Санкт-Петербургский государственный университет

Кафедра информационно-аналитических систем

Группа 20Б.08-мм

Салтыков Павел Константинович

Метрики сложности предложений и разработка микросервиса для их расчёта

Отчёт по учебной практике

Научный руководитель:
ассистент кафедры ИАС Чернышев Г. А.

Санкт-Петербург
2022

Оглавление

Введение

Завершающим этапом обучения в вузе является выпускная квалификационная работа (ВКР), при написании которой, студентам необходимо соблюдать ряд основных требований, касающихся оформления и содержания.

На данный момент ведётся разработка инструмента[1] для автоматизированной проверки текстов ВКР. Сейчас он позволяет проверять работы на наличие распространённых ошибок оформления.

Удобочитаемость предложений является одним из требований к тексту выпускной работы. В процессе чтения не должно возникать трудностей с восприятием текста.

Таким образом, в рамках данной работы стоит задача изучения метрик сложности текста, и в частности предложений. А также необходимо разработать отдельный микросервис для расчёта этих метрик, который в дальнейшем можно было бы интегрировать с валидатором текстов ВКР для отображения трудночитаемых предложений.

[1] https://github.com/darderion/map (дата обращения: 03.11.2022)

1. Постановка задачи

Целью данной работы является обзор метрик сложности предложений и создание микросервиса для расчёта этих метрик. Для выполнения этой цели были поставлены следующие задачи:

1. выполнить обзор метрик сложности текста;
2. разработать микросервис для подсчёта метрик;
3. выполнить апробацию.

2. Обзор

2.1. Индексы удобочитаемости текста

Индексы удобочитаемости — это формулы, дающие численный показатель сложности восприятия текста. Наиболее популярными являются Flesch Reading Ease и Flesch–Kincaid Grade Level [6]. Эти индексы используются в Microsoft Word[2] в качестве показателей удобочитаемости документа. Оба индекса вычисляются на основе двух параметров: средней длины предложения в словах и средней длины слова в слогах. Кроме того, эти формулы для разных языков имеют разные коэффициенты, поскольку средняя длина предложения и слова специфичны для каждого языка.

2.2. Метрики сложности текста

Для подсчёта метрик было принято решение использовать библиотеку textcomplexity [10] на языке Python. В ней реализованы различные метрики, оценивающие лингвистическую и стилистическую сложность текстов. Их можно разделить на следующие группы:

1. метрики, использующие размер выборки и размер словарного запаса (Surface-based measures);
2. метрики, основанные на характеристике предложений (Sentence-based measures);
3. метрики на основе частей речи (POS-based measures);
4. метрики, основанные на синтаксических связях в предложениях (Dependency-based measures);
5. метрики, основанные на грамматике составляющих (Constituency-based measures).

[2] Microsoft Word document's readability (дата обращения: 03.11.2022)

2.2.1. Type-Token Ratio

Наиболее популярной метрикой является Type-Token Ratio (TTR) — коэффициент лексического разнообразия [5], который оценивает то, насколько разнообразны используемые в тексте слова. Он определяется как отношение количества уникальных токенов (типов) к общему числу токенов[3].

Как и большинство метрик из первой группы, TTR зависит от длины анализируемого текста: чем он длиннее, тем меньше вероятность появления новых слов. Поэтому для больших текстов применяется метод bootstrap [2]. Его идея заключается в том, чтобы разбить текст на непересекающиеся сегменты одинаковой длины в токенах и рассчитать среднее значение метрики.

2.2.2. Длина предложения

Длина предложения является важным показателем его сложности. Чем длиннее предложение, тем сложнее оно воспринимается при чтении. Данная библиотека позволяет вычислить длину представленного массивом токенов предложения в:

- токенах,
- словах (игнорируя знаки препинания),
- символах (определяется как сумма длин всех токенов и количества промежутков между ними).

Для текста, состоящего из нескольких предложений, рассчитываются средние значения этих метрик.

2.2.3. Лексическая плотность

Коэффициент лексической плотности (Lexical density) [7] показывает долю содержательных (имеющих лексическое значение) слов в рассматриваемом тексте. Он определяется как отношение количества са-

[3]Отдельный элемент предложения (слово, знак препинания, число)

мостоятельных частей речи к общему числу слов. Таким образом, чем меньше служебной лексики используется в тексте, тем более высокой лексической плотностью он обладает.

2.2.4. Расстояние между связанными словами в предложении

Синтаксическая структура предложения может быть представлена в виде дерева зависимостей [11]. Связи между членами предложения изображаются дугами в этом дереве, а его корнем является сказуемое предложения.

Метрики четвёртой группы характеризуют сложность структуры предложения. Они основаны на анализе дерева зависимостей, которое представляется в виде ориентированного графа. Поскольку для вычисления расстояния между связанными словами необходима информация о порядке слов, каждая вершина графа содержит номер соответствующего слова в предложении. Тогда количество слов, находящихся между связанными членами предложения, будет определяться как модуль разности значений в смежных вершинах графа.

Для оценки сложности структуры предложения будет полезно среднее и максимальное значение такого расстояния.

2.2.5. Высота дерева составляющих предложения

Помимо дерева зависимостей, структура предложения также может быть представлена деревом составляющих [11] (рис. 1). Оно строится

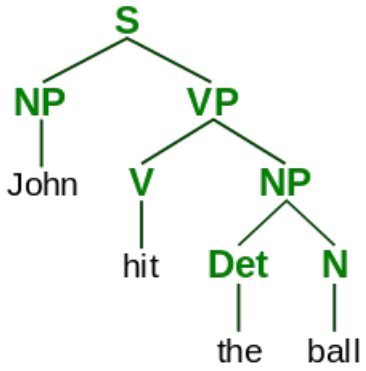

Рис. 1: Дерево составляющих, пример взят из работы [8]

на основе контекстно-свободной грамматики, которая описывает фразовую структуру предложения. Дерево составляющих содержит слова предложения в качестве листьев, а его поддеревья соответствуют составляющим этого предложения.

Таким образом, сложность структуры предложения будет определяться строением и характеристиками данного дерева, например, его высотой.

3. Реализация микросервиса

3.1. Технологии и инструменты

3.1.1. Формат CoNLL-U

Подсчёт большей части метрик в библиотеке textcomplexity основан на использовании представления текста в формате CoNLL-U. В связи с этим, перед автором данной работы также стояла задача реализовать преобразование исходного текста в такой формат.

Формат CoNLL-U [1] — это формат морфологической разметки текста, используемый в обработке естественных языков. В данном формате каждый токен представлен одной строкой, которая состоит из 10 разделенных символами табуляции полей:

1. ID: индекс токена в предложении (натуральное число);
2. FORM: форма слова или знак препинания;
3. LEMMA: лемма — словарная форма слова;
4. UPOS: универсальный тег части речи[4];
5. XPOS: тег части речи для конкретного языка;
6. FEATS: список морфологических признаков;
7. HEAD: индекс главного слова в синтаксической связи;
8. DEPREL: тип синтаксической связи[5] с HEAD;
9. DEPS: дополнительные синтаксические связи;
10. MISC: примечание.

[4]Универсальные теги частей речи: https://universaldependencies.org/u/pos/index.html (дата обращения: 03.11.2022)

[5]Universal Dependency Relations: https://universaldependencies.org/u/dep/index.html (дата обращения: 03.11.2022)

3.1.2. Обработка естественного языка

Для обработки естественного языка, а также преобразования текста в CoNLL-U формат была использована библиотека spaCy [9]. В данной библиотеке обработка текста происходит по принципу конвейера (pipeline, рис. 2).

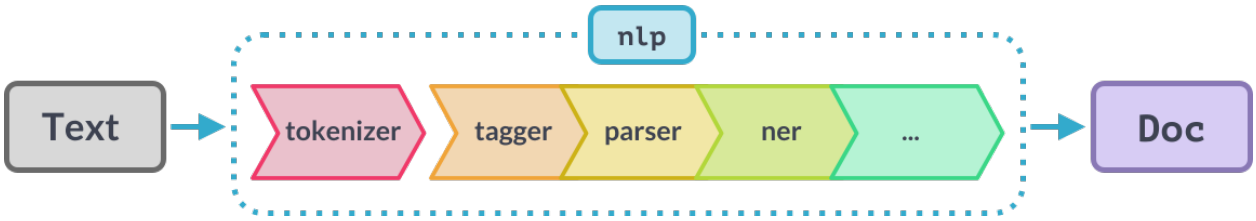

Рис. 2: spaCy pipeline, рисунок взят из работы [4]

Сначала tokenizer разделяет текст на токены и создаёт объект Doc, содержащий их. Далее, по порядку вызываются компоненты конвейера, модифицирующие Doc. Также имеется возможность добавлять пользовательские компоненты конвейера.

Таким образом, для обработки текста и преобразования его в формат CoNLL-U применялся обученный pipeline для русского языка с добавленным конвейерным компонентом conll_formatter[6].

3.1.3. FastAPI

FastAPI [3] — современный, высокопроизводительный веб-фреймворк на python. Он автоматически генерирует документацию согласно спецификации OpenAPI, а также имеет встроенную валидацию данных и сериализацию. Поэтому он был выбран для создания API микросервиса.

3.2. Получение результатов

Для получения значений метрик необходимо отправить POST-запрос, указав в параметрах строки необходимые метрики и сформировав тело запроса в формате JSON:

[6] https://pypi.org/project/spacy-conll/ (дата обращения: 03.11.2022)

Листинг 1: Пример тела запроса

```
[
    {
        "id": 0,
        "sentence": "Рассчитать метрики сложности предложения.",
        "language": "ru"
    }
]
```

Ответ будет также в формате JSON:

Листинг 2: Пример ответа

```
[
  {
    "id": 0,
    "measures": [
      {
        "name": "sentence_length_tokens",
        "value": 5
      }
    ],
    "error": null
  }
]
```

4. Апробация

Для взаимодействия с микросервисом было использовано приложение Postman[7]. Оно предоставляет возможность составлять и отправлять HTTP-запросы. Также этот инструмент позволяет писать скрипты, которые будут выполняться перед отправкой запроса (например, для формирования тела запроса) или после получения ответа.

Были сделаны запросы на вычисление 33 метрик для разного количества предложений со средней длиной 20 токенов. Время обработки составило:

- 1,92 с $\pm$ 0,06 с для 100 предложений,
- 6,89 с $\pm$ 0,18 с для 300 предложений,
- 11,24 с $\pm$ 0,28 с для 500 предложений.

[7]Postman: https://www.postman.com/ (дата обращения: 03.11.2022)

Заключение

В ходе данной работы были выполнены следующие задачи:

1. выполнен обзор метрик сложности текста;
2. разработан микросервис для подсчёта метрик;
3. выполнена апробация.

Исходный код разработанного микросервиса доступен в GitHub репозитории[8]. В будущем планируется интеграция валидатора текстов выпускных квалификационных работ с микросервисом, чтобы на основании разных метрик отображать трудночитаемые предложения.

[8]Исходный код микросервиса: https://github.com/PavelSaltykov/text-complexity-service (дата обращения: 03.11.2022)

Список литературы

[1] CoNLL-U Format. — URL: https://universaldependencies.org/format.html (online; accessed: 2022-11-03).

[2] Evert Stefan, Wankerl Sebastian, Nöth Elmar. Reliable measures of syntactic and lexical complexity: The case of Iris Murdoch. — 2017. — URL: https://birmingham.ac.uk/documents/college-artslaw/corpus/conference-archives/2017/general/paper364.pdf (online; accessed: 2022-11-03).

[3] FastAPI. — URL: https://fastapi.tiangolo.com/ (online; accessed: 2022-11-03).

[4] Language Processing Pipelines. — URL: https://spacy.io/usage/processing-pipelines (online; accessed: 2022-11-03).

[5] NLP Basics: Measuring The Linguistic Complexity of Text. — 2019. — URL: https://towardsdatascience.com/linguistic-complexity-measures-for-text-nlp-e4bf664bd660 (online; accessed: 2022-11-03).

[6] Wikipedia contributors. Flesch–Kincaid readability tests — Wikipedia, The Free Encyclopedia. — URL: https://en.wikipedia.org/wiki/Flesch-Kincaid_readability_tests (online; accessed: 2022-11-03).

[7] Wikipedia contributors. Lexical density — Wikipedia, The Free Encyclopedia. — URL: https://en.wikipedia.org/wiki/Lexical_density (online; accessed: 2022-11-03).

[8] Wikipedia contributors. Parse tree — Wikipedia, The Free Encyclopedia. — URL: https://en.wikipedia.org/wiki/Parse_tree (online; accessed: 2022-11-03).

[9] spaCy — Industrial-Strength Natural Language Processing. — URL: https://spacy.io/ (online; accessed: 2022-11-03).

[10] textcomplexity. — URL: https://github.com/tsproisl/textcomplexity (online; accessed: 2022-11-03).

[11] Автоматическая обработка текстов на естественном языке и анализ данных / Большакова Е.И., Воронцов К.В., Ефремова Н.Э. et al. — М.: НИУ ВШЭ, 2017. — P. 19. — URL: https://publications.hse.ru/books/208965165 (online; accessed: 2022-11-03).