МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра анализа данных и искусственного интеллекта

КУРСОВАЯ РАБОТА

ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ МАШИННОГО ОБУЧЕНИЯ В СЕГМЕНТЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Работу выполнил _____ И. Е. Романов
(подпись)

Направление подготовки 01.04.02 Прикладная математика и информатика

Направленность (профиль) Математическое и информационное обеспечение экономической деятельности

Научный руководитель
д-р. пед. наук,
канд. физ.-мат. наук, проф. _____ С. В. Юнов
(подпись)

Нормоконтролер
канд. физ.-мат. наук, доц. _____ Г.В. Калайдина
(подпись)

Краснодар
2021

РЕФЕРАТ

Курсовая работа 32 с., 8 рис., 11 источников.

ТЕХНОЛОГИИ МАШИННОГО ОБУЧЕНИЯ, НЕЙРОННЫЕ СЕТИ, ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА, ФОНДОВЫЙ РЫНОК, БИРЖА, МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ, iOS, SWIFT, CREATE ML, CORE ML

Цель работы – реализация приложения для мобильного устройства, которое помогает частным инвесторам быстро получать информацию о текущем общественном мнении рынка относительно конкретной компании на фондовом рынке.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В последние несколько лет в России наблюдается бум частных инвестиций. Как следствие, в сфере появляется достаточно много новых людей, а порог входа для них достаточно высокий. Без определенного опыта разобраться в многообразии торгующихся на рынке компаний не просто.

Такой рост интереса к акциям и облигациям со стороны именно частных инвесторов рождает потребность в инструментах получения информации в новом виде, которые позволили бы максимально обезопасить свои вложения на бирже.

1 Общие концепции программного продукта

1.1 Описание бизнес-процесса, автоматизируемого при помощи программной системы, и экономическая актуальность решения

Представленная программная система представляет собой решение для автоматизации работы по сбору сведений о компаниях на фондовом рынке с целью дальнейшего вложения средств и получения прибыли. Она переводит на новый уровень бизнес-процесс, осуществляемый вручную.

Описание бизнес-процесса: инвестор подбирает компании для инвестирования, исходя из трех аспектов:

– Информация о деятельности самой компании: производится поиск таких компаний, которые создают новые продукты и услуги, меняющие правила игры, квартальным и годовым ростом прибыли не менее определенного порогового значения. Также ищут еще неприбыльные компании, недавно вышедшие на рынок, которые генерируют огромную выручку.

– Общественное мнение о компании (настоящие и будущие клиенты, покупатели продукции или самих акций, инфлюенсеры, общество в целом).

– Исследование спроса и предложеня на сами акции, исследование биржевых графиков и тенденций.

При ручном поиске и исследовании такой информации инвестору пропустить через себя и проанализировать такой объем информации, который доступен для автоматизированного поиска в Сети и анализа нейросетевыми инструментами, корректно не представляется возможным за конечное время.

1.2 Цели и задачи работы

Целью работы является реализация приложения для мобильного устройства Predictor, которое помогает инвесторам быстро получать информацию о мнении рынка относительно конкретной компании посредствам оценки содержания постов социальной сети Twitter.

Предполагается сделать упор на реализации продукта, готового к использованию в виде конечного решения, способного помочь начинающим игрокам рынка.

Итак, общей задачей работы была разработка мобильного приложения для автоматизированого сбора информации о компаниях на фондовом рынке.

В ее составе были выделены следущие подзадачи:

– Формирование датасета сообщений необходимого размера: несколько тысяч штук.

– Подбор необходимых параметров и обучение при помощи фреймворка CreateML модели, готовой к интеграции в мобильное приложение для платформы iOS.

– Получение аккаунта разработчика на платформе Twitter Developer и внедрение Twitter API в приложение. Выгрузка сообщений (твитов) группами.

– Разработка и тестирование вывода нейронной сети для этих данных на основе инструмента Core ML 2 и модуля Natural Language Processing.

– Имплементация графического интерфейса пользователя и отображение результатов вывода.

– Тестирование продукта, улучшение точности модели, обработка возможных сбоев и исключительных ситуаций.

1.3 Сценарий работы готовой программной системы

Предполагается реализация программной системы в виде конечного продукта – мобильного приложения. Программная система начинает свою работу, когда пользователь вводит название компании, идентификатор компании в Nasdaq или юзернейм в Twitter.

Далее происходит сопоставление рыночного идентификатора Nasdaq, юзернейма аккаунта компании в twitter и ее названия. Здесь производится проверка, введено ли валидное значение одного из этих трех параметров, и далее, соответствует ли компания идентификатору в списке Nasdaq.

Затем происходит выгрузка последних твитов о компании, ограниченное неким пороговым значением. Также нужно установить некое нижнее пороговое значение на количество твитов. Если их общее количество будет меньше данного порога, то будем считать, что «общественное мнение» составить нельзя.

Если твитов достаточно, то они все подаются на вход (как текстовый датасет) нейросети, обученной на размеченных данных (также твитах пользователей о крупной компании), которая присваивает каждому сообщению (-1) негативный, (+1) позитивный или (0) нейтральный индекс.

Таким образом формируется общий индекс отношения пользователей к компании.

Параллельно происходит выгрузка текущей динамики цен на акции данной компании на рынке из публичного API.

Эти данные подаются на вход также нейросети, которая на основе краткосрочных и долгосрочных изменений прогнозирует цену на акции компании для заданного промежутка времени (выбираемого из короткого промежутка грядущих дат).

В итоге эти значения составляют общий индекс, прогнозирующий перспективность и благоприятность инвестиций в акции данной компании.

Результат будет наглядно представлен пользователю в интерфейсе приложения.

Для наглядного представления программной системы, на рисунке 1 представлена ее функциональная диаграмма в виде черного ящика.

Рисунок 1 – Функциональная диаграмма автоматизированной системы (черный ящик)

Для наглядного представления программной системы, на рисунке 1 представлена ее функциональная диаграмма в виде черного ящика.

Произведём декомпозицию бизнес-процесса на подпроцессы (рисунок 2). Здесь выделены три основные ветви функциональных процессов программной системы.

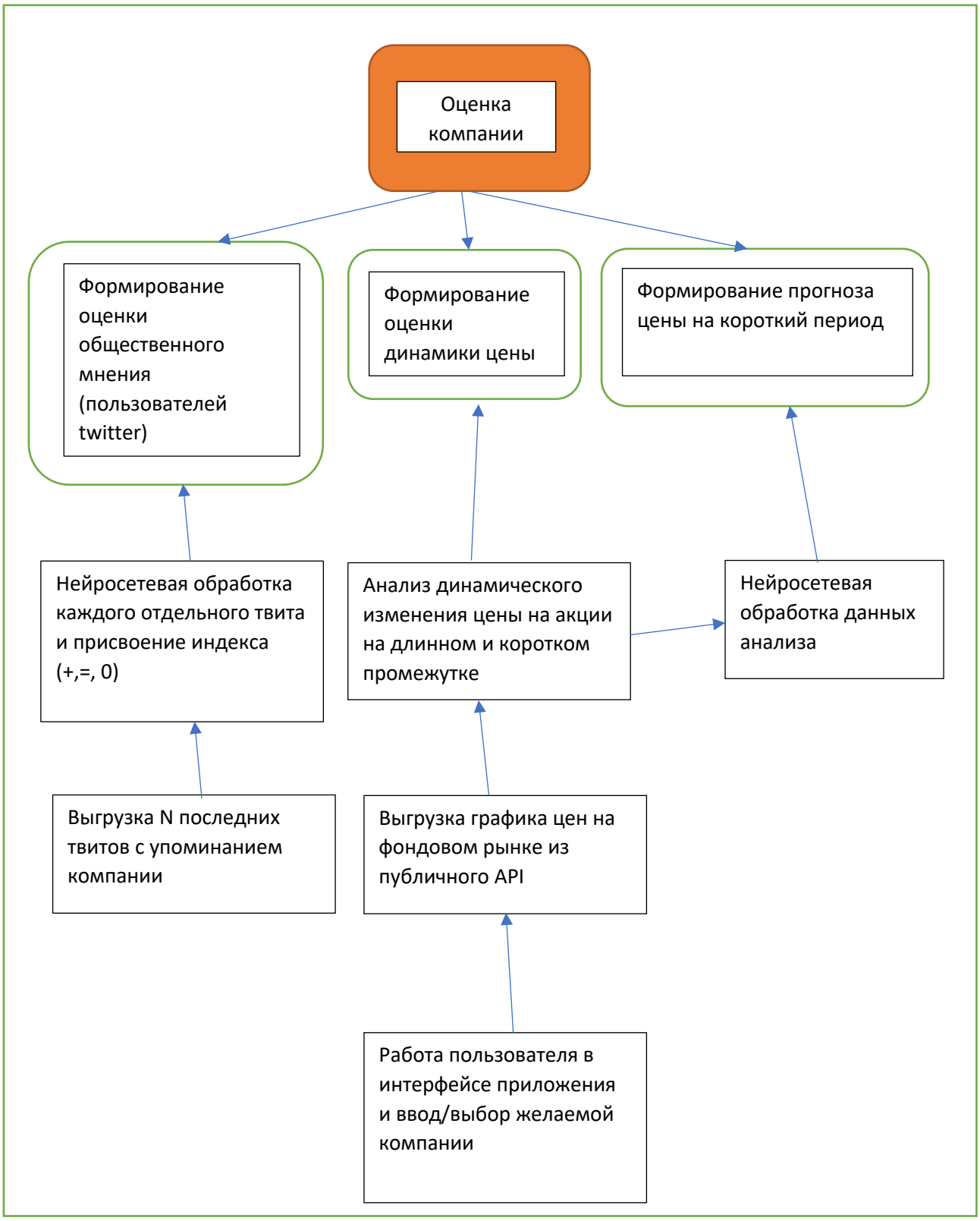

Рисунок 2 – Подпроцессы функционирования программной системы

1.4 Выбор формы реализации и платформы разработки, обоснование выбора

В качестве платформы конечной реализации проектируемой программной системы были выбраны мобильные устройства, а именно – смартфоны. В качестве аргументов в пользу данной альтернативы против других решений (веб-сервисов, десктопных приложений и т.п.) можно привести слудущее:

– Портативность и возможность использовать продукт в любых условиях

– Возможность быстро донести продукт до конечного пользователя

– Общий коммерческий успех рынка мобильных приложений

О выборе мобильной между нативной и кроссплатформенной мобильной разработкой, а также выборе конкретной операционной системы речь пойдет в следующем разделе.

1.5 Платформа и технические инструменты разработки

Для разработки приложения была выбрана мобильная платформа iOS, а также среда Xcode. Для аргументации выбора приведем подробное описание этого продукта.

Xcode - это интегрированная среда разработки (IDE), предоставляемая Apple сторонним программистам, таким как вы, для разработки приложений для iOS и OS X, написанных на Objective-C и Swift. IDE – это программа (или набор программ), которая включает в себя инструменты для различных аспектов создания программного обеспечения. Xcode, в частности, объединяет редактор кода, компилятор, отладчик, конструктор интерфейса, упаковщик приложений и симулятор (технически это отдельная программа, запускаемая из Xcode).

Компилятор Xcode – это программа, которая переводит код из Objective-C или Swift на машинный язык, родной язык процессора. Xcode также содержит прекомпилятор, который сканирует код по мере его ввода на предмет синтаксических ошибок и других проблем, чтобы уменьшить вероятность того, что ваша сборка содержит код, который вызовет сбой.

Отладчик – отличительная особенность среды. Руководящие указания и предварительные компиляторы «Good Practice» помогают снизить вероятность написания ошибок. Хорошие инструменты отладки позволяют программисту сканировать код во время выполнения процессов, наблюдая за воздействием элементов приложения и изменяя себя, чтобы увидеть, где они сталкиваются.

Interface Builder - в XCode это название инструмента, который позволяет использовать раскадровки для создания визуального макета пользовательского интерфейса (UI) аналогично программному обеспечению для проектирования и верстки.

Application Bundler - собирает программу в виде пакета приложения для отправки потребителю и распространения. Xcode содержит функциональные возможности для отправки вашего приложения в App Store для проверки. Вы можете в конечном итоге сделать это самостоятельно с побочными проектами.

Симулятор - Xcode содержит компоненты для частичной репликации условий запуска вашего приложения на мобильном устройстве. Для разработки очень полезно увидеть, как код может вести себя после запуска на реальном устройстве, но по ряду причин он не может быть вполне реальным, поэтому он не может заменить тестирование вашей сборки на реальном мобильном устройстве, в частности в случае разработки AR-приложений по причине отсутствия камеры.

Также среду разработки от Apple выгодно отличают:

а) автодополнение,

б) мгновенное исправление,

в) возможность тестирования кода в интерфейсе «игровой площадки» с мгновенным результом прекомпиляции,

г) перетаскивание элементов интерфейс,

д) наличие официальной документации и онлайн-ресурсов для обучения непосредственно от Apple.

1.5.1 Выбор языка реализации

Существуют лишь два языка разработки для iOS: Objective-C и Swift. По причине морального устаревания первого остановимся на Swift.

Объявленный в 2014 году, язык программирования Swift быстро стал одним из самых быстрорастущих языков в истории. Его использование облегчает написание программного обеспечения, которое, по заявлениям Apple, становится невероятно быстрым и безопасным. Цели компании для Swift амбициозны: сделать программирование простых вещей легким, а трудных - возможным.[7]

Для студентов изучение Swift стало отличным введением в современные концепции программирования и лучшие практики. А поскольку язык свободно распространяемый, навыки Swift смогут быть применены к еще более широкому кругу платформ, от мобильных устройств до настольных компьютеров и облачных вычислений.

Цель проекта Swift - создать наилучший доступный язык для различных применений: от системного программирования до мобильных и настольных приложений, с масштабированием до облачных сервисов. Самое главное, Swift разработан для того, чтобы облегчить разработчику написание и сопровождение программ. Для достижения этой цели разработчики проекта провозглашабт основные принципы языка и разработки на нем:

а) Безопасность. Наиболее очевидный способ написания кода также должен быть ориентирован на безопасность. Неопределенное поведение является врагом безопасности, и ошибки разработчиков должны быть

обнаружены до того, как программное обеспечение будет запущено. Выбор безопасности иногда означает, что Swift будет слишком строг к разработчику, но, по мнению создателей, ясность экономит время в долгосрочной перспективе.

б) Скорость. Swift предназначен для замены языков на основе C (C, C ++ и Objective-C). Таким образом, Swift должен быть сопоставим с этими языками по производительности для большинства задач. Производительность также должна быть предсказуемой и последовательной, а не только быстрой в коротких пакетах, которые требуют очистки позже. Есть много языков с современным функционалом – но быстрыми они бывают редко.

в) Выразительность. Swift извлекает выгоду из десятилетий прогресса в области компьютерных наук, предлагая синтаксис, который приятно использовать, при это с набором современных функций, которых ожидают разработчики. Но проект никогда не будет завершен. Создатели следят за прогрессом языка и совершенствуют структуру, чтобы сделать Swift еще лучше.

Набор нструментов разработчика являются важной частью экосистемы Swift. Интегрируются инструменты для быстрой сборки, диагностики и возможностей интерактивной разработки. Такой набор может сделать программирование намного более мощным, как это делают игровые площадки на базе Swift в Xcode, или REPL на основе веб – при работе с серверным кодом Linux.

С самого начала Swift был спроектирован так, чтобы быть более безопасным, чем языки на основе C, и исключать распространенные случаи небезопасного кода. Автоматическое управление памятью, прединиализация переменных, проверки на переполнение и выход за границы массивов позволяют сосредоточиться на смысле, а не на синтаксисе**.** Построение синтаксиса таково, чтобы было проще определить задумку программиста – большинство ключевых слов состоят всего из трех символов.

Еще одна особенность безопасности заключается в том, что по умолчанию объекты Swift никогда не могут иметь значение nil, а попытка создать или использовать объект nil приведет к ошибке во время компиляции. Это делает написание кода намного чище и безопаснее, и предотвращает общую причину сбоев во время выполнения. Тем не менее, есть случаи, когда nil случается, и для этих ситуаций Swift имеет инновационную функцию, известную как опционалы (optionals). Опционал может содержать nil, но синтаксис Swift заставляет безопасно справиться с ним, используя специальный синтаксис, чтобы указать компилятору, какая обработка значения требуется в данном случае и справиться с ней безопасно.

2 Проектирование программной системы

2.1 Анализ настроений в Twitter для прогнозирования тенденций на фондовом рынке

Прогнозирование движения фондового рынка – хорошо известная область исследования проблема, представляющая интерес. Современные социальные сети прекрасно отражают общественное мнение и мнение о текущих событиях. В частности, Twitter привлек большое внимание исследователей для изучения общественного мнения. Прогнозирование фондового рынка на основе общественных настроений, выраженных в Twitter, остается интригующей областью исследований. Предыдущие исследования пришли к выводу, что совокупное общественное настроение, собранное из сообщений в Twitter, вполне может быть коррелировано с индексом Dow Jones Industrial Average (DJIA). Тезис работы Паголу и Челла [3] состоит в том, чтобы проследить, насколько изменения цен на акции компании, подъемы и падения, коррелируют с общественным мнением, выражаемым в твитах об этой компании. Понимание мнения и эмоции автора по фрагменту текста является целью анализа. В статье были использованы два различных текстовых представления, Word2vec и N-gram, для анализа общественных настроений в твитах. Авторы применили анализ и контролируемые принципы машинного обучения к сообщениям, извлеченным из Twitter, и проанализировали корреляцию между движениями фондового рынка компании и настроениями в твитах. Если провести несложные логические параллели, становится понятным, что позитивные новости и твиты в социальных сетях о компании определенно побудят других игроков инвестировать в ее акции, и в результате цена акций этой компании возрастет. В заключении статьи наглядно показано, что существует сильная корреляция между ростом и падением цен на акции и общественными настроениями в твитах.

2.2 Проект как программная система

На этапе проектирования к проекту как к программной системе должен быть предъявлен ряд требований:

– К требованиям к надежности программной системы можно отнести достаточную точность нейросетевой модели, устойчивость к исключительным ситуациям (необходимо тестирование и их обработка).

– Требования к безопасности практически могут быть опущены, поскольку пользователи не оставляют никаких данные в приложении, а также отсутствует своя серверная среда, атаки и перегрузки исключены.

– Требования к эргономичности ограничиваются интуитивно понятным интерфейсом, степени протестированности и минимальной необходимости в действиях со стороны пользователя.

– Требования к численности и квалификации персонала и режиму его работы: для разработки продукта, учитывая небольшую сложность реализации достаточно одного мобильного разработчика с опытом на рынке. Для последующей поддержки продукта (реагированию на изменения в структуре API, изменению потребности пользователей) достаточно ставки на неполный рабочий день.

Также необходимо выполнить системное описание существующих подобных автоматизированных систем.

Систем, полностью аналогичных представленной на рынке не было найдено, таким образом альтернативой продукту для начинающего инвестора является ручной поиск информации.

Но при ручном поиске и исследовании такой информации инвестору необходимо пропустить через себя и проанализировать такой объем информации, который доступен для автоматизированного поиска в Сети и анализа нейросетевыми инструментами, корректно не представляется возможным за конечное время.

2.2.1 Структура программной системы

Структура системы на данном этапе разработки включает 3 компонента, а именно:

– публичный API твиттера, из которого извлекаются, загружаются данные;

– мобильное приложение для взаимодействия с пользователем и вывода результатов;

– Обученная и интегрированная в приложение при помощи фреймворка CreateML модель CoreML 2.

Представленная структура отражена на рисунке 3.

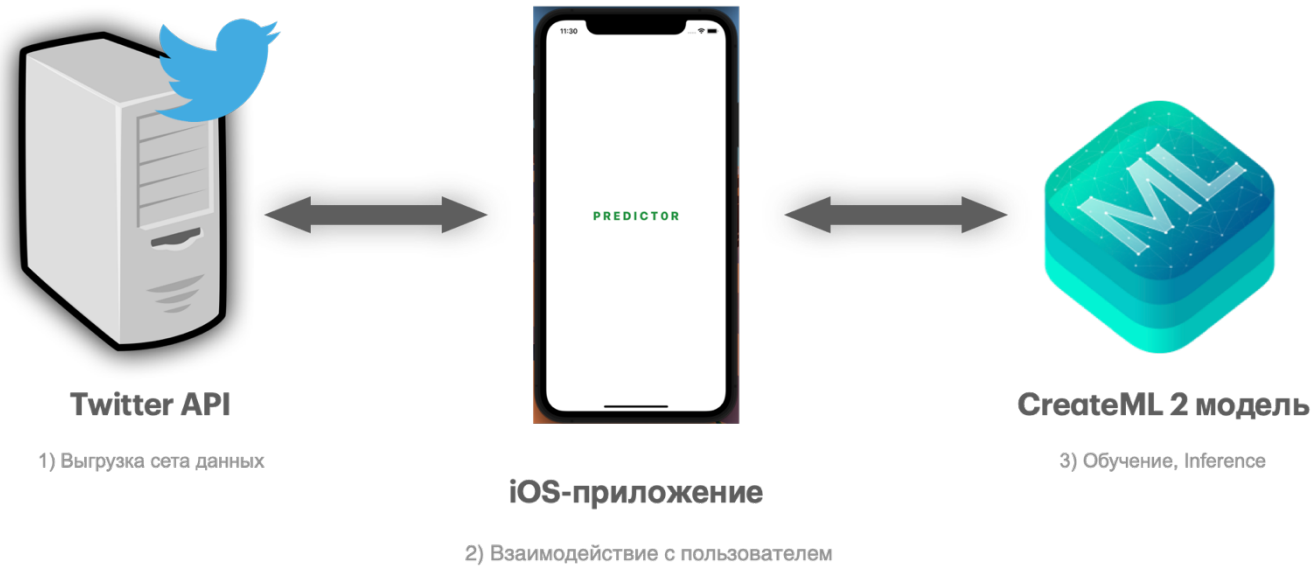

Рисунок 3 – Структура программного продукта

2.3 Получение и подготовка данных

Был найден и дополнен размеченный датасет твитов о популярной компании, этот набор данных не очень большой (3 тысячи строк таблицы), но он оказался достаточным, чтобы обучить алгоритм до оптимальной точности.

Данные из этого набора представлены в виде CSV-файла и размечены 3 видами лейблов для позитивного, негативного и нейтрального окраса сообщения (Рис. 4).

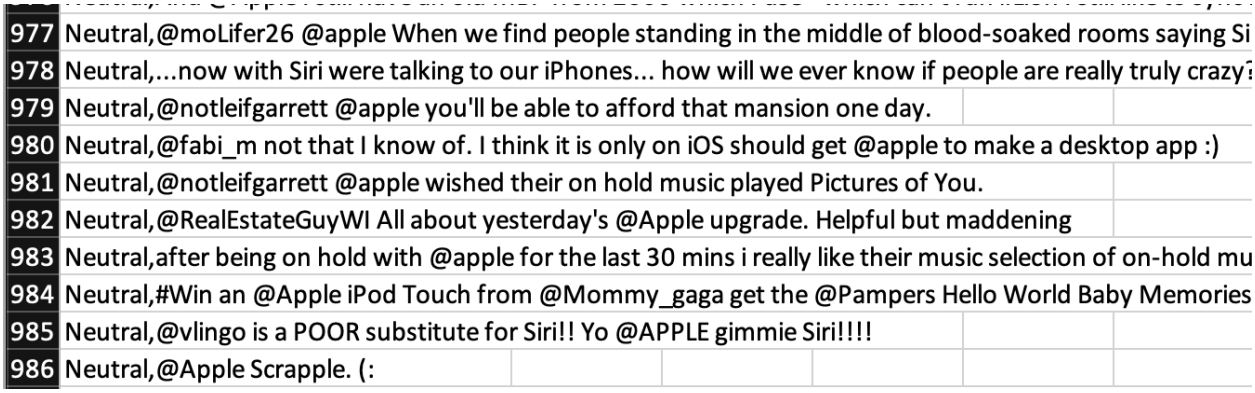

Рисунок 4 – Размеченный датасет для обучения

На этих данных нейронная сеть была обучена при помощи фреймворка Create ML, точность распознавания составила около 75 процентов, что является хорошим результатом для анализа естественного языка.

Данные для процесса вывода нейронной сети загружаются из публичного API Twitter и сохраняются локально (максимально возможное количество данных ограничено – 100 штук), к сожалению твиттер не дает возможности загружать больше за раз. Далее происходит анализ содержимого при помощи локального inference модели при помощи Core ML 2 и выдается оценка для каждого сообщения.

Для доступа к массовой выгрузке сообщений был получен доступ к аккаунту разработчика твиттера. Компания понимает, что массовым использованием нейросетевых инструментов заниматься семантическим анализом содержимого их сообщений может практически кто угодно в своих целях, а такая информация – ценнейший ресурс самой компании. Поэтому Twitter старается ограничить или просто закрыть доступ к своему публичному API почти для всех разработчиков. С этим же связано

ограничение на 100 выгружаемых за раз сообщений. Аккаунт был получен лишь после диалога с поддержкой в течение двух недель.

2.4 Фреймворк CreateML

Далее, подробнее об работе с моделью.

Для обучения модели использовался Create ML – это инструмент, созданный компанией Apple для создания и обучения пользовательских моделей машинного обучения прямо на компьютере Mac.

Create ML используется внутри знакомых разработчику инструментов, таких как Swift и Xcode playgrounds, для создания и обучения пользовательских моделей машинного обучения на Mac. Разработчик может обучать модели выполнять такие задачи, как распознавание изображений, извлечение смысла из текста или нахождение связей между числовыми значениями.

Create ML имеет множество типов моделей на выбор. Есть возможность выбрать тип модели в приложении и добавьте свои данные и параметры, чтобы начать обучение.

Функции Create ML:

– Создание и обучение мощных моделей прямо на устройстве с помощью простого интерфейса.

– Обучение нескольких моделей, с использованием различных наборов данных в пределах одного проекта.

– Оценка производительности модели, с использованием функции Continuity с помощью камеры iPhone и микрофона на компьютере Mac.

– Возможность приостановить, сохранить, возобновить и продлить процесс тренировки.

– Обучение моделей с высокой скоростью прямо на компьютере Mac, используя преимущества процессора и видеокарты.

– Поддержка внешних графических карт с Mac для еще более высокой производительности обучения модели.

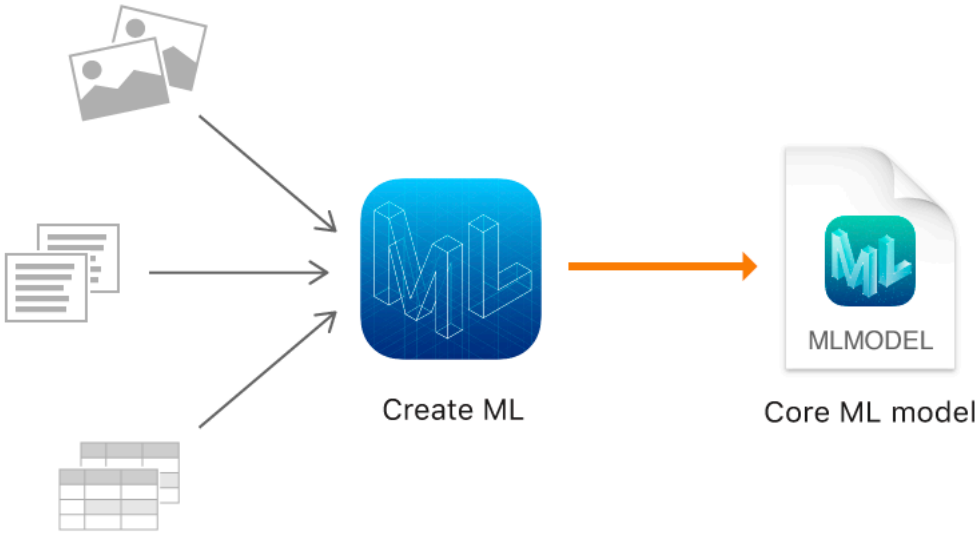

Рисунок 5 – Схема работы Create ML

Взаимодействия Create ML и Core ML между собой показано на рисунке 5.

Поддерживаются следующие типы моделей:

– для работы с изображениями,
– видео,
– физической активностью,
– звуком,
– текстом,
– таблицами.

2.5 Core ML

Core ML – это фреймворк машинного обучения, используемый в продуктах Apple (macOS, iOS, watchOS и tvOS) для быстрого прогнозирования или вывода с простой интеграцией предварительно

обученных моделей машинного обучения на периферии, что позволяет в режиме реального делать прогнозы на тестовых данных прямо на устройстве.

Core ML используется для так называемого «Машинного обучения на периферии» (ML on the edge). Данный подход обладает следующими преимуществами:

1) Низкая задержка и результаты, близкие к реальному времени: не нужно совершать вызов сетевого API, отправляя данные, а затем ожидая ответа. Это может иметь решающее значение для таких приложений, как обработка последовательных кадров с камеры на устройстве.

2) Доступность в автономном режиме, конфиденциальность и привлекательные затраты, поскольку приложение работает без подключения к сети, без вызовов API, а данные никогда не покидают устройство. Позволяет использовать свое мобильное устройство для идентификации исторических плиток в метро, каталогизации частных фотографий с отдыха в режиме полета или обнаружения ядовитых растений в дикой природе.

Недостатки подхода следующие:

1) Размер приложения: Добавляя модель на устройство, вы увеличиваете размер приложения, и некоторые точные модели могут быть довольно большими.

2) Использование системы: Прогнозирование и вывод на мобильном устройстве требуют большого количества вычислений, что увеличивает разрядку батареи. Старые устройства могут с трудом обеспечивать прогнозы в реальном времени.

Обучение модели: В большинстве случаев модель на устройстве должна постоянно обучаться за пределами устройства с использованием новых пользовательских данных. Как только модель будет переобучена, приложение необходимо будет обновить с помощью новой модели, и в зависимости от размера модели это может затруднить передачу данных по сети для пользователя. Обратитесь к проблеме размера приложения,

перечисленной выше, и теперь у нас есть потенциальная проблема с пользовательским интерфейсом.

Учитывая преимущества и необходимость в написании собственного веб-сервера с запущенной там моделью в случае отказа от локального машинного обучения, такой подход оказался оптимальным для задачи реализации программной системы предсказания общественного мнения.

Фреймворк Core ML предлагает широкую функциональность, а именно распознавание изображений в реальном времени, распознавание лиц, анализ текста и идентификация говорящего по голосу, и еще многие инновации, ставшие возможными благодаря машинному обучению с использованием Core ML.

Принцип работы фреймворка следующий: Core ML использует модель машинного обучения, которая предварительно обучается в облаке, затем преобразуется в формат Core ML и добавляется непосредственно в проект Xcode.

2.5.1 Обработка естественного языка в Core ML

Для работы с моделью использовался модуль “Natural Language Processing”.

Структура MLTextClassifier представляет классификатор текста для обучения ML-модели, которую затем можно включить в свое приложение для классификации текста на естественном языке. Модель учится связывать метки с элементами входного текста, которые могут быть предложениями, абзацами или даже целыми документами.

После обучения текстового классификатора модель сохраняется в файл модели Core ML. Затем используется экземпляр класса NLModel из среды естественного языка для чтения файла модели в свое приложение.

MLTextClassifier предоставляет три типа алгоритмов обработки естественного языка:

1) CRF (Conditional Random Field) – модель условного случайного поля. Условные случайные поля — дискриминативная ненаправленная вероятностная графическая модель, являющаяся разновидностью марковских случайных полей (Markov random field). Наиболее распространенной в применении является модель линейного CRF (linear chain CRF). Данная модель чаще всего применяется для решения задач разметки и сегментации последовательностей. CRF — это алгоритм машинного обучения с учителем. Линейный CRF хорошо подходит для решения задач сегментации и разметки последовательности, например:

- автоматическое выделение (именованных) сущностей
- анализ тональности
- частеречное тэгирование
- автоматическое распознавание речи

2) Max Ent (maximum entropy model) – модель максимальной энтропии. В статистике называемая марковская модель с максимальной энтропией (MEMM), или же условная марковская модель (CMM), это графическая модель маркировки последовательности, которая сочетает в себе черты скрытых марковских моделей (HMMs) и модели максимальной энтропии (MaxEnt). MEMM – это дискриминационная модель, которая расширяет стандарт классификатора максимальной энтропии, предполагая, что неизвестные значения, которые должны быть изучены, связаны в Цепь Маркова, а не являются условно независимыми друг от друга. MEMM находят применение в обработке естественного языка, особенно в сфере распознавания речи и извлечения информации.

3) Transfer Learning (A transfer learning model) – трансферное обучение. Идея трансферного обучения строится на том, что знания, накопленные в модели, подготовленной для выполнения одной задачи — скажем, распознавания цветов на фотографии — могут быть перенесены на другую модель, чтобы помочь в построении прогнозов для другой, родственной задачи, например, для задачи выявления меланомы.

Существуют различные подходы к трансферному обучению, но один из них — тонкая настройка (finetuning) — находит особенно широкое применение. При таком подходе команда берет предварительно обученную модель и удаляет или переучивает последние слои этой модели, чтобы сфокусироваться на новой, схожей задаче.

В ходе тестирования и переобучения с целью подбора лучших параметров модели оказалось, что самый быстрым для обучения оказался алгоритм максимальной энтропии, а самым точным оказалось трансферное обучение.

3 Результат работы над проектом

3.1 Демонстрация результатов разработки

В ходе работы были выбраны оптимальные инструменты разработки и реализован конечный продукт – готовое iOS-приложение.

Разработка была завершена успешно, и далее на рисунках будет продемонстрирован результат последовательного прохождения жизненного цикла продукта.

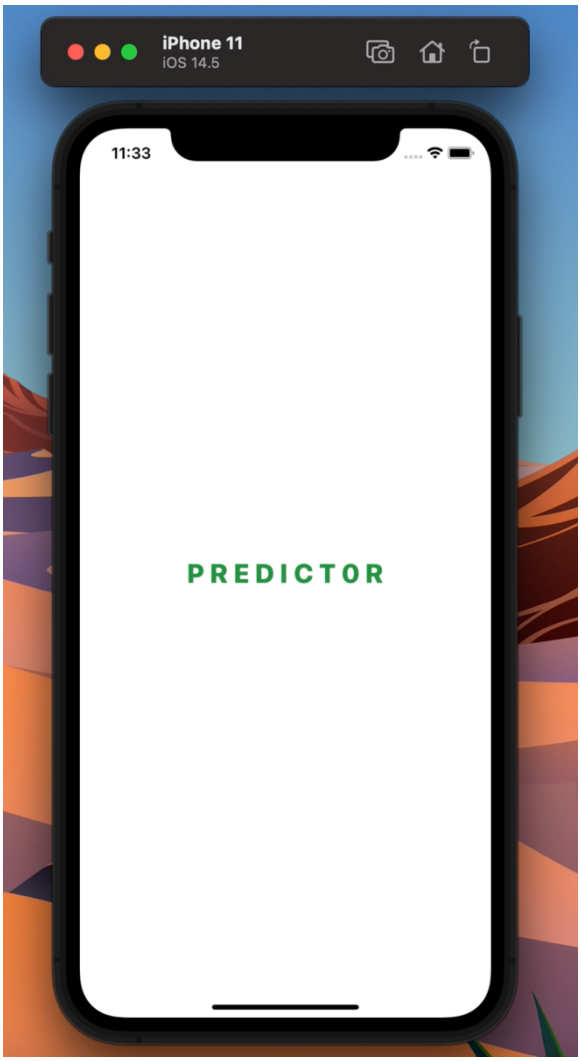

Рисунок 6 – Экран загрузки приложения

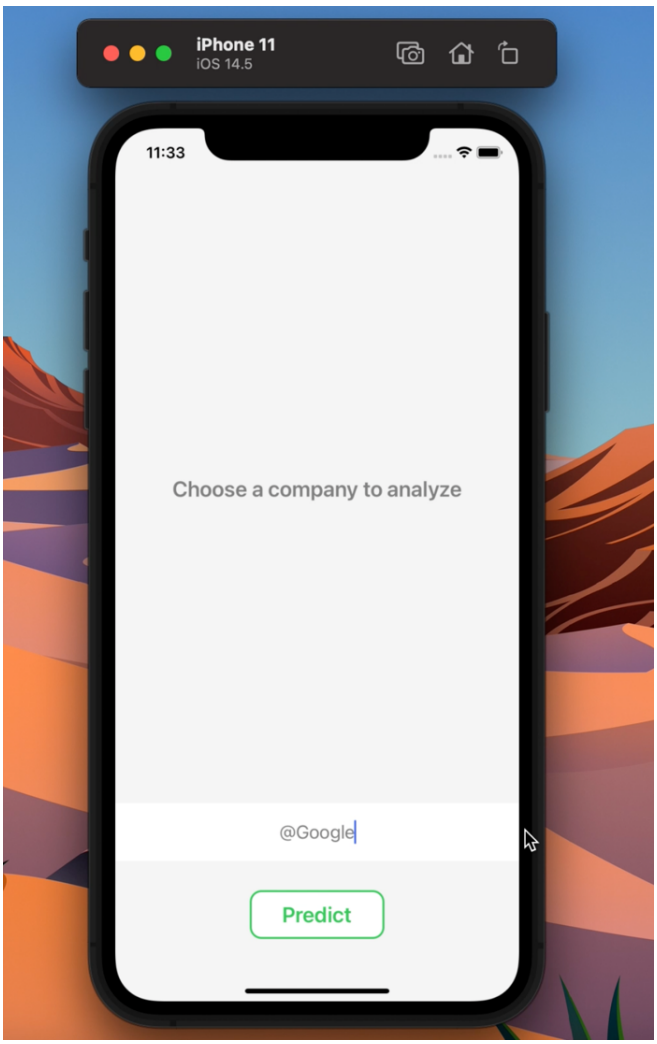

Рисунок 7 – Начальный интерфейс пользователя

Рисунок 6 представляет исходный экран загрузки приложения. Начальный интерфейс после инициализации приложения отображен на рисунке 7. Следующий шаг жизненного цикла представлен на рисунке 8.

Рисунок 8 – Результат предсказания по конкретной компании

3.2 Процесс тестирования и отладки

В процессе тестирования были выявлены следующие исключительные ситуации, ошибки и непредвиденные обстоятельства:

– Отсутствие подключения;

– Истечение ключей Twitter Public API;

– Блокировка по превышению количества запросов;

– Очень низкая точность срабатывания модели для текстов, максимально отличающихся от содержавшихся в обучающем датасете;

– Не найдено твитов для выбранного запроса;

– Неизвестная исключительная ситуация;

– Ошибки на устаревших версиях iOS.

– Неверное отображение интерфейса на нестандартных размерах экрана.

В случае, если любая из исключительных ситуаций произошла, пользователю выводится сообщение с локализованным описанием и модальным окном с кнопками для повторения действия, либо отмены.

3.3 Шаги по дальнейшему совершенствованию продукта

В качестве шагов по дальнейшему расширению функциональности планируется:

– Мультиязычность. В данный момент твиты фильтруются по языку, оставляя только англоязычные сообщения.

– Обход ограничения твиттера по загрузке всего 100 твитов на страницу, загружать несколько страниц параллельно, и затем объединять их в общий тестовый датасет для увеличения выборки мнений пользователей.

– Добавление функциональности выгрузки динамики цены из доступных публичных API, предсказания ее изменений для краткосрочной и долгосрочной перспективы.

– Объединение двух предыдущих пунктов в один алгоритм для достижения принципиально нового уровня предсказания динамики цен.

– Покрытие проекта Unit-тестами, улучшение точности моделей, переработка алгоритмов.

– Совершенствование графического интерфейса на манер современных приложений.

– Сокрытие исходного кода приложения.

– Работа над безопасностью и шифрованием траффика.

– Дистрибьюция приложения в Apple App Store.

– SEO-оптимизация страницы приложения в магазине, реклама и промоутирование приложения при помощи таргентированной рекламы пользователям, начинающим свой путь на инвестиционном рынке.

ЗАКЛЮЧЕНИЕ

В качестве результата курсовой работы была разработана система автоматизированного сбора информации о компаниях на фондовом рынке. Спроектированная структура програмного продукта была ориентирована на подготовку системы для реального пользователя.

В результате изучения и системного представления объекта и процесса автоматизации по выбранной теме были приобретены навыки системного анализа реальных объектов и процессов на предмет автоматизации решаемых задач. Была произведена декомпозиция бизнес-процесса, включающая в себя схематичное представление его этапов. Такой подход к задаче позволил лучше понять структуру программной системы и облегчил процесс ее построения.

Также в результате работы был проведен обоснованный выбор средств разработки специального ПО, разработана структура ПС, проведено обучение и подбор параметров модели машинного обучения, произведена интеграция с мобильным приложением.

Произведено первичное тестирование, отладка параметров модели, увеличение точности вывода, и тестирование приложения на реальных устройствах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Sentiment Analysis of Twitter Data for Predicting Stock Market October 28, 2016 // (Engl.). – URL: https://arxiv.org/abs/1610.09225 [23 March 2021]

2 Apple Inc. App Development with Swift: учебное пособие / Apple Education – Купертино, 2019. – 1027 с. : – URL: https://books.apple.com/us/book/app-development-with-swift/id1219117996 (дата обращения: 03.05.2021).

3 Apple Inc. The Swift Programming Language, / Apple Education – Купертино, 2019. – 500 с. – URL: https://books.apple.com/gb/book/the-swift-programming-language-swift-5-2/id881256329 (дата обращения: 03.05.2021).

4 Twitter Public API [Электронный ресурс]. – Бостон. – URL: https://developer.twitter.com/en/apply-for-access (дата обращения: 03.05.2021).

5 Apple Inc. The Swift Programming Language, / Apple Education – Купертино, 2019. – 500 с. – URL: https://books.apple.com/gb/book/the-swift-programming-language-swift-5-2/id881256329 (дата обращения: 03.05.2021).

6 Документация Apple по Swift: официальный сайт [Электронный ресурс]. – Купертино. – URL: https://swift.org/documentation/ (дата обращения: 5.04.2021)

7 Документация Apple по Core ML официальный сайт [Электронный ресурс]. – Купертино. – URL: https://developer.apple.com/documentation/coreml (дата обращения: 25.01.2021)

8 Grand Central Dispatch Tutorial for Swift 4 [Электронный ресурс]. – Бостон. – URL: https://www.raywenderlich.com/5370-grand-central-dispatch-tutorial-for-swift-4-part-1-2 (дата обращения: 25.01.2021)

9 Getting Started with Core Data Tutorial [Электронный ресурс]. – Бостон. – URL: https://www.raywenderlich.com/7569-getting-started-with-core-data-tutorial (дата обращения: 25.01.2021)

10 Документация Apple по SceneKit: официальный сайт. – Купертино. – URL: https://developer.apple.com/documentation/scenekit (дата обращения: 25.01.2021)

11 Документация Apple по UIKit: официальный сайт. – Купертино. – URL: https://developer.apple.com/documentation/uikit (дата обращения: 02.03.2021)