تقرير عن المشروع

المطوِر: أمزرت محمد أنور

رابط Github: رابط

مقدمة:

في هذا المشروع، تم تطوير تطبيق ويب يتيح للمستخدمين تحميل ملفات PDF، البحث عن عبارات محددة داخلها، والحصول على ملخص لمحتويات هذه الملفات. يعتمد التطبيق على كل من **Flask** لتشغيل الخادم الجانبي (Back-end) و**Node.js** للتفاعل مع واجهة المستخدم.

الأجزاء الرئيسية للمشروع

1. **إعداد Flask:**
   - استخدمت مكتبة **Flask** إنشاء تطبيق ويب يدعم تحميل ملفات PDF وعمليات البحث.
   - تم إعداد مسار لتحميل الملفات، حيث يتحقق من نوع الملف (PDF) ويحفظه في مجلد محدد.

```python
@app.route('/upload', methods=['POST'])
def upload_file():
    if 'pdf_file' not in request.files:
        return jsonify({"error": "No file part"}), 400
    file = request.files['pdf_file']
    if file.filename == '':
        return jsonify({"error": "No selected file"}), 400
    if not file.filename.endswith('.pdf'):
        return jsonify({"error": "File must be a PDF"}), 400

    file_path = os.path.join(UPLOAD_FOLDER, "yourfile.pdf")
    file.save(file_path)
    print(f"File uploaded and saved as: {file_path}")
    return jsonify({"message": "File uploaded successfully!"})
```

2. **بحث في محتوى PDF:**
   - استخدمت مكتبة **Flask** إنشاء تطبيق ويب يدعم تحميل ملفات PDF وعمليات البحث.
   - تم تسجيل عدد مرات ظهور العبارات، بالإضافة إلى الصفحات وأرقام الأسطر التي تم العثور عليها فيها مع الأخذ بعين الاعتبار التركيبة الخاصة باللغتين الفرنسية والانجليزية.

```python
def searchInPDF_english(pdf_path, key):
    occurrences = 0
    pages_with_lines = {}

    doc = fitz.open(pdf_path)
    for pno in range(len(doc)):
        page = doc[pno]
        content = page.get_text("text")
        lines = content.splitlines()

        for line_number, line in enumerate(lines):
            if key.lower() in line.lower():
                occurrences += line.lower().count(key.lower())

                if pno + 1 not in pages_with_lines:
                    pages_with_lines[pno + 1] = []
                pages_with_lines[pno + 1].append(line_number + 1)

    return occurrences, pages_with_lines
```

```python
def searchInPDF_french(pdf_path, key):
    occurrences = 0
    pages_with_lines = {}
    phrase_pattern = re.compile(re.escape(key), re.IGNORECASE)

    # Open the PDF file using PyMuPDF
    doc = fitz.open(pdf_path)

    for pno in range(len(doc)):
        page = doc[pno]
        content = page.get_text("text")

        # Split the content into lines
        lines = content.splitlines()

        # Combine lines into a single page content
        page_content = ' '.join(lines)

        # Find all matches for the key in the full page content
        matches = list(phrase_pattern.finditer(page_content))

        for match in matches:
            occurrences += 1  # Count each match occurrence

            # Determine the exact line where this match starts
            match_start = match.start()
            char_count = 0
            for line_num, line in enumerate(lines, start=1):
                char_count += len(line) + 1  # Include the newline character
                if char_count >= match_start:
                    # Store this line in pages_with_lines if it's not already there
                    if pno + 1 not in pages_with_lines:
                        pages_with_lines[pno + 1] = []
                    if line_num not in pages_with_lines[pno + 1]:
                        pages_with_lines[pno + 1].append(line_num)
                    break
```

3. **توليد الملخص:**
   - تم استخدام مكتبة **Transformers** ونموذج **facebook/bart-large-cnn** لتوليد ملخص لمحتوى PDF. حيث يتم تلخيص النصوص الطويلة إلى نصوص أقصر تلخص الفكرة الأساسية.

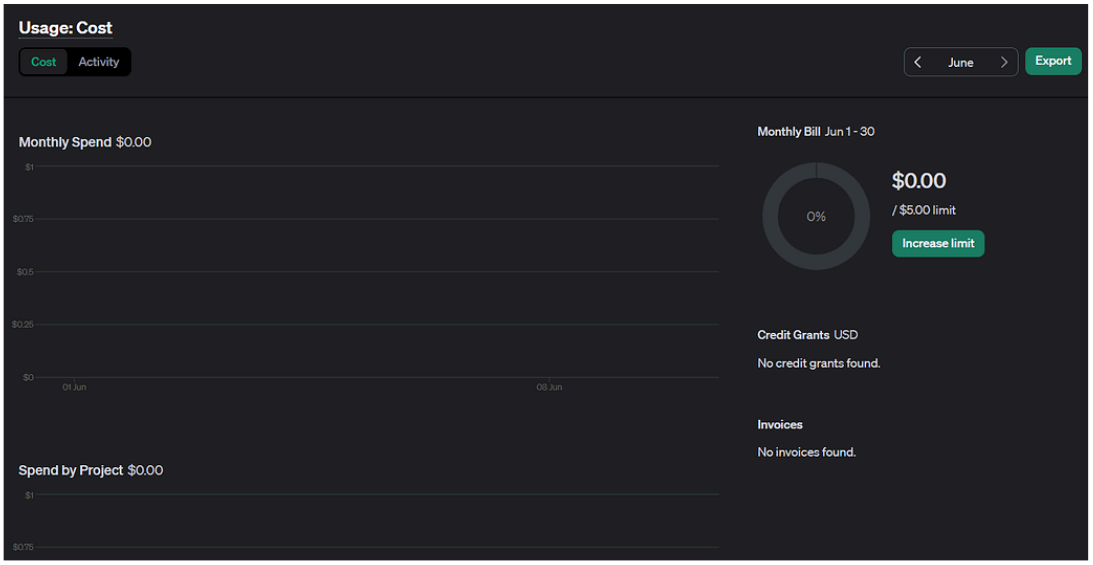
```python
# Function to summarize the extracted PDF text
def summarize_text(text):
    # Handle cases where text might be too long for summarization
    max_length = 1024  # Maximum tokens for the model
    if len(text) > max_length:
        text = text[:max_length]  # Truncate to fit model input

    summary = summarizer(text, max_length=150, min_length=30, do_sample=False)
    return summary[0]['summary_text']  # Return the summary text
```

توضيح هام: استعنت ب huggingface بدلا عن OpenAI لأن هذه الأخيرة أنهت الاستخدام المجاني المحدود لخدماتها مؤخرا.

الدليل:

D dkmernproject787067 Jun 10

I don't understand what's wrong. I tried it with different mails and numbers but it did the same thing and when I try to use the playground it gives the "insufficient quota" error. Is OpenAi free trial credits have been discontinued?



OpenAi free trial credits have been discontinued?

> Solved by jr.2509 in post #2
>
> Yes, the free credits have been discontinued earlier this year. OpenAI now for most users operates under pre-paid billing. In order to use the API you need to fund your account with a minimum balance of USD 5. Let us know if you have any further questions.

Is OpenAI no longer free?

OpenAI basically no longer offers a Free Tier. 23 août 2024

4. واجهة المستخدم باستخدام Node.js:

- تم إعداد واجهة مستخدم رسومية (GUI) باستخدام HTML وCSS. تتضمن واجهة المستخدم نماذج لتحميل الملفات وإجراء عمليات البحث.
- يتم استخدام **Node.js** للتعامل مع عمليات تحميل الملفات وإرسال استعلامات البحث إلى خادم Flask.

5. التفاعل مع PDF في المتصفح:

- تم تضمين إمكانية عرض ملفات PDF في واجهة المستخدم باستخدام مكتبة **PDF.js**، مما يسمح للمستخدمين بمشاهدة المحتوى مباشرة داخل المتصفح.
- تم توفير آلية لتظليل العبارات التي تم البحث عنها لتسهيل اكتشافها من قبل المستخدم.

التحديات والحلول

- التعامل مع النصوص الطويلة: تم تجاوز حد الطول المسموح به لتوليد الملخصات عبر تقطيع النصوص.
- تحسين البحث: تم تحسين دالة البحث في النصوص الفرنسية عبر استخدام تعبيرات عادية لضمان العثور على العبارات بدقة أكبر.

تحقيق شرط خدمة ويب RESTful

1. **التفاعل عبر HTTP:**
   - يتم استخدام بروتوكول HTTP لتبادل البيانات بين العميل والخادم. على سبيل المثال، يتم استخدام طلبات POST لتحميل ملفات PDF وطلبات POST أخرى للبحث عن النصوص داخل هذه الملفات.
2. **نقاط النهاية (Endpoints):**
   - تم تحديد نقاط النهاية في الخادم (Server) عبر Flask، حيث يتم إنشاء مسارات مثل `/upload` و`/search` التي يمكن للعميل استخدامها لإرسال البيانات واستلامها.
   - على سبيل المثال:
     - `POST /upload`: لتحميل ملف PDF.
     - `POST /search`: للبحث عن عبارات معينة في ملف PDF.

3. تنسيق البيانات:

- يتم استخدام JSON كتنسيق لتبادل البيانات بين العميل والخادم. حيث ترسل بيانات البحث (الاستعلام واللغة) إلى الخادم بتنسيق JSON، ويستقبل العميل النتائج بتنسيق JSON أيضًا.

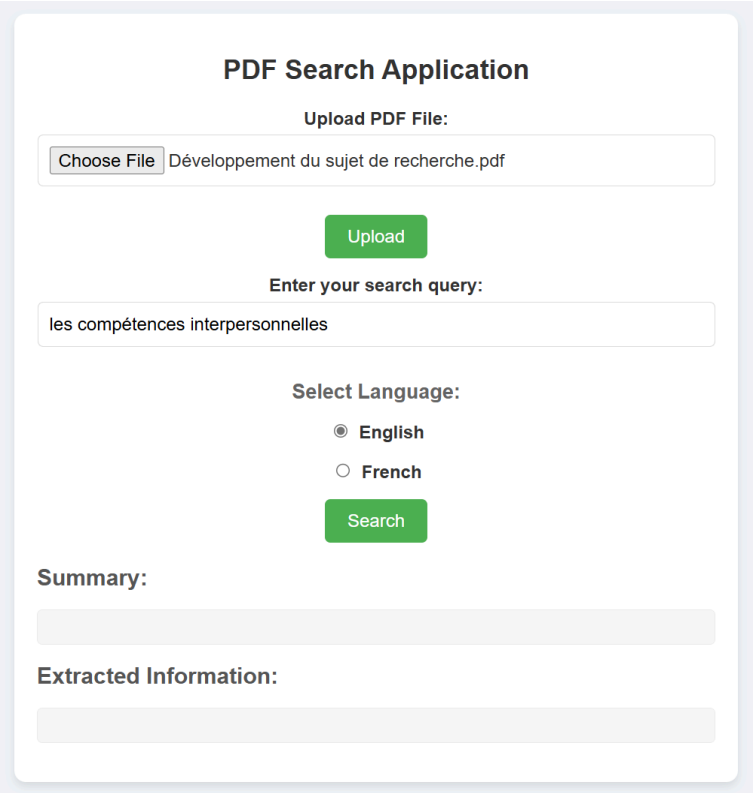
4. التفاعل بين العميل والخادم:

- يتم تنفيذ الطلبات من جانب العميل باستخدام **JavaScript**، حيث يتم استدعاء الوظائف عبر واجهة برمجة التطبيقات (API) باستخدام مكتبة Axios للقيام بطلبات HTTP.
- يتم التعامل مع الاستجابة من الخادم وتحديث واجهة المستخدم بناءً على النتائج المستلمة.

الخاتمة

هذا المشروع يعد مثالًا عمليًا على كيفية دمج تقنيات الويب المختلفة لإنشاء تطبيقات مفيدة. تم تحقيق الوظائف الأساسية بنجاح، مما يجعل من السهل على المستخدمين تحميل ملفات PDF والبحث فيها والحصول على ملخصات، مما يسهل عليهم الوصول إلى المعلومات الهامة بسرعة وفعالية.

واجهة التطبيق



أمثلة استخدام ومقارنة مع محرك بحث المتصفح: