федеральное государственное бюджетное образовательное учреждение

высшего образования «Вятский государственный университет»

ПРОГРАММА ДЛЯ ЭВМ

**Программа для прогнозирования с использованием обработки естественного языка**

Фрагменты программы для ЭВМ

Листов 3

Авторы:

Жаркова Любовь Николаевна

Киров

2023 г.

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.layers import Input, LSTM, Dense

from tensorflow.keras.models import Model

import numpy as np

import pandas as pd

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

from tensorflow.keras.preprocessing.text import Tokenizer

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

# предобработка данных

text\_data = pd.read\_csv('your\_dataset.csv')

text\_data = text\_data.astype(str)

text\_data = text\_data.apply(lambda x: x.astype(str).str.lower())

# определение максимальной длины входной и выходной последовательностей

max\_text\_length = 100

max\_summary\_length = 10

# разделение данных на обучающий и тестовый наборы

x\_train, x\_test, y\_train, y\_test = train\_test\_split(text\_data['text'], text\_data['summary'], test\_size=0.2, random\_state=0)

# токенизация текста и суммаризации

x\_tokenizer = Tokenizer()

x\_tokenizer.fit\_on\_texts(list(x\_train))

# определение размера словаря для текста

x\_vocab\_size = len(x\_tokenizer.word\_index) + 1

# преобразование текста и суммаризации в последовательности

x\_train\_sequence = x\_tokenizer.texts\_to\_sequences(x\_train)

x\_test\_sequence = x\_tokenizer.texts\_to\_sequences(x\_test)

# токенизация суммаризации

y\_tokenizer = Tokenizer()

y\_tokenizer.fit\_on\_texts(list(y\_train))

y\_vocab\_size = len(y\_tokenizer.word\_index) + 1

y\_train\_sequence = y\_tokenizer.texts\_to\_sequences(y\_train)

y\_test\_sequence = y\_tokenizer.texts\_to\_sequences(y\_test)

y\_train\_padded = pad\_sequences(y\_train\_sequence,  maxlen=max\_summary\_length, padding='post')

y\_test\_padded = pad\_sequences(y\_test\_sequence, maxlen=max\_summary\_length, padding='post')

# Модель seq2seq

latent\_dim = 300

# Encoder

encoder\_input = Input(shape=(max\_text\_length,))

#  LSTM слой в кодера

encoder = LSTM(latent\_dim, return\_sequences=True, return\_state=True)

encoder\_output, state\_h, state\_c = encoder(encoder\_input)

encoder\_states = [state\_h, state\_c]

# Decoder

decoder\_input = Input(shape=(max\_summary\_length,))

decoder = LSTM(latent\_dim, return\_sequences=True, return\_state=True)

decoder\_output, \_, \_ = decoder(decoder\_input, initial\_state=encoder\_states)

# Полносвязанный слой для прогнозирования слов

decoder\_dense = Dense(y\_vocab\_size, activation='softmax')

decoder\_output = decoder\_dense(decoder\_output)

# Компиляция и обучение модели

model.compile(optimizer='rmsprop', loss='sparse\_categorical\_crossentropy')

model.fit([x\_train\_padded, y\_train\_padded[:,:-1]], y\_train\_padded.reshape(y\_train\_padded.shape[0],y\_train\_padded.shape[1], 1)[:,1:] , epochs=50, batch\_size=64, validation\_data=([x\_test\_padded,y\_test\_padded[:,:-1]], y\_test\_padded.reshape(y\_test\_padded.shape[0],y\_test\_padded.shape[1], 1)[:,1:]))