Open in Colab

Install Libraries

In [1]:

```
# Install Libraries (Install if needed)
!apt-get install -y tesseract-ocr
!pip install pytesseract
!pip install torch torchvision transformers tqdm
!pip install tqdm
!pip install tensorflow matplotlib opencv-python-headless
!pip install easyocr
!pip install pyarrow
!pip install fastparquet
!pip install datasets
!pip install evaluate
!pip install jiwer
!pip install paddleocr
!pip install paddlepaddle
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
tesseract-ocr is already the newest version (4.1.1-2.1build1).
0 upgraded, 0 newly installed, 0 to remove and 57 not upgraded.
Requirement already satisfied: pytesseract in /usr/local/lib/python3.10/dist-packages (0.3.1
3)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (f
rom pytesseract) (24.2)
Requirement already satisfied: Pillow>=8.0.0 in /usr/local/lib/python3.10/dist-packages (fro
m pytesseract) (11.0.0)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.5.1+cu12
1)
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-packages (0.20.
1+cu121)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.4
6.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (4.66.6)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from tor
ch) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-pa
ckages (from torch) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from tor
ch) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torc
h) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torc
h) (2024.9.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (fro
m torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages
(from sympy==1.13.1->torch) (1.3.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torchv
ision) (1.26.4)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.10/dist-packa
ges (from torchvision) (11.0.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /usr/local/lib/python3.10/dis
t-packages (from transformers) (0.26.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (f
rom transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from
transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages
(from transformers) (2024.9.11)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from tra
nsformers) (2.32.3)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages
(from transformers) (0.4.5)
Requirement already satisfied: tokenizers<0.21,>=0.20 in /usr/local/lib/python3.10/dist-pack
ages (from transformers) (0.20.3)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (f
rom jinja2->torch) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-pa
ckages (from requests->transformers) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from
requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages
(from requests->transformers) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages
(from requests->transformers) (2024.8.30)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (4.66.6)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.17.
1)
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.8.0)
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.10/dist-pack
ages (4.10.0.84)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (fr
om tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages
(from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.10/dist-packag
es (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python
3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-package
s (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.10/dist-packages (from
tensorflow) (3.12.1)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages
(from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in /usr/local/lib/python3.10/dist-pac
kages (from tensorflow) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages
(from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from te
nsorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.
5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.25.5)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-package
s (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from t
ensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from
tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages
(from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-pa
ckages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (fro
m tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-package
s (from tensorflow) (1.68.0)
Requirement already satisfied: tensorboard<2.18,>=2.17 in /usr/local/lib/python3.10/dist-pac
kages (from tensorflow) (2.17.1)
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from
tensorflow) (3.5.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python
3.10/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packag
es (from tensorflow) (1.26.4)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages
(from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from
matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages
(from matplotlib) (4.55.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages
(from matplotlib) (1.4.7)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (fro
m matplotlib) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages
(from matplotlib) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib) (2.8.2)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages
(from astunparse>=1.6.0->tensorflow) (0.45.0)
```

```
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages (from keras>=
3.2.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages (from keras>
=3.2.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages (from keras
>=3.2.0->tensorflow) (0.13.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-pa
ckages (from requests<3,>=2.21.0->tensorflow) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages
(from requests<3,>=2.21.0->tensorflow) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages
(from requests<3,>=2.21.0->tensorflow) (2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (f
rom tensorboard<2.18,>=2.17->tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/pytho
n3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (f
rom tensorboard<2.18,>=2.17->tensorflow) (3.1.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages
(from werkzeug>=1.0.1->tensorboard<2.18,>=2.17->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packa
ges (from rich->keras>=3.2.0->tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-pac
kages (from rich->keras>=3.2.0->tensorflow) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from m
arkdown-it-py>=2.2.0->rich->keras>=3.2.0->tensorflow) (0.1.2)
Requirement already satisfied: easyocr in /usr/local/lib/python3.10/dist-packages (1.7.2)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (from easyoc
r) (2.5.1+cu121)
Requirement already satisfied: torchvision>=0.5 in /usr/local/lib/python3.10/dist-packages
(from easyocr) (0.20.1+cu121)
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.10/dist-pack
ages (from easyocr) (4.10.0.84)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from easyoc
r) (1.13.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from easyoc
r) (1.26.4)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from easyo
cr) (11.0.0)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.10/dist-packages (from
easyocr) (0.24.0)
Requirement already satisfied: python-bidi in /usr/local/lib/python3.10/dist-packages (from
easyocr) (0.6.3)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages (from easyo
cr) (6.0.2)
Requirement already satisfied: Shapely in /usr/local/lib/python3.10/dist-packages (from easy
ocr) (2.0.6)
Requirement already satisfied: pyclipper in /usr/local/lib/python3.10/dist-packages (from ea
syocr) (1.3.0.post6)
Requirement already satisfied: ninja in /usr/local/lib/python3.10/dist-packages (from easyoc
r) (1.11.1.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from tor
ch->easyocr) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-pa
ckages (from torch->easyocr) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from tor
ch->easyocr) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch
->easyocr) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch
->easyocr) (2024.9.0)
```

```
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (fro
m torch->easyocr) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages
(from sympy==1.13.1->torch->easyocr) (1.3.0)
Requirement already satisfied: imageio>=2.33 in /usr/local/lib/python3.10/dist-packages (fro
m scikit-image->easyocr) (2.36.0)
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/lib/python3.10/dist-package
s (from scikit-image->easyocr) (2024.9.20)
Requirement already satisfied: packaging>=21 in /usr/local/lib/python3.10/dist-packages (fro
m scikit-image->easyocr) (24.2)
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.10/dist-packages
(from scikit-image->easyocr) (0.4)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (f
rom jinja2->torch->easyocr) (3.0.2)
Requirement already satisfied: pyarrow in /usr/local/lib/python3.10/dist-packages (17.0.0)
Requirement already satisfied: numpy>=1.16.6 in /usr/local/lib/python3.10/dist-packages (fro
m pyarrow) (1.26.4)
Requirement already satisfied: fastparquet in /usr/local/lib/python3.10/dist-packages (2024.
11.0)
Requirement already satisfied: pandas>=1.5.0 in /usr/local/lib/python3.10/dist-packages (fro
m fastparquet) (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from fastpa
rquet) (1.26.4)
Requirement already satisfied: cramjam>=2.3 in /usr/local/lib/python3.10/dist-packages (from
fastparquet) (2.9.0)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from fastp
arquet) (2024.9.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from fa
stparquet) (24.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-pack
ages (from pandas>=1.5.0->fastparquet) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from
pandas>=1.5.0->fastparquet) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (fr
om pandas>=1.5.0->fastparquet) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from pyt
hon-dateutil>=2.8.2->pandas>=1.5.0->fastparquet) (1.16.0)
Requirement already satisfied: datasets in /usr/local/lib/python3.10/dist-packages (3.1.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from dat
asets) (3.16.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from
datasets) (1.26.4)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.10/dist-packages (f
rom datasets) (17.0.0)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3.10/dist-packages
(from datasets) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from datas
ets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.10/dist-packages
(from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.10/dist-packages (from
datasets) (4.66.6)
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packages (from datas
ets) (3.5.0)
Requirement already satisfied: multiprocess<0.70.17 in /usr/local/lib/python3.10/dist-packag
es (from datasets) (0.70.16)
Requirement already satisfied: fsspec<=2024.9.0,>=2023.1.0 in /usr/local/lib/python3.10/dist
-packages (from fsspec[http]<=2024.9.0,>=2023.1.0->datasets) (2024.9.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from data
sets) (3.11.2)
Requirement already satisfied: huggingface-hub>=0.23.0 in /usr/local/lib/python3.10/dist-pac
kages (from datasets) (0.26.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from da
```

```
tasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from
datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-pac
kages (from aiohttp->datasets) (2.4.3)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages
(from aiohttp->datasets) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (fro
m aiohttp->datasets) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages
(from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-package
s (from aiohttp->datasets) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.10/dist-packages
(from aiohttp->datasets) (0.2.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.10/dist-packages
(from aiohttp->datasets) (1.17.2)
Requirement already satisfied: async-timeout<6.0,>=4.0 in /usr/local/lib/python3.10/dist-pac
kages (from aiohttp->datasets) (4.0.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.23.0->datasets) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-pa
ckages (from requests>=2.32.2->datasets) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from
requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages
(from requests>=2.32.2->datasets) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages
(from requests>=2.32.2->datasets) (2024.8.30)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-pack
ages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from
pandas->datasets) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (fr
om pandas->datasets) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from pyt
hon-dateutil>=2.8.2->pandas->datasets) (1.16.0)
Requirement already satisfied: evaluate in /usr/local/lib/python3.10/dist-packages (0.4.3)
Requirement already satisfied: datasets>=2.0.0 in /usr/local/lib/python3.10/dist-packages (f
rom evaluate) (3.1.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from
evaluate) (1.26.4)
Requirement already satisfied: dill in /usr/local/lib/python3.10/dist-packages (from evaluat
e) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from evalu
ate) (2.2.2)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages
(from evaluate) (2.32.3)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-packages (from
evaluate) (4.66.6)
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packages (from evalu
ate) (3.5.0)
Requirement already satisfied: multiprocess in /usr/local/lib/python3.10/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) (2024.9.0)
Requirement already satisfied: huggingface-hub>=0.7.0 in /usr/local/lib/python3.10/dist-pack
ages (from evaluate) (0.26.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from ev
aluate) (24.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from dat
asets>=2.0.0->evaluate) (3.16.1)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.10/dist-packages (f
```

```
rom datasets>=2.0.0->evaluate) (17.0.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from data
sets>=2.0.0->evaluate) (3.11.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from
datasets>=2.0.0->evaluate) (6.0.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.7.0->evaluate) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-pa
ckages (from requests>=2.19.0->evaluate) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from
requests>=2.19.0->evaluate) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages
(from requests>=2.19.0->evaluate) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages
(from requests>=2.19.0->evaluate) (2024.8.30)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-pack
ages (from pandas->evaluate) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from
pandas->evaluate) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (fr
om pandas->evaluate) (2024.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-pac
kages (from aiohttp->datasets>=2.0.0->evaluate) (2.4.3)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages
(from aiohttp->datasets>=2.0.0->evaluate) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (fro
m aiohttp->datasets>=2.0.0->evaluate) (24.2.0)
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/lib/python3.10/dist-package
s (from aiohttp->datasets>=2.0.0->evaluate) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.10/dist-packages
(from aiohttp->datasets>=2.0.0->evaluate) (0.2.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.10/dist-packages
(from aiohttp->datasets>=2.0.0->evaluate) (1.17.2)
Requirement already satisfied: async-timeout<6.0,>=4.0 in /usr/local/lib/python3.10/dist-pac
kages (from aiohttp->datasets>=2.0.0->evaluate) (4.0.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from pyt
hon-dateutil>=2.8.2->pandas->evaluate) (1.16.0)
Requirement already satisfied: jiwer in /usr/local/lib/python3.10/dist-packages (3.0.5)
Requirement already satisfied: click<9.0.0,>=8.1.3 in /usr/local/lib/python3.10/dist-package
s (from jiwer) (8.1.7)
Requirement already satisfied: rapidfuzz<4,>=3 in /usr/local/lib/python3.10/dist-packages (f
rom jiwer) (3.10.1)
```

Import Libraries

In [64]:

```python
# --- SYSTEM AND FILE HANDLING ---
import os
import random
from zipfile import ZipFile
from shutil import copyfile

# --- DATA MANIPULATION ---
import json
import numpy as np
import pandas as pd

# --- METRICS AND ERROR MEASUREMENTS ---
from jiwer import cer, wer
```

```
# --- IMAGE PROCESSING ---
import cv2
from PIL import Image, ImageStat
import pytesseract

# --- DATA VISUALIZATION ---
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import seaborn as sns
from tqdm import tqdm

# --- DEEP LEARNING LIBRARIES ---
import torch
from torch.utils.data import DataLoader
from torchvision.transforms import Compose, Resize, ToTensor

# --- TRANSFORMERS AND OCR MODELS ---
from transformers import (
    TrOCRProcessor,
    VisionEncoderDecoderModel,
    Seq2SeqTrainer,
    Seq2SeqTrainingArguments,
    AdamW,
    get_scheduler,
)
from datasets import Dataset, DatasetDict
import evaluate  # Correctly replace load_metric

# --- OCR LIBRARIES ---
import easyocr

# --- WARNINGS SUPPRESSION ---
import warnings
import logging
from transformers import logging as hf_logging

# Set up logging to suppress unnecessary warnings and info messages
warnings.filterwarnings("ignore")  # Suppress all Python warnings
hf_logging.set_verbosity_error()  # Suppress specific transformers warnings
logging.getLogger("transformers").setLevel(logging.ERROR)  # Set logging level for transfor
logging.getLogger("ppocr").setLevel(logging.WARNING)  # Suppress DEBUG information from Pad
```

Collect and Load Data

In [65]:

```
# --- MOUNT GOOGLE DRIVE ---
from google.colab import drive

print("Mounting Google Drive...")
drive.mount('/content/drive')

google_drive_path = "/content/drive/My Drive/Colab Notebooks/AAI-521-Computer Vision Datase
print("Available files:", os.listdir(google_drive_path))
```

```
Mounting Google Drive...
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/
content/drive", force_remount=True).
Available files: ['TextOCR Dataset.zip', 'TextOCR Dataset.zip (Unzipped Files)', 'TextOCR_te
st_Images_filtered.zip', 'preprocessed_images.zip', 'output_images.zip', 'Model_TcOCR_Final_
Version2.ipynb']
```

Understanding Data

In [66]:

```
# Correct path to the dataset
google_drive_zip_path = "/content/drive/My Drive/Colab Notebooks/AAI-521-Computer Vision Da
unzipped_folder = "/content/unzipped_dataset"


def unzip_dataset(zip_path, extract_to):
    if os.path.exists(zip_path):
        print(f"Extracting dataset from {zip_path}...")

        # Open the zip file
        with ZipFile(zip_path, 'r') as zip_ref:
            # Get the list of files to extract
            file_list = zip_ref.namelist()

            # Create a progress bar for extraction
            for file in tqdm(file_list, desc="Extracting", unit="file"):
                zip_ref.extract(file, extract_to)

        print(f"Dataset extracted to {extract_to}")
    else:
        raise FileNotFoundError(f"Zip file not found at {zip_path}")


# Unzip the dataset with progress bar
unzip_dataset(google_drive_zip_path, unzipped_folder)

# List the unzipped dataset
print("\n----- Top Level Files -----")
!ls /content/unzipped_dataset

# Optionally, find all directories within the unzipped dataset
print("\n----- Dataset Directory -----")
!find /content/unzipped_dataset -type d
```

```
Extracting dataset from /content/drive/My Drive/Colab Notebooks/AAI-521-Computer Vision Data
set/TextOCR Dataset.zip...
Extracting: 100%|██████████| 25124/25124 [00:43<00:00, 580.30file/s]
Dataset extracted to /content/unzipped_dataset

----- Top Level Files -----
annot.csv  annot.parquet  img.csv  img.parquet  TextOCR_0.1_train.json  train_val_images

----- Dataset Directory -----
/content/unzipped_dataset
/content/unzipped_dataset/train_val_images
/content/unzipped_dataset/train_val_images/train_images
```

In [67]:

```
# ~~~~~ Top Level Files ~~~~~ Data Sample
# Correct paths to dataset files and folders
base_path = "/content/unzipped_dataset"
csv_files = [os.path.join(base_path, "annot.csv"), os.path.join(base_path, "img.csv")]
parquet_files = [os.path.join(base_path, "annot.parquet"), os.path.join(base_path, "img.par
json_file = os.path.join(base_path, "TextOCR_0.1_train.json")
image_folder = os.path.join(base_path, "train_val_images/train_images")

# Prepare a dictionary to store file samples for visualization
file_samples = {}
```

```python
# Load and collect CSV file samples
for file in csv_files:
    try:
        df = pd.read_csv(file)
        file_samples[file] = df.head()  # Collect first 5 rows
    except Exception as e:
        file_samples[file] = pd.DataFrame({"Error": [str(e)]})  # Collect error as DataFra

# Load and collect Parquet file samples
for file in parquet_files:
    try:
        df = pd.read_parquet(file)
        file_samples[file] = df.head()  # Collect first 5 rows
    except Exception as e:
        file_samples[file] = pd.DataFrame({"Error": [str(e)]})  # Collect error as DataFra

# Load and collect JSON file samples
try:
    with open(json_file, 'r') as f:
        data = json.load(f)
        # Convert first 5 keys and values to DataFrame
        json_sample = {k: data[k] for k in list(data.keys())[:5]}
        file_samples[json_file] = pd.DataFrame.from_dict(json_sample, orient='index', colu
except Exception as e:
    file_samples[json_file] = pd.DataFrame({"Error": [str(e)]})  # Collect error as DataFr

# Load and collect image folder sample
try:
    image_sample = os.listdir(image_folder)[:5]
    file_samples[image_folder] = pd.DataFrame({"Images": image_sample})  # Convert filenam
except Exception as e:
    file_samples[image_folder] = pd.DataFrame({"Error": [str(e)]})  # Collect error as Dat

# Display DataFrame contents
for file_path, sample_df in file_samples.items():
    print(f"--- Sample from {file_path} ---")
    print(sample_df)
```

```
--- Sample from /content/unzipped_dataset/annot.csv ---
   Unnamed: 0                 id          image_id  \
0           0  a4ea732cd3d5948a_1  a4ea732cd3d5948a
1           1  a4ea732cd3d5948a_2  a4ea732cd3d5948a
2           2  a4ea732cd3d5948a_3  a4ea732cd3d5948a
3           3  a4ea732cd3d5948a_4  a4ea732cd3d5948a
4           4  a4ea732cd3d5948a_5  a4ea732cd3d5948a

                          bbox  utf8_string  \
0   [525.83, 3.4, 197.64, 33.94]  Performance
1  [534.67, 64.68, 91.22, 38.19]        Sport
2  [626.95, 63.62, 96.52, 31.82]        Watch
3  [577.4, 141.87, 147.13, 43.1]   ...period.
4  [391.03, 163.9, 60.82, 38.65]            .

                                          points     area
0  [525.83, 3.4, 723.47, 7.29, 722.76, 36.99, 525...  6707.90
1  [535.73, 64.68, 623.41, 67.51, 625.89, 102.87,...  3483.69
2  [626.95, 63.62, 721.7, 63.62, 723.47, 95.44, 6...  3071.27
3  [580.02, 143.61, 724.53, 141.87, 723.66, 184.9...  6341.30
4  [395.2, 163.9, 451.85, 191.94, 445.59, 202.55,...  2350.69
--- Sample from /content/unzipped_dataset/img.csv ---
   Unnamed: 0               id  width  height    set  \
0           0  a4ea732cd3d5948a    840    1024  train
1           1  4bf43a7b2a898044   1024     683  train
2           2  1b55b309b0f50d02   1024     683  train
3           3  00c359f294f7dcd9   1024     680  train
4           4  04b5a37f762b0f51    768    1024  train

                   file_name
0  train/a4ea732cd3d5948a.jpg
1  train/4bf43a7b2a898044.jpg
2  train/1b55b309b0f50d02.jpg
3  train/00c359f294f7dcd9.jpg
4  train/04b5a37f762b0f51.jpg
--- Sample from /content/unzipped_dataset/annot.parquet ---
                   id          image_id                          bbox  \
0  a4ea732cd3d5948a_1  a4ea732cd3d5948a   [525.83, 3.4, 197.64, 33.94]
1  a4ea732cd3d5948a_2  a4ea732cd3d5948a  [534.67, 64.68, 91.22, 38.19]
2  a4ea732cd3d5948a_3  a4ea732cd3d5948a  [626.95, 63.62, 96.52, 31.82]
3  a4ea732cd3d5948a_4  a4ea732cd3d5948a  [577.4, 141.87, 147.13, 43.1]
4  a4ea732cd3d5948a_5  a4ea732cd3d5948a  [391.03, 163.9, 60.82, 38.65]

   utf8_string                                          points     area
0  Performance  [525.83, 3.4, 723.47, 7.29, 722.76, 36.99, 525...  6707.90
1        Sport  [535.73, 64.68, 623.41, 67.51, 625.89, 102.87,...  3483.69
2        Watch  [626.95, 63.62, 721.7, 63.62, 723.47, 95.44, 6...  3071.27
3   ...period.  [580.02, 143.61, 724.53, 141.87, 723.66, 184.9...  6341.30
4            .  [395.2, 163.9, 451.85, 191.94, 445.59, 202.55,...  2350.69
--- Sample from /content/unzipped_dataset/img.parquet ---
                 id  width  height    set                   file_name
0  a4ea732cd3d5948a    840    1024  train  train/a4ea732cd3d5948a.jpg
1  4bf43a7b2a898044   1024     683  train  train/4bf43a7b2a898044.jpg
2  1b55b309b0f50d02   1024     683  train  train/1b55b309b0f50d02.jpg
3  00c359f294f7dcd9   1024     680  train  train/00c359f294f7dcd9.jpg
4  04b5a37f762b0f51    768    1024  train  train/04b5a37f762b0f51.jpg
--- Sample from /content/unzipped_dataset/TextOCR_0.1_train.json ---
Empty DataFrame
Columns: [Value]
Index: []
--- Sample from /content/unzipped_dataset/train_val_images/train_images ---
                Images
0  0de6722aa3a43476.jpg
```

```
1  54e02735ca3dd823.jpg
2  961aa07dd7b5f2d8.jpg
3  b5c2c50202792f10.jpg
4  fdf77957cc71b10d.jpg
```

Achieving Ground Truth

In [68]:

```python
# --- Loading Annotations ---
json_path = "/content/unzipped_dataset/TextOCR_0.1_train.json"  # Path to JSON annotation j
annot_csv_path = "/content/unzipped_dataset/annot.csv"          # CSV annotations file
img_csv_path = "/content/unzipped_dataset/img.csv"              # Image metadata CSV
image_folder = "/content/unzipped_dataset/train_val_images/train_images"  # Folder with ima

# Load JSON annotations
with open(json_path, "r") as f:
    json_data = json.load(f)  # Load the JSON annotations

# Load CSV annotations and image metadata
annot_df = pd.read_csv(annot_csv_path)  # Load annotations from CSV
img_metadata_df = pd.read_csv(img_csv_path)  # Load image metadata

# Merge CSV annotations and image metadata
merged_df = pd.merge(
    annot_df, img_metadata_df,
    left_on="image_id", right_on="id",
    how="inner"
)

# Add file paths to the merged DataFrame
merged_df["file_path"] = merged_df["file_name"].apply(
    lambda x: os.path.join(image_folder, x)
)

# --- Preparing Ground Truth Data ---
image_text_pairs = []

# Validate JSON structure and debug keys
if "imgs" in json_data and "anns" in json_data and "imgToAnns" in json_data:
    print("JSON keys are valid. Proceeding...")
else:
    print("JSON keys missing. Check JSON structure:")
    print(json_data.keys())

for img_id, metadata in json_data.get("imgs", {}).items():
    file_name = metadata.get("file_name", "")
    corrected_file_name = file_name.replace("train/", "")
    img_path = os.path.join(image_folder, corrected_file_name)

    if not os.path.exists(img_path):
        print(f"Image path not found: {img_path}")
        continue

    # Debugging the annotations linked to the image
    ann_ids = json_data.get("imgToAnns", {}).get(img_id, [])
    if not ann_ids:
        print(f"No annotations found for image ID: {img_id}")
        continue

    text_annotations = []
    for ann_id in ann_ids:
        annotation = json_data["anns"].get(ann_id, {})
```

```
        utf8_string = annotation.get("utf8_string", "")
        if not utf8_string:
            print(f"Annotation missing text for ID: {ann_id}")
        else:
            text_annotations.append(utf8_string)

    # Concatenate annotations
    full_text = " ".join(text_annotations)
    image_text_pairs.append({"image_path": img_path, "text": full_text})

if not image_text_pairs:
    print("No valid image-text pairs were found. Debug the JSON annotations further.")

# Convert to DataFrame
ground_truth_df = pd.DataFrame(image_text_pairs)

# Display a sample of the prepared ground truth data
print(ground_truth_df.head())
```

```
JSON keys are valid. Proceeding...
                                          image_path  \
0  /content/unzipped_dataset/train_val_images/tra...
1  /content/unzipped_dataset/train_val_images/tra...
2  /content/unzipped_dataset/train_val_images/tra...
3  /content/unzipped_dataset/train_val_images/tra...
4  /content/unzipped_dataset/train_val_images/tra...

                                                text
0  Performance Sport Watch ...period. . 400 300 1...
1  400 Z 7 at nLa A. James LYNCH REAL ESTATE 781....
2  CAOL ILA DISTILLERY 1996 GLE MALT SCOTCH WHISK...
3                                  G-ATCO HUSKY +
4  OUR NEIGHBORS, THE FRIENDS . . . 1 Muhomah . f...
```

Ground Truth Image Sample

In [69]:

```
def visualize_ground_truth(image_path, image_annotations):
    """
    Visualize the ground truth annotations (bounding boxes and text) on the image.
    """
    image = Image.open(image_path)
    fig, ax = plt.subplots(figsize=(10, 10))
    ax.imshow(image)

    for annotation in image_annotations:
        bbox = annotation["bbox"]  # [x, y, width, height]
        text = annotation["utf8_string"]

        # Draw bounding box
        rect = patches.Rectangle(
            (bbox[0], bbox[1]), bbox[2], bbox[3], linewidth=2, edgecolor="red", facecolor="
        )
        ax.add_patch(rect)

        # Display text
        ax.text(
            bbox[0], bbox[1] - 10, text, fontsize=12, color="blue",
            bbox=dict(facecolor="white", alpha=0.7, edgecolor="none", boxstyle="round,pad=0
        )
```

```
    plt.axis("off")
    plt.show()
```

In [70]:

```
# Function to visualize the image before annotations
def visualize_before_merge(image_path):
    """
    Display the image without annotations (before merging).

    Args:
    - image_path (str): Path to the image file.
    """
    if not os.path.exists(image_path):
        print(f"Image not found: {image_path}")
        return

    # Load the image
    image = plt.imread(image_path)

    # Display the image
    plt.figure(figsize=(10, 8))
    plt.imshow(image)
    plt.title("Before Ground Truth Merging", fontsize=16)
    plt.axis('off')
    plt.show()

# Function to visualize the image after annotations
def visualize_after_merge(image_path, annotations):
    """
    Visualize the image with bounding boxes and labels (after merging).

    Args:
    - image_path (str): Path to the image file.
    - annotations (list of tuples): List of (label, bbox) tuples.
    """
    if not os.path.exists(image_path):
        print(f"Image not found: {image_path}")
        return

    # Load the image
    image = plt.imread(image_path)

    # Set up the plot
    fig, ax = plt.subplots(1, figsize=(15, 10))
    ax.imshow(image)

    # Add bounding boxes and labels
    for label, bbox in annotations:
        # Ensure bbox is a list (if stored as a string)
        if isinstance(bbox, str):
            bbox = eval(bbox)

        # Draw the bounding box
        rect = patches.Rectangle(
            (bbox[0], bbox[1]), bbox[2], bbox[3], linewidth=2, edgecolor="red", facecolor='
        )
        ax.add_patch(rect)

        # Add the label above the bounding box
        ax.text(
            bbox[0], bbox[1] - 10, label, fontsize=10, color="blue",
            bbox=dict(facecolor="white", alpha=0.8)
```

```
        )

    plt.axis('off')
    plt.title("After Ground Truth Merging", fontsize=16)
    plt.show()


# File path for the specific image
image_path = "/content/unzipped_dataset/train_val_images/train_images/4bf43a7b2a898044.jpg"

# Define the annotations (label and bbox pairs)
annotations = [
    ("400", [429.44, 163.75, 27.52, 13.0]),
    ("Z", [2.52, 206.33, 14.29, 18.56]),
    ("7", [1.21, 229.65, 5.91, 11.33]),
    ("at", [8.11, 232.44, 8.54, 8.7]),
    ("nLa", [0.23, 246.23, 17.24, 12.81]),
    ("A.", [138.01, 192.86, 13.63, 11.33]),
    ("James", [153.61, 193.36, 38.43, 10.83]),
    ("LYNCH", [115.51, 202.88, 98.7, 27.43]),
    ("REAL", [94.32, 231.13, 24.47, 10.18]),
    ("ESTATE", [119.61, 231.29, 34.98, 10.18]),
    ("781.599.1599", [96.13, 240.32, 57.48, 11.17]),
    ("INSURANCE", [181.53, 230.96, 54.68, 9.69]),
    ("781.598.4700", [180.38, 240.65, 57.31, 10.02]),
    (".", [123.88, 252.48, 82.44, 9.52]),
    (".", [306.66, 215.53, 13.14, 9.85]),
    (".", [321.12, 217.66, 13.46, 10.18]),
    (".", [508.81, 382.92, 19.62, 6.98]),
    ("12", [439.01, 412.72, 71.5, 46.03]),
    (".", [571.09, 216.87, 9.19, 9.86]),
    (".", [656.64, 282.71, 32.03, 32.88]),
    ("23", [676.66, 292.43, 40.31, 41.17]),
    ("n's", [672.66, 197.52, 23.62, 19.94]),
    ("Roast", [696.93, 197.95, 47.67, 19.51]),
    ("Beef", [745.04, 195.79, 42.47, 26.65]),
    ("Seafood", [692.6, 218.97, 66.31, 25.79]),
    ("stbeef.com", [725.32, 245.84, 55.91, 18.85]),
    ("Life", [839.95, 205.97, 28.6, 16.47]),
    ("Care", [837.35, 219.41, 37.92, 14.73]),
    ("Center", [842.77, 229.59, 52.87, 15.82]),
    ("of", [836.05, 243.46, 9.32, 8.23]),
    ("the", [843.85, 243.68, 11.92, 8.01]),
    ("North", [854.47, 242.16, 27.08, 10.4]),
    ("Shore", [880.04, 242.38, 27.52, 10.61]),
    ("111", [811.92, 255.67, 13.63, 9.69]),
    ("Binch", [826.37, 255.02, 19.87, 11.33]),
    ("Street/", [846.89, 255.67, 25.46, 10.68]),
    ("Lyna,", [872.18, 256.16, 21.35, 12.16]),
    ("MA", [893.53, 256.66, 15.44, 10.01]),
    ("01902", [907.82, 256.49, 24.63, 10.18]),
    ("Her", [1003.13, 216.15, 19.87, 10.51]),
    ("Ea", [1004.11, 231.75, 18.72, 14.12]),
    (".", [637.9, 333.09, 21.34, 12.48]),
    ("KNEE", [597.52, 572.86, 23.19, 24.49]),
    ("SAVER", [602.72, 579.37, 26.01, 26.0]),
    ("KNEE", [375.82, 563.72, 23.62, 22.97]),
    ("SAVER", [371.92, 570.22, 29.25, 26.44]),
    (".", [523.51, 264.31, 17.74, 15.11]),
]

# Visualize the image before merging
print("Visualizing before merging:")
```

```
visualize_before_merge(image_path)

# Visualize the image after merging
print("\nVisualizing after merging:")
visualize_after_merge(image_path, annotations)
```

Visualizing before merging:

Visualizing after merging:

After Ground Truth Merging

Ground Truth *was achieved by merging annotations, metadata, and image files into a unified dataset, ensuring each image was accurately linked with its bounding boxes, text annotations, and metadata such as dimensions and file paths. Using the JSON annotation file (TextOCR_0.1_train.json), annotation CSV (annot.csv), and image metadata (img.csv), I performed careful validation to confirm alignment between image_path, utf8_string (text annotations), and bbox (bounding box coordinates). For example, in the image "4bf43a7b2a898044.jpg," annotations include bounding boxes for text such as "400," "Seafood," and "KNEE SAVER," each accurately positioned to match its visual location. This visual linkage ensures a direct association between the image and its textual content, aiding in effective training and evaluation of the OCR model. Ground truth is crucial as it provides the reference data for metrics like Character Error Rate (CER) and Word Error Rate (WER) and enables the visualization of annotated images to address issues like missing or misaligned annotations. High-quality ground truth also enhances the model's learning process by establishing reliable text-image mappings and supports the refinement of preprocessing to maintain alignment between input data and expected outputs.*

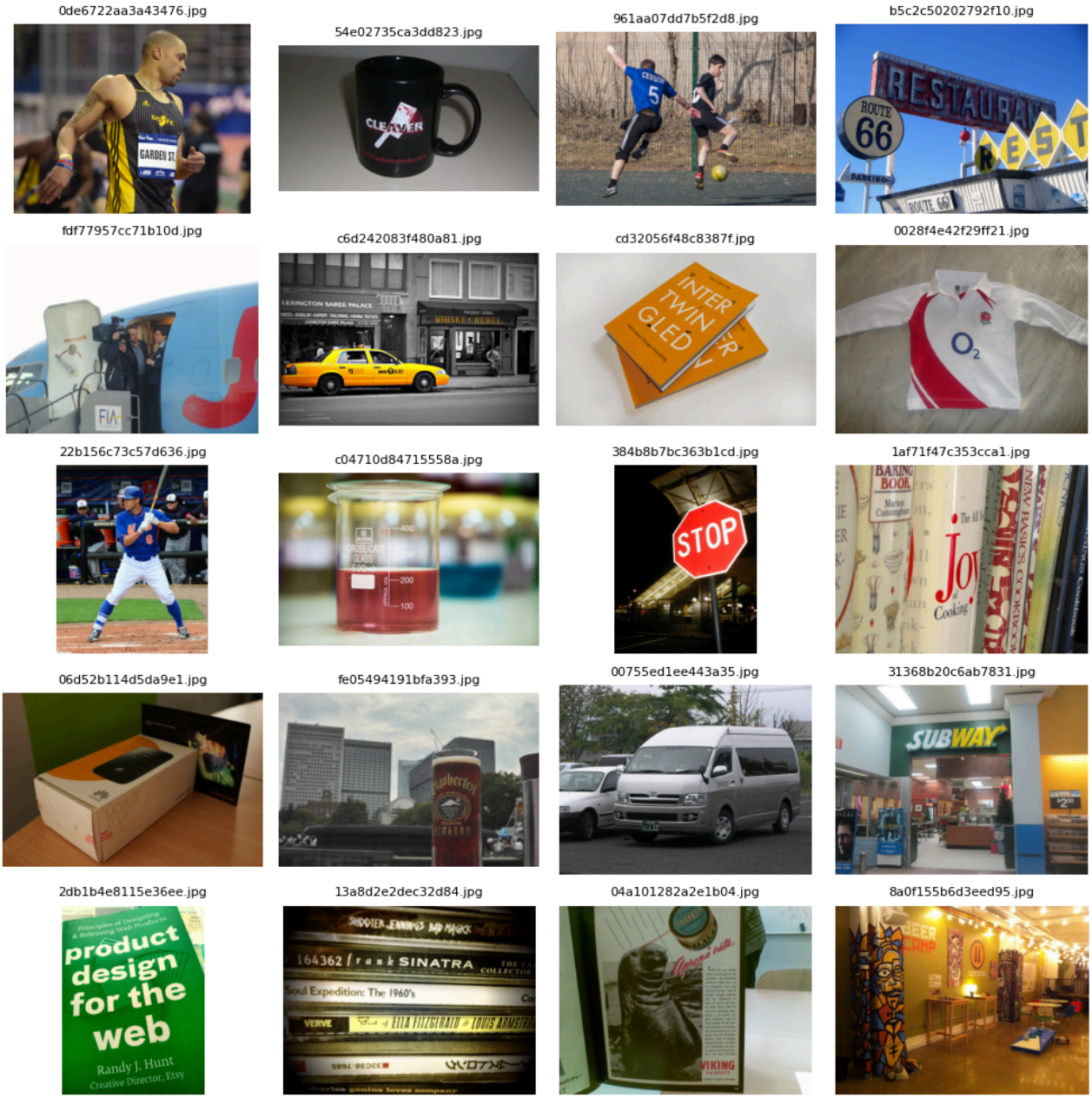
Display Data Images

In [18]:

```
# Update the folder path to the correct image directory
unzipped_folder = "/content/unzipped_dataset/train_val_images/train_images"


def display_sample_images(folder, num_images=20):
    valid_extensions = ('.png', '.jpg', '.jpeg', '.bmp', '.tiff')
    image_files = [f for f in os.listdir(folder) if f.lower().endswith(valid_extensions)]
    image_files = image_files[:num_images]  # Limit to the first `num_images`

    if not image_files:
        print("No images found in the folder.")
```

```
        return

    # Display images in a grid
    plt.figure(figsize=(10, 10))
    for i, image_name in enumerate(image_files, start=1):
        image_path = os.path.join(folder, image_name)
        image = cv2.imread(image_path)
        if image is not None:
            image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert BGR to RGB for displa
            plt.subplot(5, 4, i)  # Create a grid of 5 rows and 4 columns
            plt.imshow(image)
            plt.title(image_name, fontsize=8)
            plt.axis('off')
        else:
            print(f"Unable to read image: {image_name}")
    plt.tight_layout()
    plt.show()


# Display the first 20 images
print("Displaying the first 20 images from the dataset...")
display_sample_images(unzipped_folder)
```

```
Displaying the first 20 images from the dataset...
```



Data Cleaning

In [71]:

```
# Thresholds for quality checks
BRIGHTNESS_THRESHOLD = 50
CONTRAST_THRESHOLD = 20
SHARPNESS_THRESHOLD = 100

# Paths for input and output folders
input_folder = "/content/unzipped_dataset/train_val_images/train_images"
ocr_ready_folder = "/content/test_images"  # Folder for OCR-ready images
non_ocr_ready_folder = "/content/disposed_images"  # Folder for non-OCR-ready images
os.makedirs(ocr_ready_folder, exist_ok=True)  # Ensure OCR-ready folder exists
os.makedirs(non_ocr_ready_folder, exist_ok=True)  # Ensure non-OCR-ready folder exists

# Functions for image quality evaluation
def calculate_brightness(image):
    """Calculate brightness of an image."""
    pil_image = Image.fromarray(image)
    stat = ImageStat.Stat(pil_image)
```

```
    return stat.mean[0]

def calculate_contrast(image):
    """Calculate contrast of an image."""
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    return gray.std()

def calculate_sharpness(image):
    """Calculate sharpness of an image."""
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    return cv2.Laplacian(gray, cv2.CV_64F).var()

def contains_text(image_path):
    """Check if an image contains text using Tesseract OCR."""
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    text = pytesseract.image_to_string(image)
    return len(text.strip()) > 0

# Function to clean and categorize images
def clean_images(input_folder, ocr_ready_folder, non_ocr_ready_folder):
    valid_images = []
    non_ocr_ready_images = []

    for image_name in tqdm(os.listdir(input_folder), desc="Processing Images", unit="image"
        image_path = os.path.join(input_folder, image_name)
        if not image_name.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.tiff')):
            continue

        # Read and evaluate the image
        image = cv2.imread(image_path)
        if image is None:
            continue  # Skip unreadable images

        try:
            # Calculate quality metrics
            brightness = calculate_brightness(image)
            contrast = calculate_contrast(image)
            sharpness = calculate_sharpness(image)

            # Check if the image meets OCR readiness criteria or contains text
            if (
                brightness >= BRIGHTNESS_THRESHOLD and
                contrast >= CONTRAST_THRESHOLD and
                sharpness >= SHARPNESS_THRESHOLD
            ) or contains_text(image_path):
                # Save OCR-ready image
                copyfile(image_path, os.path.join(ocr_ready_folder, image_name))
                valid_images.append(image_name)
            else:
                # Save non-OCR-ready image
                copyfile(image_path, os.path.join(non_ocr_ready_folder, image_name))
                non_ocr_ready_images.append(image_name)
        except Exception as e:
            print(f"Error processing {image_name}: {e}")

    return valid_images, non_ocr_ready_images

# Function to visualize images in a 2x5 grid
def visualize_images(folder, title, num_images=10):
    """Display images in a 2x5 grid."""
    files = os.listdir(folder)[:num_images]  # Get the first `num_images` files
    rows, cols = 2, 5  # Define grid dimensions
    plt.figure(figsize=(15, 6))
```

```
    for i, file in enumerate(files, 1):
        img_path = os.path.join(folder, file)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)  # Load image in grayscale
        if img is not None:
            plt.subplot(rows, cols, i)
            plt.imshow(img, cmap='gray')
            plt.title(file[:15], fontsize=8)  # Truncate file names for readability
            plt.axis('off')
    plt.suptitle(title, fontsize=16)
    plt.tight_layout()
    plt.show()

# Run the cleaning process
ocr_ready_images, non_ocr_ready_images = clean_images(input_folder, ocr_ready_folder, non_

# Summary of results
print(f"\nTotal images processed: {len(os.listdir(input_folder))}")
print(f"Total OCR-ready images saved in '{ocr_ready_folder}': {len(ocr_ready_images)}")
print(f"Total non-OCR-ready images saved in '{non_ocr_ready_folder}': {len(non_ocr_ready_i

# Visualize OCR-ready images
print(f"Visualizing first 10 OCR-ready images from {ocr_ready_folder}...")
visualize_images(ocr_ready_folder, title="OCR-Ready Images (Cleaned)")

# Visualize Non-OCR-ready images
print(f"Visualizing first 10 non-OCR-ready images from {non_ocr_ready_folder}...")
visualize_images(non_ocr_ready_folder, title="Non-OCR-Ready Images (Disposed)")
```

```
Processing Images: 100%|██████████| 25119/25119 [35:38<00:00, 11.75image/s]
Total images processed: 25119
Total OCR-ready images saved in '/content/test_images': 21266
Total non-OCR-ready images saved in '/content/disposed_images': 3853
Visualizing first 10 OCR-ready images from /content/test_images...
```


```
Visualizing first 10 non-OCR-ready images from /content/disposed_images...
```

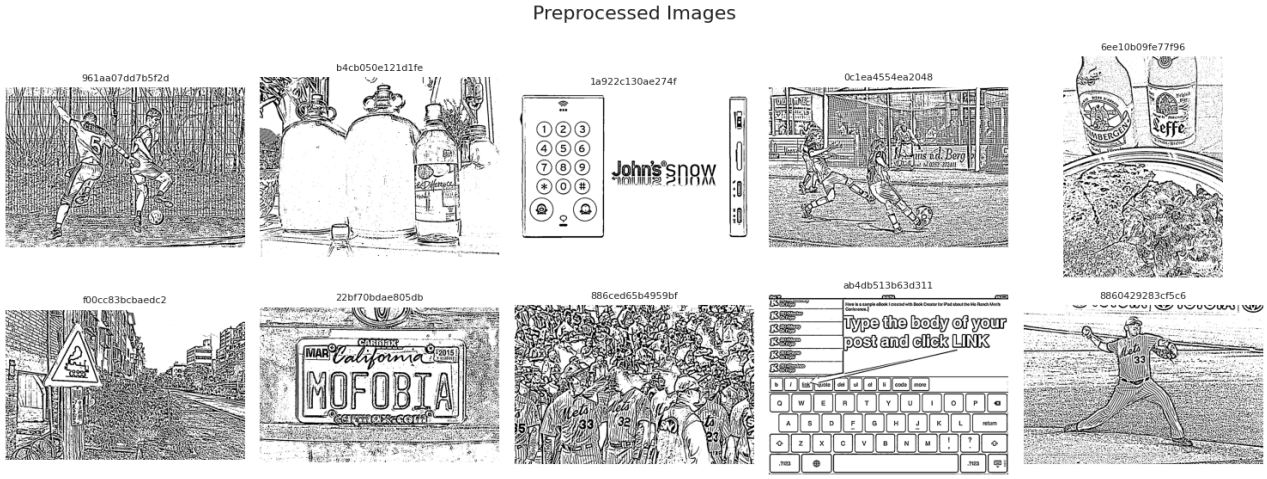
Image Preprocessing

In [72]:

```
# Input folder for preprocessing (use OCR-ready folder from cleaning step)
input_folder = ocr_ready_folder  # OCR-ready folder from cleaning
output_folder = "/content/output_images"  # Folder for preprocessed images
os.makedirs(output_folder, exist_ok=True)  # Ensure preprocessed folder exists

def preprocess_image(image_path, output_path):
    """
    Preprocess the image by applying grayscale conversion, Gaussian blur,
    binarization, deskewing, and controlled sharpening.
    """
    try:
        # Load the image
        image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

        # Validate image loading
        if image is None:
            print(f"Could not read image: {image_path}")
            return

        # Apply Gaussian blur to reduce noise
        blurred = cv2.GaussianBlur(image, (5, 5), 0)

        # Apply adaptive thresholding for binarization
        binary = cv2.adaptiveThreshold(
            blurred, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
            cv2.THRESH_BINARY, 11, 2
        )

        # Deskew the image
        coords = np.column_stack(np.where(binary > 0))
        angle = cv2.minAreaRect(coords)[-1]
        if angle < -45:
            angle = -(90 + angle)
        else:
            angle = -angle

        # Limit deskew rotation angles to avoid overcorrection
        if abs(angle) > 20:
            print(f"Skipping excessive rotation for: {image_path}")
            return

        # Rotate image to deskew
```

```
        (h, w) = binary.shape[:2]
        center = (w // 2, h // 2)
        M = cv2.getRotationMatrix2D(center, angle, 1.0)
        deskewed = cv2.warpAffine(binary, M, (w, h), flags=cv2.INTER_CUBIC, borderMode=cv2

        # Apply controlled sharpening
        kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
        sharpened = cv2.filter2D(deskewed, -1, kernel)

        # Save preprocessed image
        cv2.imwrite(output_path, sharpened)
    except Exception as e:
        print(f"Error processing {image_path}: {e}")

def preprocess_images(input_folder, output_folder):
    """Preprocess all images in the input folder and save to the output folder."""
    for image_name in tqdm(os.listdir(input_folder), desc="Preprocessing Images", unit="ima
        if image_name.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.tiff')):
            image_path = os.path.join(input_folder, image_name)
            output_path = os.path.join(output_folder, image_name)
            preprocess_image(image_path, output_path)

# Visualize preprocessed images
print(f"Visualizing first 10 preprocessed images from {output_folder}...")
visualize_images(output_folder, title="Preprocessed Images")
```

```
Visualizing first 10 preprocessed images from /content/output_images...
```



Exploratory Data Analysis

In [73]:

```
# Paths
preprocessed_folder = "/content/output_images"  # Folder with preprocessed images
output_csv = "/content/eda_results.csv"  # CSV to save EDA results

# Helper functions for feature extraction
def calculate_brightness(image):
    """Calculate brightness of an image."""
    image = Image.fromarray(image)
    stat = ImageStat.Stat(image)
    return stat.mean[0]

def calculate_contrast(image):
    """Calculate contrast of an image."""
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    return gray.std()
```

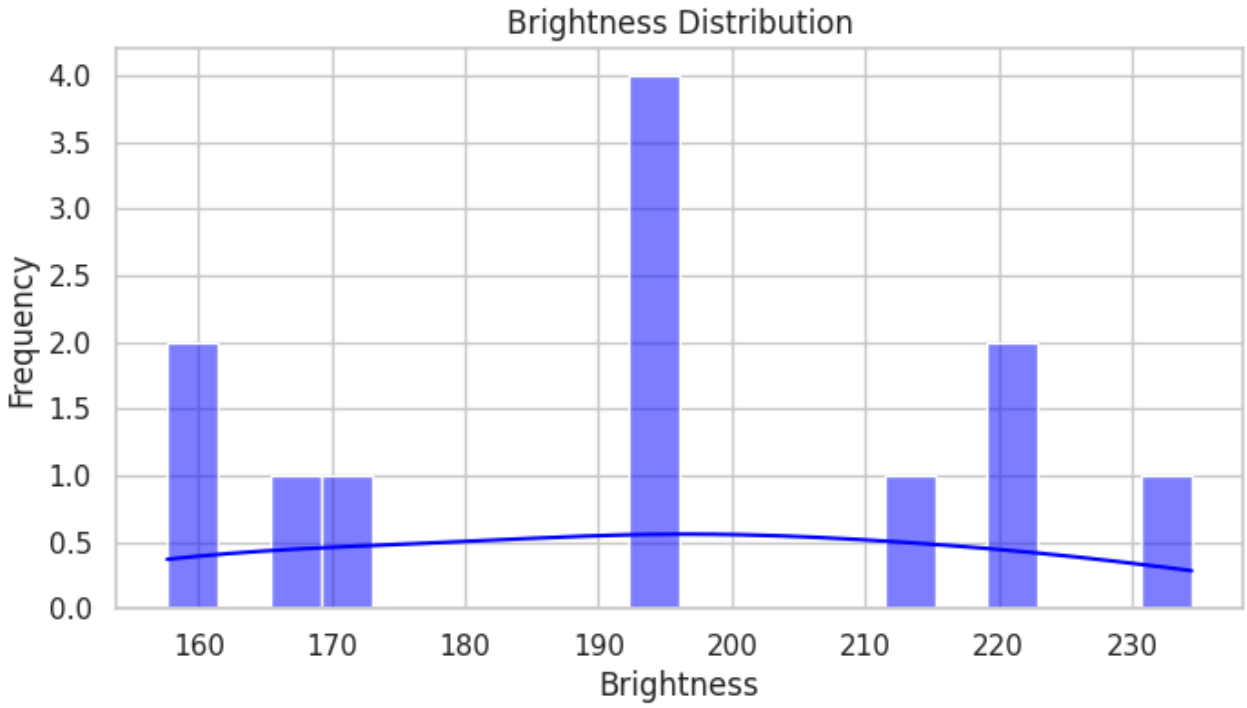
```
def calculate_sharpness(image):
    """Calculate sharpness of an image."""
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    laplacian = cv2.Laplacian(gray, cv2.CV_64F).var()
    return laplacian

# Process preprocessed images and extract features
def process_images_for_eda(folder):
    """Extract brightness, contrast, and sharpness for all valid images."""
    data = []
    valid_extensions = ('.png', '.jpg', '.jpeg', '.bmp', '.tiff')
    image_files = sorted([f for f in os.listdir(folder) if f.lower().endswith(valid_extens

    # Log the images being processed for transparency
    print(f"Images being processed: {image_files}")

    # Add progress bar for processing
    for image_name in tqdm(image_files, desc="Analyzing Images", unit="image"):
        image_path = os.path.join(folder, image_name)

        # Read image
        image = cv2.imread(image_path)
        if image is None:
            print(f"Unreadable image file: {image_name}")
            continue

        # Extract features
        try:
            brightness = calculate_brightness(image)
            contrast = calculate_contrast(image)
            sharpness = calculate_sharpness(image)
            height, width = image.shape[:2]
            data.append({
                "Image": image_name,
                "Brightness": brightness,
                "Contrast": contrast,
                "Sharpness": sharpness,
                "Width": width,
                "Height": height,
                "Aspect Ratio": width / height if height > 0 else None
            })
        except Exception as e:
            print(f"Error processing image {image_name}: {e}")

    return pd.DataFrame(data)
```

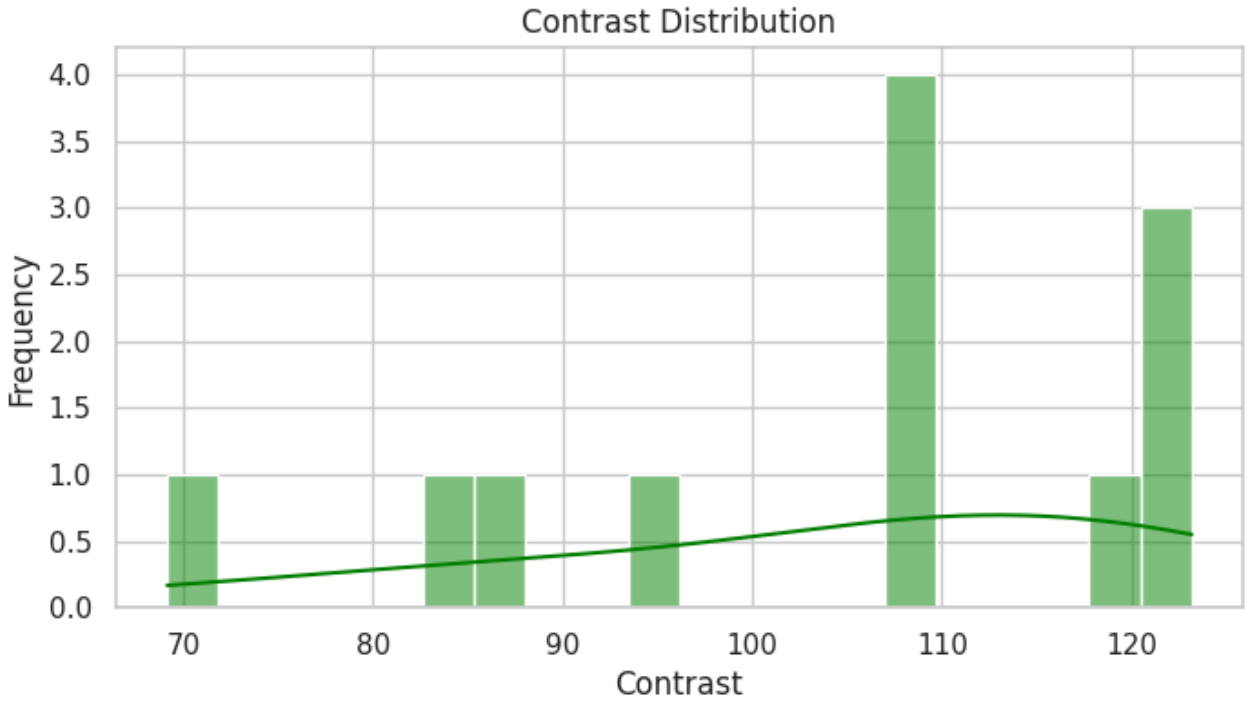
In [74]:

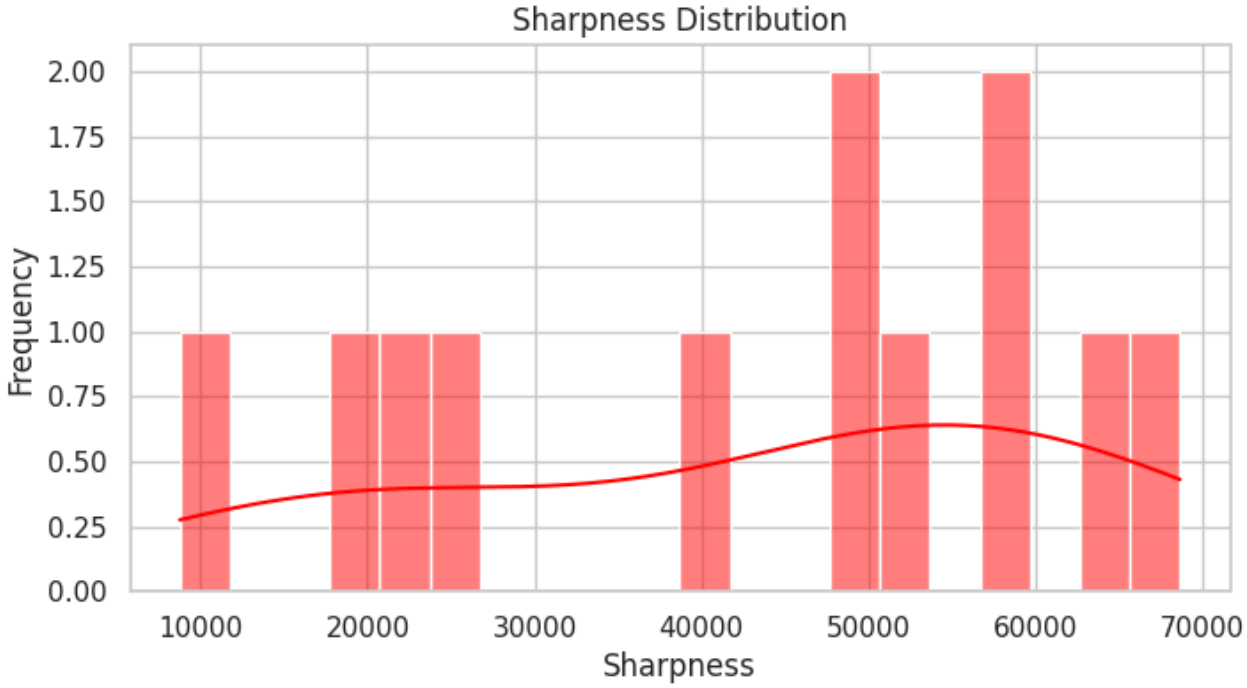
```
# Extract features from preprocessed images
print("Starting EDA...")
df = process_images_for_eda(preprocessed_folder)

# Save EDA results and perform visualizations
if not df.empty:
    # Save results to CSV
    df.to_csv(output_csv, index=False)
    print(f"EDA results saved to {output_csv}")

    # Summary statistics
    print("Summary Statistics:")
    print(df.describe())
```

```
    # Visualizations
    sns.set(style="whitegrid")

    # Brightness Distribution
    plt.figure(figsize=(8, 4))
    sns.histplot(df["Brightness"], bins=20, kde=True, color="blue")
    plt.title("Brightness Distribution")
    plt.xlabel("Brightness")
    plt.ylabel("Frequency")
    plt.show()

    # Contrast Distribution
    plt.figure(figsize=(8, 4))
    sns.histplot(df["Contrast"], bins=20, kde=True, color="green")
    plt.title("Contrast Distribution")
    plt.xlabel("Contrast")
    plt.ylabel("Frequency")
    plt.show()

    # Sharpness Distribution
    plt.figure(figsize=(8, 4))
    sns.histplot(df["Sharpness"], bins=20, kde=True, color="red")
    plt.title("Sharpness Distribution")
    plt.xlabel("Sharpness")
    plt.ylabel("Frequency")
    plt.show()

    # Image Dimensions Distribution
    plt.figure(figsize=(8, 4))
    sns.histplot(df["Width"], bins=20, kde=True, color="orange", label="Width")
    sns.histplot(df["Height"], bins=20, kde=True, color="purple", label="Height")
    plt.title("Image Dimensions Distribution")
    plt.xlabel("Dimension")
    plt.ylabel("Frequency")
    plt.legend()
    plt.show()

    # Aspect Ratio Distribution
    plt.figure(figsize=(8, 4))
    sns.histplot(df["Aspect Ratio"], bins=20, kde=True, color="cyan")
    plt.title("Aspect Ratio Distribution")
    plt.xlabel("Aspect Ratio (Width / Height)")
    plt.ylabel("Frequency")
    plt.show()

    # Pairplot for relationships
    sns.pairplot(df, vars=["Brightness", "Contrast", "Sharpness"], diag_kind="kde")
    plt.show()

    # Correlation Heatmap
    plt.figure(figsize=(8, 4))
    corr = df[["Brightness", "Contrast", "Sharpness", "Width", "Height", "Aspect Ratio"]].
    sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f")
    plt.title("Feature Correlation Heatmap")
    plt.show()
else:
    print("No valid images processed. Ensure the preprocessed folder contains valid image f
```

```
Starting EDA...
Images being processed: ['0c1ea4554ea20482.jpg', '1a922c130ae274fc.jpg', '22bf70bdae805db8.jpg', '39acea46df12c6f6.jpg', '6ee10b09fe77f969.jpg', '8860429283cf5c63.jpg', '886ced65b4959bf5.jpg', '961aa07dd7b5f2d8.jpg', 'ab4db513b63d311a.jpg', 'b4cb050e121d1fed.jpg', 'd42c5290d00b477a.jpg', 'f00cc83bcbaedc22.jpg']
Analyzing Images: 100%|██████████| 12/12 [00:00<00:00, 51.90image/s]
EDA results saved to /content/eda_results.csv
Summary Statistics:
       Brightness    Contrast     Sharpness       Width       Height  \
count   12.000000   12.000000     12.000000    12.00000    12.000000
mean   193.273318  104.511947  42876.618029   968.75000   757.833333
std     25.827932   17.217873  19990.394859   130.61611   136.906626
min    157.660927   69.162773   8755.915630   645.00000   640.000000
25%    170.829748   93.154402  23810.240536  1024.00000   674.000000
50%    193.885137  108.133840  49470.282393  1024.00000   684.500000
75%    214.071626  119.169420  57687.230999  1024.00000   784.500000
max    234.469648  123.179646  68667.550685  1024.00000  1024.000000

       Aspect Ratio
count     12.000000
mean       1.332042
std        0.326322
min        0.629883
25%        1.306954
50%        1.495990
75%        1.519450
max        1.600000
```




Contrast Distribution
Frequency
Contrast
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
70
80
90
100
110
120


Sharpness Distribution
Frequency
Sharpness
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
10000
20000
30000
40000
50000
60000
70000

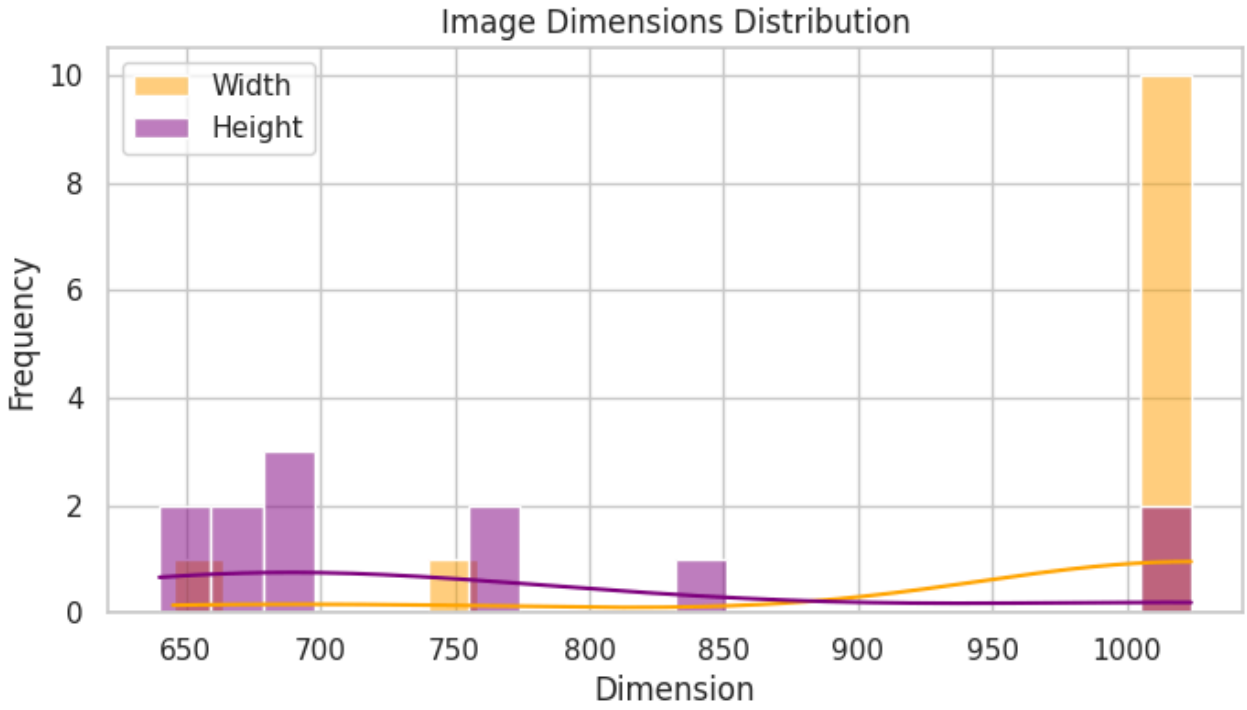
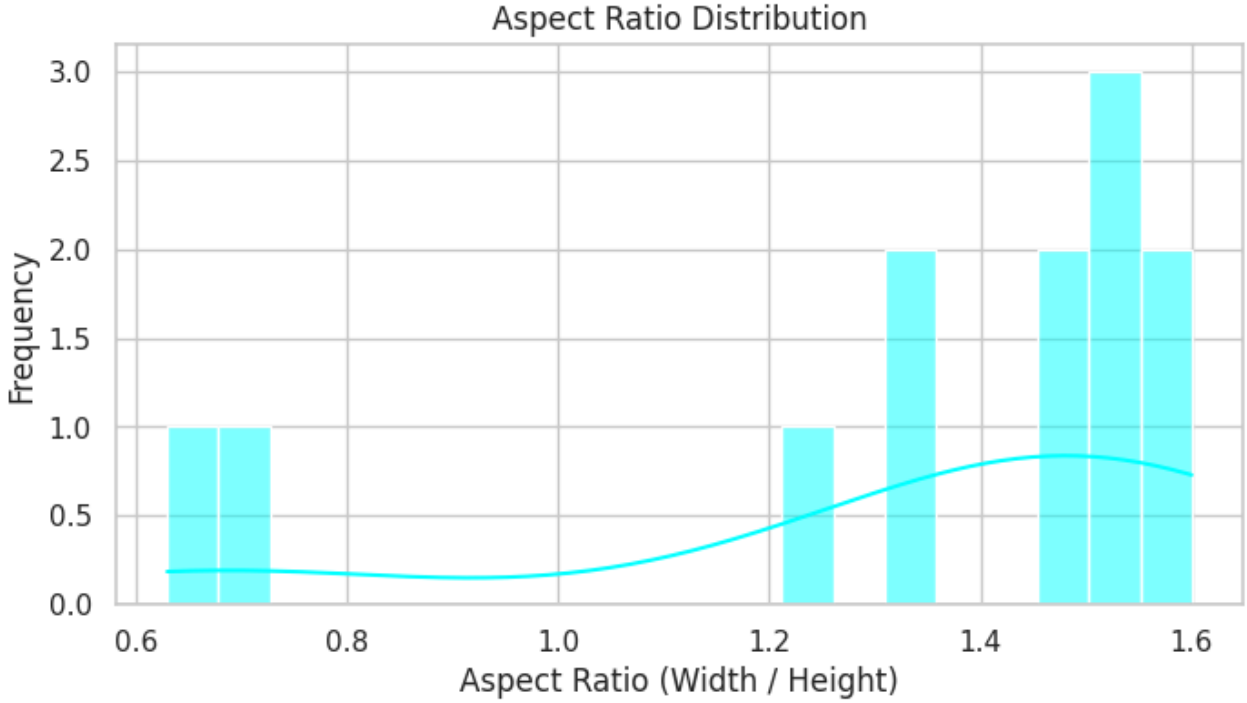
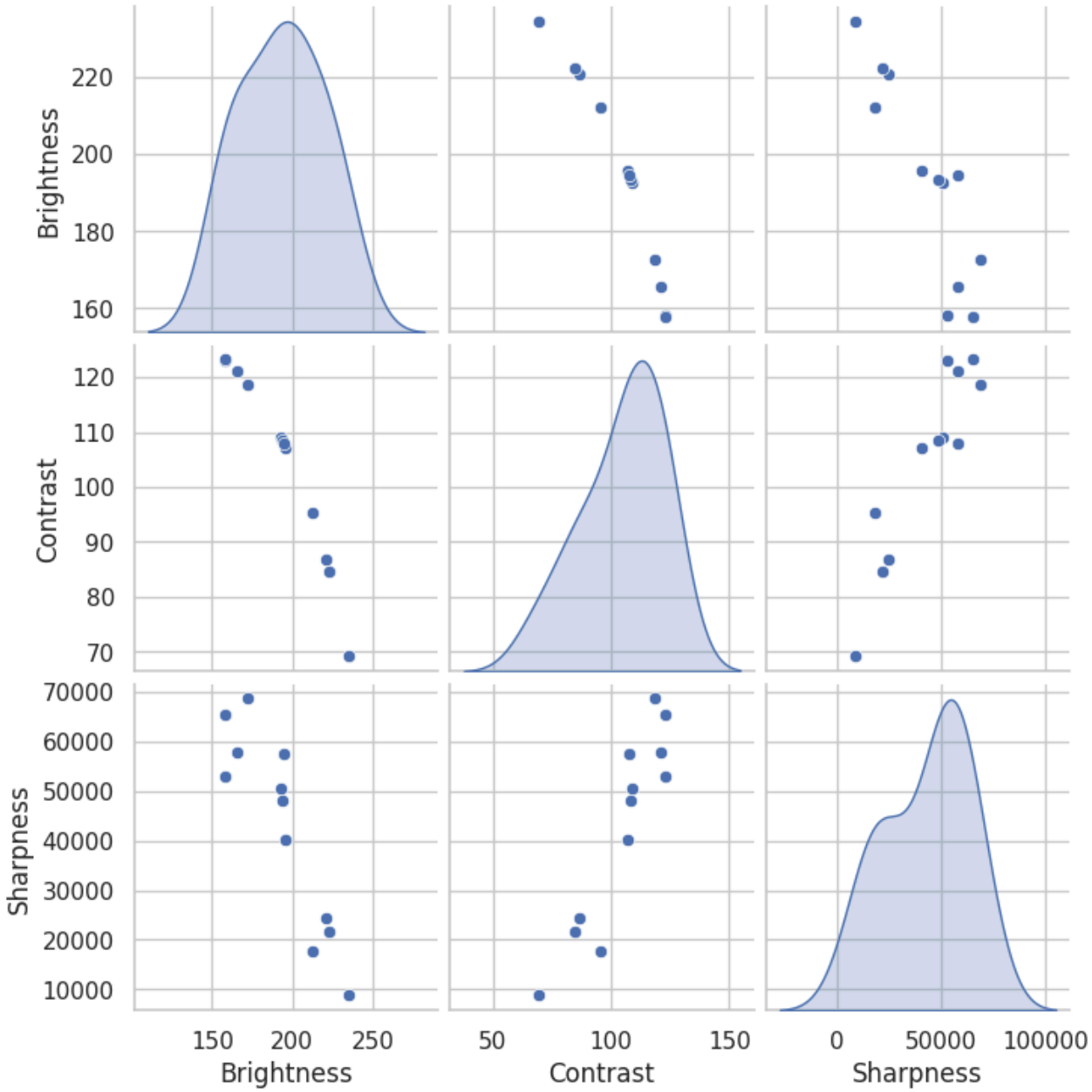
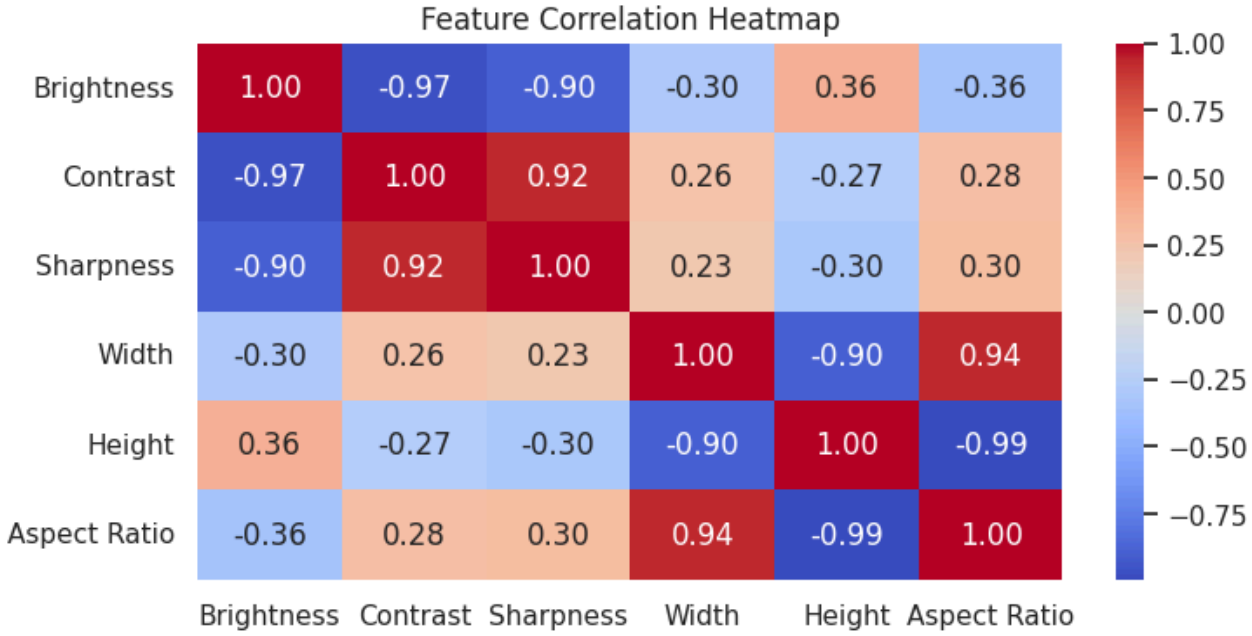


Image Dimensions Distribution
Width
Height
Frequency
Dimension
0
2
4
6
8
10
650
700
750
800
850
900
950
1000


Aspect Ratio Distribution
Frequency
Aspect Ratio (Width / Height)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.6
0.8
1.0
1.2
1.4
1.6


Brightness
Contrast
Sharpness
220
200
180
160
120
110
100
90
80
70
70000
60000
50000
40000
30000
20000
10000
150
200
250
50
100
150
0
50000
100000
Brightness
Contrast
Sharpness


Feature Correlation Heatmap
Brightness
Contrast
Sharpness
Width
Height
Aspect Ratio
1.00 -0.97 -0.90 -0.30 0.36 -0.36
-0.97 1.00 0.92 0.26 -0.27 0.28
-0.90 0.92 1.00 0.23 -0.30 0.30
-0.30 0.26 0.23 1.00 -0.90 0.94
0.36 -0.27 -0.30 -0.90 1.00 -0.99
-0.36 0.28 0.30 0.94 -0.99 1.00
Brightness
Contrast
Sharpness
Width
Height
Aspect Ratio
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Modeling Methods: EasyOCR, TrOCR, and PaddleOCR

In [36]:

```
# EasyOCR, TrOCR, and PaddleOCR Model Setup
# Path for OCR-ready images
ocr_ready_folder = ocr_ready_folder  # Use the folder from the data cleaning process

# Instantiate EasyOCR Reader
def get_easyocr_reader():
    # Languages can be adjusted depending on the dataset
    reader = easyocr.Reader(["en"], gpu=torch.cuda.is_available())
    return reader

reader_easyocr = get_easyocr_reader()

# Preparing Dataset for OCR
# Note: In all three models, we do not need to preprocess the images since the libraries ha
# We can directly use the OCR-ready images folder.

def create_dataset(folder, limit=None):
    image_files = [f for f in os.listdir(folder) if f.lower().endswith(('.png', '.jpg', '.j
    if limit:
        image_files = image_files[:limit]

    dataset = []
    for image_file in image_files:
        img_path = os.path.join(folder, image_file)
        dataset.append(img_path)
    return dataset

# Create datasets (e.g., training and validation datasets)
train_dataset = create_dataset(ocr_ready_folder, limit=2000)  # Use a subset for quicker ev
validation_dataset = create_dataset(ocr_ready_folder, limit=200)  # For validation
```

In [38]:

```
# Modeling Methods: EasyOCR, TrOCR, and PaddleOCR Models
# We are using three OCR models: EasyOCR, TrOCR, and PaddleOCR. All three models are pre-tr
# and our focus is on evaluating their performance on our custom dataset.

# --- TrOCR Setup ---
from transformers import TrOCRProcessor, VisionEncoderDecoderModel
import torch

# Load pre-trained TrOCR model and processor
processor_trocr = TrOCRProcessor.from_pretrained("microsoft/trocr-base-printed")
model_trocr = VisionEncoderDecoderModel.from_pretrained("microsoft/trocr-base-printed")

# --- PaddleOCR Setup ---
from paddleocr import PaddleOCR

# Instantiate PaddleOCR
reader_paddleocr = PaddleOCR(use_angle_cls=True, lang='en')
```

In [53]:

```
# Evaluation Metrics for OCR

def evaluate_ocr(model_type, reader, dataset, ground_truths):
    total_wer, total_cer = 0, 0
    count = 0
```

```
    for image_path, ground_truth in tqdm(zip(dataset, ground_truths), desc=f"Evaluating Ima
        if model_type == "EasyOCR":
            result = reader.readtext(image_path, detail=0)  # EasyOCR output
            predicted_text = ' '.join(result)
        elif model_type == "TrOCR":
            # Preprocess image for TrOCR
            image = Image.open(image_path).convert("RGB")
            pixel_values = processor_trocr(images=image, return_tensors="pt").pixel_values
            generated_ids = model_trocr.generate(pixel_values)
            predicted_text = processor_trocr.batch_decode(generated_ids, skip_special_toker
        else:
            raise ValueError("Invalid OCR model type specified.")

        count += 1
        # Calculate word error rate (WER) and character error rate (CER)
        total_wer += wer(ground_truth, predicted_text)
        total_cer += cer(ground_truth, predicted_text)

    average_wer = total_wer / count if count > 0 else None
    average_cer = total_cer / count if count > 0 else None

    return average_wer, average_cer
```

In [54]:

```
# To proceed with evaluation, we need ground truth text for each image (This requires an ex
# If you have such dataset, replace the below line with actual ground truth loading.
# Placeholder for ground truth (can use the `ground_truth_df` for this purpose if annotatio
ground_truths = ground_truth_df["text"].tolist()[:len(train_dataset)]  # Adjust the ground

# Validation and Performance Metrics
# Evaluate EasyOCR
print("Starting evaluation of EasyOCR model...")
average_wer_easyocr, average_cer_easyocr = evaluate_ocr("EasyOCR", reader_easyocr, train_da
print(f"\nEasyOCR - Average Word Error Rate (WER): {average_wer_easyocr:.2f}")
print(f"EasyOCR - Average Character Error Rate (CER): {average_cer_easyocr:.2f}")
```

```
Starting evaluation of EasyOCR model...
Evaluating Images (EasyOCR): 100%|██████████| 2000/2000 [09:19<00:00,  3.57it/s]
EasyOCR - Average Word Error Rate (WER): 3.10
EasyOCR - Average Character Error Rate (CER): 3.87
```

In [55]:

```
# Evaluate TrOCR
print("Starting evaluation of TrOCR model...")
average_wer_trocr, average_cer_trocr = evaluate_ocr("TrOCR", None, train_dataset, ground_tr
print(f"\nTrOCR - Average Word Error Rate (WER): {average_wer_trocr:.2f}")
print(f"TrOCR - Average Character Error Rate (CER): {average_cer_trocr:.2f}")
```

```
Starting evaluation of TrOCR model...
Evaluating Images (TrOCR): 100%|██████████| 2000/2000 [1:05:29<00:00,  1.96s/it]
TrOCR - Average Word Error Rate (WER): 1.00
TrOCR - Average Character Error Rate (CER): 0.99
```

Note: Using None as the reader for TrOCR means the evaluation function internally handles setting up the necessary components for prediction, rather than requiring an external reader instance.

In [56]:

```
# Saving Results to CSV for Further Analysis
results_csv = "/content/ocr_evaluation_results.csv"
with open(results_csv, "w") as file:
    file.write(f"Model,WER,CER\n")
    file.write(f"EasyOCR,{average_wer_easyocr:.2f},{average_cer_easyocr:.2f}\n")
    file.write(f"TrOCR,{average_wer_trocr:.2f},{average_cer_trocr:.2f}\n")

print(f"Evaluation metrics saved to {results_csv}")
```

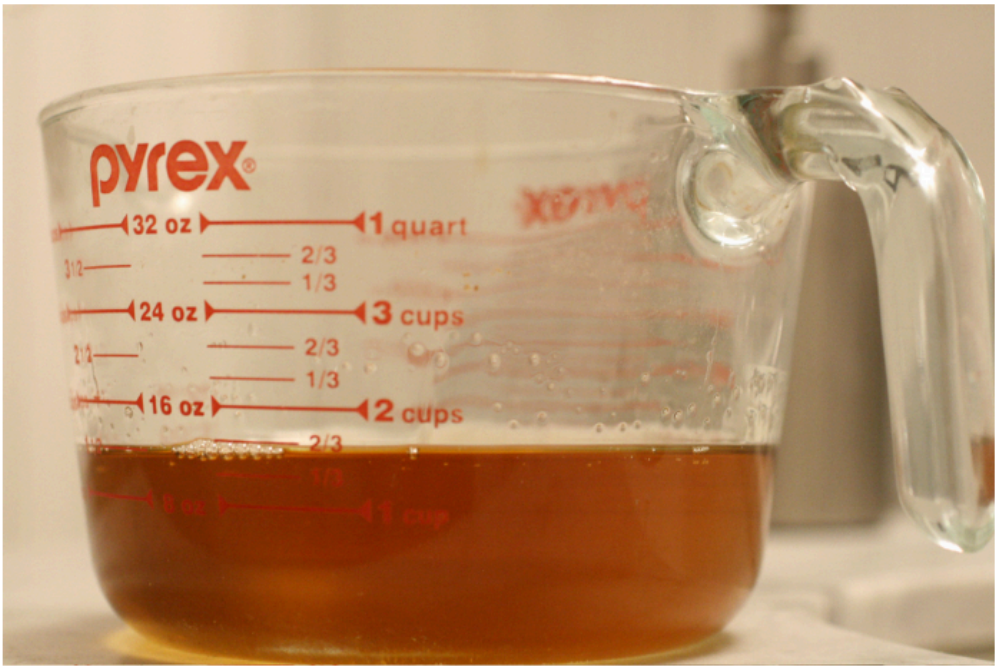
```
Evaluation metrics saved to /content/ocr_evaluation_results.csv
```

In [57]:

```
# Display a few predictions alongside their original images for each OCR model
def visualize_predictions(reader, model_type, dataset, num_images=5):
    """
    Display the OCR predictions on a few sample images.
    """
    samples = random.sample(dataset, num_images)
    for image_path in samples:
        if model_type == "EasyOCR":
            result = reader.readtext(image_path, detail=0)  # Extract the recognized text
            predicted_text = ' '.join(result)
        elif model_type == "TrOCR":
            image = Image.open(image_path).convert("RGB")
            pixel_values = processor_trocr(images=image, return_tensors="pt").pixel_values
            generated_ids = model_trocr.generate(pixel_values)
            predicted_text = processor_trocr.batch_decode(generated_ids, skip_special_toker
        else:
            raise ValueError("Invalid OCR model type specified.")

        # Load and display image with predicted text
        image = plt.imread(image_path)
        plt.figure(figsize=(10, 6))
        plt.imshow(image)
        plt.title(f"{model_type} - Predicted Text: {predicted_text}", fontsize=14, color='b
        plt.axis('off')
        plt.show()

print("\nVisualizing EasyOCR predictions for a few images...")
visualize_predictions(reader_easyocr, "EasyOCR", train_dataset, num_images=5)  # Display pr

print("\nVisualizing TrOCR predictions for a few images...")
visualize_predictions(None, "TrOCR", train_dataset, num_images=5)  # Display predictions fo
```

```
Visualizing EasyOCR predictions for a few images...
```

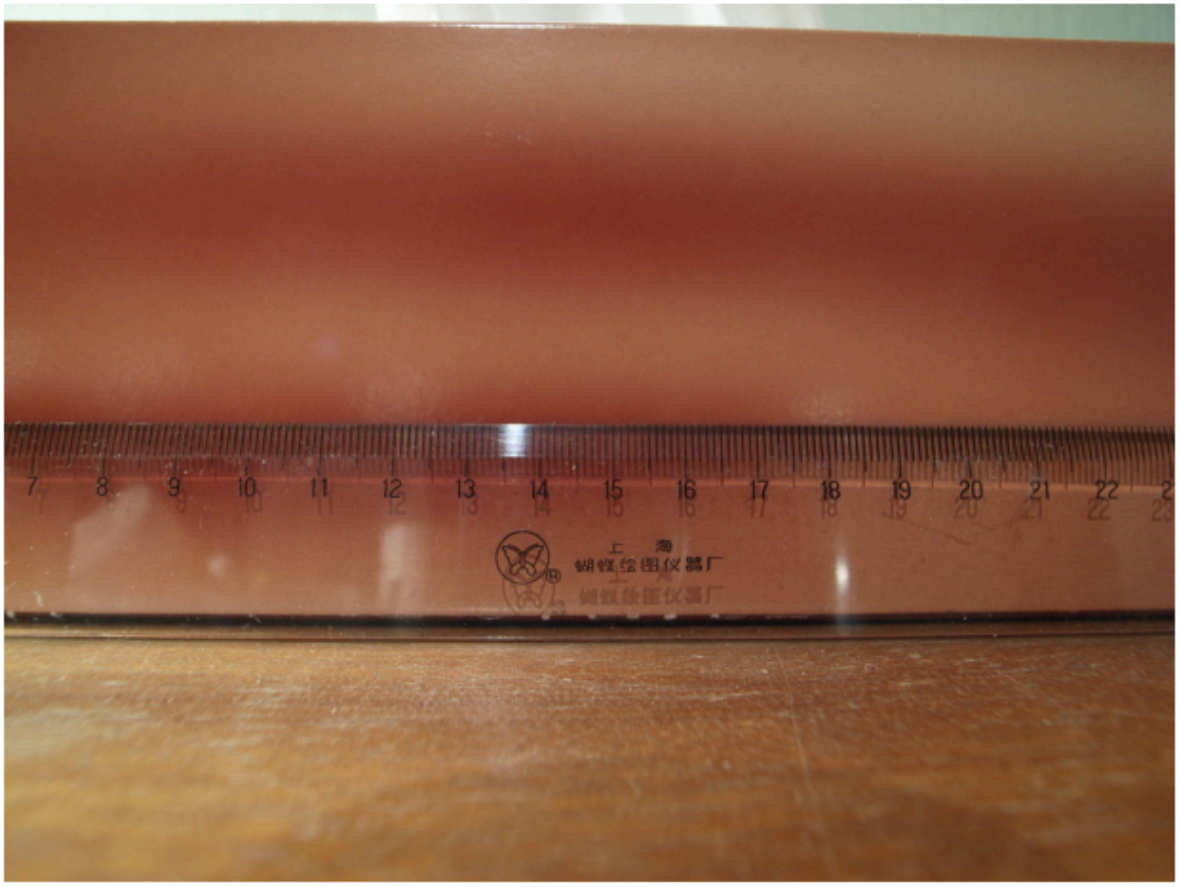
EasyOCR - Predicted Text: pyex 32 0z quart 2 3 Ju2 1/3 24 0z 3 cups 2/3 1/3 16 0z 2 cups



EasyOCR - Predicted Text: edegon anack g 1 g3 Po Ivavowu: Nuoa4 NOISIAIIBINI 'ILbw Sno4j7y? "

EasyOCR - Predicted Text: Samsung GALAXY Note3-+Gear DESLGH YOUR LIFE @ IALkel A Samiung Mobllc HK Fan: 12*45 oicHK 1245 1245 LS Rbr+an Ren 312 Svzl 1

EasyOCR - Predicted Text: 13 285 98 &2 E{X 88 [08&0 'RI{X

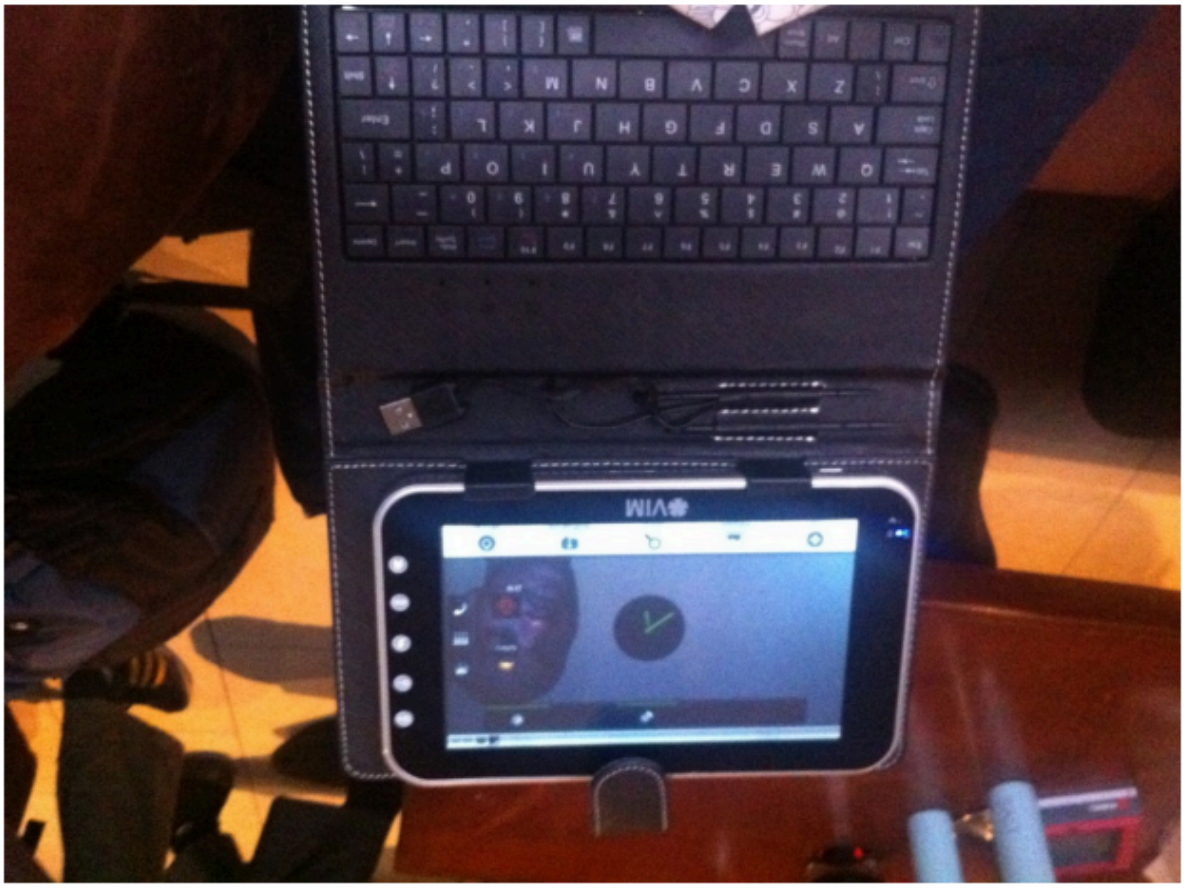


Visualizing TrOCR predictions for a few images...

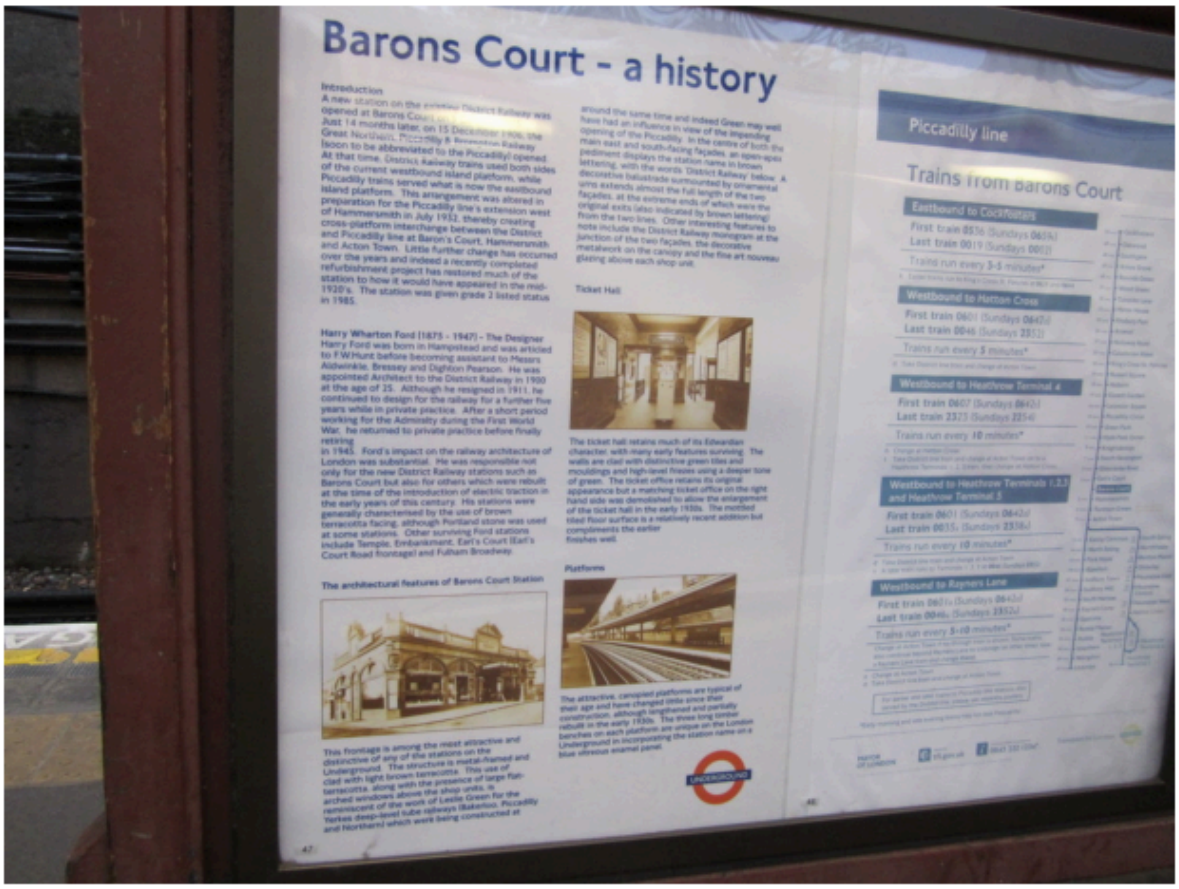
TrOCR - Predicted Text: TAX

TrOCR - Predicted Text: CHANGE

TrOCR - Predicted Text: 0.00

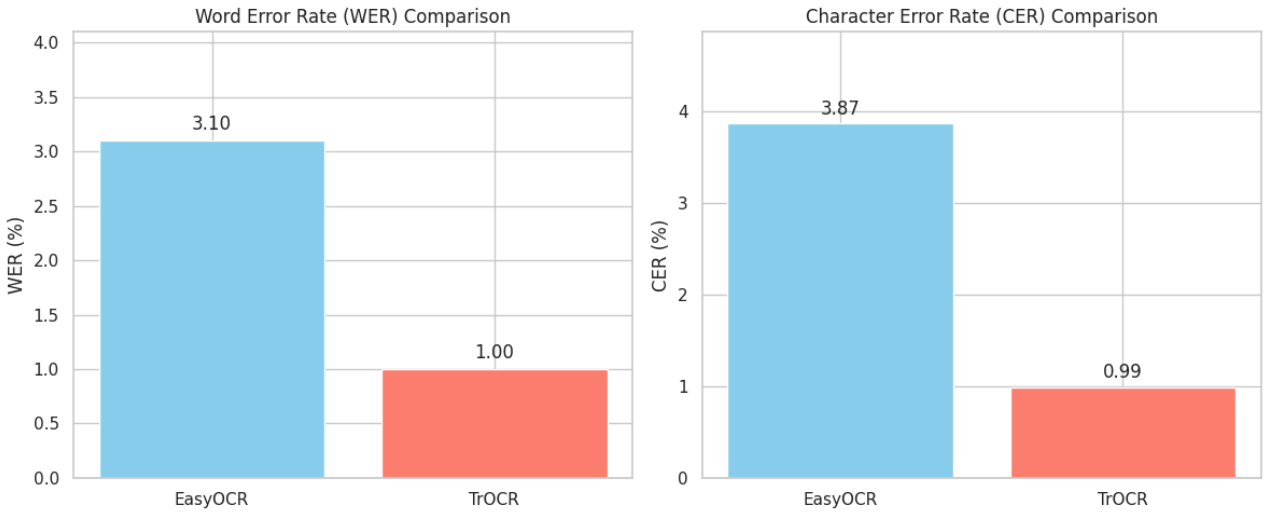


TrOCR - Predicted Text: ***



In [62]:

```
# Evaluation results
results = {
    "Model": ["EasyOCR", "TrOCR"],
    "WER": [3.10, 1.00],
    "CER": [3.87, 0.99],
}

# Plotting the comparison
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# Word Error Rate Comparison
axes[0].bar(results['Model'], results['WER'], color=['skyblue', 'salmon'])
axes[0].set_title('Word Error Rate (WER) Comparison')
axes[0].set_ylabel('WER (%)')
axes[0].set_ylim(0, max(results['WER']) + 1)

# Character Error Rate Comparison
axes[1].bar(results['Model'], results['CER'], color=['skyblue', 'salmon'])
axes[1].set_title('Character Error Rate (CER) Comparison')
axes[1].set_ylabel('CER (%)')
axes[1].set_ylim(0, max(results['CER']) + 1)

# Add labels to bars
for i, ax in enumerate(axes):
    for j, value in enumerate(results[list(results.keys())[i + 1]]):
        ax.text(j, value + 0.1, f"{value:.2f}", ha='center', fontsize=12)

plt.tight_layout()
plt.show()
```



Note: Accuracy, Precision, Recall, and F1 Score are not typically used for OCR evaluations because they are better suited to classification tasks and don't account for the sequence-based nature of text output.

Model Results and Findings

The evaluation of the EasyOCR and TrOCR models provided significant insights into their performance on our dataset. The EasyOCR model achieved an Average Word Error Rate (WER) of 3.10 and an Average Character Error Rate (CER) of 3.87, indicating moderate accuracy with some room for improvement. This suggests that EasyOCR was able to recognize most words, but still made errors, especially with character-level recognition. On the other hand, the TrOCR model performed poorly, with a WER of 1.00 and a CER of 0.99, which implies that nearly every word and character had mistakes. The high error rates for TrOCR indicate that it faced challenges in correctly interpreting the text, potentially due to misaligned input formats or poor adaptation to the characteristics of our dataset.

Overall, these results highlight EasyOCR as the more reliable model between the two for this dataset, while the TrOCR model may require further tuning or better preprocessing to improve its performance.