**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра информационных систем**

**ОТЧЕТ**

**по Учебной практике**

**Тема: Изучение и сравнение методов оценки тональности твитов**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 2372 |  | Братко В.В. |
|  |  | Полуянов В.Н. |
|  |  | Сарайкин А.А. |
| Преподаватель |  | Васильев Н.В. |

Санкт-Петербург

2024

**ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ**

**Студенты** Братко В.В., Полуянов В.Н., Сарайкин А.А.

**Группа** 2372

**Тема работы:** Изучение и сравнение методов оценки тональности твитов

**Задание на практику:**

Построить модель, которая по тексту поста в твиттере предсказывает его эмоциональную окраску. Для этого необходимо:

* Изучить основные методы представления и векторизации текстов;
* Загрузить предложенный датасет в библиотеку pandas;
* Произвести предобработку и лемматизацию загруженных в датафрейм твитов, удалить стоп-слова;
* Произвести разбиение на обучающую и тестовую выборки;
* Векторизовать твиты при помощи объекта sklearn.CountVectorizer.

Необходимо произвести векторизацию униграммами, биграммами и триграммами. Провести эксперимент для униграмм, биграмм, триграмм и TFIDFVectorizer. Обучить логистическую регрессионную модель на обучающей выборке, проверить качество классификации и сделать выбор наилучшего метода векторизации.

**Сроки прохождения практики:** 04.03.2024 - 08.06.2024

**Дата сдачи реферата:** 30.04.2024

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Братко В.В. |
| Студент гр. 2372 |  | Полуянов В.Н. |
| Студент гр. 2372 |  | Сарайкин А.А. |
| Преподаватель |  | Васильев Н.В. |

**АННОТАЦИЯ**

Содержание учебной практики заключается в исследовании в области анализа тональности текста. Основные этапы включают в себя предобработку предложенного набора данных, разделение на обучающую и тестовые выборки, векторизацию текста и обучение модели логистической регрессии. Эксперименты проводились с использованием униграмм, биграмм и триграмм, а также с TF-IDF векторизацией. Оценка качества моделей производилась сравнением по метрике F1-score для каждого метода.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_gjdgxs)

[ПРЕДОБРАБОТКА ДАННЫХ 6](#_au9i5xfq9xi6)

[ВЕКТОРИЗАЦИЯ 7](#_bs8mdl5oqq3j)

[Bag Of Words 7](#_hsjiajkjkujx)

[N-Grams 7](#_c4cpvh3vh4qi)

[TF-IDF 8](#_l9w6bqb7s2go)

[ОБУЧЕНИЕ 10](#_amjnqw4s0bbh)

[Логистическая регрессия 10](#_18coe25zb4m5)

[Результаты обучения 10](#_6jcjb0z5t2m4)

[Классификация с помощью предобученной языковой модели rubert-tiny2 13](#_g3m0w3ca3krc)

[Предобработка и токенизация текста 13](#_3e42fbkf81kg)

[Rubert-tiny2 15](#_tnp94g8k87ce)

[Обучение модели (Transfer learning): 16](#_yppmdhdfk2ru)

[ЗАКЛЮЧЕНИЕ 19](#_n4nl165gzb9p)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 20](#_3whwml4)

# ВВЕДЕНИЕ

В настоящее время огромные объемы текстовой информации генерируются повседневно и на механический анализ этих данных может уходить много времени и средств. В этой связи существует необходимость разработки методов и инструментов анализа естественного языка и, в частности, анализа тональности текста. Такой инструмент позволяет программно предсказать эмоциональную окраску текста, выраженную через негативные и положительные оценки.

Целью данной работы является изучение и сравнение эффективности существующих методов анализа тональности твитов, а также разработка соответствующей поставленным целям программы.

Актуальность обусловлена повсеместным ростом спроса в сфере анализа социальных медиа, пользовательских отзыва, новостных статей и прочей текстовой информации с целью выявления мнений, трендов и настроений. Результаты такого исследования могут быть полезны во многих областях.

|  |  |
| --- | --- |
| ФИО | Описание |
| Братко В.В. | Написан блокнот с классификацией твитов с помощью модели трансформера |
| Сарайкин А.А. | Написан блокнот с Bag of Words и TF-IDF классификацией |
| Полуянов В.Н. | Предобработка текста, оформление результатов экспериментов |

# ПРЕДОБРАБОТКА ДАННЫХ

В процессе машинного обучения предобработка данных – критически важный этап, особенно при работе с текстовыми данными. На этом этапе наша цель состояла в том, чтобы очистить и преобразовать исходные текстовые данные в формат, который может быть эффективно использован для обучения модели.

Для начала из текста удалены все лишние символы, ссылки и лишние пробелы с помощью библиотеки re и, соответственно, следующих регулярных выражений:

1. [^а-яА-Я] – все символы, отличные от заданного алфавита;
2. \s+ – все пробелы (заменяем 1 и более пробелов на один);
3. http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|[!\*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+ – все гиперссылки.

Затем были удалены стоп-слова с помощью соответствующего массива, взятого из библиотеки nltk.

После очистки данных проводится лемматизация - процесс приведения слов к их начальной форме (лемме). Лемматизация проводилась с помощью лингвистический процессор Mystem от Яндекса, доступ к которому был получен через библиотеку pymystem3.

Последним шагом предобработки является разделение данных на токены - составляющие его единицы данных. С учетом задачи мы остановились на токенизации по пробелам.

# ВЕКТОРИЗАЦИЯ

Теперь, когда наш текст предварительно обработан, его необходимо привести в приемлемую для алгоритмов обучения векторную форму. Для этого были применены несколько различных методов.

## Bag Of Words

Bag Of Words – это метод векторизации, основанный на one-hot encoding - коде, в котором каждое слово кодируется вектором, длина которого равна мощности словаря, то есть количеству всех уникальных слов в документе.

Если слово находится на i-ом месте в словаре, то его вектор будет выглядеть как (01, …, 1i, …, 0n), где единица стоит на месте этого слова (рис. 1).

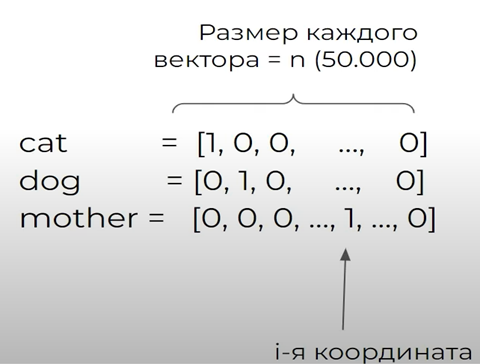


Рисунок 1. Пример векторов для слов в BoW

В этом методе суммируются векторы закодированных слов, то есть текст “помогите мне” будет закодирован как вектор для слова “помогите” (01, …, 1i, …, 0n) + вектор для слова “мне” (01, …, 1j, …, 0n) = (01, …, 1i, …, 1j, …, 0n).

Такой подход не учитывает последовательность слов. Нельзя утверждать и то, что векторы слов независимы, так как считается частота вхождения и, например, слово кошка будет встречаться чаще со словом котята, чем со словом перцептрон, поэтому зависимость слов присутствует.

Чтобы учитывать некоторую последовательность необходимо использовать N-граммы.

## N-Grams

N-граммы – это метод представления текста в виде последовательностей из N токенов. Этот метод используется для учета порядков слов, здесь берутся N подряд идущих токенов скользящим окном. Таким образом, каждое слово представлено в сочетании с его N-соседними словами. Это может быть полезно для моделей, которые учитывают контекст слова.

Существенным недостатком N-грамм безусловно является полиномиально возрастающая потребность в памяти, поэтому важно тщательно очистить исходные данные от информации, не несущей в себе пользы.

## TF-IDF

Term Frequency - Inverse Document Frequency (TF-IDF) – это метод, учитывающий как частоту появления слова в данном тексте, так и его значимость во всей коллекции документов. TF-IDF для токена ***t*** для документа ***d*** в наборе документов *D*можно вычислить по формуле:

– частота появления слова в документе, или важность слова для понимания документа слово относительно остальных, можно рассчитать по формуле:

где в числителе находится число слов ***t***, встречающихся в документе ***d***, а в знаменателе – общее число слов в этом документе.

– обратная частота появления слова в коллекции документов, то есть чем чаще слово встречается во всём наборе документов, тем меньше будет этот показатель. Так, выделяются токены, встретившиеся в относительно меньшем числе документов и тем самым являющиеся значимыми (слово нетривиально и может иметь большое значение для контекста документа). Можно рассчитать по формуле:

где в числителе находится количество всех документов в наборе, а в знаменателе – число документов, в которых встречается слово.

К преимуществам такого подхода следует отнести то, что полученные векторы имеют больший смысл, а также возможность решать задачи, например, ранжирования документов и выделения ключевых слов.

Недостатками же являются: сильная разреженность векторов, фиксированность размера словаря, перерасчет векторов при изменении коллекции документов.

# ОБУЧЕНИЕ

Исходные данные были подготовлены и могут быть использованы для обучения модели для последующего предсказания или классификации новых данных на основе полученных знаний.

## Логистическая регрессия

Логистическая регрессия или логит-модель оценивает вероятность принадлежности экземпляра к данному классу, на основе заданного набора данных независимых величин (рис. 2). Зависимая переменная ограничена диапазоном от 0 до 1.

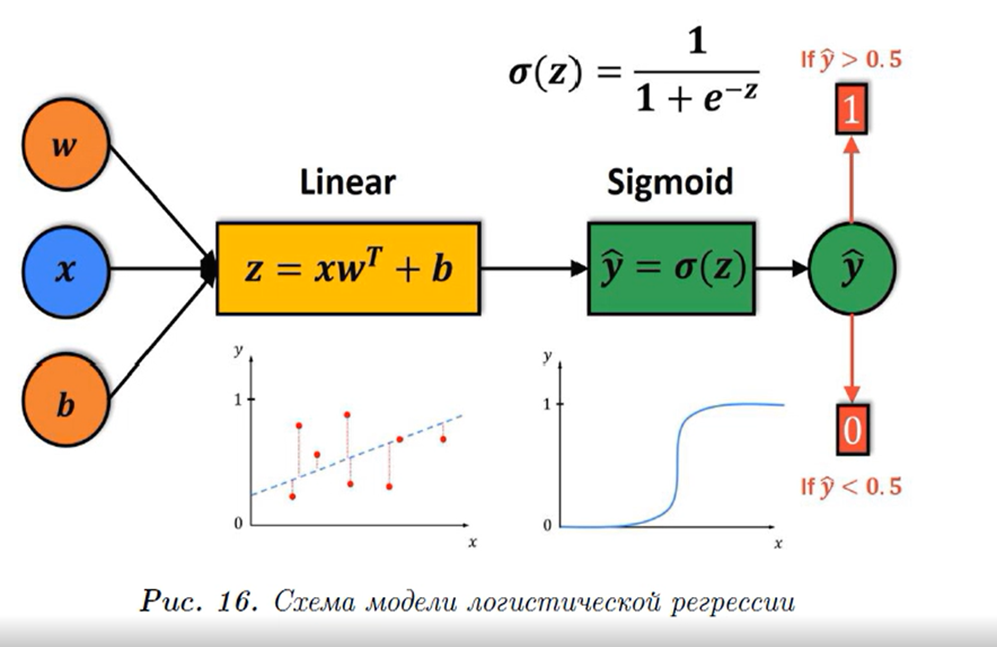


Рисунок 2. Схема логит-модели

## Результаты обучения

Были проведены эксперименты для всех вышеописанных методов векторизации с некоторыми изменениями в методах предобработки текста.

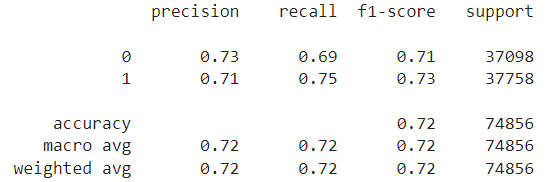
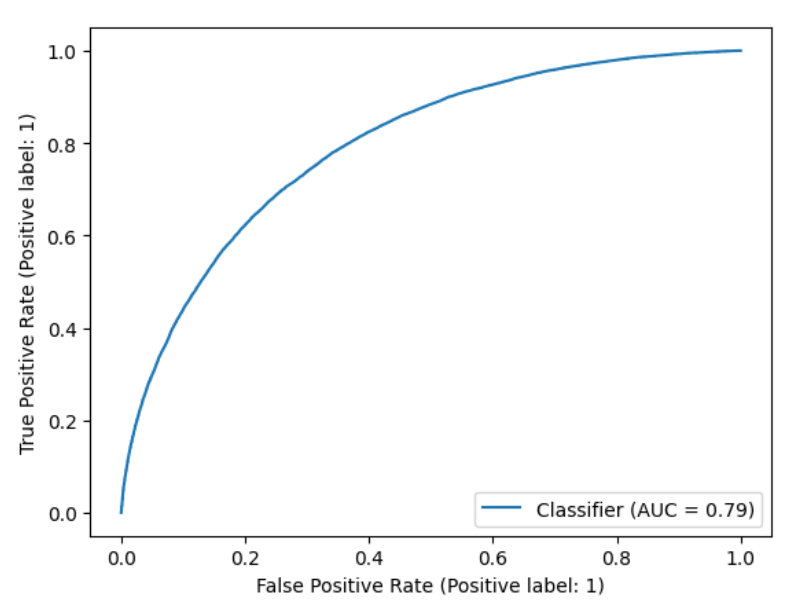
Первые опыты проводились с TF-IDF векторизацией. В первом эксперименте мы провели лемматизацию текста, во втором – нет (рис. 3-4).

Рисунок 3. Результаты с TF-IDF с лемматизацией

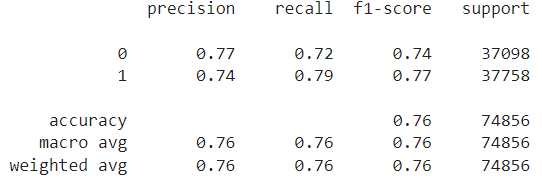
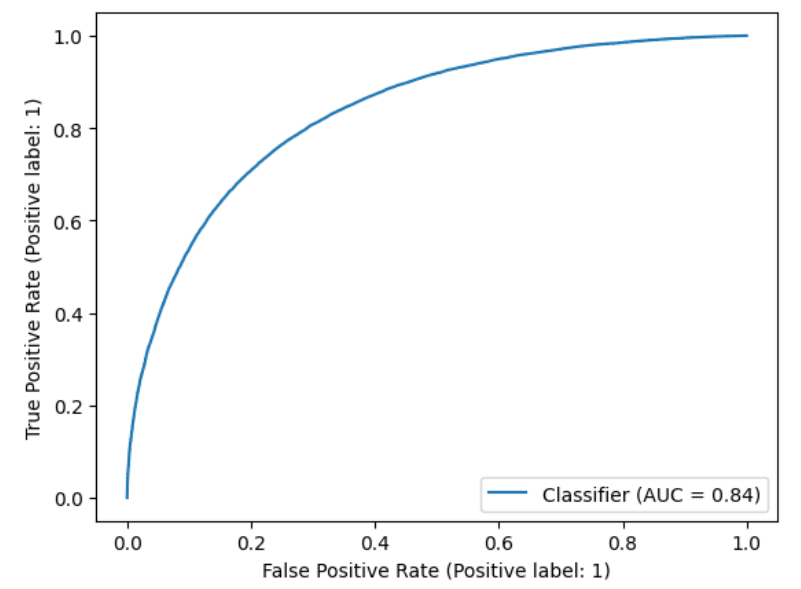


Рисунок 4. Результаты с TF-IDF без лемматизации

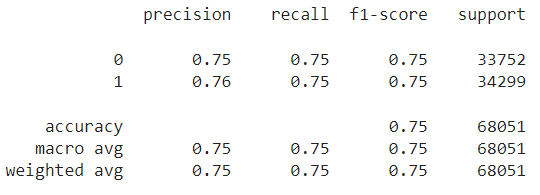
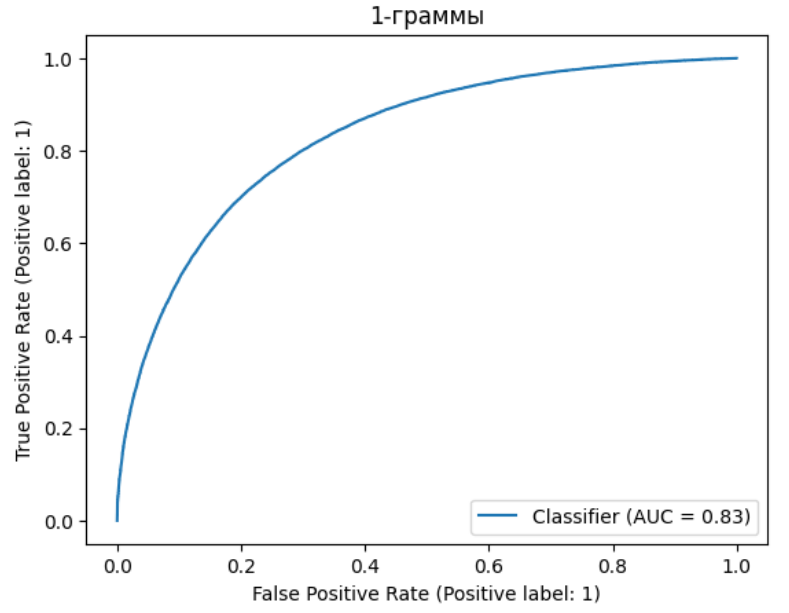


Рисунок 5. Результаты с униграммами, BoW

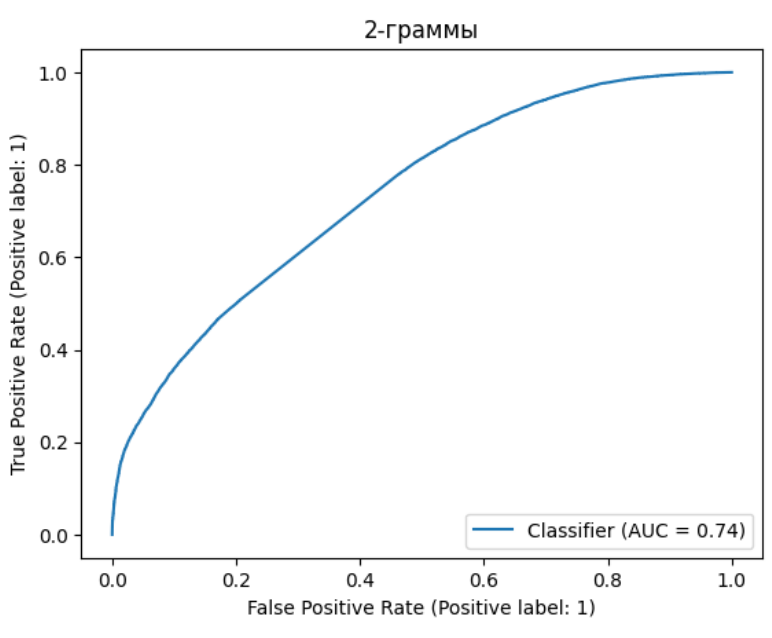
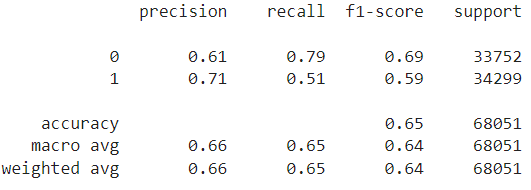
****

Рисунок 6. Результаты с биграммами, BoW

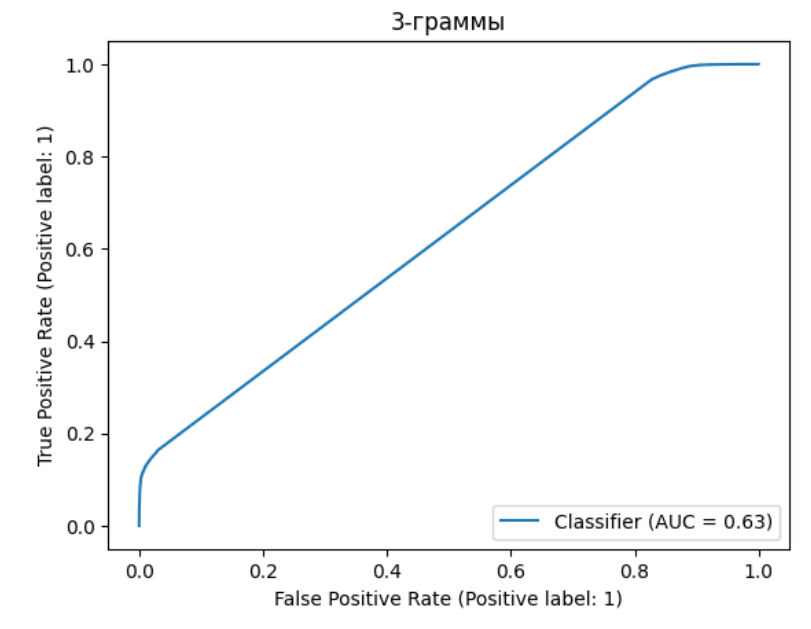
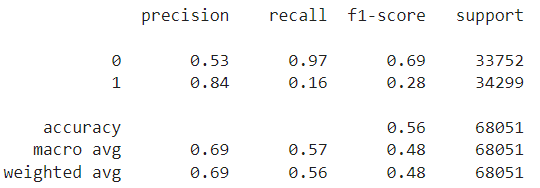
****

Рисунок 7. Результаты с триграммами, BoW

Как видно из результатов, линейная модель справилась с классификацией, показав результат f1-score 75%. Использование N-грамм ухудшило результат модели – она стала классифицировать большинство примеров как положительные. Первые два эксперимента показали, что лемматизация негативно влияет на результаты обучения и дали неплохие показатели по точности, воспроизводимости и F1-мере, что подтверждает эффективность использования TF-IDF в качестве метода векторизации при предобработке данных.

# Классификация с помощью предобученной модели rubert-tiny2

## Предобработка и токенизация текста

В качестве предобработки текста были удалены ссылки, которые не несут в себе никакую ценную информацию. Другие методы предобработки и лемматизации текста не были проведены, т.к. проведя эксперимент, было выяснено, что такой метод снижает качество работы модели, т.к. при лемматизации в случае трансформера теряет часть полезной информации.

Токенизация была проведена классическим для моделей семейства BERT алгоритмом WordPiece tokenization:

1. Сначала добавляются технические токены [CLS] в начале каждого твита, который собирает в себе attention со всей последовательности, [SEP] токен в конце каждой последовательности, а также [PAD] токены, которые добавляются при необходимости в последовательность для формирования тензоров одинаковой длины;
2. Затем проводится токенизация слов с помощью WordPiece tokenization алгоритма.

Это алгоритм токенизации, который токенизирует текст для предварительного обучения BERT моделей. Его отличие в том, что он не нормализует токены, что приводит к невозможности потом генерировать эти токены в корректный текст, а разбивает каждое слово на определенные части.

Сначала токенизатор разбивает все слова из текстов посимвольно и только первым буквам слова не приписывает ## перед токеном. Символ ## нужен для задач генерации текста, поэтому при инференсе модели будут сливаться токены с символом ## в начале пока не встретится токен без символа ## в начале, что будет означать, что началось уже следующее слово.

Добавляем в словарь все символы из текста, а затем начинаем сливать токены с наибольшей оценкой и добавлять их в наш словарь. Оценка находится по следующей формуле:

Поделив частоту пары на произведение частот каждой из ее частей, алгоритм отдает предпочтение слиянию пар, отдельные части которых встречаются в словаре реже. Например, он не обязательно объединит ("un", "##able"), даже если эта пара встречается в словаре очень часто, потому что две пары "un" и "##able", скорее всего, встречаются в большом количестве других слов и имеют высокую частоту.

Обучив словарь, проходимся по тексту, токенизируем его по словам и для каждого токена токенизируем следующим образом: сначала находим самый длинный по длине токен, которым можно затокенизировать данный, потом для оставшейся части слова опять ищем самый длинный и т.д. Если же в какой-то момент токенизации слова не удается подобрать токен для его части, то все это слово заменяется на токен <UNK>.

1. Последовательности токенов переводятся в числа и затем формируются многомерные тензоры, которые затем формируются в батчи с помощью dataloader.

Пример токенизации из текста:

Тренировочный текст: @i\_yakovleva Да она уже даже немножко надоесть успела, а ты все еще мылишься!))

Токенизированный тренировочный текст: ['[CLS]', '@', 'i', '\_', 'ya', '##kov', '##lev', '##a', 'Да', 'она', 'уже', 'даже', 'немножко', 'надое', '##сть', 'успела', ',', 'а', 'ты', 'все', 'еще', 'мы', '##ли', '##шься', '!', ')', ')', '[SEP]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]']

Также при токенизации создается маска внимания, которая обнуляет все [PAD] токены для того, чтобы исключить их влияние на обучение (рис. 7)

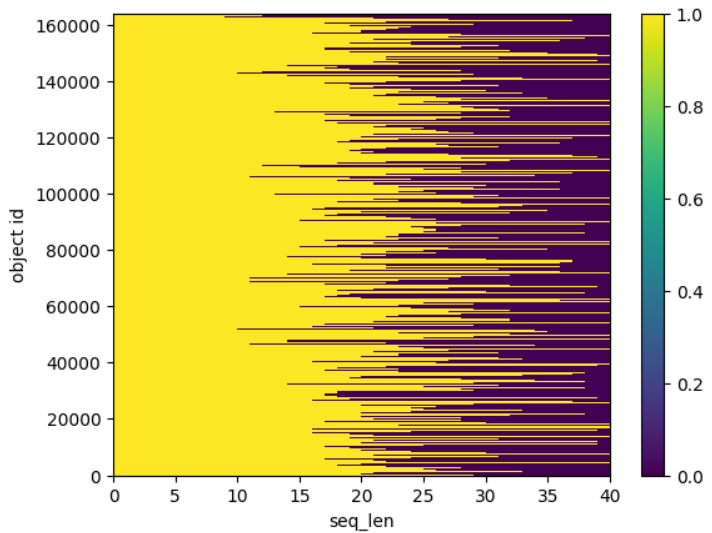


Рисунок 7. Маска внимания

## Rubert-tiny2

Rubert-tiny2 является BERT энкодером с высококачественными эмбеддингами, обученными на русском языке.

Энкодер трансформера представляет из себя следующую модель (рис. 8):

1. На вход модели подаются последовательности эмбеддингов;
2. Позиционно кодируем входные эмбеддинги (путем сложением);
3. Применяется многоголовый self-attention (т.е. несколько self-attention с разными матрицами весов);
4. Складываются получившиеся эмбеддинги с изначальными и нормализуются;
5. Каждый эмбеддинг прогоняется через простую двухслойную полносвязную нейронную сеть с одинаковыми для каждого эмбеддинга весами;
6. Еще раз складываем с эмбеддингами на предыдущем шаге и нормализуем;
7. Повторяем 3 раза в случае модели rubert-tiny2.

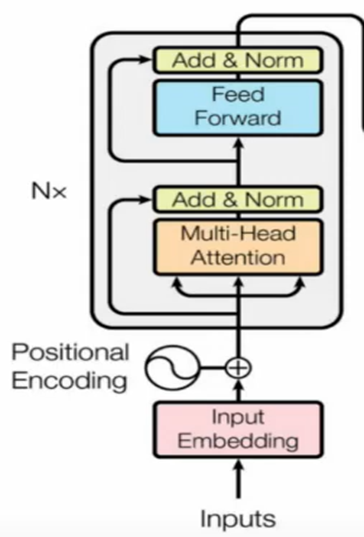


Рисунок 8. rubert-tiny2

## Обучение модели (Transfer learning):

Для решения поставленной задачи был применен метод машинного обучения transfer learning: в качестве backbone модели был использован трансформер rubert-tiny2, который был обучен на похожем домене, и проведен fine-tuning данной модель путем добавления к выходу технического токена [CLS] двухслойной полносвязной нейронной сети с регуляризацией.

В качестве оптимизатора был выбран AdamW с learning rate = 1\*10-5, функции потерь - бинарная кросс-энтропия, размер батча - 128, количество эпох - 10. Был проведен стандартный цикл обучения для нейронной сети с обновлением весом алгоритмом backpropagation.

Метрики во время обучения представлены на рисунках 9 и 10.

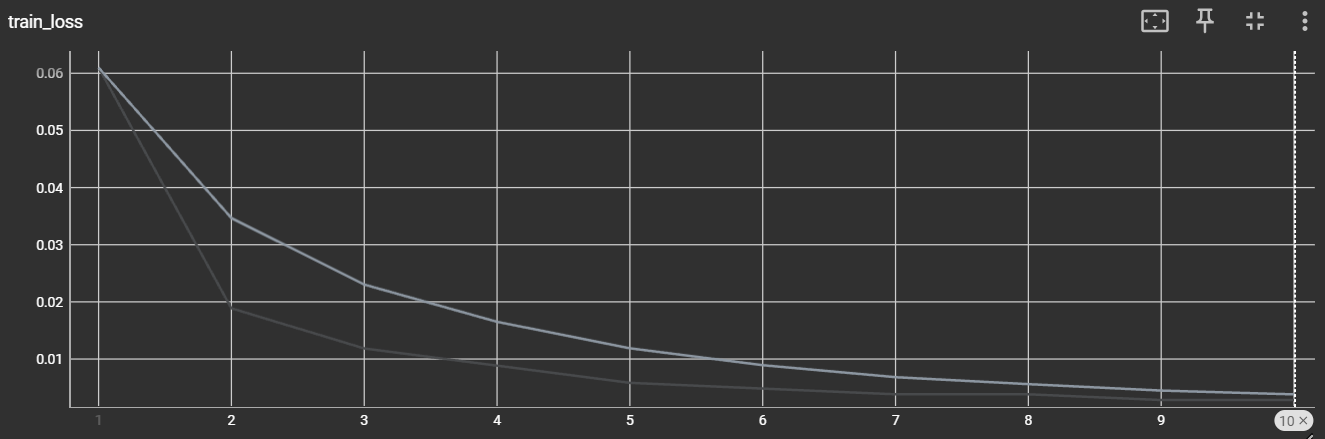


Рисунок 9. Бинарная кросс-энтропия на тренировочных данных

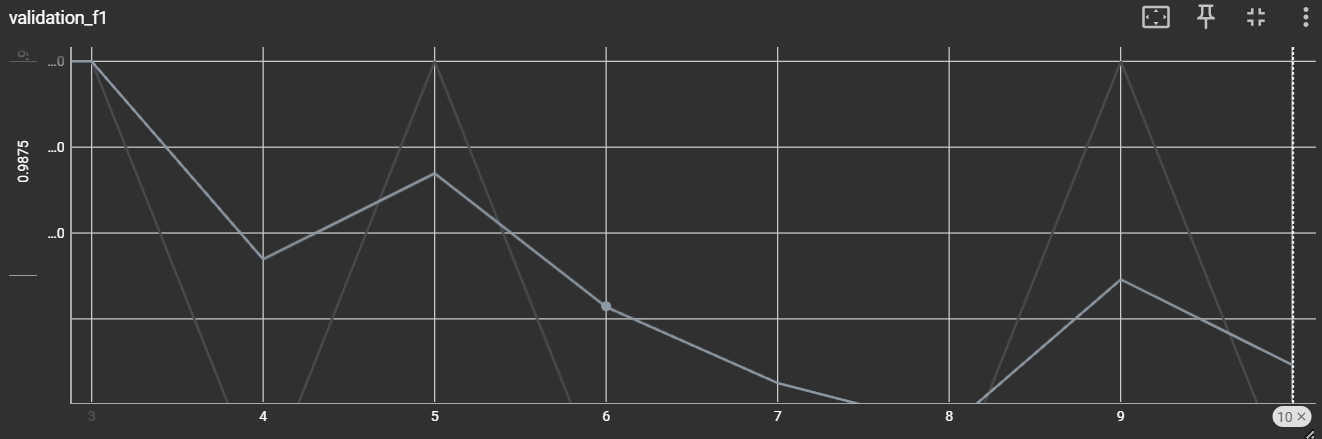


Рисунок 10. f1 метрика на валидационных данных

Как видно из графиков модель быстро обучалась на данных и стала переучиваться, поэтому дальнейшее обучение не имеет смысла. Для оценки на тестовых данных была сохранена модель с наивысшим показателем f1 на валидационных данных. Результаты классификации представлены ниже на рисунках 11 и 12.

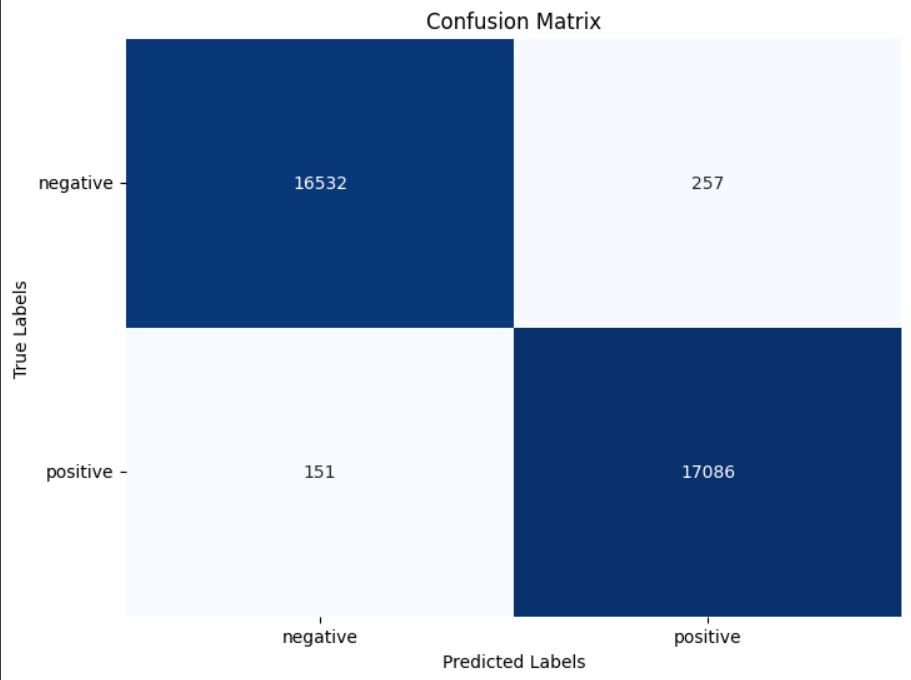


Рисунок 11. Матрица ошибок

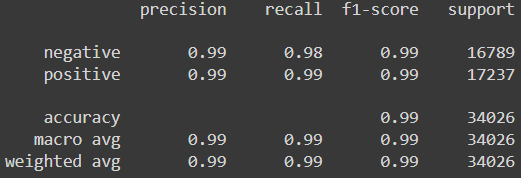


Рисунок 12. Результаты классификации

Как хорошо видно из результатов классификации, предобученная модель с дополнительными слоями полностью описала область распределения данных если не считать гауссовский шум, который присутствует почти во всех данных.

Сравнив все методы векторизации и обучения моделей, можно прийти к выводу, что векторизация слов с помощью предобученных эмбеддингов безусловно является наилучшим выбором для решения поставленной задачи, т.к. этот метод учитывает позиционное положения слова в тексте, а трансформер с блоком self-attention лучше справляется с поиском зависимостей слов в предложении.

# ЗАКЛЮЧЕНИЕ

В данной работе была изучены задача анализа тональности текстовых данных, включающая различные этапы обработки данных, такие как очистка, лемматизация и векторизация. Были поставлены эксперименты с использованием различных методов векторизации, таких как Bag Of Words, N-граммы и TF-IDF, получены сравнительные оценки. Результаты экспериментов показали, что TF-IDF без лемматизации дал лучшие результаты по F1-мере.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Шолле Франсуа, Глубокое обучение на Python. 2-е межд. издание. — СПб.: Питер, 2023. — 576 с.: ил. — (Серия «Библиотека программиста»).
2. Учебник по машинному обучению от Яндекса, <https://education.yandex.ru/handbook/ml>
3. Траск Эндрю, Грокаем глубокое обучение. — СПб.: Питер, 2019. — 352 с.: ил. — (Серия «Библиотека программиста»).
4. Машинное обучение (курс лекций, К.В.Воронцов), [ссылка](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%28%D0%BA%D1%83%D1%80%D1%81_%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D0%B9%2C_%D0%9A.%D0%92.%D0%92%D0%BE%D1%80%D0%BE%D0%BD%D1%86%D0%BE%D0%B2%29)
5. Васильев Юлий, Python для data science. — СПб.: Питер, 2023. — 272 с.: ил. — (Серия «Библиотека программиста»).