Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки Прикладная математика и информатика

РАБОТА ПРОВЕРЕНА
Рецензент,

_____
«____»_____2018г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н., доцент
_____А.А.Замышляева
«____»_____2018 г.

Разработка приложения для автоматизации тестирования сайтов
ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–01.03.02.2018.40.ПЗ ВКР

Руководитель работы, ассистент
_____/ В.А. Сурин
«____»_____2018 г.
Автор работы
студент группы ЕТ-412
_____/А.Р. Абдрашитов
«____»_____2018 г.
Нормоконтролер, к.э.н., доцент
_____/Д.А. Дрозин
«____»_____2018 г.

Челябинск 2018

АННОТАЦИЯ

Абдрашитов А.Р. Разработка приложения для автоматизации тестирования сайтов. –Челябинск: ЮУрГУ, ЕТ-412, 43 с., 37 ил., библиогр. список – 19 наим., 1 прил.

Целью данной работы является реализация приложения для автоматизации процедуры тестирования сайта.

В первом разделе был проведен анализ предметной области, сделан обзор методов, применяющихся для тестирования сайтов, а также рассмотрены уже реализованные решения. Приводится обоснование выбранного метода, способа и средств его реализации.

Во втором разделе описана математическая модель сверточной нейронной сети, применяющейся в данной работе для обнаружения рекламы, которая может присутствовать на сайте.

В третьем разделе описывается архитектура разрабатываемой нейронной сети, позволяющей обнаружить рекламу.

В четвертом разделе описан пользовательский интерфейс приложения, предназначенного для автоматизации тестирования сайта, а также описана работа нейронной сети, улучшающей работу реализованного приложения путем обнаружения рекламы на сайте. Была проведена апробация разработанного приложения на тестовых данных.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире технологий ежедневно появляется большое количество программ, решающих различные задачи и выполняющих разнообразные функции. Таким образом, пользователю предоставляется широкий выбор программ, позволяющих достичь нужную ему цель. Вполне очевидно, что программа, обладающая лучшими показателями качества по сравнению с аналогами, будет более востребованной среди заинтересованных пользователей. В связи с этим для разработчиков были установлены определенные стандарты качества ПО. Для проверки соответствия данным стандартам проводится специальная процедура, называемая тестирование программного обеспечения.

Тестирование ПО – это процедура, при которой осуществляется проверка соответствия между реальным и ожидаемым поведением ПО. Данная процедура выполняется на конечном наборе данных, которые подбираются определенным образом.

Сама по себе тема тестирования ПО относительно нова и связана с бурным развитием систем автоматизированной разработки ПО и технологий в 90-е года в США. Среди производителей ПО интерес к данной теме ежедневно возрастает, так как усиливается конкуренция, следовательно, необходимо уделять больше внимания качеству продукции, сохраняя баланс с приемлемой ценой. Кроме того, вопрос о качестве имеет особую ценность, ведь сегодня пользователю требуется не только комфортная работа с той или иной программой, но и надежность, благодаря которой возможна автоматизированное управление оборудованием в больницах, диспетчерскими системами, воздушными суднами, атомными реакторами и т. д.

На сегодняшний день, обеспечение высокого качества разрабатываемого ПО – это первоочередная задача большинства компаний, так как это позволяет «обойти» конкурентов и заслужить доверие клиентов. Именно поэтому любая компания по разработке ПО прежде чем выпустить свою продукцию на рынок, осуществляет ее тестирование, направленное на обнаружение и устранение как можно большего числа ошибок. При этом компании могут иметь собственные отделы тестирования или осуществлять данную процедуру, обращаясь к услугам сторонних организаций.

Цели тестирования заключаются в следующем:

- повышение вероятности правильной работы приложения, предназначенного для тестирования
- повышение вероятности соответствия приложения, предназначенного для тестирования, необходимым требованиям;
- предоставление актуальной информации о текущем состоянии продукта.

Таким образом, целью данной работы является реализация приложения для автоматизации процедуры тестирования сайта. Для достижения данной цели необходимо решить следующие задачи:

- выполнить обзор методов тестирования и существующих решений;

- выбрать наиболее подходящий метод тестирования;
- разработать математическую модель, необходимую для реализации выбранного метода;
- выполнить программную реализацию;
- осуществить проверку работу программы, проверив сайт на наличие некорректных ссылок

1 МЕТОДЫ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ПРИМЕРЫ РЕШЕНИЙ В ОБЛАСТИ ТЕСТИРОВАНИЯ САЙТОВ

1.1 Виды тестирования

Все виды тестирования подразделяются на 3 основные группы:

- функциональные виды тестирования;
- нефункциональные виды тестирования;
- виды тестирования, связанные с изменениями.

1.1.1 Функциональные виды тестирования

Функциональное тестирование

В результате данного тестирования делается вывод о надлежащем функционировании объекта, который подлежит тестированию. При этом для данного вида тестирования характерны следующие особенности:

Все функциональные требования транслируются в тест-кейсы, которые позволяют проверить, что система функционирует также, как описано в спецификации. При этом применяются техники «черного ящика», когда стратегии тестирования касаются только внешнего мира и не затрагивается внутреннее устройство тестируемого объекта.

Осуществляется проверка на то, все ли функциональные требования имеют программную реализацию.

Тестирование безопасности

В результате тестирования безопасности делается оценка уязвимости программного обеспечения к различным атакам. Компьютерные системы довольно часто подвергаются различным хакерским атакам. Под термином хакерская атака принято понимать совокупность действий, связанных с несанкционированным проникновением в систему в целях дальнейшей манипуляции ее работы. Тестирование безопасности позволяет проверить то, как фактически реагирует на проникновение система, используя определенные защитные механизмы. В ходе данного вида тестирования испытателю предлагается сыграть роль хакера. При этом он может выполнять различные действия:

- попытки узнать пароль, используя внешние средства;
- атаки системы с помощью специальных программ, анализирующих механизмы защиты;
- подавление системы с целью выведения ее из строя и ее отказа обслуживать других клиентов;
- специальное введение ошибок для проникновения в систему в процессе восстановления;
- обзор открытых данных с целью найти ключ для входа в систему.

1.1.2 Нефункциональные виды тестирования

Тестирование производительности

В результате данного вида тестирования определяется, насколько быстро работает система или ее часть под определенной нагрузкой. Кроме того, могут осуществляться процессы по подтверждению других параметров системы, например, масштабируемость, надежность. При попытке достичь некоторых количественных пределов, которые обуславливаются характеристиками тестируемой системы и ее операционного окружения, используется особый подвид таких тестов. Суть тестирование на производительность заключается в следующем:

- показать, что система удовлетворяет установленным критериям производительности;
- выяснить, какая часть системы является причиной низкой производительности;
- определить время реакции на действия, которые осуществляет пользователь, время отклика на запрос и т. д.

Стрессовое тестирование

Стрессовое тестирование применяется для выяснения пределов пропускной способности программы. При этом определяется надежность системы, испытывающей экстремальные или диспропорциональные нагрузки. Также в результате данного вида тестирования выясняется, достаточна ли производительность системы для того, чтобы выдержать нагрузку, превышающую ожидаемый максимум.

Нагрузочное тестирование

Данный вид тестирования осуществляется для оценки поведения системы под заданной ожидаемой нагрузкой. Такой нагрузкой, например, может быть определённое количество пользователей, которые работают с программой одновременно и совершают определенное число операций за заданный интервал времени. При наблюдении за базой данных, сервером, сетью и т.д. нагрузочное тестирование позволяет идентифицировать некоторые узкие места программы.

Тестирование стабильности

Главной целью тестирования стабильности заключается в том, чтобы убедиться, что программа способна выдержать определенную нагрузку в течении достаточно длительного времени. При данном тестировании ведется наблюдение за потреблением памяти, которое осуществляет программа, для выявления потенциальных утечек. Тестирование стабильности позволяется выяснить имеется ли деградация производительности, при которой происходит снижение скорости обработки информации или увеличение времени ответа программы после достаточно длительной работы в сравнении со временем ответа, которое показывала программа в начале теста.

Тестирование удобства использования (проверка эргономичности)

В результате данного вида тестирования выясняется, насколько удобен некоторый искусственный объект (например, веб-страница, интерфейс, устройство) при его применении. При проверке эргономичности измеряется

эргономичность системы, при этом данная проверка сосредоточена на определенном объекте. В то же время исследования взаимодействия человека с компьютером формулируются общие универсальные принципы.

При тестировании удобства использования оценивается удобство продукта в процессе его применения, при этом в качестве тестировщиков выступают сами пользователи. На основе полученных результатов делаются соответствующие выводы.

При тестировании программы пользователю предлагают решить основные задачи, которые решает данная программа, в «лабораторных» условиях. В процессе тестирования пользователь может высказать все свои замечания. Данный процесс фиксируется в протоколе или на различные аудио- и видеоустройства для дальнейшего более детального анализа.

Если при таком тестировании пользователь испытывает некоторые трудности, например, сложности в понимании инструкция, управления и т.д., то разработчикам следует доработать продукт и повторить процедуру тестирования.

При данном тестировании также могут присутствовать модераторы, поскольку нередко наблюдение за тем, как люди взаимодействуют с продуктом, дает возможность найти более оптимальные решения. Задача модератора следить за тем, чтобы респондент не отвлекался на посторонние задачи.

После проведения тестирования удобства использования возникает большая трудность в обработке большого объема полученных данных, поэтому для надлежащего анализа во время тестирования фиксируются:

- речь модератора и респондента;
- выражение лица респондента (съемка на камеру);
- изображение экрана компьютера, с которым работает респондент;
- события, которые связаны с действиями пользователя:
- перемещение курсора;
- нажатие клавиши мыши;
- использование клавиатуры.

Данные потоки данных синхронизируются по тайм-кодам, что позволяет соотносить их между собой во время анализа.

В тестировании также могут участвовать наблюдатели, которые делают различные заметки. Таким образом, фрагменты записи теста комментируются наблюдателем. Обычно модератором выступает разработчик, а наблюдателем – заказчик, испытателем, может быть покупатель.

Существует еще один подход к данному виду тестирования. Пользователю предлагается «идеальный» сценарий решения задачи, обычно, это решения, на которое ориентировался разработчик. При тестировании программы пользователи регистрируют отклонения от заданного сценария для дальнейшего анализа. После обработки информации, полученной после тестирования, осуществляется доработка программы, после чего можно получить интерфейс удовлетворительный с точки зрения пользователя.

Тестирование интерфейса пользователя

Данный вид тестирования применяется для того, чтобы убедится, что пользовательский интерфейс программы соответствует принятым требованиям. При этом осуществляется проверка на то, как обрабатываются ввод с клавиатуры и мыши, а также как отображаются различные элементы графического интерфейса (кнопки, меню, текст и т.д.).

Тестирование локализации

Данный вид тестирования относится к группе нефункционального тестирования. Основная цель – проверить правильность работы программы в определенном окружении. Роль такого окружения могут выполнять:

- аппаратная платформа;
- периферия (устройства ввода, вывода, хранения)
- сетевые устройства;
- операционная система (Windows, Unix и др.);
- базы данных (MS SQL, MySQL Oracle и др.);
- системное ПО (веб-серверы, антивирусы и др.);
- веб-браузеры (Google Chrome, Opera, Mozilla Firefox и др.)

1.1.3 Виды тестирования, связанные с изменениями

Дымовое тестирование

Данный вид тестирования представляет собой короткий цикл тестов, которые выполняются с целью подтвердить, что после сборки кода программа работает надлежащим образом.

При дымовом тестировании осуществляется поверхностное тестирование наиболее важных модулей системы на выполнение требуемых задач и наличия различных дефектов. Если такие дефекты не найдены, то считается что система прошла дымовое тестирование, и оно передается для проведения полного цикла тестирования. В противном случае система подлежит доработке.

Регрессионное тестирование

Регрессионное тестирование направлено на выявление изменений, сделанных в программе или окружающей среде (например, переход на другую ОС, базу данных, сервер приложения и т.д), с целью подтверждения того, что программа или система обладает той же функциональностью, как и прежде. Регрессионными тестами могут быть как функциональные, так и нефункциональные тесты.

Обычно для данного вида тестирования используются тесты, которые были написаны на более ранних стадиях разработки или тестирования. При этом осуществляется автоматизация регрессионных тестов для ускорения последующего тестирования.

Сэм Канер описал 3 основных типа регрессионного тестирования:

«Регрессия багов (Bug regression) - попытка доказать, что исправленная ошибка на самом деле не исправлена.

Регрессия старых багов (Old bugs regression) - попытка доказать, что недавнее изменение кода или данных сломало исправление старых ошибок, т.е. старые баги стали снова воспроизводиться.

Регрессия побочного эффекта (Side effect regression) - попытка доказать, что недавнее изменение кода или данных сломало другие части разрабатываемого приложения» [1].

1.2 Существующие решения

Расширение PerfectPixel

Для работы с данным расширением для браузера требуется заранее подготовить макет тестируемой страницы. Затем зайти на нужную web-страницу, вручную добавить макет и визуально сравнить все элементы на странице с наложенным макетом.

Поэтому данное расширение имеет ряд недостатков для работы тестирования:

- для каждого браузера отдельно нужно установить расширение;
- каждый макет добавляется вручную;
- так как проверка происходит вручную, можно пропустить несоответствие.

Gemini

Gemini — инструмент с открытым исходным кодом, позволяющий автоматизировать регрессионное тестирование отображения веб-страниц. Gemini снимает области экрана с элементами на странице и сравнивает их с эталонными изображениями элементов. Если изображения не совпадают, их отличия указываются в отчёте.

Тесты разрабатываются на JavaScript и запускаются в реальных браузерах, используя протокол Selenium WebDriver. С Gemini можно работать как в командной строке, так и через графический-интерфейс gemini-gui. Gemini разработан в Яндексе и используется для регрессионного тестирования библиотек блоков и интерфейсов сервисов.

Данное приложение имеет ряд недостатков:

- сложная установка;
- сложная настройка;
- долгая и сложная подготовка тестов;
- необходимо знание языка программирования JavaScript.

BackstopJS

Суть метода заключается в том, что вы делаете скриншоты, а затем после внесения каких-либо правок вы делаете новые скриншоты и сравниваете их с предыдущими. Если есть какие-либо отличия - тесты об этом говорят и показывают. Можно делать скриншоты как сайтов, так и отдельных блоков.

Данное приложение имеет свои преимущества:

- имеется возможность для задания порога несоответствия (в процентах);
- можно тестировать отдельные необходимые блоки на сайте;

Также существует ряд недостатков:

- сложная установка и настройка приложения;
- сложный запуск тестов.

Расширение Check My Links

Данное расширение необходимо установить в браузере. Затем зайти на сайт, который нужно протестировать на корректность ссылок, нажать на иконку расширения программа начнет тестирования, проверенные ссылки будут выделены зеленым цветом, а нерабочие – красным.

Расширение имеет недостаток: происходит тестирования указанной страницы, а не всего домена.

Расширение Site Spider Results

Данное расширение проверяет все ссылки, которые принадлежат указанному домену, но проверка происходит в открытом браузере, что осложняет работу.

XENU Link Sleuth

Бесплатная программа, которая реализует проверку сайта на наличие некорректных ссылок, при работе было обнаружено, что проверяются не все ссылки на сайте.

Вывод

Проведя анализ существующих приложений в области тестирования сайтов, нами было принято решение написать программу, которая могла бы выполнять как регрессионное тестирование сайта, так и проверять домен на наличие некорректных ссылок.

1.3 Описание необходимого функционала

Рассмотрев существующие приложения в области тестирования сайтов, было принято решение реализовать собственное, которое бы объединило в себе сразу два вида тестирования: регрессионное, а также проверку домена на наличие некорректных ссылок.

До начала разработки приложения необходимо указать функционал MVP(от англ. minimum viable product — минимально жизнеспособный продукт) — простейший работающий прототип продукта, которым тестируют спрос до полномасштабной разработки.

Основными преимуществом нашего решения, по сравнению с существующими, будет являться простая настройка и работа с приложением.

Наше приложение должно обладать следующим базовым функционалом:

Для регрессионного тестирования:

- возможность создать новый проект или тест-кейс;
- для тест-кейса указывается название теста, сайт для тестирования, браузер и ряд дополнительных настроек;
- при записи теста можно редактировать список действий, которые входят в тест-кейс;
- запуск проекта со всеми тест-кейсами;
- при обнаружении несоответствия в каком-либо тест-кейсе, прервать данный и продолжить тестирование со следующего тест-кейса;
- отображать результаты тестирование в консоли приложения;
- возможность посмотреть, в чем заключаются различия;

- возможность переименовывать тест-кейсы или проекты;
- возможность удалять тест-кейсы или проекты.

Для тестирования сайта на корректность ссылок:

- создание проекта тестирование с указанием начальной страницы;
- возможность переименовывать и удалять проекты;
- выводить в консоль результаты тестирования.

1.4 Модернизация первоначальной идеи

При разработке решения была обнаружена следующая ситуация, при которой обнаруживалось несоответствие при прохождении теста – на сайте присутствует реклама, которая постоянно меняется. Появилась необходимость в определении такого случая и его игнорирования.

Для решения данной проблемы построим нейронную сеть, которая сможет определять, является ли различия рекламой или нет. Нейронной сети придется распознавать изображения, лучше всего для этой цели подходит сверточная нейронная сеть [2-3].

Сверточная сеть обладает рядом преимуществ в отличии от классических многослойных персептронов:

- уменьшение количества обучаемых нейронов, а как следствие, ускорение обучения сети и уменьшение необходимого количества обучающих данных;
- за счет большого количества абстрактных слоев, они обеспечивают частичную устойчивость к изменениям масштаба, смещениям, поворотам, смене ракурса и прочим искажениям;
- удобное распараллеливание вычислений, а, следовательно, уменьшение времени работы сети при работе на графических процессорах [4].

Выводы по разделу

В данном разделе был проведен анализ методов тестирования и сделан обзор уже реализованных решений, были выявлены их основные особенности.

После выполненного анализа в качестве основного метода тестирования было выбрано регрессионное тестирование в сочетании с проверкой на наличие некорректных ссылок.

Кроме того, для улучшения результатов тестирования было принято решение разработать модель, позволяющую обнаружить рекламу на сайте. В качестве такой модели взята сверточная нейронная сеть для определения слов, которые могут присутствовать в рекламе.

Таким образом, нам предстоит сочетать известные методы тестирования с современными методами машинного обучения, что является отличительной особенностью данной работы.

2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ НЕЙРОННОЙ СЕТИ

В предыдущем разделе было решено использовать свёрточную нейронную сеть для выявления рекламы. При этом будем полагать, что реклама – это изображения, на которых присутствуют характерные слова. Таким образом, разрабатываемая модель нейронной сети должна осуществлять поиск слов, которые могут присутствовать в рекламе.

2.1 Математическая модель свёрточной нейронной сети

Для описания математической модели нейронной сети будем использовать следующий обозначения.

Под будем понимать рассматриваемый в данный момент слой нейронной сети, где - количество слоев в сети. За обозначим количество карт признаков на слое , а за функцию активации рассматриваемого слоя Также, под переменной будем понимать -ую карту признаков на слое [5].

2.1.1 Математическая модель свёрточного слоя

Введем в рассмотрение свёрточный слой . В подобной архитектуре нейронной сети является нечетным числом. Тогда, для карты признаков будет иметь место следующее:

– =
;

–
;

– -
.

Таким образом, карта признаков сверточного слоя будет вычисляться следующим образом:

, (2.1)

[6].

Предположим, что размер входных карт признаков равен а размер применяемой к ним свертки равняется , тогда размер выходной карты признаков вычисляется как:

. (2.2)

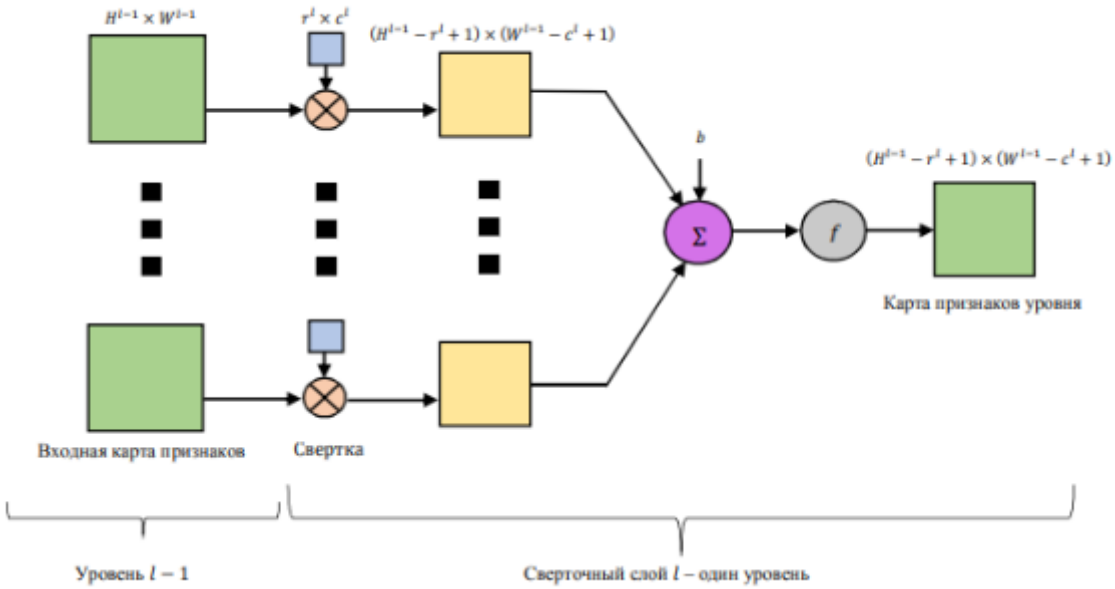

Рисунок 2.1 – Сверточный слой

2.1.2 Математическая модель субдискретизирующего слоя

Введем в рассмотрение субдискретизирующий слой В сверточной нейронной сети принято принимать четным числом. Для карты признаков введем следующее обозначение: – фильтр, применяемый к на слое и – добавочное пороговое значение [7].

Далее будем действовать следующим образом: разделим карту признаков -ого слоя на непересекающиеся блоки размером пикселя. Затем выберем наибольший элемент в каждом блоке и в результате получим матрицу: элементами которой будут являться соответствующие значения максимальных элементов. Таким образом, формула для вычисления элементов будет иметь вид:

- - - -

(2.3)

Карты признаков субдискретизирующего слоя вычисляется, как:

. (2.4)

Благодаря представленным выше рассуждениям, становится возможным посчитать размер карты признаков субдискретизирующего слоя :

(2.5)

___ ___.

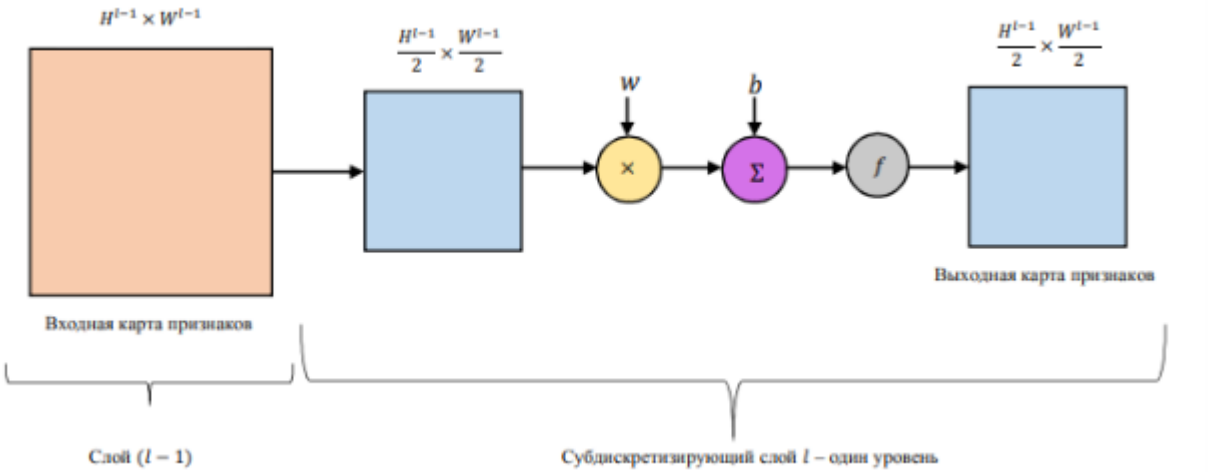

Рисунок 2.2 – Субдискретизирующий слой

2.2 Функция активации

В качестве функции активации возьмем функцию под названием «выпрямитель» (rectifier) [8]. Нейроны с данной функцией активации называются ReLU(rectified linear unit). ReLU реализует простой пороговый переход в нуле и имеет следующую формулу:

(2.6)

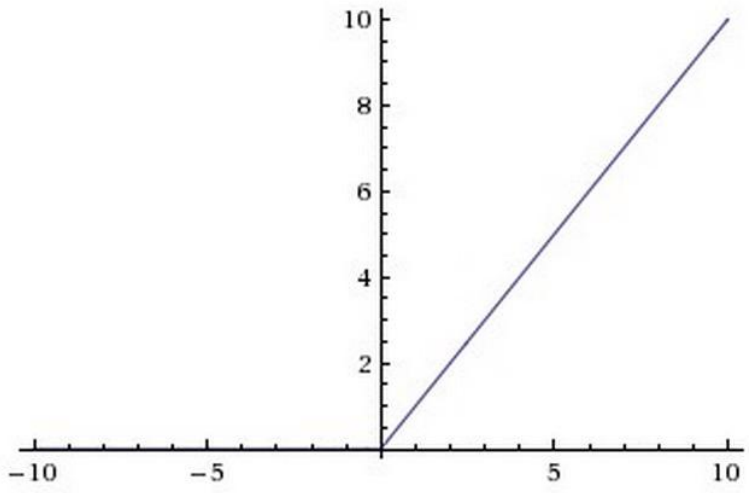

Рисунок 2.3 – Функция активации ReLU

ReLU обладает следующими преимуществами:

- вычисление не требует выполнения сложных операций в сравнении с сигмоидой и гиперболическим тангенсом;
- не подвержен насыщению;
- существенно повышает скорость сходимости стохастического градиентного по сравнению с сигмоидой и гиперболическим тангенсом.

2.3 Математическая модель сети LSTM

Любая рекуррентная нейронная сеть имеет форму цепочки повторяющихся модулей нейронной сети [9]. В обычной RNN структура одного такого модуля очень проста, например, он может представлять собой один слой с функцией активации (гиперболический тангенс).

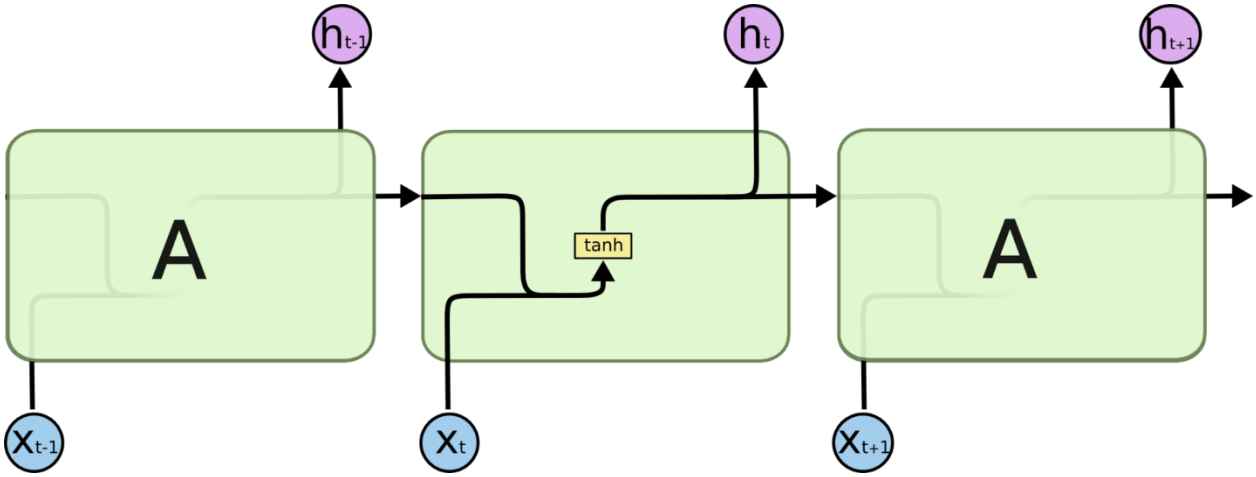

Рисунок 2.4 – Архитектура рекуррентный сети

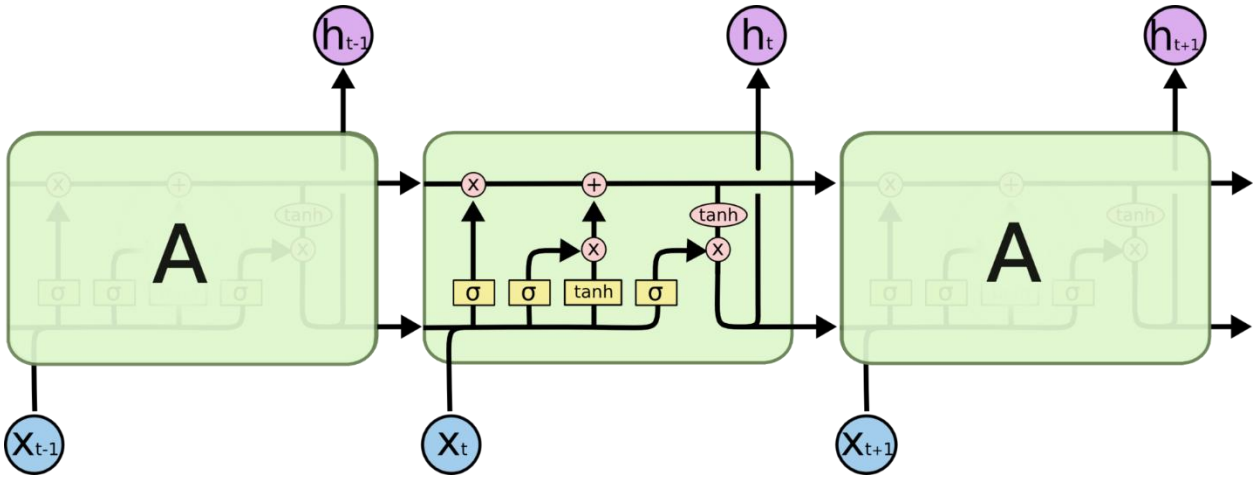

Рисунок 2.5 – LSTM архитектура рекуррентный сети

Долгая краткосрочная память (Long short-term memory; LSTM) – особая разновидность архитектуры рекуррентных нейронных сетей, способная к обучению долговременным зависимостям [10–11]. Структура LSTM также напоминает цепочку, но модули выглядят иначе. Вместо одного слоя нейронной сети они содержат целых четыре, и эти слои взаимодействуют особенным образом [12–14].

Ключевой компонент LSTM – это состояние ячейки (cell state) – горизонтальная линия, проходящая по верхней части схемы (рисунок 2.6).

Состояние ячейки напоминает конвейерную ленту. Она проходит напрямую через всю цепочку, участвуя лишь в нескольких линейных преобразованиях. Информация может легко течь по ней, не подвергаясь изменениям.

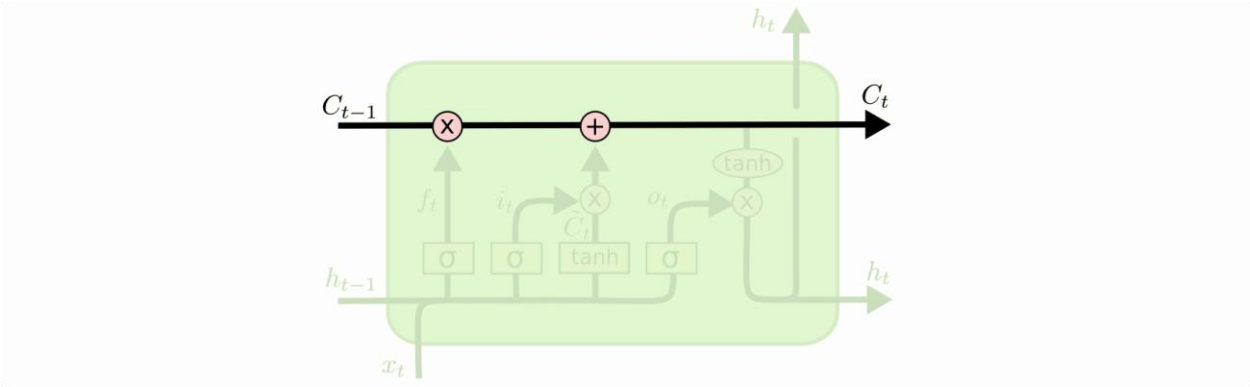

Рисунок 2.6 – Состояние ячейки

Тем не менее, LSTM может удалять информацию из состояния ячейки; этот процесс регулируется структурами, называемыми фильтрами (gates).

Фильтры позволяют пропускать информацию на основании некоторых условий. Они состоят из слоя сигмоидальной нейронной сети и операции поточечного умножения.

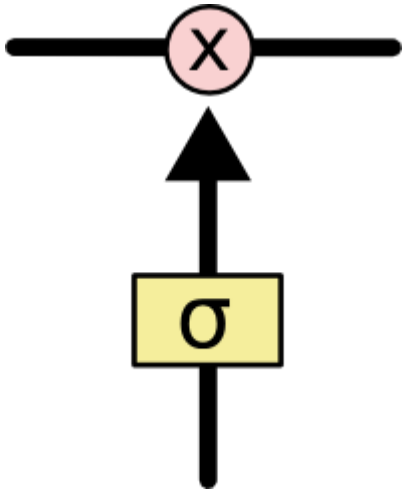

Рисунок 2.7 – Сигмоидальный слой

Сигмоидальный слой возвращает числа от нуля до единицы, которые обозначают, какую долю каждого блока информации следует пропустить дальше по сети. Ноль в данном случае означает “не пропускать ничего”, единица – “пропустить все”. В LSTM три таких фильтра, позволяющих защищать и контролировать состояние ячейки.

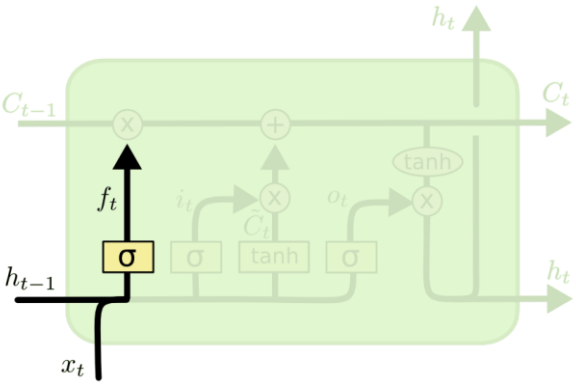

Рисунок 2.8 – Слой фильтра забывания

Первый шаг в LSTM – определить, какую информацию можно удалить из состояния ячейки. Это решение принимает сигмоидальный слой, называемый “слоем фильтра забывания” (forget gate layer) (рисунок 2.8). Он смотрит на и и возвращает число от 0 до 1 для каждого числа из состояния ячейки . 1 означает “сохранить”, а 0 – “выбросить”.

, (2.7)

где выходной вектор из предыдущего блока;
входной вектор;
матрица параметров;
.

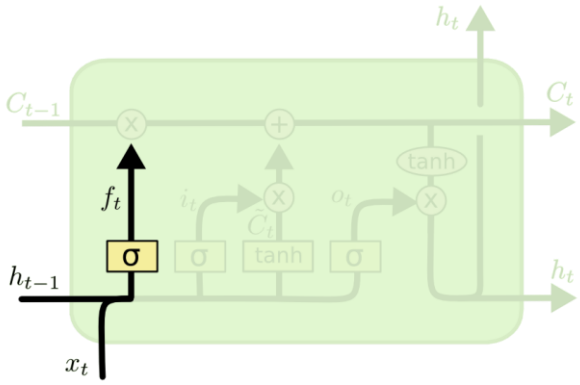

Рисунок 2.8 – Слой фильтра забывания

Следующий шаг – решить, какая новая информация будет храниться в состоянии ячейки. Этот этап состоит из двух частей. Сначала сигмоидальный слой под названием “слой входного фильтра” (input layer gate) (рисунок 2.9) определяет, какие значения следует обновить. Затем -слой строит вектор новых значений-кандидатов , которые можно добавить в состояние ячейки.

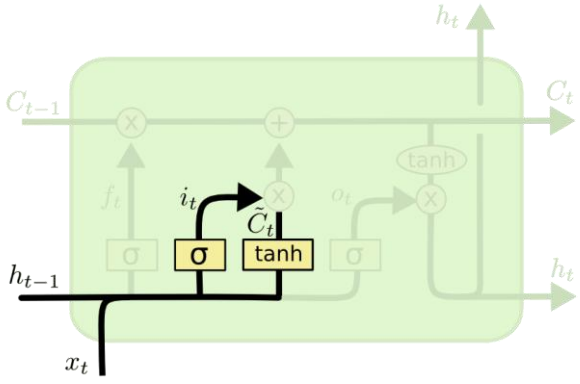

Рисунок 2.9 – Слой входного фильтра

, (2.8)

, (2.9)

где матрицы параметров;

Затем заменим старое состояние на новое:

. (2.10)

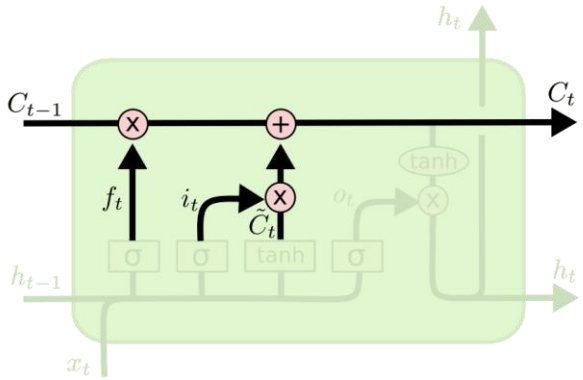

Рисунок 2.10 – Замена старого состояния ячейки

Заключительный этап – вычисление выходных значений. Выходные данные будут основаны на состоянии ячейки. Применим сигмоидальный слой к входному сигналу и - слой к состоянию нашей ячейки. Затем перемножим два этих значения – полученный результат является выходом LSTM блока.

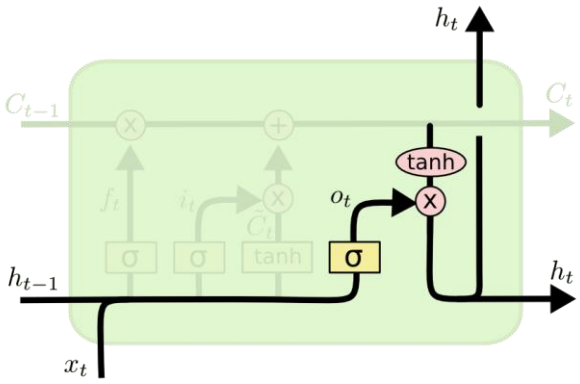

Рисунок 2.11 – Вычисление выходных значений

, (2.11)

, (2.12)

где матрицы параметров;

;

2.3.1 Структура Bidirectional Long Short-Term Memory сети

В работе используется усложненный вариант LSTM сети – BLSTM [15-16].

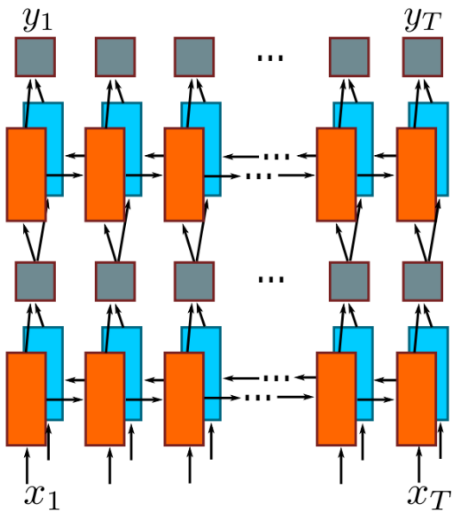

Рисунок 2.12 – Архитектура B-LSTM

В представленной структуре сети оранжевые и голубые блоки являются LSTM сетью, описанной в предыдущем пункте.

2.4 Функция потерь

Обозначим обучающая выборка, где – изображение, а надпись на изображении .

В качестве функции потерь будем использовать логарифмическую вероятность (log-likelihood), вычисление которой основано на методе CTC loss [6, 17]:

, (2.13)

где надпись, полученная рекуррентными и свёрточными слоями из изображения ;

надпись на изображении .

2.5 Метод оптимизации

Рассмотрим метод оптимизации Adagrad (adaptive gradient). Для каждого параметра сети будем хранить сумму квадратов его обновлений в момент времени :

, (2.14)

где значение функции потерь для параметра сети в момент времени

Параметры сети будем обновлять по формуле:

$$\frac{}{} \quad , \tag{2.15}$$

где скорость обучения сети;

сглаживающий параметр, чтобы избежать деления на 0.

Достоинство Adagrad — отсутствие необходимости точно подбирать скорость обучения. Достаточно выставить её в меру большой, чтобы обеспечить хороший запас, но не такой громадной, чтобы алгоритм расходился.

Но, существует и недостаток у метода Adagrad. в формуле (2.15) может увеличиваться сколько угодно, что через некоторое время приводит к слишком маленьким обновлениям и параличу алгоритма. Метод Adadelta [18] является модификацией метода Adagrad, которая призвана исправить этот недостаток.

Мы всё так же собираемся обновлять меньшие веса, которые слишком часто обновляются, но вместо полной суммы обновлений, будем использовать усреднённый по истории квадрат градиента.

Используем экспоненциально затухающее скользящее среднее:

$$, \tag{2.16}$$

где коэффициент сохранения;

скользящее среднее в момент .

Тогда вместо формулы (2.15) получим:

$$\frac{}{} \tag{2.17}$$

В формуле (2.17) знаменатель является корнем из среднего квадратов градиентов, отсюда RMSProp - root mean square propagation:

$$\frac{}{} . \tag{2.18}$$

Добавим в знаменатель формулы (2.17) стабилизирующий член, пропорциональный RMS от На шаге мы ещё не знаем значение , поэтому обновление происходит в 3 этапа: сначала накапливаем квадрат градиента, затем обновляем , после чего обновляем :

$$, \tag{2.19}$$

$$\frac{}{} \tag{2.20}$$

$$\frac{}{} \tag{2.21}$$

Выводы по разделу

В данном разделе была разработана математическая модель нейронной сети, позволяющей обнаружить рекламу на сайте. Полученная модель сочетает принцип сверточных и рекуррентных нейронных сетей.

Было сделано математическое описание сверточных и субдискретизирующих слоев, необходимых для выделения признаков, а также двунаправленной рекуррентной сети долгой краткосрочной памяти, применяющейся для распознавания последовательности букв.

Была выбрана функция активации, описан метод оптимизации нейронной сети с указанием функции потерь, которая характеризует ошибку, полученную на стадии обучения.

3 АРХИТЕКТУРА НЕЙРОННОЙ СЕТИ

3.1 Общая структура нейронной сети

Архитектура нейронной сети включает в себя три основные группы слоев : слои сверточной нейронной сети (сверточные и субдискретизирующие), рекуррентные слои слой транскрипции. Общая структура данной сети представлена на рисунке 3.1.

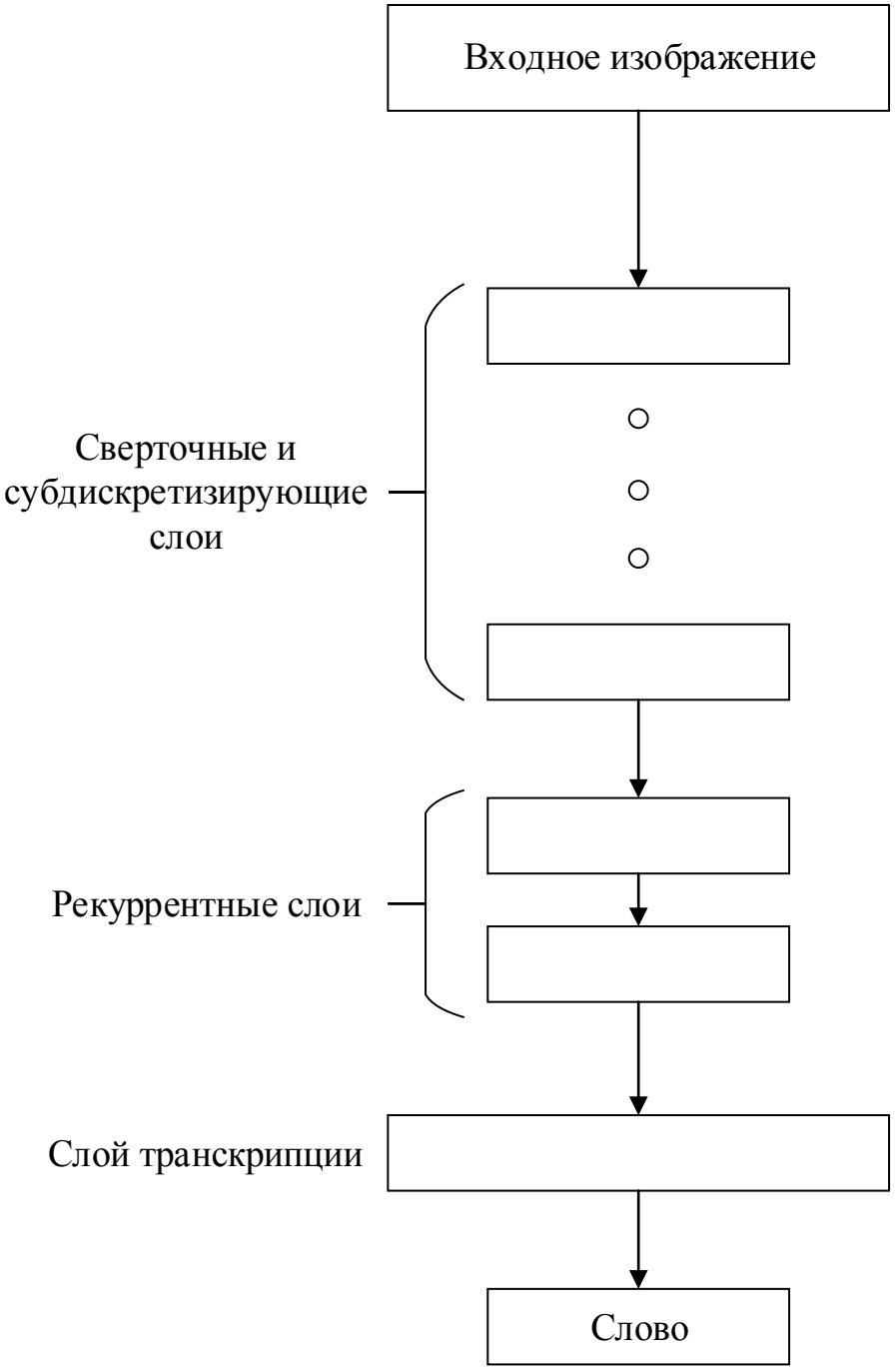

Рисунок 3.1 – Общая структура нейронной сети

Далее сделаем более детальное описание указанных слоев.

3.2 Свёрточные и субдискретизирующие слои

Первый слой – сверточный. Ему на вход подается изображение размером пикселей, к которому применяется свертка размером пикселя. Количество использованных фильтров – 64.

Второй слой в сети – субдискретизирующий (max pooling). Основной функцией данного слоя является уменьшение размерности изображения. Фильтр «пуллинга» имеет размер пикселя.

Третий слой в сети – сверточный, принимающий на вход 64 карты признаков, полученных на предыдущем слое. На данном этапе в качестве фильтров используются матрицы размерности пикселя, формируя на выходе 128 карт признаков.

Следующий слой, как и второй, является субдискретизирующим слоем. На вход данного слоя подаются 128 карт признаков размером пикселей, а выходом являются 128 карт признаков размером пикселей.

Пятый слой является свёрточным. На его вход поступает 128 карт признаков размером пикселей, к которым применяется свертка размерности пикселя. На выходе формируется 256 карт признаков размером пикселей.

Затем идет слой, которому на вход поступает 256 карт признаков размером пикселей, которые также пропускаются через фильтр размера пикселя. Выход является таким же, как у пятого слоя.

Седьмым слоем является слой субдискретизации, ядром «пуллинга» пикселя. На выходе получаются 256 карт признаков размером пикселей.

Затем идет опять свёрточный слой, на вход которого поступает 256 карт признаков размером пикселей с предыдущего слоя. Данные карты проходят через ядро свертки размером пикселя и формируют на выходе уже 512 карт, размером пикселей.

Девятый слой похож на предыдущий, он отличается лишь входом, на который поступает 512 карт признаков размером пикселей.

Затем идет слой субдискретизации, с ядром «пуллинга» пикселя. На выходе получаются 512 карт признаков размером пикселей.

Последний слой – свёрточный, на вход которого подается 512 карт признаков размером пикселей. Ядро свертки имеет размер пикселя. На выходе формируются 512 карт признаков, размерности пикселей.

3.3 Рекуррентные слои

Архитектура данных рекуррентных слоев представляет собой два слоя BLSTM (Bidirectional Long Short Term Memory), рассмотренных в главе 2.

Эти слои используются для работы с последовательностями, в данной нейронной сети – последовательность букв.

Мы используем указанную структуру, которая обеспечивает более высокий уровень абстракций, по сравнению с обычной структурой, и позволяет достигнуть большей производительности [9].

На выходе свёрточных слоев был, получен массив размерности . Чтобы подать его на вход BLSTM нужно его преобразовать, сжав его до двумерного массива размерности . Затем этот массив клонируется, и первая копия подается на вход нейронам, которые переносят данные вперед (forward layers), а над второй копией массива производится операция реверса и он подается на последний нейрон из тех, которые переносят данные в обратном направлении (backward layers).

На следующем этапе процедура повторяется. Затем над матрицей проводятся аффинные преобразования. И после этого выходная матрица имеет размер .

3.4 Слой транскрипции

На данном слое происходит преобразование матрицы размера в слово, которое удалось распознать нейронной сети. Работа данного слоя основана на методе CTC (Connectionist Temporal Classification). Более подробно работа данного слоя описана в работе Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks [17].

3.5 Исходные данные

В базу данных, которая применяется в данной работе для обучения нейронной сети, входит набор изображений Synth 90k. Это автоматически созданные изображения с английскими словами, всего используется около 9 миллионов изображений с использованием 90 тысяч английских слов.

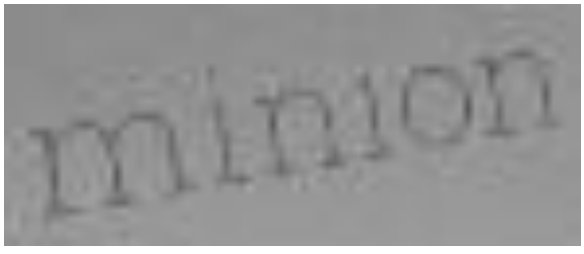

Рисунок 3.2 – Пример изображения со словом «minion»

Рисунок 3.3 – Пример изображения со словом «decouple»

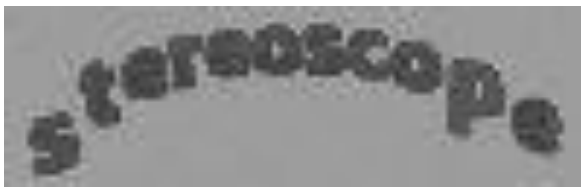

Рисунок 3.4 – Пример изображения со словом «stereoscope»

3.6 Оборудование и программное обеспечение

Разработка модели нейронной сети осуществлялась в среде программирования Pycharm на языке Python v3.6 с использованием библиотеки TensorFlow [19].

TensorFlow – это библиотека программного обеспечения с открытым исходным кодом для высокопроизводительных численных вычислений. Его гибкая архитектура позволяет легко проводить вычисления на различных платформах (процессорах, графических процессорах, TPU), от персональных компьютеров и мобильных устройств до кластеров серверов. TensorFlow разработана инженерами команды Google Brain, представляет собой гибкое вычислительное ядро для работы с алгоритмами машинного и глубокого обучения в различных областях.

Обучение сети происходило на видеокарте NVIDIA GeForce GTX 1060 6 Gb и использованием CUDA 9.0. CUDA – это архитектура параллельных вычислений от NVIDIA, позволяющая существенно увеличить вычислительную производительность благодаря использованию GPU (графических процессоров).

Выводы по разделу

В данном разделе была описана архитектура разработанной нейронной сети, которая способна распознавать слова на изображениях. Была сделана общая схема с обозначением основных структурных составляющих. Подробно описаны гиперпараметры сверточных и субдискретизирующих слоев, а также принцип работы двунаправленной рекуррентной сети долгой краткосрочной памяти и слоя транскрипции.

Сделано описание средств, которые использовались для реализации сети, а также оборудования, применявшееся для обучения разработанной модели.

4 ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

При запуске программы, перед пользователем открывается окно (рисунок 4.1) с выбором типа тестирования:

- регрессионное тестирования сайта;
- проверка домена на наличие нерабочих ссылок.

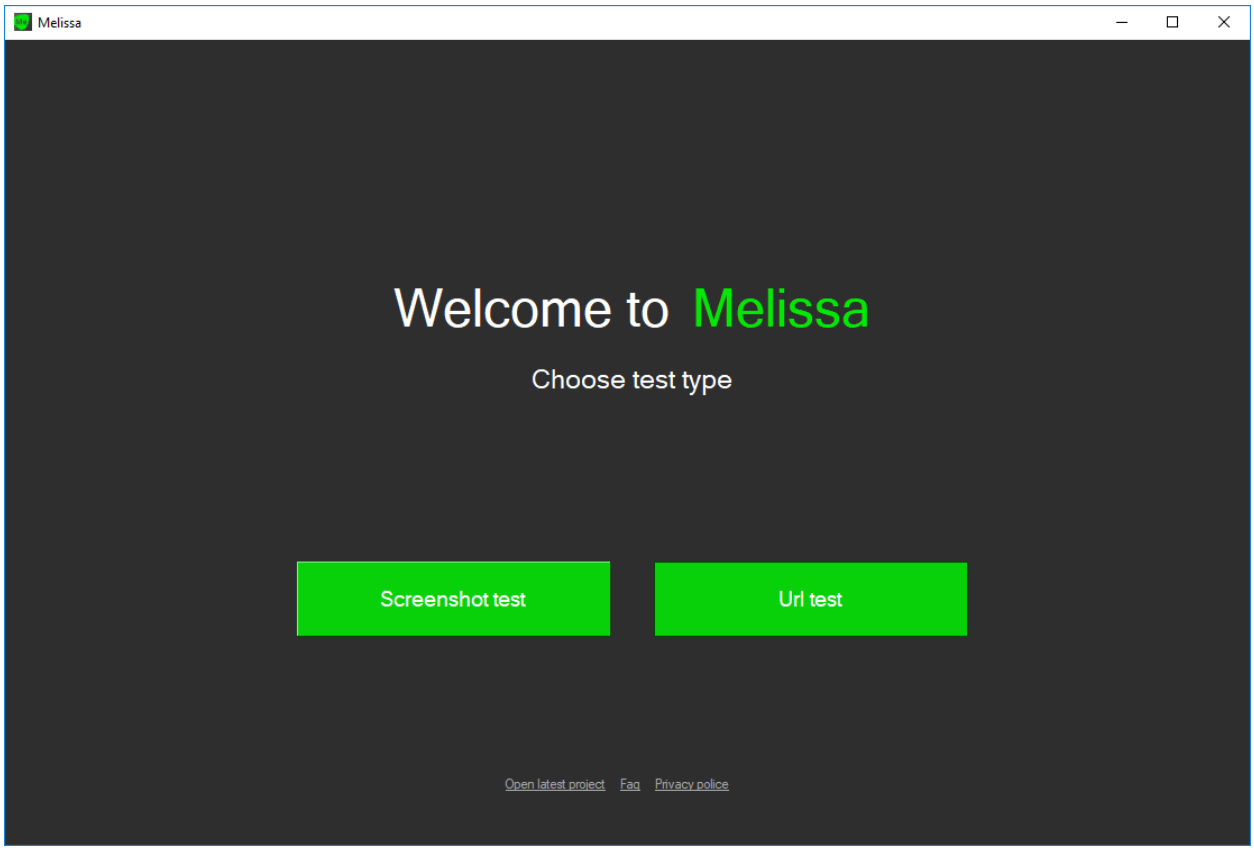

Рисунок 4.1 – Начальное окно программы с выбором тестирования.

При нажатии кнопки «Screenshot test» будет выбрано регрессионное тестирование, а при нажатии «Url test» можно будет провести тестирование домена.

4.1 Регрессионное тестирование

В данном пункте подробнее рассмотрим, как происходит тестирования выбранного сайта и какие функции доступны пользователю.

При первом запуске будет предложено создать новый проект для тестирования (рисунок 4.2).

В поле для ввода необходимо ввести названия будущего проекта и нажать кнопку «Create» для продолжения.

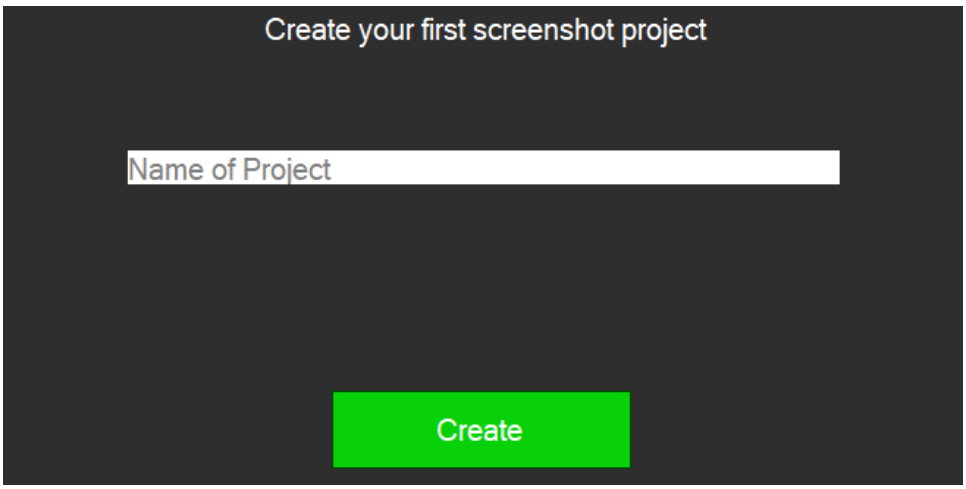

Рисунок 4.2 – Окно для создания нового проекта.

В поле для ввода необходимо ввести названия будущего проекта и нажать кнопку «Create» для продолжения.

На слудующем этапе проимходит создание тест-кейса, который будет добавлен в проект (рисунок 4.3).

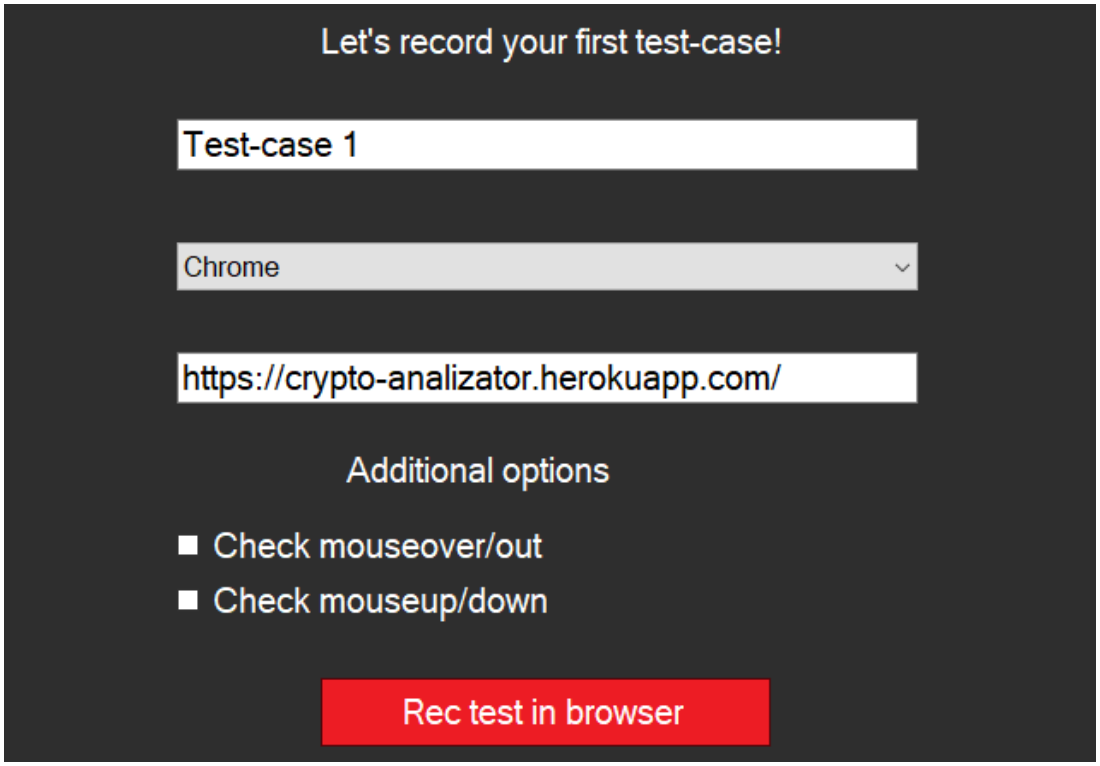

Рисунок 4.3 – Окно для создания нового тест-кейса.

В первом поле необходимо указать названия нашего тест-кейса, во втором можно выбрать браузер, в котором хотим провести тест: Chrome или Phantom, в следующем поле необходимо указать адрес сайта, который мы хотим протестировать. Также можно выбрать дополнительные параметры:

– учитывать событие mouseover/out;
– учитывать событие mouseup/down.

Событие mouseover происходит, когда мышь появляется над элементом, а mouseout – когда уходит из него.

Событие mousedown возникает при нажатии мыши, а mouseup при отпускнии кнопки.

Когда установлены все необходимые параметры, нужно нажать на кнопку «Rec test in browser» для начала записи теста. После нажатия на данную кнопку откроется окно браузера с указанной web-страницей (рисунок 4.4).

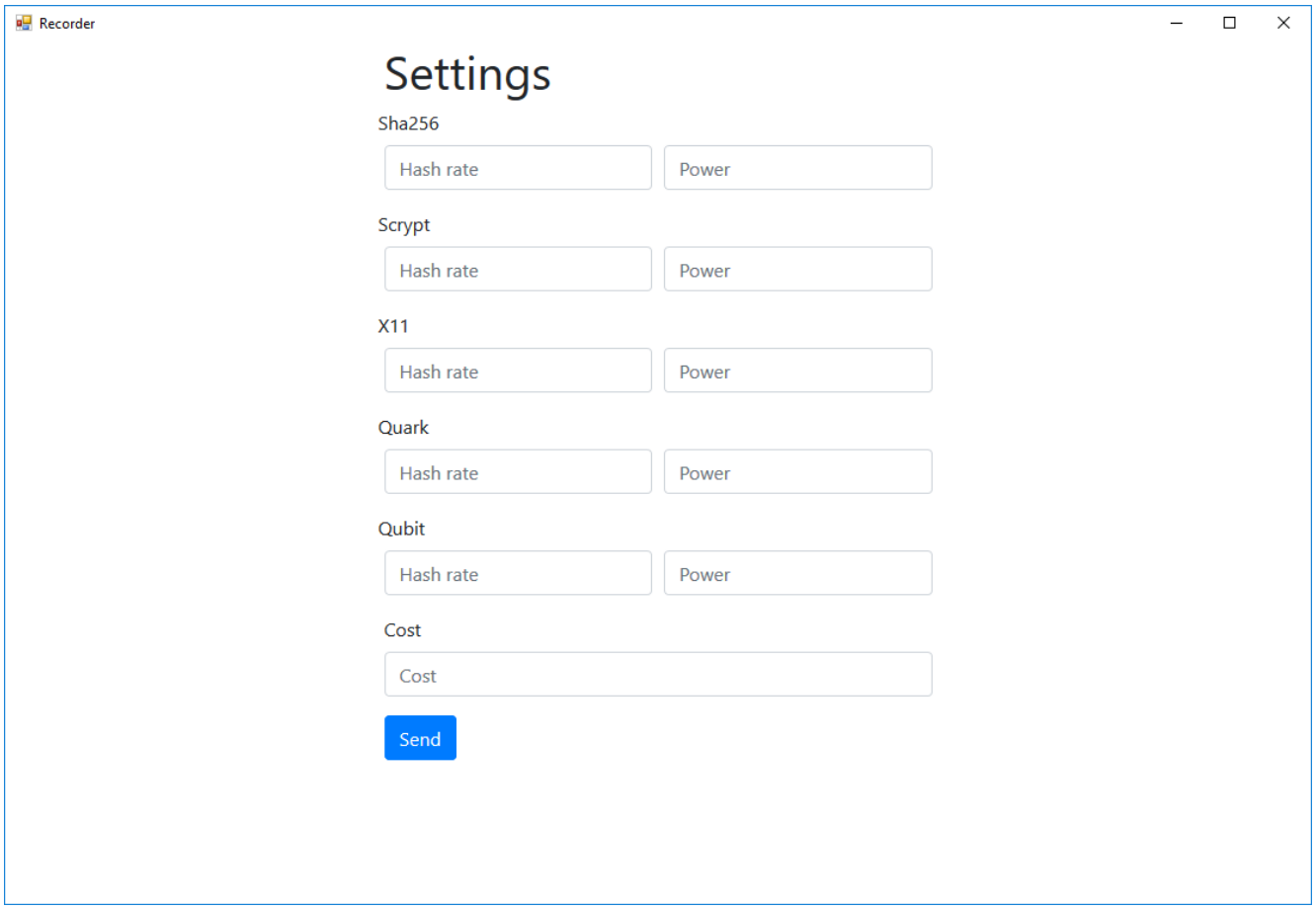

Рисунок 4.4 – Окно браузера с указанной web-страницей.

В окне браузера выполянем все необходимые действия и затем закрываем его.

После этого появится новое окно с записанными командами, которые можно отредактировать (рисунок 4.5).

Всего мы рассматриваем 7 различных команд:

- mouseover – мышь появляется над элементом;
- mouseout – мышь покидает элемент;
- focus – фокусировка на элементе;
- click – клик по элементу левой кнопкой мыши;
- mousedown – событие возникает при нажатии кнопки мыши;
- mouseup – событие возникает при отпускании кнопки мыши;
- type – ввод текста.

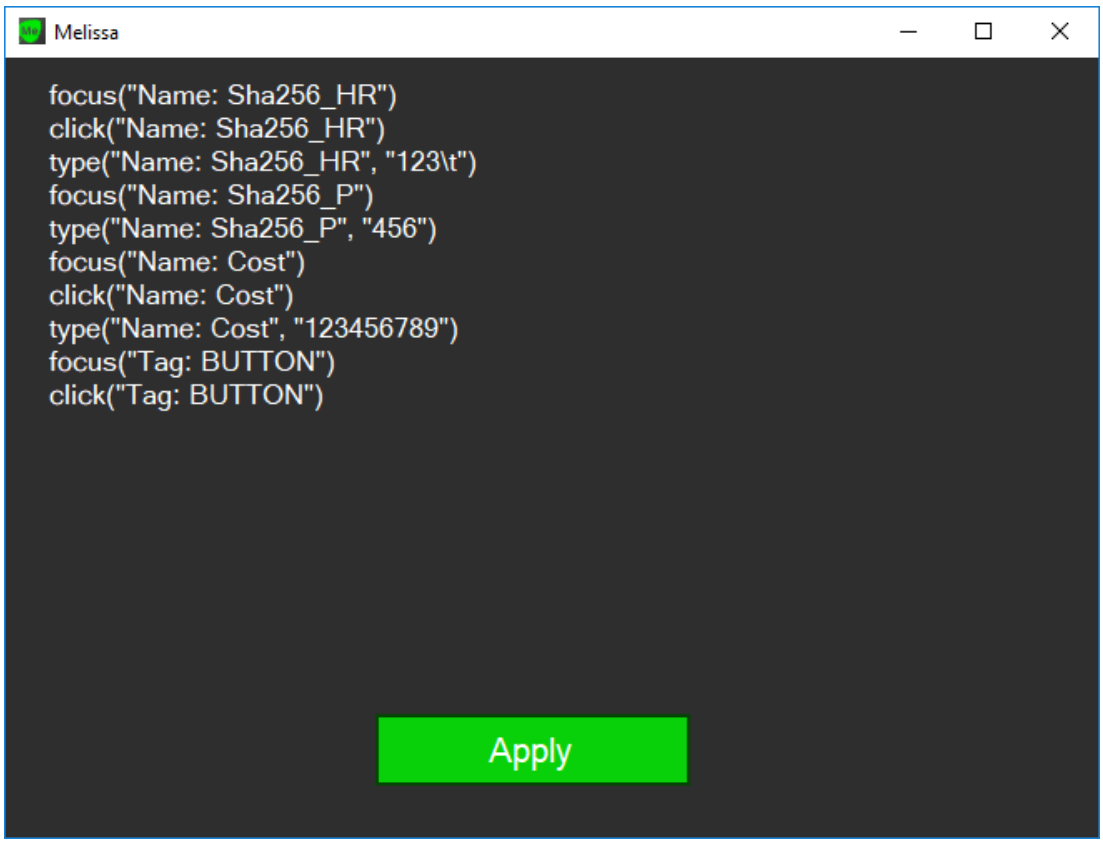

Рисунок 4.5 – Окно с записанными командами.

После каждой команды в скобках указывается тип идентификатора и уникальный идентификатор элемента, над которым была совершена команда. Для команды «type» дополнительно указывается строка с введенным текстом.

После редактирования команд может появится окно (рисунок 4.6) с информацией об ошибке в какой-либо команде, данная команда будет подсвечена красным цветом.

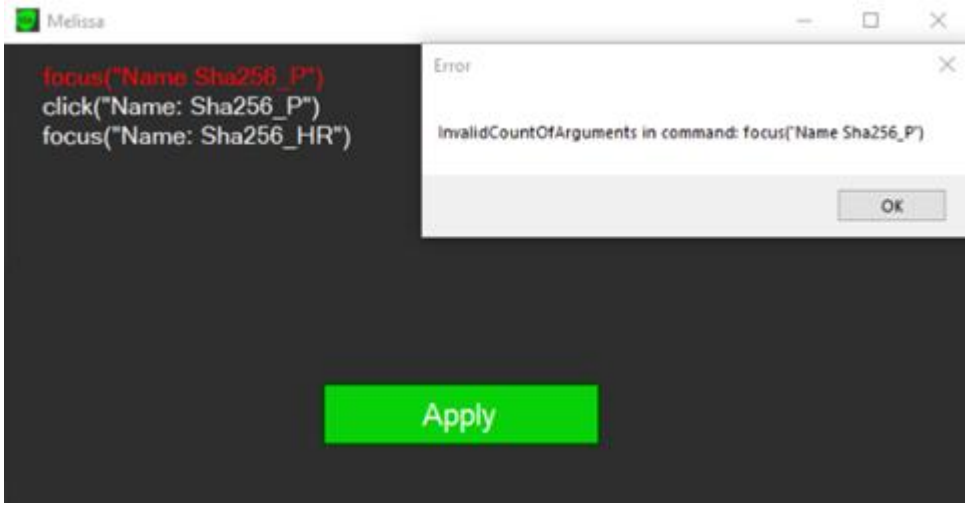

Рисунок 4.6 – Окно с ошибкой в записи первой команды

Правила записи команд слудующие:

- необходимо правильно записать тип команды;
- в круглых скобках указываются параметры командыВ кавычках необходимо указать тип идентификатора затем двоеточие и его значение( );
- для ввода текста через запятую в кавчках указать необходимый текст( ).

После того, как все команды будут верно записаны продолжится запись теста. Программа автоматически повторит все записанные команды в браузереи сделаетпосле каждой команды скриншот, чтобы запомнить состояние.

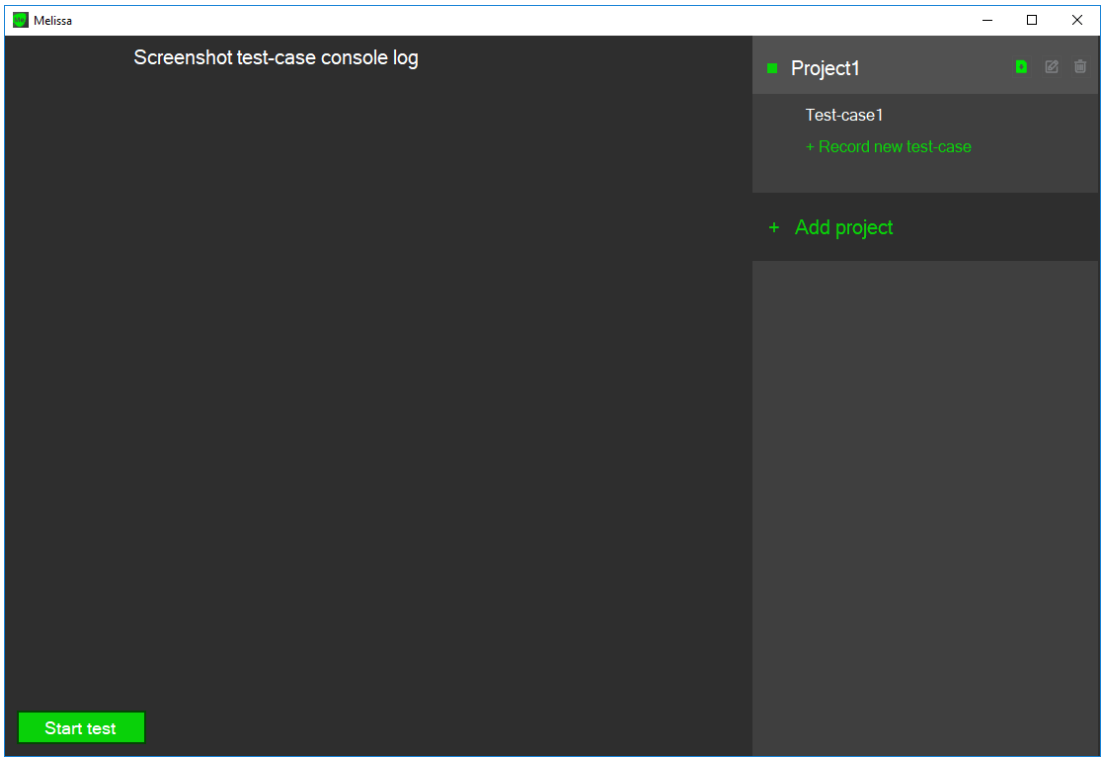

Рисунок 4.7 – Основное окно программы после записи теста

После того как у нас появился перый записанный тест-кейс (рисунок 4.7), можно начать тестирование. Для этого выбирем проект, нажав на него в правой части окна, и затем нажимает кнопку «Start test». Программа откроект браузер и повторит все записанные до этого команды и будет сравнивать скриншоты после каждой выполненной команды. Информация о результате сравнения будет повяляться в основной части окна, которую будем называть консолью.

В данном случае (рисунок 4.8) на сайте не было никаких изменений, поэтому все команды успешно пройдены, это можно наблюдать в основной части окна. Также в консоле отображается название проекта, которые сейчас тестируем и тест-кейс. Если в проекте несколько тест-кейсов, то они будут протестированы все поочереди.

Так же возможна ситуация, когда на сайте произошли некоторые измененения, тогда при выполнении команд могут обнаружится несоотетствия текущих

скриншотов с исходными, которые были сделаны при создании тест-кейса (рисунок 4.9).

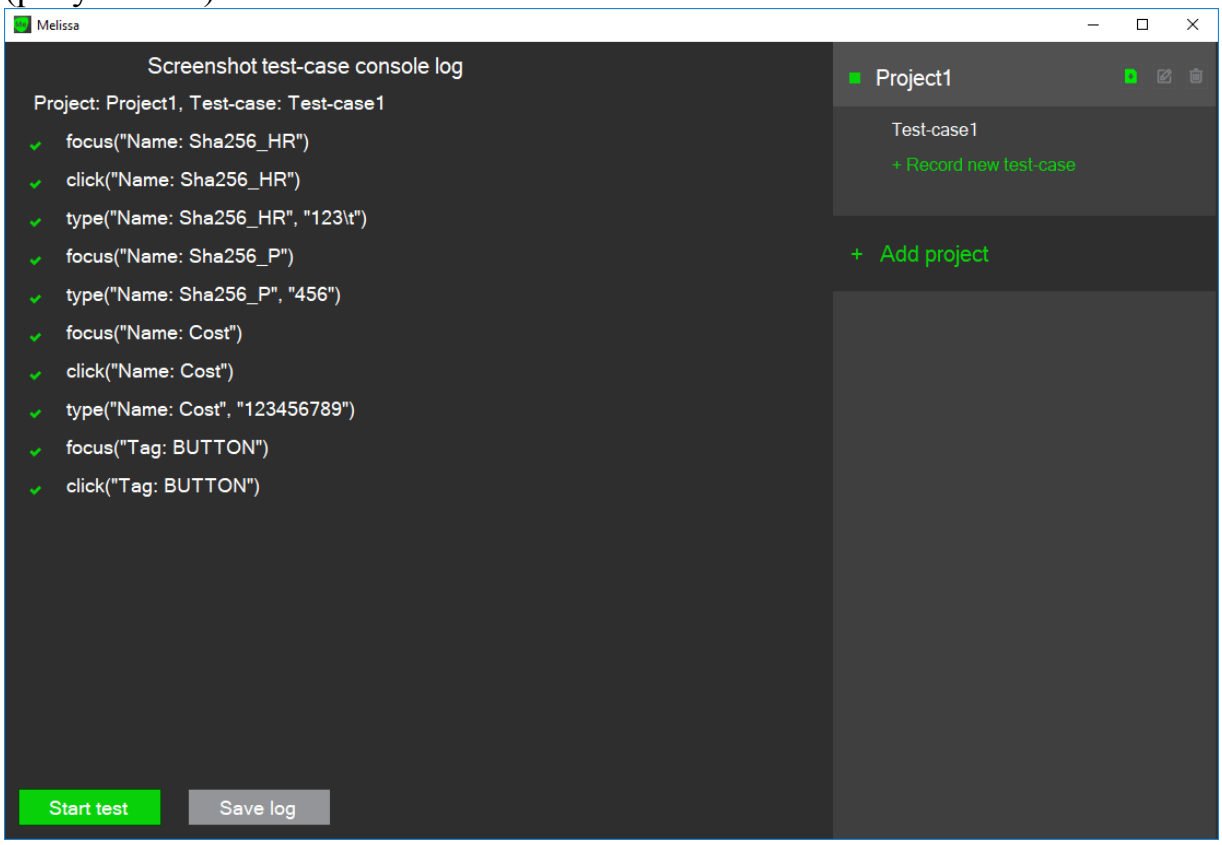

Рисунок 4.8 – Окно после успешного проведения теста

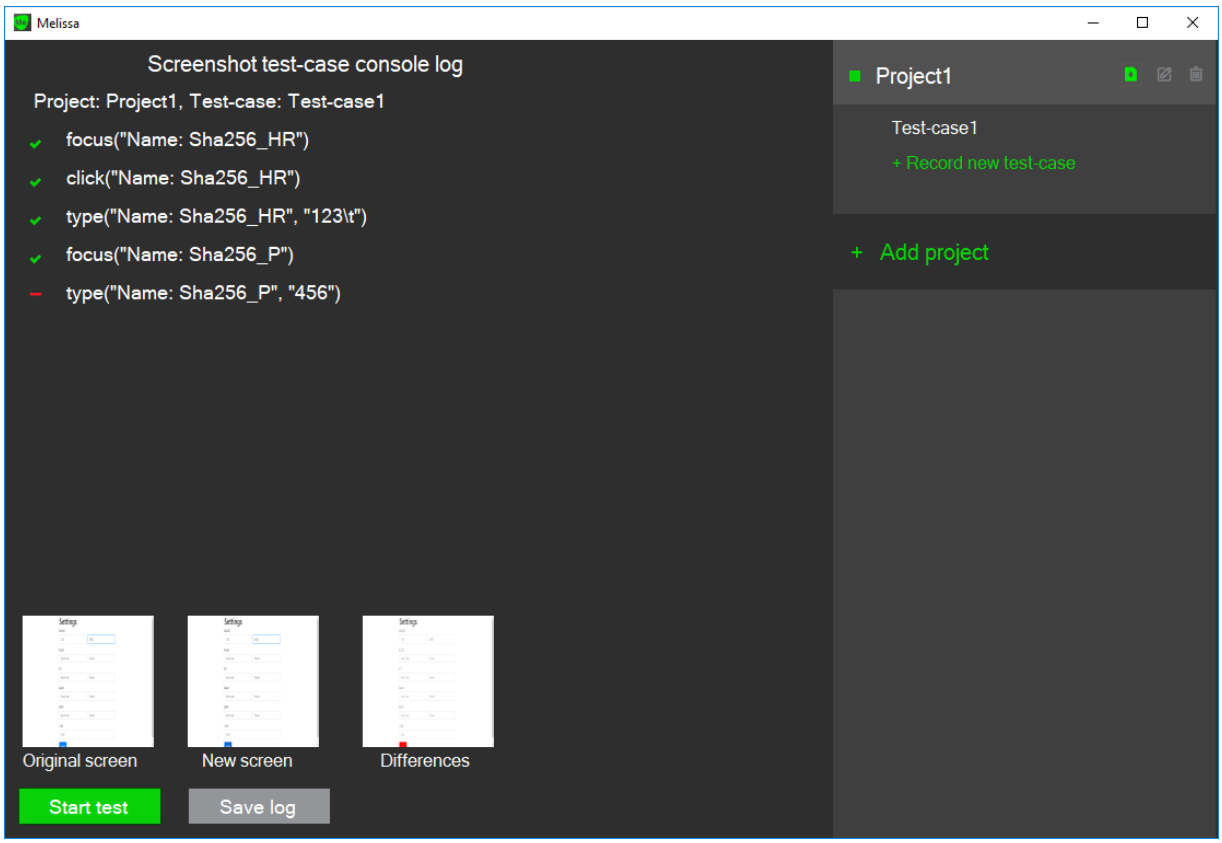

Рисунок 4.9 – Основное окно после обнаружения несоответствия

В данном случае (рисунок 4.9) отображаются 3 маленьких изображения: оригинальный снимок, текущий снимок и их различия. Изображение с различиями можно увеличть, чтобы подробнее рассмотреть, в каком месте происходит несоответствие.

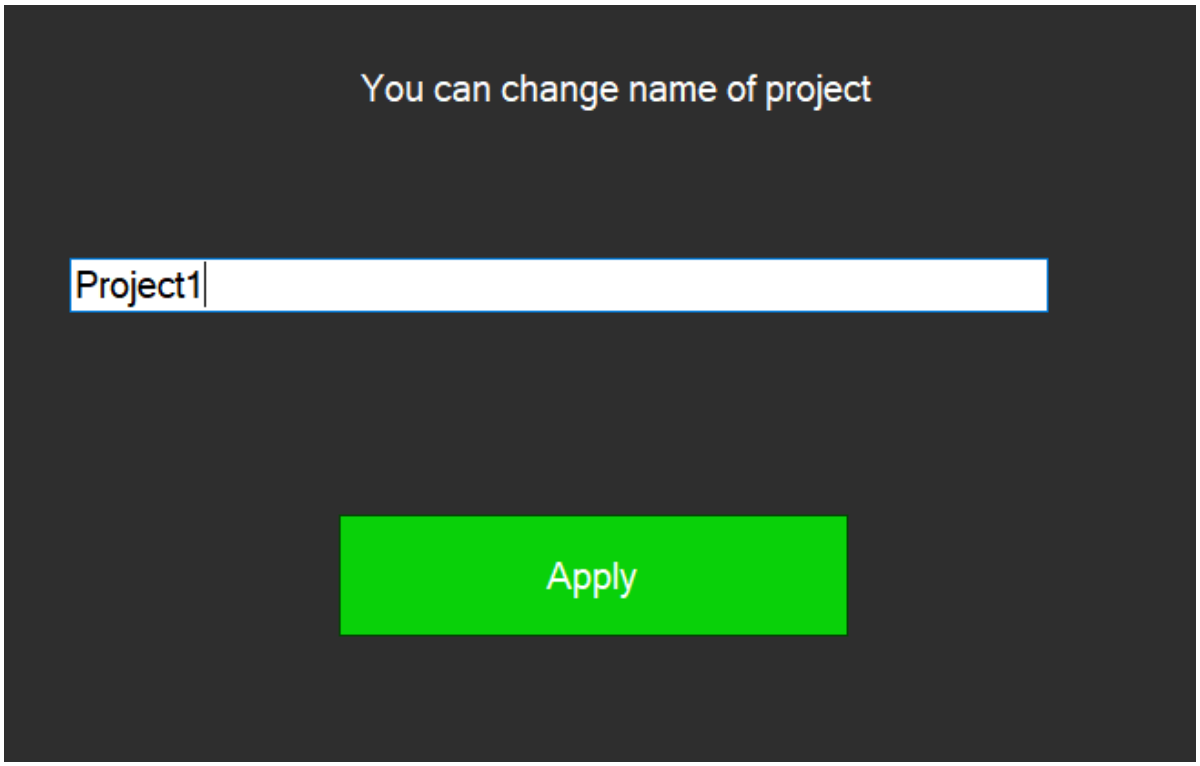

Рисунок 4.10 – Окно для изменения названия проекта

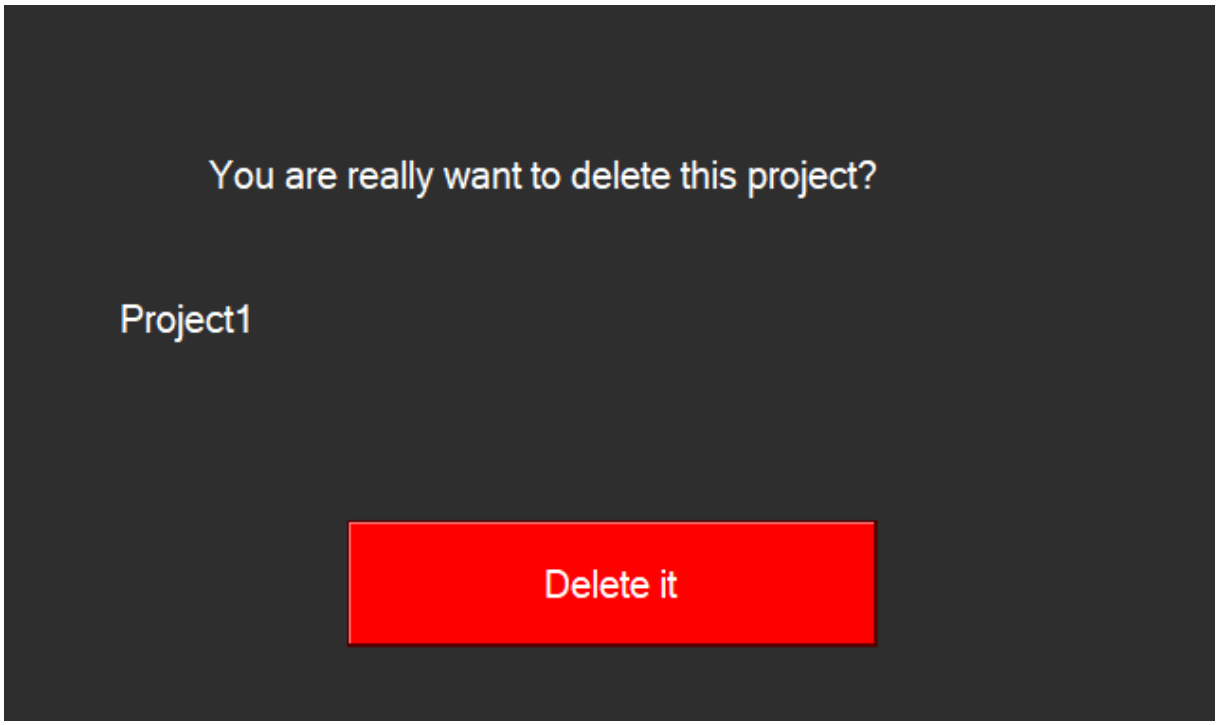

Рисунок 4.11 – Окно для удаления проекта

В программе присутствует функционал для редактирования проектов и тест-кейсов (рисунки 4.10 – 11). Можно изменять названия у тест-кейсов и проектов, а также в случае необходимости можно удалить ненужный проект или тест-кейс.

Также можно добавлять новые проекты или тест-кейся в уже существующие проекты (рисунок 4.7). При нажатии надпись «Add project» появится окно (рисунок 4.2) и можно будет добавить новый проект, а при нажатии на «Record new test-case» появится окно (рисунок 4.3) и можно будет записать еще один тест-кейс.

4.2 Тестирование домена

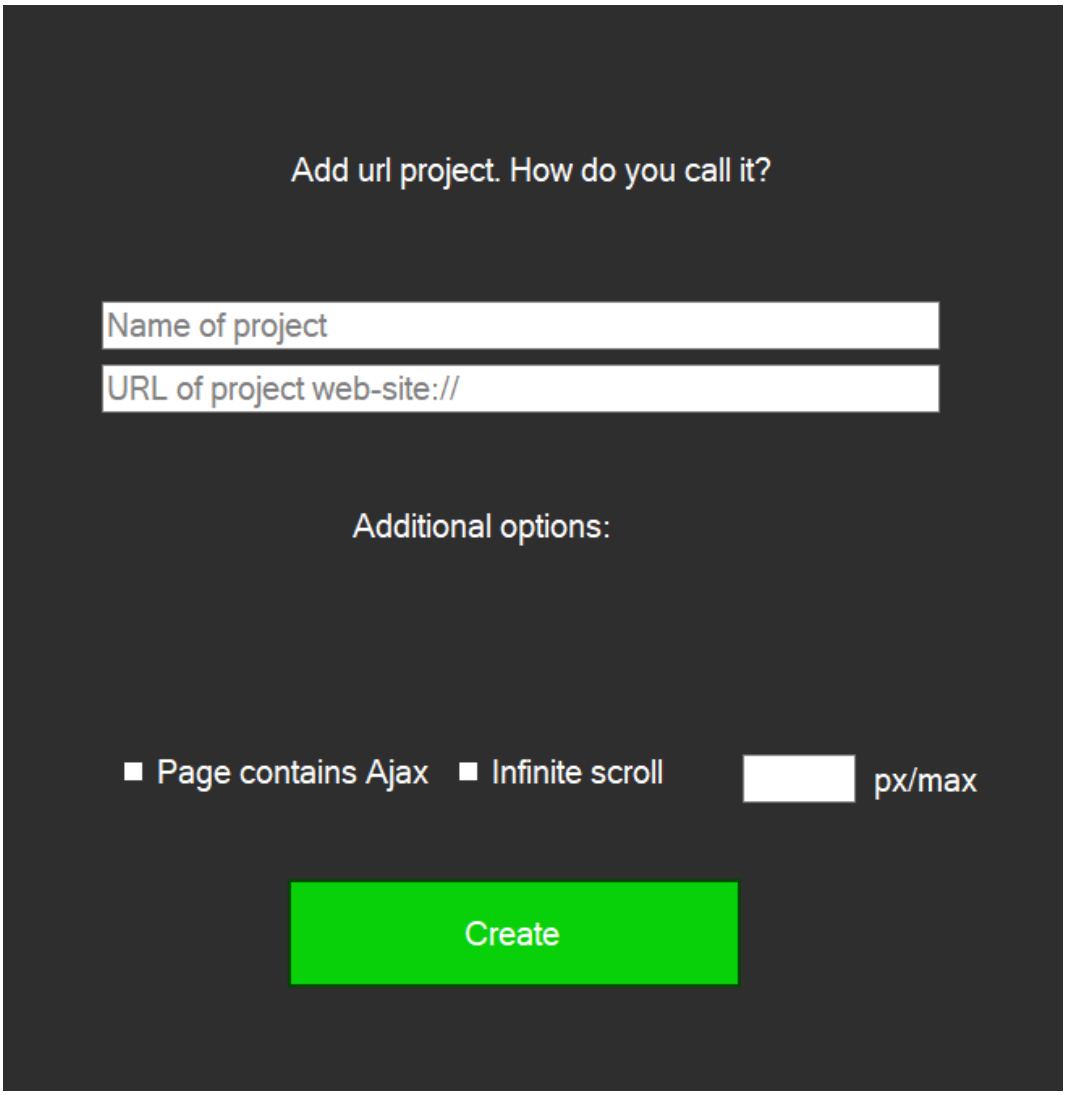

Рисунок 4.12 – Окно для создания проекта

На рисунке 4.12 показано, как выглядит окно для создания нового проекта по тестированию домена.

Для создания необходимо указать название будущего проекта, сайт а также указать, присутствует ли на сайте бесконечная прокрутка.

Затем, после нажатия на кнопку откроется основное окно приложения (рисунок 4.13).

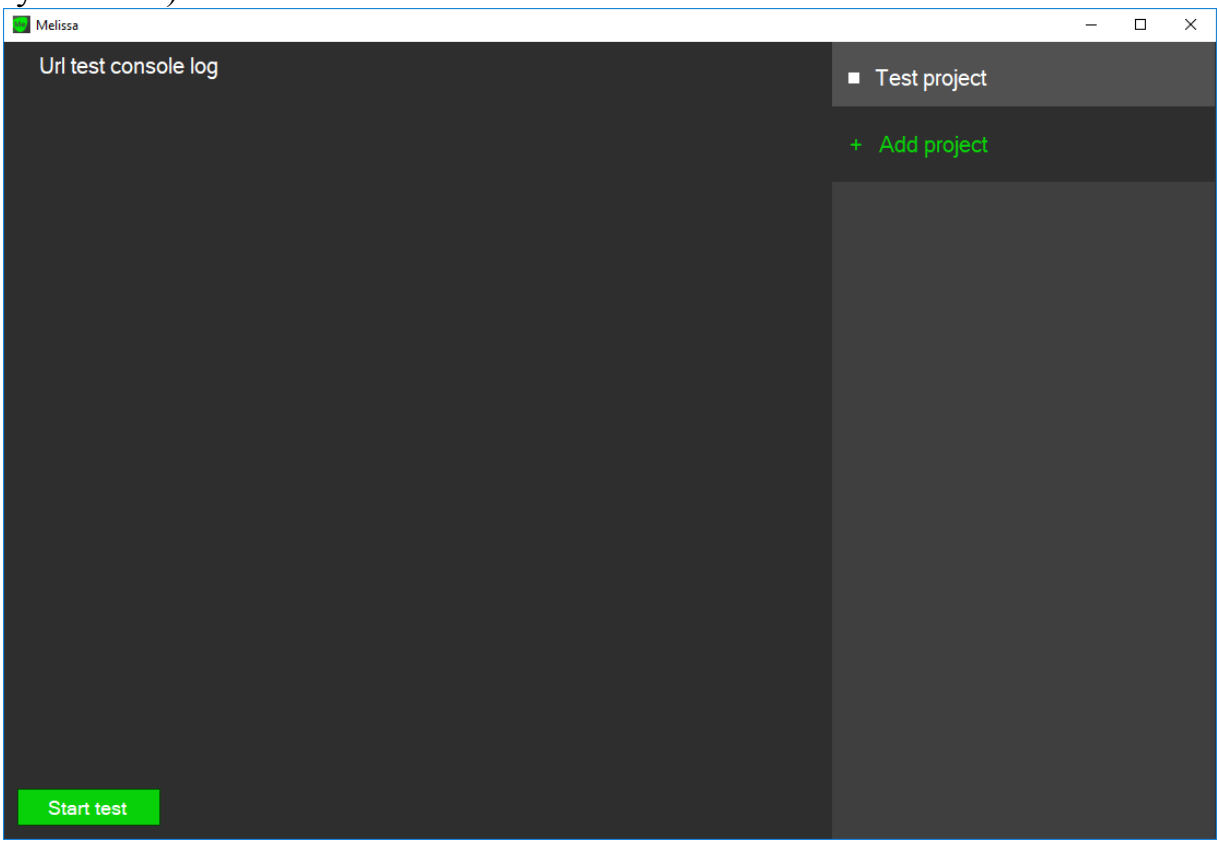

Рисунок 4.13 – Основное окно приложения

Здесь также присутствует возможность удалять ненужные проекты. Также реализован функционал для изменения названия проекта и тестируемого домена (рисунок 4.14).

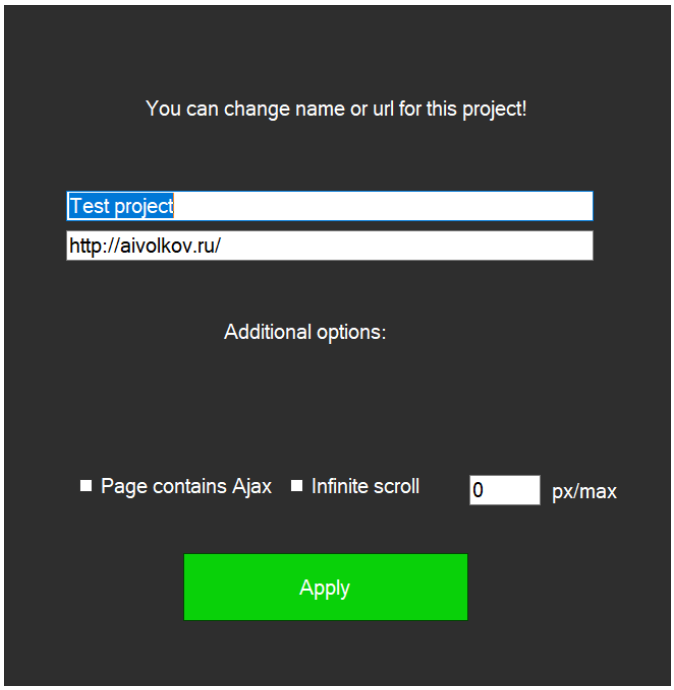

Рисунок 4.14 – Окно для редактирования проекта

Если нужно выйти из редактироваия проекта, необходимо дважды кликнуть по форме редактипования. Это также относится к формам для удаления и изменению названий, а также создания проектов (рисунки 4.2, 4.3, 4.10 – 12).

Для запуска тестирования проекта, необходимо его выбрать в списке проектов а затем нажать на кнопку «Start test» (рисунок 4.13).

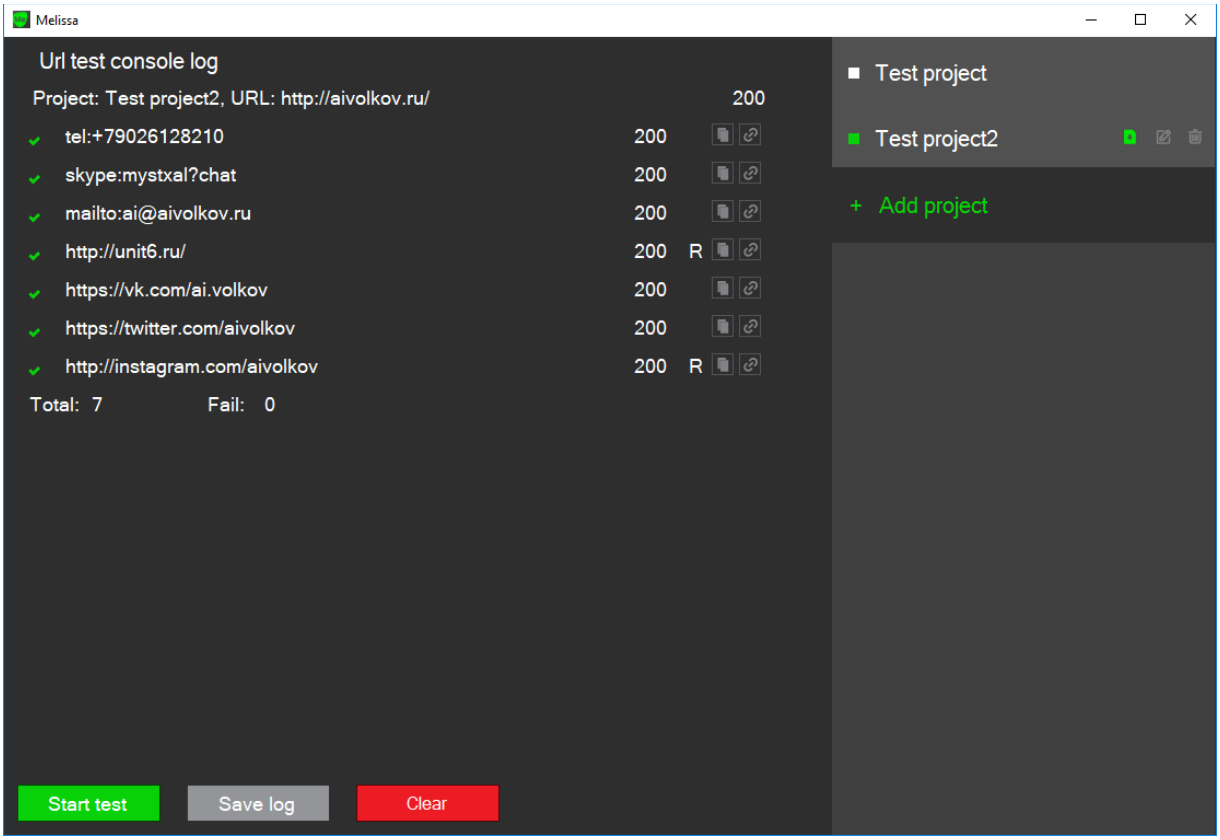

Рисунок 4.15 – Окно с результатами тестирования

На рисунке 4.15 показано, как выглядит окно основное окно программы после тестирования. В консоле указаны все найденные ссылки, если слева от нее располагается зеленая галочка, то ссылка без ошибок, если красный минус, то присутствуешь какая-то ошибка. Справа от кажой ссылки укащывается целое число больще 100 и меньше 600 – это код состояния HTTP. Корректным состоянием считается код с значением от 200 до 300.

Также справа от каждого кода присутствуют 2 кнопки:

– для копирования соответствующей ссылки;
– для открытия этой ссылки в браузере.

4.3 Пример работы нейронной сети

Сначала найдем все области на изображении, в которой могут быть слова с помощью библиотеки OpenCV (рисунок 4.16).

Рисунок 4.16 – Пример изображения с найденными областями

Затем по очереди эти изображения подаются нейронной сети для обнаружения слов. Потом происходит поиск полученного результата в наборе слов, которые могут встречаться в рекламе.

Неточность в работе связана с наличием области такого же цвета как текст слева от буквы «e».

Далее подадим следующее изображение (рисунок 4.18), на котором без ошибок удалось прочитать слово «go». При поиски этого слова в имеющемся наборе совпадений не было обнаружено.

Рисунок 4.17 – Пример входного изображения

На следующем изображении (рисунок 4.19) удалось прочитать «entire», его также не удалось найти в наборе.

Рисунок 4.18 – Пример входного изображения

Рисунок 4.19 – Пример входного изображения

На рисунке 4.20 нейросети удалось прочитать слово «the», но оно не относится к рекламе.

Рисунок 4.20 – Пример входного изображения

Рисунок 4.21 – Пример входного изображения

На рисунках 4.20 и 4.21 удалось прочитать слова «store» и «tostoref» соответственно. При сопоставлении этих результатов с набор имеющихся слов обнаружилось совпадение со словом «store». На основе этого совпадения можно сделать вывод, что на изображении присутствует реклама.

Неточность в работе связана с наличием области такого же цвета как текст слева от буквы «е».

Выводы по разделу

В данном разделе был представлен интерфейс программы, а также ее функциональные возможности. Описано каждое окно программы и его предназначение.

Продемонстрирована работа приложения по тестированию сайта, а также работа разработанной нейронной сети, которая способна обнаружить слова, характерные для рекламы и отнести определенное изображение к данной категории.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы было разработано приложение для автоматизации тестирования сайта.

Данное приложение осуществляет регрессионный вид тестирования, а также проверку сайта на наличие некорректных ссылок. Для регрессионного тестирования реализована сверточная нейронная сеть, которая позволяет обнаруживать рекламу, присутствующую на сайте. Таким образом, достигается более высокое качество тестирования по сравнению с уже реализованными приложениями, предназначенными для тестирования сайта.

В ходе работы были решены следующие задачи:

1) выполнен обзор методов тестирования и существующих решений;
2) выбраны основные методы – регрессионное тестирование и проверка на наличие некорректных ссылок в сочетании с нейронной сетью, позволяющей повысить качество тестирования, путем обнаружения рекламы;
3) разработана математическая модель нейронной сети, позволяющей обнаружить рекламу на сайте;
4) выполнена программная реализация;
5) осуществлена проверка работы программы, путем проверки сайта на наличие некорректных ссылок.

Разработанное приложение имеет следующие преимущества:

1) обладает удобным интерфейсом;
2) достигается высокая скорость работы;
3) реализована фильтрация рекламы.

К недостаткам можно отнести то, что осуществляется обнаружение только текстовой рекламы, при этом текст должен быть на английском языке. В дальнейшем планируется продолжение работы для расширения функционала приложения, а также его реализация под другие ОС.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Канер, С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений: Пер. с англ./Джэк Фолк, Енг Кен Нгуен. – К.: Издатество «ДиаСофт», 2001.–544 с.

2 Krizhevsky A. ImageNet Classification with Deep Convolutional Neural Networks / I. Sutskever, G. Hinton // USA, Neural Information Processing Systems. – 2012.

3 Jaderberg M. Deep structured output learning for unconstrained text recognition/ K. Simonyan, A. Vedaldi, A. Zisserman// USA, International Conference on Learning Representations. – 2015.

4 Jaderberg M. Reading text in the wild with convolutional neural networks / K. Simonyan, A. Vedaldi, A. Zisserman // International Journal of Computer Vision, V.116, №1, P. 1–20. – 2015.

5 Ciresan D. Multi–column deep neural networks for image classification / U. Meier, J. Schmidhuber // Conference on Computer Vision and Pattern Recognition. – 2012

6 Baoguang S. An End-to-End Trainable Neural Network for Image–based Sequence Recognition and Its Application to Scene Text Recognition / B. Xiang, Y. Cong // USA, Cornell University Library. – 2015.

7 Stutz D. Understanding Convolutional Neural Networks / Germany, Seminar Report, Fakultät für Mathematik, Informatik und Naturwissenschaften. – 2014.

8 URL:http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/ (дата обращения 25.04.18)

9 Graves A. Speech recognition with deep recurrent neural networks / Canada, Vancouver, International Conference on Acoustics, Speech and Signal Processing. – 2013.

10 Hochreiter S. Long short-term memory / J. Schmidhuber // Neural Computation vol. 9, №8, P. 1735-1780, 1997.

11 Wang K. End-to-end scene text recognition / B. Babenko, S. Belongie // Spain, Barcelona, International Conference on Computer Vision, 2011.

12 URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата обращения 30.04.18)

13 Bengio Y. Learning longterm dependencies with gradient descent is difficult / P. Y. Simard, and P. Frasconi. // NN, V.5, №2, P. 157–166, 1994.

14 Gers F.A. Learning precise timing with LSTM recurrent networks / N. N. Schraudolph, and J. Schmidhuber // JMLR, V.3, P. 115–143, 2002.

15 Zhiyong C. Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction / R. Ke // USA, Cornell University Library. – 2018.

16 Frinken V. Deep BLSTM Neural Networks for Unconstrained Continuous Handwritten Text Recognition / France, Nancy, International Conference on Document Analysis and Recognition. – 2015.

17 Graves A. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks / S. Fernandez, F. Gomez, J. Schmidhuber // USA, Pittsburgh, International Conference on Machine Learning. – 2006.

18 Zeiler M. Adadelta: an adaptive learning rate method // USA, Cornell University Library. – 2012.

19 Schrimpf M. Should I use TensorFlow / USA, Cornell University Library. – 2016.

ПРИЛОЖЕНИЕ 1 ТЕКСТ ПРОГРАММЫ

cnn_basenet.py – модуль, который содержит основные методы, для нейронной сети.

```
"""
The base convolution neural networks mainly implement some useful cnn functions
"""
import tensorflow as tf
import numpy as np
from abc import ABCMeta


class CNNBaseModel(metaclass=ABCMeta):
    """
    Base model for other specific cnn ctpn_models
    """
    def __init__(self):
        pass

    @staticmethod
    def conv2d(inputdata, out_channel, kernel_size, padding='SAME', stride=1,
w_init=None, b_init=None,
               nl=tf.identity, split=1, use_bias=True, data_format='NHWC',
name=None):
        """
        Packing the tensorflow conv2d function.
        :param name: op name
        :param inputdata: A 4D tensorflow tensor which ust have known number of
channels, but can have other
        unknown dimensions.
        :param out_channel: number of output channel.
        :param kernel_size: int so only support square kernel convolution
        :param padding: 'VALID' or 'SAME'
        :param stride: int so only support square stride
        :param w_init: initializer for convolution weights
        :param b_init: initializer for bias
        :param nl: a tensorflow identify function
        :param split: split channels as used in Alexnet mainly group for GPU
memory save.
        :param use_bias:  whether to use bias.
        :param data_format: default set to NHWC according tensorflow
        :return: tf.Tensor named ``output``
        """
        with tf.variable_scope(name):
            in_shape = inputdata.get_shape().as_list()
            channel_axis = 3 if data_format == 'NHWC' else 1
            in_channel = in_shape[channel_axis]
            assert in_channel is not None, "[Conv2D] Input cannot have unknown
channel!"
            assert in_channel % split == 0
            assert out_channel % split == 0

            padding = padding.upper()

            if isinstance(kernel_size, list):
                filter_shape = [kernel_size[0], kernel_size[1]] + [in_channel /
split, out_channel]
            else:
                filter_shape = [kernel_size, kernel_size] + [in_channel / split,
```

```
out_channel]

            if isinstance(stride, list):
                strides = [1, stride[0], stride[1], 1] if data_format == 'NHWC'
else [1, 1, stride[0], stride[1]]
            else:
                strides = [1, stride, stride, 1] if data_format == 'NHWC' else [1,
1, stride, stride]

            if w_init is None:
                w_init = tf.contrib.layers.variance_scaling_initializer()
            if b_init is None:
                b_init = tf.constant_initializer()

            w = tf.get_variable('W', filter_shape, initializer=w_init)
            b = None

            if use_bias:
                b = tf.get_variable('b', [out_channel], initializer=b_init)

            if split == 1:
                conv = tf.nn.conv2d(inputdata, w, strides, padding,
data_format=data_format)
            else:
                inputs = tf.split(inputdata, split, channel_axis)
                kernels = tf.split(w, split, 3)
                outputs = [tf.nn.conv2d(i, k, strides, padding,
data_format=data_format)
                           for i, k in zip(inputs, kernels)]
                conv = tf.concat(outputs, channel_axis)

            ret = nl(tf.nn.bias_add(conv, b, data_format=data_format) if use_bias
else conv, name=name)

        return ret

    @staticmethod
    def relu(inputdata, name=None):
        """

        :param name:
        :param inputdata:
        :return:
        """
        return tf.nn.relu(features=inputdata, name=name)

    @staticmethod
    def sigmoid(inputdata, name=None):
        """

        :param name:
        :param inputdata:
        :return:
        """
        return tf.nn.sigmoid(x=inputdata, name=name)

    @staticmethod
    def maxpooling(inputdata, kernel_size, stride=None, padding='VALID',
data_format='NHWC', name=None):
        """

        :param name:
```

```
        :param inputdata:
        :param kernel_size:
        :param stride:
        :param padding:
        :param data_format:
        :return:
        """
        padding = padding.upper()

        if stride is None:
            stride = kernel_size

        if isinstance(kernel_size, list):
            kernel = [1, kernel_size[0], kernel_size[1], 1] if data_format ==
'NHWC' else \
                [1, 1, kernel_size[0], kernel_size[1]]
        else:
            kernel = [1, kernel_size, kernel_size, 1] if data_format == 'NHWC'
else [1, 1, kernel_size, kernel_size]

        if isinstance(stride, list):
            strides = [1, stride[0], stride[1], 1] if data_format == 'NHWC' else
[1, 1, stride[0], stride[1]]
        else:
            strides = [1, stride, stride, 1] if data_format == 'NHWC' else [1, 1,
stride, stride]

        return tf.nn.max_pool(value=inputdata, ksize=kernel, strides=strides,
padding=padding,
                              data_format=data_format, name=name)

    @staticmethod
    def squeeze(inputdata, axis=None, name=None):
        """

        :param inputdata:
        :param axis:
        :param name:
        :return:
        """
        return tf.squeeze(input=inputdata, axis=axis, name=name)
```

cnn_model.py – модуль содержит модель нейронной сети

```
import numpy as np
import tensorflow as tf
from tensorflow.contrib import layers as tflayers
from tensorflow.contrib import rnn

from crnn_model import cnn_basenet


class ShadowNet(cnn_basenet.CNNBaseModel):
    """
        Implement the crnn model for squence recognition
    """
    def __init__(self, phase, hidden_nums, layers_nums, seq_length, num_classes):
        """

        :param phase:
        """
        super(ShadowNet, self).__init__()
```

```
        self.__phase = phase
        self.__hidden_nums = hidden_nums
        self.__layers_nums = layers_nums
        self.__seq_length = seq_length
        self.__num_classes = num_classes
        return

    @property
    def phase(self):
        """

        :return:
        """
        return self.__phase

    @phase.setter
    def phase(self, value):
        """

        :param value:
        :return:
        """
        if not isinstance(value, str):
            raise TypeError('value should be a str \'Test\' or \'Train\'')
        if value.lower() not in ['test', 'train']:
            raise ValueError('value should be a str \'Test\' or \'Train\'')
        self.__phase = value.lower()
        return

    def __conv_stage(self, inputdata, out_dims, name=None):
        """
        Traditional conv stage in VGG format
        :param inputdata:
        :param out_dims:
        :return:
        """
        conv = self.conv2d(inputdata=inputdata, out_channel=out_dims,
kernel_size=3, stride=1, use_bias=False, name=name)
        relu = self.relu(inputdata=conv)
        max_pool = self.maxpooling(inputdata=relu, kernel_size=2, stride=2)
        return max_pool

    def __feature_sequence_extraction(self, inputdata):
        """
        Implement the 2.1 Part Feature Sequence Extraction
        :param inputdata: eg. batch*32*100*3 NHWC format
        :return:
        """
        conv1 = self.__conv_stage(inputdata=inputdata, out_dims=64, name='conv1')
# batch*16*50*64
        conv2 = self.__conv_stage(inputdata=conv1, out_dims=128, name='conv2')  #
batch*8*25*128
        conv3 = self.conv2d(inputdata=conv2, out_channel=256, kernel_size=3,
stride=1, use_bias=False, name='conv3')  # batch*8*25*256
        relu3 = self.relu(conv3) # batch*8*25*256
        conv4 = self.conv2d(inputdata=relu3, out channel=256, kernel size=3,
stride=1, use_bias=False, name='conv4')  # batch*8*25*256
        relu4 = self.relu(conv4)  # batch*8*25*256
        max_pool4 = self.maxpooling(inputdata=relu4, kernel_size=[2, 1],
stride=[2, 1], padding='VALID')  # batch*4*25*256
        conv5 = self.conv2d(inputdata=max_pool4, out_channel=512, kernel_size=3,
stride=1, use_bias=False, name='conv5')  # batch*4*25*512
```

```
        relu5 = self.relu(conv5)  # batch*4*25*512
        if self.phase.lower() == 'train':
            bn5 = self.layerbn(inputdata=relu5, is_training=True)
        else:
            bn5 = self.layerbn(inputdata=relu5, is_training=False)  #
batch*4*25*512
        conv6 = self.conv2d(inputdata=bn5, out_channel=512, kernel_size=3,
stride=1, use_bias=False, name='conv6')  # batch*4*25*512
        relu6 = self.relu(conv6)  # batch*4*25*512
        if self.phase.lower() == 'train':
            bn6 = self.layerbn(inputdata=relu6, is_training=True)
        else:
            bn6 = self.layerbn(inputdata=relu6, is_training=False)  #
batch*4*25*512
        max_pool6 = self.maxpooling(inputdata=bn6, kernel_size=[2, 1], stride=[2,
1])  # batch*2*25*512
        conv7 = self.conv2d(inputdata=max_pool6, out_channel=512, kernel_size=2,
stride=[2, 1], use_bias=False, name='conv7')  # batch*1*25*512
        relu7 = self.relu(conv7)  # batch*1*25*512
        return relu7

    def __map_to_sequence(self, inputdata):
        """
        Implement the map to sequence part of the network mainly used to convert
the cnn feature map to sequence used in
        later stacked lstm layers
        :param inputdata:
        :return:
        """
        shape = inputdata.get_shape().as_list()
        assert shape[1] == 1  # H of the feature map must equal to 1
        return self.squeeze(inputdata=inputdata, axis=1)

    def __sequence_label(self, inputdata):
        """
        Implement the sequence label part of the network
        :param inputdata:
        :return:
        """
        with tf.variable_scope('LSTMLayers'):
            # construct stack lstm rcnn layer
            # forward lstm cell
            fw_cell_list = [rnn.BasicLSTMCell(nh, forget_bias=1.0) for nh in
[self.__hidden_nums, self.__hidden_nums]]
            # Backward direction cells
            bw_cell_list = [rnn.BasicLSTMCell(nh, forget_bias=1.0) for nh in
[self.__hidden_nums, self.__hidden_nums]]

            stack_lstm_layer, _, _ =
rnn.stack_bidirectional_dynamic_rnn(fw_cell_list, bw_cell_list, inputdata,

dtype=tf.float32)

            if self.phase.lower() == 'train':
                stack_lstm_layer = self.dropout(inputdata=stack_lstm_layer,
keep_prob=0.5)

            [batch_s, _, hidden_nums] = inputdata.get_shape().as_list()  # [batch,
width, 2*n_hidden]
            rnn_reshaped = tf.reshape(stack_lstm_layer, [-1, hidden_nums])  #
[batch x width, 2*n_hidden]
```

```
            w = tf.Variable(tf.truncated_normal([hidden_nums, self.__num_classes],
stddev=0.1), name="w")
            # Doing the affine projection

            logits = tf.matmul(rnn_reshaped, w)

            logits = tf.reshape(logits, [batch_s, -1, self.__num_classes])

            raw_pred = tf.argmax(tf.nn.softmax(logits), axis=2,
name='raw_prediction')

            # Swap batch and batch axis
            rnn_out = tf.transpose(logits, (1, 0, 2), name='transpose_time_major')
# [width, batch, n_classes]

        return rnn_out, raw_pred

    def build_shadownet(self, inputdata):
        """

        :param inputdata:
        :return:
        """
        # first apply the cnn feature extraction stage
        cnn_out = self.__feature_sequence_extraction(inputdata=inputdata)

        # second apply the map to sequence stage
        sequence = self.__map_to_sequence(inputdata=cnn_out)

        # third apply the sequence label stage
        net_out, raw_pred = self.__sequence_label(inputdata=sequence)

        return net_out
```

train.py – модуль для обучения нейронной сети.

```
"""
Train shadow net script
"""
import os
import tensorflow as tf
import os.path as ops
import time
import numpy as np
import argparse

from crnn_model import crnn_model
from local_utils import data_utils, log_utils
from global_configuration import config


logger = log_utils.init_logger()


def init_args():
    """

    :return:
    """
    parser = argparse.ArgumentParser()
    parser.add_argument('--dataset_dir', type=str, help='Where you store the
dataset')
```

```
    parser.add_argument('--weights_path', type=str, help='Where you store the
pretrained weights')

    return parser.parse_args()


def train_shadownet(dataset_dir, weights_path=None):
    """

    :param dataset_dir:
    :param weights_path:
    :return:
    """
    # decode the tf records to get the training data
    decoder = data_utils.TextFeatureIO().reader
    images, labels, imagenames = decoder.read_features(ops.join(dataset_dir,
'train_feature.tfrecords'),
                                                       num_epochs=None)
    inputdata, input_labels, input_imagenames = tf.train.shuffle_batch(
        tensors=[images, labels, imagenames], batch_size=32, capacity=1000+2*32,
min_after_dequeue=100, num_threads=1)

    inputdata = tf.cast(x=inputdata, dtype=tf.float32)

    # initializa the net model
    shadownet = crnn_model.ShadowNet(phase='Train', hidden_nums=256,
layers_nums=2, seq_length=25, num_classes=37)

    with tf.variable_scope('shadow', reuse=False):
        net_out = shadownet.build_shadownet(inputdata=inputdata)

    cost = tf.reduce_mean(tf.nn.ctc_loss(labels=input_labels, inputs=net_out,
sequence_length=25*np.ones(32)))

    decoded, log_prob = tf.nn.ctc_beam_search_decoder(net_out, 25*np.ones(32),
merge_repeated=False)

    sequence_dist = tf.reduce_mean(tf.edit_distance(tf.cast(decoded[0], tf.int32),
input_labels))

    global_step = tf.Variable(0, name='global_step', trainable=False)

    starter_learning_rate = config.cfg.TRAIN.LEARNING_RATE
    learning_rate = tf.train.exponential_decay(starter_learning_rate, global_step,
                                               config.cfg.TRAIN.LR_DECAY_STEPS,
config.cfg.TRAIN.LR_DECAY_RATE,
                                               staircase=True)
    update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)

    with tf.control_dependencies(update_ops):
        optimizer =
tf.train.AdadeltaOptimizer(learning_rate=learning_rate).minimize(loss=cost,
global_step=global_step)

    # Set tf summary
    tboard_save_path = 'tboard/shadownet'
    if not ops.exists(tboard_save_path):
        os.makedirs(tboard_save_path)
    tf.summary.scalar(name='Cost', tensor=cost)
    tf.summary.scalar(name='Learning_Rate', tensor=learning_rate)
    tf.summary.scalar(name='Seq_Dist', tensor=sequence_dist)
    merge_summary_op = tf.summary.merge_all()
```

```
    # Set saver configuration
    saver = tf.train.Saver()
    model_save_dir = 'model/shadownet1'
    if not ops.exists(model_save_dir):
        os.makedirs(model_save_dir)
    train_start_time = time.strftime('%Y-%m-%d-%H-%M-%S',
time.localtime(time.time()))
    model_name = 'shadownet_{:s}.ckpt'.format(str(train_start_time))
    model_save_path = ops.join(model_save_dir, model_name)

    # Set sess configuration
    sess_config = tf.ConfigProto()
    sess_config.gpu_options.per_process_gpu_memory_fraction =
config.cfg.TRAIN.GPU_MEMORY_FRACTION
    sess_config.gpu_options.allow_growth = config.cfg.TRAIN.TF_ALLOW_GROWTH

    sess = tf.Session(config=sess_config)

    summary_writer = tf.summary.FileWriter(tboard_save_path)
    summary_writer.add_graph(sess.graph)

    # Set the training parameters
    train_epochs = config.cfg.TRAIN.EPOCHS

    with sess.as_default():
        if weights_path is None:
            logger.info('Training from scratch')
            init = tf.global_variables_initializer()
            sess.run(init)
        else:
            logger.info('Restore model from {:s}'.format(weights_path))
            saver.restore(sess=sess, save_path=weights_path)

        coord = tf.train.Coordinator()
        threads = tf.train.start_queue_runners(sess=sess, coord=coord)

        for epoch in range(train_epochs):
            _, c, seq_distance, preds, gt_labels, summary = sess.run(
                [optimizer, cost, sequence_dist, decoded, input_labels,
merge_summary_op])

            # calculate the precision
            preds = decoder.sparse_tensor_to_str(preds[0])
            gt_labels = decoder.sparse_tensor_to_str(gt_labels)

            accuracy = []

            for index, gt_label in enumerate(gt_labels):
                pred = preds[index]
                totol_count = len(gt_label)
                correct_count = 0
                try:
                    for i, tmp in enumerate(gt_label):
                        if tmp == pred[i]:
                            correct count += 1
                except IndexError:
                    continue
                finally:
                    try:
                        accuracy.append(correct_count / totol_count)
                    except ZeroDivisionError:
```

```
                        if len(pred) == 0:
                            accuracy.append(1)
                        else:
                            accuracy.append(0)
                accuracy = np.mean(np.array(accuracy).astype(np.float32), axis=0)
                #
                if epoch % config.cfg.TRAIN.DISPLAY_STEP == 0:
                    logger.info('Epoch: {:d} cost= {:9f} seq distance= {:9f} train
accuracy= {:9f}'.format(
                        epoch + 1, c, seq_distance, accuracy))

                summary_writer.add_summary(summary=summary, global_step=epoch)
                saver.save(sess=sess, save_path=model_save_path, global_step=epoch)

            coord.request_stop()
            coord.join(threads=threads)

        sess.close()

        return


if __name__ == '__main__':
    # init args
    args = init_args()

    train_shadownet(args.dataset_dir,args.weights_path)
    print('Done')
```

test.py – модуль для работы с нейронной сетью

```
"""
Use shadow net to recognize the scene text
"""
import tensorflow as tf
import os.path as ops
import numpy as np
import cv2
import argparse
import matplotlib.pyplot as plt
try:
    from cv2 import cv2
except ImportError:
    pass

from crnn_model import crnn_model
from global_configuration import config
from local_utils import log_utils, data_utils
from detect import text_detect
import os
logger = log_utils.init_logger()

def init_args():
    """

    :return:
    """
    parser = argparse.ArgumentParser()
    parser.add_argument('--image_path', type=str, help='Where you store the
image')
    parser.add_argument('--weights_path', type=str, help='Where you store the
weights')
```

```
    return parser.parse_args()

def recognize(image_path, weights_path, is_vis=True):
    """
    :param image_path:
    :param weights_path:
    :param is_vis:
    :return:
    """
    image = cv2.imread(image_path, cv2.IMREAD_COLOR)
    image = cv2.resize(image, (100, 32))
    image = np.expand_dims(image, axis=0).astype(np.float32)

    inputdata = tf.placeholder(dtype=tf.float32, shape=[1, 32, 100, 3],
name='input')

    net = crnn_model.ShadowNet(phase='Test', hidden_nums=256, layers_nums=2,
seq_length=25, num_classes=37)

    with tf.variable_scope('shadow'):
        net_out = net.build_shadownet(inputdata=inputdata)

    decodes, _ = tf.nn.ctc_beam_search_decoder(inputs=net_out,
sequence_length=25*np.ones(1), merge_repeated=False)

    decoder = data_utils.TextFeatureIO()

    # config tf session
    sess_config = tf.ConfigProto()
    sess_config.gpu_options.per_process_gpu_memory_fraction =
config.cfg.TRAIN.GPU_MEMORY_FRACTION
    sess_config.gpu_options.allow_growth = config.cfg.TRAIN.TF_ALLOW_GROWTH

    # config tf saver
    saver = tf.train.Saver()

    sess = tf.Session(config=sess_config)

    with sess.as_default():
        saver.restore(sess=sess, save_path=weights_path)

        preds = sess.run(decodes, feed_dict={inputdata: image})

        preds = decoder.writer.sparse_tensor_to_str(preds[0])

        logger.info('Predict image {:s} label
{:s}'.format(ops.split(image_path)[1], preds[0]))
        if is_vis:
            plt.figure('CRNN Model Demo')
            plt.imshow(cv2.imread(image_path, cv2.IMREAD_COLOR)[:, :, (2, 1, 0)])
            plt.show()

        sess.close()

    return

if __name__ == '__main__':
    # Inti args
    args = init_args()

    path = args.image_path
```

```
    rect = text_detect(path)
    # recognize the image
    labels = ''
    files =
os.listdir("C:\\Users\\User\\PycharmProjects\\Project\\tools\\temp1\\")
    for file in files:
        recognize(image_path=file, weights_path=args.weights_path)
```

detect.py – модуль для выявления областей на изображении

```
import os.path as ops
from PIL import Image
import cv2
import os
import shutil

k = 0.3
def clear_directory(dir):
    for root, dirs, files in os.walk(dir):
        for f in files:
            os.unlink(os.path.join(root, f))
        for d in dirs:
            shutil.rmtree(os.path.join(root, d))

def text_detect(file_name,ele_size=(8,3)): #
    clear_directory('temp1')
    img = cv2.imread(file_name)
    img = cv2.resize(img, (0, 0), fx=k, fy=k)
    if len(img.shape)==3:
        img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img_sobel = cv2.Sobel(img,cv2.CV_8U,1,0
    img_threshold =
cv2.threshold(img_sobel,0,255,cv2.THRESH_OTSU+cv2.THRESH_BINARY)
    element = cv2.getStructuringElement(cv2.MORPH_RECT,ele_size)
    img_threshold = cv2.morphologyEx(img_threshold[1],cv2.MORPH_CLOSE,element)
    contours = cv2.findContours(img_threshold,0,1)
    Rect = [cv2.boundingRect(i) for i in contours[1] if i.shape[0]>100]
    RectP = [(int(i[0]-i[2]*0.08),int(i[1]-
i[3]*0.08),int(i[0]+i[2]*1.1),int(i[1]+i[3]*1.1)) for i in Rect]
    ind=0
    for i in RectP:
        image_obj = Image.open(file_name)
        cropped_image = image_obj.crop(int(ind /k) for ind in i)
        path = 'temp1\\' + str(ind) + '.jpg'
        cropped_image = cropped_image.convert('L')
        cropped_image.save(path)
        ind = ind + 1
    return RectP

if __name__ == '__main__':
    file_name = 'path to image'
    captch_ex(file_name)
    img1 = cv2.imread(path)
    img1 = cv2.resize(img1, (0, 0), fx=1, fy=1)
    rect = text_detect(path)
    for i in rect:
        i = tuple([int(x/k) for x in i])
        cv2.rectangle(img1,i[:2],i[2:], (0, 255, 0))
    # cv2.imwrite('img-out.png')
    cv2.imshow('captcha_result1', img1)
    cv2.waitKey()
```