In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries
installed
# It is defined by the kaggle/python Docker image: https://github.com/ka
ggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) wi
ll list all files under the input directory

import os
# for dirname, _, filenames in os.walk('/kaggle/input'):
#     for filename in filenames:
#         print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) t
hat gets preserved as output when you create a version using "Save & Run
All"
# You can also write temporary files to /kaggle/temp/, but they won't be
saved outside of the current session
```

In [2]:
```
!pip install PyMuPDF
!pip install python-Levenshtein
!pip install editdistance
```

```
Collecting PyMuPDF
  Downloading pymupdf-1.25.4-cp39-abi3-manylinux2014_x86_64.manylin
ux_2_17_x86_64.whl.metadata (3.4 kB)
Downloading pymupdf-1.25.4-cp39-abi3-manylinux2014_x86_64.manylinux
_2_17_x86_64.whl (20.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20.0/20.0 MB 86.3 MB/s eta
0:00:00
Installing collected packages: PyMuPDF
Successfully installed PyMuPDF-1.25.4
Collecting python-Levenshtein
  Downloading python_levenshtein-0.27.1-py3-none-any.whl.metadata
(3.7 kB)
Collecting Levenshtein==0.27.1 (from python-Levenshtein)
  Downloading levenshtein-0.27.1-cp310-cp310-manylinux_2_17_x86_64.
manylinux2014_x86_64.whl.metadata (3.6 kB)
Collecting rapidfuzz<4.0.0,>=3.9.0 (from Levenshtein==0.27.1->pytho
n-Levenshtein)
  Downloading rapidfuzz-3.12.2-cp310-cp310-manylinux_2_17_x86_64.ma
nylinux2014_x86_64.whl.metadata (12 kB)
Downloading python_levenshtein-0.27.1-py3-none-any.whl (9.4 kB)
Downloading levenshtein-0.27.1-cp310-cp310-manylinux_2_17_x86_64.ma
nylinux2014_x86_64.whl (161 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 161.5/161.5 kB 10.3 MB/s et
a 0:00:00
Downloading rapidfuzz-3.12.2-cp310-cp310-manylinux_2_17_x86_64.many
linux2014_x86_64.whl (3.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/3.1 MB 86.7 MB/s eta 0:
00:00
Installing collected packages: rapidfuzz, Levenshtein, python-Leven
shtein
Successfully installed Levenshtein-0.27.1 python-Levenshtein-0.27.1
rapidfuzz-3.12.2
Requirement already satisfied: editdistance in /usr/local/lib/pytho
n3.10/dist-packages (0.8.1)
```

In [3]:
```
!pip install scikit-image
!pip install python-docx
!pip install transformers[torch]
!pip install accelerate -U
!pip install evaluate
!pip install jiwer
```

```
Requirement already satisfied: scikit-image in /usr/local/lib/pytho
n3.10/dist-packages (0.25.0)
Requirement already satisfied: numpy>=1.24 in /usr/local/lib/python
3.10/dist-packages (from scikit-image) (1.26.4)
Requirement already satisfied: scipy>=1.11.2 in /usr/local/lib/pyth
on3.10/dist-packages (from scikit-image) (1.13.1)
Requirement already satisfied: networkx>=3.0 in /usr/local/lib/pyth
on3.10/dist-packages (from scikit-image) (3.4.2)
Requirement already satisfied: pillow>=10.1 in /usr/local/lib/pytho
n3.10/dist-packages (from scikit-image) (11.0.0)
Requirement already satisfied: imageio!=2.35.0,>=2.33 in /usr/loca
l/lib/python3.10/dist-packages (from scikit-image) (2.36.1)
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/li
b/python3.10/dist-packages (from scikit-image) (2024.12.12)
Requirement already satisfied: packaging>=21 in /usr/local/lib/pyth
on3.10/dist-packages (from scikit-image) (24.2)
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/p
ython3.10/dist-packages (from scikit-image) (0.4)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.1
0/dist-packages (from numpy>=1.24->scikit-image) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python
3.10/dist-packages (from numpy>=1.24->scikit-image) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.
10/dist-packages (from numpy>=1.24->scikit-image) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.10/dis
t-packages (from numpy>=1.24->scikit-image) (2025.0.1)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.10/
dist-packages (from numpy>=1.24->scikit-image) (2022.0.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python
3.10/dist-packages (from numpy>=1.24->scikit-image) (2.4.1)
Requirement already satisfied: intel-openmp>=2024 in /usr/local/li
b/python3.10/dist-packages (from mkl->numpy>=1.24->scikit-image) (2
024.2.0)
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python
3.10/dist-packages (from mkl->numpy>=1.24->scikit-image) (2022.0.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python
3.10/dist-packages (from tbb==2022.*->mkl->numpy>=1.24->scikit-imag
e) (1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in /usr/local/li
b/python3.10/dist-packages (from mkl_umath->numpy>=1.24->scikit-ima
ge) (2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in /us
r/local/lib/python3.10/dist-packages (from intel-openmp>=2024->mkl-
>numpy>=1.24->scikit-image) (2024.2.0)
Collecting python-docx
  Downloading python_docx-1.1.2-py3-none-any.whl.metadata (2.0 kB)
Requirement already satisfied: lxml>=3.1.0 in /usr/local/lib/python
3.10/dist-packages (from python-docx) (5.3.0)
Requirement already satisfied: typing-extensions>=4.9.0 in /usr/loc
al/lib/python3.10/dist-packages (from python-docx) (4.12.2)
Downloading python_docx-1.1.2-py3-none-any.whl (244 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 244.3/244.3 kB 12.6 MB/s et
a 0:00:00
Installing collected packages: python-docx
Successfully installed python-docx-1.1.2
Requirement already satisfied: transformers[torch] in /usr/local/li
b/python3.10/dist-packages (4.47.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.1
0/dist-packages (from transformers[torch]) (3.17.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in /us
r/local/lib/python3.10/dist-packages (from transformers[torch]) (0.
29.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python
3.10/dist-packages (from transformers[torch]) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/py
thon3.10/dist-packages (from transformers[torch]) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python
3.10/dist-packages (from transformers[torch]) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/
python3.10/dist-packages (from transformers[torch]) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.1
0/dist-packages (from transformers[torch]) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/loca
l/lib/python3.10/dist-packages (from transformers[torch]) (0.21.0)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/li
b/python3.10/dist-packages (from transformers[torch]) (0.4.5)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python
3.10/dist-packages (from transformers[torch]) (4.67.1)
Requirement already satisfied: torch in /usr/local/lib/python3.10/d
ist-packages (from transformers[torch]) (2.5.1+cu121)
Requirement already satisfied: accelerate>=0.26.0 in /usr/local/li
b/python3.10/dist-packages (from transformers[torch]) (1.2.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/
dist-packages (from accelerate>=0.26.0->transformers[torch]) (5.9.
5)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/p
ython3.10/dist-packages (from huggingface-hub<1.0,>=0.24.0->transfo
rmers[torch]) (2024.12.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/l
ocal/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.24.
0->transformers[torch]) (4.12.2)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.1
0/dist-packages (from numpy>=1.17->transformers[torch]) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python
3.10/dist-packages (from numpy>=1.17->transformers[torch]) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.
10/dist-packages (from numpy>=1.17->transformers[torch]) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.10/dis
t-packages (from numpy>=1.17->transformers[torch]) (2025.0.1)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.10/
dist-packages (from numpy>=1.17->transformers[torch]) (2022.0.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python
3.10/dist-packages (from numpy>=1.17->transformers[torch]) (2.4.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.1
0/dist-packages (from torch->transformers[torch]) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/
dist-packages (from torch->transformers[torch]) (3.1.4)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/pyth
on3.10/dist-packages (from torch->transformers[torch]) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/li
b/python3.10/dist-packages (from sympy==1.13.1->torch->transformers
[torch]) (1.3.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/loc
al/lib/python3.10/dist-packages (from requests->transformers[torc
h]) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/pytho
n3.10/dist-packages (from requests->transformers[torch]) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/li
b/python3.10/dist-packages (from requests->transformers[torch]) (2.
3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/li
b/python3.10/dist-packages (from requests->transformers[torch]) (20
25.1.31)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/py
thon3.10/dist-packages (from jinja2->torch->transformers[torch])
(3.0.2)
Requirement already satisfied: intel-openmp>=2024 in /usr/local/li
b/python3.10/dist-packages (from mkl->numpy>=1.17->transformers[tor
ch]) (2024.2.0)
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python
3.10/dist-packages (from mkl->numpy>=1.17->transformers[torch]) (20
22.0.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python
3.10/dist-packages (from tbb==2022.*->mkl->numpy>=1.17->transformer
s[torch]) (1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in /usr/local/li
b/python3.10/dist-packages (from mkl_umath->numpy>=1.17->transforme
rs[torch]) (2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in /us
r/local/lib/python3.10/dist-packages (from intel-openmp>=2024->mkl-
>numpy>=1.17->transformers[torch]) (2024.2.0)
Requirement already satisfied: accelerate in /usr/local/lib/python
3.10/dist-packages (1.2.1)
Collecting accelerate
  Downloading accelerate-1.5.2-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: numpy<3.0.0,>=1.17 in /usr/local/li
b/python3.10/dist-packages (from accelerate) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/py
thon3.10/dist-packages (from accelerate) (24.2)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/
dist-packages (from accelerate) (5.9.5)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/
dist-packages (from accelerate) (6.0.2)
Requirement already satisfied: torch>=2.0.0 in /usr/local/lib/pytho
n3.10/dist-packages (from accelerate) (2.5.1+cu121)
Requirement already satisfied: huggingface-hub>=0.21.0 in /usr/loca
l/lib/python3.10/dist-packages (from accelerate) (0.29.0)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/li
b/python3.10/dist-packages (from accelerate) (0.4.5)
Requirement already satisfied: filelock in /usr/local/lib/python3.1
0/dist-packages (from huggingface-hub>=0.21.0->accelerate) (3.17.0)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/p
ython3.10/dist-packages (from huggingface-hub>=0.21.0->accelerate)
(2024.12.0)
Requirement already satisfied: requests in /usr/local/lib/python3.1
0/dist-packages (from huggingface-hub>=0.21.0->accelerate) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/pytho
n3.10/dist-packages (from huggingface-hub>=0.21.0->accelerate) (4.6
7.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/l
ocal/lib/python3.10/dist-packages (from huggingface-hub>=0.21.0->ac
celerate) (4.12.2)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.1
0/dist-packages (from numpy<3.0.0,>=1.17->accelerate) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python
```

```
3.10/dist-packages (from numpy<3.0.0,>=1.17->accelerate) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.
10/dist-packages (from numpy<3.0.0,>=1.17->accelerate) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.10/dis
t-packages (from numpy<3.0.0,>=1.17->accelerate) (2025.0.1)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.10/
dist-packages (from numpy<3.0.0,>=1.17->accelerate) (2022.0.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python
3.10/dist-packages (from numpy<3.0.0,>=1.17->accelerate) (2.4.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.1
0/dist-packages (from torch>=2.0.0->accelerate) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/
dist-packages (from torch>=2.0.0->accelerate) (3.1.4)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/pyth
on3.10/dist-packages (from torch>=2.0.0->accelerate) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/li
b/python3.10/dist-packages (from sympy==1.13.1->torch>=2.0.0->accel
erate) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/py
thon3.10/dist-packages (from jinja2->torch>=2.0.0->accelerate) (3.
0.2)
Requirement already satisfied: intel-openmp>=2024 in /usr/local/li
b/python3.10/dist-packages (from mkl->numpy<3.0.0,>=1.17->accelerat
e) (2024.2.0)
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python
3.10/dist-packages (from mkl->numpy<3.0.0,>=1.17->accelerate) (202
2.0.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python
3.10/dist-packages (from tbb==2022.*->mkl->numpy<3.0.0,>=1.17->acce
lerate) (1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in /usr/local/li
b/python3.10/dist-packages (from mkl_umath->numpy<3.0.0,>=1.17->acc
elerate) (2024.2.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/loc
al/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.
21.0->accelerate) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/pytho
n3.10/dist-packages (from requests->huggingface-hub>=0.21.0->accele
rate) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/li
b/python3.10/dist-packages (from requests->huggingface-hub>=0.21.0-
>accelerate) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/li
b/python3.10/dist-packages (from requests->huggingface-hub>=0.21.0-
>accelerate) (2025.1.31)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in /us
r/local/lib/python3.10/dist-packages (from intel-openmp>=2024->mkl-
>numpy<3.0.0,>=1.17->accelerate) (2024.2.0)
Downloading accelerate-1.5.2-py3-none-any.whl (345 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 345.1/345.1 kB 1.6 MB/s eta
0:00:00
Installing collected packages: accelerate
  Attempting uninstall: accelerate
    Found existing installation: accelerate 1.2.1
    Uninstalling accelerate-1.2.1:
      Successfully uninstalled accelerate-1.2.1
Successfully installed accelerate-1.5.2
Collecting evaluate
  Downloading evaluate-0.4.3-py3-none-any.whl.metadata (9.2 kB)
Requirement already satisfied: datasets>=2.0.0 in /usr/local/lib/py
thon3.10/dist-packages (from evaluate) (3.3.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python
3.10/dist-packages (from evaluate) (1.26.4)
Requirement already satisfied: dill in /usr/local/lib/python3.10/di
st-packages (from evaluate) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/
dist-packages (from evaluate) (2.2.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/p
ython3.10/dist-packages (from evaluate) (2.32.3)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/pytho
n3.10/dist-packages (from evaluate) (4.67.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/
dist-packages (from evaluate) (3.5.0)
Requirement already satisfied: multiprocess in /usr/local/lib/pytho
n3.10/dist-packages (from evaluate) (0.70.16)
Requirement already satisfied: fsspec>=2021.05.0 in /usr/local/lib/
python3.10/dist-packages (from fsspec[http]>=2021.05.0->evaluate)
(2024.12.0)
Requirement already satisfied: huggingface-hub>=0.7.0 in /usr/loca
l/lib/python3.10/dist-packages (from evaluate) (0.29.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.
10/dist-packages (from evaluate) (24.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.1
0/dist-packages (from datasets>=2.0.0->evaluate) (3.17.0)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/py
thon3.10/dist-packages (from datasets>=2.0.0->evaluate) (19.0.1)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.1
0/dist-packages (from datasets>=2.0.0->evaluate) (3.11.12)
```

```
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python
3.10/dist-packages (from datasets>=2.0.0->evaluate) (6.0.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/l
ocal/lib/python3.10/dist-packages (from huggingface-hub>=0.7.0->eva
luate) (4.12.2)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.1
0/dist-packages (from numpy>=1.17->evaluate) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python
3.10/dist-packages (from numpy>=1.17->evaluate) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.
10/dist-packages (from numpy>=1.17->evaluate) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.10/dis
t-packages (from numpy>=1.17->evaluate) (2025.0.1)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.10/
dist-packages (from numpy>=1.17->evaluate) (2022.0.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python
3.10/dist-packages (from numpy>=1.17->evaluate) (2.4.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/loc
al/lib/python3.10/dist-packages (from requests>=2.19.0->evaluate)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/pytho
n3.10/dist-packages (from requests>=2.19.0->evaluate) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/li
b/python3.10/dist-packages (from requests>=2.19.0->evaluate) (2.3.
0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/li
b/python3.10/dist-packages (from requests>=2.19.0->evaluate) (2025.
1.31)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/loca
l/lib/python3.10/dist-packages (from pandas->evaluate) (2.9.0.post
0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/pytho
n3.10/dist-packages (from pandas->evaluate) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/pyt
hon3.10/dist-packages (from pandas->evaluate) (2025.1)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/loca
l/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->eval
uate) (2.4.6)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/p
ython3.10/dist-packages (from aiohttp->datasets>=2.0.0->evaluate)
(1.3.2)
Requirement already satisfied: async-timeout<6.0,>=4.0 in /usr/loca
l/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->eval
uate) (5.0.1)
```

```
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/pyth
on3.10/dist-packages (from aiohttp->datasets>=2.0.0->evaluate) (25.
1.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/
python3.10/dist-packages (from aiohttp->datasets>=2.0.0->evaluate)
(1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/li
b/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->evaluat
e) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/p
ython3.10/dist-packages (from aiohttp->datasets>=2.0.0->evaluate)
(0.2.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/
python3.10/dist-packages (from aiohttp->datasets>=2.0.0->evaluate)
(1.18.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.1
0/dist-packages (from python-dateutil>=2.8.2->pandas->evaluate) (1.
17.0)
Requirement already satisfied: intel-openmp>=2024 in /usr/local/li
b/python3.10/dist-packages (from mkl->numpy>=1.17->evaluate) (2024.
2.0)
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python
3.10/dist-packages (from mkl->numpy>=1.17->evaluate) (2022.0.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python
3.10/dist-packages (from tbb==2022.*->mkl->numpy>=1.17->evaluate)
(1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in /usr/local/li
b/python3.10/dist-packages (from mkl_umath->numpy>=1.17->evaluate)
(2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in /us
r/local/lib/python3.10/dist-packages (from intel-openmp>=2024->mkl-
>numpy>=1.17->evaluate) (2024.2.0)
Downloading evaluate-0.4.3-py3-none-any.whl (84 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.0/84.0 kB 6.0 MB/s eta
0:00:00
Installing collected packages: evaluate
Successfully installed evaluate-0.4.3
Collecting jiwer
  Downloading jiwer-3.1.0-py3-none-any.whl.metadata (2.6 kB)
Collecting click>=8.1.8 (from jiwer)
  Downloading click-8.1.8-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: rapidfuzz>=3.9.7 in /usr/local/lib/p
ython3.10/dist-packages (from jiwer) (3.12.2)
Downloading jiwer-3.1.0-py3-none-any.whl (22 kB)
Downloading click-8.1.8-py3-none-any.whl (98 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.2/98.2 kB 7.9 MB/s eta
0:00:00
Installing collected packages: click, jiwer
  Attempting uninstall: click
    Found existing installation: click 8.1.7
    Uninstalling click-8.1.7:
      Successfully uninstalled click-8.1.7
Successfully installed click-8.1.8 jiwer-3.1.0
```

In [4]:

```
!pip install -q git+https://github.com/huggingface/transformers.git
```

```
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
  Building wheel for transformers (pyproject.toml) ... done
```

In [5]:

```
import os
os.environ["PYTORCH_CUDA_ALLOC_CONF"] = "expandable_segments:True"
```

In [6]:

```
import numpy as np
from skimage.io import imread
from skimage.color import rgb2gray
from skimage.filters import threshold_otsu
import matplotlib.pyplot as plt
from heapq import heappop, heappush
from transformers import TrOCRProcessor, VisionEncoderDecoderModel
import torch
torch.cuda.empty_cache()
from torch.utils.data import Dataset, DataLoader
```

In [7]:

```
import shutil
import os
import docx
```

In [8]:
```
# def read_transcriptions(image_paths, file_path):
#     # Load the DOCX file
#     doc = docx.Document(file_path)

#     # Initialize the dictionary to store text from each page
#     transcript = {}
#     text = ''
#     image_index = -1
#     num_images = len(image_paths)

#     for para in doc.paragraphs:
#         stripped_text = para.text.strip()

#         # Skip "PDF pX" markers so they don't appear in extracted text
#         if stripped_text.startswith("PDF"):
#             continue

#         if "PDF" in stripped_text or "END OF EXTRACT" in stripped_tex
t:
#             if image_index >= 0:
#                 transcript[image_paths[image_index]] = text.strip()
#                 print(f"Saved transcription for {image_paths[image_ind
ex]}")
#             else:
#                 print(f"Skipping an index {image_index} due to mismatc
h.")
#             image_index += 1
#             text = ''
#         elif stripped_text != '':
#             text += stripped_text + "\n"
#         print(f"Index {image_index}, Extracted Text: {text[:100]}...")

#     # Add the last page text if it exists
#     if text.strip() and image_index < num_images:
#         transcript[image_paths[image_index]] = text.strip()
#     print(f"Total transcriptions: {len(transcript)} / {num_images}")
#     return transcript
```

In [9]:
```
# def read_transcriptions(image_paths, file_path):
#     # Load the DOCX file
#     doc = docx.Document(file_path)

#     # Initialize the dictionary to store text from each page
#     transcript = {}
#     text = ''
#     image_index = -1  # Start before the first image
#     num_images = len(image_paths)

#     for para in doc.paragraphs:
#         stripped_text = para.text.strip()

#         # Skip "PDF pX" markers so they don't appear in extracted text
#         if stripped_text.startswith("PDF"):
#             # Before moving to the next page, save the existing text
#             if image_index >= 0 and image_index < num_images:
#                 transcript[image_paths[image_index]] = text.strip()
#                 print(f"Saved transcription for {image_paths[image_ind
ex]}")

#             # Move to the next page
#             image_index += 1
#             text = ''
#             continue

#         # If valid text, append it
#         if stripped_text:
#             text += stripped_text + "\n"
#         print(f"Index {image_index}, Extracted Text: {text[:100]}...")
#     # Add the last page text if it exists
#     if text.strip() and 0 <= image_index < num_images:
#         transcript[image_paths[image_index]] = text.strip()

#     return transcript
```

In [10]:
```
def read_transcriptions(image_paths, file_path):
    doc = docx.Document(file_path)
    transcript = {}
    text = ''
    image_index = -1
    num_images = len(image_paths)

    for para in doc.paragraphs:
        stripped_text = para.text.strip()
        # print(f"Processing: {stripped_text}")  # Debugging

        # If we detect a new page marker, store the previous page's text
        if stripped_text.startswith("PDF p"):
            if image_index >= 0 and image_index < num_images:
                transcript[image_paths[image_index]] = text.strip()
                print(f"✅ Saved text for: {image_paths[image_inde
x]}")  # Debugging

            # Move to next image
            image_index += 1
            text = ''  # Reset collected text for the new page
            continue

        # Collect text for the current page
        if stripped_text:
            text += stripped_text + "\n"

    # Save text for the last image
    if text.strip() and 0 <= image_index < num_images:
        transcript[image_paths[image_index]] = text.strip()
        print(f"✅ Saved text for: {image_paths[image_index]}")  # Deb
ugging

    return transcript
```

In [11]:

```
# Without augmentations

# import os
# import re
# import numpy as np
# from skimage.io import imread
# from skimage.color import rgb2gray

# def get_segmented_image_paths(input_path, segmented_image_dir):
#     # Get the base name from the input path
#     base_name = os.path.basename(input_path)

#     # List all files in the directory
#     files = os.listdir(segmented_image_dir)

#     # Filter the files to get only those that start with the base name
and end with "_line_*.png"
#     segmented_files = [os.path.join(segmented_image_dir, f) for f in f
iles if f.startswith(base_name) and "_line_" in f and f.endswith(".pn
g")]

#     return segmented_files

# def extract_line_number(file_path):
#     # Extract the line number(s) from the file name using regex
#     match = re.search(r'_line_(\d+(_\d+)?).png$', file_path)
#     if match:
#         return list(map(int, match.group(1).split('_')))
#     return [-1]

# def sort_segmented_image_paths(image_paths):
#     # Sort the list of paths based on the extracted line number
#     sorted_paths = sorted(image_paths, key=extract_line_number)
#     return sorted_paths

# def align_segments_with_transcriptions(image_path, line_segmentation_p
ath, transcription):
#     segmented_image_paths = get_segmented_image_paths(image_path, line
_segmentation_path)
#     segmented_image_paths = sort_segmented_image_paths(segmented_image
_paths)
#     line_text = transcription.split('\n')
#     count = 0
#     data = []

#     for path in segmented_image_paths:
#         transcript = ""

#         line_img = imread(path)[:,:,:3]

#         if line_img.ndim > 2:
#             line_img = rgb2gray(line_img)
#         try:
#             transcript = line_text[count]
#             count += 1
#             data.append((line_img, transcript))
#         except Exception as e:
#             print(f"Error processing {image_path} - {path}: {e}")
#             print(len(segmented_image_paths),len(line_text))
#             print('\n')
#             continue
#     return data
```

In [12]:
```
#with augmentations
import os
import re
import numpy as np
from skimage.io import imread
from skimage.color import rgb2gray
import albumentations as A

# def extract_numbers(filename):
#     numbers = list(map(int, re.findall(r'\d+', filename)))  # Extract
all numbers
#     return tuple(numbers)  # Return a tuple for sorting

def get_segmented_image_paths(input_path, segmented_image_dir):
    base_name = os.path.basename(input_path)
    files = os.listdir(segmented_image_dir)
    segmented_files = [os.path.join(segmented_image_dir, f) for f in fi
les if base_name.replace(".png", "") in f and "_line_" in f and f.endsw
ith(".png")]
    # segmented_files = [os.path.join(segmented_image_dir, f) for f in f
iles if "_line_" in f and f.endswith(".png")]

    return segmented_files

def extract_line_number(file_path):
    match = re.search(r'_line_(\d+(_\d+)?).png$', file_path)
    if match:
        return list(map(int, match.group(1).split('_')))
    return [-1]

def sort_segmented_image_paths(image_paths):
    sorted_paths = sorted(image_paths, key=extract_line_number)
    return sorted_paths

# def apply_augmentations(image):
#     augmentations = [
#         A.Rotate(limit=3, p=1.0),
#         A.GaussNoise(var_limit=(10.0, 50.0), p=1.0),
#         A.ElasticTransform(alpha=0.3, sigma=100.0, p=1.0),
#         A.OpticalDistortion(distort_limit=0.03, shift_limit=0.03, p=1.
0),
#         A.CLAHE(clip_limit=2, tile_grid_size=(4, 4), p=1.0),
#         A.Affine(scale=(0.95, 1.05), translate_percent=(0.02, 0.02), s
hear=(-2, 2), p=1.0),
#         A.Perspective(scale=(0.01, 0.03), p=1.0),
#         A.RandomBrightnessContrast(brightness_limit=0.1, contrast_limi
t=0.1, p=1.0),
#         A.GaussianBlur(blur_limit=(3, 7), p=1.0),
#         A.GridDistortion(num_steps=3, distort_limit=0.02, p=1.0),
#         A.HueSaturationValue(hue_shift_limit=10, sat_shift_limit=10, v
al_shift_limit=10, p=1.0),
#         A.MedianBlur(blur_limit=3, p=1.0),
#     ]

#     # Ensure the image is in np.uint8 format to avoid type mismatch
#     image = image.astype(np.uint8)

#     augmented_images = []
#     for transform in augmentations:
#         augmented_images.append(transform(image=image)['image'])
#     return augmented_images
def apply_augmentations(image):
    augmentations = [
        A.Rotate(limit=3, p=1.0),
        A.GaussNoise(var_limit=(10.0, 50.0), p=1.0),
        A.ElasticTransform(alpha=0.3, sigma=100.0, p=1.0),
        A.OpticalDistortion(distort_limit=0.03, shift_limit=0.03, p=1.
0),
        A.CLAHE(clip_limit=2, tile_grid_size=(4, 4), p=1.0),
        A.Affine(scale=(0.95, 1.05), translate_percent=(0.02, 0.02), sh
ear=(-2, 2), p=1.0),
        A.Perspective(scale=(0.01, 0.03), p=1.0),
        A.RandomBrightnessContrast(brightness_limit=0.1, contrast_limit
=0.1, p=1.0),
        A.GaussianBlur(blur_limit=(3, 7), p=1.0),
        A.GridDistortion(num_steps=3, distort_limit=0.02, p=1.0),
        A.HueSaturationValue(hue_shift_limit=10, sat_shift_limit=10, va
l_shift_limit=10, p=1.0),
        A.MedianBlur(blur_limit=(3, 5), p=1.0),  # Ensures odd kernel s
ize
    ]

    # Ensure the image is in the correct format
    if len(image.shape) == 2:  # If grayscale (H, W)
        image = np.stack([image] * 3, axis=-1)  # Convert to (H, W, 3)

    image = np.clip(image, 0, 255).astype(np.uint8)  # Ensure valid dty
```

```
pe

    augmented_images = []
    for transform in augmentations:
        augmented_images.append(transform(image=image)['image'])

    return augmented_images

# def align_segments_with_transcriptions(image_path, line_segmentation_p
ath, transcription):

#     segmented_image_paths = get_segmented_image_paths(image_path, line
_segmentation_path)
#     segmented_image_paths = sort_segmented_image_paths(segmented_image
_paths)
#     print("Length of Segmented Images: ",len(segmented_image_paths))
#     line_text = transcription.split('\n')

#     count = 0
#     data = []

#     for path in segmented_image_paths:
#         transcript = ""

#         line_img = imread(path)[:,:,:3]

#         if line_img.ndim > 2:
#             line_img = rgb2gray(line_img)

#         try:
#             height = line_img.shape[0]
#             transcript = line_text[count]
#             count += 1

#             # Apply augmentations and store results
#             augmented_images = apply_augmentations(line_img)
#             for aug_img in augmented_images:
#                 data.append((aug_img, transcript))

#             # Store the original image and transcript
#             data.append((line_img, transcript))
#         except Exception as e:
#             print(f"Error processing {image_path} - {path}: {e}")
#             continue
#     return data

def align_segments_with_transcriptions(image_path, line_segmentation_path, transcription):

    segmented_image_paths = get_segmented_image_paths(image_path, line_segmentation_path)
    segmented_image_paths = sort_segmented_image_paths(segmented_image_paths)
    # segmented_image_paths = line_segmentation_path
    print(f"Length of Segmented Images: ", len(segmented_image_paths))

    line_text = transcription.split('\n')
    print(f"Length of Segmented Transcription: ", len(line_text))
    count = 0
    data = []
    # print("Segmented paths:", segmented_image_paths)
    # print("Non-file paths:", [p for p in segmented_image_paths if not os.path.isfile(p)])
    for path in segmented_image_paths:
        if not os.path.isfile(path):
            print(f"Skipping {path}: Not a valid file!")
            continue
        transcript = ""
        line_img = imread(path)

        # Ensure proper image format
        # if line_img.ndim == 3:
        #     line_img = rgb2gray(line_img)
        if line_img.ndim == 3:
            if line_img.shape[2] == 4:  # Check if image has 4 channels (RGBA)
                line_img = rgba2rgb(line_img)  # Convert RGBA to RGB
            line_img = rgb2gray(line_img)  # Convert RGB to grayscale
        try:
            height = line_img.shape[0]

            if count < len(line_text):
                transcript = line_text[count]
            else:
                print(f"Warning: No matching transcription for {path}")
                transcript = ""

            count += 1
            # Apply augmentations (ensure function returns a list)
            augmented_images = apply_augmentations(line_img) or []
            for aug_img in augmented_images:
                data.append((aug_img, transcript))

            # Store original image with transcript
            data.append((line_img, transcript))
        except Exception as e:
            print(f"Error processing {image_path} - {path}: {e}")
            continue

    return data
```

```
/usr/local/lib/python3.10/dist-packages/albumentations/__init__.py:
24: UserWarning: A new version of Albumentations is available: 2.0.
5 (you have 1.4.20). Upgrade using: pip install -U albumentations.
To disable automatic update checks, set the environment variable NO
_ALBUMENTATIONS_UPDATE to 1.
  check_for_updates()
```

In [13]:
```
# !pip install zipfile
```

In [14]:

```
# import zipfile

# # Define paths
# zip_path = "/kaggle/input/your-dataset-name/your-file.zip"  # Update w
ith your actual ZIP file name
# extract_path = "/kaggle/working/unzipped_data"  # Where files will be
extracted

# # Create the extraction directory if it doesn't exist
# os.makedirs(extract_path, exist_ok=True)

# # Extract the ZIP file
# with zipfile.ZipFile(zip_path, 'r') as zip_ref:
#     zip_ref.extractall(extract_path)

# print("Extraction completed! Files are now available in:", extract_pat
h)
```

In [15]:

```
from skimage.color import rgba2rgb
```

In [16]:
```
aligned_data = {}

folder_path = '/kaggle/input/spanish-ocr-3/pdfs - Copy'
# Loop through each PDF folder
for pdf_folder in os.listdir(folder_path):
    pdf_path = os.path.join(folder_path, pdf_folder)

    # Define paths
    page_images_path = os.path.join(pdf_path, "pages")
    segmented_images_path = os.path.join(pdf_path, "segmented")

    transcription_files = [f for f in os.listdir(pdf_path) if f.endswit
h('.docx')]
    if not transcription_files:
        print(f"Skipping {pdf_folder}: No transcription file found")
        continue
    transcription_path = os.path.join(pdf_path, transcription_files[0])
    # transcription_path = os.path.join(pdf_path, "transcription.docx")

    # Check if required files exist
    if not os.path.exists(page_images_path) or not os.path.exists(segme
nted_images_path) or not os.path.exists(transcription_path):
        print(f"Skipping {pdf_folder}: Missing required files")
        continue

    # Get sorted list of page images
    image_paths = sorted(
        [os.path.join(page_images_path, f) for f in os.listdir(page_ima
ges_path) if f.endswith(('.png', '.jpg', '.jpeg'))]
    )

    if not image_paths:
        print(f"Skipping {pdf_folder}: No page images found")
        continue
    # print(f"Number of Images found in {pdf_folder}: ",len(image_path
s))
    # print("All images:", image_paths)
    # Read transcription
    transcript = read_transcriptions(image_paths, transcription_path)
    # print(f"{pdf_folder} Transcript: ",transcript)
    if not transcript:
        print(f"Error: No transcriptions extracted for {pdf_folder}")
    # Process each image and store aligned data
    # print("Checking transcription file:", transcription_path)
    print("Extracted transcript keys:", transcript.keys())
    for img_path in image_paths:
        if img_path in transcript:
            aligned_data[img_path] = align_segments_with_transcriptions
(img_path, segmented_images_path, transcript[img_path])
        if img_path not in transcript:
            print(f"Skipping {img_path}: No matching transcription foun
d!")

# Now `aligned_data` contains the structured (image, text) pairs for all
PDFs.
```

```
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Constitu
ciones sinodales Calahorra 1602/pages/Constituciones sinodales Cala
horra 1602_page_1.png
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Constitu
ciones sinodales Calahorra 1602/pages/Constituciones sinodales Cala
horra 1602_page_2.png
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Constitu
ciones sinodales Calahorra 1602/pages/Constituciones sinodales Cala
horra 1602_page_3.png
Extracted transcript keys: dict_keys(['/kaggle/input/spanish-ocr-3/
pdfs - Copy/Constituciones sinodales Calahorra 1602/pages/Constituc
iones sinodales Calahorra 1602_page_1.png', '/kaggle/input/spanish-
ocr-3/pdfs - Copy/Constituciones sinodales Calahorra 1602/pages/Con
stituciones sinodales Calahorra 1602_page_2.png', '/kaggle/input/sp
anish-ocr-3/pdfs - Copy/Constituciones sinodales Calahorra 1602/pag
es/Constituciones sinodales Calahorra 1602_page_3.png'])
Length of Segmented Images:  38
Length of Segmented Transcription:  38
Length of Segmented Images:  15
Length of Segmented Transcription:  15
Length of Segmented Images:  38
Length of Segmented Transcription:  38
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/PORCONE
S.228.35  1636/pages/PORCONES.228.35  1636_page_1.png
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/PORCONE
S.228.35  1636/pages/PORCONES.228.35  1636_page_3.png
Extracted transcript keys: dict_keys(['/kaggle/input/spanish-ocr-3/
pdfs - Copy/PORCONES.228.35  1636/pages/PORCONES.228.35  1636_page_
1.png', '/kaggle/input/spanish-ocr-3/pdfs - Copy/PORCONES.228.35  1
636/pages/PORCONES.228.35  1636_page_3.png'])
Length of Segmented Images:  18
Length of Segmented Transcription:  18
Length of Segmented Images:  38
Length of Segmented Transcription:  38
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Ezcaray
- Vozes/pages/Ezcaray - Vozes_page_1.png
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Ezcaray
- Vozes/pages/Ezcaray - Vozes_page_2.png
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Ezcaray
- Vozes/pages/Ezcaray - Vozes_page_3.png
Extracted transcript keys: dict_keys(['/kaggle/input/spanish-ocr-3/
pdfs - Copy/Ezcaray - Vozes/pages/Ezcaray - Vozes_page_1.png', '/ka
ggle/input/spanish-ocr-3/pdfs - Copy/Ezcaray - Vozes/pages/Ezcaray
```

```
- Vozes_page_2.png', '/kaggle/input/spanish-ocr-3/pdfs - Copy/Ezcar
ay - Vozes/pages/Ezcaray - Vozes_page_3.png'])
Length of Segmented Images:  20
Length of Segmented Transcription:  20
Length of Segmented Images:  23
Length of Segmented Transcription:  23
Length of Segmented Images:  23
Length of Segmented Transcription:  23
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Mendo -
Principe perfecto/pages/Mendo - Principe perfecto_page_1.png
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Mendo -
Principe perfecto/pages/Mendo - Principe perfecto_page_2.png
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Mendo -
Principe perfecto/pages/Mendo - Principe perfecto_page_3.png
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Mendo -
Principe perfecto/pages/Mendo - Principe perfecto_page_4.png
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Mendo -
Principe perfecto/pages/Mendo - Principe perfecto_page_5.png
Extracted transcript keys: dict_keys(['/kaggle/input/spanish-ocr-3/
pdfs - Copy/Mendo - Principe perfecto/pages/Mendo - Principe perfec
to_page_1.png', '/kaggle/input/spanish-ocr-3/pdfs - Copy/Mendo - Pr
incipe perfecto/pages/Mendo - Principe perfecto_page_2.png', '/kagg
le/input/spanish-ocr-3/pdfs - Copy/Mendo - Principe perfecto/pages/
Mendo - Principe perfecto_page_3.png', '/kaggle/input/spanish-ocr-
3/pdfs - Copy/Mendo - Principe perfecto/pages/Mendo - Principe perf
ecto_page_4.png', '/kaggle/input/spanish-ocr-3/pdfs - Copy/Mendo -
Principe perfecto/pages/Mendo - Principe perfecto_page_5.png'])
Length of Segmented Images:  20
Length of Segmented Transcription:  20
Length of Segmented Images:  8
Length of Segmented Transcription:  8
Length of Segmented Images:  33
Length of Segmented Transcription:  33
Length of Segmented Images:  37
Length of Segmented Transcription:  37
Length of Segmented Images:  37
Length of Segmented Transcription:  37
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Buendia
- Instruccion/pages/Buendia - Instruccion_page_1.png
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Buendia
- Instruccion/pages/Buendia - Instruccion_page_2.png
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Buendia
- Instruccion/pages/Buendia - Instruccion_page_3.png
Extracted transcript keys: dict_keys(['/kaggle/input/spanish-ocr-3/
pdfs - Copy/Buendia - Instruccion/pages/Buendia - Instruccion_page_
1.png', '/kaggle/input/spanish-ocr-3/pdfs - Copy/Buendia - Instrucc
ion/pages/Buendia - Instruccion_page_2.png', '/kaggle/input/spanish
-ocr-3/pdfs - Copy/Buendia - Instruccion/pages/Buendia - Instruccio
n_page_3.png'])
Length of Segmented Images:  22
Length of Segmented Transcription:  22
Length of Segmented Images:  28
Length of Segmented Transcription:  28
Length of Segmented Images:  23
Length of Segmented Transcription:  23
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Paredes
- Reglas generales/pages/Paredes - Reglas generales_page_1.png
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Paredes
- Reglas generales/pages/Paredes - Reglas generales_page_2.png
✅ Saved text for: /kaggle/input/spanish-ocr-3/pdfs - Copy/Paredes
- Reglas generales/pages/Paredes - Reglas generales_page_3.png
Extracted transcript keys: dict_keys(['/kaggle/input/spanish-ocr-3/
pdfs - Copy/Paredes - Reglas generales/pages/Paredes - Reglas gener
ales_page_1.png', '/kaggle/input/spanish-ocr-3/pdfs - Copy/Paredes
- Reglas generales/pages/Paredes - Reglas generales_page_2.png', '/
kaggle/input/spanish-ocr-3/pdfs - Copy/Paredes - Reglas generales/p
ages/Paredes - Reglas generales_page_3.png'])
Length of Segmented Images:  12
Length of Segmented Transcription:  12
Length of Segmented Images:  9
Length of Segmented Transcription:  9
Length of Segmented Images:  34
Length of Segmented Transcription:  34
```

In [17]:

```python
print("Aligned Text Data")
print(len(aligned_data))
```

```
Aligned Text Data
19
```

In [18]:

```python
# !pip install --upgrade transformers torch torchvision torchaudio
```

In [19]:
```
!python -c "import transformers; print(transformers.__version__)"
!python -c "import torch; print(torch.__version__)"
```

```
4.51.0.dev0
2.5.1+cu121
```

In [20]:
```
# from transformers import TrOCRProcessor, VisionEncoderDecoderModel, Tr
ainer, TrainingArguments, EarlyStoppingCallback, TrainerCallback
```

In [21]:
```
import wandb

wandb.login(key="d4c899160235a84e61d6d5c85c5b502c7d322ed3")
```

```
wandb: Using wandb-core as the SDK backend.  Please refer to http
s://wandb.me/wandb-core for more information.
wandb: Currently logged in as: gary_hornbill (gary_hornbill-indian-
institute-of-technology-ism-dhanbad). Use `wandb login --relogin` t
o force relogin
wandb: WARNING If you're specifying your api key in code, ensure th
is code is not shared publicly.
wandb: WARNING Consider setting the WANDB_API_KEY environment varia
ble, or running `wandb login` from the command line.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.ne
trc
```

Out[21]:
```
True
```

In [22]:
```
# Sequence Likelihood Calibration
import albumentations as A
import torch
from torch.utils.data import Dataset, DataLoader, random_split
from transformers import TrOCRProcessor, VisionEncoderDecoderModel, Tra
iner, TrainingArguments, EarlyStoppingCallback, TrainerCallback
from PIL import Image
import numpy as np
from torch.nn.utils.rnn import pad_sequence
from torch.optim import AdamW
from torch.optim.lr_scheduler import CosineAnnealingWarmRestarts
from evaluate import load
import torch.nn.functional as F
import random
from transformers import get_cosine_schedule_with_warmup
import os
import gc

os.environ['CUDA_LAUNCH_BLOCKING'] = '1'
os.environ['TORCH_USE_CUDA_DSA'] = '0'

# Load the CER and WER metrics
cer_metric = load("cer")
wer_metric = load("wer")

model_path = "qantev/trocr-large-spanish"
processor_path = "qantev/trocr-large-spanish"

# model_path = "./finetuned_transformer_model_calibration_v1"
# processor_path = "./finetuned_transformer_model_calibration_v1"

processor = TrOCRProcessor.from_pretrained(processor_path, do_rescale=F
alse)
model = VisionEncoderDecoderModel.from_pretrained(model_path)
model.gradient_checkpointing_enable()

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    if isinstance(logits, tuple):
        logits = logits[0]
    predictions = logits.argmax(-1)
    decoded_preds = processor.tokenizer.batch_decode(predictions, skip_
special_tokens=True)
```

```
    decoded_labels = []
    for label in labels:
        label_filtered = [token for token in label if token != -100]
        decoded_label = processor.tokenizer.decode(label_filtered, skip
_special_tokens=True)
        decoded_labels.append(decoded_label)
    cer_score = cer_metric.compute(predictions=decoded_preds, reference
s=decoded_labels)
    wer_score = wer_metric.compute(predictions=decoded_preds, reference
s=decoded_labels)
    return {"cer": cer_score, "wer": wer_score}

class LineDataset(Dataset):
    def __init__(self, processor, model, line_images, texts, target_siz
e=(256, 64), max_length=512, apply_augmentation=False):
        self.line_images = line_images
        self.texts = texts
        self.processor = processor
        self.processor.image_processor.max_length = max_length
        self.processor.tokenizer.model_max_length = max_length
        self.model = model
        self.model.config.max_length = max_length
        self.target_size = target_size
        self.max_length = max_length
        self.apply_augmentation = apply_augmentation

        if apply_augmentation:
            self.transform = A.Compose([
                A.OneOf([
                    A.Rotate(limit=2, p=1.0),
                    A.GaussNoise(var_limit=(10.0, 30.0), p=1.0),
                    A.ElasticTransform(alpha=0.3, sigma=50.0, p=1.0),
                    A.OpticalDistortion(distort_limit=0.03, shift_limit
=0.03, p=1.0),
                    A.CLAHE(clip_limit=2, tile_grid_size=(4, 4), p=1.
0),
                    A.Affine(scale=(0.95, 1.05), translate_percent=(0.0
2, 0.02), shear=(-2, 2), p=1.0),
                    A.Perspective(scale=(0.01, 0.03), p=1.0),
                    A.GaussianBlur(blur_limit=(3, 5), p=1.0),
                    A.MedianBlur(blur_limit=1, p=1.0)
                ], p=0.8),
            ])
        else:
            self.transform = A.Compose([])

        # self.processed_images = []
        # for image in line_images:
        #     if isinstance(image, Image.Image):
        #         image = np.array(image)
        #     if image.ndim == 2:
        #         image = np.expand_dims(image, axis=-1)
        #         image = np.repeat(image, 3, axis=-1)
        #     self.processed_images.append(image)

    def __len__(self):
        return len(self.line_images)

    def __getitem__(self, idx):
        image = self.line_images[idx]
        # image = self.processed_images[idx]
        text = self.texts[idx]

        if isinstance(image, Image.Image):
            image = np.array(image)

        if image.ndim == 2:
            image = np.expand_dims(image, axis=-1)
            image = np.repeat(image, 3, axis=-1)

        image = (image * 255).astype(np.uint8)

        if self.apply_augmentation:
            augmented = self.transform(image=image)
            image = augmented['image']

        image = Image.fromarray(image)
        image = image.resize(self.target_size, Image.BILINEAR)
        image = np.array(image) / 255.0
        image = np.transpose(image, (2, 0, 1))

        encoding = self.processor(images=image, text=text, return_tenso
rs="pt")
        encoding['labels'] = encoding['labels'][:, :self.max_length]
        encoding = {k: v.squeeze() for k, v in encoding.items()}
        return encoding

def collate_fn(batch):
    pixel_values = torch.stack([item['pixel_values'] for item in batc
h])
    labels = pad_sequence([item['labels'] for item in batch], batch_fir
st=True, padding_value=-100)
    return {'pixel_values': pixel_values, 'labels': labels}

# Temperature Scaling
class TemperatureScalingModel(torch.nn.Module):
    def __init__(self, model, temperature=0.9995831251144409):
        super(TemperatureScalingModel, self).__init__()
        self.model = model
        self.temperature = torch.nn.Parameter(torch.ones(1) * temperatu
re)
        self.config = model.config

    def forward(self, pixel_values, labels=None):
        if labels is not None:
            outputs = self.model(pixel_values=pixel_values, labels=labe
ls)
            outputs.logits = outputs.logits.float() / self.temperature
            return outputs
        else:
            logits = self.model(pixel_values=pixel_values).logits
            return logits.float() / self.temperature

    def generate(self, pixel_values, **kwargs):
        return self.model.generate(pixel_values, **kwargs)

    def get_output_embeddings(self):
        return self.model.get_output_embeddings()

    def make_weights_contiguous(self):
        """Make all model weights contiguous before saving"""
        for name, param in self.named_parameters():
            if not param.is_contiguous():
                param.data = param.data.contiguous()

    def save_pretrained(self, save_directory):
        if not os.path.exists(save_directory):
            os.makedirs(save_directory)

        self.make_weights_contiguous()

        if hasattr(model, 'encoder') and hasattr(model.encoder, 'embedd
ings'):
            patch_embeddings = model.encoder.embeddings.patch_embedding
s
            if hasattr(patch_embeddings, 'projection'):
                if not patch_embeddings.projection.weight.is_contiguous
():
                    patch_embeddings.projection.weight.data = patch_emb
eddings.projection.weight.data.contiguous()
                    changes_made = True
                    self.log("Made patch embeddings projection weight c
ontiguous")

        # Save the base model
        self.model.save_pretrained(save_directory)

        # Save the temperature parameter
        temperature_path = os.path.join(save_directory, "temperature.p
t")
        torch.save(self.temperature, temperature_path)

    def gradient_checkpointing_enable(self, gradient_checkpointing_kwar
gs=None):
        """Expose the gradient_checkpointing_enable method from the base
model."""
        if hasattr(self.model, "gradient_checkpointing_enable"):
            self.model.gradient_checkpointing_enable(gradient_checkpoin
ting_kwargs=gradient_checkpointing_kwargs)

    @classmethod
    def from_pretrained(cls, load_directory):
        base_model = VisionEncoderDecoderModel.from_pretrained(load_dir
ectory)
        temperature_path = os.path.join(load_directory, "temperature.p
t")
        temperature = torch.load(temperature_path)

        model = cls(base_model, temperature.item())
        return model

    @property
    def device(self):
        return self.model.device

class CosineAnnealingWarmRestartsSchedulerCallback(TrainerCallback):
    def __init__(self, optimizer, T_0, T_mult=1, eta_min=0):
        self.scheduler = CosineAnnealingWarmRestarts(optimizer, T_0, T_
mult, eta_min)

    def on_step_end(self, args, state, control, **kwargs):
        self.scheduler.step()


class SLiCTrainer(Trainer):
    def __init__(self, *args, label_smoothing=0.1, gamma=2.0, **kwarg
s):
        super().__init__(*args, **kwargs)
        self.label_smoothing = label_smoothing
        self.gamma = gamma

    def compute_loss(self, model, inputs, return_outputs=False,num_item
s_in_batch=None):
        torch.cuda.empty_cache()
        labels = inputs.get("labels")
        pixel_values = inputs.get("pixel_values")

        # Initializing lambda
        lambda_reg = 0.01
        with torch.amp.autocast('cuda'):  # Added for mixed precision t
raining
            outputs = model(pixel_values=pixel_values, labels=labels)
            logits = outputs.logits if hasattr(outputs, 'logits') else
outputs
            logits = logits.float()

            if torch.isnan(logits).any():
                print("Warning: NaN values detected in logits")
                logits = torch.nan_to_num(logits, nan=0.0)

            vocab_size = logits.size(-1)
            labels = torch.clamp(labels, min=0, max=vocab_size - 1)

            # labels = self.smooth_labels(labels, vocab_size)

            temperature = model.temperature.item() if hasattr(model, 't
emperature') else 1.0
            similarity_scores = self.calculate_similarity(logits, label
s)
```

```
            slic_loss = self.compute_calibration_loss(logits, labels, s
imilarity_scores)
            del similarity_scores
            kl_loss = self.compute_kl_divergence(logits, labels)
            focal_loss = self.compute_focal_loss(logits, labels)

            total_loss = slic_loss + lambda_reg * (kl_loss + focal_los
s)
            if torch.isnan(total_loss) or torch.isinf(total_loss):
                print("Warning: Invalid loss value detected")
                total_loss = torch.tensor(1.0, device=total_loss.devic
e,requires_grad=True)
        # outputs = model(pixel_values=pixel_values, labels=labels)
        # logits = outputs.logits if hasattr(outputs, 'logits') else out
puts
        # logits = logits.float()

        # # Ensure labels are within the valid range
        # vocab_size = logits.size(-1)
        # labels = torch.clamp(labels, min=0, max=vocab_size - 1)

        # temperature = model.temperature.item() if hasattr(model, 'temp
erature') else 1.0

        # # Compute similarity scores
        # similarity_scores = self.calculate_similarity(logits, labels)

        # # Apply SLIC Loss
        # slic_loss = self.compute_calibration_loss(logits, labels, simi
larity_scores)

        # # Apply KL Divergence Loss
        # kl_loss = self.compute_kl_divergence(logits, labels)

        # # Apply Focal Loss
        # focal_loss = self.compute_focal_loss(logits, labels)

        # # Total Loss
        # total_loss = slic_loss + lambda_reg * (kl_loss + focal_loss)

        # clearing unnecessary tensors
        del slic_loss, kl_loss, focal_loss
        torch.cuda.empty_cache()
        return (total_loss, outputs) if return_outputs else total_loss

    def compute_calibration_loss(self, logits, labels, similarity_score
s):
        batch_size, seq_len, vocab_size = logits.size()
        device = logits.device

        # Compute log probabilities
        log_probs = F.log_softmax(logits, dim=-1)

        # Generate positive and negative samples
        pos_samples = labels.unsqueeze(-1)
        neg_samples = torch.randint(0, vocab_size, (batch_size, seq_le
n, 1), device=device)

        # Compute losses
        l_rank = self.compute_rank_loss(log_probs, pos_samples, neg_sam
ples)
        l_margin = self.compute_margin_loss(log_probs, similarity_score
s, pos_samples, neg_samples)

        # Combine losses (you may want to add weights to different loss
components)
        calibration_loss = l_rank + l_margin

        del log_probs, pos_samples, neg_samples, l_rank, l_margin
        torch.cuda.empty_cache()

        return calibration_loss

    def compute_rank_loss(self, log_probs, pos_samples, neg_samples):
        beta = 0.1
        l_rank = torch.max(torch.zeros_like(log_probs[:, :, 0]),
                           beta - log_probs.gather(-1, pos_samples).squ
eeze(-1) +
                           log_probs.gather(-1, neg_samples).squeeze(-
1)).mean()
        # print("Rank Loss : ",l_rank)
        return l_rank

    def compute_margin_loss(self, log_probs, similarity_scores, pos_sam
ples, neg_samples):
        beta = 0.1
        l_margin = torch.max(torch.zeros_like(log_probs[:, :, 0]),
                            beta * (similarity_scores.gather(-1, pos_s
amples).squeeze(-1) -
                                    similarity_scores.gather(-1, neg_s
amples).squeeze(-1)) -
                            log_probs.gather(-1, pos_samples).squeeze
(-1) +
                            log_probs.gather(-1, neg_samples).squeeze
(-1)).mean()

        # print("Margin Loss : ",l_margin)
        return l_margin


    def compute_kl_divergence(self, logits, labels):
        log_probs = F.log_softmax(logits, dim=-1)
        target_probs = F.one_hot(labels, num_classes=logits.size(-1)).f
loat()
        kl_div_loss = F.kl_div(log_probs, target_probs, reduction='batc
hmean')

        # print("KL Divergence Loss : ",kl_div_loss)
        return kl_div_loss

    def compute_focal_loss(self, logits, labels, gamma=2.0):
        ce_loss = F.cross_entropy(logits.view(-1, logits.size(-1)), lab
els.view(-1), reduction='none')
        pt = torch.exp(-ce_loss)
        focal_loss = ((1 - pt) ** gamma * ce_loss).mean()
        # print("Focal Loss : ",focal_loss)
        return focal_loss

    def calculate_similarity(self, logits, labels):
        batch_size, seq_len, vocab_size = logits.size()
        device = logits.device

        # Get token embeddings
        token_embeddings = self.model.get_output_embeddings().weight  # 
Shape: [vocab_size, embedding_dim]

        # Compute logits_softmax once
        logits_softmax = F.softmax(logits, dim=-1)  # Shape: [batch_siz
e, seq_len, vocab_size]

        # Compute Fu(e, e)
        # Instead of computing the full matrix, we'll compute it on-the-
fly

        # Compute Ru(e, e)
        # We only need the first column to be 1, rest are 0
        # ru_ee_first_col = torch.zeros(vocab_size, 1, device=device)
        # ru_ee_first_col[0] = 1.0  # Assuming 0 is the padding token in
dex

        # # Compute similarity scores
        # # We'll do this in two parts to save memory

        # # Part 1: Fu(e, e) contribution
        # similarity_scores = torch.matmul(logits_softmax, token_embeddi
ngs)  # [batch_size, seq_len, embedding_dim]
        # similarity_scores = torch.matmul(similarity_scores, token_embe
ddings.t())  # [batch_size, seq_len, vocab_size]

        # # Part 2: Ru(e, e) contribution
        # ru_contribution = torch.matmul(logits_softmax, ru_ee_first_co
l)  # [batch_size, seq_len, 1]
        chunk_size = min(vocab_size, 1024)
        similarity_scores = torch.zeros_like(logits)

        # Part 1: Chunked similarity computation
        for i in range(0, vocab_size, chunk_size):
            end_idx = min(i + chunk_size, vocab_size)
            chunk_logits = logits_softmax[:, :, i:end_idx]
            token_embeddings = self.model.get_output_embeddings().weigh
t[i:end_idx]

            # First multiplication
            intermediate = torch.matmul(chunk_logits, token_embeddings)
            # Second multiplication
            sim_chunk = torch.matmul(intermediate, token_embeddings.t
())

            similarity_scores[:, :, i:end_idx] = sim_chunk

            del chunk_logits, intermediate, sim_chunk
            torch.cuda.empty_cache()

        # Part 2: Add ru_contribution
        ru_ee_first_col = torch.zeros(vocab_size, 1, device=device)
        ru_ee_first_col[0] = 1.0
        ru_contribution = torch.matmul(logits_softmax, ru_ee_first_col)
        # Combine both parts
        similarity_scores = (similarity_scores + ru_contribution) / 2

        return similarity_scores

    # def smooth_labels(self, labels, vocab_size):
    #     # Create one-hot encoded labels
    #     one_hot = F.one_hot(labels, num_classes=vocab_size).float()

    #     # Apply label smoothing
    #     smoothed = one_hot * (1 - self.label_smoothing) + \
    #                self.label_smoothing / vocab_size

    #     # Handle padding tokens (where labels == -100)
    #     mask = (labels == -100).unsqueeze(-1)
    #     smoothed = torch.where(mask, torch.zeros_like(smoothed), smoot
hed)

    #     return smoothed



    def beam_search_decode(self, model, pixel_values, beam_size=10, max
_length=128):
        # Implement beam search decoding
        batch_size = pixel_values.size(0)
        device = pixel_values.device

        encoder_outputs = model.encoder(pixel_values=pixel_values)
        input_ids = torch.full((batch_size * beam_size, 1), model.confi
g.decoder_start_token_id, dtype=torch.long, device=device)
        beam_scores = torch.zeros((batch_size, beam_size), dtype=torch.
float, device=device)
        beam_scores[:, 1:] = -1e9
        beam_scores = beam_scores.view(-1)

        for step in range(max_length):
            outputs = model.decoder(input_ids=input_ids, encoder_hidden
_states=encoder_outputs.last_hidden_state.repeat_interleave(beam_size,
dim=0))
            next_token_logits = outputs.logits[:, -1, :]
            next_token_scores = F.log_softmax(next_token_logits, dim=-
1)
```

```
            next_token_scores = next_token_scores + beam_scores[:, Non
e].expand_as(next_token_scores)
            vocab_size = next_token_scores.size(-1)

            next_token_scores = next_token_scores.view(batch_size, beam
_size * vocab_size)
            next_tokens = torch.argmax(next_token_scores, dim=-1)
            next_tokens = next_tokens % vocab_size
            next_beam_scores = torch.gather(next_token_scores, -1, next
_tokens.unsqueeze(-1)).squeeze(-1)

            input_ids = torch.cat([input_ids, next_tokens.unsqueeze(-
1)], dim=-1)
            beam_scores = next_beam_scores.view(-1)

            if (next_tokens == model.config.eos_token_id).any():
                break

        return input_ids.view(batch_size, beam_size, -1)[:, 0, :]

    def create_scheduler(self, num_training_steps: int, optimizer: torc
h.optim.Optimizer = None):
        """
        Create a scheduler using CosineAnnealingWarmRestarts
        """
        if optimizer is None:
            optimizer = self.optimizer
        return CosineAnnealingWarmRestarts(optimizer, T_0=num_training_
steps // 10, T_mult=2, eta_min=1e-6)

# class CustomCallback(TrainerCallback):
#     def on_epoch_end(self, args, state, control, **kwargs):
#         print("Epoch ended, clearing cache.")
#         torch.cuda.empty_cache()
#         gc.collect()

class MemoryManagementCallback(TrainerCallback):
    def on_step_end(self, args, state, control, **kwargs):
        torch.cuda.empty_cache()
        gc.collect()

    def on_epoch_end(self, args, state, control, **kwargs):
        torch.cuda.empty_cache()
        gc.collect()
    def on_evaluate(self, args, state, control, **kwargs):
        torch.cuda.empty_cache()
        gc.collect()

    def on_evaluate_begin(self, args, state, control, **kwargs):
        torch.cuda.empty_cache()
        gc.collect()

# class AggressiveMemoryManagementCallback(TrainerCallback):
#     def __init__(self, memory_threshold=0.85):
#         self.memory_threshold = memory_threshold

#     def check_memory(self):
#         if torch.cuda.is_available():
#             memory_allocated = torch.cuda.memory_allocated()
#             memory_reserved = torch.cuda.memory_reserved()
#             memory_total = torch.cuda.get_device_properties(0).total_m
emory
#             memory_ratio = (memory_allocated + memory_reserved) / memo
ry_total

#             if memory_ratio > self.memory_threshold:
#                 torch.cuda.empty_cache()
#                 gc.collect()

#     def on_step_end(self, args, state, control, **kwargs):
#         self.check_memory()

#     def on_evaluate(self, args, state, control, **kwargs):
#         torch.cuda.empty_cache()
#         gc.collect()

#     def on_epoch_end(self, args, state, control, **kwargs):
#         torch.cuda.empty_cache()
#         gc.collect()

# def print_gpu_memory():
#     if torch.cuda.is_available():
#         print(f"GPU Memory allocated: {torch.cuda.memory_allocated()/1
024**2:.2f} MB")
#         print(f"GPU Memory cached: {torch.cuda.memory_reserved()/1024*
*2:.2f} MB")
```

```
# class MemoryMonitorCallback(TrainerCallback):
#     def on_log(self, args, state, control, logs=None, **kwargs):
#         print_gpu_memory()

class MemoryMonitorCallback(TrainerCallback):
    def __init__(self, print_steps=100):
        self.print_steps = print_steps
        self.step_count = 0

    def print_memory_usage(self, prefix=""):
        if torch.cuda.is_available():
            allocated = torch.cuda.memory_allocated() / 1024**2
            reserved = torch.cuda.memory_reserved() / 1024**2
            print(f"{prefix} GPU Memory: Allocated: {allocated:.2f}MB, 
Reserved: {reserved:.2f}MB")

    def on_step_begin(self, args, state, control, **kwargs):
        self.step_count += 1
        if self.step_count % self.print_steps == 0:
            self.print_memory_usage(f"Step {self.step_count} Begin -")

    def on_step_end(self, args, state, control, **kwargs):
        if self.step_count % self.print_steps == 0:
            self.print_memory_usage(f"Step {self.step_count} End -")

    def on_epoch_begin(self, args, state, control, **kwargs):
        self.print_memory_usage(f"Epoch {state.epoch} Begin -")

    def on_epoch_end(self, args, state, control, **kwargs):
        self.print_memory_usage(f"Epoch {state.epoch} End -")

    def on_evaluate(self, args, state, control, **kwargs):
        self.print_memory_usage("Evaluation -")

    def on_log(self, args, state, control, logs=None, **kwargs):
        self.print_memory_usage("Logging -")

class ErrorRecoveryCallback(TrainerCallback):
    def on_error(self, args, state, control, **kwargs):
        print("Error occurred, attempting recovery...")
        torch.cuda.empty_cache()
        gc.collect()

        # Save current state
        if hasattr(state, 'epoch'):
            print(f"Error occurred at epoch {state.epoch}")
        if hasattr(state, 'global_step'):
            print(f"Error occurred at step {state.global_step}")

# class VitCheckpointCallback(TrainerCallback):
#     def __init__(self, save_every_n_epochs=1, verbose=True):
#         self.save_every_n_epochs = save_every_n_epochs
#         self.verbose = verbose
#         self.trainer = None

#     def on_train_begin(self, args, state, control, **kwargs):
#         # self.trainer = trainer
#         if 'trainer' in kwargs:
#             self.trainer = kwargs['trainer']

#     def log(self, message):
#         if self.verbose:
#             print(message)

#     def make_tensor_contiguous(self, model):
#         """Make specific tensors contiguous and track changes"""
#         changes_made = False

#         if hasattr(model, 'encoder') and hasattr(model.encoder, 'embed
dings'):
#             patch_embeddings = model.encoder.embeddings.patch_embeddin
gs
#             if hasattr(patch_embeddings, 'projection'):
#                 if not patch_embeddings.projection.weight.is_contiguou
s():
#                     patch_embeddings.projection.weight.data = patch_em
beddings.projection.weight.data.contiguous()
#                     changes_made = True
#                     self.log("Made patch embeddings projection weight
contiguous")

#         return changes_made

#     def save_checkpoint(self, trainer, model, epoch, directory):
#         """Attempt to save checkpoint with error handling"""
#         try:
#             os.makedirs(directory, exist_ok=True)
#             trainer.save_model(directory)
#             self.log(f"Successfully saved checkpoint to {directory}")
#             return True
#         except Exception as e:
#             self.log(f"Error saving checkpoint: {str(e)}")
#             return False

#     def on_epoch_end(self, args, state, control, **kwargs):
#         epoch = state.epoch
#         if epoch % self.save_every_n_epochs == 0:
#             model = self.trainer.model
#             # trainer = kwargs.get('trainer')

#             if model is None or trainer is None:
#                 self.log("Model or trainer not found in kwargs")
#                 return

#             checkpoint_dir = f"checkpoint-epoch-{int(epoch)}"

#             # First attempt: Make specific tensors contiguous
#             self.make_tensor_contiguous(model)
#             if self.save_checkpoint(trainer, model, epoch, checkpoint_
dir):
#                 return

#             # Second attempt: Try with full model clone
#             self.log("Attempting fallback save method with model clon
e...")
#             try:
#                 model_to_save = type(model)(**model.config.to_dict())
#                 state_dict = {k: v.detach().clone().contiguous()
#                             for k, v in model.state_dict().items()}
#                 model_to_save.load_state_dict(state_dict)

#                 if self.save_checkpoint(trainer, model_to_save, epoch,
checkpoint_dir):
#                     return

#             except Exception as clone_error:
#                 self.log(f"Fallback save method failed: {str(clone_err
or)}")

#                 # Third attempt: Emergency minimal save
#                 try:
#                     self.log("Attempting emergency minimal save...")
#                     minimal_state = {
#                         'epoch': epoch,
#                         'global_step': state.global_step,
#                         'optimizer_state': trainer.optimizer.state_dic
t(),
#                         'model_config': model.config.to_dict()
#                     }
#                     torch.save(minimal_state, f"{checkpoint_dir}_minim
al.pt")
#                     self.log("Saved minimal checkpoint")
#                 except Exception as emergency_error:
#                     self.log(f"Emergency save failed: {str(emergency_e
rror)}")

#     def on_train_end(self, args, state, control, **kwargs):
#         """Save final checkpoint"""
#         try:
#             self.log("Saving final checkpoint...")
#             model = self.trainer.model
#             # trainer = kwargs.get('trainer')

#             if model is not None and trainer is not None:
#                 self.make_tensor_contiguous(model)
#                 self.save_checkpoint(trainer, model, 'final', "final_c
heckpoint")
#         except Exception as e:
#             self.log(f"Error saving final checkpoint: {str(e)}")

class EvaluationLogger(TrainerCallback):
    def __init__(self, log_file="training_metrics.csv"):
        self.best_cer = float('inf')
        self.best_wer = float('inf')
        self.log_file = log_file

        # Initialize log file with headers
        with open(self.log_file, 'w') as f:
            f.write("Step,Epoch,CER,WER,Loss,Best_CER,Best_WER\n")

    def on_evaluate(self, args, state, control, metrics=None, **kwarg
s):
        torch.cuda.empty_cache()
        gc.collect()
        if metrics:
            current_step = state.global_step
            current_epoch = state.epoch

            cer = metrics.get('eval_cer', 'N/A')
            wer = metrics.get('eval_wer', 'N/A')
            loss = metrics.get('eval_loss', 'N/A')

            # Update best scores
            if isinstance(cer, (int, float)):
                self.best_cer = min(self.best_cer, cer)
            if isinstance(wer, (int, float)):
                self.best_wer = min(self.best_wer, wer)

            # Log to file
            with open(self.log_file, 'a') as f:
                f.write(f"{current_step},{current_epoch:.2f},{cer:.4
f},"
                       f"{wer:.4f},{loss:.4f},{self.best_cer:.4f},"
                       f"{self.best_wer:.4f}\n")

            # Print summary
            print("\n" + "="*50)
            print(f"Evaluation Summary:")
            print(f"Step: {current_step} (Epoch {current_epoch:.2f})")
            print(f"CER: {cer:.4f} (Best: {self.best_cer:.4f})")
            print(f"WER: {wer:.4f} (Best: {self.best_wer:.4f})")
            print(f"Loss: {loss:.4f}")
            print("="*50)
        torch.cuda.empty_cache()
        gc.collect()


class ChunkedEvaluationCallback(TrainerCallback):
    def __init__(self, chunk_size=32):
        self.chunk_size = chunk_size
        self.cer_metric = load("cer")
        self.wer_metric = load("wer")

    def on_evaluate(self, args, state, control, model=None, eval_datalo
ader=None, **kwargs):
        print("\nStarting chunked evaluation...")
        model.eval()
        device = model.device
        all_preds = []
        all_labels = []
        total_eval_loss = 0
        num_chunks = 0

        try:
            # Process evaluation data in chunks
            for i in range(0, len(eval_dataloader.dataset), self.chunk_
size):
                torch.cuda.empty_cache()
                chunk_end = min(i + self.chunk_size, len(eval_dataloade
r.dataset))

                # Get chunk of data
                chunk_data = [eval_dataloader.dataset[j] for j in range
(i, chunk_end)]
                chunk_inputs = eval_dataloader.collate_fn(chunk_data)

                # Move to device
                chunk_inputs = {k: v.to(device) for k, v in chunk_input
s.items()}

                # Forward pass
                with torch.no_grad():
                    outputs = model(**chunk_inputs)
                    logits = outputs.logits
                    loss = outputs.loss

                # Get predictions
                predictions = logits.argmax(-1).cpu()
                labels = chunk_inputs['labels'].cpu()

                # Decode predictions and labels
                decoded_preds = processor.tokenizer.batch_decode(predic
tions, skip_special_tokens=True)

                # Decode labels
                decoded_labels = []
                for label in labels:
                    label_filtered = [token for token in label if token
!= -100]
                    decoded_label = processor.tokenizer.decode(label_fi
ltered, skip_special_tokens=True)
                    decoded_labels.append(decoded_label)

                # Accumulate results
                all_preds.extend(decoded_preds)
                all_labels.extend(decoded_labels)
                total_eval_loss += loss.item()
                num_chunks += 1

                # Print progress
                print(f"\rProcessed {chunk_end}/{len(eval_dataloader.da
taset)} examples", end="")

                # Clear memory
                del chunk_inputs, outputs, logits, predictions, labels
                torch.cuda.empty_cache()
                gc.collect()

        except Exception as e:
            print(f"\nError during chunked evaluation: {str(e)}")
            raise e

        # Compute metrics
        try:
            cer = self.cer_metric.compute(predictions=all_preds, refere
nces=all_labels)
            wer = self.wer_metric.compute(predictions=all_preds, refere
nces=all_labels)
            avg_loss = total_eval_loss / num_chunks

            # Print metrics
            print(f"\nEvaluation Results at step {state.global_step}:")
            print(f"CER: {cer:.4f}")
            print(f"WER: {wer:.4f}")
            print(f"Loss: {avg_loss:.4f}")

            # Return metrics in the format the trainer expects
            return {
                "eval_cer": cer,
                "eval_wer": wer,
                "eval_loss": avg_loss
            }

        except Exception as e:
            print(f"\nError computing metrics: {str(e)}")
            raise e

from typing import Optional
from safetensors.torch import save_file
# Modify the trainer class to use chunked evaluation
class ChunkedEvalTrainer(SLiCTrainer):
    def __init__(self, *args, chunk_size=32, **kwargs):
        super().__init__(*args, **kwargs)
        self.chunk_size = chunk_size
        self.chunked_eval_callback = ChunkedEvaluationCallback(chunk_si
ze=chunk_size)

    def _save(self, output_dir: Optional[str] = None, state_dict=None):
        """Override _save to ensure tensors are contiguous before savin
g"""
        # Make the problematic tensor contiguous
        if hasattr(self.model, 'encoder') and hasattr(self.model.encode
r, 'embeddings'):
            patch_embeddings = self.model.encoder.embeddings.patch_embe
ddings
            if hasattr(patch_embeddings, 'projection'):
                patch_embeddings.projection.weight.data = patch_embeddi
ngs.projection.weight.data.contiguous()

        # If state dict is not provided, get it from model
        if state_dict is None:
            state_dict = self.model.state_dict()

        # Ensure all tensors in state dict are contiguous
        state_dict = {k: v.contiguous() if torch.is_tensor(v) and not
v.is_contiguous() else v
                     for k, v in state_dict.items()}

        self.model.config.save_pretrained(output_dir)
        if hasattr(self, 'processor'):
            self.processor.save_pretrained(output_dir)

        if hasattr(self.model, "generation_config"):
            self.model.generation_config.save_pretrained(output_dir)

        # Call parent class's _save
        return super()._save(output_dir, state_dict)

    def save_model(self, output_dir: Optional[str] = None, _internal_ca
ll: bool = False):
        """Override save_model to ensure tensors are contiguous"""
        if hasattr(self.model, 'encoder') and hasattr(self.model.encode
r, 'embeddings'):
            patch_embeddings = self.model.encoder.embeddings.patch_embe
ddings
            if hasattr(patch_embeddings, 'projection'):
                patch_embeddings.projection.weight.data = patch_embeddi
ngs.projection.weight.data.contiguous()

        self.model.config.save_pretrained(output_dir)

        if hasattr(self, 'processor'):
            self.processor.save_pretrained(output_dir)

        if hasattr(self.model, "generation_config"):
            self.model.generation_config.save_pretrained(output_dir)

        self.processor.save_pretrained(output_dir)

        return super().save_model(output_dir, _internal_call)

    def evaluate(self, eval_dataset=None, ignore_keys=None, metric_key_
prefix="eval"):
        # Use the chunked evaluation instead of standard evaluation
        eval_dataloader = self.get_eval_dataloader(eval_dataset)
        metrics = self.chunked_eval_callback.on_evaluate(
            self.args,
            self.state,
            self.control,
            model=self.model,
            eval_dataloader=eval_dataloader
        )

        self.log(metrics)
        return metrics

class EfficientCheckpointCallback(TrainerCallback):
    def __init__(self, max_checkpoints=1, verbose=True):
        self.max_checkpoints = max_checkpoints
        self.checkpoint_list = []
        self.verbose = verbose

    def log(self, message):
        if self.verbose:
            print(message)
    def cleanup_checkpoint(self, checkpoint_path):
        """Remove non-essential files and old checkpoints"""
        try:
            if os.path.exists(checkpoint_path):
                # Keep only essential files in the checkpoint
                if os.path.isdir(checkpoint_path):
                    essential_extensions = {'.bin', '.json', '.txt', '.
model', '.safetensors'}
                    for file in os.listdir(checkpoint_path):
                        file_path = os.path.join(checkpoint_path, file)
                        # Remove if not an essential file
                        if not any(file.endswith(ext) for ext in essent
ial_extensions):
                            os.remove(file_path)
                            self.log(f"Removed non-essential file: {fil
e}")

                # Remove entire checkpoint if it's old
                if checkpoint_path not in [os.path.join(self.args.outpu
t_dir, cp)
                                        for cp in self.checkpoint_list
[-self.max_checkpoints:]]:
                    shutil.rmtree(checkpoint_path)
                    self.log(f"Removed old checkpoint: {checkpoint_pat
h}")
        except Exception as e:
            self.log(f"Error during cleanup: {str(e)}")

    def on_save(self, args, state, control, **kwargs):
        self.args = args
        current_checkpoint = f"checkpoint-{state.global_step}"
        self.checkpoint_list.append(current_checkpoint)

        # Clean up checkpoints
        checkpoint_dir = args.output_dir
        if os.path.exists(checkpoint_dir):
            # Clean up old checkpoints
            for item in os.listdir(checkpoint_dir):
                if item.startswith("checkpoint-"):
                    checkpoint_path = os.path.join(checkpoint_dir, ite
m)
                    self.cleanup_checkpoint(checkpoint_path)

    def on_train_end(self, args, state, control, **kwargs):
        """Final cleanup at the end of training"""
        self.log("\nPerforming final checkpoint cleanup...")
        # Keep only the best model if it exists
        best_model_path = os.path.join(args.output_dir, "best_model")
        if os.path.exists(best_model_path):
            self.log("Keeping best model checkpoint")
            # Remove all other checkpoints
            for item in os.listdir(args.output_dir):
                if item.startswith("checkpoint-"):
                    path = os.path.join(args.output_dir, item)
                    shutil.rmtree(path)
                    self.log(f"Removed checkpoint: {path}")

def train_transformer_with_slic(line_images, texts, target_size=(256, 6
4), batch_size=2, max_length=512, val_split=0.1, initial_seed=42, label
_smoothing=0.1, gamma=2.0):
    torch.cuda.empty_cache()
    gc.collect()

    device = torch.device("cuda" if torch.cuda.is_available() else "cp
u")

    processor = TrOCRProcessor.from_pretrained(processor_path, do_resca
le=False)
    # base_model = VisionEncoderDecoderModel.from_pretrained(model_path)
    base_model = VisionEncoderDecoderModel.from_pretrained(
        model_path,
        torch_dtype=torch.float32,
        low_cpu_mem_usage=True
    )
    base_model.gradient_checkpointing_enable()

    model = TemperatureScalingModel(base_model)
    del base_model
    torch.cuda.empty_cache()
    model.config.use_cache = False

    model.config.pad_token_id = processor.tokenizer.pad_token_id
    model.config.decoder_start_token_id = processor.tokenizer.bos_token
_id



    dataset = LineDataset(processor, model, line_images, texts, target_
size, max_length, apply_augmentation=True)
```

```
    print(f"Total dataset size: {len(dataset)}")

    val_size = int(len(dataset) * val_split)
    train_size = len(dataset) - val_size
    print(f"Train size: {train_size}, Validation size: {val_size}")

    train_dataset, val_dataset = random_split(dataset, [train_size, val
_size])


    model.to(memory_format=torch.channels_last)
    model = model.to(device)
    training_args = TrainingArguments(
        output_dir="./results",
        num_train_epochs=10,
        per_device_train_batch_size=batch_size,
        per_device_eval_batch_size=batch_size,
        logging_dir="./logs",
        logging_steps=100,
        save_steps=300,
        save_total_limit=1,
        eval_steps=300,
        eval_strategy='steps',
        learning_rate=3e-5,
        weight_decay=0.01,
        load_best_model_at_end=True,
        metric_for_best_model="cer",
        greater_is_better=False,
        logging_first_step=True,
        gradient_accumulation_steps=8,
        fp16=True,
        warmup_ratio=0.1,
        report_to="all",
        dataloader_num_workers=0,
        dataloader_pin_memory=True,
        lr_scheduler_type="cosine",
        warmup_steps=500,
        eval_accumulation_steps=1
    )

    # Optimizer
    optimizer = AdamW(model.parameters(),
                      lr=training_args.learning_rate,
                      weight_decay=1e-2,
                             betas=(0.9, 0.999),
                             eps=1e-8)

    # Learning rate scheduler
    num_training_steps = len(train_dataset) * training_args.num_train_e
pochs // (training_args.per_device_train_batch_size * training_args.gra
dient_accumulation_steps)
    scheduler = get_cosine_schedule_with_warmup(optimizer, num_warmup_s
teps=1000, num_training_steps=num_training_steps)



    # trainer = SLiCTrainer(
    #     model=model,
    #     args=training_args,
    #     train_dataset=train_dataset,
    #     eval_dataset=val_dataset,
    #     data_collator=collate_fn,
    #     optimizers=(optimizer, scheduler),
    #     compute_metrics=compute_metrics,
    #     label_smoothing=label_smoothing,
    #     gamma=gamma,
    # )
    trainer = ChunkedEvalTrainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=val_dataset,
        data_collator=collate_fn,
        optimizers=(optimizer, scheduler),
        compute_metrics=compute_metrics,
        label_smoothing=label_smoothing,
        gamma=gamma,
        chunk_size=32
    )

    trainer.add_callback(EarlyStoppingCallback(early_stopping_patience=
3))
    # trainer.add_callback(TrainerCallback(on_epoch_end))
    # trainer.add_callback(CustomCallback())
    trainer.add_callback(MemoryManagementCallback())
    # trainer.add_callback(MemoryMonitorCallback(print_steps=400))
    trainer.add_callback(ErrorRecoveryCallback())
    # trainer.add_callback(VitCheckpointCallback(save_every_n_epochs=1,
verbose=True))
```

```
    trainer.add_callback(EvaluationLogger())
    trainer.add_callback(EfficientCheckpointCallback())



    # trainer.add_callback(CosineAnnealingWarmRestartsSchedulerCallback
(optimizer, T_0=num_training_steps // 10, T_mult=2, eta_min=1e-6))
    trainer.processor = processor
    trainer.train()

    # Save the model and processor
    save_dir = "large_transformer_model_v1"
    model.save_pretrained(save_dir)
    processor.save_pretrained(save_dir)

    return trainer, model.temperature.item()
```

```
Downloading builder script: 100%          5.60k/5.60k [00:00<00:00, 504kB/s]

Downloading builder script: 100%          4.49k/4.49k [00:00<00:00, 454kB/s]

preprocessor_config.json: 100%               364/364 [00:00<00:00, 34.6kB/s]

  Using a slow image processor as `use_fast` is unset and a slow proc
  essor was saved with this model. `use_fast=True` will be the defaul
  t behavior in v4.52, even if the model was saved with a slow proces
  sor. This will result in minor differences in outputs. You'll still
  be able to use a slow processor with `use_fast=False`.

tokenizer_config.json: 100%               1.38k/1.38k [00:00<00:00, 136kB/s]

vocab.json: 100%                           798k/798k [00:00<00:00, 28.0MB/s]

merges.txt: 100%                           456k/456k [00:00<00:00, 2.29MB/s]

tokenizer.json: 100%                     2.11M/2.11M [00:00<00:00, 10.3MB/s]

special_tokens_map.json: 100%                957/957 [00:00<00:00, 94.4kB/s]

config.json: 100%                         4.97k/4.97k [00:00<00:00, 451kB/s]

pytorch_model.bin: 100%                   2.44G/2.44G [00:20<00:00, 432MB/s]

model.safetensors: 100%                  2.44G/2.44G [01:33<00:00, 25.8MB/s]
```

```
Config of the encoder: <class 'transformers.models.vit.modeling_vi
t.ViTModel'> is overwritten by shared encoder config: ViTConfig {
  "attention_probs_dropout_prob": 0.0,
  "encoder_stride": 16,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.0,
  "hidden_size": 1024,
  "image_size": 384,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-12,
  "model_type": "vit",
  "num_attention_heads": 16,
  "num_channels": 3,
  "num_hidden_layers": 24,
  "patch_size": 16,
  "pooler_act": "tanh",
  "pooler_output_size": 1024,
  "qkv_bias": false,
  "torch_dtype": "float32",
  "transformers_version": "4.51.0.dev0"
}

Config of the decoder: <class 'transformers.models.trocr.modeling_t
rocr.TrOCRForCausalLM'> is overwritten by shared decoder config: Tr
OCRConfig {
  "activation_dropout": 0.0,
  "activation_function": "relu",
  "add_cross_attention": true,
  "attention_dropout": 0.0,
  "bos_token_id": 0,
  "classifier_dropout": 0.0,
  "d_model": 1024,
  "decoder_attention_heads": 16,
  "decoder_ffn_dim": 4096,
  "decoder_layerdrop": 0.0,
  "decoder_layers": 12,
  "decoder_start_token_id": 2,
  "dropout": 0.1,
  "encoder_hidden_size": 1024,
  "eos_token_id": 2,
  "init_std": 0.02,
  "is_decoder": true,
    "layernorm_embedding": false,
    "max_position_embeddings": 1024,
    "model_type": "trocr",
    "pad_token_id": 1,
    "scale_embedding": true,
    "tie_word_embeddings": false,
    "torch_dtype": "float32",
    "transformers_version": "4.51.0.dev0",
    "use_cache": false,
    "use_learned_position_embeddings": false,
    "vocab_size": 50265
  }
```

```
generation_config.json: 100%                 420/420 [00:00<00:00, 38.3kB/s]
```

```
In [ ]:
```

In [23]:
```
line_images = []
texts = []
for page_data in aligned_data.values():
    for line_img, text in page_data:
        line_images.append(line_img)
        texts.append(text)

# trainer, learned_temperature = train_transformer_with_slic(line_images, texts, target_size=(256,64), batch_size=1, max_length=128, val_split=0.1)
trainer, learned_temperature = train_transformer_with_slic(line_images, texts, target_size=(128,32), batch_size=2, max_length=128, val_split=0.1)



print(learned_temperature)
torch.save({"temperature": learned_temperature}, "learned_temperature.pth")
# model.save_pretrained("path/to/save/model")
# processor.save_pretrained("path/to/save/processor")
```

```
Config of the encoder: <class 'transformers.models.vit.modeling_vi
t.ViTModel'> is overwritten by shared encoder config: ViTConfig {
  "attention_probs_dropout_prob": 0.0,
  "encoder_stride": 16,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.0,
  "hidden_size": 1024,
  "image_size": 384,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-12,
  "model_type": "vit",
  "num_attention_heads": 16,
  "num_channels": 3,
  "num_hidden_layers": 24,
  "patch_size": 16,
  "pooler_act": "tanh",
  "pooler_output_size": 1024,
  "qkv_bias": false,
  "torch_dtype": "float32",
  "transformers_version": "4.51.0.dev0"
}

Config of the decoder: <class 'transformers.models.trocr.modeling_t
rocr.TrOCRForCausalLM'> is overwritten by shared decoder config: Tr
OCRConfig {
  "activation_dropout": 0.0,
  "activation_function": "relu",
  "add_cross_attention": true,
  "attention_dropout": 0.0,
  "bos_token_id": 0,
  "classifier_dropout": 0.0,
  "d_model": 1024,
  "decoder_attention_heads": 16,
  "decoder_ffn_dim": 4096,
  "decoder_layerdrop": 0.0,
  "decoder_layers": 12,
  "decoder_start_token_id": 2,
  "dropout": 0.1,
  "encoder_hidden_size": 1024,
  "eos_token_id": 2,
  "init_std": 0.02,
  "is_decoder": true,
  "layernorm_embedding": false,
  "max_position_embeddings": 1024,
  "model_type": "trocr",
  "pad_token_id": 1,
  "scale_embedding": true,
  "tie_word_embeddings": false,
  "torch_dtype": "float32",
  "transformers_version": "4.51.0.dev0",
  "use_cache": false,
  "use_learned_position_embeddings": false,
  "vocab_size": 50265
}
```

```
Total dataset size: 6188
Train size: 5570, Validation size: 618
```

```
wandb: WARNING The `run_name` is currently set to the same value as
`TrainingArguments.output_dir`. If this was not intended, please sp
ecify a different run name by setting the `TrainingArguments.run_na
me` parameter.
wandb: Tracking run with wandb version 0.19.1
wandb: Run data is saved locally in /kaggle/working/wandb/run-20250
329_205007-4lh6q7iq
wandb: Run `wandb offline` to turn off syncing.
wandb: Syncing run ./results
wandb: ⭐️ View project at https://wandb.ai/gary_hornbill-indian-ins
titute-of-technology-ism-dhanbad/huggingface
wandb: 🚀 View run at https://wandb.ai/gary_hornbill-indian-institu
te-of-technology-ism-dhanbad/huggingface/runs/4lh6q7iq
`loss_type=None` was set in the config but it is unrecognised.Using
the default loss: `ForCausalLMLoss`.
```

[3480/3480 10:15:33, Epoch 9.97/10]

Step	Training Loss	Validation Loss

```
Starting chunked evaluation...
Processed 618/618 examples
Evaluation Results at step 300:
CER: 0.4430
WER: 0.6812
Loss: 10.3025


/usr/local/lib/python3.10/dist-packages/transformers/configuration_
utils.py:396: UserWarning: Some non-default generation parameters a
re set in the model config. These should go into either a) `model.g
eneration_config` (as opposed to `model.config`); OR b) a Generatio
nConfig file (https://huggingface.co/docs/transformers/generation_s
trategies#save-a-custom-decoding-strategy-with-your-model).This war
ning will become an exception in the future.
Non-default generation parameters: {'max_length': 128, 'no_repeat_n
gram_size': 3}
  warnings.warn(


Removed non-essential file: scheduler.pt
Removed non-essential file: scaler.pt
Removed non-essential file: optimizer.pt
Removed non-essential file: rng_state.pth

Starting chunked evaluation...
Processed 618/618 examples
Evaluation Results at step 600:
CER: 0.3169
WER: 0.5471
Loss: 10.8303


/usr/local/lib/python3.10/dist-packages/transformers/configuration_
utils.py:396: UserWarning: Some non-default generation parameters a
re set in the model config. These should go into either a) `model.g
eneration_config` (as opposed to `model.config`); OR b) a Generatio
nConfig file (https://huggingface.co/docs/transformers/generation_s
trategies#save-a-custom-decoding-strategy-with-your-model).This war
ning will become an exception in the future.
Non-default generation parameters: {'max_length': 128, 'no_repeat_n
gram_size': 3}
  warnings.warn(
Removed non-essential file: scheduler.pt
Removed non-essential file: scaler.pt
Removed non-essential file: optimizer.pt
Removed non-essential file: rng_state.pth
Removed old checkpoint: ./results/checkpoint-300

Starting chunked evaluation...
Processed 618/618 examples
Evaluation Results at step 900:
CER: 0.2235
WER: 0.4051
Loss: 11.9249
```

```
/usr/local/lib/python3.10/dist-packages/transformers/configuration_
utils.py:396: UserWarning: Some non-default generation parameters a
re set in the model config. These should go into either a) `model.g
eneration_config` (as opposed to `model.config`); OR b) a Generatio
nConfig file (https://huggingface.co/docs/transformers/generation_s
trategies#save-a-custom-decoding-strategy-with-your-model).This war
ning will become an exception in the future.
Non-default generation parameters: {'max_length': 128, 'no_repeat_n
gram_size': 3}
  warnings.warn(
```

```
Removed old checkpoint: ./results/checkpoint-600
Removed non-essential file: scheduler.pt
Removed non-essential file: scaler.pt
Removed non-essential file: optimizer.pt
Removed non-essential file: rng_state.pth

Starting chunked evaluation...
Processed 618/618 examples
Evaluation Results at step 1200:
CER: 0.1558
WER: 0.3052
Loss: 12.1059
```

```
/usr/local/lib/python3.10/dist-packages/transformers/configuration_
utils.py:396: UserWarning: Some non-default generation parameters a
re set in the model config. These should go into either a) `model.g
eneration_config` (as opposed to `model.config`); OR b) a Generatio
nConfig file (https://huggingface.co/docs/transformers/generation_s
trategies#save-a-custom-decoding-strategy-with-your-model).This war
ning will become an exception in the future.
Non-default generation parameters: {'max_length': 128, 'no_repeat_n
gram_size': 3}
  warnings.warn(


Removed old checkpoint: ./results/checkpoint-900
Removed non-essential file: scheduler.pt
Removed non-essential file: scaler.pt
Removed non-essential file: optimizer.pt
Removed non-essential file: rng_state.pth

Starting chunked evaluation...
Processed 618/618 examples
Evaluation Results at step 1500:
CER: 0.1314
WER: 0.2555
Loss: 13.0753


/usr/local/lib/python3.10/dist-packages/transformers/configuration_
utils.py:396: UserWarning: Some non-default generation parameters a
re set in the model config. These should go into either a) `model.g
eneration_config` (as opposed to `model.config`); OR b) a Generatio
nConfig file (https://huggingface.co/docs/transformers/generation_s
trategies#save-a-custom-decoding-strategy-with-your-model).This war
ning will become an exception in the future.
Non-default generation parameters: {'max_length': 128, 'no_repeat_n
gram_size': 3}
  warnings.warn(
Removed non-essential file: scheduler.pt
Removed non-essential file: scaler.pt
Removed non-essential file: optimizer.pt
Removed non-essential file: rng_state.pth
Removed old checkpoint: ./results/checkpoint-1200

Starting chunked evaluation...
Processed 618/618 examples
Evaluation Results at step 1800:
CER: 0.0764
WER: 0.1413
Loss: 13.8873


/usr/local/lib/python3.10/dist-packages/transformers/configuration_
utils.py:396: UserWarning: Some non-default generation parameters a
re set in the model config. These should go into either a) `model.g
eneration_config` (as opposed to `model.config`); OR b) a Generatio
nConfig file (https://huggingface.co/docs/transformers/generation_s
trategies#save-a-custom-decoding-strategy-with-your-model).This war
ning will become an exception in the future.
Non-default generation parameters: {'max_length': 128, 'no_repeat_n
gram_size': 3}
  warnings.warn(


Removed old checkpoint: ./results/checkpoint-1500
Removed non-essential file: scheduler.pt
Removed non-essential file: scaler.pt
Removed non-essential file: optimizer.pt
Removed non-essential file: rng_state.pth

Starting chunked evaluation...
Processed 618/618 examples
Evaluation Results at step 2100:
CER: 0.0632
WER: 0.1199
Loss: 14.6452
```

```
/usr/local/lib/python3.10/dist-packages/transformers/configuration_
utils.py:396: UserWarning: Some non-default generation parameters a
re set in the model config. These should go into either a) `model.g
eneration_config` (as opposed to `model.config`); OR b) a Generatio
nConfig file (https://huggingface.co/docs/transformers/generation_s
trategies#save-a-custom-decoding-strategy-with-your-model).This war
ning will become an exception in the future.
Non-default generation parameters: {'max_length': 128, 'no_repeat_n
gram_size': 3}
  warnings.warn(
```

```
Removed old checkpoint: ./results/checkpoint-1800
Removed non-essential file: scheduler.pt
Removed non-essential file: scaler.pt
Removed non-essential file: optimizer.pt
Removed non-essential file: rng_state.pth

Starting chunked evaluation...
Processed 618/618 examples
Evaluation Results at step 2400:
CER: 0.0499
WER: 0.0993
Loss: 15.2085
```

```
/usr/local/lib/python3.10/dist-packages/transformers/configuration_
utils.py:396: UserWarning: Some non-default generation parameters a
re set in the model config. These should go into either a) `model.g
eneration_config` (as opposed to `model.config`); OR b) a Generatio
nConfig file (https://huggingface.co/docs/transformers/generation_s
trategies#save-a-custom-decoding-strategy-with-your-model).This war
ning will become an exception in the future.
Non-default generation parameters: {'max_length': 128, 'no_repeat_n
gram_size': 3}
  warnings.warn(
Removed non-essential file: scheduler.pt
Removed non-essential file: scaler.pt
Removed non-essential file: optimizer.pt
Removed non-essential file: rng_state.pth
Removed old checkpoint: ./results/checkpoint-2100

Starting chunked evaluation...
Processed 618/618 examples
Evaluation Results at step 2700:
CER: 0.0438
WER: 0.0852
Loss: 15.5766


/usr/local/lib/python3.10/dist-packages/transformers/configuration_
utils.py:396: UserWarning: Some non-default generation parameters a
re set in the model config. These should go into either a) `model.g
eneration_config` (as opposed to `model.config`); OR b) a Generatio
nConfig file (https://huggingface.co/docs/transformers/generation_s
trategies#save-a-custom-decoding-strategy-with-your-model).This war
ning will become an exception in the future.
Non-default generation parameters: {'max_length': 128, 'no_repeat_n
gram_size': 3}
  warnings.warn(


Removed non-essential file: scheduler.pt
Removed non-essential file: scaler.pt
Removed non-essential file: optimizer.pt
Removed non-essential file: rng_state.pth
Removed old checkpoint: ./results/checkpoint-2400

Starting chunked evaluation...
Processed 618/618 examples
Evaluation Results at step 3000:
CER: 0.0388
WER: 0.0789
Loss: 15.7433
```

```
/usr/local/lib/python3.10/dist-packages/transformers/configuration_
utils.py:396: UserWarning: Some non-default generation parameters a
re set in the model config. These should go into either a) `model.g
eneration_config` (as opposed to `model.config`); OR b) a Generatio
nConfig file (https://huggingface.co/docs/transformers/generation_s
trategies#save-a-custom-decoding-strategy-with-your-model).This war
ning will become an exception in the future.
Non-default generation parameters: {'max_length': 128, 'no_repeat_n
gram_size': 3}
  warnings.warn(


Removed non-essential file: scheduler.pt
Removed non-essential file: scaler.pt
Removed non-essential file: optimizer.pt
Removed non-essential file: rng_state.pth
Removed old checkpoint: ./results/checkpoint-2700

Starting chunked evaluation...
Processed 618/618 examples
Evaluation Results at step 3300:
CER: 0.0393
WER: 0.0765
Loss: 15.9789


/usr/local/lib/python3.10/dist-packages/transformers/configuration_
utils.py:396: UserWarning: Some non-default generation parameters a
re set in the model config. These should go into either a) `model.g
eneration_config` (as opposed to `model.config`); OR b) a Generatio
nConfig file (https://huggingface.co/docs/transformers/generation_s
trategies#save-a-custom-decoding-strategy-with-your-model).This war
ning will become an exception in the future.
Non-default generation parameters: {'max_length': 128, 'no_repeat_n
gram_size': 3}
  warnings.warn(


Removed old checkpoint: ./results/checkpoint-3000
Removed non-essential file: scheduler.pt
Removed non-essential file: scaler.pt
Removed non-essential file: optimizer.pt
Removed non-essential file: rng_state.pth
```

```
/usr/local/lib/python3.10/dist-packages/transformers/configuration_
utils.py:396: UserWarning: Some non-default generation parameters a
re set in the model config. These should go into either a) `model.g
eneration_config` (as opposed to `model.config`); OR b) a Generatio
nConfig file (https://huggingface.co/docs/transformers/generation_s
trategies#save-a-custom-decoding-strategy-with-your-model).This war
ning will become an exception in the future.
Non-default generation parameters: {'max_length': 128, 'no_repeat_n
gram_size': 3}
  warnings.warn(


Removed non-essential file: scheduler.pt
Removed non-essential file: scaler.pt
Removed non-essential file: optimizer.pt
Removed non-essential file: rng_state.pth
Removed old checkpoint: ./results/checkpoint-3300


Could not locate the best model at ./results/checkpoint-3000/pytorc
h_model.bin, if you are running a distributed training on multiple
nodes, you should activate `--save_on_each_node`.
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-23-e2a67e4c2caf> in <cell line: 9>()
      7 
      8 # trainer, learned_temperature = train_transformer_with_slic(line_images, texts, target_size=(256,64), batch_size=1, max_length=128, val_split=0.1)
----> 9 trainer, learned_temperature = train_transformer_with_slic(line_images, texts, target_size=(128,32), batch_size=2, max_length=128, val_split=0.1)
     10 
     11 

<ipython-input-22-ed051a0b65a5> in train_transformer_with_slic(line_images, texts, target_size, batch_size, max_length, val_split, initial_seed, label_smoothing, gamma)
   1012     # trainer.add_callback(CosineAnnealingWarmRestartsSchedulerCallback(optimizer, T_0=num_training_steps // 10, T_mult=2, eta_min=1e-6))
   1013     trainer.processor = processor
-> 1014     trainer.train()
   1015 
   1016     # Save the model and processor

/usr/local/lib/python3.10/dist-packages/transformers/trainer.py in train(self, resume_from_checkpoint, trial, ignore_keys_for_eval, **kwargs)
   2243                 hf_hub_utils.enable_progress_bars()
   2244         else:
-> 2245             return inner_training_loop(
   2246                 args=args,
   2247                 resume_from_checkpoint=resume_from_checkpoint,

/usr/local/lib/python3.10/dist-packages/transformers/trainer.py in _inner_training_loop(self, batch_size, args, resume_from_checkpoint, trial, ignore_keys_for_eval)
   2716         if self.args.should_save and self.state.best_model_checkpoint is not None and self.args.save_total_limit == 1:
   2717             for checkpoint in checkpoints_sorted:
-> 2718                 if not os.path.samefile(checkpoint, self.st
ate.best_model_checkpoint):
   2719                     logger.info(f"Deleting older checkpoint
[{checkpoint}] due to args.save_total_limit")
   2720                     shutil.rmtree(checkpoint, ignore_errors
=True)

/usr/lib/python3.10/genericpath.py in samefile(f1, f2)
     99     """
    100     s1 = os.stat(f1)
--> 101     s2 = os.stat(f2)
    102     return samestat(s1, s2)
    103

FileNotFoundError: [Errno 2] No such file or directory: './results/
checkpoint-3000'
```

```
In [ ]:
```