DocxReader

Богатенкова Анастасия

3 февраля 2021 г.

ИСП РАН

Файлы, содержащие необходимую информацию

ИСП РАН

- **word/document.xml** содержит все параграфы и таблицы документа с их свойствами, ссылки на колонтитулы и заметки;
- **word/numberings.xml** содержит информацию обо всех типах списков, используемых в документе и их настройках;
- **word/styles.xml** содержит стили применимые к документу;
- **word/header1.xml, word/footer1.xml, word/endnote.xml, word/footnote.xml** содержат колонтитулы, заметки.

Основные составляющие DocxReader

ИСП РАН

- **docx_reader** – непосредственно ридер с реализованными методами can_read() и read();
- **numbering_extractor** – модуль для извлечения текстового содержимого нумерации списков;
- **styles_extractor** – модуль для анализа стилей и применения стилей к параграфам;
- **properties_extractor** – модуль для выделения из xml-параграфов различных свойств;
- **data_structures** – набор вспомогательных структур, таких как параграф и текстовый элемент (run по документации), таблица и свойства параграфа.

Содержимое docx_reader

ИСП РАН

- **get_paragraph_list, get_document_bs_tree** – используются в DocxImagesCreator;
- **_process_tables** – формирует список таблиц с помощью DocxTable;
- **_process_lines** – формирует список из Paragraph, из которого с помощью **_get_lines_with_meta** получается список из LineWithMeta.

Содержимое data_structures

ИСП РАН

- **base_props** – базовый класс для Run и Paragraph, нужен для удобства инициализации свойств;
- **Paragraph** – соответствует параграфу docx, нужен для парсинга xml-содержимого параграфов;
- **Run** – соответствует так называемому run в docx, в основном нужен для получения текста run;
- **DocxTable** – нужен для парсинга xml-содержимого таблиц;
- **ParagraphInfo** – анализирует содержимое Paragraph, формирует его текст и аннотации.

base_props

Хранит в себе общие для Paragraph и Run свойства: отступ, выравнивание, размер шрифта, жирность, курсив, подчеркивание. Можно инициализировать поля по умолчанию либо на основе уже существующего Paragraph или Run.

У параграфов и у их частей могут присутствовать и отсутствовать некоторые общие свойства, которые применяются по иерархии, поэтому показалось удобным сделать общий базовый класс.

Помимо свойств base_props хранит список run, информацию о стиле и уровне вложенности списка (если есть).

- Метод parse() формирует свойства параграфа в соответствии с иерархией стилей.
- Метод _get_numbering_formatting() формирует Run с нумерацией, если в параграфе присутствует нумерация.
- Метод _make_run_list() формирует список Run со всеми их свойствами.

Помимо свойств base_props хранит текст части параграфа.

- Метод get_text() формирует текст на основе xml.
- Метод __eq__() сравнивает два элемента без учета текста, отступа и выравнивания (сравнение используется при формировании списка Run в параграфе).

DocxTable

В отличие от Table проекта класс DocxTable нужен для разбора xml-содержимого тега tbl и для подсчета uid таблицы на основе этого содержимого.

- uid – возвращает uid таблицы;
- get_cells() – возвращает список списков с текстом ячеек таблицы. Текст каждой ячейки – склеенный переносом строки список параграфов.

ParagraphInfo

Преобразует свойства параграфа в более близкий для проекта вид – аннотации. Конструктор класса на основе Paragraph формирует текст, uid и свойства параграфа.

Свойства параграфа - список из списков [start, end, props], где start – начало свойства, end – конец (не включается), props – словарь свойств (все свойства для данного участка текста). Одинаковые аннотации пока не объединяются.

Инициализирует self.last_properties – вспомогательный словарь для объединения аннотаций.

Методы ParagraphInfo

- _get_hierarchy_level() – специальный метод для определения HierarchyLevel параграфа;
- get_info() – формирует для параграфа словарь с его текстом, уровнем, типом, uid и списком аннотаций; объединяет аннотации;
- make_annotation() – либо расширяет одну из аннотаций в self.last_properties, либо возвращает из self.last_properties одну из аннотаций, заменяя ее новой;
- get_text – возвращает текст параграфа.

Содержимое properties_extractor

Набор функций, которые на основе Run или Paragraph и xml-содержимого обновляют свойства параграфа или run.

- change_paragraph_properties() – изменяет отступ, размер шрифта и выравнивание параграфа (остальные свойства параграфа не учитываются в итоге);
- change_run_properties() – изменяет жирность, курсив, подчеркивание, размер шрифта;
- change_indent() – изменяет отступ;
- change_size() – изменяет размер шрифта;
- change_js() – изменяет выравнивание.

StylesExtractor – методы

- в конструкторе сохраняются xml-деревья с настройками стилей по умолчанию;
- _find_style() – находит xml-дерево стиля по его style_id;
- parse() – изменяет свойства Paragraph или Run в соответствии с тем стилем, style_id которого передан (далее описано подробнее);
- _apply_styles() – применяет поочередно каждый стиль из переданного списка стилей к Paragraph или Run;
- _get_styles_hierarchy() – на основе style_id ищет базовые стили и формирует список всех стилей иерархии для заданного;
- _get_style_level() – если имя стиля начинается с heading, возвращает его номер, меньший на единицу.

StylesExtractor – метод parse()

- сначала применяются стили по умолчанию;
- если style_id равен None или не найден в дереве, работа метода завершается;
- применяем _get_style_level() если можем;
- получаем и применяем иерархию стилей;
- создаем Run с нумерацией и добавляем к списку runs параграфа (если в стиле есть указание на то, что параграф пронумерован, а в самом параграфе этого нет).

Содержимое numbering_extractor

- numFmtList – словарь с типами списков и их первыми элементами;
- get_next_item() – функция получения элемента с конкретным номером для заданного типа списка;
- getSuffix - словарь с типами пробелов между нумерацией и остальным текстом параграфа;
- класс AbstractNum соотвествует abstractNum в docx, описывает свойства типа списка;
- класс Num соотвествует num в docx, наследует все свойства одного из AbstractNum и перегружает некоторые из них, для каждого списка документа есть свой num;
- класс NumberingExtractor строит соответствие между Num и AbstractNum и вычисляет текст и свойства нумерации списков.

AbstractNum

Описывается тип многоуровневого списка, у каждого уровня этого типа списка свойства различаются.

- в конструкторе инициализируются свойства, общие для всех уровней (ссылка на свойства другого AbstractNum и сброс нумерации, если список прервался) и словарь свойств для каждого уровня;
- parse() – парсит список уровней, заполняя свойства каждого уровня (тип списка, тип пробела, начальное значение, стиль, сброс нумерации).

Взаимно однозначно соответствует многоуровневому списку документа.

- в конструкторе Num проходит по иерархии AbstractNum (если в одном есть ссылка на другой), сохраняет нужное дерево и парсит его, получая информацию о свойствах уровней. Если необходимо, перегружает некоторые свойства.
- get_level_info() возвращает словарь со свойствами для конкретного уровня списка.

NumberingExtractor

ИСП РАН

Хранит текущее состояние парсинга списков и список всех Num документа.

- self.numerations – словарь текущих номеров для списков с конкретным уровнем и abstractNumId;
- self.prev_num_id и self.prev_abstract_num_id предыдущие списки для текущего списка;
- self.prev_ilvl и self.prev_numId – предыдущие уровень и numId для списка с конкретным abstractNumId;
- self.shifts используется, если в xml есть неправильные списки;
- self.levels_count – счетчик уровней для текущего списка;
- self.num_list – список всех Num документа.

NumberingExtractor._get_list_text()

Самая сложная функция. Даны уровень списка и его numId.

- сначала проверяем, корректна ли нумерация (может получится так, что информации о списке нет), исправляем в случае ошибки;
- если список сменился (другой abstractNumId), удаляем информацию о предыдущем уровне;
- обновляем или увеличиваем счетчик нумерации элементов списка в соответствии с текущими данными;
- обновляем информацию о предыдущих элементах списка;
- получаем текст элемента по уровню и numId с помощью _get_next_number() (может возникать ошибка, если нет информации о текущем номере списка, тогда текущий номер устанавливается $= 1$).

NumberingExtractor._get_next_number()

ИСП РАН

Метод является вспомогательным:

- По информации из NumberingExtractor, уровню списка и его numId вычисляет тип нумерации и номер элемента списка.
- С помощью функции get_next_item() вычисляет текстовое представление нумерации элемента списка и возвращает его.

NumberingExtractor.parse()

Изменяет свойства переданных параграфа и run.

- если параграф пронумерован, по numId и уровню (если есть) находятся стиль, свойства параграфа и свойства run (если есть);
- находится текст списка с помощью _get_list_text();
- исходные свойства параграфа и свойства run изменяются в соответствии с найденными;
- к run добавляется текст, к параграфу добавляется уровень списка.