DAFTAR REFERENSI

Akash, T. (2023). *How to build a GPT model?* Diakses dari LeeWayHertz: https://www.leewayhertz.com/build-a-gpt-model/

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., . . . Amodei, D. (2020). *Language Models are Few-Shot Learners. OpenAI.*

Chou, Y.-Y., Lin, H.-T., & Liu, T.-L. (2021). *Adaptive And Generative Zero-Shot Learning.*

Dale, R. (2021). GPT-3: *What's it good for? Natural Language Engineering*, 113-118.

Fang, J. (2021). *An Application of Customized* GPT-2 *Text Generator for Modern Content Creators.*

Farhad, P., Moloud, A., Luo, Y., Zhou, X., Lim, C. P., Wang, X.-Z., & Wu, Q. J. (2023). *A Review of Generalized Zero-Shot Learning Methods*. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 4051-4070.

Fatima, N., Imran, A. S., Kastrati, Z., Daudpota, S. M., & Soomro, A. (2022). A *Systematic Literature Review on Text Generation Using Deep Neural Network Models*. 53490-53503.

Generative AI Models Explained. (2022, Oktober 13). Diakses dari altextsoft: https://www.altexsoft.com/blog/generative-ai/

Gillioz, A., Casas, J., Mugellini, E., & Khaled, O. A. (2020). *Overview of the Transformer-based Models for NLP Tasks*. *Federated Conference on Computer Science and Information*, 179-183.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). *Generative Adversarial Nets.*

Heryanto, E. (2012, Januari 31). Kumpulan Pantun Karangan Sutan Takdir Alisjahbana. Diakses dari scribd: https://www.scribd.com/doc/79923239/Kumpulan-pantun

Ilahiyah, S., & Nilogiri, A. (2018). Implementasi *Deep Learning* Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan *Convolutional Neural Network*. Jurnal UNMUH Jember, 49-56.

Iqbal, T., & Qureshi, S. (2020). *The Survey: Text Generation Models In Deep Learning*.

Journal of King Saud University – Computer and Information Sciences, 2516-2526.

Irmanda, H. N., Astriratma, R., Chamidah, N., & Santoni, M. M. (2021). Pembuat Sampiran Pantun Otomatis berbasis *Pattern-matching*. Jurnal SISFOKOM (Sistem Informasi dan Komputer), 306-311.

Iswara, P. D. (2012). Pantun Bertema untuk Bahan Ajar di Sekolah Dasar, Sekolah Menengah Pertama, Sekolah Menengah Atas dan Perguruan Tinggi. 1-30.

Kamnis, S. (2023). *Generative Pre-Trained Transformers (GPT) For Surface Engineering*. *Surface and Coatings Technology*.

Kurniasari, D., & Widya, A. (2021, Oktober 8). Pengertian Teknik Pengolahan Data dan Macam-Macam Jenisnya. Diakses dari DQLab: https://dqlab.id/pengertian-teknik-pengolahan-data-dan-macam-macam-jenisnya

Lamb, A. (2021). *A Brief Introduction to Generative Models*. *Department of Informatics and Operations Research*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep Learning*. *NATURE*, 436-444.

Lee, A. (2022, Desember 8). *What Is a Pretrained AI Model?* Diakses dari NVIDIA: https://blogs.nvidia.com/blog/2022/12/08/what-is-a-pretrained-ai-model/

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., . . . Kiela, D. (2021). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*.

Li, T., Beirami, A., Sanjabi, M., Smith, V., & Harchaoui, Z. (2023). *On Tilted Losses in Machine Learning*: *Theory and Applications*. *Journal of Machine Learning Research*, 1-79.

Liddy, E. D. (2001). *Natural Language Processing.* Marcel Decker, Inc.

Lo, K. L., Ariss, R., & Kurz, P. (2022). GPoeT-2: *A* GPT-2 *Based Poem Generator*.

Łukawski, K. (2021, Juni 10). *Garbage in, Garbage out – Preparing Your Data Set for Machine*. Diakses dari Codete: https://codete.com/blog/garbage-in-garbage-out-preparing-your-data-set-for-machine-learning

Luo, R., Sun, L., Xia, Y., Qin, T., Zhang, S., Poon, H., & Liu, T.-Y. (2023). BioGPT: *Generative Pre-trained Transformer for Biomedical Text Generation and Mining.*

Mahesh, B. (2020). *Machine Learning Algorithms - A Review*. *International Journal of Science and Research (IJSR)*, 381-386.

McKinsey. (2023, Januari 19). *What is generative AI?* Diakses dari McKinsey & Company: https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-generative-ai

Muhaling, S. S. (2015, Mei 24). Kumpulan Pantun Anak. Diakses dari scribd: https://www.scribd.com/doc/266390396/Kumpulan-Pantun-Anak

Pebrianto, A. (2017). Analisis Kesulitan Siswa Dalam Menulis Pantun Sesuai Dengan Syarat Pantun Di Smp Negeri 16 Surakarta.

Pipa, G., & Khalil, F. (2022). *Transforming The Generative Pretrained Transformer Into Augmented Business Text Writer*. *Journal of Big Data*, 1-21.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving Language Understanding.*

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language Models are Unsupervised Multitask Learners.*

Ridwan, E. (2022, Oktober 23). Pantun adalah Apa? Ini Penjelasan, Kaidah dan Contoh yang Benar. Diakses dari detikSulSel: https://www.detik.com/sulsel/berita/d-6363560/pantun-adalah-apa-ini-penjelasan-kaidah-dan-contoh-yang-benar

Rothe, S., Narayan, S., & Severyn, A. (2020). *Leveraging Pre-trained Checkpoints for Sequence Generation Tasks.*

Rustiawati, T. (2016). Penerapan Model *Quantum Learning* Melalui Teknik Permainan Kata Dalam Pembelajaran Menulis Pantun.

Sahid, G., Hambali, M. A., Wirawan, C., Fhadli, M., Rahmadani, S., Alvinwatner, & Akmal. (2022). *GPT2-medium-indonesian*. Diakses dari Hugging Face: https://huggingface.co/indonesian-nlp/gpt2-medium-indonesian

Setyadiharja, R. (2020). Khazanah Negeri Pantun. Yogyakarta: Deepublish.

Shree, P. (2020, November 10). *The Journey of Open AI GPT models*. From Medium: https://medium.com/walmartglobaltech/the-journey-of-open-ai-gpt-models-32d95b7b7fb2

Singh, T. (2021, Oktober 25). *Streamlit: A must learn tool for data Scientist*. Diakses dari Medium: https://medium.com/crossml/streamlit-2256000541ad

Syuhada, A. S., Simanullang, A. M., Lewa, D. S., & Marthin, S. J. (2021). Pembelajaran Mesin (*Machine Learning)*.

Tineges, R., & Davita, A. W. (2021, Juni 17). Tahapan *Text Preprocessing* dalam Teknik Pengolahan Data. Diakses dari DQLab: https://dqlab.id/pengertian-teknik-pengolahan-data-dan-macam-macam-jenisnya

Tran, K. (2020, April 6). *What is PyTorch?* Diakses dari Towards Data Science: https://towardsdatascience.com/what-is-pytorch-a84e4559f0e3

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). *Attention Is All You Need*. *31st Conference on Neural Information Processing Systems (NIPS 2017),*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., . . . Rush, A. M. (2020). Huggingface *Transformers: State-of-the-Art Natural Language Processing*. 38-45.

Yenduri, G., Srivastava, G., Maddikunta, P. K., Jhaveri, R. H., Wang, W., Vasilakos, & Gadekallu, T. R. (2023). Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions. *Computation and Language*, 1-40.

LAMPIRAN

- ***Dataset Loader***

```
!gdown https://drive.google.com/uc?id=1kBI1DuPQAje8vgwtu1kednDZDXMLacWJ
```

- ***Utility Script Loader* (run_language_modeling_py & run_generation_py)**

```
!gdown https://drive.google.com/uc?id=1rD_Y8MVlOsUPCF6rofbxUfCRsFymvMH4
```

- ***Unzip Utility Script***

```
import os
import zipfile
local_zip = '/content/Utility_Scripts.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/content')
zip_ref.close()
```

- ***Prepare Dataset with pandas***

```
import pandas as pd
from sklearn.model_selection import train_test_split
import os

data = pd.read_csv("/content/Dataset_18_05_2023_sorted.csv")
print(data.shape)
#data.head()
data
```

- ***Show* jenis-jenis Pantun dalam dataset**

```
# Menghitung jumlah jenis pantun
jenis_pantun = data['tipe'].value_counts()

print(jenis_pantun)
```

- ***Last Preprocessing split train data & 5% test data***

```
train, test = train_test_split(data.teks.values,
                    test_size=0.05,
                    random_state=42)
```

```
print('Len train: ', len(train))
print('Len test: ', len(test))

with open('train.txt', 'w+') as f:
  for text in train:
    f.write('<BOS> ' + repr(text)[1:-1] + ' <EOS>\n')

with open('test.txt', 'w+') as f:
  for text in test:
    f.write('<BOS> ' + repr(text)[1:-1] + ' <EOS>\n')
```

- ***Install Requirements* (Transformers)**

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

- ***Fine-Tuning with Utility Script* run_language_modeling.py**

```
!python /content/Utility_Scripts/run_language_modeling_py.py \
--output_dir='./gpt2-pantun' \
--model_type=gpt2 \
--model_name_or_path='indonesian-nlp/gpt2-medium-indonesian' \
--do_train \
--train_data_file='train.txt' \
--do_eval \
--eval_data_file='test.txt' \
--per_device_train_batch_size=8 \
--per_device_eval_batch_size=8 \
--line_by_line \
--logging_steps=6 \
--save_steps=500 \
--learning_rate 5e-5 \
--num_train_epochs=3
```

- ***Show perplexity evaluation score***

```
# Pakai Epoch: 3 (228 iterasi)
!head './gpt2-pantun/eval_results_lm.txt'
```

- ***(optional) Save Model to Google Drive***

```
!zip -r /content/gpt2-pantun.zip /content/gpt2-pantun
from google.colab import drive
drive.mount('/content/gdrive')
%cp /content/gpt2-pantun.zip /content/gdrive/MyDrive/riki_TA/epoch3
```

- ***(optional) Load Model from Google Drive***

```
import os
import zipfile
# epoch 3 dari new :
https://drive.google.com/file/d/1EZXgTbY1zanslVQ7WZgaaHMBGxPWwfuM/view?usp=sharing
!gdown
https://drive.google.com/uc?id=1EZXgTbY1zanslVQ7WZgaaHMBGxPWwfuM

# Unzip
local_zip = '/content/gpt2-pantun.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('./')
zip_ref.close()

# Move Folder
%cp -r /content/content/gpt2-pantun gpt2-pantun

# Remove empty folder
!rm -rf /content/content

# See Perplexity
!head './gpt2-pantun/eval_results_lm.txt'
```

- ***Generate Pantun after fine-tuning the model with Utility Script* run_generation.py**

```
!python /content/Utility_Scripts/run_generation_py.py \
--model_type gpt2 \
--model_name_or_path '/content/gpt2-pantun/' \
--length 100 \
--prompt "<BOS>" \
--stop_token "<EOS>" \
--k 50 \
--num_return_sequences 50 > generated-pantun.txt
```

- **Menggabungkan Kumpulan Pantun-pantun yang sudah di-*generate* kedalam satu txt**

```
with open('generated-pantun-alpha.txt', 'r') as file1, open('generated-pantun.txt', 'r') as file2:
   content1 = file1.read()
```

```
    content2 = file2.read()

# Menggabungkan isi dari txt1 dan txt2
merged_content = content1 + content2

# Menulis isi gabungan ke txt3
with open('generated-pantun-alpha.txt', 'w') as file3:
    file3.write(merged_content)
```

- ***Show pantun generate result***

```
result = []
with open('generated-pantun-alpha.txt', 'r') as f:
    lines = f.readlines()
    for line in lines:
        if line.startswith('<BOS>'):
            line_edit = line[5:]
            line_edit2 = line_edit.replace('  ', '')
            result.append(line_edit2.replace(r' \\n ', '\n'))

for idx, pantun in enumerate(result):
    print(' ===========\n  Pantun {}\n ===========\n'.format(idx))
    print(pantun)
```

- ***Install* Streamlit *Requirements & show IP for use* streamlit *in* Google Colabs**

```
!pip install streamlit
!pip install streamlit-chat
import requests

response = requests.get('https://ipv4.icanhazip.com')
public_ip = response.text.strip()

print("Public IP:", public_ip)
```

- ***Make* streamlit *web configuration***

```
!mkdir .streamlit
%cd .streamlit

# Make theme to dark
%%writefile config.toml
[theme]
base="dark"
```

```
# %cd ..
```

- ***Make* Streamlit *web script (app.py)* disini dilakukan olahan lanjutan sehingga didapat pantun yang sempurna**

```
%%writefile app.py
import streamlit as st
from streamlit_chat import message
import random

# Configuration
if 'batas' not in st.session_state:
    st.session_state.batas = 0

if 'batas2' not in st.session_state:
    st.session_state.batas2 = 0

result = []
with open('generated-pantun-alpha.txt', 'r') as f:
    lines = f.readlines()
    for line in lines:
        if line.startswith('<BOS>'):
            line_edit = line[5:]
            line_edit2 = line_edit.replace('  ', '')
            result.append(line_edit2.replace(r' \\n ', '\n'))


def check_akhiran_contoh(x):
  sajak1 = x[0][-1]
  sajak2 = x[1][-1]
  if sajak1 in ("a","i","u","e","o"):
    if sajak2 in ("a","i","u","e","o"):
      i3 = 1
      i4 = 1
      return sajak1, sajak2, i3, i4
    elif sajak2 in ('b', 'c', 'd', 'f', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', 'y', 'z'):
      sajak2 = x[1][-2:]
      i3 = 1
      i4 = 2
      return sajak1, sajak2, i3, i4
    else:
      sajak2 = x[1][-3:]
      i3 = 1
      i4 = 3
      return sajak1, sajak2, i3, i4
  elif sajak1 in ('b', 'c', 'd', 'f', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', 'y', 'z'):
```

```
    if sajak2 in ("a","i","u","e","o"):
      sajak1 = x[0][-2:]
      i3 = 2
      i4 = 1
      return sajak1, sajak2, i3, i4
    elif sajak2 in ('b', 'c', 'd', 'f', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', 'y', 'z'):
      sajak1 = x[0][-2:]
      sajak2 = x[1][-2:]
      i3 = 2
      i4 = 2
      return sajak1, sajak2, i3, i4
    else:
      sajak1 = x[0][-2:]
      sajak2 = x[1][-3:]
      i3 = 2
      i4 = 3
      return sajak1, sajak2, i3, i4
  else:
    if sajak2 in ("a","i","u","e","o"):
      sajak1 = x[0][-3:]
      i3 = 3
      i4 = 1
      return sajak1, sajak2, i3, i4
    elif sajak2 in ('b', 'c', 'd', 'f', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', 'y', 'z'):
      sajak1 = x[0][-3:]
      sajak2 = x[1][-2:]
      i3 = 3
      i4 = 2
      return sajak1, sajak2, i3, i4
    else:
      sajak1 = x[0][-3:]
      sajak2 = x[1][-3:]
      i3 = 3
      i4 = 3
      return sajak1, sajak2, i3, i4


def check_akhiran_prompt(x1,x2):
  sajak1 = x1[-1]
  sajak2 = x2[-1]
  if sajak1 in ("a","i","u","e","o"):
    if sajak2 in ("a","i","u","e","o"):
      i3 = 1
      i4 = 1
      return sajak1, sajak2, i3, i4
    elif sajak2 in ('b', 'c', 'd', 'f', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', 'y', 'z'):
      sajak2 = x2[-2:]
      i3 = 1
```

```
      i4 = 2
      return sajak1, sajak2, i3, i4
    else:
      sajak2 = x2[-3:]
      i3 = 1
      i4 = 3
      return sajak1, sajak2, i3, i4
  elif sajak1 in ('b', 'c', 'd', 'f', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', 'y', 'z'):
    if sajak2 in ("a","i","u","e","o"):
      sajak1 = x1[-2:]
      i3 = 2
      i4 = 1
      return sajak1, sajak2, i3, i4
    elif sajak2 in ('b', 'c', 'd', 'f', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', 'y', 'z'):
      sajak1 = x1[-2:]
      sajak2 = x2[-2:]
      i3 = 2
      i4 = 2
      return sajak1, sajak2, i3, i4
    else:
      sajak1 = x1[-2:]
      sajak2 = x2[-3:]
      i3 = 2
      i4 = 3
      return sajak1, sajak2, i3, i4
  else:
    if sajak2 in ("a","i","u","e","o"):
      sajak1 = x1[-3:]
      i3 = 3
      i4 = 1
      return sajak1, sajak2, i3, i4
    elif sajak2 in ('b', 'c', 'd', 'f', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', 'y', 'z'):
      sajak1 = x1[-3:]
      sajak2 = x2[-2:]
      i3 = 3
      i4 = 2
      return sajak1, sajak2, i3, i4
    else:
      sajak1 = x1[-3:]
      sajak2 = x2[-3:]
      i3 = 3
      i4 = 3
      return sajak1, sajak2, i3, i4

def on_input_change():
    batas = st.session_state.batas
    batas2 = st.session_state.batas2
    user_input = st.session_state.user_input
```

```
  st.session_state.past.append(user_input)
  inputan_split = user_input.split()
  print("")
  numbering = 1
  kata = ""
  baris_split = user_input.split("\n")

  if len(baris_split) == 2:
    if batas != 0 :
      baris_split = user_input.split("\n")
      baris3,baris4 = baris_split
      a, b, c, d = check_akhiran_prompt(baris3,baris4)
      for i in range(len(result)):
        pantun_ke = result[i].split('\n')
        if len(pantun_ke) >= 4 and pantun_ke[2] != "" and pantun_ke[3] != "":
          if pantun_ke[0][-c:] == a:
            if pantun_ke[1][-d:] == b:
              kata += (f"""===Pantun {numbering}===
{pantun_ke[0]}
{pantun_ke[1]}
{baris3}
{baris4}
""")
              numbering += 1
            # else:
            #   continue
        else:
          continue
    elif batas2 != 0 :
      baris_split = user_input.split("\n")
      baris1,baris2 = baris_split
      a, b, c, d = check_akhiran_prompt(baris1,baris2)
      for i in range(len(result)):
        pantun_ke = result[i].split('\n')
        # print(pantun_ke)
        if len(pantun_ke) >= 4 and pantun_ke[2] != "" and pantun_ke[3] != "":
          if pantun_ke[2][-c:] == a:
            if pantun_ke[3][-d:] == b:
              kata+=(f"""===Pantun {numbering}===
{baris1}
{baris2}
{pantun_ke[2]}
{pantun_ke[3]}
""")
              numbering += 1
              st.session_state.batas2 -= batas2
          #  else:
          #    continue
```

```
          # else:
          #   continue
        else:
          continue


    for xyz in inputan_split:
      if xyz.lower() == "contoh":
        flag = random.randint(0,len(result)-1)
        for i in range(1,len(result)):
          x = result[flag].split('\n')
          while True:
            if len(x) >= 4 and x[2] != "" and x[3] != "":
              break
            else:
              flag = random.randint(0,len(result)-1)
              x = result[flag].split('\n')
          a,b,c,d = check_akhiran_contoh(x)
          pantun_ke = result[i].split('\n')
          if len(pantun_ke) >= 4 and pantun_ke[2] != "" and pantun_ke[3] != "":
            if pantun_ke[2][-c:] == a:
              if pantun_ke[3][-d:] == b:
                kata += (f"""===Pantun {numbering}===
{x[0]}
{x[1]}
{pantun_ke[2]}
{pantun_ke[3]}

""")
                flag = random.randint(0,len(result)-1)
                numbering += 1
            #   else:
            #     continue
            # else:
            #   continue
            else:
              flag = random.randint(0,len(result)-1)
      elif xyz.lower() == "sampiran":
        st.session_state.batas += 1
        kata+=("Tolong berikan isi, nanti saya akan memberikan anda sampiran")
      elif xyz.lower() == "isi":
        st.session_state.batas2 += 1
        kata+=("Tolong berikan sampiran, nanti saya akan memberikan anda isi")
    st.session_state.generated.append(f"{kata}")


def on_btn_click():
    del st.session_state.past[:]
```

```
    del st.session_state.generated[:]


st.session_state.setdefault('past', [])
st.session_state.setdefault('generated', [])

st.title("PANTUN GENERATOR by Ricky Khairul Faza")

chat_placeholder = st.empty()

with chat_placeholder.container():
    for i in range(len(st.session_state['generated'])):
        message(st.session_state['past'][i], is_user=True, key=f"{i}_user")
        message(st.session_state['generated'][i], key=f"{i}")

    st.button("Bersihkan Pesan", on_click=on_btn_click)

with st.container():
    st.text_area("User Input:", on_change=on_input_change, key="user_input")
```

- ***(Optional) Delete fine-tuning model folder***

```
!rm -rf /content/gpt2-pantun
```

- **Isi dari *Hugging Face Transformers Python Package Script* run_language_modeling.py untuk *fine-tuning***

```
# coding=utf-8
# Copyright 2018 The Google AI Language Team Authors and The HuggingFace Inc. team.
# Copyright (c) 2018, NVIDIA CORPORATION.  All rights reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
"""
```

```
Fine-tuning the library models for language modeling on a text file (GPT, GPT-2, BERT, RoBERTa).
GPT and GPT-2 are fine-tuned using a causal language modeling (CLM) loss while BERT and RoBERTa are fine-tuned
using a masked language modeling (MLM) loss.
"""

import sys
import logging
import math
import os
from dataclasses import dataclass, field
from typing import Optional

from transformers import (
    CONFIG_MAPPING,
    MODEL_WITH_LM_HEAD_MAPPING,
    AutoConfig,
    AutoModelWithLMHead,
    AutoTokenizer,
    DataCollatorForLanguageModeling,
    HfArgumentParser,
    LineByLineTextDataset,
    PreTrainedTokenizer,
    TextDataset,
    Trainer,
    TrainingArguments,
    set_seed,
)


logger = logging.getLogger(__name__)


MODEL_CONFIG_CLASSES = list(MODEL_WITH_LM_HEAD_MAPPING.keys())
MODEL_TYPES = tuple(conf.model_type for conf in MODEL_CONFIG_CLASSES)


@dataclass
class ModelArguments:
    """
    Arguments pertaining to which model/config/tokenizer we are going to fine-tune, or train from scratch.
    """

    model_name_or_path: Optional[str] = field(
        default=None,
        metadata={
```

```
            "help": "The model checkpoint for weights initialization. Leave None if you want to train a model from scratch."
        },
    )
    model_type: Optional[str] = field(
        default=None,
        metadata={"help": "If training from scratch, pass a model type from the list: " + ", ".join(MODEL_TYPES)},
    )
    config_name: Optional[str] = field(
        default=None, metadata={"help": "Pretrained config name or path if not the same as model_name"}
    )
    tokenizer_name: Optional[str] = field(
        default=None, metadata={"help": "Pretrained tokenizer name or path if not the same as model_name"}
    )
    cache_dir: Optional[str] = field(
        default=None, metadata={"help": "Where do you want to store the pretrained models downloaded from s3"}
    )


@dataclass
class DataTrainingArguments:
    """
    Arguments pertaining to what data we are going to input our model for training and eval.
    """

    train_data_file: Optional[str] = field(
        default=None, metadata={"help": "The input training data file (a text file)."}
    )
    eval_data_file: Optional[str] = field(
        default=None,
        metadata={"help": "An optional input evaluation data file to evaluate the perplexity on (a text file)."},
    )
    line_by_line: bool = field(
        default=False,
        metadata={"help": "Whether distinct lines of text in the dataset are to be handled as distinct sequences."},
    )

    mlm: bool = field(
        default=False, metadata={"help": "Train with masked-language modeling loss instead of language modeling."}
    )
```

```
    mlm_probability: float = field(
        default=0.15, metadata={"help": "Ratio of tokens to mask for masked language modeling loss"}
    )

    block_size: int = field(
        default=-1,
        metadata={
            "help": "Optional input sequence length after tokenization."
            "The training dataset will be truncated in block of this size for training."
            "Default to the model max input length for single sentence inputs (take into account special tokens)."
        },
    )
    overwrite_cache: bool = field(
        default=False, metadata={"help": "Overwrite the cached training and evaluation sets"}
    )


def get_dataset(args: DataTrainingArguments, tokenizer: PreTrainedTokenizer, evaluate=False):
    file_path = args.eval_data_file if evaluate else args.train_data_file
    if args.line_by_line:
        return LineByLineTextDataset(tokenizer=tokenizer, file_path=file_path, block_size=args.block_size)
    else:
        return TextDataset(
            tokenizer=tokenizer, file_path=file_path, block_size=args.block_size, overwrite_cache=args.overwrite_cache
        )


def main():
    # To make it usable as kaggle script, skip commit run
    if len(sys.argv) == 1:
        return


    # See all possible arguments in src/transformers/training_args.py
    # or by passing the --help flag to this script.
    # We now keep distinct sets of args, for a cleaner separation of concerns.

    parser = HfArgumentParser((ModelArguments, DataTrainingArguments, TrainingArguments))
    model_args, data_args, training_args = parser.parse_args_into_dataclasses()

    if data_args.eval_data_file is None and training_args.do_eval:
```

```
        raise ValueError(
            "Cannot do evaluation without an evaluation data file. Either supply a file to --eval_data_file "
            "or remove the --do_eval argument."
        )

    if (
        os.path.exists(training_args.output_dir)
        and os.listdir(training_args.output_dir)
        and training_args.do_train
        and not training_args.overwrite_output_dir
    ):
        raise ValueError(
            f"Output directory ({training_args.output_dir}) already exists and is not empty. Use --overwrite_output_dir to overcome."
        )

    # Setup logging
    logging.basicConfig(
        format="%(asctime)s - %(levelname)s - %(name)s -   %(message)s",
        datefmt="%m/%d/%Y %H:%M:%S",
        level=logging.INFO if training_args.local_rank in [-1, 0] else logging.WARN,
    )
    logger.warning(
        "Process rank: %s, device: %s, n_gpu: %s, distributed training: %s, 16-bits training: %s",
        training_args.local_rank,
        training_args.device,
        training_args.n_gpu,
        bool(training_args.local_rank != -1),
        training_args.fp16,
    )
    logger.info("Training/evaluation parameters %s", training_args)

    # Set seed
    set_seed(training_args.seed)

    # Load pretrained model and tokenizer
    #
    # Distributed training:
    # The .from_pretrained methods guarantee that only one local process can concurrently
    # download model & vocab.

    if model_args.config_name:
        config = AutoConfig.from_pretrained(model_args.config_name, cache_dir=model_args.cache_dir)
    elif model_args.model_name_or_path:
```

```
        config = AutoConfig.from_pretrained(model_args.model_name_or_path, cache_dir=model_args.cache_dir)
    else:
        config = CONFIG_MAPPING[model_args.model_type]()
        logger.warning("You are instantiating a new config instance from scratch.")

    if model_args.tokenizer_name:
        tokenizer = AutoTokenizer.from_pretrained(model_args.tokenizer_name, cache_dir=model_args.cache_dir)
    elif model_args.model_name_or_path:
        tokenizer = AutoTokenizer.from_pretrained(model_args.model_name_or_path, cache_dir=model_args.cache_dir)
    else:
        raise ValueError(
            "You are instantiating a new tokenizer from scratch. This is not supported, but you can do it from another script, save it,"
            "and load it from here, using --tokenizer_name"
        )

    if model_args.model_name_or_path:
        model = AutoModelWithLMHead.from_pretrained(
            model_args.model_name_or_path,
            from_tf=bool(".ckpt" in model_args.model_name_or_path),
            config=config,
            cache_dir=model_args.cache_dir,
        )
    else:
        logger.info("Training new model from scratch")
        model = AutoModelWithLMHead.from_config(config)

    special_tokens_dict = {'bos_token': '<BOS>', 'eos_token': '<EOS>', 'pad_token': '<PAD>'}
    num_added_toks = tokenizer.add_special_tokens(special_tokens_dict)
    model.resize_token_embeddings(len(tokenizer))

    if config.model_type in ["bert", "roberta", "distilbert", "camembert"] and not data_args.mlm:
        raise ValueError(
            "BERT and RoBERTa-like models do not have LM heads but masked LM heads. They must be run using the --mlm "
            "flag (masked language modeling)."
        )

    if data_args.block_size <= 0:
        data_args.block_size = tokenizer.max_len
        # Our input block size will be the max possible for the model
    else:
        data_args.block_size = min(data_args.block_size, tokenizer.max_len)
```

```
    # Get datasets

    train_dataset = get_dataset(data_args, tokenizer=tokenizer) if training_args.do_train
else None
    eval_dataset = get_dataset(data_args, tokenizer=tokenizer, evaluate=True) if
training_args.do_eval else None
    data_collator = DataCollatorForLanguageModeling(
        tokenizer=tokenizer, mlm=data_args.mlm,
mlm_probability=data_args.mlm_probability
    )

    # Initialize our Trainer
    trainer = Trainer(
        model=model,
        args=training_args,
        data_collator=data_collator,
        train_dataset=train_dataset,
        eval_dataset=eval_dataset,
        prediction_loss_only=True,
    )

    # Training
    if training_args.do_train:
        model_path = (
            model_args.model_name_or_path
            if model_args.model_name_or_path is not None and
os.path.isdir(model_args.model_name_or_path)
            else None
        )
        trainer.train(model_path=model_path)
        trainer.save_model()
        # For convenience, we also re-save the tokenizer to the same directory,
        # so that you can share your model easily on huggingface.co/models =)
        if trainer.is_world_master():
            tokenizer.save_pretrained(training_args.output_dir)

    # Evaluation
    results = {}
    if training_args.do_eval:
        logger.info("*** Evaluate ***")

        eval_output = trainer.evaluate()

        perplexity = math.exp(eval_output["eval_loss"])
        result = {"perplexity": perplexity}

        output_eval_file = os.path.join(training_args.output_dir, "eval_results_lm.txt")
```

```
        if trainer.is_world_master():
            with open(output_eval_file, "w") as writer:
                logger.info("***** Eval results *****")
                for key in sorted(result.keys()):
                    logger.info("  %s = %s", key, str(result[key]))
                    writer.write("%s = %s\n" % (key, str(result[key])))

        results.update(result)

    return results


def _mp_fn(index):
    # For xla_spawn (TPUs)
    main()


if __name__ == "__main__":
    main()
```

- **Isi dari *Hugging Face Transformers Python Package Script* run_generation.py untuk *generate* pantun saat pengujian model**

```
#!/usr/bin/env python3
# coding=utf-8
# Copyright 2018 Google AI, Google Brain and Carnegie Mellon University Authors and the HuggingFace Inc. team.
# Copyright (c) 2018, NVIDIA CORPORATION.  All rights reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
""" Conditional text generation with the auto-regressive models of the library (GPT/GPT-2/CTRL/Transformer-XL/XLNet)
"""
```

```
import sys
import argparse
import logging

import numpy as np
import torch

from transformers import (
    CTRLLMHeadModel,
    CTRLTokenizer,
    GPT2LMHeadModel,
    GPT2Tokenizer,
    OpenAIGPTLMHeadModel,
    OpenAIGPTTokenizer,
    TransfoXLLMHeadModel,
    TransfoXLTokenizer,
    XLMTokenizer,
    XLMWithLMHeadModel,
    XLNetLMHeadModel,
    XLNetTokenizer,
)


logging.basicConfig(
    format="%(asctime)s - %(levelname)s - %(name)s - %(message)s",
datefmt="%m/%d/%Y %H:%M:%S", level=logging.INFO,
)
logger = logging.getLogger(__name__)

MAX_LENGTH = int(10000)  # Hardcoded max length to avoid infinite loop

MODEL_CLASSES = {
    "gpt2": (GPT2LMHeadModel, GPT2Tokenizer),
    "ctrl": (CTRLLMHeadModel, CTRLTokenizer),
    "openai-gpt": (OpenAIGPTLMHeadModel, OpenAIGPTTokenizer),
    "xlnet": (XLNetLMHeadModel, XLNetTokenizer),
    "transfo-xl": (TransfoXLLMHeadModel, TransfoXLTokenizer),
    "xlm": (XLMWithLMHeadModel, XLMTokenizer),
}

# Padding text to help Transformer-XL and XLNet with short prompts as proposed by
Aman Rusia
# in https://github.com/rusiaaman/XLNet-gen#methodology
# and https://medium.com/@amanrusia/xlnet-speaks-comparison-to-gpt-2-ea1a4e9ba39e
PADDING_TEXT = """In 1991, the remains of Russian Tsar Nicholas II and his family
(except for Alexei and Maria) are discovered.
The voice of Nicholas's young son, Tsarevich Alexei Nikolaevich, narrates the
remainder of the story. 1883 Western Siberia,
```

```
a young Grigori Rasputin is asked by his father and a group of men to perform magic.
Rasputin has a vision and denounces one of the men as a horse thief. Although his
father initially slaps him for making such an accusation, Rasputin watches as the
man is chased outside and beaten. Twenty years later, Rasputin sees a vision of
the Virgin Mary, prompting him to become a priest. Rasputin quickly becomes famous,
with people, even a bishop, begging for his blessing. <eod> </s> <eos>"""


def set_seed(args):
    np.random.seed(args.seed)
    torch.manual_seed(args.seed)
    if args.n_gpu > 0:
        torch.cuda.manual_seed_all(args.seed)


#
# Functions to prepare models' input
#


def prepare_ctrl_input(args, _, tokenizer, prompt_text):
    if args.temperature > 0.7:
        logger.info("CTRL typically works better with lower temperatures (and lower top_k).")

    encoded_prompt = tokenizer.encode(prompt_text, add_special_tokens=False)
    if not any(encoded_prompt[0] == x for x in tokenizer.control_codes.values()):
        logger.info("WARNING! You are not starting your generation from a control code so you won't get good results")
    return prompt_text


def prepare_xlm_input(args, model, tokenizer, prompt_text):
    # kwargs = {"language": None, "mask_token_id": None}

    # Set the language
    use_lang_emb = hasattr(model.config, "use_lang_emb") and model.config.use_lang_emb
    if hasattr(model.config, "lang2id") and use_lang_emb:
        available_languages = model.config.lang2id.keys()
        if args.xlm_language in available_languages:
            language = args.xlm_language
        else:
            language = None
            while language not in available_languages:
                language = input("Using XLM. Select language in " + str(list(available_languages)) + " >>> ")
```

```
        model.config.lang_id = model.config.lang2id[language]
        # kwargs["language"] = tokenizer.lang2id[language]

    # TODO fix mask_token_id setup when configurations will be synchronized between
models and tokenizers
    # XLM masked-language modeling (MLM) models need masked token
    # is_xlm_mlm = "mlm" in args.model_name_or_path
    # if is_xlm_mlm:
    #     kwargs["mask_token_id"] = tokenizer.mask_token_id

    return prompt_text


def prepare_xlnet_input(args, _, tokenizer, prompt_text):
    prompt_text = (args.padding_text if args.padding_text else PADDING_TEXT) +
prompt_text
    return prompt_text


def prepare_transfoxl_input(args, _, tokenizer, prompt_text):
    prompt_text = (args.padding_text if args.padding_text else PADDING_TEXT) +
prompt_text
    return prompt_text


PREPROCESSING_FUNCTIONS = {
    "ctrl": prepare_ctrl_input,
    "xlm": prepare_xlm_input,
    "xlnet": prepare_xlnet_input,
    "transfo-xl": prepare_transfoxl_input,
}


def adjust_length_to_model(length, max_sequence_length):
    if length < 0 and max_sequence_length > 0:
        length = max_sequence_length
    elif 0 < max_sequence_length < length:
        length = max_sequence_length  # No generation bigger than model size
    elif length < 0:
        length = MAX_LENGTH  # avoid infinite loop
    return length


def main():
    # To make it usable as kaggle script, skip commit run
    if len(sys.argv) == 1:
        return
```

```
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--model_type",
        default=None,
        type=str,
        required=True,
        help="Model type selected in the list: " + ", ".join(MODEL_CLASSES.keys()),
    )
    parser.add_argument(
        "--model_name_or_path",
        default=None,
        type=str,
        required=True,
        help="Path to pre-trained model or shortcut name selected in the list: " + ", ".join(MODEL_CLASSES.keys()),
    )

    parser.add_argument("--prompt", type=str, default="")
    parser.add_argument("--length", type=int, default=20)
    parser.add_argument("--stop_token", type=str, default=None, help="Token at which text generation is stopped")

    parser.add_argument(
        "--temperature",
        type=float,
        default=1.0,
        help="temperature of 1.0 has no effect, lower tend toward greedy sampling",
    )
    parser.add_argument(
        "--repetition_penalty", type=float, default=1.0, help="primarily useful for CTRL model; in that case, use 1.2"
    )
    parser.add_argument("--k", type=int, default=0)
    parser.add_argument("--p", type=float, default=0.9)

    parser.add_argument("--padding_text", type=str, default="", help="Padding text for Transfo-XL and XLNet.")
    parser.add_argument("--xlm_language", type=str, default="", help="Optional language when used with the XLM model.")

    parser.add_argument("--seed", type=int, default=42, help="random seed for initialization")
    parser.add_argument("--no_cuda", action="store_true", help="Avoid using CUDA when available")
    parser.add_argument("--num_return_sequences", type=int, default=1, help="The number of samples to generate.")
    args = parser.parse_args()
```

```
    args.device = torch.device("cuda" if torch.cuda.is_available() and not args.no_cuda else "cpu")
    args.n_gpu = 0 if args.no_cuda else torch.cuda.device_count()

    set_seed(args)

    # Initialize the model and tokenizer
    try:
        args.model_type = args.model_type.lower()
        model_class, tokenizer_class = MODEL_CLASSES[args.model_type]
    except KeyError:
        raise KeyError("the model {} you specified is not supported. You are welcome to add it and open a PR :)")

    tokenizer = tokenizer_class.from_pretrained(args.model_name_or_path)
    model = model_class.from_pretrained(args.model_name_or_path)
    model.to(args.device)

    args.length = adjust_length_to_model(args.length, max_sequence_length=model.config.max_position_embeddings)
    logger.info(args)

    prompt_text = args.prompt if args.prompt else input("Model prompt >>> ")

    # Different models need different input formatting and/or extra arguments
    requires_preprocessing = args.model_type in PREPROCESSING_FUNCTIONS.keys()
    if requires_preprocessing:
        prepare_input = PREPROCESSING_FUNCTIONS.get(args.model_type)
        preprocessed_prompt_text = prepare_input(args, model, tokenizer, prompt_text)
        encoded_prompt = tokenizer.encode(
            preprocessed_prompt_text, add_special_tokens=False, return_tensors="pt", add_space_before_punct_symbol=True
        )
    else:
        encoded_prompt = tokenizer.encode(prompt_text, add_special_tokens=True, return_tensors="pt")
    encoded_prompt = encoded_prompt.to(args.device)

    if encoded_prompt.size()[-1] == 0:
        input_ids = None
    else:
        input_ids = encoded_prompt

    output_sequences = model.generate(
        input_ids=input_ids,
        max_length=args.length + len(encoded_prompt[0]),
        temperature=args.temperature,
```

```
        top_k=args.k,
        top_p=args.p,
        repetition_penalty=args.repetition_penalty,
        do_sample=True,
        num_return_sequences=args.num_return_sequences,
    )

    # Remove the batch dimension when returning multiple sequences
    if len(output_sequences.shape) > 2:
        output_sequences.squeeze_()

    generated_sequences = []

    for generated_sequence_idx, generated_sequence in enumerate(output_sequences):
        print("=== GENERATED SEQUENCE {} ===".format(generated_sequence_idx +
1))
        generated_sequence = generated_sequence.tolist()

        # Decode text
        text = tokenizer.decode(generated_sequence, clean_up_tokenization_spaces=True)

        # Remove all text after the stop token
        text = text[: text.find(args.stop_token) if args.stop_token else None]

        # Add the prompt at the beginning of the sequence. Remove the excess text that was
used for pre-processing
        total_sequence = (
            prompt_text + text[len(tokenizer.decode(encoded_prompt[0],
clean_up_tokenization_spaces=True)) :]
        )

        generated_sequences.append(total_sequence)
        print(total_sequence)

    return generated_sequences


if __name__ == "__main__":
    main()
```

- **Uji Plagiasi**

INSTITUT TEKNOLOGI INDONESIA

Jl. Raya Puspiptek, Tangerang Selatan - 15314
(021) 7562757
www.iti.ac.id institutteknologiindonesia @kampusITI Institut Teknologi Indonesia

SURAT KETERANGAN
0976/SKCP/PERPUST-ITI/2023

Yang bertanda tangan di bawah ini menerangkan bahwa:

Nama Mahasiswa	: **Ricky Khairul Faza**
Nomor Identitas	: **1151900034**
Status Pemohon	: **Mahasiswa**

Telah menyerahkan dokumen uji plagiasi kepada Perpustakaan Institut Teknologi Indonesia dengan judul sebagai berikut:

Implementasi Generative Pre-Trained Transformers Sebagai Pantun Generator

Berdasarkan hasil pengecekan dokumen dinyatakan persentase kemiripan dokumen di atas adalah sebesar 24 **%**.

Demikian kami sampaikan untuk dapat digunakan sebagaimana mestinya.

Tangerang Selatan, 26 September 2023
Petugas Perpustakaan
Institut Teknologi Indonesia

PERPUSTAKAAN

M. Ichfan Alawi, S.IP

Implementasi Generative Pre-Trained Transformers Sebagai Pantun Generator

ORIGINALITY REPORT

24%	23%	7%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	repository.iti.ac.id Internet Source	2%
2	docplayer.info Internet Source	1%
3	repository.uin-suska.ac.id Internet Source	1%
4	Submitted to Universitas Pamulang Student Paper	1%
5	123dok.com Internet Source	1%
6	glints.com Internet Source	1%
7	jurnal.atmaluhur.ac.id Internet Source	1%
8	producthp.blogspot.com Internet Source	1%
9	eprints.ums.ac.id Internet Source	1%

10	files.osf.io Internet Source	1 %
11	Submitted to Fakultas Ekonomi Universitas Indonesia Student Paper	1 %
12	repository.nurulfikri.ac.id Internet Source	1 %
13	repository.its.ac.id Internet Source	1 %
14	aclanthology.org Internet Source	1 %
15	huggingface.co Internet Source	<1 %
16	ceritaihsan.com Internet Source	<1 %
17	core.ac.uk Internet Source	<1 %
18	id.123dok.com Internet Source	<1 %
19	es.scribd.com Internet Source	<1 %
20	phob13.wordpress.com Internet Source	<1 %
21	www.scribd.com Internet Source	

		<1 %
22	repository.unsri.ac.id Internet Source	<1 %
23	text-id.123dok.com Internet Source	<1 %
24	oto.iti.ac.id Internet Source	<1 %
25	seputarilmu.com Internet Source	<1 %
26	widuri.raharja.info Internet Source	<1 %
27	www.sciencegate.app Internet Source	<1 %
28	repository.stieipwija.ac.id Internet Source	<1 %
29	repository.umj.ac.id Internet Source	<1 %
30	algorit.ma Internet Source	<1 %
31	repository.ub.ac.id Internet Source	<1 %
32	eprints.undip.ac.id Internet Source	<1 %

33	arxiv.org Internet Source	<1 %
34	file.upi.edu Internet Source	<1 %
35	informatics.uii.ac.id Internet Source	<1 %
36	repository.ppns.ac.id Internet Source	<1 %
37	staging2.dqlab.id Internet Source	<1 %
38	dspace.uii.ac.id Internet Source	<1 %
39	eprints.poltekkesjogja.ac.id Internet Source	<1 %
40	id.scribd.com Internet Source	<1 %
41	www.stuffspec.com Internet Source	<1 %
42	digilib.uinsby.ac.id Internet Source	<1 %
43	www.researchgate.net Internet Source	<1 %
44	adoc.pub Internet Source	<1 %

45	etheses.uin-malang.ac.id Internet Source	<1 %
46	garuda.kemdikbud.go.id Internet Source	<1 %
47	hashdork.com Internet Source	<1 %
48	www.topbots.com Internet Source	<1 %
49	Submitted to Universidad de Chile Student Paper	<1 %
50	digilib.esaunggul.ac.id Internet Source	<1 %
51	jantungmelayu.com Internet Source	<1 %
52	nanikhermin.blogspot.com Internet Source	<1 %
53	Submitted to Heriot-Watt University Student Paper	<1 %
54	repository.usu.ac.id Internet Source	<1 %
55	jurnal.polibatam.ac.id Internet Source	<1 %
56	library.binus.ac.id Internet Source	<1 %

57	pdfs.semanticscholar.org Internet Source	<1 %
58	repository.ar-raniry.ac.id Internet Source	<1 %
59	www.skema.edu Internet Source	<1 %
60	Submitted to University of Surrey Student Paper	<1 %
61	github.com Internet Source	<1 %
62	ichi.pro Internet Source	<1 %
63	repository.umsu.ac.id Internet Source	<1 %
64	repository.usd.ac.id Internet Source	<1 %
65	Fika Hastarita Rachman, Imamah Imamah. "Pendekatan Data Science untuk Mengukur Empati Masyarakat terhadap Pandemi Menggunakan Analisis Sentimen dan Seleksi Fitur", Jurnal Edukasi dan Penelitian Informatika (JEPIN), 2022 Publication	<1 %
66	Submitted to STIKOM Surabaya Student Paper	<1 %

67	Submitted to Universitas Muhammadiyah Surakarta Student Paper	<1 %
68	anzdoc.com Internet Source	<1 %
69	fkkumj.ac.id Internet Source	<1 %
70	link.springer.com Internet Source	<1 %
71	mafiadoc.com Internet Source	<1 %
72	repository.uinjkt.ac.id Internet Source	<1 %
73	repository.unej.ac.id Internet Source	<1 %
74	begawe.unram.ac.id Internet Source	<1 %
75	journal.maranatha.edu Internet Source	<1 %
76	journal.pancabudi.ac.id Internet Source	<1 %
77	jurnal.unimed.ac.id Internet Source	<1 %
78	pelajarancg.blogspot.com Internet Source	<1 %

79	sads.instiki.ac.id Internet Source	<1 %
80	smart.stmikplk.ac.id Internet Source	<1 %
81	www.slideshare.net Internet Source	<1 %
82	123docz.net Internet Source	<1 %
83	nugieanggipurwanto.blogspot.com Internet Source	<1 %
84	pt.slideshare.net Internet Source	<1 %
85	repository.unhas.ac.id Internet Source	<1 %
86	www.mdpi.com Internet Source	<1 %
87	docshare.tips Internet Source	<1 %
88	edoc.site Internet Source	<1 %
89	eprintslib.ummgl.ac.id Internet Source	<1 %
90	infobankterbaru.blogspot.com Internet Source	<1 %

91	kar.kent.ac.uk Internet Source	<1 %
92	lib.ui.ac.id Internet Source	<1 %
93	library.universitaspertamina.ac.id Internet Source	<1 %
94	sayutialfarizi.blogspot.com Internet Source	<1 %
95	zephyrnet.com Internet Source	<1 %
96	Michael Henry Tessler, Noah D. Goodman. "Warm (for Winter): Inferring Comparison Classes in Communication", Cognitive Science, 2022 Publication	<1 %
97	Yaiza Serrano, Sergi Roda, Victor Guallar, Alexis Molina. "Efficient and accurate sequence generation with small-scale protein language models", Cold Spring Harbor Laboratory, 2023 Publication	<1 %
98	Helena Nurramdhani Irmanda, Ria Astriratma, Nurul Chamidah, Mayanda Mega Santoni. "Pembuat Sampiran Pantun Otomatis berbasis Pattern-matching", Jurnal	<1 %

Sisfokom (Sistem Informasi dan Komputer), 2021
Publication

99	katakatasmsyoko.blogspot.com Internet Source	<1 %
100	kc.umn.ac.id Internet Source	<1 %
101	repository.uki.ac.id Internet Source	<1 %

Exclude quotes	On	Exclude matches	< 1 words
Exclude bibliography	On		