Documentation:
Streamline all providers' Tv & Radio offer from PDF to a consolidated Excel

0. Overview

This project is a telecom offers aggregator that processes PDF files from Voo, Orange, Telenet and scrapes Base's offer from there website. It parses section titles and all channels text data into .tsv files and ultimately generates a final Excel report with all the processed information. The system relies on enablers scripts calling parsers, it includes automated post-processing and data cleaning functions to ensure that the final report is structured and accurate.

1. App reads the Channel Grouping reference data.
2. App reads all PDFs one by one.
3. If a PDF contains multiple pages, a prompt asks which pages to process.
4. Pages to process are saved in config so future runs are uninterrupted.
5. Section titles are extracted per PDF.
6. Text inside these sections is extracted per PDF.
7. Data is processed per provider and per year.
8. Regions are determined from PDF's filename or its internal codes.
9. Basic or Option offers are defined by section titles and internal codes.
10. TV or Radio is determined based on sections.
11. HD (>< SD) is specified only if mentioned in the Channel name (to-do).
12. Channel Group Level is matching channels with their grouping. (to-improve).
13. Summary table with counters is generated on a second Excel sheet.

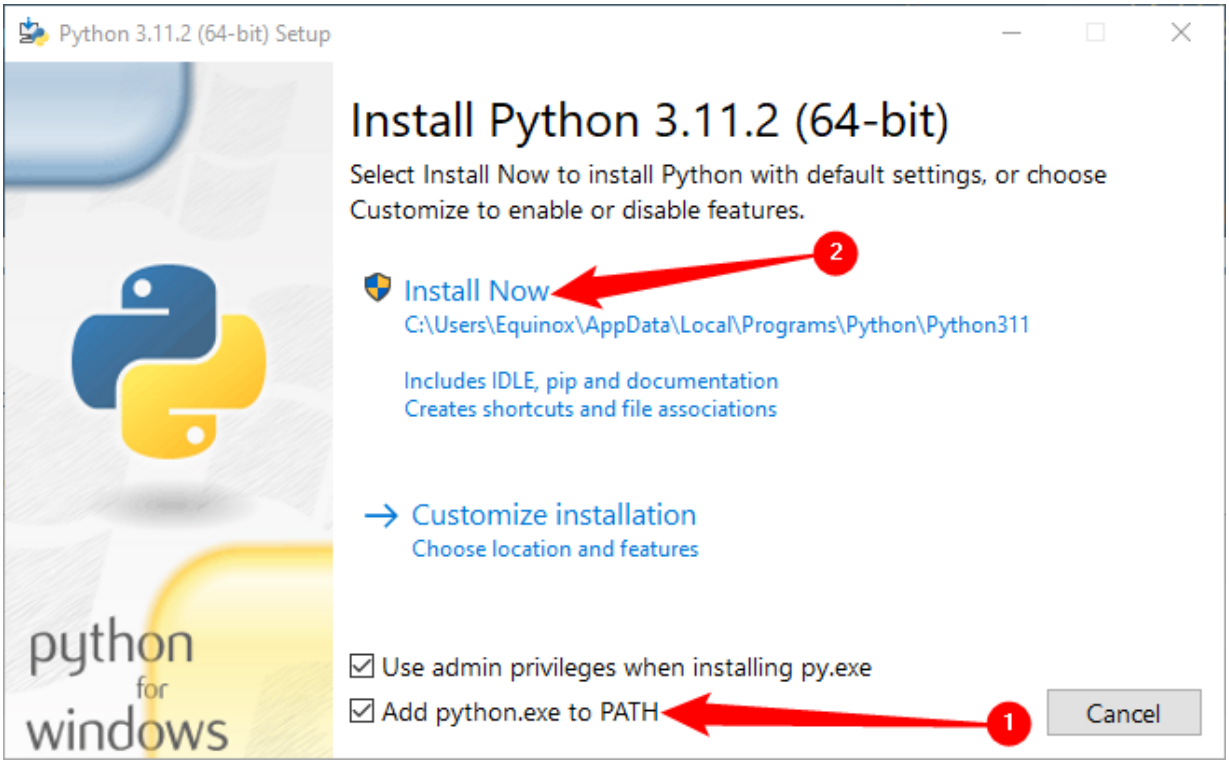
1 First Setup

Prerequisites:

Python

Download from Microsoft store at the following address. https://www.microsoft.com/store/productId/9NRWMJP3717K?ocid=pdpshare

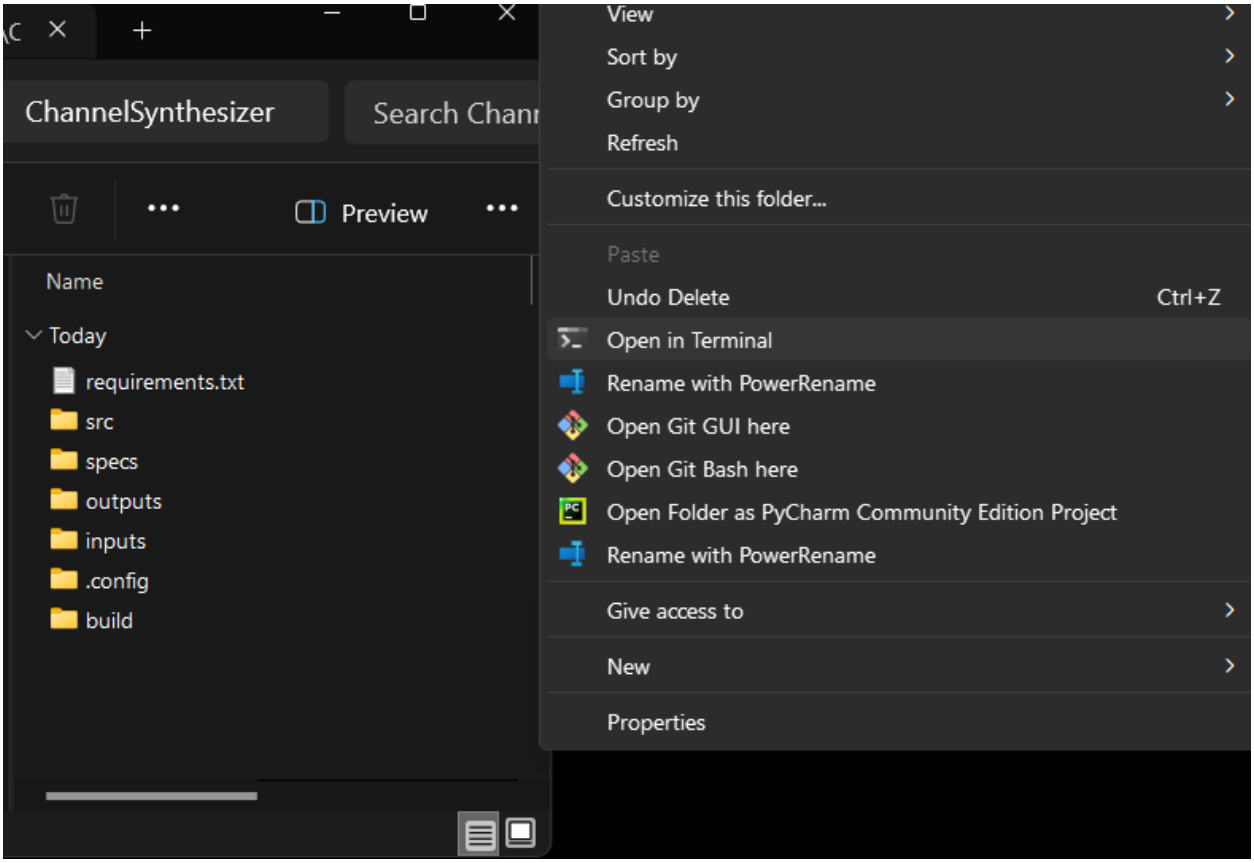
During installation process, ensure checking Add python.exe to PATH, otherwise Python won't be recognized as an available action.



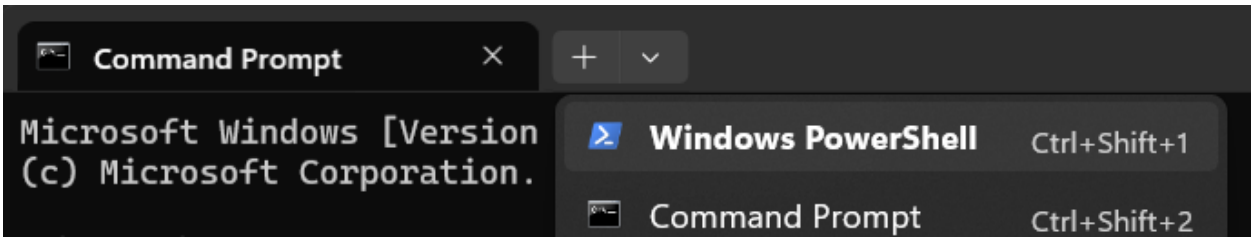
Test correct setup in whatever Windows terminal (Command Prompt or PowerShell) by typing **python *–version*** or ***python3 –version***. One of these should return a version number. If you already have another version of Python installed, you might want to skip installing 3.11. But keep that in mind if the project fails to set itself up.

Virtual Environment

Using Windows Explorer, navigate to the directory containing all the program's folders, right-click inside and select Open in Terminal.



And choose PowerShell in its dropdown menu, like so:



Now inside the Terminal, copy and paste these commands one by one:

python -m venv env

.\env\Scripts\activate

If typing the command ***pip*** alone returns 'command not found'/'not available', you need to install it using these two.

Invoke-WebRequest -Uri https://bootstrap.pypa.io/get-pip.py -OutFile get-pip.py

python .\get-pip.py

Otherwise, we simply update virtual env pip's version and install the dependencies.

pip install --upgrade pip or sometimes ***python -m pip install --upgrade pip***

pip install -r requirements.txt --no-deps

pip install openpyxl

2. Run instructions

Open a PowerShell terminal from inside the project's structure like in step 1. Activate the virtual environment needed for the project. Paste the following command:

.\env\Scripts\activate

Make sure neither PDF nor Excel files involved in this application are currently opened. Now using Windows Explorer, add all the PDF you need streamlined into one central Excel file. You must add them into the **ChannelSynthesizer*inputs**pdf*** folder, otherwise they won't be processed.

You must have a Channel Grouping reference file with a sheet called **Content_Channel_Grouping** containing column **CHANNEL_NAME** and column **CHANNEL_NAME_GROUP**. These exact values are needed for the process. You can find them hardcoded in **src\main.py** and **src\utils.py**.

There may be several Channel Grouping files, the program processes the most recent one based on the following filename format: Channel_Grouping_Latest_{TIMESTAMP}.xlsx. ie: **Channel_Grouping_Latest_20240607.xlsx**. These Channel Groupings must be in the folder called ***inputs*\.**

In PowerShell after activating the venv, paste the following command to start processing:

python .\src\main.py

The consolidated Excel files will open by itself in Excel as a result. It is generated by default in the ***outputs**xlsx*** location with the name **consolidated_report.xlsx**.

3. Workflow

3.1. Input Files and Data

PDF Files: The system takes in telecom provider PDFs (Voo, Orange, Telenet) that contain TV and radio channel information. They must be placed in folder ***inputs\pdf***.

Base offer's URL is configurable in the file ***src\main.py*** script on line 99. They don't produce PDF but only use

Channel Grouping Excel File: A file named Channel_Grouping_Latest_YYYYMMDD.xlsx provides metadata for merging channel names and grouping. Requires a sheet called **Content_Channel_Grouping** and the columns **CHANNEL_NAME_GROUP** and **CHANNEL**_NAME, these values are configurable in ***src\main.py*** and ***src\utils.py.***

3.2 Structure

main.py orchestrates all the workflow, Handles the end-to-end workflow, from loading PDF inputs to generating the final Excel report. It sends the input coordinates to the section, text and excel enablers in ***src\enablers***. These then call their respective parsing scripts in ***src\parsers*** and ***src\parsers\providers***. The Excel generation and post-processing is handled in ***src\utils.py***.

utils.py contains helper functions to read and process data, perform cleanups, and manage file operations.

src\outputs\section: Stores the parsed section data in .tsv format.

src\outputs\text: Stores the parsed text data in .tsv format.

outputs\xlsx: Where the consolidated Excel is initially saved and found.

inputs: Directory that stores the input files, such as the Channel_Grouping_Latest_*.xlsx file and the PDF files from providers in pdf\.

3.3. PDF Parsing

The system processes the PDF files in two stages:

Section Extraction: Extracts the sections of the PDF where relevant channel information is located. The per-PDF result is visible in location ***src\outputs\section***.

Text Extraction: Extracts the actual text, such as channel names and descriptions, from the designated sections. The per-PDF result is visible in location ***src\outputs\text***.

3.4. Data Consolidation

Once the parsing is complete, the system:

Merges the parsed data with the channel grouping data from the Channel_Grouping_Latest Excel file.

Processes the data to ensure the regions (Flanders, Brussels, Wallonia, Germanophone) are correctly assigned inside a function called create_consolidated_excel() in file ***src\utils.py***. Cleans up invalid channel entries and duplicates.

3.5. Excel Report Generation

The final output is an Excel file containing the consolidated and cleaned data:

TV/Radio Channels: The Excel file includes all the channels, grouped by provider, with details such as region availability, HD/SD status, and whether the channel is a basic or optional offering.

Summary Report: A summary table is also generated to give an overview of the number of channels in each region and provider.

3.6. Post-processing

The system handles post-processing to standardize channel group naming conventions, ensuring consistency between providers.

The cleaning phase removes any unwanted data, including irrelevant entries or rows that don’t follow the expected patterns.

4. Key Functions

main.py

main()

- Handles the following:
- Searches for the latest Channel_Grouping_Latest_YYYYMMDD.xlsx file and verifies that it's not open.
- Processes the PDFs using the process_sections() and process_pdfs() functions.
- Generates the Excel report by merging parsed data with the channel grouping data, followed by cleaning operations.
- Adds extra data from BASE provider.
- Opens the final report using the system's default app.

utils.py

create_consolidated_excel()

- Builds the consolidated Excel from all the parsed and scraped data.

get_provider_and_year()

- Identifies the provider and year from the file name of the PDF.

read_section_names()

- Reads section names from a text file for section identification during parsing.

parse_tsv()

- Parses the .tsv files generated by the PDF parsers and cleans out unwanted or irrelevant data.

ensure_region_columns_exist()

- Ensures that all required region columns (Flanders, Brussels, Wallonia, Germanophone) are present in the data.

post_process_orange_regions()

- Verifies the correctness of the region data for Orange and fixes any inconsistencies.

create_summary_table()

- Generates a summary table from the consolidated data, showing the total channels per provider, region, and type (Basic or Option).

enablers

sections.py

- Calling the parsing of the sections from the PDFs located in ***inputs\pdf***.

text.py

- Extracts the text from specific sections of the PDFs located in ***inputs\pdf***.

excel.py

- Send input data to the Excel generation.

parsers\providers

base.py

- Scrapes BASE's offer table and apply very little transformation to only extract wanted data.

orange.py

- Responsible of parsing all PDFs containing ***Orange*** in their filename.

telenet.py

- Responsible of parsing all PDFs containing ***Telenet*** in their filename.

voo.py

- Responsible of parsing all PDFs containing ***Voo*** in their filename.

JSON Configuration

The system includes a JSON configuration as file ***.config\page_selection.json*** that specifies which pages to parse from each PDF, ensuring that the same PDF is not parsed twice for identical content and that un-

parsable pages (ie: image-based with few text) don't contaminate the dataset. This allows for automated, repeatable parsing without re-prompting for each file.

5. Troubleshooting

Common Issues

File Not Found: If the script can't find the Channel_Grouping_Latest file, ensure that it's placed in the inputs/ directory and follows the naming format Channel_Grouping_Latest_YYYYMMDD.xlsx.

Sheet, Columns or Data not found: Ensure correct Sheet, Columns values in ***src\utils.py***.

Permission Denied: If you receive a message about a file being open, close the Excel file and rerun the script.

Debugging

Print statements are in main.py to trace the flow of execution and verify that each step (loading, parsing, merging, etc.) is completed successfully. You can see the log stream in your terminal after entering command ***python .\src\main.py***.

6. Future Enhancements

Additional Providers: Adding more providers to the system can be done by updating the parser and modifying the regular expressions to identify new channel groupings.

Error Handling: Improve error handling for malformed PDFs and missing data in Excel files.

Error Handling: Improve error handling for malformed PDFs and missing data in Excel files.

Cloud Integration: The script can be adapted for cloud storage, allowing PDFs and output reports to be managed in cloud platforms like AWS or Google Cloud.

Channel Grouping and Maintenance: Ensure that all channels are derived from the official channel grouping file for a straightforward maintenance process.

Additional Channel Providers: Ensure that Proximus and Scarlet are added to the listing and properly mapped in the channel grouping file.

Region-Specific Data Extraction: Ensure correct region-based data extraction, e.g., having a channel listed once for regions it is available in, with binary values indicating availability across regions.

Address regions separately when multiple PDFs are available (e.g., Brussels and Flanders for Telenet).

BASE Provider: Determine and clarify the availability of BASE TV channels across regions, including "chaînes néerlandophones" and "chaînes francophones."

7. Contact Information

If you need further assistance or have questions about the system, please contact the previous developer at david.botton@hainaut-promsoc.be. For urgent technical issues, reach out to the project lead at yasser.eljasouli-ext@bd-orange.com.