استخراج مدونة ضخمة من أرشيف الإنترنت: دراسة حالة

إريك سي. فايغ

الملخص:

يصف هذا المقال كيف نزّلت مكتبات جامعة كنتاكي أكثر من 86 ألف عدد من الصحف التاريخية كانت قد رفعتها إلى أرشيف الإنترنت منذ عام 2015، وذلك لاستضافتها محليًا. ويشرح المؤلف جمع البيانات الوصفية اللازمة (المعرّف والمنشئ والتاريخ وحجم المادة) بصيغة (JSON) عبر واجهة البحث المتقدم، لأن واجهة (scrape) لا تتجاوز عشرة آلاف سجل، ثم حساب مساحة التخزين المطلوبة. ويعرض الموجّهات التي استخدمها مع (ChatGPT) لتوليد نصوص برمجية صغيرة بلغة بايثون، والتعديلات اليدوية القليلة التي أجراها عليها، ومنها ضبط مسارات الملفات وتنقية أسماء المجلدات، مع فاصل زمني بين التنزيلات احترامًا لإرشادات الأرشيف. واستمر التنزيل قرابة شهرين، وأخفق فيه 6,489 تنزيلًا أُعيدت لاحقًا، واستُعين بملفات البيان وقيم المجموع الاختباري في فحص الملفات. ويخلص إلى أن العملية كانت يسيرة بالتخطيط الجيد، وأن الذكاء الاصطناعي نجح في توليد نصوص برمجية محدودة النطاق.

الكلمات المفتاحية: أرشيف الإنترنت، الصحف الرقمية، واجهات برمجة التطبيقات، الذكاء الاصطناعي التوليدي، لغة بايثون

الخلفية:

منذ عام 2015، ترفع مكتبات جامعة كنتاكي أعدادًا من الصحف التاريخية إلى أرشيف الإنترنت. ونحن الآن بصدد تغيير برنامجنا الخاص بالصحف لاستضافتها محليًا، وتوسيع نطاق المجموعة ليشمل لا الأعداد الممسوحة ضوئيًا من مصادر تناظرية فحسب، بل أيضًا محتوى الصحف المولود رقميًا والمحتوى المحصود من الويب بلغة ترميز النص التشعبي (HTML).

وكان جمع نسخ محلية من المحتوى الذي وضعناه في أرشيف الإنترنت سيصبح مرهقًا، إذ كان سيستلزم الوصول إلى نظم التخزين الأرشيفي طويل الأمد القائمة على الأشرطة، وهو ما يعني وقتًا وجهدًا من عدد من الموظفين. وبديلًا من ذلك، يمكن الاستعانة بعملية مؤتمتة لسحب الملفات مجددًا من أرشيف الإنترنت، ويمكن أن يؤدي الذكاء الاصطناعي دورًا في تسريع التطوير اللازم. وكانت لدي بعض الخبرة في بايثون، وتوقعت أن يكون الذكاء

الاصطناعي أنجح في صياغة نصوص برمجية صغيرة ضيقة التركيز منه في صياغة شيفرة أكثر تعقيدًا. [1] وكان للفضول دور هنا أيضًا. فمع تنامي الاهتمام الكبير باستخدام الذكاء الاصطناعي، بدا تجريبه في مشروع برمجي صغير ومحدد المعالم أمرًا ممتعًا، مع إمكانية التخلي عنه إن لم تكن النتائج ناجحة.

وكشف البحث الأولي أيضًا أن أرشيف الإنترنت يشجع الزحف على الويب عبر ملف (robots.txt) الخاص به، ويتيح الوصول إلى واجهة برمجة تطبيقات لجمع البيانات الوصفية والملفات. [7] [4]

جمع البيانات الوصفية للكائنات:

تمثلت الخطوة الأولى في جمع البيانات الوصفية اللازمة لاسترجاع مواد مجموعة صحف كنتاكي الرقمية من أرشيف الإنترنت، التي كانت تضم 86,819 مادة.

ويتطلب ذلك في حده الأدنى المعرّفات الخاصة بأرشيف الإنترنت التي أُسندت إلى الكائنات. وعند النظر إلى عنوان (URL) لكائن ما، يقع المعرّف في نهاية العنوان تمامًا، بعد آخر شرطة مائلة. فعلى سبيل المثال، عنوان الكائن التالي `https://archive.org/details/xt74qr4nm47j` له معرّف هو «(xt74qr4nm47j)». وفي أرشيف الإنترنت ترتبط بالكائن عناوين (URL) مختلفة تتبع أنماطًا محددة تحيل إلى قيمة المعرّف. فمثلًا، لتنزيل ملف (zip.) يضم جميع الملفات المرتبطة بالكائن، بما فيها ما رُفع في الأصل، يُستخدم نمط العنوان التالي [5]: `https://archive.org/compress/xt74qr4nm47j`.

وكان من المرغوب فيه الحصول على بيانات وصفية إضافية لتيسير إعادة تسمية الكائنات وفرزها بعد تنزيلها، ولحساب إجمالي متطلبات التخزين للملفات قبل التنزيل. وتحديدًا، استُرجعت أربعة حقول: (creator) و(identifier) و(date) و(item_size). وكانت قيم البيانات الوصفية '(creator)' (المنشئ) مفيدة في فرز المحتوى المنزَّل في مجلدات منظمة بحسب عنوان الصحيفة. واستُخدمت البيانات الوصفية '(identifier)' (المعرّف) لبناء عناوين التنزيل، وأُضيفت البيانات الوصفية '(date)' (التاريخ) في بداية أسماء ملفات .(zip) المنزَّلة لإضافة معلومات دلالية. واستُخدمت البيانات الوصفية '(item_size)' (حجم المادة) لحساب مقدار مساحة التخزين المحلية التي نحتاج إليها.

ومن الجوانب الأخرى للحصول على البيانات الوصفية ضمان أن تكون بصيغة قابلة للتحليل الآلي. ولحسن الحظ، يتيح أرشيف الإنترنت عدة خيارات لجمع البيانات الوصفية منسقة بوصفها بيانات مهيكلة. فبالنسبة إلى المجموعات التي تضم أقل من عشرة آلاف مادة، يسترجع عنوان (URL) يستخدم (/scrape/) البيانات. [4]

```
https://archive.org/services/search/v1/scrape?
fields=identifier,item_size,creator,date&q=collection%3Akentuckynewspapers
```

ولما كانت مجموعتنا تضم أكثر من 10,000 مادة بكثير، فإن هذا لم يكن ليصلح لنا. وبدلًا من ذلك، استخدمت واجهة البحث المتقدم على الويب للبحث عما أحتاج إليه، إذ تتيح صفحة نتائج البحث المتقدم أيضًا خيار تصدير النتائج بوصفها بيانات مهيكلة. [6]

اخترت (JSON) صيغةً للمخرجات. وتضم البيانات الحقول (creator) و(date) و(identifier) و(item_size) ، داخل (response.docs)، مع معلومات عن البحث وعدد النتائج. ويَرِد فيما يلي مقتطف من هذه المخرجات.

```
callback({"responseHeader":{"status":0,"QTime":469,"params":
{"query":"collection:kentuckynewspapers","qin":"collection:
\"kentuckynewspapers\"","fields":"creator,date,identifier,item_size","wt":"json","sort":
```

"creatorSorter asc","rows":1000,"json.wrf":"callback","start":0}},"response":
{"numFound":86819,"start":0,"docs":[{"creator":"Adair County news.
(The)","date":"1917-11-14T00:00:00Z","identifier":"xt744j09xt7p","item_size":13630085},
{"creator":"Adair county community
voice","date":"2012-10-18T00:00:00Z","identifier":"kd99882j6j5s","item_size":91957159},
{"creator":"Adair county community
voice","date":"2012-09-20T00:00:00Z","identifier":"kd9tm71v5w8w","item_size":93942368},
{"creator":"Adair county community
voice","date":"2012-10-04T00:00:00Z","identifier":"kd9dz02z1d9w","item_size":92310777},
{"creator":"Adair county community
voice","date":"2012-09-27T00:00:00Z","identifier":"kd9jq0sq9446","item_size":95768429},
{"creator":"Adair county community
voice","date":"2012-09-20T00:00:00Z","identifier":"kd9pg1hh6t91","item_size":93942985},
{"creator":"Adair county community
voice","date":"2012-09-13T00:00:00Z","identifier":"kd9610vq304m","item_size":95027685},
{"creator":"Adair progress
(The)","date":"2015-06-11T00:00:00Z","identifier":"kd91n7xk8900","item_size":118977672},
{"creator":"Adair progress
(The)","date":"2012-08-16T00:00:00Z","identifier":"kd90z70v8f74","item_size":83747017},
{"creator":"Adair progress
(The)","date":"2014-02-06T00:00:00Z","identifier":"kd9x34mk6r4d","item_size":89747837},
{"creator":"Adair progress
(The)","date":"2012-07-26T00:00:00Z","identifier":"kd9d79571k58","item_size":107745767},
…

حساب احتياجات التخزين:

كان حساب متطلبات التخزين أساسيًا للتخطيط، كي تعمل العملية المؤتمتة من دون أن تصطدم بنقص في مساحة التخزين.

وكان حساب مساحة التخزين اللازمة بسيطًا نسبيًا، إذ كانت جميع المعلومات اللازمة ممثلة في مخرجات (JSON)، في عنصر البيانات الوصفية (item_size).

وحين شرعت في استخدام الذكاء الاصطناعي لتوليد شيفرة نصوص برمجية صغيرة، كانت المقالات المتعلقة بهندسة الموجّهات الفعالة مفيدة. واخترت استخدام النسخة المجانية من (ChatGPT) (GPT-4o mini) لسهولة استخدامها ودقتها المثبتة في البرمجة بلغة بايثون: فقد وجدت دراسة نُشرت عام 2024 عن توليد شيفرة بايثون والذكاء الاصطناعي أنه «من بين جميع النماذج المختبرة، أظهر (GPT-4) أعلى كفاءة في مهام توليد الشيفرة، محققًا معدل نجاح بلغ 86.23% في المجموعة الفرعية الصغيرة التي اختُبرت. وكان أداء النماذج القائمة على (GPT) هو الأفضل مقارنة بالنموذجين الآخرين، (Bard) و(Claude)». [2] وكان نهجي أن أبدأ ببساطة شديدة بموجّهات تهدف في البداية إلى إنتاج شيفرة لأداء مهام معالجة بسيطة، ثم أتوسع إلى صياغة موجّهات أكثر تطورًا تجمع عمليات متعددة. كما أمضيت وقتًا في تجريب الموجّهات ثم مراجعة الشيفرة الناتجة، مستخدمًا التعلم بأمثلة قليلة بتقديم عناصر نموذجية داخل الموجّه، مع إجراء تكرارات متعددة أحيانًا بصياغات ومستويات تفصيل مختلفة، وساعيًا إلى تغطية جميع التفاصيل اللازمة بأقل عدد ممكن من الكلمات. ويَرِد فيما يلي الموجّه المستخدم لتوليد الشيفرة. [9] [3]

اكتب نصًا برمجيًا بلغة بايثون يعمل من سطر الأوامر ويعالج ملف (json.) موجودًا في مسار كامل محدد في الشيفرة. يحتوي ملف (JSON) على بيانات مغلفة في دالة استدعاء لاحق (callback) على طريقة (JavaScript)، مثل (callback)({...}). ينبغي أن يستخرج النص البرمجي محتوى (JSON)، ويحلله، ويجمع قيم الحقل (item_size) الموجود في القائمة الواقعة في (response.docs). ينبغي أن يطبع النص البرمجي ويكتب في ملف سجل الحجم الإجمالي بالبايت و(KB) و(MB) و(GB) و(TB). ينبغي أن يكون مسار ملف السجل محددًا في الشيفرة. أدرج معالجة للأخطاء في حال كان الملف مشوهًا، واطبع رسالة تشير إلى اكتمال العملية بنجاح.

وكان من الإصلاحات السريعة التي احتجت إلى إجرائها في الشيفرة الناتجة التأكد من ضبط مسارات الملفات بوصفها سلاسل نصية خام لتجاهل تسلسلات أحرف الهروب. [10] وقد أجريت هذا التغيير مباشرة في الشيفرة بعد توليدها.

تنزيل الكائنات:

بعد أن صارت لدي جميع العناوين اللازمة لتنزيل الكائنات، بدأت أفكر في نهج مسؤول لاسترجاع البيانات. وكنت أعتزم مراعاة الإرشادات المحددة في ملف (robots.txt) الخاص بأرشيف الإنترنت. [7] وبدا إدراج توقف مدته من 10 إلى 30 ثانية بين كل تنزيل وآخر أمرًا معقولًا. [11]

الموجّه الأولي:

اكتب نصًا برمجيًا بلغة بايثون يحمّل ملف (JSON) موجودًا في مسار محدد في الشيفرة. يحتوي ملف (JSON) على بيانات مغلفة في دالة استدعاء لاحق على طريقة (JavaScript)، مثل (callback)({...}). ينبغي أن يستخرج النص البرمجي محتوى (JSON)، ويحلله، ويقرأ قيم (identifier) و(creator) و(date). لكل معرّف، أنشئ عنوان تنزيل بالصيغة `https://archive.org/compress/{identifier}`. أنشئ مجلدًا فرعيًا تحت مسار أساسي محدد في الشيفرة، يحمل اسم المنشئ (creator) مع استبدال الأحرف غير الصالحة أو المسببة للمشكلات بشرطات سفلية وحذف المسافات البيضاء في البداية والنهاية. نزّل ملف (ZIP) المقابل للمعرّف واحفظه باسم `{date}_{identifier}.zip` في المجلد الفرعي المناسب. انتظر فترة عشوائية بين 10 و30 ثانية قبل الانتقال إلى التنزيل التالي. إذا أخفق تنزيل ما، فسجّل العنوان الذي أخفق في ملف سجل محدد في الشيفرة.

موجّه تحسيني:

أضف شيفرة تجمع التنزيلات في مجموعات من 1000 وتنشئ مجلدات فرعية في مجلد التنزيلات تحمل الأسماء 0001 و0002 وهكذا.

موجّه تحسيني:

أضف شيفرة بايثون بحيث يُتحقق من كل ملف .(zip) بعد تنزيله باستخدام (zipfile.is_zipfile()) لضمان أنه أرشيف (ZIP) صالح. إذا أعادت (is_zipfile()) القيمة (False)، فينبغي تسجيل رسالة خطأ في ملف سجل (مثل (download.log)) تشير إلى أن الملف ليس ملف (ZIP) صالحًا. وإذا اجتاز الملف هذا الفحص، فينبغي فتحه بعد ذلك باستخدام (zipfile.ZipFile()) واختباره باستخدام الطريقة (testzip()). وإذا أعادت (testzip()) اسم ملف، مما يدل على ملف تالف داخل الأرشيف، فينبغي تسجيل ذلك أيضًا مع رسالة تحذير أو خطأ مناسبة

وقادتني الدروس المستفادة من الموجّهات السابقة إلى إدراج مثال دالة الاستدعاء اللاحق - بالصيغة (callback) ({...}) - بوصفه موجّهًا بأمثلة قليلة كي يحلل النص البرمجي ملف (JSON) بفعالية. وبعد توليد الشيفرة، ضبطت مسارات الملفات مرة أخرى بوصفها سلاسل نصية خام. وإضافة إلى ذلك، حدد الموجّه تجميع التنزيلات في مجموعات يسهل التعامل معها داخل مجلدات فرعية مرتبة تسلسليًا، وإدراج التحقق من ملفات .(zip) المنزَّلة.

وأُجريت الاختبارات باستخدام أجزاء صغيرة من ملف (JSON). وكان من الأخطاء التي وقعت أثناء بعض التنزيلات أن النص البرمجي بلغة بايثون لم يتمكن من إنشاء اسم مجلد صالح. فقد تضمنت قيم البيانات الوصفية (creator) أحرفًا غير صالحة لتسمية المجلدات. وعالج تعديل يدوي سريع لشيفرة بايثون لتنقية البيانات الوصفية (creator) هذه المشكلة.

الشيفرة الأصلية:

```
# Subdirectory under group: creator name
save_dir = os.path.join(group_dir, creator)
```

الشيفرة المحدَّثة:

```
# Sanitize creator
sanitized_creator = re.sub(r'[^A-Za-z0-9]', '_', creator)
# Subdirectory under group: creator name
save_dir = os.path.join(group_dir, sanitized_creator)
```

ونجحت اختبارات إضافية على أجزاء فرعية صغيرة أخرى من ملف (JSON)، ثم شُغّل النص البرمجي فعليًا.

النتائج:

عمل النص البرمجي (download_zips.py) على فترات متقطعة قرابة شهرين. وحدثت ثلاثة انقطاعات بسبب إعادة تشغيل الجهاز المحلي لتحديثات (Windows). وأخفق 6,489 تنزيلًا بسبب مشكلات مؤقتة في الاتصال بالشبكة، وبسبب ما بدا أنه ملفات تالفة.

وتجري مراجعة الملفات المنزَّلة عنوانًا تلو الآخر. وتُنزَّل مرة ثانية أي مواد مفقودة بسبب التنزيلات غير الناجحة. ويستخدم أرشيف الإنترنت بيانًا للملفات يتضمن قيم المجموع الاختباري لجميع الملفات الموجودة داخل أرشيف . (zip)، وكان ذلك مفيدًا في مراجعة الملفات وتقدير ما إذا كانت تالفة أم لا.

الخاتمة:

كان تنزيل محتوانا الذي سبق رفعه إلى أرشيف الإنترنت عبر الأتمتة أمرًا مباشرًا وسهل الإنجاز مع شيء من التخطيط الدقيق. فأرشيف الإنترنت مُهيّأ للسماح بحصاد المحتوى. ويمكن صياغة عناوين (URL) باستخدام خيارات البحث المتقدم في واجهة الويب، وتجمع هذه العناوين نتائج منسقة بوصفها بيانات مهيكلة مثل (JSON) و(XML). وباستخدام الذكاء الاصطناعي لتوليد نصوص برمجية صغيرة بلغة بايثون ضيقة التركيز، أنتج (ChatGPT) شيفرة بايثون سليمة البنية حققت مراحل التطوير المنشودة، مما أتاح الاختبار السريع، ولم يترك سوى مشكلات طفيفة أُصلحت بتصحيح الأخطاء أو بإدخال موجّهات تحسينية.

المراجع:

1. الشيفرة المشار إليها في هذا المقال متاحة على: https://github.com/libmanuk/IADownload/

1. Chang Y, et al. 2024. A survey on evaluation of large language models. ACM Trans Intell Syst Technol. 15(3):1–45. https://doi.org/10.1145/3641289
2. Coello CEA, et al. 2024. Effectiveness of ChatGPT in coding: a comparative analysis of popular large language models. Digital. 4(1):114–125. https://doi.org/10.3390/digital4010005
3. Cooper N, et al. 2024. Harnessing large language models for coding, teaching and inclusion to empower research in ecology and evolution. Methods Ecol Evol. 15(10):1757–1763. https://doi.org/10.1111/2041-210X.14325
4. Internet Archive. Archive.org about search [Internet]. [accessed 2025 May 6]. https://archive.org/help/aboutsearch.htm
5. Internet Archive. Downloading – a basic guide – Internet Archive Help Center [Internet]. [accessed 2025 May 6]. https://help.archive.org/help/downloading-a-basic-guide/
6. Internet Archive. Search – a basic guide – Internet Archive Help Center [Internet]. [accessed 2025 May 6]. https://help.archive.org/help/search-a-basic-guide/
7. Internet Archive. robots.txt [Internet]. [accessed 2025 May 6]. https://archive.org/robots.txt
8. Internet Archive. Internet Archive: digital library of free & borrowable texts, movies, music & Wayback Machine [Internet]. [accessed 2025 May 6]. https://archive.org/about/
9. Knoth N, et al. 2024. AI literacy and its implications for prompt engineering strategies. Comput Educ Artif Intell. 6:100225. https://doi.org/10.1016/j.caeai.2024.100225
10. Real Python. What are Python raw strings? – Real Python [Internet]. [accessed 2025 May 6]. https://realpython.com/python-raw-strings/
11. Infatica. 2021. Responsible web scraping: an ethical way of data collection [Internet]. [accessed 2025 May 6]. https://infatica.io/blog/responsible-web-scraping/