Проект по разработке модуля парсинга PDF файлов и экспорта целевых данных в БД

Концепция проекта

Код модуля сделан на языке Python. Код написан в виде отдельных функций. Функции разбиты по файлам (модулям проекта) в соответствии с описанной ниже логикой. В качестве целевой библиотеки для PDF парсинга выбрана PyMuPDF (модуль fitz) как самая эффективная с точки зрения требуемой памяти и скорости работы. Для работы с данными выбрана библиотека Pandas, так как она имеет встроенные функции для коннекта с БД и гибкие настройки по изменению структуры и набора требуемых полей.

Модуль разбит на 3 этапа, что обеспечивает эффективную работу функций внутри каждого из этапов и удобство доработки кода при необходимости.

Точность модуля составила 100% (из 963 строк только 3 строки оказались поврежденными)

Алгоритм работы модуля

№	Цель этапа	Решаемые задачи	Файл с кодом / Используемые библиотеки
1	Трансформировать PDF файл в структурированный массив строковых данных	1) Принять на вход pdf файл, состоящий из двух разделов 5.2 и 5.3 (номер страницы, разделяющей разделы, заранее известен). 2) Трансформировать PDF файл в нормализованный (“очищенный”) и структурированный двухуровневый список, обеспечив соответствие список = параграф (5.2.Y.XXX или 5.3.XXX соответственно). Таким образом, чтобы длина списка была равна количеству параграфов = 1149 (в т.ч. 180 параграфов в разделе 5.3) 3) Разбить общий массив данных на два - master (5.3) & slave (5.2), для повышения скорости обработки информации на дальнейших этапах	pdffiletotext.py `fitz`

2	Трансформировать двухмерный массив корректных данных в дата фрейм	1) Принять на вход два двухуровневых списка (master & slave)
		2) Определить целевую структуру данных, необходимых для сбора, очередность сбора, анализа и проверки корректности собранных данных как из master, так и slave списков. Структура json файла имеет избыточное количество полей, но на данном этапе это необходимо для дополнительных проверок и кросс чеков
		3) Экспортировать данные из мастер файла в json файл и затем обновить json файл данными из slave раздела. Обеспечить достижение 100% заполнения и корректности всех данных (% брака составил = 0.31% см прил. #1)
		4) Импортировать данные из json в дата фрейм пандас. Набор столбцов дата фрейма должен строго соответствовать ТЗ. Количество строк дата фрейма = 963
			dataframe_create.py
			`pandas`
			`json`

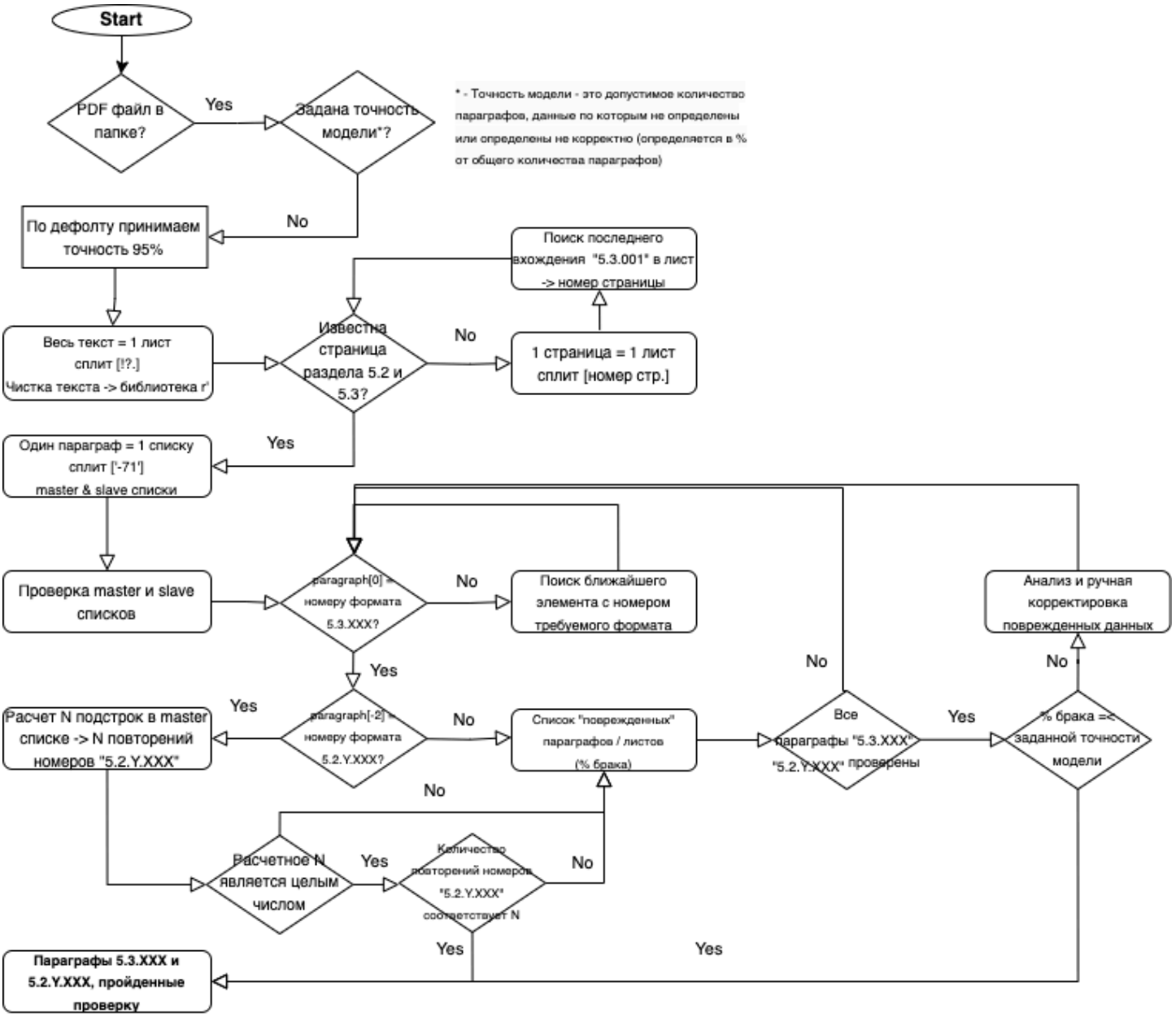
№	Цель этапа	Решаемые задачи	Файл с кодом / Используемые библиотеки

3	Обеспечить эффективный экспорт целевых данных в БД	1) Принимает на вход дата фрейм из п.3 и экспортирует полученные данные в базу данных PostgreSQL (названия столбцов из дата фрейма переносятся в psql без изменений). Все авторизационные данные для связи с PSQL занесены в переменные окружения. Количество строк в БД = 963 .	dataframe_export.py
			`psycopg2`
			`sqlalchemy`
			`dotenv`
			`os`

4		Для повышения скорости и качества работы скрипта налажены процессы проверки данных на всех этапах выше	cross_checks.py

Логика работы модуля

Этап #1
Цель: Трансформировать PDF файл в структурированный массив строковых данных



Для запуска программы необходимо:

1) Скопировать с GitHub все папки и файлы на свой ПК. Необходимо сохранить структуру расположения папок
2) скопировать PDF файл с именем “SAE J1939-71.pdf” в папку "PDF_files" (по дефолту данный файл уже там находится).
3) прописать переменные окружения со своими авторизационными данными для доступа к postgreSQL.
    a) Необходимо через терминал зайти в папку **/code_script** и последовательно запустить следующие команды:
    export DB_NAME=<указать наименование БД> (наименование любое)
    export DB_HOST=localhost
    export DB_PORT=5432
    export DB_LOGIN=postgres
    export DB_PASSWORD=<указать свой пароль>
    b) в папке “code_script” создать файл “.env” и прописать там построчно следующие данные:

```
DB_LOGIN=postgres
DB_PASSWORD=<указать свой пароль>
```

```
DB_PORT=5432
DB_HOST=localhost
DB_NAME=<указать наименование БД>
```

4) создать БД:
    a) через терминал запустить программу:
    createdb -U postgres <указать наименование БД> (должно совпадать с п.3)
    b) через DBeaver (или любую другую СУБД) создать профиль БД PostgreSQL с аналогичным именем (проверить подключение)
5) установить все зависимости из файла requirements.txt (файл должен находиться в корневом каталоге проекта). запустить через терминал IDE команду **pip install -r requirements.txt**
6) запустить на выполнение файл **code_script/main.py**
7) Результат работы - файл с расширением .sql - в папке:
Backup_files/spare_parts_202402221933.sql и количество записей БД

Приложение #1

Список поврежденных строк в json файле

{'MPRGR': '5.3.074', 'ID': 'DE00', 'PGN': '56832', 'Data_Length': '8', 'Length': 41, '_Name_': 'Service Component Identification', 'SPN Doc': '5.2.5.102', 'Scaling': None, '_Range_': None, 'SPN': None}

{'MPRGR': '5.3.103', 'ID': 'FE94', 'PGN': '65172', 'Data_Length': '8', 'Length': 38, '_Name_': 'Sea Water Pump Outlet Pressure', 'SPN Doc': '5.2.7.???', 'Scaling': None, '_Range_': None, 'SPN': None}

{'MPRGR': '5.3.145', 'ID': 'FE6B', 'PGN': '65131', 'Data_Length': 'Variable', 'Length': 34, '_Name_': '10/1/1998', 'SPN Doc': '1626 -71', 'Scaling': None, '_Range_': None, 'SPN': None}

Всего повреждено 3 строки из 963, что составляет 0.31%