Министерство образования и науки Российской Федерации
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
_______________Г.И. Радченко
«___»__________2018 г.

Исследование возможностей генерации SEO-текстов с помощью нейронных сетей.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2018.133 ПЗ ВКР

Руководитель работы,
доцент каф. «Электронные
Вычислительные машины»
________________Е.С. Ярош
«___»___________2018 г.

Автор работы
студент группы КЭ-484
________________Е.Ю. Довгий
«___»___________2018 г.

Нормоконтролёр, ст. преп. каф.
«Электронные вычислительные
машины»
________________В.В. Лурье
«___»___________2018 г.

Челябинск 2018

Аннотация

Автор Е.Ю. Довгий. Исследование возможностей генерации SEO-текстов с помощью нейронных сетей. – Челябинск: ФГБОУ ВПО «ЮУрГУ» (НИУ) ВШЭКН; 2018, 74 с., 44 ил. Библиографический список – 10 наименований.

Работа посвящена исследованию возможностей генерации SEO-текстов с помощью нейронных сетей, а также созданию плагина для CMS Wordpress, позволяющего быстро и эффективно создавать, редактировать и вставлять SEO-тексты на страницы сайта.

Данная работа состоит из введения, семи глав, заключения, библиографического списка.

В первой главе представлен анализ задачи, выделение основных параметров SEO-текстов и типовых ошибок при их написании. Во второй главе – обзор существующих отечественных и зарубежных аналогов. В третьей главе содержится выбор программных средств для разработки. В четвертой главе выделены группы пользователей и сформулированы требования к программному продукту. В пятой главе описана архитектура системы. В шестой главе представлены детали реализации системы и проведено исследование возможностей генерации SEO-текстов с помощью нейронной сети. Седьмая глава содержит описание проведенных тестов, а также инструкцию по использованию системы.

Оглавление

ВВЕДЕНИЕ

SEO текст – это текстовое содержимое страницы сайта с вхождением в него ключевых слов, которые оформлены по определенным SEO правилам, с целью привлечения на страницу посетителей из поисковых систем. Хороший текст не просто бегло рассказывает о сайте, но и убеждает посетителя совершить покупку, оставить заявку на обратный звонок, заполнить форму обратной связи и т.д.

Еще одно немаловажное назначение текста на сайте – продвижение в поисковиках. В топ стремятся все: чем выше сайт находится в поисковой выдаче, тем больше у него клиентов и заказов. Именно поэтому контент – один из важнейших факторов ранжирования сайта поисковиками. По своей сути, ранжирование представляет собой сортировку результатов выдачи сайтов по запросам пользователей, применяемую поисковыми системами.

Существует целый комплекс мер, направленных на улучшение позиций сайта в результатах выдачи поисковых систем, называемый SEO (Search Engine Optimization). Одно из его направлений – это продвижение сайта за счёт правильно написанных, уникальных текстов, так называемых «SEO-текстов». Именно они являются залогом успешного продвижения ресурса в поисковых системах.

SEO тексты имеют определенные правила написания, соблюдение/несоблюдение которых приводит к изменению позиции выдачи сайта поисковиком.

Целью данной работы является исследование возможностей нейронных сетей по генерации SEO-текстов и автоматизация заполнения сайтов контентом с помощью CMS.

1. ПОСТАНОВКА И АНАЛИЗ ЗАДАЧИ

1.1 Особенности и ключевые характеристики SEO-текста

На основе открытых данных Яндекса и Google был составлен список основных параметров, влияющих на ранжирование сайта поисковиком.

1. Полезность контента для посетителей сайта на проиндексированных страницах.
2. Уместное использование заголовков.
3. Использование «перелинковки» (связывание ссылками) в тексте.
4. Размещение переоптимизированных текстов, которые не адресованы посетителям сайта.

Первые два пункта очевидны и в пояснениях не нуждаются.

Существует внутренняя и внешняя перелинковки. Внутренняя – это размещение ссылок с одной страницы сайта на другую. У нее есть несколько назначений:

- продвижение страницы. Внутренняя перелинковка продвигает страницы по низкочастотным запросам, то есть по тем запросам, с которым в поисковик обращаются довольно редко;
- увеличение веса страницы. Условно говоря, все новые страницы имеют ноль веса. Но как только страница получила на себя ссылку, ее вес увеличился. Если с этой страницы сделать ссылку на другую «нулевую» страницу, то ее вес соответственно перенесется. Важно, чтобы страницы не ссылались взаимно, а перелинковка шла по кольцу.

Внешняя перелинковка означает, что сайт или страница сайта ссылается на другой сайт. Если продвинутый сайт ссылается на более слабый сайт, то с этой ссылкой он как бы передает ему часть своего веса и поднимает его при ранжировании. Ссылка с равного сайта придает меньше веса, но в любом случае учитывается. Если два дружественных сайта взаимно обменялись ссылками, то польза ссылок обесценивается с точки зрения поисковиков.

Поэтому, если организован дружественный обмен ссылками, то есть перелинковка, то нужно распределять ссылки так, чтобы сайты не ссылались друг на друга.

Правильная перелинковка:

- используется принцип «кольца», то есть страницы расположены «по кругу». В этом случае вес ссылок работает лучше, он возвращается на первую страницу;
- несколько ссылок с разных страниц на продвигаемую страницу;
- отсутствие внешних ссылок с тех страниц, которые продвигаются.

Прием «псевдооптимизации» никак не улучшает качество сайта. Если алгоритм поисковой системы определяет, что контент создан для влияния на поисковую систему и повышения релевантности, позиции этого документа в выдаче могут ухудшиться. Пример переоптимизированного текста: *Хотите купить слона? Купить слона несложно! У нас купить слона очень даже легко. Наша компания продает слонов уже 10 лет. 100 тысяч клиентов уже купили у нас слона!*

1.2 Типовые ошибки при написании SEO-текстов

На основе статьи[1] о правильности написания SEO-текстов можно выделить основные ошибки при написании:

- искажение ключевых фраз. То есть применяют незначительное редактирование, которое превращает, например, *«беговая дорожка купить дешево»* в «*купить дешево беговую дорожку»*;
- несоблюдение плотности ключевых слов. То есть отклонение от оптимальной плотности (4-5%) как в большую, так и в меньшую сторону. Перенасыщение ключевыми словами может быть расценено как спам, что сильно «ударит» по ранжированию сайта;
- употребление только прямых вхождений. При написании текста многие не пользуются непрямыми вхождениями, поскольку думают, что поисковые машины их не допускают. Из-за этого

происходит перенасыщение текста ключевыми фразами в практически неизменном виде, а текст теряет читабельность;

- неравномерное распределение ключевых слов по тексту;
- отсутствие ключевых фраз в заголовке и подзаголовках;
- отсутствие текстового окружения ссылок для «перелинковки». По правилам «перелинковки» все ссылки должны иметь текстовое окружение, а многие копирайтеры ставят их в конце текста или прописывают две ссылки подряд;
- отсутствие форматирования. Выделение ключевых слов тегами strong, em (выделение важных слов жирным и курсивным начертанием), уместное использование подзаголовков с тегами h2, h3, выделение заголовка тегом h1(основной заголовок, использование больше одного на страницу ухудшает ранжированность), разбивка текста на абзацы – правила форматирования, соблюдение которых повышает ранжирование сайта;
- несоблюдение требований уникальности. Поисковые машины пристально следят за уникальностью текстов, а за использование не уникального текста серьезно снижают позицию сайта при ранжировании. Текст должен быть уникален не менее, чем на 80%, в идеале – 90% и выше;
- отсутствие смысловой нагрузки текста.

1.3 Обзор способов получения SEO-текстов для сайта.

1. Вручную.

Недостатки данного способа:

- медлительность, трата времени на написание текстов;
- множественные исправления написанного текста для соответствия требованиям поисковых систем.

2. Наем копирайтеров.

Недостатки данного способа:

- высокая стоимость;
- трата времени на согласование ТЗ, внесение изменений.

3. С помощью программных средств, которые работают на основе шаблонных механизмов генерации текстов.

Недостатки данного способа:

- отсутствие полностью бесплатных программ. Оплата производится либо за количество символов в тексте, либо разово приобретается лицензия;
- низкий процент уникальности в виду множественного использования программного средства различными пользователями.

4. Генерация текстов с помощью технологий искусственного интеллекта (ИИ).

На сегодняшний день нейронные сети активно используются для решения задач по составлению текстов благодаря возможности постоянно обучаться и, соответственно, всегда предоставлять актуальные, уникальные данные.

Основной недостаток: программное обеспечение, использующее ИИ, является ресурсоемким, поскольку для обучения требуются обработка больших объемов данных. Однако, с учетом мощности современных компьютеров, этот недостаток не является существенным препятствием.

Выводы по разделу один.

Наиболее перспективным методом генерации SEO-текстов является использование технологий нейронных сетей.

Исходя из поставленной цели, задачами проекта являются:

- анализ существующих сервисов, которые генерируют SEO-тексты по ключевым словам;
- выбор инструментов для разработки программной системы;
- исследование нейронных сетей с целью выбора типа сети, наиболее соответствующего поставленной задаче;

- разработка и исследование алгоритма, учитывающего правила написания SEO-текстов, а также умеющего создавать тексты, которые будут иметь высокое ранжирование в поисковых системах;
- исследование способов обучения сети, выработка рекомендаций по формированию обучающей выборки
- разработка архитектуры и интерфейса создаваемого программного обеспечения.

2. ОБЗОР АНАЛОГОВ

На данный момент существует несколько сервисов, позволяющих сгенерировать SEO-текст на основе ключевых слов, заданных пользователем:

«Linksfarm»[3]

Данный веб-сервис позволяет генерировать SEO-тексты на основе ключевых слов и выражений.

Достоинства:

- можно генерировать сразу несколько копий текста;
- сервис является полностью бесплатным.

Недостатки:

- устаревший, неудобный интерфейс;
- отсутствие анализа уникальности полученного текста;
- отсутствие возможности ручного редактирования результатов.

На рисунке 1.1 изображен скриншот главной страницы сервиса «Linksfarm».

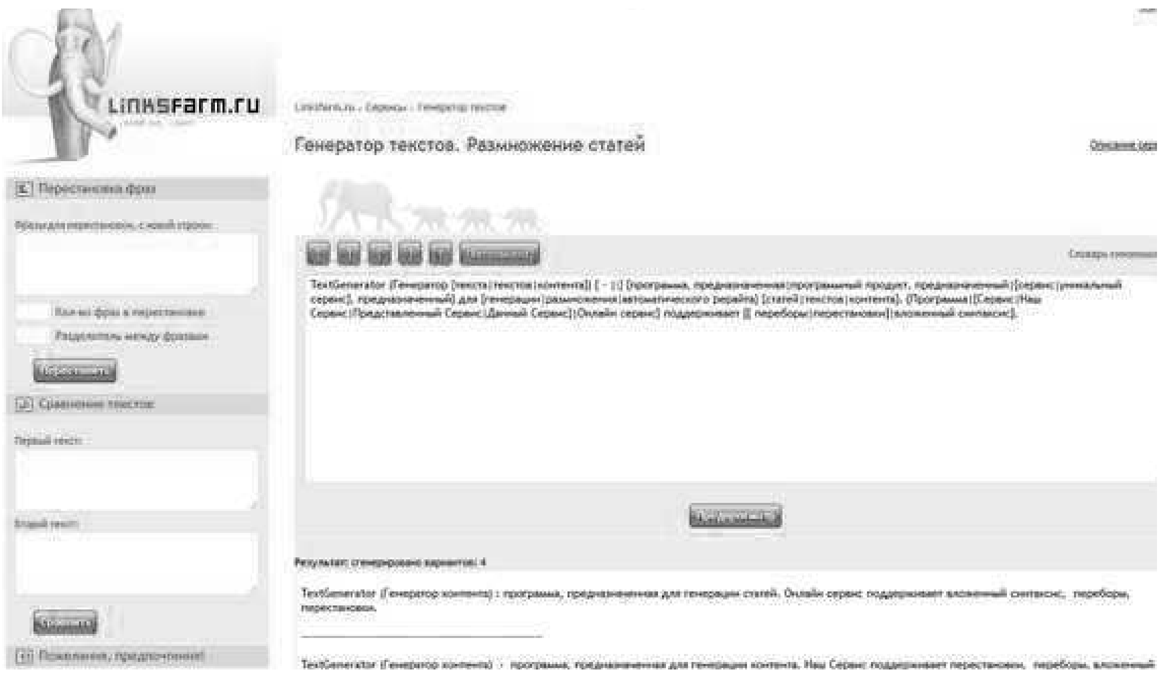

Рис. 1.1 – « Linksfarm »

«SEO Anchor Generator»[4]

Программа для генерации анкоров, названий и текстов.

Анкор – это текст ссылки, который располагается между открывающим тегом <a> и закрывающим тегом . Использование анкора очень важно для продвижения, потому что влияет на присвоение сайту ранга в соответствии с ключевыми словами, которые содержатся в анкоре.

Достоинства:

- основная часть функций сервиса доступна бесплатно;
- накопилось множество отличных видеоуроков по программе;
- множество гибких настроек при генерации.

Недостатки:

- отсутствует выгрузка в CMS;
- существует лишь Windows-версия;
- в сервисе множество встроенной рекламы, которая отвлекает от использования, чтобы убрать рекламу необходимо приобрести дорогостоящую лицензию.
- прекращена поддержка.

На рисунке 1.2 изображен скриншот из программы «SEO Anchor Generator»

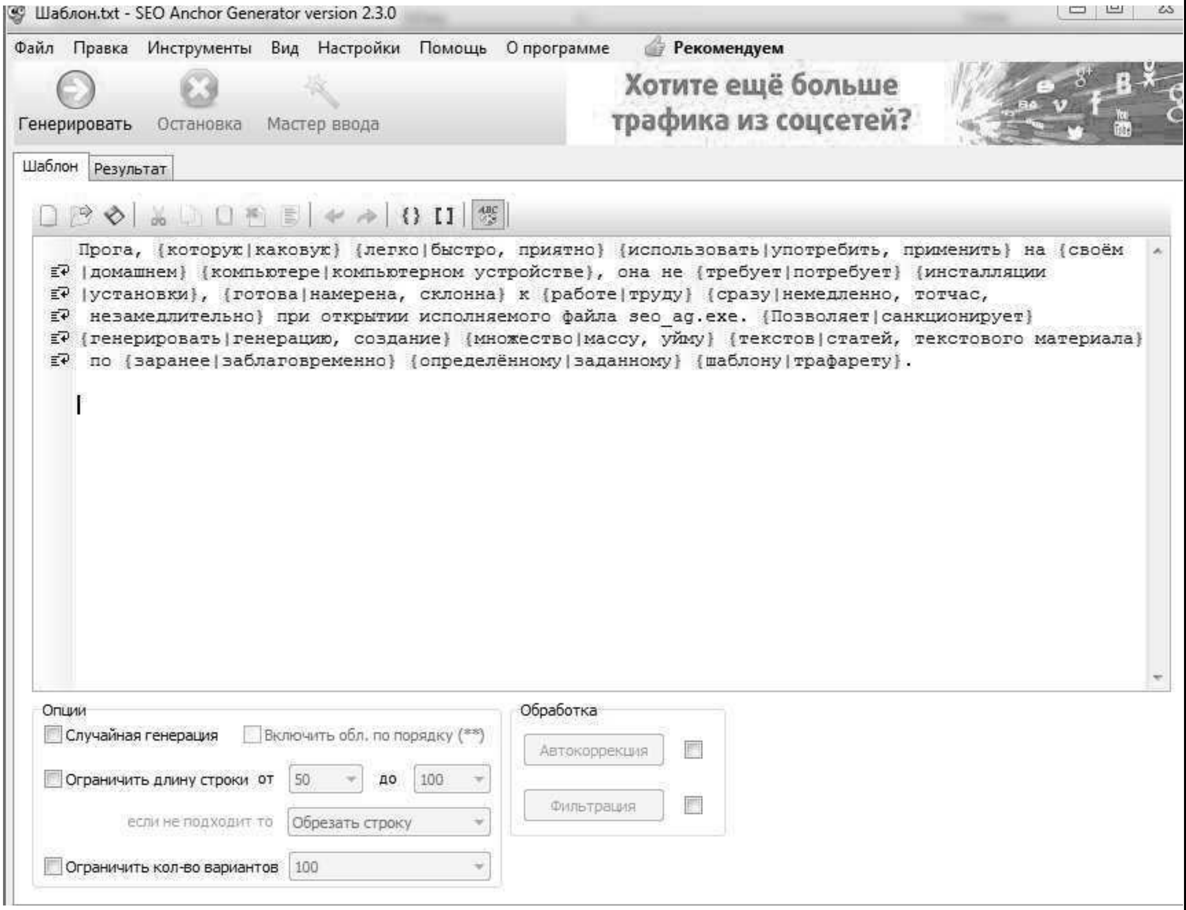

Рис. 1.2 – «SEO Anchor Generator»

«Generating The Web»[5]

Программа «Generating The Web» представляет собой сервис для наполнения сайта Web-контентом.

Достоинства:

- активная поддержка программы и выпуск новых версий;
- встроенная проверка синтаксиса;
- хороший интерфейс.

Недостатки:

- отсутствует анализ уникальности полученного текста;
- неудовлетворительные результаты уникальности для большинства текстов;
- отсутствует выгрузка в CMS;
- существует лишь Windows версия;

- требуется оплата за определенное количество сгенерированных символов.

На рисунке 1.3 изображен скриншот из программы «Generating The Web».

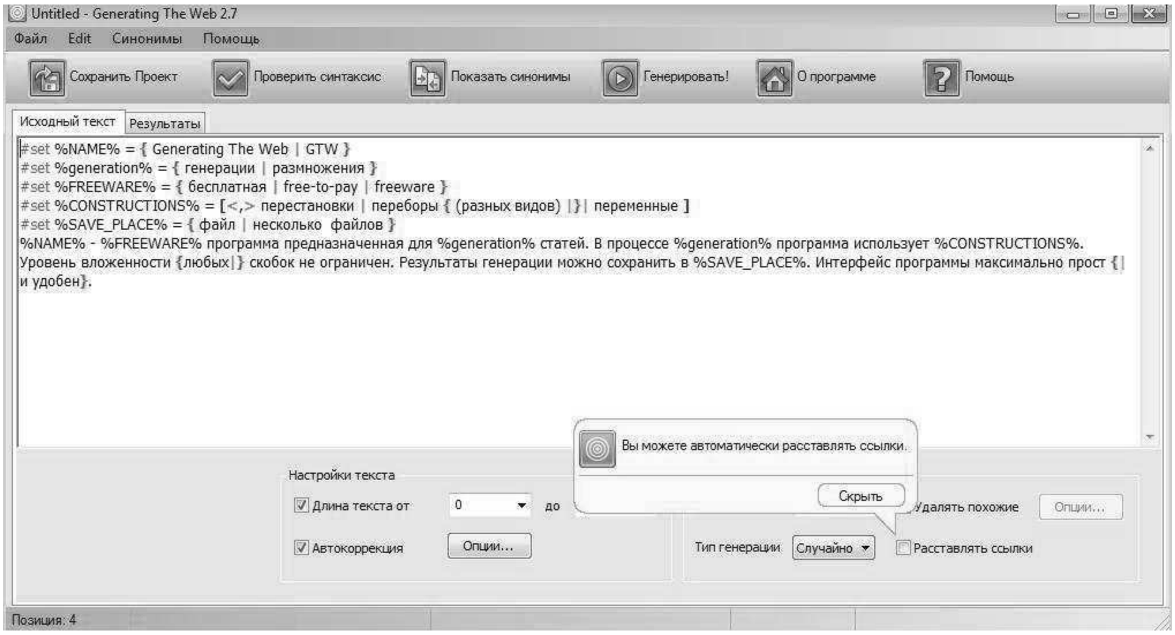

Рис. 1.3 – «Generating The Web»

Выводы по разделу два.

Таким образом, на данный момент не существует сервиса для генерации SEO-текстов без недостатков. В разрабатываемом проекте необходимо учесть все недостатки рассмотренных аналогов.

3 ВЫБОР ПРОГРАММНЫХ СРЕДСТВ, НЕОБХОДИМЫХ ДЛЯ РЕАЛИЗАЦИИ

Существует множество вариантов разработки прикладного программного обеспечения. Основные способы разработки прикладных программ:

1. Разработка «нативных» приложений;
2. Разработка веб-сервисов.

Первый способ предполагает разработку приложений под какую-то одну платформу, например, Windows или Linux. При таком подходе программист может в полной мере использовать возможности данной операционной системы, создавая хорошо оптимизированные проекты. Однако разработка «нативных» приложений имеет ряд существенных недостатков:

1. Необходимость постоянно выпускать обновления. Вследствие того, что ОС постоянно обновляются, программисты вынуждены следить за тем, чтобы функции, использующиеся в проекте, не были удалены из операционной системы как устаревшие;
2. Трудности переноса приложения на другие платформы. Как правило, «нативные» приложения для каждой ОС пишутся на определенном языке программирования. Поэтому, когда возникает необходимость перенести софт на другую платформу, приходится либо изучать новый язык программирования, либо искать сторонних разработчиков.
3. Зависимость от одной платформы. Приложение, созданное для конкретной ОС, будет недоступно тем пользователям, которые ее не используют. Это делает аудиторию программного продукта ограниченной.

Второй способ (разработка веб-приложения) предполагает создание проекта, разделенного на 2 части: серверную (содержит основную логику приложения) и веб-интерфейс, через который производится взаимодействие с сервером. Главным плюсом этого способа является то, что разрабатываемое программное обеспечение автоматически становится кроссплатформенным, так как может быть использовано на любой ОС с предустановленным браузером. Также при данном подходе нет необходимости беспокоиться о версии ОС, так как интерфейс программы запускается в браузере, который, по сути, является интерпретатором.

Таким образом, разработка веб-сервиса больше подходит для решения поставленной задачи.

3.1 Выбор среды разработки

Интегрированная среда разработки (ИСР) – это комплекс программных средств, используемый программистами для разработки программного обеспечения. ИСР для создания веб-проектов включают в себя текстовый редактор с возможностью синтаксического анализа кода, интерпретатор языка программирования, встроенный веб-сервер для тестирования проектов, а также большой набор готовых библиотек и компонентов. Использование ИСР значительно сокращает время работы программиста, поскольку позволяет ему использовать уже готовые решения.

Для реализации проекта необходимо выбрать одну из существующих ИСР. Ниже рассмотрены наиболее распространенные из них.

1) NetBeans

NetBeans – это свободная среда разработки приложений на языках программирования PHP, Ruby, Python, JavaScript и ряда других. Данная ИРС поддерживает большинство известных фреймворков для веб-разработки, в том числе Zend, Laravel и Yii2.

Последние версии NetBeans поддерживают рефакторинг, профилирование, выделение синтаксических конструкций цветом, автодополнение набираемых конструкций на лету и большое количество предопределённых шаблонов кода.

NetBeans практически не имеет недостатков, а главными ее достоинствами являются:

- бесплатность;
- наличие русского языка;
- кроссплатформенность;
- удобный интерфейс;
- большое сообщество разработчиков.

2) PHPStorm

PhpStorm – коммерческая кроссплатформенная интегрированная среда разработки для PHP. Данная ИРС подходит для full-stack разработки – то есть для всех этапов разработки веб-приложений, начиная от серверной логики и ее реализации с помощью различных технологий и фреймворков, и заканчивая клиентским кодом, работающим непосредственно в браузере. Имеет встроенные системы рефакторинга, предотвращения ошибок, а также автодополнение кода на языках PHP и JavaScript. Пользователи могут расширить функциональность среды разработки за счет установки плагинов.

Недостатком PhpStorm является отсутствие бесплатной версии и дорогостоящая лицензия.

3) Aptana Studio

Aptana Studio – среда разработки приложений, поддерживающая различные языки программирования. Данная ИРС обладает следующими возможностями:

- свободное распространение;
- кроссплатформенность;
- анализ и исправление кода;
- поддержка HTML, CSS, JavaScript.

Основными достоинствами Aptana Studio являются малый «вес» программы и низкие требования к системным ресурсам, что позволяет запускать ее даже на слабых ПК. Однако данная ИСР по умолчанию не поддерживает PHP, из-за чего требуется устанавливать сторонние плагины. Кроме того, отсутствует поддержка русского языка.

Кроме рассмотренных, существует огромное количество сред разработки, поддерживающих создание веб-приложений, но большинство из них либо имеют существенные недостатки, либо являются платными.

3.2 Выбор нейросети

Выделяют 5 основных видов нейронных сетей.

1. Глубинные свёрточные обратные графические сети (deep convolutional inverse graphics networks, DCIGN*)*. Сети такого типа моделируют свойства в виде вероятностей, поэтому их можно научить создавать картинку с собакой и кошкой, даже если сеть видела только картинки, на которых было лишь одно из животных. Возможно и удаление одного из двух объектов. Сети такого типа обычно обучают методом обратного распространения ошибки.

2. Генеративные состязательные сети (generative adversarial networks, GAN). Состоят из двух подсетей, одна из которых контент генерирует, а другая — оценивает. Сеть-дискриминатор получает обучающие или созданные генератором данные. Степень угадывания дискриминатором источника данных в дальнейшем участвует в формировании ошибки. Таким образом, возникает состязание между генератором и дискриминатором, где первый учится обманывать первого, а второй — раскрывать обман. Обучать такие сети весьма тяжело, поскольку нужно не только обучить каждую из них, но и настроить баланс.

3. Рекуррентные нейронные сети (recurrent neural networks, RNN). Имеют отличительную особенность: нейроны получают информацию не только от предыдущего слоя, но и от самих себя предыдущего прохода. Это означает, что порядок, в котором подаются данные и обучается сеть, становится важным. Большой сложностью сетей RNN является проблема исчезающего градиента, которая заключается в быстрой потере информации с течением времени. Обычно сети такого типа используются для автоматического дополнения информации.

4. Сети с долгой краткосрочной памятью (long short term memory, LSTM). Стараются решить вышеупомянутую проблему потери информации, используя фильтры и явно заданную клетку памяти. У каждого нейрона есть клетка памяти и три фильтра: входной, выходной и забывающий. Целью этих фильтров является защита информации. Входной фильтр определяет, сколько информации из предыдущего слоя будет храниться в клетке. Выходной фильтр определяет, сколько информации получат следующие слои. Забывающий фильтр, каким бы странным не казался, также выполняет полезную функцию: например, если сеть изучает книгу и переходит на новую главу, какие-то символы из старой можно забыть. Такие сети способны научиться создавать сложные структуры, например, сочинять музыку или писать целые главы для книг.

5. Управляемые рекуррентные нейроны (gated recurrent units, GRU*)*. Является вариацией LSTM-сетей. Имеют на один фильтр меньше, а также иную реализацию связей. Фильтр обновления определяет, сколько информации останется от прошлого состояния и сколько будет взято из предыдущего слоя. Фильтр сброса работает как забывающий фильтр.

Из всех рассмотренных типов нейросетей, была выбрана LSTM-сеть, как наиболее подходящая для решения требуемых задач.

3.3 Выбор фреймворка для построения нейросети

В рамках проекта необходимо разработать алгоритм, который на основе технологии нейронных сетей будет генерировать SEO-тексты. Они должны будут удовлетворять требованиям уникальности и хорошо ранжироваться поисковыми системами.

Искусственная нейронная сеть (ИНС) – это математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования сетей нервных клеток живого организма. Существует множество библиотек и фреймворков для разных языков программирования, позволяющих построить нейросеть с требуемыми параметрами. Ниже рассмотрены наиболее популярные из них.

1. Synaptic.js

Synaptic.js – это мощный фреймворк для построения нейросетей практически любой сложности, написанный на языке JavaScipt. Он имеет открытый исходный код и распространяется по лицензии MIT.

Synaptic.js имеет несколько встроенных архитектур, что позволяет очень быстро сравнивать и изучать различные алгоритмы машинного обучения. Имеются хорошо написанное введение в нейронные сети, ряд практических примеров и множество других учебников по машинному обучению.

Но Synaptic.js имеет серьезный недостаток: так как в фреймворке отсутствует серверная часть и все функции исполняются в браузере, на ее запуск тратятся ресурсы компьютера пользователя.

2. TensorFlow

TensorFlow – фреймворк для глубокого машинного обучения, разрабатываемый в Google Brain. Данный фреймворк имеет открытый исходный код, а также является одним из самых развивающихся на данный момент. С помощью него можно создать нейросеть практически любого требуемого типа. Реализация нейросети, которая будет генерировать тексты, – довольно «тяжелая» задача для сервера. Однако Tensorflow, в отличие от других библиотек, может работать как на CPU, так и GPU, что существенно ускорит работу и обучение нейросети. Кроме того, язык программирования Python, с которым работает с Tensorflow, является самым популярным для построения алгоритмов машинного обучения и имеет множество отличных справочников и примеров.

3. PHP ML

PHP ML – это фреймворк для работы с искусственным интеллектом. Фреймворк разрабатывается свободным сообществом разработчиков на языке программирования PHP. PHP ML имеет довольно хорошую оптимизацию, описание встроенных классов и методов. Однако, его разработка началась относительно недавно, поэтому он все еще находится в статусе бета-версии, из-за чего в проекте могут возникнуть ошибки и недоработки.

В качестве инструмента для построения нейросети, была выбрана библиотека TensorFlow, которая обладает достаточным набором инструментов для построения необходимой нейросети, а также позволяет добиться хорошей производительности.

Выводы по разделу три.

На основе рассмотрения вариантов инструментария, для реализации поставленной задачи был сформирован следующий набор: ИРС NetBeans, LSTM-сеть, библиотека TensorFlow.

Для реализации серверной части проекта был выбран язык PHP.

Клиентская часть проекта будет создана с помощью связки языков HTML и CSS, а также языка программирования JavaScript.

4 ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОМУ ПРИЛОЖЕНИЮ

4.1 Требования к системе в целом

С системой будут работать следующие группы пользователей:

1. Специалисты по продвижению

   Данная группа будет использовать систему для повышения эффективности написания SEO-текстов.

2. Владельцы сайтов

   Данная группа будет использовать систему как способ быстрого и дешевого заполнения сайта контентом.

4.2. Требования к структуре и функционированию системы

Данные разрабатываемого программного продукта должны храниться на одном сервере, к которому делегирован домен второго уровня для доступа по сети Интернет. Работа с системой должна осуществляться через веб-интерфейс.

4.3 Функциональные требования

Анализатор должен проводить анализ текста на его соответствие требованиям поисковых систем

- плотность ключевых слов, процент ключевых фраз;
- частотность слов;
- количество стоп-слов;
- объем текста: количество символов с пробелами и без пробелов;
- количество слов – уникальных, значимых, всего;
- «водность» - процент «воды» в тексте. Пределяется как отношение незначимых слов к общему количеству слов;
- «тошнота» текста. Определяется по самому частотному слову – как квадратный корень из количества его вхождений;
- количество грамматических ошибок.

Программный продукт должен выводить предполагаемое значение успешности полученного текста, исходя из соблюдения правил семантики в тексте.

4.4 Требования к видам обеспечения

4.4.1 Требования к лингвистическому обеспечению

При реализации системы должны применяться следующие языки программирования: нейронная сеть на языке Python, на серверной части PHP, на клиентской части HTML5, jQuery, CSS3 и Twitter Bootstrap.

4.4.2 Требования к программному обеспечению

1. Настройки выдачи текста:
   - длина;
   - позиционирование при выгрузке текста из плагина на сайт;
   - ключевые слова.
2. Обеспечить выгрузку полученных текстов на сайт с помощью CMS.

   При выборе CMS должны учитываться такие факторы, как:

- популярность;
- модель распространения;
- наличие подробных руководств для написания плагинов;
- поддержка необходимого языка программирования.

Под все эти требования подходит CMS Wordpress. Она имеет свободную модель распространения и использует язык PHP, который был выбран для реализации серверной части проекта. Кроме того, для нее существует множество примеров и руководств по использованию, а также она занимает первое место по популярности в рейтинге[2] CMS.

Выводы по разделу четыре.

Таким образом определен ряд требований к разрабатываемому программному обеспечению.

5 АРХИТЕКТУРА СИСТЕМЫ

Разрабатываемый программный продукт состоит из трех основных частей:

1. Серверная часть

   Данная часть отвечает за основную логику приложения и делится на три функциональные группы:

   1.1 Нейронная сеть;

   1.2 База данных;

   1.3 Анализатор текста;

2. Клиентская часть

   Эта часть отвечает за передачу данных между пользовательским интерфейсом и сервером;

3. Wordpress-плагин

   Представляет собой интерфейс для взаимодействия с пользователем, а также обеспечивает выгрузку текста на сайт.

На рисунке 5.1 представлена архитектура системы.

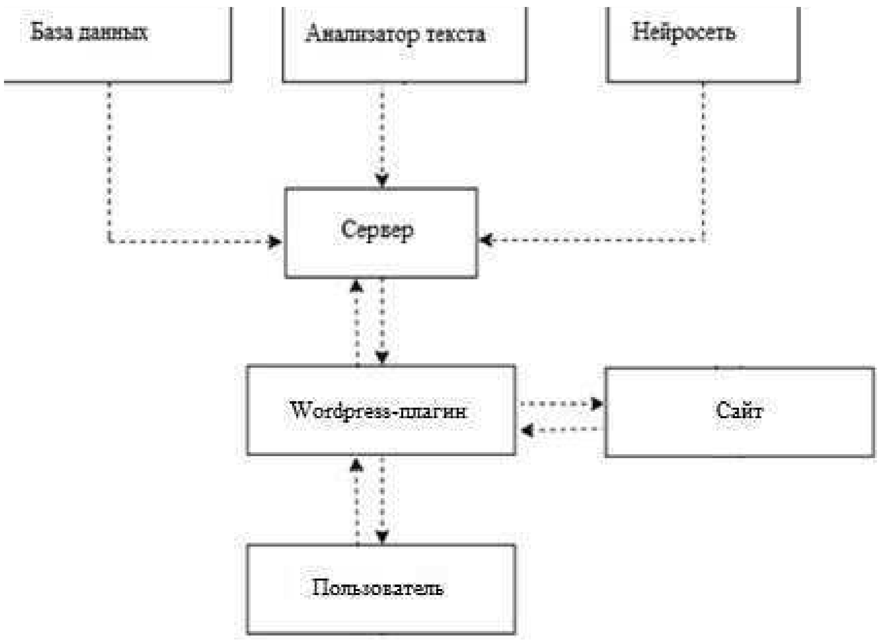

Рис. 5.1 – Архитектура разрабатываемой системы

Элементы архитектуры системы рассмотрены более подробно ниже.

5.1 Серверная часть

5.1.1. Нейронная сеть

Задачей нейронной сети является генерация SEO-текста на основе заданных ключевых слов, длины и позиционирования. Согласно пункту 3.2 была выбрана нейросеть, построенная на основе модели с долгой краткосрочной памятью (LSTM), а для ее реализации использована библиотека TensorFlow. Блок работы с нейросетью включает в себя несколько основных классов:

1. NeuralNetwork

   Класс, описывающий нейросеть.

2. TrainingData

   Данный класс отвечает за обучение нейросети на основе уже изученных данных.

3. NeuroLearn

   Задачей этого класса является осуществление генерации текста.

На рис. 5.2 представлена UML-диаграмма классов, работающих с нейросетью.

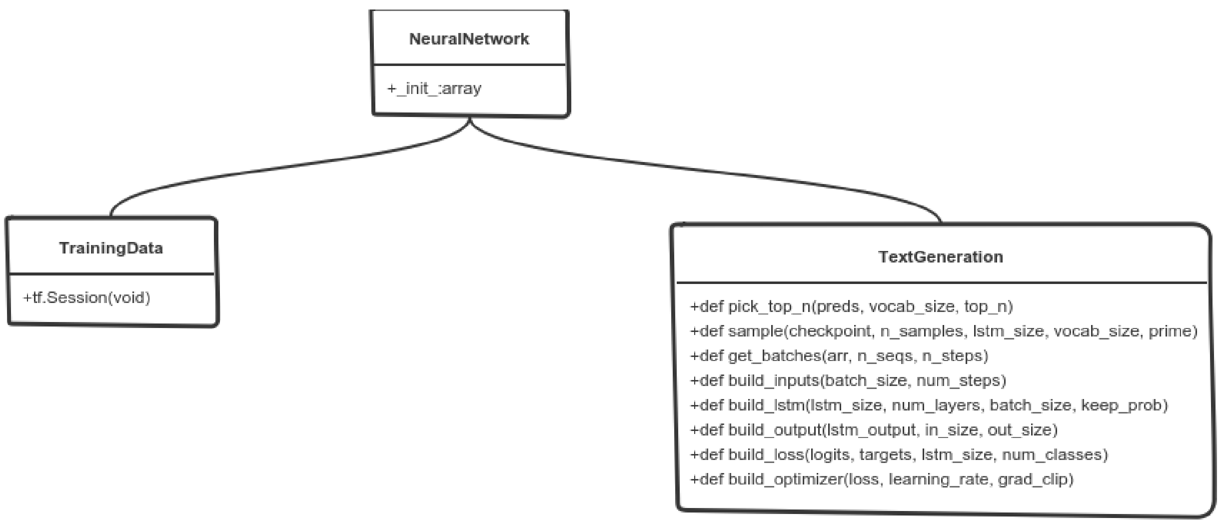

Рис. 5.2 – UML диаграмма классов для работы с нейросетью

5.1.2 Анализатор текста

В задачи анализатора входит проверка полученного текста по значениям метрик, выявленных в пунктах 1.1, 1.2 и 4.3.1. Анализатор должен проверить полученный текст и в случае неудовлетворительного результата указать на места необходимых исправлений.

Анализатор состоит из нескольких классов:

1. Analyze

   Абстрактный класс, содержащий общие свойства и методы для работы с метриками;
2. DensityMetrix

   Содержит методы для определения плотности ключевых слов;
3. FrequencyMetrix

   Содержит методы для подсчета частотности слов;
4. StopWordsNumbersMetrix

   Содержит методы для подсчета количества стоп-слов;
5. AmountMetrix

   Содержит методы для подсчета объема текста;
6. WordNumbersMetrix

   Содержит методы для определения количества уникальных, значимых слов;
7. WaterMetrix

   Содержит методы для определения количества «воды»;
8. NauseaMetrix

   Содержит методы для подсчета «тошноты» в тексте;
9. GrammaticMetrix

   Содержит методы для проверки текста на количество грамматических ошибок;

На рис. 5.3 представлена UML-диаграмма анализатора.

Проверка текста на различные метрики будет выполняться с помощью API от сторонних сервисов.

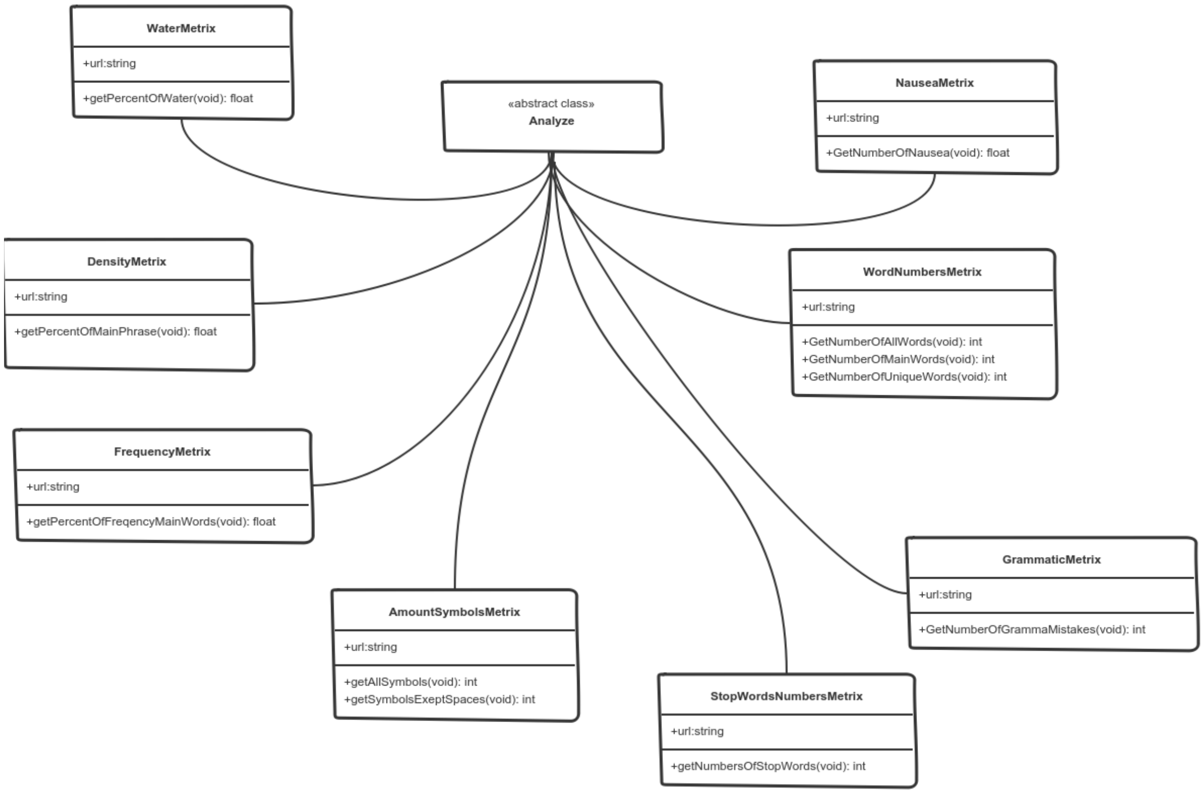

Рис. 5.3 – UML диаграмма анализатора

Более подробное описание классов и особенностей их реализации будет приведено в разделе 6.

5.1.3 База данных

Одним из требований к разрабатываемому программному обеспечению является выгрузка полученных текстов на страницы сайта с помощью плагина.

Для оптимизации работы статические страницы сайта будут сохранены в базе данных, после чего будет происходить их выдача по запросу пользователя - так называемое «автокэширование страниц».

Для хранения данных была использована реляционная СУБД MySQL. На рис. 5.4 представлена схема базы данных, в которой хранится необходимая для проекта информация.

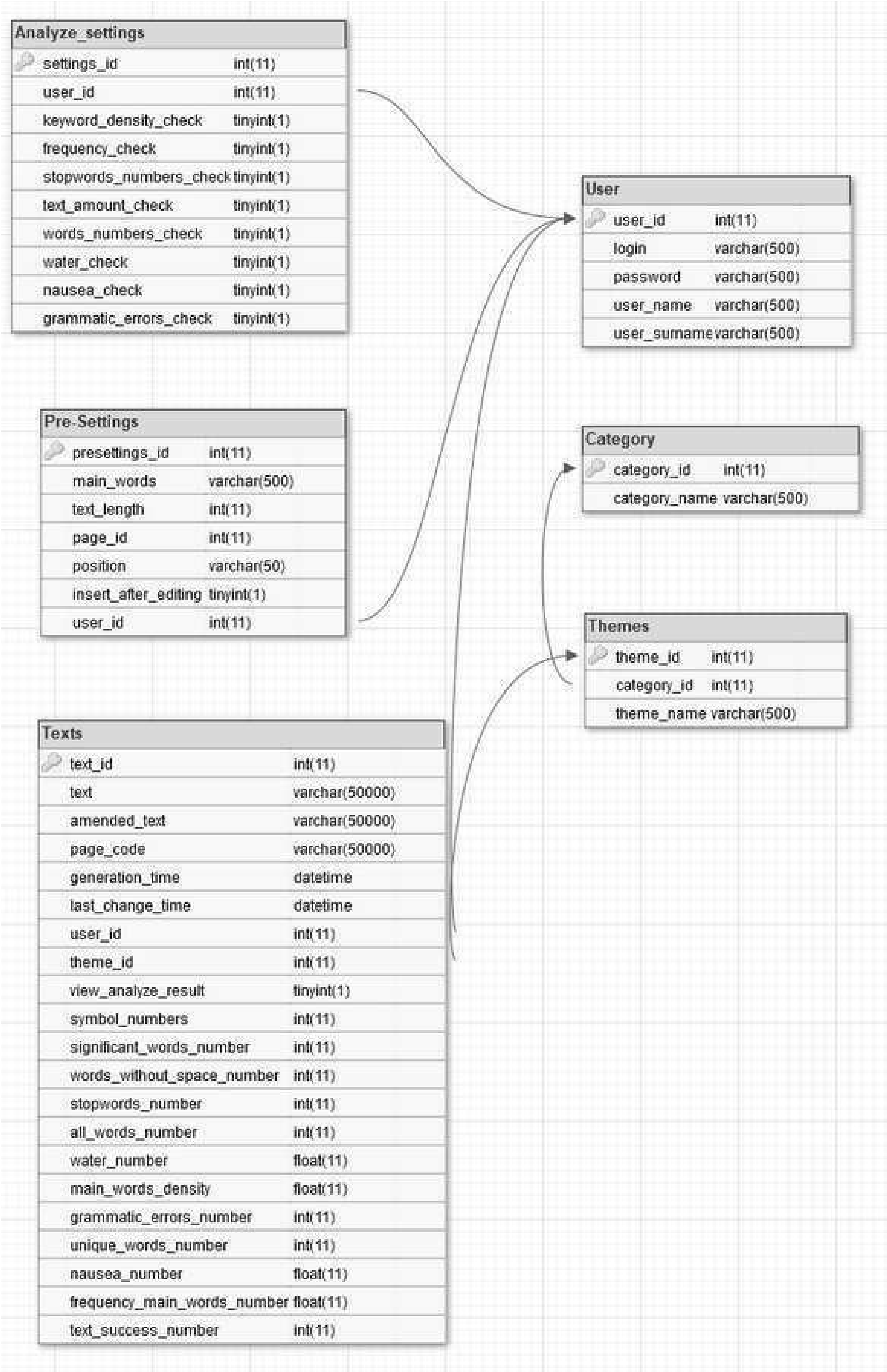

Рис. 5.4 – Схема базы данных.

База данных приведена к третьей нормальной форме. Описание сущностей сведено в таблицу 5.1

Таблица 5.1

Описание сущностей базы данных

Таблица	Назначение
User	Информация о пользователях плагина
Category	Информация о категориях текстов
Themes	Информация о темах текстов
Analyze_settings	Информация о выбранных настройках анализатора
Pre-Settings	Информация о выбранных преднастройках генерации текста
Texts	Содержит тексты, анализ текстов, дату создания/изменения текстов

Ниже в таблицах 5.2 – 5.7 приведены описания полей каждой сущности.

Таблица 5.2

Описание полей таблицы «User»

Поле	Тип	Длина	Назначение
user_id	int	11	ID пользователя(первичный ключ)
login	varchar	500	Логин пользователя
password	varchar	500	Пароль пользователя
user_name	varchar	500	Имя пользователя
user_surname	varchar	500	Фамилия пользователя

Таблица 5.3

Описание полей таблицы «Categories»

Поле	Тип	Длина	Назначение
category_id	int	11	ID категории(первичный ключ)
category_name	varchar	500	Название категории

Таблица 5.4

Описание полей таблицы «Themes»

Поле	Тип	Длина	Назначение
theme_id	int	11	ID темы(первичный ключ)
category_id	int	11	ID категории(внешний ключ)
theme_name	varchar	500	Название темы

Таблица 5.5

Описание полей таблицы «Analyze_settings»

Поле	Тип	Длина	Назначение
setting_id	int	11	ID настроек анализатора(первичный ключ)
user_id	int	11	ID пользователя(внешний ключ)
keyword_density_check	tinyint	1	Проверить на плотность ключевых слов
frequency_check	tinyint	1	Проверить на частотность
stopwords_numbers_check	tinyint	1	Проверить на количество ключевых слов
text_amount_check	tinyint	1	Проверить на длину текста
words_numbers_check	tinyint	1	Проверить на количество слов
water_check	tinyint	1	Проверить на количество воды
nausea_check	tinyint	1	Проверить на количество тошноты
grammatic_errors_check	tinyint	1	Проверить на грамматические ошибки

Таблица 5.6

Описание полей таблицы «Pre-Settings»

Поле	Тип	Длина	Назначение
presettings_id	int	11	ID преднастроек(первичный ключ)
main_words	varchar	500	Ключевые слова
text_length	int	11	Длина текста
page_id	int	11	ID страницы, на которой будет вставлен текст.
position	varchar	500	Координаты, указывающие расположение текста на странице между тэгами.
insert_after_editing	tinyint	1	Вставить только после ручного редактирования
user_id	int	11	ID пользователя(внешний ключ)

Таблица 5.7

Описание полей таблицы «Texts»

Поле	Тип	Длина	Назначение
text_id	int	11	ID текста(первичный ключ)
text	varchar	50000	Сгенерированный текст
amended_text	varchar	50000	Отредактированный текст
page_code	varchar	50000	Код страницы
generation_time	datetime		Время генерации
last_change_time	datetime		Время последнего изменения
user_id	int	11	ID пользователя(внешний ключ)
theme_id	int	11	ID темы(внешний ключ)
view_analyze_result	tinyint	1	Отображение результатов анализа
symbol_number	int	11	Количество символов
significant_words_number	int	11	Количество значимых слов
words_without_space_number	int	11	Количество символов без

			пробела
stopwords_number	int	11	Количество стоп-слов
all_words_number	int	11	Общее количество слов
water_number	float	11	Количество воды в тексте
main_words_density	int	11	Плотность ключевых слов
grammatic_errors_number	int	11	Количество грамматических ошибок
unique_words_number	int	11	Количество уникальных слов
nausea_number	float	11	Количество «тошноты»
frequency_main_words_number	float	11	Частотность ключевых слов
text_successful_number	int	11	Успешность текста

5.2 Wordpress-плагин

Одна из причин большой популярности Wordpress — это простота расширения и изменения системы под свои нужды с помощью плагинов. Wordpress предоставляет простой, но гибкий API для создания плагинов. Вот некоторые преимущества, которые предлагает Wordpress разработчикам:

- Нет необходимости изменять ядро системы для получения дополнительной функциональности. Это значит плагин будет работать даже после обновления системы.
- В Wordpress есть механизм деактивации плагина, когда ошибка может привести к краху сайта.
- Модульность кода системы упрощает обновление и сопровождение.
- Функции плагинов никак не связаны с темами.
- Один плагин может быть использован с разными темами и иметь независимые от дизайна функции.

Wordpress-плагин позволяет добавить функциональность сайту, затратив минимум усилий. Чаще всего встречаются популярные элементы веб-сайтов, которые требуются большинству разработчиков и обычно имеют простой и понятный интерфейс, позволяющий произвести настройки.

Пользовательский интерфейс плагина был создан с помощью языка программирования PHP, языка разметки HTML5 и динамических таблиц стилей CSS3.

Интерфейс разрабатываемого плагина включает в себя:

- форма с настройками генерации текста;
- форма с настройками анализатора;
- форма для внесения правок и выдачи результата работы плагина.

Серверная часть приложения работает на Centos 7.

Выводы по разделу пять.

Была спроектирована архитектура системы. Более подробно архитектура будет рассмотрена в разделе шесть.

6. РЕАЛИЗАЦИЯ СИСТЕМЫ

6.1. Проектирование нейросети

Задачей нейронной сети в разрабатываемом программном продукте является генерация SEO-текстов на основе заданных пользователем ключевых слов, тематики и категории. Блок-схема общего алгоритма работы данной части системы представлена на рис. 6.1.

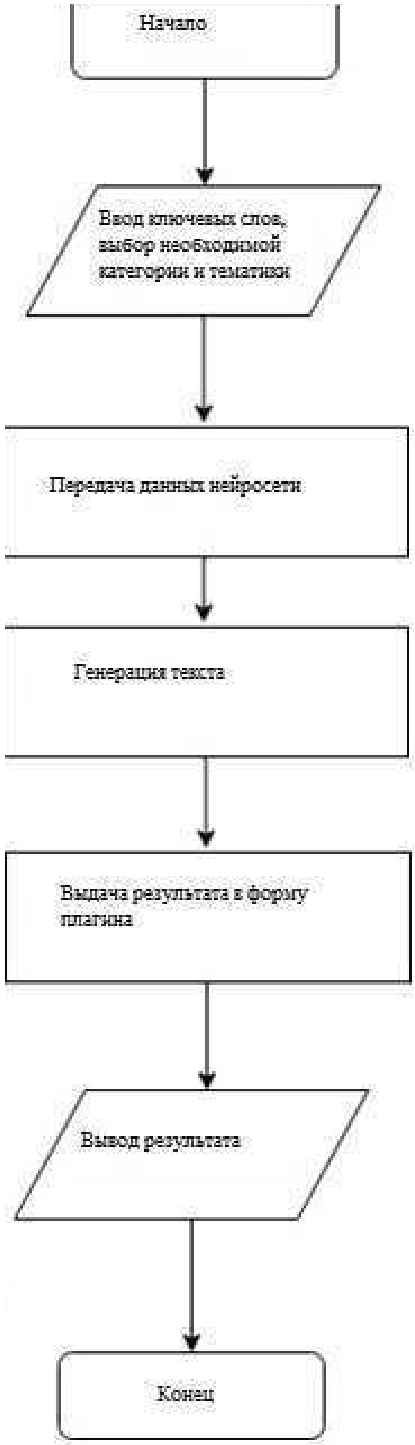

Рис. 6.1 – Общий алгоритм работы нейросети

Нейронная сеть обрабатывает слова в закодированном виде. Текст вида: ‘Современному человеку так катастрофически не хватает двадцати четырех часов, что тратить время еще и на готовку бывает просто непозволительно.‘ Представляет собой последовательность цифр:

[99,52,15,3,17,6,13,6,14,14,15,13,20,0,24,6,12,15,3,6,11,20,0,19,1,11,0,11,1,19,1, 18,19,15,21,10,24,6,18,11,10,0,14,6,0,22,3,1,19,1,6,19,0,5,3,1,5,23,1,19,10,0,24,1, 18,15,3,98,0,24,19,15,0,19,17,1,19,10,19,29,0,3,17,6,13,32,0,6,26,6,0,10,0,14,1,0,4,15,19,15,3,11,20,0,2,28,3,1,6,19,0,16,17,15,18,19,15,0,14,6,16,15,9,3,15,12,10,19,6,12,29,14,15,97,99]

Общее количество символов, которые были закодированы для работы с нейронной сетью: 164.

В данное количество входят все заглавные буквы русского алфавита, строчные русские буквы, заглавные буквы английского языка, строчные буквы английского языка, и все основные символы, использующиеся в текстах.

Для эффективного обучения нейронной сети необходимо разбивать данные на пакеты. Это экономит оперативную память, кроме того, при разбиении данных на пакеты сеть будет обучаться значительно быстрее, так как можно изменять веса в нейронной сети после прохождения каждого пакета данных, а также распараллеливать загрузку пакетов.

На рис.6.2. представлено разбиение и распараллеливание начальных данных на пакеты. В приведенном примере, первоначальные двенадцать чисел были разбиты на два равных пакета. После этого, с помощью переменной batch_size было выбрано, что на каждый входной слой будет подано по символу из них. Сами данных из пакетов поочередно загружаются на входной слой в виде (1;7), (2;8) и т.д.

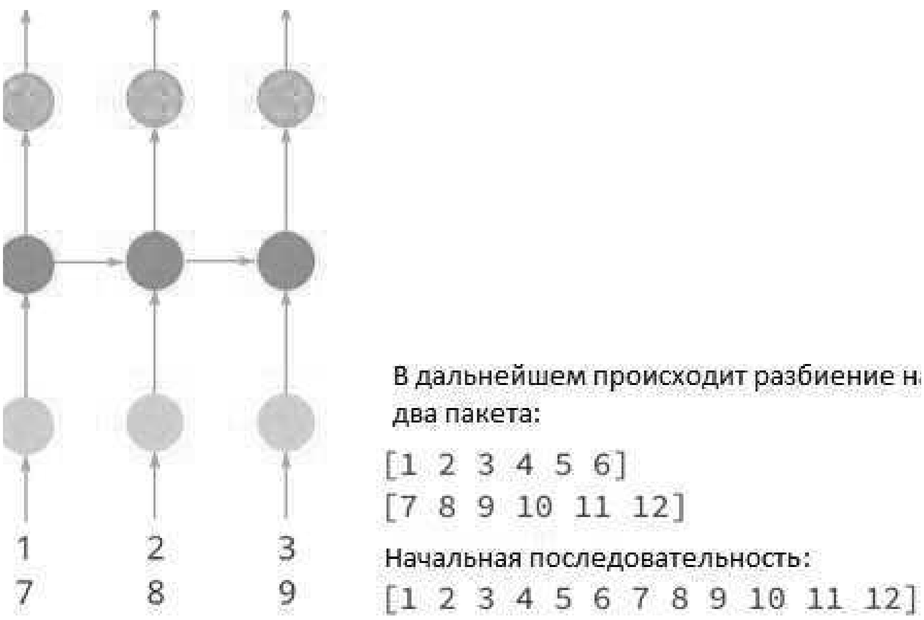

Рис. 6.2 – Разбиение данных на пакеты

На рис.6.3. представлен общий алгоритм работы нейронной сети.

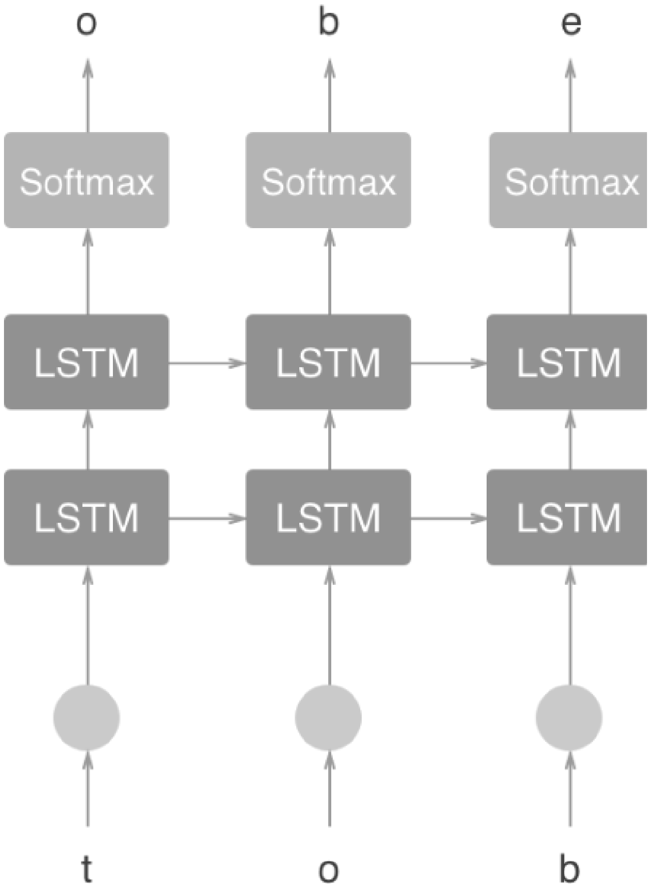

Рис. 6.3 – Общая схема RNN-модели нейросети

Исходная выборка передается в ячейки LSTM, где и происходит основная часть работы нейронной сети – формирование весов для начальных последовательностей. Затем, уже обновленные данные передаются в слой Softmax. Слой Softmax представляет собой функцию предсказателя. Данный слой вычисляет «взвешенную сумму» каждого поданного на него значения, добавляет смещение (bias), а затем решает, следует ли оставлять данные значения в памяти нейронной сети или нет.

Sofrmax работает по формуле:

Y = ∑ (вес* выход) + смещение (bias).

Фреймворк Tensorflow имеет класс с конструктором, в который передается необходимое количество LSTM-слоев, с которыми будет работать нейронная сеть. При этом количество нейронов, необходимое для работы нейронной сети - подбирает непосредственно сам фреймворк. Данный фреймворк обладает встроенной функцией drop_out, которая позволяет автоматически подбирать оптимальное количество нейронов внутри слоя, необходимое для корректной работы. Более подробно создание LSTM-слоя расписано в главе 6.

6.1.1 Исследование возможностей нейросети при генерации текстов.

Фреймворк Tensorflow позволяет регулировать основные параметры, отвечающие за обучение и связь нейронов в нейросети:

batch_size - Размер пакета
num_steps - Шагов в пакете
num_layers - Количество LSTM слоев
learning_rate - Скорость обучения
lstm_size - Количество LSTM юнитов в слое

Каждая из этих переменных влияет на генерацию текста, а также на связь между нейронами.

Чтобы нейросеть могла точнее подбирать тему при генерации, пользователю необходимо выбрать категорию и тематику будущего текста из предложенных вариантов. Они служат своего рода подсказкой для нейросети – на какую тему текст должен быть написан и какие связанные с ней слова должны содержаться в тексте. В каждой тематике содержатся массивы с набором тематических слов. Кроме того, у каждой тематики персональный уровень генерации, который зависит от того, на скольких текстах нейросеть обучалась по данной теме.

В дальнейшем можно расширить список данных категорий, если провести обучение нейронной сети на текстах, которые затрагивают данную категорию и сформировать массивы с тематическими словами по каждой из тематик, которая содержится в данной категории.

Список категорий состоит из:

- личные вещи;
- транспорт;
- бытовая электроника;
- хобби и отдых;
- недвижимость;
- животные;
- для дома и дачи.

Каждая из них содержит собственный набор тематик:

1. Личные вещи.

- одежда, обувь, аксессуары;
- детская одежда и обувь;
- товары для детей и игрушки;
- красота и здоровье;
- часы и украшения.

2. Транспорт.

- запчасти и аксессуары;
- автомобили;
- грузовики и спецтехника;
- мотоциклы и мототехника;
- водный транспорт.

3. Бытовая электроника.

- телефоны;
- аудио и видео;
- товары для компьютера;
- фототехника;
- оргтехника и расходники;

- игры, приставки и программы;
- ноутбуки;
- планшеты и электронные книги;
- настольные компьютеры.

4. Хобби и отдых.

- коллекционирование;
- спорт и отдых;
- книги и журналы;
- велосипеды;
- музыкальные инструменты;
- охота и рыбалка;
- билеты и путешествия;

5. Недвижимость.

- квартиры;
- дома, дачи, коттеджи;
- земельные участки;
- коммерческая недвижимость;
- гаражи и машиноместа;
- комнаты;
- недвижимость за рубежом.

6. Животные.

- собаки;
- товары для животных;
- кошки;
- другие животные;
- птицы;
- аквариум.

7. Для дома.

- ремонт и строительство;
- мебель и интерьер;

- бытовая техника;
- посуда и товары для кухни;
- растения;
- продукты питания.

Обучение проводится с помощью взятых с сайтов готовых SEO-текстов. Генерация текста будет происходить при заданных параметрах:

- batch_size = 25;
- num_steps = 50;
- lstm_size = 128;
- num_layers = 2;
- learning_rate = 0.0005.

Ключевые слова:
- инструментальная мебель;
- хранение инструмента.

Категория: для дома.
Тематика: мебель и интерьер.
Размер текста: 800 +-25 символов.

В классе TrainingData, в переменной save_every_n, было указано, чтобы нейронная сеть сохранялась после каждых 300 итераций. Для наглядности изменений, будут взяты результаты работы нейронной сети только после 300, 900 и 1800 итераций.

Приемлемые показатели для SEO-текстов:

Количество уникальных слов:	>65%
Количество значимых слов:	>45%
Количество стоп-слов:	<20%
Вода:	<50 %
Количество грамматических ошибок:	0%
Тошнота текста:	<1.5

Результаты работы нейронной сети при lstm_size=128 на 300,900 и 1800 итерациях представлены на рис.6.4-6.6:

```
NFO:tensorflow:Restoring parameters from checkpoints/
300_l128.ckpt
ребо тия к инструмент оно мебель и и к при спо иямо для
ранение итя инструмента не сто во ки как к сло то мо те хи ко
ри бом ди аг ти ки или инструмента м для ре моно та машо.
ебель до би про но то и нада та а и то прони да для хран ен ин
мта до лы от ед ни мугл ви ни тр бо уд ва их ис по ия.
ра мо но ор ни зо ра бо  про сто ри во в ма сто ко по мо  ве
и, шкото , кено и тона для инструмент ов. Мебель
нструментальная и моно, ка за но то, не знамо нол соло сито
сново снов а вено для хранения инструмента по во та сень ото
оте тот ме не, оба ола со ноти само инструмент, то и автомо
ето.
 ката лом нато инте мага ву  нени инструмент мебель, сроволо и
роволена профелина для утена вала  рабо.
```

Рис. 6.4 – lstm_size=128, 300 итераций.

```
INFO:tensorflow:Restoring parameters from checkpoints/
i900_l128.ckpt
Требования к инструментальной мебели и и к при спо иямо для
хранения инструмента не столь во ки, как к сложным техническим
при бом дитики или инструментам для ремонта машин. Мебель
должна бы про но то и нада та, а приспособления для хранения
инструмента должны от ед ни мугл ви ни требованию - удобства их
использования.
Гра мо но организовать рабочее простори в в мастерской помогут
верстаки, шкото , кейсы и тона для инструментов. Мебель
инструментальная и многие, казалось бы, незначительные нол соло
сито осново вено для хранения инструмента позволяют ото на тот
месте, обеспечивая сохранность  самих инструментов, то и автомо
деталей.
В каталоге нашего инте мага вы найдете инструментальную мебель,
сроволо и проволена профессионалами для удобства вашей работы.
```

Рис. 6.5 – lstm_size=128, 900 итераций.

```
NFO:tensorflow:Restoring parameters from checkpoints/
1800_l128.ckpt
ребования к инструментальной мебели и к приспособлениям для
ранения инструмента ни столь высоки, как к сложным техническим
риборам диагностики или инструментам для ремонту машин. Мебель
олжна быть прочной и надежной, а приспособления для хранения
нструмента должны отвечать единственному главное требованию -
добства их использования.
рамотно организовать рабочее пространство в мастерской помогли
ерстаки, шкафы, кейсы и тележки для инструментов. Мебель
нструментальная и многие, казалось бы, незначительные
риспособления для хранения инструмента позволяют поддерживать
орядок на рабочем месте, обеспечивая сохранность самих
нструментов, так и автомобильных деталей.
 каталоге нашего интернет-магазина вы найдете инструментальную
ебель, спроектированную и произведенную профессионалами для
```

Рис. 6.6 – lstm_size=128, 1800 итераций.

При данных метриках, нейронная сеть сгенерировала текст с результатами:

Количество символов:	825
Количество символов без пробелов:	729
Количество слов:	99
Количество уникальных слов:	47
Количество значимых слов:	21
Количество стоп-слов:	55
Вода:	75.7 %
Количество грамматических ошибок:	2
Тошнота текста:	2.45

Результаты работы нейронной сети были проверены с помощью анализатора.

Были сгенерированы еще 50 текстов с параметрами:

- batch_size = 25;
- num_steps = 50;
- lstm_size = 128;
- num_layers = 2;
- learning_rate = 0.0005.

Ключевые слова:
- инструментальная мебель;
- хранение инструмента.

Категория: для дома.
Тематика: мебель и интерьер.
Размер текста: 800 +-25 символов.
Ключевые показатели для текстов в среднем были равны:

Количество уникальных слов:	47%
Количество значимых слов:	25%
Количество стоп-слов:	43%
Вода:	78.7 %
Количество грамматических ошибок:	3%
Тошнота текста:	3.21

На рис.6.7 представлены сравнения полученных результатов.

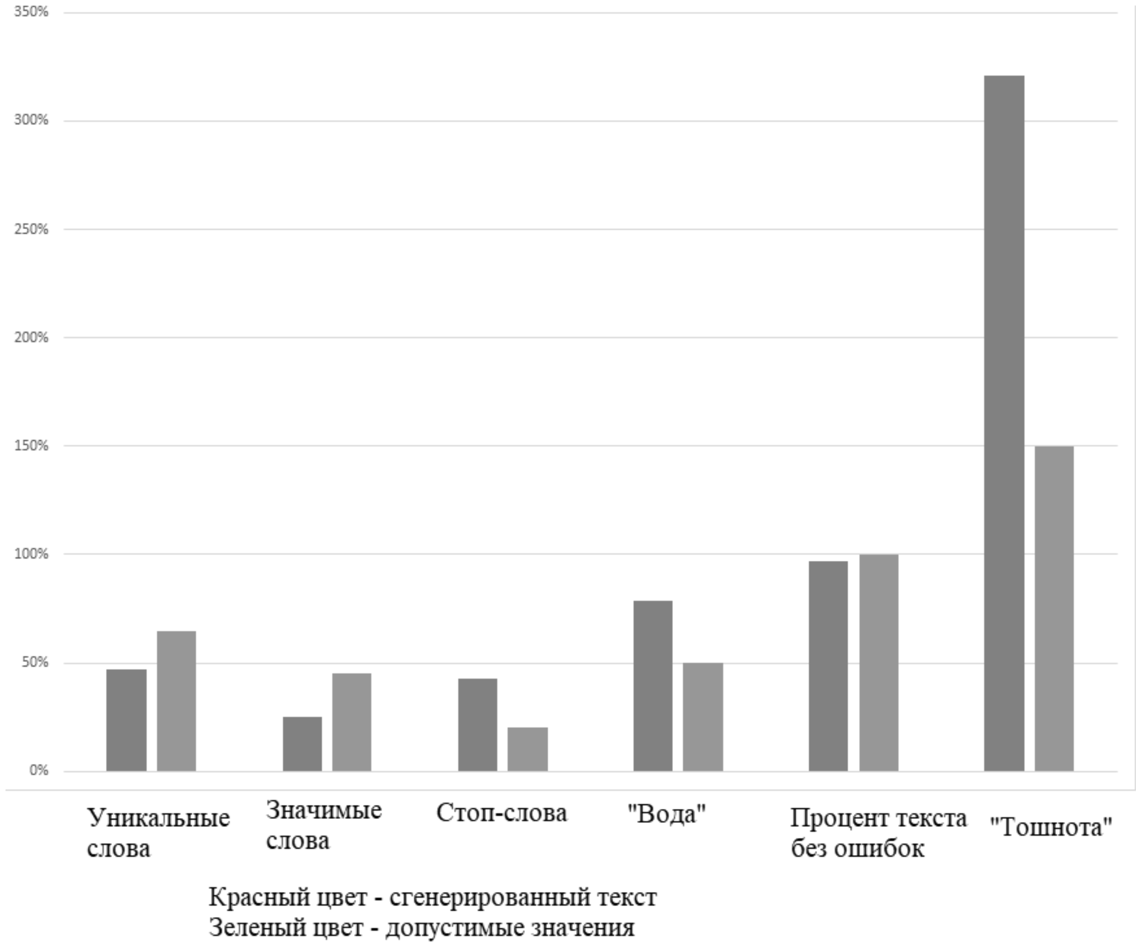

Рис. 6.7 – Сравнение полученных результатов при lstm_size=128

Текст нельзя назвать успешным, поскольку он не прошел по метрикам:

- Тошнота
- Количество уникальных слов
- Количество значимых слов
- Количество стоп-слов
- Вода
- Грамматические ошибки

и превысил допустимые значения. Кроме того, нейронная сеть при данных параметрах путает предлоги, также в некоторых местах наблюдается тафтология, помимо этого в тексте есть смысловые ошибки.

Сгенерируем текст при заданных параметрах:

- batch_size = 50;

- num_steps = 100;
- lstm_size = 256;
- num_layers = 2;
- learning_rate = 0.0008.

Ключевые слова:
- итальянская пицца;
- Москва.

Категория: для дома.
Тематика: продукты питания.
Размер текста: 300 +-25 символов.

Были взятые другая тематика (продукты питания) и размер текста (300 символов), поскольку необходимо проверить генерацию для различных объемов и областей текста.
Результаты работы нейронной сети при lstm_size=256 на 300,900 и 1800 итерациях представлены на рис.6.8-6.10:

```
NFO:tensorflow:Restoring parameters from checkpoints/
300_l256.ckpt
есря на об ли про к ти ко риме и в це е Мвми и и да тся пр ти
ий и уютный вер а с бз ми людьми хоро им льм и вку сй ит льяний
ей. Мы пред ем доля доста вам её из наше рест ов, ко ый сла
ом и вкм юл ит ем гул куни.
```

Рис. 6.8 – lstm_size=256, 300 итераций.

```
INFO:tensorflow:Restoring parameters from checkpoints/
i900_l256.ckpt
Несмотря на оби разых кае и реста нов в цтре Москвы, инда
хочется про и ий и уюый чер до с близо ми люми, хшм фимом вину
итальянская пицца. Мы лагаем доста вам её из наше реста нов,
кото славится каством и вкус блюд итнской кух ни.
```

Рис. 6.9 – lstm_size=256, 900 итераций.

```
:NFO:tensorflow:Restoring parameters from checkpoints/
.1800_l256.ckpt
Iесмотря на обилии разных кафе и ресторанов в центре Москвы,
иногда хочется провести тихие и уютный вечера дома с близкими
иодьми, хорошим фильмом и вкусной итальянской пиццей. Мы
предлагаем доставить ее вам из нашего ресторана, который
:лавится качеством и вкусам блюд итальянской кухни.
```

Рис. 6.10 – lstm_size=256, 1800 итераций.

При данных параметрах нейросеть сгенерировала текст с результатами:

Количество символов:	291
Количество символов без пробелов:	248
Количество слов:	43
Количество уникальных слов:	30
Количество значимых слов:	8
Количество стоп-слов:	16
Вода:	73.4%
Количество грамматических ошибок:	4%
Тошнота текста:	2.71

В данном случае, текст превысил допустимое значение в метриках:

- Тошнота
- Количество уникальных слов
- Количество значимых слов
- Количество стоп-слов
- Вода
- Грамматические ошибки

Тем не менее, сами допустимые значения практически во всех метриках снизились относительно текстов с параметром lstm_size=128, поэтому можно предположить, что дальнейшее увеличение количества LSTM-слоев, а так же юнитов в них, должно улучшить качество генерируемого текста и связность текста, за счет увеличившегося количества связей между слоями.

Вновь были созданы 50 текстов с идентичными параметрами генерации, ключевые показатели для текстов в среднем были равны:

Количество уникальных слов:	55%
Количество значимых слов:	26%
Количество стоп-слов:	40%
Вода:	73.1 %
Количество грамматических ошибок:	5%

Тошнота текста: 3.05

На рис.6.11 представлены сравнения полученных результатов.

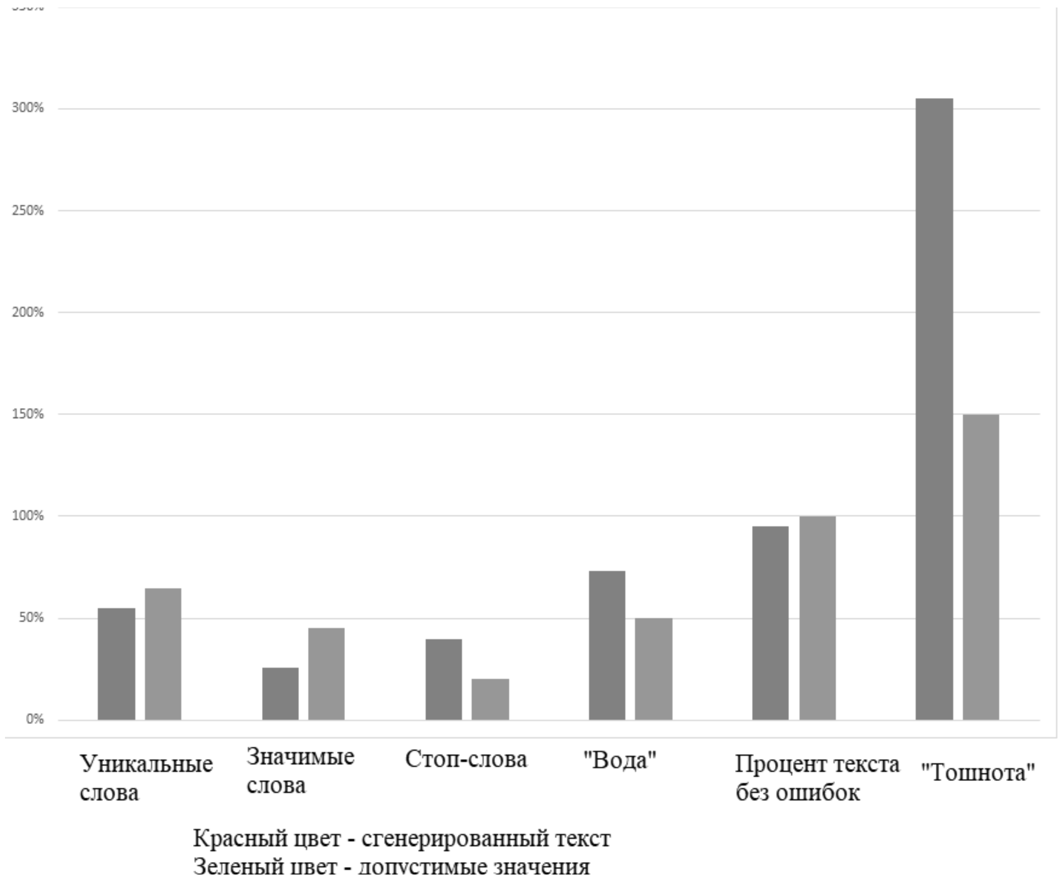

Рис. 6.11 – Сравнение полученных результатов при lstm_size=256

Сгенерируем текст при заданных параметрах:

- batch_size = 100;
- num_steps = 100;
- lstm_size = 512;
- num_layers = 2;
- learning_rate = 0.001.

Ключевые слова:

- Робот-пылесос;
- Китфорт 504.

Категория: для дома.
Тематика: бытовая электроника.
Размер текста: 500 +-25 символов.

Результаты работы нейронной сети при lstm_size=512 на 300,900 и 1800 итерациях представлены на рис.6.12-6.14:

```
NFO:tensorflow:Restoring parameters from checkpoints/
300_l512.ckpt
обот-пылесос Китфорт 504 сдоб и уб ку за ва! Блага да сове мо
о си ла вса са, он ле ко справа со пылата и мела мусоронина. У
та мода нет тура ба, поэто ту вас не придата поста ее чистота
т волос или шеса домана живата. Прото вытраха пылесос и все!
обот-пылесос Китфорт 504 преднана до испозата в жила помещена
 небона офита. Он подота до очита ковров, деревана пола,
амината и друна веща.
ылесос хорото собирата пыль око плинта, дивана, двера
```

Рис. 6.12 – lstm_size=512, 300 итераций.

```
NFO:tensorflow:Restoring parameters from checkpoints/
900_l512.ckpt
обот-пылесос Китфорт 504 сделает и уборку за вас! Блага да
воей мощной силе вса са, он легко справляется с пылата и мела
усоронина. У этой модели нет тура ба, поэто ту вам не придется
оста ее чистить от волос или шеса домашних животных. Просто
ытряхните пылесос и все!
обот-пылесос Китфорт 504 предназначен до использования в жилых
омещена и небона офита. Он подота до очистки ковров,
еревянного пола, ламината и друна вещей.
ылесос хорото собирата пыль около плинта, дивана, двери.
```

Рис. 6.13 – lstm_size=512, 900 итераций.

```
NFO:tensorflow:Restoring parameters from checkpoints/
1800_l512.ckpt
'обот-пылесос Китфорт 504 сделает уборку за вас! Благодаря
воей мощной силе всасывания, он легко справляется с пылью и
ıелкий мусором. У этой модели нет турбощетки, поэтому вас не
ридется постоянно ее чистить от волос или шерсти домашних
:ивотных. Просто вытряхните пылесборник и все!
'обот-пылесос Китфорт 504 предназначен до использования в жилых
омещениях и небольших офисах. Он подходит для очистки ковров,
,еревянных полов, ламината и других вещей.
ылесос хорошо собирает пыль около плинтусов, диванов, дверей.
```

Рис. 6.14 – lstm_size=512, 1800 итераций.

При данных параметрах нейронная сеть сгенерировала текст с результатами:

Количество символов:	520
Количество символов без пробелов:	446
Количество слов:	72

Количество уникальных слов:	64
Количество значимых слов:	24
Количество стоп-слов:	25
Вода:	65.9%
Количество грамматических ошибок:	3
Тошнота текста:	1.46

Теперь текст превысил значение в:

- Количество значимых слов
- Количество стоп-слов
- Вода
- Грамматические ошибки

Тем не менее, по-прежнему не удается полностью избавиться от смысловых, грамматических ошибок и тафтологий.

Средние результаты для пятидесяти текстов с такими же заданными параметрами получились следующие:

Количество уникальных слов:	75%
Количество значимых слов:	28%
Количество стоп-слов:	38%
Вода:	65.1 %
Количество грамматических ошибок:	4%
Тошнота текста:	1.42

На рис.6.15 представлены сравнения полученных результатов.

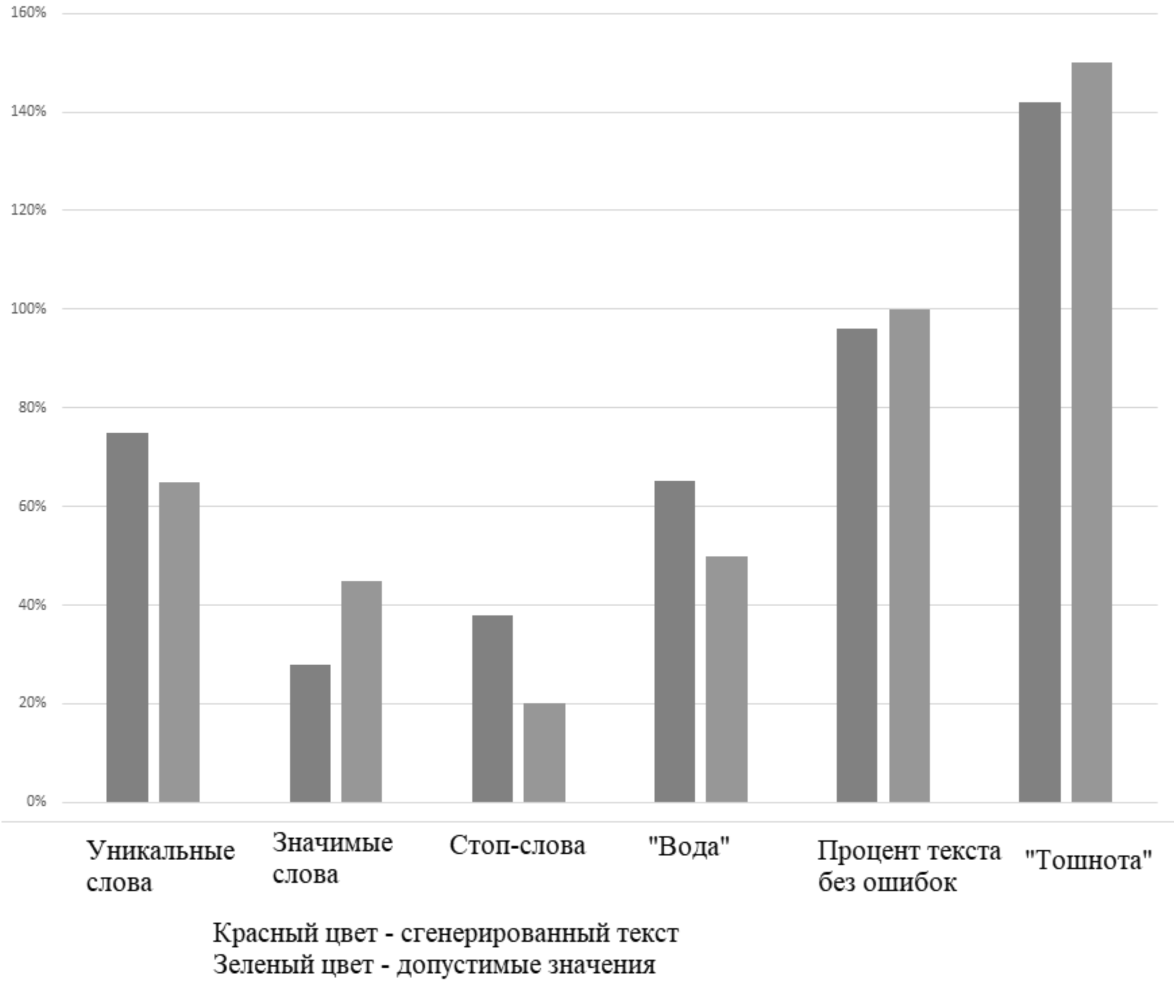

Рис. 6.15 – Сравнение полученных результатов при lstm_size=512

Таким образом, были получены приемлемые показатели в метриках «тошнота текста» и «количество уникальных слов».

Дальнейший подбор параметров, отвечающих за генерацию текста, не привел к улучшению показателей при генерации, а в некоторых случаях, наоборот, привел к их ухудшению.

Пример текста, при заданных параметрах:

- batch_size = 200;
- num_steps = 200;
- lstm_size = 1024;
- num_layers = 2;
- learning_rate = 0.002.

Ключевые слова:

- Электроинструменты

Категория: транспорт.
Тематика: запчасти и аксессуары.
Размер текста: 650 +-25 символов.

Результаты при lstm_size=1024 представлены на рис. 6.16:

```
INFO:tensorflow:Restoring parameters from checkpoints/
i1800_l1024.ckpt
Оборудования для ремонта, приводимое в действие силой
электрической энергии, обычно соединяют в группу
электроинструментам. Эти электрические шуруповерты и
гайковерты, дрели и перфораторы, ручной и промышленный
электроинструмент.
Наша компания предлагает проверенное качеством лучших
производителей электроинструмента - Makita, Bosch, Hitachi. В
каталоге этого интернет-магазина найдете так же аккумуляторы и
запчасти для электроинструмента
Современный электроинструмент позволяет выполнять ремонтные
работы быстро и легко, без дополнительных физических нагрузок.
Каждый гараж сегодня оснащен электроинструментов для выполнения
тяжелых задач.
```

Рис. 6.16 – lstm_size=1024, 1800 итераций.

При данных параметрах нейронная сеть сгенерировала текст с результатами:

Количество символов:	645
Количество символов без пробелов:	571
Количество слов:	72
Количество уникальных слов:	43
Количество значимых слов:	20
Количество стоп-слов:	30
Вода:	65.9%
Количество грамматических ошибок:	2%
Тошнота текста:	1.8

Текст превысил допустимое значение во всех параметрах.

Кроме того, текст потерял связность, имеет имеет смысловые и грамматические ошибки.

Средние результаты для пятидесяти текстов с такими же заданными параметрами получились следующие:

Количество уникальных слов:	55%
Количество значимых слов:	32%
Количество стоп-слов:	28%
Вода:	64 %
Количество грамматических ошибок:	6%
Тошнота текста:	1.8

На рис.6.17 представлены сравнения полученных результатов.

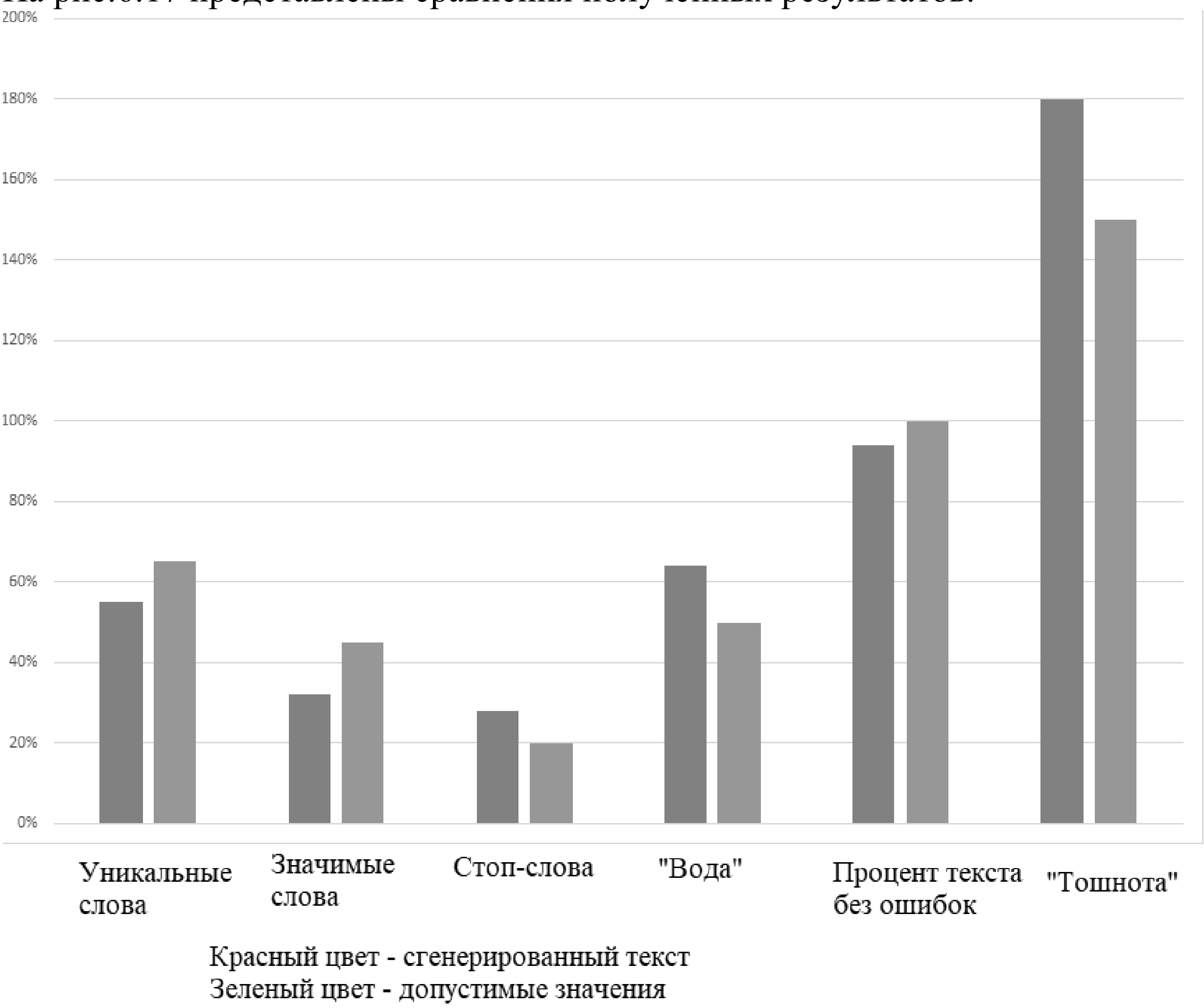

Рис. 6.17 – Сравнение полученных результатов при lstm_size=1024

На основе проведенного исследования, оптимальными значениями нейронной сети для генерации SEO-текстов можно считать:

- batch_size = 100;
- num_steps = 100;
- lstm_size = 512;
- num_layers = 2;
- learning_rate = 0.001.

Однако и при них данная нейронная сеть не может генерировать полностью готовый для вставки на сайт текст. Тем не менее, тексты, созданные ею, годятся для использования в виде наброска, который в дальнейшем, при редактировании, можно использовать для вставки на сайт.

Дальнейшее ухудшение результатов при увеличении показателей свыше lstm_size=512 можно объяснить так называемым «переобучением» сети, когда лишние связи между слоями лишь ухудшают общую логику работы сети.

6.1.2 Реализация нейросети.

Как было указано в п. 5.1.2, работа с нейронной сетью осуществляется посредством трех основных классов:

1. NeuralNetwork

   В данном классе производится описание нейросети. В нем происходит инициализация и построение LSTM-ячеек.

1.1 __init

   Конструктор класса, отвечающий за создание экземпляра нейросети.

   Листинг класса NeuroNetwork представлен на рис. 6.18;

```
lass NeuralNetwork:

    def __init__(self, num_classes, batch_size=100, num_steps=100,
                       lstm_size=512, num_layers=2, learning_rate=0.001,
                       grad_clip=5, sampling=False):

        if sampling == True:
            batch_size, num_steps = 1, 1
        else:
            batch_size, num_steps = batch_size, num_steps

        tf.reset_default_graph()

        self.inputs, self.targets, self.keep_prob = build_inputs(batch_size, num_steps)

        cell, self.initial_state = build_lstm(lstm_size, num_layers, batch_size, self.keep_prob)


        x_one_hot = tf.one_hot(self.inputs, num_classes)

        outputs, state = tf.nn.dynamic_rnn(cell, x_one_hot, initial_state=self.initial_state)
        self.final_state = state

        self.prediction, self.logits = build_output(outputs, lstm_size, num_classes)

        self.loss = build_loss(self.logits, self.targets, lstm_size, num_classes)
        self.optimizer = build_optimizer(self.loss, learning_rate, grad_clip)
```

Рис. 6.18 – Класс NeuralNetwork

2. TrainingData

В данном классе происходит обучение модели.

2.1. _ tf.Session()

Открытая сессия (метод), с помощью которого происходит сохранение статуса LSTM-ячейки, который нейронная сеть передает на вход следующей ячейки, таким образом обеспечивая преемственность. Кроме того, в нем происходит сохранение состояния модели нейронной сети через каждые 20 циклов обучения.

Листинг класса TrainingData представлен на рис. 6.19;

```
lass TrainingData:
pochs = 20
ave_every_n = 300

odel = NeuralNetwork(len(vocab), batch_size=batch_size, num_steps=num_steps,
                     lstm_size=lstm_size, num_layers=num_layers,
                     learning_rate=learning_rate)

aver = tf.train.Saver(max_to_keep=100)
ith tf.Session() as sess:
   sess.run(tf.global_variables_initializer())


   counter = 0
   for e in range(epochs):
       new_state = sess.run(model.initial_state)
       loss = 0
       for x, y in get_batches(encoded, batch_size, num_steps):
           counter += 1
           start = time.time()
           feed = {model.inputs: x,
                   model.targets: y,
                   model.keep_prob: keep_prob,
                   model.initial_state: new_state}
           batch_loss, new_state, _ = sess.run([model.loss,
                                                model.final_state,
                                                model.optimizer],
                                                feed_dict=feed)

           end = time.time()
           print('Epoch: {}/{}... '.format(e+1, epochs),
                 'Training Step: {}... '.format(counter),
                 'Training loss: {:.4f}... '.format(batch_loss),
                 '{:.4f} sec/batch'.format((end-start)))

           if (counter % save_every_n == 0):
               saver.save(sess, "checkpoints/i{}_l{}.ckpt".format(counter, lstm_size))

   saver.save(sess, "checkpoints/i{}_l{}.ckpt".format(counter, lstm_size))
```

Рис. 6.19 – Класс TrainingData

3. TextGeneration

Данный класс отвечает за генерацию текста.

3.1. def pick_top_n(preds, vocab_size, top_n):

Метод, необходимый для уменьшения шума предсказаний, top_n отвечает за количество символов, из которых нейронная сеть будет выбирать при генерации, отбрасывая остальные варианты:

Листинг метода def pick_top_n представлен на рис. 6.20;

```
ef pick_top_n(preds, vocab_size, top_n):
   p = np.squeeze(preds)
   p[np.argsort(p)[:-top_n]] = 0
   p = p / np.sum(p)
   c = np.random.choice(vocab_size, 1, p=p)[0
   return c
```

Рис. 6.20 – Метод def pick_top_n

3.2. def sample(checkpoint, n_samples, lstm_size, vocab_size, prime):

Метод, в котором задаются ключевые параметры, а также происходит непосредственная генерация текста.

Листинг метода def sample представлен на рис. 6.21;

```
def sample(checkpoint, n_samples, lstm_size, vocab_size, prime):
    samples = [c for c in prime]
    model = NeuralNetwork(len(vocab), lstm_size=lstm_size, sampling=True)
    saver = tf.train.Saver()
    with tf.Session() as sess:
        saver.restore(sess, checkpoint)
        new_state = sess.run(model.initial_state)
        for c in prime:
            x = np.zeros((1, 1))
            x[0,0] = vocab_to_int[c]
            feed = {model.inputs: x,
                    model.keep_prob: 1.,
                    model.initial_state: new_state}
            preds, new_state = sess.run([model.prediction, model.final_state],
                                         feed_dict=feed)

        c = pick_top_n(preds, len(vocab))
        samples.append(int_to_vocab[c])

        for i in range(n_samples):
            x[0,0] = c
            feed = {model.inputs: x,
                    model.keep_prob: 1.,
                    model.initial_state: new_state}
            preds, new_state = sess.run([model.prediction, model.final_state],
                                         feed_dict=feed)

            c = pick_top_n(preds, len(vocab))
            samples.append(int_to_vocab[c])

    return ''.join(samples)
```

Рис. 6.21 – Метод def sample

3.3. def get_batches(arr, n_seqs, n_steps):

Метод, отвечающий за генерацию предсказаний следующего символа.

Листинг метода def get_batches представлен на рис. 6.22;

```
def get_batches(arr, n_seqs, n_steps):
    characters_per_batch = n_seqs * n_steps
    n_batches = len(arr)//characters_per_batch

    arr = arr[:n_batches * characters_per_batch]
    arr = arr.reshape((n_seqs, -1))

    for n in range(0, arr.shape[1], n_steps):
        x = arr[:, n:n+n_steps]
        y = np.zeros_like(x)
        y[:, :-1], y[:, -1] = x[:, 1:], x[:, 0]
        yield x, y
```

Рис. 6.22 – Метод def get_batches

3.4. def build_inputs(batch_size, num_steps):

Метод, определяющий входные параметры.

Листинг метода def build_inputs представлен на рис. 6.23;

```
def build_inputs(batch_size, num_steps):
    inputs = tf.placeholder(tf.int32, [batch_size, num_steps], name='inputs')
    targets = tf.placeholder(tf.int32, [batch_size, num_steps], name='targets')

    keep_prob = tf.placeholder(tf.float32, name='keep_prob')

    return inputs, targets, keep_prob
```

Рис. 6.23 – Метод def build_inputs

3.5. def build_lstm(lstm_size, num_layers, batch_size, keep_prob):

Метод, в котором происходит построение LSTM-ячейки.

Листинг метода def build_lstm представлен на рис. 6.24;

```
ef build_lstm(lstm_size, num_layers, batch_size, keep_prob):
    def build_cell(lstm_size, keep_prob):
        lstm = tf.contrib.rnn.BasicLSTMCell(lstm_size)
        drop = tf.contrib.rnn.DropoutWrapper(lstm, output_keep_prob=keep_prob)
        return drop

    cell = tf.contrib.rnn.MultiRNNCell([build_cell(lstm_size, keep_prob) for _ in range(num_layers)])
    initial_state = cell.zero_state(batch_size, tf.float32)
```

Рис. 6.24 – Метод def build_lstm

3.6. def build_output(lstm_output, in_size, out_size):

Метод, в котором происходит построение выходного слоя.

Листинг метода def build_output представлен на рис. 6.25;

```
ef build_output(lstm_output, in_size, out_size):
    seq_output = tf.concat(lstm_output, axis=1)
    x = tf.reshape(seq_output, [-1, in_size])

    with tf.variable_scope('softmax'):
        softmax_w = tf.Variable(tf.truncated_normal((in_size, out_size), stddev=0.1))
        softmax_b = tf.Variable(tf.zeros(out_size))

    logits = tf.matmul(x, softmax_w) + softmax_b
    out = tf.nn.softmax(logits, name='predictions')
    return out, logits
```

Рис. 6.25 – Метод def build_output

3.7. def build_loss(logits, targets, lstm_size, num_classes):

Метод, в котором происходит расчет функции потери. Она представляет из себя меру несогласия наблюдаемых и предсказываемых данных.

Листинг метода def build_loss представлен на рис. 6.26;

```
ef build_loss(logits, targets, lstm_size, num_classes):
    y_one_hot = tf.one_hot(targets, num_classes)
    y_reshaped = tf.reshape(y_one_hot, logits.get_shape())
    loss = tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=y_reshaped)
    loss = tf.reduce_mean(loss)
    return loss
```

Рис. 6.26 – Метод def build_loss

3.8. def build_optimizer(loss, learning_rate, grad_clip):

Метод-оптимизатор работы нейросети. Необходим для проведения стохастического градиентного спуска. С помощью аппроксимации значительно уменьшается значение ошибки при предсказании.

Листинг метода def build_optimizer представлен на рис. 6.27;

```
ef build_optimizer(loss, learning_rate, grad_clip):
    tvars = tf.trainable_variables()
    grads, _ = tf.clip_by_global_norm(tf.gradients(loss, tvars), grad_clip)
    train_op = tf.train.AdamOptimizer(learning_rate)
    optimizer = train_op.apply_gradients(zip(grads, tvars))

    return optimizer
```

Рис. 6.27 – Метод def build_optimizer

6.2 Реализация базы данных

Исходный код скрипта для создания базы данных в MySQL представлен на рис. 6.28:

```
REATE TABLE `Analyze_settings` (
   `settings_id` int(11) NOT NULL AUTO_INCREMENT,
   `user_id` int(11) NOT NULL AUTO_INCREMENT,
   `keyword_density_check` tinyint(1) NOT NULL AUTO_INCREMENT,
   `frequency_check` tinyint(1) NOT NULL AUTO_INCREMENT,
   `stopwords_numbers_check` tinyint(1) NOT NULL AUTO_INCREMENT,
   `text_amount_check` tinyint(1) NOT NULL AUTO_INCREMENT,
   `words_numbers_check` tinyint(1) NOT NULL AUTO_INCREMENT,
   `water_check` tinyint(1) NOT NULL AUTO_INCREMENT,
   `nausea_check` tinyint(1) NOT NULL AUTO_INCREMENT,
   `grammatic_errors_check` tinyint(1) NOT NULL AUTO_INCREMENT,
   PRIMARY KEY (`settings_id`)
;

REATE TABLE `User` (
   `user_id` int(11) NOT NULL AUTO_INCREMENT,
   `login` varchar(500) NOT NULL,
   `password` varchar(500) NOT NULL,
   `name` varchar(500) NOT NULL,
   `surname` varchar(500) NOT NULL,
   PRIMARY KEY (`user_id`)
;

REATE TABLE `Themes` (
   `theme_id` int(11) NOT NULL AUTO_INCREMENT,
   `category_id` int(11) NOT NULL AUTO_INCREMENT,
   `theme_name` varchar(500) NOT NULL AUTO_INCREMENT,
   PRIMARY KEY (`theme_id`)
;

REATE TABLE `Category` (
   `category_id` int(11) NOT NULL AUTO_INCREMENT,
   `category_name` varchar(500) NOT NULL AUTO_INCREMENT,
   PRIMARY KEY (`category_id`)
;

REATE TABLE `Pre-Settings` (
   `presettings_id` int(11) NOT NULL AUTO_INCREMENT,
   `main_words` varchar(500) NOT NULL,
   `text_length` int(11) NOT NULL,
   `page_id` int(11) NOT NULL,
   `position` varchar(50) NOT NULL,
   `insert_after_editing` tinyint(1) NOT NULL,
   `user_id` int(11) NOT NULL,
   PRIMARY KEY (`presettings_id`)
;

REATE TABLE `Texts` (
   `result_id` int(11) NOT NULL AUTO_INCREMENT,
   `text` varchar(50000) NOT NULL,
   `amended_text` varchar(50000) NOT NULL,
   `generation_time` DATETIME NOT NULL,
   `last_change_time` DATETIME NOT NULL,
   `page_code` varchar(50000) NOT NULL,
   `user_id` int(11) NOT NULL,
   `theme_id` int(11) NOT NULL,
   PRIMARY KEY (`result_id`)
;

.TER TABLE `Analyze_settings` ADD CONSTRAINT `Analyze_settings_fk0` FOREIGN KEY (`user_id`) REFERENCES `User`(`user_id`);

.TER TABLE `Themes` ADD CONSTRAINT `Themes_fk0` FOREIGN KEY (`category_id`) REFERENCES `Category`(`category_id`);

.TER TABLE `Pre-Settings` ADD CONSTRAINT `Pre-Settings_fk0` FOREIGN KEY (`user_id`) REFERENCES `User`(`user_id`);

.TER TABLE `Texts` ADD CONSTRAINT `Texts_fk0` FOREIGN KEY (`user_id`) REFERENCES `User`(`user_id`);

.TER TABLE `Texts` ADD CONSTRAINT `Texts_fk1` FOREIGN KEY (`theme_id`) REFERENCES `Themes`(`theme_id`);
```

Рис. 6.28 – Скрипт для создания базы данных

6.3 Реализация анализатора

Анализатор проводит анализ полученного текста на необходимые метрики. Блок-схема общего алгоритма работы данной части системы представлена на рис. 6.29:

Рис. 6.29 – Общий алгоритм работы анализатора.

В начале работы анализатору передается список метрик, которые он должен проверить с помощью API от сторонних сервисов либо с помощью собственных методов.

Затем с помощью заданных методов анализатор проверяет текст на соответствие каждой из метрик, возвращая полученные результаты в плагин.

Успешность текста выдается на основе соответствия необходимых метрик в тексте требуемым значениям. После этого данные выводятся пользователю плагина.

Анализатор состоит из нескольких классов, которые отвечают за расчёт значений отдельных групп метрик. Данный класс WaterMetrix обращается к API от text.ru

1. WaterMetrix

   Класс, отвечающий за подсчет общего количества «воды» в тексте. Метод данного класса:

1.1 check

   Метод, считающий общее количество воды в тексте. В начале данной функции создается переменная $text, в которую загружается редактированный текст из плагина. С помощью метода check(), происходит обращение к стороннему API от сервиса text.ru, после чего в переменную $answer записывается результат проверки текста на соответствие данной метрике.

   Листинг класса WaterMetrix представлен на рис. 6.30;

```
class WaterMetrix
{
    private $apiKey;
    private $serviceUrl;

    function __construct($apiKey, $serviceUrl)
    {
        $this->apiKey = $apiKey;
        $this->serviceUrl = $serviceUrl;
    }

    function check($text) {
        $text = strip_tags($text);
        $text = htmlspecialchars_decode($text);

        $textData = [
            'text' => $text,
            'key' => $this->apiKey,
        ];

        $textData = json_encode($textData, JSON_UNSCAPED_UNICODE);

        $ch = curl_init($this->serviceUrl);
        curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
        curl_setopt($ch, CURLOPT_POSTFIELDS, $textData);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($ch, CURLOPT_HTTPHEADER, [
                'Content-Type: application/json',
                'Content-Length: ' . strlen($textData)
            ]
        );
        $answer = curl_exec($ch);

        $answer = json_decode($answer);

        if (isset($answer->error)) {
            return false;
        }

        return $answer->waternumber;
    }
}
```

Рис. 6.30 – Листинг класса WaterMetrix

2. CalcTextSymbolsMetrix

Класс, отвечающий за подсчет общего количества символов, а также количества символов без пробелов в тексте. Методы данного класса:

2.1 getAllSymbols

Метод, считающий общее количество символов в тексте. В начале данной функции создается переменная $text, в которую загружается редактированный текст из плагина. С помощью метода mb_strlen(), происходит подсчет общего количества символов.

2.2 getSymbolsExeptSpaces

Метод, считающий количество символов без пробелов в тексте. В начале данной функции создается переменная $text, в которую загружается редактированный текст из плагина. Далее, происходит преобразование переменной и отсеивания из нее лишних символов.

3. Листинг класса CalcTextSymbolsMetrix представлен на рис. 6.31;

```
class CalcTextSymbolsMetrix
{

    public static function getCalcAllSymbols($text)
    {
        $text = iconv(mb_detect_encoding($text, mb_detect_order(), true), "UTF-8", $text);
        $result = mb_strlen($text);

        if (is_int($result)) {
            return $result;
        }

        return false;
    }

    public static function getCalcSymbolsExeptSpaces($text)
    {
        $text = iconv(mb_detect_encoding($text, mb_detect_order(), true), "UTF-8", $text);
        $text = preg_replace('!\s+!', ' ', $text);
        $text = str_replace("\t", ' ', $text);
        $text = explode(' ', $text);

        $result = count($text);

        if (is_int($result)) {
            return $result;
        }

        return false;
    }
}
```

Рис. 6.31 – Листинг класса CalcTextSymbols

Остальные методы анализатора выполнены схожим образом и описываться не будут.

Выводы по разделу шесть.

Были проведены исследования и рассмотрены возможности генерации SEO-текстов с помощью нейронной сети. В ходе исследования, были подобраны оптимальные параметры для работы нейронной сети и генерации текста.

7. ПРИМЕРЫ РАБОТЫ

7.1 Запуск обучения

Обучение нейронной сети происходит в фоновом режиме. За загрузку обучающей выборки в нейросеть отвечает консольный скрипт, запуск которого настроен с помощью серверной утилиты crontab. Скрипт можно запустить также и вручную, указав в качестве входного параметра путь к файлу с обучающей выборкой. На рис. 7.1 показан процесс вызова скрипта через SSH-клиент PUTTY.

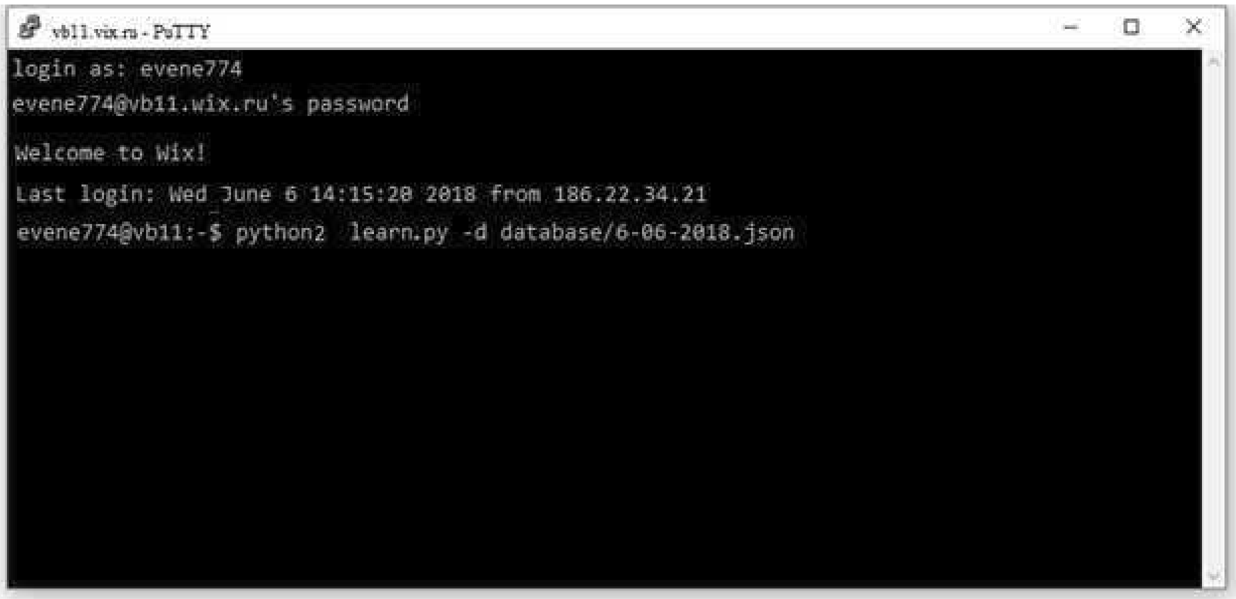

Рис. 7.1 – Вызов скрипта

7.2 Примеры работы плагина

При тестировании работоспособности разрабатываемого программного обеспечения в плагине было сгенерировано несколько текстов разной тематики с разным объемом слов. В дальнейшем каждый текст был проверен анализатором, успешно отредактирован и вставлен на сайт с помощью плагина. Ниже приведены результаты работы программного обеспечения для заданных значений настроек пре-генерации:

1. Ключевые слова: гаражное оборудование

   Тематика: автомобили

   Длина текста: 900

   Результаты генерации представлены на рис. 7.4

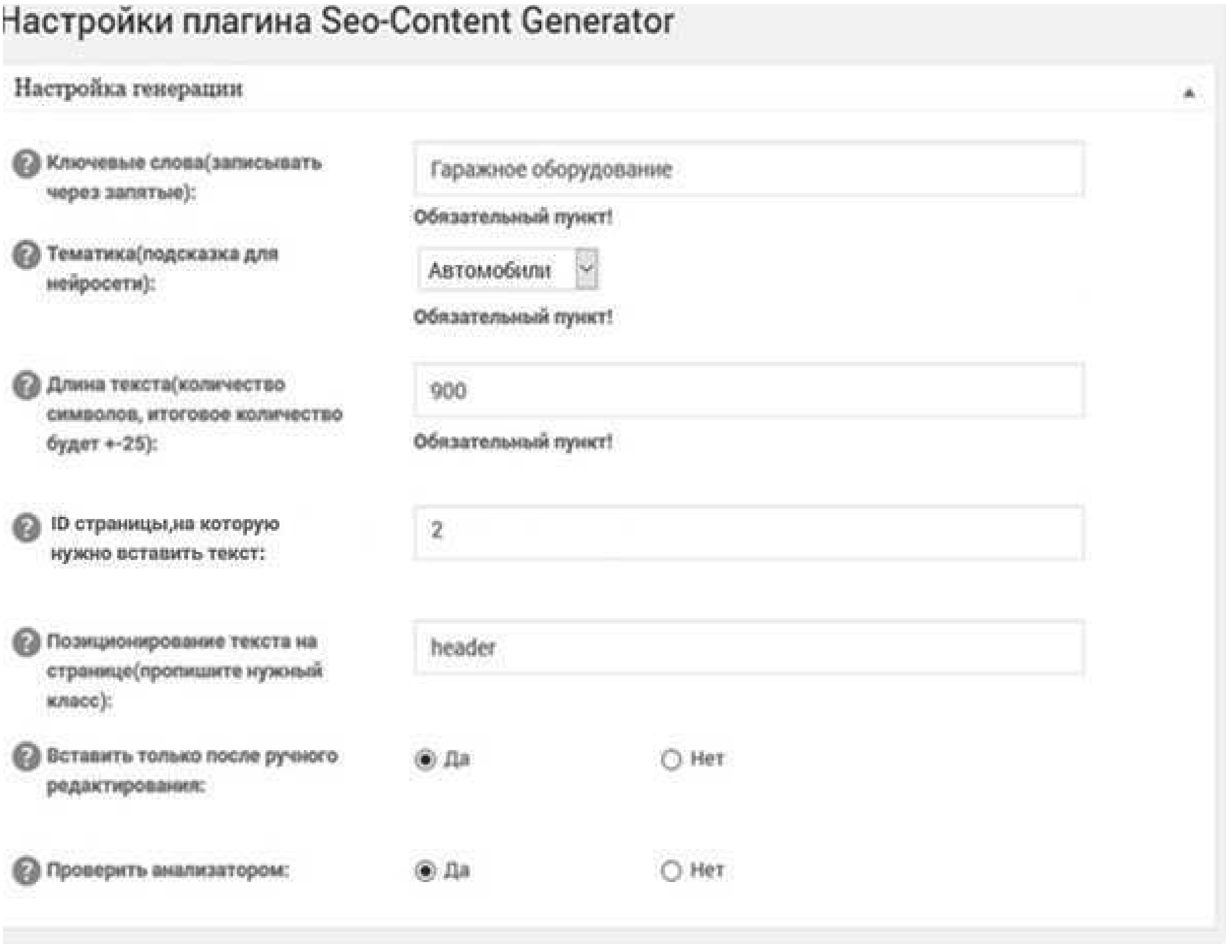

Рис. 7.2 – Форма с настройками генерации текста.

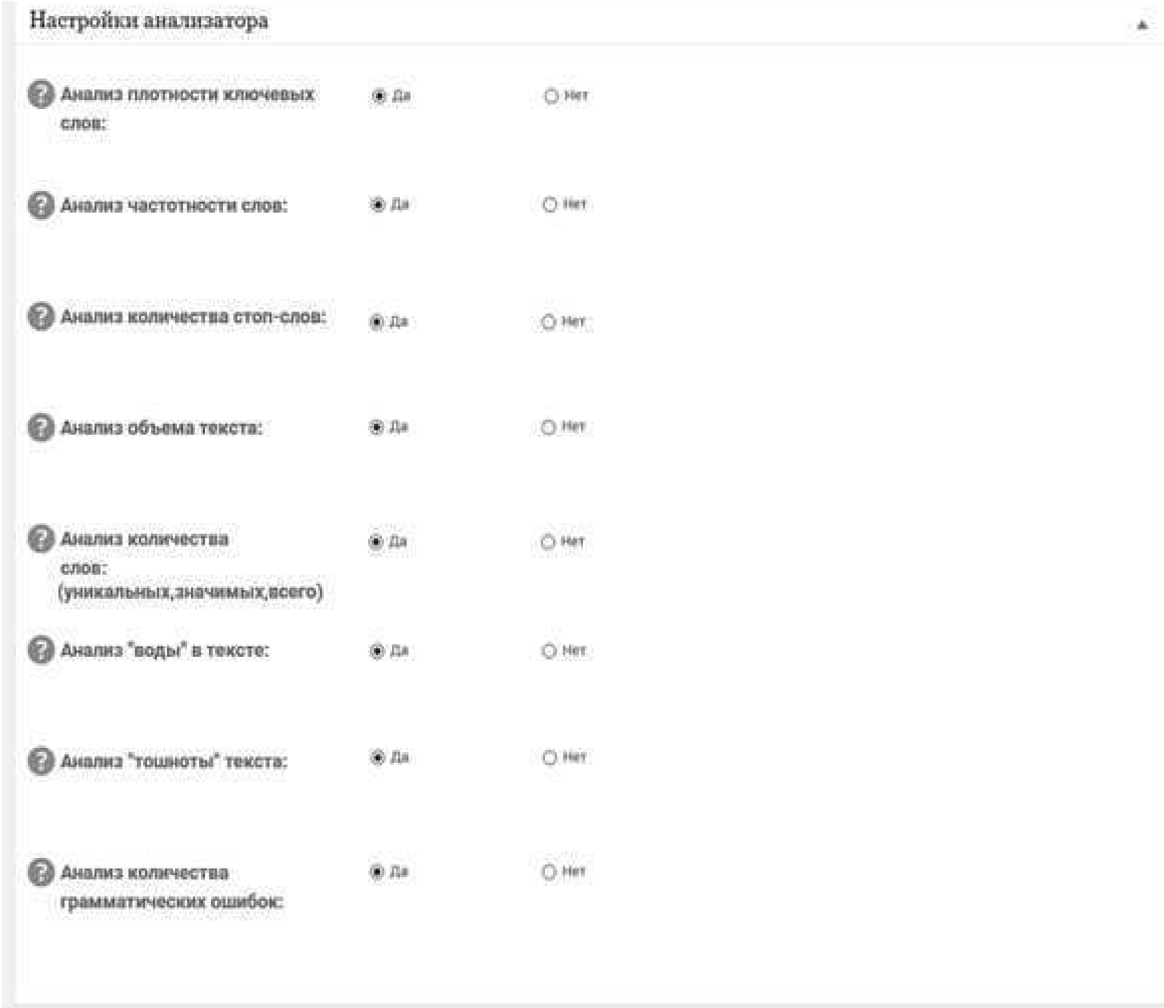

Рис. 7.3 – Форма с настройками анализатора.

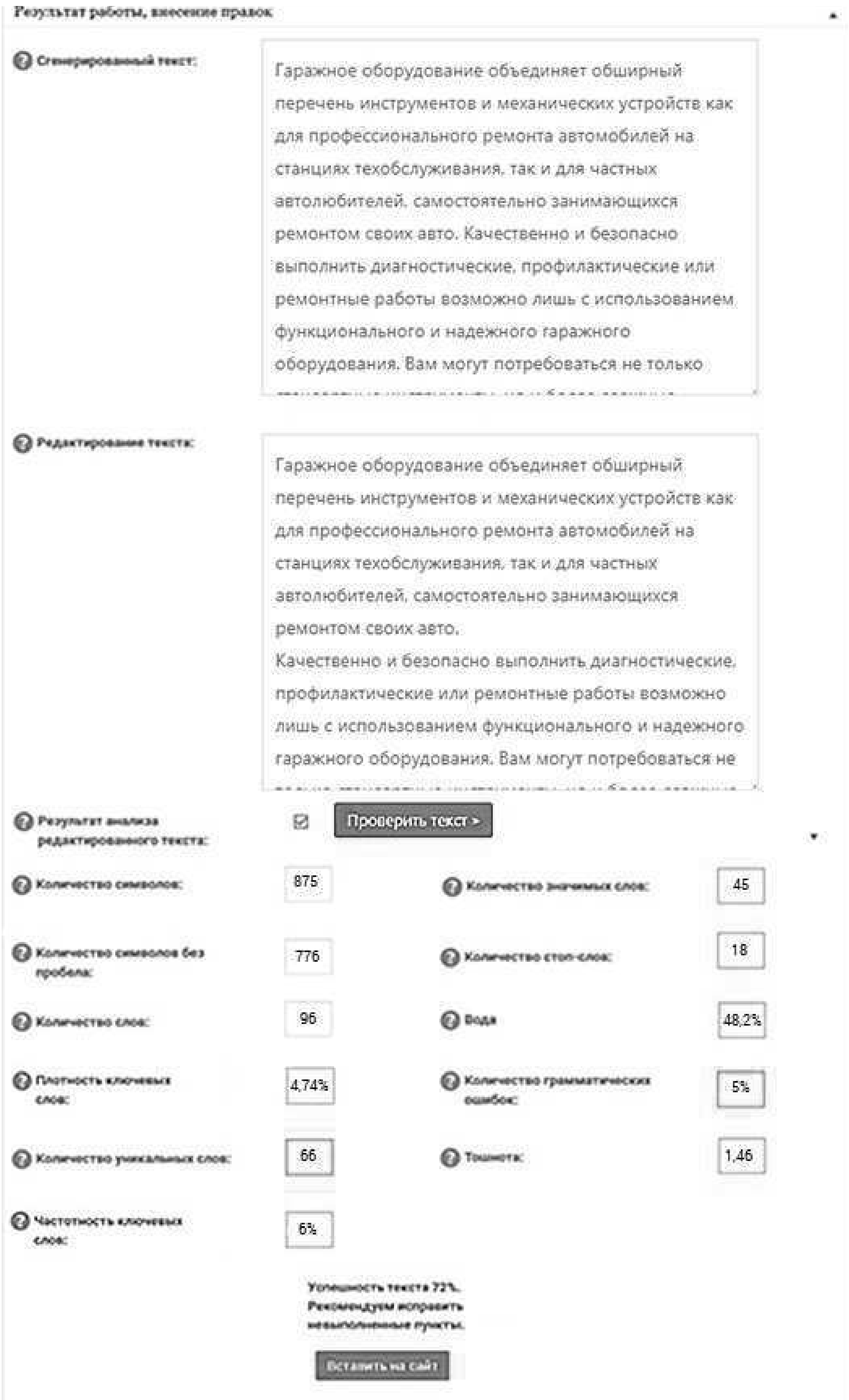

Рис. 7.4 – Результат генерации и форма для работы с текстом.

2. Ключевые слова: спорт и отдых

Тематика: лазерное оборудование

Длина текста: 700

Результаты работы генерации представлены на рис.7.5

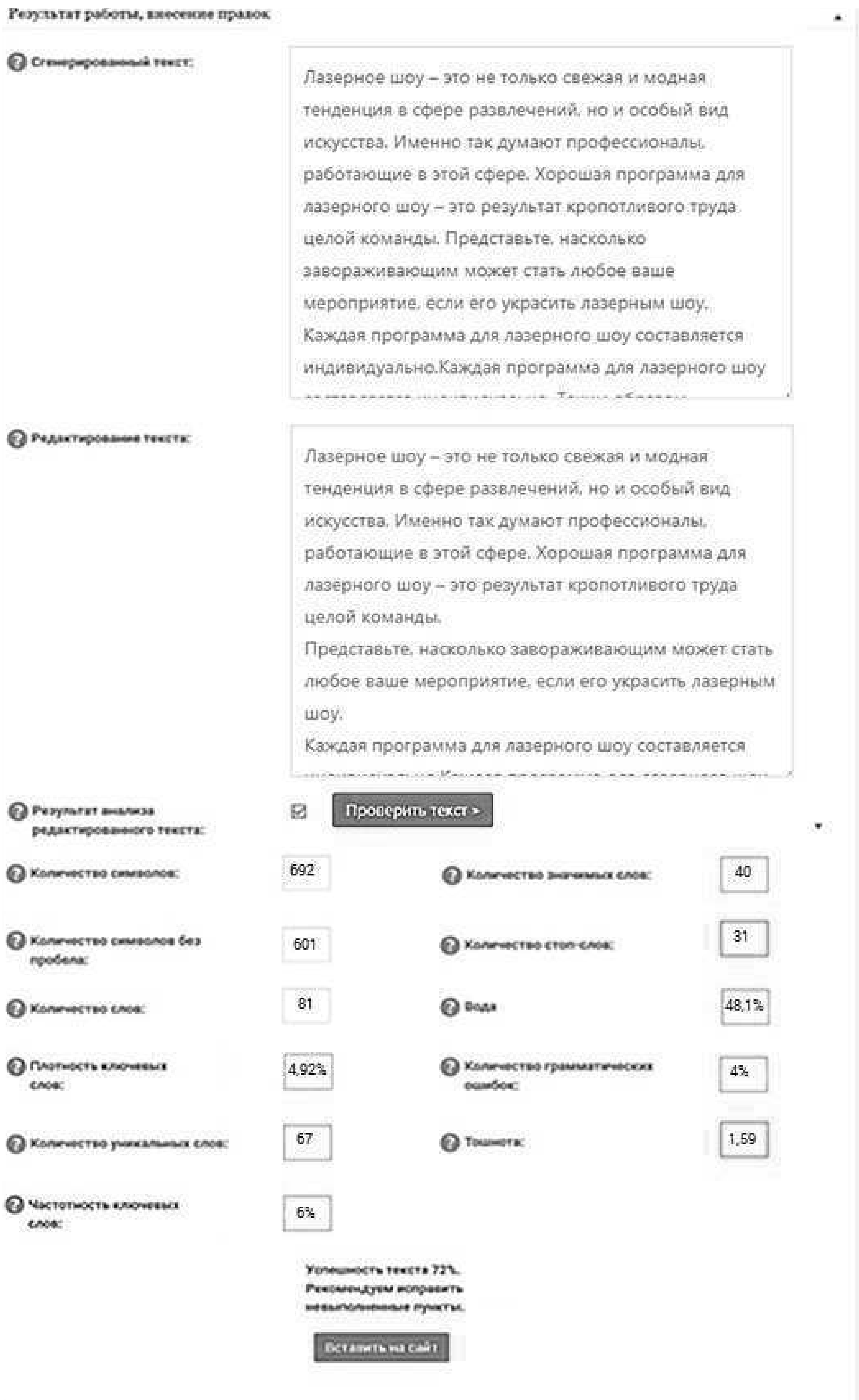

Рис. 7.5 – Результат генерации и форма для работы с текстом.

7.3 Инструкция по использованию

Согласно сформулированным в п.4 требованиям, система доступна для использования по сети Internet. Взаимодействие с сервером, на котором находится уже обученная нейронная сеть, производится посредством Wordpress-плагина, который можно установить и открыть в любом современном браузере.

Для генерации SEO-текста и его вставки на страницу необходимо:

1. Скачать плагин из магазина

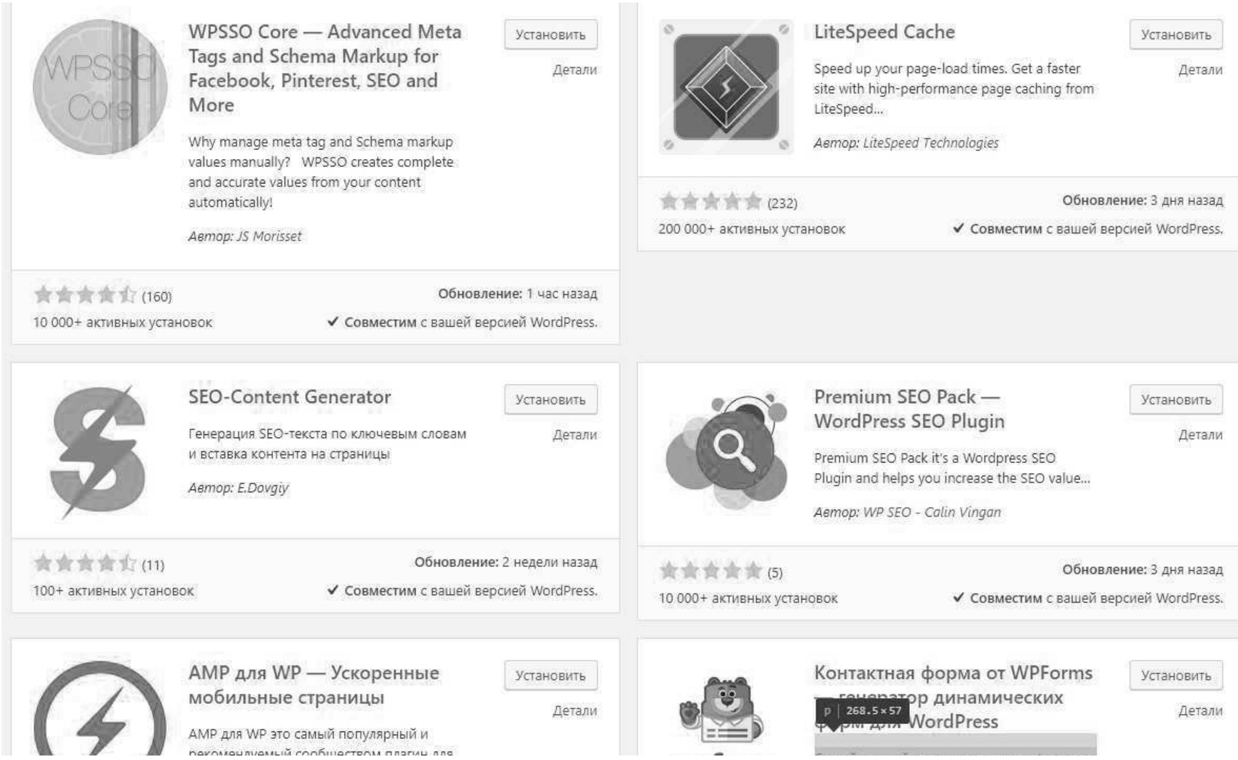

Рис. 7.7 – Расположение плагина в магазине

2. Перейти к форме с настройками генерации текста.
3. Задать необходимые параметры генерации.
4. Перейти к форме с настройками анализатора.
5. Отметить необходимые для проверки метрики.
6. Перейти к форме для работы с текстом.
7. Отредактировать текст и вставить его на желаемую страницу.

Выводы по разделу семь

В данной главе были рассмотрены примеры работы плагина при генерации текстов различного размера и тематик. На основе этого раздела можно сделать вывод о том, что система работоспособна и удовлетворяет требованиям, сформулированным в разделе 4.

ЗАКЛЮЧЕНИЕ

В ходе данной работы, были проведены исследования и рассмотрены возможности генерации SEO-текстов с помощью LSTM нейронной сети, построенной на фреймворке Tensorflow и написанной на языке программирования Python. В ходе исследования, были подобраны оптимальные параметры для работы нейронной сети и генерации текста. Тем не менее, даже при этих параметрах, нейронная сеть не смогла сгенерировать тексты, которые бы удовлетворяли всем необходимым метрикам. Тем не менее, сгенерированный текст можно использовать как набросок при написании SEO-текста, однако полностью готовый для вставки на сайт SEO-текст данная нейронная сеть выдать не способна.

Кроме того, был разработан Wordpress-плагин, генерирующий SEO-текст, на основе заданных ключевых слов и тематики. Плагин может быть использован владельцами сайтов в целях их быстрого и бесплатного наполнения контентом.

В ходе выполнения работы решены следующие задачи:

1. Проанализирован ряд программных продуктов для генерации SEO-текста;
2. Сформулированы требования к создаваемому программному обеспечению;
3. Спроектирована структура приложения;
4. Проведены работы по реализации программного продукта:
   - 4.1 Спроектирована база данных;
   - 4.2 Подключена и настроена нейронная сеть;
   - 4.3 Проведено исследование возможностей генерации SEO-текстов нейронной сетью, подобраны оптимальные параметры;
   - 4.4 Написан код для серверной и клиентской частей системы;
   - 4.5 Сверстан пользовательский интерфейс;
5. Проведено тестирование приложения.

В настоящее время плагин уже доступен для скачивания из Wordpress-магазина. На основе отзывов пользователей будут составляться дальнейшие планы по развитию проекта и внедрения в него различных компонентов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. http://www.blog.upsale.ru/seo/oshybki_seo_tekstov.html
2. https://www.itrack.ru/research/cmsrate/
3. http://www.linksfarm.ru/
4. http://seogenerator.ru/
5. http://myseogid.ru/tools/97-generating-the-web.html
6. PHP. Объекты, шаблоны и методики программирования / М. Зандстра – Москва: Изд. Вильямс, 2016. – 560 с.
7. Изучаем Python / М. Лутц – Москва: Изд. Вильямс, 2015. – 1280 с.
8. Wordpress для профессионалов / Т. Хассей – Москва: Изд. Эксмо, 2014. – 545 с.
9. MySQL. Оптимизация производительности / Б. Шварц – Москва: Изд. Символ-Плюс, 2014. – 1264 с.
10. Глубокое обучение. Погружение в мир нейронных сетей / С. Николенко – Москва: Изд. Питер, 2018. – 480 с.