Проект: «Анализ эмоциональной тональности в социальных медиа»

Введение

В современном информационном обществе социальные медиа играют ключевую роль в формировании общественного мнения, воздействуя на множество аспектов нашей жизни, включая политику, экономику, культуру и социальные взаимодействия. Одним из наиболее значимых и актуальных аспектов анализа в социальных медиа является оценка эмоциональной тональности сообщений. Анализ эмоциональной тональности позволяет понять настроение, отношение и реакции пользователей на определенные события, продукты или услуги.

В данном исследовании представляется обзор современных методов и подходов к анализу эмоциональной тональности в социальных медиа.

Цель проекта

Цель данного проекта заключается в разработке и реализации метода анализа эмоциональной тональности сообщений в социальных медиа с целью выявления общественного настроения по различным событиям, темам или брендам. Путем анализа и интерпретации пользовательских сообщений и комментариев в социальных медиа планируется:

1. **Мониторирование общественного мнения**: Проект направлен на создание инструмента, способного систематически отслеживать и анализировать тональность выражений и высказываний пользователей в социальных медиа. Это позволит оценивать общественное настроение и выявлять ключевые темы или проблемы, которые активно обсуждаются в сети.
2. **Определение трендов**: Анализ эмоциональной тональности сообщений позволит выявить тенденции и тренды, которые формируются в социальных медиа. Это может быть полезным для прогнозирования развития событий, а также для адаптации стратегий в социальных медиа для более эффективного взаимодействия с аудиторией.
3. **Оценка реакции на события**: Проект позволит оценить эмоциональную реакцию пользователей на различные события, такие как политические события, мировые новости, рекламные кампании или производственные инциденты. Это поможет заинтересованным сторонам, таким как компании, правительства или общественные организации, оценить эффективность своих действий и принимать информированные решения на основе обратной связи из социальных медиа.
4. **Предоставление инсайтов для принятия решений**: Результаты анализа эмоциональной тональности могут использоваться для принятия стратегических решений в различных сферах, включая маркетинг, общественную политику, управление репутацией и общественные отношения.

Цель проекта заключается не только в технической разработке метода анализа, но и в его практическом применении для получения ценной информации о настроениях и реакциях в социальных медиа, что может иметь важное значение для различных сфер деятельности.

Описание применяемых технологий:

1. **TensorFlow**: TensorFlow - это популярная открытая библиотека для машинного обучения и глубокого обучения, разработанная компанией Google. Она используется для создания и обучения нейронных сетей, включая сверточные нейронные сети (CNN) и рекуррентные нейронные сети (RNN), которые часто применяются в анализе текста и обработке естественного языка.
2. **Matplotlib**: Matplotlib - это библиотека визуализации данных для языка программирования Python и его численных расширений. Она используется для создания различных типов графиков и диаграмм, что может быть полезно для визуализации результатов анализа данных.
3. **Pandas**: Pandas - это библиотека Python для обработки и анализа данных. Она предоставляет структуры данных и функции, которые упрощают работу с табличными данными и временными рядами.
4. **NumPy**: NumPy - это библиотека Python для научных вычислений. Она предоставляет поддержку многомерных массивов и матриц, а также широкий спектр математических функций, что делает ее полезной для работы с данными и их анализа.
5. **NLTK**: NLTK (Natural Language Toolkit) - это библиотека Python для обработки текста и естественного языка. Она предоставляет множество инструментов для токенизации, лемматизации, стемминга, анализа частей речи и многого другого, что может быть полезно при предварительной обработке текстовых данных.
6. **Scikit-learn**: Scikit-learn - это библиотека машинного обучения для Python. Она предоставляет широкий спектр алгоритмов машинного обучения, включая классификацию, регрессию, кластеризацию и многие другие, а также инструменты для оценки и визуализации результатов.
7. **WordCloud**: WordCloud - это библиотека Python для создания облаков слов, которые позволяют визуализировать частоту встречаемости слов в тексте, что может быть полезно для анализа текстовых данных и выявления ключевых тем или терминов.
8. **Keras**: Keras - это высокоуровневый API для нейронных сетей, написанный на Python и способный работать поверх библиотек машинного обучения, таких как TensorFlow, Theano или Microsoft Cognitive Toolkit. Он упрощает создание, обучение и оценку моделей глубокого обучения.
9. **Сonv1D, Bidirectional, LSTM, Dense, Input, Dropout, SpatialDropout1D**: Эти классы и функции предоставляют различные типы слоев и компонентов для построения нейронных сетей в Keras. Например, Conv1D используется для сверточных слоев, LSTM - для рекуррентных слоев с долгой краткосрочной памятью, Dense - для полносвязных слоев, а Dropout - для регуляризации модели и предотвращения переобучения.

Описание датасета

Датасет содержит 1 600 000 твитов, извлеченных с использованием API Twitter. Твиты были аннотированы (0 = отрицательный, 2 = нейтральный, 4 = положительный) и могут использоваться для определения настроения.

В нем содержатся следующие 6 полей:

target: полярность твита (0 = отрицательный, 2 = нейтральный, 4 = положительный)

ids: идентификатор твита (2087) date: дата твита (Сб Май 16 23:58:44 UTC 2009)

flag: запрос (lyx). Если запроса нет, то это значение NO\_QUERY.

user: пользователь, который сделал твит (robotickilldozr)

text: текст твита (Lyx is cool)

Давайте посмотрим на нашу часть датасета



Natural Language Processing(NLP)

Обработка естественного языка (Natural Language Processing, NLP) - это область исследований в рамках искусственного интеллекта, которая занимается разработкой методов и моделей для обработки и анализа естественного языка, используемого людьми для коммуникации. Естественный язык может представляться как текстом, так и звуком, и является основным средством общения между людьми.

Одним из ключевых аспектов NLP является понимание и интерпретация естественного языка компьютерами. Это включает в себя задачи такие как распознавание речи, разбор и синтаксический анализ текста, анализ тональности, машинный перевод и многое другое. В основе этих задач лежит использование методов машинного обучения и глубокого обучения, а также традиционных методов обработки естественного языка.

Классификация текста - это одна из важнейших задач в области анализа настроений в рамках NLP. Это процесс, в ходе которого тексты или документы классифицируются по определенным категориям или настроениям. В контексте анализа настроений, эти категории могут включать в себя положительные, нейтральные и отрицательные настроения, что позволяет оценить эмоциональный окрас текста или мнения пользователей. Анализ настроений широко применяется в различных сферах, таких как маркетинг, общественные отношения, исследования общественного мнения и многое другое.

Классификация настроений представляет собой идеальную задачу для изучения в области NLP, так как она объединяет в себе множество концепций и методов, используемых в обработке естественного языка и машинном обучении. Решение подобных задач может помочь понять и применить различные методы анализа текста, а также научиться работать с различными типами данных, используемыми в NLP, такими как текстовые документы, твиты, отзывы и другие. Кроме того, выполнение проектов в области анализа настроений может предоставить ценный опыт и навыки для дальнейшей карьеры в области искусственного интеллекта и анализа данных.

Основные аспекты NLP

1. Токенизация: Одним из первых шагов в обработке текста является его разделение на отдельные элементы, так называемые "токены". Токенизация может быть выполнена на уровне слов или даже на уровне символов, в зависимости от конкретной задачи.

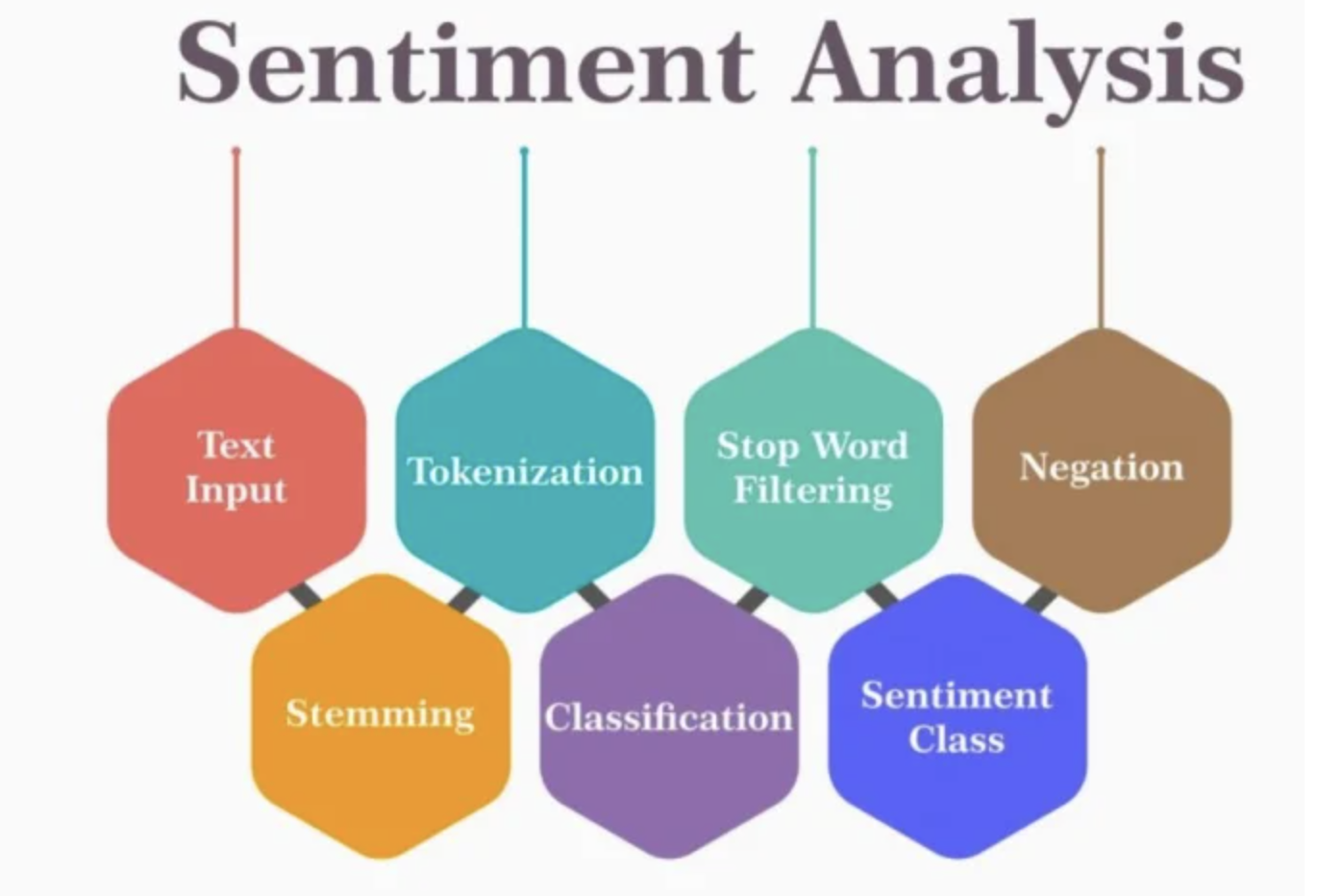
2. Лемматизация и стемминг: Часто в тексте встречаются различные формы одного и того же слова (например, "ходит", "ходил", "ходила"). Лемматизация и стемминг - это процессы приведения слов к их базовой форме (лемме) или к их основе (стему), что позволяет уменьшить размер словаря и улучшить качество анализа.

3. Удаление стоп-слов: В тексте часто встречаются слова, которые не несут смысловой нагрузки, такие как предлоги, союзы и артикли. Удаление таких слов, называемых стоп-словами, помогает сократить размер текста и улучшить качество анализа.

4. Извлечение признаков: После предварительной обработки текста необходимо преобразовать его в числовой формат, который может быть использован для обучения моделей машинного обучения. Для этого часто используются методы векторизации, такие как мешок слов или TF-IDF.

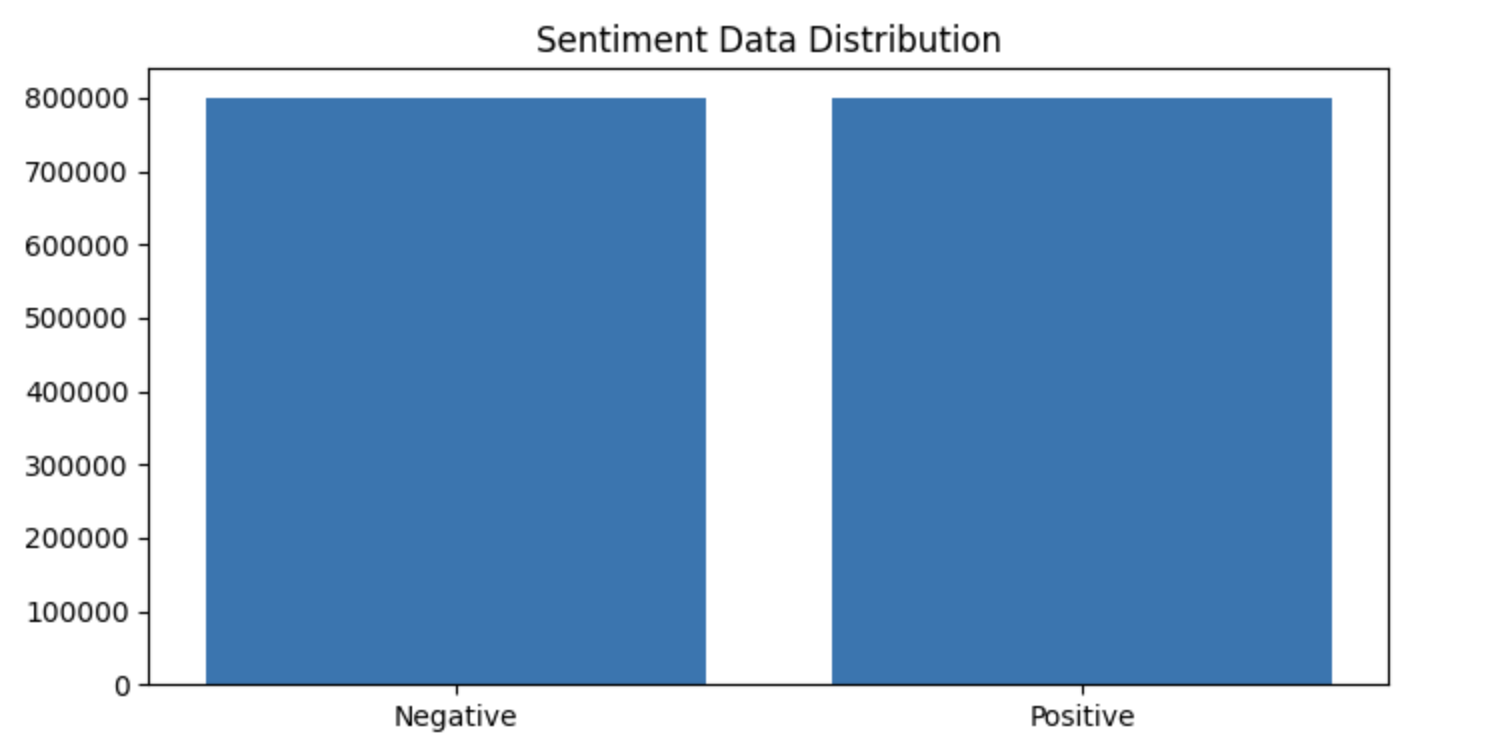
5. Модели глубокого обучения: Современные методы обработки естественного языка все чаще основаны на глубоком обучении, включая рекуррентные нейронные сети (RNN), сверточные нейронные сети (CNN) и трансформеры. Эти модели позволяют добиться высокой точности в задачах анализа текста, таких как классификация, генерация и машинный перевод.

6. Применение в реальном мире: Обработка естественного языка находит широкое применение в реальном мире, включая автоматизацию процессов в бизнесе (например, обработка текстовых запросов клиентов), анализ общественных мнений в социальных медиа, машинный перевод, создание чат-ботов и многое другое.



Распределение данных о настроениях

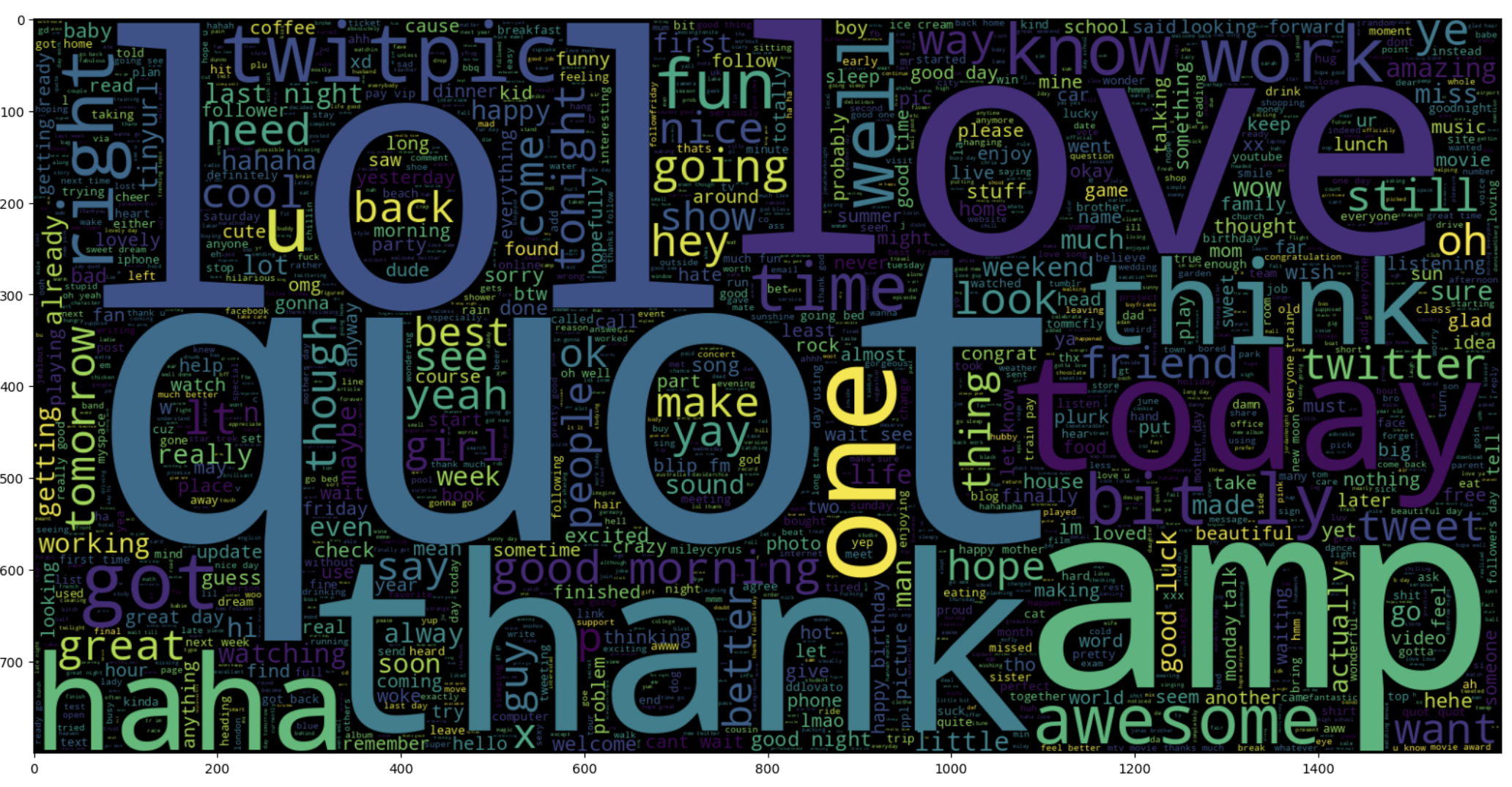
Вот расшифровка меток. Мы сопоставляем 0 -> Отрицательное значение и 1 -> Положительное значение, как указано в описании набора данных. Теперь, когда мы решили, мы проанализируем набор данных по его распределению. Потому что важно, чтобы у нас было практически небольшое количество примеров для данных классов.



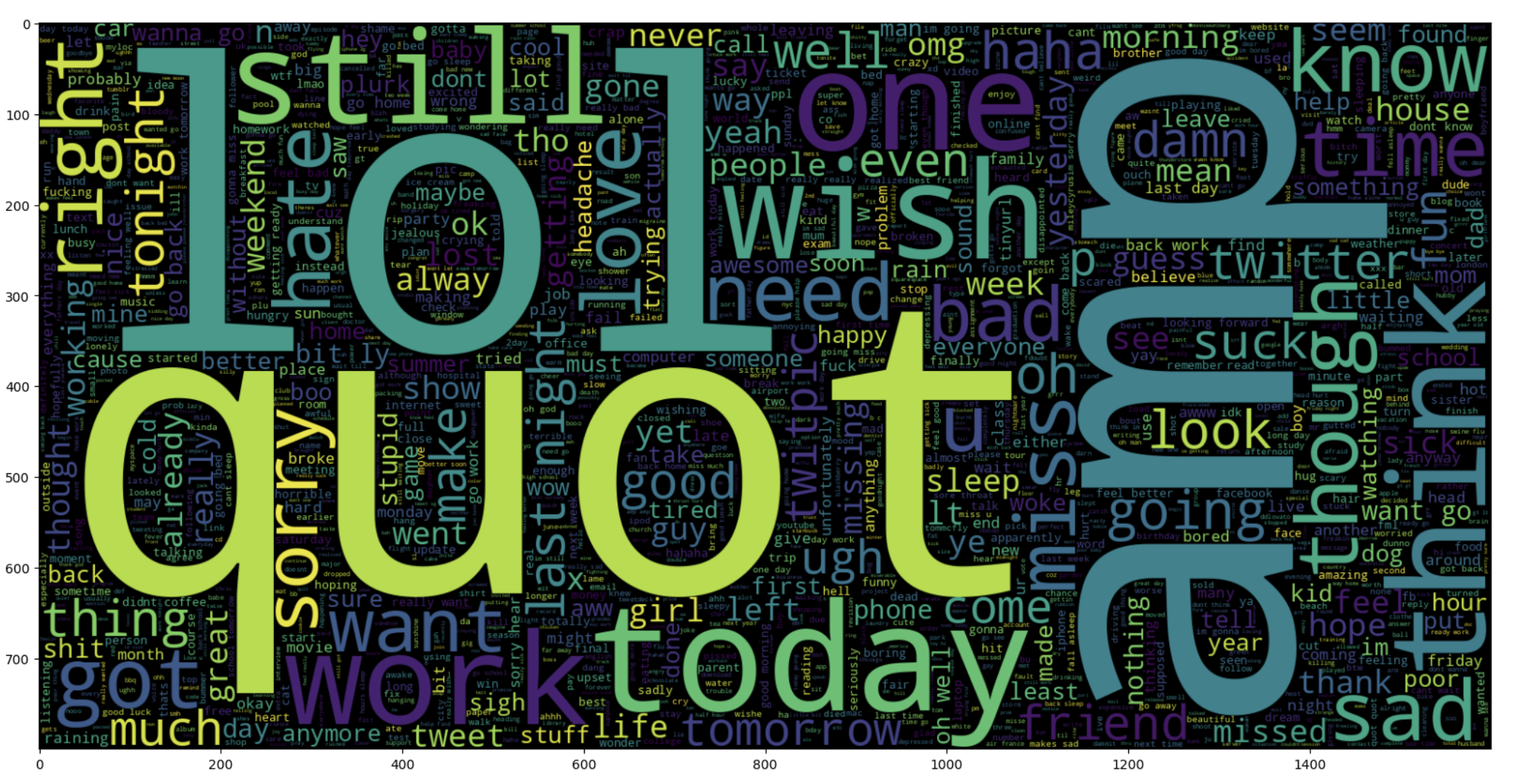
Построение Word Cloud

Теперь мы проанализируем предварительно обработанные данные, чтобы разобраться в них. Мы построим облака слов для положительных и отрицательных твитов из нашего набора данных и посмотрим, какие слова встречаются чаще всего.

Word Cloud c позитивными словами:



Word Cloud с негативными словами:



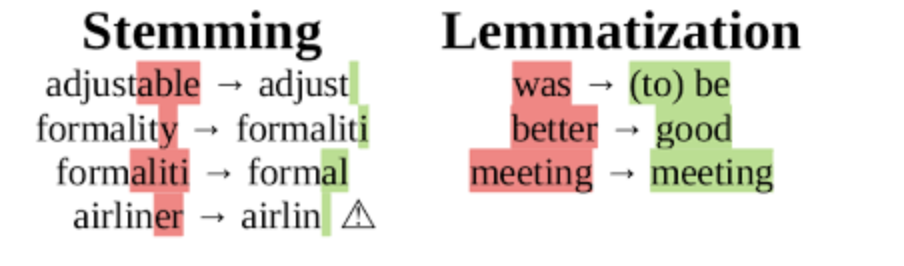
Предварительная обработка текста

Тексты твитов часто содержат упоминания других пользователей, ссылки, эмотиконы и знаки пунктуации. Эти элементы могут быть нежелательными при обучении модели языка, так как они не несут смысловой нагрузки и могут повлиять на качество обучения. Поэтому перед использованием текстов для обучения модели необходимо провести их очистку с использованием различных методов предварительной обработки.

Лемматизация/Стемминг:

Лемматизация и стемминг - это процессы приведения слов к их базовой форме или корню. На английском языке могут существовать различные формы слова, такие как глагол в разных временах или существительное во множественном числе, и обе эти техники помогают сократить разнообразие форм слова до единой базовой формы.

Стемминг обычно основан на правилах удаления окончаний слов, что может привести к неправильной обработке некоторых слов, но в целом обеспечивает хорошую производительность. Лемматизация же использует словарь и морфологический анализ для точного определения базовой формы слова, что делает ее более точной, но требует больше вычислительных ресурсов.



Стоп-слова:

Стоп-слова - это общие слова, которые не несут смысловой нагрузки и часто встречаются в текстах, такие как "и", "в", "на" и т. д. При анализе текста они могут добавлять лишний шум и не давать явных признаков для классификации или анализа. Поэтому перед обработкой текста их удаляют из текстовых данных. Вот некоторые из них  


Разделение на тренировочный и обучающий набор данных и токенизация

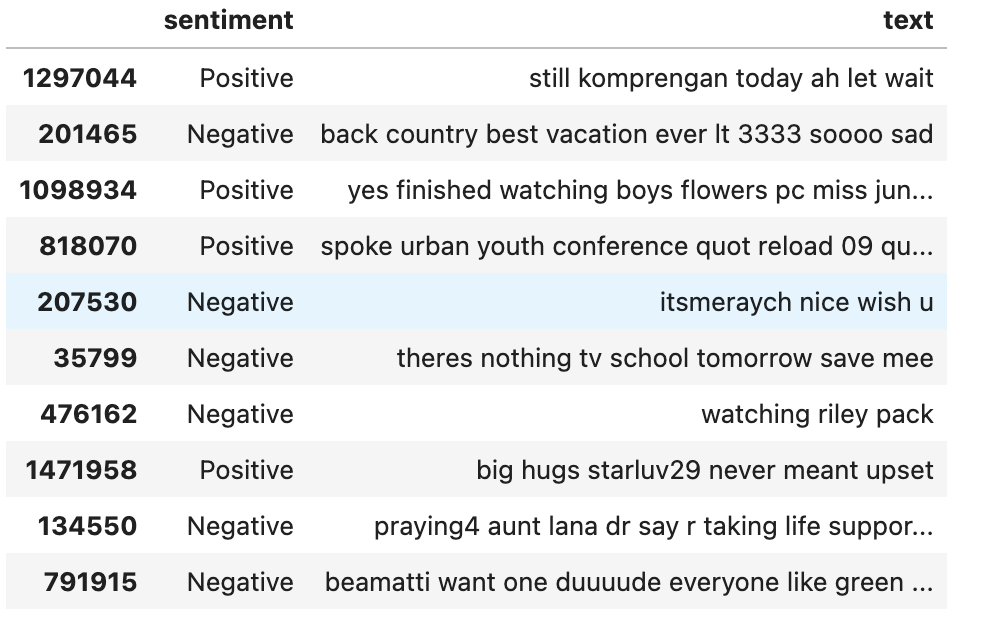
Разделение данных на тренировочный и тестовый наборы является важным шагом в машинном обучении. Этот процесс позволяет оценить производительность модели на новых данных, которые она ранее не видела, что помогает оценить обобщающую способность модели.

1. train\_test\_split: Это функция, предоставляемая библиотекой scikit-learn, которая разделяет набор данных на два набора: тренировочный и тестовый. Она принимает на вход набор данных (`df` в моем случае) и параметр `test\_size`, который определяет размер тестового набора. В вашем коде параметр `test\_size` установлен как `1-TRAIN\_SIZE`, что означает, что размер тестового набора равен `1 - TRAIN\_SIZE`, где `TRAIN\_SIZE` - это размер тренировочного набора.

2. random\_state: Этот параметр определяет случайное начальное состояние для генератора псевдослучайных чисел, используемого для разделения данных. Установка `random\_state` в определенное значение гарантирует воспроизводимость результатов при каждом запуске кода.

3. Размеры тренировочного и тестового наборов: После выполнения разделения данных мы выводим размеры полученных наборов с помощью функции `len()`. Размер тренировочного набора равен 160 000, а размер тестового набора - 1 440 000. Это означает, что 10% исходных данных используются в качестве тестового набора, а остальные 90% - в качестве тренировочного.

4. Использование разделенных наборов: Полученные наборы данных (`train\_data` и `test\_data`) могут быть далее использованы для обучения и тестирования моделей машинного обучения соответственно. Обычно тренировочный набор используется для обучения модели, а тестовый - для оценки ее производительности и обобщающей способности.

Давайте посмотрим на часть нашей тренировочной таблицы:  


Определение: Токенизация - это процесс разделения текста на более мелкие единицы, называемые токенами. Токены могут быть словами, числами, символами или другими элементами, в зависимости от конкретной задачи.

Цель: Основная цель токенизации состоит в том, чтобы разбить текст на более мелкие части, чтобы его можно было обработать и анализировать более эффективно. Токенизация является первым шагом в предварительной обработке текста и часто необходима для последующих задач, таких как анализ тональности, машинный перевод, извлечение информации и другие.

1. Типы токенизации:

- Токенизация на уровне слов: Текст разбивается на отдельные слова. Например, предложение "Я люблю кофе" будет токенизировано на три слова: "Я", "люблю", "кофе".

- Токенизация на уровне символов: Текст разбивается на отдельные символы. Например, предложение "кофе" будет токенизировано на четыре символа: "к", "о", "ф", "е".

- Токенизация на уровне предложений: Текст разбивается на отдельные предложения. Этот тип токенизации используется, когда необходимо анализировать текст по предложениям, а не по словам или символам.

2. Методы токенизации: В зависимости от конкретной задачи и языка программирования можно использовать различные методы токенизации. В Python, например, для токенизации текста можно использовать функции из библиотеки NLTK (Natural Language Toolkit), регулярные выражения или встроенные функции разделения строк.

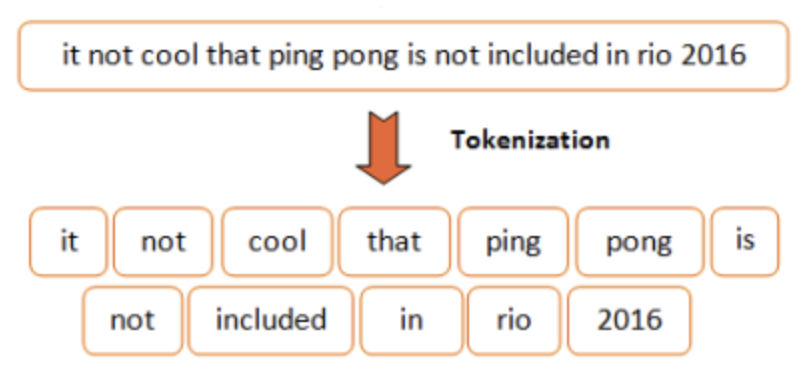
3. Обработка специальных случаев: При токенизации могут возникать специальные случаи, такие как обработка смайликов, аббревиатур, чисел и др. Некоторые методы токенизации могут автоматически обрабатывать эти случаи, в то время как другие требуют дополнительной настройки или обработки.

Токенизация является важным шагом в обработке текста и может существенно повлиять на качество и результаты последующего анализа. Правильный выбор метода и правильная настройка параметров токенизации играют важную роль в успешной обработке текстовых данных.

Описание переменных токенизации:

- word\_index содержит индекс для каждого слова.

- vocab\_size представляет собой общее количество слов в корпусе данных.



Word Embedding

Вставка слов (Word Embedding) - это техника в области обработки естественного языка (Natural Language Processing, NLP), которая представляет слова в виде векторов чисел в многомерном пространстве. Это позволяет компьютеру эффективно обрабатывать и анализировать текст, улавливая семантические и синтаксические аспекты слов.

Вот некоторые ключевые аспекты вставки слов:

1. Представление слов в векторном формате: Вместо того чтобы представлять слова как отдельные единицы, вставка слов преобразует каждое слово в вектор чисел определенной размерности. Каждое измерение вектора может представлять семантические или контекстуальные характеристики слова.

2. Захват семантического и синтаксического значения: Вставки слов способны улавливать семантические и синтаксические свойства слов, такие как их значение, контекст и отношения с другими словами. Например, слова с похожим смыслом будут иметь близкие векторные представления в многомерном пространстве.

3. Использование в машинном обучении: Вставки слов широко используются в задачах обработки естественного языка и машинного обучения. Они являются ключевым компонентом в различных моделях NLP, таких как рекуррентные нейронные сети (RNN), сверточные нейронные сети (CNN) и трансформеры.

4. Обучение вставок слов: Вставки слов могут быть обучены на больших текстовых корпусах с помощью различных алгоритмов, таких как Word2Vec, GloVe, FastText и т. д. Эти алгоритмы позволяют компьютеру выучить семантические отношения между словами на основе их контекста в тексте.

5. Применение в передаче обучения: Для задач с ограниченным количеством данных можно использовать предварительно обученные вставки слов, которые были обучены на больших текстовых наборах данных. Это позволяет модели получить представление о словах на основе обширного контекста и использовать его для решения конкретной задачи.

6. Размерность векторов и контекст: Выбор размерности векторов и контекста, который учитывается при обучении вставок слов, может существенно влиять на качество результирующих векторов. Большая размерность векторов может захватывать более сложные семантические отношения, но требует больше вычислительных ресурсов.

В целом, вставки слов являются мощным инструментом для работы с текстовыми данными в машинном обучении и NLP, позволяя компьютеру эффективно понимать и обрабатывать естественный язык.

Обучение модели LSTM модели

Давайте рассмотрим детально долгую краткосрочную память (LSTM) - это специальный тип рекуррентной нейронной сети (RNN), который эффективно работает с последовательными данными, такими как тексты, аудио и временные ряды. LSTM были разработаны для решения проблемы затухания и взрывного градиента, с которыми сталкиваются обычные RNN при обучении на длинных последовательностях.

Вот основные компоненты LSTM:

1. Клетка памяти (Memory Cell): Это главная часть LSTM, которая хранит информацию о прошлых состояниях и взаимодействует с текущим входом и предыдущим скрытым состоянием. Клетка памяти помогает сохранять информацию на протяжении длинных последовательностей и избегать затухания градиента.

2. Фильтры забывания (Forget Gate): Этот механизм определяет, какую информацию следует сохранить и какую следует забыть из предыдущего состояния клетки памяти. Фильтры забывания принимают на вход текущий вход и скрытое состояние предыдущего временного шага и генерируют вес, который умножается на предыдущее состояние клетки памяти.

3. Входной затвор (Input Gate): Этот механизм решает, какая информация должна быть обновлена в клетке памяти на основе текущего входа и предыдущего скрытого состояния. Он определяет, какую часть нового входа следует добавить к клетке памяти.

4. Выходной затвор (Output Gate): Этот механизм решает, какая информация из клетки памяти будет использоваться для предсказания на текущем временном шаге. Он определяет, какая часть содержимого клетки памяти будет передана на выход.

5. Функция активации (Activation Function): LSTM использует различные функции активации, такие как сигмоид и тангенс гиперболический, для контроля потока информации внутри сети и принятия решения о том, какая информация должна быть сохранена или отброшена.

6. Обучаемые параметры (Trainable Parameters): LSTM содержат множество обучаемых параметров, которые настраиваются в процессе обучения сети, включая веса для фильтров забывания, входных и выходных затворов, а также веса для обновления клетки памяти и вычисления выходного значения.

Преимущества LSTM включают способность эффективно обрабатывать длинные последовательности, сохранять долгосрочные зависимости в данных и избегать проблем затухания и взрывного градиента. Они широко применяются в задачах анализа текста, машинного перевода, распознавания речи, генерации текста и других приложениях, где важна обработка последовательных данных.

Архитектура LSTM

Рекуррентные нейронные сети могут обрабатывать последовательность данных и изучать последовательность входных данных, чтобы выдавать либо последовательность, либо скалярное значение в качестве выходных данных. В нашем случае нейронная сеть выдает предсказание скалярного значения.

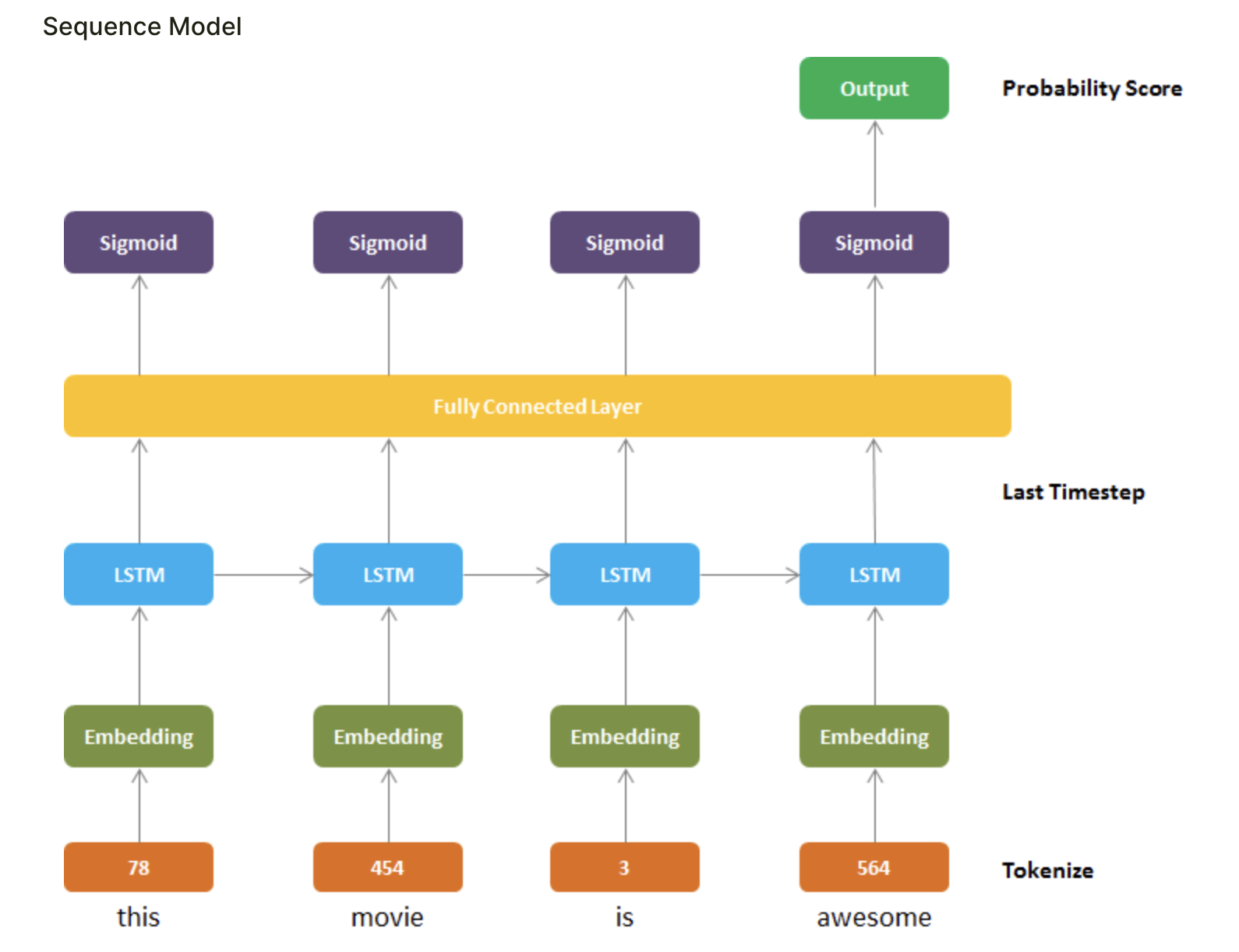
Для архитектуры модели мы используем

1) Слой встраивания - генерирует вектор встраивания для каждой входной последовательности.

2) Слой Conv1 - используется для свертки данных в векторы объектов меньшего размера.

3) LSTM - Долговременная кратковременная память, это вариант RNN, который имеет ячейку состояния памяти для изучения контекста слов, которые находятся дальше по тексту, чтобы нести контекстуальное значение, а не просто соседние слова, как в случае с RNN.

4) Плотные - Полностью соединенные слои для классификации



Алгоритм оптимизации и обучение модели

1. Adam (Adaptive Moment Estimation):

- Описание: Adam - это алгоритм оптимизации, используемый для обновления весов модели в процессе обучения с использованием градиентного спуска. Он является комбинацией методов Momentum и RMSprop, что делает его адаптивным и эффективным для широкого спектра задач машинного обучения.

- Работа: Adam поддерживает адаптивные скорости обучения для каждого параметра, обновляя их в соответствии с оценками первого и второго моментов градиентов. Это позволяет алгоритму быстро сходиться к оптимальным значениям весов модели.

2. Обратные вызовы (Callbacks):

- Описание: Обратные вызовы - это специальные функции, которые выполняются в конце каждой эпохи обучения модели или при выполнении определенного условия. Они используются для настройки поведения обучения и мониторинга процесса обучения.

- LR-планировщик (Learning Rate Scheduler):

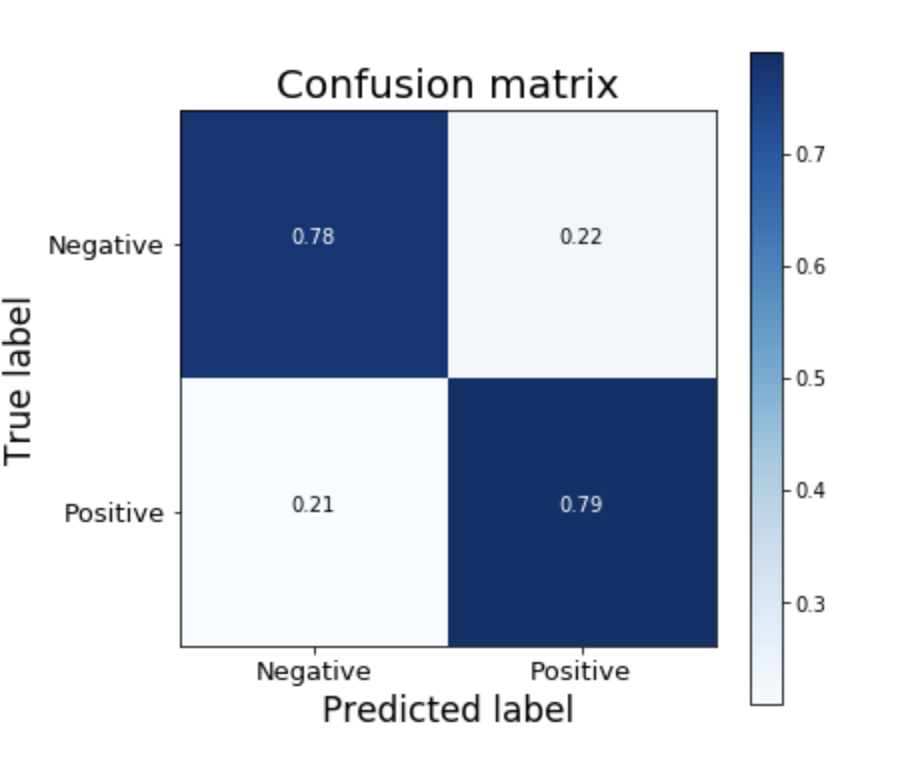
- Описание: LR-планировщик изменяет скорость обучения в определенные моменты времени или на основе определенных условий. В данном случае скорость обучения экспоненциально снижается после определенного числа эпох для улучшения сходимости модели.

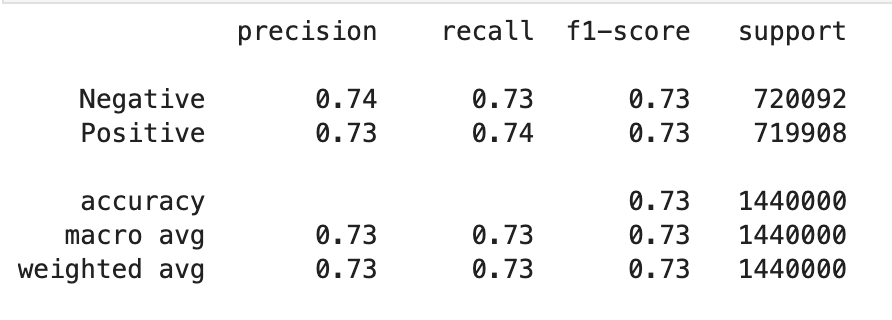
- Контрольная точка модели (Model Checkpoint):

- Описание: Контрольная точка модели сохраняет текущее состояние модели в файл на диске после каждой эпохи или при достижении определенных метрических показателей. В данном случае модель сохраняется с минимальной потерей достоверности (validation loss) для последующего использования или восстановления.

Эти аспекты алгоритма оптимизации и обратных вызовов помогают эффективно управлять процессом обучения модели, улучшая ее сходимость и сохраняя наилучшие результаты для последующего использования.

Нормализованная матрица ошибок и метрики





Эти метрики представляют собой оценки качества работы модели классификации для двух классов: "Negative" (отрицательные отзывы) и "Positive" (положительные отзывы). Давайте рассмотрим каждую метрику подробнее:

1. Точность (Precision):

- Определение: Точность - это доля правильно классифицированных положительных примеров относительно всех примеров, которые модель предсказала как положительные.

- Формула: Точность = TP / (TP + FP), где TP - количество истинно положительных примеров, а FP - количество ложно положительных примеров.

- Значение: Для класса "Negative" точность составляет 0.74, что означает, что модель правильно предсказывает 74% отрицательных отзывов относительно всех отзывов, которые она предсказывает как отрицательные. Для класса "Positive" точность также составляет 0.73, что означает, что модель правильно предсказывает 73% положительных отзывов относительно всех отзывов, которые она предсказывает как положительные.

2. Полнота (Recall):

- Определение: Полнота - это доля правильно классифицированных положительных примеров относительно всех фактически положительных примеров.

- Формула: Полнота = TP / (TP + FN), где TP - количество истинно положительных примеров, а FN - количество ложно отрицательных примеров.

- Значение: Для класса "Negative" полнота составляет 0.73, что означает, что модель правильно идентифицирует 73% отрицательных отзывов относительно всех фактически отрицательных отзывов. Для класса "Positive" полнота также составляет 0.74, что означает, что модель правильно идентифицирует 74% положительных отзывов относительно всех фактически положительных отзывов.

3. F1-мера (F1-score):

- Определение: F1-мера - это гармоническое среднее точности и полноты и является мерой общей точности классификации.

- Формула: F1-мера = 2 \* (Precision \* Recall) / (Precision + Recall).

- Значение: F1-мера для класса "Negative" и "Positive" составляет 0.73, что означает, что модель обладает хорошим балансом между точностью и полнотой для обоих классов.

4. Поддержка (Support):

- Определение: Поддержка - это количество фактических экземпляров каждого класса в наборе данных.

- Значение: Для класса "Negative" поддержка составляет 720092, а для класса "Positive" - 719908. Эти значения указывают на распределение классов в наборе данных.

5. Точность классификации (Accuracy):

- Определение: Точность классификации - это доля правильно классифицированных примеров относительно всех примеров в наборе данных.

- Формула: Точность классификации = (TP + TN) / (TP + TN + FP + FN), где TN - количество истинно отрицательных примеров.

- Значение: Точность классификации составляет 0.73, что означает, что модель правильно классифицирует 73% всех примеров в наборе данных.

Список литературы

1. https://www.linkedin.com/pulse/social-machine-learning-h2o-twitter-python-marios-michailidis

2. <https://www.kaggle.com/datasets/kazanova/sentiment140>

3. <https://keras.io/>

4. <https://www.tensorflow.org/?hl=ru>

5. <https://nlp.stanford.edu/projects/glove/>

6. <https://www.coursera.org/learn/nlp-sequence-models>

7. <https://karpathy.github.io/2015/05/21/rnn-effectiveness/>

8. <https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010>

9. <https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html>

10. <https://habr.com/ru/companies/Voximplant/articles/446738/>