Paper SM04

Spot It: A Shiny Application for Visual QC at Scale

Pravin Lakkaraju, Servier Pharmaceuticals, USA
Samiar Ashtiany, Phastar, United Kingdom

Abstract

In the pharmaceutical industry, automating repetitive, labour-intensive tasks is crucial for efficiency and accuracy. Spot-It is an innovative application designed to automate visual oversight quality control, addressing the challenge of managing large-scale document comparison. By handling various file types, including RTF, DOC, Excel, and PDF, Spot-It converts and organizes documents, ensuring content consistency and precision. This automation significantly reduces manual effort, minimizes errors, and enhances productivity for statisticians and programmers. By streamlining oversight QC processes, Spot-It supports the pharmaceutical industry's need for stringent quality control, aligning with current trends in automation and digital transformation. This paper provides an in-depth look at Spot-It's capabilities, mechanisms, and benefits, highlighting its potential to revolutionise quality control in the pharmaceutical sector.

Introduction

Quality control is a cornerstone of ensuring regulatory compliance, accuracy, and the integrity of data across clinical trials and regulatory submissions. Oversight QC is a labour-intensive and time-consuming task, often requiring comparison of multiple complex documents and outputs. Traditionally, this process relies heavily on manual effort, which is not only resource-intensive but also prone to human error.

As automation and digital tools become more important in the pharmaceutical world, there is a growing need for simple and effective ways to improve QC processes. Automating repetitive QC tasks can save time and reduce errors, thereby improving workflow and efficiency.

This paper introduces Spot-It, a cutting-edge application designed to transform visual QC at scale. Built on the robust and flexible Shiny framework, Spot-It addresses the challenges of large-scale document comparison by automating the process of converting, organizing, and verifying various file types, including RTF, DOC, Excel, and PDF.

In this paper, we will explore the challenges faced by traditional QC methods and provide an in-depth overview of Spot-It's capabilities, demonstrate its benefits through a case study. By the end of this paper, readers will gain a comprehensive understanding of how Spot-It can transform oversight QC processes and contribute to the industry's ongoing digital transformation.

Challenges in Quality Control

Quality control in the pharmaceutical industry is a critical but demanding process. Oversight QC, an aspect of quality control, presents unique challenges which would benefit from automation. These include:

- **Manual oversight of outputs:** Oversight QC often involves comparing reports, protocols and RTF outputs for consistency/differences. This manual process is slow and increases the likelihood of discrepancies being overlooked.

- **Handling various types of content:** Oversight QC must account for differences in formatting, annotations, and embedded tables or charts, which complicate document reviews. For instance, reviewing table shells containing hundreds of embedded tables with detailed statistical summaries can be particularly challenging.
- **Compliance with regulatory standards:** Stringent regulatory requirements mandate a high level of precision, making even minor oversights in QC potentially costly.
- **Scalability and timeliness:** As the number of documents increases with growing projects or organizations, traditional oversight QC methods struggle to scale efficiently while maintaining accuracy.

These specific challenges in oversight QC underline the need for innovative solutions that can enhance efficiency, reduce errors, and support the industry's demand for accuracy and compliance at scale.

Spot-It: An Overview

Spot-It is a comprehensive application designed to streamline oversight QC processes. Built on a hybrid architecture, Spot-It combines an R Shiny front-end with a Python backend to deliver seamless document comparison, data extraction, and report generation capabilities. This integration ensures that users can perform complex QC tasks with a click of a button.

Key Features

- **Document Comparison:** Spot-It supports side-by-side comparisons of RTF files, helping users identify discrepancies in text, formatting, and structure. This functionality is critical for ensuring the consistency and accuracy of regulatory submissions and outputs.
- **Data Extraction:** The app extracts structured elements, such as titles and footnotes from table shell documents and organizes them into Excel sheets. This simplifies handling large datasets and reduces manual work for QC teams.
- **User-Friendly Interface:** Spot-It's dashboard provides easy navigation through its core functionalities, including file uploads, document comparisons, and data extractions. The sidebar menu ensures quick access to each feature.
- **Efficient Data Manipulation:** Using Python's computational power, Spot-It performs tasks such as text extraction, document comparison, and report generation effectively.
- **Real-Time Insights:** Users receive visual feedback in the click of a button, allowing for quick outcomes and decision-making.
- **Scalable Design:** Spot-It handles large volumes of data and diverse file types without compromising speed or accuracy, making it suitable for both small to large teams.

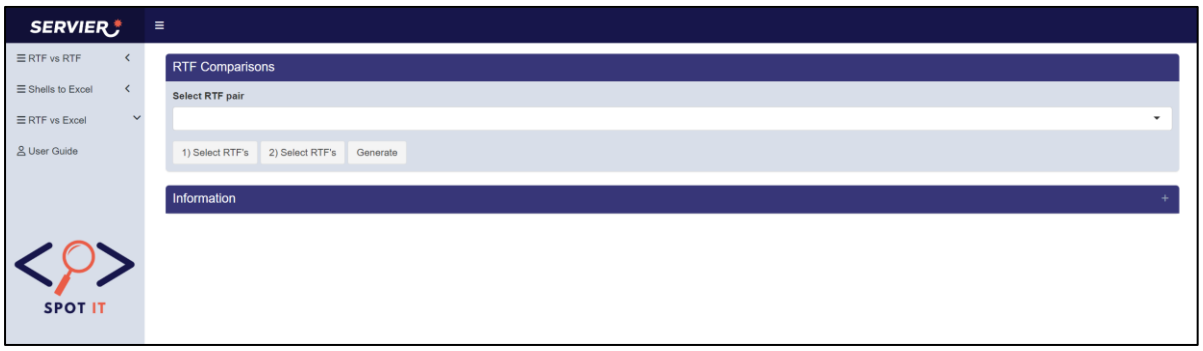

Figure 1: Application front-end and layout

Why Spot-It Matters

Spot-It simplifies traditionally time-consuming and error-prone tasks in oversight QC. Its ability to automate document comparisons and extract data into structured formats will ultimately Improve productivity for QC teams and enhanced accuracy

This overview provides a high-level understanding of Spot-It's capabilities and impact, setting the stage for the detailed mechanisms and workflow discussed in the next section.

Mechanisms and Workflow

This section provides a more detailed explanation of how the application operates, with each core feature.

RTF vs RTF Comparison

RTF output comparison of tables and listings is vital for maintaining consistency across regulatory documents, where even small discrepancies can lead to large compliance issues. For instance, a minor mismatch in statistical summaries or table titles could result in a rework during regulatory reviews.

Workflow

The RTF vs RTF comparison process follows a structured workflow to ensure accuracy and user convenience:

1. **File Upload**: Users upload two RTF files via the intuitive R Shiny interface. The application validates the files to ensure compatibility.
2. **Pre-Processing**: The Python backend (Functions.py) processes the documents using *striprtf* to convert to text, handling encoding, parsing content structures, and identifying key elements such as text blocks and formatting markers.
3. **Line-by-Line Comparison**: Using a combination of string-matching techniques and approximate similarity algorithms, Spot-It identifies discrepancies in:
    - Text content (e.g., mismatched words or missing sections).
    - Formatting (e.g., inconsistent fonts or alignments).
    - Structural elements (e.g., table layouts or headers).
4. **Highlighting Discrepancies**: Differences are color-coded based on severity:
    - **Yellow**: Minor string differences (e.g., extra spaces or punctuation).
    - **Orange**: Significant line-level differences (e.g., multiple word changes).
    - **Red**: Missing or mismatched content.
5. Extra Lines:
    - When one document has additional lines, they are flagged and stored separately.
    - Extra lines are not confused with mismatches but are clearly marked as absent in the counterpart document.
6. **Results Display**: Discrepancies are presented in a side-by-side format for easy review, allowing users to quickly identify and address inconsistencies.
7. **Report Generation**: A detailed summary of the differences is generated and available for download in Excel format. The report categorizes discrepancies and includes hyperlinks to specific locations in the documents.

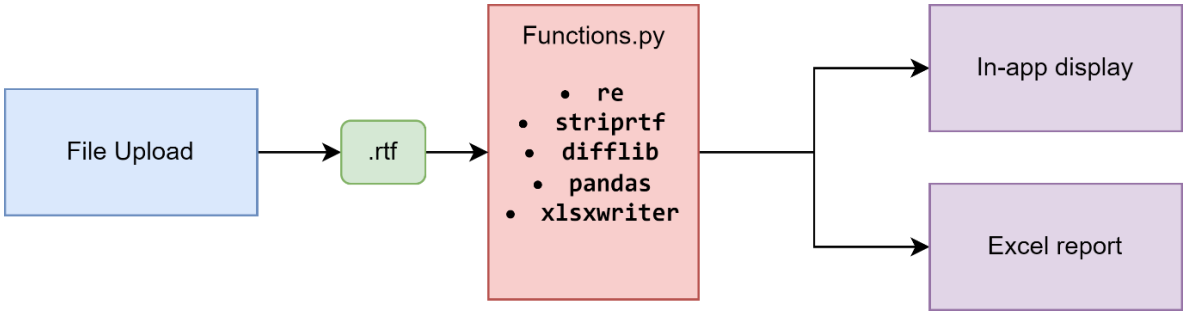

Figure 2: RTF vs RTF workflow

Technical Details

Spot-It leverages a combination of Python libraries and custom algorithms to achieve high accuracy and efficiency in RTF comparisons:

- **Regular Expressions (re)**: Used extensively for parsing and identifying patterns in text, enabling precise extraction of discrepancies.
- **String Matching**: Functions use token-based comparisons and approximate similarity algorithms (via Python's difflib.SequenceMatcher) to identify differences efficiently.
- **Custom Highlighting Logic**: Tailored algorithms pinpoint and categorize differences at word and line levels, handling complex cases like embedded tables and formatting anomalies.

Shells to Excel Conversion

The Shells to Excel Conversion feature in Spot-It streamlines the extraction and organization of data from table shell documents. By converting Word documents into structured Excel sheets, this feature simplifies the handling of complex datasets, making it easier for quality control (QC) teams to manage and review information efficiently.

Workflow

The Shells to Excel Conversion process follows a straightforward workflow:

1. **File Upload**: Users upload table shell Word documents through the R Shiny interface. The application verifies the file format and ensures compatibility.
2. **Data Extraction**: The Python backend scans the documents to identify key elements such as table numbers, titles, footnotes, and other structural components. These elements are parsed and extracted using robust regular expression-based logic.
3. **Data Structuring**: Extracted elements are organized into a structured Excel format with predefined columns, such as Table Number, Title, Type, and Footnotes. This ensures consistency and ease of use for downstream processes.
4. **Output Generation**: The final Excel file is made available for download, containing neatly categorized data for review and further analysis.

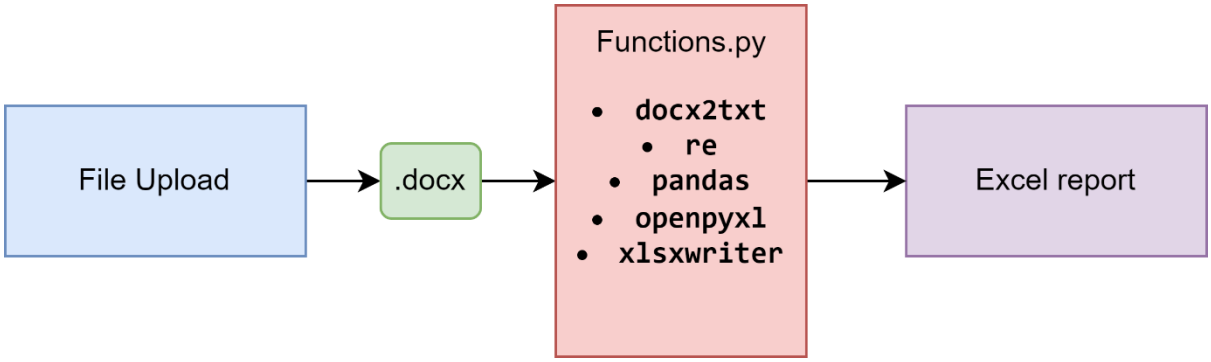

Figure 3: Shells to Excel workflow

Technical Details

Spot-It uses a combination of text parsing and data manipulation techniques to deliver reliable results during the Shells to Excel Conversion:

- **Document Parsing**:
  - Utilizes docx2txt for reading and extracting text from Word documents.
  - Employs re for pattern matching and identifying structured elements like table numbers and footnotes.
- **Data Organization**:
  - Leverages pandas for efficient data manipulation and organization.
  - Formats and writes structured outputs to Excel using openpyxl and xlsxwriter.
- **Error Handling**:
  - Includes mechanisms to handle inconsistencies in document formatting, such as missing elements or non-standard layouts, ensuring robustness across various inputs.

RTF vs Excel

The RTF vs Excel feature in Spot-It ensures consistency between textual RTF outputs and their structured Excel counterparts. By systematically comparing RTF documents with Excel table shells, this feature helps quality control (QC) teams identify discrepancies and ensure alignment in critical documentation.

Workflow

The RTF vs Excel process involves the following steps:

1. **File Upload**: Users upload RTF documents and the corresponding Excel files. The application validates file formats to ensure compatibility.
2. **Data Mapping**: The Python backend extracts and maps relevant elements from both file types, such as titles, footnotes, and table numbers, to prepare for comparison.
3. **Comparison**:
   - Differences in text, formatting, and structure are identified.
   - Missing or mismatched entries are flagged for review.
4. **Results Display**: The application visually highlights discrepancies, categorizing them by severity to facilitate prioritization.
5. **Report Generation**: A detailed Excel report summarizes the findings, offering an organized view of discrepancies for easy resolution.

Technical Details

Spot-It employs a combination of advanced tools and methodologies for accurate RTF and Excel comparisons:

- **File Parsing**:
  - Extracts structured data from RTF documents using striprtf.
  - Reads and processes Excel files with pandas for efficient data handling.
- **Comparison Logic**:
  - Uses difflib.SequenceMatcher to detect text similarities and differences.
  - Employs custom algorithms to align and validate content between RTF and Excel files.
- **Output Formatting**:
  - Generates comprehensive Excel reports using xlsxwriter, featuring color-coded highlights for discrepancies.

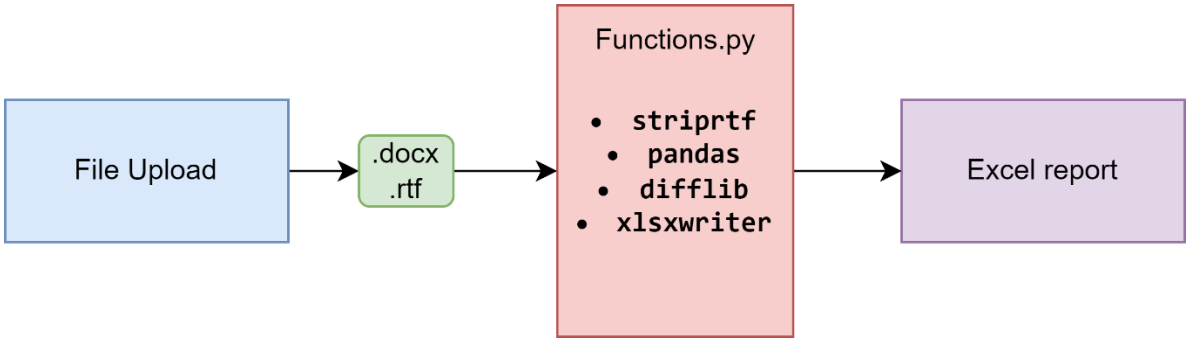

Figure 4: RTF vs Excel workflow

Overall Workflow Diagram

To provide a complete view, the following workflow illustrates Spot-It's operations:

1. **File Upload**: Users upload files via the R Shiny interface.
2. **Data Processing**: The Python backend processes files for comparisons and data extraction.
3. **Results Display**: Processed outputs are displayed on the dashboard for review and validation.
4. **Export**: Users download outputs in their desired formats.

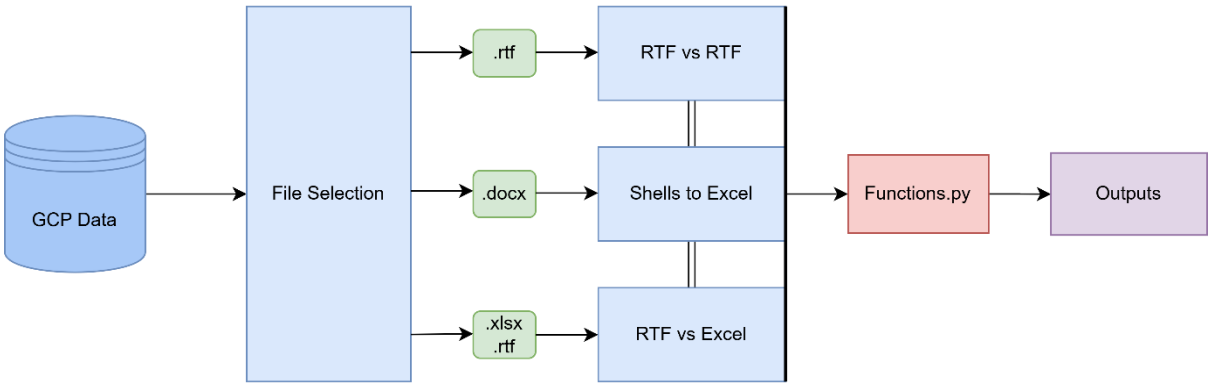

Figure 5: Overall workflow diagram

Spot-It in Action: Application Demonstration

To showcase Spot-It's capabilities, this section provides a visual walkthrough of the application in use, featuring screenshots of outputs at each stage. These examples highlight the interface, key features, and outputs, demonstrating how Spot-It streamlines quality control processes.

RTF vs RTF Comparison Output

We start by uploading two sets of RTF files, then pressing generate. In the following example, 4 pairs of RTF files have been uploaded. The output example with highlighting is shown below for the pair T1-1.rtf and T1-1b.rtf. The pairs can be selected using the drop-down box.

Figure 6: RTF vs RTF output example

This example comparison shows all three highlights in action as well as any extra lines.

The report tab can then be selected to show all of the comparison pairs

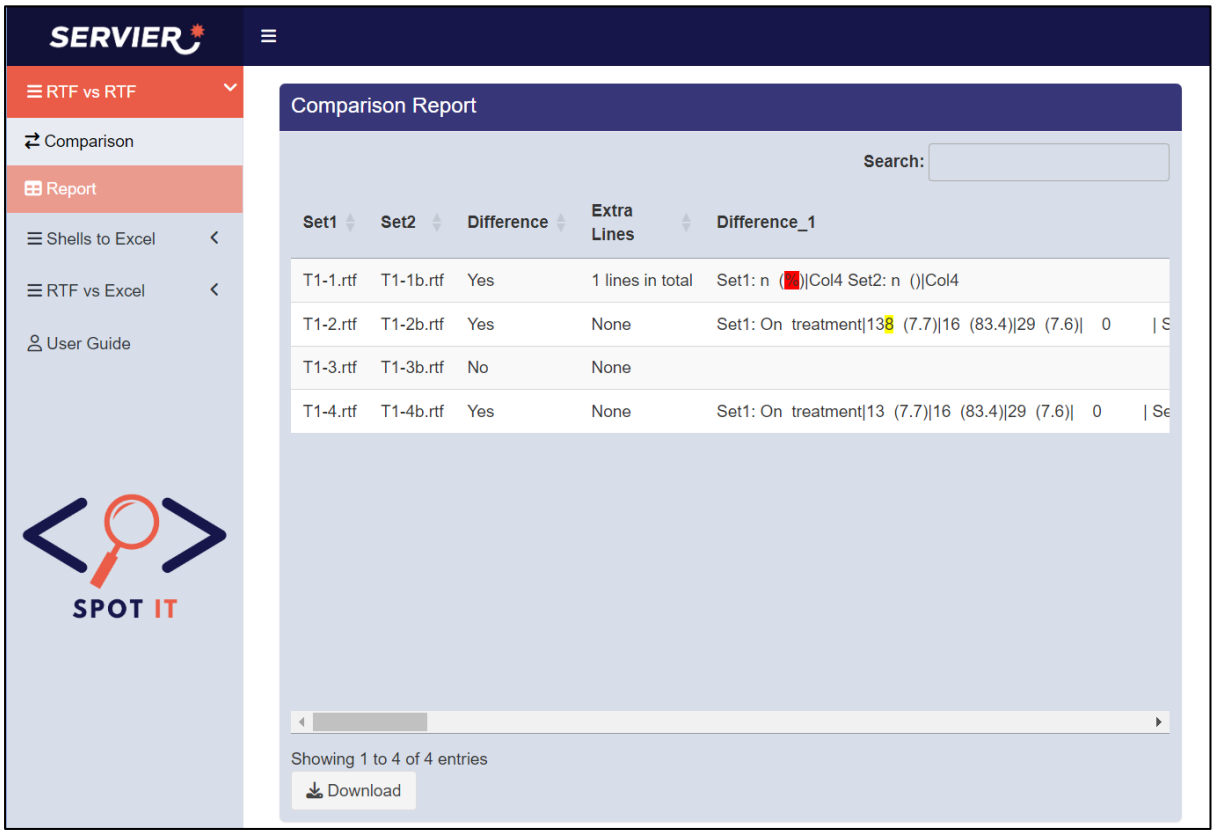

Figure 7: RTF vs RTF report example

Pressing the download button outputs the Excel file of all the comparisons

Set1	Set2	Difference	Extra Lines	Difference_1	Difference_2
T1-1.rtf	T1-1b.rtf	Yes	1 lines in total	Set1: n (%)|Col4 Set2: n ()|Col4	Set1: On treatment|13 (7.7)|16 (83.4)|29 (7.6)| 0 Set2: On treatment|14 (7.7)|16 (83.4)|29 (7.6)| 0
T1-2.rtf	T1-2b.rtf	Yes	None	Set1: On treatment|138 (7.7)|16 (83.4)|29 (7.6)| 0 Set2: On treatment|13 (7.7)|16 (83.4)|29 (7.6)| 0	Set1: Adverse event| 18 (57) | 3 (2.0) | 1 (4.0) | 0 (1.2) | Set2: Adverse event| 18 (57) | 1 (2.0) | 1 (4.0) | 0 (1.2) |
T1-3.rtf	T1-3b.rtf	No	None		
T1-4.rtf	T1-4b.rtf	Yes	None	Set1: On treatment|13 (7.7)|16 (83.4)|29 (7.6)| 0 Set2: On treatment|13 (7.70)|16 (83.40)|29 (7.60)| 0	Set1: Discontinued treatment| 5 (9.3)| 65 (1.6)| 76 (2.4)|12 (100) | Set2: Discontinued treatment| 5 (9.3)| 65 (1.6)| 76 (2.4)|12 (100.0) |

Figure 8: RTF vs RTF report output in excel

Shells to Excel Conversion Output

Extracted data from Word table shell documents are presented in a structured Excel format. The output includes organized columns for table numbers, titles, footnotes, and types. In the example below, a table shell .docx is uploaded and the generate

button is pressed.

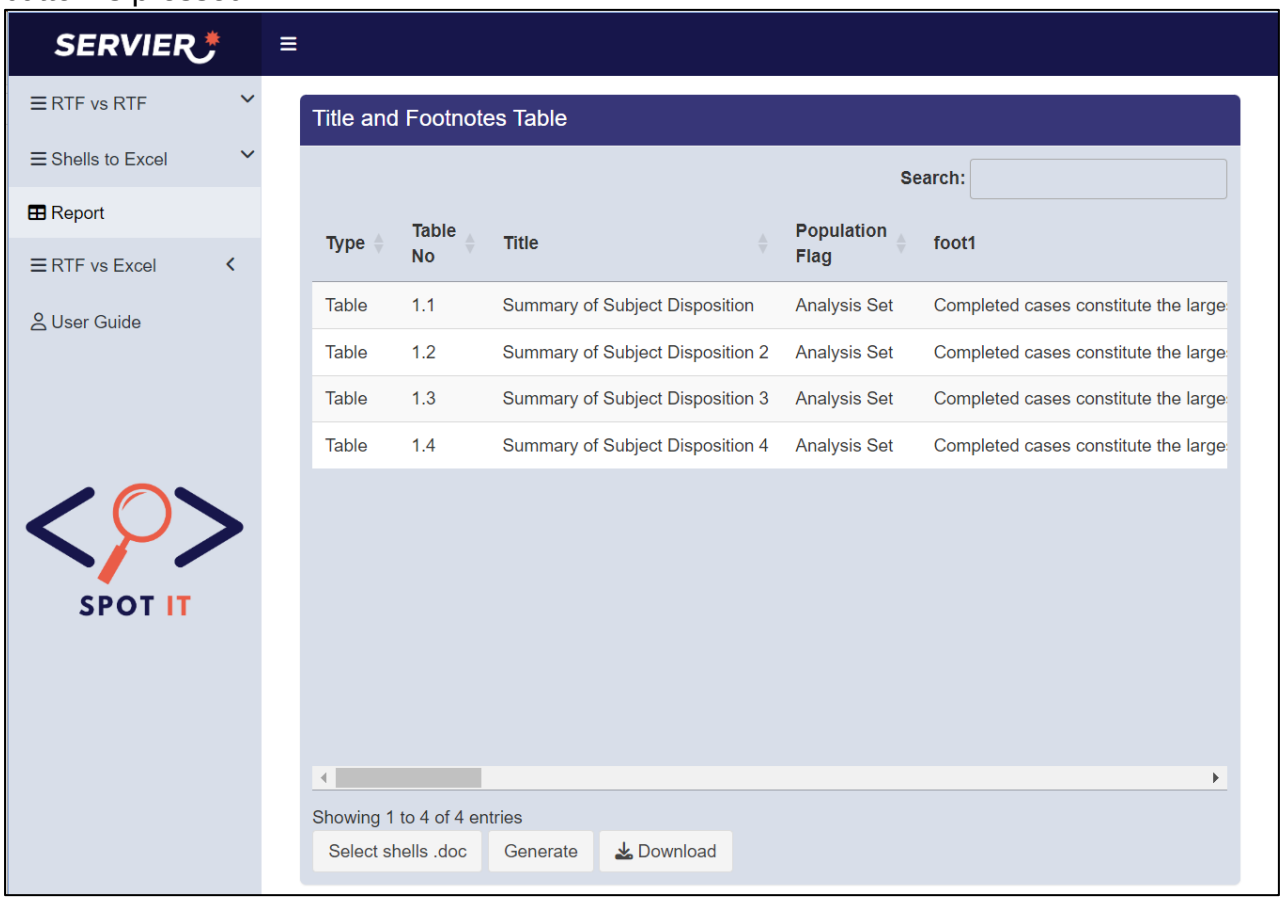

Figure 9: Shells to excel output example

The full report can be downloaded as an Excel output

Type	Table No	Title	Population Flag	foot1	foot2
Table	1.1	Summary of Subject Disposition	Analysis Set	Completed cases constitute the largest category, a	[1] Includes all cases that met the inclusion criter
Table	1.2	Summary of Subject Disposition 2	Analysis Set	Completed cases constitute the largest category, a	[1] Includes all cases that met the inclusion criter
Table	1.3	Summary of Subject Disposition 3	Analysis Set	Completed cases constitute the largest category, a	[1] Includes all cases that met the inclusion criter
Table	1.4	Summary of Subject Disposition 4	Analysis Set	Completed cases constitute the largest category, a	[1] Includes all cases that met the inclusion criter

Figure 10: Shells to excel output in excel

The backend code is designed to robustly capture as many table, listing, and figure titles and footnotes as possible from the shells document. However, achieving perfect accuracy for all table shells is challenging due to variations in document format and structure. Since table shells often lack a consistent format, it is crucial to double-check and update the output from this section before using it in subsequent processes.

RTF vs Excel Comparison Output

Once the excel output in the previous section is checked and edited, it can be used in the final section to compare title and footnote discrepancies between a list of RTF files and their counterparts in the table shells document. Below is an example output.

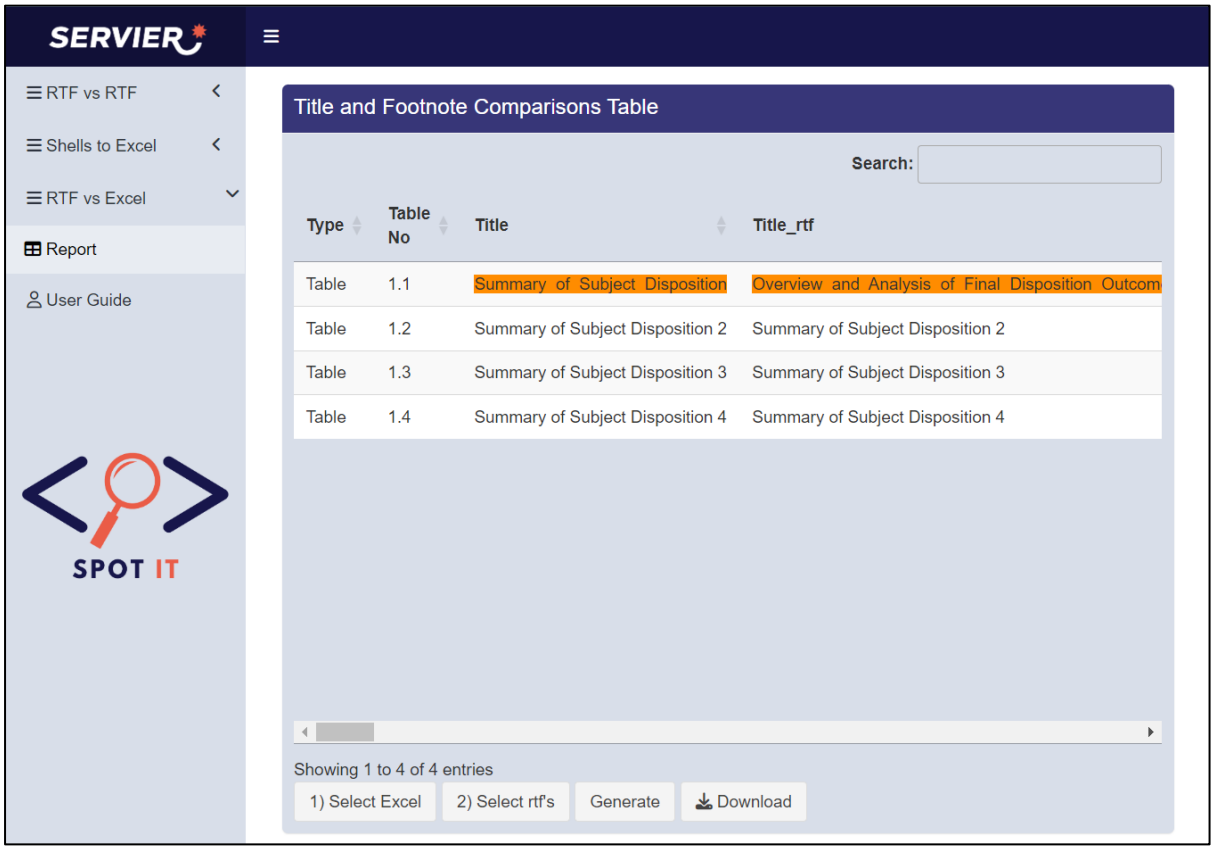

Figure 11: RTF vs Excel output example

Similarly, the final report can be downloaded as an excel file.

Type	Table No	Population Flag	Title	Title_rtf	Difference in title	Difference in footnotes	foot1
Table	1.1	Analysis Set	Summary of Subject Disposition	Overview and Analysis of Final Disposition Outcomes	Difference found	foot1; foot8; foot9	Completed cases constitute the largest category,
Table	1.2	Analysis Set	Summary of Subject Disposition 2	Summary of Subject Disposition 2	No difference found	foot1; foot4; foot7; foot8; foot9	Completed cases constitute the largest category,
Table	1.3	Analysis Set	Summary of Subject Disposition 3	Summary of Subject Disposition 3	No difference found	foot8; foot9	Completed cases constitute the largest category,
Table	1.4	Analysis Set	Summary of Subject Disposition 4	Summary of Subject Disposition 4	No difference found	foot1; foot8; foot9	Completed cases constitute the largest category,

Figure 12: RTF vs Excel output in excel

Challenges and Future Enhancements

Spot-It has proven to be a transformative tool in automating quality control (QC) processes for the pharmaceutical industry. It significantly reduces manual effort from days to minutes. The challenges faced during development, particularly in the backend, highlight the complexity of creating a solution that is both reliable and scalable. However, these challenges have been met with innovative solutions that deliver high-quality outputs tailored to the demands of QC teams.

One key challenge included the use of coloured text in Excel as it required precise formatting logic to maintain the consistency of styles/highlighting and ensure correct export results. The use of the xlsxwriter package enabled application of formatting, but ensuring compatibility and accuracy across varying requirements added complexity.

Another key challenge faced was accounting for the numerous differences in table shell formatting in the ‘Shells to Excel’ section. The back-end needed to account for different file structures, footnote/title positioning while maintaining accuracy and optimising text extraction. Implementing robust algorithms was critical to managing these differences, especially when handling inconsistent data. The integration of key Python libraries such as re was essential for preprocessing text data before export.

Looking forward, the planned enhancements - including AI-driven parsing - position Spot-It as a future-ready solution. By combining innovation with practicality, Spot-It exemplifies the potential of digital transformation in the pharmaceutical industry.

Tech Stack

The application is developed using a combination of Python and R, leveraging the strengths of both programming languages. Below is a categorized overview of the technologies and libraries used:

Core Libraries

Python was employed for data processing, text extraction, and file generation:

- **Data Analysis**: `pandas, numpy`
- **NLP and Text Processing**: `docx2txt, re, striprtf (via rtf_to_text)`
- **Data Structures and Comparison**: `collections (e.g., OrderedDict, Counter), difflib`
- **File Generation**: `xlsxwriter`

R was used for creating a web-based interface and for visualization.

- **User Interface and Dashboard**: `shiny`, `shinydashboard`, `shinydashboardPlus`, `shinyFiles`
- **Data Handling and Integration**: `reticulate`, `dplyr`, `readxl`, `openxlsx`, `writexl`
- **Visualization and Styling**: `bbcShiny`, `fresh`, `shinyalert`, `DT`

Integration

The integration of Python and R is facilitated by the reticulate package, enabling seamless communication between the two environments. This hybrid approach allows Python to handle computationally intensive tasks while R manages the user interface and visualization. This cohesive tech stack ensures the application is both powerful and user-friendly.

Conclusion

Spot-it represents a significant step in automating critical pharmaceutical processes, in particular oversight QC, eliminating the need for labour-intensive manual reviews and ensuring accuracy and consistency throughout.

The use cases and workflows illustrated in this paper highlight the application's transformative potential, aligning with the industry's shift toward digital transformation. As organizations increasingly adopt automation, tools like Spot-It stand poised to set a new benchmark for innovation and efficiency in quality control.

References

1. The pandas development team. (2023). *pandas: Python Data Analysis Library*. Retrieved from https://pandas.pydata.org
2. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … & Oliphant, T. E. (2020). *Array programming with NumPy*. *Nature*, 585(7825), 357–362. Retrieved from https://numpy.org
3. Shah, A. (n.d.). *docx2txt: A pure python-based utility to extract text and images from docx files*. GitHub. Retrieved from https://github.com/ankushshah89/python-docx2txt
4. McNamara, J. (n.d.). *xlsxwriter: A Python module for creating Excel XLSX files*. Retrieved from https://xlsxwriter.readthedocs.io
5. striprtf Contributors. (n.d.). *striprtf: Convert RTF documents to plain text*. Python Package Index (PyPI). Retrieved from https://pypi.org/project/striprtf/
6. Chang, W., Cheng, J., Allaire, J., Xie, Y., & McPherson, J. (2021). *shiny: Web Application Framework for R*. RStudio. Retrieved from https://shiny.rstudio.com
7. Ushey, K., Allaire, J., & Tang, Y. (2023). *reticulate: Interface to 'Python'*. RStudio. Retrieved from https://rstudio.github.io/reticulate/
8. Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag. Retrieved from https://ggplot2.tidyverse.org
9. Wickham, H., François, R., Henry, L., & Müller, K. (2023). *dplyr: A Grammar of Data Manipulation*. Retrieved from https://dplyr.tidyverse.org
10. Xie, Y., Cheng, J., & Tan, X. (2021). *DT: A Wrapper of the JavaScript Library 'DataTables'*. RStudio. Retrieved from https://rstudio.github.io/DT/
11. Chang, W. & Borges, B. (2023). *shinydashboard: Create Dashboards with 'Shiny'*. RStudio. Retrieved from https://rstudio.github.io/shinydashboard/

Contact Information

- **Pravin Lakkaraju**
  Company: Servier Pharmaceuticals
  Address: 200 Pier 4 Blvd., Boston, MA 02210
  Email: pravinlakkaraju1@gmail.com
- **Samiar Ashtiany**
  Company: Phastar
  Address: 2D Bollo Ln, Chiswick, London W4 5LE
  Email: samiar.ashtiany@phastar.com