Извлечение иерархической логической структуры из текстовых документов в формате docx

---

Автор: Богатенкова Анастасия Олеговна
Научный руководитель: Гомзин Андрей Геннадьевич
Научный консультант: Козлов Илья Сергеевич

2 июня 2021 г.

Актуальность задачи

- В мире каждый год создаётся большое количество документов.
- Для автоматической обработки (например, поиска по документам, суммаризации и т. д.) необходимо представить документ в удобном для обработки виде, т. е. выделить его структуру.
- Для этих целей в ИСП РАН разрабатывается программный модуль docreader, основаный на проекте dedoc[1].

[1] https://github.com/ispras/dedoc

Типы структуры документа

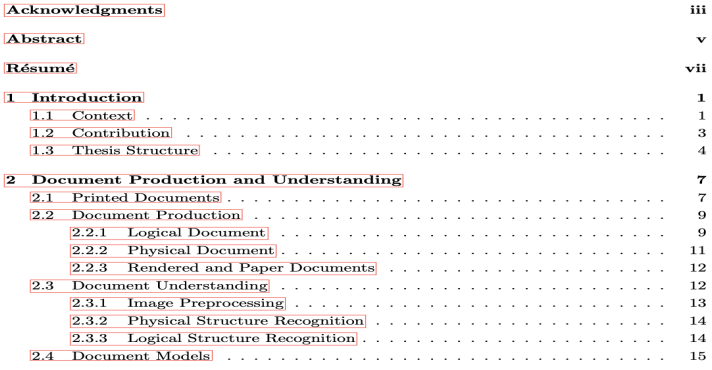
Для документов можно определить три основных типа структуры:

1. **Физическая структура** – внешний вид документа (**жирность** шрифта, две колонки).
2. **Логическая структура** описывает разбиение документа на компоненты (например, законы состоят из глав, главы разбиваются на статьи и т. д.).
3. **Семантическая структура** связана с задачей понимания содержимого текста.

В данной работе рассматривается логическая структура документа.

Логическая структура документа

Компоненты логической структуры зависят от предметной области. Например, научные статьи состоят из аннотации, введения, обзора существующих работ и других секций.



Особенности docx

- Большое количество документов создаётся и хранится в формате docx.
- Формат docx в основном ориентирован на отображение физической структуры документа.
- Библиотеки по работе с форматом существуют[2], однако не позволяют выделить в документах все стили и текст нумерации элементов списков.
- В документации формата документ описывается последовательностью **параграфов**, разделённых переносом строки. Далее будем использовать термин «параграф» в этом смысле.

[2] Рассматривается язык программирования python

Постановка задачи

- Формально описать извлекаемую иерархическую логическую структуру;
- Реализовать метод построения иерархической логической структуры из документов в формате docx на примере технических заданий;
- Провести оценку качества реализованного метода;
- Реализовать извлечение текста и необходимых метаданных из документов в формате docx. Встроить реализованный метод в открытый проект по обработке документов dedoc[3].

[3] https://github.com/ispras/dedoc

Структура технического задания

Технические задания имеют определённую логическую структуру и описываются ГОСТами[4], однако на практике от них часто отходят. По итогам изучения существующих работ и анализа примеров технических заданий, предложена структура ТЗ следующего вида: дерево произвольной глубины, в котором каждому узлу соответствует параграф документа с типом (заголовок, содержание, часть, элемент списка или текст).

[4] http://technicaldocs.ru/гост19/шаблоны/техническое_задание

Описание метода

Ставится задача построения дерева с типизированными узлами. Задача построения дерева требует решения следующих подзадач:

Определение типа параграфов

Каждому параграфу сопоставляется один из 5 классов: *заголовок*, *содержание*, *часть*, *элемент списка* или *текст*.

Построение дерева

Определение вложенности параграфов на основе попарного сравнения.

Схема работы метода

Описание классификаторов

Классификация параграфов

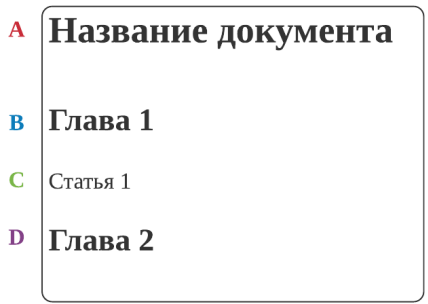
- Классификация параграфов на 5 классов: заголовок, содержание, часть, элемент списка или текст.
- Формирование векторов признаков для параграфов, обучение классификатора (XGBoost) и корректировка ответов классификатора.

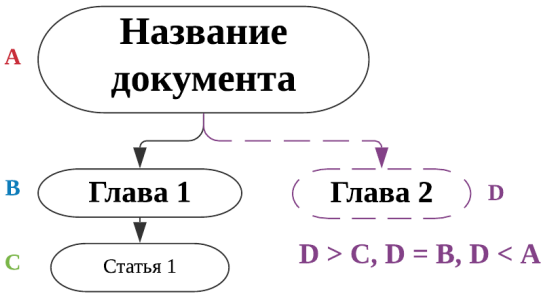
Сравнение параграфов

- Классификациия **пар** параграфов документа на 3 класса: больше, меньше или равно.
- Формирование векторов признаков для **пар** параграфов и обучение классификатора пар параграфов (XGBoost).

Построение дерева

- Добавляем параграф «Глава 2» в дерево.
- Сравниваем параграфы попарно для определения места вставки.





Результат работы алгоритма построения дерева

Название документа (0 уровень)

Глава 1 (1 уровень)

Содержимое главы 1 (2 уровень)

Статья 1 (2 уровень)

Содержимое статьи 1 (3 уровень)

- Первый элемент маркированного списка (4 уровень)
- Второй элемент маркированного списка (4 уровень)

Статья 2 (2 уровень)

Содержимое статьи 2 (3 уровень)

Глава 2 (1 уровень)

1. Первый элемент нумерованного списка (2 уровень)
2. Второй элемент нумерованного списка (2 уровень)

Набор данных для классификации параграфов

Был размечен набор данных для обучения

- 22 документа, скачанных с сайта государственных закупок[5];
- 1405 параграфов.

[5] https://zakupki.gov.ru

Набор данных классификации пар параграфов

Данные для сравнения пар параграфов

Для каждого параграфа рассматривается окно (± 4 параграфа относительно текущего параграфа). Каждой паре нужно сопоставить один из трех классов: больше, меньше или равно.

Набор данных содержал документы, использовавшиеся при классификации параграфов; размечено 4438 пар параграфов.

Оценка качества для классификаторов

Задача	Количество признаков	Алгоритм	Достоверность (accuracy)	F-мера
Метод классификации параграфов[6] (Kim)	8	RandomForest	0.91	0.86
Классификация параграфов	234	XGBoost	**0.96**	**0.94**
Сравнение параграфов	28	XGBoost	**0.98**	**0.97**

[6] A Machine-Learning Based Approach for Extracting Logical Structure of a Styled Document

Построение дерева (оценка качества)

- Реализовано построение деревьев документов типа «Техническое задание».
- Размечено 5 документов для тестирования: для каждого документа построено дерево. В размеченных данных сравнивалось 714 пар строк.
- На тестовой выборке усреднённое расстояние Робинсона-Фулдса[7] между построенными и размеченными деревьями оказалось равным 0.073.
  - Расстояние Робинсона-Фулдса означает количество специальных операций изменения (добавлений или удалений ребер с узлами), нужных для того, чтобы одно дерево превратить в другое (для нормировки делится на максимальное число операций).

[7] A generalized Robinson-Foulds distance for labeled trees

Практическая часть

При реализации использовался язык программирования *Python3*.
Реализованы следующие методы:

- Обработчик docx документов (1217 строк).
- Извлечение признаков для классификаторов и их обучение (1123 строк).
- Построение дерева и его визуализация, разметка деревьев для тестирования и сравнение деревьев (329 строк).
- Адаптация существующего метода для сравнения (297 строк).

Заключение

- Исследована логическая структура документов, предложена структура в виде дерева произвольной глубины с типизированными узлами.
- Разработан и реализован метод по извлечению предложенной структуры из документов, представленных в формате docx, на примере технических заданий.
- Составлен и размечен набор технических заданий.
- Проведена экспериментальная оценка реализованных методов.
- Реализовано извлечение текста и метаданных из документов в формате docx. Код обработчика docx документов встроен в открытый проект dedoc[8].

[8] https://github.com/ispras/dedoc

Создание набора данных

Процесс создания обучающего набора документов:

1. Создание изображений для разметки из docx файла. На каждом из изображений один из параграфов обведён в рамку.
2. Разметка каждого изображения аннотаторами с помощью системы разметки[9].
3. Получение меток для параграфов, соответствующих размеченным изображениям.

[9] https://github.com/dronperminov/ImageClassifier

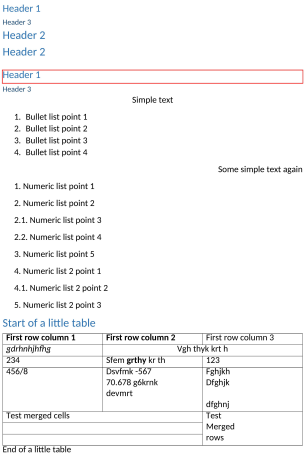
Извлечение метаданных

На языке python написан обработчик docx документов. Код обработчика встроен в проект dedoc. Пример извлечения метаданных (жирность, размер шрифта, выравнивание и т. д.) для параграфа документа.


```
"text": "Header 1",
"annotations": [
  {
    "start": 0,
    "end": 8,
    "name": "indentation",
    "value": "0"
  },
  {
    "start": 0,
    "end": 8,
    "name": "alignment",
    "value": "left"
  },
```

Создание изображений из docx документов

Страницы docx документа преобразуются в изображения с параграфами, обведенными в рамку.



Header 1

Header 3

Header 2

Header 2

Header 1

Header 3

Simple text

1. Bullet list point 1
2. Bullet list point 2
3. Bullet list point 3
4. Bullet list point 4

Some simple text again

1. Numeric list point 1
2. Numeric list point 2
2.1. Numeric list point 3
2.2. Numeric list point 4
3. Numeric list point 5
4. Numeric list 2 point 1
4.1. Numeric list 2 point 2
5. Numeric list 2 point 3

Start of a little table

First row column 1	**First row column 2**	First row column 3
gdrhnhjhfhg	Vgh thyk krt h	
234	Sfem **grthy** kr th	123
456/8	Dsvfmk -567 70.678 g6krnk devmrt	Fghjkh Dfghjk dfghnj
Test merged cells		Test Merged rows

End of a little table

Признаки параграфов

- Признаки, полученные с помощью регулярных выражений для начала и конца строки (в том числе для списков).
- Номер параграфа, длина текста параграфа, число слов в тексте параграфа.
- Индикаторы того, если ли в параграфе жирный, курсивный, подчёркнутый текст, текст из заглавных букв.
- Выравнивание, отступ от левого края, размер шрифта текста.
- Признаки для названий стилей («heading», «title», «list item», «contents»).
- Признаки трех предыдущих и трех следующих параграфов.

Признаки пар параграфов

- Разница значений размеров шрифта и отступов от левого края.
- Разница значений индикаторов жирности, курсива, подчеркивания.
- Разница выравниваний и типов параграфов (каждому типу поставлено в соответствие число по приоритетам типов).
- Разница индикаторов соответствия параграфов стилям.
- Индикатор начала списка после строки, заканчивающейся двоеточием.
- Разница индикаторов соответствия параграфов шаблонам конкретных типов списков.