Отчет о разработке приложения для определения тональности текста

Введение

Цель данного проекта состояла в разработке веб-приложения, способного определять тональность текста. Приложение было построено с использованием Django и включало в себя машинное обучение для классификации текста на позитивную и негативную тональность. В этом отчете я предоставлю подробное описание процесса разработки и использованных технологий.

Шаги разработки

1. Исследование и подготовка данных

Первым шагом было проведение исследования и подготовка данных. Я использовал датасет IMDB Large Movie Review, который содержал отзывы о фильмах, размеченные на позитивные и негативные. Датасет был загружен и предобработан, включая удаление лишних символов, токенизацию и приведение текста к нижнему регистру.

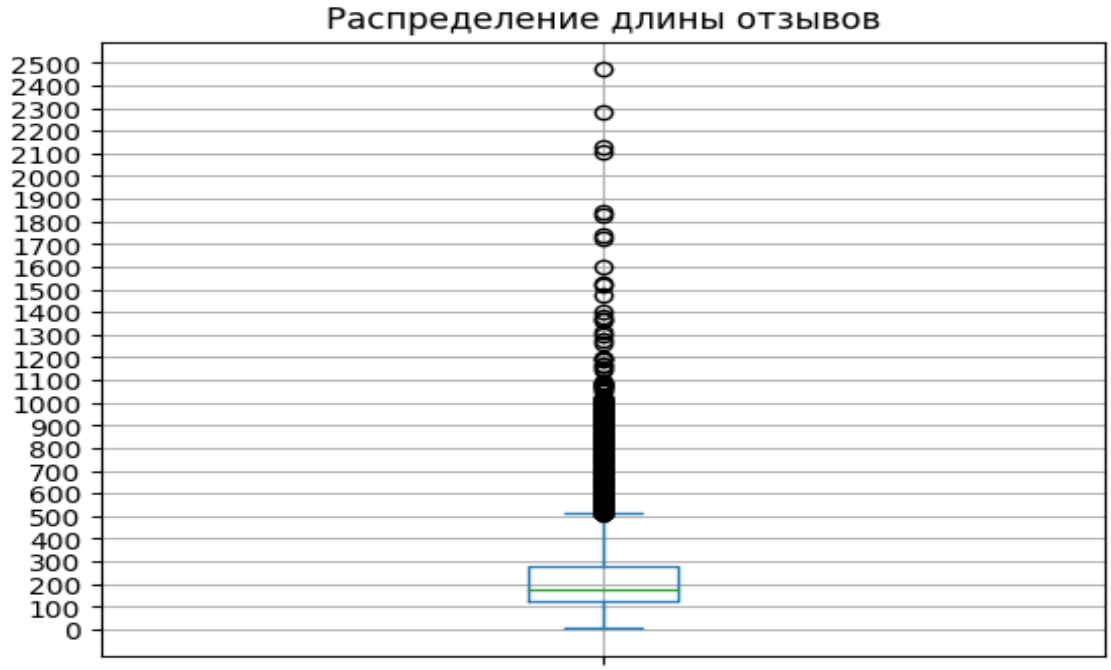
Использованные библиотеки:

tarfile , pandas, nltk, string, pymorphy2, collections

2. Анализ данных

В результате анализа данных текста были получены следующие результаты:

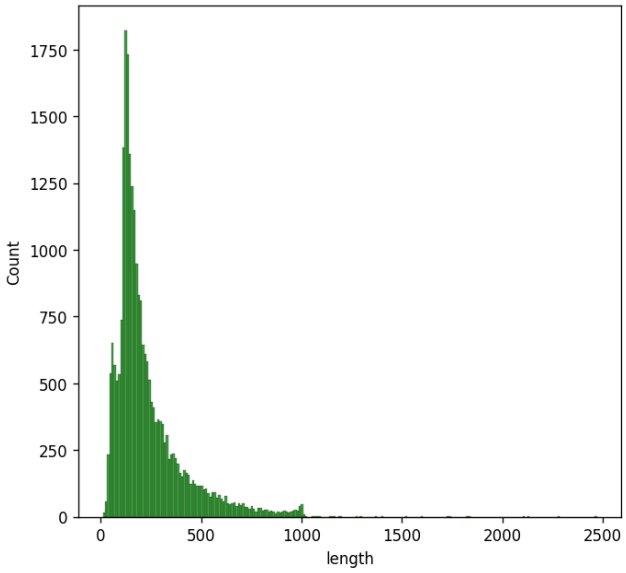
- основном отзывы содержат от 100 до 300 слов



- длина положительных и отрицательных отзывов существенно не отличается

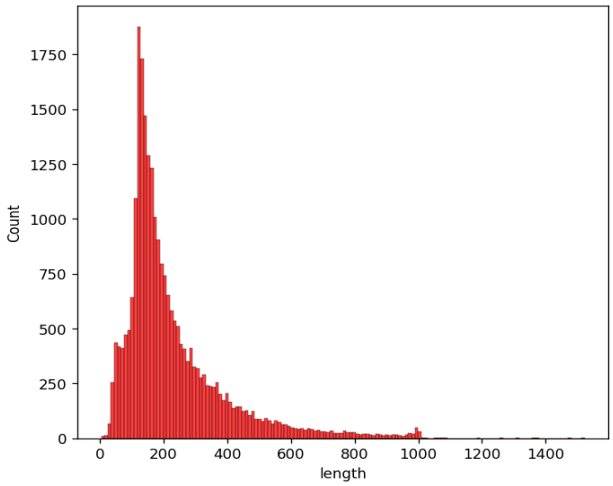
Распределение длины текста положительных отзывов

	length
count	25000.0
mean	232.85
std	177.5
min	10.0
25%	125.0
50%	172.0
75%	284.0
max	2470.0



Распределение длины текста отрицательных отзывов

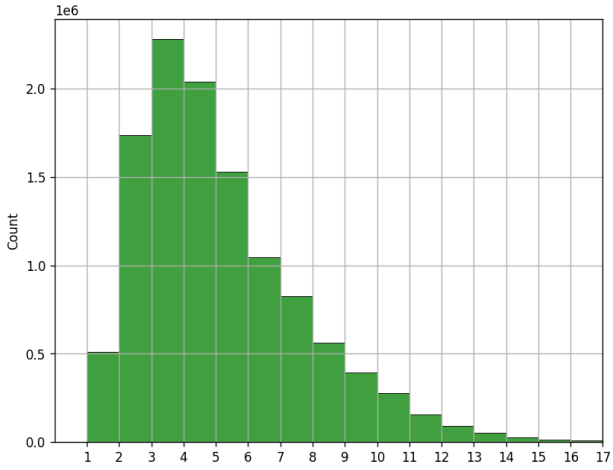
	length
count	25000.0
mean	229.46
std	164.95
min	4.0
25%	128.0
50%	174.0
75%	278.0
max	1522.0



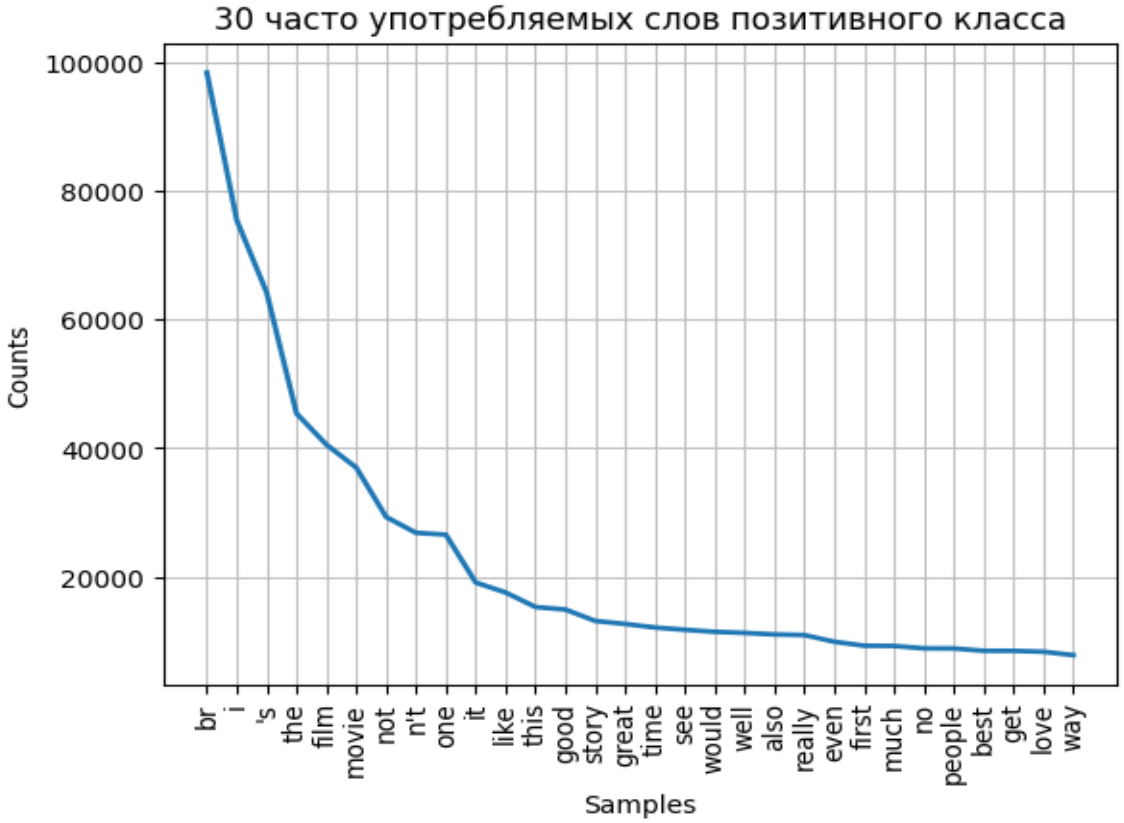
- Датасет содержал много слов состоящих из одной буквы, в последующем они были удалены, кроме слова “i”. Наиболее часто встречаются слова состоящие от 2 до 5 букв.

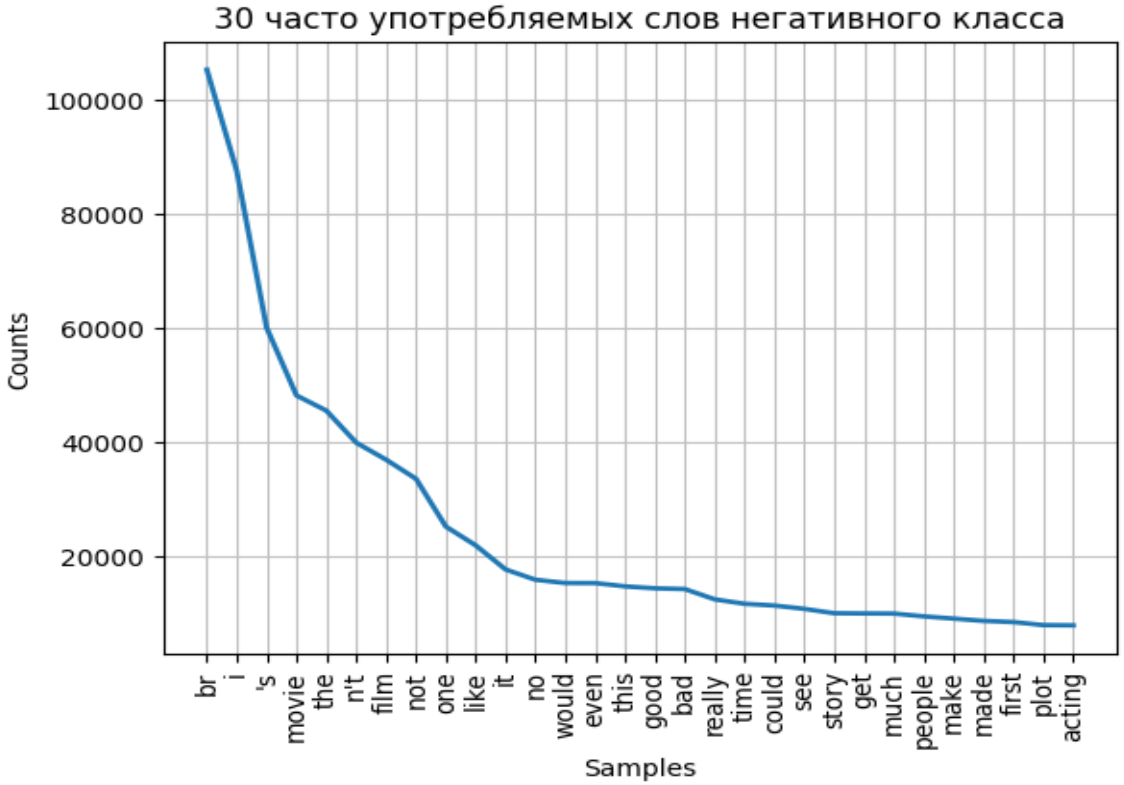
Распределение длины слов до очистки текста

	0
count	11557847.0
mean	4.67
std	2.6
min	1.0
25%	3.0
50%	4.0
75%	6.0
max	78.0



- Были выявлены самые частотные слова позитивного и негативного класса





3. Подготовка данных для обучения

Текст был преобразован в числовой формат при помощи TfidfVectorizer . TfidfVectorizer является классом в библиотеке sklearn (scikit-learn). Он используется для преобразования текстовых данных в векторное представление с использованием метода TF-IDF (Term Frequency-Inverse Document Frequency). TF-IDF - это статистическая мера, используемая для оценки важности термина в контексте коллекции документов. Категориальные переменные меток класса были заменены на 0, 1 – негативный и позитивный класс соответственно.

4. Обучение модели

Были обучены три модели: LogisticRegression , Random Forest Classifier(с оптимизацией гиперпараметров), нейросеть LSTM.

По результатам проверки качества модели на тестовой выборке, были получены следующие результаты:

LogisticRegression	f1_score: 0.92
Random Forest Classifier	f1_score: 0.97
Random Forest Classifier	f1_score: 0.83
(с оптимизацией гиперпараметров)	
нейросеть LSTM	accuracy: 0.43

В результате проверки на случайном тексте, точные результаты были у моделей LogisticRegression и Random Forest Classifier(с оптимизацией гиперпараметров). Так как у LogisticRegression более высокий скор и она более легкая, что позволяет ее легче развернуть в облаке, для использования в веб-приложении была выбрана именно она.

5. Создание веб-приложения с использованием Django

Затем я приступил к разработке веб-приложения с использованием Django. Веб-приложение создавал в IDE PyCharm, так оно больше подходит для этой задачи. Я создал Django-проект и определил модель для хранения текстовых данных. Затем я создал представления (views) для обработки запросов от пользователей и реализовал логику классификации текста с использованием обученной модели. Также были созданы HTML-шаблоны для отображения пользовательского интерфейса.

6. Тестирование и отладка

После завершения разработки я провел тестирование приложения. Я проверил его работу на разных текстовых примерах и убедился, что предсказания соответствуют ожидаемым результатам. Если возникали ошибки или проблемы, я осуществлял отладку и исправлял их.

7. Разворачивание приложения

Последним шагом было развертывание приложения для общего доступа. Я выбрал платформу для развертывания приложения PythonAnywhere.

Развертывание приложения на PythonAnywhere:

- Создание аккаунта на PythonAnywhere и настройка виртуальной среды.
- Загрузка проекта и настройка веб-приложения на PythonAnywhere.
- Проверка работоспособности приложения через доступную ссылку.

В результате приложение было доступно всем пользователям по ссылке.

Итоги:

В результате разработки приложения для определения тональности текста на позитивную и негативную, были достигнуты следующие результаты:

- Была создана модель машинного обучения, способная классифицировать тексты на основе их тональности.

- Было разработано веб-приложение с использованием Django, позволяющее пользователям вводить тексты и получать предсказания от модели.

- Приложение было развернуто в облаке и стало доступно всем пользователям по ссылке.

Это приложение может быть использовано для различных задач, связанных с анализом тональности текста, и может быть легко настроено и масштабировано для дополнительных требований и функциональности.

Используемые технологии

- **Python** - основной язык программирования
- **Django** - фреймворк для веб-разработки
- **scikit-learn** - библиотека для машинного обучения
- **PythonAnywhere** - платформа для развертывания приложений в облаке
- **GoogleColab** – среда разработки
- **PyCharm** – среда разработки

-