News Retrieval and Multimedia Exploration System

Smart Systems

Faculty of Computer and Data Sciences, Alexandria University.

Authored by: Ahmed Ehab Abdelhalem
ID:2203147
Authored by: Youssef Mohamed Helmy
ID:2203163

Libraries

```python
import feedparser
import re
from translate import Translator
from gtts import gTTS
import time
from pytube import YouTube
from pytube import Search
import pygame
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel

```

1. **Feedparser: Used for parsing RSS and Atom feeds**
    - **We used it to parse RSS from sky news**
2. **Re: Stands for regular expressions, a powerful tool for text manipulation**
    - **We used it to search in feeds of sky news**
3. **Translate: Used for translating texts**
    - **We used it to translate news in sky news feeds**
4. **Gtts (Google Text-to-Speech): Converts text into spoken words using Google Text-to-Speech API.**
    - **We used it to converts news into spoken words**
5. **Time: A standard Python module for working with time-related functions.**
    - **We used it to set time for spoken words**
6. **Pytube: A library for downloading YouTube videos**
    - **We used it to download videos about our news**
7. **Pygame: Used for multimedia applications.**
    - **We used it to load and play voice records of gtts**
8. **Sklearn (TfidfVectorizer): Used for applying TF-IDF method in natural language processing**
    - **it helped us in recommendation system because it find unique words and turn it to victors**
9. **Sklearn (linear_kernel): Used for applying linear kernel method which calculates dot product**
    - **it helped us in recommendation system because it takes word victors and find cosine similarity of this vectors**

Functions

```python
def get_news_feeds(feedURL):
    feeds = feedparser.parse(feedURL)
    news_list = []
    for line in feeds.entries:
        news_list.append({
            'title': line.title,
            'description': line.description
        })
    return news_list

```

This function takes string (feedURL) we use that URL to get feeds using parse(), then create empty list to save news in it, then make for loop to check for titles and descriptions in RSS and get it and append it in our list, then we get this list

```python
def recommend_news(user_input, news_list):
    news_corpus = [f"{item['title']} {item['description']}" for item in news_list]

    tfidf_vectorizer = TfidfVectorizer(stop_words='english')

    tfidf_matrix = tfidf_vectorizer.fit_transform(news_corpus)

    user_tfidf = tfidf_vectorizer.transform([user_input])

    cosine_similarities = linear_kernel(user_tfidf, tfidf_matrix).flatten()

    top_indices = cosine_similarities.argsort()[:-6:-1]

    recommendations = [news_list[i] for i in top_indices]
    return recommendations

```

This function takes input from user and our news list, then it initializes our corpus using combinations of titles and descriptions in our list, then we set our TF-IDF function to use English as a language then we will turn our unique words and user input to vectors using TF-IDF method

Note: there is simple example explain how TF-IDF method works

Example

- Sen 1 : good boy
- Sen 2 : good girl
- Sen 3 : boy gilr good
- First we calculate TF → $TF = \frac{\text{No.of rep in a sen}}{\text{No.of words in a sen}}$
- $IDF = \log\left(\frac{\text{no. of } sentances}{\text{No.of sentences containing words}}\right)$

	sen1	sen2	sen3
Good	1/2	1/2	1/3
Boy	1/2	0	1/3
girl	0	1/2	1/3

X

words	IDF
Good	Log(3/3)=0
Boy	Log(3/2)
girl	Log(3/2)

	good	boy	girl	output
Sen 1	0	½*log(3/2)	0	
Sen 2	0	0	½*log(3/2)	
Sen 3	0	1/3*log(3/2)	1/3*log(3/2)	

Sen 1, Sen2, Sen3 are our corpus
First: we apply TF to every word in every sen and insert outputs in matrix, outputs with higher value are more important to specific sen
Second: we apply IDF to every unique word in corpus and insert outputs in matrix, outputs with higher value are more unique in corpus
Finally: we multiply the two matrices and the new matrix is a matrix of word vectors

Returning to our main function the best way to check similarity between user input vector and news words vectors is cosine similarity so from machine learning methods we used linear kernel which is essentially the same as the cosine similarity, it's a simple dot product of the vectors that calculates the dot product between the TF-IDF vector of the user's input and each news item in the TF-IDF matrix, resulting in a measure of similarity. The higher the value, the more similar the user input is to a particular news item, then flatten() to make results in one dimensional array

Then, we sort our top 5 recommendations news from with highest values and update our news list

```python
def download_video(youtube_url, output_path):
    try:
        youtube_object = YouTube(youtube_url)
        youtube_stream = youtube_object.streams.get_highest_resolution()
        if youtube_stream:
            youtube_stream.download(output_path)
            print("Video downloaded successfully.")
        else:
            print("No suitable streams found for the video.")
    except Exception as e:
        print(f"Error downloading video: {e}")

```

This function takes YouTube URL and path to download video in it, it's main idea of it to download selected video from YouTube so it search for YouTube video and get highest resolution from it and if that done it download video in selected path and if there was an unexpected error it print an error message

```python
def NewsDownloader(lang, feedURL, WordToSearch):
    pygame.mixer.init()
    translator = Translator(to_lang=lang)

    feeds = feedparser.parse(feedURL)

    for line in feeds.entries:
        if re.search(WordToSearch, line.description, re.IGNORECASE) or re.search(WordToSearch, line.title, re.IGNORECASE):

            translated_title = translator.translate(line.title)
            title_tts = gTTS(translated_title, lang=lang)
            title_tts_file = f"{line.title[:10]}_title.mp3"
            title_tts.save(title_tts_file)
            print("Newspaper title:",line.title)
            pygame.mixer.music.load(title_tts_file)
            pygame.mixer.music.play()
            time.sleep(10)

            translated_desc = translator.translate(line.description)
            desc_tts = gTTS(translated_desc, lang=lang)
            desc_tts_file = f"{line.title[:10]}_description.mp3"
            desc_tts.save(desc_tts_file)
            print("Newspaper description:\n",line.description)
            pygame.mixer.music.load(desc_tts_file)
            pygame.mixer.music.play()
            time.sleep(10)

```

This is the function that search and display our news from sky news

First: it take feed URL and language name and input word then, we initialize our player and translator using selected language

Second: we search in feeds about the input word in descriptions or titles in feeds.

- **Translate the title using translator**
- **Read title with spoken words using (Google Text-to-Speech)**
- **Save voice recode to mp3 file**
- **Print feed title**
- **Load and play our mp3 record of reading title**
- **Set 10 sec to talk**
- **Do the same thing to description of feed**

```
            search_query = line.title
            search_results = Search(search_query)

            if search_results.results:
                first_video = search_results.results[0]
                print("Title:", first_video.title)
                print("URL:", first_video.watch_url)

                # Download video
                output_path = f"{line.title[:10]}_video.mp4"
                download_video(first_video.watch_url, output_path)

            else:
                print("No videos found for the search query.")

            print("=" * 60)
```

Third: take title line and search in YouTube for video using it, if search is done we will get first result and print video title and URL, then download and save video in the input path if there is no results print message

```
if __name__ == "__main__":

    feed_urls = {
        '1': "https://feeds.skynews.com/feeds/rss/home.xml",
        '2': "https://feeds.skynews.com/feeds/rss/uk.xml",
        '3': "https://feeds.skynews.com/feeds/rss/world.xml",
        '4': "https://feeds.skynews.com/feeds/rss/us.xml",
        '5': "https://feeds.skynews.com/feeds/rss/business.xml",
        '6': "https://feeds.skynews.com/feeds/rss/politics.xml",
        '7': "https://feeds.skynews.com/feeds/rss/technology.xml",
        '8': "https://feeds.skynews.com/feeds/rss/entertainment.xml",
        '9': "https://feeds.skynews.com/feeds/rss/strange.xml",
    }

    while True:
        print("Categories:")
        print("1. Home")
        print("2. UK")
        print("3. World")
        print("4. US")
        print("5. Business")
        print("6. Politics")
        print("7. Technology")
        print("8. Entertainment")
        print("9. Strange")
        print("0. Exit")

```

This is feeds links that we will use and we will make menu to select feed category

```
        option = input("Enter category to search about using feeds.skynews.com: ")

        if option == '0':
            break  # Exit the loop

        feed_url = feed_urls.get(option)

        if feed_url:
            user_input = input("Enter keywords to search for: ")

            news_list = get_news_feeds(feed_url)
            recommendations = recommend_news(user_input, news_list)

            print("\nTop 5 Recommended News:")
            for i, news in enumerate(recommendations, 1):
                print(f"{i}. Title: {news['title']}")

            choose = input("Enter a keyword from titles to search for: ")
            NewsDownloader("ar", feed_url, choose)
        else:
            print("Enter a valid number, please try again")
```

Our input will get one of feeds URL and we will use our previous functions to get feed URL and search for recommendations and get news in spoken words and videos