HAAG NLP Summarization Week 8

Michael Bock

October 2024

1 Slack Questions

What did you accomplish this week?

- Finished a training script using Tom's experiment and the huggingface trainer tutorials
- Downloaded DNO labels from UPenn database. This was much easier than I thought, and should have probably been done sooner.

What are you planning on working on next?

- Test the Ollama pipeline on the DNO issues I downloaded
- Provide Clearinghouse.net settlements and opinions, since the summarization subteam is starting to focus more on multi document summarization
- If there is time, try to train a fine tuned BERT model with the pipeline I made. I have a hypothesis that fine tuned models will have the same performance as prompted models because the only thing added to a fine tuned model is an output layer which structures the model's output. However, there are tools like langchain which structure chatbot outputs, thus performing the task that the output layer does. So in theory, the outputs should be the same.

What is blocking you from progressing?

- None

2 Abstract

We introduce LLaMA, a collection of foundation language models ranging from 7B to 65B parameters. We train our models on trillions of tokens, and show that it is possible to train state-of-the-art models using publicly available datasets exclusively, without resorting to proprietary and inaccessible datasets. In particular, LLaMA-13B outperforms GPT-3 (175B) on most benchmarks, and LLaMA-65B is competitive with the best models, Chinchilla-70B and PaLM-540B. We release all our models to the research community.

Link: https://arxiv.org/abs/2302.13971v1

2.1 Brief Analysis

The main idea behind llama is expanding access through reducing inference time. LLaMa cites scaling laws from Hoffman et. al that say that smaller models with more data perform better than larger models with less data. So what LLaMa does with that is take several large, unlabeled, publicly available datasets and train on those with smaller models, producing models like LLaMa 7B which can be run on consumer grade desktops. This is a very useful idea, and its been enabling me to run llama and classify civil rights cases in the clearinghouse dataset on my local computer instead of having to pay for an expensive API or cloud instance. Training small models on more data opens up the ability to try to prompt chatbots for classifications of our DNO issues dataset and gives us an avenue of investigation in generative classification.

3 Scripts and Code Blocks

get_dataset.py

```python
import numpy as np
from tqdm import tqdm
import sys
sys.path.append('../../')
from summarizers.ocr import extract_text_from_pdf
import os
import pypdf

dno = mysql.connector.connect(
        host="127.0.0.1",
        user="report",
        password="<UPENN DB PASSWORD OMITTED>",
        port=3307,
        database='sla2_prod'
       )

cursor = dno.cursor()

cursor.execute("select * from taxonomy_term_data where vid=43;")
terms = cursor.fetchall()
term_dict = {}
for x in terms:
    term_dict[x[0]] = x[2]
json.dump(term_dict, open("terms.json", 'w'))

cursor.execute("select etp.entity_id event_nid, fc.field_case_target_id case_nid, (
    select group_concat(substr(fm.uri,11) order by d.delta separator '\n') from
    field_data_field_documents d, file_managed fm where d.entity_type='node' and d.
    entity_id=etp.entity_id and fm.fid=d.field_documents_fid) docs from
    field_data_field_event_type etp inner join field_data_field_case fc on fc.
    entity_type='node' and fc.entity_id=etp.entity_id where etp.field_event_type_tid
    =1774;")

myresult = cursor.fetchall()
issues_columns = np.zeros((len([x for x in myresult if x[-1] is not None and os.path
    .exists(os.path.join('pdfs', os.path.split(x[-1])[-1]))]) - 1, len(terms)))
case_names = []
case_text = []
case_num = 0
```

```python
for i, x in enumerate(tqdm(myresult)):
    try:
        if x[-1] is not None and os.path.exists(os.path.join('pdfs', os.path.split(x[-1])[-1])):
            case_text.append(extract_text_from_pdf(os.path.join('pdfs', os.path.split(x[-1])[-1])))
            cursor.execute(f"select field_d_o_issues_tid from field_data_field_d_o_issues where entity_id={x[0]};")
            issues_in_case = cursor.fetchall()
            for issue in issues_in_case:
                issues_columns[case_num, list(term_dict.keys()).index(issue[0])] = int(1)
            case_num += 1
            case_names.append(x[-1])
    except pypdf.errors.PdfStreamError:
        pass
df_dict = {}
for tid in term_dict.keys():
    df_dict[term_dict[tid]] = issues_columns[:, list(term_dict.keys()).index(tid)]
    print(len(df_dict[term_dict[tid]]))
df_dict['Name'] = case_names
df_dict['Text'] = case_text
print(len(case_text))
print(len(case_names))

pd.DataFrame.from_dict(df_dict).to_csv('dno_labels.csv', index = False)
```

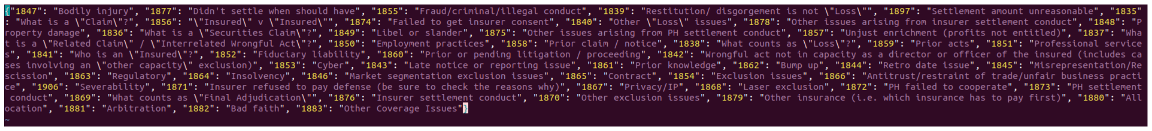
4 Documentation

This is the script I used to scrape the Upenn database to get DNO issues and assign them to our PDFs. I did this because without labels, the PDFs aren't useful. I can try classifying the PDFs with Ollama(the tool I use to run llama 3.1) without the issue labels, but without those labels I wouldn't be able to see if Ollama was correct or evaluate it in any other way. So this script is very improtant to being able to create issue classifiers. To run the script, you first need to ssh into UPenn's server: `ssh -L 3307:ssla-upenn.cluster-ctch2zttdbj0.us-east-1.rds.amazonaws.com:3306 report@lt.law.upenn.edu` Then, leave the terminal with that ssh session running and run the `get_dataset.py` script: `python3 get_dataset.py`. The script will dump a csv which you can then use to train a classifier and a `terms.json` file, which contains a mapping from `tid`(I think this stands for term ID or taxonomy ID) to the names of the DNO issues.

5 Scription Validation(Optional)

https://youtu.be/Ij_9BfirZfo

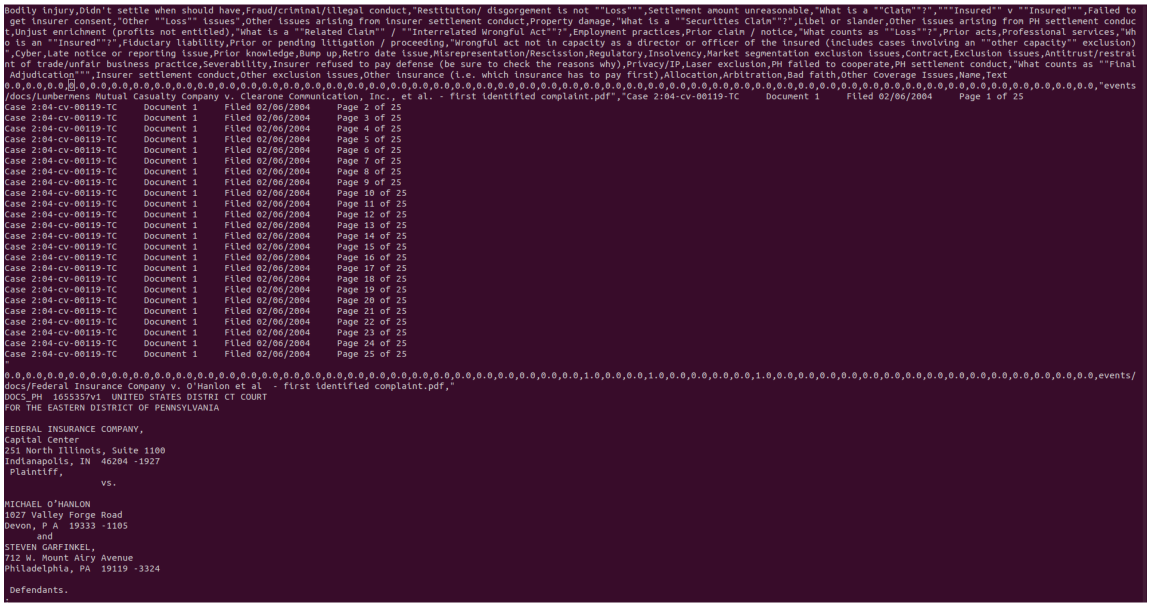
6 Results Visualization



```
{"1847": "Bodily injury", "1877": "Didn't settle when should have", "1855": "Fraud/criminal/illegal conduct", "1839": "Restitution/ disgorgement is not \"Loss\"", "1897": "Settlement amount unreasonable", "1835"
: "What is a \"Claim\"?", "1856": "\"Insured\" v \"Insured\"", "1874": "Failed to get insurer consent", "1840": "Other \"Loss\" issues", "1878": "Other issues arising from insurer settlement conduct", "1848": "P
roperty damage", "1836": "What is a \"Securities Claim\"?", "1849": "Libel or slander", "1875": "Other issues arising from PH settlement conduct", "1857": "Unjust enrichment (profits not entitled)", "1837": "Wha
t is a \"Related Claim\" / \"Interrelated Wrongful Act\"?", "1850": "Employment practices", "1858": "Prior claim / notice", "1838": "What counts as \"Loss\"?", "1859": "Prior acts", "1851": "Professional service
s", "1841": "Who is an \"Insured\"?", "1852": "Fiduciary liability", "1860": "Prior or pending litigation / proceeding", "1842": "Wrongful act not in capacity as a director or officer of the insured (includes ca
ses involving an \"other capacity\" exclusion)", "1853": "Cyber", "1843": "Late notice or reporting issue", "1861": "Prior knowledge", "1862": "Bump up", "1844": "Retro date issue", "1845": "Misrepresentation/Re
scission", "1863": "Regulatory", "1864": "Insolvency", "1846": "Market segmentation exclusion issues", "1865": "Contract", "1854": "Exclusion issues", "1866": "Antitrust/restraint of trade/unfair business practi
ce", "1906": "Severability", "1871": "Insurer refused to pay defense (be sure to check the reasons why)", "1867": "Privacy/IP", "1868": "Laser exclusion", "1872": "PH failed to cooperate", "1873": "PH settlement
 conduct", "1869": "What counts as \"Final Adjudication\"", "1876": "Insurer settlement conduct", "1870": "Other exclusion issues", "1879": "Other insurance (i.e. which insurance has to pay first)", "1880": "All
ocation", "1881": "Arbitration", "1882": "Bad faith", "1883": "Other Coverage Issues"}
~
```

Figure 1: Terms

7 Proof of work



```
Bodily injury,Didn't settle when should have,Fraud/criminal/illegal conduct,"Restitution/ disgorgement is not ""Loss""",Settlement amount unreasonable,"What is a ""Claim""?","""Insured"" v ""Insured""",Failed to
 get insurer consent,"Other ""Loss"" issues",Other issues arising from insurer settlement conduct,Property damage,"What is a ""Securities Claim""?",Libel or slander,Other issues arising from PH settlement conduc
t,Unjust enrichment (profits not entitled),"What is a ""Related Claim"" / ""Interrelated Wrongful Act""?",Employment practices,Prior claim / notice,"What counts as ""Loss""?",Prior acts,Professional services,"Wh
o is an ""Insured""?",Fiduciary liability,Prior or pending litigation / proceeding,"Wrongful act not in capacity as a director or officer of the insured (includes cases involving an ""other capacity"" exclusion)
",Cyber,Late notice or reporting issue,Prior knowledge,Bump up,Retro date issue,Misrepresentation/Rescission,Regulatory,Insolvency,Market segmentation exclusion issues,Contract,Exclusion issues,Antitrust/restrai
nt of trade/unfair business practice,Severability,Insurer refused to pay defense (be sure to check the reasons why),Privacy/IP,Laser exclusion,PH failed to cooperate,PH settlement conduct,"What counts as ""Final
 Adjudication""",Insurer settlement conduct,Other exclusion issues,Other insurance (i.e. which insurance has to pay first),Allocation,Arbitration,Bad faith,Other Coverage Issues,Name,Text
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,"events
/docs/Lumbermens Mutual Casualty Company v. Clearone Communication, Inc., et al. - first identified complaint.pdf","Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 1 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 2 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 3 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 4 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 5 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 6 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 7 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 8 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 9 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 10 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 11 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 12 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 13 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 14 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 15 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 16 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 17 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 18 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 19 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 20 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 21 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 22 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 23 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 24 of 25
Case 2:04-cv-00119-TC     Document 1     Filed 02/06/2004     Page 25 of 25
"
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,events/
docs/Federal Insurance Company v. O'Hanlon et al  - first identified complaint.pdf,"
DOCS_PH  1655357v1  UNITED STATES DISTRI CT COURT
FOR THE EASTERN DISTRICT OF PENNSYLVANIA

FEDERAL INSURANCE COMPANY,
Capital Center
251 North Illinois, Suite 1100
Indianapolis, IN  46204 -1927
 Plaintiff,
                 vs.

MICHAEL O'HANLON
1027 Valley Forge Road
Devon, P A  19333 -1105
      and
STEVEN GARFINKEL,
712 W. Mount Airy Avenue
Philadelphia, PA  19119 -3324

 Defendants.
```

Figure 2: CSV that we can use to make classifiers. Each DNO issue has a column and the value is 1 if the DNO issue is present in a case and 0 if its not.

8 Next Week's proposal

- Test the Ollama pipeline on the DNO issues I downloaded
- Provide Clearinghouse.net settlements and opinions, since the summarization subteam is starting to focus more on multi document summarization
- If there is time, try to train a fine tuned BERT model with the pipeline I made. I have a hypothesis that fine tuned models will have the same performance as prompted models because the only thing added to a fine tuned model is an output layer which structures the model's output. However, there are tools like langchain which structure chatbot outputs, thus performing the task that the output layer does. So in theory, the outputs should be the same.

HAAG Research Report
NLP - Sentencias / NLP - Gen Team
Week 9

Víctor C. Fernández

October 2024

CONTENTS

1 WEEKLY PROJECT UPDATES

What progress did you make in the last week?

- Researched ways of tokenizing and training models provided via Hugging face.
- Moved code to PACE/ICE environment and successfully loaded and ran Ollama.
- Pulled and executed current data with the new Nemotron model provided by Nvidia.
- Met with the NLP-Sentencias team on Sunday 13th to align on our goals and distribute our tasks more efficiently, as well as discussing over target conference.
- Created webinar events for Tuesday, October 22nd and Tuesday, November 12th.
- Meeting with the NLP team on October 18th for our weekly meeting.
- Meeting with Dr. Alexander and Nathan Dahlberg on October 18th to get further insights on NLP research.

What progress are you making next?

- Finish verifying results from Nemotron model in PACE environment to verify correct outcome.
- Align with team to define naming convention and pipeline for full process execution.
- Work on getting an abstract ready for our paper
- Meet with the NLP team on October 25th for our weekly meeting.
- Meet with Dr. Alexander and Nathan Dahlberg on October 25th to get further insights on NLP research.

Is there anything blocking you from making progress?

No significant blockers at this time.

2 ABSTRACTS

1. **Title:** Segmentation of legal documents
   - **URL:** https://dl.acm.org/doi/10.1145/1568234.1568245 (requires accessing with GaTech credentials to view actual paper)

   - **Abstract:** An overwhelming number of legal documents is available in digital form. However, most of the texts are usually only provided in a semi-structured form, i.e. the documents are structured only implicitly using text formatting and alignment. In this form the documents are perfectly understandable by a human, but not by a machine. This is an obstacle towards advanced intelligent legal information retrieval and knowledge systems. The reason for this lack of structured knowledge is that the conversion of texts in conventional form into a structured, machine-readable form, a process called segmentation, is frequently done manually and is therefore very expensive.
     We introduce a trainable system based on state-of-the-art Information Extraction techniques for the automatic segmentation of legal documents. Our system makes special use of the implicitly given structure in the source digital file as well as of the explicit knowledge about the target structure. Our evaluation on the French IPR Law demonstrates that the system is able to learn an effective segmenter given only a few manually processed training documents. In some cases, even only one seen example is sufficient in order to correctly process the remaining documents.

   - **Summary:** This paper addresses the semi-structured nature of legal texts, which are understandable by humans but not directly usable by machines for tasks like advanced information retrieval or knowledge extraction. The authors propose a trainable approach that makes use of both implicit formatting and explicit knowledge about the structure of the documents. The paper focuses on segmenting a corpus of French Intellectual Property Law into individual articles, demonstrating that the system can learn effectively from only a few manually annotated examples. The proposed system achieved high accuracy in segmenting legal documents and shows promise for broader applications beyond the specific dataset.

- **Relevance:** Our project also involves processing semi-structured legal documents to extract specific information, in this case, dates that are critical for identifying delays in civil case resolutions. The methodology described in the paper, particularly the use of trainable segmentation systems and Information Extraction techniques, can be applied to fine-tuning our models for extracting dates from the varied and complex structures present in Dominican legal texts. The project aims to enhance the efficiency of the judicial system by structuring unstructured legal data, making this paper a valuable reference for guiding the development of our own Natural Language Processing (NLP) tools. Additionally, we may be able to use some of this structure to design and define our own paper.

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. Altered the date extraction to provide the model with a different input and see if it led to better results. here.

```python
def extract_dates_from_json(input_folder, output_folder):
    # Create output folder if it doesn't exist
    os.makedirs(output_folder, exist_ok=True)

    # Iterate through each file in the input folder
    for filename in os.listdir(input_folder):
        if filename.endswith(".json"):  # Ensure we're only
         ↪  processing JSON files
            input_file_path = os.path.join(input_folder, filename)
            # Open and read the content of the JSON file
            with open(input_file_path, 'r', encoding='utf-8') as
             ↪  json_file:
                data = json.load(json_file)
            # Extract only the first and second keys for each
             ↪  entry in the list
            extracted_data = []
            for entry in data:
                keys = list(entry.keys())  # Get the list of keys
                 ↪  in order
                if len(keys) >= 2:  # Ensure there are at least
                 ↪  two keys
                    extracted_data.append({
                        "date": entry[keys[1]],   # First key and
                         ↪  its value
                        "date event": entry[keys[3]]            #
                         ↪  Second key and its value
                    })

                        # Define the output file path
            output_file_path = os.path.join(output_folder,
             ↪  filename)
            # Save the extracted data into a new JSON file
            with open(output_file_path, 'w', encoding='utf-8') as
             ↪  output_file:
                json.dump(extracted_data, output_file,
                 ↪  ensure_ascii=False, indent=4)
    print("Extraction complete. Check the output folder:",
     ↪  output_folder)
```

Code 1—Code for normalizing dates files

Below is the flow that would take place for this code:

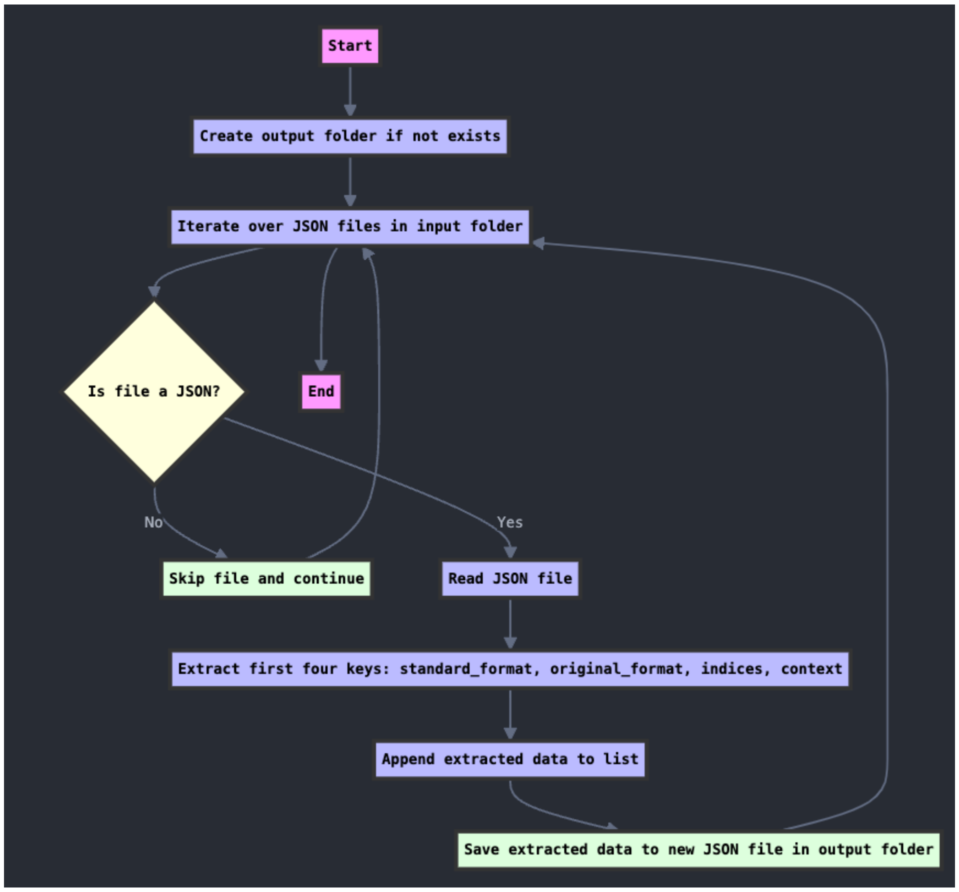

Start
Create output folder if not exists
Iterate over JSON files in input folder
Is file a JSON?
End
No
Yes
Skip file and continue
Read JSON file
Extract first four keys: standard_format, original_format, indices, context
Append extracted data to list
Save extracted data to new JSON file in output folder

Figure 1—Date normalizing process

2. New input template for querying the LLM block in charge of retrieving the context of the identified date here.

```
Analiza el siguiente texto:

{{DOCUMENT_CONTENT}}

Por favor, según la información en el texto, sustituye
↪   "TO_BE_FILLED_IN" con la opción adecuada que represente lo que
↪   indica la fecha y devuelve solo un JSON.
Utilizando solo las siguientes opciones para la respuesta:
{{OPTIONS}}
Importante: Incluye solo el JSON en la respuesta.

Estos son los datos a rellenar:
{{MODEL_OUTPUT_FORMAT}}
```

Code 2—Input template for querying the model, containing placeholders to be replaced

3. In terms of the models, pulled and ran the code using the new Nvidida Nemotron model which seems to have surpassed both ChatGPT-4o and Sonnet 3.5 in its results. Code may be found here.
   Below is the flow that explains the process happening within this code:

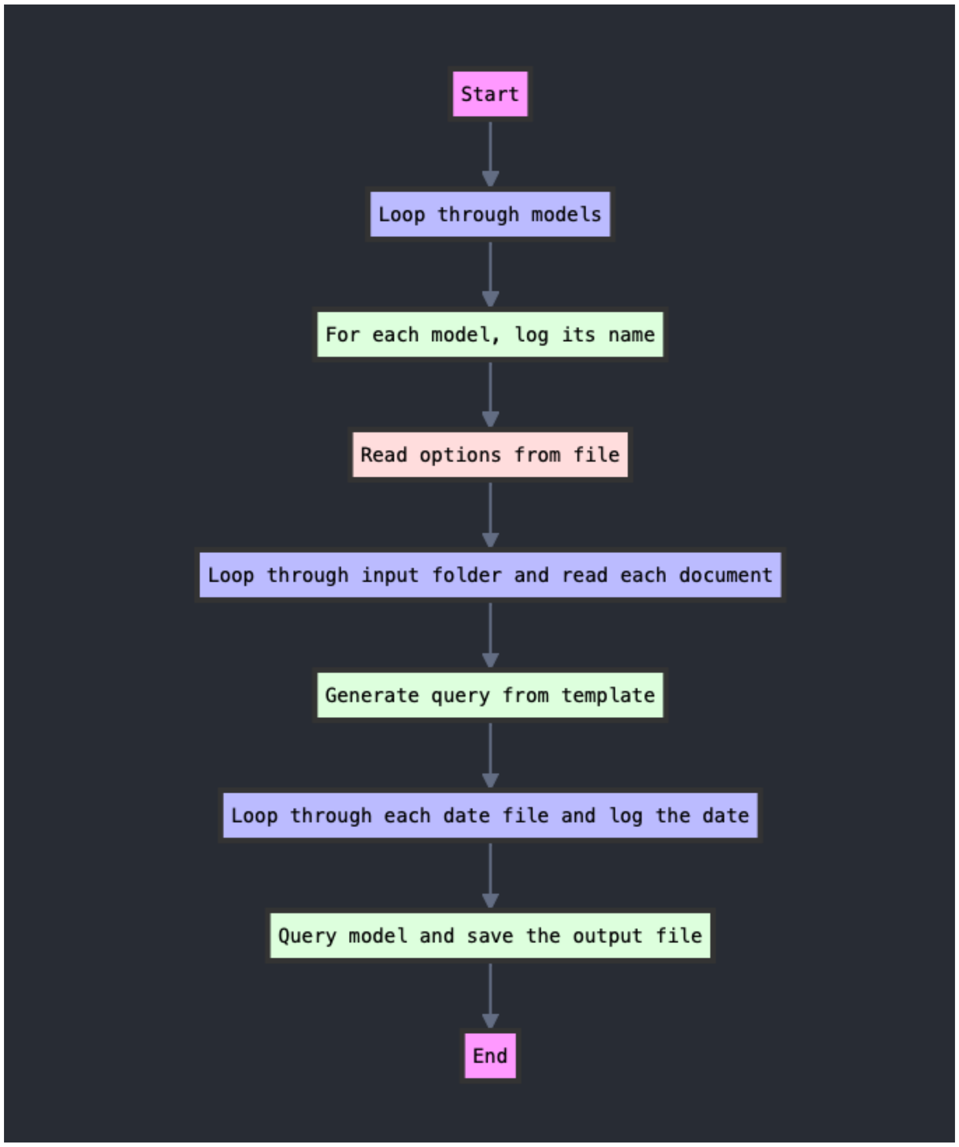

Start
Loop through models
For each model, log its name
Read options from file
Loop through input folder and read each document
Generate query from template
Loop through each date file and log the date
Query model and save the output file
End

Figure 2—Date normalizing process

4 DOCUMENTATION

As previously stated, 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:

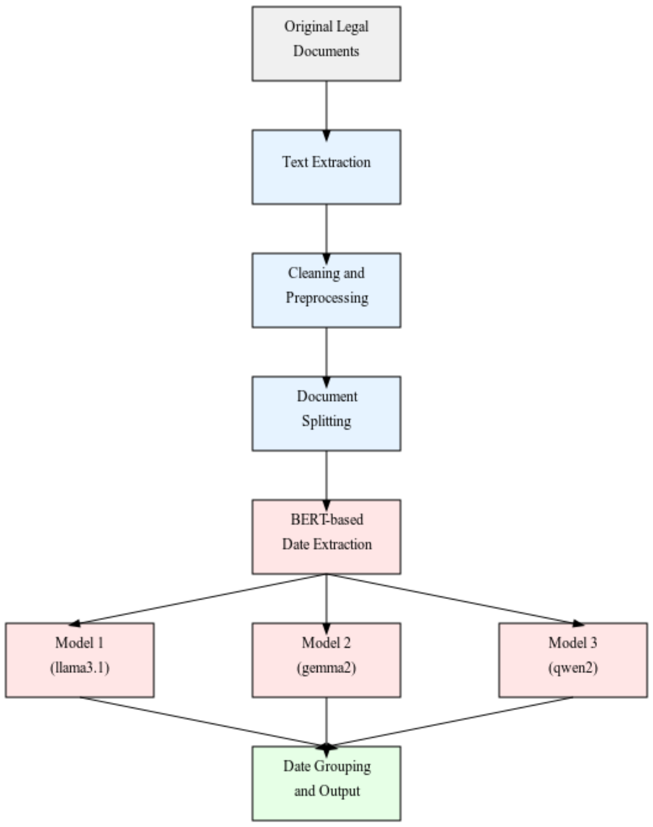


Figure 3—Full date extraction process

This week, similarly to the previous week, my focus has been on the second to last step, using a model to retrieve the context of the date. I've been carrying out this action mainly using the 70B parameters https://huggingface.co/nvidia/Llama-3.1-Nemotron-70B-Instruct-HF. This one seems to have provided better results than the lighter version of Llama 3.2 when running it in PACE.

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 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 179 files generated as training/test data we may use for the different models, extracted from original documents. Below is an example output using Llama-3.1-Nemotron-70B-Instruct-HF with the indicated files.

The model was triggered with the following hyperparameters:

- Temperature = 0.0000001,
- Top_k = 10,
- 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 10 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 file content were generated upon the models results, retrieving the context for the date given as an input to the model.

```
{"standard_format": "2024/01/31", "original_format": "31 de enero
↪   de 2024", "context": "TO_BE_FILLED_IN"}

{"standard_format": "2024/01/31", "original_format": "31 de enero
↪   de 2024", "context": "fecha de fallo reservado"}

{
  "execution_details": {
    "model_name": "nemotron",
    "hyperparameters": {
      "temperature": 1e-07,
      "top_k": 10,
      "top_p": 0.5,
      "seed": 42
    },
    "processing_time": 5.701178550720215,
    "timestamp": "2024-10-17 07:38:17"
  }
}
```

Code 3—Example output retrieved from the model

This output is based on the provided output template where the model informs the field for the date context returning a response that includes both the input date and the identified context for such date.

7 PROOF OF WORK

The implemented system returns in general terms, results that follow the correct structure, although these dates contained in original input are in some small cases being altered in the output.

In this case, a Nemotron 70B model was triggered with 179 files, once for each date contained in the files, with the following hyperparameters:

- Temperature = 0.0000001,
- Top_k = 10,
- Top_p = 0.5
- Seed = 42

The seed and the low temperature should guarantee stable results over multiple executions. With this model, results were far more stable and accurate than with the Llama 3.2 3B model, although there were still some cases were initial input dates were altered. Below is an example where the input date gets modified in the output:

```
{"standard_format": "2015/05/18", "original_format": "18 de mayo
↪   del 2015", "context": "TO_BE_FILLED_IN"}

{"standard_format": "2023/05/26", "original_format": "26 de mayo
↪   de 2023", "context": "fecha de fallo reservado"}

{
  "execution_details": {
    "model_name": "nemotron",
    "hyperparameters": {
      "temperature": 1e-07,
      "top_k": 10,
      "top_p": 0.5,
      "seed": 42
    },
    "processing_time": 5.616535902023315,
    "timestamp": "2024-10-17 07:38:35"
  }
}
```

Code 4—Example output with modified date

All generated files and content may be found here. All documents were generated correctly without any issues in the output generation process. Only matter to highlight is the difference in dates and between consecutive calls.

Once we have a stable process, we'll be able to better assess the model's accuracy. This is due to having "flexible" outputs, where the model can either choose from a range of options or generate a new response. In this last case, it would be very complicated to assess if the generated response is correct, since unless the response is extractive and allows us to use metrics such as ROUGE, there is no clear known way that doesn't involve human feedback, to efficiently validate the generated response. In the case of the other tags, we would need to first generate a set of data large enough that we can then compare the results.

For now, we're focusing on getting it to work with a small manageable sample that we can manually validate. Once this is in place, we'll have to grow the dataset to better extrapolate the results.

8 NEXT WEEK'S PROPOSAL

1. Finalize Berta model training and implementing end to end pipeline.
2. Running full pipeline to generate and evaluate output.
3. Compare outputs from different models to select best performing one.

Week 9 | HAAG - NLP | Fall 2024

Alejandro Gomez

October 18th, 2024

1 Time-log

1.1 What progress did you make in the last week?

- This week I heavily focused on QA and also focused on scaffolding to kickstart our publication. Last week I had finetuned an NER model but it looked to be overfitting so this week, I made sure all the data had the proper shape before training, made sure the data was then labelled properly using the B-I-O-PAD scheme for NER and then made sure the hyperparameters were set properly to avoid overfitting by having the model exit early when training. I also used a dataset aside from the training/validation that is blind to the model and I to see how it performs with the little data I was working with and I was pleasantly surprised, but need to continue iterating.

1.2 What are you planning on working on next?

- I started an abstract and set up the LaTex template for our conference publication target as well as the general scaffolding for the code we will be submitting to the conference so I will be working on iterating this to start building momentum on that front. This will require merging the teams code and refactoring to a clean state the data preprocessing and training pipeline to begin with. Concurrently, I need to gather more data and finetune the model on a larger dataset to see if that can increase the accuracy and confidence score of the NER in its current state.

1.3 Is anything blocking you from getting work done?

N/A

2 Article Review

2.1 Abstract

Legal language is considered to be a key obstacle to the comprehen- sibility of court decisions for laypeople. While differences between written 'standard' and legal language have already been analysed with regard to syntactic peculiarities, there is still a lack of findings on the influence of divergent word meanings on comprehensibility. We present the course and the preliminary results of a study elaborating such ambiguities on the basis of over half a million German court decisions. As these differences are highly language-dependent, our study consequentially relates (only) to German doi[BW23]

2.2 Summary

This article builds on a concept from a previous paper where I learned about context NER since recognizing words is dependent on their context for labeling. This paper discusses the ambiguities in a language outside ouf English (in this case German) and how legalese affects NER. This paper was relevant for this week because it helps me better understand the implications of NER with a language that is not English since our dataset will be in Spanish. It's also highly relevant because we finalized our target conference and this paper is from a past conference so I can begin to understand the general outline of an accepted paper to this AI/CS/Law conference.

3 Scripts and Code Blocks

3.1 Code

```

# sample result

snippet_of_result =     {
        "sentence": "504-2023-SORD-0164 N mero de caso  nico  (NUC): 2022-0150130 En la ciudad de Santo Domingo de Guzm n , Distrito Nacional, capital de la Rep blica Dominicana, a los veinticinco (25) d as del mes de enero del a o dos mil veintitr s (2023); a o ciento setenta y nueve (179) de la Independencia y ciento sesenta (160) de la Restauraci n.",
        "entities": [
            {
                "entity_group": "DATE",
                "score": 0.872123122215271,
                "word": "a los veinticinco (25) d as del mes de enero del a o dos mil veintitr s (2023)",
                "start": 156,
                "end": 234
            },
            {
                "entity_group": "DATE",
                "score": 0.6776415705680847,
                "word": "a o ciento setenta y nueve",
                "start": 236,
                "end": 262
            },
            {
                "entity_group": "DATE",
                "score": 0.5101915001869202,
                "word": "79)",
                "start": 265,
                "end": 268
            }
        ]
    },

"""
```

Listing 1: result

The above is a snippet of the resulting data from the finetuned model after feeding it a ”sentencia” which was a document never seen by the model. With only 180 chunks of data for training, I think the model is on the right track as the results above are precise but not accurate. This is why I will be using more data to see if result improved.

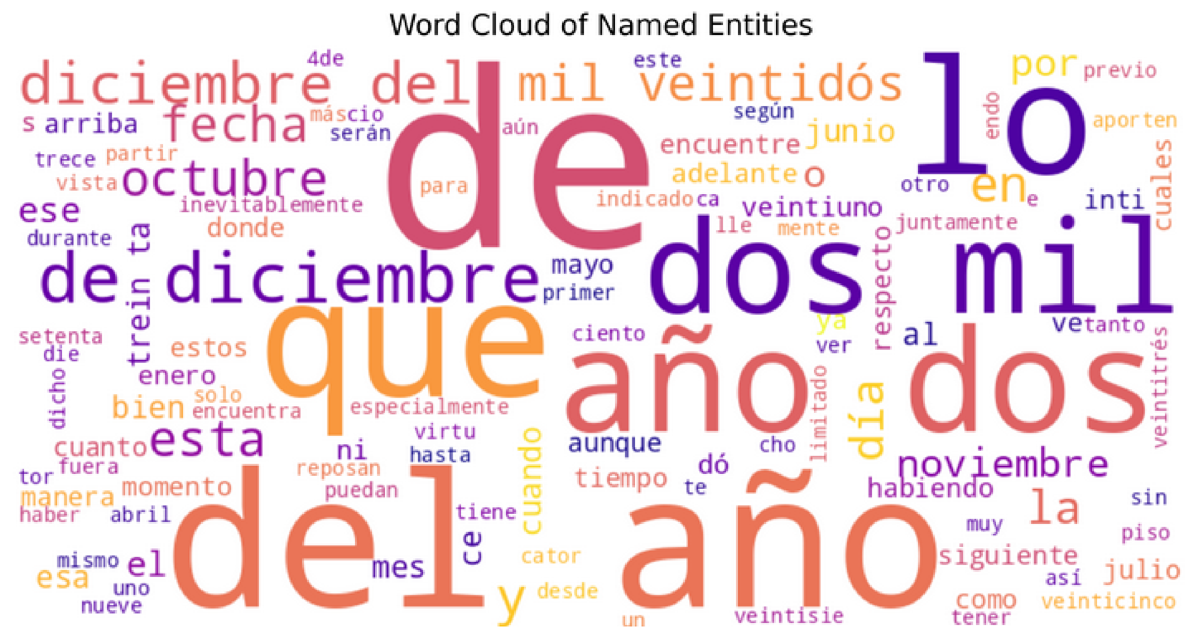


Figure 1: word cloud

This visual shows the makeup of the results so we can see that the items being identified are definitely related to verbal instances of Spanish dates but need to make sure they are accurate.

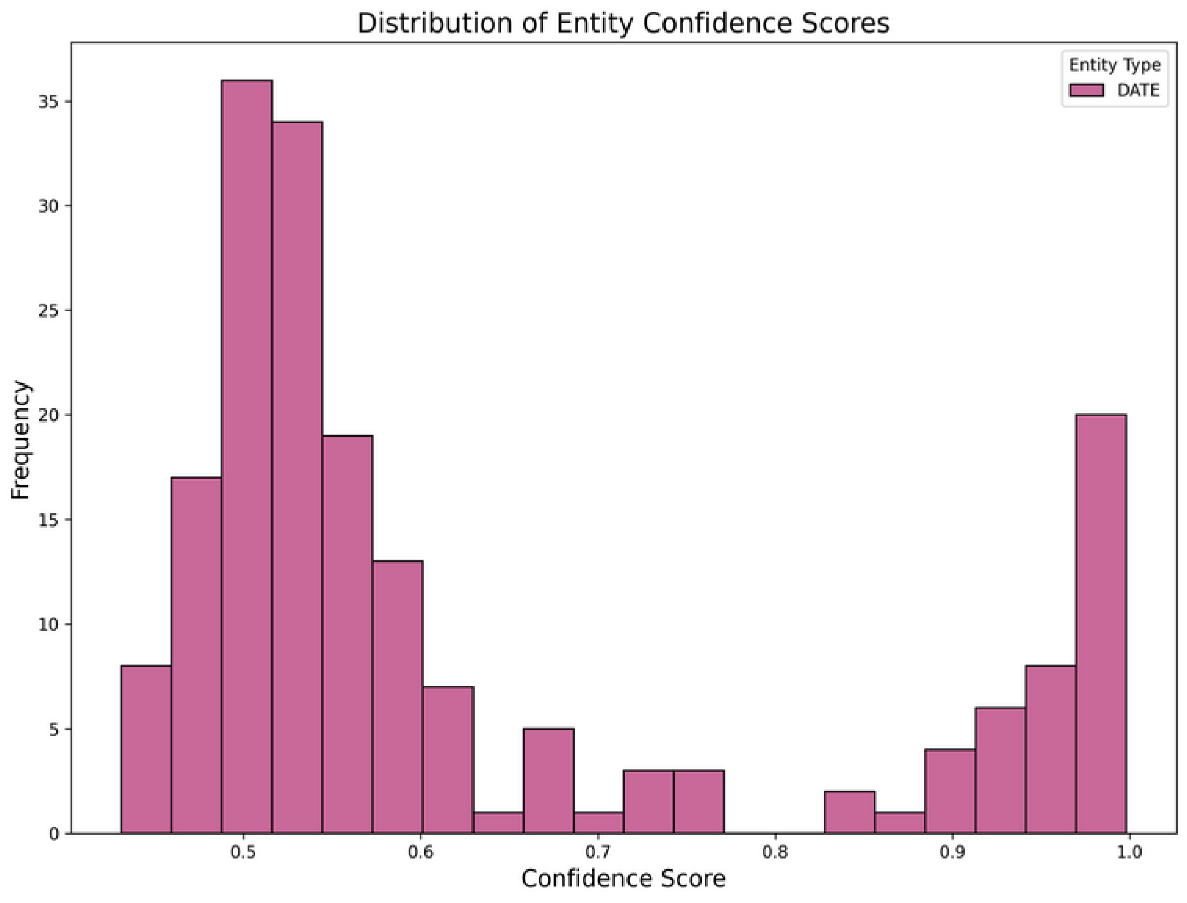


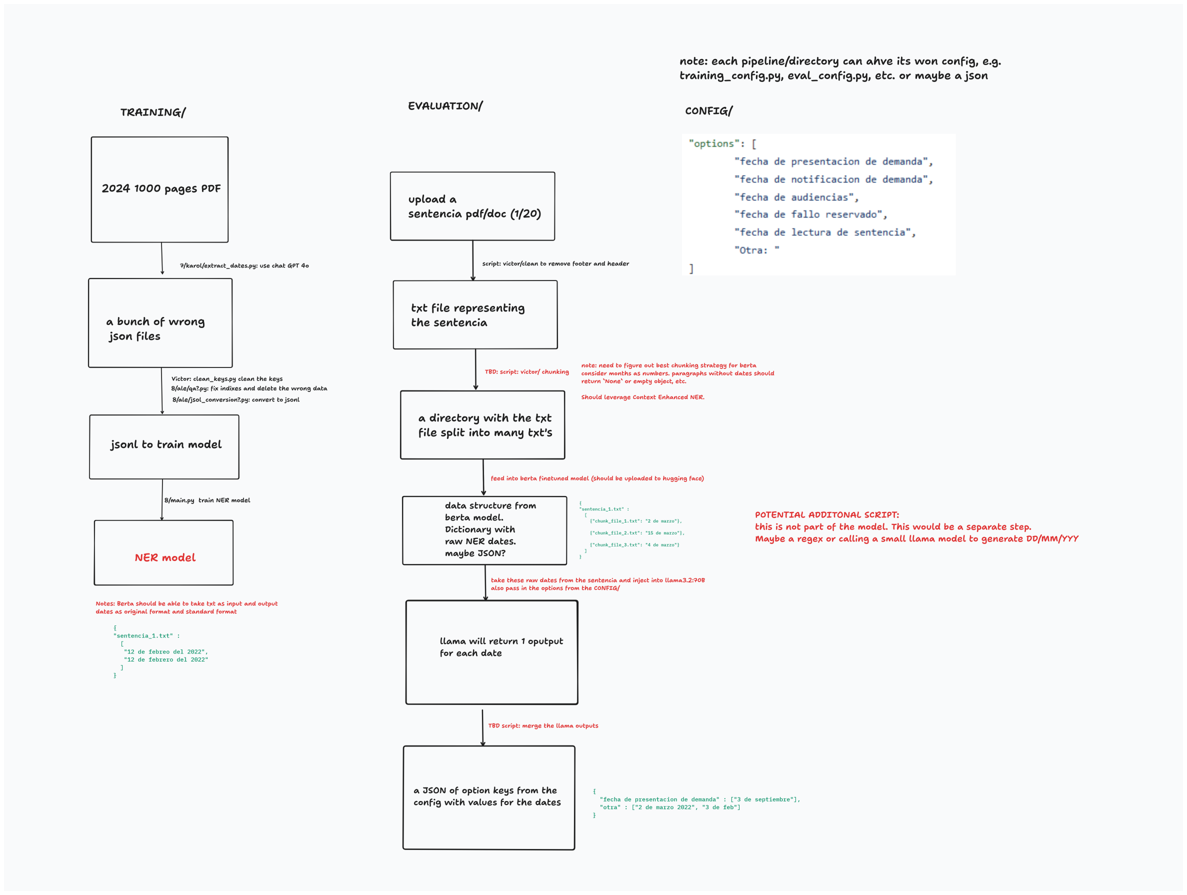
Figure 2: confidence score

This visual shows strong indexing toward weak confidence in the output so this is another factor that leads me to believe that if more data were provided to the model, then it could increase in accuracy and confidence.

3.2 List of Scripts

- The main script that finetunes the existing model and creates a new finetuned model
- script to graph the metrics of finetuned model
- script to test the model with a test sentencia
- The results of the test dataset on the finetuned model
- Script to visualize the results output by the test dataset on the finetuned model
- QA'ing the training dataset
- dataset for training
- QA'ing the data has been labelled in B-I-O-PAD format properly by checking txt files
- an assortment of graphs and visuals in various formats
- HuggingFace NER finetuning example
- HuggingFace NER finetuning simple tutorial with existing dataset

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/
EVALUATION/
CONFIG/
2024 1000 pages PDF
7/karol/extract_dates.py: use chat GPT 4o
a bunch of wrong json files
Victor: clean_keys.py clean the keys
8/ale/qa?.py: fix indixes and delete the wrong data
8/ale/jsol_conversion?.py: convert to jsonl
jsonl to train model
8/main.py train NER model
NER model
Notes: Berta should be able to take txt as input and output dates as original format and standard format
upload a sentencia pdf/doc (1/20)
script: victor/clean to remove footer and header
txt file representing the sentencia
TBD: script: victor/ chunking
note: need to figure out best chunking strategy for berta consider months as numbers. paragraphs without dates should return `None` or empty object, etc.
Should leverage Context Enhanced NER.
a directory with the txt file split into many txt's
feed into berta finetuned model (should be uploaded to hugging face)
data structure from berta model. Dictionary with raw NER dates. maybe JSON?
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
take these raw dates from the sentencia and inject into llama3.2:70B also pass in the options from the CONFIG/
llama will return 1 oputput for each date
TBD script: merge the llama outputs
a JSON of option keys from the config with values for the dates

```
"options": [
        "fecha de presentacion de demanda",
        "fecha de notificacion de demanda",
        "fecha de audiencias",
        "fecha de fallo reservado",
        "fecha de lectura de sentencia",
        "Otra: "
]
```

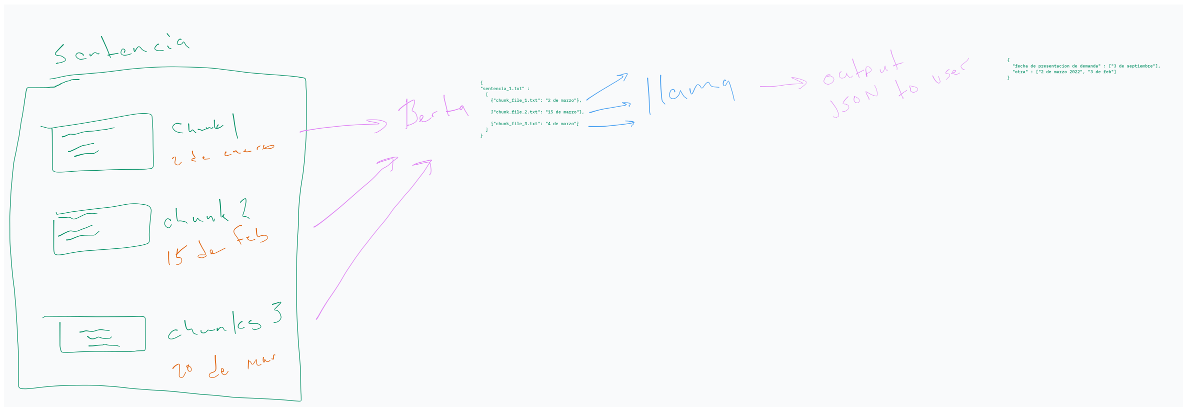
```
{
"sentencia_1.txt" :
  [
   "12 de febreo del 2022",
   "12 de febrero del 2022"
  ]
}
```

```
{
"sentencia_1.txt" :
  [
    {"chunk_file_1.txt": "2 de marzo"},

    {"chunk_file_2.txt": "15 de marzo"},

    {"chunk_file_3.txt": "4 de marzo"}
  ]
}
```

```
{
  "fecha de presentacion de demanda" : ["3 de septiembre"],
  "otra" : ["2 de marzo 2022", "3 de feb"]
}
```

Figure 3: pipeline visualization for code structure

This is how the team and I set up what our code structure would be for our training pipeline and our evaluation pipeline.



```
{
"sentencia_1.txt" :
  [
    {"chunk_file_1.txt": "2 de marzo"},

    {"chunk_file_2.txt": "15 de marzo"},

    {"chunk_file_3.txt": "4 de marzo"}
  ]
}
```

```
{
  "fecha de presentacion de demanda" : ["3 de septiembre"],
  "otra" : ["2 de marzo 2022", "3 de feb"]
}
```

Figure 4: crude pipeline visualization

This crude sketch shows how the model would function at high level from an end-user perspective.

3.4 Script Validation (optional)

N/A

3.5 Results Visualization

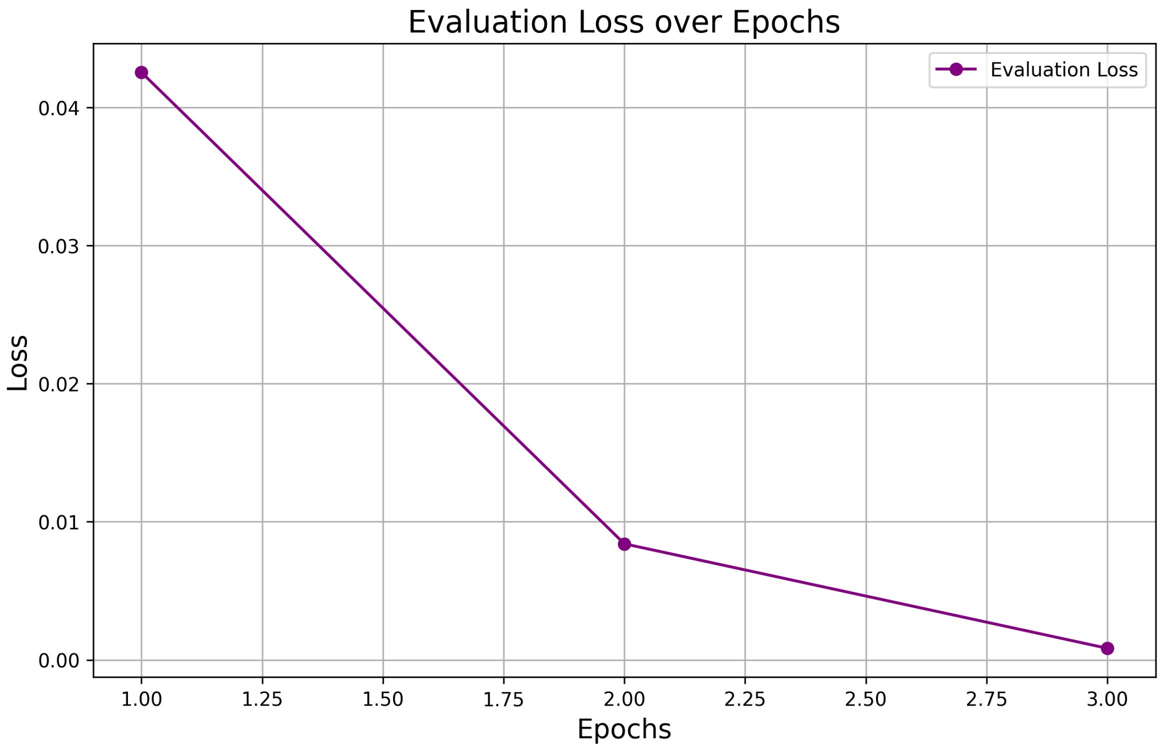


Figure 5: eval loss

This graph demonstrates the same as last week - the model is minimizing the errors over each training epoch - a favorable outcome, but notice the epochs terminate at 3.

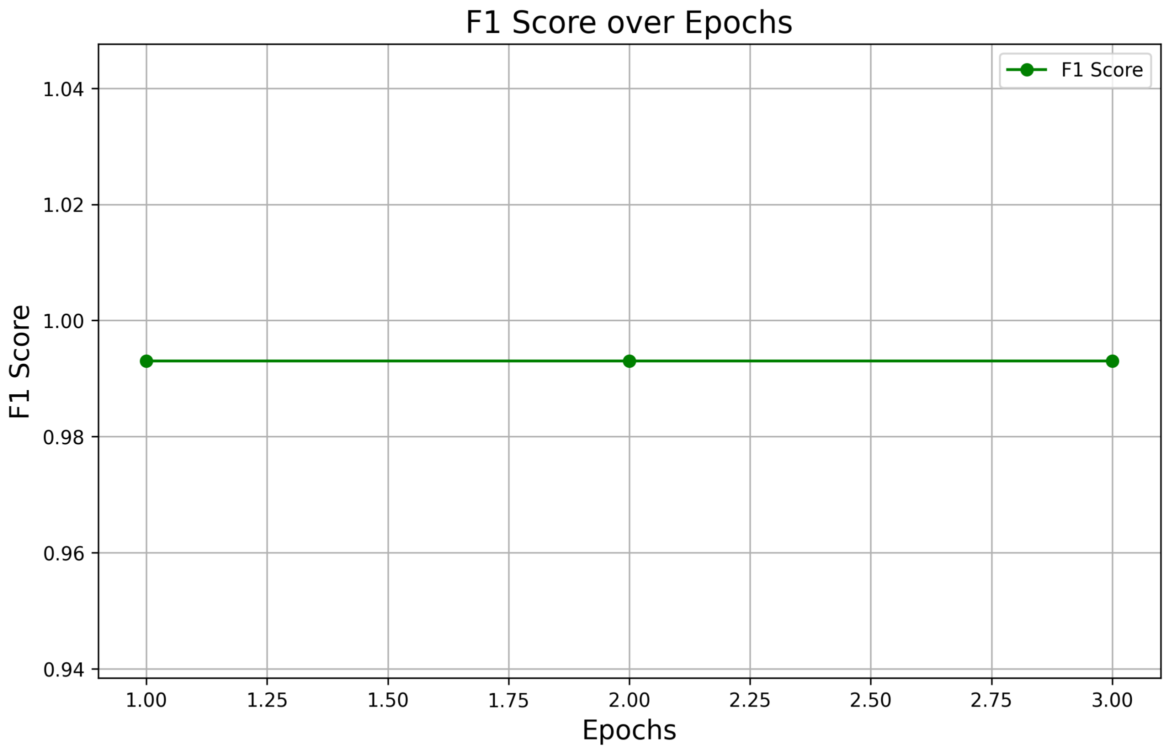


Figure 6: f1 score

This f1 score is suspicious in its constant state. Hopefully the modifcation I make to the fine tuning can give us a shape closer to the graph of y=log(x)

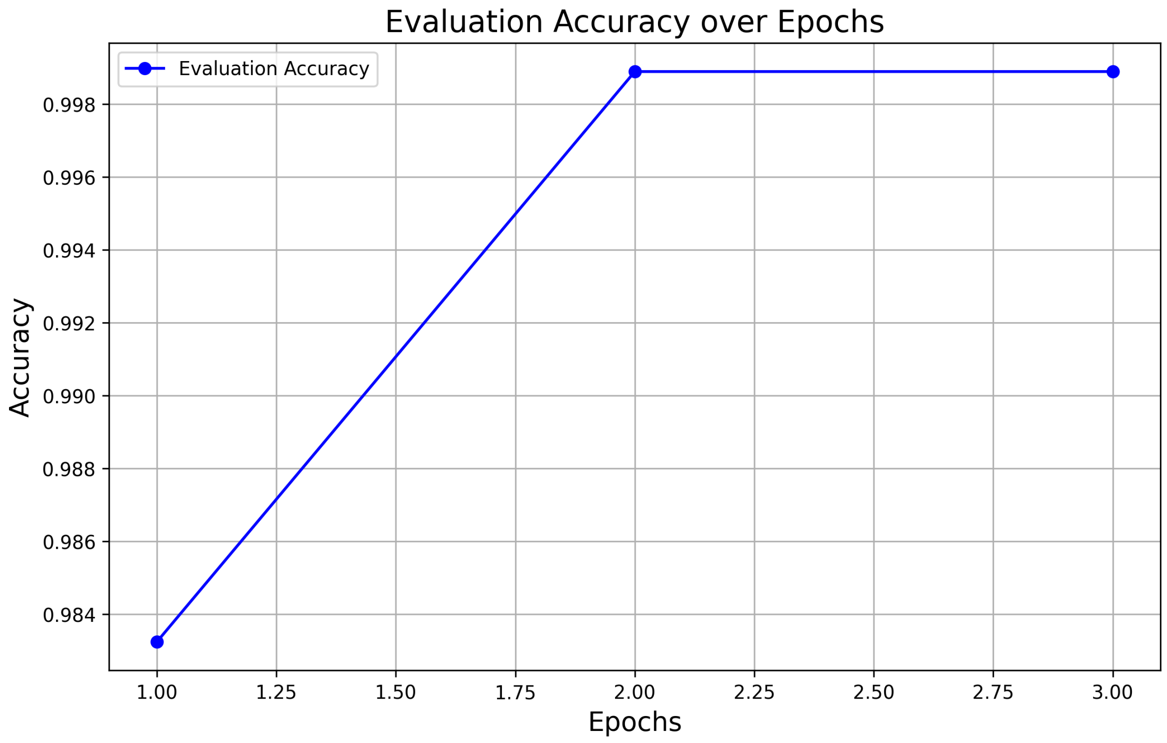


Figure 7: eval accuracy

This graphs shows an increase in accuracy over training epochs and it no longer shows a dip which may signal that the early exit is beneficial for the model training.

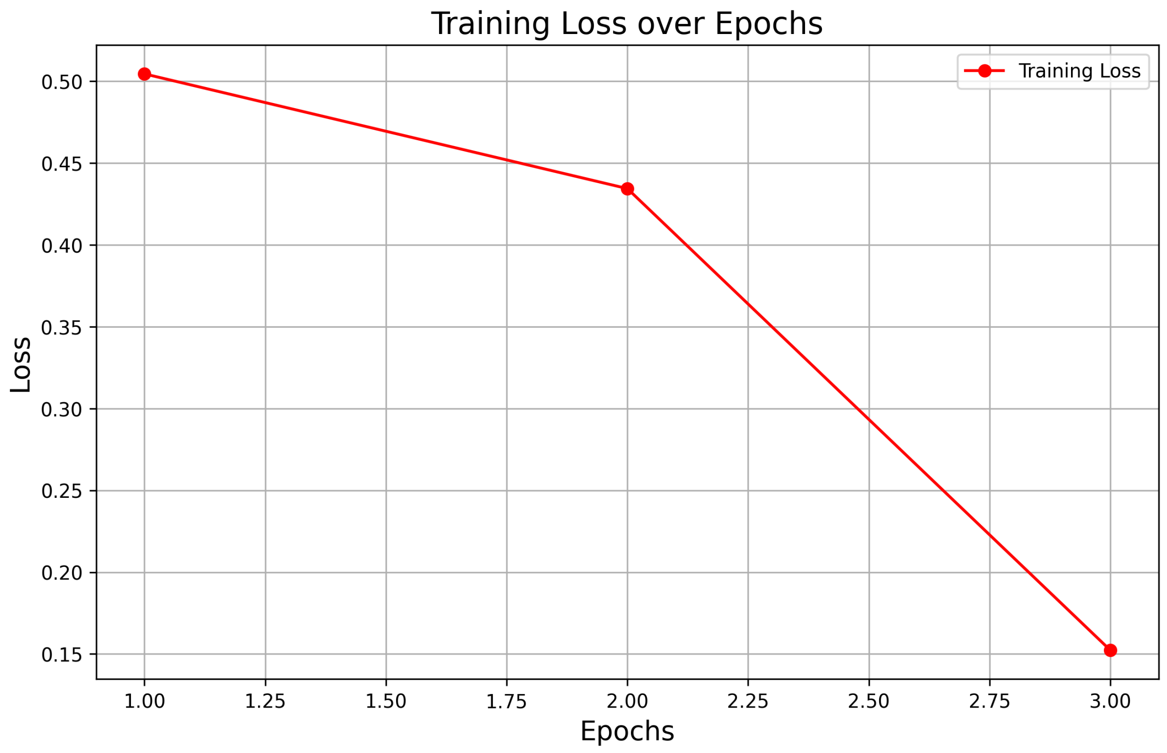


Figure 8: loss

Training loss trends downward which shows the model is reducing its errors - this is expected outcome and I am looking to keeping this shape with the next iteration of the training model.

3.6 Results Visualization Summary

The graph shapes look nearly the same as last week with some minor improvements in shape but the model is still exiting early after 3 epochs when the hyperparameter is set for 20 epochs. More epochs would give me more confidence in the shapes we are seeing.

3.7 Proof of Work

Scripts in GitHub Repo

4 Next Week's Proposal

- The NLP DR team has consistently been meeting during the weekend to review work and prepare. We will do such again this weekend. We also have a presentation with a judge from DR so we will need to discuss how to preapre for that.
- I'll need to continue iterating on our publication's abstract and setting up the formal code we will submit. While doing this, I need to make sure our NER model is precise and accurate for the date recognition, so I will be honing in on this.
- Update current documentation, e.g. NLP website.

References

[BW23] Gregor Behnke and Niklas Wais. On the semantic difference of judicial and standard language. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, ICAIL '23, page 382–386, New York, NY, USA, 2023. Association for Computing Machinery.

HAAG NLP Sentencias — Week 9 Report
NLP-Gen Team

Karol Gutierrez

October 18, 2024

1 Weekly Project Update

1.1 What progress did you make in the last week?

- Scripts to fix training data to make it consistent.
- Script to validate correctness of training data.
- Tutorial for Hugging Face model training using data.
- 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?

- Improve performance of fine tuned model.
- Work on cluster analysis of context for dates.
- 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: Deep Patient: An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records [MLKD16].

2.1 Abstract

BERT has achieved impressive performance in several NLP tasks. However, there has been limited investigation on its adaptation guidelines in specialised domains. Here we focus on the legal domain, where we explore several approaches for applying BERT models to downstream legal tasks, evaluating on multiple datasets. Our findings indicate that the previous guidelines for pre-training and finetuning, often blindly followed, do not always generalize well in the legal domain. Thus we propose a systematic investigation of the available strategies when applying BERT in specialised domains. These are: (a) use the original BERT out of the box, (b) adapt BERT by additional pre-training on domain-specific corpora, and (c) pre-train BERT from scratch on domain-specific corpora. We also propose a broader hyper-parameter search space when fine-tuning for downstream tasks and we release LEGAL-BERT, a family of BERT models intended to assist legal NLP research, computational law, and legal technology applications.

2.2 Summary

The paper presents Deep Patient, an unsupervised deep learning model trained on large-scale electronic health records (EHR) to predict future patient outcomes. The model captures temporal patterns and latent representations of patients' medical histories by using a stacked denoising autoencoder to learn patient representations in an unsupervised manner. These representations are then used for downstream tasks such as predicting future diagnoses.

- Data Handling: The model is trained on EHR data from over 700,000 patients, leveraging unsupervised learning to discover hidden patterns in the data that correlate with future medical conditions.
- Predictive Performance: Deep Patient outperforms traditional models in predicting future diseases, showcasing the strength of deep learning methods in clinical predictions without manual feature engineering.
- Model Generalization: The unsupervised approach allows the model to generalize across various types of diseases and medical conditions, demonstrating its broad application in healthcare prediction tasks.
- Clinical Impact: The model has the potential to improve clinical decision-making by identifying high-risk patients earlier, enabling more proactive medical interventions.

2.3 Relevance

The paper is highly relevant to our Sentencias project as it showcases the utility of unsupervised learning in dealing with complex, unstructured data, which is similar the procedural text found in judicial decisions. As legal documents involve sequences of events or stages, unsupervised learning can help discover patterns that may not be immediately obvious. For us, this approach could inform how court cases are structured and predicted.

3 Scripts and code blocks

The code is in the private repository repository. The progress for this week is in `./karol/week9/` and I also updated the old scripts and data from `./karol/week8/`.

3.1 Code developed

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

- I created a script to fix the incorrect JSON elements, fix the indexes and remove the wrong entries, thus updating the data folder in Figure 2
- Script that validates the integrity of such data.
- Script that converts data into HuggingFace dataset in Figure 3.
- Use BERT Spanish tokenizer with HuggingFace trainer to produce a trained model in Figure 4
- Plot results.

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`.

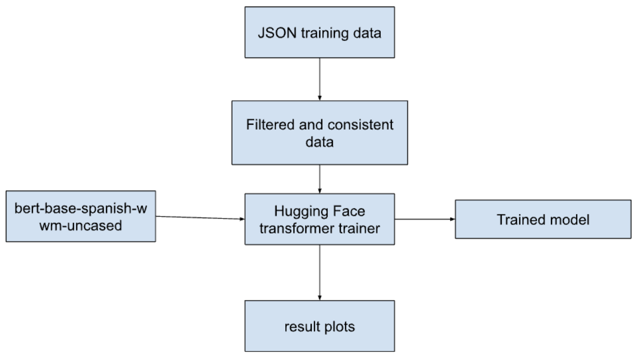

JSON training data
Filtered and consistent data
bert-base-spanish-wwm-uncased
Hugging Face transformer trainer
Trained model
result plots

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

5 Script Validation

Figure 5 shows the validation process for the generated data.

6 Results Visualization

Loss and accuracy plots are generated using the dataset, figures 6 and 7.

7 Proof of Work

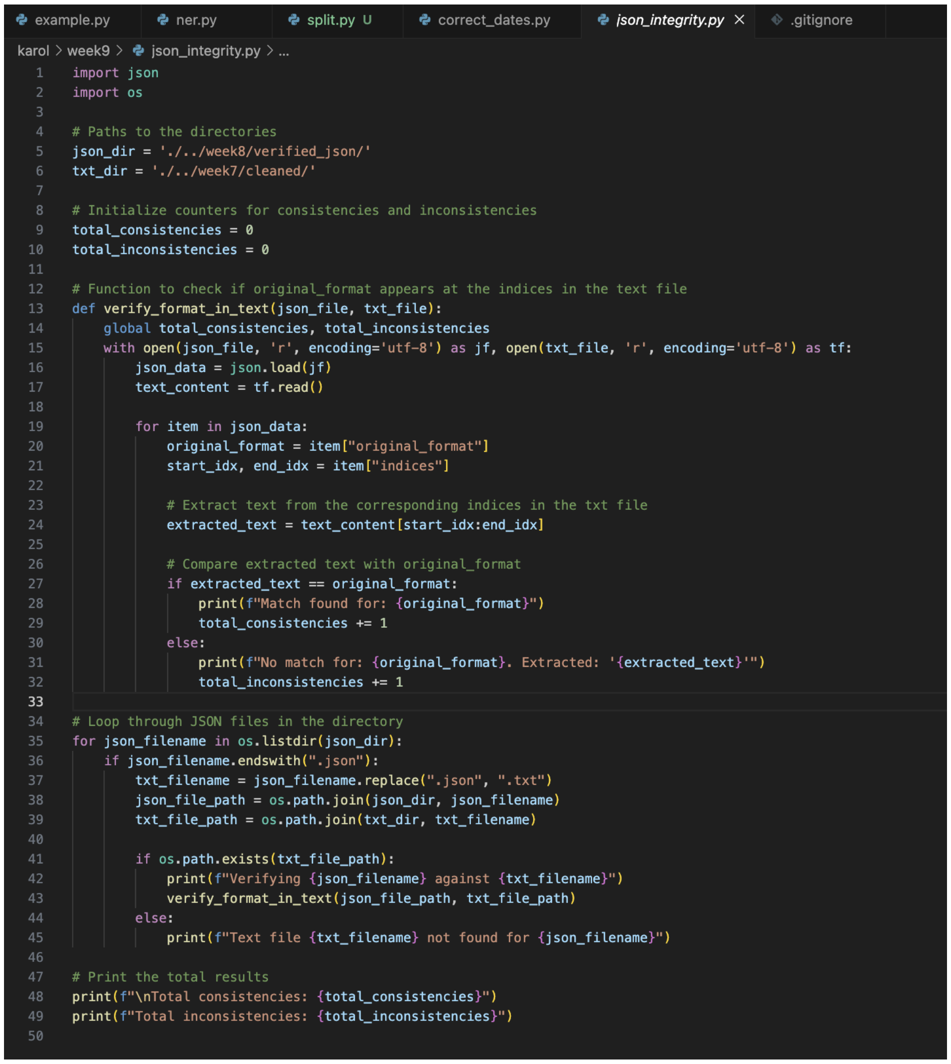
Figures 8 and Figure 9 show the process of generation of data working as well as a sample of the actual data. Figure 10 shows the training process working.

8 Next Week's Proposal

Refer to section 1.2 for details (avoid repetition).

References

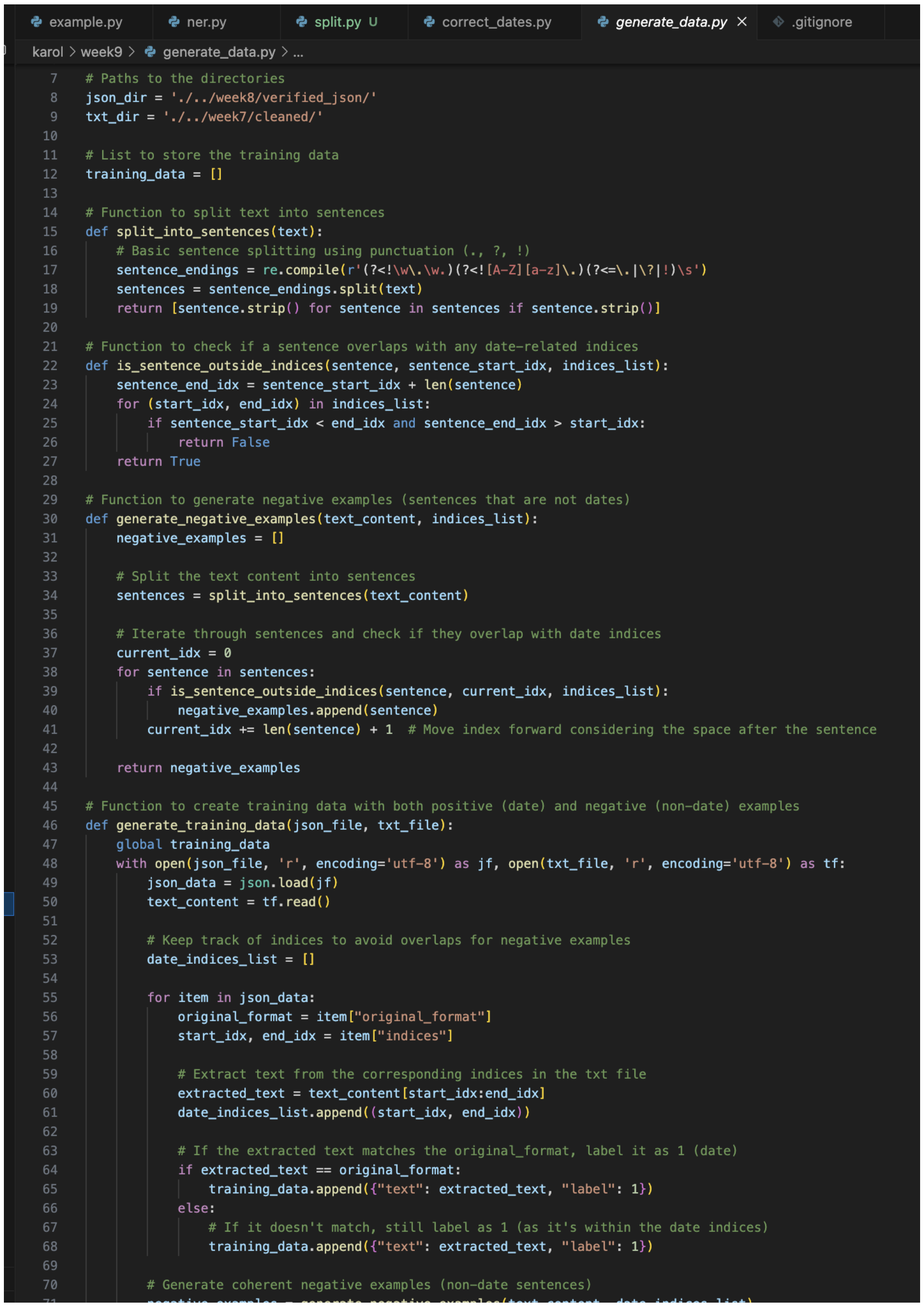
[MLKD16] Riccardo Miotto, Li Li, Brian A Kidd, and Joel T Dudley. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, 6:26094, 2016.


example.py | ner.py | split.py U | correct_dates.py | json_integrity.py × | .gitignore

karol > week9 > json_integrity.py > ...

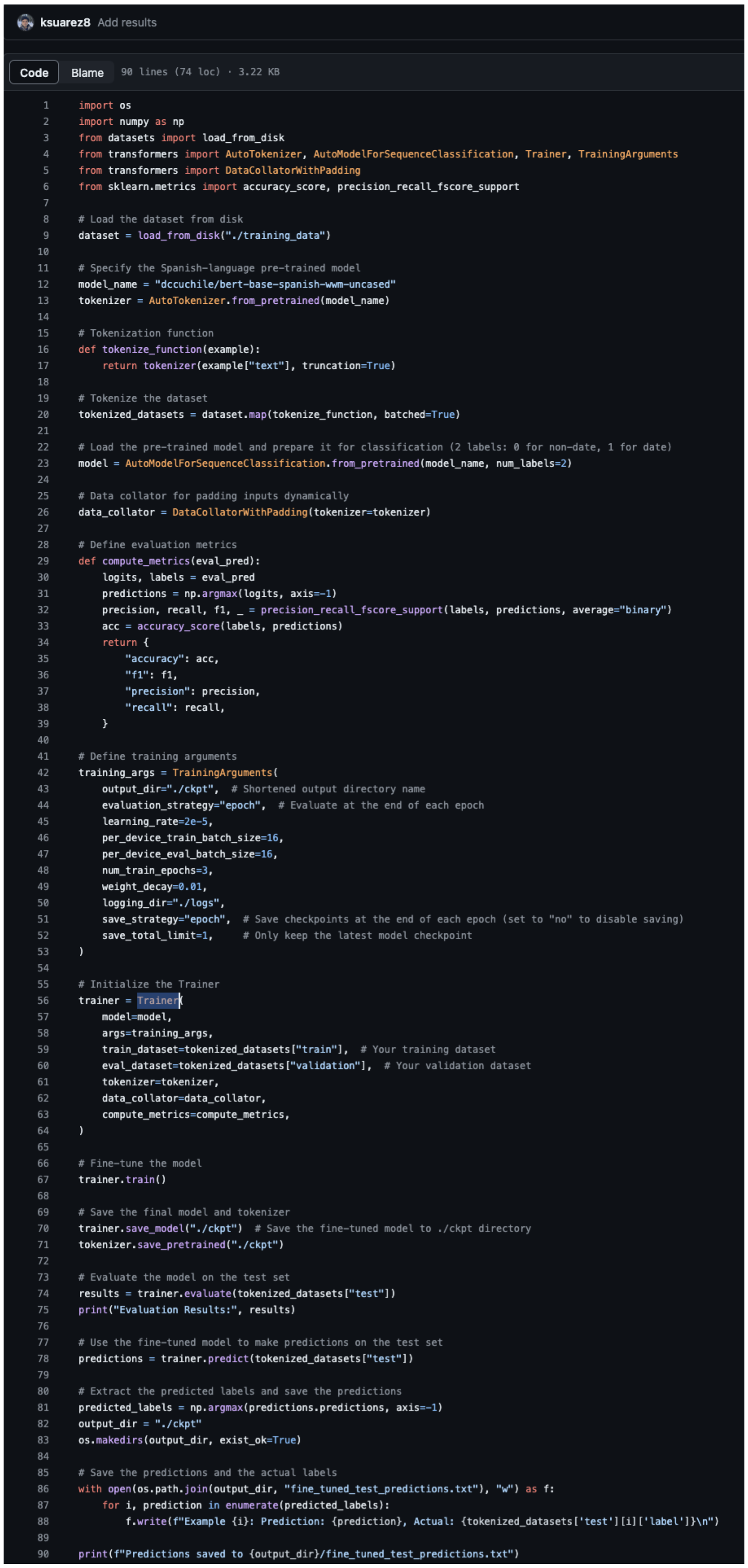
```python
import json
import os

# Paths to the directories
json_dir = './../week8/verified_json/'
txt_dir = './../week7/cleaned/'

# Initialize counters for consistencies and inconsistencies
total_consistencies = 0
total_inconsistencies = 0

# Function to check if original_format appears at the indices in the text file
def verify_format_in_text(json_file, txt_file):
    global total_consistencies, total_inconsistencies
    with open(json_file, 'r', encoding='utf-8') as jf, open(txt_file, 'r', encoding='utf-8') as tf:
        json_data = json.load(jf)
        text_content = tf.read()

        for item in json_data:
            original_format = item["original_format"]
            start_idx, end_idx = item["indices"]

            # Extract text from the corresponding indices in the txt file
            extracted_text = text_content[start_idx:end_idx]

            # Compare extracted text with original_format
            if extracted_text == original_format:
                print(f"Match found for: {original_format}")
                total_consistencies += 1
            else:
                print(f"No match for: {original_format}. Extracted: '{extracted_text}'")
                total_inconsistencies += 1

# Loop through JSON files in the directory
for json_filename in os.listdir(json_dir):
    if json_filename.endswith(".json"):
        txt_filename = json_filename.replace(".json", ".txt")
        json_file_path = os.path.join(json_dir, json_filename)
        txt_file_path = os.path.join(txt_dir, txt_filename)

        if os.path.exists(txt_file_path):
            print(f"Verifying {json_filename} against {txt_filename}")
            verify_format_in_text(json_file_path, txt_file_path)
        else:
            print(f"Text file {txt_filename} not found for {json_filename}")

# Print the total results
print(f"\nTotal consistencies: {total_consistencies}")
print(f"Total inconsistencies: {total_inconsistencies}")

```

Figure 2: Code to validate integrity of filtered data



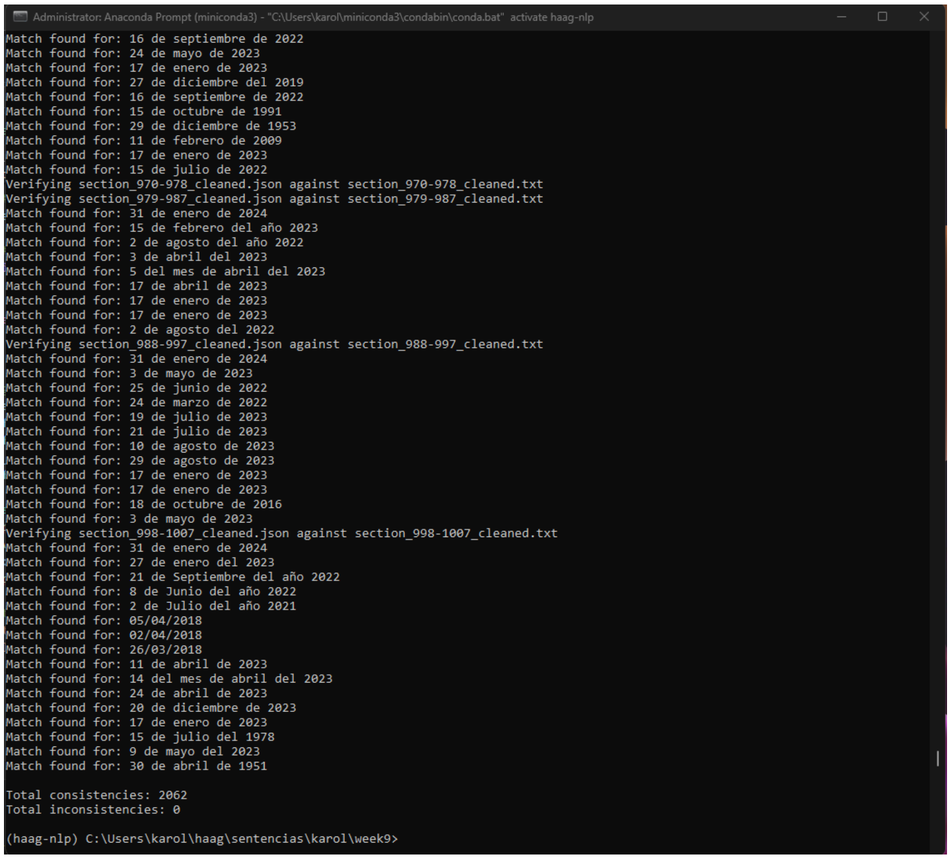
```python
# Paths to the directories
json_dir = './../week8/verified_json/'
txt_dir = './../week7/cleaned/'

# List to store the training data
training_data = []

# Function to split text into sentences
def split_into_sentences(text):
    # Basic sentence splitting using punctuation (., ?, !)
    sentence_endings = re.compile(r'(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?|!)\s')
    sentences = sentence_endings.split(text)
    return [sentence.strip() for sentence in sentences if sentence.strip()]

# Function to check if a sentence overlaps with any date-related indices
def is_sentence_outside_indices(sentence, sentence_start_idx, indices_list):
    sentence_end_idx = sentence_start_idx + len(sentence)
    for (start_idx, end_idx) in indices_list:
        if sentence_start_idx < end_idx and sentence_end_idx > start_idx:
            return False
    return True

# Function to generate negative examples (sentences that are not dates)
def generate_negative_examples(text_content, indices_list):
    negative_examples = []

    # Split the text content into sentences
    sentences = split_into_sentences(text_content)

    # Iterate through sentences and check if they overlap with date indices
    current_idx = 0
    for sentence in sentences:
        if is_sentence_outside_indices(sentence, current_idx, indices_list):
            negative_examples.append(sentence)
        current_idx += len(sentence) + 1  # Move index forward considering the space after the sentence

    return negative_examples

# Function to create training data with both positive (date) and negative (non-date) examples
def generate_training_data(json_file, txt_file):
    global training_data
    with open(json_file, 'r', encoding='utf-8') as jf, open(txt_file, 'r', encoding='utf-8') as tf:
        json_data = json.load(jf)
        text_content = tf.read()

        # Keep track of indices to avoid overlaps for negative examples
        date_indices_list = []

        for item in json_data:
            original_format = item["original_format"]
            start_idx, end_idx = item["indices"]

            # Extract text from the corresponding indices in the txt file
            extracted_text = text_content[start_idx:end_idx]
            date_indices_list.append((start_idx, end_idx))

            # If the extracted text matches the original_format, label it as 1 (date)
            if extracted_text == original_format:
                training_data.append({"text": extracted_text, "label": 1})
            else:
                # If it doesn't match, still label as 1 (as it's within the date indices)
                training_data.append({"text": extracted_text, "label": 1})

        # Generate coherent negative examples (non-date sentences)
        negative_examples = generate_negative_examples(text_content, date_indices_list)
```

Figure 3: Code for generation of dataset



```python
import os
import numpy as np
from datasets import load_from_disk
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments
from transformers import DataCollatorWithPadding
from sklearn.metrics import accuracy_score, precision_recall_fscore_support

# Load the dataset from disk
dataset = load_from_disk("./training_data")

# Specify the Spanish-language pre-trained model
model_name = "dccuchile/bert-base-spanish-wwm-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)

# Tokenization function
def tokenize_function(example):
    return tokenizer(example["text"], truncation=True)

# Tokenize the dataset
tokenized_datasets = dataset.map(tokenize_function, batched=True)

# Load the pre-trained model and prepare it for classification (2 labels: 0 for non-date, 1 for date)
model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=2)

# Data collator for padding inputs dynamically
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

# Define evaluation metrics
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    precision, recall, f1, _ = precision_recall_fscore_support(labels, predictions, average="binary")
    acc = accuracy_score(labels, predictions)
    return {
        "accuracy": acc,
        "f1": f1,
        "precision": precision,
        "recall": recall,
    }

# Define training arguments
training_args = TrainingArguments(
    output_dir="./ckpt",  # Shortened output directory name
    evaluation_strategy="epoch",  # Evaluate at the end of each epoch
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
    logging_dir="./logs",
    save_strategy="epoch",  # Save checkpoints at the end of each epoch (set to "no" to disable saving)
    save_total_limit=1,     # Only keep the latest model checkpoint
)

# Initialize the Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],  # Your training dataset
    eval_dataset=tokenized_datasets["validation"],  # Your validation dataset
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)

# Fine-tune the model
trainer.train()

# Save the final model and tokenizer
trainer.save_model("./ckpt")  # Save the fine-tuned model to ./ckpt directory
tokenizer.save_pretrained("./ckpt")

# Evaluate the model on the test set
results = trainer.evaluate(tokenized_datasets["test"])
print("Evaluation Results:", results)

# Use the fine-tuned model to make predictions on the test set
predictions = trainer.predict(tokenized_datasets["test"])

# Extract the predicted labels and save the predictions
predicted_labels = np.argmax(predictions.predictions, axis=-1)
output_dir = "./ckpt"
os.makedirs(output_dir, exist_ok=True)

# Save the predictions and the actual labels
with open(os.path.join(output_dir, "fine_tuned_test_predictions.txt"), "w") as f:
    for i, prediction in enumerate(predicted_labels):
        f.write(f"Example {i}: Prediction: {prediction}, Actual: {tokenized_datasets['test'][i]['label']}\n")

print(f"Predictions saved to {output_dir}/fine_tuned_test_predictions.txt")
```

Figure 4: Model training


Administrator: Anaconda Prompt (miniconda3) - "C:\Users\karol\miniconda3\condabin\conda.bat" activate haag-nlp

```
Match found for: 16 de septiembre de 2022
Match found for: 24 de mayo de 2023
Match found for: 17 de enero de 2023
Match found for: 27 de diciembre del 2019
Match found for: 16 de septiembre de 2022
Match found for: 15 de octubre de 1991
Match found for: 29 de diciembre de 1953
Match found for: 11 de febrero de 2009
Match found for: 17 de enero de 2023
Match found for: 15 de julio de 2022
Verifying section_970-978_cleaned.json against section_970-978_cleaned.txt
Verifying section_979-987_cleaned.json against section_979-987_cleaned.txt
Match found for: 31 de enero de 2024
Match found for: 15 de febrero del año 2023
Match found for: 2 de agosto del año 2022
Match found for: 3 de abril del 2023
Match found for: 5 del mes de abril del 2023
Match found for: 17 de abril de 2023
Match found for: 17 de enero de 2023
Match found for: 17 de enero de 2023
Match found for: 2 de agosto del 2022
Verifying section_988-997_cleaned.json against section_988-997_cleaned.txt
Match found for: 31 de enero de 2024
Match found for: 3 de mayo de 2023
Match found for: 25 de junio de 2022
Match found for: 24 de marzo de 2022
Match found for: 19 de julio de 2023
Match found for: 21 de julio de 2023
Match found for: 10 de agosto de 2023
Match found for: 29 de agosto de 2023
Match found for: 17 de enero de 2023
Match found for: 17 de enero de 2023
Match found for: 18 de octubre de 2016
Match found for: 3 de mayo de 2023
Verifying section_998-1007_cleaned.json against section_998-1007_cleaned.txt
Match found for: 31 de enero de 2024
Match found for: 27 de enero del 2023
Match found for: 21 de Septiembre del año 2022
Match found for: 8 de Junio del año 2022
Match found for: 2 de Julio del año 2021
Match found for: 05/04/2018
Match found for: 02/04/2018
Match found for: 26/03/2018
Match found for: 11 de abril de 2023
Match found for: 14 del mes de abril del 2023
Match found for: 24 de abril de 2023
Match found for: 20 de diciembre de 2023
Match found for: 17 de enero de 2023
Match found for: 15 de julio del 1978
Match found for: 9 de mayo del 2023
Match found for: 30 de abril de 1951

Total consistencies: 2062
Total inconsistencies: 0

(haag-nlp) C:\Users\karol\haag\sentencias\karol\week9>
```

Figure 5: Proof of work for integrity testing

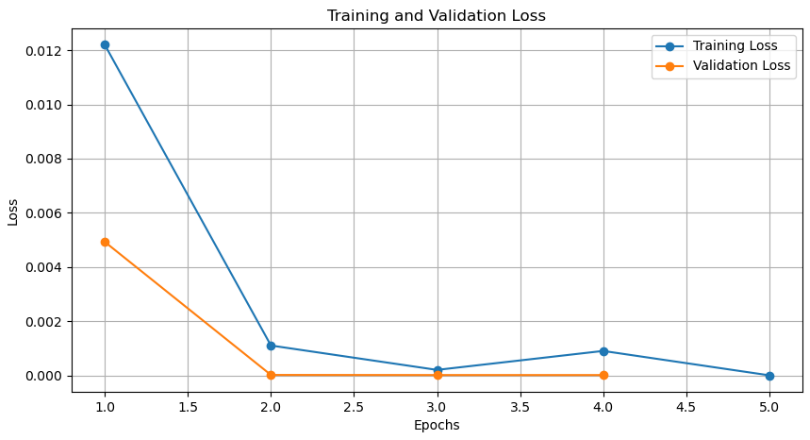


Figure 6: Loss during training

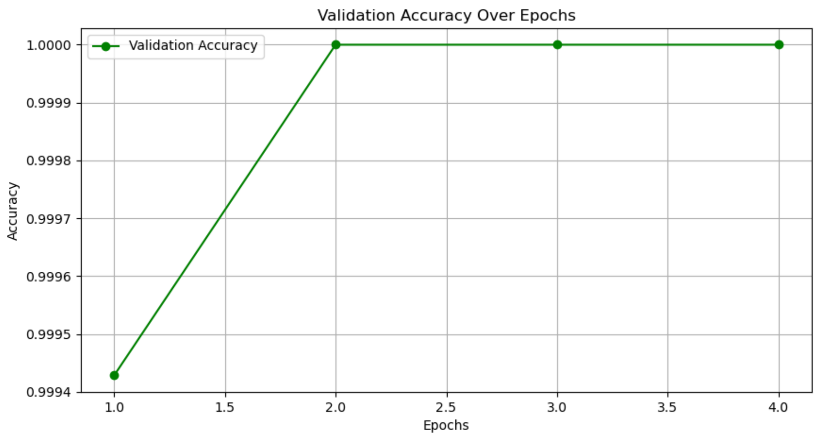
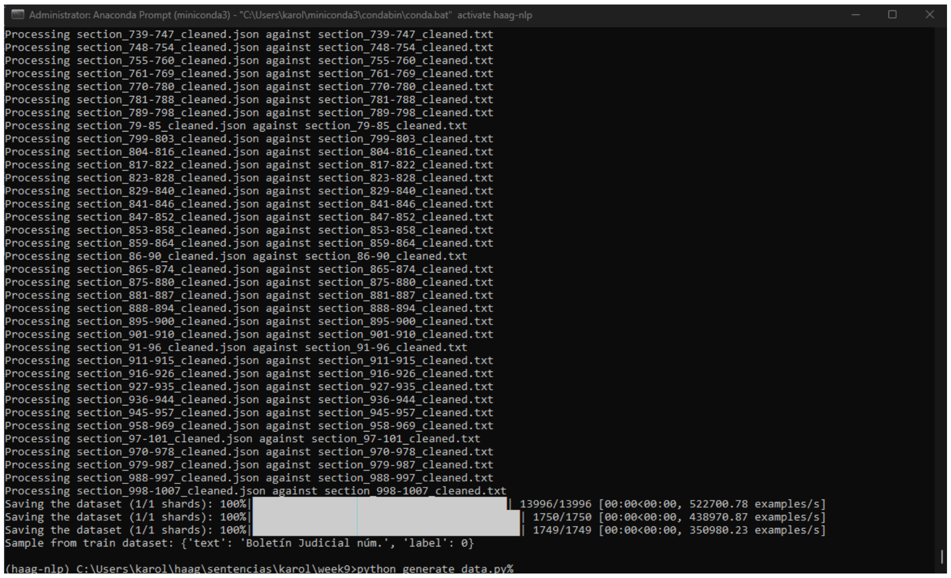
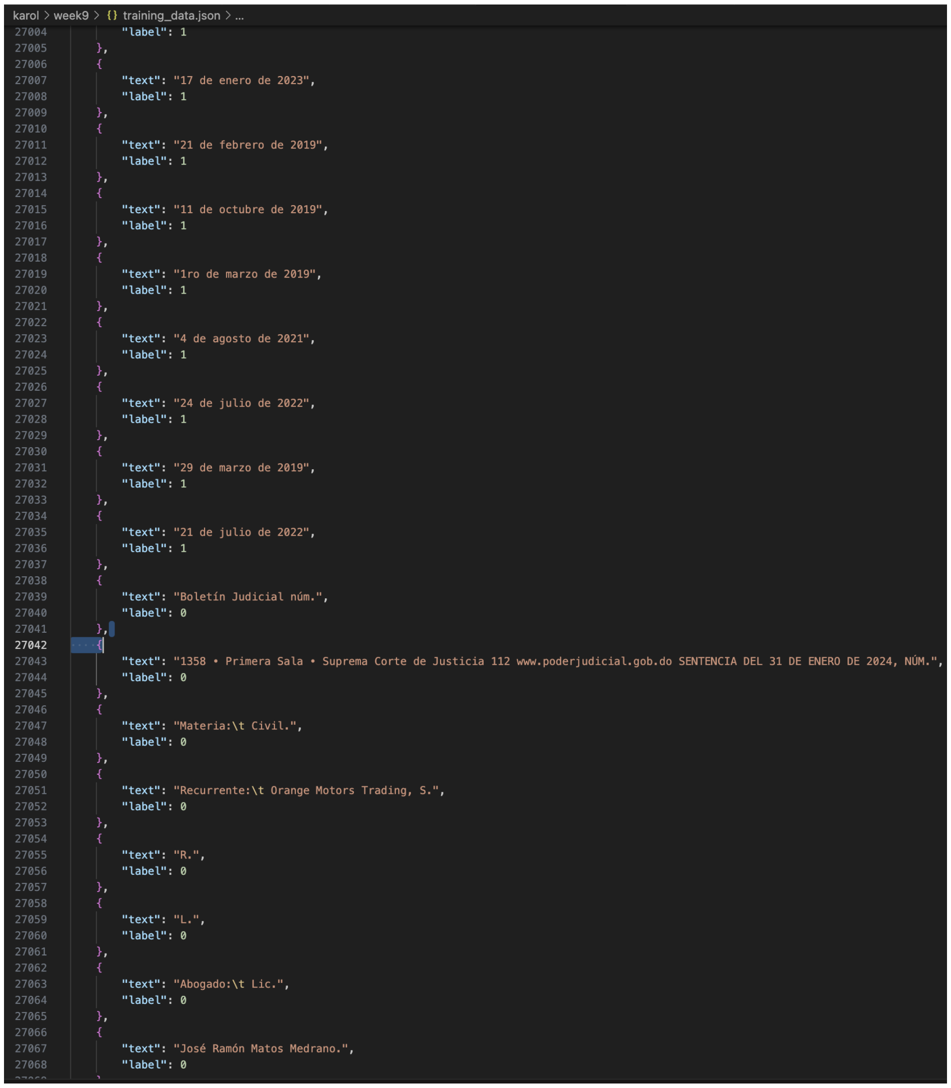


Figure 7: Accuracy during training



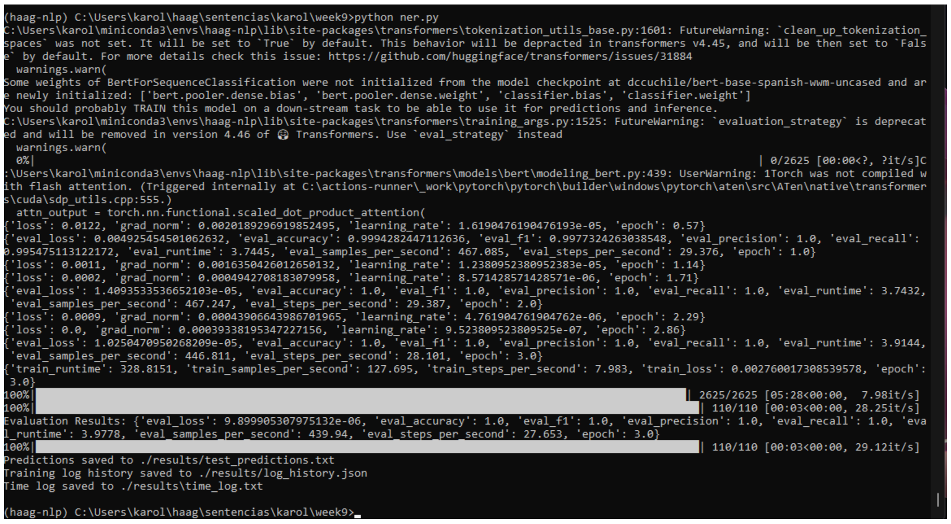
```
Administrator: Anaconda Prompt (miniconda3) - "C:\Users\karol\miniconda3\condabin\conda.bat"  activate haag-nlp
Processing section_739-747_cleaned.json against section_739-747_cleaned.txt
Processing section_748-754_cleaned.json against section_748-754_cleaned.txt
Processing section_755-760_cleaned.json against section_755-760_cleaned.txt
Processing section_761-769_cleaned.json against section_761-769_cleaned.txt
Processing section_770-780_cleaned.json against section_770-780_cleaned.txt
Processing section_781-788_cleaned.json against section_781-788_cleaned.txt
Processing section_789-798_cleaned.json against section_789-798_cleaned.txt
Processing section_79-85_cleaned.json against section_79-85_cleaned.txt
Processing section_799-803_cleaned.json against section_799-803_cleaned.txt
Processing section_804-816_cleaned.json against section_804-816_cleaned.txt
Processing section_817-822_cleaned.json against section_817-822_cleaned.txt
Processing section_823-828_cleaned.json against section_823-828_cleaned.txt
Processing section_829-840_cleaned.json against section_829-840_cleaned.txt
Processing section_841-846_cleaned.json against section_841-846_cleaned.txt
Processing section_847-852_cleaned.json against section_847-852_cleaned.txt
Processing section_853-858_cleaned.json against section_853-858_cleaned.txt
Processing section_859-864_cleaned.json against section_859-864_cleaned.txt
Processing section_86-90_cleaned.json against section_86-90_cleaned.txt
Processing section_865-874_cleaned.json against section_865-874_cleaned.txt
Processing section_875-880_cleaned.json against section_875-880_cleaned.txt
Processing section_881-887_cleaned.json against section_881-887_cleaned.txt
Processing section_888-894_cleaned.json against section_888-894_cleaned.txt
Processing section_895-900_cleaned.json against section_895-900_cleaned.txt
Processing section_901-910_cleaned.json against section_901-910_cleaned.txt
Processing section_91-96_cleaned.json against section_91-96_cleaned.txt
Processing section_911-915_cleaned.json against section_911-915_cleaned.txt
Processing section_916-926_cleaned.json against section_916-926_cleaned.txt
Processing section_927-935_cleaned.json against section_927-935_cleaned.txt
Processing section_936-944_cleaned.json against section_936-944_cleaned.txt
Processing section_945-957_cleaned.json against section_945-957_cleaned.txt
Processing section_958-969_cleaned.json against section_958-969_cleaned.txt
Processing section_97-101_cleaned.json against section_97-101_cleaned.txt
Processing section_970-978_cleaned.json against section_970-978_cleaned.txt
Processing section_979-987_cleaned.json against section_979-987_cleaned.txt
Processing section_988-997_cleaned.json against section_988-997_cleaned.txt
Processing section_998-1007_cleaned.json against section_998-1007_cleaned.txt
Saving the dataset (1/1 shards): 100%|████████████████████| 13996/13996 [00:00<00:00, 522700.78 examples/s]
Saving the dataset (1/1 shards): 100%|████████████████████| 1750/1750 [00:00<00:00, 438970.87 examples/s]
Saving the dataset (1/1 shards): 100%|████████████████████| 1749/1749 [00:00<00:00, 350980.23 examples/s]
Sample from train dataset: {'text': 'Boletín Judicial núm.', 'label': 0}

(haag-nlp) C:\Users\karol\haag\sentencias\karol\week9>python generate_data.py%
```

Figure 8: Proof of generation of dataset



```
karol > week9 > {} training_data.json > ...
        "label": 1
    },
    {
        "text": "17 de enero de 2023",
        "label": 1
    },
    {
        "text": "21 de febrero de 2019",
        "label": 1
    },
    {
        "text": "11 de octubre de 2019",
        "label": 1
    },
    {
        "text": "1ro de marzo de 2019",
        "label": 1
    },
    {
        "text": "4 de agosto de 2021",
        "label": 1
    },
    {
        "text": "24 de julio de 2022",
        "label": 1
    },
    {
        "text": "29 de marzo de 2019",
        "label": 1
    },
    {
        "text": "21 de julio de 2022",
        "label": 1
    },
    {
        "text": "Boletín Judicial núm.",
        "label": 0
    },
    {
        "text": "1358 • Primera Sala • Suprema Corte de Justicia 112 www.poderjudicial.gob.do SENTENCIA DEL 31 DE ENERO DE 2024, NÚM.",
        "label": 0
    },
    {
        "text": "Materia:\t Civil.",
        "label": 0
    },
    {
        "text": "Recurrente:\t Orange Motors Trading, S.",
        "label": 0
    },
    {
        "text": "R.",
        "label": 0
    },
    {
        "text": "L.",
        "label": 0
    },
    {
        "text": "Abogado:\t Lic.",
        "label": 0
    },
    {
        "text": "José Ramón Matos Medrano.",
        "label": 0
```

Figure 9: Generated data format



```
(haag-nlp) C:\Users\karol\haag\sentencias\karol\week9>python ner.py
C:\Users\karol\miniconda3\envs\haag-nlp\lib\site-packages\transformers\tokenization_utils_base.py:1601: FutureWarning: `clean_up_tokenization_
spaces` was not set. It will be set to `True` by default. This behavior will be depracted in transformers v4.45, and will be then set to `Fals
e` by default. For more details check this issue: https://github.com/huggingface/transformers/issues/31884
  warnings.warn(
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at dccuchile/bert-base-spanish-wwm-uncased and ar
e newly initialized: ['bert.pooler.dense.bias', 'bert.pooler.dense.weight', 'classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
C:\Users\karol\miniconda3\envs\haag-nlp\lib\site-packages\transformers\training_args.py:1525: FutureWarning: `evaluation_strategy` is deprecat
ed and will be removed in version 4.46 of 🤗 Transformers. Use `eval_strategy` instead
  warnings.warn(
  0%|                                                                                                 | 0/2625 [00:00<?, ?it/s]C
:\Users\karol\miniconda3\envs\haag-nlp\lib\site-packages\transformers\models\bert\modeling_bert.py:439: UserWarning: 1Torch was not compiled w
ith flash attention. (Triggered internally at C:\actions-runner\_work\pytorch\pytorch\builder\windows\pytorch\aten\src\ATen\native\transformer
s\cuda\sdp_utils.cpp:555.)
  attn_output = torch.nn.functional.scaled_dot_product_attention(
{'loss': 0.0122, 'grad_norm': 0.0020189296919852495, 'learning_rate': 1.6190476190476193e-05, 'epoch': 0.57}
{'eval_loss': 0.004925454501062632, 'eval_accuracy': 0.9994282447112636, 'eval_f1': 0.9977324263038548, 'eval_precision': 1.0, 'eval_recall':
0.995475113122172, 'eval_runtime': 3.7445, 'eval_samples_per_second': 467.085, 'eval_steps_per_second': 29.376, 'epoch': 1.0}
{'loss': 0.0011, 'grad_norm': 0.0016350426012650132, 'learning_rate': 1.2380952380952383e-05, 'epoch': 1.14}
{'loss': 0.0002, 'grad_norm': 0.0004942708183079958, 'learning_rate': 8.571428571428571e-06, 'epoch': 1.71}
{'eval_loss': 1.4093533536652103e-05, 'eval_accuracy': 1.0, 'eval_f1': 1.0, 'eval_precision': 1.0, 'eval_recall': 1.0, 'eval_runtime': 3.7432,
 'eval_samples_per_second': 467.247, 'eval_steps_per_second': 29.387, 'epoch': 2.0}
{'loss': 0.0009, 'grad_norm': 0.00043906643986701965, 'learning_rate': 4.761904761904762e-06, 'epoch': 2.29}
{'loss': 0.0, 'grad_norm': 0.00039338195347227156, 'learning_rate': 9.523809523809525e-07, 'epoch': 2.86}
{'eval_loss': 1.0250470950268209e-05, 'eval_accuracy': 1.0, 'eval_f1': 1.0, 'eval_precision': 1.0, 'eval_recall': 1.0, 'eval_runtime': 3.9144,
 'eval_samples_per_second': 446.811, 'eval_steps_per_second': 28.101, 'epoch': 3.0}
{'train_runtime': 328.8151, 'train_samples_per_second': 127.695, 'train_steps_per_second': 7.983, 'train_loss': 0.002760017308539578, 'epoch':
 3.0}
100%|████████████████████████████████████████████████████████████████████████████████████████████| 2625/2625 [05:28<00:00,  7.98it/s]
100%|██████████████████████████████████████████████████████████████████████████████████████████████| 110/110 [00:03<00:00, 28.25it/s]
Evaluation Results: {'eval_loss': 9.899905307975132e-06, 'eval_accuracy': 1.0, 'eval_f1': 1.0, 'eval_precision': 1.0, 'eval_recall': 1.0, 'eva
l_runtime': 3.9778, 'eval_samples_per_second': 439.94, 'eval_steps_per_second': 27.653, 'epoch': 3.0}
100%|██████████████████████████████████████████████████████████████████████████████████████████████| 110/110 [00:03<00:00, 29.12it/s]
Predictions saved to ./results/test_predictions.txt
Training log history saved to ./results/log_history.json
Time log saved to ./results\time_log.txt

(haag-nlp) C:\Users\karol\haag\sentencias\karol\week9>_
```

Figure 10: Proof of work for training process

Week 9 Research Report

Thomas Orth (NLP Summarization / NLP Gen Team)

October 2024

0.1 What did you work on this week?

1. Received validation from interview team, serving as expert reviewers. Llama 3.2 approach outperformed baseline models.
2. Implemented chunking for Summary Chain-of-Thought to experiment with summary quality
3. Implemented Domain Summary Chain-of-Thought to grab more elements of good legal summaries
4. Transitioned to running with Anthropic models, starting with Haiku
5. Generated initial Haiku summaries, performing better than llama 3.2 3b
6. Started investigating Sonnet but API was having issues last night
7. Met with Thuan to get started with Summarization project
8. Reviewed Thuan feedback on Haiku summaries, iterate on the prompts to improve

0.2 What are you planning on working on next?

1. Complete Sonnet tests
2. Meet during all hands next week for next steps with summarization work
3. Begin looking into finetuning options

0.3 Is anything blocking you from getting work done?

1. None

1 Abstracts

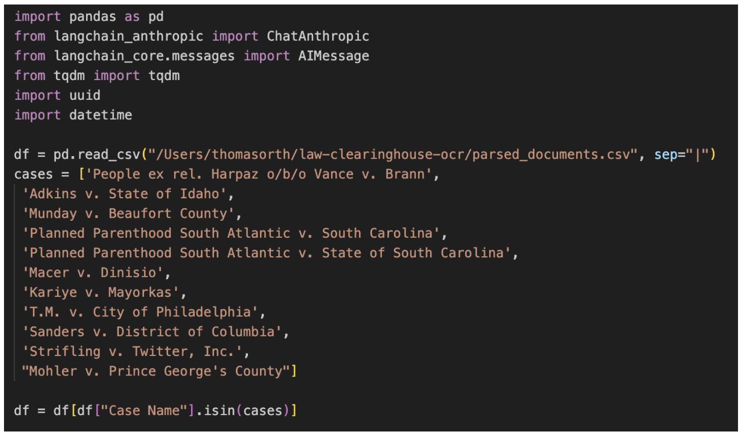
- Title: Little Giants: Exploring the Potential of Small LLMs as Evaluation Metrics in Summarization in the Eval4NLP 2023 Shared Task. Conference / Venue: ACL 2023, Proceedings of the 4th Workshop on Evaluation and Comparison of NLP Systems. Link: https://aclanthology.org/2023.eval4nlp-1.17.pdf
- Abstract: This paper describes and analyzes our participation in the 2023 Eval4NLP shared task, which focuses on assessing the effectiveness of prompt-based techniques to empower Large Language Models to handle the task of quality estimation, particularly in the context of evaluating machine translations and summaries. We conducted systematic experiments with various prompting techniques, including standard prompting, prompts informed by annotator instructions, and innovative chain-of-thought prompting. In addition, we integrated these approaches with zero-shot and one-shot learning methods to maximize the efficacy of our evaluation procedures. Our work reveals that combining these approaches using a "small", open source model (orca_mini_v3_7B) yields competitive results.
- Summary: This paper made a case for using a small language model vs a prohibitively large LLM to evaluate quality. This would allow for researchers with resource constraints to perform SOTA quality estimation.
- 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.
- The two best models in terms of accuracy and cost tradeoff is Claude 3.5 Sonnet and Claude 3 Haiku

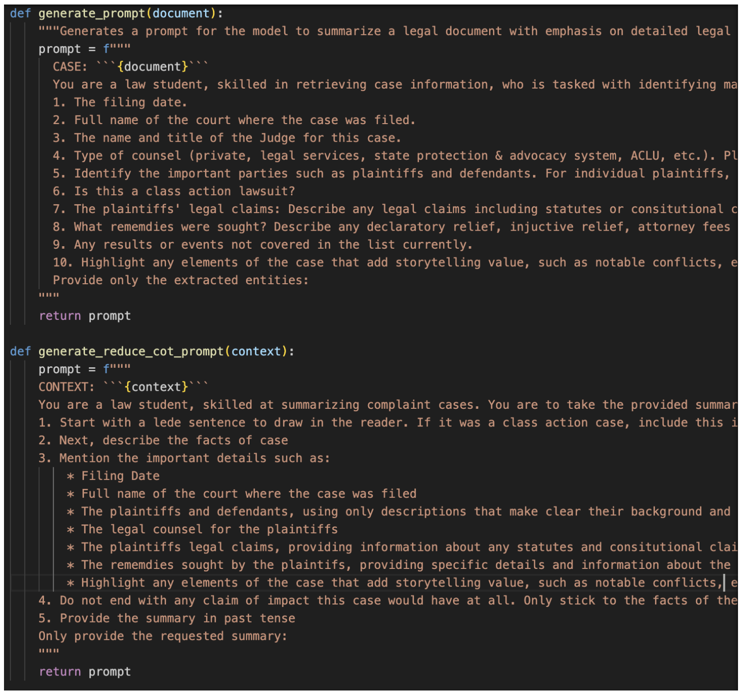
3 Scripts

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/anthropic/domain_specific_scot.py
    - Brief Description: Run a domain specific version of Summary Chain-of-thought (CoT) on complaints with Anthropic models.
    - 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, Save summaries
        (c) Third Block: Evaluate via manual inspection
    - Screenshot of code:



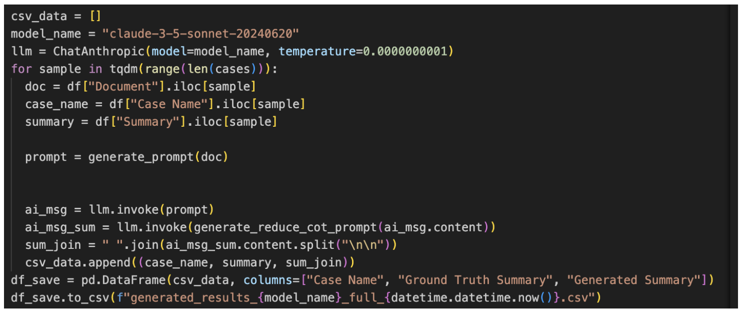
```
import pandas as pd
from langchain_anthropic import ChatAnthropic
from langchain_core.messages import AIMessage
from tqdm import tqdm
import uuid
import datetime

df = pd.read_csv("/Users/thomasorth/law-clearinghouse-ocr/parsed_documents.csv", sep="|")
cases = ['People ex rel. Harpaz o/b/o Vance v. Brann',
 'Adkins v. State of Idaho',
 'Munday v. Beaufort County',
 'Planned Parenthood South Atlantic v. South Carolina',
 'Planned Parenthood South Atlantic v. State of South Carolina',
 'Macer v. Dinisio',
 'Kariye v. Mayorkas',
 'T.M. v. City of Philadelphia',
 'Sanders v. District of Columbia',
 'Strifling v. Twitter, Inc.',
 "Mohler v. Prince George's County"]

df = df[df["Case Name"].isin(cases)]
```

Figure 1: Domain Summary CoT Anthropic Part 1



```
def generate_prompt(document):
    """Generates a prompt for the model to summarize a legal document with emphasis on detailed legal
    prompt = f"""
      CASE: ```{document}```
      You are a law student, skilled in retrieving case information, who is tasked with identifying ma
      1. The filing date.
      2. Full name of the court where the case was filed.
      3. The name and title of the Judge for this case.
      4. Type of counsel (private, legal services, state protection & advocacy system, ACLU, etc.). Pl
      5. Identify the important parties such as plaintiffs and defendants. For individual plaintiffs,
      6. Is this a class action lawsuit?
      7. The plaintiffs' legal claims: Describe any legal claims including statutes or consitutional c
      8. What rememdies were sought? Describe any declaratory relief, injuctive relief, attorney fees
      9. Any results or events not covered in the list currently.
      10. Highlight any elements of the case that add storytelling value, such as notable conflicts, e
      Provide only the extracted entities:
    """
    return prompt

def generate_reduce_cot_prompt(context):
    prompt = f"""
    CONTEXT: ```{context}```
    You are a law student, skilled at summarizing complaint cases. You are to take the provided summar
    1. Start with a lede sentence to draw in the reader. If it was a class action case, include this i
    2. Next, describe the facts of case
    3. Mention the important details such as:
        * Filing Date
        * Full name of the court where the case was filed
        * The plaintiffs and defendants, using only descriptions that make clear their background and
        * The legal counsel for the plaintiffs
        * The plaintiffs legal claims, providing information about any statutes and consitutional clai
        * The rememdies sought by the plaintifs, providing specific details and information about the
        * Highlight any elements of the case that add storytelling value, such as notable conflicts, e
    4. Do not end with any claim of impact this case would have at all. Only stick to the facts of the
    5. Provide the summary in past tense
    Only provide the requested summary:
    """
    return prompt
```

Figure 2: Domain Summary CoT Anthropic Part 2



```
csv_data = []
model_name = "claude-3-5-sonnet-20240620"
llm = ChatAnthropic(model=model_name, temperature=0.0000000001)
for sample in tqdm(range(len(cases))):
  doc = df["Document"].iloc[sample]
  case_name = df["Case Name"].iloc[sample]
  summary = df["Summary"].iloc[sample]

  prompt = generate_prompt(doc)


  ai_msg = llm.invoke(prompt)
  ai_msg_sum = llm.invoke(generate_reduce_cot_prompt(ai_msg.content))
  sum_join = " ".join(ai_msg_sum.content.split("\n\n"))
  csv_data.append((case_name, summary, sum_join))
df_save = pd.DataFrame(csv_data, columns=["Case Name", "Ground Truth Summary", "Generated Summary"])
df_save.to_csv(f"generated_results_{model_name}_full_{datetime.datetime.now()}.csv")
```

Figure 3: Domain Summary CoT Anthropic Part e

4. Thomas-Orth/anthropic/domain_specific_scot_chunked.py

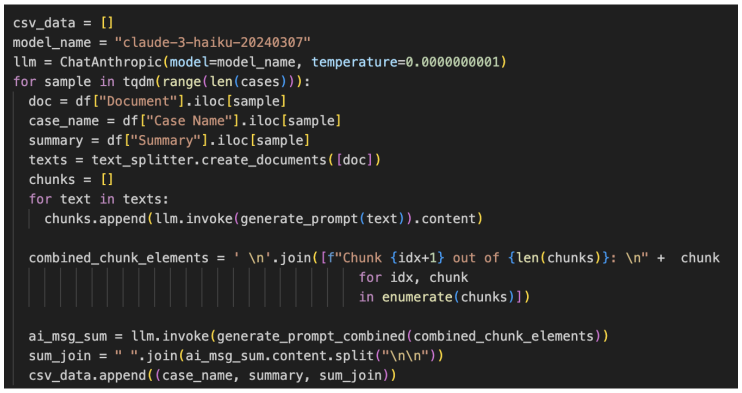
   - Brief Description: Run a domain specific version of Summary Chain-of-thought (CoT) on complaints with Anthropic models.
   - Status: Tested by running the pipeline to completion without issue
   - Important Code Blocks:

     (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 (showing only the chunking part that is what is different between this and the last code file):



```python
csv_data = []
model_name = "claude-3-haiku-20240307"
llm = ChatAnthropic(model=model_name, temperature=0.0000000001)
for sample in tqdm(range(len(cases))):
  doc = df["Document"].iloc[sample]
  case_name = df["Case Name"].iloc[sample]
  summary = df["Summary"].iloc[sample]
  texts = text_splitter.create_documents([doc])
  chunks = []
  for text in texts:
    chunks.append(llm.invoke(generate_prompt(text)).content)

  combined_chunk_elements = ' \n'.join([f"Chunk {idx+1} out of {len(chunks)}: \n" +  chunk
                                          for idx, chunk
                                          in enumerate(chunks)])

  ai_msg_sum = llm.invoke(generate_prompt_combined(combined_chunk_elements))
  sum_join = " ".join(ai_msg_sum.content.split("\n\n"))
  csv_data.append((case_name, summary, sum_join))
```

Figure 4: Domain Summary CoT Chunked Anthropic

5. Flow Diagram:

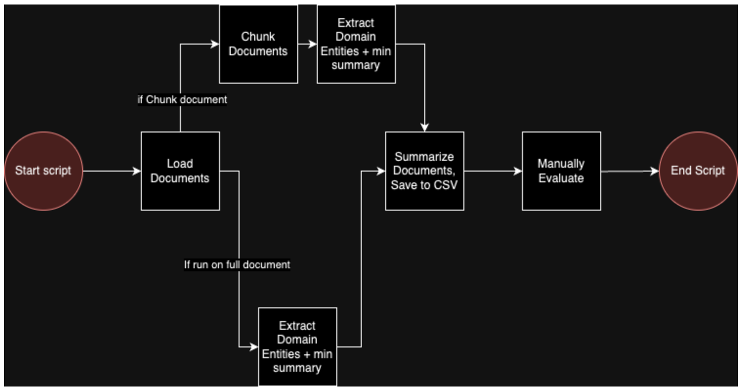


Figure 5: Flow diagram

6. 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 Anthropic account, generate an API Key, and set "ANTHROPIC_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 Domain CoT example

Below is an example summary made by the Domain Specific Summary Chain of Thought technique and Haiku:

This was a class action lawsuit filed on November 18, 1987, in the United States District Court for the District of Maryland. The plaintiffs included incarcerated individuals at the Talbot County Detention Center (TCDC), such as a 50-year-old Black man awaiting trial for 15 months, a young Black woman serving a sentence, a 20-year-old white man incarcerated since July 1987, and a 34-year-old Black man incarcerated since September 1987. The organizational plaintiffs were relatives or visitors of the incarcerated individuals. The defendants were the Jail Administrator, the County Manager of Talbot County, and members of the Talbot County Council. The plaintiffs were represented by attorneys from the ACLU of Maryland and the ACLU National Prison Project. They alleged violations of their rights under the First, Sixth, Eighth, and Fourteenth Amendments to the U.S. Constitution, stemming from the poor conditions at the TCDC, including overcrowding, lack of basic necessities, inadequate medical and mental health care, and restrictions on visitation and access to courts. The plaintiffs sought declaratory and injunctive relief to address these issues, as well as an award of reasonable costs and attorneys' fees.

This summary is improved over the Llama 3.2 3b summary as it gets more facts correct such as the TCDC true meaning and includes narrative elements about the jail conditions

6 Proof of work

Its a known phenomenon that larger LLMs have an emergent property to reason better than smaller LLMs. This can explain why Haiku, a 20b parameter model, outperforms Llama 3.2 3b, a much smaller model.

6.1 Known Limitations

Haiku is not the best model offered by Anthropic. More tests needed with Sonnet to compare truly the best performing model from them and how that impacts summarization quality.