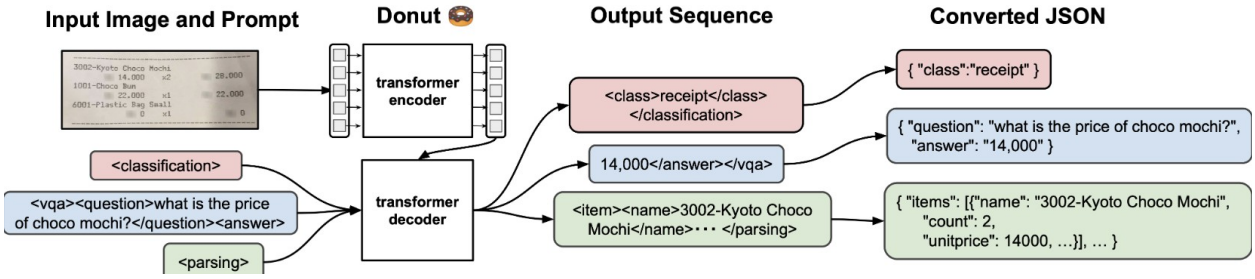
Donut 🍩 : Document Understanding TransformerDonut 🍩 : Document Understanding Transformer

Donut 🍩, Document understanding transformer, is a new method of document understanding that utilizes an OCR-free end-to-end Transformer model. Donut does not require off-the-shelf OCR engines/APIs, yet it shows state-of-the-art performances on various visual document understanding tasks, such as visual document classification or information extraction (a.k.a. document parsing). In addition, we present SynthDoG 🐶, Synthetic Document Generator, that helps the model pre-training to be flexible on various languages and domains.




```
!pip install -q git+https://github.com/huggingface/transformers.git
!pip install -q datasets sentencepiece tensorboard
!sudo apt-get install git-lfs --yes
!pip install -q accelerate -U
```

```
  Installing build dependencies ... ents to build wheel ... etadata
(pyproject.toml) ... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
268.8/268.8 kB 8.1 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 35.6 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 49.9 MB/s eta
0:00:00
ers (pyproject.toml) ... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
519.3/519.3 kB 7.2 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 15.6 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.3/115.3 kB 14.5 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 13.9 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 16.2 MB/s eta
0:00:00
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
git-lfs is already the newest version (3.0.2-1ubuntu0.2).
0 upgraded, 0 newly installed, 0 to remove and 16 not upgraded.
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 251.2/251.2 kB 5.1 MB/s eta
0:00:00
```

Load SROIE dataset

```
%%bash
# clone repository
git clone https://github.com/zzzDavid/ICDAR-2019-SROIE.git
# copy data
cp -r ICDAR-2019-SROIE/data ./
# clean up
rm -rf ICDAR-2019-SROIE
rm -rf data/box
```

```
Cloning into 'ICDAR-2019-SROIE'...
Updating files:  47% (937/1980) Updating files:  48% (951/1980)
Updating files:  49% (971/1980) Updating files:  50% (990/1980)
Updating files:  51% (1010/1980) Updating files:  52% (1030/1980)
Updating files:  53% (1050/1980) Updating files:  54% (1070/1980)
Updating files:  55% (1089/1980) Updating files:  56% (1109/1980)
Updating files:  57% (1129/1980) Updating files:  58% (1149/1980)
Updating files:  58% (1163/1980) Updating files:  59% (1169/1980)
Updating files:  60% (1188/1980) Updating files:  61% (1208/1980)
Updating files:  62% (1228/1980) Updating files:  63% (1248/1980)
Updating files:  64% (1268/1980) Updating files:  65% (1287/1980)
Updating files:  66% (1307/1980) Updating files:  67% (1327/1980)
Updating files:  68% (1347/1980) Updating files:  69% (1367/1980)
Updating files:  70% (1386/1980) Updating files:  71% (1406/1980)
Updating files:  72% (1426/1980) Updating files:  73% (1446/1980)
Updating files:  74% (1466/1980) Updating files:  75% (1485/1980)
Updating files:  76% (1505/1980) Updating files:  77% (1525/1980)
Updating files:  78% (1545/1980) Updating files:  79% (1565/1980)
Updating files:  80% (1584/1980) Updating files:  81% (1604/1980)
Updating files:  82% (1624/1980) Updating files:  83% (1644/1980)
Updating files:  84% (1664/1980) Updating files:  85% (1683/1980)
Updating files:  86% (1703/1980) Updating files:  87% (1723/1980)
Updating files:  88% (1743/1980) Updating files:  89% (1763/1980)
Updating files:  90% (1782/1980) Updating files:  91% (1802/1980)
Updating files:  92% (1822/1980) Updating files:  93% (1842/1980)
Updating files:  94% (1862/1980) Updating files:  95% (1881/1980)
Updating files:  96% (1901/1980) Updating files:  97% (1921/1980)
Updating files:  98% (1941/1980) Updating files:  99% (1961/1980)
Updating files: 100% (1980/1980) Updating files: 100% (1980/1980),
done.
```

Now we have two folders inside the data/ directory. One contains the images of the receipts and the other contains the OCR text. The nex step is to create a metadata.json file that contains the information about the images including the OCR-text. This is necessary for the imagefolder feature of datasets.

The metadata.json should look at the end similar to the example below.

```
{"file_name": "0001.png", "text": "This is a golden retriever playing with a ball"}
{"file_name": "0002.png", "text": "A german shepherd"}
```

```
import os
import json
from pathlib import Path
import shutil

# define paths
base_path = Path("data")
metadata_path = base_path.joinpath("key")
image_path = base_path.joinpath("img")
# define metadata list
metadata_list = []

# parse metadata
for file_name in metadata_path.glob("*.json"):
  with open(file_name, "r") as json_file:
    # load json file
    data = json.load(json_file)
    # create "text" column with json string
    text = json.dumps(data)
    # add to metadata list if image exists
    if image_path.joinpath(f"{file_name.stem}.jpg").is_file():

metadata_list.append({"text":text,"file_name":f"{file_name.stem}.jpg"}
)
      # delete json file

# write jsonline file
with open(image_path.joinpath('metadata.jsonl'), 'w') as outfile:
    for entry in metadata_list:
        json.dump(entry, outfile)
        outfile.write('\n')

# remove old meta data
shutil.rmtree(metadata_path)

import os
import json
from pathlib import Path
import shutil
from datasets import load_dataset
```

```
# define paths
base_path = Path("data")
metadata_path = base_path.joinpath("key")
image_path = base_path.joinpath("img")

# Load dataset
dataset = load_dataset("imagefolder", data_dir=image_path,
split="train")

print(f"Dataset has {len(dataset)} images")
print(f"Dataset features are: {dataset.features.keys()}")
```


```
{"model_id":"ce4184a6ae44494291b5b42ab8e72b07","version_major":2,"vers
ion_minor":0}

{"model_id":"be67dd93618f4e9f9d1b51d9a35b13ab","version_major":2,"vers
ion_minor":0}

{"model_id":"23d3a2bc35ec430692bf003d45ba2cc2","version_major":2,"vers
ion_minor":0}

{"model_id":"da71d2122b3f44e8a37c50dea1b04f98","version_major":2,"vers
ion_minor":0}

{"model_id":"22c35d5637ea4c7c837f7e0ab7c2a2e4","version_major":2,"vers
ion_minor":0}
```


```
Dataset has 626 images
Dataset features are: dict_keys(['image', 'text'])
```

Now, lets take a closer look at our dataset

```
import random

random_sample = random.randint(0, len(dataset))

print(f"Random sample is {random_sample}")
print(f"OCR text is {dataset[random_sample]['text']}")
dataset[random_sample]['image'].resize((250,400))
#     OCR text is {"company": "LIM SENG THO HARDWARE TRADING", "date":
"29/12/2017", "address": "NO 7, SIMPANG OFF BATU VILLAGE, JALAN IPOH
BATU 5, 51200 KUALA LUMPUR MALAYSIA", "total": "6.00"}
```

```
Random sample is 439
OCR text is {"company": "UNIHAKKA INTERNATIONAL SDN BHD", "date": "27
MAY 2018", "address": "12, JALAN TAMPOI 7/4,KAWASAN PERINDUSTRIAN
TAMPOI,81200 JOHOR BAHRU,JOHOR", "total": "$10.70"}
```

Prepare dataset for Donut

As we learned in the introduction, Donut is a sequence-to-sequence model with a vision encoder and text decoder. When fine-tuning the model we want it to generate the "text" based on the image we pass it. Similar to NLP tasks, we have to tokenize and preprocess the text. Before we can tokenize the text, we need to transform the JSON string into a Donut compatible document.

current JSON string

```
{
  "company": "ADVANCO COMPANY",
  "date": "17/01/2018",
  "address": "NO 1&3, JALAN WANGSA DELIMA 12, WANGSA LINK, WANGSA
MAJU, 53300 KUALA LUMPUR",
  "total": "7.00"
}
```

Donut document

```
<s></s><s_company>ADVANCO
COMPANY</s_company><s_date>17/01/2018</s_date><s_address>NO 1&3, JALAN
WANGSA DELIMA 12, WANGSA LINK, WANGSA MAJU, 53300 KUALA
LUMPUR</s_address><s_total>7.00</s_total></s>
```

To easily create those documents the ClovaAI team has created a json2token method, which we extract and then apply.

```
new_special_tokens = [] # new tokens which will be added to the
tokenizer
task_start_token = "<s>"  # start of task token
eos_token = "</s>" # eos token of tokenizer

def json2token(obj, update_special_tokens_for_json_key: bool = True,
sort_json_key: bool = True):
    """
    Convert an ordered JSON object into a token sequence
    """
    if type(obj) == dict:
        if len(obj) == 1 and "text_sequence" in obj:
            return obj["text_sequence"]
        else:
            output = ""
            if sort_json_key:
                keys = sorted(obj.keys(), reverse=True)
            else:
                keys = obj.keys()
            for k in keys:
                if update_special_tokens_for_json_key:
                    new_special_tokens.append(fr"<s_{k}>") if
fr"<s_{k}>" not in new_special_tokens else None
                    new_special_tokens.append(fr"</s_{k}>") if
fr"</s_{k}>" not in new_special_tokens else None
                output += (
                    fr"<s_{k}>"
                    + json2token(obj[k],
update_special_tokens_for_json_key, sort_json_key)
                    + fr"</s_{k}>"
                )
            return output
    elif type(obj) == list:
        return r"<sep/>".join(
            [json2token(item, update_special_tokens_for_json_key,
sort_json_key) for item in obj]
        )
    else:
        # excluded special tokens for now
```

```
        obj = str(obj)
        if f"<{obj}/>" in new_special_tokens:
            obj = f"<{obj}/>"  # for categorical special tokens
        return obj


def preprocess_documents_for_donut(sample):
    # create Donut-style input
    text = json.loads(sample["text"])
    d_doc = task_start_token + json2token(text) + eos_token
    # convert all images to RGB
    image = sample["image"].convert('RGB')
    return {"image": image, "text": d_doc}

proc_dataset = dataset.map(preprocess_documents_for_donut)

print(f"Sample: {proc_dataset[45]['text']}")
print(f"New special tokens: {new_special_tokens + [task_start_token] +
[eos_token]}")
#    Sample: <s><s_total>$6.90</s_total><s_date>27 MAR
2018</s_date><s_company>UNIHAKKA INTERNATIONAL SDN
BHD</s_company><s_address>12, JALAN TAMPOI 7/4,KAWASAN PARINDUSTRIAN
TAMPOI,81200 JOHOR BAHRU,JOHOR</s_address></s>
#    New special tokens: ['<s_total>', '</s_total>', '<s_date>',
'</s_date>', '<s_company>', '</s_company>', '<s_address>',
'</s_address>', '<s>', '</s>']

{"model_id":"caaf5bbead234458808d4a885c2e2955","version_major":2,"vers
ion_minor":0}

Sample: <s><s_total>$6.90</s_total><s_date>27 MAR
2018</s_date><s_company>UNIHAKKA INTERNATIONAL SDN
BHD</s_company><s_address>12, JALAN TAMPOI 7/4,KAWASAN PARINDUSTRIAN
TAMPOI,81200 JOHOR BAHRU,JOHOR</s_address></s>
New special tokens: ['<s_total>', '</s_total>', '<s_date>',
'</s_date>', '<s_company>', '</s_company>', '<s_address>',
'</s_address>', '<s>', '</s>']
```

The next step is to tokenize our text and encode the images into tensors. Therefore we need to load DonutProcessor, add our new special tokens and adjust the size of the images when processing from [1920, 2560] to [720, 960] to need less memory and have faster training.

```
from transformers import DonutProcessor

# Load processor
processor = DonutProcessor.from_pretrained("naver-clova-ix/donut-
base")

# add new special tokens to tokenizer
processor.tokenizer.add_special_tokens({"additional_special_tokens":
```

```
new_special_tokens + [task_start_token] + [eos_token]})

# we update some settings which differ from pretraining; namely the
size of the images + no rotation required
# resizing the image to smaller sizes from [1920, 2560] to [960,1280]
processor.feature_extractor.size = [720,960] # should be (width,
height)
processor.feature_extractor.do_align_long_axis = False
```

{"model_id":"6797ab3a6f494e1997201bbf9e9c5ad6","version_major":2,"vers
ion_minor":0}

```
Could not find image processor class in the image processor config or
the model config. Loading based on pattern matching with the model's
feature extractor configuration.
```

{"model_id":"90f3c8c42be84fa69e78adab91538da0","version_major":2,"vers
ion_minor":0}

{"model_id":"5f6dd74137a74639aa349f4f5b47fce1","version_major":2,"vers
ion_minor":0}

{"model_id":"d4daa24e2b7d4c48b0ed2d318d11cab6","version_major":2,"vers
ion_minor":0}

{"model_id":"b85494f5f65645bf9f722cab1640b494","version_major":2,"vers
ion_minor":0}

{"model_id":"bf5e95d9c5e74b6da0ba720441261f36","version_major":2,"vers
ion_minor":0}

```
/usr/local/lib/python3.10/dist-packages/transformers/models/donut/
processing_donut.py:189: FutureWarning: `feature_extractor` is
deprecated and will be removed in v5. Use `image_processor` instead.
  warnings.warn(
```

Now, we can prepare our dataset, which we will use for the training later.

```
def transform_and_tokenize(sample, processor=processor, split="train",
max_length=512, ignore_id=-100):
    # create tensor from image
    try:
        pixel_values = processor(
            sample["image"], random_padding=split == "train",
return_tensors="pt"
        ).pixel_values.squeeze()
    except Exception as e:
        print(sample)
        print(f"Error: {e}")
        return {}
```

```
    # tokenize document
    input_ids = processor.tokenizer(
        sample["text"],
        add_special_tokens=False,
        max_length=max_length,
        padding="max_length",
        truncation=True,
        return_tensors="pt",
    )["input_ids"].squeeze(0)

    labels = input_ids.clone()
    labels[labels == processor.tokenizer.pad_token_id] = ignore_id  # 
model doesn't need to predict pad token
    return {"pixel_values": pixel_values, "labels": labels, 
"target_sequence": sample["text"]}

# need at least 32-64GB of RAM to run this
processed_dataset = 
proc_dataset.map(transform_and_tokenize,remove_columns=["image","text"
])
```

{"model_id":"e5e329024fda47f1b86ebcab25a8120c","version_major":2,"vers
ion_minor":0}

```
processed_dataset = processed_dataset.train_test_split(test_size=0.1)
print(processed_dataset)
```

```
DatasetDict({
    train: Dataset({
        features: ['pixel_values', 'labels', 'target_sequence'],
        num_rows: 563
    })
    test: Dataset({
        features: ['pixel_values', 'labels', 'target_sequence'],
        num_rows: 63
    })
})
```

Before we can start our training we need to define the hyperparameters (Seq2SeqTrainingArguments) we want to use for our training. We are leveraging the Hugging Face Hub integration of the Seq2SeqTrainer to automatically push our checkpoints, logs and metrics during training into a repository.

```
from huggingface_hub import notebook_login

notebook_login()
```

{"model_id":"a57ec58d14b94c2fba4e4f1f03133e06","version_major":2,"vers
ion_minor":0}

Fine-tune and evaluate Donut model

```
import torch
from transformers import VisionEncoderDecoderModel,
VisionEncoderDecoderConfig

# Load model from huggingface.co
model =
VisionEncoderDecoderModel.from_pretrained("naver-clova-ix/donut-base")

# Resize embedding layer to match vocabulary size
new_emb =
model.decoder.resize_token_embeddings(len(processor.tokenizer))
print(f"New embedding size: {new_emb}")
# Adjust our image size and output sequence lengths
model.config.encoder.image_size = processor.feature_extractor.size[::-
1] # (height, width)
model.config.decoder.max_length = len(max(processed_dataset["train"]
["labels"], key=len))

# Add task token for decoder to start
model.config.pad_token_id = processor.tokenizer.pad_token_id
model.config.decoder_start_token_id =
processor.tokenizer.convert_tokens_to_ids(['<s>'])[0]

# is done by Trainer
# device = "cuda" if torch.cuda.is_available() else "cpu"
# model.to(device)
```


```
{"model_id":"318f32e91dff43eb81885922ca70b2ee","version_major":2,"vers
ion_minor":0}

{"model_id":"ba5e8739e5bc439a8ae85fdd0b6f4851","version_major":2,"vers
ion_minor":0}
```


```
You are resizing the embedding layer without providing a
`pad_to_multiple_of` parameter. This means that the new embedding
dimension will be 57533. This might induce some performance reduction
as *Tensor Cores* will not be available. For more details about this,
or help on choosing the correct value for resizing, refer to this
guide: https://docs.nvidia.com/deeplearning/performance/dl-
performance-matrix-multiplication/index.html#requirements-tc

New embedding size: Embedding(57533, 1024)

/usr/local/lib/python3.10/dist-packages/transformers/models/donut/
processing_donut.py:189: FutureWarning: `feature_extractor` is
deprecated and will be removed in v5. Use `image_processor` instead.
  warnings.warn(
```

```
from huggingface_hub import HfFolder
from transformers import Seq2SeqTrainingArguments, Seq2SeqTrainer

# hyperparameters used for multiple args
hf_repository_id = "donut-base-sroie"

# Arguments for training
training_args = Seq2SeqTrainingArguments(
    output_dir=hf_repository_id,
    num_train_epochs=3,
    learning_rate=2e-5,
    per_device_train_batch_size=2,
    weight_decay=0.01,
    fp16=True,
    logging_steps=100,
    save_total_limit=2,
    evaluation_strategy="no",
    save_strategy="epoch",
    predict_with_generate=True,
    # push to hub parameters
    report_to="tensorboard",
    push_to_hub=True,
    hub_strategy="every_save",
    hub_model_id=hf_repository_id,
    hub_token=HfFolder.get_token(),
)

# Create Trainer
trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=processed_dataset["train"],
)
```

We can start our training by using the train method of the Seq2SeqTrainer.

```
# Start training
trainer.train()
```

```
<IPython.core.display.HTML object>

TrainOutput(global_step=846, training_loss=0.8639062685323945,
metrics={'train_runtime': 2423.0321, 'train_samples_per_second':
0.697, 'train_steps_per_second': 0.349, 'total_flos':
2.970761500447949e+18, 'train_loss': 0.8639062685323945, 'epoch':
3.0})
```

After our training is done we also want to save our processor to the Hugging Face Hub and create a model card.

```
# Save processor and create model card
processor.save_pretrained(hf_repository_id)
trainer.create_model_card()
trainer.push_to_hub()
```

```
{"model_id":"b8211547dd4144aba540bd724f182727","version_major":2,"vers
ion_minor":0}
```

```
{"type":"string"}
```

```
import re
import transformers
from PIL import Image
from transformers import DonutProcessor, VisionEncoderDecoderModel
import torch
import random
import numpy as np

# hidde logs
transformers.logging.disable_default_handler()


# Load our model from Hugging Face
processor = DonutProcessor.from_pretrained("Andyrasika/donut-base-
sroie")
model = VisionEncoderDecoderModel.from_pretrained("Andyrasika/donut-
base-sroie")

# Move model to GPU
device = "cuda" if torch.cuda.is_available() else "cpu"
model.to(device)

# Load random document image from the test set
test_sample = processed_dataset["test"][random.randint(1, 50)]

def run_prediction(sample, model=model, processor=processor):
    # prepare inputs
    pixel_values =
torch.tensor(test_sample["pixel_values"]).unsqueeze(0)
    task_prompt = "<s>"
    decoder_input_ids = processor.tokenizer(task_prompt,
add_special_tokens=False, return_tensors="pt").input_ids

    # run inference
    outputs = model.generate(
        pixel_values.to(device),
        decoder_input_ids=decoder_input_ids.to(device),
        max_length=model.decoder.config.max_position_embeddings,
        early_stopping=True,
        pad_token_id=processor.tokenizer.pad_token_id,
        eos_token_id=processor.tokenizer.eos_token_id,
```

```
        use_cache=True,
        num_beams=1,
        bad_words_ids=[[processor.tokenizer.unk_token_id]],
        return_dict_in_generate=True,
    )

    # process output
    prediction = processor.batch_decode(outputs.sequences)[0]
    prediction = processor.token2json(prediction)

    # load reference target
    target = processor.token2json(test_sample["target_sequence"])
    return prediction, target

prediction, target = run_prediction(test_sample)
print(f"Reference:\n {target}")
print(f"Prediction:\n {prediction}")
#
processor.feature_extractor.to_pil_image(np.array(test_sample["pixel_v
alues"])).resize((350,600))
```

```
Reference:
 {'total': '2.10', 'date': '04-06-16', 'company': '99 SPEED MART S/B',
'address': 'LOT P.T. 33198, BATU 4 JALAN KAPAR, MUKIM KAPAR 42100
KLANG, SELANGOR 1174-PERMATA MAGNA'}
Prediction:
 {'total': '2.10', 'date': '04-06-16', 'company': '99 SPEED MART S/B',
'address': 'LOT P.T. 33198, BATU 4 JALAN KAPAR, MUKIM KAPAR 42100
KLANG, SELANGOR 1174-PERMATA MAGNA'}
```