**Введение**

В современном цифровом мире, где онлайн-платформы стали основным средством коммуникации, важность поддержания здорового и конструктивного диалога в интернете неоспорима. В то время как эти платформы открывают безграничные возможности для обмена знаниями и идеями, они также сталкиваются с проблемой токсичных комментариев, которые могут отравлять общественный дискурс и способствовать распространению негатива. Распознавание и фильтрация нетоксичных комментариев становится ключевым аспектом в создании здоровой цифровой среды.

Цель данного проекта заключается в разработке модели машинного обучения, способной эффективно идентифицировать нетоксичные комментарии в больших объемах текстовой информации. Такой подход позволит улучшить качество онлайн-дискуссий, обеспечивая при этом свободу слова и выражения мнений без подвергания пользователей негативному контенту.

В рамках проекта используются современные методы обработки естественного языка (NLP), включая токенизацию, удаление стоп-слов, лемматизацию и векторизацию текстовых данных. Основным методом машинного обучения, применяемым в проекте, является классификация с использованием различных алгоритмов, что позволяет точно определить, является ли конкретный комментарий токсичным или нет.

**Обзор литературы:**

В этом проекте были использованы следующие библиотеки и инструменты, каждый из которых играет ключевую роль в обработке и анализе текстовых данных:

**Pandas**: Мощная библиотека для анализа и манипуляции данными в Python. Она использовалась для чтения, очистки и подготовки данных перед анализом.

**NumPy**: Основная библиотека для научных вычислений в Python. Применялась для эффективной работы с массивами данных и выполнения математических операций.

**Scikit-learn:** Одна из ведущих библиотек машинного обучения для Python. Использовалась для обучения моделей классификации, валидации результатов и оценки эффективности модели.

**NLTK (Natural Language Toolkit):** Библиотека для работы с естественным языком, включающая инструменты для токенизации, стемминга, лемматизации и удаления стоп-слов.

**Описание данных**

В данном проекте использовался набор данных toxic\_comments.csv, предназначенный для анализа и классификации комментариев на токсичные и нетоксичные.

**Структура данных**

Набор данных состоит из следующих ключевых столбцов:

text: Содержит текст комментария. Это основное поле для анализа и обработки.

toxic: Целевой признак, который указывает на токсичность комментария. Обычно представлен в виде бинарного значения (0 или 1), где 1 обозначает токсичный комментарий, а 0 — нетоксичный.

**Предобработка данных**

Для подготовки данных к анализу были выполнены следующие шаги:

Очистка данных: удаление данных с целью оптимизации расчетов.

Очистка текста: Удаление лишних символов, таких как знаки препинания и специальные символы, для упрощения анализа текста.

Токенизация: Разбиение текста на отдельные слова или токены для дальнейшей обработки.

# Функция для токенизации текста

def tokenize\_text(text):

return word\_tokenize(text)

# Применяем токенизацию ко всем текстам в датасете

df['tokenized\_text'] = df['text'].apply(tokenize\_text)

# Просмотрим результаты токенизации для первых нескольких записей

df['tokenized\_text'].head()

Удаление стоп-слов: Исключение слов, которые часто встречаются, но не несут значимой информации для анализа, например, предлогов и союзов.

# Получаем список английских стоп-слов

stop\_words = set(stopwords.words('english'))

stop\_words

# Функция для удаления стоп-слов из токенизированного текста

def remove\_stop\_words(tokens):

return [word for word in tokens if word.lower() not in stop\_words]

# Удаляем стоп-слова из каждого токенизированного текста

df['text\_without\_stopwords'] = df['tokenized\_text'].apply(remove\_stop\_words)

# Просмотрим результаты для первых нескольких записей

df['text\_without\_stopwords'].head()

Лемматизация: Приведение слов к их основной форме для унификации различных форм слова.

# Инициализируем лемматизатор

lemmatizer = WordNetLemmatizer()

# Кэш для POS-тегов

pos\_cache = {}

# Функция для определения части речи слова

def get\_wordnet\_pos(word):

if word in pos\_cache:

return pos\_cache[word]

tag = pos\_tag([word])[0][1][0].upper()

tag\_dict = {"J": wordnet.ADJ,

"N": wordnet.NOUN,

"V": wordnet.VERB,

"R": wordnet.ADV}

pos\_cache[word] = tag\_dict.get(tag, wordnet.NOUN)

return pos\_cache[word]

# Функция для лемматизации текста

def lemmatize\_text(tokens):

return [lemmatizer.lemmatize(word, get\_wordnet\_pos(word)) for word in tokens]

# Применяем лемматизацию к каждому тексту

df['lemmatized\_text'] = df['text\_without\_stopwords'].apply(lemmatize\_text)

# Просмотрим результаты для первых нескольких записей

df['lemmatized\_text'].head()

**Векторизация:** Преобразование обработанных текстов в числовые векторы, которые затем используются в моделях машинного обучения.

from sklearn.feature\_extraction.text import TfidfVectorizer

import gc

# Объединяем лемматизированные слова обратно в строки

df['lemmatized\_text\_joined'] = df['lemmatized\_text'].apply(' '.join)

# Инициализируем векторизатор TF-IDF с ограничением максимального количества признаков

vectorizer = TfidfVectorizer(max\_features=10000) # Пример: ограничиваем словарь 10,000 словами

# Применяем векторизатор к лемматизированным текстам

tfidf\_matrix = vectorizer.fit\_transform(df['lemmatized\_text\_joined'])

# Освобождаем память

del df['lemmatized\_text\_joined']

gc.collect()

# tfidf\_matrix является разреженной матрицей

# Для просмотра результатов векторизации можно преобразовать её в DataFrame

tfidf\_df = pd.DataFrame(tfidf\_matrix.toarray(), columns=vectorizer.get\_feature\_names\_out())

# Просмотрим первые несколько строк полученной матрицы

tfidf\_df.head()

Эти этапы предобработки данных были критически важны для обеспечения качества и эффективности последующего анализа и классификации комментариев.

**Методология работы:**

При оценке производительности модели машинного обучения часто используют следующие метрики: точность (precision), полноту (recall), F1-меру (F1-score) и общую точность (accuracy).

**Точность (Precision):** Точность измеряет, какую долю объектов, классифицированных как положительные (например, класс 1), действительно являются положительными. Формула для точности:

Точность = TP / (TP + FP)

Где:

TP (True Positives) - количество верно классифицированных положительных объектов.

FP (False Positives) - количество неверно классифицированных положительных объектов.

**Полнота (Recall):** Полнота (или чувствительность) измеряет, какую долю всех положительных объектов модель смогла обнаружить. Формула для полноты:

Полнота = TP / (TP + FN)

Где:

TP (True Positives) - количество верно классифицированных положительных объектов.

FN (False Negatives) - количество положительных объектов, но классифицированных как отрицательные.

**F1-мера (F1-Score):** F1-мера является средним гармоническим между точностью и полнотой и используется для балансировки между этими двумя метриками. Она полезна, когда важны как точность, так и полнота, и нужно учесть обе метрики в одном числе. Формула для F1-меры:

F1-мера = 2 \* (Точность \* Полнота) / (Точность + Полнота)

**Общая точность (Accuracy):** Общая точность измеряет, какую долю всех объектов модель классифицировала верно (как положительные и отрицательные). Формула для общей точности:

Точность = (TP + TN) / (TP + TN + FP + FN)

Где:

TP (True Positives) - количество верно классифицированных положительных объектов.

TN (True Negatives) - количество верно классифицированных отрицательных объектов.

FP (False Positives) - количество неверно классифицированных положительных объектов.

FN (False Negatives) - количество положительных объектов, но классифицированных как отрицательные.

**Вывод проекта:**

Обработка естественного языка (NLP) - это подмножество машинного обучения, в котором мы стремимся обучать компьютеры понимать человеческие языки. В процессе обработки естественного языка мы стремимся подготовить текстовый набор данных для построения словаря для классификации текста или кластеризации.

Полный процесс обработки естественного языка включает в себя:

Сбор текстовых данных или документов

Токенизация

Удаление слов

Векторизация

Классификация или кластеризация текста

В ходе работы были полученные следующие результаты:

Точность (precision) для класса 0 составляет 0.95, что означает, что 95% объектов, которые модель классифицировала как класс 0, действительно принадлежат к этому классу. Точность для класса 1 равна 0.94, что также является высоким значением.

Полнота (recall) для класса 0 составляет 1.00, что говорит о том, что модель способна обнаруживать почти все объекты класса 0. Однако полнота для класса 1 ниже и составляет 0.58, что может указывать на то, что модель пропускает некоторые объекты класса 1.

F1-мера (f1-score) является средним гармоническим между точностью и полнотой. Для класса 0 она составляет 0.97, что является очень хорошим показателем. Для класса 1 F1-мера равна 0.71, что также является приемлемым результатом.

Общая точность модели (accuracy) составляет 0.95, что означает, что модель правильно классифицировала 95% всех объектов в выборке.

Таким образом, в ходе проекта, был реализован процесс обработки естественного языка и получены высокие результаты работы модели.