next-word-prediction-task-8

October 18, 2023

1 Next Word Prediction using NLTK

Basic Setup

[ ]:
```
!pip install nltk
# download nltk corpus (first time only)
import nltk
from nltk.corpus import reuters
from nltk import bigrams, ConditionalFreqDist
from nltk.tokenize import word_tokenize, sent_tokenize
nltk.download('all')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages
(3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
(from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from nltk) (4.66.1)
```

```
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package alpino to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/alpino.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger_ru to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/averaged_perceptron_tagger_ru.zip.
[nltk_data]    | Downloading package basque_grammars to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping grammars/basque_grammars.zip.
```

```
[nltk_data]    | Downloading package bcp47 to /root/nltk_data…
[nltk_data]    | Downloading package biocreative_ppi to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/biocreative_ppi.zip.
[nltk_data]    | Downloading package bllip_wsj_no_aux to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping models/bllip_wsj_no_aux.zip.
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping grammars/book_grammars.zip.
[nltk_data]    | Downloading package brown to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/brown.zip.
[nltk_data]    | Downloading package brown_tei to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/brown_tei.zip.
[nltk_data]    | Downloading package cess_cat to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/cess_cat.zip.
[nltk_data]    | Downloading package cess_esp to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/cess_esp.zip.
[nltk_data]    | Downloading package chat80 to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/chat80.zip.
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/city_database.zip.
[nltk_data]    | Downloading package cmudict to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/cmudict.zip.
[nltk_data]    | Downloading package comparative_sentences to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/comparative_sentences.zip.
[nltk_data]    | Downloading package comtrans to /root/nltk_data…
[nltk_data]    | Downloading package conll2000 to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/conll2000.zip.
[nltk_data]    | Downloading package conll2002 to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/conll2002.zip.
[nltk_data]    | Downloading package conll2007 to /root/nltk_data…
[nltk_data]    | Downloading package crubadan to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/crubadan.zip.
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/dependency_treebank.zip.
[nltk_data]    | Downloading package dolch to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/dolch.zip.
[nltk_data]    | Downloading package europarl_raw to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/europarl_raw.zip.
[nltk_data]    | Downloading package extended_omw to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    | Downloading package floresta to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/floresta.zip.
```

```
[nltk_data]    | Downloading package framenet_v15 to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/framenet_v15.zip.
[nltk_data]    | Downloading package framenet_v17 to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/framenet_v17.zip.
[nltk_data]    | Downloading package gazetteers to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/gazetteers.zip.
[nltk_data]    | Downloading package genesis to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/genesis.zip.
[nltk_data]    | Downloading package gutenberg to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/gutenberg.zip.
[nltk_data]    | Downloading package ieer to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/ieer.zip.
[nltk_data]    | Downloading package inaugural to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/inaugural.zip.
[nltk_data]    | Downloading package indian to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/indian.zip.
[nltk_data]    | Downloading package jeita to /root/nltk_data…
[nltk_data]    | Downloading package kimmo to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/kimmo.zip.
[nltk_data]    | Downloading package knbc to /root/nltk_data…
[nltk_data]    | Downloading package large_grammars to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping grammars/large_grammars.zip.
[nltk_data]    | Downloading package lin_thesaurus to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/lin_thesaurus.zip.
[nltk_data]    | Downloading package mac_morpho to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/mac_morpho.zip.
[nltk_data]    | Downloading package machado to /root/nltk_data…
[nltk_data]    | Downloading package masc_tagged to /root/nltk_data…
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data]    | Downloading package maxent_treebank_pos_tagger to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping taggers/maxent_treebank_pos_tagger.zip.
[nltk_data]    | Downloading package moses_sample to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping models/moses_sample.zip.
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/movie_reviews.zip.
[nltk_data]    | Downloading package mte_teip5 to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/mte_teip5.zip.
[nltk_data]    | Downloading package mwa_ppdb to /root/nltk_data…
[nltk_data]    |   Unzipping misc/mwa_ppdb.zip.
```

```
[nltk_data]    | Downloading package names to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/names.zip.
[nltk_data]    | Downloading package nombank.1.0 to /root/nltk_data…
[nltk_data]    | Downloading package nonbreaking_prefixes to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/nonbreaking_prefixes.zip.
[nltk_data]    | Downloading package nps_chat to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/nps_chat.zip.
[nltk_data]    | Downloading package omw to /root/nltk_data…
[nltk_data]    | Downloading package omw-1.4 to /root/nltk_data…
[nltk_data]    | Downloading package opinion_lexicon to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/opinion_lexicon.zip.
[nltk_data]    | Downloading package panlex_swadesh to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    | Downloading package paradigms to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/paradigms.zip.
[nltk_data]    | Downloading package pe08 to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/pe08.zip.
[nltk_data]    | Downloading package perluniprops to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping misc/perluniprops.zip.
[nltk_data]    | Downloading package pil to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/pil.zip.
[nltk_data]    | Downloading package pl196x to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/pl196x.zip.
[nltk_data]    | Downloading package porter_test to /root/nltk_data…
[nltk_data]    |   Unzipping stemmers/porter_test.zip.
[nltk_data]    | Downloading package ppattach to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/ppattach.zip.
[nltk_data]    | Downloading package problem_reports to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/problem_reports.zip.
[nltk_data]    | Downloading package product_reviews_1 to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/product_reviews_1.zip.
[nltk_data]    | Downloading package product_reviews_2 to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/product_reviews_2.zip.
[nltk_data]    | Downloading package propbank to /root/nltk_data…
[nltk_data]    | Downloading package pros_cons to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/pros_cons.zip.
[nltk_data]    | Downloading package ptb to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/ptb.zip.
[nltk_data]    | Downloading package punkt to /root/nltk_data…
[nltk_data]    |   Unzipping tokenizers/punkt.zip.
[nltk_data]    | Downloading package qc to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/qc.zip.
```

```
[nltk_data]    | Downloading package reuters to /root/nltk_data…
[nltk_data]    | Downloading package rslp to /root/nltk_data…
[nltk_data]    |   Unzipping stemmers/rslp.zip.
[nltk_data]    | Downloading package rte to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/rte.zip.
[nltk_data]    | Downloading package sample_grammars to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping grammars/sample_grammars.zip.
[nltk_data]    | Downloading package semcor to /root/nltk_data…
[nltk_data]    | Downloading package senseval to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/senseval.zip.
[nltk_data]    | Downloading package sentence_polarity to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/sentence_polarity.zip.
[nltk_data]    | Downloading package sentiwordnet to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/sentiwordnet.zip.
[nltk_data]    | Downloading package shakespeare to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/shakespeare.zip.
[nltk_data]    | Downloading package sinica_treebank to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/sinica_treebank.zip.
[nltk_data]    | Downloading package smultron to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/smultron.zip.
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    | Downloading package spanish_grammars to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping grammars/spanish_grammars.zip.
[nltk_data]    | Downloading package state_union to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/state_union.zip.
[nltk_data]    | Downloading package stopwords to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/stopwords.zip.
[nltk_data]    | Downloading package subjectivity to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/subjectivity.zip.
[nltk_data]    | Downloading package swadesh to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/swadesh.zip.
[nltk_data]    | Downloading package switchboard to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/switchboard.zip.
[nltk_data]    | Downloading package tagsets to /root/nltk_data…
[nltk_data]    |   Unzipping help/tagsets.zip.
[nltk_data]    | Downloading package timit to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/timit.zip.
[nltk_data]    | Downloading package toolbox to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/toolbox.zip.
[nltk_data]    | Downloading package treebank to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/treebank.zip.
```

```
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/twitter_samples.zip.
[nltk_data]    | Downloading package udhr to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/udhr.zip.
[nltk_data]    | Downloading package udhr2 to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/udhr2.zip.
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping corpora/unicode_samples.zip.
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping taggers/universal_tagset.zip.
[nltk_data]    | Downloading package universal_treebanks_v20 to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    | Downloading package vader_lexicon to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    | Downloading package verbnet to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/verbnet.zip.
[nltk_data]    | Downloading package verbnet3 to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/verbnet3.zip.
[nltk_data]    | Downloading package webtext to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/webtext.zip.
[nltk_data]    | Downloading package wmt15_eval to /root/nltk_data…
[nltk_data]    |   Unzipping models/wmt15_eval.zip.
[nltk_data]    | Downloading package word2vec_sample to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Unzipping models/word2vec_sample.zip.
[nltk_data]    | Downloading package wordnet to /root/nltk_data…
[nltk_data]    | Downloading package wordnet2021 to /root/nltk_data…
[nltk_data]    | Downloading package wordnet2022 to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/wordnet2022.zip.
[nltk_data]    | Downloading package wordnet31 to /root/nltk_data…
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/wordnet_ic.zip.
[nltk_data]    | Downloading package words to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/words.zip.
[nltk_data]    | Downloading package ycoe to /root/nltk_data…
[nltk_data]    |   Unzipping corpora/ycoe.zip.
[nltk_data]    |
[nltk_data]  Done downloading collection all
```

[ ]:
```
True
```

Loading the dataset

[ ]:
```
nltk.download("reuters")
corpus = reuters.sents()
```

```
[nltk_data] Downloading package reuters to /root/nltk_data…
[nltk_data]   Package reuters is already up-to-date!
```

[ ]:
```
print(corpus[1])
```

```
['They', 'told', 'Reuter', 'correspondents', 'in', 'Asian', 'capitals', 'a',
'U', '.', 'S', '.', 'Move', 'against', 'Japan', 'might', 'boost',
'protectionist', 'sentiment', 'in', 'the', 'U', '.', 'S', '.', 'And', 'lead',
'to', 'curbs', 'on', 'American', 'imports', 'of', 'their', 'products', '.']
```

[ ]:
```
len(corpus)
```

[ ]: 54716

Creating Bigrams **Sentence**: This is a Data Science Course **UniGram**: * This * is * a * Data * Science * Course

BiGram: * This is * is a * a Data * Data Science * Science Course

TriGram: * This is a * is a Data * a Data Science * Data Science Course

[ ]:
```
words = [word.lower() for s in corpus for word in s]
bigrams_list = list(bigrams(words))
```

[ ]:
```
print(bigrams_list[:10])
```

```
[('asian', 'exporters'), ('exporters', 'fear'), ('fear', 'damage'), ('damage',
'from'), ('from', 'u'), ('u', '.'), ('.', 's'), ('s', '.-'), ('.-', 'japan'),
('japan', 'rift')]
```

Creating Conditional Frequency Distribution

[ ]:
```
cfd = ConditionalFreqDist(bigrams_list)
```

[ ]:
```
cfd['the']
```

[ ]:
```
FreqDist({'company': 3126, 'u': 2264, 'dollar': 984, 'bank': 960, 'first': 839,
'government': 787, 'year': 720, 'united': 682, 'new': 678, 'market': 590, …})
```

Predicting Next Word

[ ]:
```
def predict_next_word(input_word):
    input_word = input_word.lower()
    if input_word in cfd:
        return cfd[input_word].max()
    else:
        return "Word not found in corpus"
```

[ ]:
```
input_word = "the"
next_word = predict_next_word(input_word)
print(f"The next word after '{input_word}' could be: {next_word}")
```

```
The next word after 'the' could be: company
```

2 Next Word Prediction using RNNs

Before we start: The Unreasonable Effectiveness of Recurrent Neural Networks

An LSTM Cell How a Neuron remembers the relevant past An LSTM cell is like a tiny memory unit inside a computer program. It has three main parts:

- Input Gate: Think of this as a gatekeeper. It decides which new information to let in from the current input and whether to remember it or not.
- Forget Gate: This part helps the LSTM cell decide what information to forget from its previous memory. It's like cleaning out unnecessary stuff to make room for new things.
- Output Gate: This gate decides what information the LSTM cell should pass on to the next step in the sequence. It's like the LSTM cell deciding what to say or remember.

The cell has two states Cell State and Hidden State. They are continuously updated and carry the information from the previous to the current time steps.

The cell state is the "long-term" memory, while the hidden state is the "short-term" memory.

The forget gate and input gate update the cell state. The hidden state is computed using the output gate.

[ ]:
```
import numpy as np

def sigmoid(x):
  return 1 / (1 + np.exp(-x))

def lstm_cell_explained(prev_c, prev_h, x_t, params):
    """
    LSTM cell implementation.

    Args:
    prev_c: Previous cell state (numpy array of shape (hidden_size,))
    prev_h: Previous hidden state (numpy array of shape (hidden_size,))
    x_t: Input at time step t (numpy array of shape (input_size,))
    params: Dictionary containing LSTM parameters - {'W_f', 'U_f', 'b_f',␣
 ↪'W_i', 'U_i', 'b_i', 'W_c', 'U_c', 'b_c', 'W_o', 'U_o', 'b_o'}

    Returns:
    next_c: Next cell state (numpy array of shape (hidden_size,))
    next_h: Next hidden state (numpy array of shape (hidden_size,))
    """
```

```
    W_f, U_f, b_f = params['W_f'], params['U_f'], params['b_f']
    W_i, U_i, b_i = params['W_i'], params['U_i'], params['b_i']
    W_c, U_c, b_c = params['W_c'], params['U_c'], params['b_c']
    W_o, U_o, b_o = params['W_o'], params['U_o'], params['b_o']

    # Input gate
    i_t = sigmoid(np.dot(prev_h, U_i) + np.dot(x_t, W_i) + b_i)

    # Forget gate
    f_t = sigmoid(np.dot(prev_h, U_f) + np.dot(x_t, W_f) + b_f)

    # Output gate
    o_t = sigmoid(np.dot(prev_h, U_o) + np.dot(x_t, W_o) + b_o)

    # Cell state
    c_t = np.tanh(np.dot(prev_h, U_c) + np.dot(x_t, W_c) + b_c)

    # Update cell state
    next_c = f_t * prev_c + i_t * c_t

    # Update hidden state
    next_h = o_t * np.tanh(next_c)

    return next_c, next_h
```

Basic Setup

[ ]:
```
import tensorflow as tf
import numpy as np
import random
import sys
import os
```

Loading the dataset

[ ]:
```
# Load and preprocess the Shakespeare text
path_to_file = tf.keras.utils.get_file('shakespeare.txt', 'https://storage.
  ↪googleapis.com/download.tensorflow.org/data/shakespeare.txt')

text = open(path_to_file, 'rb').read().decode(encoding='utf-8')
```

```
Downloading data from
https://storage.googleapis.com/download.tensorflow.org/data/shakespeare.txt
1115394/1115394 [==============================] - 0s 0us/step
```

Creating mappings

[ ]:
```
# Create a vocabulary
vocab = sorted(set(text))

# Create a mapping from characters to unique indices
char2idx = {char: idx for idx, char in enumerate(vocab)}
idx2char = np.array(vocab)

# Convert the text to numerical data
text_as_int = np.array([char2idx[char] for char in text])
```

[ ]:
```
print(vocab)
print(len(vocab))
```

```
['\n', ' ', '!', '$', '&', "'", ',', '-', '.', '3', ':', ';', '?', 'A', 'B',
'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R',
'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',
'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
'y', 'z']
65
```

[ ]:
```
print(char2idx)
```

```
{'\n': 0, ' ': 1, '!': 2, '$': 3, '&': 4, "'": 5, ',': 6, '-': 7, '.': 8, '3':
9, ':': 10, ';': 11, '?': 12, 'A': 13, 'B': 14, 'C': 15, 'D': 16, 'E': 17, 'F':
18, 'G': 19, 'H': 20, 'I': 21, 'J': 22, 'K': 23, 'L': 24, 'M': 25, 'N': 26, 'O':
27, 'P': 28, 'Q': 29, 'R': 30, 'S': 31, 'T': 32, 'U': 33, 'V': 34, 'W': 35, 'X':
36, 'Y': 37, 'Z': 38, 'a': 39, 'b': 40, 'c': 41, 'd': 42, 'e': 43, 'f': 44, 'g':
45, 'h': 46, 'i': 47, 'j': 48, 'k': 49, 'l': 50, 'm': 51, 'n': 52, 'o': 53, 'p':
54, 'q': 55, 'r': 56, 's': 57, 't': 58, 'u': 59, 'v': 60, 'w': 61, 'x': 62, 'y':
63, 'z': 64}
```

[ ]:
```
print(idx2char)
```

```
['\n' ' ' '!' '$' '&' "'" ',' '-' '.' '3' ':' ';' '?' 'A' 'B' 'C' 'D' 'E'
 'F' 'G' 'H' 'I' 'J' 'K' 'L' 'M' 'N' 'O' 'P' 'Q' 'R' 'S' 'T' 'U' 'V' 'W'
 'X' 'Y' 'Z' 'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o'
 'p' 'q' 'r' 's' 't' 'u' 'v' 'w' 'x' 'y' 'z']
```

[ ]:
```
print(text_as_int[0:200])
```

```
[18 47 56 57 58  1 15 47 58 47 64 43 52 10  0 14 43 44 53 56 43  1 61 43
  1 54 56 53 41 43 43 42  1 39 52 63  1 44 59 56 58 46 43 56  6  1 46 43
 39 56  1 51 43  1 57 54 43 39 49  8  0  0 13 50 50 10  0 31 54 43 39 49
  6  1 57 54 43 39 49  8  0  0 18 47 56 57 58  1 15 47 58 47 64 43 52 10
  0 37 53 59  1 39 56 43  1 39 50 50  1 56 43 57 53 50 60 43 42  1 56 39
 58 46 43 56  1 58 53  1 42 47 43  1 58 46 39 52  1 58 53  1 44 39 51 47
 57 46 12  0  0 13 50 50 10  0 30 43 57 53 50 60 43 42  8  1 56 43 57 53
 50 60 43 42  8  0  0 18 47 56 57 58  1 15 47 58 47 64 43 52 10  0 18 47
```

```
 56 57 58  6  1 63 53 59]
```

Creating Training Batches

[ ]:
```python
# Create training examples and targets
sequence_length = 100
sequences_per_epoch = len(text) // (sequence_length + 1)
print(sequences_per_epoch)

char_dataset = tf.data.Dataset.from_tensor_slices(text_as_int)
# Printing tensor slices
counter = 0
print("Individual characters converted to Tensors")
for chars in char_dataset:
  counter = counter + 1
  print(chars)
  if counter == 10:
    break

# Printing character batches
sequences = char_dataset.batch(sequence_length + 1, drop_remainder=True)

counter = 0
for seqs in sequences:
  counter = counter + 1
  print(seqs)
  if counter == 5:
    break

def split_input_target(chunk):
    input_text = chunk[:-1]
    print(input_text)
    target_text = chunk[1:]
    return input_text, target_text

dataset = sequences.map(split_input_target)
```

```
11043
Individual characters converted to Tensors
tf.Tensor(18, shape=(), dtype=int64)
tf.Tensor(47, shape=(), dtype=int64)
tf.Tensor(56, shape=(), dtype=int64)
tf.Tensor(57, shape=(), dtype=int64)
tf.Tensor(58, shape=(), dtype=int64)
tf.Tensor(1, shape=(), dtype=int64)
tf.Tensor(15, shape=(), dtype=int64)
tf.Tensor(47, shape=(), dtype=int64)
tf.Tensor(58, shape=(), dtype=int64)
```

```
tf.Tensor(47, shape=(), dtype=int64)
tf.Tensor(
[18 47 56 57 58  1 15 47 58 47 64 43 52 10  0 14 43 44 53 56 43  1 61 43
  1 54 56 53 41 43 43 42  1 39 52 63  1 44 59 56 58 46 43 56  6  1 46 43
 39 56  1 51 43  1 57 54 43 39 49  8  0  0 13 50 50 10  0 31 54 43 39 49
  6  1 57 54 43 39 49  8  0  0 18 47 56 57 58  1 15 47 58 47 64 43 52 10
  0 37 53 59  1], shape=(101,), dtype=int64)
tf.Tensor(
[39 56 43  1 39 50 50  1 56 43 57 53 50 60 43 42  1 56 39 58 46 43 56  1
 58 53  1 42 47 43  1 58 46 39 52  1 58 53  1 44 39 51 47 57 46 12  0  0
 13 50 50 10  0 30 43 57 53 50 60 43 42  8  1 56 43 57 53 50 60 43 42  8
  0  0 18 47 56 57 58  1 15 47 58 47 64 43 52 10  0 18 47 56 57 58  6  1
 63 53 59  1 49], shape=(101,), dtype=int64)
tf.Tensor(
[52 53 61  1 15 39 47 59 57  1 25 39 56 41 47 59 57  1 47 57  1 41 46 47
 43 44  1 43 52 43 51 63  1 58 53  1 58 46 43  1 54 43 53 54 50 43  8  0
  0 13 50 50 10  0 35 43  1 49 52 53 61  5 58  6  1 61 43  1 49 52 53 61
  5 58  8  0  0 18 47 56 57 58  1 15 47 58 47 64 43 52 10  0 24 43 58  1
 59 57  1 49 47], shape=(101,), dtype=int64)
tf.Tensor(
[50 50  1 46 47 51  6  1 39 52 42  1 61 43  5 50 50  1 46 39 60 43  1 41
 53 56 52  1 39 58  1 53 59 56  1 53 61 52  1 54 56 47 41 43  8  0 21 57
  5 58  1 39  1 60 43 56 42 47 41 58 12  0  0 13 50 50 10  0 26 53  1 51
 53 56 43  1 58 39 50 49 47 52 45  1 53 52  5 58 11  1 50 43 58  1 47 58
  1 40 43  1 42], shape=(101,), dtype=int64)
tf.Tensor(
[53 52 43 10  1 39 61 39 63  6  1 39 61 39 63  2  0  0 31 43 41 53 52 42
  1 15 47 58 47 64 43 52 10  0 27 52 43  1 61 53 56 42  6  1 45 53 53 42
  1 41 47 58 47 64 43 52 57  8  0  0 18 47 56 57 58  1 15 47 58 47 64 43
 52 10  0 35 43  1 39 56 43  1 39 41 41 53 59 52 58 43 42  1 54 53 53 56
  1 41 47 58 47], shape=(101,), dtype=int64)
Tensor("strided_slice:0", shape=(100,), dtype=int64)
```

Training Parameters

[ ]:
```
# Batch size
BATCH_SIZE = 64

# Buffer size to shuffle the dataset
BUFFER_SIZE = 10000

dataset = dataset.shuffle(BUFFER_SIZE).batch(BATCH_SIZE, drop_remainder=True)

vocab_size = len(vocab)
embedding_dim = 256
rnn_units = 1024

EPOCHS = 30
```

Creating the Model

[ ]:
```
def build_model(vocab_size, embedding_dim, rnn_units, batch_size):
    model = tf.keras.Sequential([
        tf.keras.layers.Embedding(vocab_size, embedding_dim,␣
 ↪batch_input_shape=[batch_size, None]),
        tf.keras.layers.LSTM(rnn_units, return_sequences=True, stateful=True,␣
 ↪recurrent_initializer='glorot_uniform'),
        tf.keras.layers.Dense(vocab_size)
    ])
    return model

model = build_model(vocab_size, embedding_dim, rnn_units, BATCH_SIZE)

# Compile the model
model.compile(optimizer='adam', loss=tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True))

# Configure checkpoints
checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt_{epoch}")

checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_prefix,
    save_weights_only=True)
```

[ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (64, None, 256)           16640

 lstm (LSTM)                 (64, None, 1024)          5246976

 dense (Dense)               (64, None, 65)            66625

=================================================================
Total params: 5330241 (20.33 MB)
Trainable params: 5330241 (20.33 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Training the Model

[ ]:
```
history = model.fit(dataset, epochs=EPOCHS, callbacks=[checkpoint_callback])
```

```
Epoch 1/30
```

```
172/172 [==============================] - 27s 91ms/step - loss: 2.5851
Epoch 2/30
172/172 [==============================] - 15s 72ms/step - loss: 1.8703
Epoch 3/30
172/172 [==============================] - 15s 74ms/step - loss: 1.6241
Epoch 4/30
172/172 [==============================] - 16s 74ms/step - loss: 1.4922
Epoch 5/30
172/172 [==============================] - 15s 76ms/step - loss: 1.4149
Epoch 6/30
172/172 [==============================] - 15s 77ms/step - loss: 1.3591
Epoch 7/30
172/172 [==============================] - 15s 78ms/step - loss: 1.3159
Epoch 8/30
172/172 [==============================] - 15s 78ms/step - loss: 1.2766
Epoch 9/30
172/172 [==============================] - 16s 76ms/step - loss: 1.2402
Epoch 10/30
172/172 [==============================] - 15s 76ms/step - loss: 1.2049
Epoch 11/30
172/172 [==============================] - 15s 77ms/step - loss: 1.1697
Epoch 12/30
172/172 [==============================] - 15s 77ms/step - loss: 1.1324
Epoch 13/30
172/172 [==============================] - 16s 76ms/step - loss: 1.0946
Epoch 14/30
172/172 [==============================] - 15s 77ms/step - loss: 1.0554
Epoch 15/30
172/172 [==============================] - 15s 77ms/step - loss: 1.0144
Epoch 16/30
172/172 [==============================] - 16s 76ms/step - loss: 0.9737
Epoch 17/30
172/172 [==============================] - 15s 79ms/step - loss: 0.9325
Epoch 18/30
172/172 [==============================] - 15s 78ms/step - loss: 0.8911
Epoch 19/30
172/172 [==============================] - 16s 78ms/step - loss: 0.8506
Epoch 20/30
172/172 [==============================] - 15s 76ms/step - loss: 0.8145
Epoch 21/30
172/172 [==============================] - 15s 76ms/step - loss: 0.7773
Epoch 22/30
172/172 [==============================] - 15s 77ms/step - loss: 0.7431
Epoch 23/30
172/172 [==============================] - 15s 78ms/step - loss: 0.7135
Epoch 24/30
172/172 [==============================] - 15s 77ms/step - loss: 0.6831
Epoch 25/30
```

```
172/172 [==============================] - 16s 79ms/step - loss: 0.6584
Epoch 26/30
172/172 [==============================] - 16s 77ms/step - loss: 0.6331
Epoch 27/30
172/172 [==============================] - 15s 77ms/step - loss: 0.6132
Epoch 28/30
172/172 [==============================] - 15s 77ms/step - loss: 0.5937
Epoch 29/30
172/172 [==============================] - 15s 77ms/step - loss: 0.5747
Epoch 30/30
172/172 [==============================] - 16s 77ms/step - loss: 0.5590
```

Predicting with the Model

[ ]:
```python
# Generate text
def generate_text(model, start_string):
    num_generate = 1000
    input_eval = [char2idx[s] for s in start_string]
    input_eval = tf.expand_dims(input_eval, 0)
    text_generated = []
    temperature = 1.0

    model.reset_states()
    for i in range(num_generate):
        predictions = model(input_eval)
        predictions = tf.squeeze(predictions, 0)
        predictions = predictions / temperature
        predicted_id = tf.random.categorical(predictions, num_samples=1)[-1, 0].
 ↪numpy()
        input_eval = tf.expand_dims([predicted_id], 0)
        text_generated.append(idx2char[predicted_id])

    return (start_string + ''.join(text_generated))

# Restore the latest checkpoint and generate text
model = build_model(vocab_size, embedding_dim, rnn_units, batch_size=1)
model.load_weights(tf.train.latest_checkpoint(checkpoint_dir))
model.build(tf.TensorShape([1, None]))

print(generate_text(model, start_string=u"ROMEO: "))
```

```
ROMEO: Be merrol,
Spit in her brethren Richmond and Petruchio.

CAMILLO:
I thank you, lads. Pover Gloucester's dead!

QUEEN ELIZABETH:
```

```
Harpon, amisad, fetch I look pale,
Which we heard here a dream to princely name,
In earth against the traff my meditation,
Ant of what thou hast spoken worse,
The fresh sinks papuled in despite of me.

KING RICHARD II:
Well, I just company, I think IVER:
What else? be but true, he deserved, your son,
Go together with this earth and credutish
det redemption! while the old weeds,
The fatal blood is sit in them,
By her force and victory.
But my heart prepare for Talk'd?

GLOUCESTER:
What, more than I, or'th, I will dry
you must return before your grace to pluch them all,
That bear the shadow dry our traded head to the wind;
Who, or thy warlike state, my Lord Noble Marcius, that will say From your
country's father.

BIANCA:
Why, since the king shall be contented
Upon the people: is't most rich in remedy and am I servant;
Or sweetle yours, ladies, Lancaster.

GLOUC
```

3 Next Word Prediction using Transformers

[ ]:
```
!pip install transformers torch
```

```
Collecting transformers
  Downloading transformers-4.33.2-py3-none-any.whl (7.6 MB)
                             7.6/7.6 MB
38.9 MB/s eta 0:00:00
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-
packages (2.0.1+cu118)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from transformers) (3.12.2)
Collecting huggingface-hub<1.0,>=0.15.1 (from transformers)
  Downloading huggingface_hub-0.17.1-py3-none-any.whl (294 kB)
                             294.8/294.8 kB
25.2 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (1.23.5)
```

```
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2023.6.3)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from transformers) (2.31.0)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from transformers)
  Downloading
tokenizers-0.13.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(7.8 MB)
                           7.8/7.8 MB
106.9 MB/s eta 0:00:00
Collecting safetensors>=0.3.1 (from transformers)
  Downloading
safetensors-0.3.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(1.3 MB)
                           1.3/1.3 MB
82.2 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.27 in
/usr/local/lib/python3.10/dist-packages (from transformers) (4.66.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from torch) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages
(from torch) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-
packages (from torch) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages
(from triton==2.0.0->torch) (3.27.4.1)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages
(from triton==2.0.0->torch) (16.0.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from huggingface-hub<1.0,>=0.15.1->transformers) (2023.6.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers)
```

```
(2023.7.22)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-
packages (from sympy->torch) (1.3.0)
Installing collected packages: tokenizers, safetensors, huggingface-hub,
transformers
Successfully installed huggingface-hub-0.17.1 safetensors-0.3.3
tokenizers-0.13.3 transformers-4.33.2
```

[ ]:
```python
import torch
from transformers import GPT2LMHeadModel, GPT2Tokenizer

# Load pre-trained GPT-2 model and tokenizer
model_name = "gpt2"  # You can also try "gpt2-medium", "gpt2-large", or␣
 ↪"gpt2-xl" for larger models
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
model = GPT2LMHeadModel.from_pretrained(model_name)

# Set the model to evaluation mode (no training)
model.eval()

# Function to generate text
def generate_text(prompt, max_length=50, temperature=0.7):
    input_ids = tokenizer.encode(prompt, return_tensors="pt")

    # Generate text
    output = model.generate(
        input_ids,
        max_length=max_length,
        num_return_sequences=1,
        no_repeat_ngram_size=2,
        top_k=50,
        top_p=0.95,
        temperature=temperature,
    )

    # Decode and return generated text
    generated_text = tokenizer.decode(output[0], skip_special_tokens=True)
    return generated_text
```

[ ]:
```python
# Generate text with a prompt
prompt = "Once upon a time"
generated_text = generate_text(prompt, max_length=100)
print(generated_text)
```

```
The attention mask and the pad token id were not set. As a consequence, you may
observe unexpected behavior. Please pass your input's `attention_mask` to obtain
reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
```

Once upon a time, the world was a place of great beauty and great danger. The world of the gods was the place where the great gods were born, and where they were to live.

The world that was created was not the same as the one that is now. It was an endless, endless world. And the Gods were not born of nothing. They were created of a single, single thing. That was why the universe was so beautiful. Because the cosmos was made of two