Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Отчет по учебному курсу
«Верификация программного обеспечения»

Тестирование системы для генерации тестовых заданий

Выполнил:
студент 4 курса группы 22407
Н. Д. Семенов

Петрозаводск — 2022

Содержание

1 Объект тестирования

В качестве объекта тестирования выступает модуль для генерации тестовых заданий по документам (источникам). Модуль позволяет генерировать различные вопросы, получать на них правильный ответ, генерировать неправильные ответы (дистракторы). Данные тестовых заданий отправляются на бэкенд веб-системы, где дальше доходят до конечного пользователя (тестируемого). Перед тем, как попасть к тестируемому, вопросы просматриваются экспертом, который может изменить текст вопроса, текст ответа, текст неправильных ответов, либо не принять сгенерированное задание полностью. На основе решения эксперта модуль принимает обратную связь, которая используется для дообучения ИИ-моделей системы. Модуль написан на языке Python.

Функциональность объекта тестирования

1. Генерация тестовых заданий для заданных документов
    1.1. Предобработка текста (парсинг)
        1.1.1. Поддержка обработки .html файлов
        1.1.2. Поддержка обработки .docx файлов
    1.2. Классификация предложения
        1.2.1. Классификация на основе системы правил
        1.2.2. Классификация на основе ИИ
    1.3. Генерация вопроса
        1.3.1. Классификация на основе системы правил
        1.3.2. Классификация на основе ИИ
    1.4. Генерация дистракторов
        1.4.1. Классификация на основе системы правил
        1.4.2. Классификация на основе ИИ
    1.5. Валидация вопросов
        1.5.1. Морфологические проверки
        1.5.2. Орфографические проверки
        1.5.3. Пунктуационные проверки
2. Дообучение ИИ-моделей на основе обратной связи

2.1. Дообучение моделей по классификации предложений
2.2. Дообучение моделей по генерации вопросов
2.3. Дообучение моделей по генерации дистракторов
2.4. Дообучение моделей по валидации вопроса

Архитектура системы

Схема архитектуры системы находится в приложении 1.

Обозначения архитектуры системы:

1. Веб-система (желтый);
2. Файловая система (ФС) (оранжевый): файловое хранилище для данных ИИ-моделей;
3. Django-сервер (DC) (красный): промежуточное звено между веб-системой и QG-модулем
4. QG-модуль (серый): модуль генерации тестовых заданий, написанный на языке Python;
5. Pipeline (или API) (зеленый): программный интерфейс (API) QG-модуля;
6. Библиотеки (синий): основные и независимые элементы QG-модуля;
7. Общесистемные компоненты (фиолетовый): компоненты, используемые как внутри библиотек QG-модуля, так и Pipeline (API) QG-модуля;

2 Стратегия тестирования

Тестирование проводится с помощью библиотеки pytest и вручную. При помощи pytest реализовано модульное, интеграционное и частично аттестационное тестирование. Примеры входных данных (предложений, вопросов, ответов, документов) находятся в приложении 3.

Модульное тестирование

Модульное тестирование проводится для каждой библиотеки, для Pipeline (API), для вспомогательных средств общесистемных компонент.

Тестируемые объекты:

1. Библиотеки
    - 1.1. Библиотека предобработки текста
        - 1.1.1. Функция read_file (чтение файла из файловой системы)
        - 1.1.2. Функция clear_text (очистка текста от управляющих, повторяющихся символов)
        - 1.1.3. Функция sentenize (разделение строки на предложения)
        - 1.1.4. Функция find_lists (поиск списков в тексте)
        - 1.1.5. Функция extend_html_content (дополнение содержимого html-файла)
        - 1.1.6. Функция parse_html (чтение html файла, удаление тегов)
        - 1.1.7. Функция parse_docx (чтение docx файла, удаление тегов)
    - 1.2. Библиотека классификации предложений
        - 1.2.1. Функция classificate_sen (классификация предложений)
        - 1.2.2. Функция classificate_list (классификация списков)
    - 1.3. Библиотека генерации вопроса и поиска верного ответа

1.3.1. Функция get_question (получение вопроса)
1.3.2. Функция get_answer (получение верного ответа)
1.4. Библиотека генерации дистракторов
1.4.1. Функция get_distractors (получение дистракторов)
1.5. Библиотека валидации вопроса
1.5.1. Функция validate_morph (исправление морфологических ошибок)
1.5.2. Функция validate_punct (исправление пунктуационных ошибок)
1.5.3. Функция validate_spell (исправление орфографических ошибок)
2. Pipeline (API)
2.1. Функция create_stanza (создание экземпляра синтаксического парсера на GPU/CPU)
2.2. Функция create_pymorphy (создание экземпляра морфлогического парсера на GPU/CPU)
2.3. Функция create_number_extractor (создание экземпляра замены числительных на GPU/CPU)
2.4. Функция get_classification_model (подгрузка модели классификации в GPU)
2.5. Функция get_quesiton_gen_model (подгрузка модели генерации вопросов в GPU)
2.6. Функция get_distractor_gen_model (подгрузка модели генерации дистракторов в GPU)
2.7. Функция get_validate_model (подгрузка модели валидации в GPU)
3. Общесистемные компоненты
3.1. Синтаксический анализатор
3.1.1. Функция get_root_id (получение id главного слова)
3.1.2. Функция tokenize_text (получение токенов предложения)
3.1.3. Функция find_root_adjs (поиск прилагательных)
3.1.4. Функция are_demonstrative_pronouns_present (поиск указательных местоимений)

3.1.5. Функция find_dependent_words_ids (поиск зависимых слов слова)
3.1.6. Функция find_distance_to_root (поиск ребер дерева от слова до главного слова в предложении)
3.1.7. Функция count_predicates (подсчет связных глагольных сущностей)
3.1.8. Функция get_number (получение числа слова)
3.1.9. Функция get_gender (получения рода слова)
3.2. Преобразование регулярных выражений
3.2.1. Функция get_pattern (компиляция регулярных выражений)

Интеграционное тестирование

Интеграционное тестирование — тестирование, при которой отдельные программные модули объединяются и тестируются в группе.
В рамках проведения интеграционного тестирования тестируется взаимодействие:

1. Класса Pipeline (внутри функции generate) и класса Parser через функцию process
2. Класса Pipeline (внутри функции generate) и класса Classificator через функцию process
3. Класса Pipeline (внутри функции generate) и класса QuestionGenerator через функцию process
4. Класса Pipeline (внутри функции generate) и класса DistractorGenerator через функцию process
5. Класса Pipeline (внутри функции generate) и класса Validator через функцию process

Аттестационное тестирование

Аттестационное тестирование – это тестирование системы по функциональным требованиям. Оно используется для проверки, соответствует ли система заявленным требованиям.
Функциональные требования разделяются на два крупных сценария:

1. Генерация тестовых заданий
2. Дообучение ИИ-моделей

Исходя из сценариев, полученных по функциональным требованиям, в рамках аттестационного тестирования будут проведены тесты с имитацией вызова QG-модуля из Django-сервера (имитация, так как Django-сервер будет реализован после реализации QG-модуля).

Тестирование производительности

В ходе тестирования производительности необходимо получить среднее время обработки одного документа при генерации вопросов. Также, необходимо оценить временной промежуток, необходимый для дообучения моделей, от момента обработки запроса на дообучение до момента завершения.

3 План тестов

Окружение тестируемого объекта

Виртуальное окружение создано с помощью дистрибутива Anaconda: его облегченной версии miniconda. Версии основных пакетов окружения:

1. python: 3.9.13
2. pytest: 7.1.3
3. cudatoolkit: 11.6.0
4. pytorch: 1.12.1
5. stanza: 1.4.2
6. transformers: 4.23.1

Полный список используемых пакетов с их версиями находится в приложении 2.

Системные требования

Тестирование должно проводиться в системе, отвечающей следующим требованием:

1. GPU: NVIDIA с поддержкой CUDA и с объемом видеопамяти не менее 8 Гб.
2. CPU: Intel Core i3 (или i5, i7, i9) 10+ поколения
3. ОЗУ: 16 Гб или выше
4. ОС: Windows x64 ИЛИ Linux

Описание модульных тестов

Список и описание модульных тестов:

1. Библиотека предобработки текста (класс Parser)
   1.1. Тест M1-1
   Тип теста: положительный
   Объект тестирования: функция read_file
   Входные данные: название файла, который существует в файловой системе и доступен для чтения.
   Ожидаемый результат: содержимое файла в виде строки.

1.2. Тест M1-2
Тип теста: негативный
Объект тестирования: функция read_file
Входные данные: название файла, который существует в файловой системе и недоступен для чтения.
Ожидаемый результат: FileIsBlockedException

1.3. Тест M1-3
Тип теста: негативный
Объект тестирования: функция read_file
Входные данные: название файла, который не существует в файловой системе.
Ожидаемый результат: FileNotFoundException

1.4. Тест M1-4
Тип теста: позитивный
Объект тестирования: функция clear_text
Входные данные: “/r/t/n ab c d.”
Ожидаемый результат: “ab c d.”

1.5. Тест M1-5
Тип теста: негативный
Объект тестирования: функция clear_text
Входные данные: пустая строка
Ожидаемый результат: пустая строка

1.6. Тест M1-6
Тип теста: позитивный
Объект тестирования: функция sentenize
Входные данные: “Текст идет. Текст снова идет. Текст кончился.”
Ожидаемый результат: [“Текст идет”, “Текст снова идет.”. “Текст кончился.”]

1.7. Тест M1-7
Тип теста: позитивный
Объект тестирования: функция find_lists
Входные данные: [“абв”, “абг:”, “1 - а”, “2 - б”, “аде”]
Ожидаемый результат: [[“абг:”, “1 - а”, “2 - б”]]

1.8. Тест M1-8

Тип теста: позитивный
Объект тестирования: функция extend_html_content
Входные данные: “<h1>Header</h1> Text ”
Ожидаемый результат: “<h1>Header.</h1> Text. ”

1.9. Тест M1-9
Тип теста: позитивный
Объект тестирования: функция parse_html
Входные данные: “ text <script>python</script>”
Ожидаемый результат: “text”

1.10. Тест M1-10
Тип теста: негативный
Объект тестирования: функция parse_html
Входные данные: “text”
Ожидаемый результат: HtmlFormatException

1.11. Тест M1-11
Тип теста: позитивный
Объект тестирования: функция parse_docx
Входные данные: “<p:Pr>text</p:Pr>”
Ожидаемый результат: “text”

1.12. Тест M1-12
Тип теста: негативный
Объект тестирования: функция parse_docx
Входные данные: “text”
Ожидаемый результат: DocxFormatException

2. Библиотека классификации предложений (класс Classificator)

2.1. Тест M2-1
Тип теста: позитивный
Объект тестирования: функция classificate_sen
Входные данные: [“абв где”, ”abv gde”, ”qwe rty”]
Ожидаемый результат: [1, 0, 4]

2.2. Тест M2-2
Тип теста: позитивный
Объект тестирования: функция classificate_list
Входные данные: [“а:”, “б:”, “в:”]
Ожидаемый результат: [5, 2, 3]

3. Библиотека генерации вопроса и поиска верного ответа (класс QuestionGenerator)
    3.1. Тест M3-1
        Тип теста: позитивный
        Объект тестирования: функция get_question
        Входные данные: “а - б”
        Ожидаемый результат: “Что б?”
    3.2. Тест M3-2
        Тип теста: позитивный
        Объект тестирования: функция get_answer
        Входные данные: “а - б”
        Ожидаемый результат: “а”
4. Библиотека генерации дистракторов (класс DistractorGenerator)
    4.1. Тест M4-1
        Тип теста: позитивный
        Объект тестирования: функция get_distractors
        Входные данные: “дом”
        Ожидаемый результат: [“коттедж”, “мотель”, “сарай”]
5. Библиотека валидации вопроса (класс Validator)
    5.1. Тест M5-1
        Тип теста: позитивный
        Объект тестирования: функция validate_morph
        Входные данные: “две забора”
        Ожидаемый результат: “два забора”
    5.2. Тест M5-2
        Тип теста: позитивный
        Объект тестирования: функция validate_morph
        Входные данные: “два забора”
        Ожидаемый результат: “два забора”
    5.3. Тест M5-3
        Тип теста: позитивный
        Объект тестирования: функция validate_punct
        Входные данные: “система а не система”
        Ожидаемый результат: “система, а не система”
    5.4. Тест M5-4

Тип теста: позитивный
Объект тестирования: функция validate_punct
Входные данные: “система, а не система”
Ожидаемый результат: “система, а не система”

5.5. Тест M5-5
Тип теста: позитивный
Объект тестирования: функция validate_spell
Входные данные: “плахая строка”
Ожидаемый результат: “плохая строка”

5.6. Тест M5-6
Тип теста: позитивный
Объект тестирования: функция validate_spell
Входные данные: “хорошая строка”
Ожидаемый результат: “хорошая строка”

6. Pipeline (API) (класс Pipeline)

6.1. Тест M6-1
Тип теста: позитивный
Объект тестирования: функция create_stanza
Входные данные: нет
Ожидаемый результат: создан экземпляр синтаксического парсера на GPU

6.2. Тест M6-2
Тип теста: негативный
Объект тестирования: функция create_stanza
Входные данные: нет, GPU недоступна / нет свободной памяти
Ожидаемый результат: создан экземпляр синтаксического парсера на CPU

6.3. Тест M6-3
Тип теста: негативный
Объект тестирования: функция create_stanza
Входные данные: нет, данные моделей недоступны по пути /src/models/stanza_resources/
Ожидаемый результат: данные скачаны повторно, создан экземпляр синтаксического парсера на GPU/CPU

6.4. Тест М6-4
Тип теста: позитивный
Объект тестирования: функция create_pymorphy
Входные данные: нет, в системе есть словари pymorphy по пути /src/models/pymorphy2/
Ожидаемый результат: создан экземпляр морфологического анализатора

6.5. Тест М6-5
Тип теста: негативный
Объект тестирования: функция create_pymorphy
Входные данные: нет, в системе нет словарей pymorphy по пути /src/models/pymorphy2/ (или файлы повреждены)
Ожидаемый результат: данные словарей скачаны повторно, создан экземпляр морфологического анализатора

6.6. Тест М6-6
Тип теста: позитивный
Объект тестирования: функция create_number_extractor
Входные данные: нет
Ожидаемый результат: создан экземпляр заменителя числительных

6.7. Тест М6-7
Тип теста: позитивный
Объект тестирования: функция get_classifiaction_model
Входные данные: название модели (путь)
Ожидаемый результат: ИИ-модель для классификации подгружена в GPU

6.8. Тест М6-8
Тип теста: негативный
Объект тестирования: функция get_classifiaction_model
Входные данные: нет, на GPU нет достаточного количества свободной памяти
Ожидаемый результат: MemoryLimitException

6.9. Тест М6-9
Тип теста: позитивный
Объект тестирования: функция get_quesiton_gen_model

Входные данные: название модели
Ожидаемый результат: ИИ-модель для классификации подгружена в GPU

6.10. Тест М6-10
Тип теста: негативный
Объект тестирования: функция get_quesiton_gen_model
Входные данные: нет, на GPU нет достаочного количества свободной памяти
Ожидаемый результат: MemoryLimitException

6.11. Тест М6-11
Тип теста: позитивный
Объект тестирования: функция get_distractor_gen_model
Входные данные: название модели
Ожидаемый результат: ИИ-модель для классификации подгружена в GPU

6.12. Тест М6-12
Тип теста: негативный
Объект тестирования: функция get_distractor_gen_model
Входные данные: нет, на GPU нет достаочного количества свободной памяти
Ожидаемый результат: MemoryLimitException

6.13. Тест М6-13
Тип теста: позитивный
Объект тестирования: функция get_validate_model
Входные данные: название модели
Ожидаемый результат: ИИ-модель для классификации подгружена в GPU

6.14. Тест М6-14
Тип теста: негативный
Объект тестирования: функция get_validate_model
Входные данные: нет, на GPU нет достаочного количества свободной памяти
Ожидаемый результат: MemoryLimitException

7. Общесистемные компоненты:

7.1. Синтаксический анализатор (класс SyntaxAnalyzer)

7.1.1. Тест М7.1-1
Тип теста: позитивный
Объект тестирования: функция get_root_id
Входные данные: [{“id”: 0, “upos”: “nsubj”}, {“id”: 1, “upos”: ‘root’}, {“id”: 2, “upos”: “nmod”}]
Ожидаемый результат: 1

7.1.2. Тест М7.1-2
Тип теста: позитивный
Объект тестирования: функция tokenize_text
Входные данные: “слово”
Ожидаемый результат: [{“id”: 0, “upos”: “root”, “text”: “слово”}]

7.1.3. Тест М7.1-3
Тип теста: негативный
Объект тестирования: функция tokenize_text
Входные данные: пустая строка
Ожидаемый результат: None

7.1.4. Тест М7.1-4
Тип теста: позитивный
Объект тестирования: функция find_root_adjs
Входные данные: [{“id”: 0, “upos”: “root”, “text”: “слово”, “head”: 0}, {“id”: 1, “upos”: “nmod”, “text”: “красное”, “head”: 0}, {“id”: 2, “upos”: “nmod”, “text”: “белое”, “head”: 0},]
Ожидаемый результат: [1, 2]

7.1.5. Тест М7.1-5
Тип теста: позитивный
Объект тестирования: функция are_demonstrative_pronouns_present
Входные данные: [{“id”: 0, “upos”: “root”, “text”: “это”, “head”: 0}]
Ожидаемый результат: True

7.1.6. Тест М7.1-6
Тип теста: позитивный
Объект тестирования: функция find_dependent_words_ids

Входные данные: tuple(1, [{“id”: 0, “upos”: “root”, “text”: “слово”, “head”: 0}, {“id”: 1, “upos”: “nmod”, “text”: “красное”, “head”: 0}, {“id”: 2, “upos”: “nmod”, “text”: “как”, “head”: 1}, {“id”: 3, “upos”: “nmod”, “text”: “слово”, “head”: 2}])

Ожидаемый результат: [1, 2, 3]

7.1.7. Тест M7.1-7

Тип теста: позитивный

Объект тестирования: функция find_distance_to_root

Входные данные: tuple(“лебедь прыгал, дом скакал и думал”, обработанная синтаксическим анализатором; 3)

Ожидаемый результат: 1

7.1.8. Тест M7.1-8

Тип теста: позитивный

Объект тестирования: функция count_predicates

Входные данные: “лебедь прыгал, дом скакал и думал”, обработанная синтаксическим анализатором

Ожидаемый результат: 2

7.1.9. Тест M7.1-9

Тип теста: позитивный

Объект тестирования: функция get_number

Входные данные: объект с полем feats, равным “Number=Sing” или “Number=Plur”

Ожидаемый результат: число (‘single’ или ‘plural’) слова

7.1.10. Тест M7.1-10

Тип теста: позитивный

Объект тестирования: функция get_gender

Входные данные: объект с полем feats, равным “Gender=Femn” или “Gender=Masc” или “Gender=Neut”

Ожидаемый результат: род (‘femn’, ‘masc’ или ‘neut’) слова

7.2. Преобразование регулярных выражений (класс RegexPattern)

7.2.1. Тест М7.2-1
Тип теста: позитивный
Объект тестирования: функция get_pattern
Входные данные: список - [“а б в”, “г д е”]
Ожидаемый результат: единое скомпилированное регулярное выражение (тип данных re.compile)

7.2.2. Тест М7.2-2
Тип теста: негативный
Объект тестирования: функция get_pattern
Входные данные: список - [“+”, “-”, “.”, “?”, “*”]
Ожидаемый результат: единое скомпилированное регулярное выражение (тип данных - re.compile)

Описание интеграционных тестов

Список и описание интеграционных тестов:

1. Взаимодействие библиотек и Pipeline (API):

1.1. Тест И1-1
Тип теста: положительный
Объект тестирования: связь класса Pipeline (внутри функции generate) и класса Parser через функцию Parser.process()
Входные данные: файл, который доступен для чтения, содержимое файла: “абв. абг.”
Ожидаемый результат: Document({“data_lists”: “абв.”, “data_no_lists”: “абг.”})

1.2. Тест И1-2
Тип теста: негативный
Объект тестирования: связь класса Pipeline (внутри функции generate) и класса Parser через функцию Parser.process()
Входные данные: название файла, который существует в файловой системе и недоступен для чтения
Ожидаемый результат: None

1.3. Тест И1-3

Тип теста: негативный

Объект тестирования: связь класса Pipeline (внутри функции generate) и класса Parser через функцию Parser.process()

Входные данные: название файла, который не существует в файловой системе

Ожидаемый результат: None

1.4. Тест И1-4

Тип теста: позитивный

Объект тестирования: связь класса Pipeline (внутри функции generate) и класса Classificator через функцию Classificator.process()

Входные данные: [{“data_lists”: [“абв”], “data_no_lists”: [“абгв”]}]

Ожидаемый результат: QuestionList({“id”: 0, “text_fragment”: “абв”, “q_class”: 0}, {“id”: 1, “text_fragment”: “абгв”, “q_class”: 1})

1.5. Тест И1-5

Тип теста: позитивный

Объект тестирования: связь класса Pipeline (внутри функции generate) и класса QuestionGenerator через функцию QuestionGenerator.process()

Входные данные: QuestionList({“id”: 0, “text_fragment”: “абв”, “q_class”: 0}, {“id”: 1, “text_fragment”: “абг”, “q_class”: 1})

Выходные данные: QuestionList({“id”: 0, “text_fragment”: “абв”, “q_class”: 0, “question”: “а”, “answer”: “бв”}, {“id”: 1, “text_fragment”: “абг”, “q_class”: 1, “question”: “аб”, “answer”: “г”})

1.6. Тест И1-6

Тип теста: позитивный

Объект тестирования: связь класса Pipeline (внутри функции generate) и класса DistractorGenerator через функцию DistractorGenerator.process()

Входные данные: QuestionList({“id”: 0, “text_fragment”: “абв”, “q_class”: 0, “question”: “a”, “answer”: “бв”}, {“id”: 1, “text_fragment”: “абг”, “q_class”: 1, “question”: “аб”, “answer”: “г”})

Выходные данные: QuestionList({“id”: 0, “text_fragment”: “абв”, “q_class”: 0, “question”: “a”, “answer”: “бв”, “distractors”: [“c”, “д”, “e”]}, {“id”: 1, “text_fragment”: “абг”, “q_class”: 1, “question”: “аб”, “answer”: “г”, “distractors”: [“c”, “д”, “e”]})

1.7. Тест И1-7

Тип теста: позитивный

Объект тестирования: связь класса Pipeline (внутри функции generate) и класса Validator через функцию Validator.process()

Входные данные: QuestionList({“id”: 0, “text_fragment”: “абв”, “q_class”: 0, “question”: “система плахая”})

Выходные данные: QuestionList({“id”: 0, “text_fragment”: “абв”, “q_class”: 0, “question”: “система плохая”})

Описание аттестационных тестов

Список и описание аттестационных тестов:

1. Генерацию тестовых заданий

1.1. Тест А1-1

Тип теста: позитивный

Объект тестирования: QG-модуль

Входные данные: список из названия одного документа (содержимое: “Ответ - содержимое.”), GPU имеет более 6 Гб свободной видеопамяти

Ожидаемый результат: [{“id”: 0, “question”: “Как называется содержимое?”, “answer”: “Ответ”, “distractors”: [“Ответ1”, “Ответ2”, “Ответ3”], “competence”: “base”, “q_class”: “100”}]

1.2. Тест А1-2

Тип теста: негативный

Объект тестирования: QG-модуль

Входные данные: список документов (источников), GPU имеет менее 6 Гб свободной видеопамяти
Ожидаемый результат: MemoryLimitException

2. Дообучение ИИ-моделей

2.1. Тест А2-1

Тип теста: позитивный
Объект тестирования: QG-модуль
Входные данные: список данных для дообучения, GPU имеет более 6 Гб свободной видеопамяти
Ожидаемый результат: перезаписанные данные ИИ-моделей в папке /src/models/

2.2. Тест А2-2

Тип теста: негативный
Объект тестирования: QG-модуль
Входные данные: список данных для дообучения, GPU имеет менее 6 ГБ свободной видеопамяти
Ожидаемый результат: MemoryLimitException

Описание тестирования производительности

Тестирование производительности должно дать представление о скорости обработки одного документа модулем. Ссылка на хранилище с документами находится в приложении 4.

Пример реализации тестов

Пример программной реализации модульного теста:

```
def test_rusvectorize(self, classificators, question_generators, distractor_generators, toolkit):
    data = rusvectorize_data

    for test in data.keys():
        question_data = classificators[0].classificate(toolkit, data[test])
        if question_data is None:
            print(data[test])
            raise ValueError

        question, answer = question_generators[0].generate(question_data, toolkit)
        question_data.question = question
        question_data.answer = answer
        question_data = distractor_generators[0].generate(question_data, [question_data], 0, toolkit)

        assert (len(question_data.distractors) == 3
                and question_data.answer not in question_data.distractors
                and DistractorType.NotFound.value not in question_data.distractors)
```

Пример программной реализации интерграционного теста:

```python
def test_pipeline_and_classificator(classificators, pipeline_stub_class):
    data = [{"data_lists": ["абв"], "data_no_lists": ["абгв"]}]
    texts = ["абв", "абгв"]
    q_classes = [0, 1]

    # имитируем работу pipeline до момента вызова классификатора
    pre_data = pipeline_stub_class.prepare_data(data)
    output = classificators.process(pre_data)

    assert isinstance(QuestionList, output) # проверяем тип возвращаемого значения
    for i in range(len(output)):
        # проверяем что поля не пустые
        assert None not in [output.id, output.text_fragment, output.q_class]

        # проверяем совпадение полей
        assert ({"id": output.id, "text_fragment": output.text_fragment, "q_class": output.q_class}
                ==
                {"id": i, "text_fragment": texts[i], "q_class": q_classes[i]})
```

Пример запуска тестов:

```
(test) blazinghorizon@ai-prototype-Z590M:~/prof-evaluation/src/tests$ pytest
========================================= test session starts =========================================
platform linux -- Python 3.9.13, pytest-7.1.3, pluggy-1.0.0
rootdir: /home/blazinghorizon/prof-evaluation/src/tests, configfile: pytest.ini
plugins: cov-4.0.0, anyio-3.6.2
collected 32 items

classificator_test.py ..........                                                          [ 31%]
distractor_generation_test.py ......                                                      [ 50%]
question_generation_test.py ................                                              [100%]

========================================= 32 passed in 23.59s =========================================
```

Оценка покрытия тестирования

В качестве метода оценивания покрытия тестирования было выбрано покрытие кода, которое рассчитывается по следующей формуле:

$$Tcov = (Ltc/Lcode) * 100\%,$$

где Tcov - тестовое покрытие Ltc - количество строк кода, покрытых тестами Lcode - общее количество строк кода.

$$Tcov = 73\%$$

4 Журнал тестирования

Модульное тестирование

№ теста	Дата	Тестировщик	Результат	Номера найденных ошибок
M1-1	01.11.2022	Семенов Н.Д.	Пройден	-
M1-2	01.11.2022	Семенов Н.Д.	Пройден	-
M1-3	01.11.2022	Семенов Н.Д.	Пройден	-
M1-4	01.11.2022	Семенов Н.Д.	Пройден	-
M1-5	01.11.2022	Семенов Н.Д.	Пройден	-
M1-6	01.11.2022	Семенов Н.Д.	Пройден	-
M1-7	01.11.2022	Семенов Н.Д.	Пройден	-
M1-8	01.11.2022	Семенов Н.Д.	Пройден	-
M1-9	01.11.2022	Семенов Н.Д.	Пройден	-
M1-10	01.11.2022	Семенов Н.Д.	Пройден	-
M1-11	01.11.2022	Семенов Н.Д.	Пройден	-
M1-12	01.11.2022	Семенов Н.Д.	Пройден	-
M2-1	01.11.2022	Семенов Н.Д.	Пройден	-
M2-2	01.11.2022	Семенов Н.Д.	Пройден	-
M3-1	01.11.2022	Семенов Н.Д.	Пройден	-
M3-2	01.11.2022	Семенов Н.Д.	Пройден	-
M4-1	12.11.2022	Семенов Н.Д.	Пройден	1
M5-1	12.11.2022	Семенов Н.Д.	Пройден	-

М5-2	12.11.2022	Семенов Н.Д.	Пройден	-
М5-3	12.11.2022	Семенов Н.Д.	Пройден	-
М5-4	12.11.2022	Семенов Н.Д.	Пройден	-
М5-5	12.11.2022	Семенов Н.Д.	Пройден	-
М5-6	12.11.2022	Семенов Н.Д.	Пройден	-
М6-1	08.11.2022	Семенов Н.Д.	Пройден	-
М6-2	08.11.2022	Семенов Н.Д.	Пройден	-
М6-3	08.11.2022	Семенов Н.Д.	Пройден	-
М6-4	08.11.2022	Семенов Н.Д.	Пройден	-
М6-5	08.11.2022	Семенов Н.Д.	Пройден	-
М6-6	08.11.2022	Семенов Н.Д.	Пройден	-
М6-7	12.11.2022	Семенов Н.Д.	Пройден	-
М6-8	12.11.2022	Семенов Н.Д.	Пройден	2
М6-9	12.11.2022	Семенов Н.Д.	Пройден	-
М6-10	12.11.2022	Семенов Н.Д.	Пройден	3
М6-11	12.11.2022	Семенов Н.Д.	Пройден	-
М6-12	12.11.2022	Семенов Н.Д.	Пройден	-
М6-13	12.11.2022	Семенов Н.Д.	Пройден	-
М6-14	12.11.2022	Семенов Н.Д.	Пройден	-
М7.1-1	01.11.2022	Семенов Н.Д.	Пройден	-
М7.1-2	01.11.2022	Семенов Н.Д.	Пройден	-
М7.1-3	01.11.2022	Семенов Н.Д.	Пройден	5
М7.1-4	01.11.2022	Семенов Н.Д.	Пройден	-

М7.1-5	01.11.2022	Семенов Н.Д.	Пройден	-
М7.1-6	01.11.2022	Семенов Н.Д.	Пройден	6
М7.1-7	01.11.2022	Семенов Н.Д.	Пройден	-
М7.1-8	01.11.2022	Семенов Н.Д.	Пройден	-
М7.1-9	01.11.2022	Семенов Н.Д.	Пройден	-
М7.1-10	01.11.2022	Семенов Н.Д.	Пройден	-
М7.2-1	01.11.2022	Семенов Н.Д.	Пройден	-
М7.2-2	01.11.2022	Семенов Н.Д.	Пройден	7

Интеграционное тестирование

№ теста	Дата	Тестировщик	Результат	Номера найденных ошибок
И1-1	12.11.2022	Семенов Н.Д.	Пройден	-
И1-2	12.11.2022	Семенов Н.Д.	Пройден	-
И1-3	12.11.2022	Семенов Н.Д.	Пройден	-
И1-4	12.11.2022	Семенов Н.Д.	Пройден	-
И1-5	12.11.2022	Семенов Н.Д.	Пройден	-
И1-6	12.11.2022	Семенов Н.Д.	Пройден	-
И1-7	12.11.2022	Семенов Н.Д.	Пройден	-
И1-8	12.11.2022	Семенов Н.Д.	Пройден	-

Аттестационное тестирование

№ теста	Дата	Тестировщик	Результат	Номера найденных ошибок
A1-1	12.11.2022	Семенов Н.Д.	Пройден	-
A1-2	-	-	Не реализован	-
A2-1	-	-	Не реализован	-
A2-2	-	-	Не реализован	-

Тестирование производительности

Средняя скорость обработки документов из приложения 4 составила 5.64 секунды.
Максимальное время обработки - 25.21 секунды.
Минимальное время обработки - 0.25 секунды.

5 Журнал ошибок

Ошибка 1.
Номер теста: M4-1
Модуль: DistractorGenerator.get_distractors()
Тип ошибки: несовпадение ожидаемого и действительного значения
Описание: функция вместо списка из 3 дистракторов вернула список из 5 дистракторов
Предлагаемое исправление: возвращать первые 3 дистрактора внутри функции
Автор отчета: Семенов Н. Д.
Ответственный за исправление: Семенов Н. Д.
Состояние: исправлено, тест пройден

Ошибка 2.
Номер теста: M6-8
Модуль: Pipeline.get_classification_model()
Тип ошибки: несовпадение ожидаемого и действительного значения
Описание: вместо возврата MemoryLimitException, вернулся обычный Exception
Предлагаемое исправление: добавить создание и возврат MemoryLimitException
Автор отчета: Семенов Н. Д.
Ответственный за исправление: Семенов Н. Д.
Состояние: исправлено, тест пройден

Ошибка 3.
Номер теста: M6-10
Модуль: Pipeline.get_quesiton_gen_model()
Тип ошибки: зацикливание
Описание: превышена максимальная глубина рекурсии
Предлагаемое исправление: проверить функцию на бесконечный цикл
Автор отчета: Семенов Н. Д.
Ответственный за исправление: Семенов Н. Д.
Состояние: исправлено, тест пройден

Ошибка 4.
Номер теста: M7.1-3
Модуль: SyntaxAnalyzer.tokenize_text()
Тип ошибки: выброс Exception
Описание: вместо возврата None, выбрасывается Exception
Предлагаемое исправление: проверить часть кода, ответственного за обработку крайних случаев (пустая входная строка)
Автор отчета: Семенов Н. Д.
Ответственный за исправление: Семенов Н. Д.
Состояние: исправлено, тест пройден

Ошибка 5.
Номер теста: M7.1-6
Модуль: SyntaxAnalyzer.find_dependent_words_ids()
Тип ошибки: ожидаемое значение не совпадает с действительным
Описание: функция возвращает число, большее на 1 от необходимого
Предлагаемое исправление: возвращать значение, меньшее на 1 от полученного
Автор отчета: Семенов Н. Д.
Ответственный за исправление: Семенов Н. Д.
Состояние: исправлено, тест пройден

Ошибка 6.
Номер теста: M7.2-2
Модуль: RegexPattern.get_pattern()
Тип ошибки: выброс Exception
Описание: если в строке содержится неэкраннированный символ “+”, выбрасывается Exception
Предлагаемое исправление: обработать экранирование “+”
Автор отчета: Семенов Н. Д.
Ответственный за исправление: Семенов Н. Д.
Состояние: исправлено, тест пройден

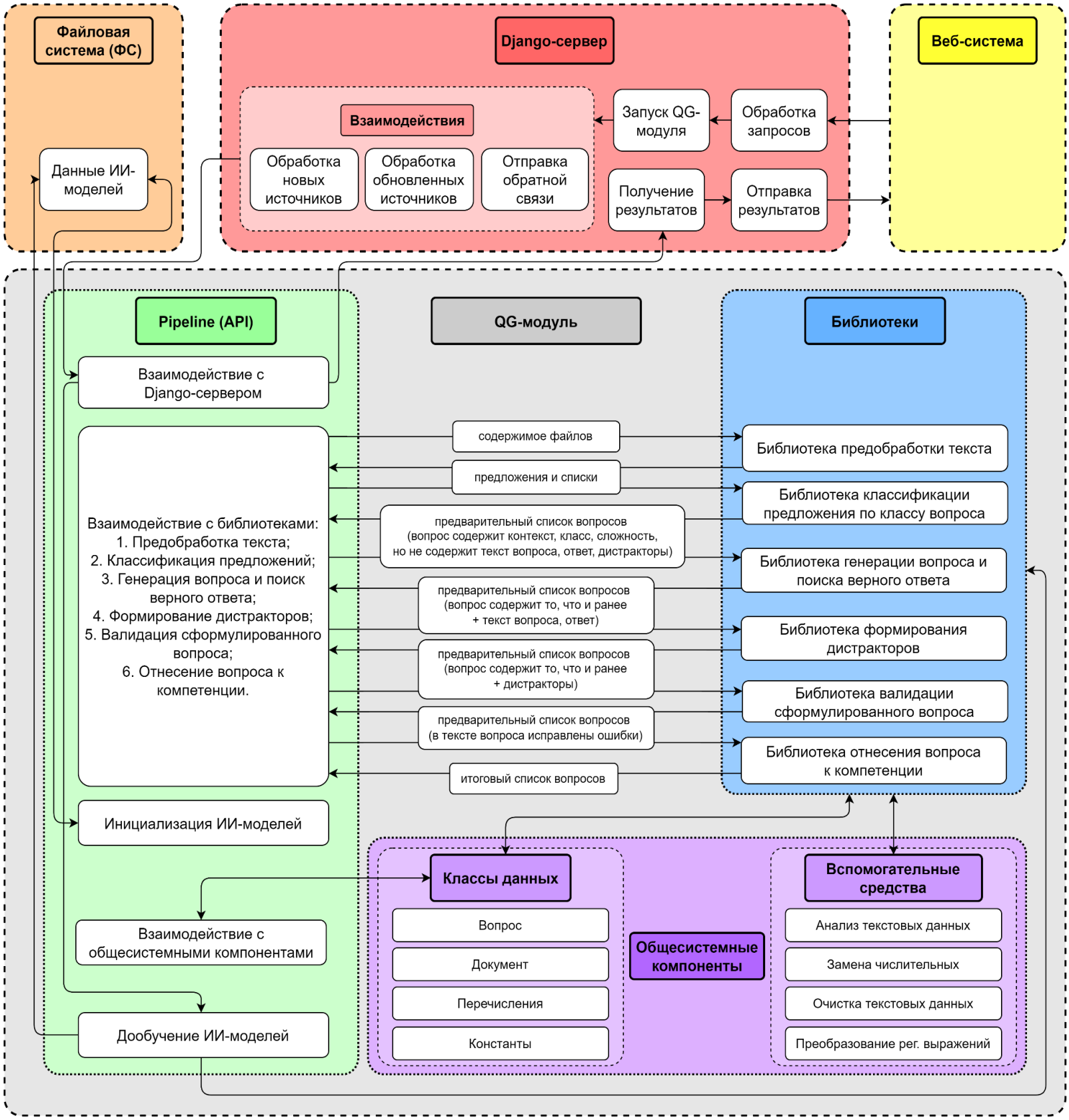
6 Результаты

Модульное, интеграционное тестирование было проведено в полном объеме. Найденные ошибки были задокументированы и исправлены. Не было полностью проведено аттестационное тестирование, планируется сделать это в будущем.

Тестирование можно считать пройденным успешно.

7 Приложение

Приложение 1 - Схема архитектуры системы



Приложение 2 - Окружение тестируемого объекта

1. beautifulsoup4=4.11.1
2. cryptography=37.0.1
3. cudatoolkit=11.6.0
4. gensim=4.2.0
5. libgcc-ng=11.2.0
6. libgomp=11.2.0
7. libidn2=2.3.2
8. libopus=1.3.1
9. libpng=1.6.37
10. libsodium=1.0.18
11. libstdcxx-ng=11.2.0
12. nest-asyncio=1.5.5
13. numpy=1.23.1
14. openpyxl=3.0.10
15. openssl=1.1.1s
16. pip=22.2.2
17. pymorphy2=0.9.1
18. pymorphy2-dicts-ru=2.4.4
19. pyopenssl=22.0.0
20. pyparsing=3.0.9
21. pysocks=1.7.1
22. pytest=7.1.3
23. python=3.9.13
24. pytorch=1.12.1
25. pyyaml=6.0
26. razdel=0.5.0
27. regex=2022.9.13
28. sentencepiece=0.1.97
29. stanza=1.4.2
30. tokenizers=0.13.1
31. transformers=4.23.1
32. zlib=1.2.12

Приложение 3 - Документы для тестирования производительности

url: https://cool-docs.ru/documents_for_testing/