english')

# Read CSV file
file_path = '/Users/z/Desktop/Search suggestion text data summarized from 1 to 10000 File for reading.csv'
data = pd.read_csv(file_path)

# Text preprocessing
def preprocess_text(text):
    return [token for token in simple_preprocess(text) if token not in stop_words]

# Convert text data to processed word lists
processed_data = [preprocess_text(text) for text in data['text_column']]

# Create dictionary
dictionary = Dictionary(processed_data)

# Filter extreme words
dictionary.filter_extremes(no_below=5, no_above=0.5)

# Bag-of-words representation
corpus = [dictionary.doc2bow(doc) for doc in processed_data]

# Run LDA model
lda_model = LdaModel(corpus=corpus,
                    id2word=dictionary,
                    num_topics=500, # Set the number of topics to 500
                    passes=5) # Number of iterations

# Save the model
lda_model.save('lda_model')

# Save the dictionary
dictionary.save('dictionary')

# Generate a list of topics ranked by weight from high to low
topics_data = []
for idx in range(lda_model.num_topics):
    topic_words = lda_model.show_topic(idx)
    topic_words = sorted(topic_words, key=lambda x: x[1], reverse=True)
    topic_words_str = " ".join([f'{word[0]}: {word[1]}' for word in topic_words])
    topics_data.append([f'Topic: {idx}', f'Words: {topic_words_str}'])

# Convert the topic list to a DataFrame
topics_df = pd.DataFrame(topics_data)

# Save the DataFrame to a CSV file
output_file_path = '/Users/z/Desktop/File output by code 4.csv'
topics_df.to_csv(output_file_path, index=False)

print(f'Output file saved at: {output_file_path}')

```

Figure 7. 4th code. The 1st picture

Analysis results: Use this code²⁰⁻²¹. The code outputs data for a total of 500 subjects. This is the first data output by this code.

Table 6. Part of the data output by 4th code

Topic	Words
0	durham: 0.00023397283803205937, lifelong: 0.00023397283803205937, release: 0.00023397283803205937, lottery: 0.00023397283803205937, dvd: 0.00023397283803205937, peer: 0.00023397283803205937, para: 0.00023397283803205937, dynamics: 0.00023397283803205937, magic: 0.00023397283803205937, rep: 0.00023397283803205937



Taking the first data row ranked first among them as an example, we can predict that many people may be interested in educational content related to "magic".

9.6 Code 5 used for data analysis

```
jupyter 5 最新状态: 8 分钟前 (已自动保存)
File Edit View Insert Cell Kernel Widgets Help
Python 3 (ipykernel)

In [ ]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE
import sys

# Read CSV file
file_path = 'Users\\Desktop\\Search suggestion text data summarized from 1 to 10000 File for reading.csv'
data = pd.read_csv(file_path)

# Select the text column for analysis
text_data = data['text_column']

# Text preprocessing and feature extraction
vectorizer = TfidfVectorizer(max_df=0.5, min_df=2, stop_words='english', use_idf=True)
X = vectorizer.fit_transform(text_data)

# Clustering algorithm
num_clusters = 30 # Assume clustering into 30 clusters
kmeans = KMeans(n_clusters=num_clusters, init='k-means++', max_iter=100, n_init=1)

# Train the clustering model
kmeans.fit(X)

# Add cluster labels to the original dataframe
data['cluster_label'] = kmeans.labels_

# Output dataframe with cluster labels to CSV file
output_file_path = 'Users\\Desktop\\File output by code 5.csv'
data.to_csv(output_file_path, index=False)

# t-SNE dimensionality reduction
tsne = TSNE(n_components=2, random_state=0, init='random') # Set the initialization method to random
X_tsne = tsne.fit_transform(X.toarray()) # Convert sparse matrix to dense matrix before dimensionality reduction

# Visualize clustering results
plt.figure(figsize=(10, 8))
colors = ['b', 'g', 'r', 'c', 'm', 'k', 'y', 'orange', 'purple', 'brown', 'pink', 'lime', 'olive', 'cyan', 'indigo', 'teal', 'lavender', '']

for i in range(num_clusters):
    points = X_tsne[kmeans.labels_ == i]
    plt.scatter(points[:, 0], points[:, 1], s=30, c=colors[i], label=f'Cluster {i+1}')

plt.title('t-SNE Visualization of Text Clusters')
plt.legend()
plt.show()

# Exit program execution
sys.exit()
```

Figure 8. Code 5

Analysis results:

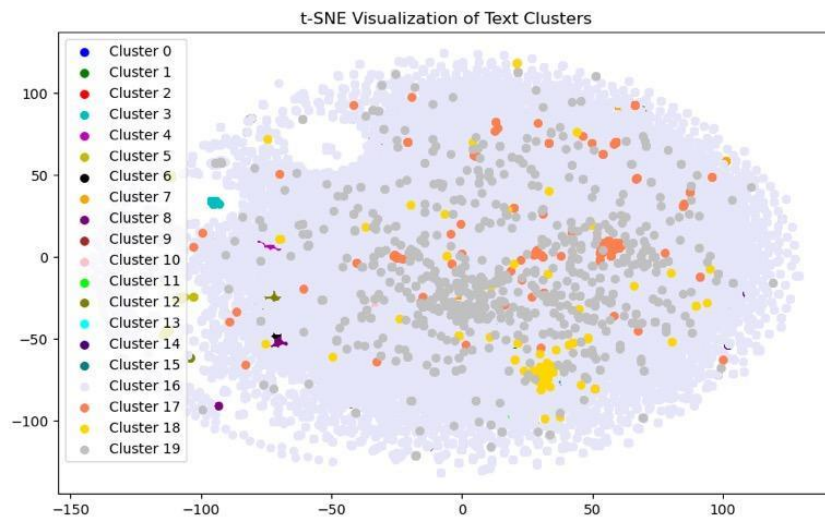
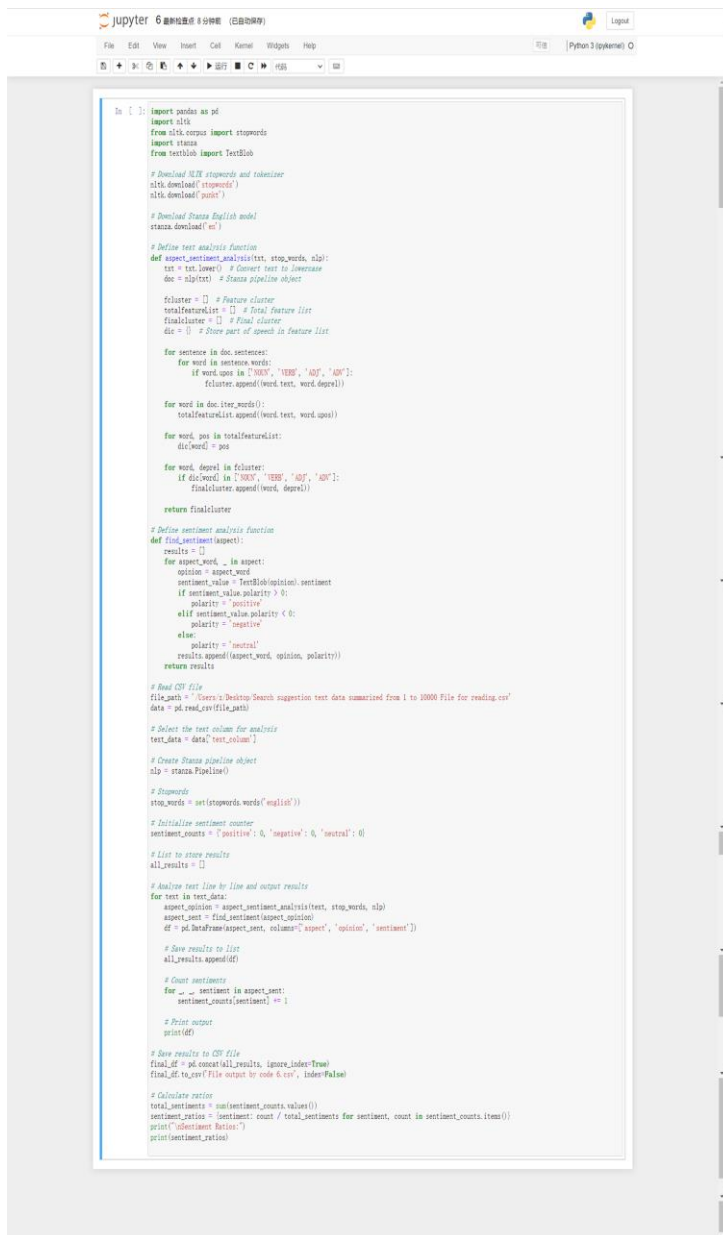


Figure 9. Picture of the output of code 5

Use this code 22-23, the data is divided into 20 categories. You can find more related data

in the same category by checking which category the relevant data you are interested.

9.7 Code 6 used for data analysis



```
In [ ]: import pandas as pd
import nltk
from nltk.corpus import stopwords
import stanza
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize

# Download NLTK stopwords and tokenizer
nltk.download('stopwords')
nltk.download('punkt')

# Download Stanza English model
stanza.download('en')

# Define text analysis function
def aspect_sentiment_analysis(text, stop_words, nlp):
    text = text.lower() # Convert text to lowercase
    doc = nlp(text) # Create pipeline object

    clusters = [] # Feature cluster
    totalfeaturelist = [] # Final feature list
    finalcluster = [] # Final cluster
    doc = [] # Store part of speech in feature list

    for sentence in doc.sentences():
        for word in sentence.words():
            if word.upos in ['NOUN', 'VERB', 'ADJ', 'ADV']:
                cluster.append(word.text, word.deprel())

    for word in doc.iter_words():
        totalfeaturelist.append(word.text, word.upos())

    for word_pos in totalfeaturelist:
        doc[word_pos] = pos

    for word_deprel in finalcluster:
        if doc[word_pos] in ['NOUN', 'VERB', 'ADJ', 'ADV']:
            finalcluster.append(word, deprel())

    return finalcluster

# Define sentiment analysis function
def find_sentiment(aspect):
    results = []
    for aspect_word, _ in aspect:
        opinion = aspect_word
        sentiment_value = TextBlob(opinion).sentiment
        if sentiment_value.polarity > 0:
            polarity = 'positive'
        elif sentiment_value.polarity < 0:
            polarity = 'negative'
        else:
            polarity = 'neutral'
        results.append(aspect_word, opinion, polarity)
    return results

# Read CSV file
file_path = 'C:/Users/Devi/Desktop/Search suggestion text data summarized from 1 to 10000 File for reading.csv'
data = pd.read_csv(file_path)

# Select the text column for analysis
test_data = data['test_column']

# Create Stanza pipeline object
nlp = stanza.Pipeline()

# Stopwords
stop_words = set(stopwords.words('english'))

# Initialize sentiment counter
sentiment_counts = {'positive': 0, 'negative': 0, 'neutral': 0}

# List to store results
all_results = []

# Analyze text line by line and output results
for text in test_data:
    aspect_opinion = aspect_sentiment_analysis(text, stop_words, nlp)
    aspect_sent = find_sentiment(aspect_opinion)
    df = pd.DataFrame(aspect_sent, columns=['aspect', 'opinion', 'sentiment'])

    # Save results to list
    all_results.append(df)

# Count sentiment
for _ in sentiment_counts:
    sentiment_counts[sentiment] += 1

# Print output
print(df)

# Save results to CSV file
final_df = pd.concat(all_results, ignore_index=True)
final_df.to_csv('File output by code 6.csv', index=False)

# Calculate ratios
total_sentiments = len(sentiment_counts.values())
sentiment_ratios = {sentiment: count / total_sentiments for sentiment, count in sentiment_counts.items()}
print('Sentiment Ratios:')
print(sentiment_ratios)
```

Figure 10. Code 6. The 1st picture

Analysis results: Use this code²⁴, the output result is {'positive': 0.02999141996374451, 'negative': 0.01191528343149841, 'neutral':

0.9580932966047571}. We can know that the results of people's sentiment analysis on education are neutral to positive. We can



learn about changes in people's sentiments
on education by collecting and analyzing data

at different time periods.

9.8 Code 7 used for data analysis

```
jupyter 7 最新检查点 9分40秒 (已自动保存)
File Edit View Insert Cell Kernel Widgets Help
Python 3 (ipykernel) O

In [ ]: import pandas as pd
import math
import csv

# Read CSV file
file_path = "Users\\x\\Desktop\\Search suggestion test data summarized from 1 to 10000 File for reading.csv"
data = pd.read_csv(file_path)

# Select the text column for analysis
text_data = data['text_column']

# Build inverted index
def build_index(documents):
    index = {}
    for doc_id, text in documents.items():
        terms = text.lower().split()
        for term in terms:
            if term not in index:
                index[term] = set()
            index[term].add(doc_id)
    return index

# Calculate cosine similarity
def cosine_similarity(vec1, vec2):
    intersection = set(vec1.keys()) & set(vec2.keys())
    numerator = sum([vec1[x] * vec2[x] for x in intersection])
    sum1 = sum([vec1[x]**2 for x in vec1.keys()])
    sum2 = sum([vec2[x]**2 for x in vec2.keys()])
    denominator = math.sqrt(sum1) * math.sqrt(sum2)

    if not denominator:
        return 0.0
    else:
        return float(numerator) / denominator

# Calculate document vector
def calculate_tf(doc):
    tf = {}
    for term in doc.split():
        tf[term] = tf.get(term, 0) + 1
    return tf

# Main function
def main():
    documents = {}
    for i, text in enumerate(text_data):
        doc_id = f'Doc{i+1}'
        documents[doc_id] = text

    index = build_index(documents)

    target_doc = 'information and communication technology for education' # Replace with your target document to search similar documents
    target_tf = calculate_tf(target_doc.lower())

    similarities = []
    for doc_id, doc_text in documents.items():
        doc_tf = calculate_tf(doc_text)
        similarity = cosine_similarity(target_tf, doc_tf)
        similarities.append((doc_id, doc_text, similarity))

    # Sort similarities from high to low
    similarities.sort(key=lambda x: x[2], reverse=True)

    # Write search results to CSV file
    with open('File output by code 7.csv', 'w', newline='') as csvfile:
        fieldnames = ['Document', 'Similarity']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

        writer.writeheader()
        for doc_text, similarity in similarities:
            writer.writerow({'Document': doc_text, 'Similarity': similarity})

    print('Similar documents to "{}":'.format(target_doc))
    for doc_text, similarity in similarities:
        print('Document "{}", Similarity: {:.2f}'.format(doc_text[:50], similarity))

if __name__ == '__main__':
    main()
```

Figure 11. Code 7. The 1st picture

Analysis results: Use this code²⁵, the
code ranks more than 50,000 lines of data by
similarity to the text 'information and

communication technology for education'.
This is the top 10 data with the largest
Similarity value output by this code:

Table 7. Part of the data output by code 7

Document	Similarity
education communication and technology	0.816496581
education communication and technology notes	0.730296743
education information technology	0.707106781
education information technology	0.707106781
education information technology	0.707106781
education information technology	0.707106781
education communication & information	0.612372436
education information technology pdf	0.612372436
education information technology pdf	0.612372436
education information technology journal	0.612372436

Through this method, we can quickly find the collected search recommendation text data that is highly relevant to the topic you are interested in.



9.9 Code 8 used for data analysis

```
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import DataLoader, Dataset
from sklearn.metrics import confusion_matrix

# Load CSV file
file_path = "C:/Users/1/Desktop/Source suggestion test data summarized from 1 to 10000 File for reading.csv"
data = pd.read_csv(file_path)

# Select the test column for analysis
test_data = data['test_column']

# Convert test to bag-of-words vector representation using CountVec
vectorizer = CountVec()
X_train = vectorizer.fit_transform(test_data)
X_test = X_train

# Custom dataset class
class CustomDataset(Dataset):
    def __init__(self, data):
        self.data = data

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        return torch.tensor(self.data[idx], dtype=torch.float32) # Convert to tensor

# Define Generator model
class Generator(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(Generator, self).__init__()
        self.relu = nn.ReLU()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        return self.relu(self.fc1(x))

# Define Discriminator model
class Discriminator(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(Discriminator, self).__init__()
        self.relu = nn.ReLU()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        return self.relu(self.fc1(x))

# Define GAN model
class GAN(nn.Module):
    def __init__(self, input_dim, hidden_dim):
        super(GAN, self).__init__()
        self.generator = Generator(input_dim, hidden_dim, input_dim)
        self.discriminator = Discriminator(input_dim, hidden_dim, 1)

        self.optimizer_G = optim.Adam(self.generator.parameters())
        self.optimizer_D = optim.Adam(self.discriminator.parameters())
        self.reconstruction_loss = nn.MSELoss(reduction='none')

    def forward(self, x):
        return self.generator(x)

    def train(self, data_loader, num_epochs=10, batch_size=32):
        for epoch in range(num_epochs):
            for i, data in enumerate(data_loader):
                self.optimizer_G.zero_grad()
                real_data = data
                batch_size = real_data.size(0) # Get current batch size
                fake_data = self.generator(torch.randn(batch_size, self.input_dim))
                real_labels = torch.ones(batch_size, 1)
                fake_labels = torch.zeros(batch_size, 1)

                # Calculate discriminator loss
                disc_real_loss = nn.MSELoss(self.discriminator(real_data), real_labels)
                disc_fake_loss = nn.MSELoss(self.discriminator(fake_data.detach()), fake_labels)
                disc_loss = disc_real_loss + disc_fake_loss
                self.optimizer_D.zero_grad()
                disc_loss.backward()
                self.optimizer_D.step()

                # Calculate generator loss
                self.optimizer_G.zero_grad()
                fake_data = self.generator(torch.randn(batch_size, self.input_dim)) # Re-generate fake data to match current batch size
                gen_loss = nn.MSELoss(self.discriminator(fake_data), real_labels)
                self.optimizer_G.zero_grad()
                gen_loss.backward()
                self.optimizer_G.step()

            # Train reconstruction error during training
            reconstruction_error = self.reconstruction_loss(real_data, self.generator(real_data))

        def detect_anomalies(self, test_data):
            # Calculate reconstruction error on the test set
            test_tensor = torch.tensor(test_data, dtype=torch.float32)
            reconstructed_data = self.generator(test_tensor)
            reconstruction_errors = torch.mean(test_tensor - reconstructed_data**2, dim=0)

            # Compute average the mean plus three times standard deviation as anomaly threshold based on reconstruction error
            threshold = reconstruction_errors.mean() + 3 * reconstruction_errors.std()

            return anomalies

# Create data loader
dataset = CustomDataset(Dataset, batch_size=32, shuffle=True)

# Define and train the model
input_dim = 1, hidden_dim=10 # Determine input dimension based on bag-of-words vector dimension
hidden_dim = 10
generator_model = Generator(input_dim, hidden_dim)
discriminator_model = Discriminator(input_dim, hidden_dim, num_epochs=10)

# Test data
test_data = X_test # Convert test data to list

# Perform anomaly detection
anomalies = generator_model.detect_anomalies(test_data)

# Create DataFrame
result_df = pd.DataFrame({'test': test_data, 'anomaly': anomalies})

# Save anomaly detection results to a CSV file
result_df.to_csv('File output to save 8.csv', index=False)
```

Figure 12. Code 8. The 1st picture

Analysis results: Use this code²⁶⁻²⁸, we screened 55,945 rows of data, and 818 rows

of data were identified as anomalies, 10 of which are as follows:

Table 8. Part of the data output by code 8

Text	Anomaly
how to teach the abc to preschoolers	TRUE
how long can you be absent from school	TRUE
what happens if your child is absent from school	TRUE
education achievement of rural population of uttar pradesh	TRUE
education achievement of rural population of uttar pradesh	TRUE
education action plan for the belt and road initiative	TRUE
education activists who is the youngest nobel prize laureate crossword	TRUE
who is the girl in the education connection commercial	TRUE
adam and eve how did adam die	TRUE
how old was adam when he left eden	TRUE

These TRUE outputs indicate that these sentences can be identified as anomalies in the text data. In this case, "abnormal" may mean that the sentences are different from other sentences, may contain some special keywords, topics, or grammatical structures, or they may not conform to the normal pattern of the dataset. In textual data, an anomaly often indicates something different or unusual from the rest of the text that may require special handling or attention.

These sentences are different from other parts of the text in that they introduce a different topic or issue, or their grammatical structure differs from other sentences.

The GANomaly²⁹ model may find that these sentences differ from other sentences in terms of vocabulary, grammar, or subject matter, and therefore mark them as exceptions. In practice, these anomalies can be further investigated to understand their causes and what they mean in the dataset.

We can also use the output results of two or more of the above eight codes at the same time to analyze and get the results. For example, use code 7 to find the search recommendation text related to 'information and communication technology for education', and then Find the popularity of search recommendation text related to 'information and communication technology for education' in the output file of code 2.

10. Data mining methods using CRISP-DM

This Part follows the CRISP-DM (Cross-Industry Standard Process for Data Mining)³⁰ framework and outlines the data mining methods used in this study. CRISP-DM provides a structured approach to data mining projects³¹. including six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment.

10.1 Business understanding

In this study, we invented and proposed a new algorithm to get the text of search suggestions on the web. We then use this new algorithm to analyze and mine education-related data to extract insights and inform educational research and practice.

10.2 Data understanding

The data understanding phase involves acquiring, exploring, and assessing the quality of the data set. This study uses a Python crawler to collect education-related search suggestion text from YouTube and evaluates the relevance and completeness of the dataset.

10.3 Data Preparation

Data preparation involves cleaning, transforming, and integrating datasets to make them suitable for analysis³². In this study, the search suggestion text was preprocessed to address issues such as noise, missing values, and data inconsistencies.

10.4 Modeling

In the modeling phase, the study applied various data mining techniques to extract patterns and insights from the prepared dataset³³. Algorithms such as machine learning including clustering, classification and sentiment analysis were used to analyze trends and themes related to education.

10.5 Evaluation

The objective of the evaluation stage is to verify the performance and validity of the data mining model³⁴. Using web scraping, data analysis, and mining methods, this study has made significant progress in predicting education-related trends and emotions. Through these methods, we found many valuable insights that further demonstrated the validity of the model.

10.6 Deployment

The deployment phase consists of assembling data mining results into actionable insights for stakeholders³⁵. The study presents findings and recommendations for data mining analysis, highlighting opportunities for improvement in educational research, policy development, and practice.

Results and discussion

This part presents the results and discussions of data mining analysis using the CRISP-DM method. These results provide valuable insights into education-related trends, topics, and sentiments extracted from search recommendation texts collected on YouTube.

We used in our code: TF-IDF analysis, BERT text embedding and similarity calculation, Naive Bayes text classification, LDA topic modeling, K-means text clustering and t-SNE visualization, Aspect-Based Sentiment Analysis, Technologies such as inverted index and cosine similarity calculation, GANomaly anomaly detection, etc. Regarding the 8 data analysis and mining codes, each code is generally independent of each other, but the results output by the code are used for comprehensive analysis.

1. Trends and patterns

The analysis reveals top trends and suggested text patterns for education-related searches, including top topics, keywords, and sentiment trends. The study identifies key themes and areas of concern for the education community.

2. Sentiment analysis

Sentiment analysis of search suggestion text reveals the emotional tone and attitudes expressed toward education-related topics³⁶. The study examined positive, negative and neutral sentiments to assess public perceptions and opinions. We concluded that people's emotions towards education are Neutral to positive.

3. Topic clustering

Subject clustering technology groups similar search suggestion texts based on semantic similarity³⁷. This study identifies

clusters of related topics and can use text from different clusters to find more text that you might be interested in, to find information that people might not have known before, and so on.

4. Outlier analysis

In addition, this study also performed outlier analysis to identify abnormal patterns or extreme values in the data set³⁸. Outlier analysis helps to understand abnormal or abnormal behaviors in the data set, providing a more accurate basis for subsequent analysis. These may be Data that people don't often pay attention to but have a certain degree of popularity. Analyzing it may yield unexpected results.

Conclusions

We conclude the main findings and implications of the data mining analysis of education-related search suggestion text. This part also outlines the possibilities for using the algorithms and methods of this study in future educational research and applications, as well as in other industries.

The results reveal educational trends, themes, and sentiments reflected in the suggested search texts. The analysis provides valuable insights for education researchers, policy makers and practitioners. In this paper, we invent and propose a new algorithm to obtain search recommendation text on the Internet. The technology can be

used to obtain search recommendation text in any field on the Internet. At present, it is the first method and algorithm in the world that can quickly and automatically obtain the recommended text of internet search. Moreover, this study is also the first international data analysis and mining of Internet search recommended texts.

Insights gained from data mining analytics have had a significant impact on education, including curriculum development, instructional practices, and educational policymaking. This study highlights further opportunities to improve educational outcomes using innovative data-driven approaches. Leveraging data creates more new possibilities.

Recommendations

For future research, we can gather and examine this data at various intervals to compare and observe how data analysis and mining outcomes evolve over time. This method allows us to study hotspots and shifts in people's perspectives. We can utilize the developed method and algorithm and suggest in the study to generate search suggestions on the Internet for research in various domains or subdivisions. It is utilized to gather data from various fields or segmentation directions for the purpose of data analysis and data mining. For instance, in several educational sectors like computer

education, mathematics education and English education, or in diverse businesses such as entertainment, education, medical care, politics, etc. This study achieved a world-first by developing a method for quickly and automatically collecting a new type of data - search recommendation text from the Internet. This successfully fosters the advancement of web crawlers' data analysis and mining technologies. Introducing additional options and potential to various aspects of life. Any to increase efficiency.

References

- 1 Higgins S, Xiao ZM, Katsipataki M. The impact of digital technology on learning: A summary for the education endowment foundation. Education endowment foundation; 2012.
- 2 Lin MH, Chen HC, Liu KS. A study of the effects of digital learning on learning motivation and learning outcome. *Eurasia J Math Sci Tech Ed* 2017;13:3553-64. doi:10.12973/eurasia.2017.00744a.
- 3 Picciano AG. The evolution of big data and learning analytics in American higher education. *JALN* 2012;16:9-20.
- 4 Romero C, Ventura S. Educational data mining: a review of the state of the art. *IEEE* 2010;40:601-18. doi:10.1109/TSMCC.2010.2053532.

- 5 Baker RS. Data mining for education. In International Encyclopedia of Education (3rd edition). Oxford, UK:2012. p.112-18.
- 6 Wang F, Hannafin MJ. Design-based research and technology-enhanced learning environments. ETR&D 2005;53:5-23.
- 7 Baker RS, Yacef K. The state of educational data mining in 2009: A review and future visions. IEDMS 2009;1:3-17.
- 8 Pardos ZA, Heffernan NT. KT-IDEM: Introducing item difficulty to the knowledge tracing model. In: Proceedings of the 19th International conference on user modeling, adaptation and personalization. Girona, Spain: 2011. P.105-19.
- 9 Brin S, Page L. The anatomy of a large-scale hypertextual web search engine. Computer science department. CA: Stanford University; 1998.
- 10 Heydon A, Najork M. Mercator: A Scalable, Extensible web crawler. In Proceedings of the 7th International world wide web conference (WWW7); 1998 April 14-18: Brisbane. Australia: 1999. p.287-301.
- 11 Fayyad U, Piatetsky-Shapiro G, Smyth P. From data mining to knowledge discovery in databases. AI magazine 1996;17:37. <https://doi.org/10.1609/aimag.v17i3.1230>
- 12 Chakrabarti S, Van den Berg M, Dom B. Focused crawling: A new approach to topic-specific web resource discovery. CN 2000.31:1623-40. doi:10.1016/S1389-1286(99)00052-3.
- 13 Han J, Kamber M, Fan M, Meng X. Data mining: concepts and techniques. China machine press. 2001.
- 14 Guyon I, Elisseeff A. An Introduction to variable and feature selection. JMLR 2003;3:1157-82.
- 15 Halevy A, Norvig P, Pereira F. The unreasonable effectiveness of data. IEEE. 2009;8:1541-672. doi:10.1109/MIS.2009.103
- 16 Dave K, Lawrence S, Pennock DM. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In: Proceedings of the 12th International world wide web conference (WWW 2003); 2003 May 20-24: Budapest, Hungary: 2003. p. 519-28.
- 17 Jones KS. A statistical interpretation of term specificity and its application in retrieval. Journal of documentation 1972;28:11-21.
- 18 Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: advances in neural information processing systems:

- 2001 November 21, Deli, India: pp. 5998-6008. 2001.
- 19 Bayes T. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions* 1993; 30:165-178.
- 20 Blei DM, Ng AY, Jordan MI. Latent dirichlet allocation. *JMLR* 2003;3:993-1022.
- 21 Hoffman MD, Bach FR, Blei DM. Online learning for latent dirichlet allocation. In: *advances in neural Information processing systems*: 2001 November 21, Deli, India. pp. 856-64. 2010.
- 22 MacQueen JB. Some methods for classification and analysis of multivariate observations. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*; 1965 January 7 : University of California. 1967. p. 281-297.
- 23 van der Maaten L, Hinton G. Visualizing data using t-SNE. *008;9:2579-605*.
- 24 Xue W, Li T. Aspect based sentiment analysis with gated convolutional networks. *arXiv preprint arXiv:1805.07043*.2018.
- 25 Williams HE, Zobel J, Bahle D. Fast phrase querying with combined indexes. *TOIS* 2004;22:573–94. <https://doi.org/10.1145/1028099.1028102>
- 26 Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An imperative style, high-performance deep learning library. *arXiv:1912.01703* 2019;10 :1-12.
- 27 Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*. 2009;4:1-58.
- 28 Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. *arXiv:1912.01703* 2014;13 :1-12.
- 29 Schlegl T, Seeböck P, Waldstein SM, Schmidt-Erfurth U, Langs G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: *International conference on information processing in medical imaging*; 2023 June 18-23: San Carlos De Bariloche: 2023. p. 146-157.
- 30 Hand DJ, Mannila H, Smyth P. *Principles of data mining*. MIT Press. 2001.
- 31 Witten IH, Frank E. *CRISP-DM: towards a standard process model for data mining*. In: *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*; 2000 April 11-13: Manchester: 2000. p. 29-40.
- 32 Batini C, Cappiello C, Francalanci C, Maurino A. Methodologies for data quality assessment and improvement. *ACM computing surveys (CSUR)* 2009;41:1-52.



- 33 Bishop CM. Pattern recognition and machine learning. Springer Link. Springer; 2006.
- 34 Fawcett T. An introduction to ROC analysis. Pattern recognition letters 2006;27:861-74.
- 35 Chen M, Mao S, Liu Y. Big data: A survey. IEEE access 2014;2:652-87.
- 36 Pang B, Lee L. Opinion mining and sentiment analysis. Foundations and trends® in information retrieval 2008;2:1-135.
- 37 Steinbach M, Karypis G, Kumar V. A comparison of document clustering techniques. In: Computer Science & Engineering (CS&E) Technical Reports: 2000 May 23: University of Minnesota: 2000. p. 1-20.
- 38 Knorr EM, Ng RT. Algorithms for mining distance-based outliers in large datasets. In: Proceedings of the 24th international conference on very large data bases; 1998 August 24: New York: 1998. p. 392-403.