Projekt 2

Uzupełnianie zdań

Tematem mojego drugiego projektu z Inteligencji Obliczeniowej jest aplikacja ***Uzupełnianie zdania (Sentence Autocomplete)***. Podczas jego tworzenia wyszkoliłam własny model LSTM oraz dotrenowałam model GPT-2. Opracowanie zawiera szczegółowy opis mojego programu oraz treningu.

---

1. Modele przetwarzania tekstu

Dane tekstowe i preprocessing

Do szkolenia swoich modeli wykorzystałam 7 opowiadań Edgara Allana Poe, dostępnych po angielsku dzięki Projektowi Gutenberg. Łącznie z tytułami i liniami odstępu mają one około 1030 długich linii w pliku tekstowym. Poe pisał bardzo poetyckim językiem, wykorzystując nietypowe słowa, co uznałam za ciekawe wyzwanie dla generatora tekstu.

Opowiadania początkowo znajdują się w osobnych plikach, dlatego pierwszym krokiem przygotowania danych jest złączenie ich w jeden plik tekstowy. Następnie tokenizuję tekst, używając WordPunctTokenizera z paczki NLTK, który używa wyrażeń regularnych do rozdrabniania danych. Usuwam znaki interpunkcyjne i nieprzydatne słowa, aby usprawnić nauczanie. Przy przygotowywaniu tokenów zostają stworzone listy dataX i dataY, przechowujące sekwencje wejściowe i wyjściowe modelu, a następnie są one przekształcone na format zrozumiały dla modelu. dataX jest przekształcone na trójwymiarową tablicę i normalizowane prostym min-max, a dataY na one-hot encoding.

Szkolenie GPT-2 nie wymagało żadnego dodatkowego preprocessingu oprócz zamiany pliku tekstowego na dataset, co się dzieje w funkcji `load_dataset`. Dane są też tokenizowane przy użyciu tokenizatora GPT-2.

Trenowanie modeli

Pierwszy model wytrenowałam na rekurencyjnej sieci LSTM. Uznałam ją za dobry wybór, ponieważ Long-Short Term Memory jest szczególnie skuteczna w zadaniach związanych z sekwencjami, w tym predykcją i generowaniem tekstu. Ma zdolność do przechowywania i przypominania sobie informacji przez dłuższy czas. Sieć jest dosyć prosta, zawiera bowiem dwie warstwy LSTM, w tym jedną dwukierunkową. Bidirectional LSTM, która pozwala przetwarzać dane wejściowe w obu kierunkach - od początku i od

końca sekwencji. Pozwala to modelowi lepiej zrozumieć sekwencję. W modelu znajdują się też dwie warstwy Dropout. Warstwa końcowa używa funkcji aktywacji softmax do przekształcania wyników w prawdopodobieństwa. Wytrenowałam sieć na 65 epokach, co zajęło mi kilkanaście godzin.

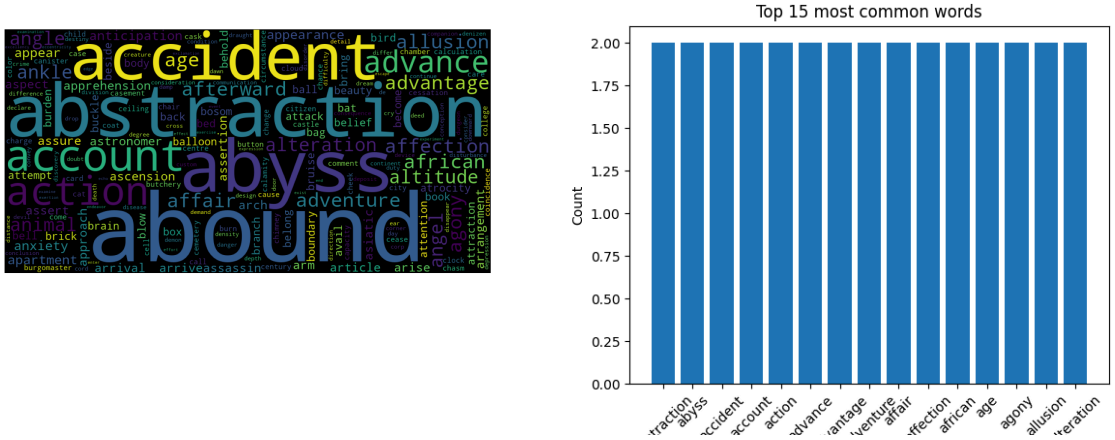
Jako dodatkowy eksperyment, dokonałam fine-tuningu modelu GPT-2. Na 3 epokach wytrenowałam go, aby dostosował się do opowiadań Edgara Allana Poe. GPT-2 bazuje na Transformerach i PyTorchu, więc one posłużyły mi do treningu. Data Collator przygotowuje dane, by mogły być użyte do trenowania modelu. Następnie przeprowadziłam proces treningowy przez zdefiniowanie konfiguracji z argumentami treningowymi i trenerem nadzorującym proces.

Statystyki i wydajność modeli

Model LSTM radził sobie coraz lepiej z każdą epoką; miał coraz niższą wartość straty. Jednak z moich obserwacji wynika, że w 64 epoce model generował najbardziej sensowne zdania. Wcześniej i później szczególnie często powtarzał niektóre słowa i wyrażenia, np. “of the car”. Przy dłuższym treningu możliwe jest, że model radziłby sobie jeszcze lepiej, jednak wyniki są całkiem satysfakcjonujące.

GPT-2 również radzi sobie dobrze, lecz lepiej generowałby sekwencje, gdyby trening składał się z większej liczby epok. 3 epoki jedynie delikatnie dostosowują model do zadania.

Jako dodatek, stworzyłam również statystyki takie jak 15 najczęściej występujących słów i chmura słów. Widać na nich, że Poe w opowiadaniach najczęściej używał słów zaczynających się na A.



zdj. 1 i 2 - chmura słów i 15 najczęściej występujących słów

2. Funkcjonalność aplikacji

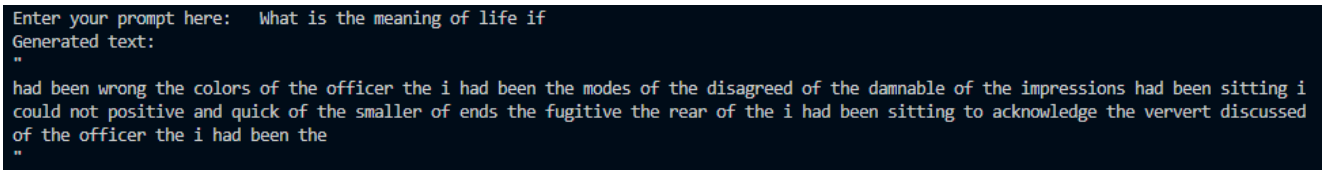
Instrukcja użycia programu (po angielsku) znajduje się w pliku README.md.

Program można uruchomić na cztery sposoby:

1) Bez flagi - wyjaśni, do czego służy program oraz pokaże flagi z ich działaniem.
2) Z flagą -t lub --train - przygotuje dane i wytrenuje model.
3) Z flagą -r lub --run - uruchomi główną aplikację generującą tekst.
4) Z flagą -p lub --plot - stworzy dodatkowe wykresy (patrz: zdj. 1 i 2).

Po uruchomieniu programu użytkownik dostaje wybór - czy chce wytrenować/uruchomić model LSTM, czy GPT-2. Następnie wykonuje zadania specyficzne dla danego modelu.

Sama aplikacja jest minimalistyczna, działająca w terminalu. Została stworzona w celu pokazywania działania generatorów. Oba działają w nieskończonej pętli, w której użytkownik wpisuje swój tekst, a generator dopisuje jego ciąg dalszy. Dodatkowo, aplikacja dla modelu LSTM pozwala na wygenerowanie losowego promptu z opowiadań. Po wpisaniu “exit” jako promptu, aplikacja się zamyka.



zdj. 3 - przykład działania modelu LSTM

Podsumowując, uczenie obu modeli było ciekawym, aczkolwiek długim doświadczeniem. Modele radzą sobie dobrze z generowaniem sekwencji. Edgar Allan Poe pisał nietypowym językiem, co stanowiło wyzwanie przy trenowaniu, jednak zarówno LSTM, jak i GPT-2, podołały mu.

Generowanie tekstu na podstawie dzieł Poego jest bardzo interesującym pomysłem, który można zgłębić i ulepszyć moje modele. Myślę, że dotrenowane na dodatkowych kilkunastu epokach mogłyby generować jeszcze lepsze zdania. Byłoby to jednak bardzo czasochłonne.

Bibliografia:

Project Gutenberg - Edgar Allan Poe: https://www.gutenberg.org/cache/epub/25525/pg25525-images.html
Huggingface - GPT-2: https://huggingface.co/docs/transformers/model_doc/gpt2
Huggingface - Transformers tutorial: https://huggingface.co/docs/transformers/index
Geeks for geeks - What is LSTM?: https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/
Additional help: ChatGPT, GitHub Copilot, presentations from lectures and tasks/tutorials from laboratories