МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский технологический университет “МИСиС”»

НИТУ МИСИС

Институт информационных технологий

Кафедра инфокоммуникационных технологий

Отчет по научно-исследовательской работе на тему «Механизм определения структуры документа»

Выполнили студенты группы МИВТ-22-5:

Маковецкий Иван

Мишталь Кирилл

Пехотин Влад

Талдытова Дина

Москва — 2024

Оглавление

Стр.

Часть 1. Постановка задачи

1.1 Вводные

В компании N планируется разработать модель машинного обучения, которая сможет определять структуру документа (вложенные списки) и внедрить в бизнес-процесс по анализу документов в предприятии K.

Структура документа — это преобразование документа в древовидную структуру, каждый узел которой имеет какое-либо логическое определение или объяснение.

1.2 Развитие

Автоматизация полного цикла внедрения ML-сервиса:
– Разметка данных.
– Обучение модели.
– Применение модели.
– Доразметка данных или активное обучение.
– Дообучение модели.

1.3 Техническое задание

1. Должна быть разработана схема взаимодействия пользователя с системой.
2. Должна быть разработана схема взаимодействия (API) между компонентами ML-сервиса.
3. Должна быть разработана модель, которая принимает на вход текст документа и возвращает структуру документа.

4. Должен быть разработан отчет о ходе работ. Отчет должен включать следующую информацию:
    а) Описание данных (определение типов данных, используемых в сервисе, и их структура).
    б) Эксперименты с моделями (описание экспериментов, проведенных при разработке модели, включая выбор датасетов, параметров обучения и результаты).

Часть 2. Ход работы

2.1 Описание данных

Предоставленные данные — нормативно-правовые акты из Единого реестра образовательных документов (EPOT), 2000 документов в формате Portable Document Format. Несмотря на применение обработки естественного языка в широком круге задач при обработке PDF, большинство алгоритмов, использующих NLP, подразумевают использование «чистых данных». Для успешного определения целевой информации необходимо выявить иерархическую структуру для удаления незначительной информации из документа, к примеру, номера страницы.

Мы придерживаемся метода, разработанного Yuta Koreeda и Christofer D. Manning в статье «Захват логической структуры визуально структурированных документов с помощью мультимодального анализатора переходов» (Capturing Logical Structure of Visually Structured Documents with Multimodal Transition Parser). Этот метод позволяет практически не использовать разметку и оперировать высокоуровневыми признаками: визуальными (отступами, интерльяжем) и текстовыми (пунктуацией, номерами в оглавлении).

На вход сервиса поступает PDF-файл для анализа, пользователь получает структуру документа в виде дерева в формате JSON.

Для каждого документа выполняются следующие шаги:

1. Создается пустой словарь `spans` для хранения информации о блоках текста.
2. Для каждого блока текста в документе выполняются следующие действия:
    - Строится матрица иерархии `(m)` с использованием функции `create_hierarchy_matrix`.
    - Для каждого индекса блока текста проверяются отношения с другими блоками текста.
    - Информация о каждом блоке текста добавляется в словарь spans.

3. Формируется путь вывода (out_path) для каждого документа и сохраняется в формате JSON с использованием json.dump.

2.2 Описание модели

2.2.1 Гипотезы

В ходе работ были предложены три основные методологии выполнения работы:

1. Использование больших языковых моделей для выявления структуры. Достоинства такого метода: отстутствие необходимости разметки данных, возможность вывода в формате, нужном пользователю, способны улавливать сложный контекст и взаимосвязь между разными частями документа, автоматически извлекать признаки и паттерны. Недостатки такого метода: требования к вычислительным ресурсам (как для обучения, так и для инференса), неясность интерпретации принятия решения моделью, возможность «галлюцинаций» в выводе, сложность применения в доменной области из-за чувствительного характера данных.
2. Использование XML-структур и заголовков для выявления структуры документа. Достоинства такого метода: XML обеспечивает явную иерархию элементов, документы с явной структурой могут быть легко интерпретированы и поняты человеком без необходимости автоматического анализа, требуется малое количество вычислительных ресурсов. Недостатки такого метода: XML может не явно представлять некоторые отношения между различными частями документа, XML-структуры могут оказаться не достаточно гибкими для представления динамичных или сложных структур, в некоторых случаях автоматический анализ XML может требовать дополнительной обработки для выделения семантической структуры.
3. Использование ручных признаков и классификатора на основе машинного обучения. Достоинства:

а) Возможность включения в модель различных признаков, таких как визуальные, текстовые и семантические, что может улучшить способность модели к анализу контента.

б) Может быть эффективным в случаях, когда у нас есть ограниченное количество обучающих данных, что важно в юридической области, где многие данные являются закрытыми.

в) Ручные признаки обычно более легко интерпретируются, что облегчает понимание принимаемых моделью решений.

г) Обучение модели на основе ручных признаков и классификатора может быть более быстрым и требовать меньше вычислительных ресурсов по сравнению с глубоким обучением.

Недостатки:

а) Модели, основанные на ручных признаках, могут оказаться менее способными к автоматическому выявлению сложных и абстрактных паттернов, которые могли бы быть выучены глубокими нейронными сетями.

б) Результаты сильно зависят от качества выбранных ручных признаков, и неправильный выбор может привести к ухудшению производительности.

в) Модель, основанная на ручных признаках, может иметь ограниченные возможности в обобщении на новые и неизвестные сценарии, особенно если они сильно отличаются от обучающих данных.

г) Сложность учета всех видов информации (визуальной, текстовой, семантической) может потребовать дополнительной тщательной обработки.

Исходя из технического задания и доменной области был выбран вариант с использованием ручных признаков и классификатора на основе машинного обучения.

2.2.2 Признаки, используемые моделью

1. Отступы (вверх, вниз или одинаковые). Описывает отношение отступов между разными уровнями текстовых блоков (1-2, 2-3).
2. Отступы после отстутствия нумерации.
3. Центрированный. Описывает, является ли текст центрированным.
4. Разрыв строки перед правым краем.
5. Перенос на новую страницу.
6. В верхних 15% страницы. Описывает, находится ли блок в верхних 15% страницы.
7. В нижних 15% страницы. Описывает, находится ли блок в нижних 15% страницы.
8. Увеличенный межстрочный интервал. Описывает, есть ли увеличенный межстрочный интервал.
9. Выровнен по ширине с пробелами посередине. Описывает, выровнен ли текст по ширине с пробелами посередине.
10. Похожий текст в похожей позиции. Описывает наличие похожего текста в похожей позиции.
11. Выделение пробелами между символами. Описывает выделение пробелами между символами.
12. Выделение скобками. Описывает выделение текста скобками.

2.2.3 Эксперименты и метрики

Конечная цель оценки эффективности системы заключается в том, чтобы понять, насколько хорошо она справляется с поставленной задачей. В данном случае система рассматривается как инструмент для извлечения информации из текста.

Первый набор метрик (IE-перспектива)

– Однозначный абзац: Метрика F1 оценивает, насколько успешно система идентифицирует пары блоков, находящихся в пределах одного абзаца.

– Сиблинги: Аналогично, система оценивается на способность выявлять отношения сиблингов между блоками.
– Отношения предок-потомок: Эта метрика оценивает способность системы выявлять отношения между блоками в иерархии предок-потомок.

Второй набор метрик (Предварительная обработка для NLP):

– Выделение границ абзацев: Эта метрика оценивает, насколько точно система определяет границы абзацев. Это важно для того, чтобы правильно передавать фрагменты текста в технологии обработки естественного языка (NLP).
– Точность удаления мусора с опущенным абзацем: Метрика отражает эффективность системы в удалении блоков с опущенными изменениями.

Был использован Random Forest в качестве классификаторов для определения переходов и указателей. Random Forest подходит для работы с категориальными признаками, которые составляют основную часть наших характеристик (features).

Random Forest представляет собой ансамбль решающих деревьев, где каждое дерево принимает решение, и окончательный результат определяется голосованием или усреднением результатов отдельных деревьев. Этот метод хорошо подходит для задач классификации, особенно когда признаки могут быть категориальными.

В ходе экспериментов были проведены исследования важности каждой функции при использовании методов жадного прямого отбора (greedy forward selection) и жадного обратного исключения функций (greedy backward elimination). В ходе исследования было выяснено, что система балансированно использует визуальные и текстовые подсказки.

Некоторые параметры, такие как «Отступ», «Больший интерлиньяж» и «Иерархия нумерации (T1)», которые частично представляют собой целевые метрики, оказались высоко оцененными во многих случаях. В то же время другие параметры, такие как «Все заглавные буквы» и «Пунктуированные», также внесли значительный вклад в точность.

2.3 Описание API

Описание представлено на рисунке 2.1. API предназначено для обработки PDF-файлов и предсказания структуры документа с использованием библиотеки `pdf_struct`.

1. **Загрузка PDF-файла:** Пользователь отправляет POST-запрос на эндпоинт `/pdf`, включая PDF-файл в теле запроса. PDF-файл загружается с использованием `UploadFile` из FastAPI.
2. **Сохранение PDF-файла:** Содержимое загруженного PDF-файла читается и сохраняется на сервере в директории `pdf` с использованием уникального имени файла.
3. **Предсказание структуры PDF:** Загруженный PDF-файл передается в функцию `pdf_struct.predict`, которая выполняет предсказание структуры документа. Параметры функции указывают формат вывода (`format='tree'`), путь к PDF-файлу (`in_path=path_to_pdf_file`), и используемую модель (`model='PDFContractEnFeatureExtractor'`).
4. **Сохранение результата в текстовый файл:** Полученная структура документа преобразуется в строку и сохраняется в текстовом файле в директории `out`. Этот файл имеет уникальное имя, содержащее текущую дату и время.
5. **Отправка результата пользователю:** Текстовый файл с результатами отправляется пользователю в качестве ответа на запрос с использованием `FileResponse`. Медиа-тип указывается как `"application/text"`, и имя файла также включает текущую дату и время.
6. **Запуск и обслуживание сервера:** Если скрипт запускается напрямую (`if __name__ == "__main__":`), то сервер запускается с использованием `uvicorn` на localhost, порту 8000, с возможностью автоматической перезагрузки и тремя рабочими процессами.

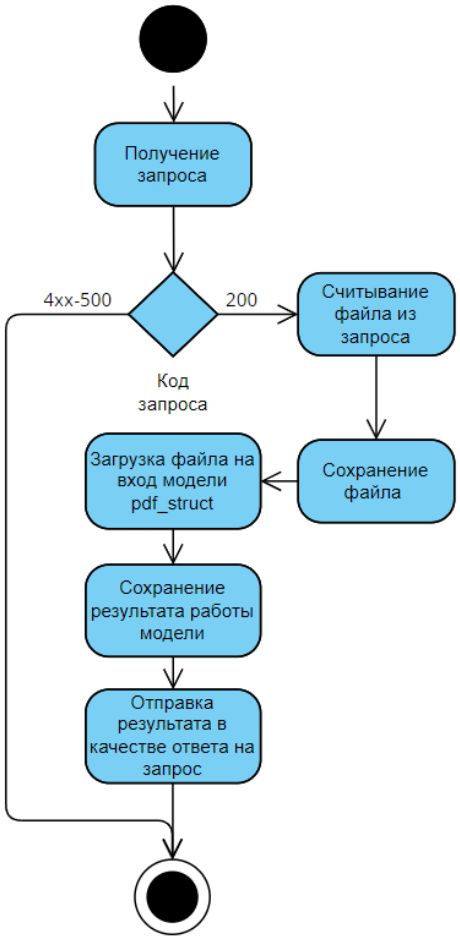

Рисунок 2.1 — Реализация API

2.4 Описание взаимодействия пользователя с системой

Код на Python использует библиотеку Aiogram для создания бота в Telegram. Описание представлено на рисунке 2.2.

1. **Настройка команд бота:** Бот настроен на использование команд, таких как `/start` и `/help`. Команды устанавливаются с описанием, которое будет отображаться при запросе команд.
2. **Обработчики сообщений:**
   a. `get_start`: Этот обработчик вызывается при команде `/start` и приветствует пользователя, предлагая выбрать тип загружаемого файла.
   b. `start_dow`: Обработчик, который запускается, когда пользователь выбирает загрузку PDF файла. Бот предлагает загрузить PDF файл.

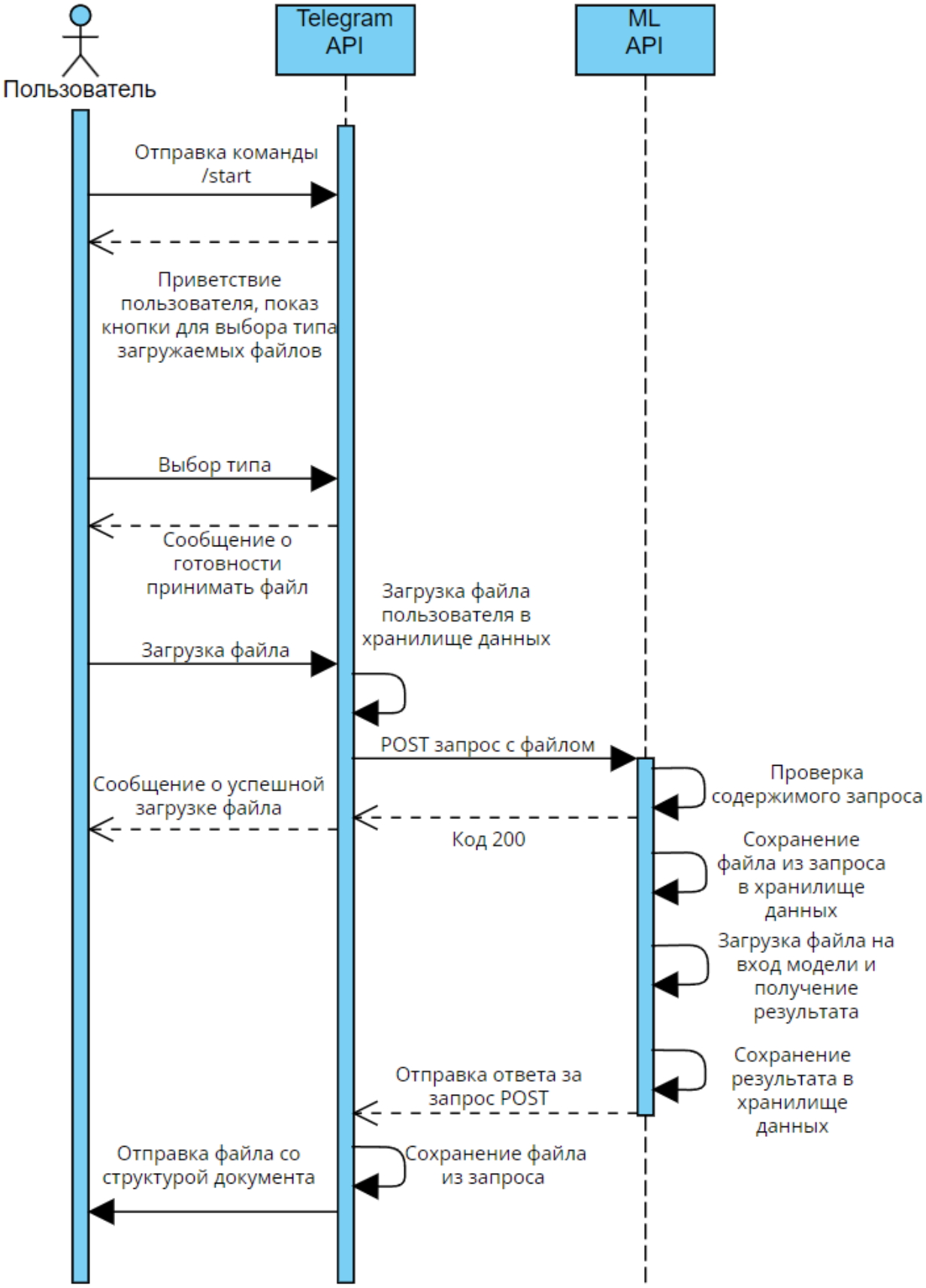

Рисунок 2.2 — Описание взаимодействия пользователя с системой

   c. `dow_file`: Обработчик загрузки файла. После того как пользователь загружает PDF файл, бот сохраняет его локально, отправляет на внешний сервер по адресу http://localhost:8000/pdf, получает ответ и отправляет результат пользователю.

3. **Клавиатура для выбора действий:** Создается клавиатура с единственной кнопкой «Загрузить PDF». Эта клавиатура используется для удобного выбора действия при взаимодействии с ботом.
4. **Основной блок кода:** В основном блоке кода создается экземпляр бота и диспетчера. Запускается функция `start()`, которая настраивает логирование, устанавливает команды бота и начинает прослушивание событий (`polling`).
5. **Состояния для управления процессом загрузки:** В коде также определено состояние `DowPDF` с использованием `StatesGroup` из Aiogram. Это состояние используется для отслеживания этапа загрузки PDF файла.
6. **Общий процесс работы бота:** Пользователь начинает с команды `/start`, затем выбирает загрузку PDF файла, загружает файл, и бот обрабатывает этот файл, отправляя его на внешний сервер и возвращая результат обратно пользователю.

Часть 3. Выводы

В результате проведенных работ была разработана система с использованием машинного обучения для определения структуры документа (вложенных списков).

Модель показала хорошие результаты на тестовом наборе данных, F1-мера для всего документа = 0.953.

Модель реализована в виде API, которое может быть использовано для обработки PDF-файлов и предсказания их структуры. API реализовано с использованием FastAPI и позволяет загружать PDF-файлы, обрабатывать их и возвращать результат пользователю в виде JSON-объекта.

Также была разработана схема взаимодействия пользователя с системой с использованием бота в Telegram. Бот позволяет пользователю загружать PDF-файлы и получать результат предсказания структуры документа.