HAAG NLP Summarization Week 10

Michael Bock

October 2024

1 Slack Questions

What did you accomplish this week?

- Tested Ollama pipeline, but got poor results
- Provided settlements to OCR team

What are you planning on working on next?

- We beleive that LLaMa had trouble understanding the insurance terminology, so Nathan advised me to fine tune. I have a script set up, but it isn't working because I can't figure out the context length of LLaMa. It says 128,000, but then I pass in that many tokens and it throws an error.

What is blocking you from progressing?

- None

2 Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Link: https://arxiv.org/abs/1706.03762

2.1 Brief Analysis

I don't believe I've ever linked Attention is all you need here, but all the work I've done is based on it. Attention is all you need introduces transformers, which are a neural network architecture that is common in natural langauge processing. A key feature of natural langauge is long range dependencies; we have words like "the" and "is" that aren't as important as others in a sentence. Most important words are far away from each other. At the time, established architectures like RNNs and CNNs relied on small perceptive fields which combined nearby features to produce a useful representation of data for classification or regression. Transformers take the opposite approach - they rely on combining features from words that are far away from each other. Transformers use an attention mechanism. Attention mechanisms weight each word in a sentence. More important words get higher weights and less important words get weights of zero. Then, the model proceeds as a normal multi-layer perceptron.

This use of attention lets the model filter out unimportant words and only focus on important words, similar to how a CNN may apply filters to extract edges or basic textures. The critical difference is that attention combines features from important words that are far away from each other. This improved feature extraction led to transformers achieving state of the art results on most tasks.

Since Attention is all you need, transformers have become the dominant neural network architecture in natural language processing. The most well known use of transformers is in chat bots like ChatGPT and other conversational AI. However, we can make use of transformers the same way we'd make use of any other backbone. I plan to use transformers for classification by taking the output features from a pretrained transformer and training classifiers with them.

3 Scripts and Code Blocks

generate_pdfs.py

```python
from summarizers.get_complaints import get_complaint_only_cases, get_orders_cases
from summarizers.ocr import read_doc, extract_text_from_pdf
from tqdm import tqdm
import pandas as pd
import os
from urllib.request import urlretrieve
import pypdf

#os.mkdir('pdfs')

data = get_orders_cases("all_cases_clearinghouse.pkl")

data_entries = []
df = {
        'ID' : [],
        'Summary': [],
        'Name': [],
        'docUrls': [],
        'Documents Text': []
    }

for entry in tqdm(data):
    try:
        if not entry or not entry.case_documents or len(entry.case_documents) < 1:
                continue
        doc_types = ''
        for i, doc in enumerate(entry.case_documents):
            text = ""
            if doc.document_type == 'Order/Opinion' or doc.document_type == 'Settlement Agreement' or doc.document_type == 'Complaint':
                print(doc.file, doc.document_type, entry.id)
                doc_types += doc.file+'|'
                doc_type = doc.document_type.replace("/", "_").replace(" ", "_")
                #urlretrieve(doc.file, f'pdfs/{entry.id}_{doc_type}_{i}.pdf')
                text += f"\n======================== CASE DOCUMENT {doc_type}_{i} =========================\n{read_doc(doc)}"
        df['Name'].append(entry.name)
        df['Summary'].append(entry.summary)
        df['Documents Text'].append(text)
        df['ID'].append(entry.id)
        df['docUrls'].append(doc_types)
    except pypdf.errors.PdfStreamError:
        pass

df = pd.DataFrame.from_dict(df)
df.to_csv('./cases_multi_doc.csv')
```

model.py

```
from datasets import load_dataset, Dataset
from transformers import AutoTokenizer
from transformers import AutoModelForSequenceClassification, AutoModel
from transformers import TrainingArguments, Trainer
import numpy as np
import evaluate
import os
import pandas as pd
from torch import nn
from labels import DNO_ISSUES

class LongFormerClassification(nn.Module):
    def __init__(self, n_issues = 51):
        super().__init__()
        self.heads = []
        self.backbone = AutoModel.from_pretrained("meta-llama/Llama-3.2-1B", token = os.environ['HF_TOKEN'])
        for _ in range(n_issues):
            self.heads.append(nn.Sequential(nn.Linear(self.backbone.config.hidden_size, 1), nn.Sigmoid()))
        print(f"LLaMa max sequence length: {self.backbone.config.max_position_embeddings}")
        print(self.backbone)

    def forward(self, input_ids, attention_mask):
        print('Input Id shape: ', input_ids.shape)
        print('Attention Mask shape: ', attention_mask.shape)
        x = self.backbone(input_ids, attention_mask)
        x = x[0]
        outputs = []
        for head in self.heads:
            outputs.append(head(x))
        return tuple(outputs)

```

```python
metric = evaluate.load("accuracy")

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)

training_args = TrainingArguments(output_dir="/home/hice1/mbock9/scratch/tutorial_runs/", eval_strategy = 'epoch', save_total_limit = 5, load_best_model_at_end = True, save_strategy = "epoch", num_train_epochs = 50)

df = pd.read_csv("dno_labels.csv", sep=",").dropna()

dataset = Dataset.from_pandas(df).train_test_split(test_size=0.2)

tokenizer = AutoTokenizer.from_pretrained("meta-llama/Llama-3.2-1B", token = os.environ['HF_TOKEN'])
tokenizer.add_special_tokens({'pad_token': '[PAD]'})

max_seq_length = tokenizer.model_max_length  # or model.config.max_position_embeddings
print(f"Model's maximum sequence length: {max_seq_length}")

def tokenize_function(examples):
    examples['Text'] = [e.replace('\n', '') for e in examples["Text"]]
    return tokenizer(examples['Text'], padding="max_length", max_length=max_seq_length, truncation=True, return_tensors='pt')

tokenized_dataset = dataset.map(tokenize_function, batched=True)
tokenized_dataset = tokenized_dataset.remove_columns(["Text"])
tokenized_dataset = tokenized_dataset.remove_columns(["Name"])
#tokenized_dataset = tokenized_dataset.rename_column("label", "labels")
tokenized_dataset.set_format("torch")

#small_train_dataset = tokenized_dataset["train"].shuffle(seed=42).select(range(1000))
#small_eval_dataset = tokenized_dataset["test"].shuffle(seed=42).select(range(1000))
#trainer = Trainer(model = model, args = training_args, train_dataset=small_train_dataset, eval_dataset=small_eval_dataset, compute_metrics = compute_metrics)
#trainer.train()
from torch.utils.data import DataLoader

train_dataloader = DataLoader(tokenized_dataset["train"], shuffle=True, batch_size=2)
eval_dataloader = DataLoader(tokenized_dataset["test"], batch_size=2)
from torch.optim import AdamW

model = LongFormerClassification()

optimizer = AdamW(model.parameters(), lr=5e-5)

from transformers import get_scheduler

num_epochs = 3
num_training_steps = num_epochs * len(train_dataloader)
lr_scheduler = get_scheduler(
    name="linear", optimizer=optimizer, num_warmup_steps=0, num_training_steps=
    num_training_steps
)
import torch

device = torch.device("cpu")#torch.device("cuda") if torch.cuda.is_available() else
    torch.device("cpu")
model.to(device)
from tqdm.auto import tqdm

progress_bar = tqdm(range(num_training_steps))
criterion = nn.BCELoss()

for epoch in range(num_epochs):
    model.train()
    for batch in train_dataloader:
        batch = {k: v for k, v in batch.items()}
        inputs = {"input_ids" : batch["input_ids"].to(device), "attention_mask":
    batch["attention_mask"].to(device)}
        outputs = model(**inputs)
        labels = {k: batch[k].to(device) for k in DNO_ISSUES}
        total_loss = sum([criterion(output, labels[k]) for output, k in zip(outputs,
     DNO_ISSUES)])
        #loss = outputs.loss
        total_loss.backward()

        optimizer.step()
        lr_scheduler.step()
        optimizer.zero_grad()
        progress_bar.update(1)

    """import evaluate

    metric = evaluate.load("accuracy")
    model.eval()
    for batch in eval_dataloader:
        batch = {k: v.to(device) for k, v in batch.items()}
        with torch.no_grad():
            outputs = model(**batch)

        logits = outputs.logits
        predictions = torch.argmax(logits, dim=-1)
        metric.add_batch(predictions=predictions, references=batch["labels"])

    print(metric.compute())"""
```

4 Documentation

The first script is how I got the dataset that I sent to the OCR team. In order to run it you only need the `all_cases_clearinghouse.pkl` pickle file, which holds a scrape of the clearinghouse api. The script takes no arguments. The output is the same as all the csvs I've been sending this semester, except it adds in a docUrl field, which has the url of every PDF in the case separated by a pipe character.

5 Scription Validation(Optional)

N/A, the script doesn't work yet.

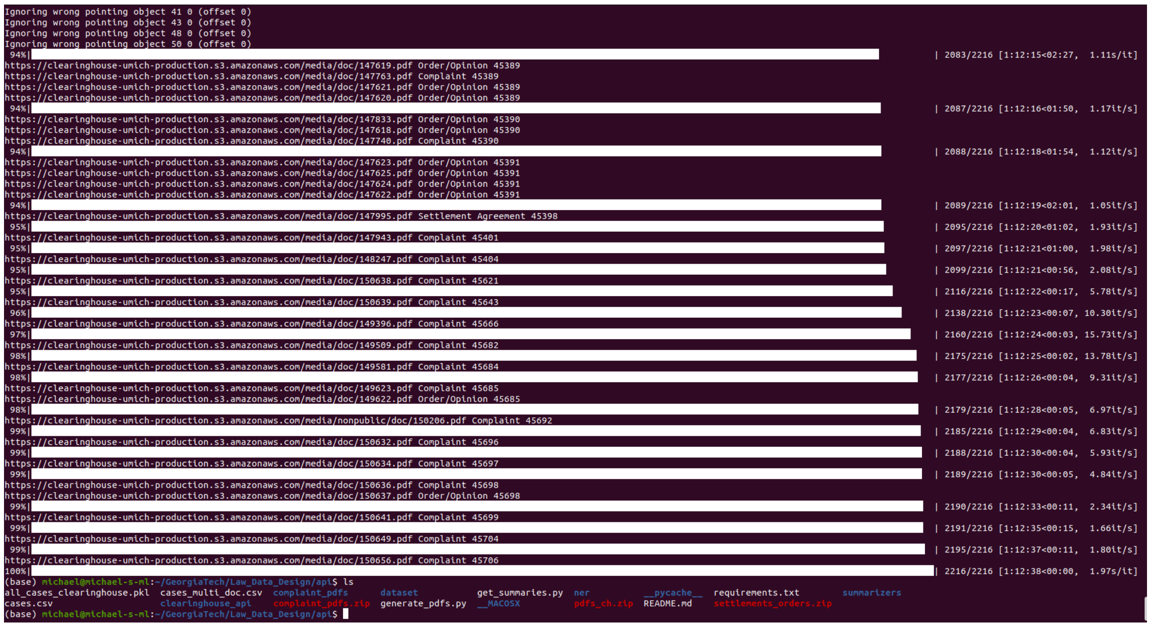
6 Results Visualization

Below is the error I am currently getting. I am running on PACE with LLaMa-1B. Huggingface and Sam Pang(another student who has fine tuned LLaMa) says that LLaMa has a context window of 128,000, but I get this error if I run more than 2048 tokens. I think this is not a coincedence, the dimensionality of the embedding is 2048 so maybe I have the wrong input shape. Huggingface documentation doesn't include an input shape so this will include trial and error to find what is wrong. Sam provided me an example on Colab of him fine tuning LLaMa, so I am going to try to run his script on my PACE account. I have the same tokenization and model as him so I think it may have something to do with my dataset that breaks huggingface.

```
---------------------------------------
Begin Slurm Prolog: Oct-25-2024 13:26:53
Job ID:    885152
User ID:   mbock9
Account:   coc
Job name:  HGX_H100_Example
Partition: coe-gpu
---------------------------------------
Lmod has detected the following error: The following module(s) are unknown:
"python/3.6"

Please check the spelling or version number. Also try "module spider ..."
It is also possible your cache file is out-of-date; it may help to try:
  $ module --ignore_cache load "python/3.6"

Also make sure that all modulefiles written in TCL start with the string
#%Module



/var/lib/slurm/slurmd/job885152/slurm_script: line 11: pip: command not found
/usr/bin/python: No module named pip
Model's maximum sequence length: 131072
Map: 100%|| 660/660 [00:35<00:00, 18.79 examples/s]
Map: 100%|| 166/166 [00:08<00:00, 20.14 examples/s]
LLaMa max sequence length: 131072
LlamaModel(
  (embed_tokens): Embedding(128256, 2048)
  (layers): ModuleList(
    (0-15): 16 x LlamaDecoderLayer(
      (self_attn): LlamaSdpaAttention(
        (q_proj): Linear(in_features=2048, out_features=2048, bias=False)
        (k_proj): Linear(in_features=2048, out_features=512, bias=False)
        (v_proj): Linear(in_features=2048, out_features=512, bias=False)
        (o_proj): Linear(in_features=2048, out_features=2048, bias=False)
        (rotary_emb): LlamaRotaryEmbedding()
      )
      (mlp): LlamaMLP(
        (gate_proj): Linear(in_features=2048, out_features=8192, bias=False)
        (up_proj): Linear(in_features=2048, out_features=8192, bias=False)
        (down_proj): Linear(in_features=8192, out_features=2048, bias=False)
        (act_fn): SiLU()
      )
      (input_layernorm): LlamaRMSNorm((2048,), eps=1e-05)
      (post_attention_layernorm): LlamaRMSNorm((2048,), eps=1e-05)
    )
  )
  (norm): LlamaRMSNorm((2048,), eps=1e-05)
  (rotary_emb): LlamaRotaryEmbedding()
)
  0%|          | 0/990 [00:00<?, ?it/s]Input Id shape:  torch.Size([2, 131072])
Attention Mask shape:  torch.Size([2, 131072])
Traceback (most recent call last):
  File "/home/hice1/mbock9/law-data-design-vip/ner/llamas/model.py", line 96, in <module>
    outputs = model(**inputs)
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/nn/modules/module.py", line 1511, in _wrapped_call_impl
    return self._call_impl(*args, **kwargs)
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/nn/modules/module.py", line 1520, in _call_impl
    return forward_call(*args, **kwargs)
  File "/home/hice1/mbock9/law-data-design-vip/ner/llamas/model.py", line 25, in forward
    x = self.backbone(input_ids, attention_mask)
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/nn/modules/module.py", line 1511, in _wrapped_call_impl
    return self._call_impl(*args, **kwargs)
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/nn/modules/module.py", line 1520, in _call_impl
    return forward_call(*args, **kwargs)
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/transformers/models/llama/modeling_llama.py", line 950, in forward
    inputs_embeds = self.embed_tokens(input_ids)
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/nn/modules/module.py", line 1511, in _wrapped_call_impl
    return self._call_impl(*args, **kwargs)
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/nn/modules/module.py", line 1520, in _call_impl
    return forward_call(*args, **kwargs)
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/nn/modules/sparse.py", line 163, in forward
    return F.embedding(
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/nn/functional.py", line 2237, in embedding
    return torch.embedding(weight, input, padding_idx, scale_grad_by_freq, sparse)
IndexError: index out of range in self
  0%|          | 0/990 [00:00<?, ?it/s]
---------------------------------------
Begin Slurm Epilog: Oct-25-2024 13:27:56
Job ID:        885152
Array Job ID:  _4294967294
User ID:       mbock9
Account:       coc
Job name:      HGX_H100_Example
Resources:     cpu=1,gres/gpu:h100=1,mem=224G,node=1
Rsrc Used:     cput=00:01:04,vmem=0,walltime=00:01:04,mem=11571236K,energy_used=0
Partition:     coe-gpu
```

```
Nodes:          atl1-1-03-012-8-0
---------------------------------------
```

7 Proof of work



```
Ignoring wrong pointing object 41 0 (offset 0)
Ignoring wrong pointing object 43 0 (offset 0)
Ignoring wrong pointing object 48 0 (offset 0)
Ignoring wrong pointing object 50 0 (offset 0)
 94%|████████████████████████████████████          | 2083/2216 [1:12:15<02:27,  1.11s/it]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/147619.pdf Order/Opinion 45389
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/147763.pdf Complaint 45389
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/147621.pdf Order/Opinion 45389
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/147620.pdf Order/Opinion 45389
 94%|████████████████████████████████████          | 2087/2216 [1:12:16<01:50,  1.17it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/147833.pdf Order/Opinion 45390
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/147618.pdf Order/Opinion 45390
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/147740.pdf Complaint 45390
 94%|████████████████████████████████████          | 2088/2216 [1:12:18<01:54,  1.12it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/147623.pdf Order/Opinion 45391
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/147625.pdf Order/Opinion 45391
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/147624.pdf Order/Opinion 45391
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/147622.pdf Order/Opinion 45391
 94%|████████████████████████████████████          | 2089/2216 [1:12:19<02:01,  1.05it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/147995.pdf Settlement Agreement 45398
 95%|████████████████████████████████████          | 2095/2216 [1:12:20<01:02,  1.93it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/147943.pdf Complaint 45401
 95%|████████████████████████████████████          | 2097/2216 [1:12:21<01:00,  1.98it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/148247.pdf Complaint 45404
 95%|████████████████████████████████████          | 2099/2216 [1:12:21<00:56,  2.08it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/150638.pdf Complaint 45621
 95%|█████████████████████████████████████         | 2116/2216 [1:12:22<00:17,  5.78it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/150639.pdf Complaint 45643
 96%|█████████████████████████████████████         | 2138/2216 [1:12:23<00:07, 10.30it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/149396.pdf Complaint 45666
 97%|██████████████████████████████████████        | 2160/2216 [1:12:24<00:03, 15.73it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/149509.pdf Complaint 45682
 98%|██████████████████████████████████████        | 2175/2216 [1:12:25<00:02, 13.78it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/149581.pdf Complaint 45684
 98%|██████████████████████████████████████        | 2177/2216 [1:12:26<00:04,  9.31it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/149623.pdf Complaint 45685
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/149622.pdf Order/Opinion 45685
 98%|██████████████████████████████████████        | 2179/2216 [1:12:28<00:05,  6.97it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/nonpublic/doc/150206.pdf Complaint 45692
 99%|███████████████████████████████████████       | 2185/2216 [1:12:29<00:04,  6.83it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/150632.pdf Complaint 45696
 99%|███████████████████████████████████████       | 2188/2216 [1:12:30<00:04,  5.93it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/150634.pdf Complaint 45697
 99%|███████████████████████████████████████       | 2189/2216 [1:12:30<00:05,  4.84it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/150636.pdf Complaint 45698
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/150637.pdf Order/Opinion 45698
 99%|███████████████████████████████████████       | 2190/2216 [1:12:33<00:11,  2.34it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/150641.pdf Complaint 45699
 99%|███████████████████████████████████████       | 2191/2216 [1:12:35<00:15,  1.66it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/150649.pdf Complaint 45704
 99%|███████████████████████████████████████       | 2195/2216 [1:12:37<00:11,  1.80it/s]
https://clearinghouse-umich-production.s3.amazonaws.com/media/doc/150656.pdf Complaint 45706
100%|██████████████████████████████████████████████| 2216/2216 [1:12:38<00:00,  1.97s/it]
(base) michael@michael-s-ml:~/GeorgiaTech/Law_Data_Design/api$ ls
all_cases_clearinghouse.pkl  cases_multi_doc.csv  complaint_pdfs      dataset          get_summaries.py  ner          __pycache__  requirements.txt       summarizers
cases.csv                    clearinghouse_api    complaint_pdfs.zip  generate_pdfs.py  __MACOSX          pdfs_ch.zip  README.md    settlements_orders.zip
(base) michael@michael-s-ml:~/GeorgiaTech/Law_Data_Design/api$ 
```

Figure 1: Retrieving URLs from clearinghouse api

8 Next Week's proposal

- We beleive that LLaMa had trouble understanding the insurance terminology, so Nathan advised me to fine tune. I have a script set up, but it isn't working because I can't figure out the context length of LLaMa. It says 128,000, but then I pass in that many tokens and it throws an error.

HAAG Research Report
NLP - Sentencias / NLP - Gen Team
Week 10

Víctor C. Fernández

October 2024

1 WEEKLY PROJECT UPDATES

What progress did you make in the last week?

- Implemented a benchmark for evaluating different models when retrieving context for the dates.
- Created graphical views comparing all models.
- Completed a basic abstract for our paper within the overleaf template.
- Created a presentation/PPT for our future call with the judge Miguel.
- Researched on creating a pipeline to orchestrate the whole process using prefect.
- Researched on creating a simple UI to load and view results using Gradio.
- Met with the NLP-Sentencias team on Monday 21st to align on our goals and distribute our tasks.
- Meeting with the NLP team on October 25th for our weekly meeting.
- Meeting with Dr. Alexander and Nathan Dahlberg on October 25th to get further insights on NLP research.

What progress are you making next?

- Researching ways to improve context retrieval for the dates.
- Implement a simple pipeline to run all processess end to end.
- Implement a simple UI to interact with the pipeline and upload and process a single file.
- Work on getting more content in for our paper.

Is there anything blocking you from making progress?

No blockers at this moment.

2 ABSTRACTS

1. **Title:** Segmentation of legal documents
   - **URL:** https://dl.acm.org/doi/10.1145/3594536.3595142 (requires accessing with GaTech credentials to view actual paper)

   - **Abstract:** The computational cost of transformer-based models has a quadratic dependence on the length of the input sequence. This makes it challenging to deploy these models in domains in which long documents are especially lengthy, such as the legal domain. To address this issue, we propose a three-stage cascading approach for long document classification. We begin by filtering out likely irrelevant information with a lightweight logistic regression model before passing the more challenging inputs to the transformer-based model. We evaluate our approach using CUAD, a legal dataset with 510 manually-annotated, long contracts. We find that the cascading approach reduces training time by up to 80% while improving baseline performance. We hypothesize that the gains in performance stem from localizing the classification task of the transformer model to particularly difficult examples.
   - **Summary:** The paper, proposes a three-stage cascading model to improve the efficiency of transformer-based models when processing lengthy legal documents. The pipeline begins by dividing the document into manageable chunks and applies a lightweight logistic regression model to filter out irrelevant segments. Only the challenging segments are passed on to a transformer model, significantly reducing computational cost. Evaluations on a legal dataset of long contracts showed that the cascading model reduced training time by up to 80% while maintaining or improving classification accuracy, demonstrating that this approach can optimize resources in NLP tasks involving long texts.

   - **Relevance:** This paper provides an efficient approach for handling lengthy legal documents. The cascading pipeline's use of lightweight classifiers for filtering less critical segments could help streamline the processing of sentencias, especially as the project focuses on optimizing NLP tools to tackle data extraction efficiently. This is particularly advantageous given the long-form nature of judicial decisions and their tendency to include substantial non-relevant content, mirroring the filtering challenges addressed in the paper.

By incorporating this cascading method, our project could reduce processing time while focusing computational resources on critical information extraction tasks, such as pinpointing case dates, ultimately supporting efforts to identify causes of judicial delays and contribute to policy recommendations.

3 SCRIPTS AND CODE BLOCKS

All scripts have been uploaded to the HAAG NLP Repo. Outputs files, processed sentencias and any other document that may contain sensitive information is located in the private NLP-Sentencias Repo.

The following code contains the logic and functions I have been working on this week.

1. Created a function to combine all results from all dates into a single file here.

```
def combine_outputs_per_file_and_model(output_folder):
    # Iterate over each original file in the output folder
    for filename in os.listdir(output_folder):
        file_path = os.path.join(output_folder, filename)
        if os.path.isdir(file_path):
            # Iterate over each model folder within the original
            ↪   file folder
            for model in os.listdir(file_path):
                model_path = os.path.join(file_path, model)
                if os.path.isdir(model_path):
                    dates_outputs = []
                    metrics_outputs = []
                    # Iterate over each output file within the
                    ↪   model folder
                    output_files = sorted([f for f in
                    ↪   os.listdir(model_path) if
                    ↪   f.endswith('.txt')])
                    for output_file in output_files:
                        if output_file.endswith('_combined.txt'):
                            continue  # Skip combined files
                        output_file_path =
                        ↪   os.path.join(model_path, output_file)
                        with open(output_file_path, 'r',
                        ↪   encoding='utf-8') as f:
                            content = f.read()
                        # Parse the content
                        sections = content.strip().split('\n\n')
                        if len(sections) >= 2:
                            # Model output is the first section
                            model_output_text =
                            ↪   sections[0].strip()
                            # Execution details is the last
                            ↪   section
                            execution_details_json =
                            ↪   sections[-1].strip()
                            # Clean up model output text
                            model_output_text =
                            ↪   re.sub(r'^```(?:json)?\s*', '',
                            ↪   model_output_text)
                            model_output_text =
                            ↪   re.sub(r'\s*```$', '',
                            ↪   model_output_text)
```

```
# Parse the model output JSON
                try:
                    model_output =
                     ↪   json.loads(model_output_text)
                except json.JSONDecodeError as e:
                    print(f"Error decoding model
                     ↪   output in file
                     ↪   {output_file_path}: {e}")
                    # Optionally, print the
                     ↪   problematic text
                    # print(f"Model output text
                     ↪   was:\n{model_output_text}\n")
                    continue
                # Extract "date" and "date event"
                try:
                    date = model_output.get("date",
                     ↪   "")
                    date_event =
                     ↪   model_output.get("date event",
                     ↪   "")
                except Exception as e:
                    print(f"Error extracting date and
                     ↪   date event in file
                     ↪   {output_file_path}: {e}")
                    date = ""
                    date_event = ""
                # Parse execution details
                try:
                    execution_details_full = json.loa⌋
                     ↪   ds(execution_details_json)
                    execution_details =
                     ↪   execution_details_full.get('e⌋
                     ↪   xecution_details',
                     ↪   execution_details_full)
                except json.JSONDecodeError as e:
                    print(f"Error decoding execution
                     ↪   details in file
                     ↪   {output_file_path}: {e}")
                    execution_details = {}
                # Add to dates_outputs
```

```
                        dates_outputs.append({
                            "date": date,
                            "date event": date_event
                        })
                        # Add to metrics_outputs
                        metrics_outputs.append(execution_deta⌋
                     ↪   ils)
                    else:
                        # Handle unexpected format
                        print(f"Unexpected format in file
                     ↪   {output_file_path}")
                        continue
                # Write dates_outputs to JSON file
                dates_output_file = os.path.join(model_path,
             ↪   f"{filename}_{model}.json")
                with open(dates_output_file, 'w',
             ↪   encoding='utf-8') as f:
                    json.dump(dates_outputs, f,
                 ↪   ensure_ascii=False, indent=2)
                # Write metrics_outputs to JSON file
                metrics_output_file = os.path.join(model_path,
             ↪   f"{filename}_{model}_metrics.json")
                with open(metrics_output_file, 'w',
             ↪   encoding='utf-8') as f:
                    json.dump(metrics_outputs, f,
                 ↪   ensure_ascii=False, indent=2)
print("Combined output files created.")
```

Code 1—Code for combining all models outputs

2. Code for validating the results obtained from the models and generate corresponding scores for benchmark here.

```python
def validate_model_outputs(models_output_folder,
↪   validation_set_folder, results_file):
    # Dictionary to store accumulated metrics per model and
    ↪   hyperparameters
    model_hyperparam_metrics = {}

    # For each document in models_output_folder
    for document_name in os.listdir(models_output_folder):
        # Log the document being processed
        log_in_color(f"Processing document: {document_name}",
        ↪   "green")
        document_path = os.path.join(models_output_folder,
        ↪   document_name)
        if not os.path.isdir(document_path):
            continue

        # Get the validation set for this document
        validation_set_file = os.path.join(validation_set_folder,
        ↪   f"{document_name}_validation.json")
        if not os.path.isfile(validation_set_file):
            print(f"Validation set for '{document_name}' not
            ↪   found.")
            continue

        with open(validation_set_file, 'r', encoding='utf-8') as
        ↪   f:
            validation_set = json.load(f)

        # For each model in this document's folder
        for model_name in os.listdir(document_path):
            # Log the model being processed
            log_in_color(f"Processing model: {model_name}",
            ↪   "blue")
            model_path = os.path.join(document_path, model_name)
            if not os.path.isdir(model_path):
                continue
            # Read the model outputs
            model_output_file = os.path.join(model_path,
            ↪   f"{document_name}_{model_name}.json")
            if not os.path.isfile(model_output_file):
                print(f"Model output file '{model_output_file}'
                ↪   not found.")
                continue

            with open(model_output_file, 'r', encoding='utf-8')
            ↪   as f:
                model_outputs = json.load(f)

            # Read the execution metrics
            metrics_file = os.path.join(model_path,
            ↪   f"{document_name}_{model_name}_metrics.json")
            if os.path.isfile(metrics_file):
                with open(metrics_file, 'r', encoding='utf-8') as
                ↪   f:
                    execution_metrics = json.load(f)
            else:
                execution_metrics = []

            # Validate the model outputs against the validation
            ↪   set
            correct = 0
            incorrect = 0
            used_model_indices = set()

            for val_date_obj in validation_set:
                val_date =
                ↪   normalize_str(val_date_obj.get('date'))
                val_event = normalize_str(val_date_obj.get('date
                ↪   event'))
                match_found = False
                # Search for val_date in model outputs
                for idx, model_output in enumerate(model_outputs):
                    # Log the model output index being processed
                    log_in_color(f"Processing model output index:
                    ↪   {idx}", "magenta")
                    if idx in used_model_indices:
                        continue  # Skip already matched outputs

                    mod_date = model_output.get('date', '')
                    mod_event = model_output.get('date event',
                    ↪   '')
                    # Check if date and date event are strings, if
                    ↪   not, cast them to strings
                    if not isinstance(mod_date, str):
                        mod_date = str(mod_date)
                    if not isinstance(mod_event, str):
                        mod_event = str(mod_event)
                    mod_date = normalize_str(mod_date)
                    mod_event = normalize_str(mod_event)

                    if val_date == mod_date:
                        used_model_indices.add(idx)
                        match_found = True

                        # Check if date events match
                        if val_event == mod_event:
                            # Correct response
                            correct += 1
                        else:
                            # Check for 'otros' in both date
                            ↪   events
                            if 'otros' in val_event.lower() and
                            ↪   'otros' in mod_event.lower():
                                correct += 1
                            else:
                                incorrect += 1
                        break
```

```
    if not match_found:
        # Date not found in model outputs
        incorrect += 1

# Any remaining model outputs are false positives
false_positives = len(model_outputs) -
↪   len(used_model_indices)
total_validation_dates = len(validation_set)
total_model_dates = len(model_outputs)

# Calculate metrics
precision = correct / (correct + false_positives) if
↪   (correct + false_positives) > 0 else 0
recall = correct / total_validation_dates if
↪   total_validation_dates > 0 else 0
f1_score = (2 * precision * recall) / (precision +
↪   recall) if (precision + recall) > 0 else 0
accuracy = correct / total_validation_dates if
↪   total_validation_dates > 0 else 0

# Get hyperparameters from the first execution detail
if execution_metrics:
    hyperparameters =
    ↪   execution_metrics[0].get('hyperparameters',
    ↪   {})
    # Sum processing times
    total_processing_time =
    ↪   sum([em.get('processing_time', 0.0) for em in
    ↪   execution_metrics])
    execution_count = len(execution_metrics)
else:
    hyperparameters = {}
    total_processing_time = 0.0
    execution_count = 0
    print(f"No execution metrics found for model
    ↪   {model_name} on document {document_name}")
```

```
# Create key for the model and hyperparameters
hyperparameters_tuple =
↪   tuple(sorted(hyperparameters.items()))
key = (model_name, hyperparameters_tuple)

# Initialize or update the metrics in the dictionary
if key not in model_hyperparam_metrics:
    model_hyperparam_metrics[key] = {
        'model_name': model_name,
        'hyperparameters': hyperparameters,
        'documents': set(),  # Set of documents
        ↪   evaluated
        'total_correct': 0,
        'total_incorrect': 0,
        'total_false_positives': 0,
        'total_validation_dates': 0,
        'total_model_dates': 0,
        'sum_precision': 0.0,
        'sum_recall': 0.0,
        'sum_f1_score': 0.0,
        'sum_accuracy': 0.0,
        'total_processing_time': 0.0,
        'execution_count': 0,
    }

metrics = model_hyperparam_metrics[key]

# Update the set of documents
metrics['documents'].add(document_name)
```

```
            # Accumulate performance metrics
            metrics['total_correct'] += correct
            metrics['total_incorrect'] += incorrect
            metrics['total_false_positives'] += false_positives
            metrics['total_validation_dates'] +=
            ↪   total_validation_dates
            metrics['total_model_dates'] += total_model_dates
            metrics['sum_precision'] += precision
            metrics['sum_recall'] += recall
            metrics['sum_f1_score'] += f1_score
            metrics['sum_accuracy'] += accuracy

            # Update processing time and execution count
            metrics['total_processing_time'] +=
            ↪   total_processing_time
            metrics['execution_count'] += execution_count

    # Now, after processing all documents and models, prepare the
    ↪   results
    final_results = []

    for key, metrics in model_hyperparam_metrics.items():
        n = len(metrics['documents'])  # Number of documents
        ↪   evaluated
        # Calculate average metrics
        average_precision = metrics['sum_precision'] / n if n > 0
        ↪   else 0
        average_recall = metrics['sum_recall'] / n if n > 0 else
        ↪   0
        average_f1_score = metrics['sum_f1_score'] / n if n > 0
        ↪   else 0
        average_accuracy = metrics['sum_accuracy'] / n if n > 0
        ↪   else 0
        average_processing_time =
        ↪   metrics['total_processing_time'] /
        ↪   metrics['execution_count'] if
        ↪   metrics['execution_count'] > 0 else 0
        # Prepare result entry
        result_entry = {
            'model_name': metrics['model_name'],
            'hyperparameters': metrics['hyperparameters'],
            'documents_evaluated': list(metrics['documents']),
            'total_correct': metrics['total_correct'],
            'total_incorrect': metrics['total_incorrect'],
            'total_false_positives':
            ↪   metrics['total_false_positives'],
            'total_validation_dates':
            ↪   metrics['total_validation_dates'],
            'total_model_dates': metrics['total_model_dates'],
            'average_precision': average_precision,
            'average_recall': average_recall,
            'average_f1_score': average_f1_score,
            'average_accuracy': average_accuracy,
            'total_processing_time':
            ↪   metrics['total_processing_time'],
            'average_processing_time_per_execution':
            ↪   average_processing_time,
        }

        final_results.append(result_entry)

    # Write the final results to the benchmark_results.json file
    with open(results_file, 'w', encoding='utf-8') as f:
        print(f"Results written to '{results_file}'")
```

Code 2—Code for validating the models output

3. Updated clusters for classifying dates here.

```
{
        "options": [
                "fecha de presentacion de demanda",
                "fecha de notificacion de demanda",
                "fecha de audiencia",
                "fecha de fallo reservado",
                "fecha de lectura de sentencia",
                "tiempo desde la independencia",
                "tiempo desde la restauracion",
                "otros: "
        ]
}
```

Code 3—Clusters used for classifying the dates in the legal documents

4. In terms of the models, the benchmark was executed comparing the following models:
   - "llama3.2"
   - "llama3.1"
   - "nemotron-mini"
   - "gemma2"
   - "mistral-nemo"
   - "qwen2"
   - "deepseek-coder-v2"
   - "phi3"
   - "mixtral"
   - "mistral-small"
   - "gemma2:27b"
   - "nemotron"
   - "llama3.1:70b"
   - "qwen2.5:72b"

4 DOCUMENTATION

Similar to what was indicated in past reports, the pipeline/flow we're currently following is the one below, where we first extract and clean the documents. Afterwards, a process takes care of diving the clean documents into smaller pieces that can be then passed as input to a new layer where a Bert based model in Spanish, that has been fine tuned to better identify dates over legal documents for the Dominican Republic, is used to retrieve the dates from the corpus. Once these dates have been identified, they will be passed on to an additional model that will then retrieve the context of the date to identify what it is representing. Finally, all dates will be grouped and included in one file, representing the output of all the pieces of the original document being put together.

The following diagram represents this flow:

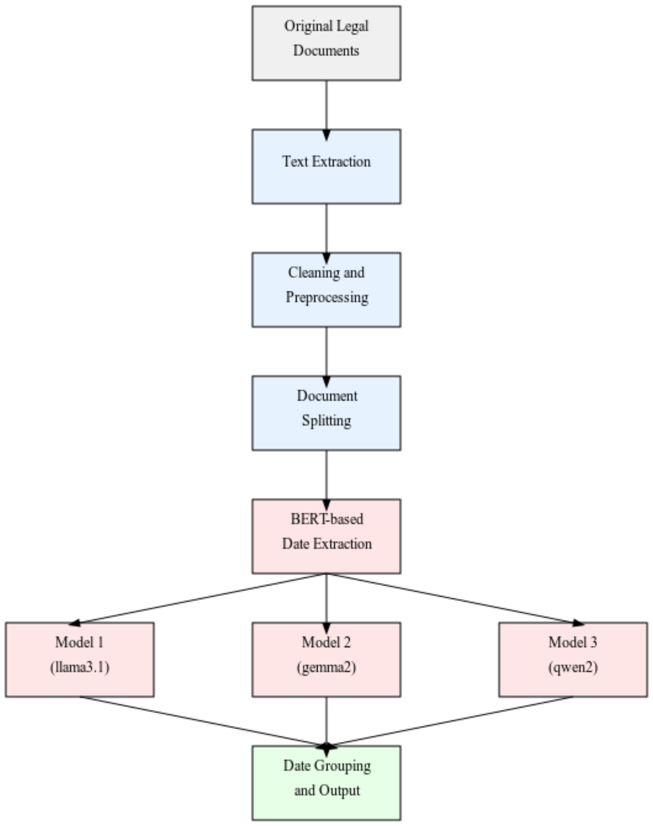


Figure 1—Full date extraction process

This week, my focus has been on the second to last step, comparing the outputs of the different models on the task of retrieving the context of each date and comparing those results to some manually extracted dates and contexts. The process for each has been as follows:

Date context extraction

- Input template generated in txt format to feed the model and retrieve the date context. This template contains placeholders to fill in:
  - Content of the piece of text extracted from the original file where a date is contained (slightly updated this one to obtain different results).
  - Options/clusters template containing the categories by which to classify the different dates retrieved.

  The output of the model will be a single text file containing a JSON object with the input date, a JSON object with the model output and a JSON object containing configuration details for the executed model such as hyperparameters used, model's name and execution time.

5 SCRIPT VALIDATION

The model was queried over a set of 5 files generating 10 outputs for each of the dates contained in the files. Additionally, performance metrics were obtained from each of the models execution, such as execution time.

Each model was triggered with the following hyperparameters:

- Temperature = 0.0000001,
- Top_k = 5,
- Top_p = 0.5
- Seed = 42

Here is a brief explanation of these hyperparameters:

- **Temperature**: A very low temperature (0.0000001) ensures that the outputs will be highly predictable. This is useful when we are looking for consistency and want results to be stable over time.
- **Top-k**: This limits the choices to only the top 5 probable words. This ensures that the model generates meaningful outputs without straying into highly unlikely predictions. It balances between randomness and relevance.

- **Top-p**: Combined with top-k, this gives fine control over the diversity of model output. A top_p value of 0.5 means the model will only consider words that make up 50% of the total probability distribution, ensuring more relevant results.
- **Seed**: Setting the seed makes the experiments reproducible, helpful for research purposes. With the same inputs and hyperparameters, in theory, we should get the same outputs every time (but in practice this doesn't always happen).

All generated files and content may be found here.

6 RESULTS VISUALIZATION

The following images provide the results compared between the different models when retrieving the context for the date given as an input to the model.

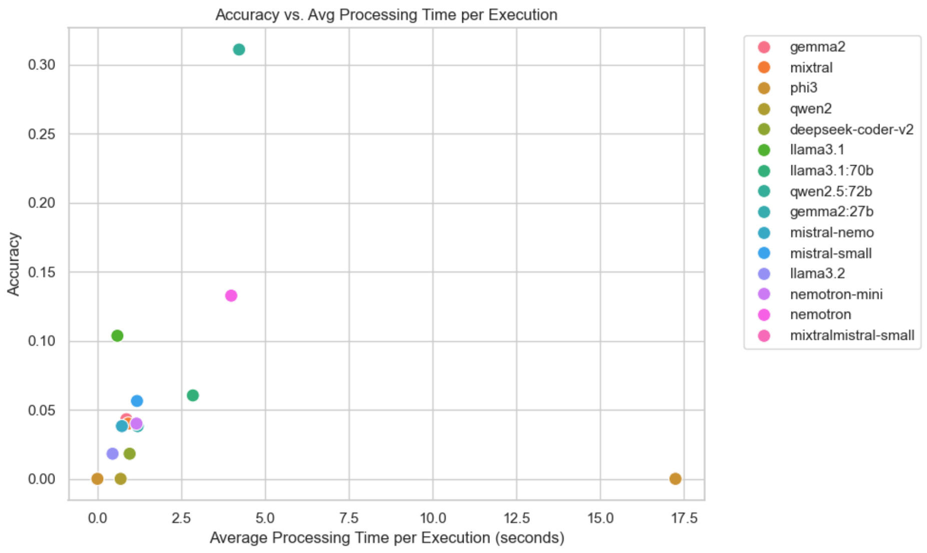


Figure 2—Accuracy vs. Processing time for each model

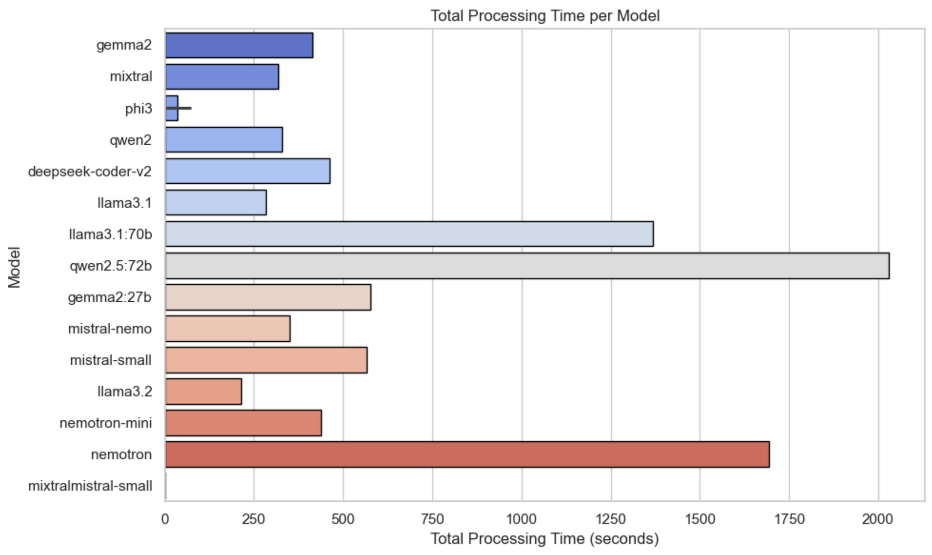


Figure 3—Total processing time for each model

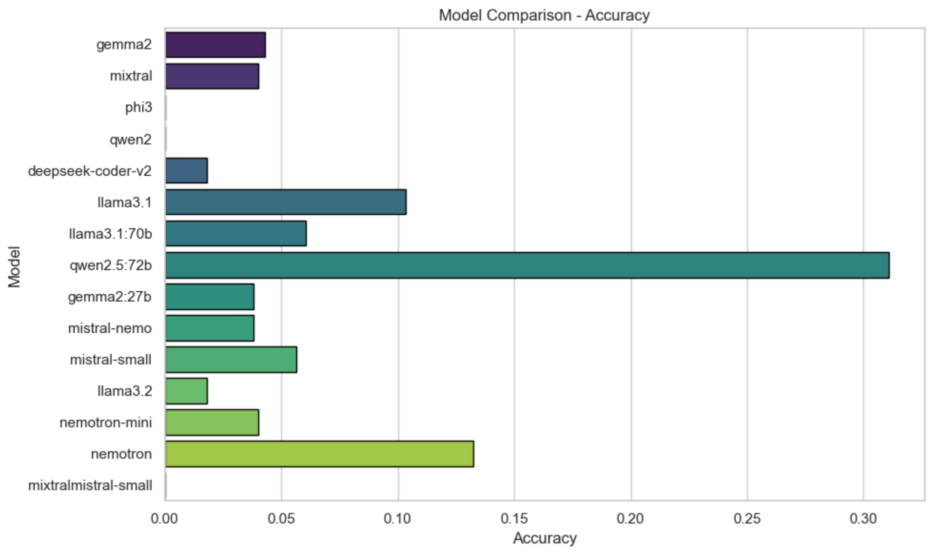


Figure 4—Accuracy comparison between models

7 PROOF OF WORK

The implemented system returns results that follow the correct structure without issues in this area for the larger models. There are some models like Phi3 which tend to generate gibberish content.

Additionally, to ensure stability in the results, each input was used 10 times to generate an output, with the expectations that given the low temperature and the exact same input the model would generate the exact same output. It wasn't the case though. Additional data preparation will have to be carried out by chunking the text into smaller pieces with the expectation that results will be more accurate and additionally should reduce processing times.

By looking at the results we can observe that the qwen2.5:72b model obtained the highest accuracy, with a 0.3 result, which is not enough to guarantee the success of the pipeline being implemented. Also, this results is fully correlated with the time the model took for generating the response, although it did not do so in the case of other models such as nemotron. where the difference in processing time when compared to qwen2.5:72b is way smaller in terms of processing time than in accuracy.

Smaller models, with less parameters, obtained clearly less accurate results when carrying out the classification, indicating as well that this configuration does have an impact on the results of the model for this specific task.

Below is an example of the 10 run obtained by one of the models, where we can clearly observe the randomness of the results. For information purposes, the hyperparameters used were the following ones, as mentioned previously:

- Temperature = 0.0000001,
- Top_k = 5,
- Top_p = 0.5
- Seed = 42

```
{
    "date": "veinticinco (25) días del mes de enero del año dos
     ↪  mil veintitrés (2023)",
    "date event": "fecha de fallo reservado"
  },
  {
    "date": "veinticinco (25) días del mes de enero del año dos
     ↪  mil veintitrés (2023)",
    "date event": "fecha de presentacion de demanda"
  },
  {
    "date": "veinticinco (25) días del mes de enero del año dos
     ↪  mil veintitrés (2023)",
    "date event": "fecha de fallo reservado"
  },
  {
    "date": "veinticinco (25) días del mes de enero del año dos
     ↪  mil veintitrés (2023)",
    "date event": "fecha de lectura de sentencia"
  },
  {
    "date": "veinticinco (25) días del mes de enero del año dos
     ↪  mil veintitrés (2023)",
    "date event": ""
  },
```

Code 4—Example output over 10 iterations with the same input on a llama3.1:70b model

```
{
  "date": "veinticinco (25) días del mes de enero del año dos
  ↪   mil veintitrés (2023)",
  "date event": "fecha de fallo reservado"
},
{
  "date": "veinticinco (25) días del mes de enero del año dos
  ↪   mil veintitrés (2023)",
  "date event": "fecha de presentacion de demanda"
},
{
  "date": "veinticinco (25) días del mes de enero del año dos
  ↪   mil veintitrés (2023)",
  "date event": "fecha de lectura de sentencia"
},
{
  "date": "veinticinco (25) días del mes de enero del año dos
  ↪   mil veintitrés (2023)",
  "date event": "fecha de fallo reservado"
},
{
  "date": "veinticinco (25) días del mes de enero del año dos
  ↪   mil veintitrés (2023)",
  "date event": "fecha de fallo reservado"
},
```

Code 5—Example output over 10 iterations with the same input on a llama3.1:70b model

The expectation here was for the model to provide the adequate date event for each of the provided dates. As it may be observed, for the 10 iterations, only 2 returned the correct result: "fecha de lectura de sentencia". Given this randomness in the results, additional actions will be take to reduce variability in the results: text chunking, model fine-tuning, classes/clusters improval, discard unrelated text.

8 NEXT WEEK'S PROPOSAL

1. Research ways to improve context retrieval for the dates.
2. Implement a simple pipeline to run all processes end to end.
3. Implement a simple UI to interact with the pipeline and upload and process a single file.
4. Work on getting more content in for our paper.

Week 10 | HAAG - NLP | Fall 2024

Alejandro Gomez

October 25th, 2024

1 Time-log

1.1 What progress did you make in the last week?

- This week I continued the drive toward our publication code. The NLP-DR team produced more raw data so I setup a pipeline to fix/clean and QA the data for training the NER model and then converts it to JSONL to be used properly for training. Following the data preprocessing, I ran the training script that finetunes the MMG/xlm-roberta-large-ner-spanish model from HuggingFace for use with Spanish legalese dates. Then I ran some scripts for metrics and charts to visualize the data and compare it to the previous iteration on the model (this will be discussed further later in the report). Using an actual sentencia, I tested the model with this "test dataset" and was able to see an improvement in performance, although there is still some work to be done. I also practiced uploading the model to HuggingFace and using the HuggingFace ecosystem libraries (namely Pipelines) to load the model from the web and use it on the "test dataset" successfully. This will be helpful as we can use this model with the HuggingFace library when we are moving into the evaluation stage of the dates. Additionally, the overleaf template where we will be writing our puiblication is now live and the team has begun contributing toward it. We have a rough draft for the abstract and will continue developments toward writing.

1.2 What are you planning on working on next?

- We have a presentation to work on and deliver to a judge in the DR so I'll be working closely with the group to set up the POC for this work to demo our current project and gather domain knowledge/acceptance criteria from the judge for this tooling/ML pipeline.
- Need to gather more data and train the model with more data to continue improving the NER.
- I need to set a temperature parameter I believe because the model can produce slightly different results on each run and I need it to be consistent.
- I need to start refactoring the pipeline so that any parameters that can be modified, should live in a config file and be pulled in at runtime.
- Now that I understand how to deploy the model to HuggingFace and use it from HugginFace, I need to start building/assembling and writing documentation for it. The labelling for the model on HugginFace also shows "1,2,3" in the JSON file instead of "DATE" in the BIO format so I need to look into this next.
- Iterate on the abstract and assemble references and resources for the publication.

1.3 Is anything blocking you from getting work done?

N/A

2 Article Review

2.1 Abstract

NLP in the legal domain has seen increasing success with the emer- gence of Transformer-based Pre-trained Language Models (PLMs) pre-trained on legal text. PLMs trained over European and US legal text are available publicly; however, legal text from other domains (countries), such as India, have a lot of distinguishing characteris- tics. With the rapidly increasing volume of Legal NLP applications in various countries, it has become necessary to pre-train such LMs over legal text of other countries as well. In this work, we attempt to investigate pre-training in the Indian legal domain. We re-train (continue pre-training) two popular legal PLMs, LegalBERT and CaseLawBERT, on Indian legal data, as well as train a model from scratch with a vocabulary based on Indian legal text. We apply these PLMs over three benchmark legal NLP tasks – Legal Statute Identification from facts, Semantic Segmentation of Court Judg- ment Documents, and Court Appeal Judgment Prediction – over both Indian and non-Indian (EU, UK) datasets. We observe that our approach not only enhances performance on the new domain (Indian texts) but also over the original domain (European and UK texts). We also conduct explainability experiments for a qualitative comparison of all these different PLMs doi[PMGG23]

2.2 Summary

This paper discusses the introduction of new solutions for Indian legal documents by leveraging pretrained-LLM's. This paper discusses how pretrained-language-models (PLM's) are growing in popularity with a few popular one's in the US and EU with respective corpus but a lack of representation for other legal language such as that in Indian, and in our case, that in Latin America. This gives the group confidence in the novelty of our problem and proposed solution. I chose the paper because it was accepted to the current conference the NLP-DR team is targetting and it can serve as a model to understand how to structure the pretraining and data collection approach. Our project has a heavy index toward pretraining and training data and this paper does a good job of explaining and going through these steps.

3 Scripts and Code Blocks

3.1 Code

```

# sample result

snippet_of_result = [
    {
        "sentence": "Circunscripci n de Sato Domingo Oeste, la cual hace constar que
    la joven Creismeryy, es hija de los se ores  ngel  Gonz lez y Lorenza Mej a
    Ramos; \n C. Extracto de acta de nacimiento de fecha treinta (30) de junio del a o
     dos mil veintiuno \n(2021), emitida por la Oficial a del Estado Civil de la 15ta.
    ",
        "entities": [
            {
                "entity_group": "DATE",
                "score": 0.8496034741401672,
                "word": "treinta (30) de junio del a o dos mil veintiuno (2021",
                "start": 191,
                "end": 245
            }
        ]
    },
    {
        "sentence": "Circunscripci n de Sato Domingo Oeste, la cual hace constar que
    la joven Emely, es hija de los se ores  ngel  Gonz lez y Lorenza Mej a Ramos; \n
     D. Declaraci n jurada de uni n libre, de fecha dieciocho (18) de mayo del a o
    dos mil veintid s (2022), instrumentado por el doctor Miguel Cabral Hern ndez,
    notario p blico de los del n mero del Distrito Nacional, mediante el cual los
    se ores  ngel  Gonz lez y Lorenza Mej a, declararon que se encontraban unidos
    sentimentalmente en pareja, mediante la figura legal de esta civil de uni n libre,
     desde el 28 de diciembre del a o 2006, hasta la fecha de dicha declaraci n , y
    que fruto de esa relaci n han procreado dos \n(2) hijas que responden a los
    nombres de Crismery Gonz lez Mej a y Emely Gonz lez Mej a; \n E.",
        "entities": [
            {
                "entity_group": "DATE",
                "score": 0.8161188364028931,
                "word": "dieciocho (18) de mayo del a o dos mil veintid s (2022",
                "start": 190,
                "end": 244
            },
            {
                "entity_group": "DATE",
                "score": 0.8216369152069092,
                "word": "28 de diciembre del a o 2006,",
                "start": 547,
                "end": 576
            }
        ]
    },
]

"""
```

Listing 1: result

For week 10, the model shows improvement in data. This can be seen qualitatively by observing that the dates above are almost perfectly extracted with a much higher confidence score this time around. You can notice there is a ")" left off at the end of the NER for the first two items so training with more data would be beneficial to prevent this type of error.

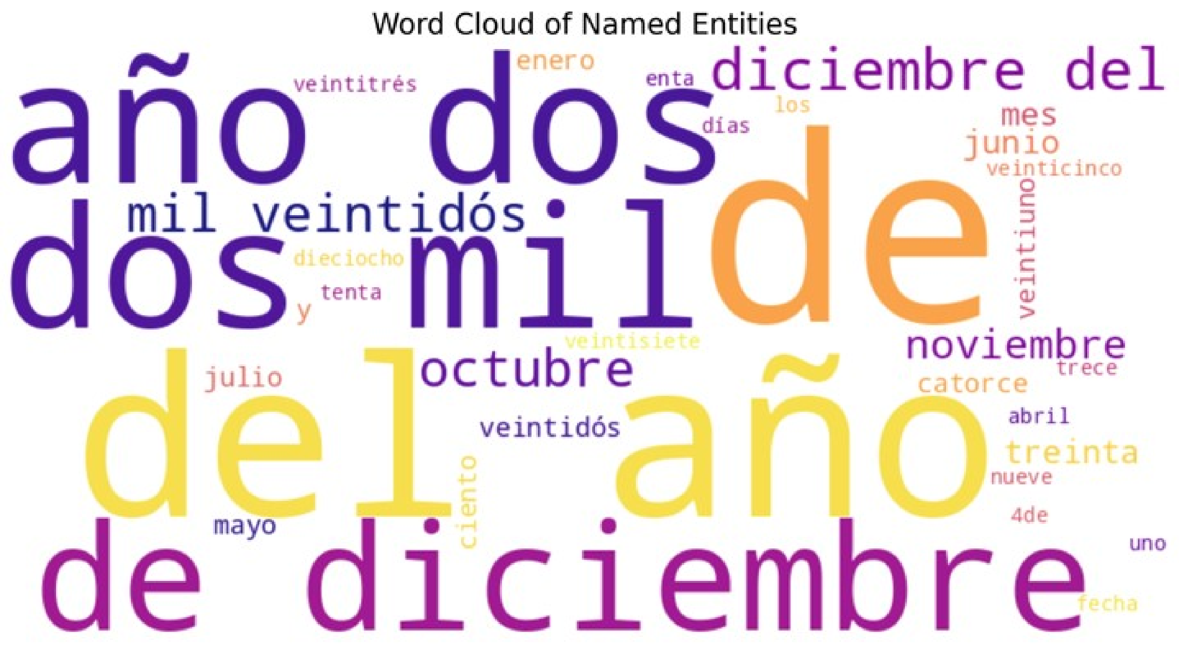


Figure 1: word cloud

The word cloud looks healthy with this week's finetuned model. All of the words displayed are typical makeup of Spanish verbal dates.

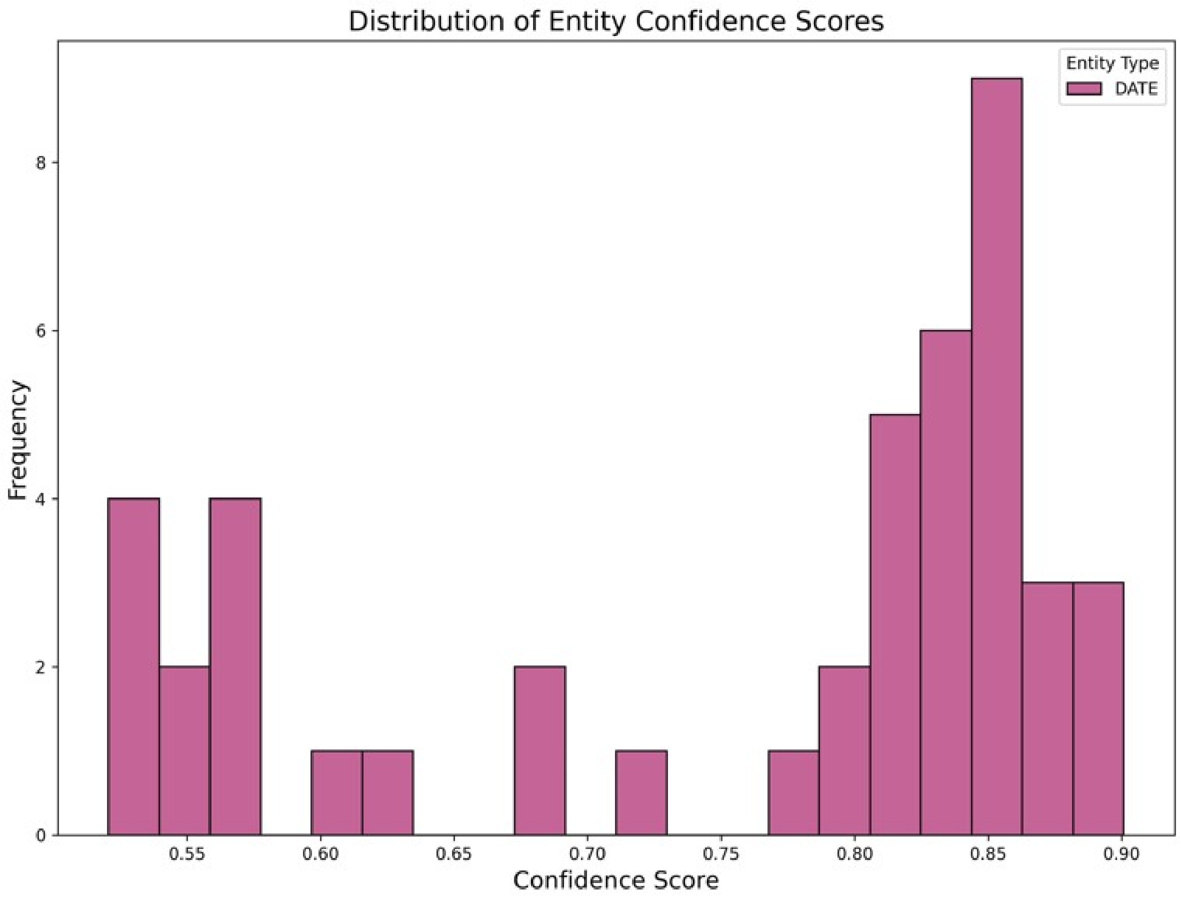
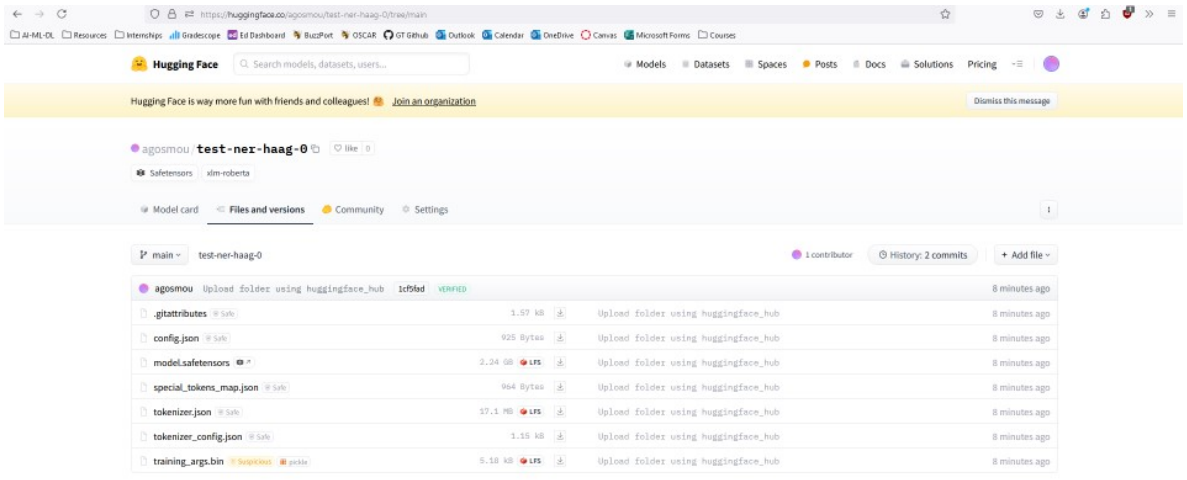


Figure 2: confidence score

This week, the model is indexing toward a higher confidence in its NER. This is obviously a good thing as long as it is confident AND correct rather than confidently incorrect. So this shows promising results as last week's model had very low confidence and generally bad NER with a few correct results in between. Good progress overall, but I need to peruse the results manually as well to ensure they are accurate and precise.



Hugging Face is way more fun with friends and colleagues! Join an organization

agosmou / test-ner-haag-0

Safetensors xlm-roberta

test-ner-haag-0 1 contributor History: 2 commits

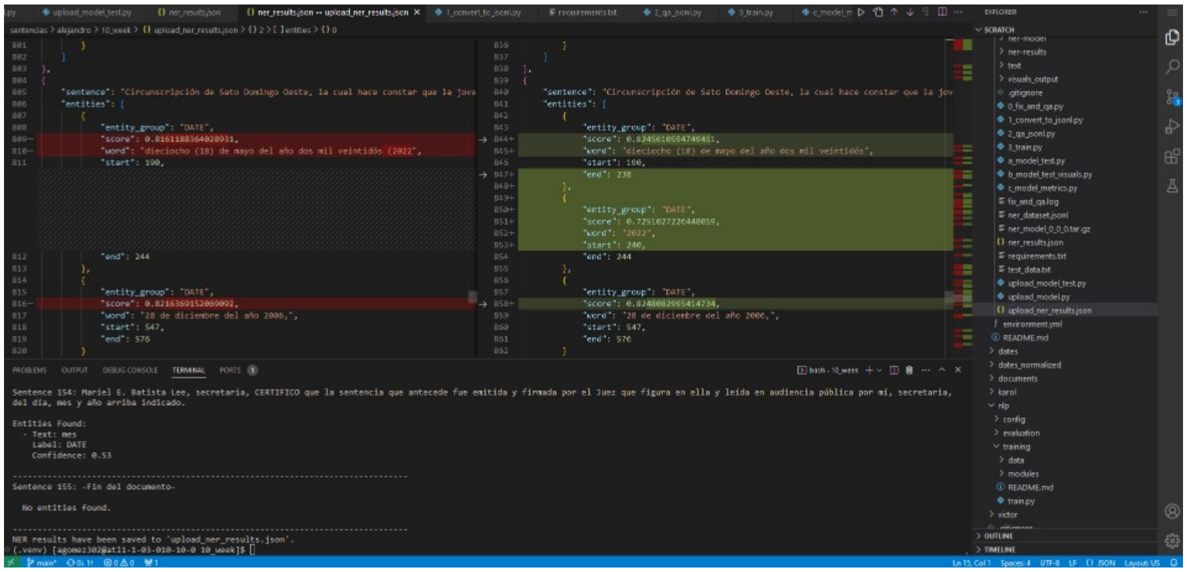
agosmou `Upload folder using huggingface_hub` 1cf5fad VERIFIED 8 minutes ago

.gitattributes	Safe	1.57 kB	Upload folder using huggingface_hub	8 minutes ago
config.json	Safe	925 Bytes	Upload folder using huggingface_hub	8 minutes ago
model.safetensors		2.24 GB LFS	Upload folder using huggingface_hub	8 minutes ago
special_tokens_map.json	Safe	964 Bytes	Upload folder using huggingface_hub	8 minutes ago
tokenizer.json	Safe	17.1 MB LFS	Upload folder using huggingface_hub	8 minutes ago
tokenizer_config.json	Safe	1.15 kB	Upload folder using huggingface_hub	8 minutes ago
training_args.bin	Suspicious pickle	5.18 kB LFS	Upload folder using huggingface_hub	8 minutes ago

Figure 3: Deploying first iteration to HuggingFace as a test run for the deployment pipeline

```python
from huggingface_hub import HfApi
api = HfApi()
api.upload_folder(
    folder_path="ner-model",
    repo_id="agosmou/test-ner-haag-0",
    token="insert-personal-access-token-from-huggingface",
    repo_type="model",

)


# docs
# pip install huggingface_hub
# https://huggingface.co/docs/huggingface_hub/v0.26.0/en/package_reference/hf_api#
    huggingface_hub.HfApi.upload_folder
```

Listing 2: HF Deployment Code



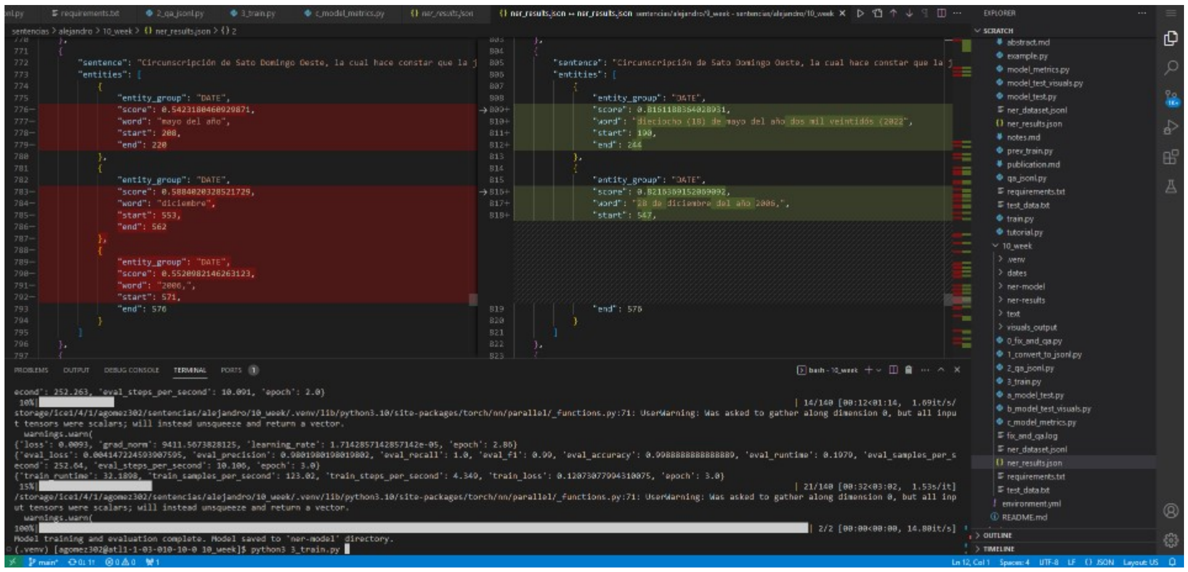
```
            }
        ]
    },
    {
        "sentence": "Circunscripción de Sato Domingo Oeste, la cual hace constar que la jove
        "entities": [
            {
                "entity_group": "DATE",
                "score": 0.8161188364028931,
                "word": "dieciocho (18) de mayo del año dos mil veintidós (2022",
                "start": 190,
                "end": 244
            },
            {
                "entity_group": "DATE",
                "score": 0.8216369152069092,
                "word": "28 de diciembre del año 2006,",
                "start": 547,
                "end": 576
            }
```

```
            }
        ]
    },
    {
        "sentence": "Circunscripción de Sato Domingo Oeste, la cual hace constar que la jov
        "entities": [
            {
                "entity_group": "DATE",
                "score": 0.8245610594749451,
                "word": "dieciocho (18) de mayo del año dos mil veintidós",
                "start": 190,
                "end": 238
            },
            {
                "entity_group": "DATE",
                "score": 0.7251027226448059,
                "word": "2022",
                "start": 240,
                "end": 244
            },
            {
                "entity_group": "DATE",
                "score": 0.8248082995414734,
                "word": "28 de diciembre del año 2006,",
                "start": 547,
                "end": 576
            }
```

```
Sentence 154: Mariel E. Batista Lee, secretaria, CERTIFICO que la sentencia que antecede fue emitida y firmada por el Juez que figura en ella y leída en audiencia pública por mí, secretaria,
del día, mes y año arriba indicado.

Entities Found:
  - Text: mes
    Label: DATE
    Confidence: 0.53

--------------------------------------------------------------------------------
Sentence 155: -Fin del documento-

  No entities found.

--------------------------------------------------------------------------------
NER results have been saved to 'upload_ner_results.json'.
(.venv) [agomez302@atl1-1-03-010-10-0 10_week]$ 
```

Figure 4: Diff between same model locally and in the cloud

The image above compares the NER on the same dataset from the local model that was trained and the HF model, which is just a deployed version of the locla model. The disparity in results will require some finetuning to make the model more deterministic.



```
    },
    {
        "sentence": "Circunscripción de Sato Domingo Oeste, la cual hace constar que la j
        "entities": [
            {
                "entity_group": "DATE",
                "score": 0.5423180460929871,
                "word": "mayo del año",
                "start": 208,
                "end": 220
            },
            {
                "entity_group": "DATE",
                "score": 0.5884020328521729,
                "word": "diciembre",
                "start": 553,
                "end": 562
            },
            {
                "entity_group": "DATE",
                "score": 0.5520982146263123,
                "word": "2006,",
                "start": 571,
                "end": 576
            }
        ]
    },
    {
```

```
    },
    {
        "sentence": "Circunscripción de Sato Domingo Oeste, la cual hace constar que la j
        "entities": [
            {
                "entity_group": "DATE",
                "score": 0.8161188364028931,
                "word": "dieciocho (18) de mayo del año dos mil veintidós (2022",
                "start": 190,
                "end": 244
            },
            {
                "entity_group": "DATE",
                "score": 0.8216389152069092,
                "word": "28 de diciembre del año 2006,",
                "start": 547,
                "end": 576
            }
        ]
    },
    {
```

```
econd': 252.263, 'eval_steps_per_second': 10.091, 'epoch': 2.0}
 10%|                                                                                              | 14/140 [00:12<01:14,  1.69it/s/
storage/ice1/4/1/agomez302/sentencias/alejandro/10_week/.venv/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:71: UserWarning: Was asked to gather along dimension 0, but all inpu
t tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn(
{'loss': 0.0093, 'grad_norm': 9411.5673828125, 'learning_rate': 1.7142857142857142e-05, 'epoch': 2.86}
{'eval_loss': 0.004147224593907595, 'eval_precision': 0.9801980198019802, 'eval_recall': 1.0, 'eval_f1': 0.99, 'eval_accuracy': 0.9988888888888889, 'eval_runtime': 0.1979, 'eval_samples_per_s
econd': 252.64, 'eval_steps_per_second': 10.106, 'epoch': 3.0}
{'train_runtime': 32.1898, 'train_samples_per_second': 123.02, 'train_steps_per_second': 4.349, 'train_loss': 0.12073077994310075, 'epoch': 3.0}
 15%|                                                                                              | 21/140 [00:32<03:02,  1.53s/it]
/storage/ice1/4/1/agomez302/sentencias/alejandro/10_week/.venv/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:71: UserWarning: Was asked to gather along dimension 0, but all inp
ut tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn(
100%|                                                                                              | 2/2 [00:00<00:00, 14.80it/s]
Model training and evaluation complete. Model saved to 'ner-model' directory.
(.venv) [agomez302@atl1-1-03-010-10-0 10_week]$ python3 3_train.py
```

Figure 5: Diff between same model last week with less data and this week with more data

This image above shows the improvement of the model as it's trained with more data by comparing the output on the same dataset between the week 9 model and the week 10 model.

3.2 List of Scripts

- Data Preprocessing
  - fixes and then QA the data provided by teammate
  - converts data to JSONL format for use in training
- Training
  - Finetune the NER model
- Testing
  - Test the local model with never-before-seen dataset, i.e. an actual sentencia
  - wordcloud and confidence charts based on ner results JSON from the above
- Deployment
  - Upload model to HugginFace Repo
  - Test the sentencia dataset with the deployed HF model
- Visualizations
  - compares week 9 and week 10 models

3.3 Documentation

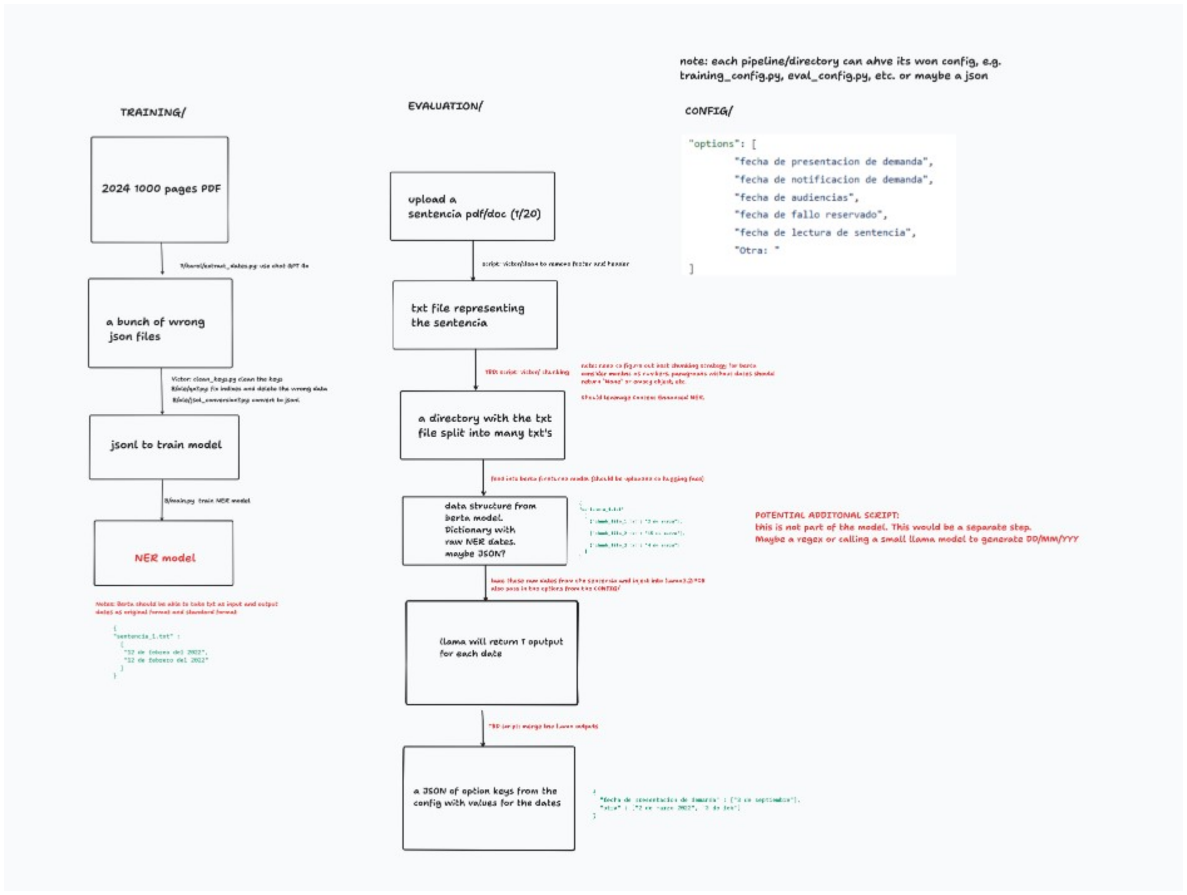

note: each pipeline/directory can ahve its won config, e.g. training_config.py, eval_config.py, etc. or maybe a json
TRAINING/
2024 1000 pages PDF
a bunch of wrong json files
jsonl to train model
NER model
EVALUATION/
upload a sentencia pdf/doc (1/20)
txt file representing the sentencia
a directory with the txt file split into many txt's
data structure from berta model. Dictionary with raw NER dates. maybe JSON?
llama will return 1 oputput for each date
a JSON of option keys from the config with values for the dates
CONFIG/
"options": [
"fecha de presentacion de demanda",
"fecha de notificacion de demanda",
"fecha de audiencias",
"fecha de fallo reservado",
"fecha de lectura de sentencia",
"Otra: "
]
POTENTIAL ADDITONAL SCRIPT:
this is not part of the model. This would be a separate step.
Maybe a regex or calling a small llama model to generate DD/MM/YYY

Figure 6: pipeline visualization for code structure

This is the code structure we assembled last week and included in the report for week 9, but I am including here to discuss some minor modifications. Over the coming weeks, the team will be working to assemble our code for submission so this flow may change slighty. This week, we decided the config will not be JSON but instead will be Python files to allow for flexibility in documentation that allows the team to include in-line comments in the config for better instructions to the user with hyperparameter tuning.

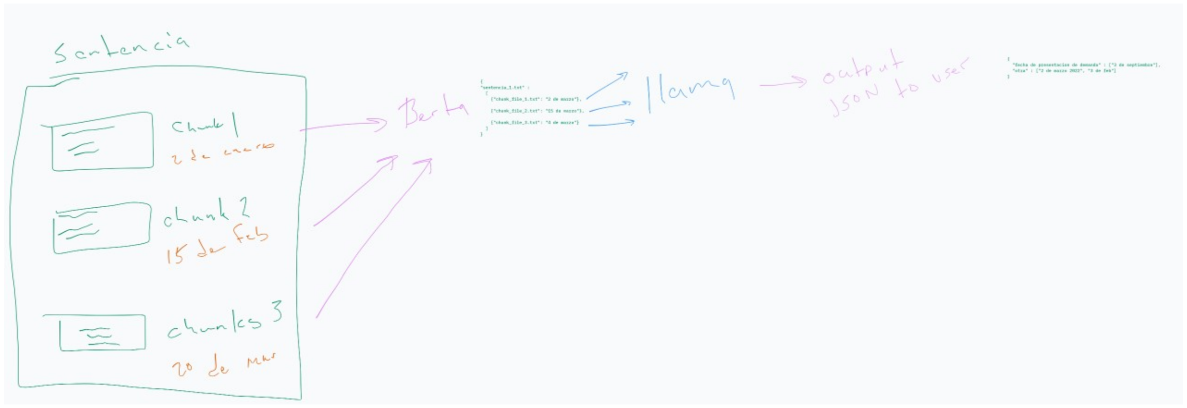


Figure 7: crude pipeline visualization

Changes in this file are potential to substitute llama3.1 70B for more potent models that we are currently benchmarking. Some models that showed promising performance this week in the team's benchmarking: qwen2.5 72B and nemotron.

3.4 Script Validation (optional)

N/A

3.5 Results Visualization

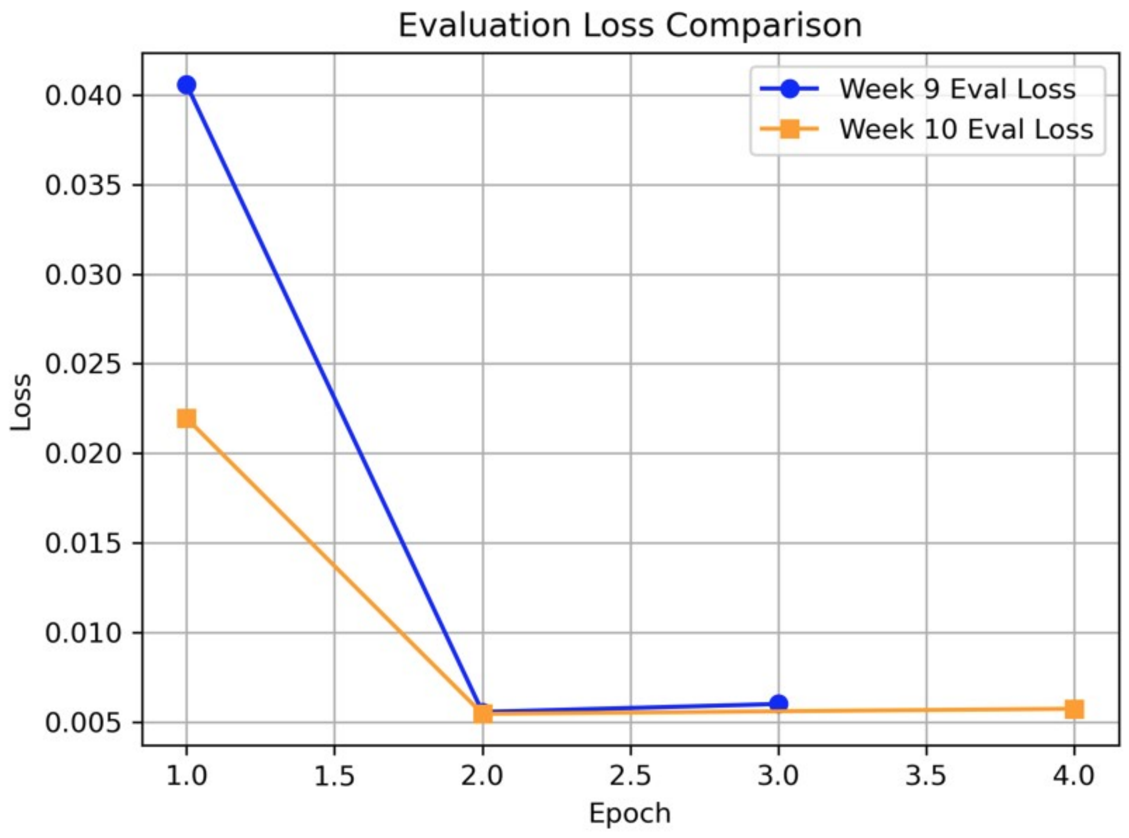


Figure 8: eval loss

Prediction errors minimize as the training epochs complete which is a favorable outcome. Both models have expected shapes for the evaluation loss over the epochs.

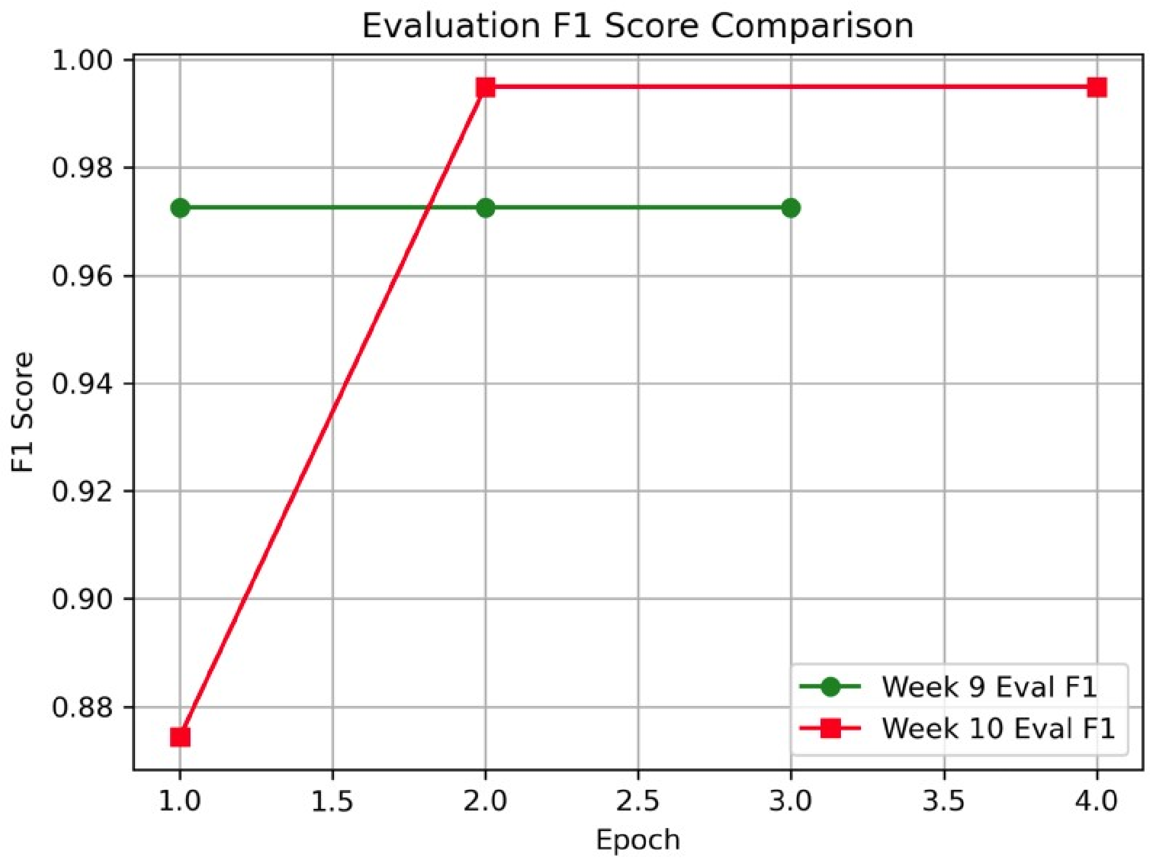


Figure 9: f1 score

F1 score looks better for the week 10 model following a shape closer to y=log(x) which is what I was hoping for with more training data.

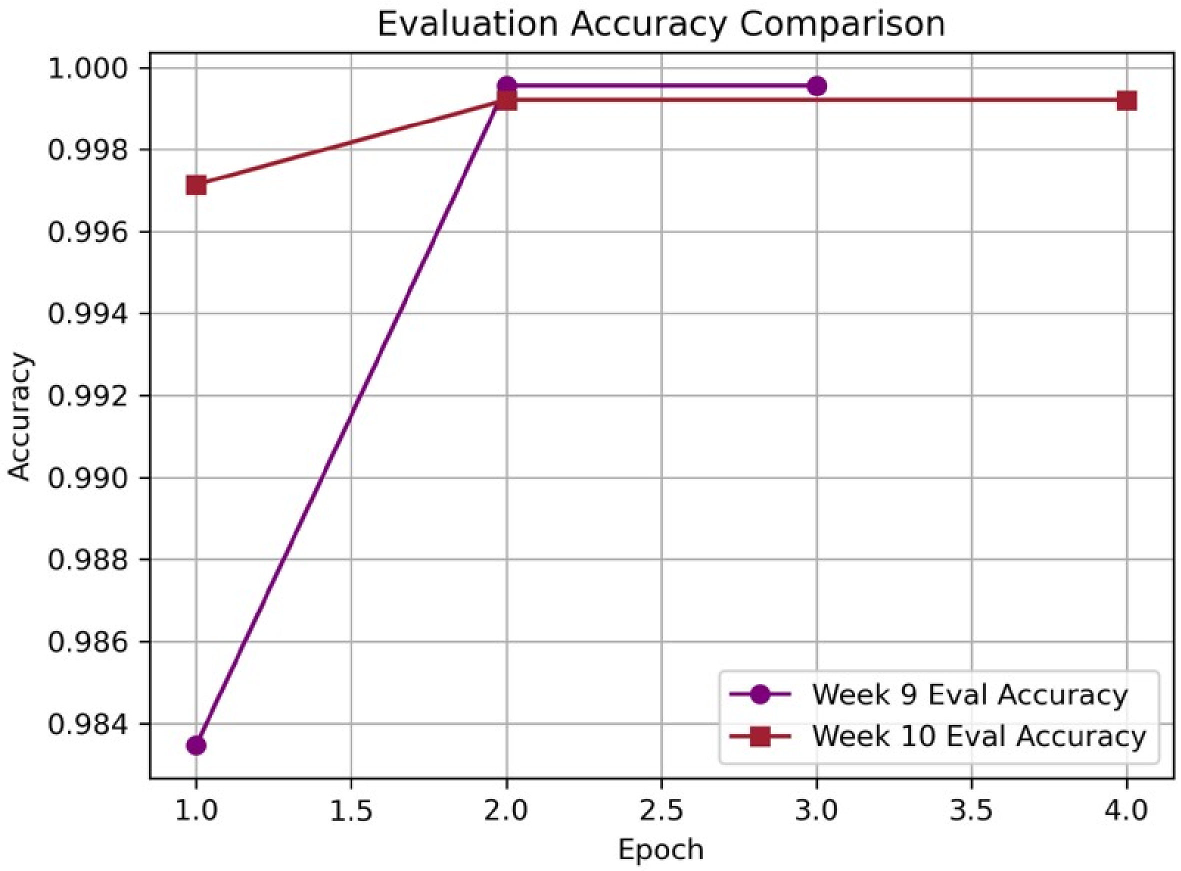


Figure 10: eval accuracy

The accuracy of the model increases generally over the epochs for both models which is a good signal.

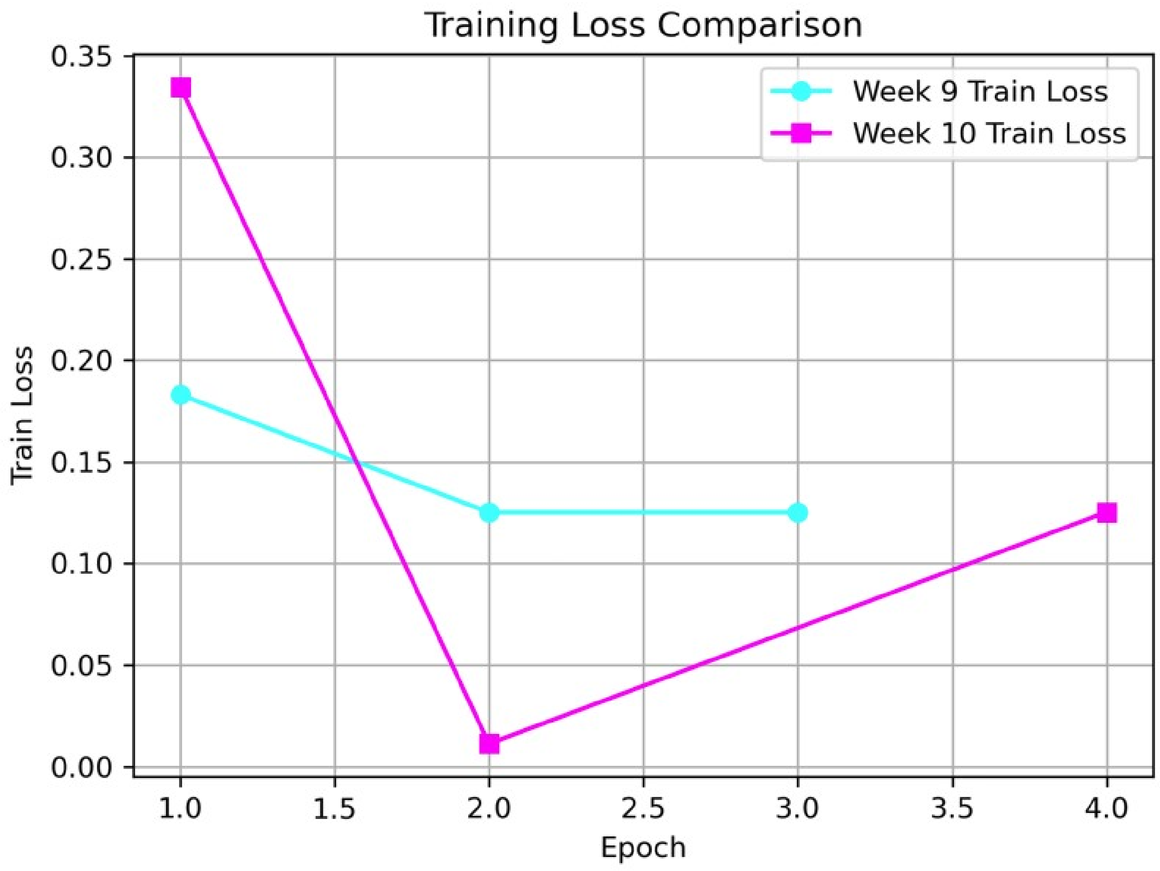


Figure 11: loss

Training loss trends downward and stabilizes in the week 9 model but in week 10 it spikes up toward epoch 4. I will have to see how I can model the behavior in week 9.

3.6 Results Visualization Summary

There were some oddities that surfaced due to the analysis of these visualizations. I retrained the week 9 model with week 9 data but the shape of the graphs looks slightly different. The week 10 training had epochs 1,2 and 4, but skipped number 3. This is very unusual because the model is performing better but it seems like there was a bug in the training steps. Additionally, the graphs look mostly OK with some strange behavior such as the training loss for week 10 where it decreases and isntead of flattening out, it shoots up.

3.7 Proof of Work

Scripts in GitHub Repo

4 Next Week's Proposal

- (See first section for full list. Brief summary below)
- Work on presentation for DR judge
- Improve model for POC during presentation and for publication
- Contribute to publication writing content
- Update current documentation, e.g. NLP website.

References

[PMGG23] Shounak Paul, Arpan Mandal, Pawan Goyal, and Saptarshi Ghosh. Pre-trained language models for the legal domain: A case study on indian law. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, ICAIL '23, page 187–196, New York, NY, USA, 2023. Association for Computing Machinery.

HAAG NLP Sentencias — Week 10 Report
NLP-Gen Team

Karol Gutierrez

October 25, 2024

1 Weekly Project Update

1.1 What progress did you make in the last week?

- Scripts to process data from large PDF file into cleaned split data by court case.
- Generate 800+ more training data files, each containing around 10 dates, using Azure Open AI Studio.
- Updated model with more data.
- Updating OneNote with video recordings and meeting notes
- Fulfill my role as Meet Manager/Documentor by working on the tasks expected for my position.
- Continuous meetings with Dr. Alexander, Nathan and team to discuss progress on project and publication options, as well as internal meetings with team to sync on next steps.

1.2 What are you planning on working on next?

- Add filter for consistency of language in output text (there is mix of Spanish and English in training data).
- Clusterize the context from the training data.
- Improve fine tuning of model.
- Continue fulfilling my role as Meet Manager/Documentor by working on the tasks expected for my position (gather notes from meetings and prepare recordings).

1.3 Is anything blocking you from getting work done?

No.

2 Literature Review

Paper: LexNLP: Natural language processing and information extraction for legal and regulatory texts [aKD18].

2.1 Abstract

LexNLP is an open source Python package focused on natural language processing and machine learning for legal and regulatory text. The package includes functionality to (i) segment documents, (ii) identify key text such as titles and section headings, (iii) extract over eighteen types of structured information like distances and dates, (iv) extract named entities such as companies and geopolitical entities, (v) transform text into features for model training, and (vi) build unsupervised and supervised models such as word embedding or tagging models. LexNLP includes pre-trained models based on thousands of unit tests drawn from real documents available from the SEC EDGAR database as well as various judicial and regulatory proceedings. LexNLP is designed for use in both academic research and industrial applications, and is distributed at https://github.com/LexPredict/lexpredict-lexnlp. Keywords: natural language processing, legal, regulatory, machine learning, segmentation, extraction, open source, Python

2.2 Summary

The paper presents LexNLP as a robust toolkit that facilitates the application of NLP techniques in legal contexts. It offers features for parsing legal documents, extracting key entities (such as parties and dates), and performing text classification tasks. The authors conduct experiments to demonstrate LexNLP's effectiveness, comparing its performance against existing tools and benchmarks in the legal NLP space.

- Data Handling: LexNLP effectively processes a wide range of legal documents, including contracts, court opinions, and statutes, enabling users to extract meaningful insights from unstructured text.
- Entity Recognition: The toolkit incorporates advanced entity recognition algorithms that accurately identify relevant legal entities, significantly improving the extraction process compared to traditional methods.
- Text Classification: LexNLP provides built-in functionalities for classifying legal texts, allowing users to categorize documents based on predefined legal categories, enhancing organization and retrieval.
- Practical Applications: The toolkit is designed to assist legal practitioners, researchers, and developers in building applications that leverage legal data, contributing to advancements in legal technology and improving access to legal information.

2.3 Relevance

The paper is directly relevant to our Sentencias project, as LexNLP provides essential tools for processing and analyzing legal text, which is crucial for extracting procedural history from court decisions. We can use similar approach tailored to the Spanish language.

3 Scripts and code blocks

The code is in the private repository repository. The progress for this week is in `./karol/week10/`.

3.1 Code developed

The following items were developed this week. The full workflow of the code is shown in Figure 1.

- I created new script to retrieve the sentencias parts from the original large PDF document from the Supreme court, this script splits the file into smaller ones for each sentencias, code seen in Figure 2
- Script that calls Azure Studio AI to use GPT-4o API to get the dates and context from each sentencia, updated script and prompt in in Figure 3
- Script that validates the integrity of such data.
- Updated code to create HuggingFace dataset in Figure **??**.
- Use BERT Spanish tokenizer with HuggingFace trainer to produce a trained model in Figure **??**
- Plot results.

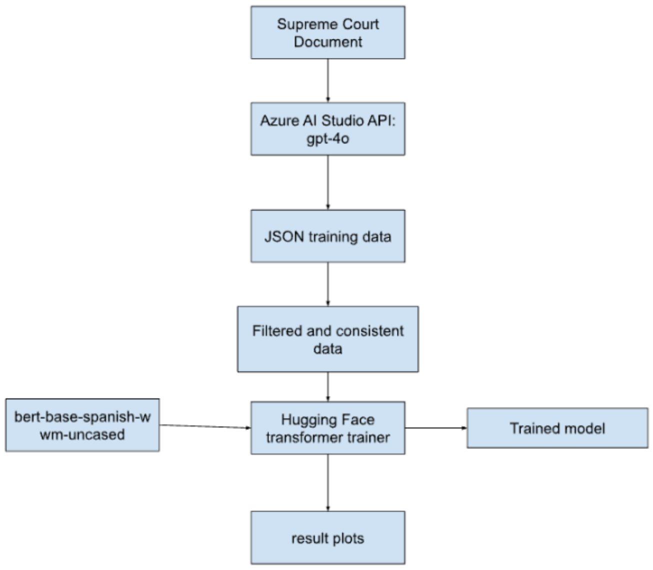

Supreme Court Document
Azure AI Studio API: gpt-4o
JSON training data
Filtered and consistent data
bert-base-spanish-wwm-uncased
Hugging Face transformer trainer
Trained model
result plots

Figure 1: Code logic workflow to process data and train model.

4 Documentation

The documentation is present in the README.md file in the repository. Refer to the repository to get the most updated instructions on how to run the code. For this week, the useful readme is in `./karol/readme.md`.

5 Script Validation

Figure 4 shows the updated validation process for the new generated data.

6 Results Visualization

For this week, the results are related to the generation of training data, some of the folder content can be seen in Figure 5.

7 Proof of Work

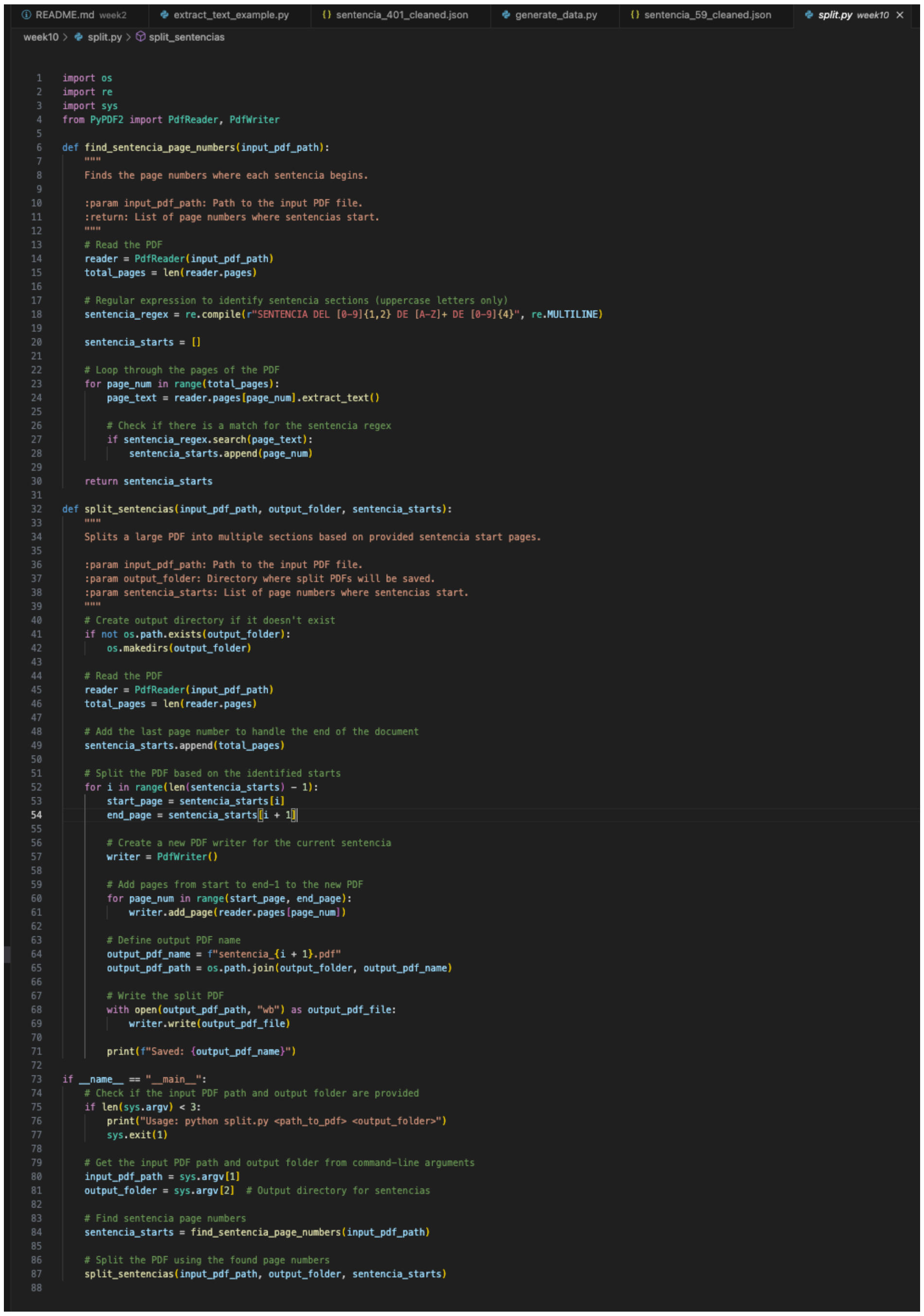
Figures 6 and Figure 7 show the final output of the data generation process, it was observed that sometimes the context was generated in English and sometimes in Spanish, this will be addressed for the next deliverable.

8 Next Week's Proposal

Refer to section 1.2 for details (avoid repetition).

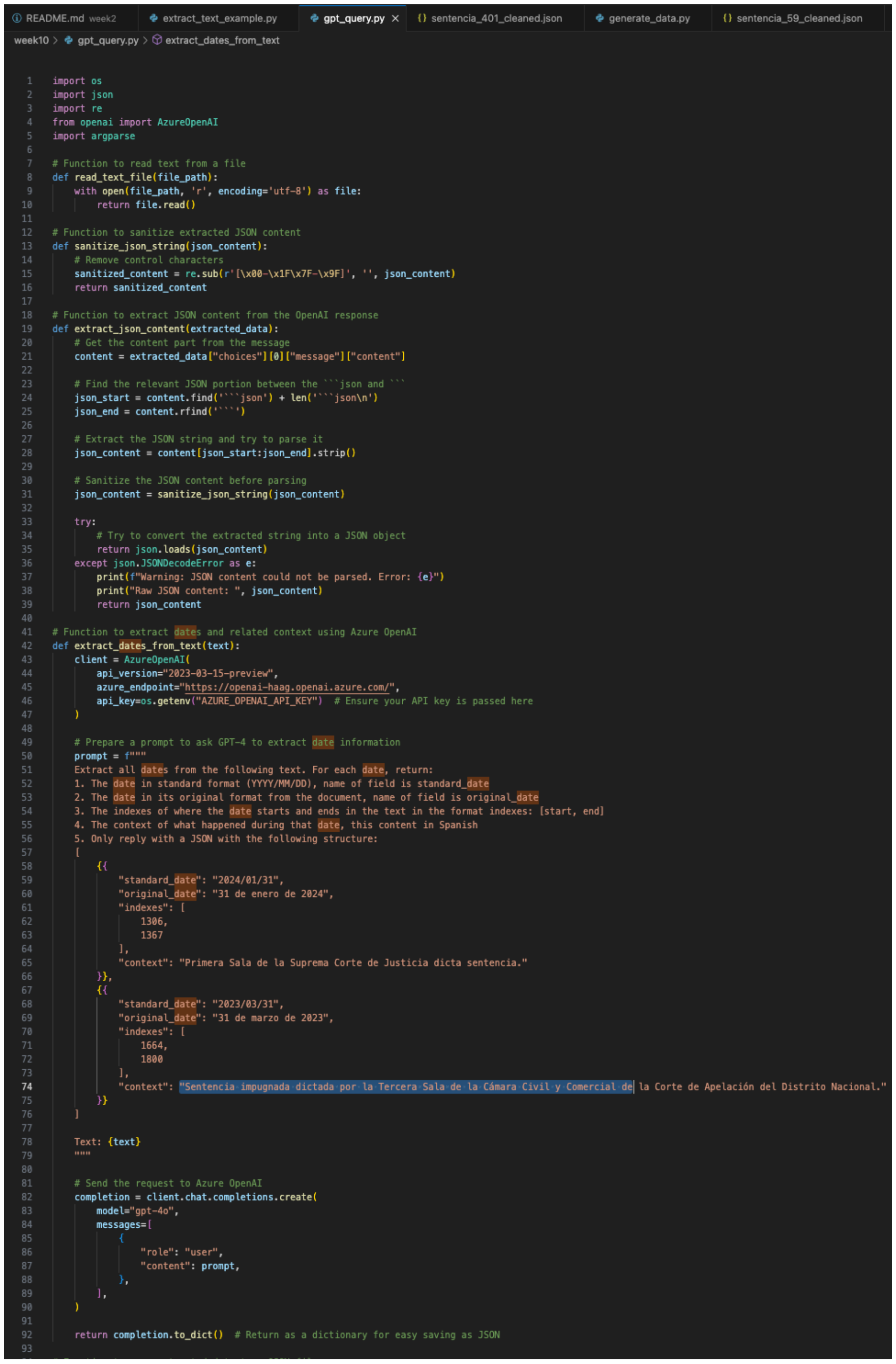
References

[aKD18] Michael J Bommarito II au2, Daniel Martin Katz, and Eric M Detterman. Lexnlp: Natural language processing and information extraction for legal and regulatory texts, 2018.


```python
import os
import re
import sys
from PyPDF2 import PdfReader, PdfWriter

def find_sentencia_page_numbers(input_pdf_path):
    """
    Finds the page numbers where each sentencia begins.

    :param input_pdf_path: Path to the input PDF file.
    :return: List of page numbers where sentencias start.
    """
    # Read the PDF
    reader = PdfReader(input_pdf_path)
    total_pages = len(reader.pages)

    # Regular expression to identify sentencia sections (uppercase letters only)
    sentencia_regex = re.compile(r"SENTENCIA DEL [0-9]{1,2} DE [A-Z]+ DE [0-9]{4}", re.MULTILINE)

    sentencia_starts = []

    # Loop through the pages of the PDF
    for page_num in range(total_pages):
        page_text = reader.pages[page_num].extract_text()

        # Check if there is a match for the sentencia regex
        if sentencia_regex.search(page_text):
            sentencia_starts.append(page_num)

    return sentencia_starts

def split_sentencias(input_pdf_path, output_folder, sentencia_starts):
    """
    Splits a large PDF into multiple sections based on provided sentencia start pages.

    :param input_pdf_path: Path to the input PDF file.
    :param output_folder: Directory where split PDFs will be saved.
    :param sentencia_starts: List of page numbers where sentencias start.
    """
    # Create output directory if it doesn't exist
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Read the PDF
    reader = PdfReader(input_pdf_path)
    total_pages = len(reader.pages)

    # Add the last page number to handle the end of the document
    sentencia_starts.append(total_pages)

    # Split the PDF based on the identified starts
    for i in range(len(sentencia_starts) - 1):
        start_page = sentencia_starts[i]
        end_page = sentencia_starts[i + 1]

        # Create a new PDF writer for the current sentencia
        writer = PdfWriter()

        # Add pages from start to end-1 to the new PDF
        for page_num in range(start_page, end_page):
            writer.add_page(reader.pages[page_num])

        # Define output PDF name
        output_pdf_name = f"sentencia_{i + 1}.pdf"
        output_pdf_path = os.path.join(output_folder, output_pdf_name)

        # Write the split PDF
        with open(output_pdf_path, "wb") as output_pdf_file:
            writer.write(output_pdf_file)

        print(f"Saved: {output_pdf_name}")

if __name__ == "__main__":
    # Check if the input PDF path and output folder are provided
    if len(sys.argv) < 3:
        print("Usage: python split.py <path_to_pdf> <output_folder>")
        sys.exit(1)

    # Get the input PDF path and output folder from command-line arguments
    input_pdf_path = sys.argv[1]
    output_folder = sys.argv[2]  # Output directory for sentencias

    # Find sentencia page numbers
    sentencia_starts = find_sentencia_page_numbers(input_pdf_path)

    # Split the PDF using the found page numbers
    split_sentencias(input_pdf_path, output_folder, sentencia_starts)
```

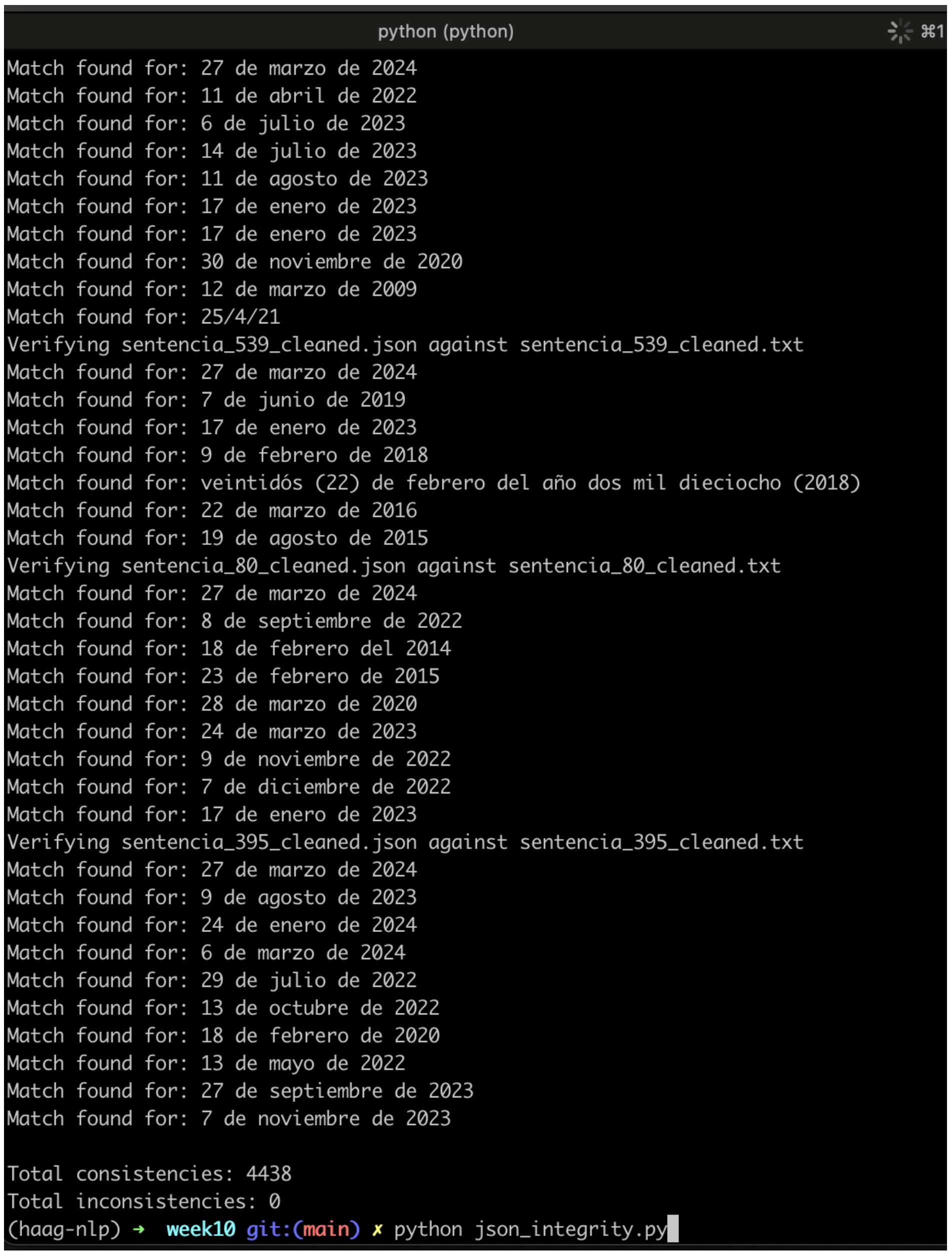
README.md week2
extract_text_example.py
sentencia_401_cleaned.json
generate_data.py
sentencia_59_cleaned.json
split.py week10
week10 > split.py > split_sentencias

Figure 2: Code to split large PDF into sentencias


README.md week2
extract_text_example.py
gpt_query.py
sentencia_401_cleaned.json
generate_data.py
sentencia_59_cleaned.json
week10 > gpt_query.py > extract_dates_from_text

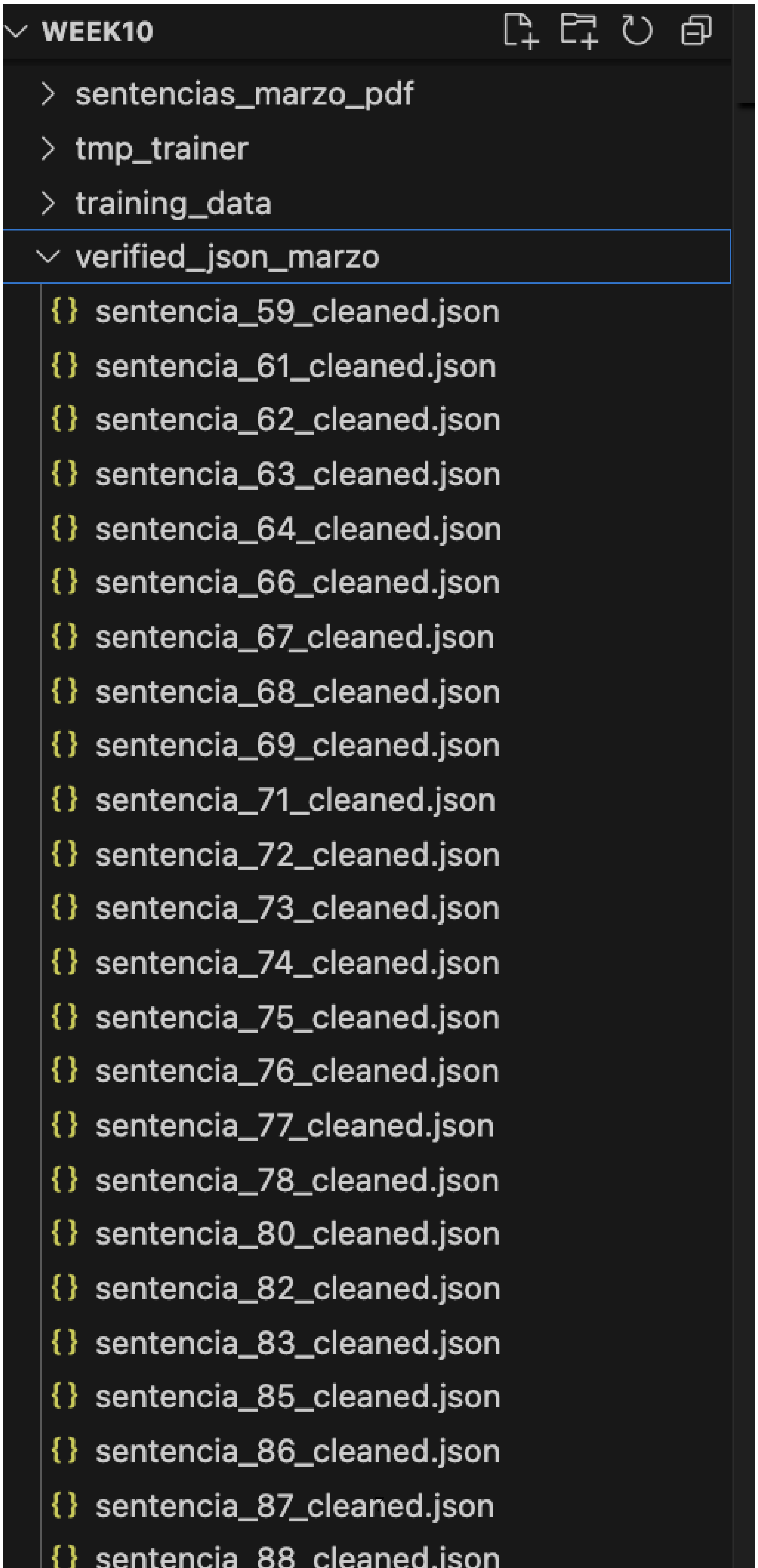
```python
import os
import json
import re
from openai import AzureOpenAI
import argparse

# Function to read text from a file
def read_text_file(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        return file.read()

# Function to sanitize extracted JSON content
def sanitize_json_string(json_content):
    # Remove control characters
    sanitized_content = re.sub(r'[\x00-\x1F\x7F-\x9F]', '', json_content)
    return sanitized_content

# Function to extract JSON content from the OpenAI response
def extract_json_content(extracted_data):
    # Get the content part from the message
    content = extracted_data["choices"][0]["message"]["content"]

    # Find the relevant JSON portion between the ```json and ```
    json_start = content.find('```json') + len('```json\n')
    json_end = content.rfind('```')

    # Extract the JSON string and try to parse it
    json_content = content[json_start:json_end].strip()

    # Sanitize the JSON content before parsing
    json_content = sanitize_json_string(json_content)

    try:
        # Try to convert the extracted string into a JSON object
        return json.loads(json_content)
    except json.JSONDecodeError as e:
        print(f"Warning: JSON content could not be parsed. Error: {e}")
        print("Raw JSON content: ", json_content)
        return json_content

# Function to extract dates and related context using Azure OpenAI
def extract_dates_from_text(text):
    client = AzureOpenAI(
        api_version="2023-03-15-preview",
        azure_endpoint="https://openai-haag.openai.azure.com/",
        api_key=os.getenv("AZURE_OPENAI_API_KEY")  # Ensure your API key is passed here
    )

    # Prepare a prompt to ask GPT-4 to extract date information
    prompt = f"""
    Extract all dates from the following text. For each date, return:
    1. The date in standard format (YYYY/MM/DD), name of field is standard_date
    2. The date in its original format from the document, name of field is original_date
    3. The indexes of where the date starts and ends in the text in the format indexes: [start, end]
    4. The context of what happened during that date, this content in Spanish
    5. Only reply with a JSON with the following structure:
    [
        {{
            "standard_date": "2024/01/31",
            "original_date": "31 de enero de 2024",
            "indexes": [
                1306,
                1367
            ],
            "context": "Primera Sala de la Suprema Corte de Justicia dicta sentencia."
        }},
        {{
            "standard_date": "2023/03/31",
            "original_date": "31 de marzo de 2023",
            "indexes": [
                1664,
                1800
            ],
            "context": "Sentencia impugnada dictada por la Tercera Sala de la Cámara Civil y Comercial de la Corte de Apelación del Distrito Nacional."
        }}
    ]

    Text: {text}
    """

    # Send the request to Azure OpenAI
    completion = client.chat.completions.create(
        model="gpt-4o",
        messages=[
            {
                "role": "user",
                "content": prompt,
            },
        ],
    )

    return completion.to_dict()  # Return as a dictionary for easy saving as JSON
```

Figure 3: GPT Call to generate training data



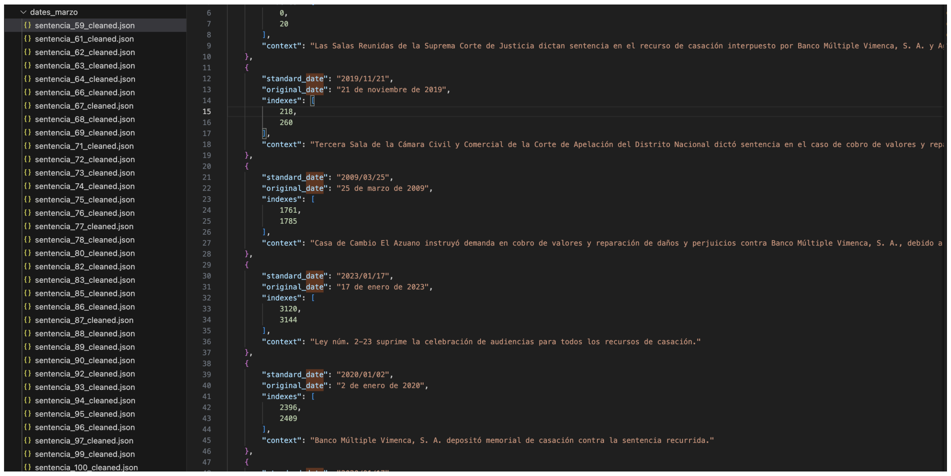
```
Match found for: 27 de marzo de 2024
Match found for: 11 de abril de 2022
Match found for: 6 de julio de 2023
Match found for: 14 de julio de 2023
Match found for: 11 de agosto de 2023
Match found for: 17 de enero de 2023
Match found for: 17 de enero de 2023
Match found for: 30 de noviembre de 2020
Match found for: 12 de marzo de 2009
Match found for: 25/4/21
Verifying sentencia_539_cleaned.json against sentencia_539_cleaned.txt
Match found for: 27 de marzo de 2024
Match found for: 7 de junio de 2019
Match found for: 17 de enero de 2023
Match found for: 9 de febrero de 2018
Match found for: veintidós (22) de febrero del año dos mil dieciocho (2018)
Match found for: 22 de marzo de 2016
Match found for: 19 de agosto de 2015
Verifying sentencia_80_cleaned.json against sentencia_80_cleaned.txt
Match found for: 27 de marzo de 2024
Match found for: 8 de septiembre de 2022
Match found for: 18 de febrero del 2014
Match found for: 23 de febrero de 2015
Match found for: 28 de marzo de 2020
Match found for: 24 de marzo de 2023
Match found for: 9 de noviembre de 2022
Match found for: 7 de diciembre de 2022
Match found for: 17 de enero de 2023
Verifying sentencia_395_cleaned.json against sentencia_395_cleaned.txt
Match found for: 27 de marzo de 2024
Match found for: 9 de agosto de 2023
Match found for: 24 de enero de 2024
Match found for: 6 de marzo de 2024
Match found for: 29 de julio de 2022
Match found for: 13 de octubre de 2022
Match found for: 18 de febrero de 2020
Match found for: 13 de mayo de 2022
Match found for: 27 de septiembre de 2023
Match found for: 7 de noviembre de 2023

Total consistencies: 4438
Total inconsistencies: 0
(haag-nlp) ➜  week10 git:(main) ✗ python json_integrity.py
```

Figure 4: Proof of work for integrity testing



WEEK10

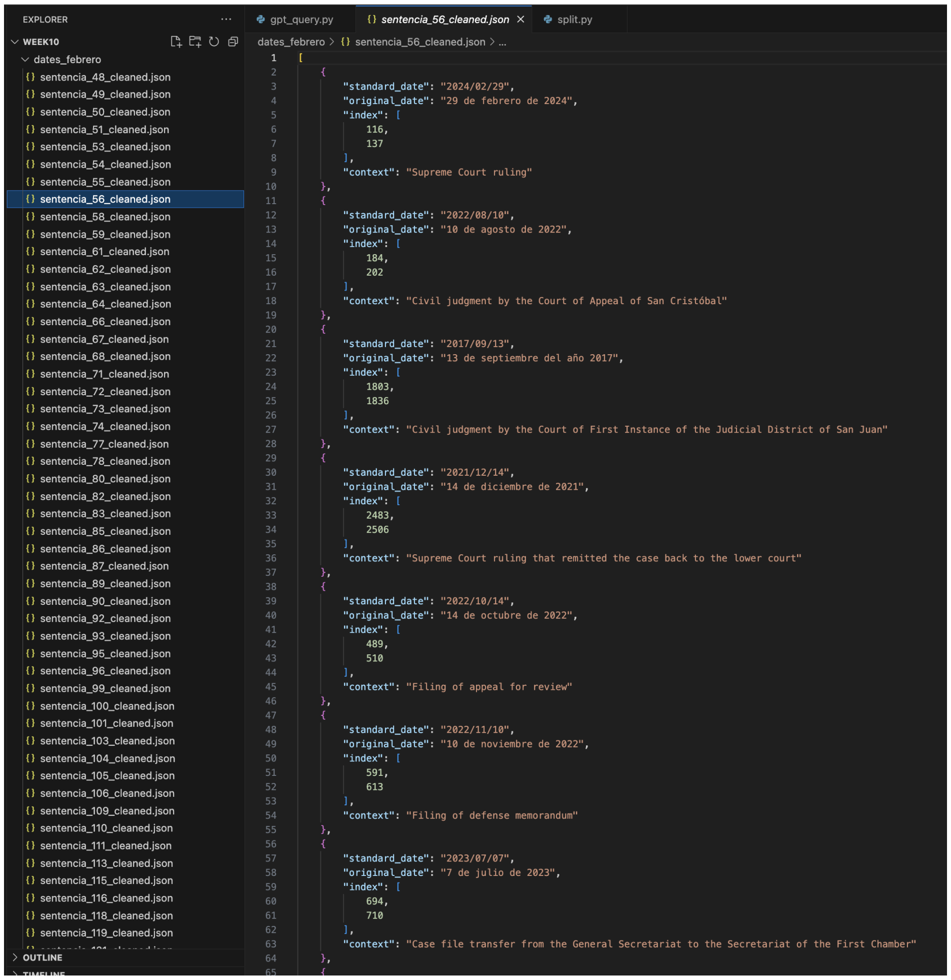
```
> sentencias_marzo_pdf
> tmp_trainer
> training_data
v verified_json_marzo
    sentencia_59_cleaned.json
    sentencia_61_cleaned.json
    sentencia_62_cleaned.json
    sentencia_63_cleaned.json
    sentencia_64_cleaned.json
    sentencia_66_cleaned.json
    sentencia_67_cleaned.json
    sentencia_68_cleaned.json
    sentencia_69_cleaned.json
    sentencia_71_cleaned.json
    sentencia_72_cleaned.json
    sentencia_73_cleaned.json
    sentencia_74_cleaned.json
    sentencia_75_cleaned.json
    sentencia_76_cleaned.json
    sentencia_77_cleaned.json
    sentencia_78_cleaned.json
    sentencia_80_cleaned.json
    sentencia_82_cleaned.json
    sentencia_83_cleaned.json
    sentencia_85_cleaned.json
    sentencia_86_cleaned.json
    sentencia_87_cleaned.json
    sentencia_88_cleaned.json
```



```
dates_marzo
sentencia_59_cleaned.json
sentencia_61_cleaned.json
sentencia_62_cleaned.json
sentencia_63_cleaned.json
sentencia_64_cleaned.json
sentencia_66_cleaned.json
sentencia_67_cleaned.json
sentencia_68_cleaned.json
sentencia_69_cleaned.json
sentencia_71_cleaned.json
sentencia_72_cleaned.json
sentencia_73_cleaned.json
sentencia_74_cleaned.json
sentencia_75_cleaned.json
sentencia_76_cleaned.json
sentencia_77_cleaned.json
sentencia_78_cleaned.json
sentencia_80_cleaned.json
sentencia_82_cleaned.json
sentencia_83_cleaned.json
sentencia_85_cleaned.json
sentencia_86_cleaned.json
sentencia_87_cleaned.json
sentencia_88_cleaned.json
sentencia_89_cleaned.json
sentencia_90_cleaned.json
sentencia_92_cleaned.json
sentencia_93_cleaned.json
sentencia_94_cleaned.json
sentencia_95_cleaned.json
sentencia_96_cleaned.json
sentencia_97_cleaned.json
sentencia_99_cleaned.json
sentencia_100_cleaned.json
```

```
            0,
            20
        ],
        "context": "Las Salas Reunidas de la Suprema Corte de Justicia dictan sentencia en el recurso de casación interpuesto por Banco Múltiple Vimenca, S. A. y A
    },
    {
        "standard_date": "2019/11/21",
        "original_date": "21 de noviembre de 2019",
        "indexes": [
            218,
            260
        ],
        "context": "Tercera Sala de la Cámara Civil y Comercial de la Corte de Apelación del Distrito Nacional dictó sentencia en el caso de cobro de valores y repa
    },
    {
        "standard_date": "2009/03/25",
        "original_date": "25 de marzo de 2009",
        "indexes": [
            1761,
            1785
        ],
        "context": "Casa de Cambio El Azuano instruyó demanda en cobro de valores y reparación de daños y perjuicios contra Banco Múltiple Vimenca, S. A., debido a
    },
    {
        "standard_date": "2023/01/17",
        "original_date": "17 de enero de 2023",
        "indexes": [
            3120,
            3144
        ],
        "context": "Ley núm. 2-23 suprime la celebración de audiencias para todos los recursos de casación."
    },
    {
        "standard_date": "2020/01/02",
        "original_date": "2 de enero de 2020",
        "indexes": [
            2396,
            2409
        ],
        "context": "Banco Múltiple Vimenca, S. A. depositó memorial de casación contra la sentencia recurrida."
    },
    {
```

Figure 6: Generated data format



```
EXPLORER
WEEK10
dates_febrero
sentencia_48_cleaned.json
sentencia_49_cleaned.json
sentencia_50_cleaned.json
sentencia_51_cleaned.json
sentencia_53_cleaned.json
sentencia_54_cleaned.json
sentencia_55_cleaned.json
sentencia_56_cleaned.json
sentencia_58_cleaned.json
sentencia_59_cleaned.json
sentencia_61_cleaned.json
sentencia_62_cleaned.json
sentencia_63_cleaned.json
sentencia_64_cleaned.json
sentencia_66_cleaned.json
sentencia_67_cleaned.json
sentencia_68_cleaned.json
sentencia_71_cleaned.json
sentencia_72_cleaned.json
sentencia_73_cleaned.json
sentencia_74_cleaned.json
sentencia_77_cleaned.json
sentencia_78_cleaned.json
sentencia_80_cleaned.json
sentencia_82_cleaned.json
sentencia_83_cleaned.json
sentencia_85_cleaned.json
sentencia_86_cleaned.json
sentencia_87_cleaned.json
sentencia_89_cleaned.json
sentencia_90_cleaned.json
sentencia_92_cleaned.json
sentencia_93_cleaned.json
sentencia_95_cleaned.json
sentencia_96_cleaned.json
sentencia_99_cleaned.json
sentencia_100_cleaned.json
sentencia_101_cleaned.json
sentencia_103_cleaned.json
sentencia_104_cleaned.json
sentencia_105_cleaned.json
sentencia_106_cleaned.json
sentencia_109_cleaned.json
sentencia_110_cleaned.json
sentencia_111_cleaned.json
sentencia_113_cleaned.json
sentencia_115_cleaned.json
sentencia_116_cleaned.json
sentencia_118_cleaned.json
sentencia_119_cleaned.json
OUTLINE
TIMELINE
```

gpt_query.py
sentencia_56_cleaned.json
split.py
dates_febrero > sentencia_56_cleaned.json > ...

```
[
    {
        "standard_date": "2024/02/29",
        "original_date": "29 de febrero de 2024",
        "index": [
            116,
            137
        ],
        "context": "Supreme Court ruling"
    },
    {
        "standard_date": "2022/08/10",
        "original_date": "10 de agosto de 2022",
        "index": [
            184,
            202
        ],
        "context": "Civil judgment by the Court of Appeal of San Cristóbal"
    },
    {
        "standard_date": "2017/09/13",
        "original_date": "13 de septiembre del año 2017",
        "index": [
            1803,
            1836
        ],
        "context": "Civil judgment by the Court of First Instance of the Judicial District of San Juan"
    },
    {
        "standard_date": "2021/12/14",
        "original_date": "14 de diciembre de 2021",
        "index": [
            2483,
            2506
        ],
        "context": "Supreme Court ruling that remitted the case back to the lower court"
    },
    {
        "standard_date": "2022/10/14",
        "original_date": "14 de octubre de 2022",
        "index": [
            489,
            510
        ],
        "context": "Filing of appeal for review"
    },
    {
        "standard_date": "2022/11/10",
        "original_date": "10 de noviembre de 2022",
        "index": [
            591,
            613
        ],
        "context": "Filing of defense memorandum"
    },
    {
        "standard_date": "2023/07/07",
        "original_date": "7 de julio de 2023",
        "index": [
            694,
            710
        ],
        "context": "Case file transfer from the General Secretariat to the Secretariat of the First Chamber"
    },
    {
```

Figure 7: Proof of generation of dataset

Week 10 Research Report

Thomas Orth (NLP Summarization / NLP Gen Team)

October 2024

0.1 What did you work on this week?

1. Concluded Anthropic tests
2. Tested TogetherAI, no real advantage over Anthropic for use
3. Attended All-hands meeting
4. Looked into commercial finetuning
5. Sent interview team the set of commercial summaries

0.2 What are you planning on working on next?

1. Start summarizing settlements once the data is provided
2. Check Anthropic workbench to see if it can improve prompting
3. Coordinate with subteams as needed

0.3 Is anything blocking you from getting work done?

1. I'll need the Settlement documents from our OCR team + our interview team needs to reach out to the law students so that we can know what kind of relevant info to extract from settlement documents.

1 Abstracts

- Title: Little Giants: Exploring the Potential of Small LLMs as Evaluation Metrics in Summarization in the Eval4NLP 2023 Shared Task. Conference / Venue: ACL 2024, Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Link: https://aclanthology.org/2024.acl-long.51/
- Abstract: Automated evaluation is crucial for streamlining text summarization benchmarking and model development, given the costly and time-consuming nature of human evaluation. Traditional methods like ROUGE

do not correlate well with human judgment, while recently proposed LLM-based metrics provide only summary-level assessment using Likert-scale scores. This limits deeper model analysis, e.g., we can only assign one hallucination score at the summary level, while at the sentence level, we can count sentences containing hallucinations. To remedy those limitations, we propose FineSurE, a fine-grained evaluator specifically tailored for the summarization task using large language models (LLMs). It also employs completeness and conciseness criteria, in addition to faithfulness, enabling multi-dimensional assessment. We compare various open-source and proprietary LLMs as backbones for FineSurE. In addition, we conduct extensive benchmarking of FineSurE against SOTA methods including NLI-, QA-, and LLM-based methods, showing improved performance especially on the completeness and conciseness dimensions. The code is available at https://github.com/DISL-Lab/FineSurE.

- Summary: This paper proposes a better LLM judge approach for summarization. They showed they could do finer grain judging than previous methods. It also finds GPT4o-mini to provide the best overall results as the backend to the code.
- Relevance: As we investigate LLM-as-a-judge approach to evaluating our outputs, this research could prove useful.

2 Relevant Info

- Summary Chain of Thought (CoT) is a technique to prompt LLMs for information to provide context for summarization. I took a domain centric approach in this experiment to extract entities the Clearinghouse is looking for specifically.
- Llama 3.2 is a popular LLM given its performance
- Ollama is a way to serve LLMs locally
- Langchain is a popular library for interacting with LLMs
- Anthropic is a company that produces the Claude family of models that compete with GPT-4.
- Together AI provides open source models through managed infrastructure like ChatGPT
- The two best models in terms of accuracy and cost tradeoff is Claude 3.5 Sonnet and Claude 3 Haiku

3 Scripts

1. All scripts uploaded to https://github.com/Human-Augment-Analytics/NLP-Gen
2. Scripts were run with the following file for testing: `https://gatech.box.com/s/g3heprllvzamua0gwdkhz5k2r34ocgwt`
3. Thomas-Orth/togetherai/domain_specific_scot_chunked.py
   - Brief Description: Run a domain specific version of Summary Chain-of-thought (CoT) on complaints with TogetherAI models, speficially a Llama model..
   - Status: Tested by running the pipeline to completion without issue
   - Important Code Blocks:
     (a) First block: Read in CSV file, choose document
     (b) Second block: Run through prompts, chunking documents, save summaries
     (c) Third Block: Evaluate via manual inspection
   - Screenshot of code: No screenshots provided due to the code being largely the same as last week, just with a different LLM backing it
4. Flow Diagram:

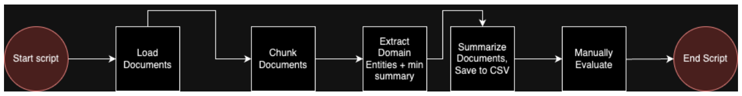

Start script
Load Documents
Chunk Documents
Extract Domain Entities + min summary
Summarize Documents, Save to CSV
Manually Evaluate
End Script

Figure 1: Flow diagram

5. Running scripts:
   (a) Download the scripts, the csv from the box link and llm.requirements.txt
   (b) Run: python -m pip install -r llm.requirements.txt
   (c) Sign up for an TogetherAI account, generate an API Key, and set "TOGETHER_API_KEY" in your environment.
   (d) Run: python [chosen python script]

4 Documentation

1. Download CSV file, with two columns: Document and Summary
2. Update scripts to point to CSV file
3. Run scripts to output generated summaries
4. Manually evaluate summary

5 Results

5.1 Haiku vs Sonnet

5.1.1 Haiku Summary for Macer Case

This was a class action lawsuit filed on November 18, 1987, in the United States District Court for the District of Maryland. The plaintiffs, which included incarcerated individuals at the Talbot County Detention Center (TCDC) and their family members/visitors, were represented by the ACLU National Prison Project and the ACLU of Maryland. The defendants were officials responsible for the operation and oversight of the TCDC, including the Jail Administrator, County Manager, and members of the Talbot County Council. The plaintiffs alleged that the conditions at the TCDC, which was described as an antiquated, overcrowded, and dilapidated facility, violated the First, Sixth, Eighth, and Fourteenth Amendments of the U.S. Constitution. Specifically, they claimed that the conditions amounted to cruel and unusual punishment for sentenced prisoners and impermissible punitive conditions for pre-trial detainees, as well as a denial of the right to visitation, the right to maintain familial relations, and access to the courts. The plaintiffs sought a declaratory judgment that the defendants' actions were unconstitutional, a permanent injunction to prevent the defendants from engaging in the unlawful practices, the retention of jurisdiction by the court until the order was carried out, and reasonable costs and attorneys' fees.

5.1.2 Sonnet Summary for Macer Case

In a class action lawsuit filed on November 18, 1987, prisoners and their families challenged the conditions at the Talbot County Detention Center in Maryland. The case, brought before the United States District Court for the District of Maryland, highlighted severe overcrowding and poor living conditions at the facility. The plaintiffs, represented by the American Civil Liberties Union (ACLU) National Prison Project and ACLU Maryland, alleged various constitutional violations against county officials, including the Jail Administrator, County Manager, and members of the Talbot County Council. The lawsuit claimed violations of the First, Sixth, Eighth, and Fourteenth Amendments to the U.S. Constitution, as well as claims under 42 U.S.C. §1983. The plaintiffs sought several remedies, including a declaratory judgment stating that the defendants' policies violated their constitutional rights, a permanent injunction against unlawful practices, and retention of court jurisdiction until the order was carried out. They also requested an award of reasonable costs and attorneys' fees under 42 U.S.C. §1988, along with other just and equitable relief. The case brought attention to significant issues within the detention center, including inadequate medical care, poor sanitation, insufficient safety measures, and restrictions on visitation and access to courts. These conditions not only affected the inmates but also impacted their families and community members who interacted with the facility.

6 Proof of work

Both Haiku and Sonnet are both powerful models from Anthropic. Through manual verification, the facts presented seem correct based on the complaint.

6.1 Known Limitations

Its tough to say which between the two models are the best or how they stack up compared to GPT4o-mini and Gemini. Sets of summaries were sent to the interview team for review to determine this.