

Generowanie tytułów publikacji naukowych na podstawie modelu GPT2-Simple oraz biblioteki arXiv – raport.

Stanisław Gren, Informatyka rok 4

I. Wstęp

1. Abstrakt

Raport przybliży szczegóły implementacji narzędzia do generowania tytułów publikacji naukowych, na podstawie wstępnie wytrenowanego modelu GPT2-Simple po dostrojeniu go tytułami artykułów naukowych z konkretnej kategorii, oraz zaproponowanego przez w repozytorium „gpt-paper-title-translator”. Projekt prezentuje również przykładowe generacje z tak przygotowanego modeli.

2. Cel pracy

Celem pracy jest stworzenie prostego narzędzia do generowania tytułów publikacji naukowych, poprzez douczenie modelu GPT2.

3. Metodyka

Praca nad projektem opiera się o wykorzystanie istniejącego już rozwiązania zaproponowanego przez użytkownika Chandan Singh, dostępnego w formie licencji MIT na githubie.

Implementacja narzędzia zakłada scaping tytułów publikacji naukowych z bazy arXiv, w poniższym przykładzie wykorzystamy artykuły z kategorii “Electrical Engineering and Systems Science”, na temat procesowania dźwięku i mowy. Wyciągnięte tytuły zostaną wykorzystane do douczenia modelu, aby finalnie wygenerować przykładowe wyniki.

Korzystano również z dokumentacji biblioreki gpt-2-simple, oraz artykułu na temat finetuningu.

II. Część teoretyczna

1. gpt-paper-title-generator

Repozytorium użytkownika Chandan Singh, które prezentuje przykładowe programy generujące tytuły naukowe. Repozytorium prezentuje przykłady z wykorzystaniem kilku różnych modeli. Na jego podstawie został zrealizowany ten projekt.

2. Scraping i baza danych arXiv

Scraping (w j. polskim “skrobanie”) to termin, którym opisuje się zautomatyzowane pobieranie danych, na przykład ze stron internetowych.



Baza danych arXiv jest darmowym, otwartym archiwum zawierającym prawie 2,4 miliona artykułów naukowych z różnych dziedzin. Wyciągając dane z tej bazy możemy określić żadaną dziedzinę oraz zakres dat publikacji artykułów. W tym wypadku wyciągniemy dane z roku 2024.

Do wyciągnięcia danych wykorzystano darmowa bibliotekę arxivscraper. Biblioteka ta umożliwia wybór konkretnej kategorii, okesu oraz podkategorii. Poniżej przedstawiono przykład użycia

```
scraper = ax.Scraper(category='eess', date_from='2024-05-01',
                    date_until='2024-12-03', t=10,
                    filters={'categories':['eess.AS']})

output = scraper.scrape()

titles = [' '.join(o['title'].split()) for o in output]
np.savetxt('titles_ref.csv', np.array(titles), fmt='%s')
```

3. gpt-2-simple

GPT-2-Simple to pakiet w języku Python, który umożliwia korzystanie z istniejących skryptów do dostrajania i generowania modeli tekstowych GPT-2 od OpenAI.

Model GPT2, czyli Generative Pre-training Transformer, jest modelem NLP, opartym na architekturze transformera. Oznacza to, że jego działanie opiera się o przewidywanie kolejnych słów w oparciu o kontekst. GPT2 służy do tworzenia tekstowych odpowiedzi na podstawie własnych zasobów zrozumienia języka.

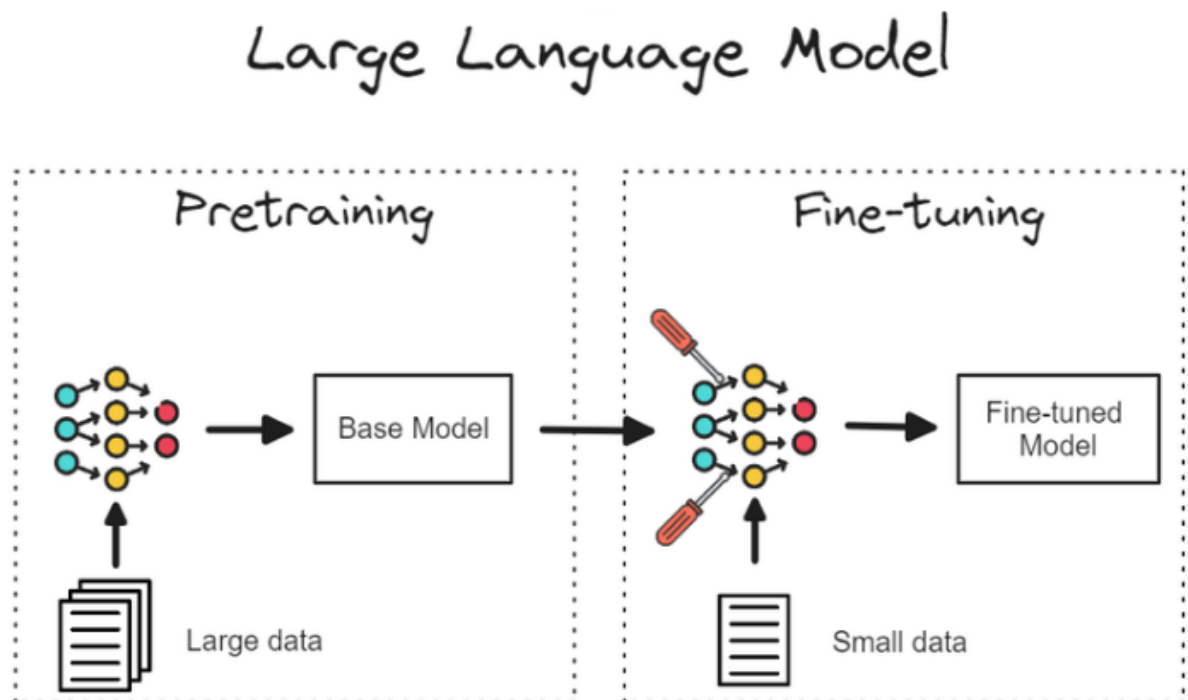
Ułatwia on proces tworzenia tekstów, oferując funkcję zapisywania wyników do pliku, co sprzyja wygodniejszemu zarządzaniu, a także wspiera użycie prefiksów, pozwalających na rozpoczęcie generowanego tekstu od określonej frazy.

Biblioteka dostarcza funkcje do finetuningu (dostrajania modelu) wykorzystując interfejs TensorFlow, o czym informacje możemy znaleźć w repozytorium.

4. Finetuning

W języku polskim oznacza dostrajanie. Proces, który umożliwia zaadaptowanie istniejącego modelu do własnych potrzeb. Wykorzystuje się go podczas użycia wstępnie wytrenowanych modeli (pretrained model), takich jak gpt2-124M.

W finetuningu używa się mniejszych zasobów danych, niż przy trenowaniu modelu od podstaw, które może być bardzo czasochłonne, a efekty niekoniecznie zadowalające, jeśli nie mamy doświadczenia. Finetuning to dobre rozwiązanie jeśli nasz zbiór danych nie jest szczególnie obszernym.



III. Część praktyczna

1. Pobranie modelu oraz instalacja bibliotek

Do programu należało doinstalować konkretne biblioteki, oprócz wymienionych wcześniej w części teoretycznej, wykorzystano również biblioteki `os` i `numpy`, które pomogą zrealizować program. Instalację bibliotek zrealizowano przy użyciu narzędzia `PiPy`.

```
import gpt_2_simple as gpt2
import os
import arxivscraper as ax
import numpy as np
```

Pierwszym krokiem było pobranie odpowiedniego modelu, wybrano model o nazwie "124M", który jak sama nazwa wskazuje, ma 124 miliony parametrów. Jest to najmniejsza wersja modelu GPT2, natomiast do celów testowych w obrębie tego projektu, jest wystarczająca. Poniżej zamieszczono kod realizujący to zadanie.

```
model_name = "124M"
if not os.path.isdir(os.path.join("models", model_name)):
    print(f"Downloading {model_name} model...")
    gpt2.download_gpt2(model_name=model_name)
else:
    print("Model exist. Scraping achrives...")
```

```
Downloading 124M model...
Fetching checkpoint: 1.05Mit [00:00, ?it/s]
Fetching encoder.json: 1.05Mit [00:01, 898kit/s]
Fetching hparams.json: 1.05Mit [00:00, ?it/s]
Fetching model.ckpt.data-00000-of-00001: 25%|██████| 127M/498M [00:10<00:23, 15.8Mit/s]
```

Kod napisano w taki sposób aby całość odbywała się automatycznie, to znaczy przy pierwszym uruchomieniu, program sprawdza czy model został już pobrany, aby nie nadpisywać go przy kolejnych uruchomieniach.

2. Scraping danych

Realizację projektu rozpoczęto od wyciągnięcia danych, na podstawie których douczymy model, oraz sformatowaniu ich w odpowiedni plik. Poniżej przedstawiono fragment kodu który jest za to odpowiedzialny.

```
scraper = ax.Scraper(category='eess', date_from='2023-05-01',
                     date_until='2023-07-03', t=10,
                     filters={'categories':['eess.AS']})

output = scraper.scrape()

titles = [' '.join(o['title'].split()) for o in output]
np.savetxt('titles_ref.csv', np.array(titles), fmt='%s')
```

Dane wyciągnięte z bazy arXiv będą dotyczyć artykułów z zakresu „Electrical Engineering and Systems Science” i podkategorii „Audio and Speech Processing”.

Po uruchomieniu kodu, rozpoczyna się proces scrapingu, udało nam się wyciągnąć 3064 tematy.

```
fetching up to 1000 records...
fetching up to 2000 records...
fetching up to 3000 records...
fetching up to 4000 records...
fetching up to 5000 records...
fetching up to 6000 records...
fetching up to 7000 records...
fetching up to 8000 records...
fetching up to 9000 records...
fetching up to 10000 records...
fetching up to 11000 records...
fetching up to 12000 records...
fetching up to 13000 records...
fetching up to 14000 records...
fetching up to 15000 records...
fetching up to 16000 records...
fetching up to 17000 records...
fetching is completed in 205.0 seconds.
Total number of records 3064
```

Przykład kilku pierwszych w utworzonym dokumencie:

```
dichotic harmony for the musical practice
convolutional neural network achieves human-level accuracy in music genre classification
acoustic scene classification: a competition review
interfacing pdm mics microphones with pfm spiking systems: application for neuromorphic auditory sensors
periodicity pitch detection in complex harmonies on eeg timeline data
bayesian restoration of audio degraded by low-frequency pulses modeled via gaussian process
sound field translation and mixed source model for virtual applications with perceptual validation
version control of speaker recognition systems
jointly fine-tuning "bert-like" self supervised models to improve multimodal speech emotion recognition
ultra-low power on-chip learning of speech commands with phase-change memories
fullsubnet: a full-band and sub-band fusion model for real-time single-channel speech enhancement
data augmentation for end-to-end code-switching speech recognition
enhancement by postfiltering for speech and audio coding in ad-hoc sensor networks
bayesian learning for deep neural network adaptation
adjust-free adversarial example generation in speech recognition using evolutionary multi-objective optimization under black-box condition
unboxing engagement in youtube influencer videos: an attention-based approach
study of pre-processing defenses against adversarial attacks on state-of-the-art speaker recognition systems
monaural speech enhancement with complex convolutional block attention module and joint time frequency losses
low-latency auditory spatial attention detection based on spectro-spatial features from eeg
reduced basis methods for numerical room acoustic simulations with parametrized boundaries
adversarial attacks and defenses for speech recognition systems
using voice and biofeedback to predict user engagement during product feedback interviews
a new class of efficient adaptive filters for online nonlinear modeling
aligned contrastive predictive coding
audio transformers: transformer architectures for large scale audio understanding. adieu convolutions
phone-level prosody modelling with gmm-based mdn for diverse and controllable speech synthesis
improving the adversarial robustness for speaker verification by self-supervised learning
information retrieval for zerospeech 2021: the submission by university of wroclaw
human perception of audio deepfakes
a novel markovian framework for integrating absolute and relative ordinal emotion information
phone duration modeling for speaker age estimation in children
binaural rendering from microphone array signals of arbitrary geometry
a few-shot learning approach for sound source distance estimation using relation networks
emotional speech synthesis for companion robot to imitate professional caregiver speech
is attention always needed? a case study on language identification from speech
a method for capturing and reproducing directional reverberation in six degrees of freedom
```

3. Trenowanie modelu oraz przykładowe generacje

Następnie wytrenowano model GPT2, za ten proces odpowiada następujący fragment kodu:

```
sess = gpt2.start_tf_sess()
gpt2.finetune(sess,
               'titles_ref.csv',
               model_name=model_name,
               steps=500,
               save_every=50,
               sample_every=25)

gpt2.generate(sess)
```

Funkcja `.start_tf_sess()` rozpoczyna sesję TensorFlow, natomiast funkcja `.finetune()` odpowiada za rozpoczęcie procesu trenowania modelu. Trening modelu opierał się o 500 kroków, zapisywanych co 50 kroków, oraz generujących dane poglądowe co 25, aby móc śledzić postępy.

Trening ograniczono wstępnie do 500 kroków, ze względu na to, że zbiór danych na którym operowano, nie jest obszerny, a nie chciano doprowadzić do przeuczenia modelu. Proces ten dodatkowo monitorowano, sprawdzając wartość training loss, aby nie spadła ona poniżej 0.1, co mogłoby sygnalizować przeuczenie. Wartość ta określa jak bardzo model dopasował się do danych.

Poniżej przedstawiono przykładowe zrzuty ekranu z tego procesu.

Trenowanie modelu:

```
[101 | 2493.80] loss=1.38 avg=1.83
[102 | 2519.27] loss=1.58 avg=1.82
[103 | 2543.23] loss=1.45 avg=1.82
[104 | 2567.52] loss=1.56 avg=1.81
[105 | 2591.52] loss=1.37 avg=1.81
[106 | 2615.34] loss=1.48 avg=1.80
[107 | 2639.35] loss=1.48 avg=1.80
[108 | 2663.23] loss=1.22 avg=1.79
[109 | 2686.89] loss=1.41 avg=1.78
[110 | 2711.27] loss=1.50 avg=1.78
[111 | 2735.15] loss=1.37 avg=1.77
[112 | 2758.99] loss=1.30 avg=1.76
[113 | 2782.71] loss=1.40 avg=1.76
[114 | 2806.57] loss=1.30 avg=1.75
[115 | 2830.63] loss=1.26 avg=1.75
[116 | 2854.93] loss=1.31 avg=1.74
[117 | 2879.59] loss=1.35 avg=1.73
[118 | 2904.36] loss=1.33 avg=1.73
[119 | 2928.66] loss=1.44 avg=1.72
[120 | 2953.23] loss=1.28 avg=1.72
[121 | 2977.82] loss=1.18 avg=1.71
[122 | 3002.30] loss=1.31 avg=1.70
[123 | 3026.70] loss=1.10 avg=1.70
[124 | 3051.02] loss=1.47 avg=1.69
[125 | 3075.48] loss=1.13 avg=1.68
```

Przykładowe dane wygenerowane na 125 kroku:


```

===== SAMPLE 1 =====
text>
<|startoftext>mamba: state-space modeling for multi-level sound classification<|endoftext|>
<|startoftext>the voice2i of chinese deepfake audio dataset<|endoftext|>
<|startoftext>an acoustic baseline for sound event localization via latent diffusion<|endoftext|>
<|startoftext>deepfake speech detector pooling<|endoftext|>
<|startoftext>japanese deepfake audio dataset: measuring the computational power of speaker recognition with low-rank adaptation and noisy diffusion<|endoftext|>
<|startoftext>multilingual speaker diarization for automatic speakers from different languages with cross-speaker retrieval<|endoftext|>
<|startoftext>self-supervised speech representation learning on speaker diarization and attention<|endoftext|>
<|startoftext>exploring a language model for the multilingual asr challenge 2022<|endoftext|>
<|startoftext>soundsxml: introducing asr with style transform<|endoftext|>
<|startoftext>improving high-quality asr in monaural meetings: lessons from real-world meetings<|endoftext|>
<|startoftext>exploration of an asr-based language model for tts systems<|endoftext|>
<|startoftext>self-supervised transcription for neural end-to-end fast speech recognition<|endoftext|>
<|startoftext>enhancing speech analysis performance with text selective tts: real-world efficiency tuning<|endoftext|>
<|startoftext>robust localization of raspberry pi audio using deep convolutional neural networks<|endoftext|>
<|startoftext>exploring noise-relief neural networks for text-to-speech synthesis<|endoftext|>
<|startoftext>taming emotion responses using discrete token audio components<|endoftext|>
<|startoftext>multi-task attention learning for target speech extraction<|endoftext|>
<|startoftext>speech tokenization with speaker embeddings and synthetic bi-streaming<|endoftext|>
<|startoftext>speech recognition models: a comparison of embeddings and synthetic data<|endoftext|>
<|startoftext>maldespac3: robust speaker verification from microphone recordings<|endoftext|>
<|startoftext>deep learning fine-grained linear context selection for real-world audio-visual retrieval<|endoftext|>
<|startoftext>on the evaluation of automatic speech recognition in a large audio-language dataset<|endoftext|>
<|startoftext>enhancing synthetic robustness in the chinese deepfake audio dataset<|endoftext|>
<|startoftext>speech tokenization with feature decomposition<|endoftext|>
<|startoftext>dasr: neural audio tagging and representation learning for text to written speech<|endoftext|>
<|startoftext>solo streaming speech token codec for spoken language identification<|endoftext|>
<|startoftext>music generation by self-supervised learning from multi-attention speaker data<|endoftext|>
<|startoftext>pitch-aware pitch bias detection in cascaded neural networks<|endoftext|>
<|startoftext>speech language models for audio domain classification in low-latency static environments<|endoftext|>
<|startoftext>the gushing effect from your time: zero-shot text to audio generation<|endoftext|>
<|startoftext>towards machine learning-based voice identity and verification via adversarial learning<|endoftext|>
<|startoftext>exploring noise-based neural networks for pathological speech detection<|endoftext|>
<|startoftext>diff-mko: a computational audio spectrogramkit with reflections<|endoftext|>
<|startoftext>mamba-based end-to-end pre-training for audio classification and unsupervised speaker identification<|endoftext|>
<|startoftext>multi-investigation dual-feature fusion for keyword spotting in medical corpus analysis<|endoftext|>

```

4. Generowanie tytułów

Po finetuningu modelu, przetestowano działanie modelu. Dla przykładu wygenerowano 10 tematów bez prefiksu. Tematy należało wyciągnąć ze zwróconej listy i dla czytelności wyczyścić znaki początku i końca. Odpowiada za to poniższy kod:

```

import gpt_2_simple as gpt2
sess = gpt2.start_tf_sess()
gpt2.load_gpt2(sess)

text = gpt2.generate(sess,
    length=40,
    temperature=0.5,
    prefix=None,
    nsamples=10,
    batch_size=1,
    return_as_list=True
)

for t in text:
    print("title ---")
    chk = t.title()
    chk = chk.replace('<|startoftext|>', '').replace('\n', '') # remove extraneous stuff

    chk = chk[:chk.index('<|endoftext|>')]
    print(chk)
    print("----")

```

Funkcja generate() przyjmuje różne atrybuty:

- Length – określa maksymalną długość generowanego tekstu
- Temperature – im niższy tym bardziej przewidywalny jest wynik, im wyższy tym bardziej kreatywny
- Prefix – ciąg znaków od którego model ma rozpocząć tytuł
- Nsamples – ile model ma wygenerować przykładów
- Batch_size – ile tytułów model ma generować jednocześnie

Przykładowe 10 tematów wygenerowanych przez model:

```
title ---
Self-Supervised Speech Representations Are More Interpretable
---
title ---
Libritts-P: A Corpus With Speaking Style And Speaker Identity Prompts For Text-To-Speech And Style Captioning
---
title ---
>Multimodal Punctuation By Speaker Characteristic Measure
---
title ---
Leveraging Self-Supervised Models For Automatic Whispered Speech Recognition
---
title ---
>Diffusion Gaussian Mixture Audio Denoise
---
title ---
>Dynamic Humtrans: Humming Transcription Using Cnns And Dynamic Programming
---
title ---
Fusion Of Discrete Representations And Self-Supervised Representations For Multilingual Automatic Speech Recognition
---
title ---
Prosodic Estimation Of Corpus Networks For Speech Foundation Models
---
title ---
>A Survey Of Interactive Large Language Models
---
title ---
Acoustic Feature Mixup For Balanced Multi-Aspect Pronunciation Assessment
---
```

Jak widać, model wygenerował różnorodne tytuły, pojawiają się natomiast dodatkowe znaki takie jak „>”

5. Prefixy

Model przetestowano również wykorzystując prefixy. Poniżej przedstawiono wyniki dla prefixów „Neural” oraz „Music”

Neural:

```
title ---
Neural Speech Enhancement In Competitively-Spaced Multi-Channel Multi-Speaker Environment Using Discrete Units
---
title ---
Neuralized Spatial Learning For Audio Spoof Detection
---
title ---
Neural Cues In Speech Signal Processing: A Novel Data Augmentation Approach
---
title ---
Neuralized Text-To-Speech Synthesis With Controllable Semantic Representations
---
title ---
Neural Signature Detection For Speaker Verification
---
title ---
Neural Signature Generation Using Large Language Models
---
title ---
Neural Sensors In Speech Synthesis
---
title ---
Neural Signature Extraction Using Neural Audio Codec Neural Network
---
title ---
Neuralized Cross-Talker Asr Using Feature Fusion And Cross-Dataset Speaker Retrieval
---
title ---
Neural Cues In Speech Separation Using Beamformer
---
```

Music:

```

title ---
Music|>Soundsil-Ds: Deep Denoising And Denoising Ofsea Change Usingilses For Audio Classification
---
title ---
Music|>Musebarrelmusic: An Age-Restricted Dataset For Chord-Based Music Generation
---
title ---
Music For All Platforms
---
title ---
Music Speech Representation Learning
---
title ---
Music|>Gibberish Is All You Need For Membership Verification In Fb
---
title ---
Music|>SamL: Speaker Adaptive Mixture Of Experts For End-To-End Asr
---
title ---
Music>Mixture Of Experts Fusion For Llm-Based Asr With Enhanced Mems Microphone Arrays And Mems-Conformers
---
title ---
Music
---
title ---
MusicEmoknob: Enhance Voice Cloning With Fine-Grained Emotion Control
---
title ---
Music>Musebarcontrol: Enhancing Text-To-Music Generation With Style Captioning
---

```

6. Interpretacja danych

Tytuły wygenerowane przez model, różnią się od siebie oraz zawierają nieznaczące błędy. Może to być spowodowane przez wiele czynników, na przykład zbyt mały zbiór danych albo zbyt krótki proces uczenia.

Tematy wygenerowane bez prefiksów wyglądają akceptowalnie, są czytelne i w miarę logiczne, co mogłoby czynić je użytecznymi po odpowiednim dostosowaniu ich.

Dodanie prefiksu „Neural”, nie zaburzyło wyników, a wręcz je poprawiło. Tematy te są czytelne i logiczne, a większość z nich wygląda na użyteczne. Inny przetestowany prefiks, czyli „Music” spowodował wygenerowanie tytułów, które nie są w większości związane z tematem, ani nie są do końca logiczne. Może to być spowodowane znikomym związkiem prefiksu z danymi na których podstawie dostrajano model.

IV. Podsumowanie

Stworzenie generatora tytułów dla artykułów na podstawie wstępnie wytrenowanego modelu GPT2, oraz dostrojenie go, pozwoliło nam uzyskać narzędzie, które mogłoby być pomocne, natomiast nie jest ono idealne i często dostarcza wyniki które nie są w żadnym stopniu użyteczne. Dostrojenie modelu na większym zbiorze danych, lub

wykorzystanie większego modelu być może wpłynęłoby pozytywnie na jakość. Niemniej jednak cel został osiągnięty, a część tematów wygląda na całkiem użyteczne.

V. Bibliografia

- a. Gpt-paper-title-generator - <https://github.com/csinva/gpt-paper-title-generator>
- b. arXiv - <https://arxiv.org/>
- c. gpt-2-simple - <https://github.com/minimaxir/gpt-2-simple>
- d. arxivscraper - <https://github.com/Mahdisadjadi/arxivscraper>
- e. What is data scraping and how can you use it? - <https://targetinternet.com/resources/what-is-data-scraping-and-how-can-you-use-it>
- f. Fine-Tuning the Model: What, Why, and How - <https://medium.com/@amanatulla1606/fine-tuning-the-model-what-why-and-how-e7fa52bc8ddf>
- g. <https://openai.com/index/better-language-models/>