Generowanie tytułów publikacji naukowych na podstawie danych ArXiv i gpt2

Kraków, 2022

Spis treści

Autorzy

Krzysztof Wodecki

Hubert Tacik

Karol Waligóra

Abstrakt

Niniejszy projekt podejmuje próbę opracowania rozwiązania pozwalającego na generowanie tytułów publikacji naukowych na podstawie najnowszych trendów. Prognozowanie postępu badań naukowych zdaje się być celem bardzo istotnym, a zarazem niezwykle trudnym do osiągnięcia. Projekt wykorzystuje dane artykułów naukowych z pierwszego półrocza roku 2022, aby na ich podstawie generować proponowane tytuły prac z dziedziny przetwarzania języka naturalnego. Do osiągnięcia celów projektu wykorzystano język Python, bazę danych archiwum ArXiv oraz model GPT-2.

1. Wstęp

Celem niniejszej pracy jest opracowanie narzędzia, które pozwala na generowanie tytułów publikacji naukowych. Ten temat jest istotny z punktu widzenia prognozowania postępu związanego z badaniami i rozwojem dziedziny sztucznej inteligencji jaką jest przetwarzanie języka naturalnego (ang. Natural Language Processing - NLP). Generowanie proponowanych tytułów publikacji odbywa się na podstawie bazy danych ArXiv [4]. Wykorzystując bibliotekę i model sieci neuronowej GPT-2 analizie zostaną poddane prace naukowe, których data publikacji obejmuje pierwsze półrocze roku 2022. Wynikiem projektu jest wytrenowana sieć neuronowa umożliwiająca generowanie proponowanego tytułu publikacji z dziedziny NLP.

2. Realizacja

Realizacja projektu opiera się na trzech głównych elementach: pobraniu danych trenujących, uczeniu modelu sieci neuronowej i generowaniu tytułów na wytrenowanej sieci. Kod źródłowy wykorzystany i przedstawiony w projekcie został opracowany na bazie projektu *gpt-paper-title-generator* [2].

2.1. Pobranie danych trenujących

Dane trenujące zostały pobrane z archiwum ArXiv[4] za pomocą biblioteki *arxivscraper* [3]. Biblioteka umożliwia filtrowanie pobieranych danych względem kategorii i daty publikacji. Pobrane dane obejmują dziedzinę NLP i pochodzą z pierwszego półrocza roku 2022. Tytuły publikacji zostały zapisane do pliku CSV w celu ułatwienia późniejszej analizy.

```
import arxivscraper as ax
import numpy as np

scraper = ax.Scraper(category='cs', date_from='2022-01-01',
                     date_until='2022-07-01', t=10,
                     filters={'categories':['cl']})

output = scraper.scrape()

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

Rysunek 1. Pobranie danych z ArXiv.

2.2. Model sieci neuronowej GPT-2 (117M)

Trenowanie sieci odbywa się na podstawie modelu 117M dostarczanego przez GPT-2. Stanowi on jeden z wielu dostępnych modeli w bibliotece GPT-2 i jest złożony z 12 wejść, 768 warstw ukrytych, 12 wyjść oraz 117 milionów parametrów (stąd nazwa) [1].

Aby skorzystać z modelu 117M należy go pobrać i zapisać. Moduł *gpt_2_simple* dedykowany dla języka Python umożliwia łatwe pobranie i zapisywanie modelu dostępnych w ramach GPT-2.

Kod przedstawiony poniżej sprawdza, czy modelu został już pobrany. Jeśli nie, pobiera go i zapisuje w katalogu *models.*

```
import gpt_2_simple as gpt2
import os.path

model_name = "117M"

if not os.path.isdir("./models"):
    gpt2.download_gpt2(model_name=model_name)
    # model is saved into current directory under /models/117M/
else:
    print("Model already exists.")
```

Rysunek 2. Pobranie modelu 117M.

Pobrany model uczony jest na podstawie danych pobranych z ArXiv (2.1.). Wykorzystana została metoda *finetune()* dostępna z poziomu biblioteki *gpt_2_simple.* Dane ustalone zostały na 1000 generacji uczenia, a co 10 iteracji do pliku zapisywany jest aktualny model i wygenerowane podczas nauki tytuły publikacji. Naukę można przerwać w każdym momencie, co inicjuje proces zapisania modelu. Po uruchomieniu skryptu model zostanie wczytany i nie ma konieczności ponownego uczenia przy każdym uruchomieniu.

```
import gpt_2_simple as gpt2

model_name = "117M"

sess = gpt2.start_tf_sess()
gpt2.finetune(sess,
              'titles_ref.csv',
              model_name=model_name,
              steps=1000,
              #reuse=True,
              save_every=10, # checkpoint nauczonego modelu co 10 iteracji
              sample_every=10) # co 10 iteracji pojawia się próbka danych
```

Rysunek 3. Trenowanie modelu.

Wygenerowane podczas uczenia tytuły mogą zostać wyświetlone, po uprzednim usunięci zbędnych ciągów znaków. Przykład został zaprezentowany na rysunku poniżej.

Slide Type Fragment

```
sample_file = 'samples/run1/samples-51'
t = open(sample_file, 'r').read()

for s in ['endoftext', 'startoftext', '<|', '|>']:
    t = t.replace(s, '')
for title in t.title().split('\n')[1:]:
    if not title == '':
        print('- ' + title)
```

```
- Texttextsubunit: A New Toolkit For Studying Machine Translation
- Possible Explanations Of Model Errors In English Language Models
- Concentric Learning Towards Scalable Learning Towards Fine-Tuning Of Latent-Variance Network
- Districts And Pretrained Representations For Neural Language Analysis
- A Case Study On The Importance Of Covid-19 Statistics In Studying Hate Speech In The Quran Forums
- Extracting Language-Aware Subtitles With Low-Cost And High-Cost Transformer
- Learning To Express Human Emotions With An Effective Language Model
- A Few-Shot Question Answering Machine Learning Algorithm For Natural Language Identification
- A Generalisation Of Bert Models
- On The Importance Of Covid—19 Data In Hate Speech Detection
- Conforming A Simple To Model Neural Network To Generate Large Scale Models
- Tendency Estimation For Text-To-Audio Conversion
- A Benchmark For Evaluating The Feasibility Of Pretrained And Distanced Models For Document Consistency Checking
- Constraining Pretrained Representations To The Data: Understanding The Impact Of Loss Of Dimension For Transforme
r In Convolutional Neural Networks
- Modeling Language Models With Text-To-Speech And Textual Descriptions: Exploring Bert Models And Their Relevance
To Text Classification Models
- A Novel Approach To Improving Language Model Training For Language Models
- An Efficient Model For Zero-Shot Dialogue Response Generation
- Theoretically-Relevant Transformer Transformer: A Benchmark For Pre-Trained Models And Their Application To Text
Classification
- Cadet: A Graph-Based Pre-Trained Language Model For End-To-End Dialogue Classification With An Iterative Learning
Model
- A Theoretical Evaluation Of Covid-19 Data From Japanese State-Of-The-Art Natural Language Processing
- The Impact Of Pre-Trained Language Models On Semantic Classification Accuracy At The Language Level
- On The Impact Of Pretraining In Pre-Trained Models From Natural Language Processing
- An Attention On Language In Relation To Natural Language Comprehension In Pre-Trained Natural Language|
- Modeling The Relationship Between Language-Level Contrastivity And Few-Shot Dialogue Adaptation And Training
- A Scientific Study Of The Impact Of Linguistic Change On Natural Language Translation
- The Impact Of Social Information On Online Comments And Posts
- Knowledge-Informed Conversational Self-Documentation With Knowledge Graph Neural Networks
- A Review Of Approaches To Explainer In Language Models
- A Few Examples Of Language-Based Model Named After A Named Entity In Nuzhi-Nlp: A Survey
- Reinforcement Learning For End-To-End Speech And Text Classification In Bert-Based Text Augmentation
- Towards A Simple And Efficient Model For Natural Language Prediction: A Systematic Approach
- The Importance Of The Covid-19 Datasets For Information Extraction In Danish Legal Literature
- The Impact Of Pre-Trained Models On Conversational Privacy In Pre-Trained Language Models
- Evaluating The Health Of Nlp
```

Rysunek 4. Wyświetlenie tytułów wygenerowanych podczas uczenia.

2.3. Generowanie tytułu publikacji

Po wytrenowaniu modelu można go wczytać i wygenerować nowe tytuły publikacji. W tym celu posłużono się metodą *load_gpt2()*, która wczytuje model. Następnie wykorzystują metodę *generate()* wygenerowano przykładowy tytuł publikacji. W celu prawidłowej prezentacji wyniku, zbędne fragmenty wygenerowanego tytułu zostały usunięte. Kod źródłowy oraz przykład działania został przedstawiony na rysunku poniżej.

Slide Type Fragment

```
# inicjalizacja sieci neuronowej gpt2 w celu generowania danych z nauczonego już modelu
import gpt_2_simple as gpt2
sess = gpt2.start_tf_sess()
gpt2.load_gpt2(sess)


text = gpt2.generate(sess,
                     length=400,
                     temperature=0.7,
                     nsamples=1,
                     batch_size=1,
                     return_as_list=True)


t = text[0].title()
t = t.replace('<|Startoftext|>', '').replace('\n', '') # remove extraneous stuff
t = t[:t.index('<|Endoftext|>')] # only get one title
t = t.replace('|', '').replace('>', '').replace('<','')
print(t)
```

```
Loading checkpoint checkpoint\run1\model-50
INFO:tensorflow:Restoring parameters from checkpoint\run1\model-50
Text-To-Speech: A Unified Multilingual Text-To-Speech (Text-To-Speech) Corpus For Scientific Text Classification In
Texts
```

Rysunek 5. Generowanie tytułów na wytrenowanej sieci.

3. Podsumowanie

O ile wyniki przedstawione na rysunku 4. przedstawiają sensowne dane, a tytuły wygenerowane podczas nauki mogłyby faktycznie stanowić temat badań naukowych, to generowanie tytułów na wyuczonym modelu nie zawsze daje tak dobre wyniki. Zdarza się, że wygenerowany tytuł zawiera bezpodstawne powtórzenia (tak jak w przykładzie na rysunku 5.) lub jego treść jest absurdalna. Znaczna część przypadków generowania daje całkiem dobre rezultaty (niekiedy wymagają delikatnego przeredagowania). W naszej opinii opracowane narzędzie może dawać pogląd na trendy badań naukowych w temacie NLP oraz proponować tematy kolejnych badań, a także stanowić podstawę przewidywania dalszego rozwoju nauki w dziedzinie NLP.

4. Bibliografia

[1]. *GPT-2,* https://en.wikipedia.org/wiki/GPT-2

[2]. *Gpt-paper-title-generator,* https://github.com/csinva/gpt-paper-title-generator

[3]. *Arxivscraper,* https://github.com/Mahdisadjadi/arxivscraper

[4]. *ArXiv*, https://arxiv.org/category_taxonomy