**Алгоритм решения задачи**

Предварительная обработка: Выполните все необходимые шаги по предварительной обработке текстов, например, удалите знаки препинания, преобразуйте их в строчные буквы или удалите стоп-слова. Для этого можно использовать такие библиотеки, как NLTK или spaCy.

Токенизация: Токенизируйте предварительно обработанные тексты, чтобы преобразовать их в последовательности лексем. Для этого шага можно использовать класс Tokenizer из Keras. Настройте токенизатор на обучающие тексты для создания словаря, а затем преобразуйте тексты в последовательности с помощью метода tokenizer's texts\_to\_sequences.

Вставка: Разделите последовательности, чтобы сделать их одинаковой длины. Это необходимо, поскольку нейронным сетям требуются входы фиксированного размера. Вы можете использовать функцию pad\_sequences из Keras, чтобы добавить вставки в последовательности.

Разделите данные: Разделите предварительно обработанные последовательности и оценки на обучающий и проверочный наборы с помощью функции train\_test\_split из scikit-learn. Укажите желаемое соотношение или количество образцов для проверочного набора.

Постройте модель: Создайте последовательную модель, используя класс Sequential из Keras. Добавьте в модель необходимые слои, такие как слой встраивания, конволюционные слои, слои объединения и плотные слои. Настройте гиперпараметры и архитектуру модели в зависимости от вашей задачи и данных.

Скомпилируйте модель: Составьте модель, используя подходящую функцию потерь и оптимизатор. Выберите метрику оценки, основанную на вашей проблеме, например, среднюю квадратичную ошибку (MSE) или среднюю абсолютную ошибку (MAE) для задач регрессии.

Обучите модель: Обучите модель на тренировочных данных, используя метод fit. Укажите количество эпох, размер партии и данные для проверки. Контролируйте процесс обучения и при необходимости корректируйте гиперпараметры.

Оцените модель: Оцените обученную модель на проверочном множестве с помощью метода evaluate. Вычислите показатели эффективности, такие как MSE или MAE, чтобы оценить точность модели.

Делаем прогнозы: Обработайте все новые входные тексты с помощью тех же шагов предварительной обработки, которые применялись к обучающим данным. Преобразуйте предварительно обработанные тексты в последовательности с помощью обученного токенизатора и заполните последовательности. Передайте набивные последовательности обученной модели для предсказания с помощью метода predict.

Постобработка прогнозов: При необходимости преобразуйте предсказанные оценки в нужный формат или шкалу. Вы можете округлить оценки, применить порог или сопоставить их с определенным диапазоном.

Вывод результатов: Распечатайте или сохраните прогнозируемые оценки для дальнейшего анализа или использования.

**Листинг программы**

mport numpy as np

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense

from sklearn.model\_selection import train\_test\_split

# Загрузка данных

# texts = [

# "Сериал очень люблю, но Академия и Земля вызывает у меня отторжение идеей... Не люблю, когда принижают ценность человека как личности, даже не смотря на ошибки личности...",

# "думал, что будет лучше идея очень интересна - ...",

# "с творчеством Головачева я познакомился посред...",

# ]

# scores = [

# 6,

# 7,

# 10

# ]

# Предварительная обработка данных

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts(texts)

sequences = tokenizer.texts\_to\_sequences(texts)

vocab\_size = len(tokenizer.word\_index) + 1

max\_length = max(len(seq) for seq in sequences)

# Преобразование оценок в массив NumPy

scores = np.array(scores)

# Разделение данных на обучающую и тестовую выборки

train\_sequences, val\_sequences, train\_scores, val\_scores = train\_test\_split(sequences, scores, test\_size=0.2, random\_state=42)

# Построение модели сверточной нейронной сети

embedding\_dim = 100

num\_filters = 128

filter\_size = 3

model = Sequential()

model.add(Embedding(vocab\_size, embedding\_dim, input\_length=max\_length))

model.add(Conv1D(num\_filters, filter\_size, activation='relu'))

model.add(GlobalMaxPooling1D())

model.add(Dense(1, activation='linear'))

model.compile(loss='mean\_squared\_error', optimizer='adam', metrics=['mean\_squared\_error'])

model.fit(pad\_sequences(train\_sequences, maxlen=max\_length), train\_scores, epochs=10, batch\_size=32, validation\_data=(pad\_sequences(val\_sequences, maxlen=max\_length), val\_scores))

**Контрольный тест**

# Preprocess the input texts

input\_texts = [

"Очень хорошо прописаны характеры. Занусси словно под микроскопом разглядывает человека, препарирует ситуацию, разбирает случай на детали, устраивает качели игр разума. Жанр не мой, но досмотрела — захватывает.",

"на мой взгляд самая неудачная книга у этого автора..."

]

input\_sequences = tokenizer.texts\_to\_sequences(input\_texts)

padded\_sequences = pad\_sequences(input\_sequences, maxlen=max\_length)

# Make predictions

predictions = model.predict(padded\_sequences)

# Convert predictions to scores

predicted\_scores = predictions.flatten().tolist()

print(predicted\_scores)

Результат

[6.444942951202393, 4.808480262756348]

**Вывод**

Этот код предварительно обработает входные тексты с помощью токенизатора и функции pad\_sequences, затем передаст предварительно обработанные последовательности модели для предсказания. Предсказанные оценки будут преобразованы в список и выведены в результат. Обученная нейронная сеть достаточно хорошо обработала тексты и предсказала скор