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

Факультет экономики, менеджмента и бизнес-информатики

Фалалеева Вероника Сергеевна

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АНАЛИЗА ПУБЛИКАЦИЙ НА АНГЛИЙСКОМ ЯЗЫКЕ НА СООТВЕТСТВИЕ НАУЧНОМУ СТИЛЮ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

Выпускная квалификационная работа

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

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

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

Пермь, 2018 год

Аннотация

Данная работа посвящена разработке системы анализа научных публикаций на английском языке на соответствие научному стилю с использованием методов машинного обучения.

Первая глава посвящена анализу существующих решений, помогающих решить проблему отсутствия программного продукта для анализа публикаций на английском языке на соответствие научному стилю. Также в данной главе проанализированы методы машинного обучения, выбраны, спроектированы и разработаны некоторые из них: метод опорных векторов, логистическая регрессия, метод деревьев решений, нейронные сети, наивный байесовский метод.

Во второй главе приводится описание проектирования программного продукта, автоматизирующего процесс проверки текста на соответствие научному стилю, технико-экономическое обоснование, выявлены риски проекта.

В третьей главе описывается реализация основного модуля программного продукта. Также в приложениях написаны техническое задание, тестовые сценарии, руководство пользователя.

Оглавление

Список сокращений и терминов

МО – машинное обучение.

ПП – программный продукт.

Классификация текстов – процесс распределения текстов на естественном языке по категориям из заранее определенного набора.

Классификатор – это алгоритм, соотносящий некие входные данные с одним или несколькими классами. В отличие от алгоритмов кластеризации эти классы должны быть определены заранее.

Нейронные сети (также называемые искусственные нейронные сети) – это система отдельных процессоров (нейронов), связанных определенным структурированным образом. Они предназначены для имитации работы человеческого мозга. Каждый отдельный нейронный блок имеет функцию, которая обрабатывает значения всех его входов.

Введение

В настоящее время в мире насчитывается около 35 тысяч научных рецензируемых журналов, большинство из которых имеет строгие требования к статьям, которые они публикуют. Главная причина жестких требований состоит в поддержании репутации издания. Это напрямую связано с привлечением новых авторов для научных публикаций, так как чем более надежный и авторитетный научный источник, тем большее количество людей будет заинтересовано в публикации именно в нем. Данные требования в большинстве случаев достаточно схожи между собой, но так как нет международного единого документа, описывающего основы академического стиля в публикациях, то каждое издание самостоятельно решает и выдвигает требования.

Проверка и исправление всех возможных правил занимает достаточно большое количество времени. Причем, для людей, не являющихся носителями английского языка и не имеющих филологическое образование, выполнить проверку является действительно сложным процессом. Именно поэтому существуют различные сервисы, помогающие ускорить процесс проверки. Все подобные сайты, которые предлагают проанализировать текст, действительно могут помочь в проверке статьи, но они решают данную проблему лишь частично. Именно поэтому проблема отсутствия какого-либо программного продукта, автоматизирующего проверку научной статьи на соответствие научному стилю, действительно является актуальной.

Данная проблема может быть решена с помощью методов машинного обучения. Используя различные методы машинного обучения можно получить результат сравнения академического уровня английского языка в загруженной в систему статье с желаемым корпусом статей. В данном случае под корпусом статей подразумеваются примеры для обучения системы в виде уже опубликованных ранее статей в научных издательствах.

Метод машинного обучения сводится к решению проблемы текстовой классификации. Данное решение заключается в разделении научных статей на классы. Каждый класс характеризуется маркерами, которые определяют научный стиль текста.

В данной выпускной квалификационной работе описана реализация приложения для анализа публикаций на английском языке на соответствие научному стилю.

Объектом ВКР является сравнение стилистических характеристик текста, а предметом – стилистический анализ текста.

Цель данной работы заключается в разработке программного продукта, анализирующего текст научных публикаций на английском языке на соответствие научному стилю. Для достижения поставленной цели требуется выполнить следующие задачи:

1) анализ существующих методов проверки текста на соответствие научному стилю и выбор основных характеристик для данной проверки;
2) анализ методов решения проблемы отсутствия анализатора текста и выбор наиболее подходящего для разработки;
3) проектирование программного проекта анализа публикаций на английском языке на соответствие академическому стилю;
4) разработка программного проекта анализа публикаций, позволяющего загружать текст на проверку соответствия академическому стилю и получать результат в виде положительного или отрицательного ответа на вопрос о публикации;
5) тестирование и отладка разработанного программного проекта анализа публикаций.

Практическая значимость данной работы определяется несколькими аспектами. Главный из них – научный аспект. Разработанная система позволит проверять научные публикации на соответствие академическому стилю самым быстрым и удобным способом, несмотря на жесткие требования издателя. Это приложение не только сократит время на проверку, но и даст возможность более тщательно подготовить материал публикации, поэтому количество научных статей будет только увеличиваться и повысит их научный уровень.

Еще одним аспектом, который входит в практическую значимость, является экономический аспект. Причинами этого являются расходы на проверку научных публикаций. Некоторые люди платят, чтобы быть уверенными в уровне английского языка для академических целей в своих статьях. Разработанная система позволит сэкономить эти деньги.

Глава 1. Изучение предметной области

В данной главе описаны основные существующие решения, маркеры, характеризующие научный стиль текста на английском языке, формализована главная задача работы, выполнено сравнение существующих методов машинного обучения.

1.1. Анализ существующих решений

На данный момент в открытом доступе не найдено полностью аналогичных программных продуктов. Однако найдены две системы [25,26], которые помогают решить проблему соответствия научному стилю. Первый программный продукт называется «Grammarly». Это веб-сайт, в котором можно загрузить любой текст и проверить его на соответствие формальному стилю письма, а также выбрать тему для лучшей проверки текста. После загрузки текста в систему пользователь может увидеть оценку своего текста, которая показывает процентное соотношение баллов, полученных в результате проверки текста по сравнению с работами загруженными на «Grammarly», то есть 100% - отличная работа без ошибок, а 0% - полное несоответствие. Проверка на соответствие выбранным теме и стилю является не основной функцией сервиса, его основная задача – проверка грамматики, поэтому данная оценка в процентном соотношении не может являться точной, так как она выражает общую оценку работы исходя из различных характеристик, таких как: грамматика, пунктуация, неправильное употребление слова в контексте, неправильный порядок слов, а только после этого идет соответствие теме и формальному стилю письма. Пример работы программы представлен на рисунке 1.1. Данный рисунок показывает оценку пользователю в правом верхнем углу.

Следующая система называется «STATISTICA» от компании StatSoft. Это большая группа различных систем, которые занимаются различными функциями, такими как: data mining, анализ данных, контроль качества, прогнозирование, обучение, консалтинг и многое другое. Один из продуктов данной компании называется «STATISTICA Text Miner», который занимается задачами структуризации текста и анализа выявленных из него данных. Например, пользователь может оценить степень похожести документов. В таком случае пользователь может выбрать статью, уже

напечатанную в издании, в котором он бы хотел опубликовать свою статью и сравнить ее с текстом своей работы. Пример работы программы представлен на рисунках 1.2-1.3.

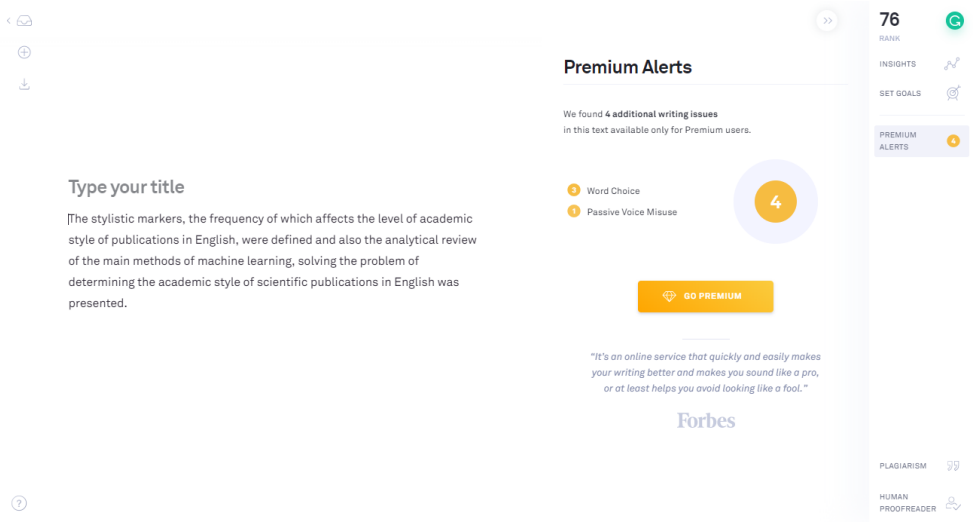

Рисунок 1.1. Пример работы программы «Grammarly»

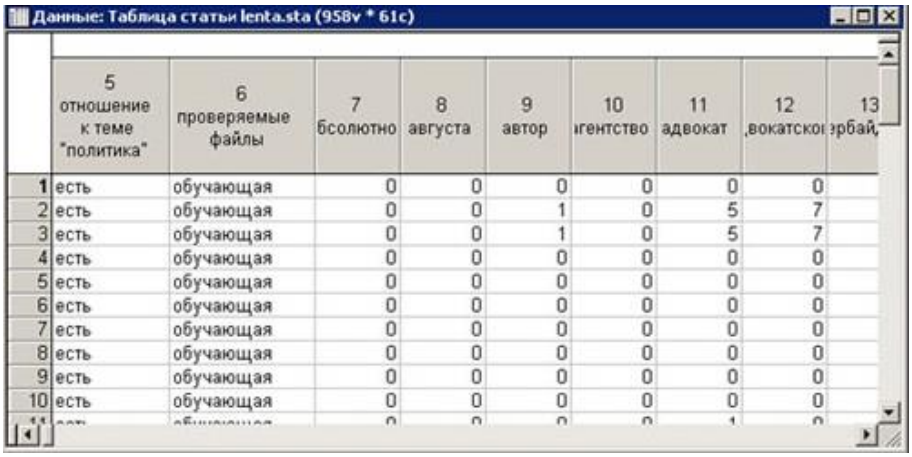

Рисунок 1.2. Пример работы программы «STATISTICA Text Miner»

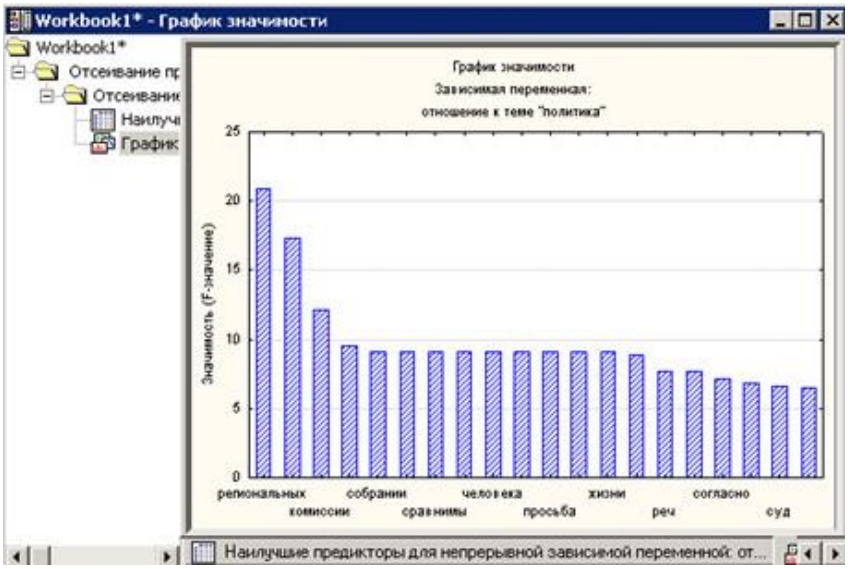

Рисунок 1.3. Пример работы программы «STATISTICA Text Miner»

1.2. Формализация задачи анализа научных публикаций

На данный момент существует большое многообразие методов, основанных на машинном обучении. Каждый подход обладает своими преимуществами и недостатками, которые рассмотрены в данной подглаве.

Алгоритмы МО делятся на две большие группы [3,16-18]:

1. Обучение без учителя.
2. Обучение с учителем.

Главная задача, решаемая алгоритмами машинного обучения, заключается в соотнесении объекта к той или иной группе объектов В целом, это анализ данных в различных информационных системах, позволяющий выполнять предсказания состояний или классификацию объектов.

Алгоритмы, основанные на обучении без учителя – это алгоритмы, решающие задачу кластеризации, то есть множество заранее не обозначенных объектов разбивается на группы путем автоматической процедуры, исходя из свойств данных объектов.

Данная задача не подходит для разрабатываемой системы, так как публикации, на которых производится обучение системы делятся на две большие группы: «компетентные» публикации и публикации «некомпетентные». Первая группа состоит из статей на английском языке, которые уже были опубликованы в научных журналах, а вторая группа состоит из статей, которые не являются научными (статьи из ненаучных или научно-популярных журналов). Поэтому задача данной работы сводится к задаче классификации, где алгоритм основан на обучении с учителем.

Алгоритмы, основанные на обучении с учителем – это алгоритмы, решающие задачу классификации, то есть множества объектов заранее разделены на некие классы некоторым образом. Формирование данных классов выполняется экспертом.

Алгоритм классификации должен, используя первоначальную классификацию как образец, отнести следующие необозначенные объекты к той или иной организованной экспертом группе исходя из свойств этих объектов. Алгоритмы классификации включают большой набор алгоритмов или семейств алгоритмов, которые часто разделяются на линейные и нелинейные классификаторы, в зависимости от формы (гиперплоскости или гиперповерхности) разделяющие классы объектов. В двумерном

случае линейные классификаторы разделяют классы единственной прямой, тогда как нелинейные классификаторы — линией, как показано на рисунке 1.1.

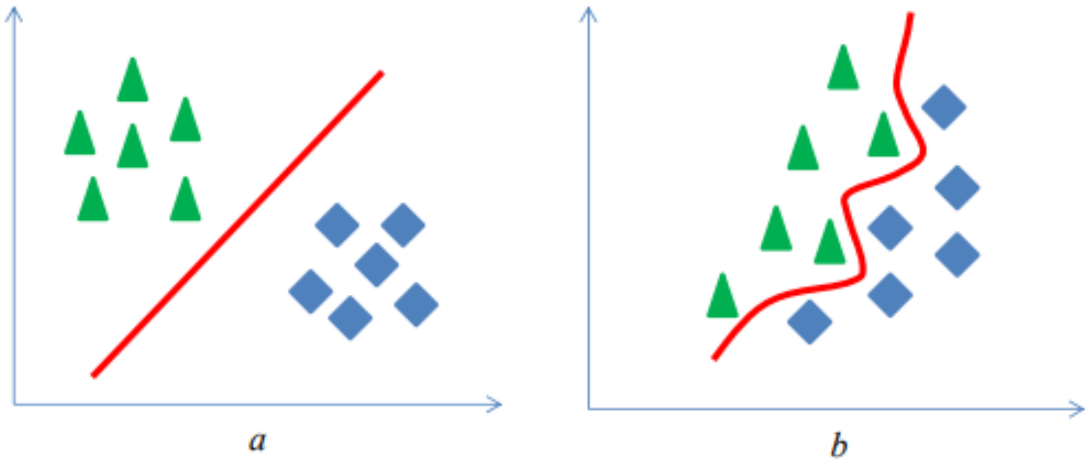

Рисунок 1.4. Линейный (a) и нелинейный (b) классификаторы

Алгоритмы обучения без учителя (линейные классификаторы) имеют следующую структуру:

1) Логическая регрессия – Logical regression (LR);

2) Наивный байесовский метод – Naive bayes Classifier (NBC);

3) Персептрон – Perceptron (P).

Алгоритмы обучения с учителем (нелинейные классификаторы) имеют следующую структуру:

1) Метод опорных векторов – Support Vector Classification (SVM);

2) Логистическая регрессия – Logistic regression (LogR);

3) Лес деревьев решений – Decision Tree (DT);

4) Нейронные сети – Neural Networks (NN);

5) Байесовский метод – Bayesian Networks (BN).

Так как в разрабатываемой системе наборы данных имеют высокую дисперсию, то необходимо использование алгоритмов обучения с учителем, использующих нелинейные классификаторы.

Также для формализации задачи классификации текстов, необходимо рассмотреть основные проблемы, которые могут встретиться в процессе разработки данной работы.

Прежде всего результат сравнения на соответствие научному стилю не может быть окончательным и однозначным, так как в разные издательства могут быть приняты разные статьи с абсолютно разными характеристиками и частотностью встречаемости

маркеров, приведенных в подглаве 1.3. Также текстовые документы могут содержать грамматические ошибки или даже опечатки, которые будут неправильно поняты программой.

Поэтому разработанное приложение не гарантирует, что его результат будет правильным с вероятностью в 100%. Это также связано с некоторыми ограничениями, которые наложены на данный проект. Основное из них заключается в ограниченной выборке данных: чем больше входных данных для разработанного приложения, тем точнее результат. Поэтому необходимо получить как можно больше статей для изучения. В данной работе для обучения будут использоваться ранее опубликованные статьи в известных английских издательствах, а для тестирования и прогнозирования результатов – студенческие статьи.

Стоит отметить, что для лучшего анализа данных пользователю будут предложены несколько корпусов статей. На данный момент их 5: право, программная инженерия и бизнес-информатика, история, политология, экономика. Пользователь самостоятельно должен выбрать корпус, с которым он хочет сравнить свою публикацию. Система должна рассчитывать некий коэффициент соответствия загруженной пользователем публикации выбранному корпусу.

Таким образом, задача анализа научных публикаций на английском языке на соответствие научному стилю сводится к задаче классификации. Публикация классифицируется с помощью методов машинного обучения, рассмотренных ранее, на два класса: соответствующий уровень академического стиля (коэффициент соответствия равен 1) или несоответствующий (коэффициент соответствия равен 0).

1.3. Описание маркеров для классификации текста

Для классификации текста выбраны следующие маркеры, которые характеризуют научный стиль текста [1,2, 19-24].

1. Архаизмы.
2. Личные местоимения, так как авторы не должны показывать субъективность в своей работе.
3. Сложные выражения – данный вид тенденции в научных статьях присутствует достаточно часто (это необязательно положительное явление, но распространенное в академических текстах)..

4. Десемантические глаголы: may, would, suggest, could, consider и тому подобные..
5. Глаголы абстрактной семантики, такие как: be, exist, have, appear, occur, alter, continue, contribute, discuss, involve, investigate, conduct, consider, illustrate, assume, find, calculate, demonstrate, identify, analyse, support, challenge, examine, affect, provide, include, classify, establish.
6. Будущее время.
7. Пассивный залог, так как он чаще используется в точных науках, а в гуманитарных, наоборот, гораздо реже.
8. Существительные.
9. Усилительные наречия, такие как: clearly, dramatically, completely, considerably, essentially, significantly, markedly, perfectly, так как любое наречие, которое не требуется для лучшего понимания темы лучше исключить.
10. Некоторые суффиксы.
11. Препозитивные определения, например: theoretical background; system perspective.
12. Прошедшее время.
13. Местоимение «I».
14. Постпозитивные определения, например: perspective of members; differences between the internalist and the externalist viewpoints; a strategic approach to mutual understanding.
15. Настоящее время.
16. Средства логической связи, такие как: however, moreover, furthermore, nevertheless и тому подобные.
17. Указательные местоимения.
18. Местоимение «We».
19. Местоимения «You», «He», «She».

Проанализировав корпус научных статей и сравнив его с корпусом статей, написанных студентами НИУ ВШЭ-Пермь, можно будет сделать вывод по поводу уровня академического стиля статьи. Например, в таблице 1.1 приведены примеры сравнения встречаемости средств логической связи. Исходя из данных, представленных

в данной таблице, можно сделать вывод, что чем сложнее средство логической связи, тем реже оно встречается в научной статье.

Таблица 1.1. Частотность встречаемости средств логической связи

Средство логической связи	Корпус научных статей		Корпус статей студентов	
	Общее кол-во	Кол-во на 1000 слов	Общее кол-во	Кол-во на 1000 слов
However	707	1.02	145	1.16
Moreover	178	0.26	124	0.95
Furthermore	183	0.26	23	0.18
Nevertheless	42	0.06	25	0.19

На рисунке 1.5. представлен график, показывающий сравнение средних значений коэффициента количества маркеров на 1000 слов корпуса «Экономика».

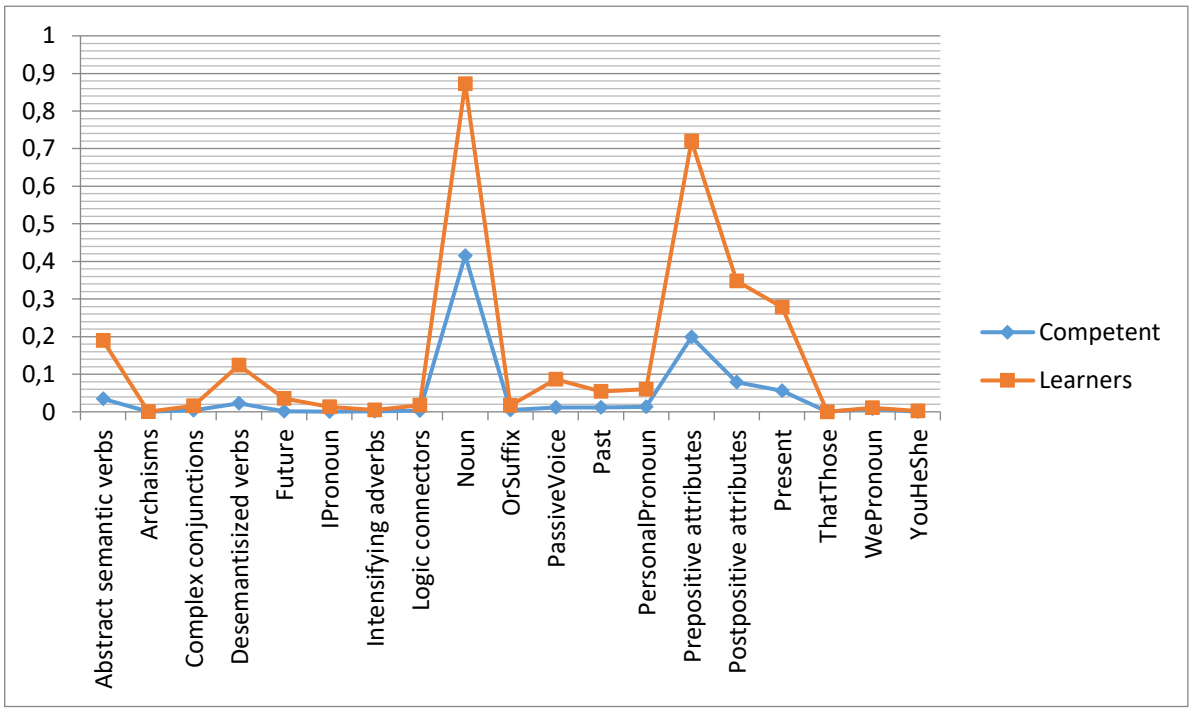

Рисунок 1.5. Сравнение средних значений маркеров корпуса «Экономика»

1.4. Анализ основных алгоритмов обучения с учителем

1.4.1. Описание метода опорных векторов

Метод опорных векторов (Support Vector Machines) – это хорошо изученный класс контролируемых методов обучения. Данная реализация подходит для прогнозирования двух возможных результатов, основанных на непрерывных или категориальных переменных. Метод опорных векторов является одним из самых ранних алгоритмов машинного обучения, а модели SVM использовались во многих приложениях, от поиска информации до классификации текста и изображений. SVM можно использовать как для задач классификации, так и регрессии [4].

Модель SVM является контролируемой моделью обучения, которая требует помеченных данных. В процессе обучения алгоритм анализирует входные данные и распознает закономерности в многомерном пространстве объектов, называемом гиперплоскостью. Все входные примеры представлены в виде точек в этом пространстве и сопоставлены с выходными категориями таким образом, что категории делятся на максимально широкие и четкие промежутки, как показано на рисунке 1.6. Для прогнозирования алгоритм SVM назначает новые примеры в ту или иную категорию, сопоставляя их в том же пространстве. Метод опорных векторов может быстро разделить классы и отличается минимальной вероятностью создания ложной связи, а также не требует больших объемов памяти.

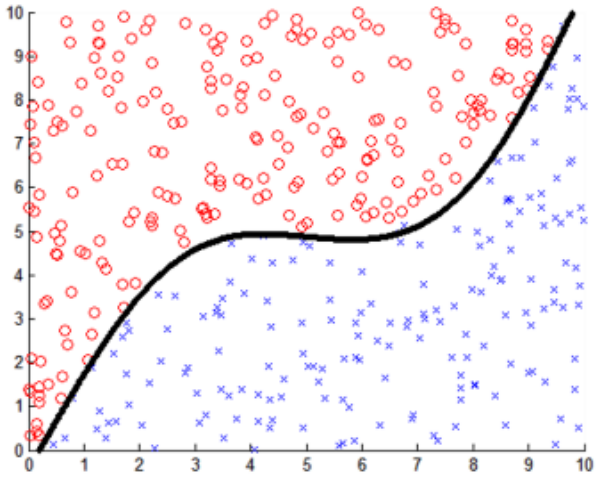

Рисунок 1.6. График метода опорных векторов

1.4.2. Описание метода логистической регрессии

Логистическая регрессия является хорошо известным методом в статистике, который используется для прогнозирования вероятности результата и особенно популярен для задач классификации. Алгоритм предсказывает вероятность возникновения события путем подгонки данных к логистической функции. Использование данного алгоритма снижает точность. Работа алгоритма представлена на рисунке 1.7.

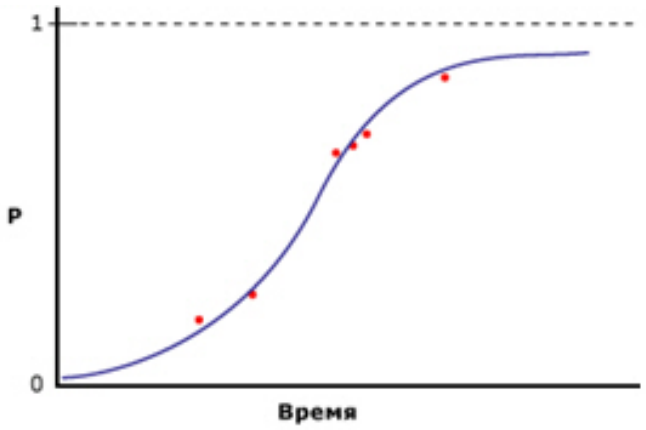

Рисунок 1.7. График алгоритма логистической регрессии

1.4.3. Описание метода деревьев решений

В наиболее простом виде дерево решений – это способ представления правил в иерархической, последовательной структуре. Основа такой структуры - ответы "Да" или "Нет" на ряд вопросов. Решение задачи классификации заключается в определении значения категориального (дискретного) выходного атрибута на основании входных данных. Для этого сначала производится оценка степени корреляции входных и выходных значений, после чего обнаруженные зависимости описываются в виде узлов дерева.

Деревья решений в целом имеют много преимуществ для задач классификации:

1) они могут захватывать нелинейные границы решения;
2) они имеют возможность обучаться и прогнозировать множество данных, так как они эффективны в вычислениях и использовании памяти;
3) выбор признаков интегрирован в процессы обучения и классификации;
4) деревья хорошо обрабатывают «зашумленные» данные и большое количество параметров.

Однако простые деревья решений могут перекрывать данные и являются менее обобщаемыми, чем древовидные ансамбли. Работа данного алгоритма представлена на рисунке 1.8.

Рисунок 1.8. График работы алгоритма дерева решений

1.4.4. Описание метода нейронных сетей

Классификация с использованием нейронных сетей является контролируемым методом обучения и, следовательно, требует набора данных с метками. Нейронная сеть – это набор взаимосвязанных слоев. Входные данные являются первым слоем и соединены с выходным слоем ациклическим графом, состоящим из взвешенных ребер и узлов.

Между входным и выходным слоями можно вставить несколько скрытых слоев. Большинство прогнозных задач можно легко выполнить только с одним или несколькими скрытыми слоями. Однако недавние исследования показали, что глубокие нейронные сети со многими слоями могут быть очень эффективными в сложных задачах, таких как распознавание изображений или речи.

Взаимосвязь между входами и выходами изучается при обучении нейронной сети входным данным. Направление графика происходит от входных данных через скрытый слой и к выходному слою. Все узлы в слое соединяются взвешенными ребрами с узлами в следующем слое.

Для вычисления выходных данных сети для определенных входных данных значение вычисляется на каждом узле в скрытых слоях и в выходном слое. Значение устанавливается путем вычисления взвешенной суммы значений узлов из предыдущего слоя. Затем к этой взвешенной сумме применяется функция активации (см. рис. 1.9).

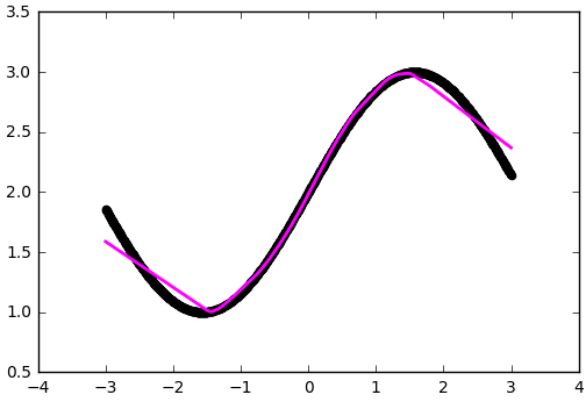

Рисунок 1.9. Алгоритм работы нейронных сетей

1.4.5. Описание Наивного Байесовского метода

Упрощенный алгоритм Байеса – это алгоритм классификации, основанный на вычислении условной вероятности значений прогнозируемых атрибутов. При этом и предполагается, что входные атрибуты являются независимыми и определен хотя бы один выходной атрибут.

Байесовский подход к линейной классификации, называемый "машиной Байеса" эффективно аппроксимирует теоретически оптимальное Байесовское среднее линейных классификаторов (в терминах производительности обобщения), выбирая один "средний" классификатор, точку Байеса. Поскольку точечная машина Байеса является Байесовской классификационной моделью, она не склонна к переобучению обучающих данных.

Для корректного использования упрощенного алгоритма Байеса необходимо учитывать, что:

1) входные атрибуты должны быть взаимно независимыми;
2) атрибуты могут быть только дискретными или дискретизированными (в процессе дискретизации множество значений непрерывного числового атрибута разбивается на интервалы и дальше идет работа с номером интервала);
3) алгоритм требует меньшего количества вычислений, чем другие алгоритмы интеллектуального анализа, поэтому он часто используется для первоначального исследования данных. По той же причине, данный алгоритм предпочтителен для анализа больших наборов данных с большим числом входных атрибутов.

1.4.6. Сравнение методов машинного обучения

Один из составляющих факторов оценки качества прогнозирования алгоритма является определение количества обучающих примеров. Автор работы [12] утверждает, что для успешного обучения модели необходимо:

$$Q = 7 \times N_x + 15, \quad (1.1)$$

где N_x – количество входных параметров модели; Q – количество примеров, обучающего множества.

После подстановки в формулу 1.1 количество примеров обучающего множества равно 148, так как количество входных параметров модели, которое в данном случае равно количеству выделенных маркеров, обсуждаемых в пункте 1.1, равно 19.

В таблице 1.3. приведено сравнение точности методов машинного обучения, рассматриваемых в данной работе. Данные в таблице показывают, что нейронные сети и наивный байесовский метод являются наиболее точными по сравнению с остальными методами машинного обучения, однако нейронные сети требуют меньшее количество примеров, что в данном случае является актуальным преимуществом в связи с недостатком количества обучающих примеров в доступных исследовательских корпусах.

Таблица 1.2. Сравнение точности методов машинного обучения

Метод	**Оценка правильности**	**Количество примеров**
Логистическая регрессия	0.974	150
Метод опорных векторов	0.946	148
Лес деревьев решений	0.811	148
Нейронные сети	1.000	110
Наивный Байесовский метод	1.000	150

Сравнение методов машинного обучения для решения задачи классификации текстов представлено в таблице 1.3.

Таблица 1.3. Сравнение методов машинного обучения

Алгоритм	**Точность**	**Время обучения**	**Линейность**	**Параметры**	**Примечание**
Логистическая регрессия	Плохая	Короткое	Используется	5	
Лес деревьев решений	Отличная	Среднее	Не используется	6	
Нейронная сеть	Отличная	Среднее	Не используется	9	Возможны дополнительные настройки

Алгоритм	Точность	Время обучения	Линейность	Параметры	Примечание
Метод опорных векторов	Плохая	Среднее	Используется	5	Предназначен для больших наборов данных
Байесовский метод	Плохая	Среднее	Используется	3	

1.4.7. Обзор инструментов разработки системы анализа публикаций на соответствие научному стилю

Машинное обучение – это хорошо признанная область исследований, которая охватывает широкий спектр приложений, содержащий множество различных алгоритмов и подходов.

Для разработки алгоритма машинного обучения используется высокоуровневый язык программирования, ориентированный на повышение производительности и читаемости кода. В то же время стандартная библиотека включает в себя большое количество полезных функций, которые помогают в развитии программы. Также Python позволяет создавать сложные веб-сервисы или интегрироваться в уже существующие системы.

Для графического описания объектного моделирования, моделирования бизнес-процессов, системного проектирования и картографирования организационной структуры используется унифицированный язык моделирования (язык UML). Он был выбран потому, что это простой метод для определения, визуализации, проектирования и документации программных систем.

В этой работе используется bottle web project. Bottle – это быстрый, простой и легкий WSGI micro Web-фреймворк для Python. Распространяется в виде одного файлового модуля и не имеет никаких зависимостей, кроме стандартной библиотеки Python.

1) Шаблоны: быстрый и встроенный шаблон на Python и поддержка шаблонов mako, jinja2 и cheetah.
2) Утилиты: удобный доступ к данным формы, загрузки файлов, cookies, заголовкам и другим метаданным, связанным с HTTP.
3) Сервер: Встроенный сервер развития HTTP и поддержка для paste, fapws3, bjoern, gae, cherrypy или любого другого сервера HTTP с поддержкой WSGI.

Глава 2. Проектирование системы анализа публикаций на английском языке на соответствие научному стилю

Проектирование системы анализа публикаций на английском языке на соответствие научному стилю включает в себя несколько основных подразделов: общее описание системы, архитектура проектируемой программы, анализ и оценка рисков, технико-экономическое обоснование.

2.1. Проектирование поведения системы

Для начала необходимо рассмотреть варианты использования системы с помощью диаграммы прецедентов, представленной на рисунке 2.1. Данная диаграмма составляет модель прецедентов, которые являются значимыми с точки зрения пользователя и имеют для него измеримый результат. Каждый прецедент соответствует отдельному сервису, предоставляемому моделируемой системой в ответ на запрос пользователя, т. е. определяет способ использования этой системы.

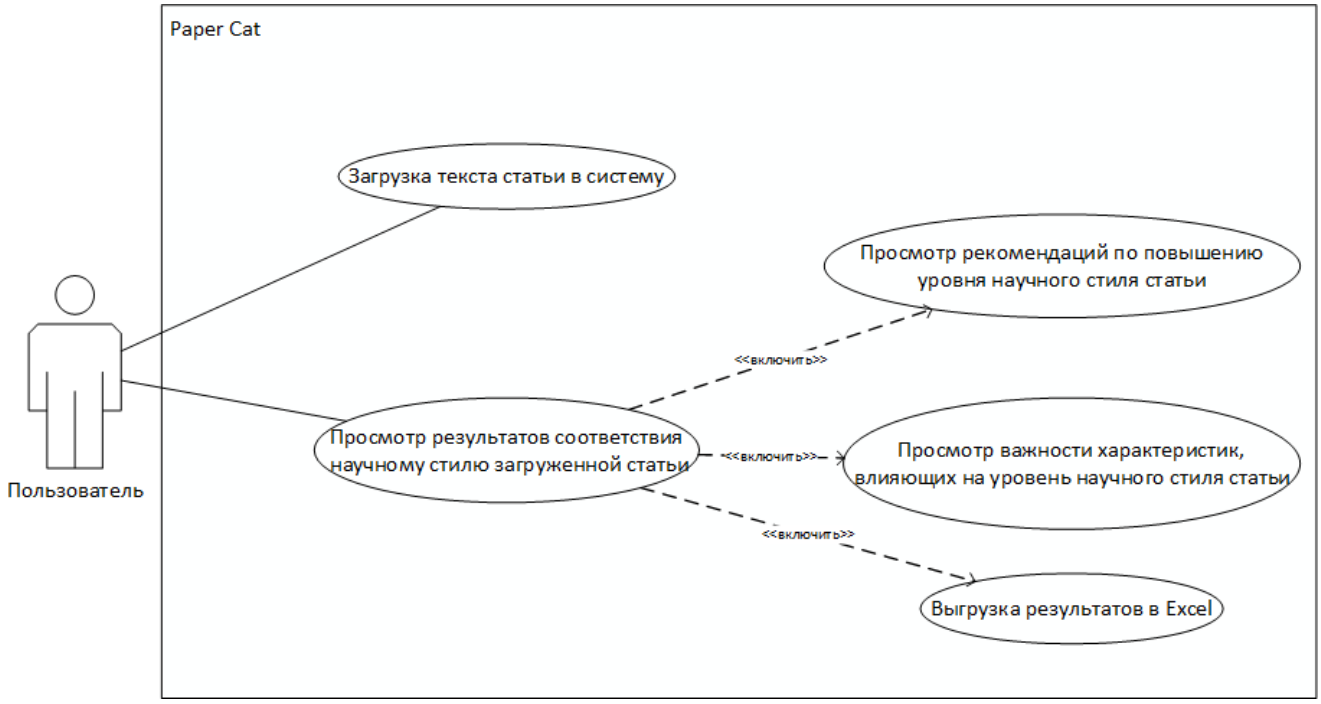

Рисунок 2.1. Диаграмма прецедентов для системы анализа текстов на соответствие научному стилю

Так как диаграмма описывает бизнес-процесс с точки зрения пользователя, необходимо использовать диаграммы активностей для более детального описания бизнес-процесса с точки зрения не только пользователя, но и системы. На рисунке 2.2.

представлена диаграмма активностей для прецедентов, описанных в диаграмме вариантов использования (рис. 2.1.).

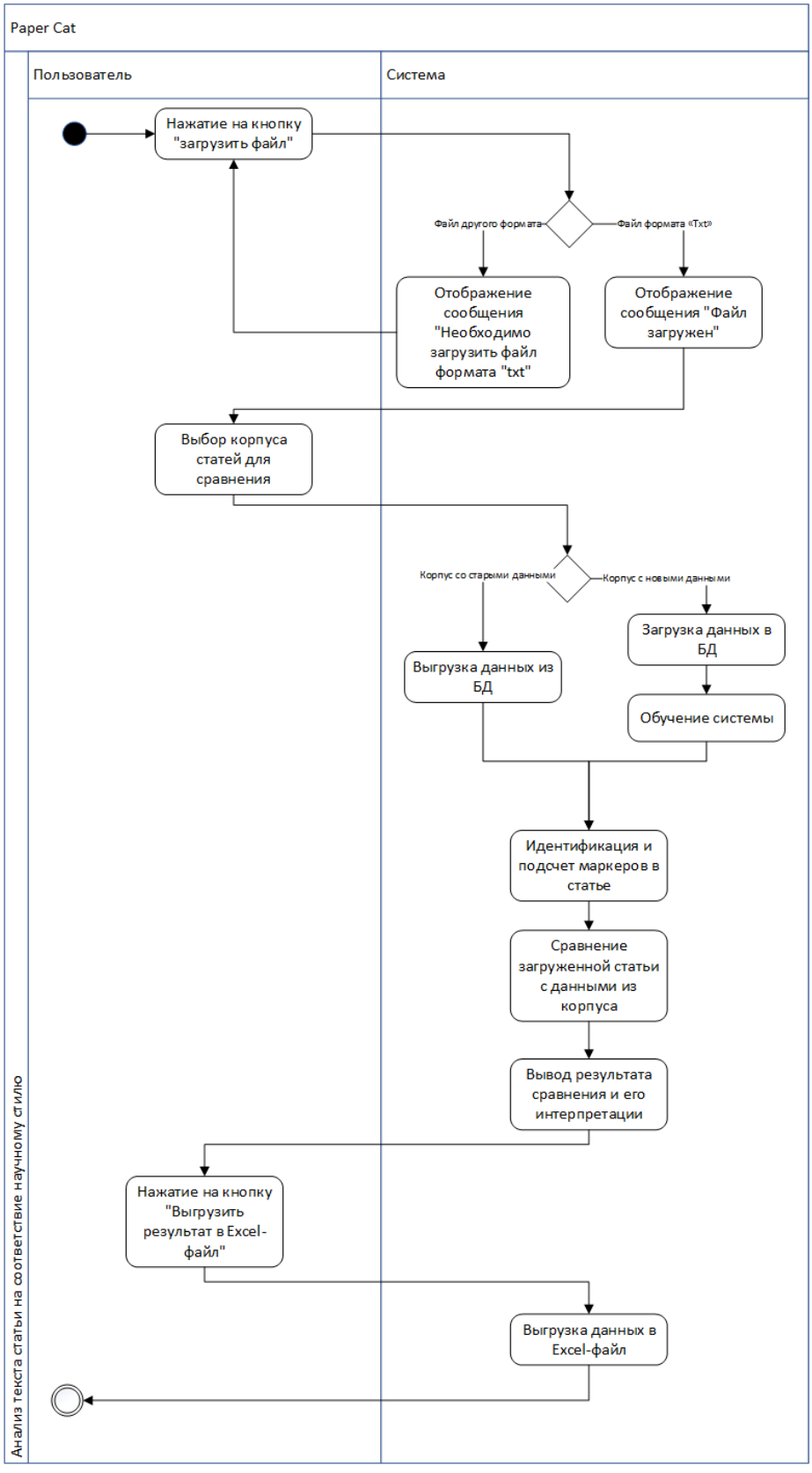

Рисунок 2.2. Диаграмма активностей для системы анализа текстов на соответствие научному стилю

Для полного понимания работы системы спроектирован пользовательский интерфейс, который представлен на рисунке 2.3.

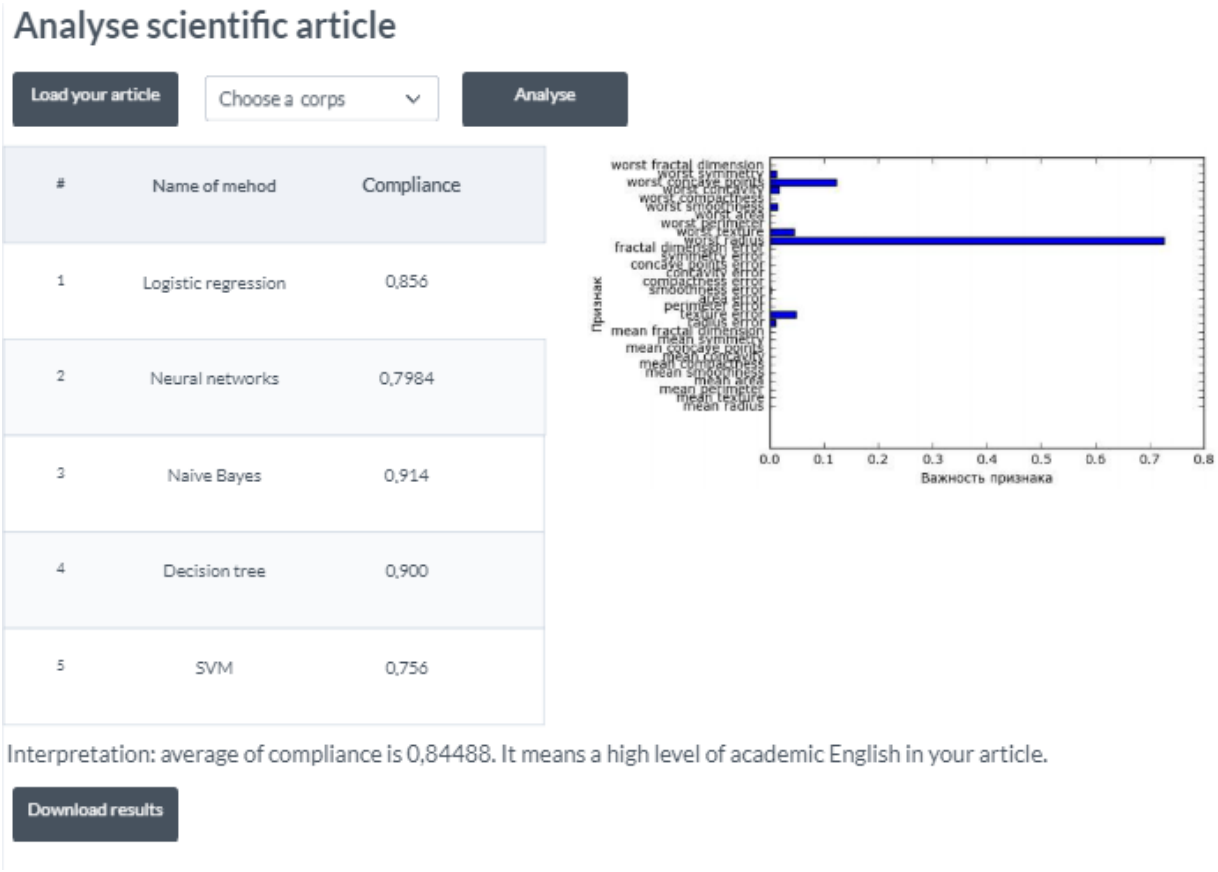

Рисунок 2.3. Проектирование интерфейса системы анализа текстов на соответствие научному стилю

2.2. Проектирование архитектуры компонент системы

Проект с открытым кодом scilit-learn [14], содержащий большое количество современных алгоритмов машинного обучения также использован в данной работе. Однако данная библиотека требует наличия нескольких дополнительных библиотек: matplotlib [15], IPython и Jupyter Notebook. Перечисленные библиотеки представлены в программе для научных вычислений Anaconda, которая используется для разработки алгоритма машинного обучения.

В таблице 1.4. представлены программные пакеты, которые использовались при разработке приложения.

Таблица 2.1. Описание использованных пакетов разработки

Название пакета	Описание	Язык программирования
MatplotLib	библиотека для построения математических графиков, в которую входят такие графики, как: линейные диаграммы, гистограммы и диаграммы разброса	Python

Название пакета	Описание	Язык программирования
NumPy	основной пакет для научных вычислений на языке Python	C, C++, Fortran
SciPy	набор функций для научных вычислений на языке Python	C, C++, Fortran
SciKit-learn	пакет для использования методов машинного обучения	Python, Cython

Архитектура системы анализа текстов на соответствие научному стилю представлена на рисунке 2.4.

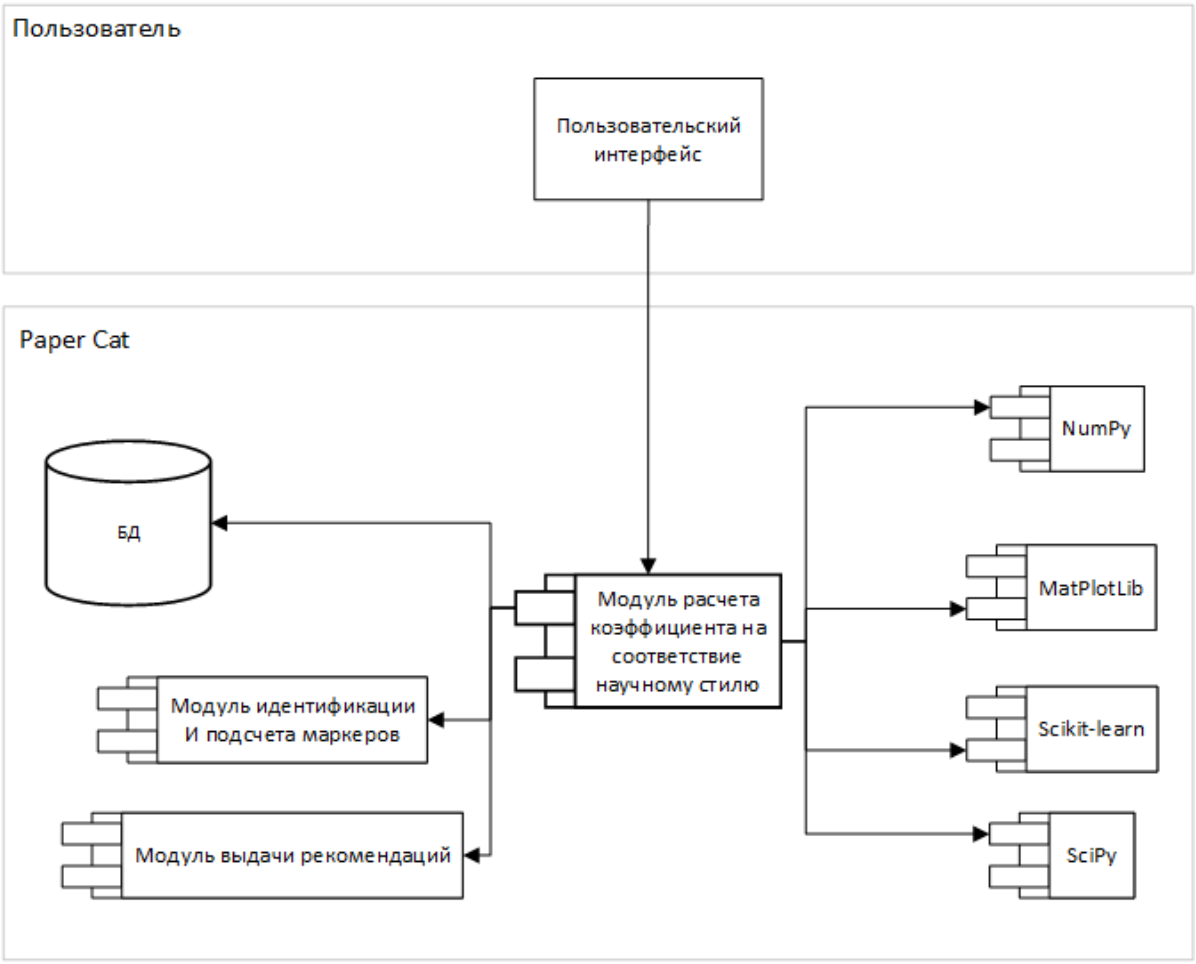

Рисунок 2.4. Проектирование архитектуры системы анализа текстов на соответствие научному стилю

Так как в данной системы решено использовать алгоритм нейронных сетей, необходимо спроектировать его структуру, которая представлена на рисунке 2.5.

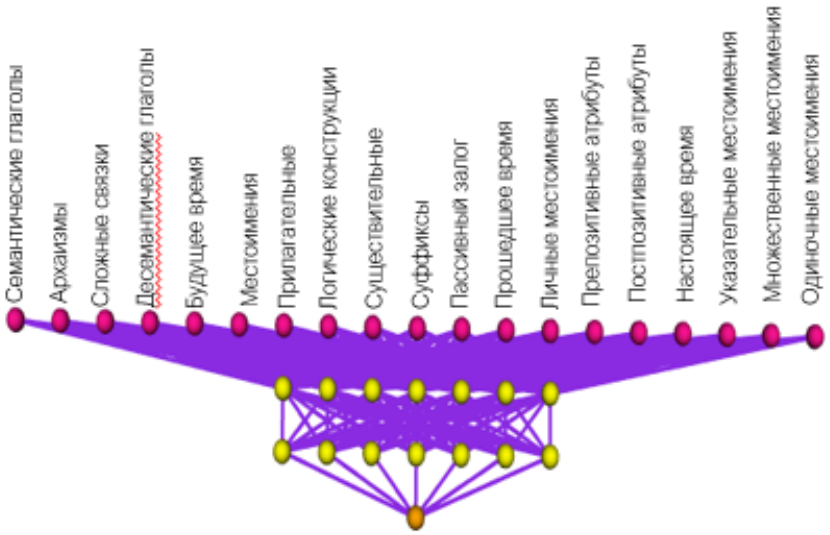

Рисунок 2.5. Проектирование структуры нейронной сети для системы анализа текстов на соответствие научному стилю

Из рисунка следует, что входных нейронов – 19, два скрытых слоя, на каждом из которых по 7 нейронов и один выходной нейрон.

2.3. Оценка рисков проекта по разработке системы анализа публикаций на соответствие научному стилю

Под риском в данном случае подразумевается событие, которое происходит в условиях неопределенности и наступление которого может иметь положительный или отрицательный эффект на целях проекта: длительность, стоимость и качество [5].

Риски проекта делятся на несколько категорий. Подробная структура категорий рисков представлена ниже.

1. Требования. Под этим пунктом подразумеваются все риски, связанные с проблемами неправильной формулировки технических требований к проекту. Например, возможна ситуация, когда требования заказчика могут быть непонятны для разработчиков или также они могут иметь двоякий смысл. Также нечеткая первоначальная формулировка пользовательских требований может привести к значительным изменениям системных требований, проявившихся на поздних стадиях разработки проекта.
2. Технология. Под этим пунктом подразумеваются риски, которые связаны с проблемами в применении всевозможных технологий. Например, для

реализации некоторого проекта необходимо использовать новую платформу, которая не применялась разработчиками ранее.

3. Сложность и взаимодействие. Здесь подразумеваются риски, которые относятся к проблемам с возможными ошибками в прогнозировании сложности предполагаемой работы.
4. Заказчик. Возможны ситуации, когда заказчик длительное время не выходит на связь, что, в свою очередь, тоже может являться причиной для задержки выпуска проекта. Также может быть ситуация, когда заказчик меняет требования к проекту, которые были даны изначально. В действительности подобная ситуация случается практически всегда, поэтому данный риск автоматически должен закладываться в смету проекта. Заказчик также может не предоставить какую-то важную для проектного отдела и разработчиков информацию, потому что для него она кажется не особенно важной.
5. Зависимости проекта. Сюда входят все случаи, когда проект зависит от чего-либо другого. Самая распространенная ситуация в данном случае происходит, если в компании на текущий момент несколько проектов, которыми уже занята команда разработчиков. В таком случае необходимо ждать, когда они закончат текущие проекты или выделить несколько человек, которые будут заниматься новым проектом. И в том, и в другом случае пострадает длительность проекта.
6. Ресурсы. Сюда входят все проблемы, связанные с ресурсами. Например, риски с человеческими ресурсами: возможны при большой текучке кадрового состава компании или, например, если один специалист компании, работающий над данным проектом, уходит в отпуск/уезжает в командировку/уходит на больничный и передает руководство над проектом другому специалисту. Также сюда же относятся всевозможные сценарии, связанные с отказом оборудования, так как оборудование также является ресурсом.
7. Расстановка приоритетов. Сюда включаются всевозможные проблемы, которые связаны с неправильной расстановкой приоритетов проекта со стороны специалиста проектного отдела.

8. Оценка. Сюда входят все риски, связанные с неправильной оценкой текущего проекта. Данная ситуация возможна, так как специалист рассчитывает смету, основываясь на своем опыте работы с подобного рода проектами. Здесь ошибки могут возникнуть на этапе прогнозирования стоимости реализации той или иной задачи, а также ее длительности.
9. Планирование. Также при ведении проекта возможны ошибки в планировании выполнения конкретных задач. Неправильно распределенное время может повести за собой увеличение длительности проекта.
10. Контроль. Сюда относятся всевозможные недочеты, которые могут быть пропущены при выполнении проекта.

Для дальнейшей оценки рисков необходимо идентифицировать и проанализировать возможные риски данного проекта. Исходя из структуры, перечисленной выше и основываясь на опыте предыдущих проектов, выявлены самые распространенные риски, встречающиеся в IT-проектах подобного рода. Однако сюда не входят такие глобальные риски, как: кризис, падение курса валют, чрезвычайные ситуации, политическая нестабильность, инфляция, нищета, терроризм и т.д., так как они являются неконтролируемыми. Также в данный список не входят риски, связанные с финансированием и расчетом бюджета, так как данный проект является выпускной квалификационной работой и не требует никаких денежных ресурсов. Ниже представлен подробный список.

1. Неоднозначная или непонятная формулировка требований.
2. Отсутствие связи с научным руководителем.
3. Изменение требований на поздних стадиях разработки.
4. Загруженность разработчика из-за других предметов и проектов.
5. Нехватка человеческих ресурсов (главный и единственный разработчик ушел на больничный).
6. Проблема с оборудованием.
7. Проблемы с изучением новой технологии разработки.
8. Ошибки в расстановке приоритетов проекта.
9. Неправильная оценка длительности этапов разработки.
10. Неправильное планирование проекта.
11. Неправильный контроль проекта.

После того, как выявлены риски данного проекта, необходимо приступать к качественному анализу рисков. На представленной ниже таблице 2.2 отображены риски, их вероятность, воздействие и ранг, где вероятность оценивается как: очень вероятно (9-10), возможно (6-8), маловероятно (2-5), невероятно (0-1), а воздействие оценивается как: катастрофическое (7-10), критичное (4-6) и некритичное (0-3). Ранг рассчитывается путем умножения значения оценки вероятности на значение оценки воздействия.

Таблица 2.2. Качественный анализ рисков проекта

Риск	Вероятность	Воздействие	Ранг
Неоднозначная формулировка требований	3	5	15
Отсутствие связи с научным руководителем	2	6	12
Изменение требований на поздних стадиях разработки	3	4	12
Загруженность разработчика	2	5	10
Нехватка человеческих ресурсов	6	7	42
Проблема с оборудованием	5	7	35
Неправильная расстановка приоритетов	4	5	20
Неправильная оценка длительности	3	5	15
Неправильное планирование	1	7	7
Неправильный контроль	3	6	18

После проведения качественного анализа рисков можно сделать вывод, что самыми опасными рисками данного проекта являются: нехватка человеческих ресурсов, проблема с оборудованием и неправильная расстановка приоритетов. В таблице 2.3 рассмотрены причины и условия данных рисков, а также представлены последствия и ущерб, нанесенный выявленными рисками.

Таблица 2.3. Анализ выявленных рисков

Риск	Причина	Условия	Последствия	Ущерб
Неоднозначная формулировка требований	Требования не ясны для разработчика	Отсутствует однозначная и понятная формулировка требований	Связь с руководителем, уточнение требований, формулирование требований заново	Увеличение длительности проекта, увеличение трудозатрат
Отсутствие связи с научным руководителем	Научный руководитель не реагирует на попытки связи	Появление вопросов, связанных с разработкой данной работы	Попытки связи через другие источники связи и через академического руководителя направления	Увеличение длительности проекта
Изменение требований на поздних стадиях разработки	Руководитель меняет требования, которые уже сформулированы в техническом задании	Другое представление проекта у руководителя, изменение решения по ходу разработки проекта	Выполнение требуемых изменений	Увеличение длительности и трудозатрат проекта
Загруженность разработчика	Разработчик занят другими	Несогласованное количество проектов	Выполнение проекта не в срок	Увеличение длительности

Риск	Причина	Условия	Последствия	Ущерб
	предметами и проектами	в учебном плане студента		проекта
Нехватка человеческих ресурсов	Разработчик взял больничный	Болезнь разработчика	Выполнение проекта не в срок	Увеличение длительности проекта
Проблема с оборудованием	Оборудование разработчика вышло из строя	Могут быть различными	Выполнение проекта не в срок	Увеличение длительности проекта
Неправильная расстановка приоритетов	Неправильная расстановка приоритетных задач при выполнении проекта	Неправильная формулировка требований проекта, отсутствие сценариев использования разрабатываемой системы	Связь с научным руководителем, проработка сценариев использования и бизнес-процессов приложения	Увеличение длительности и трудозатрат проекта
Неправильная оценка длительности проекта	Ошибки в оценке длительности этапов разработки проекта	Ошибки при установке уровня сложности этапа разработки	Выполнение проекта не в срок	Увеличение длительности и трудозатрат проекта
Неправильное планирование ресурсов проекта	Ошибки в оценке длительности этапов разработки проекта	Ошибки при установке уровня сложности этапа разработки	Выполнение проекта не в срок	Увеличение длительности и трудозатрат проекта
Неправильный контроль проекта	Пропущены недочеты или ошибки при контроле выполнения проекта и тестировании конечного продукта	Малое количество времени, которое отводится на проверку и тестирование приложения	Доработка недочетов	Увеличение длительности и трудозатрат проекта

2.4. Технико-экономическое обоснование системы анализа публикаций на соответствие научному стилю

2.3.1. Планирование комплекса работ по разработке системы

Для определения ожидаемой продолжительности работы $T_{ож}$ применяется формула 2.1.

$$T_{ож} = \frac{t_{мин} + 4t_{нв} + t_{макс}}{6}, \quad (2.1)$$

где $t_{мин}$ – кратчайшая продолжительность данной работы (оптимистическая оценка);

$t_{макс}$ – самая большая продолжительность работы (пессимистическая оценка);

$t_{нв}$ – наиболее вероятная продолжительность работы (реалистическая оценка).

Оценка трудоемкости отдельных видов работ приведена в таблице 2.4.

Таблица 2.4. Вычисление продолжительности работ на каждом этапе разработки

Виды работы	**$t_{мин}$**	**$t_{нв}$**	**$t_{макс}$**	**Ожидаемая продолжительность работы**
Исследование и обоснование разработки	15	17	18	17
Поиск аналогов и прототипов	7	8	9	8
Анализ требований	12	14	16	14
Проектирование архитектуры системы	2	3	5	3
Проектирование архитектуры компонентов системы	5	7	8	7
Программирование модулей системы	8	14	16	13
Тестирование программных модулей	19	21	23	21
Сборка и испытание программы	2	5	7	5
Анализ результатов испытаний	3	4	8	5
Оформление рабочей документации	16	23	26	23

Руководитель (в данном случае научный руководитель) выполняет функции консультанта, а именно: ставит цель и задачи работы, курирует ход работ и дает необходимые консультации по ходу выполнения разработки системы. Исполнитель (в данном случае программный инженер) отвечает за выполнение всех работ данного проекта: от выполнения постановки задачи до выполнения внедрения системы. В таблице 2.5. представлен расчет длительности проекта.

Таблица 2.5. Расчет длительности проекта

Содержание работ	**Исполнители**	**Длительность, дни**	**Загрузка**	
			дни	**%**
1. Подготовка процесса разработки и анализ требований				
1.1 Исследование и обоснование разработки				
1.1.1 Постановка задачи	Руководитель	3	1	33
	Программист			100
1.1.2 Сбор исходных данных	Руководитель	14	5	35
	Программист		14	100
1.2 Поиск аналогов и прототипов				
1.2.1 Анализ существующих методов решения задачи и программных средств	Руководитель	6	0	0
	Программист		6	100
1.2.2 Обоснование принципиальной необходимости разработки	Руководитель	2	1	50
	Программист		2	100
1.3 Анализ требований				
1.3.1 Определение и анализ требований к проектируемой программе	Руководитель	3	1	33
	Программист		3	100
1.3.2 Определение структуры входных и выходных данных	Руководитель	5	1	20
	Программист		5	100
1.3.3 Выбор технических и программных средств реализации	Руководитель	3	1	33
	Программист		3	100
1.3.4 Согласование и утверждение	Руководитель	3	1	33

Содержание работ	Исполнители	Длительность, дни	Загрузка	
технического задания	Программист		3	100
Итого по этапу 1	Руководитель	**39**	**11**	
	Программист		**39**	
2. Проектирование				
2.1 Проектирование программной архитектуры	Руководитель	3	0	0
	Программист		3	100
2.2 Техническое проектирование компонентов программы	Руководитель	7	0	0
	Программист		7	100
Итого по этапу 2	Руководитель	**10**	**0**	
	Программист		**10**	
3. Программирование и тестирование программных модулей				
3.1 Программирование модулей в выбранной среде программирования	Руководитель	13	0	0
	Программист		13	100
3.2 Тестирование программных модулей	Программист	21	0	0
	Программист		21	100
3.3 Сборка и испытание программы	Руководитель	5	2	40
	Программист		5	100
3.4 Анализ результатов испытаний	Руководитель	5	1	20
	Программист		5	100
Итого по этапу 3	Руководитель	**44**	**3**	
	Программист		**44**	
4. Оформление рабочей документации				
4.1 Оформление рабочей документации	Руководитель	22	5	
	Программист		22	100
Итого по этапу 4	Руководитель	**22**	**5**	
	Программист		**22**	
Итого по проекту	Руководитель	**115**	**19**	
	Программист		**115**	

2.3.2. Расчет затрат на разработку проекта

Расчет основной заработной платы разработчиков проекта приведен в таблице 2.6. Сведения о должностном окладе взяты из сметы компании прохождения практики прошлого года.

Таблица 2.6. Расчет основной заработной платы работников

Должность	Средняя дневная ставка, руб.	Затраты времени на разработку, чел.-дней	ОЗП, руб.
Руководитель	2500,00	19	47500,00
Программист	1950,00	115	224250,00
Итого			271750,00

Из-за того, что проектируемая система должна быть запрограммирована и отлажена с помощью компьютеров, к суммарным затратам на разработку добавляются затраты на использование машинного времени, исчисляемые как представлено в формуле 2.2.

$$M_{в} = t_{мв} S_{мч} K_{м},$$

(2.2)

где $t_{мв}$ – машинное время компьютера, необходимое для разработки программного продукта; $t_{мв}$ = 224 час.;

$S_{мч}$ – стоимость 1 часа машинного времени; $S_{мч}$ = 12 руб./час.;

$K_{м}$ – коэффициент мультипрограммности (показывает долю машинного времени, отводимого непосредственно на работу над проектом); $K_{м}$ = 1.

В таблице 2.7. представлен расчет затрат на разработку системы.

Таблица 2.7. Расчет затрат на разработку системы

Статьи затрат	Сумма, руб.
Основная заработная плата	271750,00
Затраты на машинное время	2688,00
Итого	274438,00

При внедрении системы, рассматриваемой в данном проекте, затраты на его реализацию определяются затратами на оборудование. В оборудование и материалы входит компьютер на базе процессора Intel Core-i5. Стоимость компьютера 50000 руб.

Затраты на основное и вспомогательное оборудование представлены в формуле 2.3.

$$K_{о} = \sum_{j=1}^{n} C_{bj} Q_{j} Y_{j},$$

(2.3)

где C_{bj} – балансовая стоимость j-го вида оборудования, руб. (при n=1 C_{b1} = 50000 руб.);

Q_{j} – количество единиц j-го оборудования, руб.;

Y_{j} – коэффициент загрузки j-го вида оборудования при обработке информации по решению задач предметной области рассчитывается по формуле 2.4.

$$Y_{j} = \frac{T_{j}}{\Phi_{эфj}},$$

(2.4)

где $\Phi_{эфj}$ – эффективный годовой фонд времени работы технического средства j-го вида, час./год.

Время работы технического средства j-го вида по решению s задач, час./год рассчитывается по формуле 2.5.

$$T_{j} = \sum_{k=1}^{s} t_{kj} \times U_{k},$$

(2.5)

где t_{kj} – трудоемкость однократной обработки информации по $к$-й задаче на j-м виде технических средств, часов машинного времени (t_{kj} = 6);

$U_к$ – частота (периодичность) решения $к$-й задачи, дней /год ($U_к$ =264).

Затраты на реализацию:

$К_р$ = 50000×1×6×264/(264×8) руб. = 37500 руб.

Таким образом, суммарные затраты на разработку проекта:

$К = К_п + К_р$ = 271750 + 37500 руб. = 309250 руб.

Суммарные затраты, связанные с внедрением аналога, складываются из следующих затрат:

1) затраты по оплате услуг на установку и сопровождение продукта (12000 руб.);

2) затраты на основное и вспомогательное оборудование (50000 руб.) (предполагается, что для внедрения аналога понадобится такой же компьютер, что и для проектируемой системы);

3) затраты на подготовку пользователя (оплата курсов повышения квалификации, командировочные расходы и пр.) (9000 руб.).

Итого суммарные затраты, связанные с внедрением аналога, составят 71000 руб.

Глава 3. Реализация системы анализа публикаций на английском языке на соответствие научному стилю

Основной задачей системы анализа публикаций является расчет коэффициента соответствия статьи, загруженной пользователем с компетентным корпусом статей. Для решения данной задачи отобраны методы машинного обучения, описанные в пункте 1.3. Для реализации системы анализа публикаций на английском языке на соответствие научному стилю использованы библиотеки, представленные в пункте 2.2.

Листинг программного кода функции по вычислению коэффициента соответствия:

```
def home():
        # Creation of data set
        x, y = make_blobs(center_box=[0.0, 1.0], cluster_std = 0.5,
n_samples = 150, centers=2, n_features=19, random_state=0)

        # Division on test set and train set
        x_train, x_test, y_train, y_test = train_test_split(x, y,
random_state=0)
        # Learning system
        logreg = LogisticRegression().fit(x_train, y_train)
        nb = GaussianNB().fit(x_train, y_train)
        svm = SVC().fit(x_train, y_train)
        nn = MLPClassifier(hidden_layer_sizes=(6, 2)).fit(x_train,
y_train)
        tree = DecisionTreeClassifier().fit(x_train, y_train)
        # Send message back to client
        # Comparation test data with train data
        message = "{:.3f}".format(logreg.score(x_test, y_test))
        message1 = "{:.3f}".format(nb.score(x_test, y_test))
        message2 = "{:.3f}".format(svm.score(x_test, y_test))
        message3 = "{:.3f}".format(nn.score(x_test, y_test))
        message4 = "{:.3f}".format(tree.score(x_test, y_test))
        return dict(
        m = message,
        m1 = message3,
        m2 = message1,
        m3 = message4,
        m4 = message2)
```

Возвращаемые значения данной функции передаются на вход index.tpl странице, в которой строится таблица для наглядного сравнения полученных значений.

Пример полученной таблицы представлен на рисунке 3.1.

Name of method	Compliance
Logistic Regression	0.711
Neural Networks	0.474
Naive Bayes	0.816
Decision Tree	0.632
SVM	0.789

Рисунок 3.1. Таблица соответствия анализа текстов на соответствие научному стилю

В данной таблице представлено 5 отобранных методов и значения соответствия, вычисленные с помощью них. Единица означает, что статья соответствует необходимым требованиям выбранного корпуса. Ноль означает полное несоответствие с академическим стилем в выбранном корпусе. На рисунке 3.1. представлен результат, который показывает, что загруженная статья имеет хороший уровень академического английского языка.

Также для более полного понимания пользователем, каким именно характеристикам лучше уделить большее внимание, система строит диаграмму, показывающую важность влияния каждого маркера на коэффициент соответствия в частности. Программный код построения данной диаграммы представлен ниже:

```
 markers = ('Abstract semantic verbs', 'Archaisms', 'Complex
conjunctions', 'Desemantisized verbs', 'Future', 'IPronoun',
'Intensifying adverbs', 'Logic connectors', 'Noun', 'OrSuffix',
'PassiveVoice', 'Past', 'PersonalPronoun', 'Prepositive
attributes', 'Postpositive attributes', 'Present', 'ThatThose',
'WePronoun', 'YouHeShe')
 plt.barh(range(len(markers)), tree.feature_importances_)
 plt.yticks(np.arange(len(markers)), markers)
 plt.xlabel("The importance of characteristic")
 plt.ylabel("The characteristic")
 plt.tight_layout()
 pic = plt.savefig("static\content\pic.png")
```

Данный код строит диаграмму и сохраняет ее в папке, где хранится страница сервиса. Пример диаграммы представлен на рисунке 3.2.

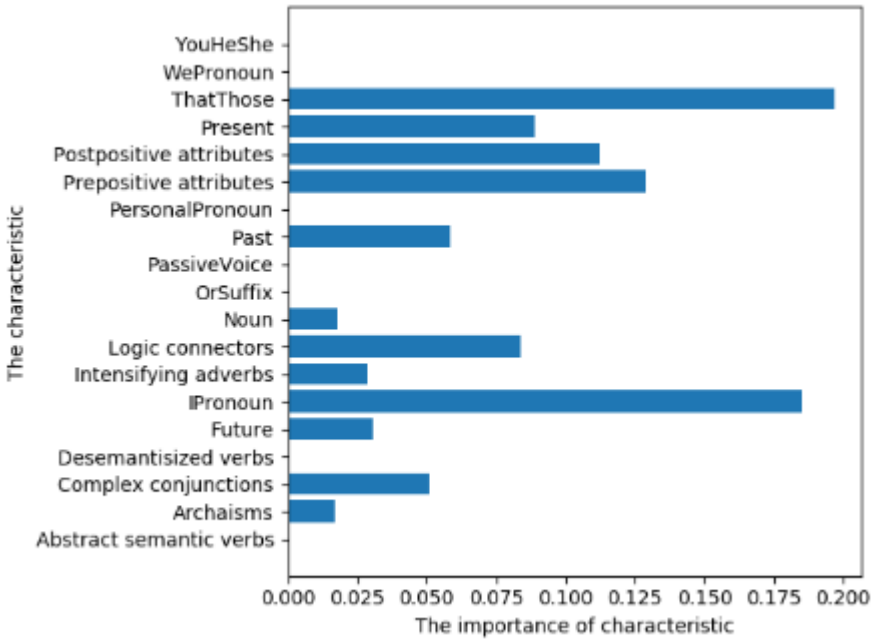

Рисунок 3.2. Диаграмма важности влияния маркеров на анализ текстов на соответствие научному стилю

Из диаграммы следует, что основное влияние на научный стиль статьи имеет маркер «ThatThose» и «IPronoun».

Также в системе предусмотрены функции загрузки файла статьи в систему и выгрузки результатов сравнения в Excel-файл.

Программный код функции экспорта результатов соответствия в Excel-файл после нажатия на кнопку «Download results» представлен ниже:

```
book = xlwt.Workbook()
sheet1 = book.add_sheet('sheet1')
data = [m, m1, m2, m3, m4]
for i,e in enumerate(supersecretdata):
    sheet1.write(i,1,e)
name = "random.xls"
book.save(name)
book.save(TemporaryFile())
```

Полный листинг программного кода представлен в Приложении Б.

На рисунках 3.3, 3.4 представлен пример работы системы.

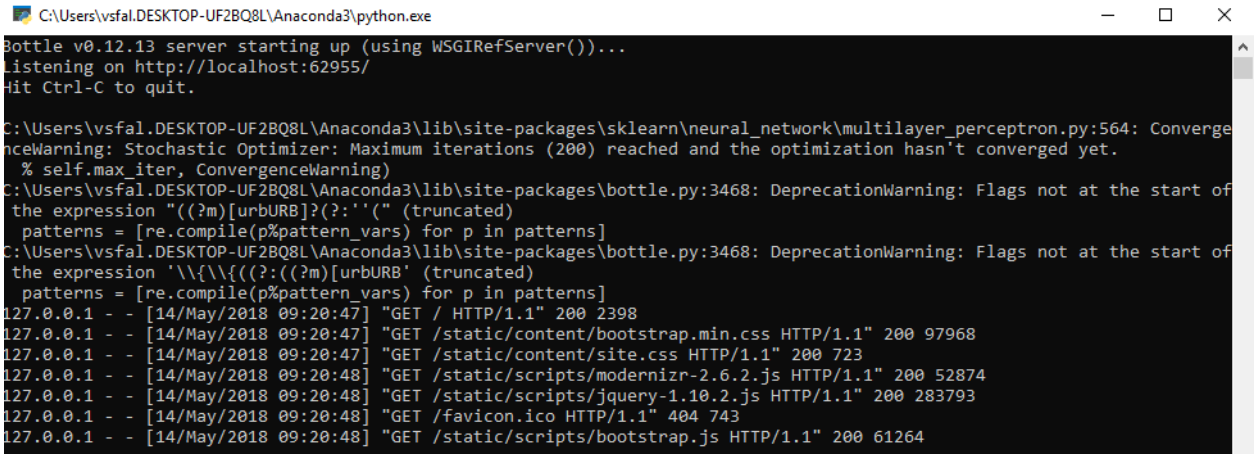

Рисунок 3.3. Запуск сервера системы анализа текстов на соответствие научному стилю

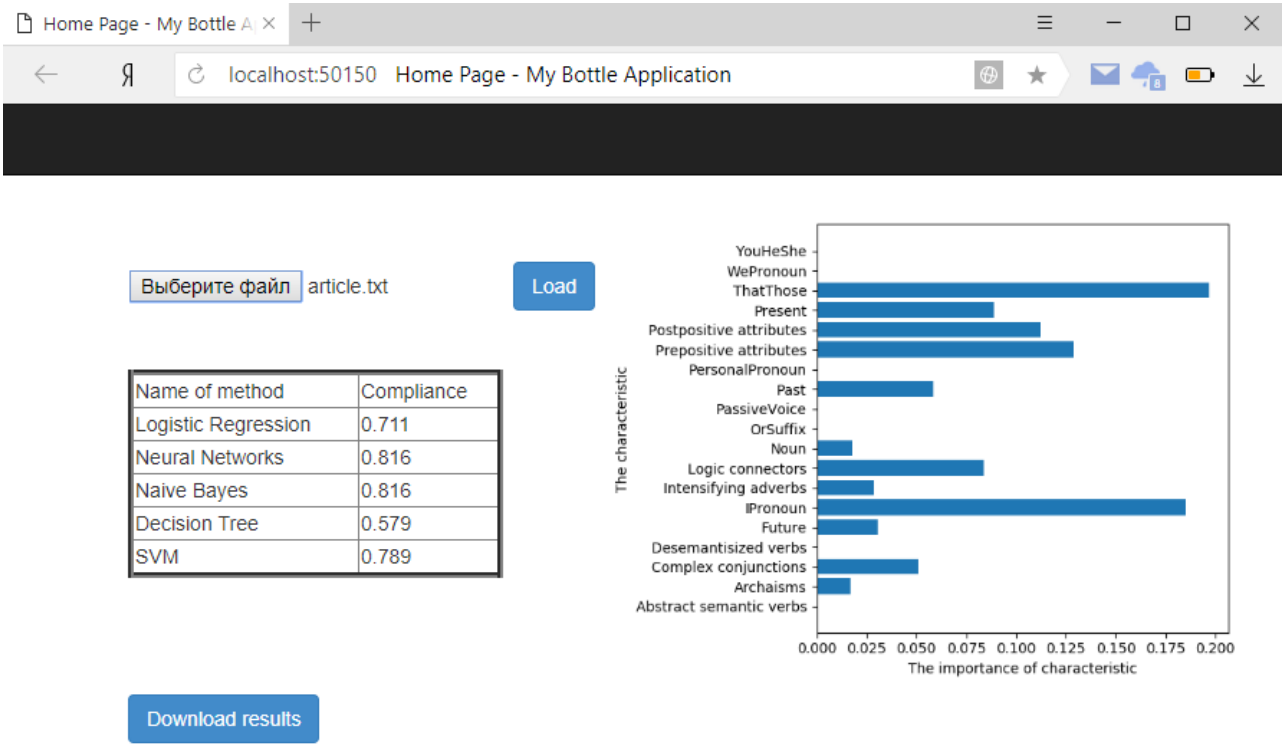

Рисунок 3.4. Пользовательский интерфейс системы анализа текстов на соответствие научному стилю

Для более удобного процесса тестирования алгоритмов используется система Anaconda в целом и Jupiterlab в частности.

Тестирование производится программным способом, программный код представлен ниже:

```
from bottle import route, view
from datetime import datetime
import numpy as np
import matplotlib.pyplot as plt
from http.server import BaseHTTPRequestHandler, HTTPServer
```

```
from sklearn.datasets import make_blobs
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_moons

# Creation of data set
x, y = make_blobs(center_box=[0.0, 1.0], cluster_std = 0.5, n_samples
= 150, centers=2, n_features=19, random_state=0)

# Division on test set and train set
x_train, x_test, y_train, y_test = train_test_split(x, y,
random_state=0)
# Learning system
logreg = LogisticRegression().fit(x_train, y_train)
nb = GaussianNB().fit(x_train, y_train)
svm = SVC().fit(x_train, y_train)
nn = MLPClassifier(hidden_layer_sizes=(6, 2)).fit(x_train, y_train)
tree = DecisionTreeClassifier().fit(x_train, y_train)
# Send message back to client
# Comparation test data with train data
print("Logistic Regression: {:.3f}".format(logreg.score(x_test,
y_test)))
print("Naive Bayes: {:.3f}".format(nb.score(x_test, y_test)))
print("SVM: {:.3f}".format(svm.score(x_test, y_test)))
print("Neural Networks: {:.3f}".format(nn.score(x_test, y_test)))
print("Decision Tree: {:.3f}".format(tree.score(x_test, y_test)))
```

Для тестирования использованы данные нескольких статей. Тесты производились от наиболее приближенных к компетентным значений маркеров к наименее приближенным. Ниже представлен результат тестирования анализа статьи с наиболее приближенными значениями маркеров:

Логистическая регрессия: 0.974

Наивный Байес: 1.000

Метод опорных векторов: 0.974

Нейронные сети: 0.868

Метод деревьев решений: 0.737

Ниже представлен результат тестирования анализа статьи с наименее приближенными значениями маркеров:

Логистическая регрессия: 0.711

Наивный Байес: 0.816

Метод опорных векторов: 0.789

Нейронные сети: 0.632

Метод деревьев решений: 0.711

Ниже представлен результат тестирования анализа статьи с минимально приближенными значениями маркеров:

Логистическая регрессия: 0.579

Наивный Байес: 0.632

Метод опорных векторов: 0.421

Нейронные сети: 0.553

Метод деревьев решений: 0.500

Из данных тестовых случаев следует, что с каждой загрузкой статей среднее значение вычисленных коэффициентов снижается. Первый случай: 0,911; второй случай: 0,732; третий случай: 0,537. Данный факт означает, что алгоритм работает должным образом.

Заключение

В результате процесса анализа методов машинного обучения выбраны методы разработки приложения для оценки публикаций на английском языке на соответствие академическому стилю. После выбора методов разработки спроектировано и разработано приложение для оценки публикаций на английском языке на соответствие академическому стилю. Данное приложение обучается на опубликованных статьях различных издательств и проверено на статьях, которые были написаны студентами НИУ ВШЭ – Пермь. Приложение проверяет загруженную статью для оценки соответствия академическому стилю и представляет результат для пользователя. Разработанное приложение является частью большого веб-сервиса, разработанного преподавателями и студентами НИУ ВШЭ – Пермь. Данное приложение предназначено для студентов и преподавателей соответствующего университета в целях улучшения уровня академического английского в своих статьях для английских издательств.

В результате данной работы поставлена цель, определены задачи по достижению этой цели. В ходе данной работы рассмотрены и описаны основные проблемы и методы их решения. После обзора основных проблем и методов их решения выбраны подходы к разработке приложения для оценки соответствия публикаций на английском языке академическому стилю.

В качестве будущей работы над этим проектом можно рассмотреть и развить взаимодействие с экспертной системой для улучшения разработанного сервиса, который может прогнозировать точный и определенный результат и давать некоторые рекомендации по исправлению загруженного документа в систему.

Библиографический список

1. Dontcheva-Navratilova O. (2013) Authorial presence in academic discourse: Functions of author-reference pronouns. Linguistica Pragensia, 23(1), 9-30.
2. «НИУ ВШЭ в Перми» – Научно-учебная группа «Разработка программного обеспечения для проведения корпусных исследований английского языка» [Электронный ресурс] URL: https://perm.hse.ru/bi/sfcr (дата обращения 08.11.2017).
3. Мухамедиев Р.И., Мухамедиева Е.Л., Кучин Я.И. (2015) Таксономия методов машинного обучения и оценка качества классификации и обучаемости. Cloud of Science, 2(3), 359-378.
4. Выбор алгоритмов машинного обучения Microsoft Azure // Техническая документация, материалы по API и примеры кода [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/azure/machine-learning/studio/algorithm-choice (дата обращения 08.11.2017).
5. Project Management Institute. (2016) PMBOK Guide – Sixth Edition.
6. Berry M. W. (2004) Survey of Text Mining I: Clustering, Classification, and Retrieval. Springer, 1, 3-23.
7. Aggarwal C. C., Zhai C.C. (2012) Mining Text Data. Springer, 1, 77-128.
8. Do Prado H. A, Ferneda E. (2007) Emerging Technologies of Text Mining: Techniques and Applications. Idea Group Reference, 1.
9. Гореликова С. Н. Природа термина и некоторые особенности терминообразования в английском языке. Оренбург: Вестник ОГУ, 2002.
10. Biber, D., Gray B. (2010) Challenging stereotypes about academic writing: Complexity, elaboration, explicitness. Journal of English for Academic Purposes, 9, 2-20.
11. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными. М., 2017.
12. Ясницкий Л.Н., Интеллектуальные системы: учебник. М.: Лаборатория знаний, 2016. – 221 с.
13. ГОСТ 19.201-78 Единая система программной документации. Техническое задание. Требования к содержанию и оформлению.

14. Machine Learning in Python // scikit-learn 0.19.1 documentation [Электронный ресурс] URL: http://scikit-learn.org/stable (дата обращения 15.04.2018).
15. A simple way to share Jupyter Notebooks // nbviewer [Электронный ресурс] URL: http://nbviewer.jupyter.org (дата обращения 15.04.2018).
16. «АОТ» – Автоматическая Обработка Текста [Электронный ресурс] URL: http://www.aot.ru/ (дата обращения 08.11.2017).
17. «Machine Learning» – Машинное обучение [Электронный ресурс] URL: www.machinelearning.ru (дата обращения 08.11.2017).
18. «ММРО» – Математические методы распознавания образов [Электронный ресурс] URL: http://mmro.ru (дата обращения 08.11.2017).
19. Арнольд И. В. Стилистика: Современный английский язык. М.: Флинта: Наука, 2002.
20. Рябцева Н. К. Научная речь на английском языке. Руководство по научному изложению. Словарь оборотов и сочетаемости общенаучной лексики. Новый словарь-справочник активного типа (на английском языке). М.: Наука, 2002.
21. Зайцев А.Б. Некоторые особенности прагматической адаптации перевода англоязычного научного текста на русский язык. Оренбург: Вестник ОГУ, 2001.
22. Biber, D. (2006) Stance in spoken and written university registers. Journal of English for Academic Purposes, 5(2), 97-116.
23. Biber, D., Gray B. (2010) Challenging stereotypes about academic writing: Complexity, elaboration, explicitness. Journal of English for Academic Purposes, 9, 2-20.
24. Поспелова Г.Б. Характеристики научного стиля в английском языке — Иностранные языки: теория и практика. М.: ЛИНГВИСТИКА, 2012.
25. «Grammarly» – Free Writing Assistant [Электронный ресурс] URL: https://app.grammarly.com (дата обращения 08.11.2017).
26. «STATISTICA» – Data Mining, анализ данных, контроль качества, прогнозирование, обучение, консалтинг [Электронный ресурс] URL: http://statsoft.ru (дата обращения 08.11.2017).

Приложение А. Техническое задание (ГОСТ 19.201-78)

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

УТВЕРЖДАЮ

Старший преподаватель кафедры
информационных технологий в бизнесе
НИУ ВШЭ – Пермь

_____ В.В. Ланин

«_____»_____ 2018 г.

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АНАЛИЗА ПУБЛИКАЦИЙ НА АНГЛИЙСКОМ ЯЗЫКЕ НА СООТВЕТСТВИЕ НАУЧНОМУ СТИЛЮ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

Техническое задание

Листов 16

ЛИСТ УТВЕРЖДЕНИЯ

РАЗРАБОТЧИК

_____ В.С.Фалалеева
«___»_____ 2018 г.

Пермь 2018

АННОТАЦИЯ

В данном программном документе приведено техническое задание на разработку модуля системы Paper Cat.

В данном программном документе, в разделе «Введение» указано наименование, краткая характеристика области применения программы (программного изделия).

В разделе «Основания для разработки» указаны документы, на основании которых ведется разработка, наименование и условное обозначение темы разработки.

В данном программном документе, в разделе «Назначение разработки» указано функциональное и эксплуатационное назначение программы (программного изделия).

Раздел «Требования к программе» содержит следующие подразделы:

1) требования к функциональным характеристикам;
2) требования к надежности;
3) условия эксплуатации;
4) требования к составу и параметрам технических средств;
5) требования к информационной и программной совместимости;
6) требования к маркировке и упаковке;
7) требования к транспортированию и хранению;
8) специальные требования.

В данном программном документе, в разделе «Требования к программной документации» указаны предварительный состав программной документации и специальные требования к ней.

В данном программном документе, в разделе «Стадии и этапы разработки» установлены необходимые стадии разработки, этапы и содержание работ.

В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.

Данное техническое задание выполнено согласно ГОСТ 19.201-78.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Наименование программы

Наименование – «Разработка приложения для анализа публикаций на английском языке на соответствие научному стилю с использованием методов машинного обучения».

1.2. Краткая характеристика области применений

Программа предназначена к применению в Пермском филиале Национального Исследовательского Университета «Высшая Школа Экономики» (далее – НИУ ВШЭ – Пермь) для определения уровня академического стиля написанной работы.

2. ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ

2.1. Основание для проведения разработки

Основанием для проведения разработки является «Положение о курсовой и выпускной квалификационной работе студентов, обучающихся по программам бакалавриата, специалитета и магистратуры НИУ ВШЭ» от 10.07.2015 № 6.18.1-01/1007-02. Тема ВКР согласована с научным руководителем – Ланиным Вячеславом Владимировичем.

2.2. Наименование и условное обозначение темы разработки

Наименование темы разработки – «Разработка приложения для анализа публикаций на английском языке на соответствие научному стилю с использованием методов машинного обучения».

Условное обозначение – «Paper Cat».

3. НАЗНАЧЕНИЕ РАЗРАБОТКИ

3.1. Функциональное назначение

Функциональным назначением программы является предоставление пользователю возможности анализа уровня академического английского в загруженном в систему текстовом файле.

3.2. Эксплуатационное назначение

Программа должна эксплуатироваться в профильных подразделениях на объектах Заказчика.

Конечными пользователями программы должны являться сотрудники и студенты НИУ ВШЭ-Пермь.

4. ТРЕБОВАНИЯ К ПРОГРАММЕ ИЛИ ПРОГРАММНОМУ ИЗДЕЛИЮ

4.1. Требования к функциональным характеристикам

4.1.1. Требования к составу выполняемых функций

Программа должна обеспечивать выполнение нижеперечисленных функций:

9) функция загрузки текстового файла в систему

10) функция проверки текстового файла на соответствие научному стилю

11) функция вывода результата проверки на соответствие научному стилю на экран

4.1.2. Требования к организации входных данных

Входные данные программы должны быть организованы в виде отдельных файлов формата txt, соответствующих спецификации.

Файлы указанного формата должны размещаться (храниться) на локальных или съемных носителях, отформатированных согласно требованиям операционной системы.

Любой файл иного формата, но с расширением txt, открываться не должен.

4.1.3. Требования к организации выходных данных

Выходные данные программы должны быть представлены на экранной форме.

4.1.4. Требования к временным характеристикам

Требования к временным характеристикам зависят от выполняемой задачи. При необходимости обновления данных системы время ожидания зависит от скорости интернета на устройстве, с которого был произведен вход в систему. Скорость выполняемой задаче не должна превышать 20 секунд. Без необходимости обновления данных в системе время ожидания результатов выполнения запросов пользователя не должно превышать 1 секунды. При формировании отчета временные рамки увеличиваются пропорционально обрабатываемым данным.

4.2. Требования к надежности

4.2.1. Требования к обеспечению надежного (устойчивого) функционирования программы

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением Заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

1) организацией бесперебойного питания технических средств;

2) использованием лицензионного программного обеспечения;

3) регулярным выполнением рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;

4) регулярным выполнением требований ГОСТ 51188-98. Защита информации. Испытания программных средств на наличие компьютерных вирусов.

4.2.2. Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать времени, необходимого на перезагрузку операционной системы и запуск программы, при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

4.2.3. Отказы из-за некорректных действий оператора

Отказы программы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

4.3. Условия эксплуатации

4.3.1. Климатические условия эксплуатации

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

4.3.2. Требования к видам обслуживания

Программа не требует проведения каких-либо видов обслуживания.

4.3.3. Требования к численности и квалификации персонала

Конечный пользователь программы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы.

4.4. Требования к составу и параметрам технических средств

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий в себя:

1) процессор – 2.3 ГГц;

2) свободная оперативная память – 4 Гб;

3) свободная память жесткого диска – 20 Гб.

4) клавиатура;

5) мышь;

6) монитор.

4.5. Требования к информационной и программной совместимости

4.5.1. Требования к информационным структурам и методам решения

Требования к информационным структурам и методам решения не предъявляются.

4.5.2. Требования к исходным кодам и языкам программирования

Исходные коды программы должны быть реализованы на языке Python. В качестве интегрированной среды разработки программы должна быть использована среда Visual Studio и Anaconda.

4.5.3. Требования к программным средствам, используемым программой

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы.

4.5.4. Требования к защите информации и программ

Требования к защите информации и программ не предъявляются.

4.6. Требования к маркировке и упаковке

4.6.1. Требования к маркировке

Требования к маркировке не предоставляются.

4.6.2. Требования к упаковке

Пакет файлов необходимых к распространению – это инсталляционный пакет, инструкция по установке и руководство пользователя.

4.7. Требования к транспортированию и хранению

Ссылка на сервис будет представлена на сайте НИУ ВШЭ-Пермь.

4.8. Специальные требования

Программа должна обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса, разработанного согласно рекомендациям компании-производителя операционной системы.

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1. Предварительный состав программной документации

Состав программной документации должен включать в себя:

1) техническое задание;
2) программу и методики испытаний;
3) руководство системного программиста;
4) руководство пользователя;
5) ведомость эксплуатационных документов.

6. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Ориентировочная экономическая эффективность не рассчитываются.

Предполагаемое число использования программы в год – 365 сеансов работы на одном рабочем месте.

7. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

7.1. Стадии разработки

Разработка должна быть проведена в три стадии:

1) разработка технического задания;

2) рабочее проектирование;

3) внедрение.

7.2. Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1) разработка программы;

2) разработка программной документации;

3) испытания программы.

На стадии внедрения должен быть выполнен этап разработки - подготовка и передача программы.

7.3. Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1) постановка задачи;

2) определение и уточнение требований к техническим средствам;

3) определение требований к программе;

4) определение стадий, этапов и сроков разработки программы и документации на неё;

5) выбор языков программирования;

6) согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77 с требованием п. Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

1) разработка, согласование и утверждение программы (в ГОСТ, похоже, опечатка – «порядка») и методики испытаний;

2) проведение приемо-сдаточных испытаний;

3) корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию на объектах Заказчика.

8. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

8.1. Виды испытаний

Приемо-сдаточные испытания должны проводиться на объекте Заказчика в сроки с 29.04.18 по 20.05.18.

Приемо-сдаточные испытания программы должны проводиться согласно разработанной Исполнителем и согласованной Заказчиком Программы и методик испытаний.

Ход проведения приемо-сдаточных испытаний Заказчик и Исполнитель документируют в Протоколе проведения испытаний.

8.2. Общие требования к приемке работы

На основании Протокола проведения испытаний Исполнитель совместно с Заказчиком подписывают Акт приемки-сдачи программы в эксплуатацию.

9. ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

НИУ ВШЭ-Пермь – Национальный Исследовательский Университет «Высшая школа экономики» Пермский филиал.

Приложение Б. Листинг программного кода

Б.1. Листинг файла routes.py

```
"""
Routes and views for the bottle application.
"""

from bottle import route, view
from datetime import datetime
import numpy as np
import matplotlib.pyplot as plt
from http.server import BaseHTTPRequestHandler, HTTPServer
from sklearn.datasets import make_blobs
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_moons

@route('/')
@route('/home')
@view('index')
def home():
        # Creation of data set
        x, y = make_blobs(center_box=[0.0, 1.0], cluster_std = 0.5,
n_samples = 150, centers=2, n_features=19, random_state=0)

        # Division on test set and train set
        x_train, x_test, y_train, y_test = train_test_split(x, y,
random_state=0)
        # Learning system
        logreg = LogisticRegression().fit(x_train, y_train)
        nb = GaussianNB().fit(x_train, y_train)
        svm = SVC().fit(x_train, y_train)
        nn = MLPClassifier(hidden_layer_sizes=(6, 2)).fit(x_train,
y_train)
        tree = DecisionTreeClassifier().fit(x_train, y_train)
        # Send message back to client
        # Comparation test data with train data
        message = "{:.3f}".format(logreg.score(x_test, y_test))
        message1 = "{:.3f}".format(nb.score(x_test, y_test))
        message2 = "{:.3f}".format(svm.score(x_test, y_test))
        message3 = "{:.3f}".format(nn.score(x_test, y_test))
        message4 = "{:.3f}".format(tree.score(x_test, y_test))
```

```
        n_features = 19
        markers = ('Abstract semantic verbs', 'Archaisms', 'Complex
conjunctions', 'Desemantisized verbs', 'Future', 'IPronoun',
'Intensifying adverbs', 'Logic connectors', 'Noun', 'OrSuffix',
'PassiveVoice', 'Past', 'PersonalPronoun', 'Prepositive attributes',
'Postpositive attributes', 'Present', 'ThatThose', 'WePronoun',
'YouHeShe')
        plt.barh(range(len(markers)), tree.feature_importances_)
        plt.yticks(np.arange(len(markers)), markers)
        plt.xlabel("The importance of characteristic")
        plt.ylabel("The characteristic")
        plt.tight_layout()
        pic = plt.savefig("static\content\pic.png")

        return dict(
        m = message,
        m1 = message3,
        m2 = message1,
        m3 = message4,
        m4 = message2)
```

Б.2. Листинг файла index.tpl

```
% rebase('layout.tpl', title='Home Page', m = m, m1 = m1, m2 = m2, m3
= m3, m4 = m4)

<table align="left" width="80%" border="0">
     <tr>
          <td>
               <table align="top" width="25%" border="0">
               <tr>
               <td>
               <input type="file" name="file" id="file">
               </td>
               <td>
               <button type="submit" id="submit" class="btn btn-
primary" >Load</button>
               </td>
               </tr>
               </table>

               <br><br><table align="top" width="80%" border="4">
     <tr>
          <td>
          Name of method
```

```
     </td>
     <td>
     Compliance
     </td>
</tr>

<tr>
     <td>
     Logistic Regression
     </td>
     <td>
     {{m}}
     </td>
</tr>

<tr>
     <td>
     Neural Networks
     </td>
     <td>
     {{m1}}
     </td>
</tr>

<tr>
     <td>
     Naive Bayes
     </td>
     <td>
     {{m2}}
     </td>
</tr>

<tr>
     <td>
     Decision Tree
     </td>
     <td>
     {{m3}}
     </td>
</tr>

<tr>
     <td>
     SVM
     </td>
```

```
          <td>
          {{m4}}
          </td>
     </tr>

</table> </br></br>



          </td>
          <td>
               <img src="../static/content/pic.png" width="450" />
          </td>
     </tr>

     <tr>
     <br><table align="left" width="25%" border="0">
     <tr>
          <td>

          <a href="/filename.pdf" download class="btn btn-
primary">Download results</a>
          </td>
     </tr></table></br></tr>
```

Б.3. Листинг файла app.py

```
"""
This script runs the application using a development server.
"""

import bottle
import os
import sys

# routes contains the HTTP handlers for our server and must be
imported.
import routes

if '--debug' in sys.argv[1:] or 'SERVER_DEBUG' in os.environ:
    # Debug mode will enable more verbose output in the console
window.
    # It must be set at the beginning of the script.
    bottle.debug(True)

def wsgi_app():
    """Returns the application to make available through wfastcgi.
This is used
```

```
    when the site is published to Microsoft Azure."""
    return bottle.default_app()

if __name__ == '__main__':
    PROJECT_ROOT = os.path.abspath(os.path.dirname(__file__))
    STATIC_ROOT = os.path.join(PROJECT_ROOT, 'static').replace('\\',
'/')
    HOST = os.environ.get('SERVER_HOST', 'localhost')
    try:
        PORT = int(os.environ.get('SERVER_PORT', '5555'))
    except ValueError:
        PORT = 5555

    @bottle.route('/static/<filepath:path>')
    def server_static(filepath):
        """Handler for static files, used with the development
server.
        When running under a production server such as IIS or Apache,
        the server should be configured to serve the static files."""
        return bottle.static_file(filepath, root=STATIC_ROOT)

    # Starts a local test server.
    bottle.run(server='wsgiref', host=HOST, port=PORT)
```

Приложение В. Сценарии тестирования

В таблице В.1. представлены тестовые сценарии, а также их ожидаемый и реальный результат.

Таблица В.1. Сравнение точности методов машинного обучения

Тест	Ожидаемый результат	Реальный результат	Соответствие
Нажатие на кнопку «Выберите файл»	Открытие диалогового окна с выбором файла	Открытие диалогового окна с выбором файла	+
Выбор файла и нажатие на кнопку «Open»	Загрузка файла в систему	Загрузка файла в систему	+
Нажатие на кнопку «Cancel»	Закрытие диалогового окна	Закрытие диалогового окна	+
Нажатие на кнопку «Load»	Вывод таблицы с коэффициентами соответствия и графиком влияния маркеров	Вывод таблицы с коэффициентами соответствия и графиком влияния маркеров	+
Нажатие на кнопку «Download results»	Выгрузка таблицы в Excel-файл	Выгрузка таблицы в Excel-файл	+

Тестирование производится программным способом, программный код представлен ниже:

```
# Creation of data set
x, y = make_blobs(center_box=[0.0, 1.0], cluster_std = 0.5, n_samples
= 150, centers=2, n_features=19, random_state=0)

# Division on test set and train set
x_train, x_test, y_train, y_test = train_test_split(x, y,
random_state=0)
# Learning system
logreg = LogisticRegression().fit(x_train, y_train)
nb = GaussianNB().fit(x_train, y_train)
svm = SVC().fit(x_train, y_train)
nn = MLPClassifier(hidden_layer_sizes=(6, 2)).fit(x_train, y_train)
tree = DecisionTreeClassifier().fit(x_train, y_train)
# Send message back to client
# Comparation test data with train data
print("Logistic Regression: {:.3f}".format(logreg.score(x_test,
y_test)))
print("Naive Bayes: {:.3f}".format(nb.score(x_test, y_test)))
print("SVM: {:.3f}".format(svm.score(x_test, y_test)))
print("Neural Networks: {:.3f}".format(nn.score(x_test, y_test)))
print("Decision Tree: {:.3f}".format(tree.score(x_test, y_test)))
```

Для тестирования использованы данные нескольких статей. Тесты производились от наиболее положительных значений маркеров к наиболее отрицательным. Ниже

представлен результат тестирования анализа статьи с положительными значениями маркеров:

Логистическая регрессия: 0.974

Наивный Байес: 1.000

Метод опорных векторов: 0.974

Нейронные сети: 0.868

Метод деревьев решений: 0.737

Ниже представлен результат тестирования анализа статьи со средними значениями маркеров:

Логистическая регрессия: 0.711

Наивный Байес: 0.816

Метод опорных векторов: 0.789

Нейронные сети: 0.632

Метод деревьев решений: 0.711

Ниже представлен результат тестирования анализа статьи с отрицательными значениями маркеров:

Логистическая регрессия: 0.579

Наивный Байес: 0.632

Метод опорных векторов: 0.421

Нейронные сети: 0.553

Метод деревьев решений: 0.500

Из данных тестовых случаев следует, что с каждой загрузкой статей среднее значение вычисленных коэффициентов снижается. Первый случай: 0,911; второй случай: 0,732; третий случай: 0,537. Данный факт означает, что алгоритм работает должным образом.

Приложение Г. Руководство пользователя

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

УТВЕРЖДАЮ

Старший преподаватель кафедры
информационных технологий в бизнесе
НИУ ВШЭ – Пермь

_____ В.В. Ланин

«_____»_____ 2018 г.

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АНАЛИЗА ПУБЛИКАЦИЙ НА АНГЛИЙСКОМ ЯЗЫКЕ НА СООТВЕТСТВИЕ НАУЧНОМУ СТИЛЮ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

Руководство пользователя

ЛИСТ УТВЕРЖДЕНИЯ

РАЗРАБОТЧИК

_____ В.С. Фалалеева
«___»_____ 2018 г.

Пермь 2018

АННОТАЦИЯ

В документе приводится описание использования приложения для анализа публикаций на английском языке.

Содержание и правила заполнения программного документа приведены с соответствующими государственными стандартами ГОСТ 34.

Содержание

1. ВВЕДЕНИЕ

Руководство пользователя написано для приложения анализа научных публикаций на английском языке на соответствие научному стилю. В данном приложении пользователь может проанализировать загруженную им самостоятельно публикацию на соответствие уровню академического английского языка. После вычислений система выводит таблицу с коэффициентами соответствия научному стилю с помощью пяти различных методов. 1 – публикация полностью соответствует научному стилю, 0 – публикация полностью не соответствует. Пользователь самостоятельно решает, какой коэффициент его устраивает.

Система производит анализ исходя из количества различных маркеров, подсчитанных в работе (подглава 1.3). На основании подсчитанных данных система выводит график важности влияния каждого маркера на коэффициент соответствия. Пользователь, просмотрев данный график, может убедиться какие маркеры стоит исправить в первую очередь.

1.1. Область применения

Данный документ применяется для написания Выпускной квалификационной работы.

Программа предназначена к применению в Пермском филиале Национального Исследовательского Университета «Высшая Школа Экономики» (далее – НИУ ВШЭ – Пермь) для определения уровня академического стиля написанной работы.

1.2. Краткое описание возможностей

Система предоставляет возможность проверить загруженную публикацию на уровень научного стиля.

1.3. Уровень подготовки пользователя

Пользователь должен иметь некоторые филологические навыки для распознавания маркеров и умения их корректировать в своей работе.

Также пользователи системы должны иметь опыт эксплуатации персонального компьютера и операционной системы Windows 7, 8, 8.1, 10 (англоязычная версия).

1.4. Эксплуатационная документация

Пользователи в обязательном порядке должны быть ознакомлены с настоящим Руководством.

2. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ

2.1. Виды деятельности, функции, для автоматизации которых предназначено данное средство

Данное средство предназначено для анализа публикаций на английском языке на соответствие научному стилю.

2.2. Условия, при соблюдении которых обеспечивается применение средства в соответствии с его назначением

Наличие у пользователей системы достаточной квалификации для грамотных действий при эксплуатации системы описаны в п. 1.3 настоящего Руководства.

3. ПОДГОТОВКА К РАБОТЕ

3.1. Порядок загрузки данных и программ

Перейти к соответствующей вкладке портала «Paper Cat».

3.2. Порядок проверки работоспособности

Пользователю не требуется проводить дополнительных действий для проверки работоспособности программы – если при клике на вкладку, система переходит на страницу (рис.6).

4. ОПИСАНИЕ ОПЕРАЦИЙ

4.1. Описание главного окна приветствия и загрузки публикации

Пользователь видит следующее окно программы (рис.1.):

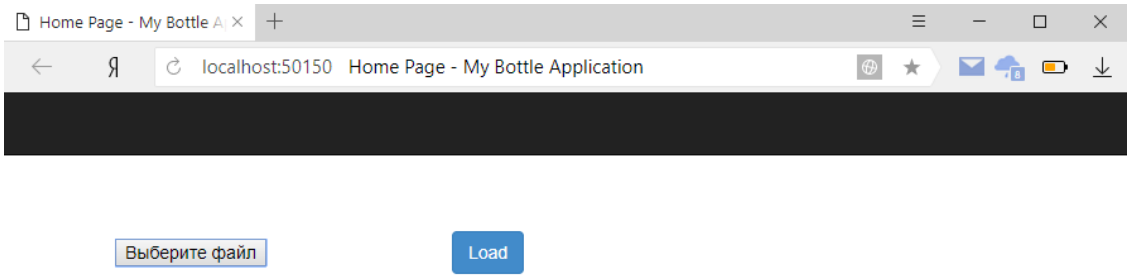

Рис. 1. Главное окно системы

После чего пользователю необходимо нажать на кнопку «Выберите файл» и выбрать текстовый файл с публикацией. После этого название файла появится рядом с кнопкой «Выберите файл», как показано на рисунке 2.

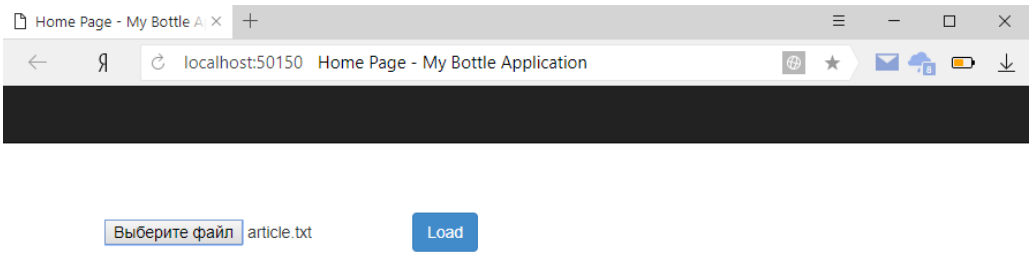

Рис. 2. Загрузка публикации

После этого пользователю необходимо нажать на кнопку «Load», затем система выведет коэффициенты соответствия и график с важностью влияния каждого маркера, как показано на рисунке 3.

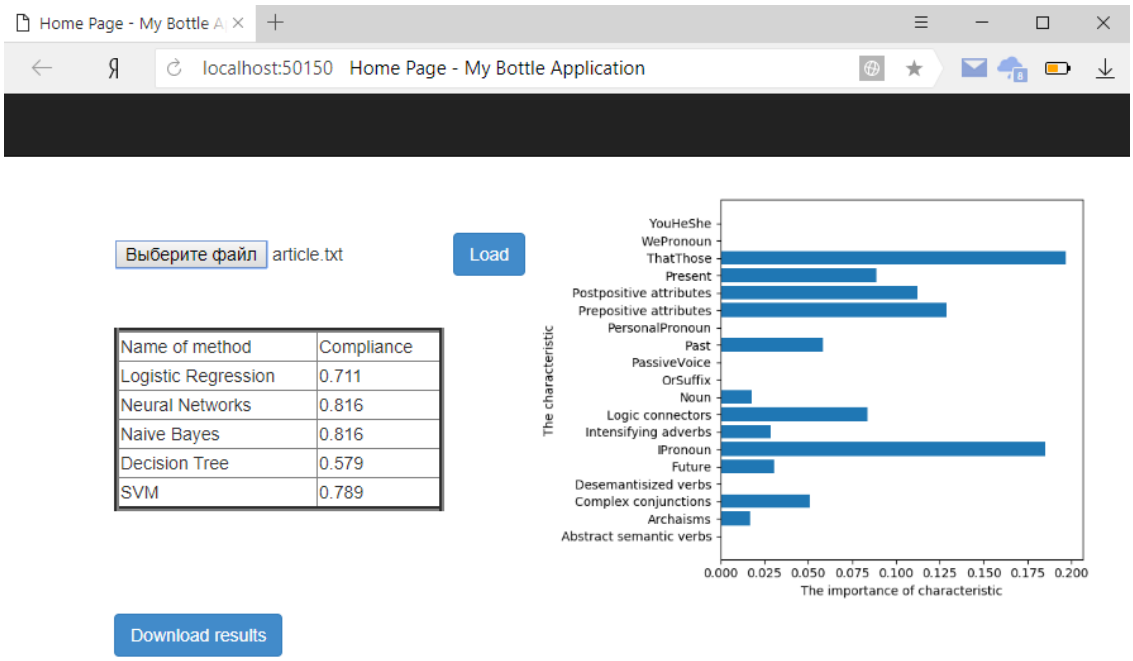

Рис. 3. Результаты анализа

Если это необходимо, пользователь может выгрузить результаты таблицы коэффициентов соответствия в Excel-файл, нажав на кнопку «Download results». Пример файла представлен на рисунке 4.

	A	B
1	Name of method	Compliance
2	Logistic Regression	0.711
3	Neural Networks	0.816
4	Naive Bayes	0.816
5	Decision Tree	0.579
6	SVM	0.789

Рис. 4. Пример выгрузки результатов расчета коэффициентов соответствия

5. АВАРИЙНЫЕ СИТУАЦИИ

5.1. Действия в случае несоблюдения условий выполнения технологического процесса

При несоблюдении технологических условий программы следует обращаться к администратору системы либо к системному администратору НИУ ВШЭ – Пермь. Специалисты проведут диагностику и исправление неполадки, руководствуясь п. 7 "Руководства системного программиста".

5.2. Действия по восстановлению программ и/или данных при отказе магнитных носителей или обнаружении ошибок в данных

Программа не защищена от отказа магнитных носителей.

При обнаружении ошибок в данных системному программисту необходимо согласовать с администратором отчёт об ошибке и устранить её самому или передать свои права команде по исправлению проекта.

5.3. Действия в случаях обнаружении несанкционированного вмешательства в данные

Несанкционированные данные не должны повлиять на структуру работы программы.