ui1.py

MIT License

Copyright 2023 auto_anki

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

from tkinter.ttk import Progressbar

```
Function for processing file
Processes the file (File's path)
and generates c_count number of cards
:param file: String representing file path
:param c_count: Input number of anki cards
```

when testing use searchquery[:10 or less]. Still working on better threading to get faster results

finish_callback()

```
import os
from user_cli import *
import sys
import gpt_prompting as gp
import gpt4 as gp4
from tkinter import messagebox
```

```
import json
import random
from docx2pdf import convert

from flask import Flask, render_template, request, jsonify, session, send_file

sys.path.append(
    '/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages')


def process_(file,c_count):
```

```
    print("Processing",file)
    try:
        if file:
            lect_name = os.path.basename(file).split(".")[0]

            if file.split("/")[-1].split(".")[1] == "pdf":
                pass
            elif file.split("/")[-1].split(".")[1] == "docx":
                convert(file,os.path.join("uploads",lect_name+'.pdf'))
                file = file[:-5] + ".pdf"

            raw_data = extract_words(file)
            raw_data = text_to_groupings(raw_data)
            keyword_data = wp.extract_noun_chunks(raw_data)
            keyword_data = wp.merge_slide_with_same_headers(keyword_data)

            keyword_data = wp.duplicate_word_removal(keyword_data)
            search_query = wp.construct_search_query(
                keyword_data)

            source_choice = request.form['source']

            if source_choice == "Google":
                with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:


                    results = executor.map(
                        get_people_also_ask_links, search_query[:c_count])
            elif source_choice == "GPT":
                with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:
                    results = executor.map(gp.get_gpt_answers, search_query[:c_count])

            results_new = [qapair for result in results for qapair in result]
            print(len(results_new),c_count)
            if len(results_new) > int(c_count):
                results_new = random.sample(results_new, int(c_count))
            auto_anki_model = get_model()
            deck = get_deck(deck_name=lect_name)
            for qapair in results_new:
                question = qapair["Question"]
                answer = qapair["Answer"]
                qa = add_question(
                    question=f'{question}', answer=f'{answer}', curr_model=auto_anki_model)
                deck.add_note(qa)
            add_package(deck, lect_name)

    except Exception as e:
        print("file process_ Error", str(e))
        messagebox.showerror("file process_ Error", str(e))
```

Function for processing url

```
Function for processing url
Processes the url (web url)
and generates c_count number of cards
:param url: String representing URL path
:param c_count: Input number of anki cards
```

```
def process_url(url,c_count):


    print("processing url", url)

    try:
        results = gp4.get_gpt_link_answers(url, c_count)
        results_json = results.replace("'", '"')
        results_list = json.loads(results_json)
        auto_anki_model = get_model()
        lect_name = url.split("/")[-1]
        deck = get_deck(deck_name=lect_name)
        for result in results_list:
            question = result["Question"]
            answer = result["Answer"]
            qa = add_question(
                question=f'{question}',
                answer=f'{answer}',
                curr_model=auto_anki_model)
            deck.add_note(qa)
        add_package(deck, lect_name)

    except Exception as e:
        print("process_url error", str(e))
        messagebox.showerror("process_url Error", str(e))


current_filename = None
```

Configuring Flask App

```
app = Flask(__name__)
app.config['SECRET_KEY'] = os.urandom(24)
```

Returns a new status object to caller

```
def new_status():
    return {'message':'Ready','flag':False}
```

```
Renders index.html page in frontend
returns: index.html
```

```
@app.route('/')
def index():


    return render_template('index.html',status_label=new_status())
```

```
API endpoint to process file
returns: dict(status)
```

```
@app.route('/upload/file', methods=['POST'])
def upload_file():


    try:
        status_label = session.get('status_label', new_status())
        global current_filename
```

Check if ‘file’ is present in the request

```
        if request.files['file']:
            c_count = 5
            if request.form['file_value']:
```

Get count of cards within range 5 to 100

```
                c_count = min(max(int(request.form['file_value']),5),100)
```

Process the file

```
            file = request.files['file']
            current_filename = file.filename
            status_label['message'], status_label['flag'] = "Processing file...", False
```

Upload the file to server

```
            upload_path = os.path.join("uploads", file.filename)
            if not os.path.exists("uploads"):
                os.makedirs("uploads")
            file.save(upload_path)
```

call process_ to process file

```
            process_(os.path.join("uploads", file.filename),c_count)
            status_label['message'], status_label['flag'] = "File processed successfully!", True

        session['status_label'] = status_label

        return jsonify(status_label)

    except Exception as e:
        print("Upload Error", str(e))
        return jsonify(status_label)
```

```
API endpoint to process URL
returns: dict(status)
```

```
@app.route('/upload/url', methods=['POST'])
def upload_url():


    try:
        global current_filename
        status_label = session.get('status_label', new_status())
```

Check if ‘url’ is present in the request

```
        if request.form['url']:
            c_count = 5
            if request.form['url_value']:
                c_count = min(max(int(request.form['url_value']),5),100)
```

Process URL

```
            url = request.form['url']
            current_filename=url.split("/")[-1]
            status_label['message'], status_label['flag'] = "Processing URL...", False
            process_url(url,c_count)
            status_label['message'], status_label['flag'] = "URL processed successfully!", True

        session['status_label'] = status_label
```

```
        return jsonify(status_label)
    except Exception as e:
        print("Upload URL Error", str(e))
        return jsonify(status_label)
```

Use session.get to get the user-specific status_label

```
@app.route('/api/status')
def api_get_status():

    status_label = session.get('status_label', new_status())
    return jsonify(status_label)
```

Resets api status with status
Used when page is visited first time

```
@app.route('/api/refresh')
def api_refresh_status():

    session['status_label'] = new_status()
    return jsonify(session['status_label'])
```

Downloads the generated anki file locally from server

```
@app.route('/download')
def download_apkg():

    global current_filename
    if current_filename:
        current_filename = current_filename.split("/")[-1]
        current_filename = current_filename.split(".")[0]+".apkg"
        file_path = os.path.join(
            "anki_decks", current_filename)

        return send_file(file_path, as_attachment=True, download_name=current_filename)

    else:
        return "No file uploaded", 400
```

Set cache control headers

```
if __name__ == '__main__':

    app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 0
    app.run(host ='0.0.0.0', port = 5000, debug = True)
```