Используя данные датасета, https://www.kaggle.com/datasets/asaniczka/tmdb-movies-dataset-2023-930k-movies/, реализовать задачу анализа аннотаций к фильмам. Результатом работы должен быть список слов из аннотаций и рецензий, влияющий на продвижение фильма.

Шаги:

1. Подготовка данных

a. Очистка текста от неалфавитных символов.

b. Токенизация.

c. Лемматизация.

d. Удаление стоп-слов.

e. Векторизация текста.

2. Выбор признаков

3. Обучение модели

Оформить выводы.

Теоретические материалы:

nltk – библиотека для обработки естественного языка, которая предоставляет множество функций для работы с текстом на Python. Рассмотрим основные возможности:

· Токенизация: разбиение текста на токены (слова, числа, знаки препинания), для последующего анализа или обработки.

· Стемминг и лемматизация: процессы нормализации текста путем приведения словоформ к их основам (стемминг) или формам словарных слов (лемматизация).

· POS-тэггинг: определение частей речи в тексте (существительное, глагол, прилагательное и т.д.).

· Анализ синтаксиса: разбор предложений на составляющие (подлежащее, сказуемое, дополнение и т.д.) для более глубокого понимания текста.

· Анализ сентимента: определение тональности текста (позитивная/негативная).

Также в nltk есть множество инструментов для обучения и классификации текстовых данных, а также для работы с корпусами текстов – наборами текстовых данных для исследования и обучения моделей. Библиотека поддерживает работу с текстом на многих языках, в том числе и на русском. Для установки и дополнительной информации можно посетить официальный сайт nltk (https://www.nltk.org/).

Токенизация текста

Токенизация текста является первым шагом в обработке естественного языка. Она заключается в разбиении текста на отдельные слова или токены. Библиотека nltk python содержит несколько методов для токенизации русского текста.

Метод word\_tokenize

Метод word\_tokenize разбивает текст на отдельные слова и знаки препинания. Каждый элемент списка представляет собой отдельный токен:

Пример использования:

from nltk.tokenize import word\_tokenize

text = "Это пример русского предложения. Оно содержит несколько слов!"

tokens = word\_tokenize(text, language='russian')

РЕКЛАМА

print(tokens)

Вывод: [‘Это’, ‘пример’, ‘русского’, ‘предложения’, ‘.’, ‘Оно’, ‘содержит’, ‘несколько’, ‘слов’, ‘!’]

Метод regexp\_tokenize

Метод regexp\_tokenize позволяет использовать регулярные выражения для токенизации текста. Этот метод более гибкий, чем word\_tokenize, и может использоваться для токенизации по определенному шаблону:

Пример использования:

from nltk.tokenize import regexp\_tokenize

text = "Стоит ехать на автобусе №23 до остановки 'Улица Пушкина'."

pattern = "w+|[^ws]+"

tokens = regexp\_tokenize(text, pattern, gaps=False)

print(tokens)

Вывод: [‘Стоит’, ‘ехать’, ‘на’, ‘автобусе’, ‘№23’, ‘до’, ‘остановки’, «‘», ‘Улица’, ‘Пушкина’, «‘», ‘.’ ]

Метод PunktSentenceTokenizer

Метод PunktSentenceTokenizer используется для токенизации предложений в тексте. Он использует обучающую выборку для определения, где заканчивается предложение и начинается новое:

Пример использования:

from nltk.tokenize import PunktSentenceTokenizer

text = "Это пример русского текста. Второе предложение следует за первым. Третье предложение заканчивает текст."

tokenizer = PunktSentenceTokenizer()

sentences = tokenizer.tokenize(text)

print(sentences)

Вывод: [‘Это пример русского текста.’, ‘Второе предложение следует за первым.’, ‘Третье предложение заканчивает текст.’]

Лемматизация – процесс приведения словоформы к начальной форме – лемме. Например, словоформы «бежала», «бежит», «бег» после лемматизации будут приведены к лемме «бежать». Лемматизация позволяет упростить обработку текста и уменьшить количество разных слов в тексте, что положительно влияет на результаты анализа.

Стемминг – это процесс удаления окончаний слов для получения основы. Например, слова «бежала», «бежит», «бег» после стемминга станут «беж», «беж», «бег». Стемминг более грубый метод, чем лемматизация, т.к. в результате стемминга получается не обязательно существующее слово.

В библиотеке nltk для русского языка доступны методы лемматизации и стемминга. Для лемматизации можно использовать класс Mystem из модуля pymystem3:

from pymystem3 import Mystem

m = Mystem()

lemmatized = m.lemmatize("бежит бегущий бег")

print(lemmatized)

В результате выполнения кода на экран будет выведено: [‘бежать’, ‘ ‘, ‘бежать’, ‘ ‘, ‘бег’]

Для стемминга можно использовать класс SnowballStemmer из модуля nltk.stem.snowball:

from nltk.stem.snowball import SnowballStemmer

stemmer = SnowballStemmer("russian")

stemmed = stemmer.stem("бежит бегущий бег")

print(stemmed)

В результате выполнения кода на экран будет выведено: беж бегущ бег

Выбор между лемматизацией и стеммингом зависит от поставленной задачи. Если нужно получить базовую форму слова, то лемматизация – более подходящий метод. Если же нужно быстро и грубо производить обработку текста, то стемминг – более быстрый и менее ресурсозатратный метод.

Частотный анализ

Частотный анализ текста — это метод исследования частоты появления отдельных слов или групп слов в тексте. Он часто используется в лингвистике, теории информации и компьютерной лингвистике, включая анализ текстового контента в социальных сетях и машинный перевод.

Частотный анализ может помочь оценить ключевые темы и ключевые слова в тексте, а также выявить чрезмерное повторение слов или фраз, что может указывать на их значимость в тексте. Он также может помочь в идентификации авторства текста или документов, поскольку люди часто используют определенные слова и фразы, которые становятся частыми в их текстах.

При проведении частотного анализа важно учитывать, что он может быть ненадежным, если не учитывать контекст, например, одно и то же слово может иметь различные значения в разных контекстах и значительно влиять на результаты частотного анализа. Поэтому необходимо применять частотный анализ в сочетании с другими методами анализа текста и учитывать контекст при интерпретации результатов.