Санкт-Петербургский государственный университет

Кафедра информационно-аналитических систем

Группа 21.Б08-мм

Платонов Илья Юрьевич

Расширение базы правил валидатора текстов выпускных квалификационных работ

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
ассистент Чернышев Г.А.

Санкт-Петербург
2023

Оглавление

Введение

Все студенты обучающиеся в высших учебных заведениях рано или поздно встречаются с задачей написания различных отчетов (курсовые работы, отчеты по различным практикам, и пр.). Однако при написании весомая часть времени уходит на различные проверки текстов этих работ. Этим занимаются как преподаватели при оценке так и обучающимся перед показом. Большая часть времени этой проверки уходит на утомительные и однообразные действия. Поэтому логично было бы автоматизировать этот процесс, что не только ускорит проверку, но и сделает ее более точной.

Для решения этой проблемы и было написано на языке Kotlin веб-приложение Mundane Assignment Police для проверки ВКР и других работ на нормы оформления. Оно предназначено для нахождения различного рода ошибок в PDF-файлах. И для расширения функционала этого приложения в рамках работы предлагается создать новые правила, чтобы увеличить количество ошибок, которые может зафиксировать приложение.

1. Постановка задачи

Целью работы является расширение функционала веб-приложения для проверки PDF-файлов с текстами выпускных квалификационных работ.

Для её выполнения были поставлены следующие задачи:

1. Разработать новые правила, расширив функционал приложения
   (a) Правило проверяющее ссылки на единообразие
   (b) Правило проверяющее символ на выход за поля
   (c) Правило проверяющее оформление ссылок на несколько статей
   (d) Правило проверяющее соседние слова на повторение
   (e) Правило находящее большую букву после скобки
2. Провести апробацию на базе данных курсовых работ

2. Обзор

В данном разделе рассматриваются работы и методы, решающие проблему проверки и анализа текста на соответствие выдвинутым стандартам.

2.1. Автоматизированная оценка текстов

Необходимость автоматизированной оценки текстов встречается во многих отраслях и соответственно появляется много работ предназначенных для автоматической оценкой текста. Одной из отраслей является, например, юриспруденция. Для рассмотрения новых законов была создана система Formalex [3]. Система оценивает текст законов, выделяя логические структуры, и проверяя не противоречат ли они уже существующим или самим себе.

Также существует необходимость в оценки текстов на их читаемость, логичность и.т.д.

Для этих целей был создан фреймворк EXPATS [2]. Данный фреймворк позволяет пользователям разрабатывать свои модели автоматической оценки текста, предлагая набор из различных способов для оценки.

2.2. Автоматизированная проверка оформления текстов

Одной из первых систем созданных для автоматизированной оценки научных текстов была система ЛИНАР [4]. Она осуществляла контроль орфографии, анализ лексического состава, стилистический контроль текста, контроль структуризации и семантический контроль. Так же в системе ЛИНАР были варианты исправления ошибок.

Однако эта система не содержала возможности проверки учебных текстов, и для решения этой проблемы создана система КОНУТ [1]. Эта система представляла из себя текстовый редактор с возможностью

запустить оценку текста. КОНУТ проверял наличие обязательных разделов, правильный порядок расположения разделов, корректная нумерация разделов, правильное оформление титульного листа и библиографии и др. Так же был встроен справочник с правилами оформления учебных работ.

Так же существовала система ГАММА, которая являлась подсистемой для MS Word. С точки зрения анализа текста она имела тот же функционал что и система КОНУТ, контроль содержания, проверка порядка секций и др. Так же она имела удобную возможность перемещения между всеми таблицами и рисунками. В данной системе было реализовано наибольшее количество правил.

2.3. Различия

Найти в открытом доступе описанные выше системы для оценки научно-учебных текстов не удалось. Потому провести проверку и сравнить их в действии с разрабатываемой программой возможности не было. Различие есть в формате обрабатываемых файлов, Mundane Assignment Police обрабатывает PDF файлы, в то время как например ГАММА могла обрабатывать только Word файлы. Также в найденных аналогах общее количество обрабатываемых правил меньше, например из правил описанных в постановке задачи реализовано было лишь правило проверяющее выход символа за поля.

3. Реализованные правила

В данном разделе описывается процесс реализации правил заявленных в постановке задачи.

3.1. Оформление ссылок на разные статьи

Нередко люди пишущие ВКР или другие работы используя несколько ссылок оформляют это так:

В рассматриваемых играх [3] [6] присутствуют элементы RPG.

Однако корректно было бы оформить это так:

В рассматриваемых играх [3, 6] присутствуют элементы RPG.

Для реализации этого правила было использовано правило проверяющие соседние символы. В нем задаются два символа которые не могут стоять рядом и затем все соседние символы проверяются. Первым символом был выбран символ '] ', после чего проверяются только его правые соседи на то, равны они ' ['. При этом символы запятой и пробела отмечаются как игнорируемые для проверки.

3.2. Выход символа за поля

Часто при оформлении ВКР и других работ по невнимательности, или ввиду иной ошибки может получиться так, что один или несколько символов выйдут за поля. Для фиксации подобных случаев будут использоваться координаты символов.

Однако проверять координату каждого символа будет излишне, потому проверку начинать следует с конца строки, и при выявлении выхода за поля будет выдаваться ошибка.

Для реализации будет использоваться класс созданный для работы со словами при помощи задающегося предиката.

3.3. Проверка ссылок на единообразие

Нередко можно встретить ситуацию, когда некоторые ссылки указаны в виде 'www.google.com',а некоторые в виде 'https://google.com'. Подобное оформление является неправильным, необходимо чтобы все ссылки были одинаковыми. Для того что бы подобное не происходило берется список всех URL и он фильтруется отбрасывая по очереди ссылки разных типов (сначала убираются URL вида "https://www", затем "www" и.т.д). И если после какой-либо из фильтраций количество ссылок меняется то все ссылки не отфильтровавшиеся выдаются как ошибочные. Если же все URL имеют один тип то список становится пустым.

3.4. Правило находящее одинаковые слова стоящие подряд

Частой ошибкой при написании ВКР является случайное написание двух одинаковых слов подряд,. И для выявления это ошибки был создан класс для реализации правил работающих со словами. Его отличие от уже существующего в том, что в новый можно передавать предикат. Тогда как в старом мы могли лишь проверять текст по ранее заданной функции.

Для нахождения в тексте двух одинаковых слов подряд текст разбивается на слова, игнорируя пунктуацию. Затем идет проверка, где если соседи совпадают, то на этом месте отмечается ошибка.

3.5. Правило находящее большую букву после скобки

Часто при создании ВКР студенты забывают, что после скобки писать слово с большой буквы нельзя.Однако если слово после скобки является аббревиатурой, то это можно делать. Поэтому просто выделять все случаи когда после символа " (" пишется заглавная буква нельзя.

Неправильным срабатыванием будет считаться ситуация, когда аббревиатура принимается за обычное слова. Для того чтобы не допустить этого используется класс, работающий со словами, при помощи задаваемого предиката.

При проверке выделяется первое слово находящееся после скобки. После этого слово проверяется на то является оно аббревиатурой, или нет. Слово считается аббревиатурой, если в нем больше одной большой буквы. И далее если слово окажется не аббревиатурой, то это место распознается как ошибочное.

4. Апробация

Тестирование проводилось на базе данных курсовых, магистерских и бакалаврских работ Математико-механического факультета СПбГУ[1]. Была обработана 21 работа, из которых было 8 бакалаврских и 13 магистерских работ (таблица 1). Таким образом было найдено 368 ошибок, и в среднем в каждой работе было по 19 ошибок.

Таблица 1: Статистика ошибок

Правило	Всего	Среднее	Среднее в бакалаврских	Среднее в магистерских
Единообразие ссылок	274	14.2	20.66	14.3
Выход за поля	4	0.2	0	0.4
Оформление ссылок на разные статьи	3	0.15	0.5	0
Повторяющиеся слова	0	0	0	0
Большая буква после скобки	45	2.36	3.66	1
Все обнаруженные ошибки	368	19	6.3	4.27

Как видно из таблицы самой частой ошибкой является отсутствие единообразия ссылок, большое количество обусловлено тем, что при несовпадение типов ссылок все ссылки одного из типов выделяются как ошибки. Правило же которое должно находить использование большой буквы после скобки, в большей части случаев срабатывает на записях вида “(Рис.1)”.

Малое же количество выходов за поля связано с тем, что правило не может работать с рисунками и таблицами, которые чаще всего и выходят за поля.

[1]https://github.com/Darderion/map-dataset — репозиторий с базой выпускных квалификационных работ на Github.

Также для дополнительной проверки верности срабатывания правила проверяющего оформление ссылки на единообразие была проведена ручная проверка пяти работ, в ходе которой не было выявлено лишних срабатываний.

Заключение

В ходе работы были выполнены следующие задачи:

1. Разработаны новые правила, расширяющие функционал приложения:
   (a) разработано и реализовано правило проверяющее правильность оформления ссылок на разные источники;
   (b) разработано и реализовано правило проверяющее выход символа за поля;
   (c) разработано и реализовано правило проверяющее ссылки на единообразие;
   (d) разработано и реализовано правило находящее повторяющиеся подряд слова;
   (e) разработано и реализовано правило находящее большую букву после скобки;
2. Проведена апробация на базе данных курсовых работ.

Открытый код доступен по ссылке на репозиторий[2] Github.

[2]https://github.com/gwyndol1ne/map — репозиторий проекта на Github.

Список литературы

[1] Bolshakova E. Computer Assistance in Writing Technical and Scientific Texts // Proceedings of 2nd International Symposium “Las Humanidades en la Educación Técnica ante el Siglo XXI”, México. — 2000. — P. 59–63.

[2] Manabe Hitoshi, Hagiwara Masato. EXPATS: A Toolkit for Explainable Automated Text Scoring // CoRR. — 2021. — Vol. abs/2104.03364. — URL: https://arxiv.org/abs/2104.03364.

[3] Performance Improvement on Legal Model Checking / Carlos Faciano, Sergio Mera, Fernando Schapachnik et al. // Proceedings of the 16th Edition of the International Conference on Articial Intelligence and Law. — ICAIL ’17. — New York, NY, USA : Association for Computing Machinery, 2017. — P. 59–68. — URL: https://doi.org/10.1145/3086512.3086518.

[4] Баева НВ, Большакова ЕИ. Проблемы автоматизации контроля учебно-научных текстов // М.: МГУ им. МВ Ломоносова, факультет вычислительной математики и кибернетики. — 2012. — URL: https://sworld.education/konfer27/381.pdf.