Метод автоматизации обработки электронных таблиц формата XLSX без потери совместимости с программным пакетом Microsoft Office

А.Б. Бетелин[1], Г.А. Прилипко[2], А.Г. Прилипко[3], С.Г. Романюк[4], Д.В. Самборский[5]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, ab@niisi.msk.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, prilipko@niisi.msk.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, aleksey.prilipko@gmail.com;

[4]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, sgrom@niisi.ras.ru;

[5]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, samborsky_d@fastmail.com

Аннотация. Для автоматизации документооборота часто требуются кроссплатформенные средства генерации или редактирования файлов электронных таблиц в формате Office Open XML. Несмотря на наличие нескольких свободно распространяемых библиотек и программных пакетов с подобными функциями, ни одно из этих средств не гарантирует полной совместимости с программным пакетом Microsoft Office. В данной статье анализируется возможность автоматизации редактирования файлов электронных таблиц без потери каких-либо атрибутов и внутренних элементов. Авторами статьи разработана программа, преобразующая содержимое файла формата Office Open XML в текстовый формат и обратно, тем самым обеспечивающая аккуратную и эффективную обработку документов электронных таблиц для пакета Microsoft Office.

Ключевые слова: документооборот, электронная таблица, Office Open XML

1. Введение

Одним из общепринятых форматов электронного документооборота является формат документов Office Open XML (OOXML) программного пакета Microsoft Office. Этот формат был принят в качестве международных стандартов ECMA-376 [1] и ISO/IEC 29500:2016 [2]. В отличие от предшествующих ему бинарных форматов документов пакета Microsoft Office, файлы формата OOXML имеют описание внутренней структуры и допускают непосредственное редактирование их содержимого. Поэтому при необходимости автоматизированного редактирования сложных XLSX-документов нет объективных препятствий для решения этой задачи без применения сложных в освоении и дорогостоящих программных продуктов.

Однако, в силу сложности устройства формата OOXML пользователи часто вынуждены использовать одно из программных средств или специализированную среду разработки, например:

- среду разработки Microsoft Office SDK [2], позволяющую автоматизировать редактирование документов с использованием языков Visual Basic for Applications (VBA) и .NET [11];
- программные пакеты LibreOffice [4] и Apache OpenOffice [5], имеющие некоторые средства для автоматизации обработки документов и интеграции с серверами баз данных;
- специальные библиотеки для используемой среды программирования, предоставляющие ограниченные возможности создания или редактирования XLSX-документов. Примерами таких библиотек служат OpenPyXL [6] для языка программирования Python, библиотеки Spreadsheet::ParseXLSX [7] и Excel::Writer::XLSX [8] для языка программирования Perl, или библиотека Apache POI [9] для среды Java.

Большинство из средств для работы с файлами XLSX после чтения документа электронной таблицы полностью преобразуют его в объект данных, имеющий определенную внутреннюю структуру. Этот объект данных не всегда во всем соответствует устройству исходного файла. Например, в случае использования программы LibreOffice XLSX-файл при чтении преобразуется в объект документа в формате Open Document Format (ODF) [10] и этот формат не вполне совместим с форматом Microsoft Excel. Аналогично работают и библиотеки OpenPyXL, Apache POI, и др. Из вышеупомянутых инструментов только Microsoft Office SDK в полной мере гарантирует сохранность всех атрибутов редактируемых электронных таблиц. Созданию

инструментов, которые бы обеспечивали полную совместимость с Microsoft Excel, препятствует сложность стандарта ECMA-376. Так, только основной документ этого стандарта содержит более 5000 страниц.

Тем не менее, возможен другой метод редактирования XLSX-файлов, дающий возможность сохранить все атрибуты электронной таблицы. Как известно, XLSX-файл является ZIP-архивом, в который помещено несколько текстовых файлов с данными электронной таблицы и элементами ее оформления. Если редактирование затрагивает только части, хорошо понятные инструменту, например, XML-файлы листов электронной таблицы, то все остальные составляющие документа останутся прежними. Следуя этому подходу, авторы разработали программу xlsx-processor, автоматизирующую редактирование электронных таблиц в формате XLSX. Данная программа написана на языке Python и использует библиотеку xml для работы с XML-файлами и библиотеку OpenPyXL для коррекции адресов формул ячеек MS Excel. Программа функционирует в операционных системах семейств GNU Linux и Microsoft Windows. На момент публикации авторы не имеют сведений о наличии другого общедоступного инструмента с функциями, аналогичными функциям утилиты xlsx-processor.

2. Алгоритм работы утилиты xlsx-processor

Утилита xlsx-processor позволяет изменять содержимое электронной таблицы так, как если бы ее данные были бы представлены в текстовом формате. Для этого xlsx-processor сначала распаковывает XLSX-файл и выводит таблицу каждого листа в текстовом формате CSV (comma-separated-values). Позже, когда пользователь отредактирует или вставит данные в эти файлы, утилита создаст новый XLSX-файл, в который поместит новые данные, но сохранит все атрибуты и элементы оформления.

Программа xlsx-processor имеет следующий синтаксис:

```
xlsx-processor.py {подкоманда} [опции]
```

где {подкоманда} определяет выполняемое действие: распаковку XLSX-файла (unpack), извлечение данных листов электронной таблицы (extract), заполнение новыми данными (fill), подстановку регулярных выражений в текстовых значениях ячеек (regsub), или компоновку нового XLSX-файла (pack).

Например, команда

```
xlsx-processor.py unpack --file example.xlsx
```

открывает файл example.xlsx как обычный ZIP-архив и записывает все его содержимое в указанный подкаталог (по умолчанию это подкаталог xlsx в текущем каталоге).

Далее, следующей командой

```
xlsx-processor.py extract
```

запускается процедура извлечения данных, которая первым делом выполняет чтение всех строк из внутреннего словаря документа — они помещаются непосредственно в XML-элементы ячеек и для этих ячеек устанавливается тип inlineStr. Затем эта команда выводит данные листов электронной таблицы в текстовые файлы в формате CSV с именами, аналогичными исходным XML–файлам – sheet1.csv, sheet2.csv, и т.д. При выводе этих файлов утилита добавляет первую колонку с номерами строк, которая позже помогает понять, какие строки были переставлены или скопированы. Если ячейка содержит многострочный текст, то при выводе в формат CSV символы перевода строки обозначаются символом \\n, что упрощает последующую обработку данных, поскольку каждый ряд электронной таблицы образует одну строку текстового файла. Кроме CSV-файлов для каждого листа электронной таблицы утилита также выводит настоящие имена всех листов в отдельный файл с именем workbook.csv.

Редактирование данных листов электронной таблицы в текстовом формате возможно при помощи любого языка программирования. Обычно в процессе редактирования таблиц некоторые строки используются в качестве шаблонов и заполняются нужными данными, тогда как ненужные строки могут быть удалены. Важно, что при этом сохраняется первая колонка, содержащая номер строки в исходном файле. Отображение номеров строк исходного файла в результирующий используется позже для автоматической коррекции адресов ячеек в формулах.

Для упрощения редактирования утилита предлагает использовать два специальных символа для содержимого ячеек: символ #, обозначающий, что данная ячейка полностью сохраняет свое значение, и символ @, который указывает, что значение сохраняется, но в формуле этой ячейки выполняется автоматическая трансляция адресов ячеек (см. в следующем разделе). То есть, если в отредактированном CSV-файле ячейка имеет значение # или @, то используется ее прежнее значение, содержащееся в XML-файле.

Ячейки, значение которых нет необходимости изменять, лучше помечать этими символами, чтобы утилита не тратила время на очистку ячеек и подстановку их старых значений. Кроме того, иногда ячейки листов электронной таблицы содержат элементы форматирования, которые теряются при выводе в CSV-формат. Поэтому чтобы сохранить вид этих ячеек, лучше оставить их прежнее представление в формате XML. С этой целью утилита допускает применение символов @ и # для целого ряда – когда строка CSV-файла начинается с этих символов, или всего листа, если заголовок CSV-файла начинается со специального символа.

Иногда требуется отредактировать ячейки, содержащие элементы форматирования текста, которые теряются при выводе в формат CSV. Примерами таких ячеек служит заголовок документа или его окончание, где могут быть указаны некоторые даты и реквизиты. Для редактирования таких ячеек служит подкоманда regsub, выполняющая подстановку регулярных выражений непосредственно в текстовых значениях элементов XML-файлов. Например, следующая команда выполнит замену строки «30.12.2023» на строку «15.01.2024» внутри листов 1 и 3:

```
xlsx-processor.py regsub --sheet 1 --sheet 3 --regsub 30.12.2023 15.01.2024
```

Подкоманда regsub использует Python-библиотеку поддержки языка регулярных выражений и потому способна выполнять гораздо более сложную обработку текста, нежели простую подстановку строк. См. документацию библиотеки re языка Python и описание функции re.subn, используемой подкомандой regsub.

После завершения редактирования пользователь запускает команду заполнения XML-файлов новыми текстовыми данными:

```
xlsx-processor.py fill
```

На этой стадии утилита анализирует, какие строки таблиц были скопированы, перемещены или удалены и делает соответствующие исправления в XML-документе каждого листа. Затем утилита записывает новые значения ячеек во все измененные или скопированные строки.

Завершает редактирование электронной таблицы упаковка отредактированных XML-файлов в новый XLSX-файл — это действие выполняет подкоманда pack:

```
xlsx-processor.py pack --output result.xlsx
```

Помимо модификации данных, для изменения оформления ячеек электронной таблицы можно также изменять стиль ячеек. Например, при необходимости выделения некоторых ячеек особым цветом фона можно использовать стиль той ячейки шаблонного документа, в которой демонстрировалась эта возможность. Чтобы задать стиль ячейки, достаточно в соответствующее ей поле CSV-файла добавить префикс вида s=style_index!, где style_index – индекс стилей. Необходимый индекс можно узнать из значения атрибута s XML-элемента некоторой ячейки, которая использует необходимый стиль. Данная процедура не предусматривает создания новых стилей ячеек, поэтому если в исходном XLSX-файле шаблона документа не существовало ячейки с нужным стилем оформления, то в нем следует создать строку с примером такой ячейки, чтобы потом в процессе редактирования стиль этой ячейки мог быть применен к другим ячейкам. Эту вспомогательную строку в процессе редактирования можно удалить.

3. Автоматическая коррекция адресации ячеек в формулах

Использование утилиты xlsx-processor для создания XLSX-файлов по заданному шаблону предусматривает копирование строк таблицы и заполнение их новыми данными. После такого редактирования может потребоваться коррекция формул электронной таблицы, поскольку копирование и сдвиг строк изменяет их нумерацию. Если бы модификация таблицы выполнялась средствами программы Microsoft Office Excel, то адреса ячеек в формулах таблицы были бы исправлены автоматически. А именно, когда в электронной таблице копируется ячейка с формулой, то адреса в этой формуле корректируются по следующим правилам: если использован относительный адрес (символ $ отсутствует), то он корректируется вектором смещения копии ячейки относительно позиции ее оригинала; если же использован адрес с абсолютной координатой (обозначенный символом $), то он при копировании не изменяется. В случае сдвигов ячейки с формулой или ячеек адресатов формулы, адреса в формуле корректируются так, чтобы сохранить ее функцию.

При редактировании XLSX-файлов с использованием утилиты xlsx-processor пользователь может либо исправить формулу самостоятельно, либо разрешить утилите выполнить коррекцию формулы автоматически. В последнем случае пользователю достаточно в позиции ячейки вместо формулы указать только один символ @.

Чтобы коррекция формул привела к желае-

мому результату, в формулах документа-шаблона следует применять относительную и абсолютную адресацию рядов. Например, для суммирования значений диапазона ячеек колонки A вместо формулы SUM(A1:A2) в ячейке A3 нужно использовать абсолютный адрес первого ряда, SUM(A$1:A2). Тогда при вставке 3 новых рядов в этот диапазон ячейка формулы будет сдвинута на 3 ряда вниз, и формула будет заменена на SUM(A$1:A5), что соответствует желаемому результату.

Результирующий XLSX-файл обычно требует обновления результата вычислений формул в некоторых ячейках, поэтому утилита устанавливает специальный атрибут fullCalcOnLoad. Этот атрибут заставляет программу Microsoft Excel сразу после чтения электронной таблицы выполнить ее полный пересчет.

4. Заключение

В данной статье был выполнен анализ существующих средств автоматизации обработки документов электронных таблиц в формате XLSX и предложен новый способ редактирования этих файлов, сохраняющий полную совместимость с пакетом Microsoft Office.

Работа выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2024-0001 «Создание и реализация доверенных систем искусственного интеллекта, основанных на новых математических и алгоритмических методах, моделях быстрых вычислений, реализуемых на отечественных вычислительных системах» (1023032100070-3-1.2.1).

A Method for Automation of Data Processing in XLSX Spreadsheet Files Without Loss of Compatibility With Microsoft Office

A.B. Betelin, G.A. Prilipko, A.G. Prilipko, S.G. Romanyuk, D.V. Samborskiy

Abstract. For automating document workflows, having cross-platform tools that can generate or edit electronic spread-sheets in the Office Open XML format is essential. While there are several free libraries and software packages available with such capabilities, they do not always ensure full compatibility with Microsoft Office. In this article we explore the feasibility of automating the processing of electronic spreadsheets without sacrificing any attrib-utes or internal elements. The authors have developed a program for editing the contents of electronic tables by converting data into plain text and back into the Office Open XML format, thereby enabling their accurate and efficient processing.

Keywords: document workflow, electronic spreadsheet, Office Open XML

Литература

1. Международный стандарт ECMA-376, Office Open XML file formats, 5th edition, December 2021, https://ecma-international.org/publications-and-standards/standards/ecma-376 (дата обращения 02.07.2024).

2. Международный стандарт ISO/IEC 29500-1:2016. Information technology. Document description and processing languages. Office Open XML File Formats, https://www.iso.org/standard/71691.html (дата обращения 02.07.2024).

3. Сайт документации Open XML SDK, https://learn.microsoft.com/en-us/office/open-xml/open-xml-sdk (дата обращения 02.07.2024).

4. Сайт документации Apache OpenOffice, https://www.openoffice.org (дата обращения 02.07.2024).

5. Сайт документации LibreOffice, https://www.libreoffice.org (дата обращения 02.07.2024).

6. Сайт документации OpenPyXL, https://pypi.org/project/openpyxl (дата обращения 02.07.2024).

7. Сайт документации Spreadsheet::ParseXLSX, https://metacpan.org/pod/Spreadsheet::ParseXLSX (дата обращения 02.07.2024).

8. Сайт документации Excel::Writer::XLSX, https://metacpan.org/pod/Excel::Writer::XLSX (дата обращения 02.07.2024).

9. Сайт документации Apache POI - the Java API for Microsoft Documents, https://poi.apache.org (дата обращения 02.07.2024).

10 Сайт документации OpenDocument Format, https://opendocumentformat.org (дата обращения 02.07.2024).

11. Programming Excel with VBA and .NET by Jeff Webb, Steve Saunders. O'Reilly Media, Inc. 2006. ISBN: 9780596007669.