بايثون لمطابقة سجلات الموردين مع فهرس المكتبة

دان لو



الملخص:

يعرض المقال استخدام بايثون و(pandas) لمقارنة سجلات موردي الموارد الإلكترونية بسجلات نظام (Sierra) وتحديد الاختلافات التي تحتاج إلى معالجة. ويشرح تصدير البيانات وتنظيف أرقام (ISBN) واستخراجها من الحقول المتعددة، ثم مطابقة السجلات بالمحتوى المشترك. وينتقل إلى مقارنة العنوان والمؤلف ودرجات التشابه لتقليص الحالات المتبقية وترتيبها للمراجعة البشرية. وتوضح التجربة خفض عدد كبير من السجلات المحتاجة للفحص إلى مجموعة أصغر خلال أيام، دون اعتبار التشابه حكمًا نهائيًا بصحة المطابقة. كما يناقش أهمية إجراء المراجعة دوريًا، وتحسين بيانات الصلاحية وروابط الإتاحة، وآثار انتقال إدارة الموارد إلى واجهات الموردين. ويبرز دور الأتمتة في دعم الفهرسة وجودة البيانات، مع إبقاء القرار المهني والخصوصية ضمن الاعتبار.

الكلمات المفتاحية: مطابقة السجلات، بايثون، الموارد الإلكترونية، جودة البيانات، الفهرسة

مشكلة اختلاف السجلات:

لاحظت أمينة التزويد لدينا فروقًا كبيرة بين سجلات الموردين (Axis360) و(OverDrive) وسجلات نظام (Sierra). ورغم إرسال الموردين تحديثات منتظمة، ظلت المزامنة التامة صعبة بمرور الوقت. ولحل الاختلافات بكفاءة وسرعة، طُلب مني كتابة برنامج بايثون يقارن التفاصيل ويحدد السجلات التي ينبغي تعديلها في الفهرس.

المقارنة بالرقم الدولي المعياري للكتاب (ISBN):

بدأت بمحاولة كشف الاختلافات بمقارنة أرقام (ISBN). واخترت مكتبة (pandas) أداةً للتحليل[1]، وهي حزمة بايثون مفتوحة شائعة في علم البيانات والتعلم الآلي، مبنية فوق (NumPy) الداعمة للمصفوفات متعددة الأبعاد، ومتوافقة جيدًا مع حزم مثل (Matplotlib) للتصور. وتبسّط أعمالًا متكررة مستهلكة للوقت: تحميل البيانات وتحويلها وتنظيفها وتطبيعها واستكمالها ودمجها وتقسيمها وعرضها وفحصها، وغيرها.

يمكن تثبيت الحزم محليًا، لكن خيارًا أفضل للبداية، بحسب المؤلفة، هو (Google Colab)[2]، خدمة دفاتر (Jupyter) السحابية ذات المستوى المجاني. وتُثبَّت فيها (pandas) و(Matplotlib) افتراضيًا؛ ويكفي استيراد الحزمة باسم مختصر:

```
import pandas as pd
```

صدرنا سجلات (MARC) المناسبة من (Sierra) إلى (CSV)، وحملناها في الدفتر ككائن (DataFrame). وهذا مثال قراءة سجلات الكتب الصوتية من (Axis360) المصدرة من (Sierra):

```
sierra_axis_audio = pd.read_csv(
"/content/drive/My Drive/Library/axis_overdrive/sierra_axis360_audio.csv")
```

يتضمن الملف عادة العنوان والمؤلف وحقول (ISBN). وقد يتجاوز عدد الأرقام في السجل اثنين، فتُخزن الإضافات مؤقتًا في أعمدة (Unnamed) الأخيرة:

```
sierra_axis_audio.columns
Index(['TITLE', 'AUTHOR', 'ISBN1', 'ISBN2', 'Unnamed: 4', 'Unnamed: 5',
       'Unnamed: 6'],
      dtype='object')
```

واحتجنا إلى تنظيف الأرقام للمقارنة؛ فحقل (MARC020) يحوي حقولًا فرعية أخرى غير (a) المخصصة للرقم. وقد يضم التصدير معلومات زائدة، مثل «9780804148757 (electronic audio bk.)» أو «9780062460110 : $62.99».

كتبت دالة تستخرج الرقم المجرد من كل عمود يلي (AUTHOR)، وتجمعها في كائن (DataFrame) جديد. وبما أن سجل (Sierra) قد يضم أرقامًا متعددة، حصلنا على قائمة كاملة بكل أرقام السجلات المصدرة:

```
def extract_isbns(df):
    isbns = pd.DataFrame()
    result = pd.DataFrame()
    for i in range(2,len(df.columns)-1):
        new_isbns = df.iloc[:,i].astype(str)
        new_isbns = new_isbns.str.split(' ', 1).str[0].str.strip()
        isbns = pd.concat([isbns, new_isbns])
        isbns = isbns.reset_index()
        isbns = isbns[isbns[0]!='nan']
    return isbns
```

أما ملف المورد فأبسط؛ يحوي حقل (ISBN) واحدًا نظيفًا، إلى جانب العنوان والمؤلف وتاريخ النشر والإضافة والصيغة والكمية والإعارات. ولم يبق سوى استيراده بوصفه كائن (DataFrame).

ثم كتبت دالة تقارن أرقام المصدرين بعد التنظيف، مع عمود (Exist) جديد في كليهما. فإذا وجد رقم المورد في (Sierra)، تعبر القيمة 1 عن وجوده، و0 عن غيابه، والعكس عند اختبار رقم (Sierra) لدى المورد. وتعيد الدالة أيضًا كائنين إضافيين للسجلات المختلفة فقط:

```
def find_mismatch(vendor_df, sierra_df, sierra_isbns_df):
    vendor_df["Exist"] = vendor_df["ISBN"].astype(str).isin(sierra_isbns_df[0])
    not_in_sierra = vendor_df[vendor_df["Exist"]==False  | vendor_df["Exist"].isna() |
vendor_df["Exist"].empty]
```

```
    sierra_isbns_df["Exist"] =
sierra_isbns_df[0].astype(str).isin(vendor_df["ISBN"].astype(str))
    exists = sierra_isbns_df[sierra_isbns_df['Exist']==1].drop_duplicates("index")
    not_exists = sierra_isbns_df[sierra_isbns_df['Exist']==0].drop_duplicates("index")
    not_exists = not_exists[~not_exists["index"].isin(exists['index'])]
    isbns = pd.concat([exists, not_exists])
    sierra_df = pd.merge(sierra_df, isbns, how="left", left_index=True,
right_on="index")
    not_in_vendor = sierra_df[sierra_df["Exist"]==False | sierra_df["Exist"].isna() |
sierra_df["Exist"].empty]

    return vendor_df, sierra_df, not_in_vendor, not_in_sierra
```

بعد التشغيل، حللت عدد الاختلافات التي حلتها مطابقة (ISBN)، وأنتجت قائمة بسجلات (MARC) المطلوب إضافتها للنظام، وأخرى بالمطلوب حذفها. وأظهرت النتائج أن معظم سجلات (Axis360) سليمة، بينما نحو ثلث سجلات (OverDrive) ينطوي على مشكلات. وبقي أكثر من 4600 سجل يواجه مشكلة في (Sierra).

المقارنة بالعنوان والمؤلف:

لتقليص المشكلة أكثر، انتقلت إلى مطابقة السجلات المتبقية بدرجة تشابه حقلي العنوان والمؤلف، مستفيدة من قائمة الاختلافات الناتجة عن الخطوة السابقة.

نظفنا الحقلين واستخرجناهما. وكتبت دالة لدمجهما لكل سجل من المصدرين؛ تحول النص إلى أحرف صغيرة، وتحذف علامات الترقيم الزائدة، وتستخرج المعلومات اللازمة، ثم تجمعها في عمود (TitleAuthor):

```
def extract_title_author(vendor_df, sierra_df, vendor_title_field, vendor_author_field,
sierra_title_field = "TITLE", sierra_author_field ="AUTHOR"):
  vendor_df['TitleAuthor'] =
vendor_df[vendor_title_field].astype(str).str.lower().apply(remove_punctuations).astype(
str) + ' ' +
vendor_df[vendor_author_field].astype(str).str.lower().apply(remove_punctuations).astype
(str)
  sierra_df['TitleAuthor'] =
sierra_df[sierra_title_field].str.lower().str.extract("([^[]*\w*)").astype(str).apply(re
move_punctuations).astype(str) + ' ' +
sierra_df[sierra_author_field].str.lower().astype(str).apply(remove_punctuations).astype
(str).str.extract("([ a-z]*)").astype(str)
```

ثم استخدمت الوحدة (difflib) لمقارنة تشابه العمودين. وهي توفر أصنافًا ودوال لفروق البيانات. وهذه دالة المقارنة وتعيين مستوى التشابه:

```
from difflib import SequenceMatcher
def get_close_matches_indexes(word, possibilities, n=3, cutoff=0.6):
    if not n >  0:
        raise ValueError("n must be > 0: %r" % (n,))
    if not 0.0 <= cutoff <= 1.0:
        raise ValueError("cutoff must be in [0.0, 1.0]: %r" % (cutoff,))
    result = []
    s = SequenceMatcher()
    s.set_seq2(word)
```

```
for idx, x in enumerate(possibilities):
    s.set_seq1(x)
    if s.real_quick_ratio() >= cutoff and \
       s.quick_ratio() >= cutoff and \
       s.ratio() >= cutoff:
        result.append((s.ratio(), idx))
```

وبعد المقارنة، أنشأت جداول جديدة لكل السجلات المختلفة، يحوي كل صف فيها سجل (Sierra) ومعه أقرب سجل من المورد بالعنوان والمؤلف. ورتبت الصفوف بدرجة التشابه من 0 إلى 1؛ فالصفر يعني اختلافًا كبيرًا، والواحد تطابقًا قويًا.

وقارنت درجات تشابه العنوان والمؤلف في السجلات المختلفة، لكلٍّ من الكتب الصوتية والإلكترونية لدى الموردين، لتقدير مقدار ما تقلّص من المشكلة.

وباعتماد 0.7 حدًا مرجعيًا، احتاج نحو ثلثي السجلات غير المتطابقة إلى جهد فهرسة إضافي. ونجحت البرامج في تقليص مشكلة 30843 سجلًا إلى 3394 خلال أيام قليلة؛ ولم يحتج الفحص اللاحق من المفهرس إلا إلى هذا العدد، مما خفض العمل اللازم للإصلاح.

خاتمة:

يزداد تحدي تحديث سجلات (MARC) للموارد الإلكترونية ومزامنتها بين النظام والموردين بسبب التغير المستمر. ويمثل الحل وسيلة جيدة لكشف الاختلافات المتراكمة وتصحيحها. وأوصي المكتبات بتبنيه مهمةً دورية لمنع تراكم الاختلافات وتفاقم المشكلة.

ويُفضل إدخال حقل مخصص في سجلات المورد يحدد مدى الصلاحية الزمني، ثم تطوير برنامج يُجدّد أو يُعدّل تلقائيًا ما اقترب من الانتهاء أو تجاوزه، باسترجاعه من النظام بالتصدير أو (API). ولا يولّد موردون كثيرون خطأ 404 عندما تفقد المكتبة إتاحة مورد ككتاب إلكتروني؛ فينبغي أن يكون البرنامج مرنًا لاكتشاف الروابط التي لم تعد تعمل كما يلزم.

وفي المقابل، يحدث تحول كبير مع خروج نظام سجلات الموارد الإلكترونية تدريجيًا من نظام المكتبة. فمثلًا، أتاحت (BiblioCommons) خدمة (BiblioCloudRecords) لإضافة موارد معينة إلى واجهة الفهرس عبر (API) دون إدخال (MARC) إلى النظام المتكامل. ويُحسّن ذلك الإتاحة، لكنه ينقل بيانات المستفيدين وإحصاءات الاستخدام من المكتبة إلى الموردين حتمًا. وزاد حضور الموارد الإلكترونية منذ الجائحة، وسيزداد أثرها في المستقبل المنظور. ولذلك يحسن أن نسأل الآن كيف تحدد المكتبات دورها في هذا المشهد الجديد.

المراجع:

[1] Pandas.2018.Python Data Analysis Library: https://pandas.pydata.org/ .

[2] Google Colaboratory.2019: https://colab.research.google.com/ .

https://pandas.pydata.org/

https://colab.research.google.com/