|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  Факультет экономики, менеджмента и бизнес-информатики |
|  |
| Горбунова Мария Андреевна  **Разработка системы сбора статистики использования маркеров стиля в научных публикациях на английском языке**  *Выпускная квалификационная работа*  по направлению подготовки *38.03.05 Бизнес-информатика*  образовательная программа «Бизнес-информатика»   |  |  | | --- | --- | | Рецензент  к.ф.-м.н., заведующая кафедрой математического обеспечения вычислительных систем ПГНИУ  С.И. Чуприна | Руководитель  Старший преподаватель кафедры информационных технологий в бизнесе НИУ ВШЭ-Пермь  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  В.В. Ланин | |

Пермь, 2017 год

# Аннотация

Данная работа посвящена описанию разработки программного модуля разрабатываемого на базе программы Gate Developer. Разработка предназначена, прежде всего, для сбора статистики использования маркеров академического стиля в научных публикациях на английском языке, но также имеет целью реализацию возможности генерации отчетов на основании собранных данных. При успешной реализации и тестировании данный программный модуль может применяться в сфере образования, при контроле качества обучения и академических исследованиях.

Первая глава содержит описание бизнес-процесса, автоматизируемого посредством плагина, обзор и выбор средства обработки естественного языка, на основе которого произведен и описан обзор существующих плагинов. Данная часть также содержит обзор генераторов отчетов, среди которых был произведен выбор одного для последующей интеграции в программный модуль. Во второй главе описаны процессы моделирования и проектирования программного продукта. А именно произведено формирование функциональных требований, разработка архитектуры программного продукта, а также выбраны средства для его реализации. Третья глава представляет собой описание процесса реализации программного модуля. В данном разделе описан ход разработки прототипа, а также описан его функционал. В заключении описаны основные результаты проделанной работы и значимость исследования для дальнейшей реализации продукта.

Данная разработка описывает процесс разработки инструментального средства для решения проблем компьютерной лингвистики, таким образом, ориентирована, прежде всего, на пользователей, которые занимаются исследованиями обработки естественного языка с помощью компьютерных средств, а также читателей, задействованных в академической деятельности в рамках данной сферы.

**Оглавление**

[Введение 5](#_Toc483466119)

[Глава 1. Анализ существующих систем и инструментов для обработки естественного языка 7](#_Toc483466120)

[1.1. Анализ бизнес-процесса сбора статистики и создания отчета 7](#_Toc483466121)

[1.2. Сравнительный анализ средств обработки языка 10](#_Toc483466122)

[1.2.1. Система AntConc 10](#_Toc483466123)

[1.2.2. Система WordSmith Tools 13](#_Toc483466124)

[1.2.3. Система Sketch Engine 14](#_Toc483466125)

[1.2.4. Система Gate Developer 16](#_Toc483466126)

[1.2.5. Веб-система CQPWeb 17](#_Toc483466127)

[1.3. Обзор существующих плагинов для программы Gate Developer 20](#_Toc483466128)

[1.3.1. Система ANNIE 21](#_Toc483466129)

[1.3.2. Группа компонентов OpenNLP 22](#_Toc483466130)

[1.3.3. Плагины Academic Corpus Tagger 23](#_Toc483466131)

[1.4. Обзор средств для генерации отчетов 24](#_Toc483466132)

[1.4.1. Генератор отчетов Crystal Reports Server 25](#_Toc483466133)

[1.4.2. Система Stimulsoft Reports 26](#_Toc483466134)

[1.4.3. Генератор отчетов Jasper Report 26](#_Toc483466135)

[1.4.4. Генератор отчетов Next Report 27](#_Toc483466136)

[1.4.5. Программа BIRT 27](#_Toc483466137)

[1.5. Вывод по первой главе 28](#_Toc483466138)

[Глава 2. Проектирование программного модуля для сбора статистики и генерации отчетов 30](#_Toc483466139)

[2.1. Формирование требований к плагину 30](#_Toc483466140)

[2.1.1. Требования к программному модулю от заказчика 30](#_Toc483466141)

[2.1.2. Обработанные функциональные требования 30](#_Toc483466142)

[2.2. Разработка архитектуры плагина 32](#_Toc483466143)

[2.3. Выбор средств разработки 33](#_Toc483466144)

[2.3.1. Выбор языка программирования 34](#_Toc483466145)

[2.3.2. Обзор интегрированных сред разработки 35](#_Toc483466146)

[2.4. Выводы по второй главе 38](#_Toc483466147)

[Глава 3. Разработка программного модуля 40](#_Toc483466148)

[3.1. Описание параметров входа и выхода 40](#_Toc483466149)

[3.2. Описание процесса создания плагина 42](#_Toc483466150)

[3.3. Особенности реализации плагина в среде Gate Developer 44](#_Toc483466151)

[3.3. Разработка шаблонов для генерации отчетов 46](#_Toc483466152)

[3.4. Результаты реализации программного модуля 49](#_Toc483466153)

[Заключение 51](#_Toc483466154)

[Библиографический список 52](#_Toc483466155)

[Приложение А. UseCase диаграмма для описания функциональных требований 54](#_Toc483466156)

[Приложение Б. Листинг программы 55](#_Toc483466157)

[Приложение В. Руководство пользователя по установке и настройке программного модуля StatsCollector 63](#_Toc483466158)

# Введение

Корпусная лингвистика является одним из наиболее перспективных и востребованных направлений в области исследований естественного языка. Данная наука занимается разработкой, созданием и использованием, так называемых текстовых корпусов, состоящих из документов, объединённых по какому-либо признаку. С помощью корпусов лингвисты решают широкий спектр задач, например, подсчитывают статистику использования различных языковых единиц, представление лингвистических данных, реализация математического анализа текста и т.д.

Проведение анализа текста без использования компьютерных инструментальных средств – это достаточно затратный по времени и силам процесс, поэтому необходимо использовать программные инструментальные средства. На данный момент существует большое количество инструментов для обработки корпусов текстов. Наиболее распространенными являются AntConc[1], WordSmith Tools[2], Gate Developer[3], Sketch Engine[4] и CQPweb[5], каждая из которых предоставляет пользователю возможность хранить и редактировать корпуса текстов. Однако, вышеупомянутые средства имеют ряд недостатков, например, подобные системы имеют недостаточные средства сбора статистики, так как они не адаптируемы к потребностям пользователей. Но необходимо отметить, что подобный функционал должен быть в системах такого рода, так как с появлением корпусов, объёмы исследуемых данных в разы увеличились, и ручная обработка стала весьма трудозатратной.

Таким образом, статистическая обработка языка ускоряет исследование, а также позволяет с помощью методов математической статистики подтвердить или опровергнуть гипотезу о каком-либо языковом явлении. Из чего следует, что данная работа актуальна и является решением проблемы недостаточного функционала систем обработки корпусов текстов.

Одним из инструментов для обработки корпусов текстов является программа Gate Developer. Данная программа позволяет создавать и изменять правила по обработке текстов. Стоит отметить, что данный инструмент широко применяется для обработки корпусов текстов. Программа является бесплатной и предоставляется с открытым исходным кодом, что очень удобно для дальнейшего расширения функционала. Так же, как и в системах упомянутых выше, в программе Gate Developer нет инструментария, позволяющего генерировать отчеты о собранной статистике.

Объектом исследования в данной работе выступают инструментальные средства обработки текстовых корпусов. Тогда как предметом – средства сбора статистической информации при обработке корпусов текстов, встроенные в инструментарий.

Таким образом, целью данной работы является разработка средства сбора статистики на базе программы Gate Developer для отображения информации о корпусе текстов, обладающего такими свойствами как генерация отчетов о собранных данных, на основе параметров, введенных пользователем, и формирование структуры отчета.

Для достижения данной цели необходимо выполнить ряд задач:

1. Провести анализ существующих средств обработки лингвистических корпусов. Выявить основные достоинства и недостатки рассмотренных программ. Рассмотреть примеры реализации плагинов в программе Gate Developer. Рассмотреть средства генерации отчетов.

2. Сформулировать функциональные требования к разрабатываемому продукту. Выбрать инструментальные средства реализации.

3. Разработать архитектуру модуля для сбора статистики.

4. Разработать плагин для сбора статистики и базовые отчетные формы.

Для исследования объекта необходимо применить такие методы как:

* моделирование информационных систем (для моделирования плагина и его архитектуры);
* формализация (при представлении требований к продукту);
* статистика (для сбора статистических данных в корпусах текстов);
* вероятностно-теоретические методы (при расчёте дополнительных показателей, выводимых в отчете по сбору статистики).

Результатом выполнения данной работы должен быть плагин, позволяющий представить пользователю определенный набор статистики в виде отчета.

Данная работа состоит из 3 глав. Первая глава содержит в себе сравнительный анализ существующих средств обработки языка, а также анализ реализованных плагинов для программы Gate Developer. Во второй главе формируются требования к разрабатываемому плагину, описывается выбор средств для его разработки, а также разрабатывается архитектура плагина. Третья глава является описанием реализации программы, ее работы и первичного тестирования.

# Глава 1. Анализ существующих систем и инструментов для обработки естественного языка

В данной главе приводится анализ бизнес-процесса сбора статистики и создания отчета и анализ существующих на данный момент инструментов для обработки естественного языка. На основе информации полученной в результате анализа будут выбраны инструменты для интеграции программного модуля.

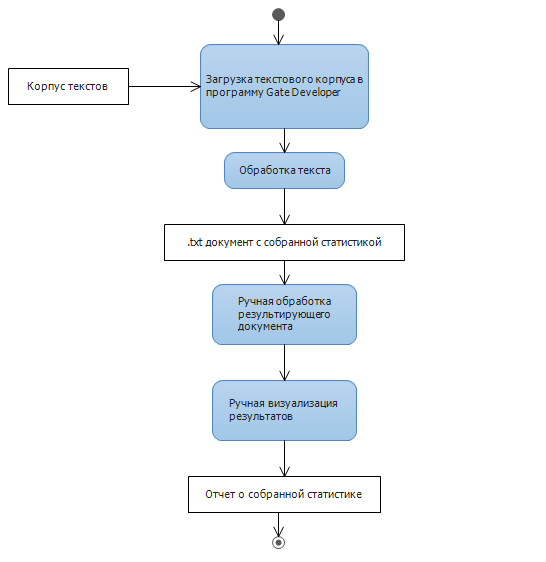
## Анализ бизнес-процесса сбора статистики и создания отчета

Для разработки программного обеспечения необходимо тщательно рассмотреть процесс, который автоматизируется с помощью программы. В рассматриваемом случае это бизнес процесс обработки текстового корпуса и генерации отчета о собранной статистике. С помощью анализа бизнес-процесса можно определить его «слабые места» и вывести функциональные требования к программному модулю.

Язык моделирования UML помогает графически описать бизнес-процесс, модель системы или организационную структуру [1]. В данном стандарте существует такой тип диаграмм как Activity Diagrams, предназначенные для описания последовательности действий необходимых для достижения конкретной цели. В данном случае этой целью является сгенерированный отчет.

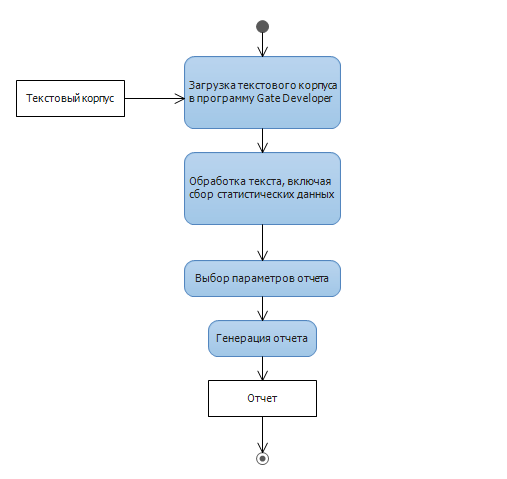
Для рассматриваемого бизнес-процесса были разработаны две диаграммы активностей – AS-IS и TO-BE. Первая показывает, как процесс функционирует в данный момент, вторая, соответственно, как он должен работать, используя плагин. Сравнительный анализ данных диаграмм должен помочь сформулировать требования к разрабатываемому программному модулю.

Диаграмма AS-IS, представленная на рисунке 1.1, содержит двух актеров: пользователя и систему Gate Developer. Первое что делает пользователь, это загружает текстовый корпус в систему Gate Developer, используя пользовательский интерфейс программы. Далее проходит обработка текста, согласно задачам пользователя, после чего система генерирует \*.txt документ с данными о частоте употребления элементов аннотированного текста. После чего, пользователь вручную вычленяет необходимые данные из полученного документа и создает отчет с помощью каких-либо текстовых или табличных редакторов. В результате, выходом процесса является отчет, составленный и оформленный по требованиям пользователя. Стоит отметить, что процесс мог занять большое количество времени и потребовать определенных усилий пользователя, не говоря о том, что многие показатели могли быть не подсчитаны. Таким образом, можно сделать вывод, что процесс на данном этапе функционирует не эффективно, так как является затратным по времени, а также требует частого вмешательства пользователя.



*Рисунок 1.1. Диаграмма активностей для бизнес-процесса "AS-IS"*

Рассмотрим диаграмму TO-BE, которая отражает вариант преобразования процесса, используя программный модуль (см. рис. 1.2). Как можно увидеть на диаграмме, теперь пользователь принимает меньшее участие в функционировании процесса. Точкой входа также является загрузка пользователем корпуса текстов. Далее происходит обработка текстов, но теперь с использованием плагина, для сбора статистических данных. Далее программный модуль должен представить интерфейс, с помощью которого пользователь программы может установить определенные параметры отчета, такие как список отображаемых пунктов и настройки отображения и оформления. Плагин производит обработку результатов и генерирует отчет, в соответствии с установленными требованиями. Выходным параметром данного процесса также является отчет, соответствующий запросам пользователя, но по сравнению с процессом AS-IS он требует меньшего вмешательства пользователя и, как следствие, меньших трудозатрат.



*Рисунок 1.2. Диаграмма активностей для бизнес-процесса "TO-BE"*

В результате моделирования процессов AS-IS и TO-BE, а также их сравнительного анализа, было выявлено несколько требований предъявляемых разрабатываемому модулю:

1. Сбор статистических данных и подсчет стандартных показателей.
2. Возможность предоставления пользовательского интерфейса для управления генерацией отчета.
3. Генерация отчета согласно пользовательским параметрам.

Прежде чем приступать к моделированию программного модуля и его реализации необходимо рассмотреть рынок инструментов для обработки текстовых корпусов и проанализировать реализованные решения. Данный шаг может помочь, точнее определить проблему, а также ознакомиться с различными подходами к решению данной задачи.

## Сравнительный анализ средств обработки языка

На данный момент реализовано огромное количество систем по обработке естественного языка. Для выяснения необходимого функционала для разрабатываемого программного модуля, необходимо провести обзор систем-аналогов, выявить существующие средства для сбора статистики и генерации отчетов, и каким образом они реализованы.

Для сравнительного анализа систем были сформулированы следующие критерии:

1. Возможность генерации отчетов.
2. Поддержка различных форматов отчетов.
3. Список параметров, заносимых в отчет.
4. Формирование пользовательских отчетов.
5. Возможность расширения функционала, а именно встраивания дополнительных программных модулей в систему.
6. Язык разработки.

Проанализируем наиболее популярные системы обработки естественного языка.

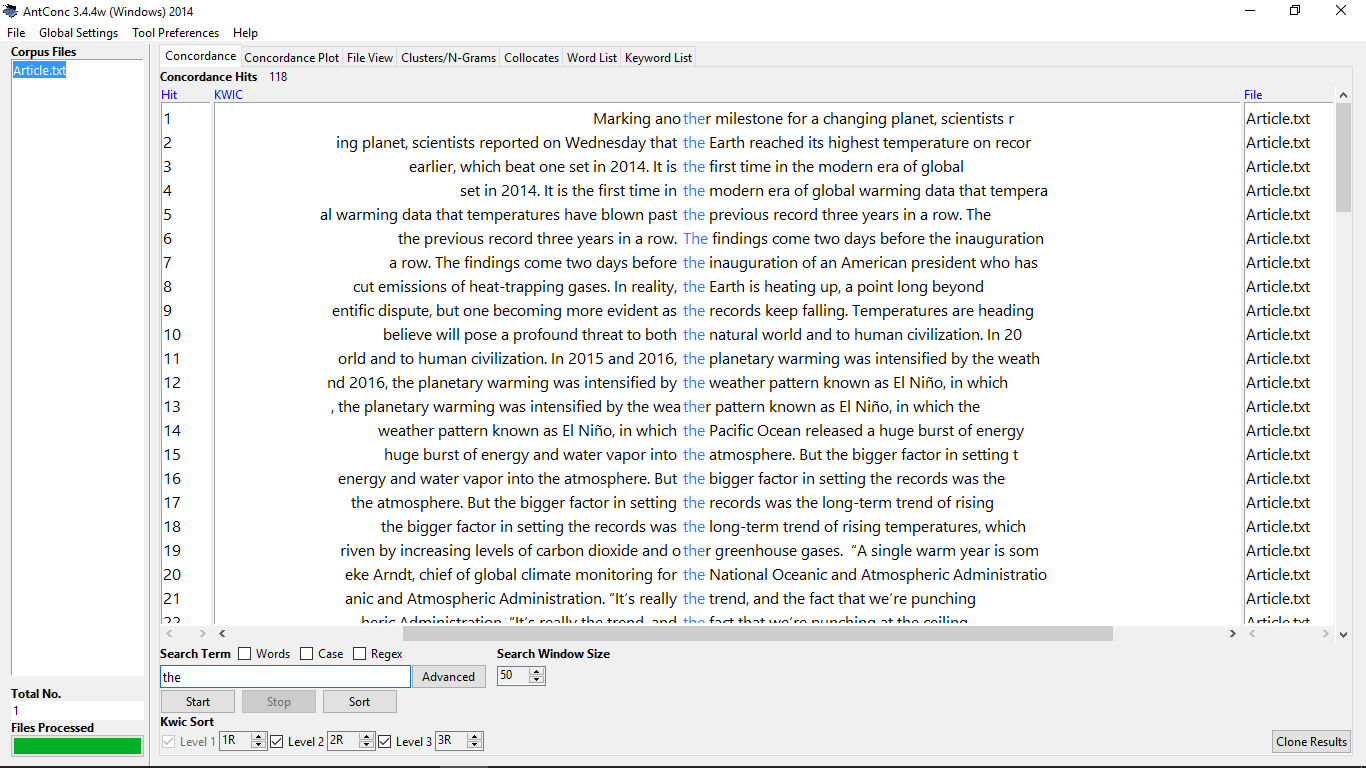
### Система AntConc

Программа является настольным приложением, подходит под такие операционные системы как MS Windows, Linux и Mac. Приложение предоставляется бесплатно на официальном сайте [1] и не требует установки.

Приложение AntConc предназначено для многоцелевой обработки текста. Система содержит в себе 7 инструментов:

1. Конкорданс. Показывает, как ключевое слово используется в разных контекстах исследуемого текста.
2. График конкорданса. График позволяет увидеть, какие файлы содержат искомое ключевое слово.
3. Просмотр файлов. Предоставляет возможность просмотреть оригинальный файл.
4. Кластеры. Позволяют просмотреть упорядоченный список кластеров, которые появляются в процессе поиска.
5. Расположение. Показывает расположение искомого элемента в тексте.
6. Список слов. Данный инструмент производит подсчет всех слов в корпусе и представляет упорядоченный список, с помощью которого можно посмотреть наиболее употребляемые элементы.
7. Список ключевых слов. Инструмент создает «Список слов» для ключевых слов, запрашиваемых пользователем.

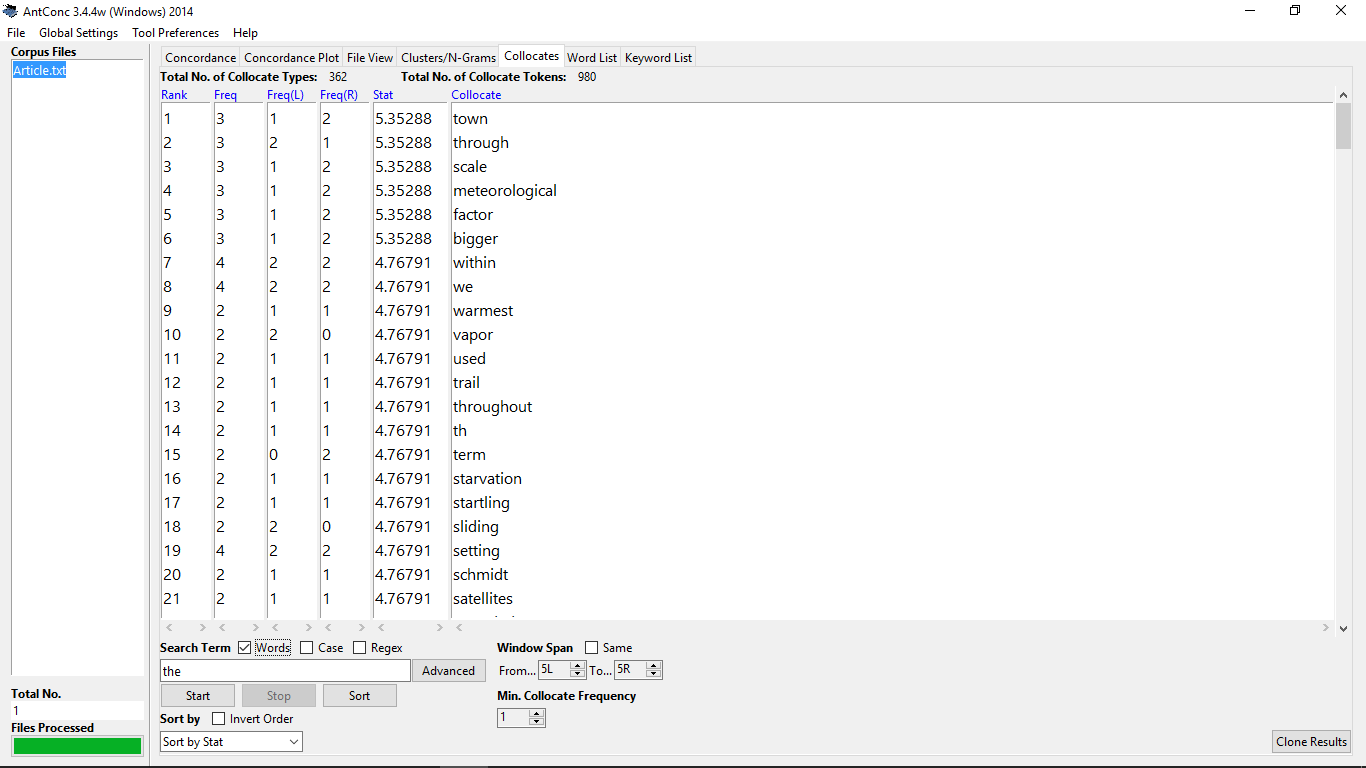
Используя описанные инструменты, лингвист может провести исследование корпуса, найти языковые единицы и сравнить частотности их использования [3]. Домашняя страница программы показана на рисунке 1.3.



*Рисунок 1.3. Домашняя страница системы AntConc*

После обработки текста, программа отображает данные в виде таблицы. Пользователю предоставляется возможность выгрузить результаты обработки текста в файлы нескольких форматов (\*.txt,\*.text, \*.html,\*.xml, \*.ant) [3, 19] .Стоит отметить, что отображаемые параметры настроить нельзя, они определены в приложении. Содержание отчета зависит от того, какой раздел открыт в окне на данный момент. Несмотря на то, что данные представляются в табличном виде, программа не позволяет загрузить их в MS Excel.

Подсчет статистики в данной системе представлен в разделе «Collocates», который представляет собой таблицу с информацией о частотности использования искомого слова в сочетании с каким-либо словом/сочетанием букв. Интерфейс данного раздела представлен на рисунке 1.4.



*Рисунок 1.4. Представление статистики в системе AntConc*

Система реализована на языке программирования Perl и поддерживает дальнейшее расширение функционала. Стоит отметить, что создание пользовательских отчетов система не поддерживает.

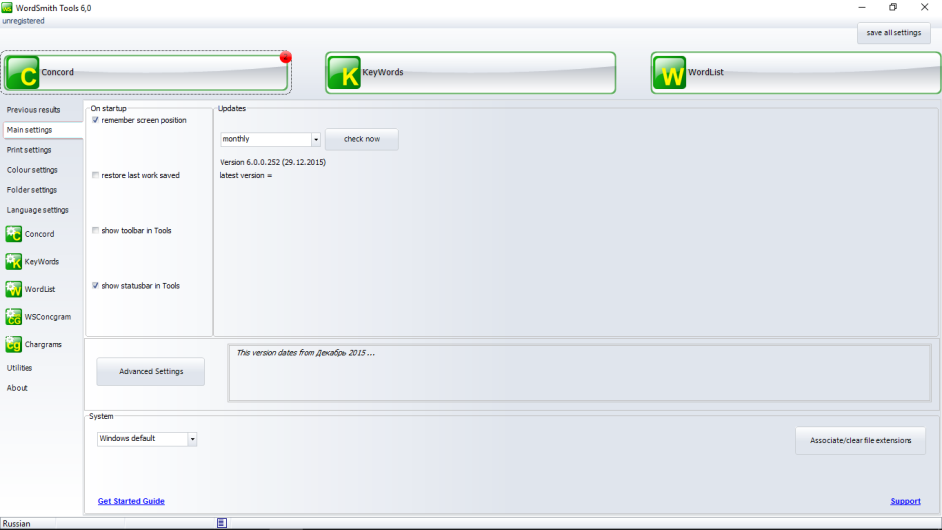
### Система WordSmith Tools

Приложение «WordSmith Tools» [18] является настольным, платным, требующим установки программным обеспечением, которое ориентировано на многоцелевой анализ корпусов текстов. Также существует бесплатная версия данного приложения с ограниченным функционалом и сроком работы. Стоит отметить, что система работает только под MS Windows, что является существенным недостатком в сравнении с другими системами [20]. Программа реализована на языке программирования Python и не поддерживает расширения функционала.

Пользователю доступны для работы три компонента:

1. Компонент WordList, который позволяет формировать список отдельных слов и кластеров слов, также отсортировать их по алфавиту или частоте.
2. Компонент KeyWords позволяет построить список ключевых слов.
3. Компонент Concord, используется для построения списков совпадений в контексте по искомому слову.

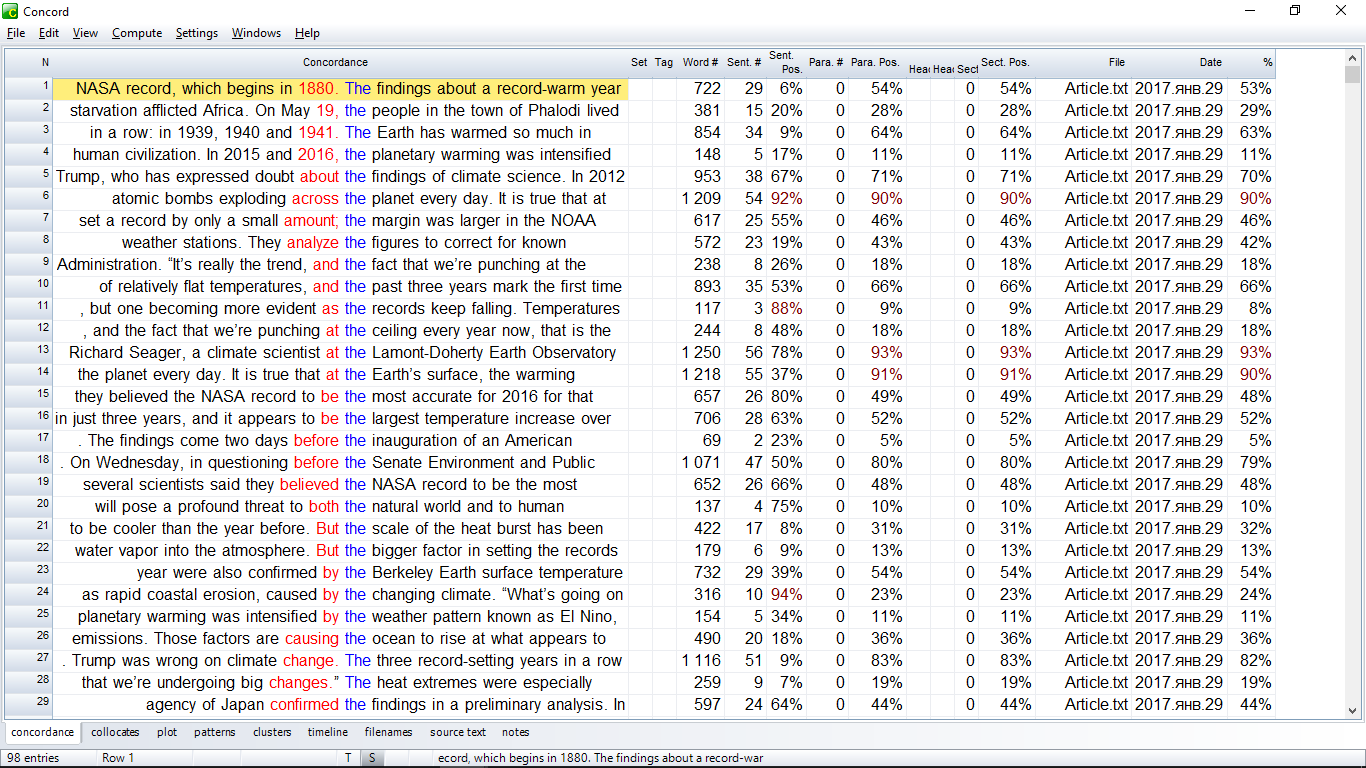
Загрузить тексты можно в форматах TXT, PDF, DOC, DOCX, HTML, XML или SGML [21]. Домашняя страница системы представлена на рисунке 1.5.



*Рисунок 1.5. Домашняя страница системы WordSmith Tools*

В приложении есть возможность сохранить результаты в форматах \*.xml, \*.txt, Excel spreadsheet и Concordance list. В отчет заносятся стандартные атрибуты – слово, частота употребления, источник, словарная форма и набор документов.

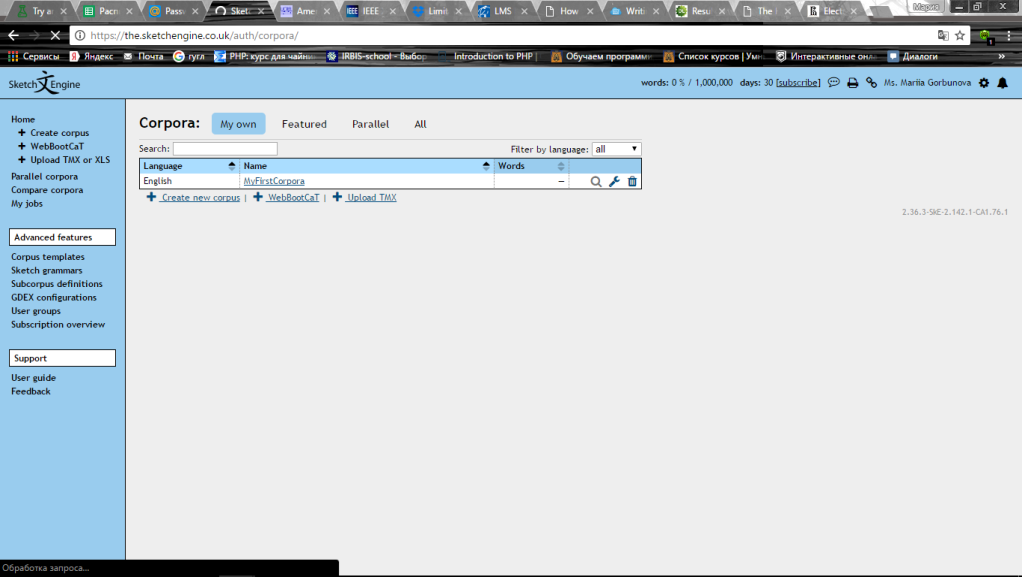
Статистика, собранная с анализируемого документа показана в разделе Concord (см. рис. 1.6). Как и в системе AntConc, WordSmith представляет результаты обработки в табличном виде с различными показателями. Преимуществом данной системы является то, что представленные результаты можно экспортировать в формате XLS. Тем не менее, добавлять пользовательские параметры в отчеты нельзя.



*Рисунок 1.6. Раздел Concord в системе WordSmith Tools*

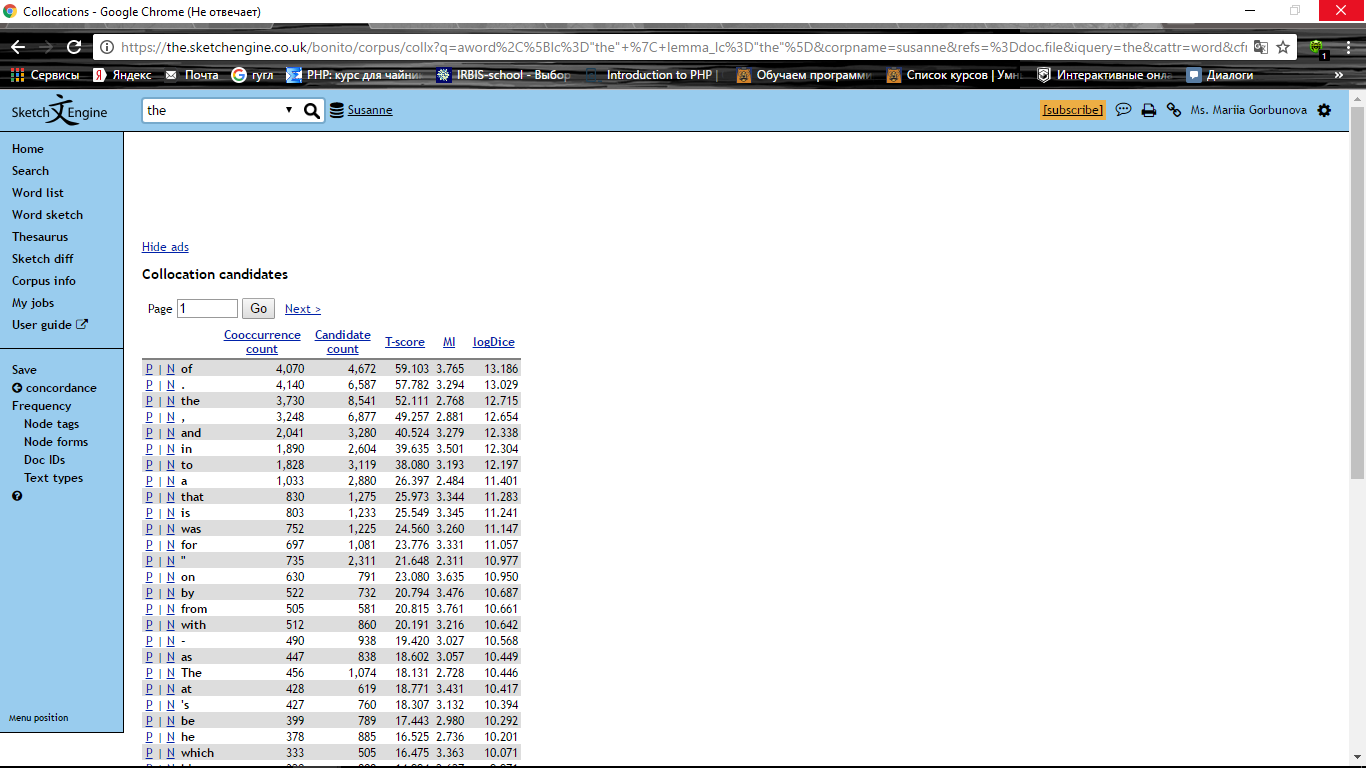
### Система Sketch Engine

Система «Sketch Engine» [22] – это веб-система, которая позволяет изучать текстовые корпуса и извлекать из них нужную информацию. Инструмент предоставляет бесплатную пробную версию на 30 дней с возможностью использования полного функционала (рис 1.7). После окончания пробного периода, пользователю предлагается выбрать несколько видов платных подписок, с разными наборами функций. Программный код системы использует такие языки программирования как C++, Python и JavaScript. Также возможно расширение функционала системы за счет подключения новых программных компонентов.

******

*Рисунок 1.7. Домашняя страница Sketch Engine*

Система поддерживает загрузку текстовых корпусов разных форматов (.doc, .docx, .htm, .html, .pdf, .ps,. tar.bz2,.tar.gz, .tei, .tgz, .tmx, .txt, .vert, .xml, .zip.) [22]. Для анализа текстов пользователю доступен большой спектр инструментов, начиная от обычного поиска слова в тексте, заканчивая специальными фильтрами для поиска предложений по определенной схеме. В системе не существует возможности для сохранения полученных результатов на компьютер пользователя. Статистика использования языковой единицы в текстовом корпусе представляется в виде таблицы, которая представлена на рисунке 1.8. То есть обработка конечных результатов сводится к копированию представленной таблицы.

******

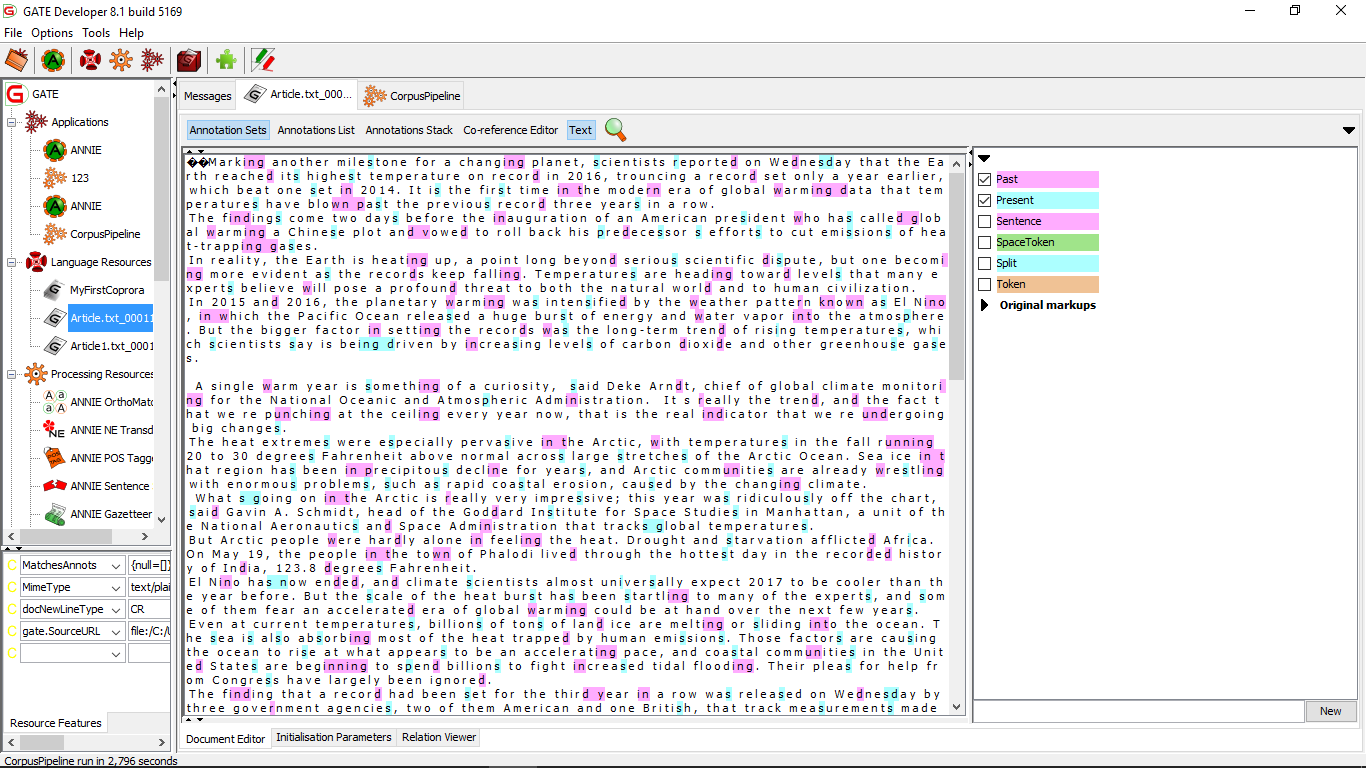
*Рисунок 1.8. Таблица со статистикой Sketch Engine*

Инструмент предоставляет в распоряжение пользователя достаточно широкий спектр возможностей для подсчёта математических и статистических показателей. В то же время выгружать результаты обработки текста не представляется возможным, что является существенным недостатком по сравнению с аналогами.

### Система Gate Developer

Gate Developer – это бесплатное настольное приложение, требующее установки, которое позволяет создавать, изменять и обслуживать программные компоненты по обработке языка [2]. Программа предоставляется с открытым исходным кодом и имеет множество примеров реализованных программных модулей.

Сильной стороной системы является аннотирование текста и дальнейшее его изучение на основе полученных аннотаций. Маркеры для разметки текста могут быть стандартными, а могут задаваться пользователем. Подсчет статистических данных программа не осуществляет, но подготавливает хорошую базу для подобных вычислений. Пример аннотированного документа представлен на рисунке 1.9.

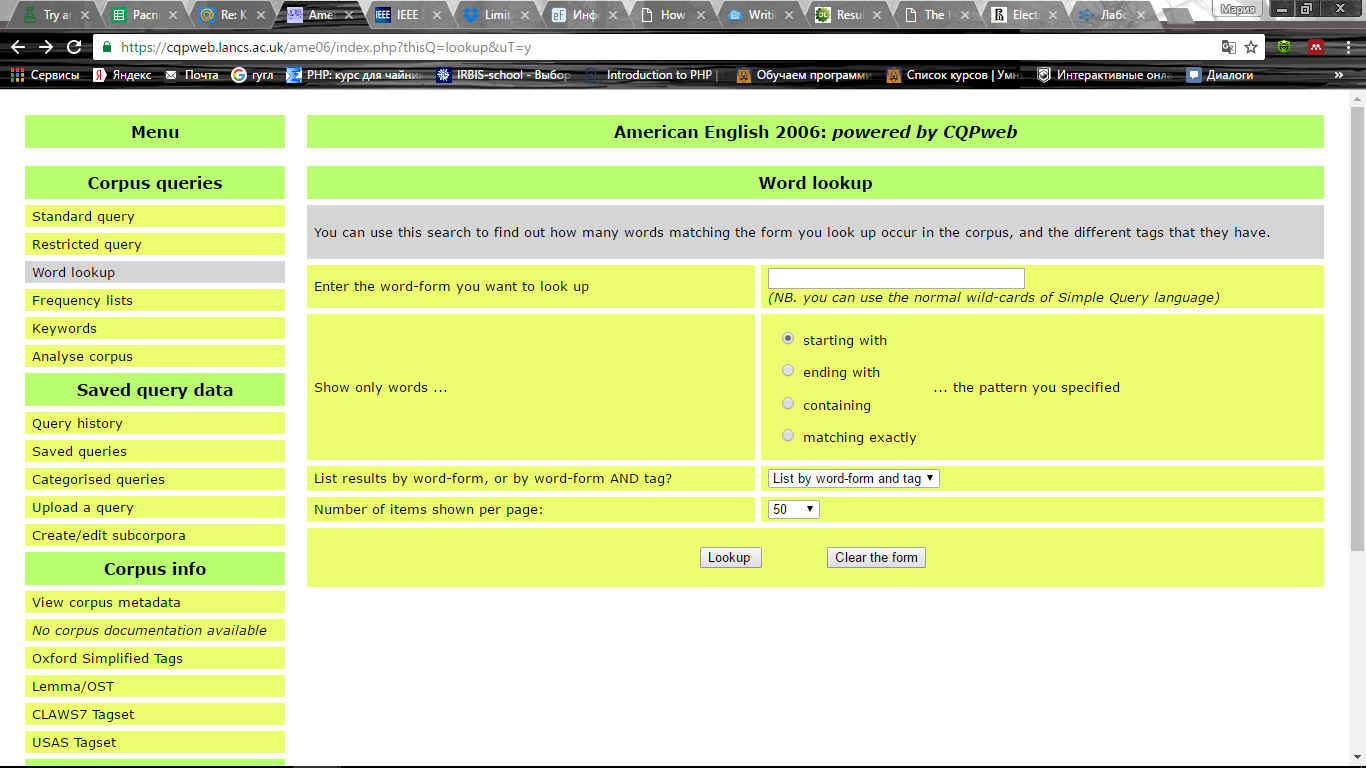


*Рисунок 1.9. Домашняя страница Gate Developer*

Пользователю предлагается производить обработку текста, используя три компонента: языковые ресурсы, обрабатывающие ресурсы и приложения. Приложения, являются результатом объединения двух других компонентов – обрабатывающие ресурсы применяются к лингвистическим. В качестве языковых ресурсов могут быть использованы документы разных форматов, таких как TXT, Plain Text, HTML, SGML, RTF, PDF, Microsoft Office, OpenOffice [4], а также дополнительные форматы документов, которые доступны посредством плагинов. Эти документы в свою очередь могут быть объединены в корпуса, а корпуса в хранилища данных (DataStore). Система позволяет выгружать XML файлы с аннотированными текстами, но не имеет возможности генерировать отчеты.

### Веб-система CQPWeb

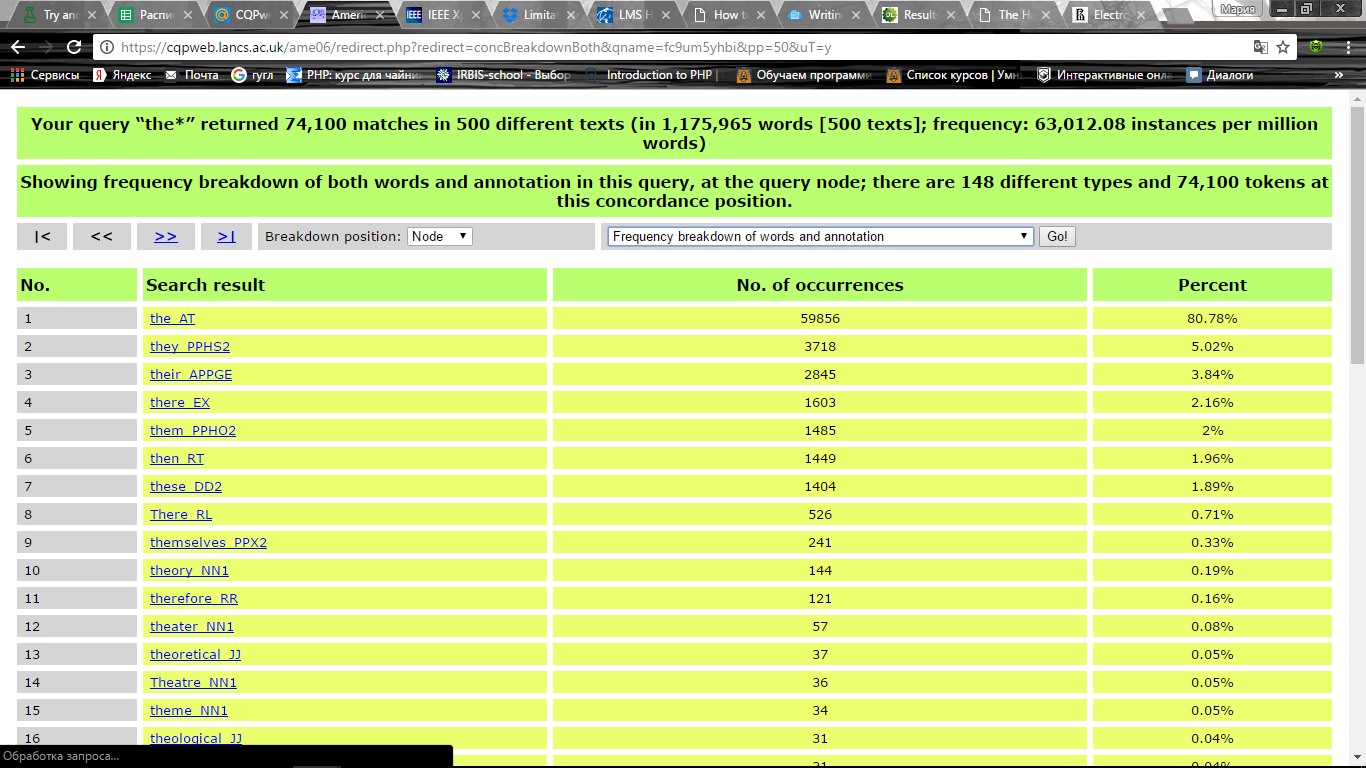
CQPWeb – это веб-система, которая позволяет пользователям исследовать большие корпуса текстов для получения какой-либо информации [17]. Данный ресурс является бесплатным, но обладает большим недостатком – он не позволяет пользователю загружать собственные корпуса текстов. Для изучения доступны 45 встроенных корпусов на различных языках. Внешний вид домашней страницы системы представлен на рисунке 1.10.



*Рисунок 1.10. Домашняя страница CQPWeb*

Система позволяет пользователю выгрузить результаты обработки текста на свой компьютер в формате «\*.txt». В данном отчете можно увидеть результаты выполнения пользовательского запроса, а именно четыре столбца: номер совпадения, само совпадение, количество подобных совпадений и процентное представление.

После того как система проанализирует предоставляемые корпуса на предмет искомого слова, на экране появляется таблица с данными статистики (см. рис 1.11).



*Рисунок 1.11. Страница с собранной статистикой CQPWeb*

Пользователь может выполнить следующие действия с данными таблицы:

1. Создать новый запрос данных.
2. Сузить границы поиска.
3. Упорядочить по частотности.
4. Распределить по группам.
5. Отсортировать.
6. Просмотреть расположение слов в тексте.
7. Скачать результаты поиска.
8. Категоризировать результаты поиска.
9. Сохранить текущее число совпадений поиска.

Система позволяет выгружать результат обработки текста, в формате TXT, который представляет собой табулированный файл с данными. Пользовательские параметры добавлять в отчет также не предоставляется возможным, но доступна возможность выбора нескольких параметров отображения, таких как метаданные, название файла, наименования колонок и т.д. Для реализации системы были использованы языки программирования Perl, Python и R. Стоит отметить, что расширение функционала системы является возможным, за счет подключения новых программных модулей.

Описанные выше сведения удобно представить в виде таблицы (см. табл. 1.1).

*Таблица 1.1. Сравнительная таблица средств обработки текста*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название системы | Генерация отчетов | Форматы экспорта | Параметры отчета | Пользовательские отчеты | Расширение функционала | Язык разработки |
| AntConc | + | + (.txt, .text, .html,.htm, .xml, .ant) | Hit, KWIC, File | - | + | Perl |
| WordSmith Tools | + | + (.txt, .xml, .xls, .rtf, WordList files) | Word, Frequency, Texts, Lemmas, Set | - | - | Python |
| Sketch Engine | - | + (.txt, .xml) | Word, Co-occurrence count, Candidate Count, T-score, MI, logDice | - | + | C ++ , Python , JavaScript , JQuery |
| Gate Developer | - | + (.xml) | Type, Set, Start, End, ID, Features | - | + | Java |
| CQPWeb | + | + (.txt) | No, Search Results, No. Of occurrences, Percent | - | + | Perl, R, PHP |

По итогам анализа систем обработки естественного языка, можно сделать ряд выводов. Ни одна из рассмотренных систем не позволяет пользователю вносить собственные параметры в отчет. Некоторые из систем не предоставляют возможности генерировать отчеты, также, как и не во всех реализована функция сбора статистики. В большинстве из рассмотренных систем отчет представляет собой файл в формате «\*.txt», с ограниченным набором параметров. Данный формат является универсальным для дальнейшей обработки, но не обеспечивает пользователю четкого графического отчета о произведенной обработке текста. Одна из систем, WordSmith Tools, позволяет отобразить отчет посредством MS Excel, что является преимуществом перед другими инструментами.

Таким образом, стоит отметить, что возможность генерировать пользовательские отчеты с графическим отображением полученных данных не реализована ни в одной из рассмотренных систем, тем не менее, подобная функция является полезной при представлении результатов обработки текста.

## Обзор существующих плагинов для программы Gate Developer

Разработка программного модуля производится на базе системы Gate Developer, поэтому необходимо рассмотреть плагины, реализованные в рамках данного инструмента. На данный момент в системе Gate Developer существует 72 встроенных и 14 подключаемых плагинов [2]. Нет необходимости рассматривать каждый из них, в рамках данной работы целесообразно проанализировать программные модули, связанные со сбором статистических данных, генерацией отчетов или выгрузкой аннотированного текста для обработки.

Корпусная обработка текстов в рамках системы GATE сосредоточена на трех видах ресурсов:

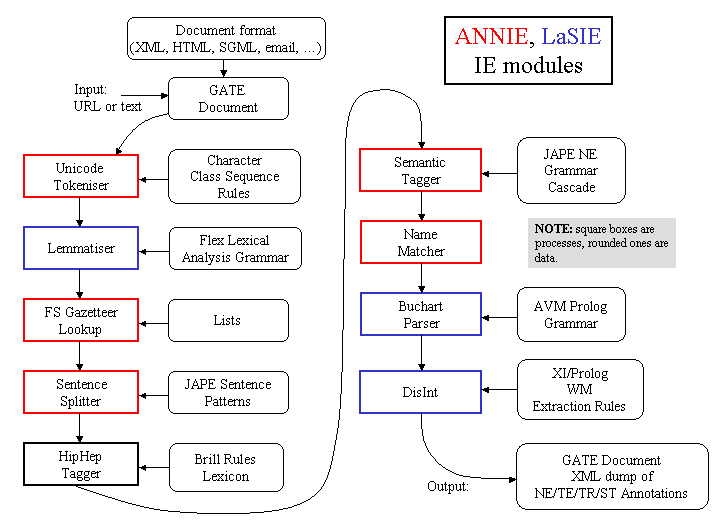
* Лингвистические ресурсы (LRS, Language Resourсes), это такие компоненты как текстовые корпуса, словари, онтологии корпусов и т.д.
* Процессные ресурсы (PRS, Processing Resources), объекты, которые представляют собой алгоритмы обработки текстов, анализаторы, генераторы.
* Визуальные ресурсы (VRS, Visual Resources), представляют собой компоненты визуализации и редактирования.

Все ресурсы, интегрированные в GATE, составляют CREOLE (a Collection of REusable Objects for Language Engineering), в который также будет добавлен программный модуль по сбору статистики. В набор плагинов CREOLE входит система ANNIE, которая предназначена для извлечения текстовой информации [2].

### Система ANNIE

Данный инструмент работает в режиме конвейера, схема которого изображена на рисунке 1.12. Система ANNIE включает в себя следующие возможности:

* Токенизация (ANNIE English Tokenizer), разбиение текста на более мелкие части, токены.
* Разметка частей речи (ANNIE POS-Tagger), автоматическая морфологическая разметка, в ходе которой словам в тексте приписываются тэги в соответствии с частью речи и грамматическими характеристиками [5].
* Разбивка на предложения (ANNIE Sentence Splitter).
* Извлечение именованных сущностей (ANNIE Gazetteer, ANNIE NE Transducer).
* Анализ кореферентности (ANNIE OrthoMatcher), отношения между частями высказывания указывающими на один и тот же объект [6].



*Рисунок 1.12. Схема функционирования компонента ANNIE*

В системе ANNIE можно выделить следующие ключевые компоненты:

* Document Reset PR, удаляет разметку со всего текста, кроме тех параметров которые пользователь отметил как «setToKeep».
* Tokenizer, используется для упомянутой выше токенизации.
* Gazetteer, выделяет именованные сущности в тексте, которые представляют собой названия городов, организаций, дней недели и т.д.
* Sentence Splitter, делит текст на предложения, разделяя их на типы в зависимости от знака в конце предложения (вопросительные, утвердительные, восклицательные).
* RegEx Sentence Splitter, также как и предыдущий инструмент, делит текст на предложения, но уже используя JAPE, преобразователь регулярных выражений.
* Part of Speech Tagger, используется для упомянутой выше разметки частей речи.
* Semantic Tagger, производит семантическую аннотацию текста используя Jape.
* Orthographic Coreference, используется для анализа кореферентности, описанного выше.
* Pronominal Coreference, находит цитаты, прямую речь, местоимения и т.д.

Группа компонентов ANNIE используется для аннотации неструктурированного текста. Данный инструмент является незаменимым в рамках системы GATE при первичной обработке текстового корпуса. Применение плагинов ANNIE значительно упрощает дальнейшую работу с корпусами.

### Группа компонентов OpenNLP

Данный инструмент представляет собой группу Java-компонентов, которые предназначены для выделения выражений, токенизации, разметки частей речи, чанкинга, парсинга, определения именованных сущностей и кореферентности.

В набор компонентов OpenNLP входят:

* OpenNLP Tokenizer;
* OpenNLP Sentence Splitter;
* OpenNLP POS Tagger;
* OpenNLP Chunker;
* OpenNLP Parser;
* OpenNLP NER.

Стоит отметить, что многие инструменты из набора OpenNLP, имеют схожие с плагинами ANNIE названия и функции. Рассмотрим те, которые не представлены в системе ANNIE:

OpenNLP Chunker – это инструмент для «чанкинга», который является частичным синтаксическим разбором, разбиением текста на синтаксически связные фрагменты [8].

OpenNLP Parser позволяет выполнить синтаксический разбор. Инструмент необходимо применять к уже аннотированному тексту. В качестве результата плагин представляет результаты парсинга аннотации в виде синтаксического дерева.

### Плагины Academic Corpus Tagger

Academic Corpus Tagger – это набор плагинов, предназначенных для разметки текста в соответствии с маркерами научного стиля академического английского языка [7]. В состав данного инструмента входят следующие плагины:

* Academic Tagger Annotations Counter, счетчик аннотированных тегов;
* Annotations Deleter, инструмент для очищения текста от уже реализованных аннотаций;
* BeHaveTagger, плагин, который выделяет глаголы *be* и *have*, а также проверяет слова стоящие за ними на предмет морфологических характеристик;
* ComplexConjExtractor, выделяет двойные союзы, отделяя их от схожих конструкций;
* PrePostPositiveExtractor, компонент помогающий определить существительные с определениями, стоящими до или после слова;
* TheTheConjunction, выделяет двойной союз *the*;
* VerbTenseTagger, аннотирует временные глагольные формы (*Past, Present, Future*);
* WordlistFinder, выделяет в тексте точные совпадения или совпадения начальной формы слов представленных в списке, который служит входным параметром плагина;
* WordlistSplitFinder, производит тоже действие что и WordlistFinder, но со словосочетаниями состоящими из 2 частей разделенных «…»;
* WordlistSuffixFinder, выделяет слова соответствующие суффиксам или частям речи указанным как критерии поиска во входном параметре-списке;
* WordlistVerbTagger, выделяет в тексте указанный список глаголов.

Описанный набор плагинов является наиболее важным в рамках данной работы, так как обладает рядом функций, которые будут необходимы при разработке и тестировании плагина для сбора статистики.

По итогам рассмотрения существующих плагинов в системе Gate Developer, можно сделать ряд выводов. Во-первых, в системе реализовано достаточно большое количество пользовательских плагинов, следовательно, существует множество примеров и пособий по этому процессу. Во-вторых, многие плагины выполняют функции, которые необходимы в ходе процесса сбора статистических данных, а значит заново реализовывать их не обязательно. В-третьих, вследствие того, что разработчики постоянно реализуют компоненты для системы Gate, проектная команда Gate создала набор инструментов и методических пособий для разработчиков, которые значительно упрощают процесс создания новых инструментов. Например, Gate Developer, Gate Embedded и Gate Cloud позволяют программистам ускорить разработку продукта, а также повысить его качество, за счет автоматизации данного процесса.

Стоит отметить, что среди рассмотренных инструментов нет ни одного предоставляющего доступ к генератору отчетов, таким образом, необходимость данной разработки снова подтверждается. Для того чтобы встроить генератор отчетов в разрабатываемый инструмент необходимо проанализировать системы, представленные на рынке и выбрать ту, которая максимально удовлетворяет критериям поставленной задачи.

## Обзор средств для генерации отчетов

Проанализировав средства для обработки языка было выявлено, что ни одно из них, включая Gate Developer, не предоставляет пользователю возможность сгенерировать отчет. Данный факт является показателем того что системы недостаточно доработаны, так как отсутствие генерации отчетов — это существенный недостаток. Для того чтобы построить программный модуль, который выполнял бы такую функцию, необходимо просмотреть решения, которые существуют на рынке в данный момент и выбрать подходящую систему. Определим, что является генератором отчетов.

Генератор отчетов – это библиотека или программа, позволяющая представить данные в структурированном виде, удобном для чтения и печати [9]. Чаще всего для пользователя достаточно затруднительно, после завершения работы с какими-либо данными, формировать отчет вручную. Инструменты для генерации отчетов значительно облегчают задачу.

Стоит отметить, что генераторы отчетов нужны не только пользователям, но и разработчикам, так как система, оснащенная функцией выведения отчетности, имеет некоторое преимущество перед аналогами.

Для дальнейшей разработки программного модуля необходимо проанализировать имеющиеся на рынке средства генерации отчетов и выбрать тот, который максимально удовлетворяет требованиям. Были выбраны 5 генераторов отчетов: Crystal Reports, StimulSoft Reports, Jasper Report, Next Report, BIRT. Для оценивания каждого из инструментов были разработаны следующие критерии:

1. Язык внедрения.
2. Возможность создания диаграмм.
3. Форматы документов отчетов, выгружаемых пользователю.
4. Возможность выбора визуального оформления отчета (шрифты, цветовое оформление, отступы и т.д.).
5. Стоимость программы.
6. Открытый код.
7. Возможность работы с базами данных.

Теперь необходимо проанализировать каждое средство генерации отчетов в отдельности.

### Генератор отчетов Crystal Reports Server

Данный инструмент разработан компанией SAP и предназначен для использования компаниями малого и среднего бизнеса [27]. Одним из важных достоинств данного инструмента является подробная документация с описанием всех функций системы. Генератор отчетов поддерживает большинство языков программирования (в том числе и Java), что позволяет разработчикам легко интегрировать его в приложения.

Данное программное средство предоставляет пользователю возможность работы со сложными цветовыми диаграммами. Также с помощью специальных инструментов можно изменять параметры оформления отчета. Стоимость системы достаточно велика для академических исследований, примерно 75 000 рублей [10], поэтому разработчики данного инструмента не предоставляют исходный код.

Система предоставляет широкий спектр возможностей для работы с базами данных, различными СУБД, а также поддерживает работу с языком запросов SQL.

Программа позволяет экспортировать отчеты в различных популярных форматах (.doc, .xls, .rtf,.pdf и т.д.).

### Система Stimulsoft Reports

StimulSoft Reports – это полный набор инструментов для построения отчетов в средах JavaScript, ASP.NET, ASP.NET MVC, WPF, Silverlight, WinRT, HTML5, Windows Forms, PHP, Java, и Flex [11]. Система поддерживает работу с различными видами таблиц, диаграмм и других средств визуализации. В данном инструменте содержится полноценный дизайнер отчета, который позволяет разработчику полностью спроектировать шаблон отчета.

Данное средство генерации отчетов предоставляется пользователям в ценовом диапазоне от 29 000 рублей [12], но также предоставляется и бесплатная демоверсия для ознакомления с продуктом. Предоставляется несколько вариантов покупки, с исходным кодом и без.

Программа позволяет работать с базами данным и запросами SQL, а также экспортировать готовые отчеты в различных форматах.

### Генератор отчетов Jasper Report

Jasper Report – это Java-библиотека для работы с отчетами. Система позволяет выгружать готовые отчеты в форматах PDF, RTF, HTML, XLS, CSV и XML [12]. Данное средство позволяет загружать данные из различных источников, от документа в формате XML до самостоятельно созданного источника данных. Также возможна неограниченная встраиваемость отчетов друг в друга. Для работы с диаграммами в системе встроена специальная библиотека.

Основным преимуществом данного инструмента перед другими является то, что он бесплатен и предоставляется с открытым исходным кодом. Так же, как и аналоги, Jasper Report позволяет оперировать запросами SQL и работать с различными базами данных. В системе есть встроенный дизайнер, который позволяет настраивать внешний вид отчетов.

### Генератор отчетов Next Report

Генератор отчетов Next Report представляет собой систему для разработки отчетов, состоящую из 3 частей (Next Report Engine, Next Report Designer, Next Report Server). Внедрение данной системы производится на языке программирования Java [14].

С помощью Next Report Designer разработчик может спроектировать дизайн отчета. Компонент «Next Report Engine» - это библиотека, которая позволяет запустить пакет Next Report внутри разрабатываемого приложения. Последний элемент, «Next Report Server», обрабатывает логику построения отчетов.

Программа позволяет экспортировать отчеты в различных форматах, таких как HTML, EXCEL, PDF и т.д. Также система работает с базами данных, SQL, диаграммами и кросс-таблицами. Пакет инструментов предоставляется бесплатно, но без исходного кода.

### Программа BIRT

Данная программа представляет собой средство, которое позволяет визуализировать данные и генерировать отчеты из приложений, написанных на Java и Java ЕЕ. Генератор отчетов BIRT, разработанный Eclipse Foundation, состоит из двух компонентов. Первый компонент это дизайнер, с помощью которого можно разрабатывать отчетные формы, второй – позволяет генерировать эти отчеты из Java-приложения [15].

Программное средство предоставляется бесплатно и с открытым кодом. Оно позволяет разрабатывать отчеты со сложными таблицами, диаграммами, графиками, изображениями и т.д. Отчеты можно экспортировать в такие форматы как HTML, PDF, Excel, Power Point, Word [16].

Программа является бесплатной и предоставляется с открытым исходным кодом. Использование данной системы удобно для разработки на языке программирования Java, так как данный программный продукт максимально адаптирован к популярной среде разработки Eclipse.

По итогам сравнительного анализа была составлена таблица (табл. 1.2), в которой представлена краткая характеристика каждого генератора отчета в соответствии с вышеупомянутыми критериями.

*Таблица 1.2 Сравнительная таблица генераторов отчетов*

|  | Язык внедрения | Возможность создания диаграмм | Форматы отчетов, выгружаемых пользователю | Возможность выбора визуального оформления | Стоимость | Открытый код | Возможность работы с базами данных |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Crystal Reports | Visual Basic, C++,C#, Java и т.д. | + | RPT, RPTR, HTML, MS Word, MS Excel, ODBC, PDF, RTF, CSV, TTX, TXT, XML | + | ~ 75 000 руб. | - | + |
| StimlSoft Reports | JavaScript, HTML5, PHP, Java, и Flex | + | PDF, XPS, MS Power Point, HTML, MHT, TXT, RTF, MS Word, MS Excel, CSV, DBF, XML | + | ~ 29 000 руб. | + (при определенных условиях покупки) | + |
| Jasper Report | Java | + | PDF, RTF, HTML, XLS, CSV и XML | + | Бесплатно | + | + |
| Next Report | Java | + | HTML, XLS, XLSX, PDF, DOCX, RTF,  CSV, TSV, TEXT и XML. | + | Бесплатно | - | + |
| BIRT | Java | + | HTML, PDF, Excel, Power Point,  Word | + | Бесплатно | + | + |

Таким образом, среди всех рассмотренных средств генерации отчетов, наиболее подходящим для решения поставленной задачи является BIRT. Данный инструмент поддерживает внедрение с помощью языка программирования Java, позволяет визуализировать данные в виде диаграмм, а также выбрать параметры отображения отчета. Также выбранное средство предоставляется бесплатно, что является важным критерием при академических исследованиях.

## Вывод по первой главе

В данной главе был проанализирован бизнес-процесс, который планируется автоматизировать с помощью программного модуля. По итогам анализа можно сказать, что на данный момент генерация отчетов о собранной статистике автоматизирована недостаточно, что объясняет необходимость разработки дополнительного программного обеспечения. Рассмотрев уже существующие решения, был сделан вывод, что ни один из представленных на рынке инструментов не соответствует разработанным критериям на 100%, но один из них был выбран в качестве основы для разработки плагина, а именно Gate Developer.

Данный инструмент уже оснащен большим количеством плагинов, которые также были рассмотрены в данной главе. Многие из проанализированных инструментов, встроенных в систему GATE, могут помочь при разработке программного продукта и уменьшить объемы реализации. В ходе рассмотрения плагинов было отмечено, что нет ни одной разработки включающей в себя генерацию отчетов. Для данного процесса необходим генератор отчетов, который и был выбран по итогам анализа данных средств. Инструмент для генерации отчетов BIRT подошел по большинству критериев, соответственно он будет использован в ходе разработки плагина. Основываясь на проанализированных данных можно спроектировать модель разрабатываемого программного модуля.

# Глава 2. Проектирование программного модуля для сбора статистики и генерации отчетов

Данная глава является описанием этапа проектирования разрабатываемого программного модуля. На основе результатов анализа проведенного выше, были выбраны компоненты будущей системы. Далее необходимо сформулировать требования к программе и разработать архитектуру, а также выбрать средства, с помощью которых она будет реализована.

## 2.1. Формирование требований к плагину

Формирование требований к программному продукту является важной стадией разработки, так как помогает четко определить какие функции должна выполнять программа и какие задачи решать. В данном исследовании требования исходят от «Заказчика», в форме пожеланий пользователя, его представлений о том, как должны работать программа. Задача состоит в том, чтобы внимательно изучить требования «Заказчика» и формализовать их.

### Требования к программному модулю от заказчика

Основной пользовательской аудиторией данного программного модуля являются специалисты в области лингвистики и филологии, люди, не ориентированные на процессы, протекающие непосредственно внутри программы. Поэтому требования сформулированы в основном к функционалу и интерфейсу. Итак, «Заказчик» сформулировал следующие требования, программный модуль должен позволять:

* + - 1. Выводить в отчет статистику аннотации по единице текста (текст, корпус, корпуса текстов).
      2. Генерировать отчеты, в которых представлены результаты сравнения показателей разных единиц текста.
      3. Отображать результаты обработки текста в виде графиков разных типов (гистограмма, график, круговая и т.д.).

Таким образом, пользователь в своих требованиях определяет основные функции программы, которые теперь необходимо описать подробнее.

### Обработанные функциональные требования

Исходя из описанных выше требований, необходимо разработать и описать процесс, который выполняется, когда пользователь использует программный модуль. Для описания функций программы, как нельзя лучше подходят диаграммы UseCase, описанные выше (см. пункт 1.1.). Построенная диаграмма отображает не только функции программы, но и последовательность выполнения тех или иных задач (см. Приложение А).

Началом работы, как и было описано ранее, является загрузка текстовых корпусов в программу Gate Developer. Перед первым запуском программного модуля пользователю необходимо добавить его в систему Gate Developer, посредством модального окна добавления плагинов CREOLE. После этого следует выбор параметров выгружаемой статистики, на основе которых программный модуль на следующем шаге выполняет сбор статистических данных. Далее все действия пользователя проходят в интерфейсе дизайнера отчетов выбранной программы (система BIRT, см. пункт 1.4.). На следующем этапе пользователь выбирает показатели, которые необходимо отобразить в отчете, среди тех, которые были указаны на предыдущем шаге. Затем пользователь выбирает функцию, которой ему необходимо воспользоваться, порядок выполнения, которых не является важным. Стоит отметить, что выбор данных функций, кроме последнего, не является обязательным. Каждая из этих функций является реализацией требований, сформулированных заказчиком. После того как пользователь выбрал данные для отображения, плагин предоставляет возможность выбрать между 5 параметрами визуализации:

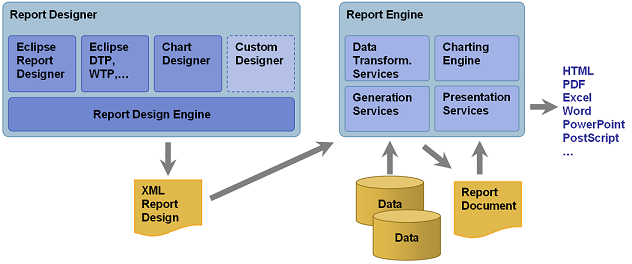
* Единицы текста для расчета частотности, которая определяет рамки определения частоты использования того или иного маркера, например, корпуса текстов, корпус, текст и т.д.
* Корпуса для сравнения, а также параметры, по которым будет производиться сравнение.
* Параметры отображения, такие как, таблица и график, для последнего должны быть возможность выбора типа (гистограмма, круговая, столбчатая и т.д.).
* Параметры оформления отчета, например, размер и цвет шрифта, отступы, цвет фона и т.д.

На основании всех выбранных параметров генератор отчета формирует XML файл, после чего пользователю предлагается выбрать формат выгрузки. Сформированный отчет выгружается на локальный компьютер пользователя. Формирование требований позволило рассмотреть программный модуль как единую систему, теперь же необходимо описать компоненты, из которых она состоит.

## Разработка архитектуры плагина

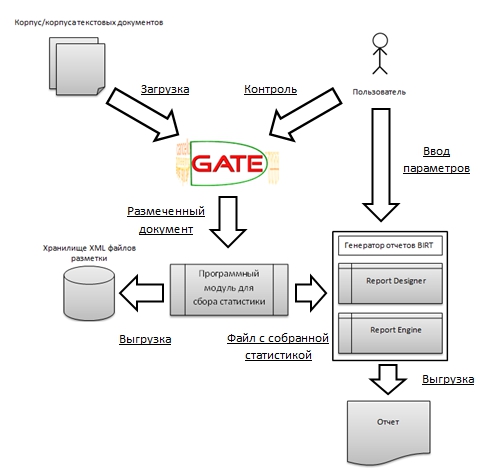
Важным этапом разработки любого программного продукта является моделирование структуры программы. Необходимо графически изобразить и описать, как компоненты программного модуля взаимодействуют друг с другом.

Архитектура программного модуля взаимодействует, прежде всего, со средой Gate Developer, которая была выбрана для интеграции плагина в пункте 1.2. Данный инструмент является связующим звеном между плагином и корпусами текста. Gate проводит разметку корпусов, после чего направляет файлы с размеченными текстами в формате XML в плагин. Программа проводит сбор статистики, после чего данные переходят в хранилище и в систему BIRT. Данный инструмент состоит из 2 компонентов: Report Designer и Report Engine (рис. 2.1). Первый выступает в качестве пользовательского интерфейса посредством элемента Custom Designer, второй – в качестве обработчика данных.



***Рисунок 2.1. Архитектура системы для генерации отчетов BIRT***

Целевым компонентом является отчет, который генерируется и выгружается через генератор отчетов BIRT. В рассматриваемой архитектуре пользователь производит ввод различных параметров, а также контролирует работу программы. Схематичное изображение архитектуры программного продукта представлено на схеме (см. рис. 2.2.).



***Рисунок 2.2. Архитектура программного модуля***

Итак, архитектура плагина построена и описана взаимосвязь компонентов, необходимо изучить средства, для реализации, которые подходят для разработки такого рода. После этого должны быть выбраны инструменты, которые будут использоваться при разработке программного продукта.

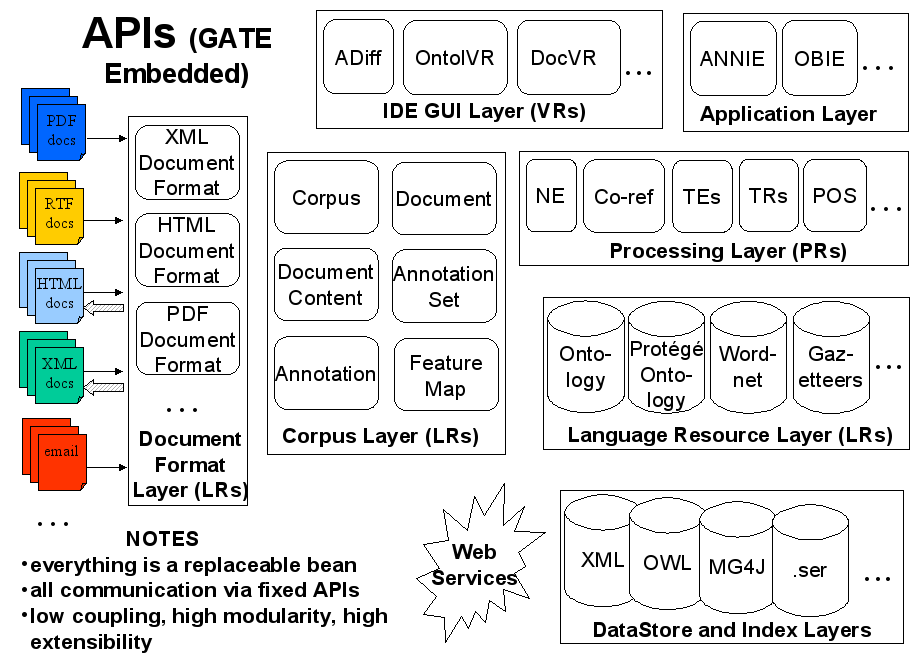
## Выбор средств разработки

Тщательное рассмотрение и последующий выбор инструментов для разработки позволит повысить качество продукта, а также сократить время исправления возможных ошибок, вызванных использованием не подходящих средств. Необходимо описать и выбрать средства разработки двух типов, а именно, язык программирования и среда разработки. Сначала выберем язык, на котором будет написан программный модуль.

### Выбор языка программирования

На данный момент на рынке существует более 8 000 языков программирования, но для данного исследования нет необходимости рассматривать их, так как в пункте 1.2. была выбрана система для интегрирования плагина – среда Gate Developer. Данная программа реализована на языке программирования Java, таким образом, вопрос о выборе языка реализации плагина не возникает.

Группа разработчиков системы Gate Developer создала инструмент для работы с языком программирования Java – Gate Embedded, фреймворк предоставляющий доступ ко всем функциям Gate Developer. Gate Embedded представляет собой набор библиотек, которые помогают программистам создавать плагины для среды. Состав данного инструмента показан на рисунке 2.3.



***Рисунок 2.3. Архитектура инструмента Gate Embedded***

Gate Embedded состоит из различных слоев, которые в свою очередь содержат в себе средства для работы с теми или иными элементами. Существует 7 основных слоев: слой форматов документов, слой корпусов, слой пользовательского интерфейса, слой работы с приложениями, слой процессных ресурсов, слой языковых ресурсов и слой хранения данных. Все эти средства в комплексе позволяют программисту получить доступ к объектам и функциям программы Gate Developer без применения сторонних библиотек. Тем не менее, для непосредственного написания Java-кода необходим инструмент, рассмотрим наиболее популярные из них.

### Обзор интегрированных сред разработки

Интегрированная среда разработки (англ. IDE – Integrated Development Environment) – это набор программных средств, с помощью которых происходит разработка программы. Чаще всего IDE содержит в себе текстовый редактор, компилятор/интерпретатор, средство сборки и отладчик.

Как было сказано выше, язык программирования, используемый в данной разработке это Java, следовательно, рассмотрим наиболее популярные IDE для разработки программ на выбранном языке. По итогам анализа информации в сети Интернет, можно выделить три среды разработки: Eclipse IDE for Java Developers, NetBeans IDE и IntelliJ IDEA. Достоинства и недостатки каждой из них сформулированы на основе, как личного опыта использования, так и отзывов других пользователей. Рассмотрим каждую из них более детально.

#### Среда разработки Eclipse IDE for Java Developers

Eclipse IDE for Java Developers – это программная платформа с открытым кодом, разработанная компанией Eclipse Foundation и предоставляемая в открытом доступе [22]. Программная среда является полностью бесплатной, при полном функционале, что сильно повышает ее популярность. Основными достоинствами данного инструмента являются:

* Кроссплатформенность, возможность работать с Windows, Linux, Solaris и Mac OS X.
* Возможность разработки расширений для программного обеспечения.
* Оснащенность полнотекстовой интерактивной документацией, с помощью которой легко ориентироваться в информации.
* Возможность интеграции с GIT.
* Огромное количество подгружаемых плагинов, которые значительно расширяют функционал программы.

Несмотря на достоинства среды разработки, ее недостатки могут периодически осложнять работу над разработкой программы. Можно выделить несколько слабых сторон данного программного обеспечения:

* Медленная скорость работы среды, из-за старых версий Java Virtual Machine, на которой основывается работа IDE [22].
* Вероятность возникновения конфликта между плагинами в ходе загрузки.
* Необходимость настройки среды перед применением.

#### Среда разработки NetBeans IDE

IDE NetBeans- это интегрированная среда разработки, поддерживаемая компанией Oracle. Данный инструмент поддерживает разработку на многих языках, таких как Java, Python, PHP, JavaScript, C, C++ и т.д. IDE позволяет разрабатывать настольные, мобильные и веб-приложения. Программное средство предоставляется с открытым исходным кодом бесплатно на официальном сайте [23].

Рассматриваемая среда обладает следующими достоинствами:

* Скорость работы, быстрее, чем Eclipse IDE, но медленнее, чем Intellij IDEA.
* Удобное и простое средство для построения пользовательского интерфейса GUI Builder.
* Поддержка различных стандартов Java (Enterprise Edition, Standard Edition, Mobility).
* Редактор кода заканчивает код при написании и предлагает список элементов.

Из недостатков можно отметить недостаточную документацию данного продукта, чтобы найти информацию о том или ином компоненте, необходимо потратить достаточно много времени и сил. Также пользователи жалуются на то что авто-завершение текста не всегда предлагает то что подразумевалось программистом [24].

#### Среда разработки Intellij IDEA

IntelliJ IDEA – это программное обеспечение, разработанное компанией JetBrains, которое позволяет программистам создавать программные продукты. Среда разработки поддерживает большое количество языков – Java, JavaScript, CoffeeScript, HTML/XHTML/HAML, CSS/SASS/LESS, XML/XSL/XPath, YAML, ActionScript/MXML, Python, Ruby, SQL, PHP, Kotlin, Clojure, C, C++. IntelliJ IDEA имеет две формы поставки: бесплатная Community Edition и платная Ultimate Edition. Первая версия предназначена для JVM и Android разработки, вторая - для производственной и веб-разработки. Среда разработки IntelliJ IDEA обладает следующими достоинствами:

* Функция «умное автодополнение», данное выражение означает, что система предоставляет список подходящих символов, основываясь не только на общей частоте употреблений, но и на «стиле» программирования разработчика, на его личной частоте использования тех или иных символов.
* Подсветка синтаксиса при вводе строк на других языках программирования, таких как SQL, HTML, CSS и JavaScript.
* Рефакторинг одинаковых фрагментов кода.
* Интеграция с системами контроля версий, такими как Git.

Несмотря на выше описанные преимущества, система имеет и недостатки, некоторые из которых вытекают из преимуществ. Перечислим некоторые из них:

* Неоправданно высокая стоимость в рамках академической разработки – приблизительно $500 в год [25].
* Долгое время выполнения компиляции.
* При подключении дополнительных модулей, многие из них могут «конфликтовать».
* Частые поломки при обновлении плагинов среды разработки.

#### Итоги обзора

По итогам рассмотрения вышеописанных сред разработки можно сделать вывод, что наиболее подходящей является Eclipse. Если составить рейтинг трех описанных IDE, не учитывая специфику разрабатываемого продукта, то он будет выглядеть так:

Средство разработки IntelliJ IDEA.

Программное средство NetBeans.

Среда разработки Eclipse IDE for Java Developers.

Данное расположение элементов списка объясняется тем, что по скорости работы и по качеству обработки кода, Eclipse уступает NetBeans и IntelliJ IDEA. Но в рамках данной работы, данная система является наиболее подходящей по ряду причин:

* + - 1. Генератор отчетов BIRT, выбранный ранее, является продуктом Eclipse, что означает, что процессы интеграции и взаимодействия между этими двумя компонентами будут выполняться эффективнее, чем в остальных средах разработки. Также стоит отметить, что существует документация, описывающая работу системы BIRT в рамках Eclipse IDE, что значительно облегчит работу.
      2. Несмотря на то, что выбранная система не может предоставить пользователю повышенную производительность, она подходит для разработки программного модуля, так как программа не предполагает подключения большого количества плагинов и объёмного кода, поэтому данный недостаток не является критичным.
      3. Средство предоставляется бесплатно с полным функционалом в отличие от других рассмотренных сред разработки, что является важным фактором при академической разработке.
      4. Средства для разработки графического интерфейса Eclipse IDE, на порядок ниже, чем у NetBeans и IntelliJ IDEA, но так как визуализация в программном модуле осуществляется с помощью генератора отчетов BIRT, данный недостаток не является существенным.
      5. Необходимость настройки среды разработки перед первым применением также не является критичным фактором, так как данный процесс тщательно описан в документации программы.

## Выводы по второй главе

В ходе второй главы были сформированы функциональные требования к разрабатываемому продукту, посредством обработки списка пожеланий заказчика. При рассмотрении процесса сбора статистики и генерации отчетов было выявлено, что программный модуль должен предоставлять пользователю возможности отображения по заданным параметрам.

Разработка программного продукта без предварительного моделирования структуры является крайне неэффективной. Поэтому в данной главе была построена архитектура программного модуля, в соответствии с которой будет выполняться разработка. Элементами структуры программы являются компоненты, выбранные и описанные в первой главе.

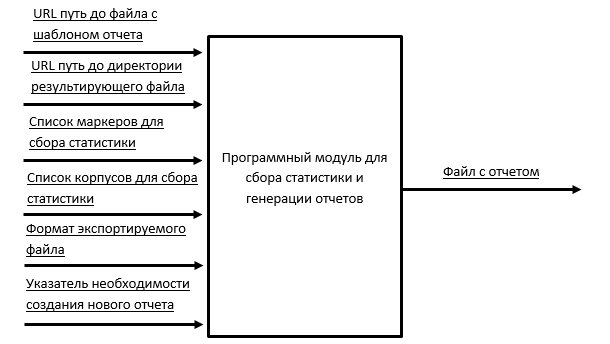
После того как были определены функции, которые должен выполнять программный модуль и построена его архитектура, был произведен сравнительный анализ инструментов разработки, таких как язык программирования и интегрированная среда разработки. В частности, язык разработки – Java – был выбран на основании средства обработки естественного языка, описанного в первой главе, системы Gate Developer, которая также реализована на этом языке. Были рассмотрены популярные среды разработки на выбранном языке программирования и выбрана Eclipse IDE for Java Developers, которая повысит удобство разработки за счет того, что генератор отчетов является программным компонентом для интеграции в данную среду. Таким образом, на данном этапе выделены функциональные требования к программному модулю, а также выбраны инструмент необходимые для разработки.

# Глава 3. Разработка программного модуля

В данной главе описывается процесс разработки программного модуля, предназначенного для сбора статистики и генерации отчета о полученных данных. Программный модуль представляет собой плагин для программы Gate Developer, который применяется к уже размеченному тексту.

## 3.1. Описание параметров входа и выхода

Перед тем как приступить к непосредственной разработке программного модуля, необходимо описать какие параметры должны быть использованы. Так как плагин должен генерировать отчет на основе шаблона и экспортировать готовый отчет, в качестве входных параметров необходимо получать путь до файла с шаблоном отчета и путь до директории, в которой необходимо сохранить отчет. Схема передачи параметров изображена на рисунке 3.1.



***Рисунок 3.1. Входные и выходные параметры программного модуля***

Опишем каждый из вышеуказанных параметров подробнее (табл. 3.1).

***Таблица 3.1. Характеристика параметров входа и выхода***

| Параметр | Название параметра | Обязательность | Тип данных | Описание |
| --- | --- | --- | --- | --- |
| URL путь до файла с шаблоном отчета | templateFileURL | - | URL | Содержит в себе путь до файла с шаблоном отчета |
| URL путь до директории результирующего файла | outputfileURL | + | URL | Содержит в себе путь до директории в которой необходимо сохранить отчет |
| Список маркеров для сбора статистики | Вводятся при создании обрабатывающего ресурса | + | Список полей типа Boolean | Список маркеров для которых необходимо собрать статистику |
| Список корпусов для сбора статистики | corpList | - | List<String> | Список корпусов для которых необходимо собрать статистику |
| Формат экспортируемого файла | outputFormat | - | String | Формат файла экспорта, если он не указан, то по умолчанию выгружается файл в формате HTML |
| Указатель необходимости создания нового отчета | isNeededNewTemplate | + | Boolean | Параметр указывает на необходимость создания нового отчета. Если значение параметра – true, то плагин открывает BIRT RCP Report Designer. |

Для того чтобы параметры отобразились в редакторе их необходимо добавить, используя средства Gate Embedded. Рассмотрим на примере добавления параметра «URL путь до файла с шаблоном отчета», для этого необходимо добавить следующие строки в программный код:

//Путь до шаблона-отчета

public URL getTemplateFileURL()

{

return templateFileURL;

}

@gate.creole.metadata.RunTime //указывается тип параметра

@gate.creole.metadata.Optional //указывается необязательность параметра

@CreoleParameter(

comment = "Filename of template report",

suffixes = ".rpttemplate",//так как тип параметра – путь к файлу – указывается тип файла

priority = 1)

public void setTemplateFileURL(URL source)

{

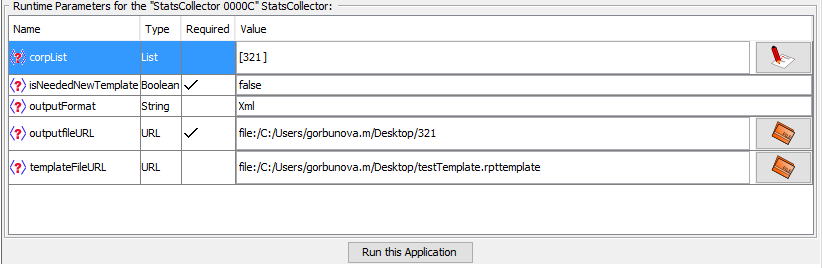
this.templateFileURL = source;

}

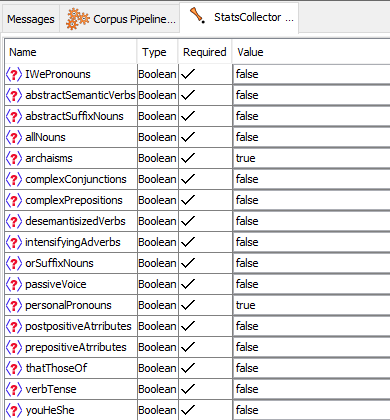
protected URL templateFileURL;

***Листинг 3.1. Добавление параметра templateFileURL***

Полный листинг добавления параметров можно увидеть в приложении Б. В результате выполнения плагина, в окне редактора параметров обрабатывающего ресурса отображаются добавленные параметры (рис. 3.2) и список необходимых маркеров (рис. 3.3).



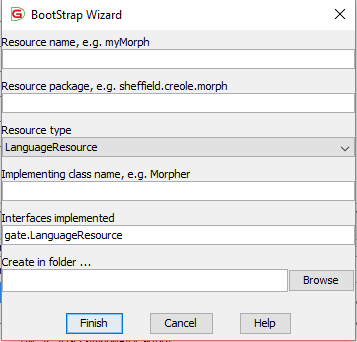
***Рисунок 3.2. Входные параметры обрабатывающего ресурса***



***Рисунок 3.3. Список маркеров для сбора статистики***

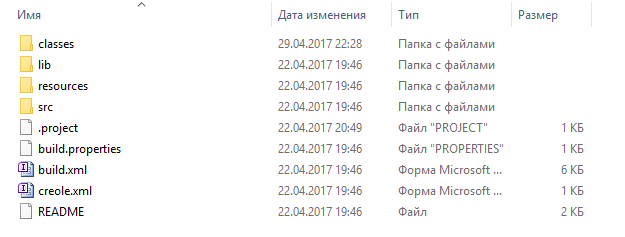
## Описание процесса создания плагина

Согласно документации разработчиков Gate [2], разработку нового плагина необходимо начинать с вызова Bootstrap Wizard. Это инструмент, который создает директорию будущего плагина согласно введенным параметрам (рис. 3.4).



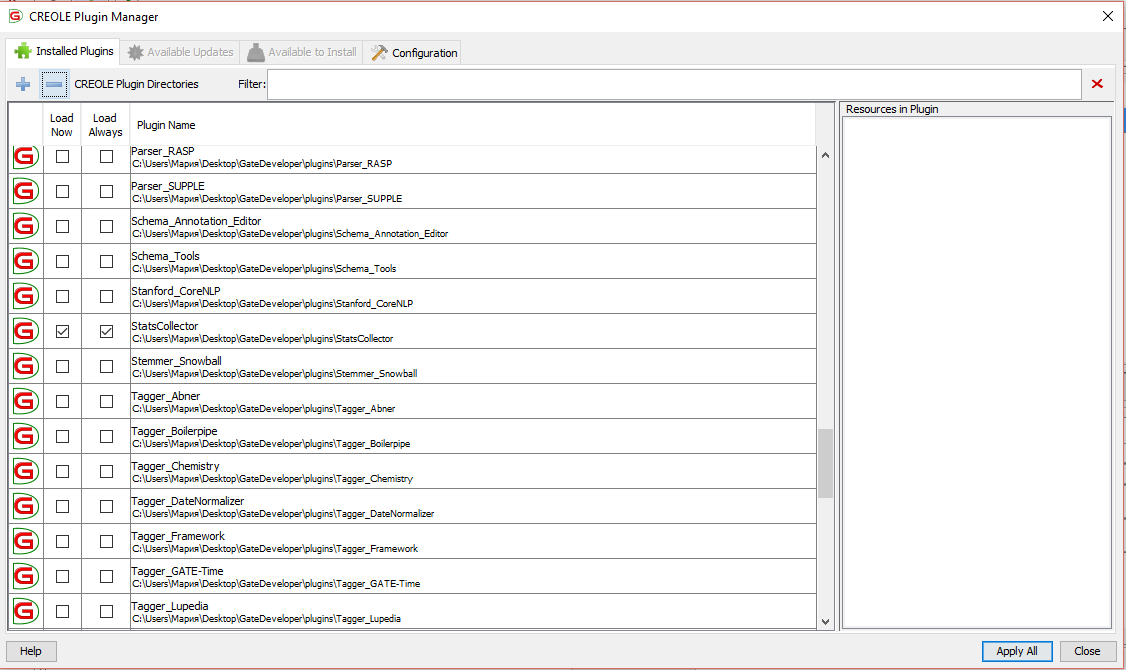
***Рисунок 3.4. Инструмент Bootstrap Wizard***

После нажатия кнопки “Finish”, в указанном месте файловой системы создается папка, содержимое которой представлено на рисунке 3.5.



***Рисунок 3.5. Корневая папка плагина***

Для того чтобы плагин заработал, необходимо добавить данную папку в папку “plugins” в директории Gate Developer, а также загрузить \*.jar файл с кодом плагина. Далее необходимо добавить плагин в систему через Creole Manager (рис. 3.6).

**

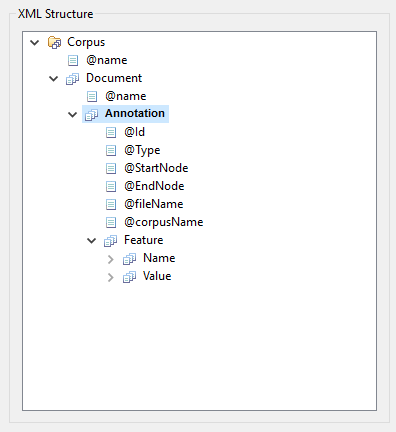
***Рисунок 3.6. Окно CREOLE Manager***

После этого плагин появляется в системе. Но так как никаких классов еще реализовано не было, плагин только отображается и не выполняет никаких практических функций.

## 3.3. Особенности реализации плагина в среде Gate Developer

Ранее было сказано, что в системе Gate Developer уже существует плагин для сбора статистических данных. На данный момент StatsCollector, программный компонент по сбору статистики, экспортирует результат работы в виде файла в формате \*.txt. Данный формат не удобен для дальнейшей обработки, к тому же результирующий файл не обладает необходимой для этого структурой. Таким образом, в плагине необходимо реализовать алгоритм, который бы выгружал аннотированные тексты в структурированную форму (XML) и передавал их в генератор отчетов, в качестве источника данных. В зависимости от значения параметра corpList, плагин формирует .xml документ с данными корпуса или же документ с данными из нескольких корпусов, с помощью функции getcorp(). В качестве входного параметра getcorp() принимает название корпуса, и формирует строку с данными. Для работы с XML и его обработкой, была использована библиотека DOM4J. В случае, если плагин применяется к нескольким корпусам, метод применяется к каждому из них и формирует общую строку для набора корпусов.

Размеченные корпуса в формате XML после обработки обладают структурой, изображенной на рисунке 3.7.



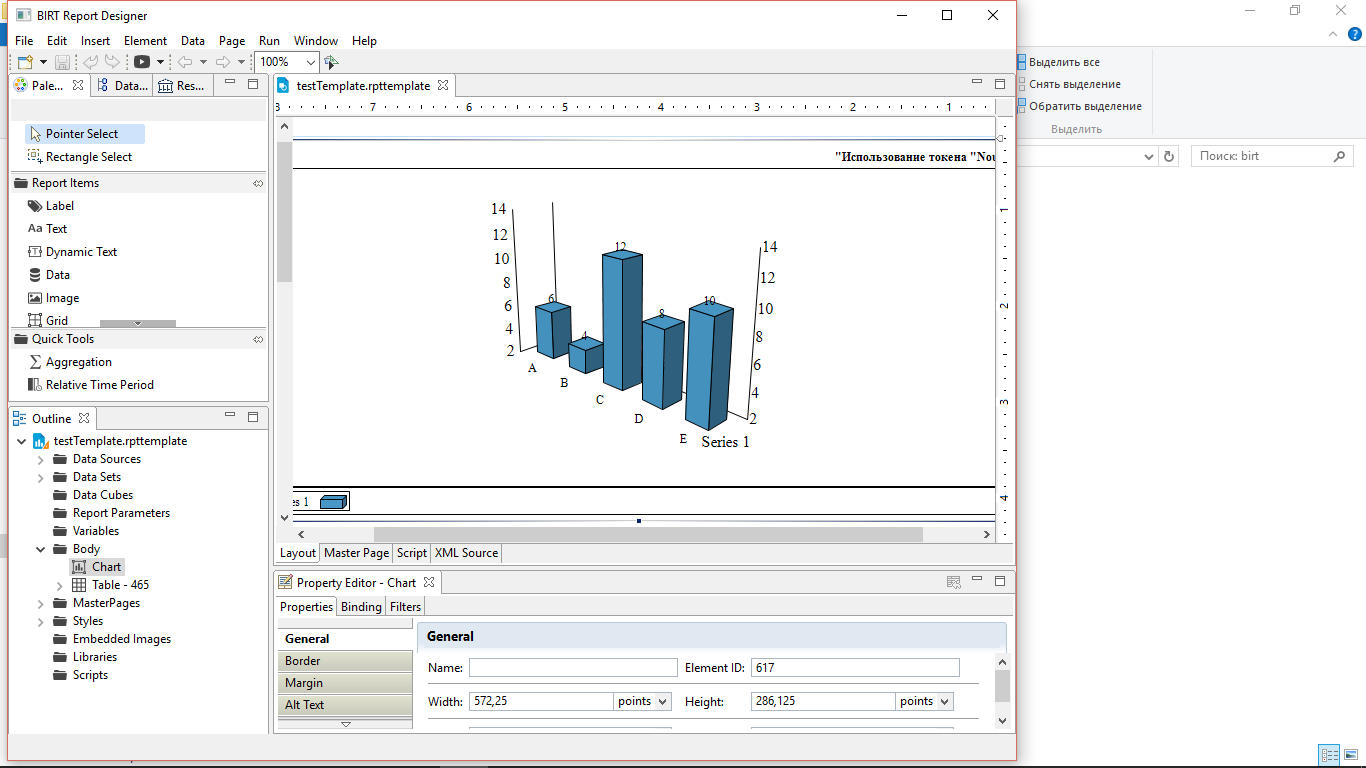
***Рисунок 3.7. Структура аннотированного текста***

Следующим этапом работы программного модуля является выгрузка XML документа в генератор отчетов в качестве DataSource. Для этого используется класс ReportGenerator, который определяет формат экспорта, генерирует отчет и выгружает его в указанную пользователем директорию. Все эти операции осуществляются посредством библиотеки BIRT API, которая позволяет внести в шаблон отчета информацию об источнике данных, с помощью класса OdaDataSourceHandle.

Чтобы редактировать программный код плагинов Gate, необходимо открыть соответствующую папку в папке “plugins” директории Gate Developer и запустить \*.jar файл через какой-либо редактор, в нашем случае IDE Eclipse. Листинг программы представлен в Приложении Б.

## 3.3. Разработка шаблонов для генерации отчетов

Для того чтобы сгенерировать отчет, пользователю необходимо передать в плагин URL адрес расположения шаблона отчета. В текущем контексте шаблон рассматривается как спроектированная структура отчета. Для создания и редактирования шаблонов отчетов существует независимый от среды разработки Eclipse интерфейс, к которому можно получить доступ посредством введения значения true во входной параметр isNeededDrawTemplate. После введения данного параметра необходимо нажать кнопку “Run Application”, после чего пользователь видит окно, изображенное на рисунке 3.8.

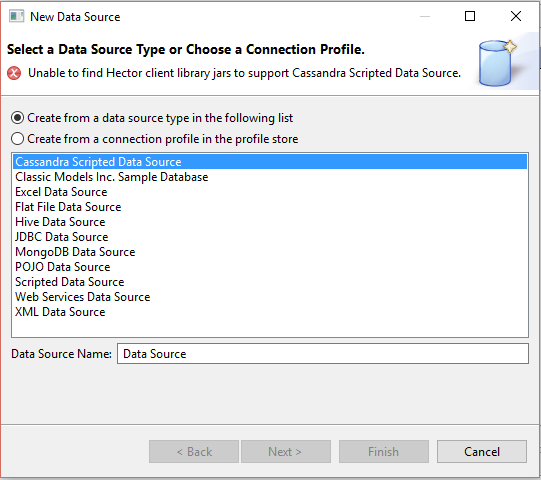


*Рисунок 3.8. Интерфейс BIRT Report Designer*

С помощью данного интерфейса пользователь может добавлять элементы из палитры и таким образом заполнять отчет. Так как на основе этого шаблона в дальнейшем будет генерироваться плагин необходимо, чтобы в нем была указана структура данных. Рассмотрим добавление данных на примере вышеуказанного xml-документа.

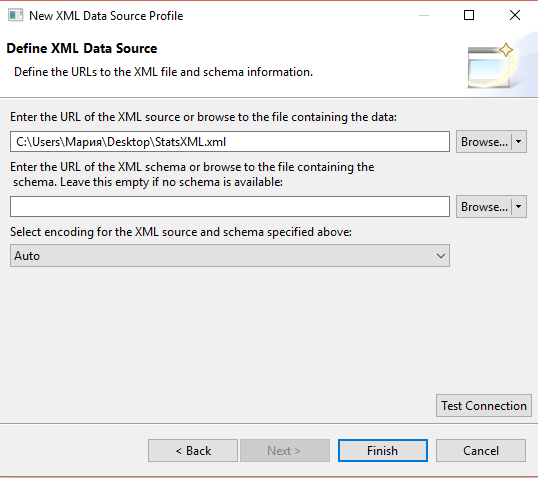
Сначала необходимо создать шаблон (.rpttemplate) или дизайн(.rptdesign),в данном случае это будет шаблон. После этого перейти на вкладку Data Explorer и с

помощью контекстного меню вызвать окно создания источника данных (рис. 3.9).



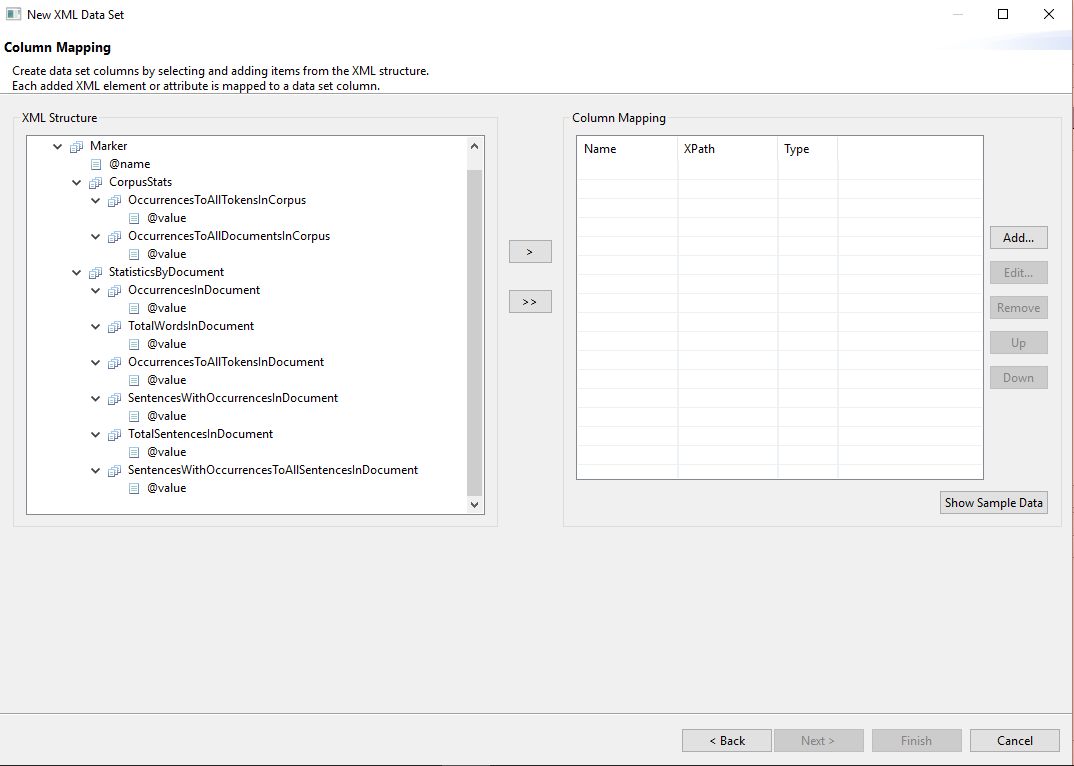
*Рисунок 3.9. Создание источника данных*

Далее необходимо последовательно выбрать «XML Data Source», а затем указать путь до самого источника данных (рис. 3.10). Схему данных программа определяет автоматически.



*Рисунок 3.10. Указание источника данных*

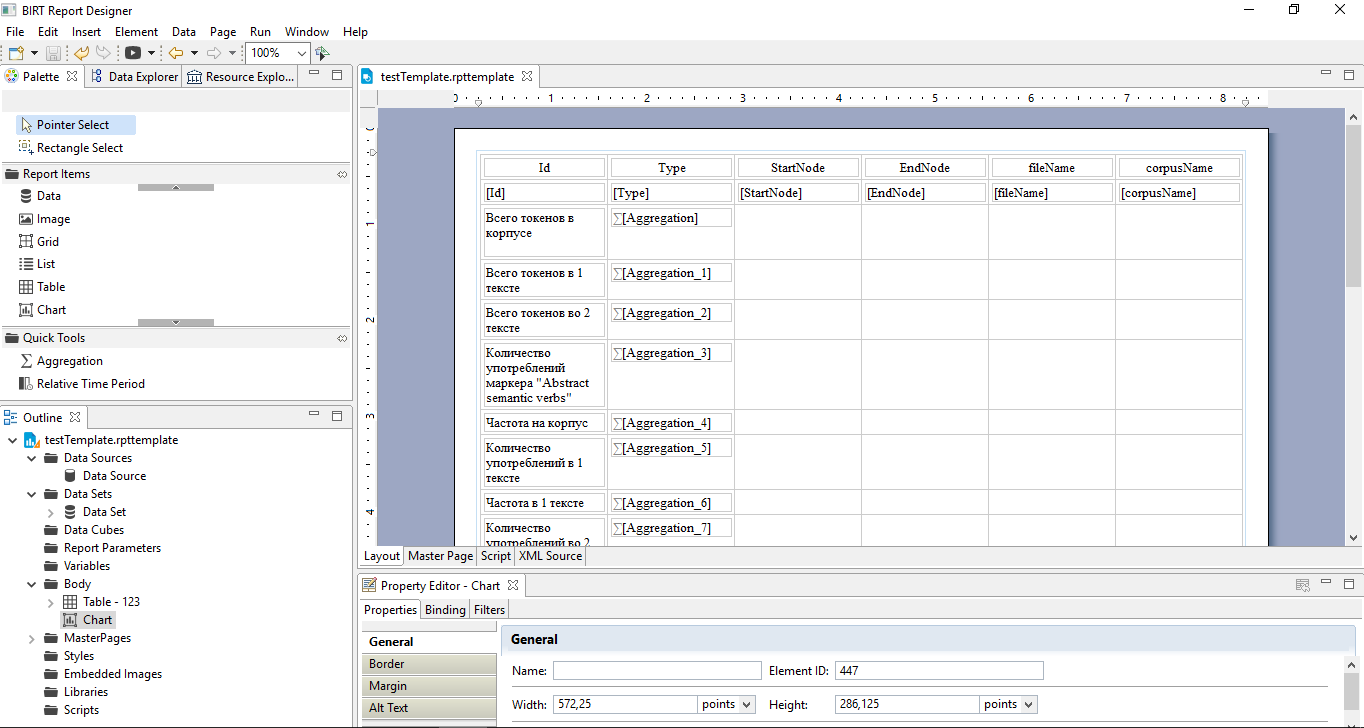
Далее необходимо создать DataSet, так же как для предыдущего действия, для этого необходимо воспользоваться контекстным меню. После того как заданы название и другие настройки набора данных, необходимо создать структуру колонок, выбрав их из структуры xml-файла (рис. 3.11).



*Рисунок 3.11. Выбор колонок из структуры документа*

После добавления данных пользователь может выполнять действия с ними, а также использовать различные элементы для создания шаблона. В конце работы необходимо сохранить шаблон в формате \*.rpttemplate (или \*.rptdesign). Указанные процедуры являются необходимыми, так как структура данных, включенная в шаблон, позволяет загрузить результаты вычисления статистики в отчет и отобразить их, так как было задано пользователем.

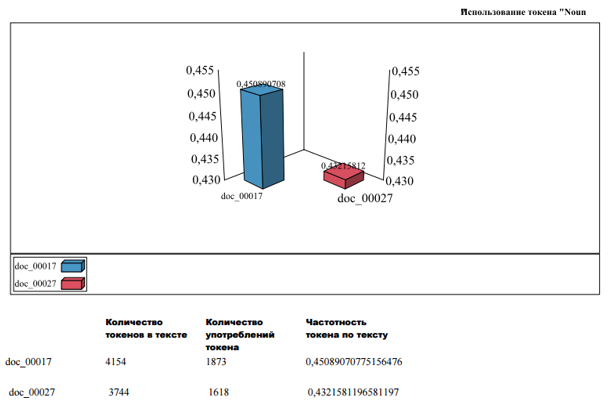
На рисунке 3.12 представлен пример шаблона. На нем мы можем увидеть функцию Aggregation, которая позволяет выполнять стандартные математические функции, а также функции математической статистики, что является применимым к данной работе.

**

*Рисунок 3.12. Шаблон отчета в программе BIRT RCP Designer*

## 3.4. Результаты реализации программного модуля

В рамках данной работы был реализован плагин, который позволяет сгенерировать отчет о собранной статистике. Для того чтобы убедиться в полноте реализации, необходимо протестировать работу программы. В качестве примеров были выбраны два текста. Задачей является загрузить их в плагин и сгенерировать отчет по шаблону. Для анализа выбран один маркер – «Noun» - так как, данные можно будет проверить посредством подсчётов «вручную». Для визуализации была выбрана столбчатая диаграмма (рис. 3.13), на которой можно увидеть разницу между частотностью использования маркера в пределах каждого из текстов.



*Рисунок 3.13. Диаграмма частотности маркера "Noun"*

Выведенные данные полностью совпадают с теми, что были подсчитаны вручную. Таким образом, можно сделать вывод, что программный модуль работает корректно. Так как использование плагина в рамках системы Gate Developer является достаточно трудоемким процессом, было разработано руководство пользователя, которое представлено в Приложении В.

# Заключение

Подводя итоги проделанной работы можно сказать, что исследования в сфере компьютерной лингвистики являются важными не только в академических рамках, но также могут применяться и в практических разработках. В ходе данной работы была описана теоретическая составляющая исследования, а также процессы моделирования и проектирования разработки.

С помощью анализа существующих средств обработки языка была выбрана среда для интегрирования плагина – Gate Developer. Анализируя генераторы отчетов, представленные на рынке, было выявлено наиболее подходящее для данной разработки средство – система BIRT. Для того чтобы при разработке лучше ориентироваться в среде Gate Developer, был произведен обзор плагинов, интегрированных в систему, которые будут использованы в ходе написания программного модуля.

В ходе моделирования программного продукта были сформулированы функциональные требования, на основе требований заказчика. А также построена архитектура разрабатываемого плагина, которая содержит все описанные компоненты. Выбор языка программирования Java для реализации был основан на решении использовать систему Gate Developer в качестве среды для интеграции модуля. Так как был определен язык реализации, выбор интегрированной среды разработки свелся к трем, наиболее популярных Java IDE. ПО результатам анализа была выбрана среда Eclipse IDE.

В рамках данной работы был разработан программный модуль, который позволяет генерировать отчет о статистике на основе шаблона, а также предоставляет доступ к интерфейсу для создания нового. В дальнейшем данная разработка будет усовершенствована, планируется расширить функционал дополнительными модулями, а также протестировать имеющийся на более нестандартных ситуациях использования.

# Библиографический список

1. Трофимов С. Варианты использования (Use Case) [Электронный ресурс]. URL: http://www.caseclub.ru/articles/use\_case.html (дата обращения: 24.05.2017).

2. Cunningham H. [и др.]. Developing language processing components with GATE (a user guide) // University of Sheffield. 2001. № Gate 2 (2006). C. 1–457.

3. Laurence Anthony’s Homepage [Электронный ресурс]. URL: http://www.laurenceanthony.net/ (дата обращения: 24.05.2017).

4. GATE.ac.uk - index.html [Электронный ресурс]. URL: https://gate.ac.uk/ (дата обращения: 24.05.2017).

5. The Stanford Natural Language Processing Group [Электронный ресурс]. URL: https://nlp.stanford.edu/software/tagger.shtml (дата обращения: 24.05.2017).

6. Лингвистический Энциклопедический Словарь. Кореферентность [Электронный ресурс]. URL: http://tapemark.narod.ru/les/243a.html (дата обращения: 24.05.2017).

7. Чанкер для русского языка [Электронный ресурс]. URL: http://web-corpora.net/wsgi/chunker.wsgi/npchunker/ (дата обращения: 24.05.2017).

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

9. генератор отчетов - это... Что такое генератор отчетов? [Электронный ресурс]. URL: http://dic.academic.ru/dic.nsf/eng\_rus/56647/генератор (дата обращения: 24.05.2017).

10. Crystal Reports [Электронный ресурс]. URL: https://msdn.microsoft.com/ru-ru/library/bb126227(v=vs.90).aspx (дата обращения: 24.05.2017).

11. Генератор отчетов для Java. Экспорт отчетов в PDF, MS Office, Open Office и т.д. :: Стимулсофт [Электронный ресурс]. URL: https://www.stimulsoft.com/ru/products/reports-java (дата обращения: 24.05.2017).

12. Бизнес аналитика. Генератор отчетов для JavaScript, .NET, ASP.NET, WPF, Flex, PHP, Silverlight, Java, ASP.NET MVC, WinRT, HTML5. Дизайнер отчетов. Создание отчетов. Сервер отчетов. Бизнес-аналитика. :: Стимулсофт [Электронный ресурс]. URL: https://www.stimulsoft.com/ru (дата обращения: 24.05.2017).

13. JasperReports® Library | Jaspersoft Community [Электронный ресурс]. URL: http://community.jaspersoft.com/project/jasperreports-library (дата обращения: 24.05.2017).

14. Business reporting software - NextReports Designer, Engine and Server [Электронный ресурс]. URL: http://www.next-reports.com/ (дата обращения: 24.05.2017).

15. BIRT Home [Электронный ресурс]. URL: http://www.eclipse.org/birt/ (дата обращения: 24.05.2017).

16. About [Электронный ресурс]. URL: http://www.eclipse.org/birt/about/project-organization/ (дата обращения: 24.05.2017).

17. Hardie A. The IMS Open Corpus Workbench ( CWB ) CQPweb System Administrator ’ s Manual 2014. C. 1–45.

18. Scott M. WordSmith Tools step by step 2010.

19. Talk T., Diachronic T., Corpus E. A Guide to using AntConc C. 1–9.

20. Сравнение утилит анализа корпусов — NLPub [Электронный ресурс]. URL: https://nlpub.ru/Сравнение\_утилит\_анализа\_корпусов (дата обращения: 24.05.2017).

21. Каталог лингвистических программ и ресурсов в Cети, ч.1 / Linguistics Software Catalogue, ch.1 [Электронный ресурс]. URL: http://rvb.ru/soft/catalogue/c01.html (дата обращения: 24.05.2017).

22. Sketch Engine | language corpus management and query system [Электронный ресурс]. URL: https://www.sketchengine.co.uk/ (дата обращения: 24.05.2017).

23. Обзор платформы Eclipse - как её использовать | Статьи о программном обеспечении [Электронный ресурс]. URL: https://hightech.in.ua/content/art-eclipse-platform (дата обращения: 24.05.2017).

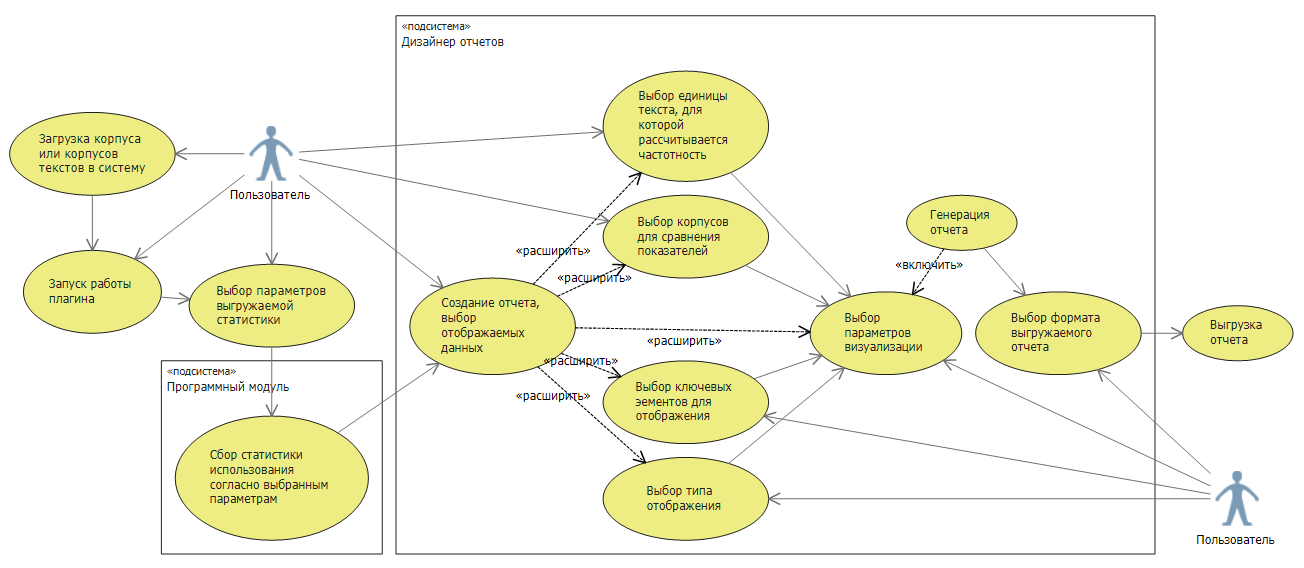
24. Welcome to NetBeans [Электронный ресурс]. URL: https://netbeans.org/ (дата обращения: 24.05.2017).

25. Обзор NetBeans IDE | Обзор IDE [Электронный ресурс]. URL: https://aboutide.wordpress.com/2011/03/23/обзор-netbeans-ide/ (дата обращения: 24.05.2017).

26. IntelliJ IDEA [Электронный ресурс]. URL: http://jetbrains.ru/products/idea/ (дата обращения: 24.05.2017).

27. SAP Crystal Reports 2011 User вЂTM s Guide SAP Crystal Reports 2011 2012.

# Приложение А. UseCase диаграмма для описания функциональных требований



***Рисунок А.1. Диаграмма описания функциональных требований***

# Приложение Б. Листинг программы

/\*

\* StatsCollector.java

\*

\* Copyright (c) 2000-2012, The University of Sheffield.

\*

\* This file is part of GATE (see http://gate.ac.uk/), and is free

\* software, licenced under the GNU Library General Public License,

\* Version 3, 29 June 2007.

\*

\* A copy of this licence is included in the distribution in the file

\* licence.html, and is also available at http://gate.ac.uk/gate/licence.html.

\*

\* Maria, 20/3/2017

\*

\* For details on the configuration options, see the user guide:

\* http://gate.ac.uk/cgi-bin/userguide/sec:creole-model:config

\*/

package jsh.academic.tagger;

import gate.util.Out;

import java.net.URL;

import org.eclipse.birt.core.framework.Platform;

import org.eclipse.birt.report.engine.api.\*;

import org.eclipse.birt.report.model.api.\*;

import org.w3c.dom.\*;

import gate.\*;

import gate.creole.\*;

import gate.creole.metadata.\*;

import gate.util.\*;

import java.awt.Desktop;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import java.net.URL;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.Iterator;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.dom4j.\*;

import org.dom4j.io.SAXReader;

import org.eclipse.birt.core.exception.BirtException;

/\*\*

\* This class is the implementation of the resource STATSCOLLECTOR.

\*/

@CreoleResource(name = "StatsCollector",

comment = "Collecting statistics in corpus and documents")

public class StatsCollector extends AbstractLanguageAnalyser

implements ProcessingResource {

boolean DEBUG\_MODE = true;

//Указатель необходимости создания шаблона отчета

public Boolean getisNeededNewTemplate() {

return isNeededNewTemplate;

}

@gate.creole.metadata.RunTime

@CreoleParameter(

comment = "If it's needed to create new template",

priority = 1)

public void setisNeededNewTemplate(Boolean source) {

this.isNeededNewTemplate = source;

}

protected Boolean isNeededNewTemplate;

//Формат экспорта

public String getoutputFormat() {

return outputFormat;

}

@gate.creole.metadata.RunTime

@gate.creole.metadata.Optional

@CreoleParameter(

comment = "Output File Format",

priority = 1)

public void setoutputFormat(String source) {

this.outputFormat = source;

}

protected String outputFormat;

//Корпуса для сравнения

public List getCorpList() {

return corpList;

}

@gate.creole.metadata.RunTime

@gate.creole.metadata.Optional

@CreoleParameter(

comment = "List of corporas to execute",

priority = 1)

public void setCorpList(List source) {

this.corpList = source;

}

protected List corpList;

//Путь до шаблона-отчета

public URL getTemplateFileURL() {

return templateFileURL;

}

@gate.creole.metadata.RunTime

@gate.creole.metadata.Optional

@CreoleParameter(

comment = "Filename of template report",

suffixes = ".rpttemplate",

priority = 1)

public void setTemplateFileURL(URL source) {

this.templateFileURL = source;

}

protected URL templateFileURL;

//Путь до файла экспорта

public URL getOutputfileURL() {

return outputfileURL;

}

@gate.creole.metadata.RunTime

@CreoleParameter(

comment = "Filename where results will be written",

suffixes = ".txt",

priority = 1)

public void setOutputfileURL(URL source) {

this.outputfileURL = source;

}

protected URL outputfileURL;

public Boolean getAllNouns() {

return AllNouns;

}

@CreoleParameter(

comment = "Count abstract semantic verbs")

public void setAllNouns(Boolean bool) {

this.AllNouns = bool;

}

/\* check to count this type of annotation \*/

protected Boolean AllNouns = true;

public Boolean getAbstractSemanticVerbs() {

return AbstractSemanticVerbs;

}

@CreoleParameter(

comment = "Count abstract semantic verbs")

public void setAbstractSemanticVerbs(Boolean bool) {

this.AbstractSemanticVerbs = bool;

}

/\* check to count this type of annotation \*/

protected Boolean AbstractSemanticVerbs = true;

public Boolean getAbstractSuffixNouns() {

return AbstractSuffixNouns;

}

@CreoleParameter(

comment = "Count nouns with abstract suffixes")

public void setAbstractSuffixNouns(Boolean bool) {

this.AbstractSuffixNouns = bool;

}

/\* check to count this type of annotation \*/

protected Boolean AbstractSuffixNouns = true;

public Boolean getArchaisms() {

return Archaisms;

}

@CreoleParameter(

comment = "Count nouns with abstract suffixes")

public void setArchaisms(Boolean bool) {

this.Archaisms = bool;

}

/\* check to count this type of annotation \*/

protected Boolean Archaisms = true;

public Boolean getDesemantisizedVerbs() {

return DesemantisizedVerbs;

}

@CreoleParameter(

comment = "Count desemantisized verbs")

public void setDesemantisizedVerbs(Boolean bool) {

this.DesemantisizedVerbs = bool;

}

/\* check to count this type of annotation \*/

protected Boolean DesemantisizedVerbs = true;

public Boolean getComplexConjunctions() {

return ComplexConjunctions;

}

@CreoleParameter(

comment = "Count complex conjunctions")

public void setComplexConjunctions(Boolean bool) {

this.ComplexConjunctions = bool;

}

/\* check to count this type of annotation \*/

protected Boolean ComplexConjunctions = true;

public Boolean getComplexPrepositions() {

return ComplexPrepositions;

}

@CreoleParameter(

comment = "Count complex prepositions")

public void setComplexPrepositions(Boolean bool) {

this.ComplexPrepositions = bool;

}

/\* check to count this type of annotation \*/

protected Boolean ComplexPrepositions = true;

public Boolean getIWePronouns() {

return IWePronouns;

}

@CreoleParameter(

comment = "Count I/We pronouns")

public void setIWePronouns(Boolean bool) {

this.IWePronouns = bool;

}

/\* check to count this type of annotation \*/

protected Boolean IWePronouns = true;

public Boolean getIntensifyingAdverbs() {

return IntensifyingAdverbs;

}

@CreoleParameter(

comment = "Count intensifying adverbs")

public void setIntensifyingAdverbs(Boolean bool) {

this.IntensifyingAdverbs = bool;

}

/\* check to count this type of annotation \*/

protected Boolean IntensifyingAdverbs = true;

public Boolean getOrSuffixNouns() {

return OrSuffixNouns;

}

@CreoleParameter(

comment = "Count nouns with -OR suffix")

public void setOrSuffixNouns(Boolean bool) {

this.OrSuffixNouns = bool;

}

/\* check to count this type of annotation \*/

protected Boolean OrSuffixNouns = true;

public Boolean getPassiveVoice() {

return PassiveVoice;

}

@CreoleParameter(

comment = "Count verbs in passive voice")

public void setPassiveVoice(Boolean bool) {

this.PassiveVoice = bool;

}

/\* check to count this type of annotation \*/

protected Boolean PassiveVoice = true;

public Boolean getPersonalPronouns() {

return PersonalPronouns;

}

@CreoleParameter(

comment = "Count personal pronouns")

public void setPersonalPronouns(Boolean bool) {

this.PersonalPronouns = bool;

}

/\* check to count this type of annotation \*/

protected Boolean PersonalPronouns = true;

public Boolean getPostpositiveAtrributes() {

return PostpositiveAtrributes;

}

@CreoleParameter(

comment = "Count postpositive attributes")

public void setPostpositiveAtrributes(Boolean bool) {

this.PostpositiveAtrributes = bool;

}

/\* check to count this type of annotation \*/

protected Boolean PostpositiveAtrributes = true;

public Boolean getPrepositiveAtrributes() {

return PrepositiveAtrributes;

}

@CreoleParameter(

comment = "Count prepositive attributes")

public void setPrepositiveAtrributes(Boolean bool) {

this.PrepositiveAtrributes = bool;

}

/\* check to count this type of annotation \*/

protected Boolean PrepositiveAtrributes = true;

public Boolean getThatThoseOf() {

return ThatThoseOf;

}

@CreoleParameter(

comment = "Count that of / those of")

public void setThatThoseOf(Boolean bool) {

this.ThatThoseOf = bool;

}

/\* check to count this type of annotation \*/

protected Boolean ThatThoseOf = true;

public Boolean getVerbTense() {

return VerbTense;

}

@CreoleParameter(

comment = "Count verb tenses")

public void setVerbTense(Boolean bool) {

this.VerbTense = bool;

}

/\* check to count this type of annotation \*/

protected Boolean VerbTense = true;

public Boolean getYouHeShe() {

return YouHeShe;

}

@CreoleParameter(

comment = "Count you/he/she")

public void setYouHeShe(Boolean bool) {

this.YouHeShe = bool;

}

/\* check to count this type of annotation \*/

protected Boolean YouHeShe = true;

public String getCorp(String corpName) {

String outstr = "<Corpus name=\"" + corpName + "\">";

String filenamecorp = "C:/Users/gorbunova.m/Desktop/" + corpName + ".xml";

SAXReader reader = new SAXReader();

try {

org.dom4j.Document document = reader.read(filenamecorp);

org.dom4j.Element root = document.getRootElement();

outstr += "<Document name = \"" + filenamecorp + "\">";

for (Iterator x = root.elementIterator("AnnotationSet"); x.hasNext();) {

org.dom4j.Element annotationSet = (org.dom4j.Element) x.next();

for (Iterator z = annotationSet.elementIterator("Annotation"); z.hasNext();) {

org.dom4j.Element annotation = (org.dom4j.Element) z.next();

annotation.addAttribute("fileName", filenamecorp);

annotation.addAttribute("corpusName", corpName);

outstr += annotation.asXML();

}

}

outstr += "</Document>";

} catch (DocumentException ex) {

Logger.getLogger(StatsCollector.class.getName()).log(Level.SEVERE, null, ex);

}

outstr += "</Corpus>";

return outstr;

}

public void execute() {

if(isNeededNewTemplate)

{

File filercp = new File("C:\\Users\\gorbunova.m\\Desktop\\birt\\BIRT.exe");

try {

Desktop.getDesktop().open(filercp);

} catch (IOException ex) {

Logger.getLogger(StatsCollector.class.getName()).log(Level.SEVERE, null, ex);

}

}else{

String outstr = "<?xml version='1.0' encoding='windows-1251'?>";

String filename1 = "corpora";

FileWriter fw1 = null;

if (corpList.size() != 0&&corpList.size()!=1) {

filename1 = "corpora";

outstr += "<Corporas>";

for (Iterator w = corpList.iterator(); w.hasNext();) {

String vv = (String) w.next();

outstr += getCorp(vv);

}

outstr += "</Corporas>";

} else {

Corpus corp = getCorpus();

//outstr+=getCorp(corp.getName());

FileWriter fw = null;

FeatureMap cf = corp.getFeatures();

int s = corp.size(); //size of corpus

gate.Document currDoc = getDocument();

int d = corp.indexOf(currDoc); //index of current document

if (d == 0) //if first - create the features

{

try {

outstr += "<Corpus name=\"" + corp.getName() + "\">";

for (int i = 0; i < corp.size(); i++) {

String filename = "C:/Users/gorbunova.m/Desktop/" + corp.get(i).getName() + ".xml";

File f = new File(filename);

if (f.exists()) {

f.delete();

}

f.createNewFile();

fw = new FileWriter(f, true);

PrintWriter pw = new PrintWriter(fw);

String str = corp.get(i).toXml();

pw.print(str);

pw.close();

fw.close();

SAXReader reader = new SAXReader();

try {

org.dom4j.Document document = reader.read(filename);

org.dom4j.Element root = document.getRootElement();

outstr += "<Document name = \"" + filename + "\">";

for (Iterator x = root.elementIterator("AnnotationSet"); x.hasNext();) {

org.dom4j.Element annotationSet = (org.dom4j.Element) x.next();

for (Iterator z = annotationSet.elementIterator("Annotation"); z.hasNext();) {

org.dom4j.Element annotation = (org.dom4j.Element) z.next();

annotation.addAttribute("fileName", filename);

annotation.addAttribute("corpusName", corp.getName());

outstr += annotation.asXML();

}

}

outstr += "</Document>";

} catch (DocumentException ex) {

Logger.getLogger(StatsCollector.class.getName()).log(Level.SEVERE, null, ex);

}

}

outstr += "</Corpus>";

filename1 = "C:/Users/gorbunova.m/Desktop/" + corp.getName() + ".xml";

} catch (IOException ex) {

Logger.getLogger(StatsCollector.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

File f1 = new File(filename1);

try {

if (f1.exists()) {

f1.delete();

}

f1.createNewFile();

fw1 = new FileWriter(f1, true);

PrintWriter pw1 = new PrintWriter(fw1);

pw1.print(outstr);

pw1.close();

fw1.close();

} catch (IOException ex) {

Logger.getLogger(StatsCollector.class.getName()).log(Level.SEVERE, null, ex);

}

try {

ReportGenerator report =new ReportGenerator(outputfileURL, templateFileURL, outputFormat);

} catch (BirtException ex) {

Logger.getLogger(StatsCollector.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

} // class StatsCollector

class ReportGenerator {

public ReportGenerator(URL outputfileURL, URL templatefileURL, String outputFormat) throws BirtException {

boolean DEBUG\_MODE = true;

EngineConfig config = new EngineConfig();

config.setEngineHome("C:\\Users\\gorbunova.m\\Downloads\\birt-runtime-4.6.0-20160607\\ReportEngine");

Platform.startup(config);

ReportEngine engine = new ReportEngine(config);

if (DEBUG\_MODE) {

Out.pr(outputfileURL + "/StatsXML.xml");

}

templatefileURL.getPath();

IReportRunnable reportRunnable = engine.openReportDesign(templatefileURL.getPath());

IRunAndRenderTask runAndRender = engine.createRunAndRenderTask(reportRunnable);

ReportDesignHandle report = (ReportDesignHandle) reportRunnable.getDesignHandle();

ElementFactory ef = report.getElementFactory();

OdaDataSourceHandle datasource = (OdaDataSourceHandle) ef.newOdaDataSource("DataSource", outputfileURL.getPath() + "StatsXML.xml");

RenderOption option = new RenderOption();

switch (outputFormat) {

case "HTML":

option.setOutputFormat("HTML");

break;

case "PDF":

option.setOutputFormat("PDF");

break;

case "TXT":

option.setOutputFormat("TXT");

break;

case "WORD":

option.setOutputFormat("WORD");

break;

default:

option.setOutputFormat("HTML");

}

option.setOutputFileName(outputfileURL.getPath()+"testReport."+option.getOutputFormat());

runAndRender.setRenderOption(option);

runAndRender.run();

runAndRender.close();

engine.destroy();

Platform.shutdown();

}

}

# Приложение В. Руководство пользователя по установке и настройке программного модуля StatsCollector

Краткое руководство пользования плагином и шаблоном отчета. Файлы необходимые для работы плагина прикреплены к выпускной работе.

1. Скопировать папки из StatsCollector в C:\Program Files (x86) \GATE\_Developer\_8.0\plugins (C:\Program Files (x86) может быть заменено на место установки GATE Developer).
2. Запустить GATE Developer и выбрать File> Manage CREOLE Plugins.
3. Загрузить плагин, нажимая на значок "+" и указывая путь к папке, содержащей ресурс.
4. В окне CREOLE Plugins Manager поставить галочку напротив StatsCollector.
5. Обрабатывающий ресурс для приложения создаётся через Processing Resources> New> StatsCollector. Необходимо указать входные данные, а именно маркеры, статистику для которых необходимо собрать.
6. Плагин должен быть применен к уже аннотированным корпусам, корпусу или документу.
7. Указать Runtime параметры.
8. Запустить плагин через нажатие кнопки “Run this Application”.
9. Если параметр isNeededNewTemplate установлен в значении true, то откроется BIRT RCP Report Designer, через который можно построить новый шаблон отчета (процесс был описан ранее).
10. В противном случае, плагин начнет работу, и экспортирует файл со сгенерированным отчетом в указанную директорию.

Для работы с реализованными плагинами использовалась Java Version 8 Update 20 (build 1.8.0\_20-b26). Подробнее о создании приложений и загрузке корпусов в GATE Developer можно узнать на сайте gate.ac.uk и в руководстве пользователя.