Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Институт прикладной математики и компьютерных наук

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Руководитель ООП
д-р техн. наук, профессор
_____ К.И. Лившиц
«09» июня 2022 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

АВТОМАТИЗИРОВАННЫЙ ПРОГНОЗ КРАТКОСРОЧНОГО ДВИЖЕНИЯ СТОИМОСТИ ЦЕННЫХ БУМАГ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ

по направлению подготовки 01.03.02 Прикладная математика и информатика,
направленность (профиль) «Математические методы в экономике»

Бикбавлеев Александр Александрович

Руководитель ВКР
канд. техн. наук, доцент
_____ М.Н. Головчинер
«06» июня 2022 г.

Автор работы
студент группы № 931822
_____ А.А. Бикбавлеев
«06» июня 2022 г.

Томск – 2022

Министерство науки и высшего образования Российской Федерации.
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Институт прикладной математики и компьютерных наук

УТВЕРЖДАЮ
Руководитель ООП
д-р техн. наук, профессор
_____ К.И. Лившиц
подпись
«12» ноября 2021 г.

ЗАДАНИЕ

по выполнению выпускной квалификационной работы бакалавра обучающемуся
Бикбавлееву Александру Александровичу
Фамилия Имя Отчество обучающегося
по направлению подготовки 01.03.02 Прикладная математика и информатика, направленность (профиль) «Математические методы в экономике»

1 Тема выпускной квалификационной работы
Автоматизированный прогноз краткосрочного движения стоимости ценных бумаг на основе нейронных сетей

2 Срок сдачи обучающимся выполненной выпускной квалификационной работы:
а) в учебный офис / деканат – 06.06.2022 б) в ГЭК – 09.06.2022

3 Исходные данные к работе:
Объект исследования – Методы и подходы к анализу текстов на предмет их позитивной или негативной эмоциональной окраски
Предмет исследования – Методы и подходы к анализу текстов на предмет их позитивной или негативной эмоциональной окраски
Цель исследования – Создание программного обеспечения для прогнозирования роста или снижения стоимости акций компании на основе анализа текста комментариев пользователей форумов и социальных сетей об компании.

Задачи:
Изучить существующие методы и подходы к анализу текстов. Изучить программные средства автоматического анализа тональности текстов. Выбрать наиболее оптимальные из них для прогнозирования роста или снижения стоимости акций. Разработать программное обеспечение для прогнозирования роста или снижения стоимости акций компании на основе анализа текста комментариев пользователей форумов и социальных сетей об этой компании.

Методы исследования:
Аналитический метод исследования с использованием аппарата теории искусственного интеллекта, нейронных сетей и математической лингвистики.

Организация или отрасль, по тематике которой выполняется работа, –
НИ ТГУ, Институт прикладной математики и компьютерных наук, кафедра компьютерной безопасности

4 Краткое содержание работы:

Разработано программное обеспечение, которое создает базу данных и анализирует тональность комментариев; строит график результатов анализа тональности и накладывает его на график реальных биржевых данных.

Проведен анализ причин расхождений полученных графиков.

Руководитель выпускной квалификационной работы
канд.техн.наук, доцент, НИ ТГУ — / М.Н. Головчинер
должность, место работы — *подпись* — *И.О. Фамилия*

Задание принял к исполнению
студент группы № 931822 НИ ТГУ — / А.А. Бикбавлеев
должность, место работы — *подпись* — *И.О. Фамилия*

АННОТАЦИЯ

Целью данной работы является разработка программного обеспечения для автоматизированного прогноза краткосрочного движения стоимости ценных бумаг компании с использованием программных средств на основе нейронных сетей. В ходе работы изучены существующие методы и подходы к анализу текстов; рассмотрены и применены наиболее оптимальные существующие программные средства (пакеты) автоматического анализа тональности текстов для прогнозирования роста или снижения стоимости акций. На примере комментариев о компании Tesla в социальных сетях выполнен анализ текстов на предмет их позитивной или негативной эмоциональной окраски (анализ тональности); построены графики результатов анализа тональности комментариев с наложением их на графики реальных биржевых данных; проведен анализ возможных причин расхождений полученных графиков.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

С каждым годом все большее количество людей получает возможность доступа к интернету. В глобальной информационной сети люди ищут ответы на свои вопросы, общаются в социальных сетях, делятся мнением на форумах, пишут отзывы о товарах и услугах. В том числе, происходит обмен мнениями о прогнозах роста или снижения стоимости ценных бумаг. И всю эту информацию возможно анализировать в автоматическом режиме.

Обработка текстов на естественном языке в последние годы активно развивается. Особый интерес представляет понимание того, какое ожидание у людей имеется на стоимость акций компании. Если дела у компании идут хорошо, то мнения людей в интернете о компании будут преимущественно позитивные, иначе – негативные. Соответственно возможен рост либо снижение стоимости акций.

Среди участников рынка ценных бумаг имеется потребность в анализе мнений людей в комментариях на форумах и в социальных сетей о той или иной компании для возможности спрогнозировать рост или снижение стоимости акций. Для распознавания мнения пользователя, то есть определения позитивной или отрицательной окраски текста, применяется направление в области искусственного интеллекта, нейронных сетей и математической лингвистики - **Обработка текстов на естественном языке** (Natural Language Processing, NLP).

В представленной работе на примере анализа комментариев о компании Tesla в социальных сетях реализован автоматизированный прогноз краткосрочного движения стоимости ценных бумаг компании с использованием программных средств на основе нейронных сетей.

Tesla (ранее **Tesla Motors**) - одна из перспективных и широко известных компаний. Она была основана в 2003 г. В данный момент компания является одним из крупнейших производителей электромобилей, её дочерняя компания

– SolarCity, является одним из крупнейших производителей солнечных панелей.

Влияние на стоимость акций Tesla имеет мнение обычных людей, акционеров и также владельцев электромобилей Tesla. В популярных социальных сетях в интернете, таких как **Twitter**, **Instagram**, **Facebook**, а также на форум-сайте **Reddit** ведется активное обсуждение той или иной модели электромобиля, качества сервиса, а также смежных с Tesla компаний, таких как SolarCity.

В данной работе для достижения цели используется метод **анализа тональности текстов**.

Данный метод обработки текстов естественного языка осуществляет автоматический анализ эмоциональной окрашенности предоставленного текста и эмоциональной оценки автора к объекту, о котором говориться в тексте. Метод применяется для прогнозирования цен на рынке биржевых акций на основе анализа тональностей текстов в потоке биржевых новостей [1], получения информации об удовлетворенности потребителя [2], распознавания ложной информации из новостных источников [3].

Задача анализа тональности текста состоит из трех этапов: предварительной обработки текста, перевода текста в вещественное пространство признаков и использования методов машинного обучения для последующей классификации тональности.

Предобработка текста – ключевой момент данного процесса, включающий в себя удаление стоп-слов, сегментацию и приведение слов к одной грамматической форме, маркировку частей речи и анализ.

Современные алгоритмы машинного обучения, используемые при решении подобных задач, ориентированы на признаковое описание объектов [4].

После предобработки анализируемый текст переводится в вещественное пространство признаков. Для этого чаще всего используются методы, основанные на статистической информации о словах, например, Bag Of Words («мешок слов») [5] или Word2Vec [6]. В этом случае каждому объекту соответствует вектор, длина которого равна количеству используемых слов во всех текстах выборки.

Заключительным шагом при анализе тональности текста является выбор подходящих для решения данной задачи алгоритмов машинного обучения. Как правило, анализ мнений на уровне документа может быть сформулирован как проблема классификации, которая определяет, какое мнение выражается - положительное, отрицательное или нейтральное. Классификаторы обучаются определять полярности рассматриваемых текстов.

В представленной работе исследуются существующие в настоящее время методы обработки естественного языка для анализа текстов комментариев в социальных сетях.

В первой главе описываются общие методы обработки естественного языка, используемые для предварительной обработки текстов.

Во второй главе исследуются существующие методы векторного представления слов в тексте: Bag Of Words («мешок слов») и Word2Vec.

В третьей главе описывается задача классификации документов и подходы к классификации тональности текстов.

Четвертая глава посвящена исследованию существующих пакетов для анализа тональности текстов и выбору оптимального программного средства для анализа комментариев в социальных сетях о компании Tesla.

В пятой главе описывается работа пакета Flair и архитектура применяемого в его основе метода контекстного вложения строк.

В шестой главе представлено описание реализации программной системы автоматизированного прогнозирования краткосрочного движения стоимости ценных бумаг компании на основе анализа тональности текстов комментариев в социальных сетях о компании Tesla.

Постановка задачи

Целью представленной выпускной квалификационной работы является создание программного обеспечения для прогнозирования роста или снижения стоимости акций компании на основе анализа текста комментариев пользователей форумов и социальных сетей об этой компании.

Для достижения поставленной цели было необходимо выполнить следующие задачи:

1) Изучить существующие методы и подходы к анализу текстов на предмет их позитивной или негативной эмоциональной окраски (далее - анализ тональности текста).

2) Изучить существующие программные средства (пакеты) автоматического анализа тональности текстов и выбрать наиболее оптимальные из них для прогнозирования роста или снижения стоимости акций.

3) Разработать программное обеспечение, которое:

- создает базу данных и автоматически заполняет её из Web-ресурса комментариями пользователей форумов и социальных сетей, которые были оставлены за определенный период времени;
- анализирует тональность комментариев с помощью выбранного программного обеспечения;
- строит график результатов анализа тональности комментариев и накладывает его на график реальных биржевых данных.

4) Провести анализ возможных причин расхождений полученных графиков.

5) Протестировать программное средство применительно к комментариям в социальных сетях о компании Tesla.

1 Методы и подходы к анализу текстов

1.1 Предварительная обработка текста

Примененный в представленной работе метод анализа тональности текста состоит из трех этапов:

- предварительная обработка текста;
- перевод текста в вещественное пространство признаков;
- использование методов машинного обучения для последующей классификации тональности.

Предобработка текста является начальным этапом процесса анализа текста естественного языка и включает:

- удаление стоп-слов;
- сегментацию и приведение слов к одной грамматической форме;
- маркировку частей речи.

Перевод текста в вещественное пространство признаков включает в себя:

- Графематический анализ [7] - выполняет деление всего текста на предложения, словосочетания или словоформы (токены). Данный этап необходим для того, чтобы перестать воспринимать буквы, как отдельные символы, а группировать их по словам, к которым они относятся;
- Морфологический анализ [7] - позволяет из каждой словоформы или морфемы выделить ядро слова;
- Синтаксический анализ [7] - это один из важнейших этапов работы с текстом. Он необходим для определения связи между словами и их роли в предложении;
- Семантический анализ [7] определяет смысл фраз. Именно на этом этапе можно понять, какую эмоциональную окраску несет в себе текст.

Использование методов машинного обучения для последующей классификации тональности включает в себя:

- Создание размеченной тренировочной и тестовой выборки, для обучения и проверки классификатора;
- нахождение функции F', максимально приближенную к функции F, где F это заранее известная функция абсолютно точной классификации, которую задаёт учитель.

1.2 Методы векторного представления слов в тексте

Слова в тексте можно кодировать различными способами, самый простой из них – сделать **словарь** из слов, встречающихся в предложении, и присвоить им порядковый номер.

Но такой способ не несет никакой смысловой нагрузки. Важно понимать, насколько схожи лексические значения слов.

Например, в тексте автор выражает свое радостное мнение, используя негативную лексику, по поводу, например, роста акций компании, чьи ценные бумаги он преждевременно приобрел. В таком случае, система будет видеть только негативные слова, но сам тон текста относительно объекта тональности оказывается положительным.

Таким образом, значение одного слова для исследователя не так важно, так как текст состоит из набора слов, поэтому встает вопрос, каким образом можно представить анализируемый текст в виде вектора, который находится в некоем «пространстве смыслов»?

Простейшее решение - установка для конкретного слова единицы в той позиции, которую слово занимает в **словаре**, тем самым получая **вектор слова** - такой подход называется «**Быстрое кодирование**» (англ. **One-hot encoding**). Тривиальный пример:

Таблетка = (0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;)

Пилюля = (0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0;)

$$\text{Таблетка} \wedge \text{Пилюля} = 0$$

Очевидно, что быстрое кодирование не способно определить семантическое сходство между двумя сходными по смыслу и тематике словами, так как пересечение векторов дает 0.

Второй подход: взять каждое слово из документа и над векторами всех слов произвести операцию ∨.

В результате сформируется вектор, который зафиксирует количество использований каждого слова в документе.

Данный подход называется «**Мешок слов**» (англ. **Bag of words**).

1.2.1 Метод Bag Of Words

При применении данного подхода берется набор текстов (далее - корпус документов).

Далее над корпусом производится **токенизация** – деление на уникальные слова. При формировании векторного представления текста токенизируются только ключевые слова, которые определяются критерием χ^2 - больше некоторого критического значения.

Словам, успешно прошедшим через фильтр χ^2 , необходимо присвоить **вес**. Осуществляется эта задача формулой **TF-IDF** (от англ. **TF** – **term frequency**, **IDF** – **inverse document frequency**).

TF-IDF – это статистическая мера, используемое для оценки **значимости** слова для данного документа.

Составляющие формулы:

TF (с англ. **Term Frequency** – частота слова) - отношение количества вхождений определенного слова к полному объёму слов в документе:

$$TF(t,d) = \frac{n_t}{\sum_k n_k},$$

где n_t – число слов вида t в документе d;

$\sum_k n_k$ – объёму слов в документе.

IDF (**inverse document frequency** – обратная частота документа) логарифм обратной частоты слова во всех документах:

$$IDF(t,D) = \log \frac{|D|}{|\{d_i \in D | t \in d_i\}|},$$

где $|D|$ – число документов в корпусе;

$|\{d_i \in D | t \in d_i\}|$ – число документов из корпуса $|D|$ в которые входит слово t.

$$\mathrm{TF-IDF(t,d,D)} = TF(t,d) * IDF(t,D) \quad (1)$$

Смысл формулы 1 заключается в том, что слова, часто встречающиеся в каком-то конкретном тексте и при этом встречающиеся редко в других текстах будут иметь больший вес.

Подход «мешок слов» весьма прост для понимания и реализации, но он имеет несколько недостатков:

- **Разреженность**. Данное ограничение возникло из-за естественного развития информационных технологий и расширения объёма информации, которую можно найти в интернете. Документы или тексты, представляющие из себя группу слов, значительно увеличились в объёме, а это значит, что и словари стали больше. Возникает ситуация, когда тексты проецируются в пространства высокой разреженности из-за очень большого объёма словарей. В итоге, возникает экспоненциальный рост сложности вычислений.

- **Невозможность** определения семантической связи между словами. Ключевой недостаток данного метода.

Свободным от данных проблем является метод **Word2Vec**.

1.2.2 Метод Word2Vec

В 2013 году мало известный аспирант из Чехии Томаш Миколов предложил идею, основанную на **контекстной близости** слов. То есть, слова, которые часто встречаются в одинаковых группах слов, имеют некую семантическую связь и будут иметь в векторном пространстве меньшее косинусное расстояние.

Модель, которую предложил Т. Миколов, может рассчитать вектор слова по окружающим словам. Здесь не выделяется одно ключевое слово, отвечающее за настроение всего текста, а слово берется вместе с контекстом. Таким образом, векторное представление близких друг к другу слов будем мало отличаться.

В том же году группой исследователей Google под руководством Томаша Миколова было разработано программное обеспечение **Word2Ve**c.

Word2Vec работает следующим образом: на вход методу приходит корпус из большого числа документов. Документы сканируются, и каждому слову из текстов сопоставляется векторное представление в пространстве смыслов; такое представление называется **эмбеддинг** (англ. embedding – встраивание).

Векторное представление слова способно определять его близость к другим словам определенного контекста. Полученные вектора используются для машинного обучения. Систему нужно обучить таким образом, чтобы каждому слову была установлена правильная вероятность встречи его в реальном тексте в нужном контексте.

Математически это записывается в виде формулы **softmax**:

$$P(w_k|w_c) = \frac{e^{s(w_k|w_c)}}{\sum_{w_i \in V} e^{s(w_i|w_c)}},$$

где V – множества всех слов;

w_k – вектор слова, относительно которого высчитывается предсказание;

w_c – вектор слов, которые должны окружать слово w_k. Вектор w_c можно получить путем усреднения векторов слов, окружающих слово w_k;

$\mathrm{s}(w_x|w_y)$ – некоторая функция, которая, определяет число, обозначающее отношение между двумя векторами w_x и w_y. Этим числом может быть, например, уже упомянутое косинусное расстояние.

Процесс тренировки осуществляется с помощью метода **CBOW** (англ. **continuous bag of words**).

CBOW – обычная модель «мешка слов», только в неё подается не одно слово, а последовательность, состоящая из нечетного числа слов, слово в середине есть целевое. Слова слева и справа от целевого это и есть окружение этого слова или же контекст. Так как важен порядок подающейся на вход информации, а порядок самих слов внутри CBOW не важен.

Ключевой задачей в CBOW является оптимизация векторов слов, составляющих контекст для целевого слова. Оптимизация проводится путем сведения векторных представлений слов контекста к минимальному значению.

В методе **Negative Sampling** предлагается идея увеличивать для целевого слова вероятность нахождения в окружении его контекста и в то же время уменьшать вероятность нахождения в том окружении, где оно встречается редко. Интуитивно можно понять, что слова, которые появляются в одном и том же контексте, скорее всего, будут иметь схожие векторные представления.

Числитель в уравнении (1) показывает это, назначая большее значение для похожих слов посредством скалярного произведения двух векторов. Если слова не встречаются в контексте друг друга, представления будут разными, и в результате числитель будет иметь маленькое значение.

Метод **Negative Sampling** реализуется формулой:

$$NegS(w_k) = \log \sigma\left(s(w_k|w_c)\right) + \sum_{i=1}^{d} \mathbb{E}_{w_i \sim P_n(w)} \log \sigma\left(-s(w_i|w_c)\right) \qquad (2)$$

В формуле 2 первый логарифм отвечает за максимизацию вероятности появления слова в окружении контекста.

Второе слагаемое должно минимизировать вероятности слов, не лежащих в контексте, но, вместо того чтобы выполнять минимизацию для абсолютно всех слов в словаре, случайным образом выбираются слова в количестве d.

Происходит отбор этих слов на основе частоты их появления, такое распределение называется «негативным распределением»:

$$P_n(w) = U(w)^{\frac{3}{4}},$$

где $\mathrm{U(w)}$ – «юниграммное» распределение слов, то есть берется по одному слову.

Степень $\frac{3}{4}$ вынуждает формулу выбирать менее часто встречающиеся и в тоже время незначимые слова, такие как предлоги и артикли.

Формула $\mathbb{E}_{\boldsymbol{w_i \sim P_n(w)}}$ означает, что $\boldsymbol{w_i}$ каждый раз берется случайным образом из распределения $P_n(w)$.

Результат. Метод **CBOW** позволяет предугадывать следующее слово, опираясь на представленный в тексте контекст, но этим Word2Vec не ограничивается.

Т. Миколов вместе со своей командой добавил еще один метод, который выполняет функцию **CBOW**, но наоборот. Этот метод работает примерно по такому же принципу, что и **CBOW,** и называется **SkipGram**.

В то время, как **CBOW** предсказывает возможный контекст, которому слово может принадлежать, **SkipGram** выполняет работу по предсказанию самого контекста по целевому слову, что включает в себя и предсказание темы, словосочетания и семантической предрасположенности рассматриваемого слова.

Такие возможности делают метод **Word2Vec** крайне полезным инструментов при анализе текстов. Метод способен улавливать семантическую связь между словами и, что примечательно, данная способность явно не заложена в методе в виде отдельного пункта или алгоритма. Поэтому хорошо обученная модель на основе **Word2Vec** способна улавливать такие смысловые лексические зависимости, как пол или количество объектов реального мира.

Наглядно это демонстрирует рисунок 1:

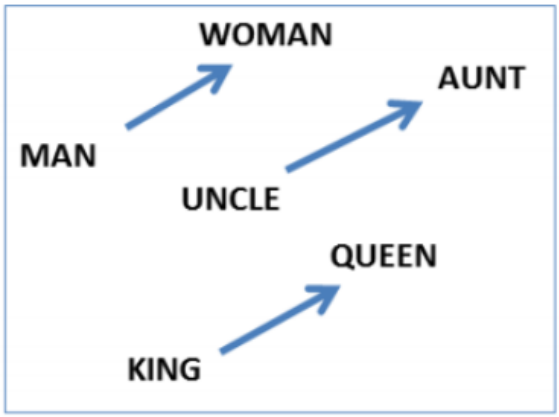


Рисунок 1 – Векторное пространство смыслов 1

Рисунок демонстрирует, что между векторным представлением группы «мужчина и женщина» и вектором группы «король и королева» наблюдается коллинеáрность в пространстве смыслов.

Это значит, что если на определенном расстоянии и в определенном направлении направить от слова вектор, то можно получить женский смысл этого слова, или направить вектор в обратном направлении и получить мужскую версию слова. Таким образом, в зависимости от точки начала вектора возможен выбор семантической принадлежности слова.

Другой пример показывает лексическую связь между словами. Например, известно, что в английском языке, если в конце существительного поставить букву «s», можно добиться множественного значения слова, то есть говорить о нем во множественном числе.

Наглядно это демонстрирует рисунок 2:

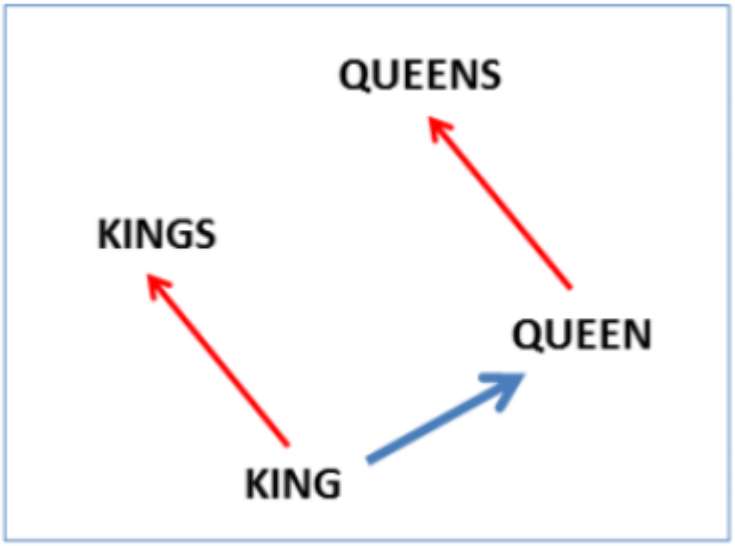


Рисунок 2 – Векторное пространство смыслов 2

На рисунке видна та же семантическая связь в смысле пола между королем и королевой, выраженная синим вектором. Если отложить от короля и королевы вектора одинаковой длины в другую сторону, то получится множественное число для этих понятий.

Таким образом, **Word2Vec** использует новаторскую идею анализа текстов, которая позволяет более точно понимать смысл слов по сравнению с предыдущими методами.

1.3 Задача классификации текстов

1.3.1 Определение задачи классификации документов

Вся задача анализа тональности текста сводиться к тому, чтобы определить, выражается положительное, отрицательное или нейтральное мнение. Иными словами - отнести данный документ (текст) d_i из множества всех документов D к тому или иному классу c_j из множества всех классов C, где:

$$D = \{d_1, d_2, d_3, \dots, d_n\}$$

Множество классов в данном случае содержит в себе два класса: положительный тон - c_1 и отрицательный тон - c_2:

$$C = \{c_1, c_2\}$$

Далее определяется целевая функция:

$$F: D \times C \rightarrow \{0, 1\}$$

Целевая функция фиксирует, принадлежит документ конкретному классу или нет.

Задача состоит в том, чтобы найти функцию $\boldsymbol{F'}$, максимально приближенную к функции $\boldsymbol{F}$.

Когда такая функция $\boldsymbol{F'}$ будет найдена, она будет называться **классификатором.** Классификаторы обучаются определять тональность рассматриваемых текстов.

Истинное значение целевой функции становиться известно системе только в процессе обучения классификатора разделять документы на классы, и, как правило, значения целевой функции определяет человек вручную.

Для того, чтобы обучить классификатор, осуществляется разбиение всего числа документов на две выборки: **учебную** и **тестовую**.

Учебная выборка представляет из себя множество документов D длины s, где $s < n$.

$$Ds = \{d_1, d_2, d_3, \dots, d_s\}$$

По данной выборке классификатор обучается правильно распознавать документы, замечая закономерные признаки принадлежности документа к какому-либо классу.

Тестовая выбора представляет из себя множество документов D длины $t = n - s$, где $s < t \leq n$.

$$Dt = \{d_{s+1}, d_{s+2}, d_{s+3}, \dots, d_n\}$$

На данном множестве тестируется правильность выводов, которые делает классификатор. Это значит, что каждый документ из множества Dt подается как входное значение классификатору F'; затем полученный результат сравнивается с результатом работы целевой функции F, правильные значения которой уже заранее известны.

Если классификатор распознает документы в процентом соотношении, как можно большем по отношению к общему числу документов в тестовой выборке, то тем эффективнее считается работа полученного классификатора.

Стоит заметить, что классификации разделяются на **булеву** и **ранжированную**:

Булева классификация – $F: D \times C \rightarrow \{0, 1\}$

Ранжированная классификация – $F: D \times C \rightarrow [0, 1]$

При булевой классификации целевая функция принимает только два значения: 0 или 1, то есть «не соответствует точно» или «точно соответствует»

При ранжированной классификации целевая функция принимает некоторое число из какого-либо диапазона, в нашем случае это от 0 до 1. Данный вид классификации демонстрирует степень принадлежности или же уверенность в том, что документ относится к определенному классу.

1.3.2 Подходы к классификации тональности документов

Осуществив значительное обобщение, можно разделить подходы к классификации тональности документов на категории:

- Метод на основе словарей тональности
- Метод машинного обучения без учителя
- Метод машинного обучения с учителем

Подход на основе словарей тональности использует словари тональности, которые составляются заранее. В общем виде, словарь представляет из себя список, где собраны различные слова, и им сопоставляется эмоциональная числовая оценка.

Например, в открытом тональном словаре русского языка «КартаСловСент» Кулагина Д.И. [8] эмоциональная оценка располагается на промежутке $[-1, 1]$. Промежуток трактуется так, что если численная оценка $[-1, 0)$, то смысловая нагрузка слова имеет отрицательный оттенок, при том, чем оценка ближе к -1 тем слово более негативное, если равна 0, то слово нейтральное, а если оценка $(0, 1]$, то смысловая нагрузка слова имеет положительный оттенок, при том, чем оценка ближе к 1, тем слово более позитивное. }8

Пример слов из таблицы «КартаСловСент» Кулагина Д.И. [8] приведен в таблице 1.

Таблица 1 – Словарь «КартаСловСент» Кулагина Д.И.

слово	эмоционально-оценочный заряд
хороший	1.0
авторитет	0.9
неплохой	0.7
довершить	0.64
нормальный	0.31
средний	0.0
абракадабра	-0.17
авантюра	-0.58
ужастик	-0.64
ужасный	-1.0

Из минусов данного метода стоит отметить большую трудоемкость процесса составления словаря; как правило, этим занимается целая группа людей. Данный метод не способен понимать семантическую близость слов, так как в словарях указываются, как правило, числовая степень позитивности / негативности слова. Поэтому вся задача классификации сводится к подсчету вхождений каждого слова в тексте и вычислении агрегированной величины эмоциональной окраски всего текста.

Подход машинного обучения без учителя – наименее точный метод анализа тональности текста, но, в тоже время, самый интересный для изучения развития, так как требует минимального вмешательства человека в процесс обучения. Такой подход порождает две проблемы:

Кластеризация – поиск групп похожих примеров в данных. Обычно осуществляется алгоритмом **k-Means.**

Оценка плотности – анализ распределения данных и осуществление его обобщения во входном множестве. Решается алгоритмом **ядерная оценка плотности,** который производит сглаживание.

Подход машинного обучения с учителем – наиболее эффективный и используемый метод анализа текста. Главной целью метода является обучение классификатора путем подачи на вход заранее размеченных

данных. Затем проводится проверка точности на тестовых множествах для проверки точности и правильности предсказания класса документа. Преимуществом метода является универсальность по отношению к словарю, а точнее, для обучения классификатора на определенную тему формируется свой собственный словарь, который ориентирован именно на работу с данной темой.

1.3.3 Численная оценка работы классификатора

Поскольку, в нашей задаче существуют только 2 класса, в процессе своей работы классификатор тональности текста делает один из двух выводов:

- P – документ определен системой как имеющий позитивную тональность;
- N – документ определен системой как имеющий негативную тональность.

Если система провела классификацию документа неверно, в дополнении к вышеуказанным результатам стоит добавить экспертное мнение:

- T – Система правильно провела классификацию документа;
- F – Система неправильно провела классификацию документа.

Теперь можно объединить результаты классификатора и экспертного мнения в таблицу 2:

Таблица 2 – Комбинации вывода классификатора и экспертного мнения

i-ый Документ		Вывод классификатора	
		положительный	негативный
Экспертное мнение	положительный	TP	FN
	негативный	FP	TN

На основе данных значений высчитываются метрики, способные оценить качество работы классификатора:

- Метрика **Аккуратность** (англ. ***Accuracy***):

$$\text{Аккуратность } = \frac{TP + TN}{TP + FN + FP + TN}$$

Определяет долю правильно классифицированных документов независимо от того, позитивные они или негативные.

У данной метрики существует недостаток: она присваивает всем документам одинаковый вес.

Данная особенность может привести к негативным последствиям: если распределение документов сильно смещено в сторону какого-либо класса или группы классов, это значит, что классификатор имеет больше информации по этим классам, а значит, будет формировать по ним более правильные выводы, в то время как другие классы будут распознавать очень плохо или вообще не будут распознаваться.

Для решения этой проблемы используются следующие две метрики:

- Метрика **Точность** (англ. **Precision**), высчитывается отдельно для негативных и положительных документов:

$$\text{Точность положительная } = \frac{TP}{TP + FP}$$

Определяет долю положительных документов от общего количества документов, определенных системой как положительные.

$$\text{Точность негативная } = \frac{TN}{TN + FN}$$

Определяет долю отрицательных документов от общего количества документов, определенных системой как отрицательные.

- Метрика **Полнота** (англ. **Recall**), высчитывается отдельно для негативных и положительных документов:

$$\text{Полнота позитивная } = \frac{TP}{TP + FN}$$

Определяет долю положительных документов от общего числа позитивных документов.

$$\text{Полнота негативная } = \frac{TN}{TN + FP}$$

Определяет долю негативных документов от общего числа негативных документов.

Метрика **F-мера (**англ. **F – measure)**:

$$F - \text{мера } = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Определяет меру, которая в себе комбинирует точность и полноту.

2 Выбор программы для определения тональности текстов

2.1 Обзор существующих систем

Несмотря на то, что анализ тональности текста - это относительно молодой раздел в области обработки текстов на естественном языке, за последние пять лет он сильно изменился и расширился.

Из-за того, что все больше людей с каждым годом входят в глобальную информационную сеть, возникают множества сообществ, форумов, где люди делятся своим мнением по поводу товаров и услуг. Данный факт не остается незамеченным компаниями, которые производят эти услуги, поэтому анализ мнений людей и тональности отзывов представляет интерес для исследователей.

Благодаря этому в мире появилось множество пакетов с открытым исходным кодом, которые предоставляют бесплатно готовые, настроенные системы для определения тональности текстов.

Рассмотрим самые используемые пакеты, которые используют метод, основанный на правилах.

Идея метода достаточно проста, она не использует машинное обучение и очень схожа с работой метода на основе словарей. Системы на основе правил используют вручную заданные правила для классификации текстов, которые представляют словарь из ключевых слов по данной тематике, а также правила совместного использования этих слов в тексте.

2.2 Пакет Textblob

Textblob - пакет написан на Python для решений задач NLP. Обладает API (программный интерфейс приложения) доступом к таким методам как анализ тональности текста, исправление орфографии и т.д. Обладает весьма простым для использования интерфейсом.

Для предложения, подаваемого на вход системе, Textblob возвращает две характеристики:

- Полярность – численное значение, которое находится на промежутке $[-1, 1]$. Положительное число указывает на то что текст имеет положительный эмоциональный окрас, отрицательное число указывает, что текст имеет негативный эмоциональный окрас;
- Субъективность – численное значение, которое находится на промежутке $[0, 1]$. Демонстрирует степень субъективности слов и фраз, которые были представлены в тексте. Объективные предложения содержат в себе фактическую информацию, которую возможно проверить, а субъективные предложения выражают чувства, убеждения и точку зрения автора.

Как было сказано выше, **Textblob** работает, используя метод анализа текстов на основе правил и словаря из фраз, для которых заранее расписаны правила и эмоциональная оценка.

При возникновении ситуации, когда Textblob встречает незнакомые ему слова, он игнорирует их. Тем не менее, такие модели разработчик хорошо обучает и размечает, прежде чем выпустить систему для массового использования другими людьми.

Поэтому, когда Textblob встречает знакомые фразы и слова, которым он может назначить полярность, а потом усреднить оценки всех ключевых объединений слов, точность программы остаётся на приемлемом уровне.

Пример работы программы TextBlob на фразе «The food was great!» представлен на рисунке 3:

```
from textblob import TextBlob

testimonial = TextBlob("The food was great!")
print(testimonial.sentiment)
```

```
Sentiment(polarity=1.0, subjectivity=0.75)
```

Рисунок 3 – Пример работы TextBlob

Программа показывает полярность 1, что соответствует действительности, субъективность равна 0,75.

2.3 Пакет VADER Sentiment Analysis

VADER Sentiment Analysis - пакет для Python, способный определять не только принадлежность текста положительному или отрицательному классу, но и силу эмоций.

VADER работает, используя словарь для того чтобы соотносить лексические характеристики и силу эмоций. Так же система использует пять эвристик, такие как: пунктуация, капитализация, модификатор степени, изменение смысла из-за союза «но», пятая эвристика анализирует триграмму перед лексическими признаками, в которых было отмечено большое количество чувственных или эмоциональных слов и фраз, для того чтобы распознать инверсию тональности. Инверсия происходит путем умножения численного значения тональности лексического признака на значение -0.74 которое установлена эмпирическим путем.

Данная модель хорошо зарекомендовала себя при анализе мнений людей в социальных сетях, а также анализе отзывов на фильмы.

VADER высчитывает вероятность того, что подаваемый на вход текст будет позитивным, негативным или нейтральным.

Пример работы программы VADER на фразе «The food was great!» представлен на рисунке 4:

```
analyzer = SentimentIntensityAnalyzer()
sentence = "The food was great!"
vs = analyzer.polarity_scores(sentence)
print("{:-<65} {}".format(sentence, str(vs)))
```

```
{'compound': 0.6588, 'neg': 0.0, 'neu': 0.406, 'pos': 0.594}
```

Рисунок 4 – Пример работы пакета VADER

Система определила текст «The food was great!» как негативный – 0%, нейтральный – 0,406%, позитивный – 0,594%. Как видно, в сумме все показатели дают 100%.

Главным недостатком подхода, основанного на правилах, является то, что система ориентируется на слова и словосочетания, но абсолютно игнорирует контекст, который используется в тексте. Например, текст «Этот фильм ужасов действительно смог меня напугать» будет расценен как отрицательный, при том, что по сути это положительный отзыв для фильма ужасов.

2.4 Пакет FastText

Далее рассмотрим программы на основе **эмбеддингов** (векторное представление в пространстве смыслов).

FastText пакет для Python от разработчиков Facebook Research.

FastText - это система, которая включает заранее заготовленные, предобученные эмбеддинги, позволяющие относить слова к определенному контексту.

FastText был разработан под руководством создателя Word2vec Томаша Миколова и в своей основе содержит метод Word2vec, который описан в 2.2.

Пример работы программы FastText представлен на рисунке 5:

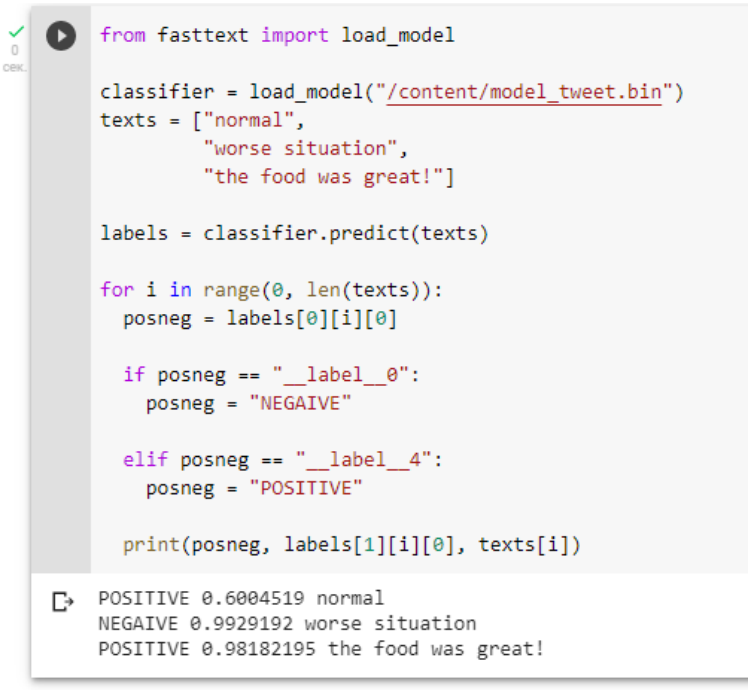


Рисунок 5 – Пример работы пакета FastText

В данном примере текст «The food was great!» определяется как позитивный, что верно, а также указывается численная степень эмоциональной нагрузки, которая равна 0,98182195, то есть, очень высокая.

Степень эмоциональной нагрузки является такой большой из-за восклицательного знака, который можно убрать и посмотреть на результат:

```
POSITIVE 0.8381438 the food was great
```

Видно, что эмоциональность текста снизилась.

Другой текст, который имеет в себе просто слово normal, соответствует своему нейтральному эмоциональному статусу 0,6004519.

Текст «worse situation» является негативным текстом и имеет высокую степень эмоциональной нагрузки 0,9929192, потому что слово «worse» означает «худший», если заменять на «bad», то эмоциональная нагрузка снизиться до значения 0.9519037.

2.5 Пакет Flair

Flair пакет для Python, простая в использовании система, которая использует все самые современные методы анализа текста. Система использует заранее обученную нейронную сеть, которая тренировалась на отзывах о фильмах на сайте IMDB.

Учитывая специфику работы принципа отзывов на фильмы, Flair весьма точно прогнозирует тональность подаваемых на вход текстов.

Отзывы на фильмы формируются следующим образом. Человек пишет свое мнение о фильме и выставляет оценку от 1 до 10 включительно. Данная система удобна, потому что мнению дается чёткая оценка по десятибалльной шкале и нейронная сеть сразу может понять насколько положительный или отрицательный эмоциональный оттенок имеет текст.

Программа выдает позитивный или негативный результат, а также демонстрирует свою уверенность, которая выражается числовым значением в диапазоне (0.5, 1). То есть вероятность всегда больше 50%, потому что система может оценивать некоторые тексты как тексты, имеющие не такой очевидный эмоциональный окрас.

Пример работы программы. Рассмотрим отзывы на фильм Андрея Тарковского «Зеркало».

Подадим такой отзыв на вход, он имеет оценку 9 звезд из 10, что по сути является показателем того, что отзыв является положительным. Пример отзыва и результат его обработки представлены на рисунках 6, 7.

9/10

Another great Tarkovsky film
brianberta 31 August 2016

Tarkovsky is a great director for people who prefer to watch challenging films. A lot of his films have deep meanings behind them, and they can inspire a lot of discussion. I like to describe him as the Russian Kubrick. Like Kubrick, he would often tell simple, yet grand stories in his films. As a result, many of his films would seem like an astonishing feat. However, I'd actually say that "The Mirror" is the easiest Tarkovsky film to interpret. It has a simple, yet impressive meaning.

A dying man named Alexei remembers different events from his past in pre-war, WW2, and post-war times. These events seem unconnected, and they don't appear to follow a particular storyline of any kind. This is also, arguably, his most personal film as many of its scenes heavily draw on Tarkovsky's own childhood such as the evacuation from Moscow to the countryside during WW2, a withdrawn father, and a mother who works as a proof-reader at a printing press.

did take me a few viewings to really love "Stalker". This film could grow on me a lot as well on subsequent viewings.

Рисунок 6 – Позитивный отзыв на фильм

```
[37] from flair.models import TextClassifier
     from flair.data import Sentence

     f = open('text.txt', 'r')
     texted = f.read()

     classifier = TextClassifier.load('en-sentiment')
     sentence = Sentence(texted)
     classifier.predict(sentence)

     # print sentence with predicted labels
     print('Sentence above is: ', sentence.labels)
```

```
ilm could grow on me a lot as well on subsequent viewings ."'/'POSITIVE' (0.9996)]
```

Рисунок 7 – Результат тональной оценки системой Flair

Видим, что программа посчитала отзыв положительным с уверенностью в 0,9996.

Рассмотрим другой отзыв. Пример отзыва и результат его обработки представлены на рисунках 8, 9.

★ 7/10

Another walk into Andrei Tarkovsky's maverick world which shows the unseen dimensions and patterns of unconventional storytelling.

SAMTHEBESTEST 8 October 2021

Zerkalo / The Mirror (1975) : Brief Review -

Another walk into Andrei Tarkovsky's maverick world which shows the unseen dimensions and patterns of unconventional storytelling. Tarkovsky's 'Andrei Rublev' (1966) is one of my favourite films from the 60s decade and his 'The Stalker' (1979) kind of left me scratching my head. It was a very slow paced film, maybe that's why, or maybe because the metaphysical elements were new to me then. But i understood one thing from these two films that Andrei Tarkovsky is not just an ordinary storyteller. He is maverick and how. When i think about Stanley Kubrick, Ingmar Bergmann and Satyajit Ray's films from 60s and 70s decade, i feel like yes this is something out of the world. This is something i haven't seen before and this could be the last step of the experimental and artistic cinema but then i meet Andrei Tarkovsky's cinema. He is beyond that and literally forces you to think about the narrative and how it is told which is simply unexpected and unimaginable. The Mirror has no extraordinary story. It is just a simple story of a dying man in his forties remembering his past. His childhood, his mother, the war, personal moments and things that tell of the recent history of all the Russian nation. Now you have read the entire story in these two small sentences but wait for the surprise. The Mirror is structured in the form of a nonlinear narrative. The film combines contemporary scenes with childhood memories, dreams, and newsreel footage. Its cinematography slips between color, black-and-white, and sepia. The narrative is unconventional and chronologically shuffled about the events which is surely confusing but brings a new experience of cinema viewing for you. In short, it's an upper level artistic and unorthodox cinema made only for smart, no sorry Smarter audience.

Рисунок 8 – Негативный отзыв на фильм

```
ox cinema made only for smart , no sorry Smarter audience ."'/'NEGATIVE' (0.6064)]
```

Рисунок 9 – Результат тональной оценки системой Flair

Система показывает 'NEGATIVE' с уверенностью 0,6064, потому что автор отзыва не совсем уверен в том, что понял смысл фильма, и система это уловила. Однако, интересно, почему отзыв имеет достаточно неплохую оценку 7, а система показала негативный отзыв? На самом деле, по всей видимости, автор поставил такую оценку из уважения к именитому режиссёру, а далее честно признался, что фильм для него сложен для понимания, соответственно не совсем ему понравился.

Стоит также отметить механику работы оценок для различных произведений культуры, таких как: фильмы, музыка и книги. Это объясняет, почему иногда люди так выставляют оценки.

Исторически сложилось так, что оценки ниже 6 являются отрицательными, а то, что меньше 6 просто выражает то, насколько произведение плохо.

Существует негласное правило, что оценки представляют примерно следующие описания:

- 6 – оценка чуть ниже среднего;
- 7 – средняя оценка;
- 8 – уже хорошее произведение, но есть не мало недостатков;
- 9 – хорошее произведение;
- 10 – шедевр в представлении автора отзыва.

Именно механика «уверенности» нейронной сети является ключевой в выборе программы для данной работы, поэтому для анализа комментариев о компании Tesla был выбран пакет Flair.

2.6 Сравнение точности классификации пакетов

Точность с которой каждый пакет классифицирует комментарии на негативные или позитивные определяется следующим образом: на вход каждой системы подается 25000 отзывов на фильмы с сайта IMDB, каждому отзыву уже присвоено правильная тональность: позитивная или негативная.

Задача систем, правильно распознать тональность у как можно большего числа отзывов, далее вычисляется процент правильно распознанных отзывов.

Также в сравнении будет участвовать нейронная сеть созданная собственными усилиями на основе эмбеддингов архитектуры LSTM (32 входных нейрона) с помощью пакета tenzorflow-keras. Данная сеть была обучена на 25000 заранее размеченных комментариях с сайта отзывов на

фильмы IMDb. Результаты сравнения точности пакетов представлены в таблице 3:

Таблица 3 – Сравнение точности пакетов

Пакет	Кол-во правильно распознанных отзывов	Точность, %
VADER	17 254	69,016
Textblob	17 358	69,432
FastText	17 738	70,952
Собственная НС	21 355	85,420
Flair	21 879	87,516

Очевидно, что наиболее успешно с задачей определения тональностей справляется пакет Flair.

3 Работа пакета Flair

Пакет Flair работает на основе метода **контекстного вложения строк** (англ. Contextual String Embeddings) [9].

3.1 Контекстное вложение строк

Данный метод предоставляет иной взгляд на контекстуальное встраивание слов, который представляет собой работу по анализу символов в тексте. Метод включает в себя следующие преимущества вышеупомянутых методов:

- Как в Word2Vec возможность обучения на больших неразмеченных данных;
- Определять значение слов в контексте их использования и применять различные эмбеддинги для многозначных слов в зависимости от контекста, в котором они используются.
- Моделировать слова и последовательности слов с упором на их символьное представление, для того, чтобы лучше отслеживать ошибки, а также морфемы, такие как приставки и суффиксы.

Метод работает на основе нейронных сетей **долгой-краткосрочной памяти** – **LSTM** (англ. **Long short-term memory**).

3.2 Архитектура LSTM

LSTM впервые была представлена в 1997 году Зеппом Хохрайтером и Юргеном Шмидхубером [10]. Особенностью архитектуры LSTM является наличие памяти о существенных деталях предыдущих контекстов. Данная архитектура по сей день является одной из лучших схем нейронных сетей и используется во многих задачах. Так как принцип памяти не очевиден, стоит подробнее описать архитектуру LSTM.

Главная идея LSTM заключается в наличии памяти, которая может сохранять ключевые данные и переносить их на следующую итерацию работы архитектуры. На каждой итерации осуществляет свою работу специальный блок LSTM, называемый **ячейкой**, именно в ячейках происходят вычисление весовых коэффициентов и сохранение информации.

Строение ячейки выглядит изображено на рисунке 10:

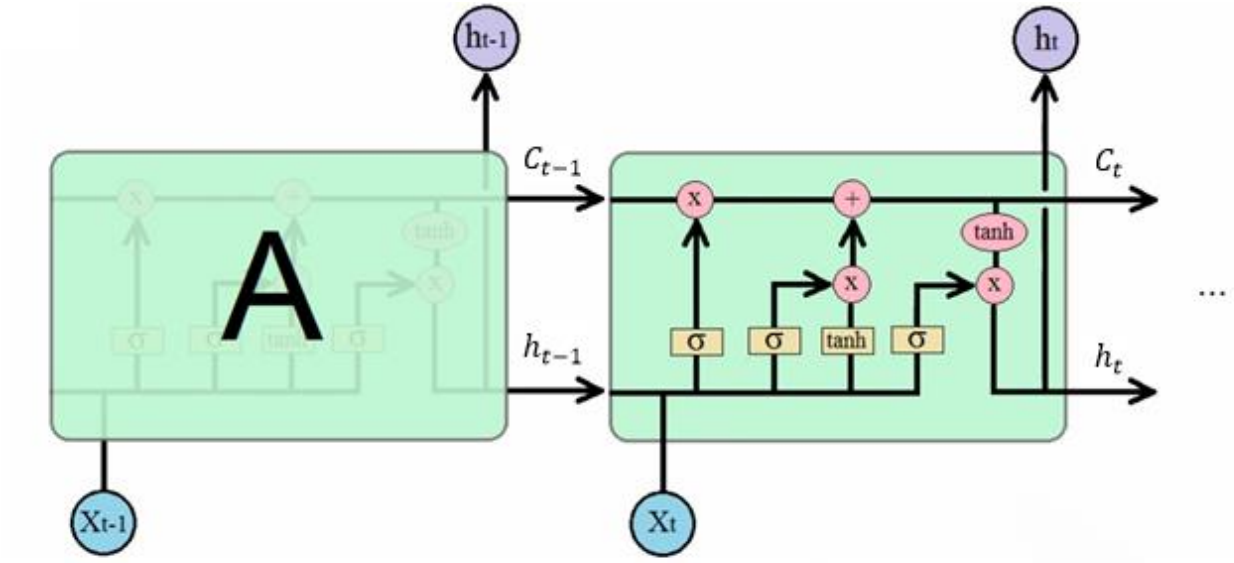


Рисунок 10 – Ячейки памяти LSTM

Где X_t– входной вектор на -ом шаге;

h_t– вектор скрытого состояния на -ом шаге;

C_t– вектор с канала памяти, который содержит в себе долгосрочную память -ом шаге;

x – поэлементное умножение вектора;

+ – поэлементное сложение вектора;

σ – полносвязный слой нейронной сети с сигмоидальной функцией активации;

– поэлементное вычисление гиперболического тангенса;

tanh – полносвязный слой нейронной сети с функцией активации на основе гиперболического тангенса.

Принцип работы системы памяти на рисунке демонстрирует канал, которому на вход поступает информация C_{t-1} с предыдущей ячейки.

Далее данная информация переписывается с помощью «функции забывания» и «функции запоминания». Таким образом, в канале C всегда содержится полезная информация, которая может включать в себя как информацию, записанную на длительном промежутке времени от старта обучения, так и записанную совсем недавно. Такая особенность позволяет избежать проблемы долговременной зависимости, которой страдают рекуррентные нейронные сети RNN. После всех манипуляций только значимая информация передается в следующую ячейку, а также на обработку вектору скрытого состояния h_t.

Строение функции забывания в ячейке изображено на рисунке 11:

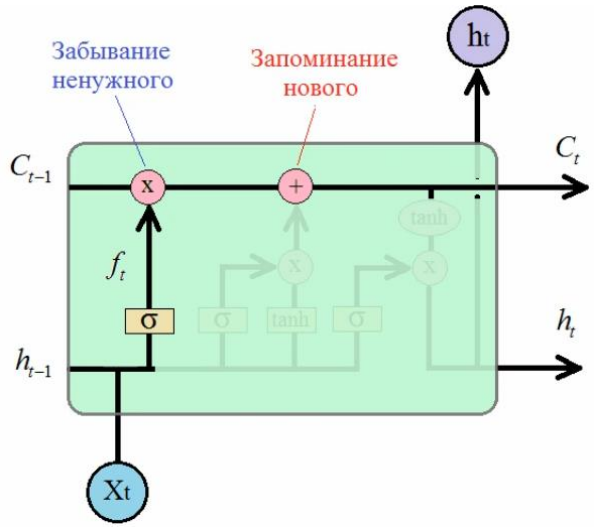


Рисунок 11 – Функция забывания в ячейке памяти LSTM

Забывание ненужной информации происходит путем поэлементного перемножения вектора f_t с вектором C_{t-1}. Сам вектор f_t формируется следующим образом:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right), \quad (3)$$

где W_f – весовые коэффициенты на полносвязном слое;

$[h_{t-1}, x_t]$ – объединение векторов h_{t-1} и x_t;

b_f – вектор смещения bias.

На выходе строиться вектор f_t такой же размерности что и C_{t-1}, в котором каждое значение - это число в диапазоне $[0, 1]$. Все эти числа являются показателями значимости для каждого значения вектора C_{t-1}, поэтому при поэлементном перемножении происходит забывание ненужной информации и оставление информации, которая может пригодиться. В формировании f_t ключевую роль играют весовые коэффициенты W_f, которые в процессе обучения подбираются таким образом, чтобы корректно ослаблять ненужную информацию, а нужную оставлять.

Строение функции забывания в ячейке изображено на рисунке 12:

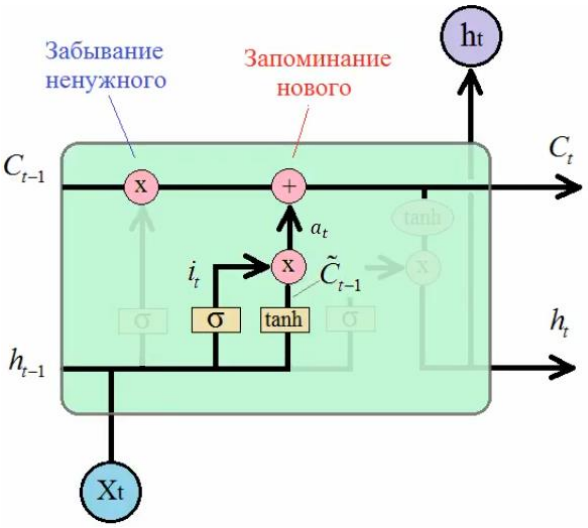


Рисунок 12 – Функция запоминания в ячейке памяти LSTM

Запоминание ненужной информации происходит путем поэлементного сложения вектора a_t с вектором C_{t-1}. Сам вектор a_t формируется следующим образом:

$$a_t = i_t \otimes \tilde{C}_{t-1},$$

где $i_t = \sigma\left(W_p \cdot [h_{t-1}, x_t] + b_i\right);$ (4)

$$\tilde{C}_{t-1} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C).$$

Все переменные в формуле 4 аналогичны переменным в формуле 3. Сигмоидальная функция $\sigma()$, как и в f_t, показывает, какую информацию стоит убрать из вектора с информацией $\tilde{C}_{t-1}$, который, в свою очередь, формируется с помощью функции активации гиперболического тангенса и каждое его значение находится в диапазоне $[-1, 1]$.

После всех манипуляций происходит поэлементное сложение вектора a_t и C_{t-1}, полученный вектор складывается с $f_t \otimes C_{t-1}$ и на выходе получается выходной вектор контента C_t.

$$C_t = f_t \otimes C_{t-1} + a_t \oplus C_{t-1}.$$

Формирование выходного скрытого состояния h_t изображено на рисунке 13.

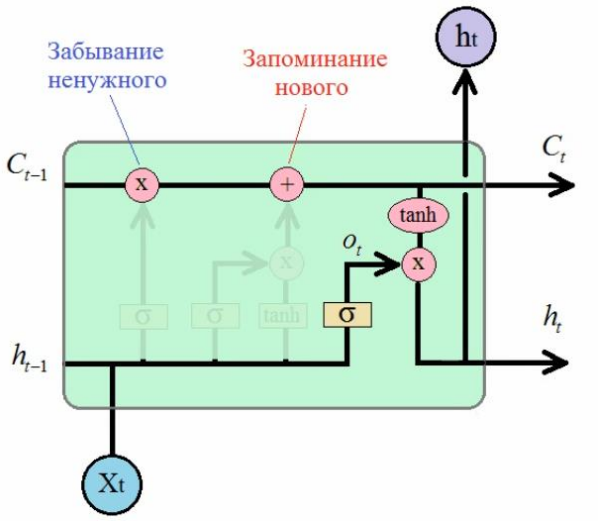


Рисунок 13 – Функция формирования скрытого состояния h_t памяти LSTM

Для того, чтобы сформировать скрытый вектор состояния h_t, вектор C_t проходит через функцию гиперболического тангенса в результате чего

нормализуется, полученный нормализованный вектор поэлементно умножается на вектор o_t, который вычисляется по формуле:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o),$$

где смысл всех элементов аналогичен смыслу элементов в формулах 3 и 4.

Далее происходит поэлементное умножение вектора o_t и $tahn(C_t)$ в результате чего строится вектор скрытого состояния h_t:

$$h_t = o_t \otimes tahn(C_t).$$

3.3 Применение LSTM в пакете Flair

В пакете Flair, архитектура LSTM используется из-за её способности гибко кодировать долгосрочные зависимости и их скрытые состояния. В Flair символы используются как атомарные единицы информации, поэтому текст передается в LSTM как последовательность символов, где для каждого элемента последовательности LSTM обучается предсказывать следующий символ.

Наглядно работа LSTM показана на рисунке 14.

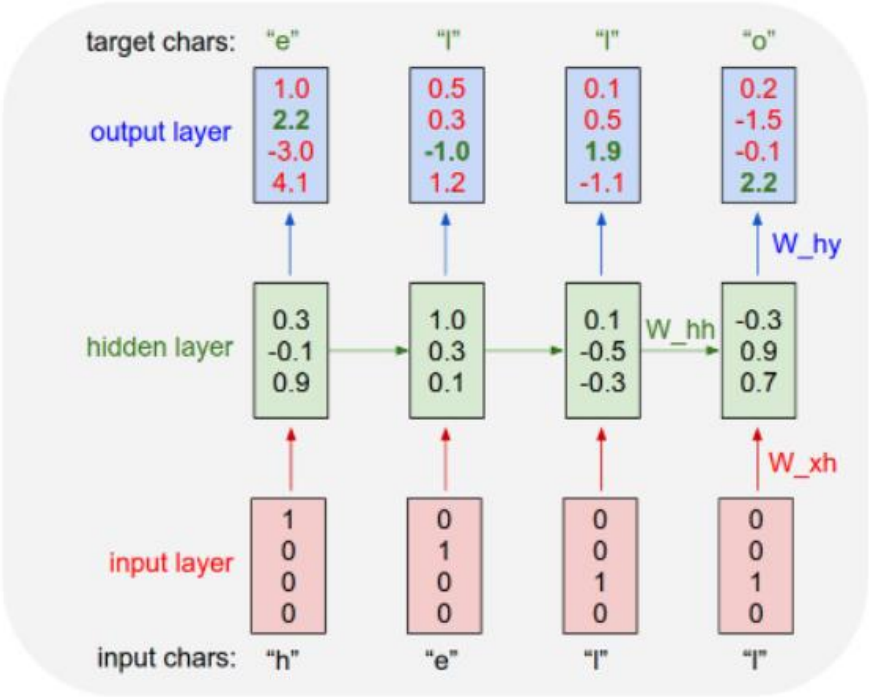


Рисунок 14 – Предсказывание следующего символа архитектурой LSTM

На входной слой «input layer» нейронной сети подается слово «hell», для каждого символа формируется вектор скрытого состояния на слое «hidden layer», который предсказывает следующий символ, который выводится на выходной слой «output layer». Таким образом, система предсказывает последний символ «o» слова «hello».

Главной целью схемы, основанной на символах, является оценка распределения $P(x_{0:T})$ над последовательностью символов $(x_0, x_1, \ldots, x_T) = x_{0:T}$.

В процессе тренировки модели происходит обучение $P(x_t|x_0, x_1, \ldots, x_{t-1})$, в результате которого определяются оценки распределения прогноза следования следующих символов на основе предыдущих.

Тогда совместное распределение по целым предложениям может быть представлено как произведение прогнозируемых распределений символов на основе предыдущих:

$$P(x_{0:T}) = \prod_{t=0}^{T} P(x_t|x_{0:t-1}).$$

В LSTM условная вероятность $P(x_t|x_{0:t-1})$ приближенно равна выходным значениям скрытого слоя h_t:

$$P(x_t|x_{0:t-1}) \approx \prod_{t=0}^{T} P(x_t|h_t;\rho), \tag{5}$$

где h_t представляет из себя последовательность предыдущих символов, вычисленные с помощью ячеек памяти C_t.

$$h_t(x_{0:t-1}) = f_h(x_{t-1}, h_{t-1}, C_{t-1};\rho) \tag{6}$$

$$C_t(x_{0:t-1}) = f_c(x_{t-1}, h_{t-1}, C_{t-1};\rho), \tag{7}$$

где ρ означает параметры модели.

Начальные значения h_{-1} и C_{-1} в начале работы сети определяются нулями. Вычисление h_t и C_t было рассмотрено в 5.2.

Вероятность каждого символа определяется формулой

$$P(x_t|h_t;V) = softmax(Vh_t + b) = \frac{\exp(Vh_t+b)}{\|\exp(Vh_t+b)\|},$$

где V и b являются частью параметров ρ и представляют из себя веса и биасы, соответственно.

Помимо описанной однонаправленной модели LSTM в пакете Flair используется двунаправленная BiLSTM. Это означает, что наряду с моделью, сканирующей текст от начала до конца, существует модель, которая работает

в обратном направлении. Данная схема позволяет создать контекстуализированные эмбеддинги слов.

В обратной LSTM условная вероятность $P^b(x_t|x_{t+1:T})$ приближенно равна выходным значениям скрытого слоя h_t:

$$P^b(x_t|x_{t+1:T}) \approx \prod_{t=0}^{T} P^b(x_t|h_t^b;\rho),$$

где h_t^b представляет из себя последовательность символов, вычисленных с помощью ячеек памяти C_t^b.

$$h_t^b(x_{t+1:T}) = f_h^b(x_{t+1}, h_{t+1}^b, C_{t+1}^b;\rho)$$
$$C_t^b(x_{t+1:T}) = f_c\big(x_{t+1}, h_{t+1}^b, C_{t+1}^b;\rho\big).$$

Стоит уточнить, что в далее в формулах 5, 6, 7 будут использоваться верхние индексы f для обозначения формул, которые работают «вперед», а индексы b означают, что данная LSTM является обратной и работает «назад».

Наглядно работа BiLSTM представлена на рисунке 15.

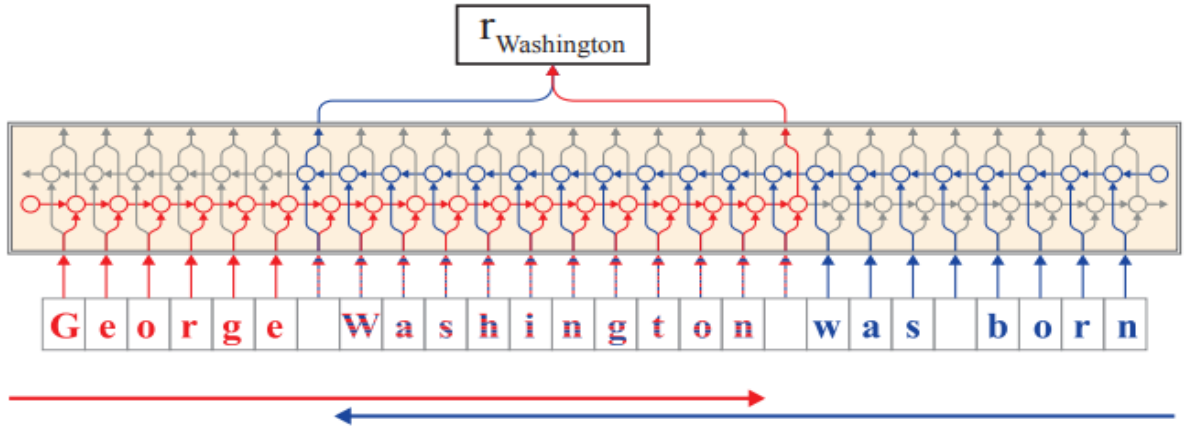


Рисунок 15 – Демонстрация работы BiLSTM

В результате прямой работы LSTM, которая на рисунке обозначена красной линией, извлекается выходное скрытое состояние h_t^f после последнего символа в слове. Так как f-LSTM обучено предсказывать вероятное продолжение текста, который идет после каждого символа, скрытое состояние кодирует семантико-синтаксическую информацию от начала текста и до конца этого слова. Аналогичным образом действует и b-LSTM, обозначенная на рисунке 1 – извлекается выходное скрытое состояние перед первым символом слова для того, чтобы получить семантико-синтаксическую информацию от конца текста до этого символа.

Выходные скрытые состояния f-LSTM и b-LSTM объединяются для формирования окончательного эмбеддинга слова и захвата семантико-синтаксической информации вместе с контекстом, в котором употребляется слово.

Если условиться, что каждое слово в тексте начинается с индексов $t_0, t_1, \dots, t_n$, можно определить контекстуализированные эмбеддинги слов следующим образом:

$$w_i^{CharCE} = \begin{bmatrix} h^f_{t_{i+1}-1} \\ h^b_{t_i-1} \end{bmatrix}.$$

Таким образом, данный подход создает эмбеддинги из скрытых состояний, которые вычисляются не только для символов слова, но и для символов контекста, в котором слово находится.

Это дает возможность создавать различные эмбеддинги для одной и той же строки слов в разных контекстах и точно фиксировать семантику контекстуального использования вместе с самой семантикой слов и предсказывать принадлежность этой строки к классу негативных текстов или позитивных.

Для того, чтобы предсказать контекст на вход сети BiLSTM подаются слова: $w_0, w_1, \dots, w_0$. выходные значения BiLSTM выражаются в виде векторов r_i^f и r_i^b прямой и обратной LSTM:

$$r_i = \begin{bmatrix} r_i^f \\ r_i^b \end{bmatrix}$$

Финальные вероятности для каждого контекста y высчитываются с помощью метода CRF:

$$\hat{P}(y_{0:n}|r_{0:n}) = \prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, r_i),$$

где $\psi_i(y', y, r) = \exp(W_{y',y} r + b_{y',y})$.

Для улучшенной работы тонального предсказания используется совмещение эмбеддингов слов CharCE, основанных на контексте и классический эмбеддингов Word2Vec:

$$w_i = \begin{bmatrix} w_i^{CharCE} \\ w_i^{\text{Word2Vec}} \end{bmatrix}.$$

Данный вид эмбеддинга, представляющий комбинацию $CharCE$ и Word2Vec, способен определять для каждого слова контекст; в зависимости от контекста у одного и того же слова могут иметься разные векторные представления тональности. Таким образом, когда программа анализирует текст, она анализирует контексты или группы из слов, которые в совокупности несут в себе эмоциональный окрас.

4 Практическая реализация

4.1 Формулировка задачи

Функциональная схема программной системы представлена на рисунке 16.

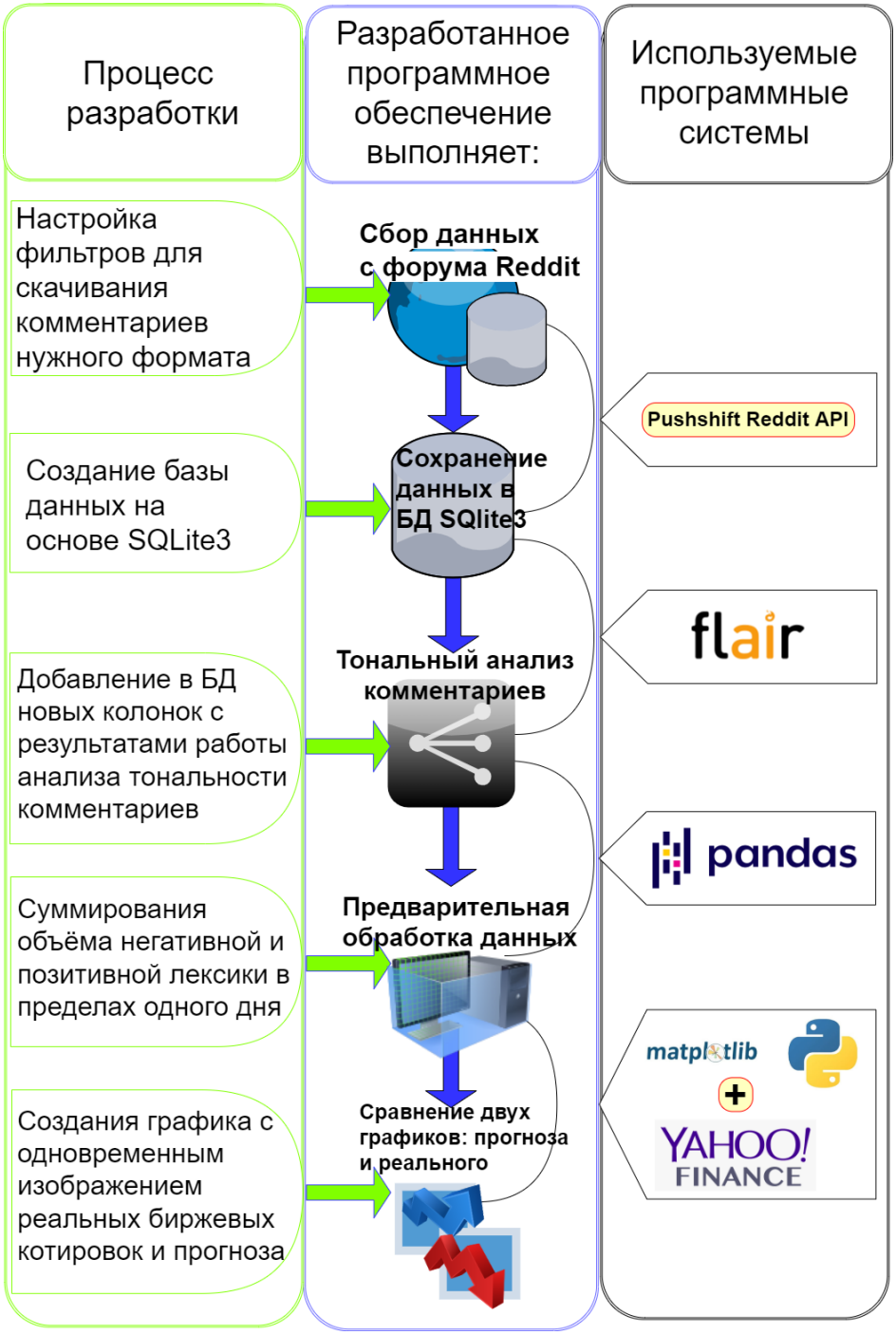


Рисунок 16 – Функциональная схема разработанного программного обеспечения

В схеме кратко описана работа разработанного программного обеспечения. В синей области представлены основные действия, выполняемые разработанным программным обеспечением. В чёрной области представлены готовые программные системы, с помощью которых осуществляется переход от одного действия в другому. В зелёной области описаны команды и действия разработчика благодаря которым программа выполняется.

1. Для того чтобы осуществить анализ комментариев о компании Tesla, необходимо создать базу данных и осуществить сбор данных с Web-ресурса.

Для хранения информации используется база данных SQLite – компактная встраиваемая СУБД, которая проста в обращении и поэтому идеально подходит для визуального анализа загруженных данных, которые занимают небольшой объём памяти на персональном компьютере.

В качестве источника данных используется сайт Reddit, сочетающий в себе черты социальной сети и форума. Reddit один из самых популярных сайтов в мире и занимает 21-е место по посещаемости по данным интернет-ресурсов Alexa Internet и SimilarWeb.

Сбор данный осуществляется с помощью программного интерфейса приложения - Pushshift Reddit API. Данный API позволяет делать запросы сайту Reddit, используя фильтры, которые позволяют загружать только те комментарии, в которых присутствует слово «Tesla».

В настоящей работы загружаются комментарии за 2019 год, так в течении этого года в мире была относительно стабильная политическая обстановка.

Вид загруженных данных представлен в таблице 4.

Таблица 4 – База данных комментариев

	index	created_date	created_date_epoch	subreddit	post_id	text_of_comment
247478	921	29-12-2019 23:48:47	1 577 663 327	electronic_cigarette	ehawvd	Everyone trashes smok but we all gotta start somewh
247479	920	29-12-2019 23:48:49	1 577 663 329	TeslaLounge	ehbzj1	I thought you couldn't use foam in Tesla tyres
247480	919	29-12-2019 23:49:00	1 577 663 340	thebachelor	ehdjij	As a current intern (who is v happy to get paid), I cou
247481	918	29-12-2019 23:50:39	1 577 663 439	HistoryMemes	eh88ub	Obviously it was before Edison invented Nicholas Te
247482	917	29-12-2019 23:50:47	1 577 663 447	electricvehicles	eh9hz8	> Taycan has great performance but is overweigh
247483	916	29-12-2019 23:51:06	1 577 663 466	teslamotors	eh9p5n	Anti-Tesla fans making stuff up in their mind. Noboc
247484	915	29-12-2019 23:51:31	1 577 663 491	wallstreetbets	eh9niz	If your total empire generates 30 billion why suicide
247485	914	29-12-2019 23:51:34	1 577 663 494	wallstreetbets	eh75pc	They going to release Tesla killer soon...
247486	912	29-12-2019 23:52:12	1 577 663 532	teslamotors	eh9p5n	This was my best friend. He was going to be the best
247487	913	29-12-2019 23:52:12	1 577 663 532	TeslaModel3	ehc3qe	It seems to be following your navigation. The speed
247488	911	29-12-2019 23:52:15	1 577 663 535	technology	egvy38	That's the whole point of autonomous cars... as in th
247489	910	29-12-2019 23:52:28	1 577 663 548	TeslaLounge	eh4j64	My MX sold 12/18 does this. Took it in and the servic
247490	909	29-12-2019 23:52:39	1 577 663 559	AmltheAsshole	ehdqyu	^^^^AUTOMOD ***The following is a copy of the a
247491	908	29-12-2019 23:52:41	1 577 663 561	newjersey	eh6d3j	laughs for 15-30 minutes in tesla. swiping to fill up
247492	907	29-12-2019 23:54:20	1 577 663 660	electricvehicles	ehcfp0	A dealership in NJ did the same with the Ioniq past n
247493	906	29-12-2019 23:56:10	1 577 663 770	teslamotors	eh6737	ALL OF THIS plus coming from a car manufacturer w
247494	905	29-12-2019 23:56:12	1 577 663 772	WarhammerCompeti	ehd0m3	A quick check of your math (at least just through Ma
247495	904	29-12-2019 23:56:39	1 577 663 799	teslamotors	eh6737	I am going to be driving one in a couple of weeks at
247496	903	29-12-2019 23:57:08	1 577 663 828	cars	ehaf61	There can, but is there? Is there any electric car on th
247497	902	29-12-2019 23:57:59	1 577 663 879	Glitch_in_the_Matrix	eh7vhk	Just read about Tesla, nothing much to say.
247498	901	29-12-2019 23:59:11	1 577 663 951	technews	egbg4v	Lmao. I love how you tell me that I know nothing of
247499	900	29-12-2019 23:59:49	1 577 663 989	ShitAmericansSay	eh0hpv	>Tesla, is now profitable businesses¶¶Lol¶¶>w

Всего было загружено 247499 комментариев, содержащих слово «Tesla» за весь 2019 год.

В таблице вся необходимая информация содержится в колонках:

- created_date – дата создания комментария;
- created_date_epoch – дата создания комментария в системе эпох, для удобства сортировки;
- subreddit – название сообщества в одном из обсуждении которого был оставлен комментарий;
- post_id – id обсуждения, в котором был оставлен комментарий;
- text_of_comment – текст комментария.

2. Анализ тональности комментариев проводится с помощью пакета – Flair. Работа программы была подробно рассмотрена в 5 главе.

Главным преимуществом пакета является возможность подсчёта вероятности принадлежности комментария к позитивному или негативному классу, а не только его обозначение как негативный или позитивный.

3. После анализа тональности каждого комментария в базе данных создаются две новые колонки с именами (Таблица 5):

Таблица 5 – База данных комментариев с результатами анализа тональности

	te	123 created_date_epoch	ABC subreddit	ABC post_id	ABC text_of_comment	123 probability	ABC sentiment
21379	00:42	1 548 964 842	teslamotors	alr2fn	I'm saying this right now, if done right this pickup w	0,6285261512	POSITIVE
21380	00:06	1 548 964 806	DeadBedrooms	akund3	> Let's go back to the car analogy. I'm selling, yo	-0,9999858141	NEGATIVE
21381	59:01	1 548 964 741	RealTesla	alawr6	> It would be very helpful to know Gigafactory er	-0,9995421171	NEGATIVE
21382	58:04	1 548 964 684	teslamotors	althml	>Instead, we have a company seemingly on the c	0,9805665016	POSITIVE
21383	55:56	1 548 964 556	teslamotors	alfryr	> Repair complexity wise, the Model 3 is much cl	-0,9980569482	NEGATIVE
21384	55:42	1 548 964 542	Futurology	aln8xg	Informative slashdot discussion: https://tech.slashdc	-0,9973909259	NEGATIVE
21385	55:10	1 548 964 510	Surface	alrbld	Oh please, the cops don't want to piss off the Detroit	-0,9439794421	NEGATIVE
21386	55:01	1 548 964 501	teslainvestorsclub	alnoeg	Tesla took 80% market share of the BEV market, all w	0,7472516298	POSITIVE
21387	54:21	1 548 964 461	idleapocalypse	alrzxv	Long before the upgrade that allowed you to max it	-0,9997087121	NEGATIVE
21388	53:05	1 548 964 385	DebateCommunism	alnhag	Tesla's first deal with Westinghouse was 3 cents for e	-0,9996907711	NEGATIVE
21389	52:58	1 548 964 378	RealTesla	alarz7	> I DO think there is a population of OCD buyers	-0,9981313348	NEGATIVE
21390	51:59	1 548 964 319	teslamotors	alfryr	I got the performance. Then I got 18" all seasons so	-0,9997848868	NEGATIVE
21391	51:37	1 548 964 297	cars	alp09d	I feel like Tesla could use a few more guys like you lo	0,9979148507	POSITIVE
21392	50:04	1 548 964 204	trees	alsrkg	I am the Tesla	0,9998742342	POSITIVE
21393	48:33	1 548 964 113	PewdiepieSubmissi	alqvmk	Jamie, pull up that video of me destroying the Tesla	-0,9996796846	NEGATIVE
21394	48:23	1 548 964 103	TeslaModel3	altal4	I would check with ChargeHub and see if there are a	-0,9703984261	NEGATIVE
21395	48:07	1 548 964 087	ClashRoyale	alnc76	One freeze is all it takes for a tower to be lost. I don't	0,7160443068	POSITIVE

- probability – процентное соотношение лексики тональности класса sentiment в тексте над противоположной, принимает значение в диапазоне [±0,5; ±99999];
- sentiment —указание класса «NEGATIVE» или «POSITIVE», которому отнесен комментарий.

4. Нормализация данных относительно фиксированного промежутка времени. В рамках одного дня производится подсчет суммы всего элементов колонки probability. Так как значение probability для негативных комментариев в таблице всегда – отрицательное число, а позитивных наоборот – положительное, то сумма даст перевес тональности в течении дня; либо в позитивную либо в негативную сторону. Таким образом, можно понять, какой, в информационном плане, был день, негативный или позитивный.

5. С помощью программного интерфейса приложения Yahoo! Finance's API осуществляется скачивание данных дневных цен на акции компании Tesla на один месяц. Далее, подсчитывается среднее арифметическое всех стоимостей акций торгуемых в месяце по дням mean1 и среднее арифметическое тональности дней в таком же месяце mean2. Каждому значению тональности дня прибавляется разница средних арифметических значений: mean1 – mean2. В результате получается наложенный график прогноза на реальный график стоимости акций.

4.2 Графики прогнозов и реальных стоимостей акций

На графике (Рисунок 17) проиллюстрировано сравнение графика прогноза и реального графика стоимости акций за январь 2019 г.

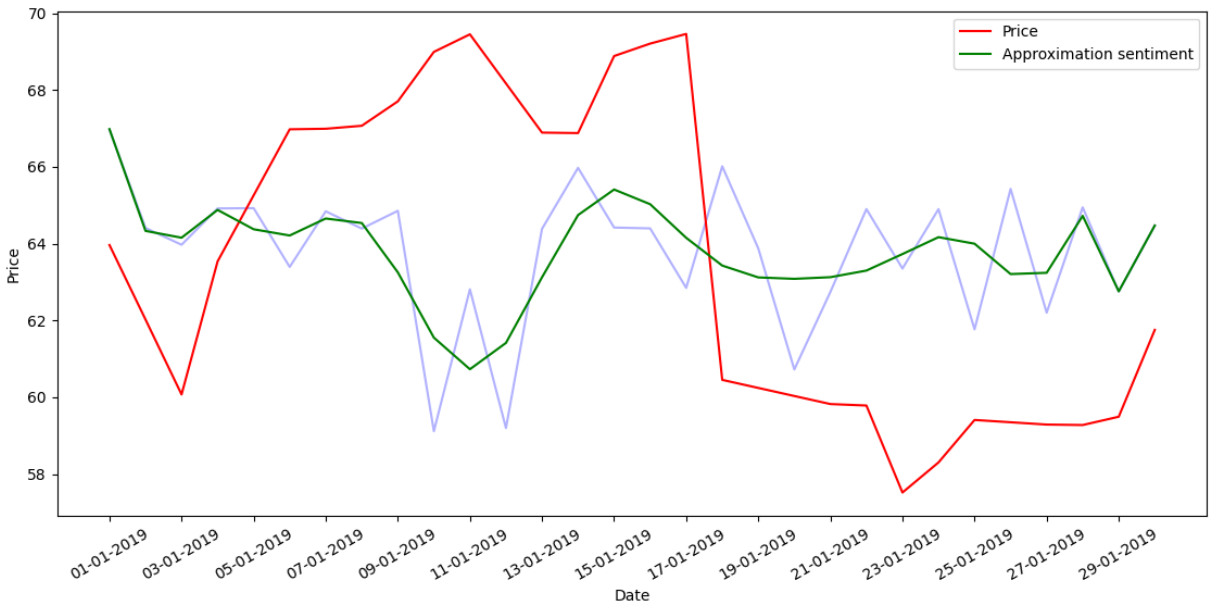


Рисунок 17 – График прогноза и график реальных данных о стоимости акций Tesla за январь 2019-го года

На рисунке 17:

- Синий график – это значения суммированных показателей лексического системы на каждый день;
- Зеленый график – аппр{оксимированные значения жёлтого графика;
- Красный график — настоящий график цен на акций.

Можно заметить, что зеленый график весьма точно показывает направление движение красного графика, более того, делает это раньше, чем начнет свое движение красный график.

Обратим особое внимание на 8 января - график прогноза и реальный график расходятся, при чем весьма существенно. Стоит отметить, что система, естественно, не способна идеально предсказывать движение цен акций,

потому что иногда на рынке и в информационном поле возникают непредвиденные события.

Рассмотрим этот случай.

До 8-го января дела у компании Tesla шли не очень благополучно, несмотря на то, что Tesla закончила 2018 год с рекордными показателями производства.

Однако инвесторов больше заботили будущие проблемы, такие как развитие конкуренции среди знаменитый автопроизводителей, поэтапный отказ от налоговых льгот для аккумуляторных автомобилей Tesla. Таким образом, информационный фон был негативным, однако 8го числа появилась новость о том, что Tesla начинает строить свои фабрики в Китае, что неизбежно привело к росту цен на акции. Но дело в том, что было очевидно, что Tesla, исходя из финансовых показателей, не сможет быстро построить эти фабрики, то есть, эта новость была, скорее всего, манипулятивной, которая, по сути, проблем компании Tesla не решала. Более того, 11-го января стали очевидны проблемы с прибыльностью компании, когда она выпустила сообщение, что сокращает производство разнообразия комплектации путем ограничения разнообразия опций, расцветок и отделки салона, и акции компании пошли вниз.

Далее можно увидеть падение акций 18-го января. В данном случае вышла новость на BBC о том, что Tesla сокращает тысячи сотрудников, чтобы сохранить прибыльность компании, закономерно стоимость акции сильно упала.

Рассмотрим график за февраль 2019 года изображенный на рисунке 18. Можно заметить, что график прогноза предсказывает снижение или рост стоимости акций за пару дней, до того, как движение стоимости происходит на бирже, что является наглядным примером успешно проведенной работы.

Рисунок 18 – График прогноза и график реальных данных об стоимости акций Tesla за февраль 2019-го года

В приложении А можно ознакомиться с большим числом графиков.

Выводы

Проведенное исследование возможностей нейронных сетей в решении задачи автоматизированного прогноза краткосрочного движения стоимости ценных бумаг с использованием методов обработки текстов на естественном языке (NLP) позволило прийти к следующим выводам:

1. В течение последних нескольких лет анализ эмоциональной окрашенности текста и эмоциональной оценки активно применяется для прогнозирования цен на рынке биржевых акций.
2. Существует несколько подходов для решения задачи тональности текста:

- Подход на основе словарей тональности;
- Подход машинного обучения без учителя;
- Подход машинного обучения с учителем.

Подход машинного обучения с учителем, использованный в данной работе, показывает самые высокие точности классификации. Это связано с

тем, что для обучения классификатора на определенную тему формируется свой собственный словарь.

3. Исследование существующих пакетов показало, что, в основном, готовые решения предназначены для работы с английским языком. Выбор англоязычного пакета Flair обусловлен наличием достаточного для анализа количества обсуждений на английском языке на заданную тему, подтверждающих высокую точность определения тональности текста.

4. Использованный в работе пакет Flair, разработанный для анализа тональности отзывов о кинофильмах на сайте IMDb, показывает высокую точность – свыше 80% при определении тональности комментариев о деятельности компаний – эмитентов акций.

Таким образом, разработанное программное средство на основе данного пакета применимо в прогнозировании роста или снижения стоимости ценных бумаг.

ЗАКЛЮЧЕНИЕ

В рамках данной работы рассмотрена задача автоматического определения тональности текста, исследованы существующие методы и пакеты анализа текстов. Разработано программный пакет для автоматического сбора исходных данных и проведения анализа тональности комментариев пользователей социальных сетей с последующим построением графиков прогноза роста или снижения стоимости акций.

В ходе работы решены следующие задачи:

1. Исследованы существующие методы и подходы анализа текстов, используемые в области искусственного интеллекта, нейронных сетей.
2. Рассмотрены существующие программные пакеты автоматического анализа тональности текстов и выбран наиболее оптимальный для определения эмоциональной окраски текста комментариев в социальных сетях.
3. Разработано программное обеспечение, реализующее:

- автоматическое заполнение базы данных комментариев из сайта Reddit на тему, связанную с деятельностью компании Tesla;
- анализ тональности собранных комментариев с помощью пакета Flair;
- формирование и наложение графиков тональности собранных комментариев и реальных биржевых данных для последующего прогноза динамики стоимости акций.

4. Построенные графики демонстрируют, что рост/снижение стоимости акций следует за ростом/снижением тональности комментариев пользователей социальных сетей о деятельности компании.

Полученные результаты показывают, что выбранный метод анализа тональности текстов применим для использования в прогнозе роста или снижения стоимости акций.

Поставленные в ВКР задачи выполнены.

СПИСОК ЛИТЕРАТУРЫ

1. Anurag Nagar. Using Text and Data Mining Techniques to extract Stock Market Sentiment from Live News Streams/ Anurag Nagar, Michael Hahsler // IPCSIT. – 2012. – Vol. 20.
2. Jeffrey Breen. Mining Twitter for Airline Consumer Sentiment // Boston Predictive Analytics Meetup Group. – 2011. – Oct 25.
3. Oluwaseun Ajao. Sentiment Aware Fake News Detection on Online Social Networks/ Oluwaseun Ajao, Deepayan Bhowmik, Shahrzad Zargari, // IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) Vol. 2019.
4. Котельников Е. В. Автоматический анализ тональности текстов на основе методов машинного обучения. / Котельников Е. В. ,Клековкина М. В. // РОМИП. – 2011.
5. Abdulaziz. An Overview of Bag of Words / Abdulaziz, Wisam & M. Ameen, Musa & Ahmed, Bilal // Importance, Implementation, Applications, and Challenges. – 2019. – P. 200-204.
6. Mikolov T. Efficient Estimation of Word Representations in Vector Space / Mikolov T., Chen K., Corrado G., Dean J. // In Proceedings of Workshop at ICLR. – 2013.
7. Автоматическая обработка текстов на естественном языке и анализ данных / Большакова Е.И., Воронцов К.В., Ефремова Н.Э., Клышински Э.С., Лукашевич Н.В., Сапин А.С. – М. : НИУ ВШЭ, 2017. – С. 17-18.
8. Кулагин Д.И. Открытый тональный словарь русского языка КартаСловСент // Компьютерная лингвистика и интеллектуальные технологии: сб. материалов ежегодной Международной конференции «Диалог». Вып. 20. – М.: РГГУ, 2021. – С. 1106-1119.
9. Alan Akbik. Contextual String Embeddings for Sequence Labeling. / Alan Akbik, Duncan Blythe, Roland Vollgraf // In Proceedings of the 27th

International Conference on Computational Linguistics, Santa Fe, New Mexico, USA. Association for Computational Linguistics, 2018. – P. 1638–1649.

10. Sepp Hochreiter. Long short-term memory / Sepp Hochreiter; Jürgen Schmidhuber. // Neural Computation. – 1997. – Vol. 9, № 8. – P. 1735– 1780.

ПРИЛОЖЕНИЕ А

Графики результатов анализа тональности комментариев и реальных биржевых данных

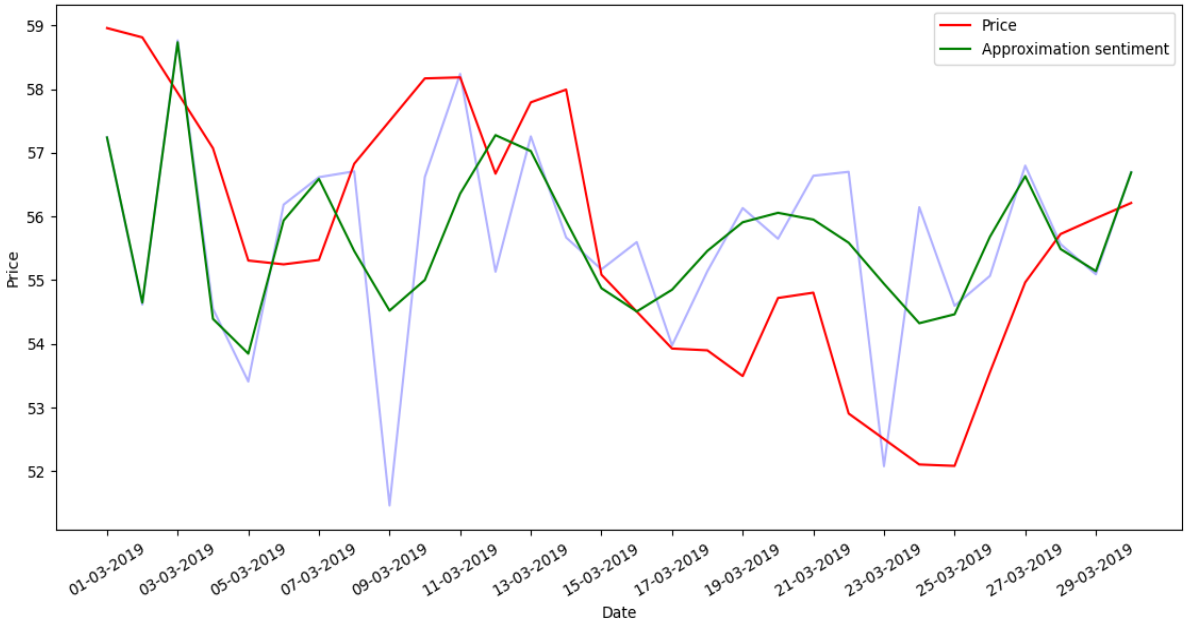


Рисунок А.1 – График прогноза и график реальных данных об стоимости акций Tesla за март 2019-го года.

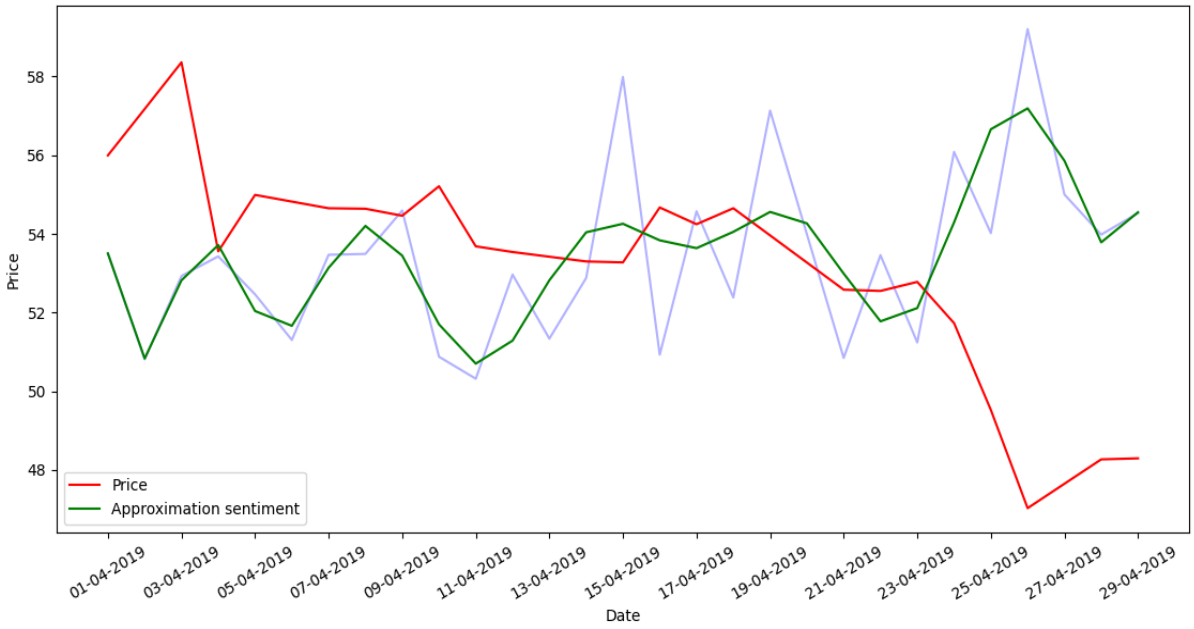


Рисунок А.2 – График прогноза и график реальных данных об стоимости акций Tesla за апрель 2019-го года.

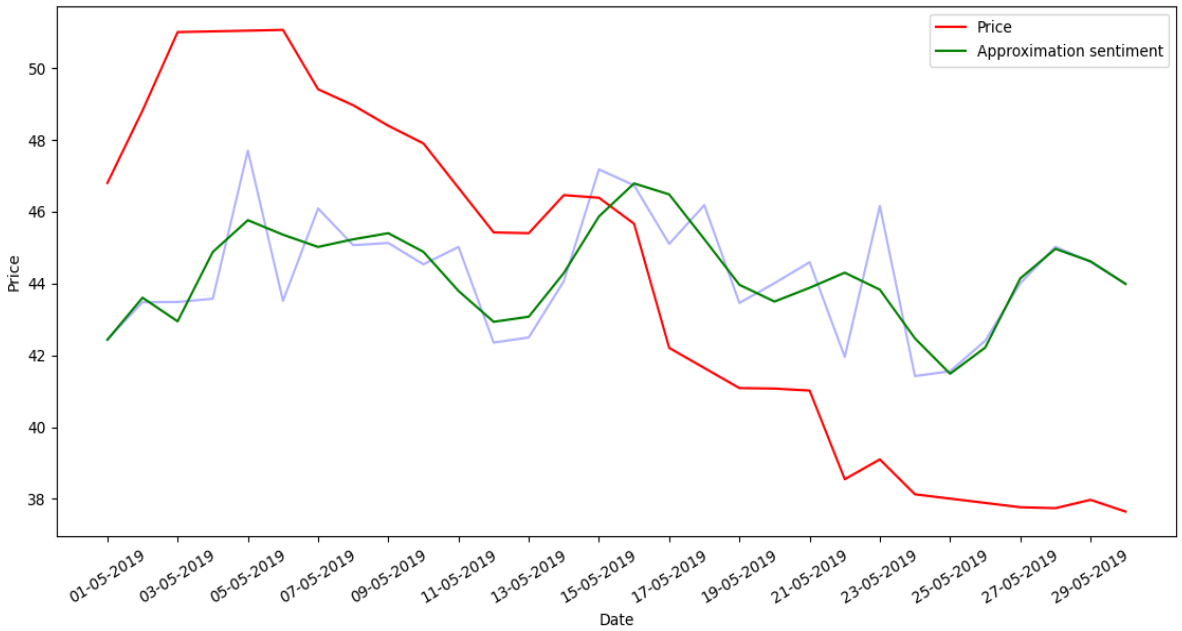


Рисунок А.3 – График прогноза и график реальных данных об стоимости акций Tesla за май 2019-го года.

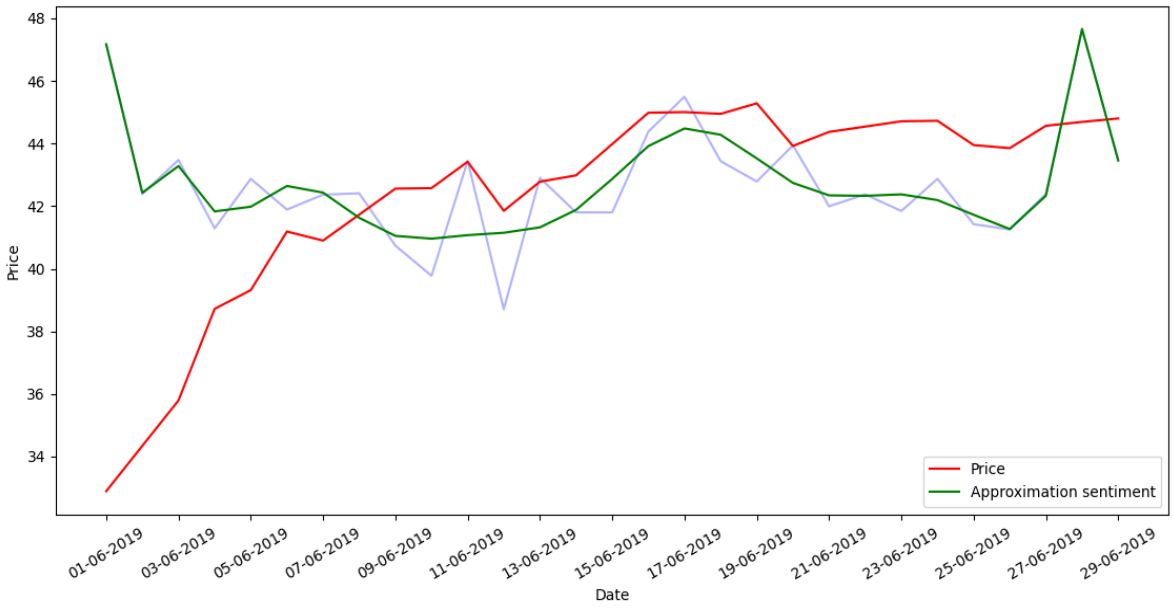


Рисунок А.4 – График прогноза и график реальных данных об стоимости акций Tesla за июнь 2019-го года.

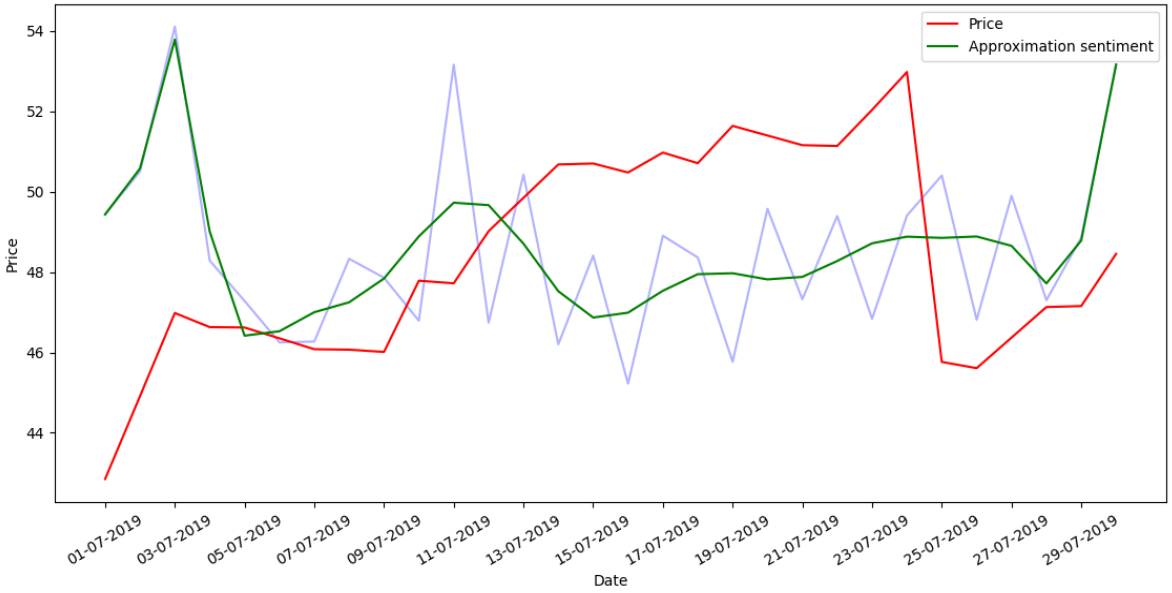


Рисунок А.5 – График прогноза и график реальных данных об стоимости акций Tesla за июль 2019-го года.

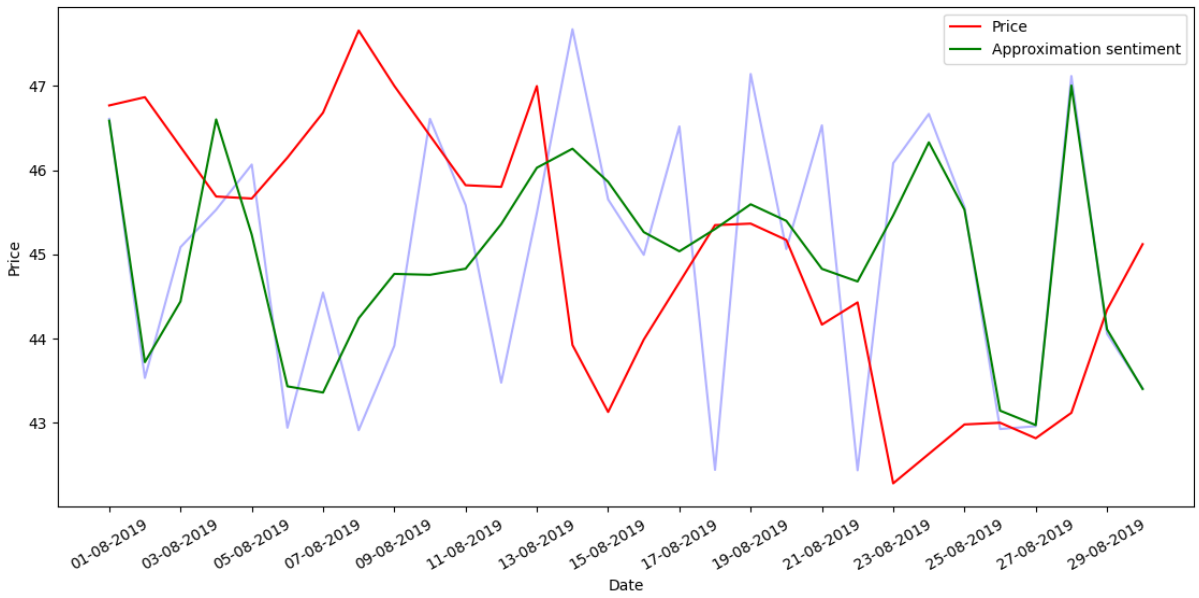


Рисунок А.6 – График прогноза и график реальных данных об стоимости акций Tesla за август 2019-го года.

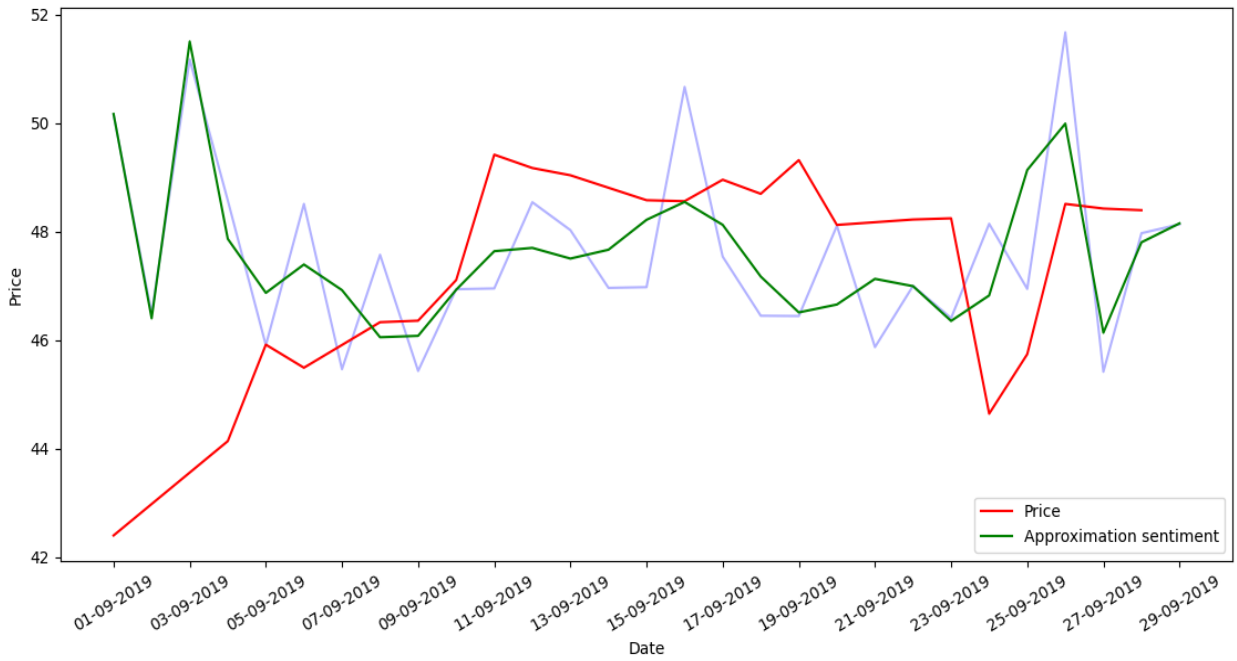


Рисунок А.7 – График прогноза и график реальных данных об стоимости акций Tesla за сентябрь 2019-го года.

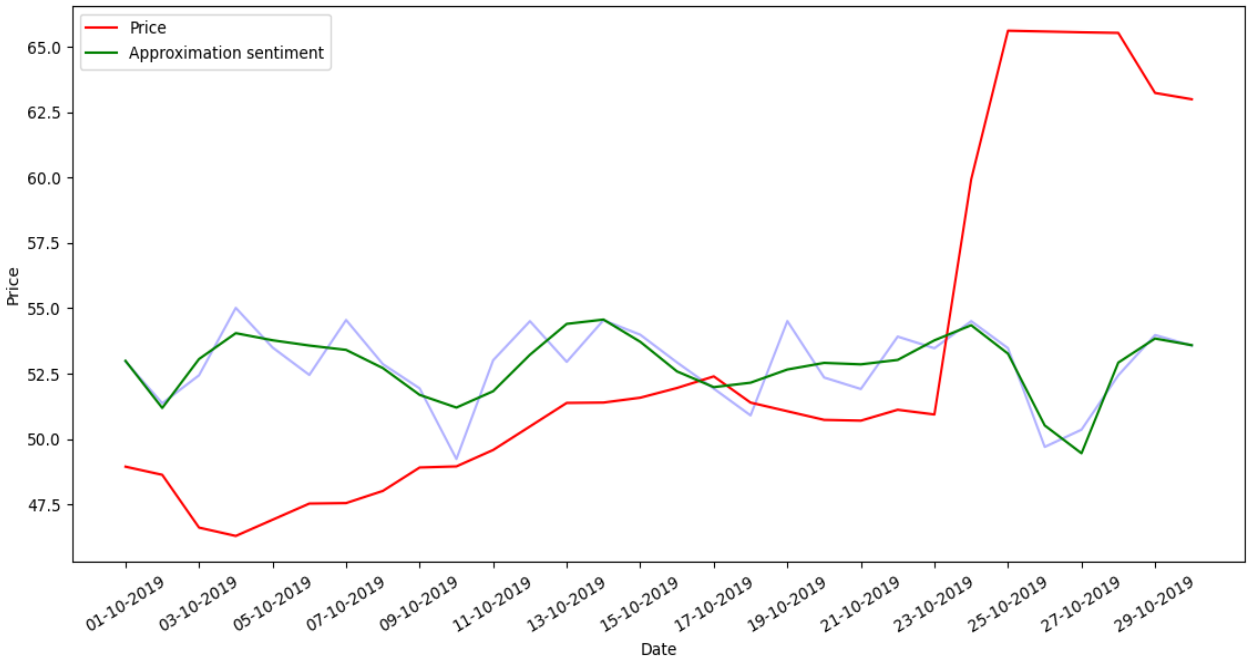


Рисунок А.8 – График прогноза и график реальных данных об стоимости акций Tesla за октябрь 2019-го года.

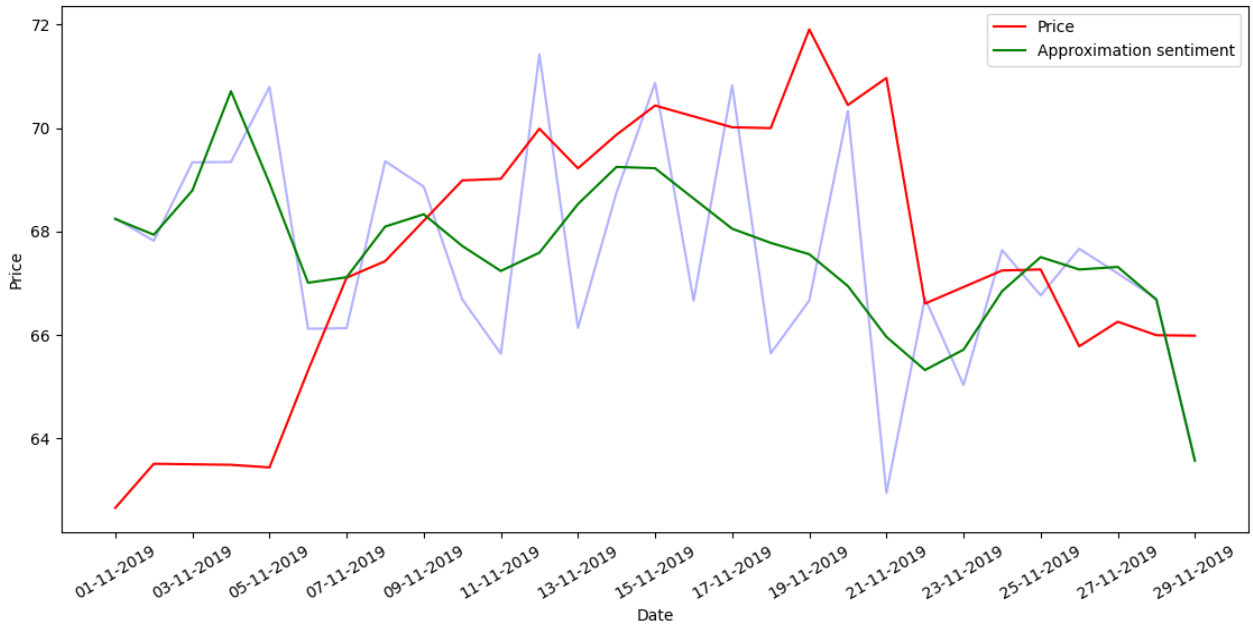


Рисунок А.9 – График прогноза и график реальных данных об стоимости акций Tesla за ноябрь 2019-го года.

ПРИЛОЖЕНИЕ Б

Код разработанного программного обеспечения

```
import pandas as pd
import praw
import datetime as dt
from datetime import datetime, timedelta
from pmaw import PushshiftAPI

from hurry.filesize import size, iec
import sqlite3
import flair
from time import perf_counter, perf_counter_ns
from timeit import default_timer as timer
from Benchmark import benchmark
import matplotlib.pyplot as plt
from requests import Session
from numba import jit, cuda
import numpy as np
import yfinance as yf
import replit
import scipy.stats
from scipy import spatial
from similaritymeasures import similaritymeasures
import collections


def epoch_date(date_string, separator = "-"):
    try:
        ret = int(dt.datetime.strptime(date_string,
f"%d{separator}%m{separator}%Y").timestamp())
    except ValueError:
        ret = False
    return ret

def epoch_to_str(ts_epoch):
    return dt.datetime.fromtimestamp(ts_epoch).strftime('%d-%m-%Y')

def get_submissions(after, before, subreddit, limit, isPrint = False):
    print("get_submissions start")
    api = PushshiftAPI()
    filter = ["title", "url"]
    listed = list(api.search_submissions(after=after,
                                         before=before,
                                         subreddit=subreddit,
                                         filter=filter,
                                         limit=limit))

    collection_of_submissions = [["created_date", "created_date_epoch",
"text_of_submission", "URL_Link"]]

    for submission in listed:
        dict_from_SM = submission[len(submission)-1]
        anotherList =
[datetime.fromtimestamp(dict_from_SM["created_utc"]).strftime('%d-%m-%Y
%H:%M:%S'),
                        dict_from_SM["created_utc"], dict_from_SM["title"],
dict_from_SM["url"]]
```

```
        collection_of_submissions.append(anotherList)

    submissions_dataframe = pd.DataFrame(collection_of_submissions[1:],
columns=collection_of_submissions[0])
    if isPrint:
        with pd.option_context('display.max_rows', None,
'display.max_columns', None, 'display.width', None, 'display.max_colwidth', -
1):  # more options can be specified also
            print(submissions_dataframe)
    print("get_submissions() finished")

def get_comments_by_subreddit(after, before, subreddit, limit = 10, isPrint =
False):

    print("get_comments start")
    api = PushshiftAPI()
    listed = []

    filter = ["created_utc", "subreddit","body","link_id", "permalink"]

    if filter == None or len(filter) == 0:
        listed = list(api.search_comments(after=after,
                                          before=before,
                                          subreddit=subreddit,
                                          limit=limit))
    else:
        listed = list(api.search_comments(after=after,
                                          before=before,
                                          subreddit=subreddit,
                                          filter=filter,
                                          limit=limit))

    collection_of_submissions = [["created_date", "created_date_epoch",
"subreddit", "post_id","text_of_comment", "URL_Link"]]

    for submission in listed:
        dict_from_SM = submission[len(submission)-1]
        anotherList =
[datetime.fromtimestamp(dict_from_SM["created_utc"]).strftime('%d-%m-%Y
%H:%M:%S'),
                       dict_from_SM["created_utc"],
                       dict_from_SM["subreddit"],
                       dict_from_SM["link_id"][3:],
                       dict_from_SM["body"],
                       "https://www.reddit.com"+dict_from_SM["permalink"]]
        collection_of_submissions.append(anotherList)

    submissions_dataframe = pd.DataFrame(collection_of_submissions[1:],
columns=collection_of_submissions[0])

    if isPrint:
        print(submissions_dataframe)

    print(f"get_comments() finished, dataframe size =
{size(submissions_dataframe.memory_usage(deep=True).sum(),system= iec)}"
          f", rows count = {submissions_dataframe.shape[0]}")

    return submissions_dataframe

def get_comments(after, before, q, limit = 10, isPrint = False):

    print("get_comments")
    api = PushshiftAPI()
```

```
    listed = []

    filter = ["created_utc", "subreddit","body","link_id", "permalink"]

    if filter == None or len(filter) == 0:
        listed = list(api.search_comments(after=after,
                                          before=before,
                                          q = q,
                                          limit=limit))
    else:
        listed = list(api.search_comments(after=after,
                                          before=before,
                                          q = q,
                                          filter=filter,
                                          limit=limit))

    collection_of_submissions = [["created_date", "created_date_epoch",
"subreddit", "post_id","text_of_comment", "URL_Link"]]

    for submission in listed:
        # dict_from_SM = submission[len(submission)-1]
        dict_from_SM = submission
        anotherList =
[datetime.fromtimestamp(dict_from_SM["created_utc"]).strftime('%d-%m-%Y
%H:%M:%S'),
                      dict_from_SM["created_utc"],
                      dict_from_SM["subreddit"],
                      dict_from_SM["link_id"][3:],
                      dict_from_SM["body"],
                      "https://www.reddit.com"+dict_from_SM["permalink"]]
        collection_of_submissions.append(anotherList)

    submissions_dataframe = pd.DataFrame(collection_of_submissions[1:],
columns=collection_of_submissions[0])

    if isPrint:
        print(submissions_dataframe)

    print(f"get_comments() finished, dataframe size =
{size(submissions_dataframe.memory_usage(deep=True).sum(),system= iec)}"
          f", rows count = {submissions_dataframe.shape[0]}")
    print("end get_comments")
    return submissions_dataframe

def get_from_sqlite3(name_of_SQLite_file = "", start_Date = "", end_Date =
"", query_SELECT = "", query_WHERE = "", isSentiment = False):

    query = ""
    epoch_1Day = 86400
    start_epoch = epoch_date(start_Date)
    end_epoch = epoch_date(end_Date)
    if end_epoch: end_epoch += epoch_1Day
    epoch_30Days = 2592000


    if query_SELECT != "":
        query = f"SELECT {query_SELECT} FROM {name_of_SQLite_file}"
    else:
        if isSentiment:
            query = f"SELECT created_date, created_date_epoch, subreddit,
post_id,text_of_comment, probability, sentiment, URL_Link FROM
{name_of_SQLite_file}"
        else:
```

```
            query = f"SELECT created_date, created_date_epoch, subreddit,
post_id,text_of_comment, URL_Link FROM {name_of_SQLite_file}"



    if start_epoch and end_epoch:
        query += f" WHERE created_date_epoch >= {str(start_epoch)} AND
created_date_epoch <= {str(end_epoch)}"
    elif start_epoch and end_epoch == False:
        query += f" WHERE created_date_epoch >= {str(start_epoch)} AND
created_date_epoch <= {str(start_epoch + epoch_30Days)}"
    elif start_epoch == False and end_epoch:
        query += f" WHERE created_date_epoch >= {str(end_epoch-epoch_30Days)}
AND created_date_epoch <= {str(end_epoch)}"

    if query_WHERE != "":
        if "WHERE" in query:
            query += f" AND {query_WHERE}"
        else:
            query += f" WHERE {query_WHERE}"

    db = sqlite3.connect(f"{name_of_SQLite_file}.db")
    df = pd.read_sql_query(query, db)
    db.close()
    return df

@jit
def get_from_sqlite4(name_of_SQLite_file = "", start_Date = "", end_Date =
"", query_SELECT = "", query_WHERE = "", isSentiment = False):

    query = ""
    epoch_1Day = 86400
    start_epoch = epoch_date(start_Date)
    end_epoch = epoch_date(end_Date)
    if end_epoch: end_epoch += epoch_1Day
    epoch_30Days = 2592000


    if query_SELECT != "":
        query = f"SELECT {query_SELECT} FROM {name_of_SQLite_file}"
    else:
        if isSentiment:
            query = f"SELECT created_date, created_date_epoch, subreddit,
post_id,text_of_comment, probability, sentiment, URL_Link FROM
{name_of_SQLite_file}"
        else:
            query = f"SELECT created_date, created_date_epoch, subreddit,
post_id,text_of_comment, URL_Link FROM {name_of_SQLite_file}"



    if start_epoch and end_epoch:
        query += f" WHERE created_date_epoch >= {str(start_epoch)} AND
created_date_epoch <= {str(end_epoch)}"
    elif start_epoch and end_epoch == False:
        query += f" WHERE created_date_epoch >= {str(start_epoch)} AND
created_date_epoch <= {str(start_epoch + epoch_30Days)}"
    elif start_epoch == False and end_epoch:
        query += f" WHERE created_date_epoch >= {str(end_epoch-epoch_30Days)}
AND created_date_epoch <= {str(end_epoch)}"

    if query_WHERE != "":
        if "WHERE" in query:
```

```
            query += f" AND {query_WHERE}"
        else:
            query += f" WHERE {query_WHERE}"

    db = sqlite3.connect(f"{name_of_SQLite_file}.db")
    df = pd.read_sql_query(query, db)
    db.close()
    return df

def do_research(df):
    df_text = df["text_of_comment"]
    sentiment_model = flair.models.TextClassifier.load('en-sentiment')
    koef = 0

    listed_probability = []
    listed_sentiment = []

    for text in df_text:
        sentence = flair.data.Sentence(text)
        sentiment_model.predict(sentence)
        if sentence.labels[0].value == "POSITIVE":
            koef = 1
        elif sentence.labels[0].value == "NEGATIVE":
            koef = -1

        listed_probability.append(sentence.labels[0].score * koef)
        listed_sentiment.append(sentence.labels[0].value)

    df.insert(5, 'probability', listed_probability)
    df.insert(6, 'sentiment', listed_sentiment)

    return df
@jit
def predi(sentiment_model, sentence):
    sentiment_model.predict(sentence)
    return sentence

def do_research2(df):
    df_text = df["text_of_comment"]
    sentiment_model = flair.models.TextClassifier.load('en-sentiment')
    koef = 0

    listed_probability = []
    listed_sentiment = []

    for text in df_text:
        sentence = flair.data.Sentence(text)
        sentence = predi(sentiment_model, sentence)
        if sentence.labels[0].value == "POSITIVE":
            koef = 1
        elif sentence.labels[0].value == "NEGATIVE":
            koef = -1

        listed_probability.append(sentence.labels[0].score * koef)
        listed_sentiment.append(sentence.labels[0].value)

    df.insert(5, 'probability', listed_probability)
    df.insert(6, 'sentiment', listed_sentiment)

    return df

def dataframe_to_sqlite(df, filename = "basket", if_exists = "fail"):
    db = sqlite3.connect(f"{filename}.db")
```

```
    df.to_sql(filename, db, if_exists=if_exists, index=True)
    db.close()


def proba(name_of_SQLite_file5, start_Date="", end_Date="", concrete_step =
0, difference_procent = 2.5, mode = ""):
    epoch_1Day = 86400

    listed = [["date", "points"]]
    indent = 0.5
    ldif = difference_procent - indent
    rdif = difference_procent + indent
    step = 0.511
    right_step = 0.51
    right_dif_procent = 0.51
    mindif = 100
    is_catch_diapzone = False

    if concrete_step == 0:
        while is_catch_diapzone == False:
            while step < 0.99999999:
                df = get_from_sqlite3(name_of_SQLite_file5,
start_Date=start_Date, end_Date=end_Date,
                                      query_WHERE= f"probability <
{str(step)} AND probability > -{str(step)}", isSentiment= True)
                df_positive = df.loc[df['sentiment'] == "POSITIVE"]
                df_negative = df.loc[df['sentiment'] == "NEGATIVE"]
                pos = df_positive.shape[0]/(df.shape[0]/100)
                neg = df_negative.shape[0]/(df.shape[0]/100)
                this_dif_procent = abs(neg - pos)

                if mode == "":
                    if ldif <= this_dif_procent <= rdif:
                        right_dif_procent = this_dif_procent
                        right_step = step
                        is_catch_diapzone = True

print("||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
|||||")
                elif mode == "minimum difference_procent":
                    if this_dif_procent < mindif:
                        mindif = this_dif_procent
                        right_step = step
                        is_catch_diapzone = True

                print(f"step {step} \n"
                      f"positives {df_positive.shape[0]} negatives
{df_negative.shape[0]} \n"
                      f"df rows {df.shape[0]} procent {pos} {neg}\n"
                      f"procent difference {this_dif_procent} \n")
                step += 0.01
            if is_catch_diapzone == False:
                step = 0.511
                right_step = 0.51
                mindif = 100
                indent += 0.1
                ldif = difference_procent - indent
                rdif = difference_procent + indent
            else:
                print(f"right_step {right_step} this_dif_procent
{right_dif_procent}")

    elif concrete_step > 0:
```

```
        right_step = concrete_step
        df = get_from_sqlite3(name_of_SQLite_file5, start_Date=start_Date,
end_Date=end_Date,
                              query_WHERE=f"probability < {str(right_step)}
AND probability > -{str(right_step)}",
                              isSentiment=True)
        df_positive = df.loc[df['sentiment'] == "POSITIVE"]
        df_negative = df.loc[df['sentiment'] == "NEGATIVE"]
        pos = df_positive.shape[0] / (df.shape[0] / 100)
        neg = df_negative.shape[0] / (df.shape[0] / 100)
        right_dif_procent = abs(neg - pos)
        print(f"right_step {right_step} this_dif_procent
{right_dif_procent}")


    df = get_from_sqlite3(name_of_SQLite_file5, start_Date=start_Date,
end_Date=end_Date,
                          query_WHERE=f"probability < {str(right_step)} AND
probability > -{str(right_step)}", isSentiment=True)
    df.sort_values(by='created_date_epoch', ascending=True, inplace=True,
ignore_index=True)

    Day = df.loc[0, "created_date_epoch"]
    last_Day = df.loc[df.shape[0]-1, "created_date_epoch"]

    while Day <= last_Day:
        df_per_day = df.loc[(df['created_date_epoch'] >= Day) &
(df['created_date_epoch'] <= Day + epoch_1Day)]
        sublist = [epoch_to_str(Day), df_per_day["probability"].sum()]
        listed.append(sublist)
        Day += epoch_1Day


    submissions_dataframe = pd.DataFrame(listed[1:], columns=listed[0])
    return submissions_dataframe


    return df

def change_date_in_stoks(df, separator = "-"):

    dates = df.index
    str_dates = [date_obj.strftime(f'%d{separator}%m{separator}%Y') for
date_obj in dates]
    df.insert(0, "date", str_dates, True)
    return df


def fillTrueStocks(dataframe_predict, dataframe_true):

    print(type(dataframe_predict))
    print(type(dataframe_true))
    isFirst = False
    start_index = 0
    end_index = 0
    boolActual = True
    boolPast = True

    for index in dataframe_predict.index:
        if
dataframe_true["date"].str.contains(dataframe_predict["date"][index], regex =
False).any():
            boolPast = boolActual
```

```
            boolActual = True
        else:
            end_index += 1
            boolPast = boolActual
            boolActual = False
        if index == 4:
            r = 3

        start_date_time_obj =
datetime.strptime(dataframe_predict["date"][start_index], '%d-%m-%Y')
        if (end_index > -1 and boolPast == False and boolActual == True) \
                or (boolActual == False and index == len(dataframe_predict)-
1):
            end_index += 1

            if (end_index > -1 and boolPast == False and boolActual == True
and start_date_time_obj.day > 1):
                end_date_time_obj =
datetime.strptime(dataframe_predict["date"][index], '%d-%m-%Y')

                end_index -= 1

                step_Open = (dataframe_true["Open"][end_date_time_obj] -
dataframe_true["Open"][start_date_time_obj]) / end_index
                step_Low = (dataframe_true["Low"][end_date_time_obj] -
dataframe_true["Low"][start_date_time_obj]) / end_index
                step_High = (dataframe_true["High"][end_date_time_obj] -
dataframe_true["High"][start_date_time_obj]) / end_index
                step_Close = (dataframe_true["Close"][end_date_time_obj] -
dataframe_true["Close"][start_date_time_obj]) / end_index
                step_Adj_Close = (dataframe_true["Adj
Close"][end_date_time_obj] - dataframe_true["Adj
Close"][start_date_time_obj]) / end_index
                step_Volume = (dataframe_true["Volume"][end_date_time_obj] -
dataframe_true["Volume"][start_date_time_obj]) / end_index

                end_index += 1

                step_Open = round(step_Open, 2) - (step_Open /
abs(step_Open)) * 0.01
                step_Low = round(step_Low, 2) - (step_Low / abs(step_Low)) *
0.01
                step_High = round(step_High, 2) - (step_High /
abs(step_High)) * 0.01
                step_Close = round(step_Close, 2) - (step_Close /
abs(step_Close)) * 0.01
                step_Adj_Close = round(step_Adj_Close, 2) - (step_Adj_Close /
abs(step_Adj_Close)) * 0.01
                step_Volume = round(step_Volume, 2) - (step_Volume /
abs(step_Volume)) * 0.01

                Index = [start_date_time_obj + timedelta(days=1)]
                Date = [(start_date_time_obj +
timedelta(days=1)).strftime("%d-%m-%Y")]
                Open = [dataframe_true["Open"][start_date_time_obj]+
step_Open]
                Low = [dataframe_true["Low"][start_date_time_obj]+ step_Low]
                High = [dataframe_true["High"][start_date_time_obj]+
step_High]
                Close = [dataframe_true["Close"][start_date_time_obj]+
step_Close]
                Adj_Close = [dataframe_true["Adj
Close"][start_date_time_obj]+ step_Adj_Close]
```

```
                Volume = [dataframe_true["Volume"][start_date_time_obj]+
step_Volume]

                for i in range(1, end_index-1):
                    Index.append(start_date_time_obj + timedelta(days=i+1))
                    Date.append((start_date_time_obj +
timedelta(days=i+1)).strftime("%d-%m-%Y"))
                    Open.append(Open[i-1] + step_Open)
                    Low.append(Low[i-1] + step_Low)
                    High.append(High[i-1] + step_High)
                    Close.append(Close[i-1] + step_Close)
                    Adj_Close.append(Adj_Close[i-1] + step_Adj_Close)
                    Volume.append(Volume[i-1] + step_Volume)

                df2 = pd.DataFrame({
                    "Index": Index,
                    "date": Date,
                    "Open": Open,
                    "High": High,
                    "Low": Low,
                    "Close": Close,
                    "Adj Close": Adj_Close,
                    "Volume": Volume
                })

                df2.set_index("Index", inplace=True)
                dataframe_true = pd.concat([dataframe_true, df2], axis=0)

            if (end_index > -1 and boolPast == False and boolActual == True
and start_date_time_obj.day == 1):
                end_date_time_obj =
datetime.strptime(dataframe_predict["date"][index], '%d-%m-%Y')
                start_date_time_obj = end_date_time_obj
                end_date_time_obj = end_date_time_obj + timedelta(days=1)
                end_index -= 1

                step_Open = (dataframe_true["Open"][end_date_time_obj] -
dataframe_true["Open"][start_date_time_obj]) / end_index
                step_Low = (dataframe_true["Low"][end_date_time_obj] -
dataframe_true["Low"][start_date_time_obj]) / end_index
                step_High = (dataframe_true["High"][end_date_time_obj] -
dataframe_true["High"][start_date_time_obj]) / end_index
                step_Close = (dataframe_true["Close"][end_date_time_obj] -
dataframe_true["Close"][start_date_time_obj]) / end_index
                step_Adj_Close = (dataframe_true["Adj
Close"][end_date_time_obj] - dataframe_true["Adj
Close"][start_date_time_obj]) / end_index
                step_Volume = (dataframe_true["Volume"][end_date_time_obj] -
dataframe_true["Volume"][start_date_time_obj]) / end_index

                end_index += 1

                step_Open = round(step_Open, 2) - (step_Open /
abs(step_Open)) * 0.01
                step_Low = round(step_Low, 2) - (step_Low / abs(step_Low)) *
0.01
                step_High = round(step_High, 2) - (step_High /
abs(step_High)) * 0.01
                step_Close = round(step_Close, 2) - (step_Close /
abs(step_Close)) * 0.01
                step_Adj_Close = round(step_Adj_Close, 2) - (step_Adj_Close /
abs(step_Adj_Close)) * 0.01
                step_Volume = round(step_Volume, 2) - (step_Volume /
```

```
abs(step_Volume)) * 0.01

                Index = [start_date_time_obj - timedelta(days=1)]
                Date = [(start_date_time_obj -
timedelta(days=1)).strftime("%d-%m-%Y")]
                Open = [dataframe_true["Open"][start_date_time_obj]-
step_Open]
                Low = [dataframe_true["Low"][start_date_time_obj]- step_Low]
                High = [dataframe_true["High"][start_date_time_obj]-
step_High]
                Close = [dataframe_true["Close"][start_date_time_obj]-
step_Close]
                Adj_Close = [dataframe_true["Adj
Close"][start_date_time_obj]- step_Adj_Close]
                Volume = [dataframe_true["Volume"][start_date_time_obj]-
step_Volume]

                for i in range(1, end_index-1):
                    Index.append(start_date_time_obj - timedelta(days=i+1))
                    Date.append((start_date_time_obj -
timedelta(days=i+1)).strftime("%d-%m-%Y"))
                    Open.append(Open[i-1] - step_Open)
                    Low.append(Low[i-1] - step_Low)
                    High.append(High[i-1] - step_High)
                    Close.append(Close[i-1] - step_Close)
                    Adj_Close.append(Adj_Close[i-1] - step_Adj_Close)
                    Volume.append(Volume[i-1] - step_Volume)

                df2 = pd.DataFrame({
                    "Index": Index,
                    "date": Date,
                    "Open": Open,
                    "High": High,
                    "Low": Low,
                    "Close": Close,
                    "Adj Close": Adj_Close,
                    "Volume": Volume
                })

                df2.set_index("Index", inplace=True)
                dataframe_true = pd.concat([dataframe_true, df2], axis=0)

            if (boolActual == False and index == len(dataframe_predict) - 1):

                end_date_time_obj =
datetime.strptime(dataframe_predict["date"][start_index + 1], '%d-%m-%Y')
                end_date_time_obj = end_date_time_obj - timedelta(days=1)
                start_date_time_obj = start_date_time_obj - timedelta(days=1)
                step_Open = (dataframe_true["Open"][end_date_time_obj] -
dataframe_true["Open"][start_date_time_obj]) / (end_index - 1)
                step_Low = (dataframe_true["Low"][end_date_time_obj] -
dataframe_true["Low"][start_date_time_obj]) / (end_index - 1)
                step_High = (dataframe_true["High"][end_date_time_obj] -
dataframe_true["High"][start_date_time_obj]) / (end_index - 1)
                step_Close = (dataframe_true["Close"][end_date_time_obj] -
dataframe_true["Close"][start_date_time_obj]) / (end_index - 1)
                step_Adj_Close = (dataframe_true["Adj
Close"][end_date_time_obj] - dataframe_true["Adj
Close"][start_date_time_obj]) / (end_index - 1)
                step_Volume = (dataframe_true["Volume"][end_date_time_obj] -
dataframe_true["Volume"][start_date_time_obj]) / (end_index - 1)
                start_date_time_obj = start_date_time_obj + timedelta(days=1)
                end_index -= 1
```

```
                step_Open = round(step_Open, 2) - (step_Open /
abs(step_Open)) * 0.01
                step_Low = round(step_Low, 2) - (step_Low / abs(step_Low)) *
0.01
                step_High = round(step_High, 2) - (step_High /
abs(step_High)) * 0.01
                step_Close = round(step_Close, 2) - (step_Close /
abs(step_Close)) * 0.01
                step_Adj_Close = round(step_Adj_Close, 2) - (step_Adj_Close /
abs(step_Adj_Close)) * 0.01
                step_Volume = round(step_Volume, 2) - (step_Volume /
abs(step_Volume)) * 0.01

                Index = [start_date_time_obj + timedelta(days=1)]
                Date = [(start_date_time_obj +
timedelta(days=1)).strftime("%d-%m-%Y")]
                Open = [dataframe_true["Open"][start_date_time_obj]+
step_Open]
                Low = [dataframe_true["Low"][start_date_time_obj]+ step_Low]
                High = [dataframe_true["High"][start_date_time_obj]+
step_High]
                Close = [dataframe_true["Close"][start_date_time_obj]+
step_Close]
                Adj_Close = [dataframe_true["Adj
Close"][start_date_time_obj]+ step_Adj_Close]
                Volume = [dataframe_true["Volume"][start_date_time_obj]+
step_Volume]

                for i in range(1, end_index-1):
                    Index.append(start_date_time_obj + timedelta(days=i+1))
                    Date.append((start_date_time_obj +
timedelta(days=i+1)).strftime("%d-%m-%Y"))
                    Open.append(Open[i-1] + step_Open)
                    Low.append(Low[i-1] + step_Low)
                    High.append(High[i-1] + step_High)
                    Close.append(Close[i-1] + step_Close)
                    Adj_Close.append(Adj_Close[i-1] + step_Adj_Close)
                    Volume.append(Volume[i-1] + step_Volume)

                df2 = pd.DataFrame({
                    "Index": Index,
                    "date": Date,
                    "Open": Open,
                    "High": High,
                    "Low": Low,
                    "Close": Close,
                    "Adj Close": Adj_Close,
                    "Volume": Volume
                })

                df2.set_index("Index", inplace=True)
                dataframe_true = pd.concat([dataframe_true, df2], axis=0)

        if boolActual == True:
            end_index = 0
            start_index = index
        else:
            if index == len(dataframe_predict)-1:
                f = 3

    dataframe_true.sort_index(inplace=True)
```

```
    return dataframe_true


def sentiment_and_stoks(name_of_SQLite_file, concrete_step = 0,
difference_procent = 2.5, mode = ""):

    df_pos = proba(name_of_SQLite_file, concrete_step = concrete_step,
difference_procent = difference_procent, mode= mode)
    print("result of sentiment", df_pos["points"].sum())

    start_date = df_pos.loc[0, "date"]
    end_date = df_pos.loc[df_pos.shape[0]-1, "date"]
    start_date = dt.datetime.strptime(start_date, "%d-%m-%Y")
    end_date = dt.datetime.strptime(end_date, "%d-%m-%Y")
    start_date = start_date.strftime("%Y-%m-%d")
    end_date = end_date.strftime("%Y-%m-%d")

    data = yf.download('TSLA', start_date, end_date)
    data = change_date_in_stoks(data)
    data = data[1:]

    mean1 = data["Adj Close"].mean()
    mean2 = df_pos["points"].mean()
    print(f"mean1 {mean1}  mean2  {mean2}")

    df_pos["points"] = df_pos["points"] + mean1 - mean2
    df_pos = df_pos[:df_pos.shape[0]-1]

    fig, ax = plt.subplots()

    minu = 0

    if minu > 0:
        df_pos = df_pos[:-minu]

    linear_model = np.polyfit(df_pos.index.values, df_pos["points"], 15)
    linear_model_fn = np.poly1d(linear_model)
    x_s = np.arange(0, df_pos.shape[0])

    data = fillTrueStocks(df_pos, data)

    ax.plot(df_pos["date"], df_pos["points"], color="blue", alpha=0.3)
    ax.plot(data["date"], data["Adj Close"], color="red", label = "Price")


    #
    ax.plot(x_s, linear_model_fn(x_s), color="green", label="Approximation
sentiment")

    plt.xticks(np.arange(0, len(df_pos), 2), rotation=30)
    plt.xlabel("Date")
    plt.ylabel("Price")
    plt.subplots_adjust(bottom=0.18)

    print(data["Adj Close"])
    print("len data", len(data["Adj Close"]))
    print("linear_model_fn(x_s)",linear_model_fn(x_s))
    print("len linear_model_fn", len(linear_model_fn(x_s)))
    print(df_pos["points"])
    print("len df_pos", len(df_pos["date"]))

    # print(scipy.stats.pearsonr(data["Adj Close"], linear_model_fn(x_s)))  #
```

```
Pearson's r
    # print(scipy.stats.spearmanr(data["Adj Close"], linear_model_fn(x_s)))
# Spearman's rho
    # print(scipy.stats.kendalltau(data["Adj Close"], linear_model_fn(x_s)))
# Kendall's tau
    # print("spatial", 1 - spatial.distance.cosine(data["Adj Close"],
linear_model_fn(x_s)))
    # print("sim", similaritymeasures.frechet_dist(data["Adj Close"],
linear_model_fn(x_s)))

    ax.legend()
    plt.show()

if __name__ == '__main__':

    pd.set_option('display.max_rows', None, 'display.max_columns', None,
'display.width', None, "display.max_colwidth", 1)


    start_epoch = int(dt.datetime(2019, 1, 1).timestamp())
    end_epoch = int(dt.datetime(2019, 12, 30).timestamp())
    subreddit = 'teslainvestorsclub'
    q = "Tesla"
    limit = 10000000

    name_of_SQLite_file = "comments_database"
    name_of_SQLite_file2 = "comments_database2"
    name_of_SQLite_file3 = "comments_database3"
    name_of_SQLite_file4 = "comments_database4"
    name_of_SQLite_file5 = "comments_database5"
    name_of_SQLite_file6 = "comments_database6"
    name_of_SQLite_file7 = "comments_database7"
    name_of_SQLite_file8 = "comments_database8"
    name_of_SQLite_file9 = "comments_database9"
    name_of_SQLite_file10 = "comments_database10"
    name_of_SQLite_file11 = "comments_database11"
    name_of_SQLite_file12 = "comments_database12"
    name_of_SQLite_file13 = "comments_database13"
    name_of_SQLite_file14 = "comments_database14"
    name_of_SQLite_file15 = "comments_database15"

    print("start")
    sentiment_and_stoks(name_of_SQLite_file15, concrete_step=0.55)
```

АНТИПЛАГИАТ
ОБНАРУЖЕНИЕ ЗАИМСТВОВАНИЙ

Отчет о проверке на заимствования №1

Автор: Бикбавлеев Александр
Проверяющий:

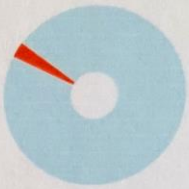
Отчет предоставлен сервисом «Антиплагиат» - http://users.antiplagiat.ru

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 3
Начало загрузки: 04.06.2022 23:00:05
Длительность загрузки: 00:00:01
Имя исходного файла:
Диплом_Черновик_Бикбавлеев_Александр_931822_2_1.pdf
Название документа:
Диплом_Черновик_Бикбавлеев_Александр_931822_2_1
Размер текста: 59 кБ
Символов в тексте: 60606
Слов в тексте: 7594
Число предложений: 391

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Начало проверки: 04.06.2022 23:00:07
Длительность проверки: 00:00:05
Комментарии: не указано
Модули поиска: Интернет Free



ЗАИМСТВОВАНИЯ	САМОЦИТИРОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
2,88%	0%	0%	97,12%

Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.
Самоцитирования — доля фрагментов текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника, автором или соавтором которого является автор проверяемого документа, по отношению к общему объему документа.
Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.
Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.
Заимствования, самоцитирования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.
Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Источник	Актуален на	Модуль поиска
[01]	1,22%	Полный текст (2/2) https://istina.msu.ru	09 Мая 2018	Интернет Free
[02]	0,46%	Полный текст (1/2) https://istina.msu.ru	09 Мая 2018	Интернет Free
[03]	0,75%	Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop (1/4) https://aclweb.org	10 Июл 2020	Интернет Free

Еще источников: 6
Еще заимствований: 0,44%

С результатом ознакомлен М.Н. Головчинер.

К.И. Лившиц