

Generowanie tytułów publikacji naukowych

Jakub Możdżeń
Łukasz Rotko

1. Abstrakt

Ostatnie osiągnięcia w dziedzinie sztucznej inteligencji zdobyły światową sławę, stając się niezastąpionymi narzędziami w życiu codziennym. Niniejszy raport prezentuje praktyczne wykorzystanie tej technologii jako wsparcia w procesie tworzenia publikacji naukowych.

Rozwiązanie, które przedstawimy jest jedną z możliwości zastosowania tej technologii jako asystenta dla autorów publikacji naukowych w szczególnie ważnym momencie tworzenia, którym jest sporządzenie tytułu publikacji. Dzięki zaawansowanym modelom, teraz możliwe jest generowanie tytułów jedynie na podstawie dostarczonej bibliografii, co zdecydowanie ułatwia ten kreatywny proces. W dalszej części raportu przedstawimy cel naszej pracy oraz omówimy fundamenty naszego innowacyjnego rozwiązania tego problemu.

2. Wstęp

Celem projektu jest stworzenie narzędzia do generacji tytułów publikacji naukowych. Staramy się przewidzieć jakie mogą być kolejne artykuły danego autora na podstawie wcześniejszych jego prac.

Projekt skupia się na pobieraniu danych z archiwum, analizowaniu ich oraz tworzeniu nowych rozwiązań. Ważne jest też przygotowanie danych w celu ich dalszego przetwarzania, czyli skupienie się na usuwaniu duplikatów oraz pozbyciu się problematycznych tytułów np. takich, które zawierają cyfry, żeby uniknąć potencjalnych błędów.

Wykorzystujemy przy tym GPT-3 bez dostrajania do generowania tytułów uzależnionych od konkretnego autora. Korzystamy z arXiv, elektronicznego archiwum artykułów naukowych do pobrania publikacji autorów.

3. Część Teoretyczna

Aby przygotować się do stworzenia asystenta generującego tytuły, najpierw zapoznaliśmy się z dokumentacją zawartą w projekcie "gpt-paper-title-generator", który jest podstawą całego narzędzia. Z tej dokumentacji dowiedzieliśmy się, że cały projekt opiera się na języku Python i wykorzystuje notatniki Jupyter do zgrabnego przedstawienia, jak działa zawarty w nim kod. Musieliśmy więc zasięgnąć do dokumentacji IDE PyCharm, które oferuje wygodną pracę z Python'em oraz notatnikami Jupyter'a.

Poprzez czytanie kodu zawartego w projekcie, byliśmy w stanie dowiedzieć się bardzo dużo o jego działaniu. Pewne ograniczenia w generowaniu danych oraz braki w plikach, zmusiły nas na podejście do tego etapu nauki w inny sposób niż planowaliśmy. Nie byliśmy w stanie stworzyć modeli od zera, jednak udało nam się odzyskać wcześniej wytrenowane modele. Musieliśmy je przekonwertować na inny format pliku, aby kontynuować pracę nad projektem i tu z pomocą przyszedł nam wpis na platformie GitHub o narzędziu "pickle2json".

Po zbieraniu wszystkich powyższych elementów, byliśmy w stanie uzyskać pierwsze tytuły publikacji naukowych.

4. Część praktyczna

W pierwszym kroku wykorzystaliśmy funkcję do przetwarzania metadanych artykułów naukowych z platformy arXiv. Usuwamy niepotrzebne znaki i tworzymy listę i słownik dla autorów i artykułów.

```
def get_metadata():
    if os.path.exists('data/arxiv_metatadata_2022_clean.pkl'):
        df = pd.read_pickle('data/arxiv_metatadata_2022_clean.pkl')
    else:
        df = pd.read_pickle('data/arxiv_metatadata_2022.pkl')
        # df['update_date'] = pd.to_datetime(df['update_date'])
        df['date'] = pd.to_datetime(df['versions'].apply(lambda x: x[0]['created'])) # converting to datetime takes a while
        df = df.drop(columns=['versions', 'update_date'])
        df = df.drop_duplicates(subset=['title', 'authors'], keep='first') # replicated submissions
        # df['num_versions'] = df['versions'].apply(len)
        df['year'] = pd.DatetimeIndex(df['date']).year
    def clean_title(s):
        s = s.replace('\n', '')
        s = s.replace('\t', '')
        s = re.sub(' +', ' ', s)
        return s
    df['title'] = df['title'].apply(clean_title)
    df['title_len'] = df['title'].str.split(' ').apply(len)
    df = df[~df['title'].str.lower().str.startswith('comment')]
    df = df.sort_values(by='date') # most recent papers at bottom
    df = df.to_pickle('data/arxiv_metatadata_2022_clean.pkl')
    return df

df = get_metadata()

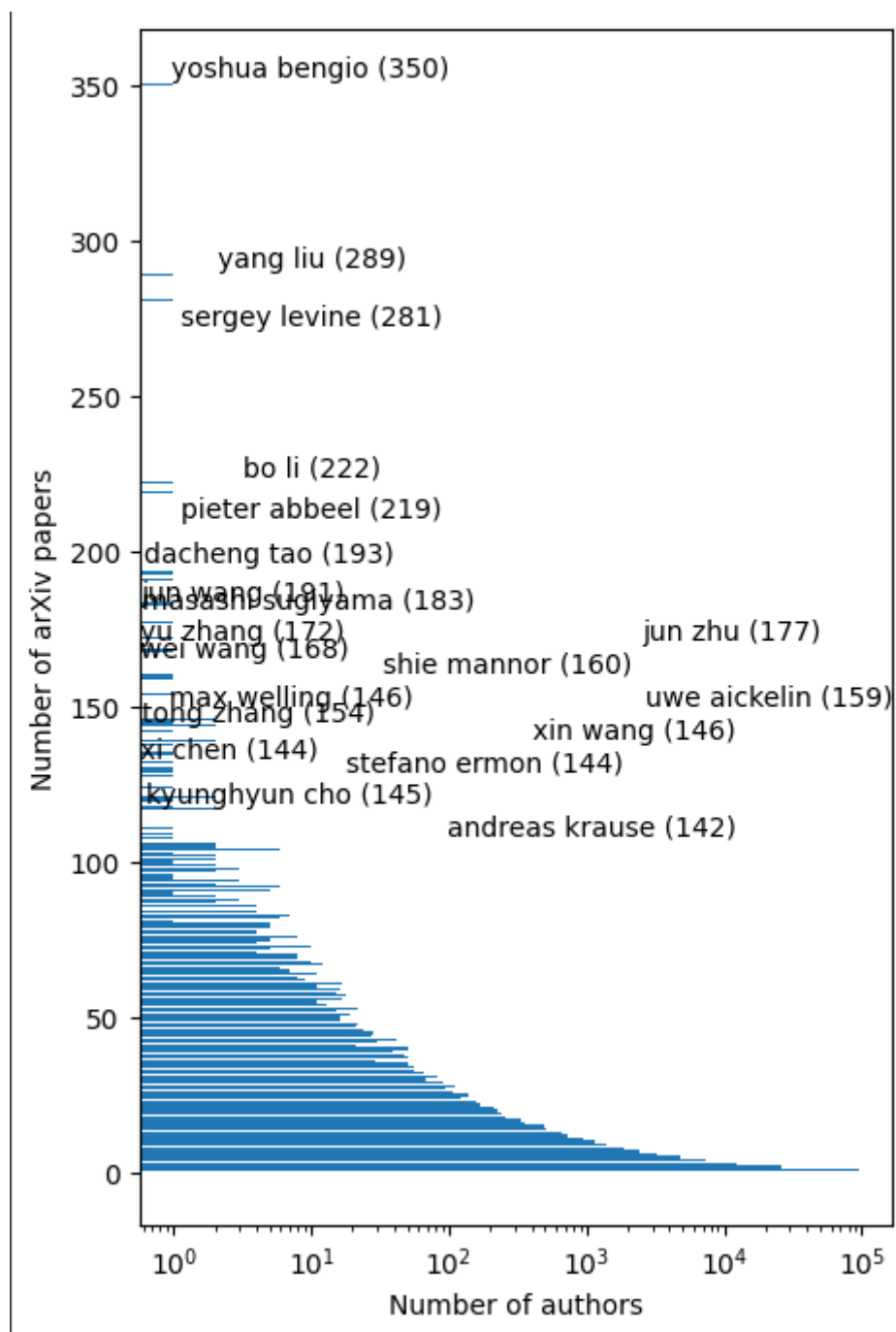
def get_authors_list(s: str) -> List:
    return [
        a.lower().strip()
        for a in re.split('and |, |\n', s)
    ]

authors_list_arr = df.authors.apply(get_authors_list)
author_dict = defaultdict(list)
for i, l in enumerate(authors_list_arr):
    for author in l:
        author_dict[author].append(i)
```

Część kodu, która usuwa duplikaty oraz odpowiada za wyszukiwanie nazw ze zmiennej keywords w celu wyrzucenia artykułów, które mogą być problematyczne dla tworzenia nowych tytułów. Sprawdza również takie rzeczy jak to czy nie ma cyfr lub długość autora ma więcej niż 3 znaki.

```
# prune the authors list
authors_list = sorted(set(list(itertools.chain.from_iterable(authors_list_arr))))
def contains_num(x):
    if True in [char.isdigit() for char in x]:
        return True
    return False
keywords = re.compile(r'universi|departm|faculty|institute|global|center|laboratory')
authors_list = [
    a for a in authors_list
    if keywords.search(a) is None
    and not any(p in a for p in string.punctuation)
    and not contains_num(a)
    and len(a) > 3
    and len(a.split()) > 1
    and len(a.split()[0]) > 1
]
author_dict_clean = {
    a: author_dict[a]
    for a in authors_list
}
assert len(authors_list) == len(author_dict_clean)
```

Wykres przedstawiający liczbę autorów oraz liczbę jego artykułów. Pobiera 20 autorów z największą ilością prac z arXiv.



Sortujemy po liczbie artykułów dla autora:

```
authors = np.array(list(authors_dict_titles.keys()))
args = np.argsort(paper_counts[::-1])
authors_dict_titles = {
    a: authors_dict_titles[a]
    for a in authors[args]
}
```

Wczytujemy dane i filtrujemy je w taki sposób, że bierzemy autorów, którzy mają więcej niż 2 tytuły.

```
# authors sorted in descending order by num papers
authors_dict_titles = pickle.load(open('data/authors_dict_titles.pkl', 'rb'))
prompt = 'Here is a list of related machine-learning papers:\n\n> '
authors_save = {}
authors = list(authors_dict_titles.keys())

# settings to save
authors = [a for a in authors if len(authors_dict_titles[a]) > 2]
gens_per_author = 5
papers_in_context = 5
```

Ustawiamy klucz API OpenAI, po czym generujemy nowe tytuły poprzez zapytania do GPT-3:

```
# run
for i, author in enumerate(tqdm(authors)):
    if not author in authors_save:
        query = prompt + '\n> '.join(authors_dict_titles[author][-papers_in_context:]) + '\n>'
        completion = openai.completions.create(
            model='text-davinci-002',
            prompt=query,
            max_tokens=200,
            stop='\n>',
            n=gens_per_author
        )
        authors_save[author] = [completion.choices[i].text for i in range(len(completion.choices))]
    if i % 200 == 0:
        pickle.dump(authors_save, open(f'gen_titles/authors_save_{i}.pkl', 'wb'))
pickle.dump(authors_save, open(f'gen_titles/authors_save_full.pkl', 'wb'))
```

Wykorzystanie kodu “pickle2json” do konwersji danych:

```

import pickle
import json
import sys
import os

# open JSON file
with open("gen_titles/authors_save_full.json", 'r', encoding='utf-8') as infile:
    json_str = infile.read()

# convert JSON string to JSON object
json_obj = json.loads(json_str)

# convert JSON object to Pickle object
obj = json_obj

# write the Pickle file
with open(
    'authors_save_full.pkl',
    'wb'
) as outfile:
    pickle.dump(obj, outfile)

```

Nowe tytuły dla jednego z autorów:

```

author = 'pieter abbeel'

print(author)
titles = authors_save_processed[author]

for title in titles:
    print('\t', title)

```

```

pieter abbeel
    hierarchicalrl: A Framework for Multi-Level Reinforcement Learning
    Investigate Neural Representation for Sequential Decision Making Tasks
    Context-Aware Neural Machine Translation
    A Simple Approach for Time-Varying Graphical Models
    Tracking the World State with Neural Maps

```

Istniejące tytuły dla jednego z autorów:

```

query = '\n> '.join(authors_dict_titles['pieter abbeel'][-5:]) + '\n>'
print(query)

AdaCat: Adaptive Categorical Discretization for Autoregressive Models
> Multi-Objective Policy Gradients with Topological Constraints
> Reducing Variance in Temporal-Difference Value Estimation via Ensemble of Deep Networks
> Temporally Consistent Video Transformer for Long-Term Video Prediction
> Real-World Robot Learning with Masked Visual Pre-training
>

```

5. Podsumowanie

Przygotowaliśmy narzędzie do generacji tytułów publikacji naukowych. Podczas prac nad projektem nabyliśmy różnorodną wiedzę oraz pokonaliśmy kilka przeszkód. Również widzimy możliwości ulepszenia asystenta poprzez zwiększenie lub zróżnicowanie materiałów jakie mu podamy, np. takie zbazowane na Języku Polskim. Niestety w tym momencie jest to dla nas niemożliwe ze względu na koszty. Podczas pracy nad projektem natknęliśmy się na przeszkodę jaką jest ograniczenie zapytań wysyłanych do modeli zarządzanych przez openAi. Przykładowe dane znacznie przekraczały darmową ilość zapytań, nie wspominając o dodawaniu własnych. Dlatego też budowaliśmy całość na danych oferowanych przez twórców “gpt-paper-title-generator”, aby można było zobaczyć asystenta w akcji.

6. Bibliografia

1. Romulo Gapuz Jr., 2020, "pickle2json"
GitHub. <https://gist.github.com/romgapuz/c7a4cedb85f090ac1b55383a58fa572c>
2. Chandan Singh, 2022, “gpt-paper-title-generator”
GitHub. <https://github.com/csinva/gpt-paper-title-generator>
3. Python Software Foundation, 2023, “Python 3.12.1 - dokumentacja”
Python 3. <https://docs.python.org/3/>
4. JetBrains s.r.o., 2023, “ProfessionalJupyter notebook support”
<https://www.jetbrains.com/help/pycharm/jupyter-notebook-support.html>