План тестирования системы помощи абитуриенту

Выполнил: Бурдин Григорий Олегович
Проверил: Кулаков Кирилл

Петрозаводск — 2020

Объект тестирования

В качестве объекта тестирования выступает вопросно-ответная система. Тестируемая система принимает от пользователя вопрос на естественном языке, в качестве результата работы системы возвращается набор документов, которые вероятнее всего содержат ответ на поставленный вопрос, а также короткий сниппет документа, который система сочла наиболее релевантным для запроса пользователя.

Требования к тестируемой системе:

1. Система должна автоматически строить поисковый индекс для всех текстовых документов в заранее определенной директории
2. Система должна принимать от пользователя вопрос на естественном языке
3. Система должна проводить поиск релевантных запросу пользователя документов
4. Система должна предоставлять пользователю список документов, ранжированных по релевантности запросу пользователя

Задачами тестирования являются:

- Обнаружение ошибок в работе системы.
- Повышение качества системы за счет оптимизации и совершенствования проблемных модулей.
- Определение качества работы системы (доля релевантных результатов для определённого набора пользовательских запросов).

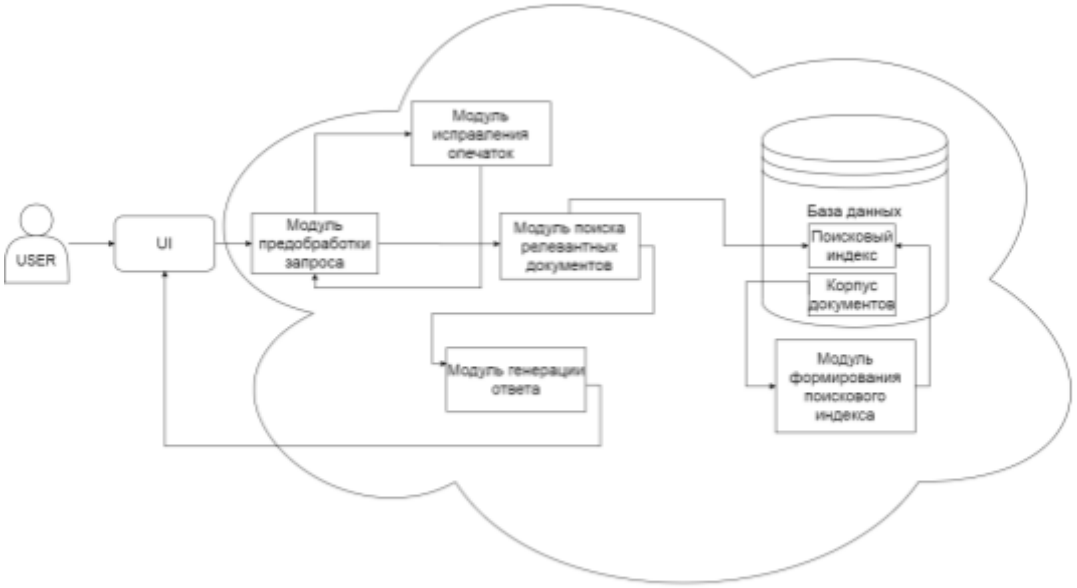

Рис. 1. Высокоуровневая архитектура системы

Основными модулями системы являются:

1. Модуль формирования поискового индекса

   Принимает корпус документов на естественном языке, производит их предобработку, строит матрицу терм-документ, по схеме TF-IDF определяет веса для элементов матрицы.

   Модуль содержит следующие функции:

   - prepareText - производит предобработку (нормировку) текста документа, удаляя лишние символы, токенизируя и лемматизирую термы
   - buildFrequencyMatrix - строит матрицу терм-документ для каждого всего корпуса документов
   - buildIndex - строит матрицу весов терм-документ для реализации поискового алгоритма

2. Модуль предобработки запроса

   Преобразует получаемый пользователем запрос в вектор термов, имеющих значение для поискового алгоритма. Вызывает модуль исправления опечаток.

   Модуль содержит следующие функции:

   - prepRequest - производит предобработку (нормировку) запроса пользователя, удаляя лишние символы, лемматизируя термы, производит токенизацию и удаление термов, которые не вошли в поисковый индекс
   - buildWeights - строит вектор весов для термов в запросе пользователя

3. Модуль исправления опечаток

   Исправляет допущенные пользователем опечатки при написании запроса.

   Модуль содержит следующие функции:

   - correct - производит попытку исправить слово, введенное пользователем с ошибкой

4. Модуль поиска релевантных документов

   По полученному на этапе предобработки запроса вектору и поисковому индексу, рассчитывает список документов, релевантных запросу пользователя

   Модуль содержит следующие функции:

   - findDocuments - производит поиск релевантных документов с использованием поискового индекса и вектора весов запрос с использованием TF-IDF схемы

5. Модуль генерации ответа

   Составляет ответ пользователю на естественном языке.

   Модуль содержит следующие функции:

   - prepDocNames - подготавливает список названий документов для использования их в формировании ответа пользователю
   - response - формирует ответ пользователю

Система реализована на языке программирования Python с использованием библиотек:

- PyMystem3 - python-оболочка для морфологического анализатора русского языка Yandex MyStem 3.1 (модули 1, 3).

- Natasha - набор открытых инструментов для обработки естественного русского. языка (модули 1, 2, 3).
- NLTK - пакет библиотек и программ для символьной и статистической обработки естественного языка (модули 1, 3).
- pandas - программная библиотека на языке Python для обработки и анализа данных (модули 1, 3, 4).
- numpy - библиотека для выполнения математических расчетов (модули 1, 3, 4).

Стратегия тестирования

Тестирование будет проводиться для всех указанных выше модулей. Перед проведением тестирования был заготовлен набор из 150 запросов пользователей, которые будут выступать в роли входных данных на различных этапах тестирования. Для каждого запроса заранее определен ожидаемый результат.
Если результатом выполнения функции является вещественное число, то сравнение с ожидаемым результатом производится с точностью до тысячных (3 знака после запятой).
Для тестирования используются текстовые файлы (16 файлов), содержащие информацию об аспектах процесса проведения приёмной кампании, а также поисковый индекс, сформированный с использованием этих файлов. Каждый из файлов содержит текст на русском языке, который предлагается, как ответ на запрос пользователя. Для проведения тестов с использованием файлов формата, отличного от txt, подойдёт любой файл указанного формата.

Таблица 1. Краткое описание файлов.

№	Описание
1	Правила и сроки подачи заявления
2	Информация о рангах и приоритетах
3	Сроки и сценарии действий по завершению приема документов
4	Документы, необходимые для зачисления
5	Зачисление без вступительных испытаний, на места в пределах особой квоты и целевой квоты
6	Первая волна зачисления
7	Второй этап зачисления, отказ от зачисления, изменение приоритета, сценарии
8	Зачисление на обучение на платной основе
9	Всё о общежитиях
10	Сроки и кабинеты для подачи документов
11	Количество дополнительных баллов и условия их начисления за индивидуальные достижения поступающих

12	Все о поступлении на заочное обучение
13	Все о поступлении в магистратуру
14	Правила подачи апелляции
15	Регламент прохождения медицинского осмотра для направлений, которым это требуется
16	Перечень средних и минимальных баллов для поступления на различные направления

Блочное тестирование

В ходе блочного тестирование проверяется работоспособность функций программы. Для этого в качестве входных параметров функции подаются различные значения, после чего результат работы функции сравнивается с ожидаемым.
Тестирование выполняется автоматически с использованием программной оболочки для Jupyter Notebook.

В блочном тестировании используются следующие функции:

1. prepareText (модуль 1)
2. prepRequest (модуль 2)
3. buildFrequencyMatrix (модуль 1)
4. prepDocNames (модуль 5)

Для проведения тестирования были выбраны данные функции, т.к. они являются важнейшими для подготовки входных данных перед дальнейшей их обработкой. Функции, выполняющиеся далее, либо несут в себе строгую логику, либо опираются на результаты выполнения выбранных для тестирования функций.

Описание тестируемых функций:

1. **prepareText**

Назначение функции: подготовка текста документа к обработке функциями построения индекса, а именно: приведение текста к нижнему регистру, удаление не кириллических символов, знаков препинания, избыточных пробелов, токенизация, лемматизация слов удаление стоп слов (на основании списка стоп слов для русского языка в библиотеке NLTK).
Важность: Тестирование функции снижает вероятность ошибки на начальном этапе формирования поискового индекса, а также прямым образом влияет на термы, которые попадают в сам индекс.
Входные параметры:

- raw_text - string: исходный текст документа

Выходные параметры:

- terms - array: термы, необходимые для формирования индекса

2. **buildFrequencyMatrix**

Назначение функции: построение матрицы терм - документ на основании исходных файлов с текстом (используется функция prepareText)
Важность: Функция строит частотные матрицы для документов, что является важной частью алгоритма построения индекса.
Входные параметры:

- docs - array: список документов, которые нужно проиндексировать

Выходные параметры:

- frequency - DataFrame: датафрейм частот термов в документах

3. **prepRequest**

Назначение функции: предварительно обработать запрос пользователя (приведение к нижнему регистру, удаление не кириллических символов, лишних пробелов, знаков препинания, токенизация, лемматизация, исключение слов, не представленных в индексе)
Важность: Функция подготавливает полученный от пользователя запрос, что прямым образом влияет на качество поиска.
Входные параметры:

- raw_requrst - string: исходный запрос пользователя

Выходные параметры:

- request - dictionary: термы, необходимые для поиска, и частота их встречаемости в запросе

4. **prepDocNames**

Назначение функции: формирование списка документов и сниппетов для ответа пользователю
Важность: Функция подготавливает список имён документов для вывода ответа. При наличии ошибок в функции, даже при верных результатах, ответ может выдаваться неправильный, что сведёт к нулю качество системы.
Входные параметры:

- index - list: список всех документов в поисковом индексе

Выходные параметры:

- docs - dictionary: словарь, где ключ - название документа, а значение - сниппет документа

Интеграционное тестирование

На этапе интеграционного тестирование проверяется корректность работы модулей, т.е. группы взаимодействующих функций.

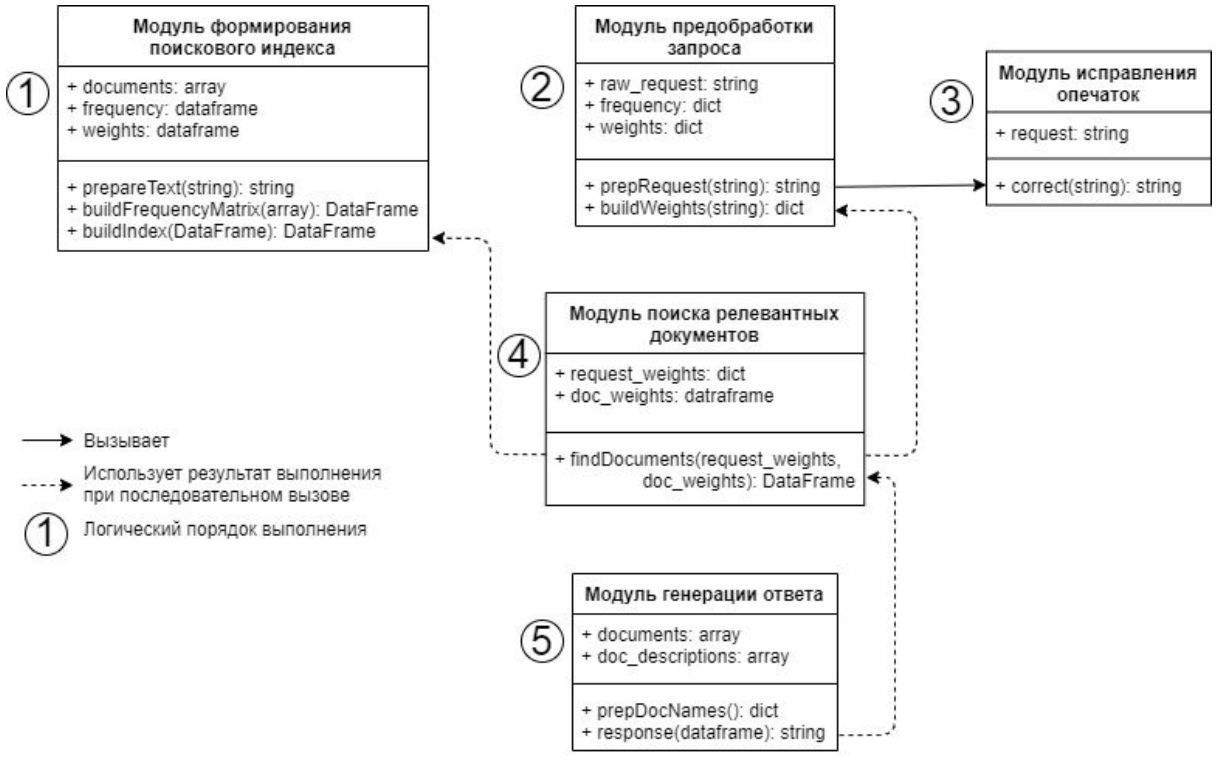

Рис.2. Схема взаимосвязи модулей в системе

В интеграционном тестировании используется модуль предобработки запроса.Тестирование производится с использованием программной оболочки для Jupyter Notebook.

Модули предобработки запроса и поиска релевантных документов

Входные данные: запрос пользователя на естественном языке.

Данные подаются на вход функции prepRequest, проверяемым результатом является выход функции findDocuments.

findDocuments
Входные параметры:

- request_weights: dict - Словарь весов термов в запросе
- doc_weights:DataFrame - датафрейм весов термов в документах

Выходные параметры:

- DataFrame - Датафрейм документов с коэффициентами релевантности

Назначение функции: По полученным весам термов запроса и весам термов документов определить коэффициент релевантности документов запросу.

Назначение модуля: нормировать запрос пользователя, построить частотный словарь и словарь весов термов в запросе пользователя, провести поиск по заранее сформированному индексу, вывести пользователю ранжированный список релевантных документов.
Выходные данные: список документов с весами.

Аттестационное тестирование

В ходе Аттестационного тестирование проверяется корректность работы системы в целом в условиях, приближенным к условиям эксплуатации. Тестирование происходит в ручном режиме.

Настройка окружения для проведения тестирования

Для проведения тестирования, необходимо:

1. Установить Jupyter Notebook на ваш компьютер
2. Загрузить библиотеки NLTK, Natasha, numpy, pandas для Python используя pip (консольная команда pip -install <НАЗВАНИЕ ПАКЕТА>)
3. Разместить директории “raw_documents”, “index” с файлами для тестирования в папке с файлом тестируемого функционала
4. Запустить .ipynb файл в Jupiter Notebook и выполнить его целиком

Тестирование производится в локальном режиме без соединения с интернетом..

Тестируемый функционал

В проведении аттестационного тестирования участвует следующий функционал:

1. Построение поискового индекса для всех текстовых документов в заранее определенной директории
2. Получение от пользователя вопроса на естественном языке
3. Поиск релевантных запросу пользователя документов
4. Предоставление пользователю списка документов, ранжированных по релевантности его запросу

В рамках тестирования проводится проверка корректности системы при корректных, некорректных и пустых входных данных. В описании ожидаемых результатах проведения теста незначительная часть ответа может быть пропущена.

Нагрузочное тестирование

Для проведение нагрузочного тестирования был сформирован список 50 запросов, которые последовательно направлялись на вход системе. Каждый следующий запрос подавался на вход системе после получения ответа на предыдущий.
Целью тестирования было выяснить, справится ли система со всеми 50 запросами за минуту.

Тестирование проводится в 4 этапа:

1. Использующиеся запросы состоят из не более чем трех слов
2. Использующиеся запросы состоят из не менее, чем трех, но не более, чем пяти
3. Использующиеся запросы состоят из не менее, чем пяти, но не более, чем десяти слов
4. Использующиеся запросы состоят из не менее, чем из десяти слов

Стоп слова учитываются при подсчете слов запроса. Словарь в ходе проведения тестирования является фиксированным. Порядок запросов определяется случайно.

Условия начала тестирования

Тестирование рекомендуется начинать, когда реализованы тестируемые модули и функции системы.

Условия окончания тестирования

Тестирование следует окончить, если более 80% тестов прошли успешно или более 50% тестов (не менее 5) при начале тестирования завершились неудачей. В таком случае следует передать тестируемые модули или функции на доработку.

Условия перехода между этапами тестирования

Переходить к следующему этапу тестирование следует в том случае, если в ходе тестирования текущего этапа более 80% тестов завершились успешно.

Детальный план тестов

Блочные тесты

Функция prepareText(raw_test: string) → terms: array

№	1
Цель теста	Проверка правильности работы функции prepareText при корректно заданных входных данных
Тип теста	Позитивный
Входные данные	raw_text: “Строка абсолютно нормального текста без лишних чего-то”
Ожидаемый результат	['строка', 'абсолютно', 'нормальный', 'текст', 'лишний']

№	2
Цель теста	Проверка правильности работы функции prepareText при пустых входных данных
Тип теста	Негативный
Входные данные	raw_text: ‘’
Ожидаемый результат	[ ]

№	3
Цель теста	Проверка правильности работы функции prepareText при наличии во входных данных не кириллических символов
Тип теста	Негативный
Входные данные	raw_text: “Строка абсолютно H0pM4Jlbnor0 текста без лишних чего-то”
Ожидаемый результат	['строка', 'абсолютно', 'текст', 'лишний']

Функция prepRequest(raw_request: string) → frequency: DataFrame

№	4
Цель теста	Проверка правильности работы функции prepareRequest при входных данных, состоящих исключительно из не кириллических символов
Тип теста	Негативный

Входные данные	raw_request: ‘lorem ipsum dolor sit amet’
Ожидаемый результат	[ ]

Функция buildFrequencyMatrix(docs: array) → frequency: DataFrame

№	5
Цель теста	Проверка правильности работы функции buildFrequencyMatrix при пяти входных файлах
Тип теста	Позитивный
Входные данные	docs: [‘0.txt’, ‘1.txt’, ‘2.txt’, ‘3.txt’, ‘4.txt’]
Окружение тестирования	Файлы ‘0.txt’, ‘1.txt’, ‘2.txt’, ‘3.txt’, ‘4.txt’ в папке ‘raw_documents’
Ожидаемый результат	

	0.txt	1.txt	2.txt	3.txt	4.txt
карета	1.0	1.0	0.0	0.0	0.0
пассажирский	1.0	0.0	0.0	0.0	0.0
повозка	1.0	0.0	0.0	0.0	0.0
конный	1.0	1.0	0.0	1.0	0.0
тяга	1.0	1.0	0.0	0.0	1.0
дилижанс	0.0	1.0	0.0	0.0	0.0
многоместный	0.0	1.0	0.0	0.0	0.0
германия	0.0	1.0	1.0	0.0	1.0
официальный	0.0	1.0	1.0	0.0	1.0
радебейль	0.0	0.0	1.0	0.0	0.0
государство	0.0	0.0	0.0	1.0	0.0
европа	0.0	0.0	0.0	1.0	0.0
берлин	0.0	0.0	0.0	1.0	0.0
немецкий	0.0	0.0	0.0	1.0	0.0
язык	0.0	0.0	0.0	0.0	3.0
немец	0.0	0.0	0.0	0.0	1.0

№	6
Цель теста	Проверка правильности работы функции buildFrequencyMatrix при отсутствии входных файлов
Тип теста	Негативный
Входные данные	docs: [ ]
Ожидаемый результат	Пустой датафрейм

№	7

Цель теста	Проверка правильности работы функции buildFrequencyMatrix при наличии в списке файла не текстового формата входных файлов
Тип теста	Негативный
Входные данные	docs: [‘0.txt’, ‘1.txt’, ‘2.txt’, ‘3.txt’, ‘4.txt’, ‘5.png’]
Окружение тестирования	Файлы ‘0.txt’, ‘1.txt’, ‘2.txt’, ‘3.txt’, ‘4.txt’, ‘5.png’ в папке “raw_documents”
Ожидаемый результат	Сообщение об ошибке “Некорректный формат файла в списке”

Функция prepDocNames(index: array) → docs: dictionary

№	8
Цель теста	Проверка корректности формирования списка имен документов функцией prepDocNames при наличии в папке файлов исключительно текстовых документов с расширением txt
Тип теста	Позитивный
Входные данные	index: [‘0.txt’, ‘1.txt’, ‘2.txt’, ‘3.txt’, ‘4.txt’, ‘5.txt]
Окружение тестирования	Файлы [‘0.txt’, ‘1.txt’, ‘2.txt’, ‘3.txt’, ‘4.txt’, ‘5.txt] в папке raw_documents

Ожидаемый результат	docs: {
	‘0.txt’ : ‘Сниппет документа 0’,
	‘1.txt’ : ‘Сниппет документа 1’,
	‘2.txt’ : ’Сниппет документа 2’,
	‘3.txt’ : ‘Сниппет документа 3’,
	‘4.txt’ : ‘Сниппет документа 4’,
	‘5.txt’ : ‘Сниппет документа 5’,
	}

№	9
Цель теста	Проверка корректности формирования списка имен документов функцией prepDocNames при наличии в папке файлов как текстовых документов с расширением txt, так и файлов с расширением png
Тип теста	Негативный
Входные данные	index: [‘0.txt’, ‘1.txt’, ‘2.txt’, ‘3.txt’, ‘4.txt’, ‘5.png]
Окружение тестирования	Файлы [‘0.txt’, ‘1.txt’, ‘2.txt’, ‘3.txt’, ‘4.txt’, ‘5.png] в папке ‘raw_documents’

Ожидаемый результат	docs: {
	‘0.txt’ : ‘Сниппет документа 0’,
	‘1.txt’ : ‘Сниппет документа 1’,
	‘2.txt’ : ’Сниппет документа 2’,
	‘3.txt’ : ‘Сниппет документа 3’,
	‘4.txt’ : ‘Сниппет документа 4’,
	}

№	10
Цель теста	Проверка корректности формирования списка имен документов функцией prepDocNames при отсутствии файлов в папке
Тип теста	Негативный
Входные данные	index: [ ]
Ожидаемый результат	[ ]

Интеграционные тесты

Модуль предобработки запроса и поиска релевантных документов

№	11
Цель теста	Проверить корректность формирования матриц частот и весов при корректном запросе
Тип теста	Позитивный
Входные данные	request: “Сроки подачи документов на поступление’’
Ожидаемый результат	response: {‘4.txt’: 0.8267, 5.txt: 0.4931, ‘2.txt’: 0.36463, ‘1.txt’: 0,19315}

№	12
Цель теста	Проверить корректность формирования матриц частот и весов при пустом запросе
Тип теста	Негативный
Входные данные	request: ‘’
Ожидаемый результат	[ ]

№	13
Цель теста	Проверить корректность формирования матриц частот и весов при запросе, состоящем из стоп слов
Тип теста	Негативный
Входные данные	request: ‘но пока где почему’
Ожидаемый результат	[ ]

№	14
Цель теста	Проверить корректность формирования матриц частот и весов при запросе на английском языке
Тип теста	Негативный
Входные данные	request: ‘how can i submit documents’
Ожидаемый результат	[ ]

Аттестационные тесты

№	15
Цель теста	Проверка формирования поискового индекса (треб. 1)
Тип теста	Позитивный
Входные данные	Ввод в строку: “Где найти список документов?”
Ожидаемый результат	Система вывела любой ответ

№	16
Цель теста	Проверка получения запроса (треб. 2)
Тип теста	Позитивный
Окружение тестирования	Файлы [‘0.txt’, ‘1.txt’, ‘2.txt’, ‘3.txt’, ‘4.txt’, ‘5.txt’] в папке ‘raw_documents’
Ожидаемый результат	Создан csv-файл поискового индекса в папке index

№	17
Цель теста	Проверить корректность получения ответа на запрос пользователя на русском языке (треб. 3, 4)
Тип теста	Позитивный
Входные данные	request: ‘Где посмотреть список документов, необходимых для зачисления?’
Ожидаемый результат	04.txt — Документы, необходимые для зачисления 06.txt — Первая волна зачисления 07.txt — Вторая волна зачисления 08.txt — Зачисление на обучение на платной основе

№	18
Цель теста	Проверить корректность получения ответа на запрос пользователя на английском языке (треб. 3, 4)
Тип теста	Негативный
Входные данные	request: ‘Where can I see the list of documents required for

	enrollment?’
Ожидаемый результат	“По вашему запросу ничего не найдено! Попробуйте переформулировать.”

№	19
Цель теста	Проверить корректность получения ответа на пустой запрос (треб. 3, 4)
Тип теста	Негативный
Входные данные	request: ‘’
Ожидаемый результат	“Пустой запрос! Введите ваш вопрос.”

Нагрузочное тестирование

№ этапа	Число пройденных запросов	Флаг
1	50	
2	37	
3	20	
4	14	

В ходе тестирование выяснилось, что система на некоторых запросах (особенно длинных) работает дольше, до 5 секунд на запрос. За отведенную минуту, система не справилась с поставленным числом запросов на этапах 2,3 и 4.

Оценка покрытия кода тестами

При ручном исследовании кода и тестов, было установлено, что:

- 35% кода покрыто блочными тестами;
- 50% кода покрыто интеграционными тестами;
- 80% кода покрыто аттестационными тестами.

Не протестированными остались модули формирования ответа пользователю и модуль исправления опечаток, поскольку на момент проведения тестирования, они находятся в разработке.

Журнал тестирования

№ теста	Результат теста	Флаг	Пояснение
1	['строка', 'абсолютно', 'нормальный', 'текст', 'лишний']		Результат соответствует ожидаемому
2	[ ]		Результат соответствует ожидаемому
3	['строка', 'абсолютно', 'текст', 'лишний']		Результат соответствует ожидаемому
4	[ ]		Результат соответствует ожидаемому
5	Получена матрица частот		Результат соответствует ожидаемому
6	[ ]		Результат соответствует ожидаемому
7	Ошибка		Не предусмотрено наличие не текстовых файлов в папке
8	Список сниппетов		Результат соответствует ожидаемому
9	Ошибка		На моменте попытки обработки не текстового файла, возникла ошибка
10	[ ]		Результат соответствует ожидаемому
11	Список весов документов		Результат соответствует ожидаемому
12	[ ]		Результат соответствует ожидаемому
13	[ ]		Результат соответствует ожидаемому
14	[ ]		Результат соответствует ожидаемому
15	Получен ответ от системы		Результат соответствует ожидаемому
16	Файл индекса создан		Результат соответствует ожидаемому
17	Список документов с названиями и пояснениями, релевантными запросу		Результат соответствует ожидаемому
18	“Пустой запрос! Введите ваш вопрос.”		Все символы были удалены при предобработке
19	“Пустой запрос! Введите ваш вопрос.”		Результат соответствует ожидаемому

Журнал найденных ошибок

№ теста	Полученный результат	Ожидаемый результат	Комментарий
7	Сообщение об ошибке	Сообщение об ошибке “Некорректный формат файла в списке”	Если в папке raw_documents есть не текстовые файлы, программа выдаёт ошибку

9	Сообщение об ошибке	docs: {
		‘0.txt’ : ‘Сниппет документа 0’,
		‘1.txt’ : ‘Сниппет документа 1’,
		‘2.txt’ : ’Сниппет документа 2’,
		‘3.txt’ : ‘Сниппет документа 3’,
		‘4.txt’ : ‘Сниппет документа 4’,
		}
			Если в папке raw_documents есть не текстовые файлы, программа выдаёт ошибку

18	Сообщение “Пустой запрос! Введите ваш вопрос.”	“По вашему запросу ничего не найдено! Попробуйте переформулировать.”	При удалении не кириллических символов в запросе на английском, если запрос состоит не из стоп слов, не выводится сообщение для запроса, он считается пустым

Результаты тестирования

В ходе тестирования были протестированы 4 функции (prepareText, prepRequest, buildFrequencyMatrix, prepDocNames), протестирована связь модулей предобработки запроса и поиска релевантных документов, а также проведено аттестационное тестирование. Были выявлены некоторые ошибки, которые рекомендованы к отладке и исправлению. Также было установлено, что модуль поиска релевантных документов не оптимизирован, из-за чего на некоторых запросах системе требуется больше времени на поиск ответа.