Пермский филиал федерального государственного автономного
образовательного учреждения высшего образования
Национальный исследовательский университет
«Высшая школа экономики»

Факультет экономики, менеджмента и бизнес-информатики

Гуляев Вадим Юрьевич

РАЗРАБОТКА ИССЛЕДОВАТЕЛЬСКОГО ПОРТАЛА ДЛЯ АНАЛИЗА И ОЦЕНКИ СТИЛЯ НАУЧНЫХ ПУБЛИКАЦИЙ

Выпускная квалификационная работа

студента образовательной программы «Программная инженерия»
по направлению подготовки *09.03.04 Программная инженерия*

Рецензент
кандидат физико-математических наук, доцент кафедры математического обеспечения вычислительных систем ПГНИУ
С.И. Чуприна

Руководитель
старший преподаватель кафедры информационных технологий в бизнесе
_____
В.В. Ланин

Пермь, 2018 год

Аннотация

Данная работа посвящена проектированию и разработке исследовательского портала для анализа стиля научных текстов на английском языке на основе методов корпусной лингвистики, в частности метода эталонных корпусов. Портал предназначен для квалифицированных лингвистов, которые хотят анализировать различные крупные корпусы и получать статистику, а также тех, кто желает проверить соответствия тому или иному функциональному стилю.

Работа включает в себя введение, четыре главы, заключение, библиографический список и приложения. Работа состоит из 72 страниц формата А4, включает в себя 35 рисунков и 5 таблиц, приложения занимают 7 страниц формата А4.

Оглавление

Введение

Английский язык является общепризнанным языком мирового научного сообщества, и, чтобы быть членом этого сообщества, необходимо владеть английским языком, и в том числе использовать его при публикации результатов научных исследований. Те, кто имеет опыт в написании статей для международных источников, уже знают, на что обращают внимание зарубежные рецензенты, но те, кто только начинает свой путь на вершину международного научного олимпа, еще не имеют представления о специфике научного стиля статей на английском языке.

Существует множество приемов приведения англоязычного текста к научному общепринятому стилю: структура статьи, клише, лексика и в том числе соответствующая терминология. К сожалению, использование вышеперечисленных средств не гарантирует обеспечение принадлежности текста к научному стилю, так как нет как такового единого стандарта или критерия соответствия статьи научному стилю, ведь данная оценка в большинстве своем субъективна и производится экспертами.

Проблемой в данном случае является определения степени принадлежности статьи к научному стилю. Изучением и решением данной проблемы занимается корпусная лингвистика. Метод решения заключается в следующем:

1. сформировать корпус – набор статей, отобранных по определенному признаку: научная область, географическое происхождение, научный журнал;
2. выделить список признаков – метрик, которые будут служить для количественной оценки того или иного текста;
3. получить усредненное значение по каждому признаку и принять как эталонное;
4. проаннотировать и сравнить произвольный текст с эталоном;
5. в зависимости от отклонения полученных значений от эталонных сформировать вердикт относительно степени принадлежности анализируемого текста к научному стилю.

Таким образом осуществляется переход от экспертной оценки к конкретным количественным показателям, что позволяет определять соответствие статьи научному стилю без привлечения экспертов. Данный подход позволяет сконцентрировать

внимание рецензентов на семантике предоставляя анализ синтаксической составляющей машинной обработке.

Решение сформулированной ранее проблемы заключается в автоматизации описанного выше процесса. Необходимо разработать приложение, которое бы позволяло формировать корпусы статей, обрабатывать их, анализировать какую-либо англоязычную статью на предмет соответствия научному академическому стилю по средствам сравнения показателей выбранных метрик с показателями корпуса и на основе результатов оценки выдавать рекомендации по исправлению текста.

Объектом данного исследования является определение принадлежности статьи на английском языке к научному стилю. Предметом – информационная система для проведения корпусных исследований.

Целью данной работы является разработка веб-портала для проведения корпусных исследований:

1. формирование корпуса;
2. обработка корпуса;
3. анализ статьи на основе корпуса.

Для достижения цели требуется решить следующие задачи:

1. проанализировать предметную область проведения корпусных исследований;
2. проанализировать существующие решения для анализа соответствия статей академическому стилю;
3. разработать и формализовать требования к порталу;
4. выполнить проектирование портала в нотации UML;
5. реализовать портал согласно сформированным требованиям.

В данном исследовании будут применяться методы корпусной лингвистики, ООП. Структурно работа приблизительно соответствует жизненному циклу разработки программных продуктов. Первая глава посвящена анализу предметной области и обзору существующих решений оценки и анализа функционального стиля. Вторая глава посвящена формулированию требований и их формализации. Третья глава посвящена проектированию архитектуры системы и макетированию интерфейса. Четвертая глава содержит описание реализации портала.

Глава 1. Методы анализа в корпусной лингвистике и решения для оценки стиля текста

Экспертное определение функционального стиля текста - это субъективный, кропотливый, трудоемкий и неавтоматизированный процесс, в котором участвует много людей. В то же время определение стиля текста является предметом изучения корпусной лингвистики. Методы корпусной лингвистики, которые основаны на анализе коллекций текстов, предоставляет возможность взглянуть

Для автоматизации процесса анализа стиля текста на основе уже полученных результатов корпусных исследований было разработано программное обеспечение для синтаксического анализа текста. Такое программное обеспечение представлено как в качестве сайтов, на которых можно найти и просмотреть интересующую информацию, так и в качестве подключаемых модулей, интеграция которых может быть использована при реализации портала.

1.1. Академический стиль в английском языке, способы анализа стиля текста

Академический стиль речи – это особый функциональный стиль речи, который используется для представления результатов научной деятельности и доказательства их истинности. Данный стиль используется в научных статьях, учебной литературе и т.д.

На данный момент у людей, которые занимаются проведением исследований, периодически возникает потребность привести свои тексты в соответствие научному стилю с целью получить возможность быть изданным в уважаемых международных журналах. Чтобы получить информацию о научном стиле, можно воспользоваться огромным количеством информационных ресурсов, представленных как в виде печатных справочных и учебных материалов, так и интернет-ресурсов. Для обучения академическому стилю письма существует множество видов учебных пособий на английском языке. Данные пособия могут быть предназначены для студентов, преподавателей и рецензентам.

Интернет-ресурсы так же представлены в разных видах и для разной аудитории. В качестве частного примера можно привести ресурсы, которые содержат списки

конструкций для использования в различных ситуациях. Такие ресурсы называются academic phrasebanks. В качестве примера можно представить Academic Phrasebank University of Manchester [5]. Данные ресурсы могут помочь во время написания выразить в тексте такие элементы как:

1. Критическая оценка.
2. Осторожное высказывание мнения.
3. Классификация.
4. Сравнение и противопоставление.
5. Понятия и определения.
6. Описания тенденций.
7. Описание величин.
8. Описание причинно-следственных связей.
9. Описание ситуаций в прошлом.

Академический стиль английского языка рассматривается и изучается уже довольно давно, было проведено большое количество исследований, среди которых есть также и русскоязычные исследования, история которых тянется с 80-х годов по настоящее время. В данных исследованиях рассматриваются общие структурные и функциональные особенности научных работ, также рассматриваются менее обширные темы, такие как выражение собственной точки зрения, описание результатов и др. [2].

Одной из задач естественного языка является классификация текстов, одним из признаков для классификации является функциональный стиль. Определение функционального стиля текста может использоваться в машинном переводе, информационном поиске и генерации текстов.

Чтобы определить функциональный стиль текста, как правило, используют «маркеры стиля». В известном исследовании описаны четыре типа таких маркеров:

1. токены;
2. синтаксические маркеры;
3. маркеры, демонстрирующие разнообразие речи;
4. частотность.

В работе [3] представлены используемые для решения задач классификации методы машинного обучения, с использованием раннее описанных признаков. Классифицируемые тексты сопоставляются с векторами признаков. Авторы выделили два типа признаков:

1. Лексические.
2. Количественные.

Один из подходов к анализу функционального стиля текста был представлен в исследовании [2]. В данной работе выделены признаки текста, которые отвечают за формальный стиль (субъективные выражения, пассивный залог, вопросы, отсутствие личных местоимений), читаемость (связки, ссылки, замена глаголов на существительные) и научный язык (по общепринятому списку научных слов и т.п.). На основе выделенных признаков создается карта (Self-Organizing Map – особая разновидность нейронной сети).

Еще одним примером использования метода для анализа стиля, является метод, который был использован для анализа характеристик в заголовках статей на тему «Computer Science». На примере корпуса статей из различных научных журналов были изучены значения параметров таких как:

1. Длина заголовка.
2. Пунктуация.
3. Частота слов.

Метод, который будет использован в данной работе при реализации портала, представляет из себя сравнение функционального стиля статей, объединённых экспертами в корпус, с функциональным стилем статей, стиль которых не соответствует стандартам научного стиля. Сравнение функциональных стилей производится по признакам, которые связаны с синтаксическими и лексическими показателями текста, выделенными при анализе литературы связанной с особенностями академического стиля текста. Образцом, с которым планируется сравнение, будет являться набор «эталонных» статей из прорецензированных источников, каждой из которых сопоставляется вектор, демонстрирующий вхождение различных лингвистических характеристик.

Данный метод примечателен тем, что он направлен на практическое применение теоретических сведений, собранных в ходе изучения в данной предметной области. Проведение сравнительного анализа основанных на коллекции текстов, объеденных по языку и функциональному стилю, относит метод к области исследований, связанных с корпусной лингвистикой.

1.2. Корпусная лингвистика, цели и задачи

Исследования в корпусной лингвистике направлены на изучение языка и его аспектов в различных проявлениях через призму текстов. Подобный подход к изучению языка позволяет сравнить идеальную модель языка с его практическим использованием. Корпусную лингвистику от традиционной отличает использование количественных характеристик и методов исследования, что позволяет собирать статистику и строить математические модели.

Корпус – это собранные по определённым критериям тексты в электронном виде. Наибольшая вариативность с точки зрения функциональных стилей наблюдаются в национальных корпусах, собранных по признаку языка написания. Специализированные корпуса формируются с учетом нескольких признаков, к примеру, стиль, жанр, автор, тематика и т.д. Также существуют параллельные корпуса, которые содержат оригинальные тексты и переводы к ним на нескольких языках.

Хранение документов определенного корпуса иногда используется как часть определения понятия «корпус». Помимо этого, немаловажной частью корпуса является метаинформация или «разметка». Разметка – это множество единиц метаинформации, которые относятся к отдельным словам или речевым конструкциям. Одним из основных видов разметки для корпуса считается разметка, сделанная на основе морфологического анализа.

Существуют различные виды разметок текста. Разметка на первом этапе – это токенизация (выделение орфографических слов), лемматизация (конвертация слов в начальную форму) и морфологический анализ. В плане автоматизации порядок может отличаться, так как бывают различный подходы к тем или иным этапам разметки и нанесения метаинформации на разные плоские тексты.

Из лингвистических типов разметки моно выделить следующие [6]:

1. Морфологическая.
2. Синтаксическая.
3. Семантическая.
4. Анафорическая.
5. Просодическая.

Типы 1-3 представляют итог применения методов разметки и присвоения единиц метаинформации (тэгов и аннотаций) на речевые конструкции. Анафорическая разметка текста позволяет определить связь между местоимениями и словами, которые они заменяют. Просодическая разметка используется в корпусах, где необходимо получить информацию о фонетических особенностях слов в тексте.

Проведение операций, связанных с ручной обработкой внушительных временных затрат, поэтому для автоматизации данного процесса разрабатывается программное обеспечение. Особыми успехами с точки зрения автоматизации отличаются морфологическая и синтаксическая разметки. Данная разметка осуществляется такими программами как парсеры и тэггеры – программными средствами, производящими морфологическую и синтаксическую разметку соответственно.

Для проведения исследований и решения собственных задач скорее всего не хватит стандартного набора разметок, необходимо добавлять и использовать собственные. В таком случае необходимо программно описать новые типы разметок и интегрировать в среду для работы с документами.

Подобные задачи решаются с помощью описания лс шаблонов по средствам регулярных выражений. Подобные шаблоны описываются и добавляются к существующему механизму разметки.

Лингвистический корпус - это инструмент, который призван решать задачи корпусной лингвистики, а также может быть использован для формирования глоссариев, машинного перевода, написания и формирования различных обучающих материалов и проверки качества инструментов автоматизированного синтеза и анализа речи. Актуальной является рассматриваемая в рамках данной работы задача проведения анализа функционального стиля текста для оценки качества научных статей на английском языке.

1.3. Анализ и оценка существующих решений

Так как для проведения корпусных исследований необходимы большие наборы данных, то есть много представленных для пользования инструменты для работы с корпусами, от подключаемых библиотек для разработчиков, до полноценных инструментов для использования лингвистами. Необходимо рассмотреть существующие инструментальные средства для решения задач анализа и анализа стиля текста. Экспертами были сформированы требования, предъявляемые к подбираемым решениям:

1. Наличие возможности выбора одного из нескольких корпусов.
2. Предоставление возможности загружать корпусы.
3. Возможность разметки текста.
4. Наличие компонента формирования отчетов на основе собранной статистики.
5. Многопользовательский доступ.

В процессе анализа существующих решений будут учитываться моменты, связанные с процессом анализа в целом. На этом основании будут сформированы нефункциональные требования.

В качестве первого аналога будет рассмотрен сайт Национального корпуса русского языка (НКРЯ) [10]. Данный инструмент сконцентрирован вокруг корпуса на русском языке. Единственная возможность, которую предоставляет сайт – это поиск. Поиск может быть произведен в привычном режиме (поиск, по введенным словам) (см. рис. 1.1), а может быть использован параметризованный поиск для более гибкого построения запроса.

Тексты, который содержатся в НКРЯ, имеют синтаксическую разметку, поэтому поиск имеет разнообразные параметры (см. рис.1.2). Данная возможность позволяет более тонко настраивать поисковый инструмент при использовании данного программного инструмента.

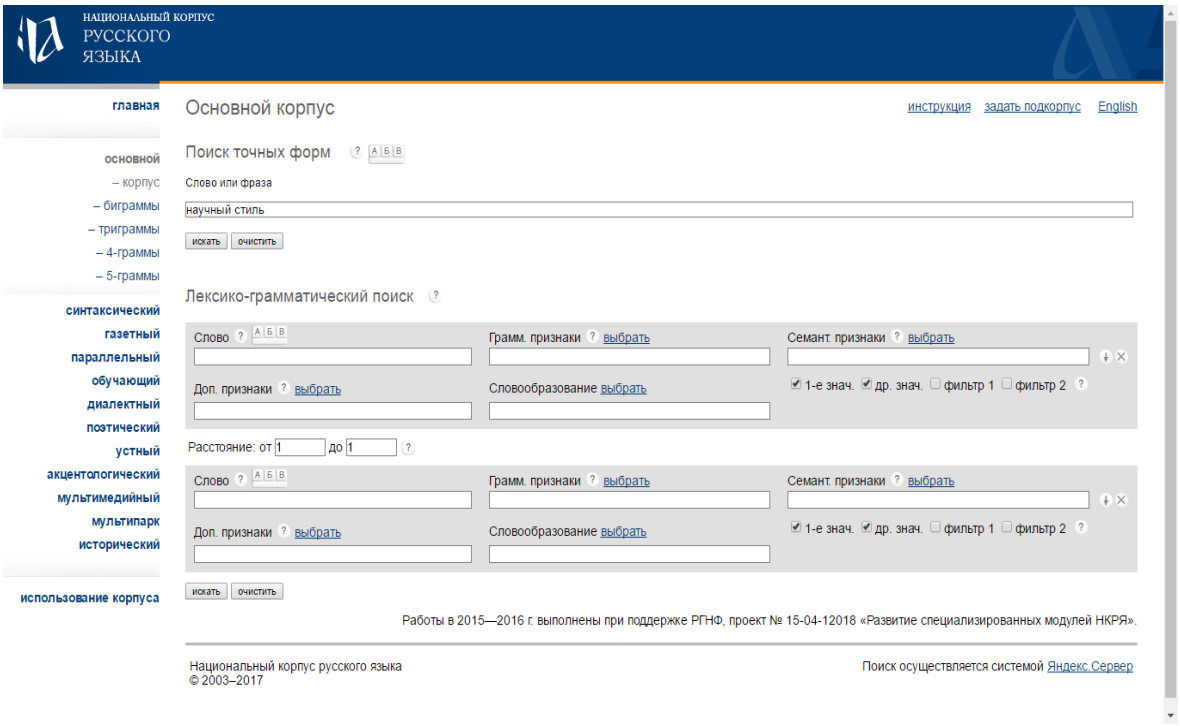

Рисунок 1.1. Поиск по Национальному корпусу русского языка

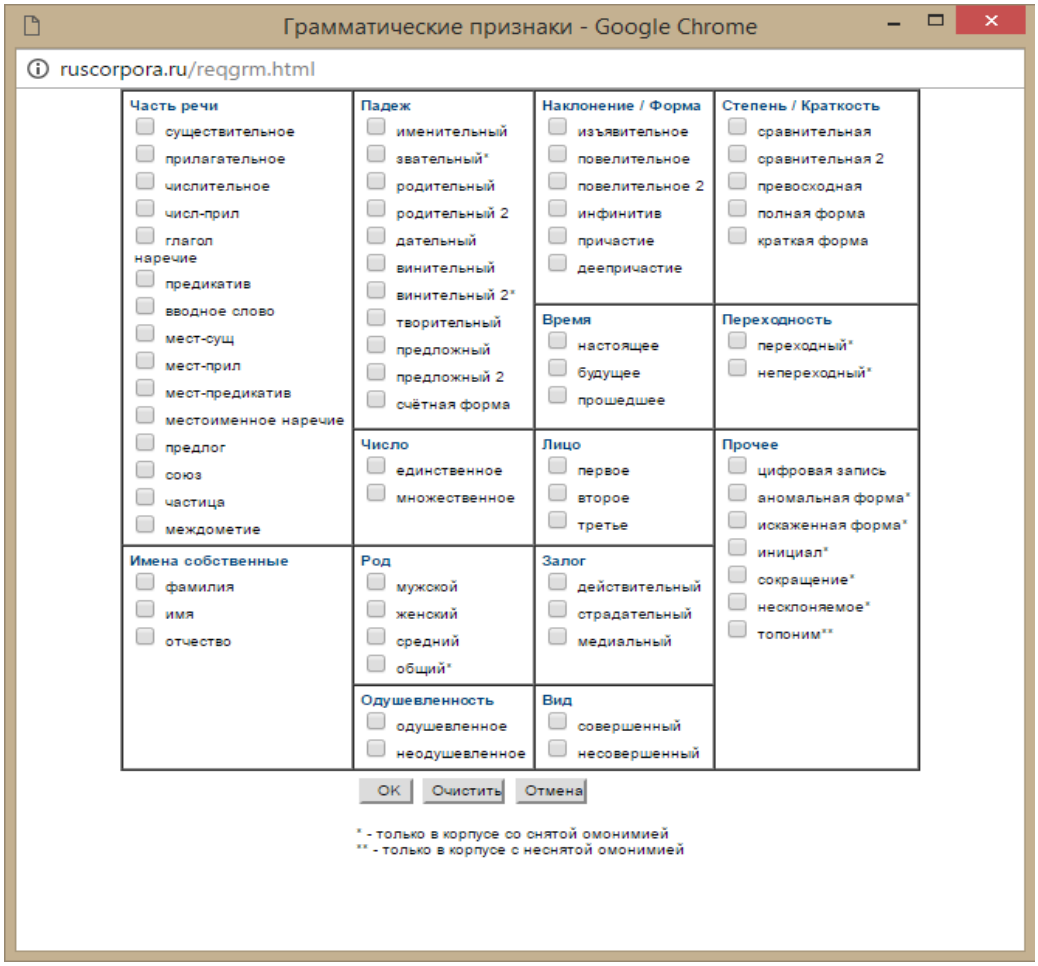

Рисунок 1.2. Параметры поиска по Национальному корпусу русского языка

Одной из главных, а в большей части и единственной функцией программ, связанных с корпусными исследованиями является поиск употребления того или иного слова/словосочетания. Подобного рода программные продукты называются конкордансерами. Такие программы предназначены для людей с низкой квалификацией. Конкорданс – это все найденные употребления искомого выражения. Программы, которые специализируются на обеспечении проведения корпусных исследований предоставляют возможность пользователям создавать подобные конкордансы, иными словами исследовать ситуации и обстоятельства употребления тех или иных выражений в рамках учебных, профессиональных или исследовательских целей.

Программы конкордансеры могут быть использованы с привязкой к какому-то конкретному корпусу. Примером такого типа программы является StringNet используемый для поиска слов и выражений с привязкой к национальному корпусу английского языка (British National Corpus, BNC). Также существуют и программы, позволяющие работать с корпусами текстов без подключения к сети интернет. Есть и программное обеспечение которое может предложить более разнообразный набор функций, например, составление списка слов, имеющихся в корпусе, поиск по заранее составленным регулярным выражениям и списка ключевых слов, проводить аннотирование.

Одним из ярких примеров программ с большим набором функциональности будут рассмотрены программы, которые представлены в виде свободного программного обеспечения, такие как среда разработки для создания приложений для обработки текстов GATE и AntConc. Эти продукты не привязаны к конкретной предметной области и подходят для решения довольно широкого спектра задач в корпусной лингвистике.

Одним из известных разработчиков программного обеспечения для проведения корпусных исследований является Энтони Лоуренс, который в свою очередь является создателем целого семейства программ, в том числе и AntConc [8]. Следует отметить такие продукты как AntWordProfiler – средство для определения принадлежности слов к тому или иному списку и разделению на уровни в зависимости от наличия или

отсутствия слова в списках, а также AntPConc – инструмент для анализа параллельных корпусов.

Продукт AntConc включает в себя ряд инструментов [8]:

1. Осуществление поиска конкордансов в виде карты, так называемого штрих-кода для выявления расположений, где находятся выражения, которые ищет пользователь (см. рис. 1.3).

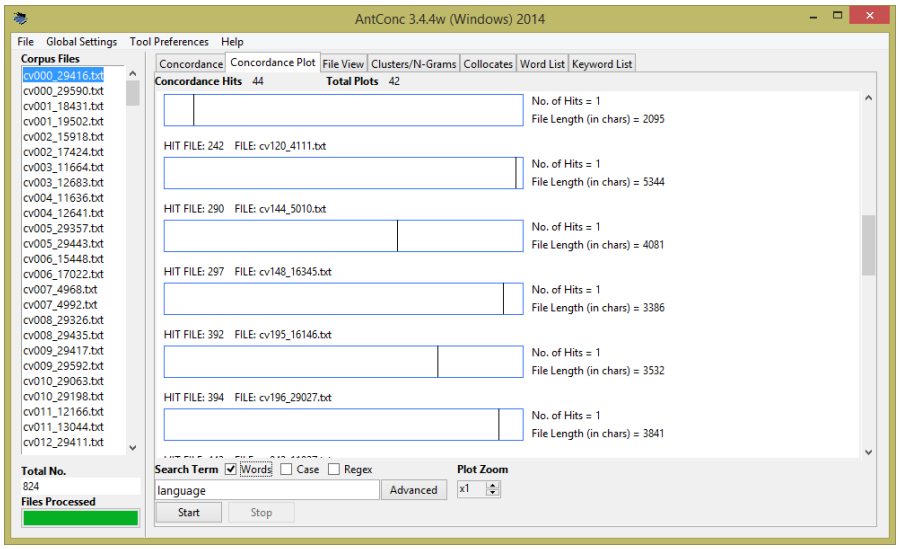

Рисунок 1.3. Осуществление поиска конкордансов в виде карты

2. Осуществление поиска конкордансов в виде ключевого слова вместе с контекстом, в котором оно употреблено (см. рис. 1.4).

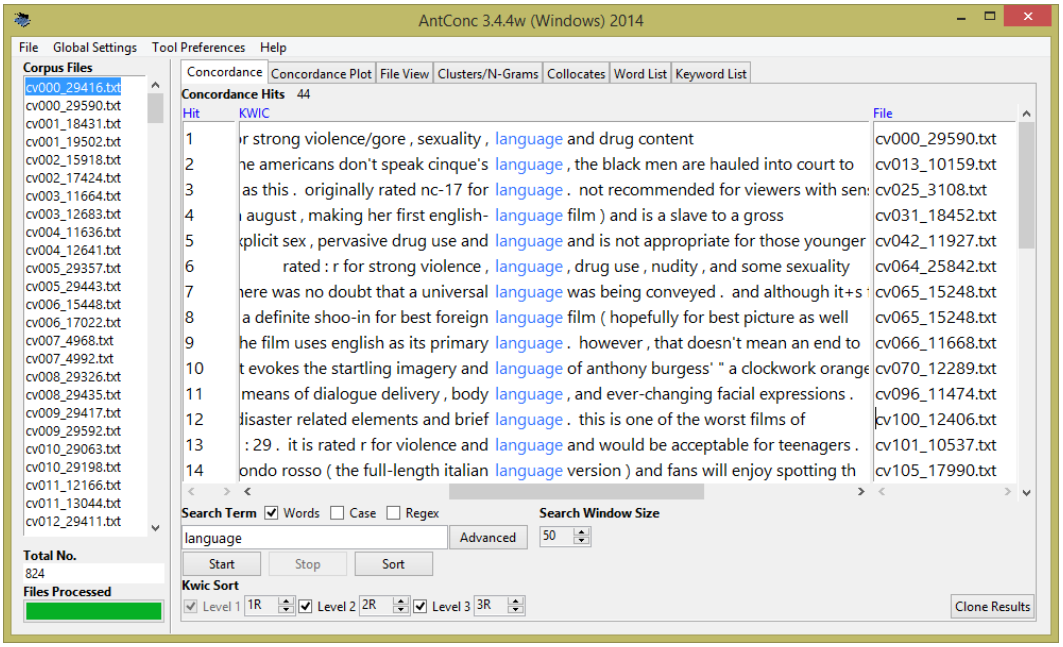

Рисунок 1.4. Осуществление поиска конкордансов в виде ключевое слово и контекст

3. Нахождение и определение устойчивых выражений, в котором находится искомое словоупотребление.
4. Формирование списка из уникальных слов с приведением статистики по количеству вхождений.
5. Формирование списка, состоящего из ключевых слов, полученных из пользовательского корпуса.
6. Возможность открывать и просматривать содержимое пользовательских файлов.
7. Выявление кластеров, демонстрирующих наиболее часто встречающиеся контексты, в которых употребляется пользовательское словоупотребление (см. рис. 1.5).

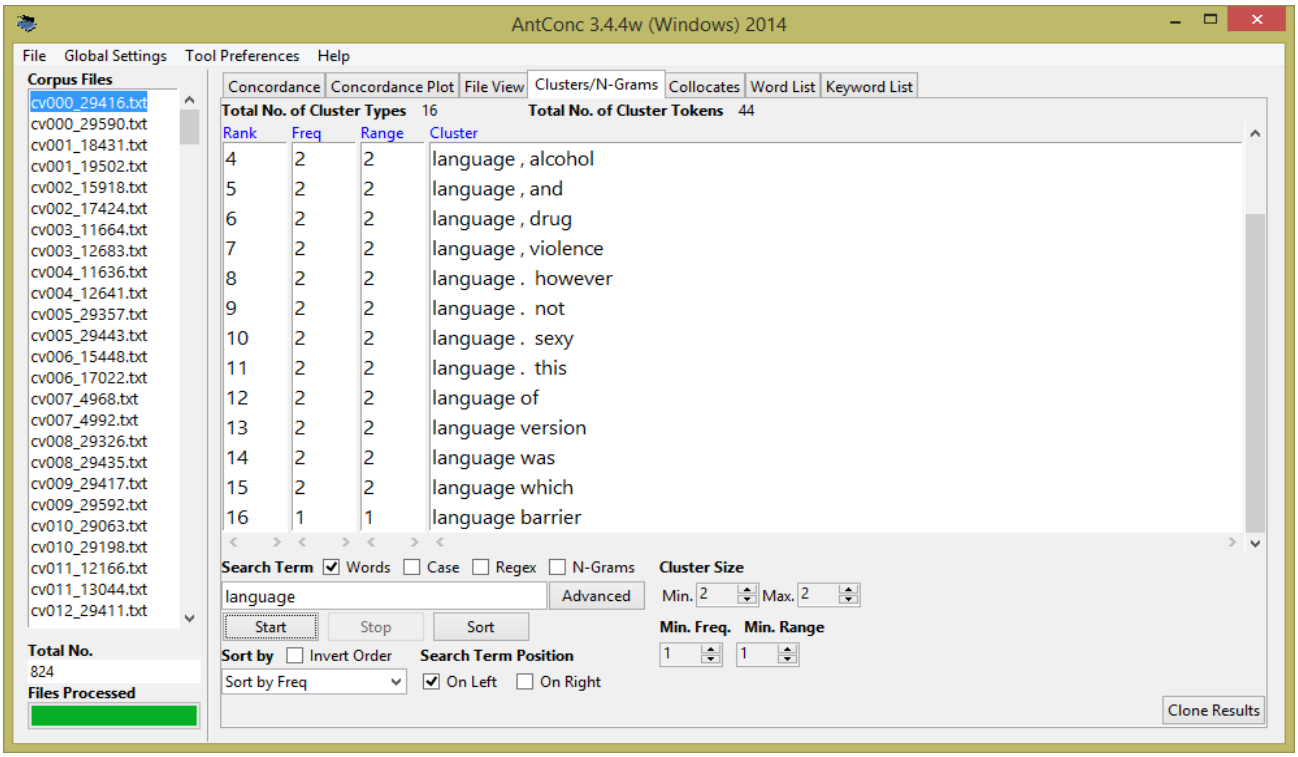

Рисунок 1.5. Выделение кластеров в AntConc

AntConc является десктопной, однопользовательской системой, позволяющей проводить пользовательскую кастомизацию. К сожалению, при всем удобстве использования данной системы для решения популярных задач корпусной лингвистики: поиск конкордансов, формирования списка ключевых и уникальных слов, данный инструмент не дает возможности подключать и использовать собственные средства для обработки текста.

GATE (General Architecture for Text Engineering) Developer [7], как и AntConc - это представитель разновидности систем для обработки корпусов. Основная разница между GATE Developer и AntConc - это целевое использование. Если AntConc - это конечный продукт для решения прикладных задач, то GATE Developer нацелен на разработку и использование собственных инструментов для аннотирования и создания мета информации о тексте/корпусе. GATE Developer – это IDE (Integration Development Environment) для разработки модулей для обработки корпусов и отдельных текстов, которая в свою очередь предоставляет возможность использовать готовый набор плагинов. Также можно воспользоваться инструментом GATE Embedded, который позволяет создавать приложения с использованием уже готовых или вновь созданных компонентов и плагинов GATE Developer, инструментов GATE Teamware для совместной работы над корпусом и GATE Cloud для произведения вычислений в облаке.

Каждое приложение, разработанное в GATE Developer – это конвейер по подготовке и обработке текста, в котором друг за другом к тексту применяются различные ресурсы. Вычислительные ресурсы для обработки текстом при создании базируются как на уже доступных в GATE Developer плагинах и компонентах, так и на разработанных в программистском сообществе GATE компоненты на языке программирования Java, что для .NET среды не очень подходит. Возможности GATE по созданию конвейера продемонстрированы далее (см. рис. 1.6.).

Набор заготовленных компонентов и плагинов GATE Developer прекрасно подходит для разметки текста методом аннотирования – присвоения метаинформации какой-либо части текста. Можно производить токенизацию – выделение обособленных элементов текста, производить морфологический и синтаксический анализ путем определения и выделения различных частей речи и синтаксических конструкций. Возможности визуализации и редактирования аннотации представлены далее (см. рис. 1.7). У пользователя есть возможность редактировать и создавать собственную аннотацию. У каждой метки свой цвет, что позволяет выделять в самом документе каждый тип отдельно.

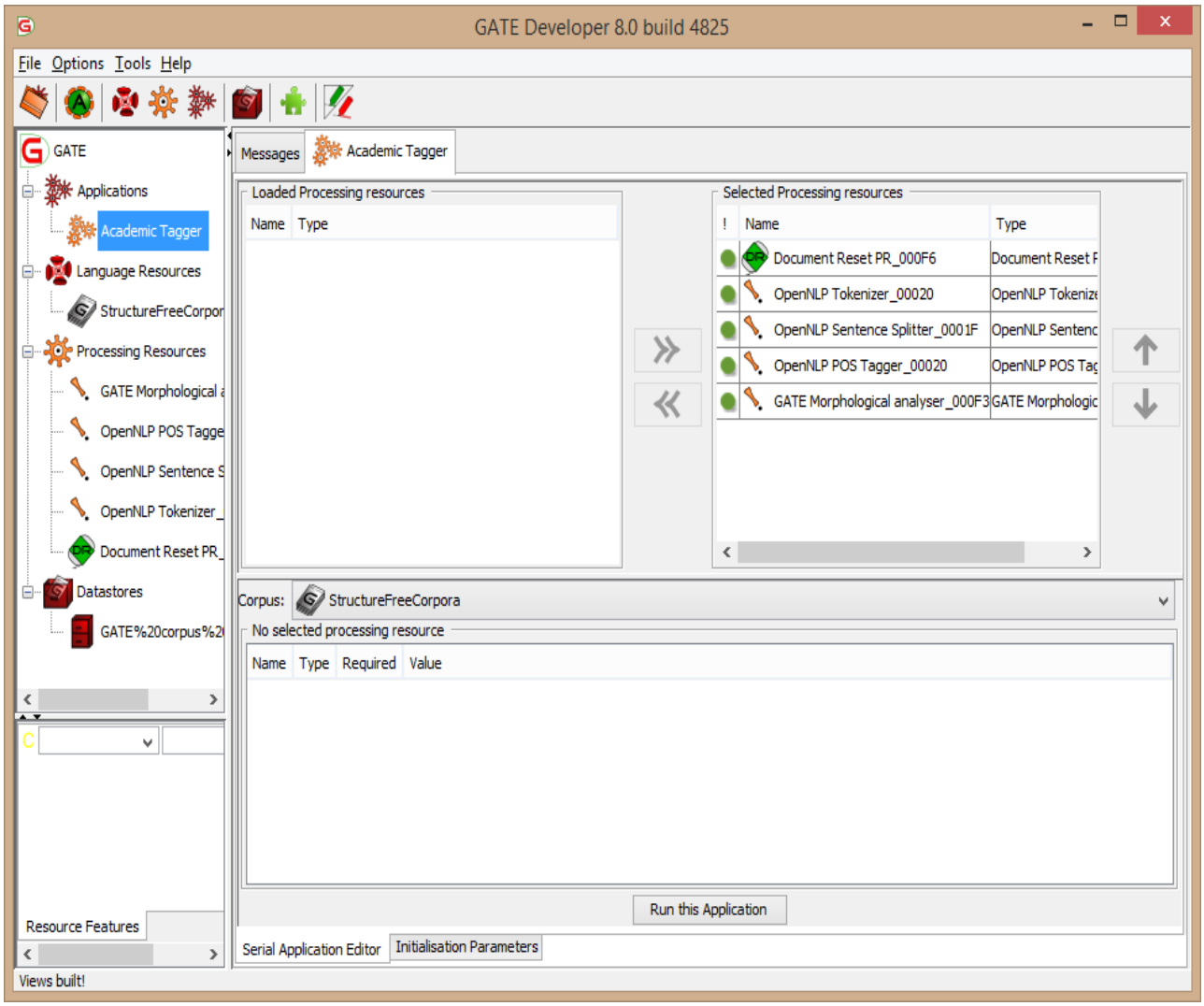

Рисунок 1.6. Графический интерфейс для создания Pipeline в GATE Developer

Разнообразие функциональных возможностей и возможности универсального подхода к решению задач по обработке текста приводит к увеличению сложности взаимодействия незнакомого с GATE пользователя с графическим интерфейсом. Элементарные задачи, например, создание собственной разметки и ее дальнейшее сохранение вызывает вопросы. Можно говорить, что данная система предназначена для профессиональных лингвистов, которые способны разрабатывать собственные компоненты, следовательно, имеющие навыки программирования на языке Java.

GATE – это не только продукт готовый к использованию, но и набор различных библиотек и компонентов. Компоненты GATE могут быть легко переиспользованы и интегрированы в разрабатываемые приложения по анализу и обработке какого-либо набора текстов.

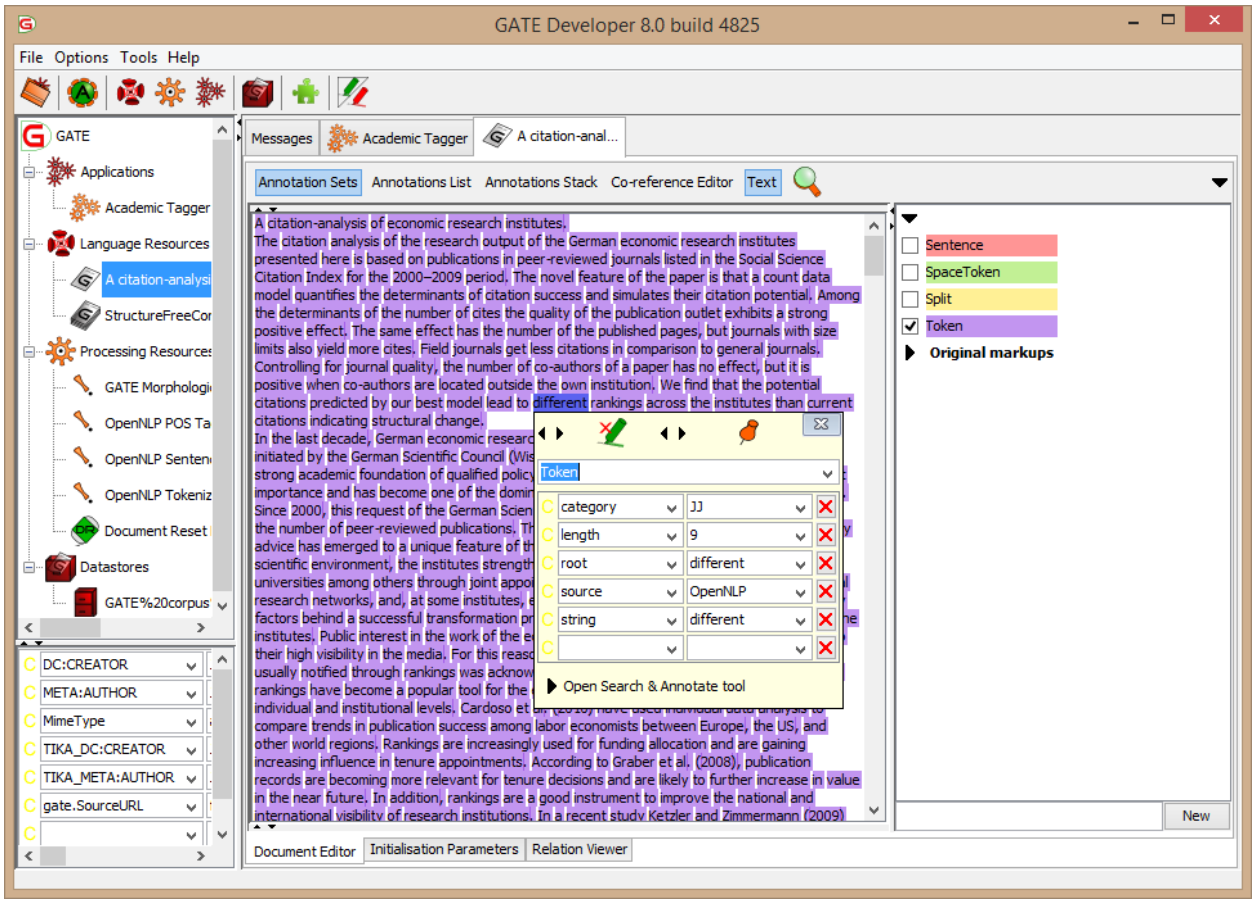

Рисунок 1.7. Работа с аннотациями

После рассмотрения существующих решений инструментальных средств для разметки и анализа текстов, предоставляется возможность подвести небольшие итоги:

1. Программные продукты сфокусированы на разной степени автоматизации процесса обработки текста. В одних требуется ручное воздействие для поиска и выделения интересующей информации, другие представляет собой конвейер, который на вход получает плоский текст, а на выходе выдает текст, дополненный различной метаинформацией.
2. Инструментальные средства предоставляются как на безвозмездной основе в качестве свободного ПО, так и как коммерческие продукты.
3. Также средства имеют различную функциональность: аннотирование, поиск конкордансов, морфологический, синтаксический и лексический анализ.
4. Предоставляемые средства по своей сущности образуют широкий диапазон в качестве используемых средств – от подключаемых библиотек до сред разработки.

Перед тем как остановится на каком-либо инструменте для работы с корпусами текстов, необходимо понимать, какие задачи будут решаться при помощи выбранного средства, количество членов группы, какой материальный фонд, квалификацию членов группы. Результаты соответствия проанализированных аналогов и предъявляемых функциональных требований представлены ниже (см. табл. 1.1).

Таблица 1.1. Соответствие программных продуктов заявленным функциональным требованиям

Требование	**НКРЯ**	**AntConc**	**GATE**
Возможность использовать и хранить больше одного корпуса	-	+	+
Загрузка в систему пользовательского корпуса	-	+	+
Автоматическое нанесение разметки	-	-	+
Сбор статистики и создание отчётов или возможность разработки соответствующего расширения.	-	-	-
Многопользовательский режим	+	-	-

По результатам сравнения GATE является наиболее подходящей системой для реализации метода аннотирования, описанного ранее. К сожалению, в виду отсутствия бесплатной возможности многопользовательского режима работы над корпусом и сложного, перенасыщенного интерфейса GATE не может быть использована как решение поставленных задач.

Благодаря рассмотрению существующих аналогов и обзора с точки зрения предъявляемых требований, можно перейти к формализации требований к разработке собственного продукта для проведения работы с корпусами.

1.4. Выводы по главе 1

Для получения информации о научном стиле для дальнейшего изучения, следует воспользоваться информационными ресурсами, представленными как в виде печатных справочных и учебных материалов, так и интернет-ресурсов. Для обучения академическому стилю письма существует множество видов учебных пособий на английском языке. Данные пособия могут быть предназначены для студентов, преподавателей и рецензентам. Корпусная лингвистика предоставляет набор методов для рассмотрения практического применения приемов приведения текста к академическому стилю основываясь на специфически подобранных корпусов.

После проведенного анализа аналогов следует отметить, что не удалось найти инструментальных средств, которые могли бы подойти для реализации метода проверки стиля текста. Исходя из данных, приведенных в таблице 1.1, самой подходящей является GATE, но очевидными недостатками являются отсутствие многопользовательского режима и сложность взаимодействия на уровне графического интерфейса. Необходимо разработать простой и удобный инструмент с многопользовательским режимом для пользователей любого уровня квалификации.

Глава 2. Формулировка и формализация требований к разработке портала

По результатам рассмотрения аналогов и определения первичных требований, определяемых методом оценки и анализа научных текстов на английском языке на основе подобранных экспертами эталонных корпусов можно сказать, что адекватной реализацией будет исследовательский портал. Исследовательский портал – это интернет-ресурс, который размещен в сети для предоставления информации о какой-либо предметной области, исследованиях в этой области и результатах проведенных исследований. Существуют порталы как внутренние так общедоступные, но так или иначе они обеспечивают многопользовательский доступ к материалам и предоставляют возможность командной работы в группе.

Для получения качественного и измеримого результата реализации портала важно сформировать формализованные функциональные и нефункциональные требования. Функциональные требования будут сформированы исходя из особенностей реализуемого метода. Нефункциональные требования будут выделены исходя из пожеланий и пользовательских сценариев фокус группы.

2.1. Метод эталонных корпусов

Метод оценки стиля текста на английском языке, который рассматривается в данной работе, основывается на сравнении с эталонным корпусом статей, авторами которых являются носители языка и которые прошли экспертный отбор по ряду критериев, характеризующих академический письменный стиль. Критерии отбора далее будут употребляться в качестве метрик или маркеров стиля.

Для того чтобы сформировать набор лингвистических критериев для оценки академического стиля английского языка и читаемости, необходимо оценить и исследовать важность различных характеристик, выделяемых в корпусной лингвистике. Данное исследование было проведено путем экспертной оценки частотности тех или иных маркеров в эталонных корпусах, работах низкого и высокого качества.

По результатам исследования экспертами были предоставлены выявленные в ходе работы список маркеров, определяющих текст как академический. При составлении

списка эксперты основывались на разного рода источниках: учебно-методические пособия, интернет-ресурсами, которые специализируются на академическом стиле текстов на английском языке. Так же в ходе обработки источников на результат повлияла гипотеза, что исследования, приводимые в русскоязычных диссертациях по большей части, не приводят доказательных и наглядных данных.

Из сформированного экспертами списка маркеров, можно составить и описать определенную иерархию, которая наглядно продемонстрирует взаимосвязь и важность выделенных критериев. Для того, чтобы сформировать такую иерархию для начала нужно список маркеров стиля, сформированный экспертами, разделить на три группы:

1. Синтаксические маркеры.
2. Лексические маркеры.
3. Грамматические маркеры.

Далее необходимо декомпозировать выделенные крупные группы на подгруппы. Декомпозиция может быть произведена внутри каждой конкретной группы, например, по способу их валидации. Следует рассмотреть маркеры, вошедшие в каждую группы отдельно.

Синтаксические маркеры можно разделить на две подгруппы:

1. Маркеры, которые описываются структурами.
2. Маркеры, которые учитывают частоту вхождения различных союзов, предлогов, средств связи и др.

В первую подгруппу относят следующие маркеры:

1. Наличие значительного количества предложений с сложноподчинённой/ сложносочинённой/простой структурой.
2. Присутствие при большом количестве существительных постпозитивных и препозитивных определений.
3. Наличие значительного количества в текстах с техническим уклоном препозитивных определительных групп.

В английском языке препозитивное определение, как правило, выделяется следующими частями речи [4]:

1. существительным;

2. прилагательным;
3. числительным;
4. герундием без предлога;
5. причастием;
6. местоимением.

Что касается постопозитивного определения, то оно представляет из себя [4]:

1. существительное;
2. количественное числительное;
3. герундий с предлогом;
4. прилагательное с зависимыми словами;
5. наречие.

Ко второй подгруппе можно отнести следующие характеристики:

1. наличие составных предлогов;
2. использование средств логических связок;
3. наличие составных и двойных союзов.

В группе лексических маркеров можно выделить три подгруппы, которые можно назвать как:

1. Маркеры, учитывающие частота вхождения терминов.
2. Маркеры, учитывающие частоту вхождения словообразований.
3. Маркеры, учитывающие частоту вхождения частей речи.

В первую подгруппу можно отнести следующие маркеры, которые представляют список из пяти пунктов:

1. Почти полное отсутствие личных местоимений.
2. Абстрактные глаголы.
3. Обилие терминов исследуемой предметной области.
4. Усилительные наречия.
5. Словосочетания.

Во вторую подгруппу входят следующие характеристики:

1. Использование абстрактных существительных.

2. Использования суффикса «or», обозначающего различные технические термины.

Следующие маркеры входят в третью подгруппу:

1. Высокая частотность вхождения существительных.
2. Низкая частотность вхождения личных местоимений.

В последнюю группу грамматических критериев входят две характеристики:

1. Пассивный залог.
2. Глаголы в настоящем времени.

В данном случае существует предположение, что, проведя корпусный анализ статей из определенной предметной области, можно получить вектор, где каждый маркер — это отдельное измерение. Портал, разрабатываемый в рамках данной работы, должен позволять загружать корпусы по сходной тематике и высчитывать показатели этого вектора.

После проведения аннотирования необходимо дать возможность просмотреть получившийся результат и самостоятельно выделять или снимать выделение заранее подготовленных маркеров. После проведения анализа аннотированного корпуса, необходимо получить агрегированную статистику по таким параметрам как:

1. Средний показатель частоты вхождения того или иного маркера по всему корпусу.
2. Граничные значения той или иной характеристики.
3. Распределение показателей по каждому маркеру в разрезе отдельных документов.
4. Процентные показатели характеристик, которые Ориентированы на оценку частоты вхождения.
5. Отклонения показателей маркеров от средних в разрезе в разрезе отдельных документов.

При подсчете статистики необходимо принимать во внимание, что статьи в корпусе будут разного размера, в таком случае нужно статистические показатели отражать в абсолютных показателях. Данные по корпусу будут использоваться лингвистом для интерпретирования и их валидации.

Создание рекомендаций для отдельной статьи по схожей с корпусом тематике будет осуществляться путем ее аннотирования и получения статистических данных по этой статье. Полученные значения представляются в виде вектора и после нахождения расстояния между вектором средних значений по корпусу будут сформированы рекомендации по корректировке относительно полученных результатов вычисления расстояния.

2.2. Формирование нефункциональных требований

Проведение анализа и последующей оценки стиля англоязычного текста методом, описанным в статье [10] это лишь одна из важнейших характеристик реализуемого портала. В ходе проектирования и подбора инструментов для реализации портала необходимо понимать потребности конечного пользователя и проблемы, которые нужно решить. В рамках данной работы предполагается, что в конечном итоге будет две категории пользователей:

1. Лингвисты, проводящие корпусные исследования.
2. Пользователи, которые хотят получить рекомендации по написанию их статьи относительно имеющихся в доступе корпусов.

Учитывая особенности реализуемого в рамках данной работы метода и результаты анализа известных инструментальных средств для корпусных исследований, которые были представлены в главе 1, можно выделить следующие требования:

1. Портал должен быть размещен в сети Интернет. Совместный доступ к данным повышает эффективность работы и сокращает необходимость в лишней коммуникации между членами команды. Доступ с любого устройства, у которого есть выход в Интернет.
2. Управление правами доступа. У пользователя должна быть возможность ограничивать доступ к публикуемым материалам.
3. Дружественный интерфейс. Необходимо создать удобный инструмент, чтобы при его использовании не возникало вопросов как у профессиональных лингвистов, так и у незнакомых с данной предметной областью пользователей.

4. Публикация результатов проведённых исследований. У пользователя должна быть возможность публиковать результаты проделанной им работы в открытом доступе.
5. Наличие описания портала, его возможностей и метода анализа. Необходимо предоставить пользователю возможность перед началом работы познакомиться с порталом, его технической и теоретической частью.

При соблюдении вышеописанных требований есть возможность произвести простое и понятное средство для анализа стиля текста на английском языке. Данный инструмент подойдет как профессиональным лингвистам, так и пользователям, которые хотят получить рекомендации по написанным статьям.

2.3. Основные термины предметной области

Для реализации той или иной системы необходимо концептуальное представление в терминах предметной области, в данном случае предметной области оценки и анализа стиля текста на английском языке на основе корпусов. Описание терминов и понятий данной предметной области приведено в приложении (см. Приложение D).

В концептуальном представлении данной предметной области пользователь занимает центральное место, так как инициация того или иного объекта происходит по его инициативе, а также ограничение доступа к этим объектам. У пользователя есть связь с такими сущностями как:

1. Отчет-анализ.
2. Отчет-сравнение.
3. Корпус.
4. Документ.

Для целостности будет определен идентификатор пользователя, зарезервированный под значение «общий», следовательно, если не указан конкретный круг лиц, которым будет доступный тот или иной материал, то будет установлен данный идентификатор для дальнейшего показа всем пользователям. Также можно установить этот идентификатор вдобавок к указанным пользователям.

Каждый отчет-анализ должен быть ассоциирован с конкретным корпусом и типом отчета, от которого зависит специфика статистики. Разметка будет ассоциирована с конкретным документом, включающая аннотации и их атрибуты, а также с пользователем, которому принадлежат данные.

Рекомендация будет проассоциирована с конкретным документом и корпусом, так как в зависимости от выбранного эталонного корпуса будут разнится и рекомендации. Ассоциация будет осуществлена с помощью реляционного хранилища. Учетные данные по каждому объекту на портале будут однозначно определять привязку нереляционных данных к тому или иному объекту.

2.4. Формирование функциональных требований

Далее будут представлены функциональные требования в UML нотации в виде диаграммы прецедентов. Требования разработаны с учетом описания особенностей реализуемого метода, требований, сформированных в параграфе 2.2, а также концептуальных понятий перечисленных и описанных в параграфе 2.3. Варианты использования для удобства разделены на три диаграммы:

1. Общая диаграмма. Данная диаграмма представляет из себя описание взаимодействия пользователя на самом верхнем уровне.
2. Диаграмма прецедентов для анализа на основе корпуса. Данная диаграмма описывает варианты взаимодействия пользователя с системой в сценариях разметки корпуса и его дальнейшего анализа.
3. Диаграмма прецедентов для получения рекомендаций. Данная диаграмма описывает возможные действия пользователя в сценариях получения рекомендаций по какой-либо статье.

Общая диаграмма прецедентов представлена далее (см. рис. 2.1.). Эта диаграмма отражает действия пользователя, не только зарегистрированного в системе, но и нового пользователя. Представлены возможности авторизации и регистрации для новых пользователей, а также крупные блоки представляющие из себя основные функциональные блоки портала.

Крупные блоки будут раскрыты в диаграмме прецедентов для анализа (см. рис. 2.2.) и диаграмме прецедентов для получения рекомендации (см. рис. 2.3.).

Подробное текстовое описание прецедентов описанных на диаграммах прецедентов представлено в приложении (см. Приложение В).

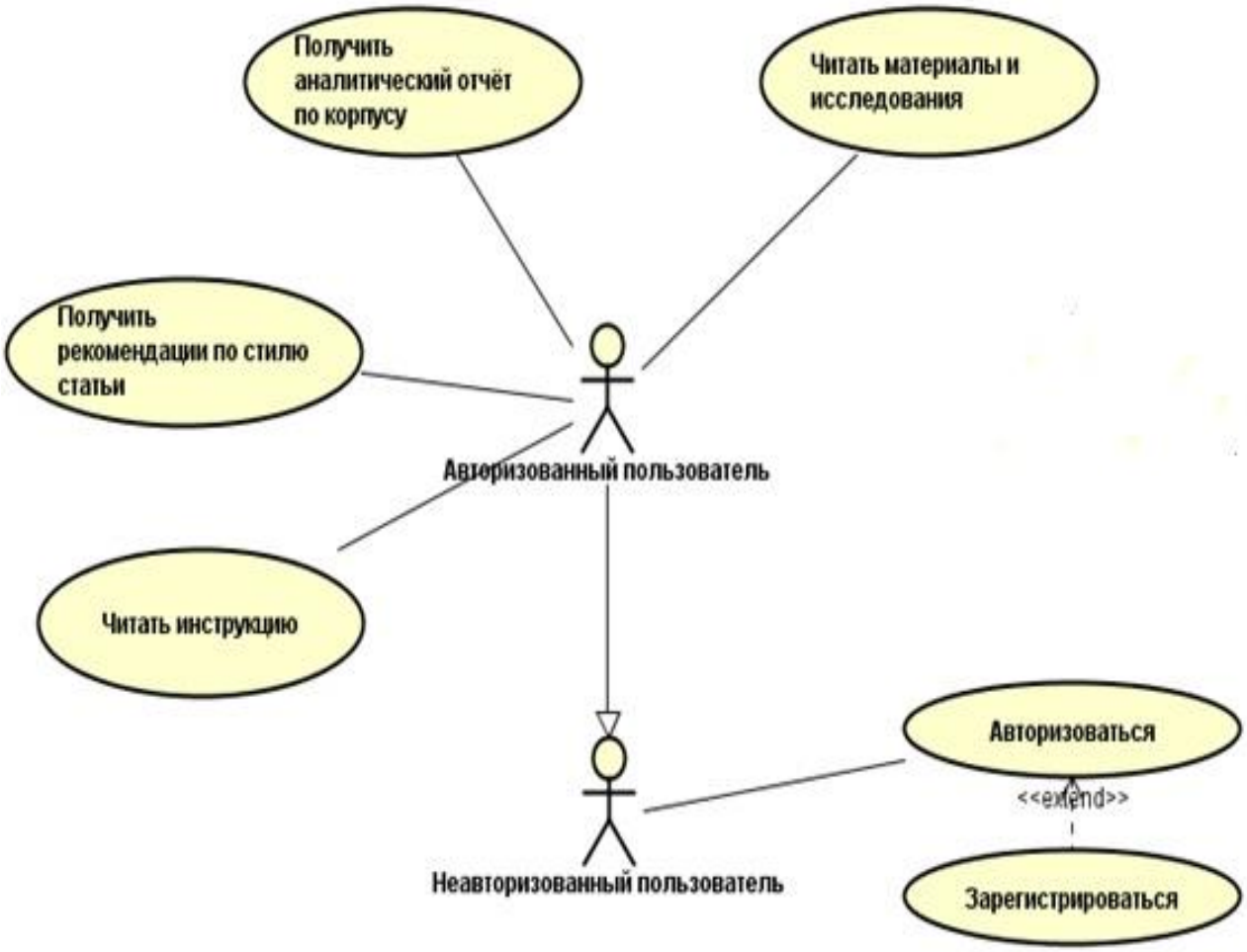

Рисунок 2.1. Общая диаграмма прецедентов

Необходимость получения статистических данных в виде какого-либо отчета продиктовано особенностями предметной области. Предполагаемый сценарий такой, что пользователь начал изучать данную тему и хочет посмотреть статистику по уже загруженным корпусам в общем доступе, после этого решает загрузить свой собственный корпус и так же проанализировать его и получить отчет. Чтобы достигнуть поставленных целей, пользователю необходимо выбрать один из ранее загруженных корпусов и перейти к его описанию, где выбрать просмотр статистики по корпусу. В случае с пользовательским корпусом, необходимо сначала загрузить корпус, затем дождаться пока процесс аннотации будет завершен и представится возможность просмотреть отчет по корпусу.

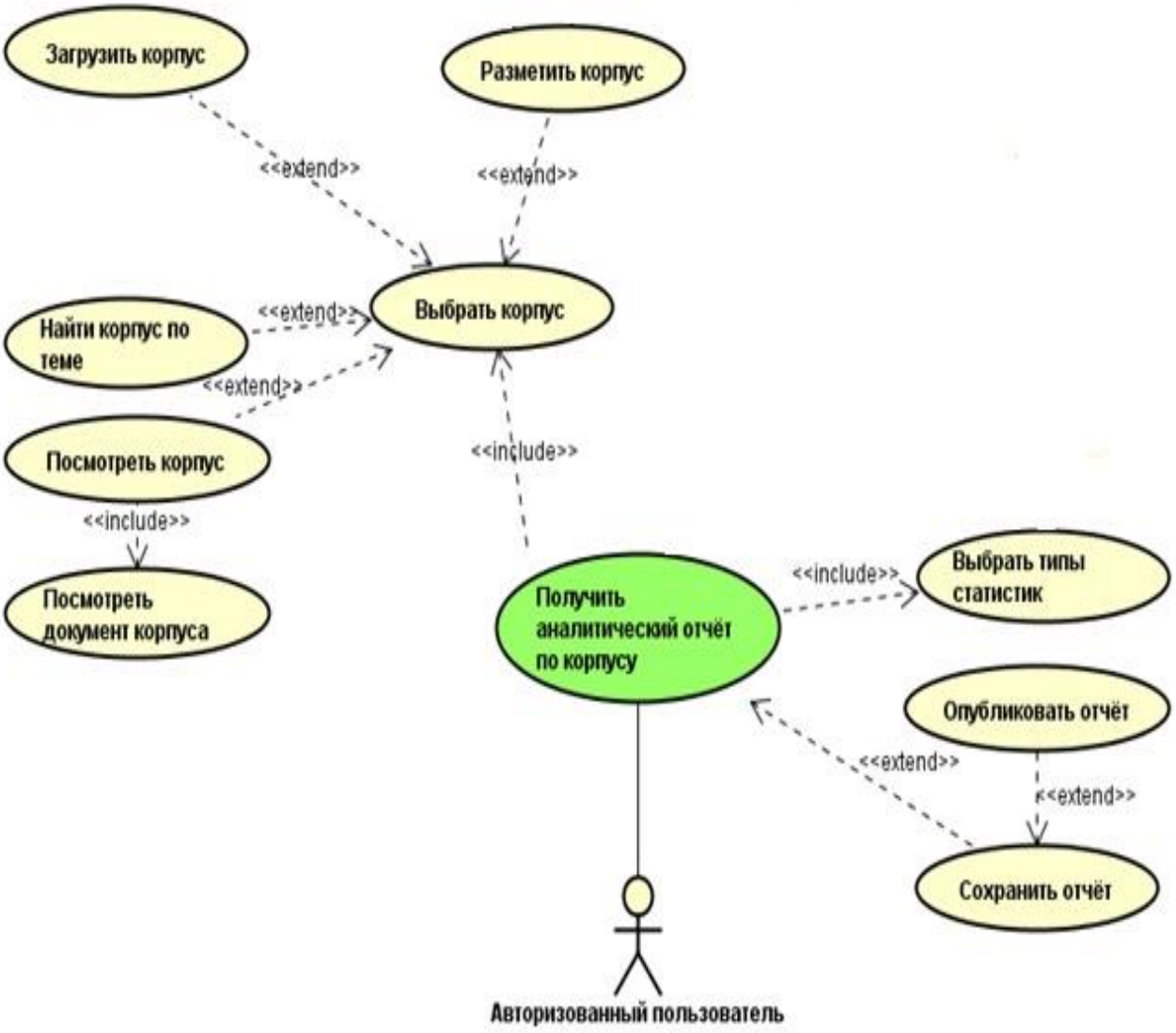

Рисунок 2.2. Диаграмма прецедентов портала для анализа корпуса

Для проведения анализа вновь созданного корпуса, необходимо выбрать размеченный корпус в открытом доступе, либо личный и просмотреть аналитический отчет. Если корпус ранее не был загружен, то необходимо запустить процесс разметки корпуса, дождаться завершения выполнения разметки и сформировать отчет. Данное действие легко выполнить, воспользовавшись заранее заготовленным шаблоном.

Автоматический процесс разметки предполагает использование какого-либо ЛС шаблона. В рамках данной работы будет использован заранее заготовленный ЛС шаблон.

После завершения процесса разметки, можно сформировать и просмотреть отчет в разрезе различных маркеров и характеристик корпуса. Сформированный по

результатам разметки отчет можно открыть, затем сохранить и в последствии опубликовать как приватно, то есть только для пользователя, загрузившего отчет, так и в пространстве для всех пользователей.

Последняя из диаграмм прецедентов представляет возможности пользователя при сценарии получения рекомендации по стилю загруженного на портал отдельного конкретного документа. Диаграмма представлена далее (см. рис. 2.3).

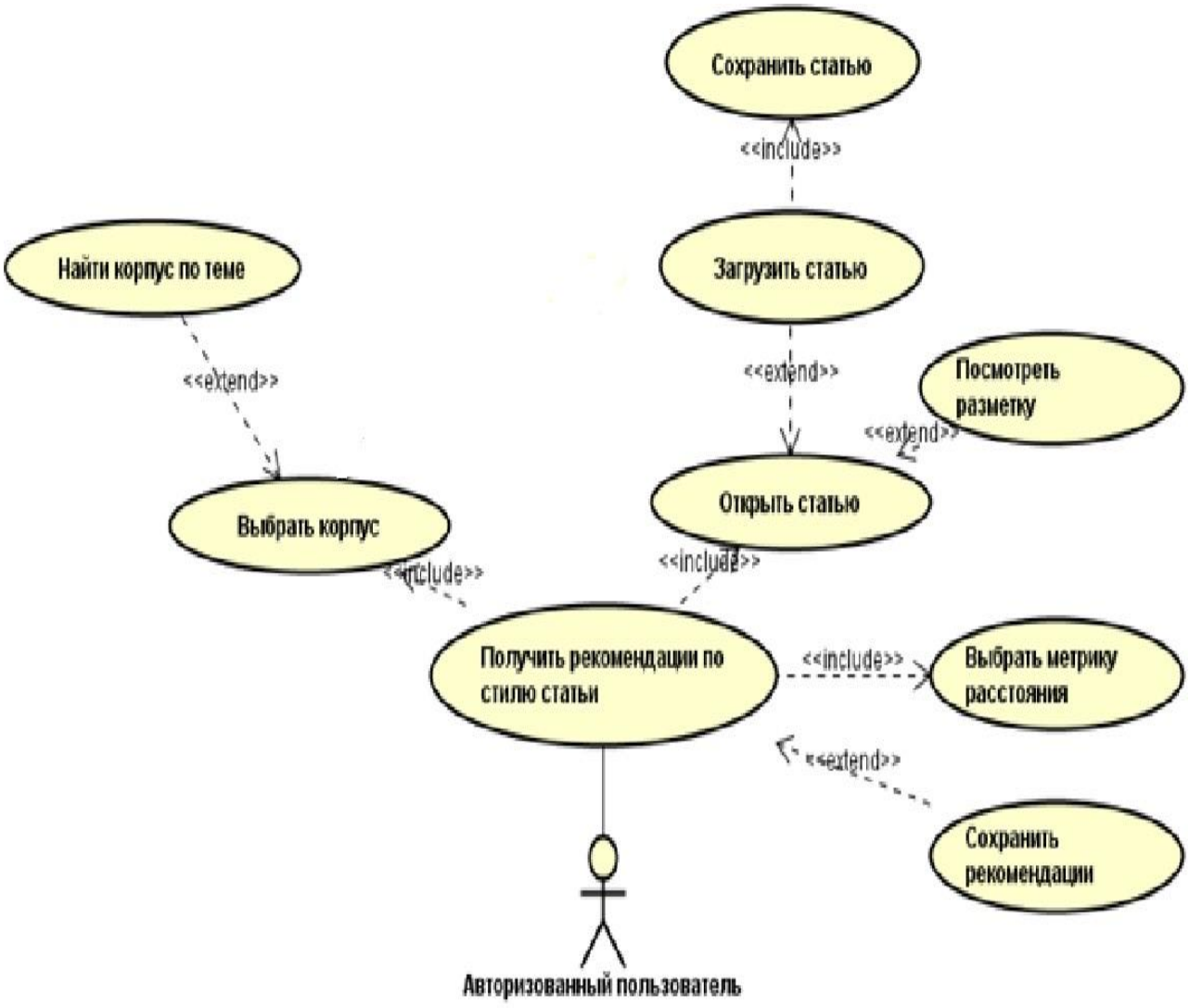

Рисунок 2.3. Диаграмма прецедентов для сценария получения рекомендаций

Чтобы получить рекомендацию, пользователю необходимо:

1. Загрузить статью.
2. Проаннотировать.
3. Получить рекомендацию.
4. Сохранить результат.

Так же можно сохранять и публиковать результаты прохождения оценки своих статей на портале как приватно, то есть только для пользователя, загрузившего отчет, так и в пространстве для всех пользователей.

2.5. Оценка средств реализации приложения

Принимая во внимание требования к реализации системы, которые были определены в ходе анализа аналогов и в ходе формирования функциональных и нефункциональных требований, разрабатываемый инструмент должен представлять из себя веб-портал, позволяющий по средствам браузера, в самом благоприятном сценарии с любого устройства. Решение о разработке было принято, исходя из данных полученных ранее, которые позволяют заключить, что не существует бесплатных инструментов, которые бы могли покрыть все потребности, описанные ранее. Рассмотренные решения не могут удовлетворить требования к удобству интерфейса, повсеместного доступа к порталу и решению задачи определения стиля текста описанным ранее методом эталонных корпусов.

Стоит отметить, что рассмотренные инструменты могут быть использованы в качестве вспомогательных элементов системы. Для минимизации трудовых затрат для решения стандартных и универсальных задач синтаксического и морфологического разбора, токенизации, выделения отдельных предложений следует воспользоваться уже существующими решениями. Например, плагины, компоненты и библиотеки GATE Developer, а также ранее созданные пользовательские компоненты для среды GATE.

Определить стек технологий для реализации веб-портала довольно трудно из-за обилия существующих технологий, но в рамках учебной программы много времени было посвящено .NET Framework и Java. Применимость данных технологий была исследована в работе [1]. Стоит сказать, что данные технологии имеют свои преимущества и недостатки и в данном случае «правильного» выбора нет, но более подходящей платформой был назван .NET Framework, но стоит сказать, что интегрировать компоненты GATE, которые написаны на Java напрямую на данный момент довольно проблематично. Данная проблема может быть решена созданием HTTP сервиса на Java, который предоставит интерфейс для обращения к располагающимся внутри компонентам GATE. Это накладывает определённые

издержки, так как HTTP ограничивает взаимодействие и следует применить некоторые усилия для использования данного канала коммуникации между приложениями.

Жизненно важной особенностью в рамках данной работы является необходимость хранения довольно большого объема данных. Предполагаемый размер обуславливается тем, что в корпусы могут быть различного размера в плане количества документов, а также от тематики, так как тема зачастую коррелирует с объемом статей. Для решения подобных задач подходят облачных технологии. В работе [1] помимо сравнения технологий для реализации веб-портала было проведено сравнение облачных провайдеров и платформ. Сравнение проводилось по следующим характеристикам:

1. Предоставляемые вычислительные ресурсы.
2. Хранилище данных и его организация.
3. Поддержка решений на разных языках программирования.
4. Ценообразование.
5. Пробный период.

По результатам проведённого анализа была выбрана платформа «Microsoft Azure», а результаты представлены в приложении (см. Приложение Е).

2.6. Выводы по главе 2

В данной главе были сформированы и формализованы функциональные и нефункциональные требования к разрабатываемому в рамках данной работы порталу для проведения анализа и последующей оценке соответствия текста проверяемой статьи академическому стилю английского языка на основе эталонных корпусов. Требования были прежде всего направлены на увеличение диапазона функций, который могут предложить рассмотренные в первой главе аналоги, но при этом сохранить понятный, простой и удобный интерфейс.

Были описаны понятия корпусной лингвистики для дальнейшего концептуального представления предметной области во время разработки. Концептуальные понятия помогут сформировать программный код в терминах пользователя и эксперта в предметной области, что упростит диалог между разработкой и пользователями.

Функциональные требования были описаны посредством нотации UML, а именно диаграмм прецедентов. Были описаны крупные блоки взаимодействия с порталом, а также их детализация:

1. Разметка корпуса.
2. Формирование отчёта по данным корпуса.
3. Формирование рекомендации по конкретному отдельному документу.

Данный способ формализации предоставляет возможность выделить функциональные блоки веб-портала и описать жизненный цикл разрабатываемой системы.

Глава 3. Проектирование портала для проведения корпусных исследований

Данная глава посвящена проектированию архитектуры и интерфейса портала. Далее будет проводиться проектирование разрабатываемого веб-портала. Ранее были описаны нефункциональные требования, а также функциональные требования. Также были определены особенности предметной области. На основе сформированных требований следует выполнить следующее:

1. Смоделировать жизненный цикл, чтобы определить порядок работы разрабатываемого инструмента.
2. Смоделировать взаимодействие модулей системы в ходе описанного на предыдущем этапе ЖЦ.
3. Спроектировать архитектуру веб-портала на уровне компонентов системы.
4. Спроектировать пользовательский интерфейс.

Также стоит уточнить, что основное внимание в ходе проектирования будет уделено двум наиболее важным сценариям: работе с корпусом, и получение оценки функционального стиля текста.

3.1. Описание жизненного цикла портала и взаимодействия его компонентов

Для описания жизненного цикла системы выбрана диаграмма последовательностей в нотации UML. Предполагаемый процесс взаимодействия пользователя с порталом довольно объемный, по этой причине следует разделить данное описание на несколько блоков, как в случае с диаграммами прецедентов. Ниже смоделирован порядок взаимодействия пользователя (см. рис. 3.1.).

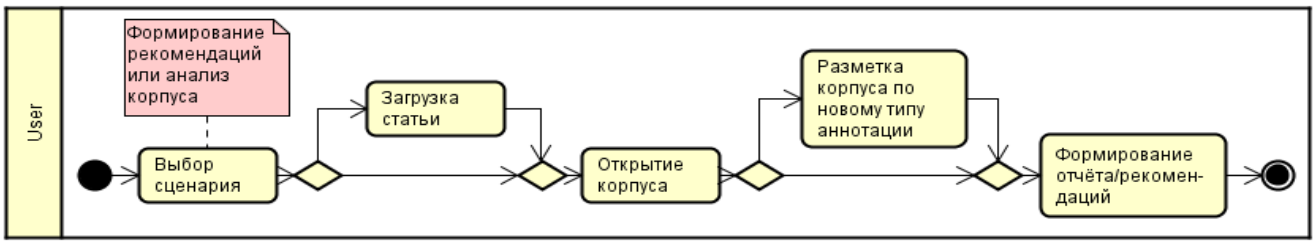

Рисунок 3.1. Общая диаграмма последовательностей

Выбор сценария - это выбор между работой с корпусом и получением оценки стиля отдельной загруженной статьи. В случае получения оценки стиля пользователю необходимо загрузить первоначально загрузить статью.

Далее будут рассмотрены сценарии связанные с открытием и разметкой корпуса, получением отчета и рекомендации. Также далее помимо пользователя в качестве участников процесса выделены модули системы, представляющие отдельные сервисы. Текстовое описание сервисов системы представлено в приложении (см. Приложение F).

Далее представлена диаграмма последовательностей, описывающая процесс «Открытие корпуса» (см. рис. 3.2). Данный процесс включает в себя следующие шаги:

1. Пользователь осуществляет поиск интересующего корпуса.
2. Сформированный запрос передается сервису поиска.
3. После нахождения необходимого корпуса данные передаются Системе.
4. Система выводит полученные данные в интерфейс.
5. Выбор корпуса пользователем.
6. Система вызывает визуальный компонент для разметки и передаёт данные корпуса.
7. Визуализация корпуса.
8. Просмотр корпуса для дальнейшего анализа.

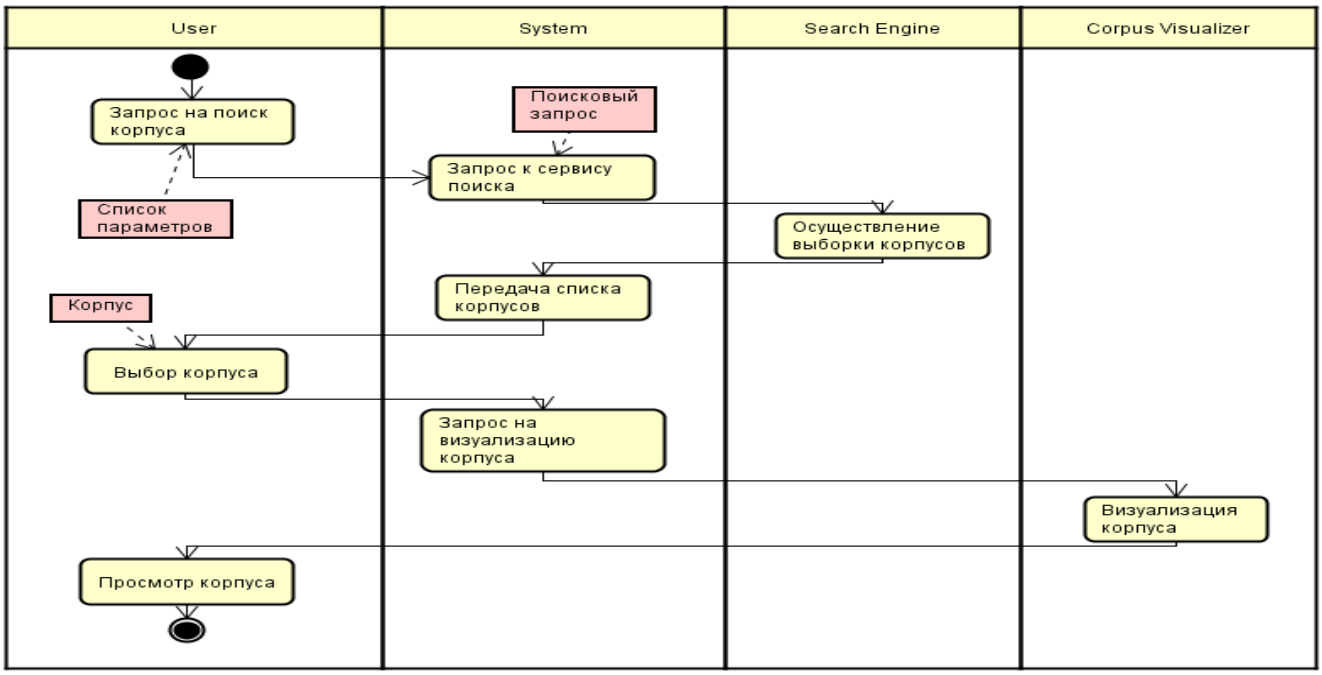

Рисунок 3.2. Диаграмма последовательностей «Открытие корпуса»

Представленная далее диаграмма моделирует процесс генерации отчётов. Диаграмма последовательностей представлена далее (см. рис. 3.3). Данный процесс содержит следующие этапы:

1. При необходимости настройки генерации отчёта, пользователь настраивает генерацию под свои нужды.
2. Пользователь передаёт запрос на генерацию отчёта по корпусу или рекомендаций по конкретному документу.
3. При инициации процесса получения рекомендаций система запрашивает разметку у компонента разметки.
4. Сервис разметки инициирует процесс аннотации и сбора метрик.
5. Система посылает запрос сервису сбора статистики для составления рекомендации или получения статистики по корпусу для генерации отчета.

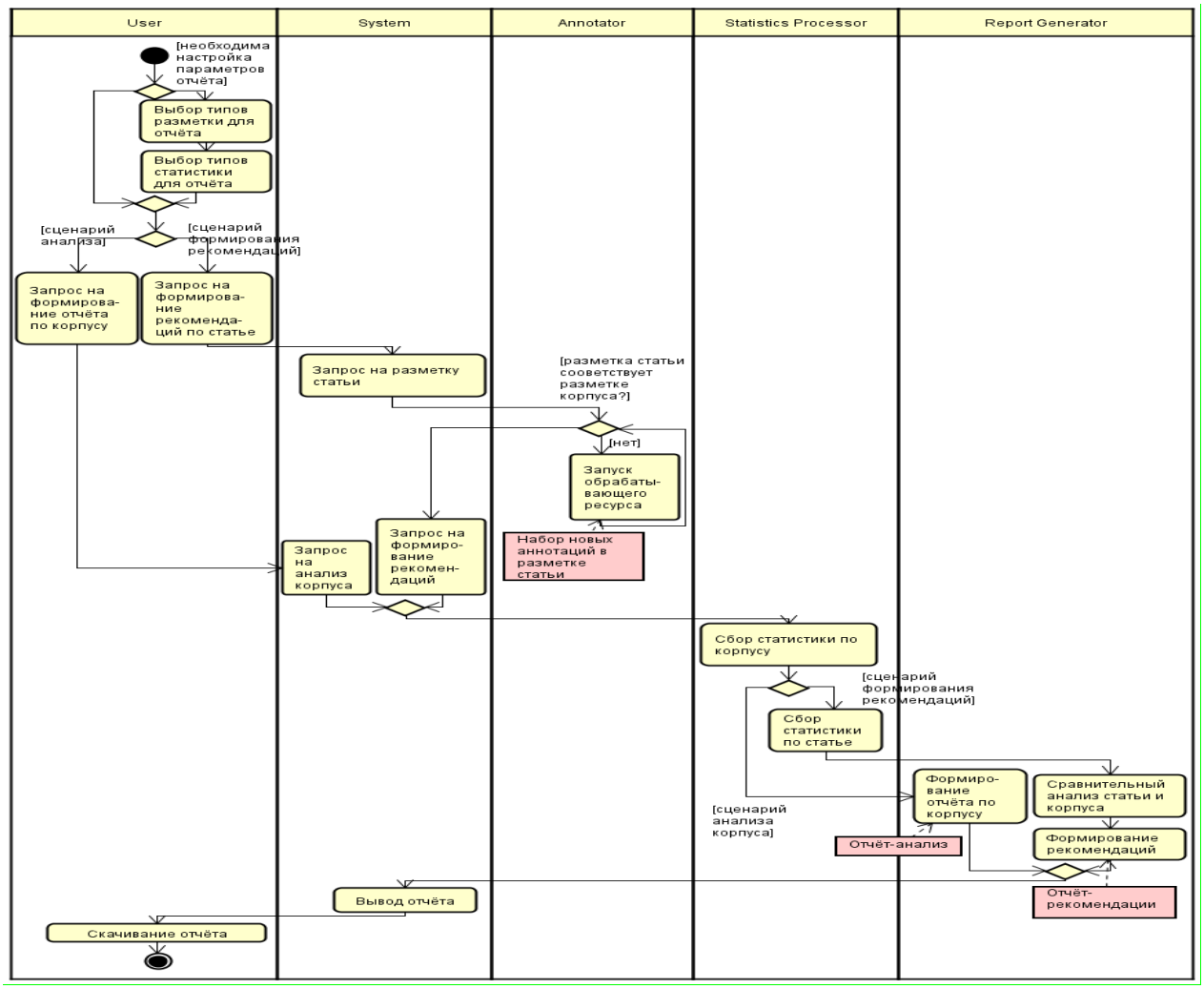

Рисунок 3.3. Диаграмма последовательностей формирования рекомендации или отчёта

6. Сервис сбора статистики возвращает статистику по корпусу, либо высчитывает для загруженного для оценки документа. После получения и форматирования данные посылаются сервису формирования отчётов.
7. Сервис для формирования отчётов создает отчёт-анализ или проводит сравнительный анализ статистических показателей статьи и корпуса и формирует рекомендацию.
8. Система выводит в интерфейс результат.
9. Пользователь получает отчёт.

Описанные диаграммы последовательностей в UML нотации моделируют основные пользовательские сценарии, которые планируется реализовать в процессе разработки портала. Результатом данного этапа является выделение модулей портала и их обязанности в рамках данной работы.

Взаимодействие модулей системы для сценариев формирования отчетов описано в виде диаграмм последовательностей, которые находятся в приложении (см. Приложение С). Данные диаграммы демонстрируют содержание запросов между сервисами системы.

3.2. Проектирование архитектуры портала

Исходя из предыдущего раздела, можно заключить, что разные функции системы на обособленные компоненты, которые взаимодействуют друг с другом покрывают основные пользовательские сценарии. Данный подход к организации разрабатываемого решения можно отнести к микро сервисной архитектуре. Такой вид архитектуры прекрасно подходит для распределённых систем, потому что хорошо масштабируется и имеет высокую взаимозаменяемость.

Данные компоненты могут быть разнесены на разные слои в рамках архитектуры данной системы:

1. Слой представления, отвечает за визуализацию работы системы.
2. Слой приложений, отвечающий за взаимодействие и работу ключевых компонентов в составе портала.
3. Слой БД, отвечающий за обработку данных по работе с корпусами и портальными учетными данными.

Разрабатываемый портал представляет из себя веб-приложением, которое работает через браузер и представляет из себя тонкий клиент. Представление отвечает за взаимодействие с пользователем. Описание архитектуры представлено ниже (см. рис. 3.4).

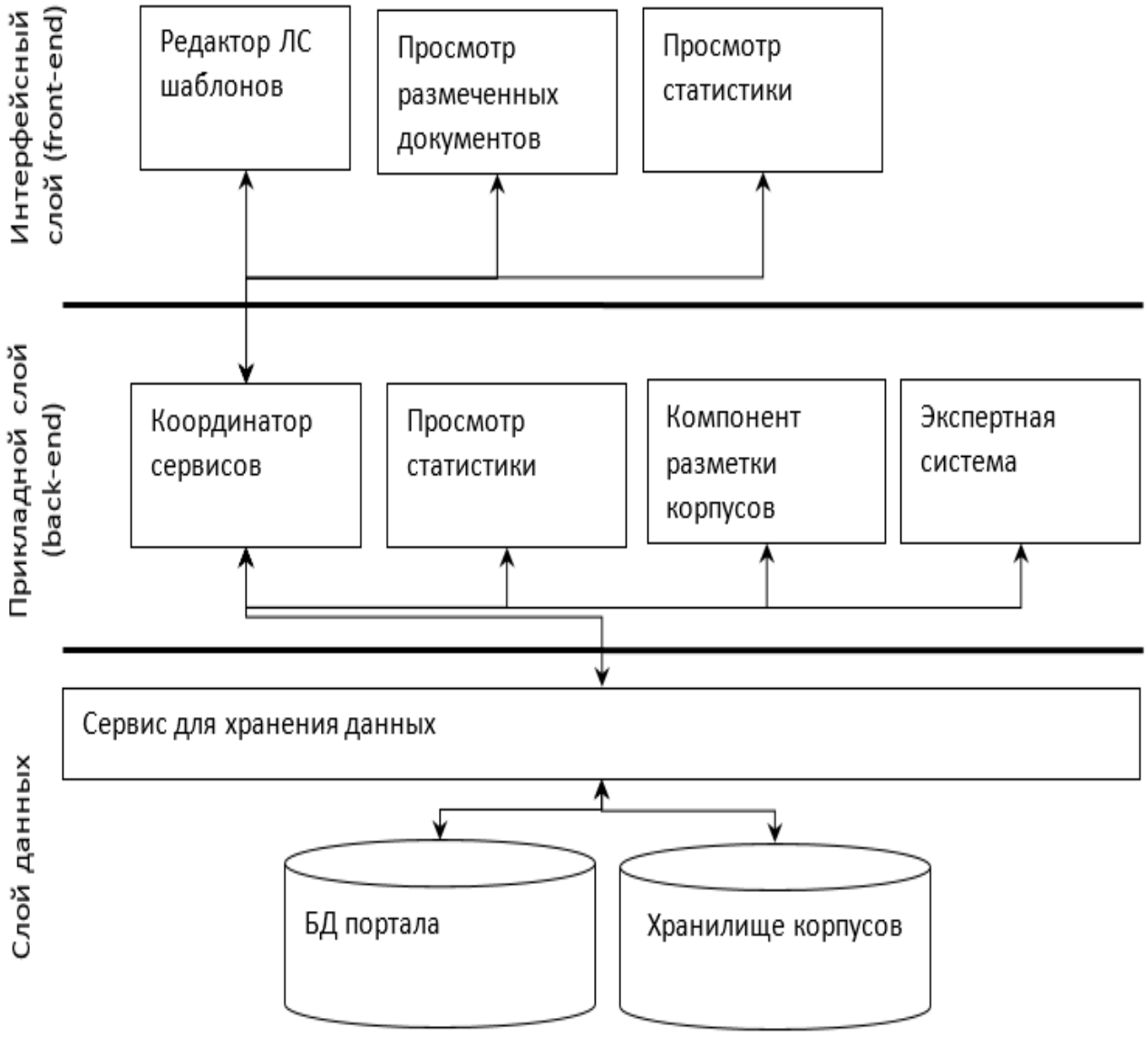

Рисунок 3.4. Описание спроектированной архитектуры системы

Системообразующим модулем в данном случае является компонент, отвечающий за обработку корпусов. Данный элемент системы выполняет анализ загруженного корпуса и формирует рекомендации по достижению функционального стиля корпуса, на основе которого производится сравнение. Метод анализа функционального стиля, который будет использован в данной работе, предполагает выполнение анализа статья в

три этапа. Каждый этап будет выполняться в рамках отдельного элемента. Данный сервис завязан на использовании готовых решений системы GATE, в которых реализованы основные методы для работы с текстом.

Первоначальный этап будет реализован в рамках компонента для разметки загруженного корпуса. Данный компонент предназначен для обработки загруженного в систему корпуса при помощи имеющихся в GATE готовых плагинов и библиотек для токенизации и синтаксического разбора текста, а также иные способы разметки. Размечать корпус потребуется только при первичной загрузке, поэтому при анализе отдельно взятой статьи на основе имеющихся в системе корпусов, достаточно будет просто сравнить размеченную статью с имеющимся данными о корпусе.

Принцип работы данного компонента можно описать как получение коллекции текстов и обрабатывающие ресурсы для нанесения разметки. После обработки корпуса, будет сформирована особым образом метаинформация, которая будет характеризовать значения тех или иных метрик относительно обработанного корпуса. Дальнейшая работа с результатом обработки осуществляется согласно методу эталонных корпусов, описанному во второй главе.

Во втором этапе предполагается использования модуля статистики для формирования количественных показателей по тем или иным метрикам корпуса. Отчет будет строиться на основе загруженного корпуса и метаинформации, полученной в результате разметки этого корпуса.

Модуль статистики в качестве результата выдает значения различных характеристик: от частоты вхождения того или иного маркера до среднего количества предложений в корпусе. Полученные данные могут выступать в качестве параметров для интересующих исследователя вычисляемых значений. Результат обработки может быть представлен в любом удобном виде.

Наиболее важной в данном случае является компонент, предназначенный для формирования отчетов по загруженной статье или размеченному корпусу. Создание отчета будет производиться по средствам готового компонента в удобном для пользователя формате. Оценка стиля и рекомендация будет сформирована на основе количественных расхождений тех или иных показателей загруженного корпуса и

отдельно взятой статьи. Результат будет выводиться в формате сообщения, в которых будут содержаться количественные расхождения показателей размеченного корпуса и размеченной статьи.

Данные будут обрабатываться следующим образом:

1. Данные пользователей и различные реляционные сущности портала будут расположены в БД портала.
2. Нереляционные данные (метаинформация, статистика, html разметка) будут расположены в хранилище корпусов.

Нереляционное хранилище предполагает хранение больших объемов данных, исходя из этого необходимо получить масштабируемое и эффективное решение.

Описанное архитектурное решение гарантирует масштабируемость и отказоустойчивость посредством слабой связности элементов системы. Сделав акцент на абстракциях, можно будет без труда подменять те или иные компоненты различными реализациями в случае необходимости.

3.3. Прототип интерфейса

Далее будет представлено описание проектов интерфейса разрабатываемого портала для основных сценариев работы с приложением. Основные сценарии – это загрузка и разметка пользовательского корпуса и загруженной статьи, и получение рекомендации на основе ранее размеченного корпуса с таким же функциональным стилем. Основное внимание уделялось диаграммам вариантов использования созданных и описанных во второй главе.

3.3.1. Прототип интерфейса стартового экрана

В данном разделе будет продемонстрирован макет интерфейса для проведения базового сценария. Данный сценарий представляет самую большую практическую ценность с точки зрения использования портала неквалифицированными лингвистами. Далее представлен прототип для выполнения сценария разметки загруженной статьи. Данный процесс включает в себя 3 этапа:

1. Загрузка статьи.
2. Аннотирование.

3. Визуальное форматирование выделения тех или иных маркеров.

Интерфейс представлен далее (см. рис. 3.5).

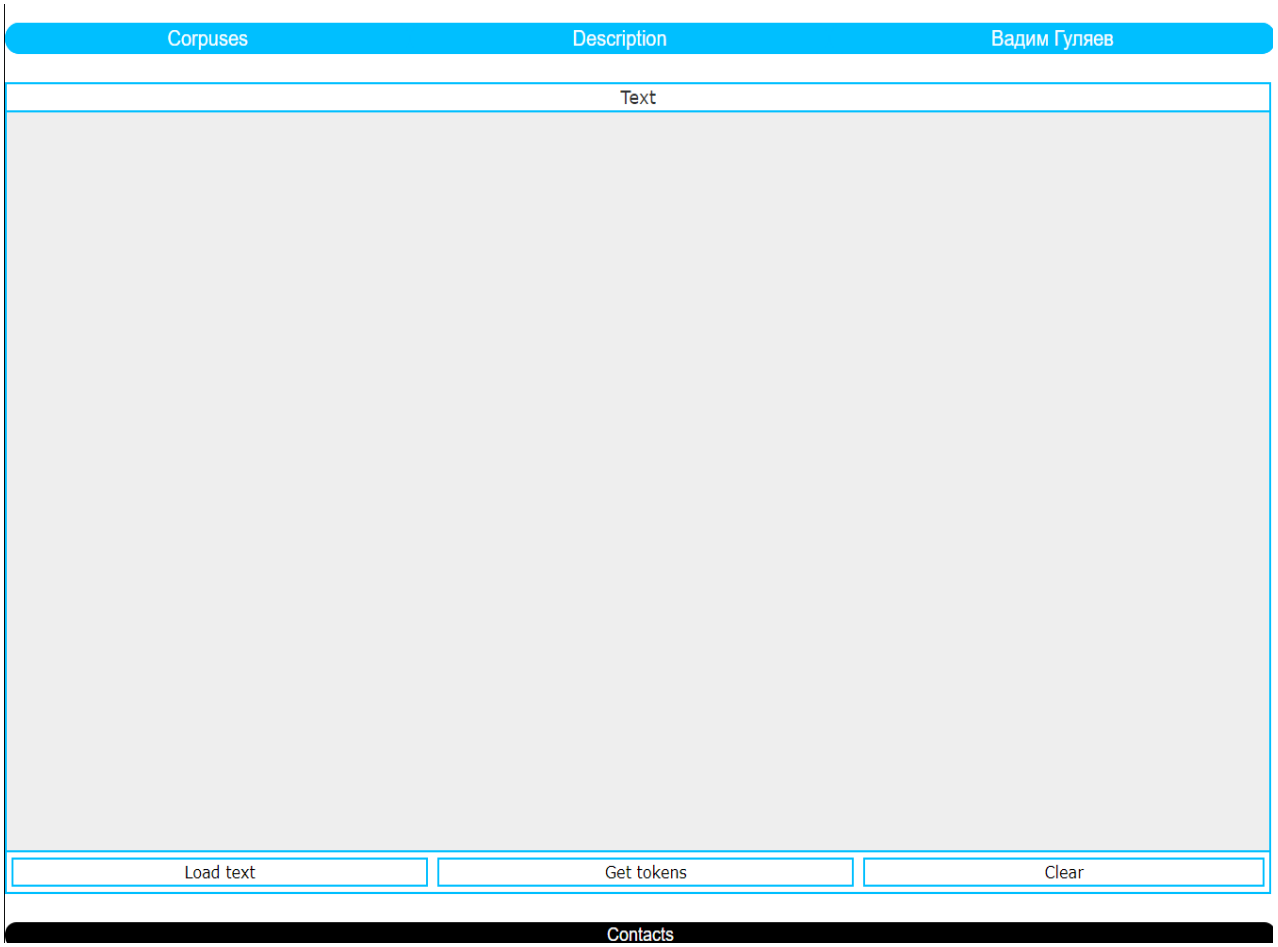

Рисунок 3.5. Проект интерфейса стартового экрана

Данный интерфейс отвечает требованиям простоты, потому что действия и их порядок соответствует правилу чтения слева направо. Пользователь, которому необходимо получить рекомендацию по исправлению текста для соответствия функциональному стилю корпуса, для достижения цели должен перейти к файловому менеджеру в зависимости от операционной системы, нажав кнопку «Load text», выбрать нужный файл в формате txt и затем нажать кнопку «Get tokens».

3.3.2. Прототип интерфейса для загрузки файла из файлового менеджера ОС

Далее будет представлен вариант с использованием данного портала в браузере, установленном на компьютере под управлением операционной системы Windows. После нажатия кнопки «Load text» будет возможность загрузить только файлы в формате txt в силу настроенного фильтра для поиска в файловом менеджере. Данное ограничение

обуславливается текущей реализацией компонента разметки. В дальнейшем в ходе развития проекта будет возможность работать с различными форматами файлов. Интерфейс представлен далее (см. рис. 3.6).

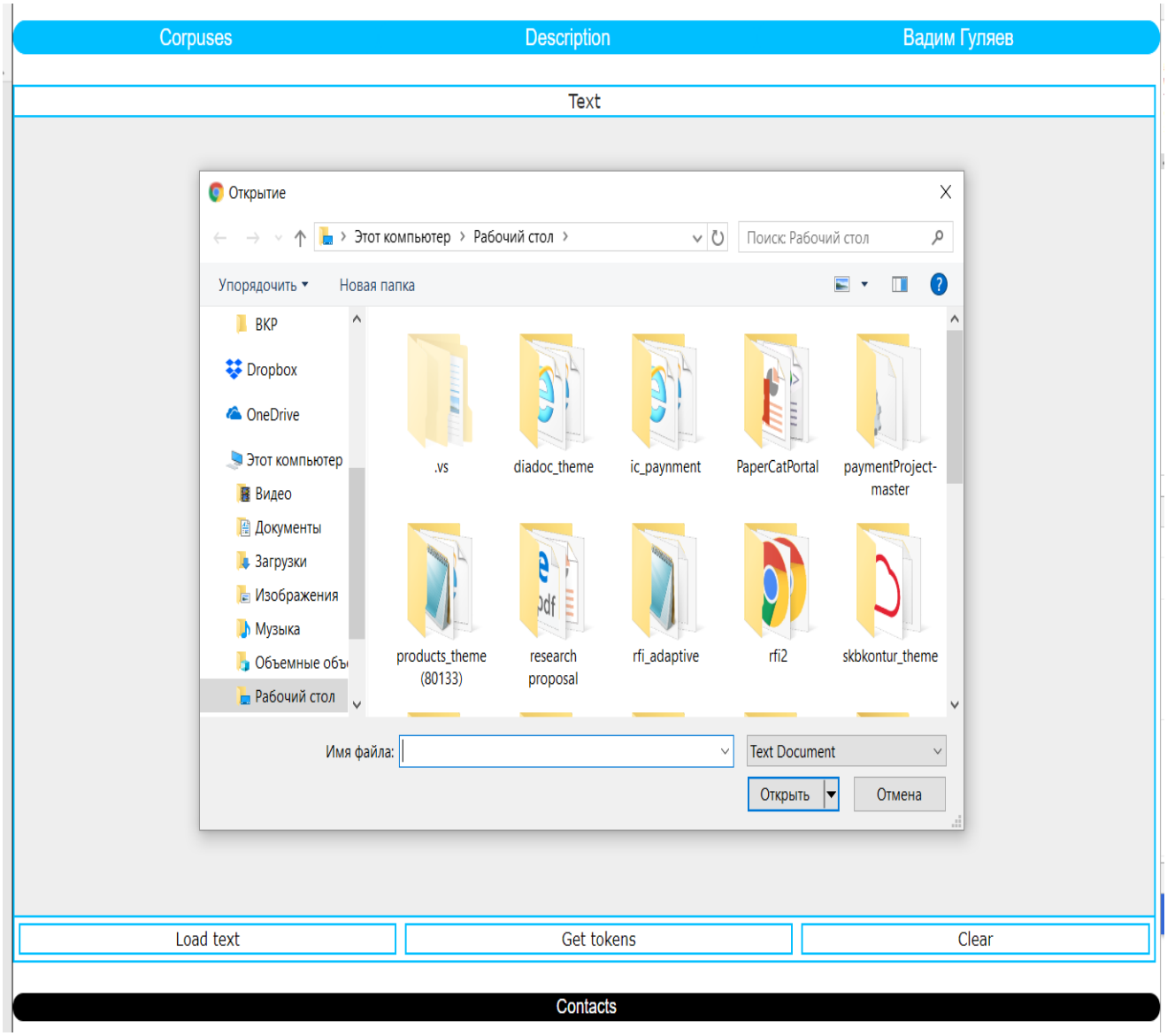

Рисунок 3.6. Проект интерфейса загрузки статьи для получения рекомендации

3.3.3. Прототип интерфейса работы с неразмеченной статье

После выбора файла, в котором содержится текст статьи для проверки функционального стиля, текст файла будет загружен в браузер и будет расположен в основном окне приложения. Текст будет доступен для просмотра и поиска по тексту с использованием встроенного в браузер поисковика текста. Кроме того, если текст не будет помещаться в отведённой браузером области, то появится соответствующий стандартный контролл для прокрутки.

Текст загруженный из файла – это основа для будущей обработки и анализа статьи модулем разметки. После обработки текста будет визуализирована разметка составленная в соответсвии с полученными данными от компонента разметки. Макет интерфейса расположен далее (см. рис. 3.7).

Рисунок 3.7. Проект интерфейса после загрузки статьи для получения рекомендации

3.3.4. Прототип интерфейса работы с компонентом визуализации

После загрузки статьи пользователю предоставляется возможность нажать кнопку «Get tokens». До того момента как пользователь загрузит текст для разметки нажатие кнопки «Get tokens» ни к чему не приведёт, и пользователь получит сообщение о том, что для получения аннотации необходимо загрузить текст.

Результатом нажатия кнопки «Get tokens» должно быть изменение интерфейса. В правой части экрана должна появиться область, где будут расположены все выявленные в тексте маркеры. Все маркеры доступны для выделения их в исходном тексте. Данная

предоставляемая возможность позволит визуализировать и помочь локализовать чрезмерно частотное нахождение в тексте тех или иных речевых единиц.

За основу взят интерфейс работы с маркерами в системе GATE. Подход используемый в GATE прост и понятен: список маркеров, имеющих уникальный цвет расположен в правой части экрана, а сами маркеры можно выделять, либо скрывать в зависимости от интересов пользователя. Проект интерфейса работы с маркерами представлен далее (см. рис. 3.8).

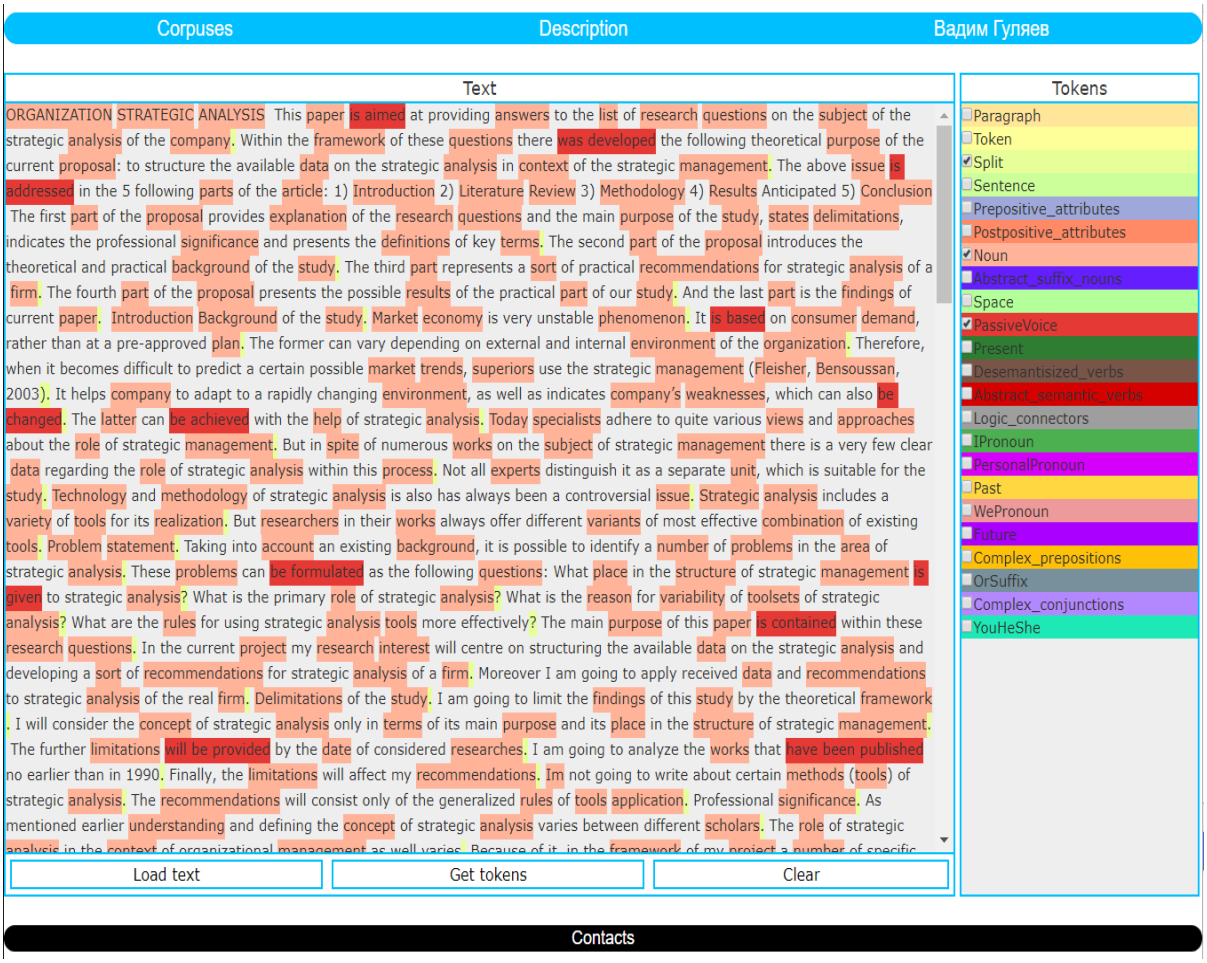

Рисунок 3.8. Проект интерфейса для работы с маркерами

После получения итогов разметки и рекомендации в виде всплывающего сообщения и предложения сохранить результаты обработки статьи в текстовом файле на жестком диске пользователя, для очистки основного рабочего пространства приложения и скрытия области с маркерами необходимо нажать кнопку «Clear».

3.3.5. Прототип интерфейса взаимодействия с корпусами в системе

После работы по основному сценарию, связанному с разметкой статьи пользователя и получением рекомендации, следует описать интерфейс работы с корпусами. Работа с корпусами в данном случае будет рассмотрена с двух точек зрения: с точки зрения просмотра и изучения корпуса и с точки зрения загрузки собственного корпуса для разметки. Работа с корпусами с целью просмотра и изучения предполагает поиск по названию стилю, то есть в поисковой строке можно искать одновременно по тому или другому параметру – поисковый запрос вернет результаты, где выявлено совпадению по стилю или названию. Интерфейс раздела с корпусами и поиском представлен далее (см. рис. 3.9).

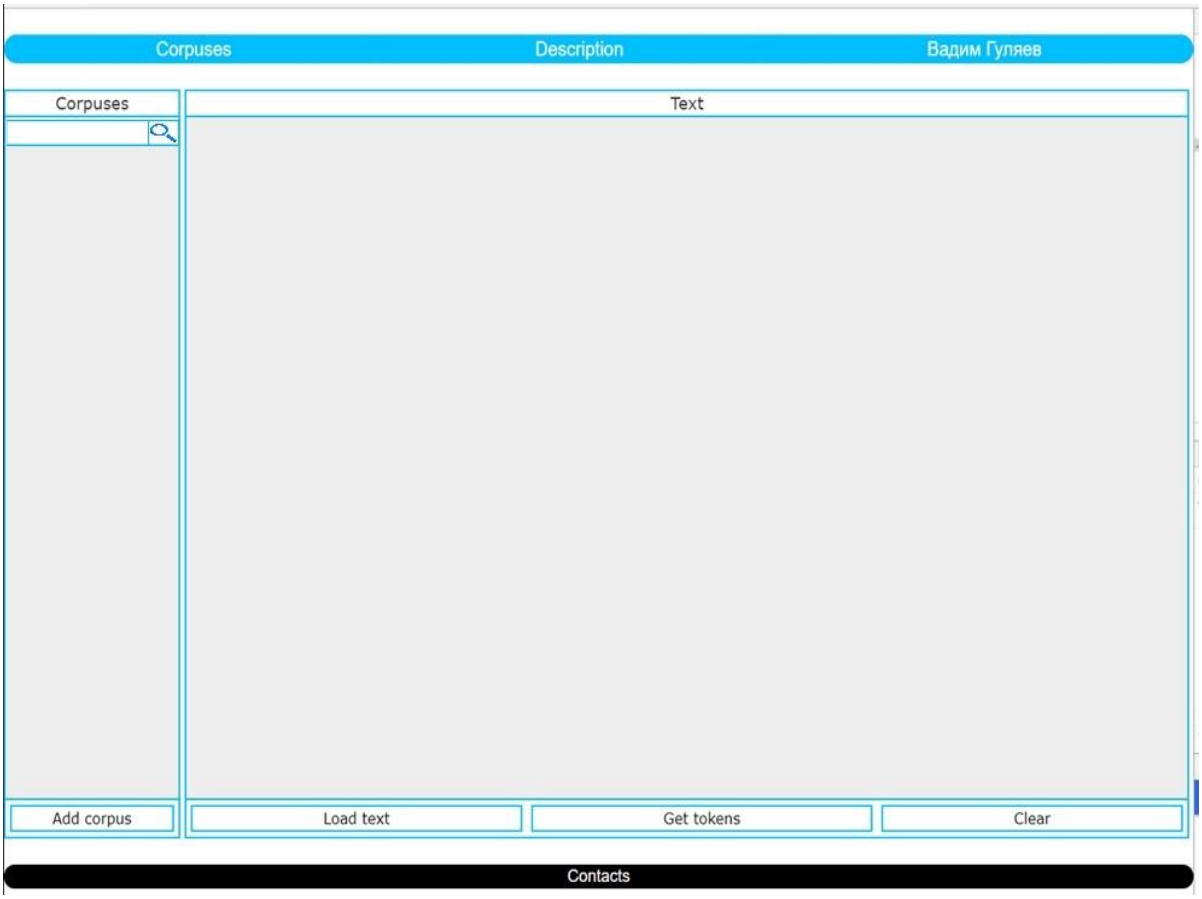

Рисунок 3.9. Проект интерфейса для работы с корпусами

Данная секция позволяет осуществлять поиск по названиям корпусов и по функциональному стилю корпуса. В случае если запрос вернул ноль элементов, то пользователь будет проинформирован о том, что для нахождения корпусов необходимо изменить запрос.

Для того чтобы загрузить собственный корпус необходимо открыть секцию для работы с корпусами, которая была описана ранее и нажать кнопку «Add corpus». После нажатия на данную клавишу по аналогии с загрузкой статьи для получения рекомендаций пользователь должен будет в файловом менеджере выбрать набор файлов в формате txt. После выбора файлов пользователю будет предложено дать корпусу название, определить права доступа (публичный/приватный), указать язык написания статей и функциональный стиль корпуса для дальнейшего поиска. После завершения данных процедур, начнется длительная операция разметки корпуса. По результатам разметки корпус станет доступен среди других корпусов.

3.4. Выводы по главе 3

В ходе проектирования архитектуры приложения была выбрана трехслойная микросервисная архитектура. Данное решение является масштабируемым и отказоустойчивым, а оперирование взаимодействием на уровне абстракций позволит подменять реализации в ходе доработки или разработки тех или иных компонентов системы. Каждый микросервис является изолированным компонентом и может быть развернут на разных машинах. Порядок и описание взаимодействия элементов веб-портала на разных уровнях архитектуры продемонстрировано в виде диаграмм активности и последовательности.

На ряду с архитектурой были представлены и описаны макеты интерфейса покрывающие базовые сценарии работы с порталом. Требования по простоте и подходящем для всех категорий пользователей интерфейсе выполнены, аргументация представлена в описании макетов. Для получения рекомендации и для начала работы с визуализацией разметки достаточно нажать две кнопки и выбрать файл в файловом менеджере системы.

Глава 4. Реализация портала

Данная глава посвящена реализации веб-портала. В данной главе будут описаны используемые программные средства и их применение в данной работе. В качестве основной технологии используется .NET Framework. Для клиентской части был использован стандартный набор инструментов:

1. HTML.
2. CSS.
3. JavaScript.

Далее будут описаны ключевые функции с точки зрения решаемых задач, таких как обработка xml файла с разметкой и конвертация в html разметку для визуализации на клиенте.

4.1. Используемые технологии

Клиентский слой веб-портала представляет собой ASP.NET приложение. Это популярное решение для написания веб-приложений на языке C#. Для обеспечения простоты организации проекта и надежности механизмов взаимодействия в качестве заготовки проекта был выбран проект ASP.NET WEB API.

4.1.1. Шаблон проекта ASP.NET WEB API

Визуально проект ASP.NET WEB API по иерархии папок в файловой системе, созданной при инициации проекта, напоминает ASP.NET MVC, но ключевым отличием шаблонов проектов заключается в том, что котроллеры в ASP.NET WEB API наследуются от ApiController, а не от Controller. ApiController и Controller никак не связаны.

Стоит также отметить, что контроллеры ASP.NET WEB API представлены в виде REST (Representation State Transfer). Методы, реализованные в контроллере, унаследованном от ApiController возвращают не ActionResult, в отличии от контроллеров, которые унаследованы от Controller. Необходимо называть методы одноименно с HTTP запросами, или прописывать атрибуты, что тот или иной метод сопоставляется с тем или иным HTTP запросом.

4.1.2. Язык разметки HTML

HTML (Hyper Text Markup Language) – язык разметки гипертекста, который является стандартом для визуального отображения веб-страниц. Данный стандарт поддерживается всеми браузерами. При помощи данного языка можно с легкостью определить структуру страницы в виде DOM (Document Object Model).

В проекте ASP.NET WEB API есть возможность работать с HTML прямо из Visual Studio. Данный момент облегчает разработку клиента. Стоит отметить, что также присутствует возможность использовать встроенные средства для работы с разметкой такие как Razor или ASPX.

4.1.3. Язык для описания стилей CSS

CSS (Cascade Style Sheet) – стандарт веб-разработки, такой же как и HTML, но его предназначение декларативно описывать внешней вид элементов DOM описанных при помощи HTML. Это довольно гибкий инструмент для дизайнеров и фронтэндеров, которые отвечают за внешний вид продуктов, а также в умелых руках является инструментом быстрого прототипирования.

В проекте ASP.NET WEB API так же, как и в случае с HTML, есть поддержка синтаксиса CSS. Также предоставляется описание стандартных элементов и методов, что упрощает разработку. Файлы со стилями подключаются в файле с размеченной страницей аналогично любому веб-проекту.

4.1.4. Язык JavaScript

JavaScript – это интерпретируемый язык программирования со слабой типизацией, который придает интерактивность веб-страницам. При помощи данного языка можно запрограммировать поведение веб-страницы на стороне браузера. Данный аспект позволяет проводить элементарные операции и часть определённой логики на клиенте, а на сервер обращаться в крайних случаях.

В Visual Studio при работе над проектом предоставляются различные средства для удобства разработки на JavaScript, в том числе отладка. Для расширения его стандартных возможностей доступных из стандартного проекта ASP.NET, необходимо подключать сторонние библиотеки.

4.2. Клиентская часть приложения

Разработка клиентской части в рамках данной выпускной квалификационной работы заключалась в применении комбинации средств веб-разработки. Как и в большинстве случаев, исходя из описания выбранных средств разработки, данный набор состоял из HTML, CSS и JavaScript.

Для начала необходимо было определить структуру документа. Visual Studio предоставляет редактор для работы с HTML. Структура страницы следующая:

1. Заголовочная часть.
2. Основной контент.
3. Подвал страницы.

Все элементы были обернуты в div контейнеры для удобства дальнейший стилизации, так как некоторые стандартные элементы управления не поддаются преобразованию внешнего вида и все современные методы при работе с CSS направлены на блочный элемент div.

Заголовочная часть содержит три кнопки:

1. Загрузка корпусов.
2. Описание портала.
3. Вход в систему.

Код заголовочной части представлен далее (см. рис. 4.1):

```
<div>
    <button onclick="getCorpuses()">Corpuses</button>
</div>
<div>
    <button onclick="showDescription()">Description</button>
</div>
<div>
    <button onclick="authorize()">Login</button>
</div>
```

Рисунок 4.1. Код разметки для заголовочной части

Следующим логическим блок является контейнер-контент. В данном контейнере содержится область для работы с текстом. «Плоский текст» после разметки заменяется на html разметку с соответствующими классами.

Панель управления для главного сценария содержит три кнопки. Две кнопки для загрузки и разметки статьи, а кнопка «Clear» для очистки рабочего пространства. Код данного блока представлен далее (см. рис. 4.2):

```
<div id="main-content" class="main-content">
    <div id="corpuses" class="corpuses invisible">
        <div id="corpus_contentHead" class="contentHead">Corpuses</div>
        <div id="corpus_content" class="content">
        </div>
        <div id="corpus_operations" class="operations">
            <div class="operation">
                <button onclick="addCorpus()">Add corpus</button>
            </div>
        </div>
    </div>
    <div id="workspace" class="workspace">
        <div name="centerHead" id="workspace_contentHead" class="contentHead">Text</div>
        <div id="content" class="content">
        </div>
        <div id="operations" class="operations">
            <div class="operation">
                <button onclick="load()">Load text</button>
            </div>
            <div class="operation">
                <button onclick="getTokensFromServer()">Get tokens</button>
            </div>
            <div class="operation">
                <button onclick="clearContent()">Clear</button>
            </div>
        </div>
    </div>
    <div id="tokens" class="tokens invisible"></div>
```

Рисунок 4.2. Код основной части страницы

Последним логическим блоком является так называемый подвал. Там содержится ссылка на НУГ «Разработка программного обеспечения для проведения корпусных исследований английского языка». Код представлен ниже (см. рис. 4.3):

```
<div class="footer">
    <div>
        <a href="https://perm.hse.ru/bi/sfcr/" target="_blank">Contacts</a>
    </div>
</div>
```

Рисунок 4.3. Код нижней части страницы

На этом формирование структуры документа закончилось. Далее придавался внешний вид стандартным элементам управления в HTML. Стилизация производилась при помощи CSS. За счет определённых классов у элементов в документе стало возможным воспользоваться селекторами CSS и гибко придавать красивый внешний вид выбранным элементам.

Расположение блоков и элементов внутри этих блоков произведено при помощи flex-box. Данная технология позволяет отойти от устаревшей табличной верстки. Гибкий и удобный метод позволил добиться желаемого результата, а именно работы не только с компьютеров, но и с мобильных устройств. Пример стилизации основных блоков приведен ниже (см. рис. 4.4):

```
.corpuses{
  display: flex;
  flex-grow: 1;
  border: solid 2px deepskyblue;
  flex-direction: column;
  background: #eee;
  height: 100%;
  min-width: 15%;
  max-width: 15%;
  animation: anim .5s ease-in-out;
  margin-right: 5px;
  margin-left: 3px;
  flex: 1 1 0;
  overflow: auto;
}

.page{
    display: flex;
    flex-direction: column;
    height: 100%;
}

.footer{
    list-style: none;
    background: black;
    display: -webkit-box;
    display: -moz-box;
    display: -ms-flexbox;
    display: -webkit-flex;
    display: flex;
    flex-grow: 0;
    flex-flow: row;
    justify-content: center;
    margin-top: 2%;
    margin-bottom: 2%;
    border-radius: 100px;
}
```

Рисунок 4.4. Верстка с использованием flexbox

Для того, чтобы пользователь подождал результатов разметки загруженной статьи, необходимо было создать приятную анимацию для ожидания. Для каждого маркера был сформирован свой собственный цвет. Также для интерактивности JavaScript файл реагирует на все события, происходящие на странице.

Запросы на сервер для получения и передачи каких-либо данных происходит с использованием технологии AJAX. На ряду с этим используется библиотека jQuery для простого взаимодействия с DOM из JavaScript. Процесс получения разметки продемонстрирован ниже (см. рис. 4.5).

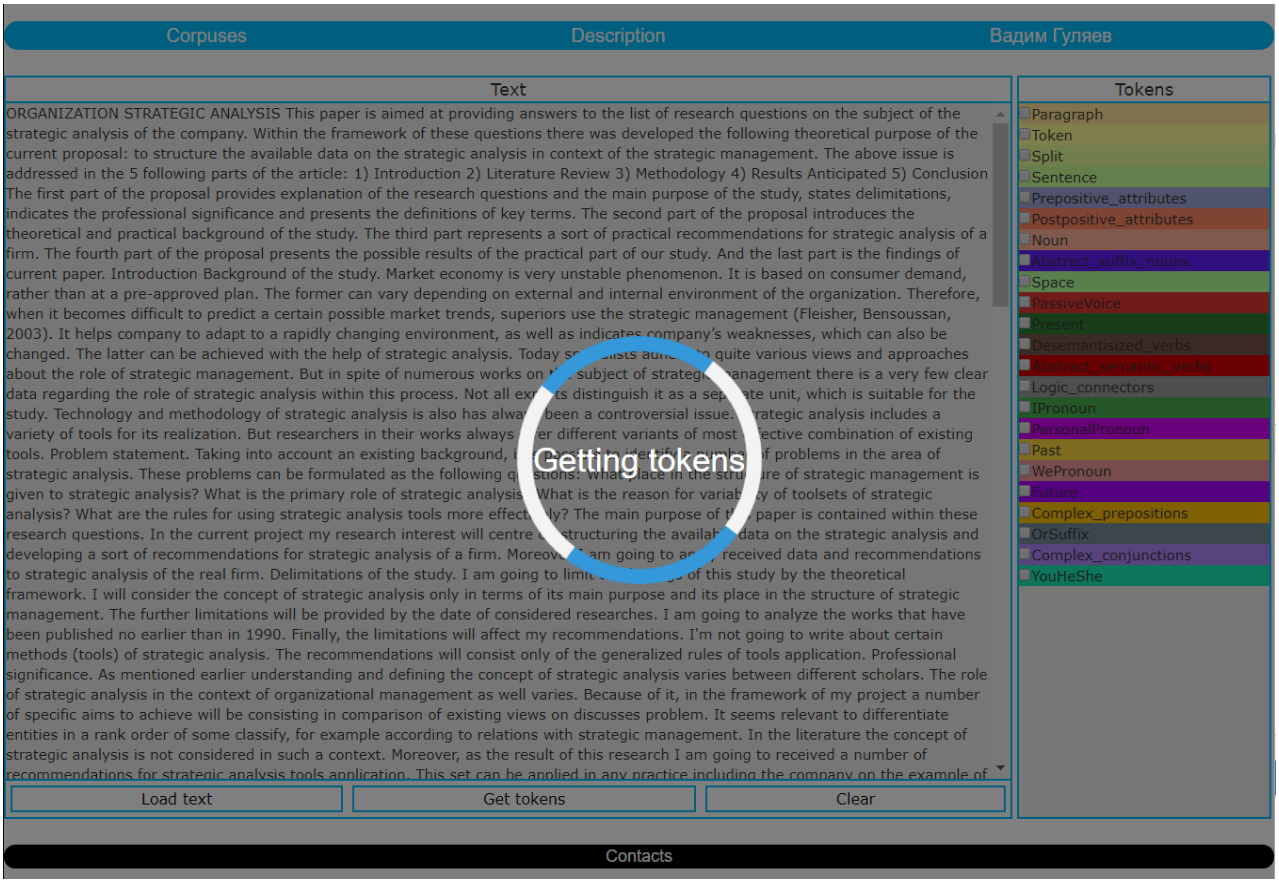

Рисунок 4.5. Демонстрация процесса ожидания разметки

Процесс работы с визуальным представлением результатов разметки статьи происходит посредством присвоения нескольких классов обозначающих тот или иной маркер. Код для получения списка маркеров, полученных в ходе разметки представлен далее (см. рис. 4.6).

Также необходимо раскрасить сам текст статьи после разметки. Данная задача была решена посредством деления плоского текста загруженного для анализа функционального стиля на много различных блоков «div» для дальнейшей перекраски заднего фона в зависимости от того в какие маркеры входит тот или иной участок текста. Результат работы представлен далее (см. рис. 4.7).

```
function getTokensFromServer() {
    if (textIsLoaded) {
        if (!tokenIsLoaded) {
            $.ajax({
                url: 'GetTokens',
                data: param = "",
                contentType: 'application/json; charset=utf-8',
                dataType: 'json',
                success: function (data) {
                    document.getElementById("overlay").style.display = "block";
                    document.getElementById("loader").style.display = "block";
                    tokenIsLoaded = true;

                    document.getElementById("tokens").innerHTML += "<div id=\"contentHead\" class=\"contentHead\"> Tokens </div>";
                    for (i = 0; i < data.length; i = i + 1) {
                        let tokneDiv = "<div class=\"" + data[i] + "\"><input value=\"" + data[i] + "\" type=\"checkbox\" onclick=\"changeBG(this.value)\">" + data[i] + "</div>";
                        document.getElementById("tokens").innerHTML += tokneDiv;
                    }
                    document.getElementById("tokens").classList.toggle("invisible");

                    colorText();
                },
                error: function () {
                    console.log('Error!');
                }
            });
        }
    } else {
        alert('Please, load the text before getting tokens.');
    }
}
```

Рисунок 4.6. Код для загрузки маркеров, полученных в ходе разметки

Рисунок 4.7. Демонстрация работы с компонентом визуальной разметки

4.3. Серверная часть приложения

Серверная часть в данном приложении отвечает за взаимодействие с другими компонентами системы, обработку результатов разметки, оперирование данными портала. Для корректной передачи данных в слой хранилища данных необходимо определить контракты взаимодействия и модели данных для сериализации и десериализации.

Основной метод для обработки полученных из компонента разметки xml файла, преобразует данные в список токенов. В классе токен реализован интерфейс IComparable, который позволяет выполнять сортировку типизированной коллекции. Код класса токен представлен ниже (см. рис. 4.8).

```
[DataContract]
12 references
public class Token: IComparable<Token>
{
    [DataMember(Name = "tokenName")]
    public string TokenName;
    [DataMember(Name = "startPosition")]
    public int StartPosition;
    [DataMember(Name = "finishPosition")]
    public int FinishPosition;

    2 references | 0 exceptions
    public Token(
        string tokenName,
        int startPosition,
        int finishPosition)
    {
        TokenName = tokenName;
        StartPosition = startPosition;
        FinishPosition = finishPosition;
    }

    1 reference | 0 exceptions
    int IComparable<Token>.CompareTo(Token otherToken)
    {
        if (FinishPosition > otherToken.FinishPosition)
            return -1;
        if (FinishPosition == otherToken.FinishPosition)
            return 0;
        return 1;
    }
}
```

Рисунок 4.8. Класс токен

Метод, который извлекает данные из xml файла и проводя вычисления конвертирует исходные данные в коллекцию токенов. Код метода представлен ниже (см. рис. 4.9).

```
private static List<Token> GetTokensFromXml(string text)
{
    var xmlDoc = XDocument.Parse(text);
    var tokenList = xmlDoc
        .Descendants("AnnotationSet")
        .Descendants("Annotation")
        .Select(obj => new Token(
            obj.Attribute("Type")?
                .Value.ToString() == "SpaceToken"
                ? "Space"
                : (obj.Attribute("Type")?.Value.ToString() == "paragraph"
                    ? "Paragraph"
                    : string.Join("_", obj.Attribute("Type")?.Value.ToString().Split(' '))),
            Convert.ToInt32(obj.Attribute("StartNode")?.Value.ToString()),
            Convert.ToInt32(obj.Attribute("EndNode")?.Value.ToString())
        ))
        .ToList();

    return tokenList;
}
```

Рисунок 4.9. Метод конвертации xml файла в коллекцию токенов

Затем идет формирование html разметки для передачи на клиент. Формирование проходит в два этапа. Так как токен содержит описание в виде стартовый символ и конечный символ, то нужно найти пересечения всех отрезков, чтобы в случае вхождения того или иного участка текста в несколько маркеров, отнести его сразу ко всем.

Получение пересечений всех токенов вынесено в отдельный метод. Код метода представлен ниже (см. рис. 4.10).

```
private static List<int> GetEdgeList(IEnumerable<Token> tokenList)
{
    var edgeList = new List<int>();

    foreach (var token in tokenList)
    {
        edgeList.Add(token.StartPosition);
        edgeList.Add(token.FinishPosition);
    }

    edgeList = edgeList.Distinct().ToList();
    edgeList.Sort();
    return edgeList;
}
```

Рисунок 4.10. Метод вычисления точек на отрезке в виде текста

Последним этапом является формирование html разметки. Но перед этим необходимо определить какие токены спроецированы на выделенные на предыдущем шаге граничные значения. Код метода представлен ниже (см. рис. 4.11).

```
private static List<Token> GetNewTokenList(IReadOnlyList<int> edgeList, List<Token> tokenList)
{
    var newTokenList = new List<Token>();

    for (var i = 0; i < edgeList.Count - 1; i++)
    {
        var tokenNameList = tokenList
            .Where(x => x.StartPosition <= edgeList[i] && x.FinishPosition >= edgeList[i + 1])
            .Select(x => x.TokenName)
            .Distinct()
            .ToList();

        var resultTokenName = tokenNameList.Aggregate(string.Empty, (current, tokenName) => current + (tokenName + " "));

        resultTokenName = resultTokenName.Trim();
        newTokenList.Add(new Token(resultTokenName, edgeList[i], edgeList[i + 1]));
    }
    return newTokenList;
}
```

Рисунок 4.11. Метод проецирования маркеров на полученные отрезки

Финальный шаг формирование html разметки для передачи на клиент. Код представлен ниже (см. рис. 4.12).

```
private static string GetResult(IReadOnlyList<Token> newTokenList, string content)
{
    var result = string.Empty;

    for (var i = 0; i < newTokenList.Count && newTokenList[i].FinishPosition < content.Length; i = i + 1)
    {
        var startPosition = newTokenList[i].StartPosition;
        var finishPosition = newTokenList[i].FinishPosition;
        var length = finishPosition - startPosition;
        var tokenName = newTokenList[i].TokenName;

        result += ColorPartOfText(content.Substring(startPosition, length), tokenName);
    }

    return result;
}
1 reference | 0 exceptions
private static string ColorPartOfText(string str, string token)
{
    var result = $"<div name='{token}' class='' title='In {token}'>{str}</div>";
    return result;
}
```

Рисунок 4.12. Метод проецирования маркеров на полученные отрезки

Формирование рекомендаций проходит тривиально, вычисление разницы между количественными показателями метрик, усредненных по корпусу и полученных из разметки. Передача на клиент осуществляется в виде строки, полученной в ходе сериализации в формат JSON, благодаря датаконтрактам описанных в классах.

4.4. Тестирование портала

В ходе работы было проведено ручное тестирование, направленное на отладку того или иного фрагмента программы. Проводилось юнит тестирование. Также было проведено системное тестирование, которое было направлено на проверку нефункциональных требований. И было проведено интеграционное тестирование, которое было направлено на проверку взаимодействия всех компонентов, разработанного портала.

Особое внимание было уделено проверке взаимодействия компонентов системы, потому что в распределённых системах всегда есть риск потери данных. В первую очередь проверялись ситуации, связанные с основной функциональностью, так как эти места наиболее уязвимы из-за частоты использования.

4.5. Выводы по главе 4

В данной главе был описан стек технологий использованный при реализации портала. Были описаны причины использования тех или иных инструментов. Был описан процесс и особенности реализации клиентской части с учетом специфик разработки в Visual Studio. На ряду с клиентской частью была описана реализация серверной части. В качестве итога разработан тонкий клиент и серверная часть на основе шаблона ASP.NET WEB API

Помимо реализации компонентов данного приложения стоит отметить, что клиент является адаптивным под мобильные устройства. Описан процесс тестирования портала. Кроме того, тестирование проводилось в том числе и на мобильных устройствах.

Заключение

В ходе данной работы был проведен анализ предметной области и определение специфики проведения корпусных исследований. Исходя из особенностей предметной области, оценки существующих решений и мнения экспертов были сформулированы функциональные и нефункциональные требования к порталу для анализа и оценки стиля англоязычных научных публикаций.

Говоря о вариантах использования портала можно заключить, что существует два основных пользовательских сценария:

1. Один сценарий относится к профессиональным лингвистам, которым интересно оперировать количественными показателями при анализе корпуса.
2. Другой сценарий характеризует использование портала неквалифицированными пользователями, которые нуждаются в оценке стиля заложенным в портале методом эталонных корпусов и получении рекомендаций.

С точки зрения организации всей системы в целом сценарий с анализом собственных корпусов требует оперирования большими объемами данных, что требует надежных коммуникаций между компонентами системы.

Требования к порталу были формализованы и описаны посредством диаграмм вариантов использования, с подробным описанием прецедентов. Взаимодействие компонентов внутри портала описано с помощью диаграмм последовательностей.

Так же стоит отметить организацию всей системы в целом. Предложенная трехслойная архитектура позволяет абстрагироваться от конкретных реализаций и оперировать на уровне абстракций, что обеспечивает легкую заменимость компонентов и масштабируемость.

В основе системы лежит ранее разработанный компонент разметки, разработанный на основе инструмента GATE Developer. Интеграция с данным компонентом является основополагающей, но взаимодействие только с этим компонентом разметки это лишь временно. Возможно подключение различных компонентов разметки с тем же интерфейсом.

Разработка велась с привязкой к терминам предметной области, чтобы в момент обсуждения концептуальных вопросов было проще вести диалог с экспертами, мнение которых является основополагающим на всем времени работы. Также стоит сказать, что работа над порталом проходила в рамках научно-учебной группы.

Помимо участия в научно-исследовательской деятельности результаты исследования были апробированы на всероссийской научно-практической конференции «Математика и междисциплинарные исследования» в соавторстве с другими участниками проекта и были награждены дипломом первой степени в рамках секции «Прикладная лингвистика».

Библиографический список

1. Бармина Е. И. Система для обработки корпусов текстов / Е. И. Бармина, Р. Н Бушуев, Н. В. Котельникова, В. В. Ланин, О. А. Плотникова // Математика и междисциплинарные исследования – 2016. Пермь: Пермский государственный национальный исследовательский университет, 2016. – С. 245-250.

2. Евстигнеева Г. А. Способы выражения причинно-следственных отношений в научном стиле речи: (На материале учебных текстов).–дис. канд. филол. наук. – Киев, 1983.

3. Ермакова Л. М. Методы автоматической классификации текстов по функциональным стилям / Л. М. Ермакова, М. А. Абашев, Р. В. Никитин, Р. И. Ушаков // Вестник Пермского университета. Математика. Механика. Информатика. Выпуск 4(27). – Пермь, 2014.

4. Academic Phrasebank [Электронный ресурс] // URL: http://www.phrasebank.manchester.ac.uk/ (дата обращения: 22.03.2018).

5. Anthony, L. Characteristic features of research article titles in computer science / IEEE Transactions on Professional Communication 44 (3), 2001. – P. 187-194.

6. GATE.ac.uk [Офиц. сайт] // URL: http://gate.ac.uk/ (дата обращения: 28.04.2018).

7. AntConc Homepage // Laurence Anthony's Website URL: http://www.laurenceanthony.net/software/antconc/ (дата обращения: 28.04.2018).

8. Luyckx K., Daelemans, W. Shallow text analysis and machine learning for authorship attribution // Computational Linguistics in the Netherlands 2004: selected papers from the Fifteenth CLIN Meeting / van der Wouden T. [Ed.], e.a., Utrecht, LOT, 2005, –P. 149-160.

9. Scholz T., Conrad S. Style Analysis of Academic Writing // Natural Language Processing and Information Systems: 16th International Conference on Applications of Natural Language to Information Systems, Proceedings. NLDB 2011, Alicante, Spain, June 28-30, 2011. – P. 246-249.

10. Strinyuk S. A., Shuchalova Y., Lanin V. Academic Papers Evaluation Software / Application of Information and Communication Technologies (AICT), 2015 9th International Conference on,14-16 Oct. 2015. Rostov-on-Don : IEEE, 2015. – P. 506-510.

Приложение А. Элементы лексико-синтаксических шаблонов

- Лексические характеристики:
  - «имя» токена (конкретная форма слова);
  - лексическая категория (e.g. «прилагательное»);
  - корень слова;
  - концептуальная категория (e.g. «человек»).
- Логические операторы OR, AND, NOT.
- Специальные символы:
  - $ – 0 или 1 токен;
  - 0 или более токенов;
  - 1 или более токенов.
- Присваивание переменной значения из компонентов шаблона: ?X =.
- Группирующие операторы: <>, [].
- Повторение:
  - * – 0 или более раз;
  - + – 1 или более раз.
- Диапазон:
  - *N – от 0 до N;
  - +N – от 1 до N.
- Необязательные конституенты: {}.

Приложение В. Описание прецедентов

Таблица В.1. Описание прецедентов

Название	Актор	Описание
Авторизоваться	Неавторизованный пользователь	Пользователь входит в свой аккаунт на портале
Зарегистрироваться	Неавторизованный пользователь	Пользователь создаёт свой аккаунт на портале
Получить рекомендации по стилю статьи	Авторизованный пользователь	Пользователь получает файл с рекомендациями по стилю статьи на основе сравнения собственной статьи и одного из корпусов
Получить аналитический отчёт по корпусу	Авторизованный пользователь	Пользователь получает файл с результатами анализа разметки одного из корпусов
Читать инструкцию	Авторизованный пользователь	Пользователь читает инструкцию к порталу
Читать форум	Авторизованный пользователь	Пользователь просматривает форум
Читать материалы и исследования	Авторизованный пользователь	Пользователь просматривает материалы и исследования, загруженные на портал
Выбрать корпус	Авторизованный пользователь	Пользователь выбирает корпус для дальнейшего анализа
Посмотреть корпус	Авторизованный пользователь	Пользователь просматривает данные корпуса и список входящих в него документов
Посмотреть документ корпуса	Авторизованный пользователь	Пользователь просматривает содержимое документа вместе с разметкой
Загрузить корпус	Авторизованный пользователь	Пользователь загружает в систему свой корпус
Найти корпус по теме	Авторизованный пользователь	Пользователь ищет корпус с помощью введенных параметров и ключевых слов
Разметить корпус	Авторизованный пользователь	Пользователь редактирует разметку (набор аннотаций) корпуса
Выбрать существующий тип аннотации	Авторизованный пользователь	Пользователь автоматически размечает корпус с помощью существующего в системе типа аннотации (которому соответствует обрабатывающий ресурс, наносящий аннотации данного типа)
Выбрать шаблон	Авторизованный пользователь	Пользователь выбирает для создания типа аннотации существующий в системе лексико-синтаксический шаблон
Сохранить разметку	Авторизованный пользователь	Пользователь сохраняет изменения, произведённые с разметкой (множеством аннотаций) в корпусе
Опубликовать разметку	Авторизованный пользователь	Пользователь разрешает одному или нескольким пользователям доступ к данной версии разметки корпуса
Выбрать типы статистик	Авторизованный пользователь	Пользователь выбирает виды статистики, которые будут рассчитаны в отчёте
Сохранить отчёт	Авторизованный пользователь	Пользователь сохраняет и/или скачивает файл с результатами анализа
Опубликовать отчёт	Авторизованный пользователь	Пользователь разрешает одному или нескольким пользователям доступ к файлу с результатами анализа

Название	Актор	Описание
Открыть статью	Авторизованный пользователь	Пользователь выбирает статью для формирования рекомендаций
Загрузить статью	Авторизованный пользователь	Пользователь загружает статью в систему
Сохранить статью	Авторизованный пользователь	Пользователь сохраняет статью в системе
Посмотреть разметку	Авторизованный пользователь	Пользователь просматривает содержимое файла вместе с разметкой
Выбрать метрику расстояния	Авторизованный пользователь	Пользователь выбирает метрику расстояния или другие параметры сравнения статьи и эталонного корпуса
Сохранить рекомендации	Авторизованный пользователь	Пользователь сохраняет или скачивает файл с рекомендациями

Приложение С. Диаграммы последовательностей

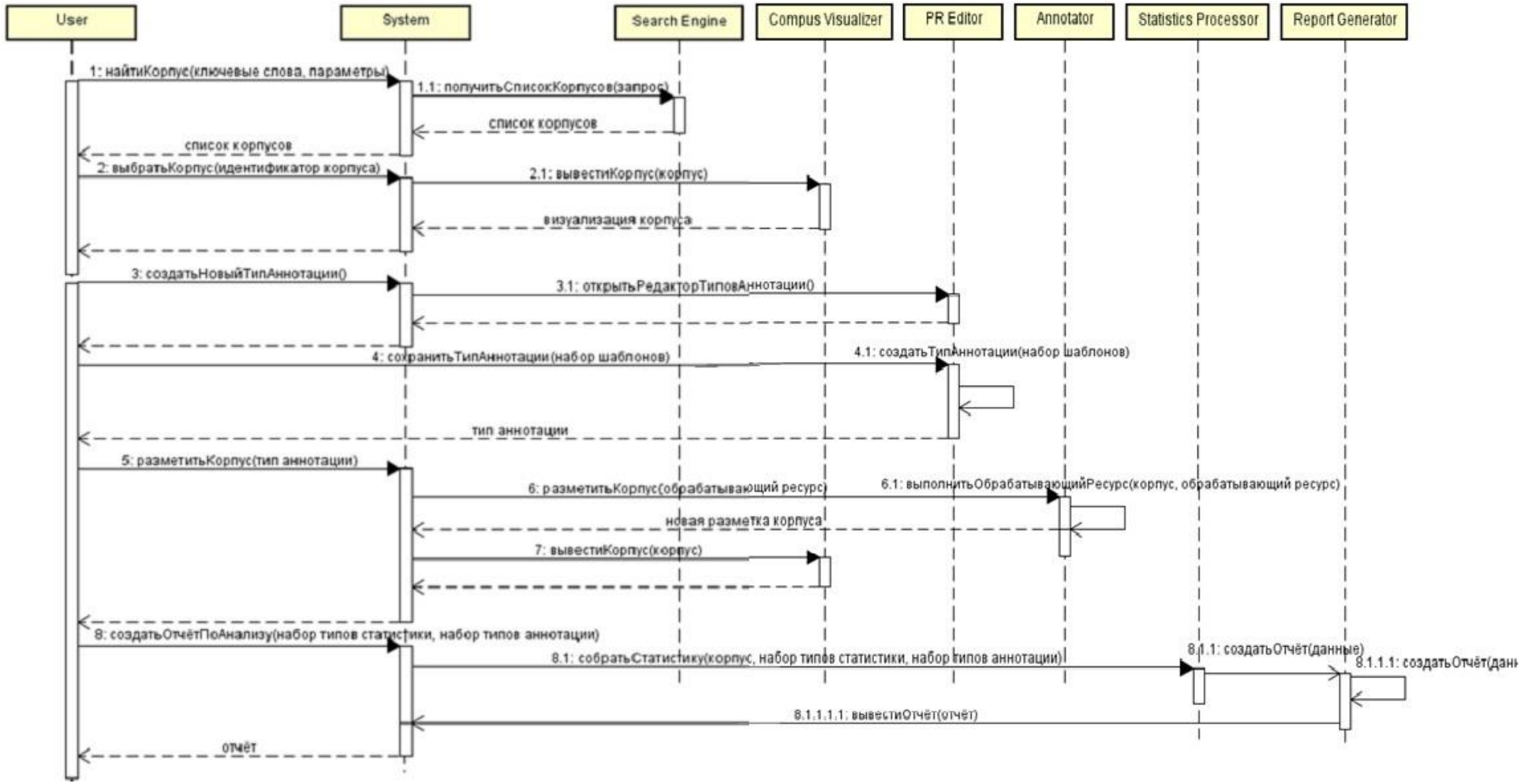

Рисунок С.1. Диаграмма последовательностей процесса анализа корпуса

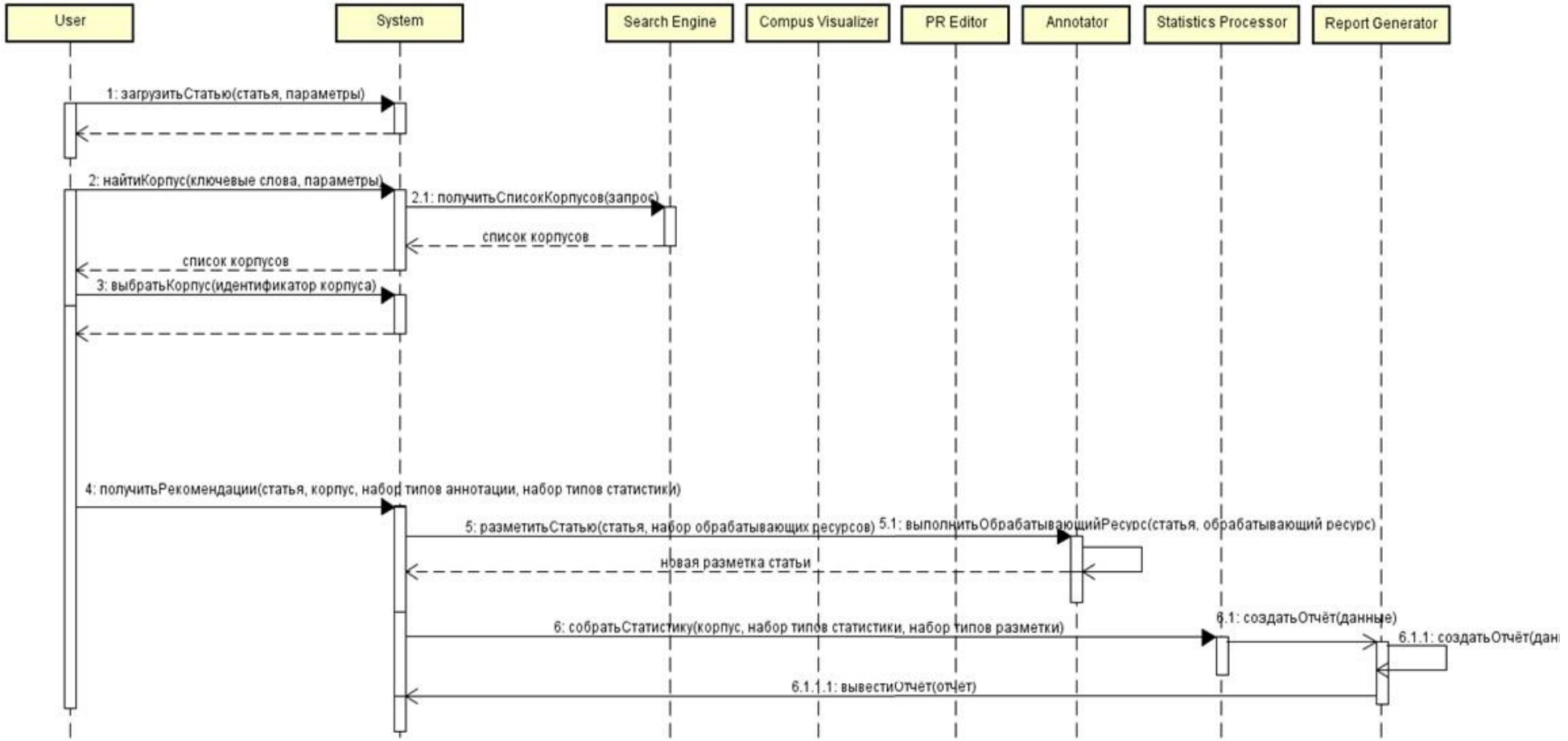

Рисунок С.2. Диаграмма последовательностей процесса формирования рекомендаций

Приложение D. Описание терминов предметной области

Таблица D.1. Описание терминов и понятий предметной области

Понятие	***Описание***	***Возможные атрибуты***
Пользователь	Пользователь портала, который может загрузить корпус или документ. Может ограничивать права доступа к загруженным материалам.	Имя, возраст, контактные данные, идентификатор пользователя
Документ	Часть корпуса или отдельно загруженная для получения рекомендации статья.	Идентификатор пользователя, источник, идентификатор документа, название, идентификатор корпуса, год
Корпус	Коллекция документов-статей, имеющая структурированную мета-информацию.	Описание, название, идентификатор пользователя, идентификатор корпуса, год
Разметка	Мета-информация о документе.	Идентификатор документа
Аннотация	Мета-информация о единице текст	Номер начального символа, номер конечного символа
Свойство аннотации	Описание свойств аннотации.	Ключ, значение
Тип аннотации	Название маркера.	Название, описание
Лексико-синтаксический шаблон	Структурированный формально описанный шаблон для автоматического выделения соответствующих конструкций в тексте элементов (см. Приложение А)	Название, описание
Список слов	Список слов, которые могут быть частью лексико-синтаксического шаблона.	Название, описание
Отчёт-анализ	Количественные показатели по соответствующим маркерам	Название, описание
Отчёт-сравнение	Рекомендация по стилю текста сравниваемого документа и выбранного корпуса в качестве эталонного	Название, описание

Приложение Е. Сравнительный анализ облачных решений

Таблица Е.1. Результат сравнительного анализа облачных платформ

Критерий	Amazon EC2	Google App Engine	Microsoft Azure
Вычислительные ресурсы предоставляемых виртуальных машин			
ROM ГБ	От 4 до 48 000.	Без ограничений	От 4 до 64 000.
RAM ГБ	От 1 до 244.	От 3,75 до 208.	От 0,768 до 448.
Количество процессоров	От 1 до 40.	От 1 до 32.	От 1 до 32.
Операционные системы на виртуальных машинах	Linux, Windows Server.	Linux, Windows Server.	Linux, Windows Server.
Хранение данных			
Реляционное хранилище. Предоставляемый размер хранилища	До 64 ТБ.	До 500 ГБ.	До 500ГБ.
Поддержка формата BLOB.	До 5 Тб размер одного объекта. Ограничений на общий объем данных нет.	До 2 ГБ размер одного объекта.	До 200 ГБ размер одного объекта. Одна учетная запись может содержать до 500 ТБ.
Разработка решений			
Developer tools	+	+	+
EclipseTools	+	+	+
VisualStudioTools	+	+ (Python)	+
Java SDK	+	+	+
Mobile SDK	iOS, Android, Fire OS.	iOS, Android.	iOS, Android, Windows Phone.
PHP SDK	+	+	+
Python SDK	+	+	+
Ruby SDK	+	+	+
.NET SDK	+	-	+

Критерий	Amazon EC2	Google App Engine	Microsoft Azure
Мониторинг состояния сервисов	Amazon CloudWatch.	Диагностическое API.	Диагностическое API.
Политика ценообразования	Плата производится по факту использования.	Плата производится по факту использования.	Плата производится по факту использования.
Пробное использование	12 месяцев, 750 часов/месяц на виртуальную машину, BLOB хранилище на 5 Гб, 20000 запросов на получение и 2000 на отправку. База данных NoSQL размером 25 Гб и 200 миллионов запросов в месяц.	300 долларов на первые 60 дней для использования всех доступных служб Google app engine.	10 000 рублей на первые 30 дней на любые службы Azure. По программе «BizSpark» можно получить 750 $ на 5 разработчиков в течение 3 лет. А также необходимо программное обеспечение от компании «Microsoft».

Приложение F. Описание компонентов системы

Таблица F.1. Описание компонентов системы

Обозначение	**Название компонента**	**Функции**
System	Компонент, отвечающий за взаимодействие с пользователем (далее Система)	Осуществление взаимодействия с пользователем на уровне интерфейса, диспетчеризация.
Search Engine	Сервис для осуществления поиска	Выполнение поиска по запросу
Corpus Visualizer	Компонент визуализации разметки	Визуализация разметки для пользователя
Annotator	Компонент разметки корпуса	Автоматическая разметка документа или корпуса, при использовании обрабатывающего ресурса
Statistics Processor	Компонент сбора статистики	Сбор данных и формирование статистики
Report Generator	Компонент формирования отчётов	Формирование отчетов