

HAAG NLP Summarization Week 7

Michael Bock

October 2024

1 Slack Questions

What did you accomplish this week?

- BERT wasn't working well, so tried getting a chatbot to classify. It was able to classify issues in a civil rights case correctly, but not case types. This indicates that maybe my choice labels isn't good.
- I was able to get pretty good looking classifications using ChatOllama and LLaMa 3.2 on the clearinghouse.net data. Once we get the UPenn data, this should be an avenue of investigation. It lacks some properties that Dr. Alexander mentioned we would eventually want, like being able to classify novel issues or saying it doesn't know, but I *might* be able to prompt it to do that. I also have another special loss function I've been working on that should be able to turn any of the huggingface classifiers into models which should be able to accomodate unknown examples and novel issues.

What are you planning on working on next?

- If Dr. Alexander says to, start trying to get the UPenn data instead of continuing exploration on the clearinghouse data, which I was using as a surrogate until the UPenn data arrived.
- Georgia Tech is on break this weekend, so I may continue on the clearinghouse data just because that's the only data I have right now. This won't yield any good results for the purposes of DNO classification, but it will allow me to see if my loss function for the novel classes can work on any issues classification data and it will allow me to have functioning training scripts ready for when the UPenn data comes. A goal of this would be that we can just run the training script on the UPenn data once it comes with no adjustments and hopefully get something that performs well.

What is blocking you from progressing?

- Need UPenn data to get definitive results

2 Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity

task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities

<https://arxiv.org/pdf/1301.3781>

2.1 Brief Analysis

One thing I've been interested for novel classes or being able to say the model doesn't know what issues a case involves are latent spaces. Latent spaces are like the insides of neural networks. As a neural network classifies, it must extract important features that describe classes well. During extraction, neural networks for rich latent spaces where examples from classes cluster together.

For language models, this used to be a very difficult task. Before neural networks, language models used statistical N gram methods, which group multiple words together and pair that with an ID to input to a neural network. So for example, the word **Languauge** may have ID 1 and is a 1-gram. A 2-gram, like **New York** will have a different numerical ID for the neural network.

Word2Vec uses a neural network to turn these numerical IDs into a feature-rich, continuous space for other neural networks to build off. It proposes two models to build a latent space. First, is Continuous Bag of Words. Similar to masked language modeling, CBoW predicts the current word using surrounding words as context. Second is Skip-Gram modeling, which predicts surrounding words given a continuous input word. It predicts the context given the word.

Word2vec has succeeded in enabling downstream language modeling tasks. One cool property of word2vec is that you can use the embedding created by word2vec to solve simple queries. For example, in word2vec "king"- "man" = "queen". One good example I use to understand the that the latent space of word2vec is feature-rich is those videos on tiktok where there is an emoji and word and you drag them together to create a new word and emoji. Those are built using word2vec embeddings and they add together the two terms in the embedding space to get a new word.

Word2vec is a bit antiquated for this project, but the idea that it has a feature-rich latent space that we can exploit to make classifications remains relevant. In order to classify, we can use pretrained model that has already found a feature-rich latent space. Then, we can add a classification head to that model that will output a final classification output. This won't require as much data as training a model from scratch because it won't require the model to learn all the parts of the english language or all the parts of legal lexicon.

We can also exploit a feature-rich latent space to be able to know when a model doesn't know the answer. One way to do this is by understanding that examples of the same class must cluster together in the latent space. If a classification of a certain class is really far away from its cluster, this is an indication that the language model doesn't actually know what the correct classification is.

3 Scripts and Code Blocks

chat_llm.py

```
1 import pandas as pd
2 from langchain_ollama import ChatOllama
3 from langchain_core.messages import AIMessage
```

```

4 import sys
5 sys.path.append('../mistral')
6 from mistral_datasets import DocumentClassificationDataset, ISSUE_IDS
7
8 def generate_prompt(document, issue):
9     """Generates a prompt for the model to summarize a legal document with emphasis
10     on detailed legal claims and chronological storytelling."""
11     prompt = f"""
12     CASE: '{document}'
13     You are a lawyer trying to identify this case with a civil rights issue. Is
14     the civil rights issue this case is associated with {issue}? Answer only \"Yes\"
15     or \"No\".
16     Do not give any explanation. The answer must contain only one word
17     """
18     return prompt
19 cases = DocumentClassificationDataset(None, '../all_cases_clearinghouse.pkl', n =
20     1)
21 doc, label = cases[0]
22 with open('coarse_issues', 'r') as f:
23     for issue in f.readlines():
24         print(issue[:-1])
25         prompt = generate_prompt(doc, issue[:-1])
26         llm = ChatOllama(model="llama3.2")
27         ai_msg = llm.invoke(prompt)
28         print(ai_msg.content)

```

coarse_issues

```

1 Reproductive rights
2 Constitutional Clause
3 Special Case
4 General
5 Prison Crowding
6 Discrimination
7 Disability
8 Mental Disability
9 Medical
10 COVID-19
11 Benefits Programs
12 EEOC
13 Voting
14 Death Penalty
15 Immigration or the Border
16 Injunction Content

```

4 Documentation

The above script invokes a chat bot and asks it whether or not a case uses different legal issues. It uses Ollama and llama 3.2. The script conducts 2 experiments. I prompt llama by first giving it the case, and then asking it whether or not the case is associated with one of the issues from coarse_issues. I require them LLM to only output a 1 word response so it is easier to parse. If the requested issue is part of the case, the LLM responds **Yes** otherwise it responds **No**. In the second experiments, I test it with finer issues. Some of the terms used to describe issues like **General** and **Special Case** I observed to be too vague for the model to predict accurately. So I added in more issues related to the case I was testing with: **General/Misc.:** **Personal injury,**

General/Misc.: Food service / nutrition / hydration, General/Misc.: Sanitation / living conditions, Jails, Prisons, Detention Centers, and Other Institutions: Visiting, General/Misc.: Access to lawyers and General/Misc.: Bathing and hygiene.

5 Script Validation(Optional)

Results from my LLaMa experiment, can be replicated using `ollama pull llama3.2 && python chat_llm.py` :

```
1 Iterate over complaints
2   0%|

      | 0/1 [00:00<?, ?it/s]['General/Misc.: Personal injury', '
General/Misc.: Food service / nutrition / hydration', 'General/Misc.: Sanitation
 / living conditions', 'Jails, Prisons, Detention Centers, and Other
Institutions: Visiting', 'General/Misc.: Access to lawyers or judicial system',
'General/Misc.: Bathing and hygiene']
3
4 Reproductive rights
5 No
6 Constitutional Clause
7 Yes
8 Special Case
9 Yes
10 General
11 Yes
12 Prison Crowding
13 Yes
14 Discrimination
15 Yes
16 Disability
17 No
18 Mental Disability
19 No
20 Medical
21 Yes
22 COVID-19
23 No
24 Benefits Programs
25 No
26 EEOC
27 No
28 Voting
29 No
30 Death Penalty
31 No
32 Immigration or the Border
33 Yes
34 Injunction Content
35 Yes
36 General/Misc.: Personal injury
37 Yes
38 General/Misc.: Food service / nutrition / hydration
39 Yes
40 General/Misc.: Sanitation / living conditions
41 Yes
```

```
42 Jails , Prisons , Detention Centers , and Other Institutions : Visiting
43 Yes
44 General/Misc.: Access to lawyers or judicial system
45 Yes
46 General/Misc.: Bathing and hygiene
47 Yes
```

The fine issues are the final 6. I took them from the labels of the case shown. Since they all say Yes, I think this shows that LLaMa is able to understand that these issues are part of the given case. I still need to test it with many different cases, but I may change directions depending on whether Dr. Alexander wants me to focus more on acquiring the UPenn data or exploring classification methods.

6 Results Visualization

Figure 1

7 Proof of work

```
100%
Reproductive rights
No
Constitutional Clause
Yes
Special Case
Yes
General
Yes
Prison Crowding
Yes
Discrimination
Yes
Disability
No
Mental Disability
No
Medical
Yes
COVID-19
No
Benefits Programs
No
EEOC
No
Voting
No
Death Penalty
No
Immigration or the Border
Yes
Injunction Content
Yes
General/Misc.: Personal Injury
Yes
General/Misc.: Food service / nutrition / hydration
Yes
General/Misc.: Sanitation / living conditions
Yes
Jails, Prisons, Detention Centers, and Other Institutions: Visiting
Yes
General/Misc.: Access to lawyers or judicial system
Yes
General/Misc.: Bathing and hygiene
Yes
```




Figure 1: The model being correct on the fine issues

8 Next Week's proposal

- Fix BERT using tutorials provided by Huggingface, Tom, and Nathan.
- Expand upon my test with the fine issues from the clearinghouse dataset. Produce a confusion matrix on a small subset of the data to decide for sure if ChatBot prompting can be used for classification on the clearinghouse dataset.

HAAG Research Report
NLP - Sentencias / NLP - Gen Team
Week 8

V́ctor C. Ferńandez
October 2024

CONTENTS

1	Weekly Project Updates	2
2	Abstracts	3
3	Scripts and Code Blocks	4
4	Documentation	11
5	Script Validation	13
6	Results Visualization	14
7	Proof of Work	15
8	Next Week's Proposal	17

1 WEEKLY PROJECT UPDATES

What progress did you make in the last week?

- Researched ways of training a Hugging face model.
- Moved code to PACE/ICE environment and currently figuring out how to load and run Ollama.
- Refactored code for bulk processing all test dates data generated by Karol last week.
- Met with the NLP-Sentencias team on Sunday 6th to align on our goals and distribute our tasks more efficiently.
- Mailed several professors for our speaker series, having 1 new seminar coming in November.
- Meeting with the NLP team on October 11th for our weekly meeting.
- Meeting with Dr. Alexander and Nathan Dahlberg on October 11th to get further insights on NLP research.

What progress are you making next?

- Finalize configuring PACE environment and then running larger models to verify results vs smaller models.
- Train Hugging Face model to retrieve dates from text.
- Compare outputs from different models to select best performing one.
- Meet with the NLP team on October 18th for our weekly meeting.
- Meet with Dr. Alexander and Nathan Dahlberg on October 18th 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:** NLP-CIC-WFU at SocialDisNER: Disease Mention Extraction in Spanish Tweets Using Transfer Learning and Search by Propagation

- **URL:** <https://aclanthology.org/2022.smm4h-1.6.pdf>

- **Abstract:** Named entity recognition (e.g., disease mention extraction) is one of the most relevant tasks for data mining in the medical field. Although it is a well-known challenge, the bulk of the efforts to tackle this task have been made using clinical texts commonly written in English. In this work, we present our contribution to the SocialDisNER competition, which consists of a transfer learning approach to extracting disease mentions in a corpus from Twitter written in Spanish. We fine-tuned a model based on mBERT and applied post-processing using regular expressions to propagate the entities identified by the model and enhance disease mention extraction. Our system achieved a competitive strict F1 of 0.851 on the testing data set.

- **Summary:** This paper presents a solution for extracting disease mentions from Spanish-language tweets, as part of the SocialDisNER competition. The authors employ a transfer learning approach based on a fine-tuned multi-lingual BERT (mBERT) model, specifically designed for clinical texts. The approach includes pre-processing the data to adhere to the BIO tagging scheme and post-processing to enhance the extraction of disease mentions. The system deals with common challenges in social media text, such as misspellings and irregular structures (e.g., hashtags, URLs). Additionally, a propagation mechanism is applied to detect repetitive disease mentions. The proposed model achieved competitive results, with a strict F1 score of 0.851 on the test dataset, indicating a strong performance in the task of disease extraction from Spanish social media content.

- **Relevance:** Although the paper's domain is disease mention extraction from social media, its underlying techniques, such as transfer learning with multi-lingual models (mBERT) and the use of pre-processing and post-processing steps to handle irregularities in natural language text, are directly applicable to the challenges we face in our legal document processing. The Dominican "Sentencias" contain varying linguistic patterns, and leveraging a model fine-

tuned for Spanish-language text extraction can provide valuable insights for our date extraction task. Furthermore, the error analysis and handling of irregularities like hashtags in the paper can inform strategies to improve the accuracy of extracting structured data from the unstructured legal texts in our project.

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. Training data normalization. The intention of this code is to normalize all the files for the training data that were generated from original documents, since they were generated using ChatGPT and it turns out the keys in the different files are different. For instance, one file contains the key "standard_format" for the standard format date, whereas the other contains "standard_date" for the same field. The file for this code may be found [here](#).

```

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({
                        "standard_format": entry[keys[0]], #
                            ↪ First key and its value
                        "original_format": entry[keys[1]], #
                            ↪ Second key and its value
                        "indices": entry[keys[2]], #
                            ↪ Third key and its value
                        "context": entry[keys[3]] #
                            ↪ Fourth 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:

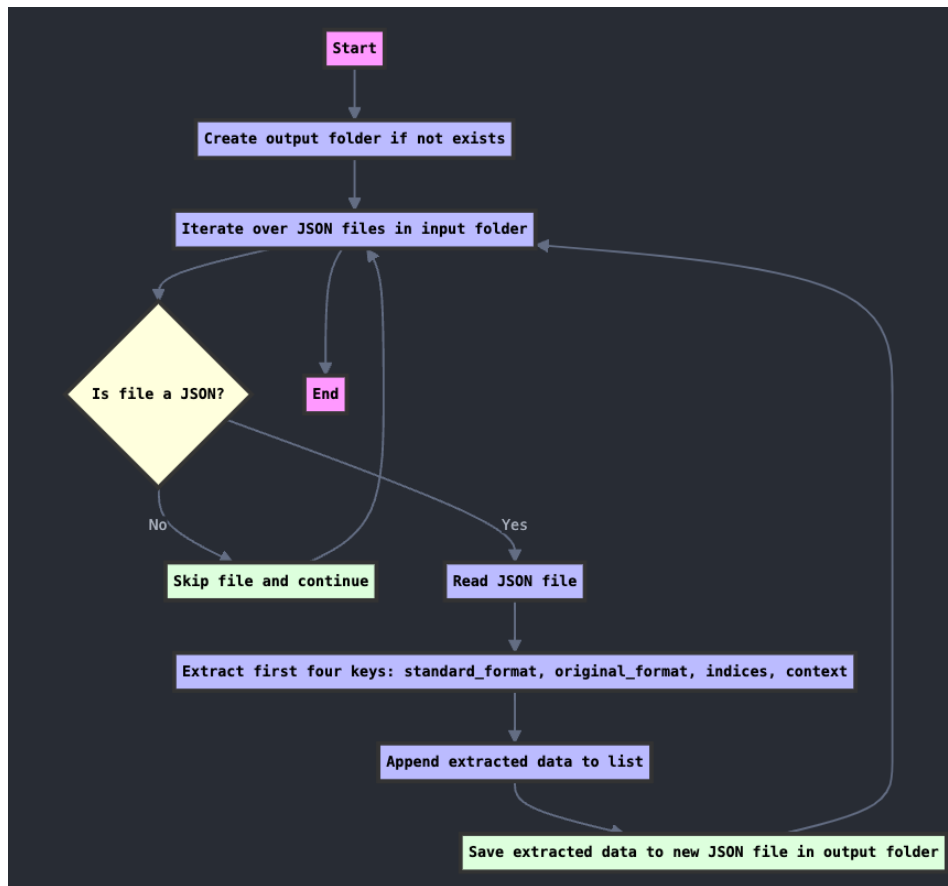


Figure 1—Date normalizing process

2. Once data has been normalized and we can ensure that all the keys in all the files are the same, then there is an additional code that will generate new json

files that only contain the standard date and the date as it is shown in the document. The logic for this code is very similar to the previous one with the difference that each date will now return 3 keys: "standard_format", "original_format" and "context". This last one having a default value of "TO_BE_FILLED_IN". The code for this file may be found [here](#).

3. 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 el contexto adecuado y devuelve solo un
- JSON:

{{MODEL_OUTPUT_FORMAT}}

Utilizando solo las siguientes opciones para la respuesta:

{{OPTIONS}}

Importante: Incluye solo el JSON en la respuesta.
```

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

4. New options template. Instead of having a single file with output and options, a new options file was created containing all the different categories on to which a date could be mapped, including an "Others" section for dates that do not fit into the defined categories. This file may be found [here](#).

```

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

```

Code 3—Possible categories for mapping dates

5. Refactored code using the `OllamaModelProcessor` class created to handle querying models and passing hyperparameters. This code mainly puts together all the previous pieces of code, generating a variable input query for the model and then processes the output, adding the original date at the top and storing it into a file. This way, we can also assess if the model not only returns the right context but also maintains the dates in their original form without generating hallucinations. Code may be found [here](#).

```

def generate_query(query_template: str, document_content: str,
    options: str):
    with open(query_template, 'r', encoding='utf-8') as f:
        query_template_content = f.read()
        # Replace the placeholders in the query template with the
        # actual content
        query = query_template_content.replace("{{DOCUMENT_CONTENT}}",
            document_content)
        query = query.replace("{{OPTIONS}}", options)
    return query

def log_in_color(text: str, color: str):
    print(colored(text, color))

```

```

def generate_output(ollama_models: list, query_template: str,
    → input_folder: str, dates_folder: str, output_folder: str,
    → model_hyperparameters: dict = {}):
    # Instantiate the OllamaModelProcessor

    for model in ollama_models:
        # Log the model being processed:
        log_in_color(f"Processing model: {model}", "green")
        # Step 1: Instantiate the OllamaModelProcessor
        processor = OllamaModelProcessor(model,
            → **model_hyperparameters)
        # Step 2: Get the output options from the
            → date_options.json file contained in the option key
        options_file = "date_options.json"
        with open(options_file, 'r', encoding='utf-8') as f:
            options_content = json.load(f)
        # Now set the options to be the value of the options key
        options = json.dumps(options_content["options"])

    for filename in os.listdir(input_folder):
        if filename.endswith(".txt"): # Process only txt files
            # Log the file being processed:
            log_in_color(f"Processing file: {filename}",
                → "blue")
            # We append locate the output folder under a
                → folder with the file name first and then a
                → folder with the model name
            file_output_folder = os.path.join(output_folder,
                → filename, model)
            # Create the output folder if it doesn't exist
            os.makedirs(file_output_folder, exist_ok=True)
            # Read the content of the document
            document_path = os.path.join(input_folder,
                → filename)
            with open(document_path, 'r', encoding='utf-8')
                → as f:
                document_content = f.read()

```

```

query_without_dates =
    → generate_query(query_template,
    → document_content, options)

# Now we query the model replacing the last place
    → holder with each date independently
# Read the dates JSON file with same name as the
    → document
dates_file = os.path.join(dates_folder,
    → f"{os.path.splitext(filename)[0]}.json")

with open(dates_file, 'r', encoding='utf-8') as
    → f:
    date_objects = json.load(f)

for i, date_object in enumerate(date_objects):
    # Log the date position being processed:
    log_in_color(f"Processing date: {i}",
        → "magenta")
    expected_output = json.dumps(date_object)
    query = query_without_dates.replace("{}MODEL_
        → OUTPUT_FORMAT}",
        → expected_output)

    output = processor.query_model(query)
    output_path = os.path.join(file_output_folder,
        → f"{os.path.splitext(filename)[0]}_{i}.txt
        → ")
    with open(output_path, 'w', encoding='utf-8')
        → as f:
        # First write a line with the date object
            → passed to the model
        f.write(expected_output + "\n\n")
        f.write(output)

```

Code 4—Model querying functions with variable inputs

Below is the flow that explains the process happening within this code:



Figure 2—Date normalizing process

4 DOCUMENTATION

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:

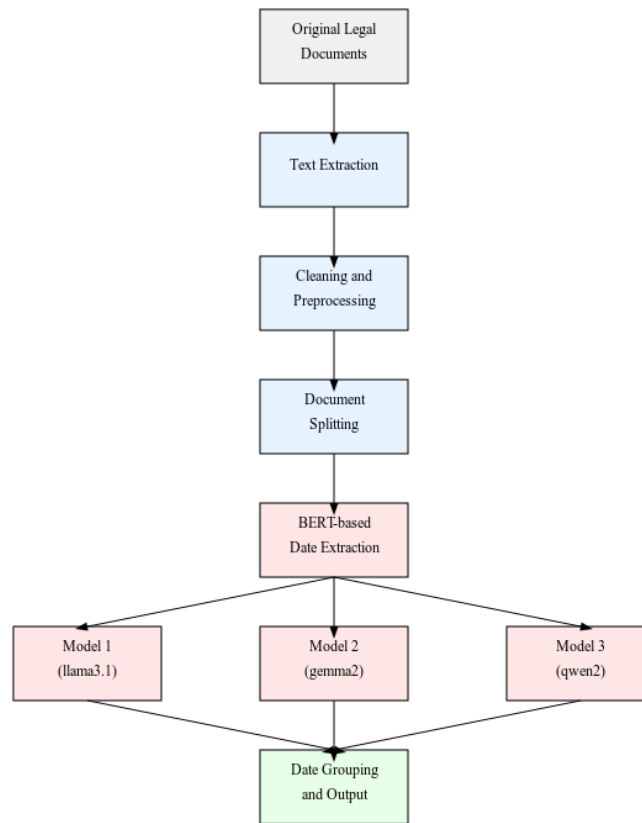


Figure 3—Full date extraction process

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 3B version of Llama3.2 as a proof of concept. Given prior experience extracting data, the next step will be to run the working code within PACE to run a larger model that will probably return better results.

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 Llama3.2 3B 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": "2023/08/21",
  "original_format": "21 de agosto de 2023",
  "context": "fecha de depósito del memorial de casación"
}

{
  "execution_details": {
    "model_name": "llama3.2",
    "hyperparameters": {
      "temperature": 1e-07,
      "top_k": 10,
      "top_p": 0.5,
      "seed": 42
    }
  },
  "processing_time": 2.586211919784546,
  "timestamp": "2024-10-10 00:45:01"
}
}
```

Code 5—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 generally being altered in the output.

In this case, a Llama 3.2 3B 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. This wasn't the actual case and although results were very similar in content with a Llama 3.2 3B model, they weren't exactly the same for the sample text used. Additionally, the model modified the initial date passed as an input. This leads to additional checks to consider, since now we'll also need to validate the model is actually returning the correct date that was passed initially. Below is an example where the input date gets modified in the output:

```

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

{"standard_format": "2023/08/21", "original_format": "21 de agosto
  de 2023", "context": "fecha de dep\u00f3sito del memorial de
  casaci\u00f3n"}

{
  "execution_details": {
    "model_name": "llama3.2",
    "hyperparameters": {
      "temperature": 1e-07,
      "top_k": 10,
      "top_p": 0.5,
      "seed": 42
    },
    "processing_time": 2.586211919784546,
    "timestamp": "2024-10-10 00:45:01"
  }
}

```

Code 6—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 configuring PACE environment and then running larger models to verify results vs smaller models.
2. Train Hugging Face model to retrieve dates from text.
3. Compare outputs from different models to select best performing one.

Week 8 | HAAG - NLP | Fall 2024

Alejandro Gomez

October 11th, 2024

1 Time-log

1.1 What progress did you make in the last week?

- This week, my team went full force on our recent pivot where we decided we would be working on identifying dates from the legal texts and then providing context. Our target for this week was to complete up to the identification model. This included the cleaning/preprocessing of the JSON data used for finetuning as the indices were inaccurate. I was able to successfully QA our JSON to understand its errors so that I could cleanse it. Then I was able to follow a basic HuggingFace tutorial adapted my custom use-case to be able to finetune the pretrained NER Spanish model with our current dataset. I'll discuss the results of this activity down below because there was significant improvement in results, but an overfitting suspicion remains, which may be able to be mitigated by using far more training data and validation data as well as a test dataset.

1.2 What are you planning on working on next?

- For the upcoming week, I want to compare with my team a few things. I want to compare our data preprocessing/cleaning scripts to see which we should put into the formal ML pipeline we are working on together. I also want to compare with another teammate their approach to training a llama model. I was tasked with training a NER model so I'd like to compare our experiences and results and understand if there are takeaways from each other. We also need to hone in on a conference paper that we will be targetting and write up a rough draft of an Abstract for our slated publication.

1.3 Is anything blocking you from getting work done?

N/A

2 Article Review

2.1 Abstract

Recently, many studies have illustrated the robustness problem of Named Entity Recognition (NER) systems: the NER models often rely on superficial entity patterns for predictions, without considering evidence from the context. Consequently, even state-of-the-art NER models generalize poorly to out-of-domain scenarios when out-of-distribution (OOD) entity patterns are introduced. Previous research attributes the robustness problem to the existence of NER dataset bias, where simpler and regular entity patterns induce shortcut learning. In this work, we bring new insights into this problem by comprehensively investigating the NER dataset bias from a dataset difficulty view. We quantify the entity-context difficulty distribution in existing datasets and explain their relationship with model robustness. Based on our findings, we explore three potential ways to de-bias the NER datasets by altering entity-context distribution, and we validate the feasibility with intensive experiments. Finally, we show that the de-biased datasets can transfer to different models and even benefit existing model-based robustness-improving methods, indicating that building more robust datasets is fundamental for building more robust NER systems. [doi\[MWZ+23\]](#)

2.2 Summary

This paper was extremely valuable for me because it goes over the flaw of NER in which the named entities will have bias from the training and may not pick up on context. One example from the paper is: "President Bush told Mr. Apple in this week 's interview..." In this case, Apple is the last name of an individual but the model may label it as an ORG (organization) which would be inaccurate. This is where the context is needed. This leads to a discussion on context-enhanced NER - a subject the team hinted at briefly. This is important since context is a critical component to the dates we want to track in order to make predictions about them toward the end of the research pipeline. Given this new information, I feel empowered to suggest to the team: "perhaps when we identify a named entity, we can have the model grab context based on the 20 pre- and 20 post- tokens to the identified entity. This paper was valuable knowledge and I feel that It strengthened my understanding for NER's limitations and suggested a method to mitigate that I could try to apply.

3 Scripts and Code Blocks

3.1 Code

```
1
2 # this is only a representation of data for demonstration purposes
3
4 # INCORRECT JSON
5
6 incorrect_json_indices = [
7     {
8         "standard_format": "2024/01/31",
9         "original_format": "31 de enero de 2024",
10        "index": [
11            106,
12            123
13        ],
14        "context": "SENTENCIA DEL 31 DE ENERO DE 2024, N M. SCJ-PS-24-0001"
15    },
16    {
17        "standard_format": "2018/12/18",
18        "original_format": "dieciocho (18) del mes de diciembre del a o dos mil
19        dieciocho (2018)",
20        "index": [
21            1645,
22            1702
23        ],
24        "context": "revoca parcialmente la sentencia n m. 035-18-SCON-01770, de fecha
25        dieciocho (18) del mes de diciembre del a o dos mil dieciocho (2018), dictada por
26        la Segunda Sala de la C mara Civil y Comercial del Juzgado de Primera Instancia
27        del Distrito Nacional"
28    },
29 ]
30
31 # section_51-56_cleaned.json
32 correct_json_indices = [
33     {
34         "standard_format": "2023/03/31",
35         "original_format": "31 de marzo de 2023",
36         "indices": [
37             285,
38             304
39         ],
40         "context": "Sentencia impugnada: Tercera Sala de la C mara Civil y Comercial
41         de la Corte de Apelaci n del Distrito Nacional, del 31 de marzo de 2023."
42     },
43     {
44         "standard_format": "2018/12/18",
45         "original_format": "dieciocho (18) del mes de diciembre del a o dos mil
46         dieciocho (2018)",
47         "indices": [
```

```

43         2598,
44         2666
45     ],
46     "context": "revoca parcialmente la sentencia n m. 035-18-SCON-01770, de fecha
dieciocho (18) del mes de diciembre del a o dos mil dieciocho (2018), dictada por
la Segunda Sala de la C mara Civil y Comercial del Juzgado de Primera Instancia
del Distrito Nacional"
47 },
48 ]
49 # section_51-56_cleaned.txt
50
51 txt_file = ""
52
53
54 Bolet n Judicial n m. 1358 Primera Sala Suprema Corte de Justicia 3 Segunda
Sala www.poderjudicial.gob.do SENTENCIA DEL 31 DE ENERO DE 2024, N M. SCJ-PS
-24-0001 Sentencia impugnada: Tercera Sala de la C mara Civil y Comercial de la
Corte de Apelaci n del Distrito Nacio- nal, del 31 de marzo de 2023.
55 Materia: Civil.
56 Recurrente: Aim Josefina Grand.
57 Abogado: Lic. Juan F. De Jes s M.
58 Recurridos: Asociaci n Cibao de Ahorros y Pr stamos y compartes.
59 Abogados: Licda. Olga Mar a Veras L. y Lic. Nardo Au- gusto Matos Beltr .
60 Jueza ponente: Pilar Jim nez Ortiz.
61 Decisi n: Declara Caducidad.
62
63 EN NOMBRE DE LA REP BLICA La PRIMERA SALA DE LA SUPREMA CORTE DE JUSTICIA, competente
para conocer de los recursos de casaci n en materia civil y comercial,
regularmente constituida por los jueces Pilar Jim nez Ortiz, presidente,
Justiniano Montero Montero, Samuel Arias Arzeno y Vanessa Acosta Peralta, miembros,
asistidos del secretario general, en la sede de la Suprema Corte de Justicia,
ubicada en Santo Domingo de Guzm n, Distrito Nacional, en fecha 31 de enero de
2024, a o 180 de la Inde- pendencia y a o 161 de la Restauraci n, dicta la
siguiente sentencia: En ocasi n del recurso de casaci n interpuesto por la
se ora Aim Josefina Grand, quien tiene como abogado apoderado al Lcdo. Juan F.
64 De Jes s M.; de generales que constan en el expediente.
65 Bolet n Judicial n m. 1358 Primera Sala Suprema Corte de Justicia 4 www.
poderjudicial.gob.do En este proceso figuran como partes recurridas a) Asociaci n
Cibao de Ahorros y Pr stamos, debidamente representada por su presiden- te
ejecutivo, Jos Luis Ventura Casta os, quien tiene como abogados apoderados a los
Lcdos. Olga Mar a Veras L. y Nardo Augusto Matos Beltr ; b) Constructora Armando
Toros C. por A. Jos Rosado Torres, y Consorcio de Propietarios del Condominio
Residencia Torres Las Perlas; quienes no depositaron constituci n de abogado,
memorial de defensa ni notificaci n del memorial de defensa ante esta Corte de
Casaci n.
66 Contra la sentencia civil n m. 1303-2023-SSEN-00149, de fecha 31 de marzo de 2023,
dictada por la Tercera Sala de la C mara Civil y Co- mercial de la Corte de
Apelaci n del Distrito Nacional, cuyo dispositivo copiado textualmente dispone lo
siguiente: Primero: Pronuncia el defecto por falta de concluir de la parte re-
currida, Consorcio de Propietarios del Condominio Torres Las Perlas, no obstante
haber sido citado. Segundo: Acoge, en cuanto al fondo, el recurso de apelaci n
interpuesto por se ora Aim Josefina Grand, revoca parcialmente la sentencia n m
. 035-18-SCON-01770, de fecha dieciocho (18) del mes de diciembre del a o dos mil
dieciocho (2018), [REST OF TEXT REMOVED FOR DEMONSTRATION PURPOSE] Comercial de la
Corte de Apelaci n del Distrito Nacional, para la notificaci n de esta sentencia.
67
68 ""

```

Listing 1: cleaned data

Given the above conditions I ran a Python script that would fix the indices by iterating every chunk and searching for the corresponding substring. If the substring did not exist because of a hallucination from an LLM on the generation of this data, then it was removed from this data set. This script is available in [week 8 repo](#).

```

1
2 {"text": "Bolet n Judicial n m. 1358 Primera Sala Suprema Corte de Justicia
922 www.poderjudicial.gob.do SENTENCIA DEL 31 DE ENERO DE 2024, N M. SCJ-PS
-24-0114 Sentencia impugnada:\t C mara Civil, Comercial y de Trabajo de la Corte
de Apelaci n de San Juan de la Ma- guana, del 21 de febrero de 2023.\nMateria:\t
Civil.\nRecurrente:\t Alejandro Portes Roa.\nAbogado:\t Lic. Freddy Ota o de los

```

```

Santos.\nRecurrido:\t Mario Enrique Ramirez Ramirez.\nJueza ponente:\t Pilar
Jimenez Ortiz.\nDecision: Rechaza.\n\nEN NOMBRE DE LA REPUBLICA LA PRIMERA SALA
DE LA SUPREMA CORTE DE JUSTICIA, competente para conocer de los recursos de
casacion en materia civil y comercial, regularmente constituida por los jueces
Pilar Jimenez Ortiz, presidente, Justiniano Montero Montero, Samuel Arias Arzeno y
Vanessa Acosta Peralta, miembros, asistidos del secretario general, en la sede de
la Suprema Corte de Justicia, [REST OF TEXT REMOVED FOR DEMONSTRATION PURPOSE].\n\n
Firmado: Pilar Jimenez Ortiz, Justiniano Montero Montero, Samuel Arias Arzeno y
Vanessa Acosta Peralta.\nCesar Josue Garcia Lucas, secretario general de la
Suprema Corte de Justicia, CERTIFICO, que la sentencia que antecede fue dada y
firmada por los jueces que figuran en ella, en la fecha arriba indicada. www.
poderjudicial.gob.do\n", "entities": [{"start": 113, "end": 132, "label": "DATE"},
{"start": 271, "end": 292, "label": "DATE"}, {"start": 2009, "end": 2029, "label":
"DATE"}, {"start": 2246, "end": 2265, "label": "DATE"}, {"start": 3083, "end":
3102, "label": "DATE"}, {"start": 3281, "end": 3300, "label": "DATE"}, {"start":
3479, "end": 3497, "label": "DATE"}, {"start": 3569, "end": 3588, "label": "DATE"},
{"start": 3872, "end": 3891, "label": "DATE"}, {"start": 7936, "end": 7955, "label
": "DATE"}]}

```

Listing 2: dataset prep for huggingface model trainer

Given the shape of the data above, it was necessary to transform this data so that it could be used with the HuggingFace transformers library. I ran a script that iterated our current data to create a jsonl file so that each text chunk was a json in a single line with all of the NER's in that same line. This can be found in the [week 8 repo](#). See above for an example of a single line representing one text chunk for the data used for model training/validation. The final output contained nearly 180 of these.

3.2 Documentation

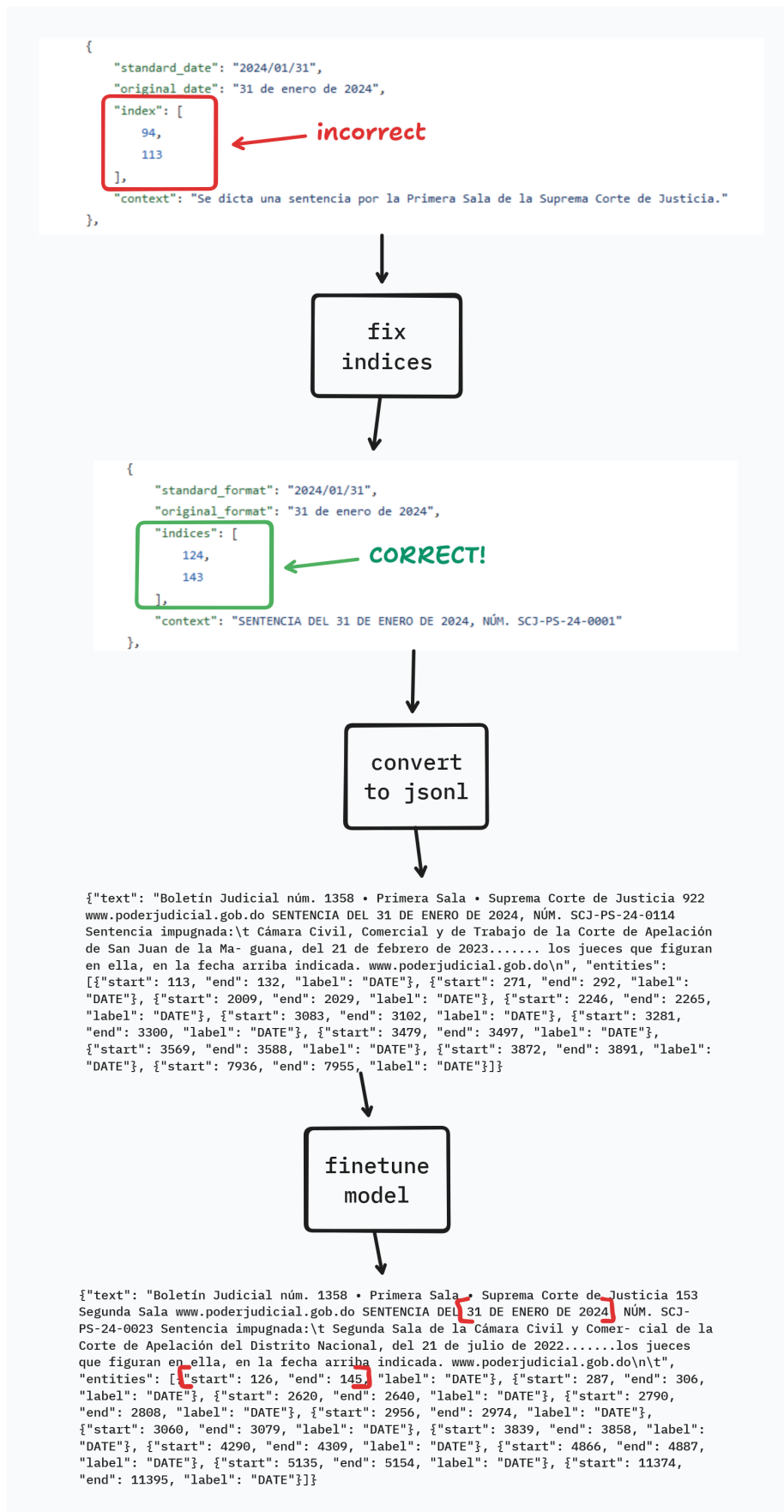


Figure 1: pipeline step visualization

3.3 Script Validation (optional)

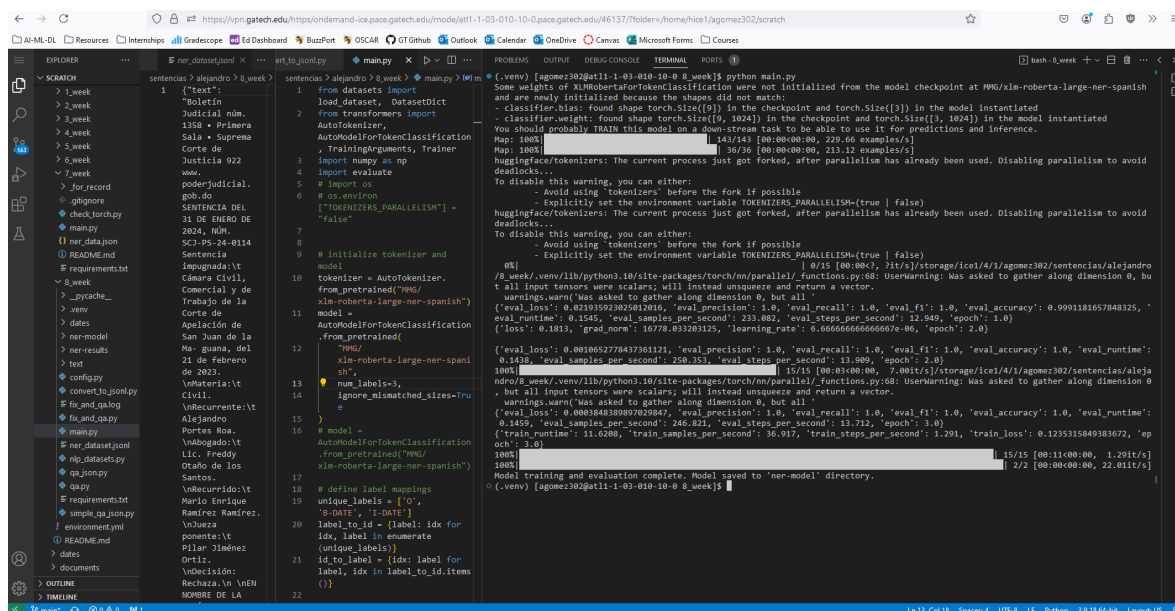


Figure 2: screenshot of model training output

This screenshot shows the the PACE-ICE development environment that I leveraged to quickly train the model as I was developing and to run the preprocessing scripts needed for the data. On the terminal on the right, the output from the model training shows the metrics that are visualized below over the course of 3 training epochs. The open tabs also show the main.py code which has the code to load the dataset and train the model. Next to this on the left, the dataset to be loading in the format of jsonl can be seen. Finally, to the far left, a directory structure with a variety of files make up week 8 efforts: QA scripts, cleaning scripts, data preparation, model training, raw data directories, etc.

3.4 Results Visualization

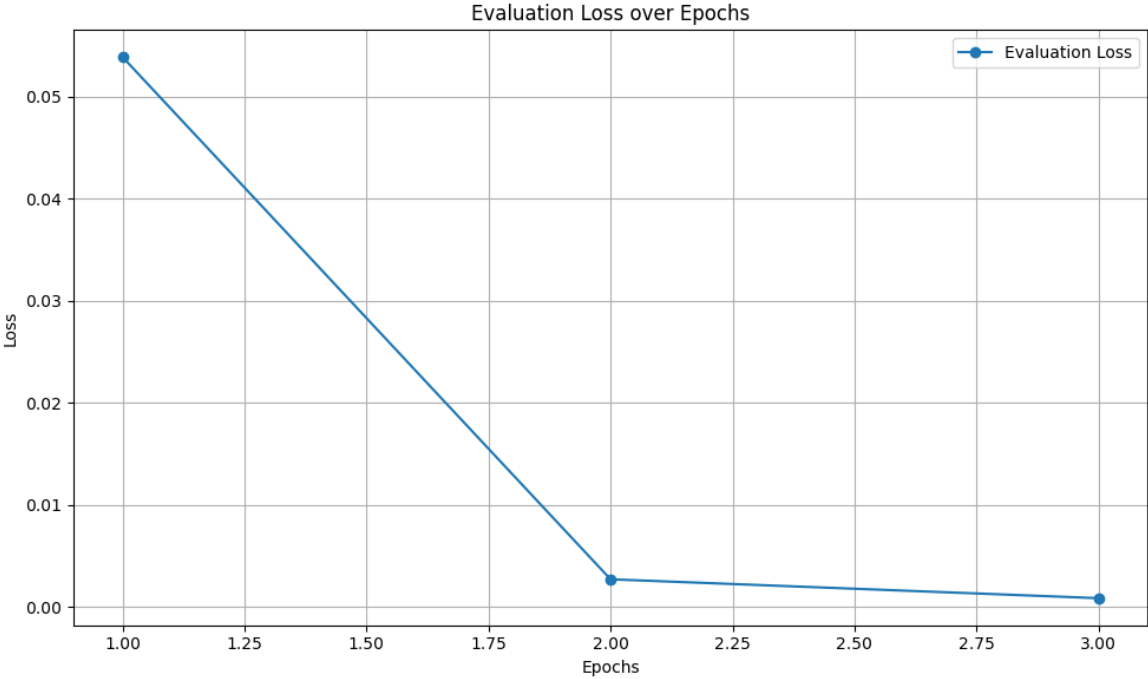


Figure 3: eval loss

This graph demonstrates the model is minimizing the errors over each training epoch - a favorable outcome.

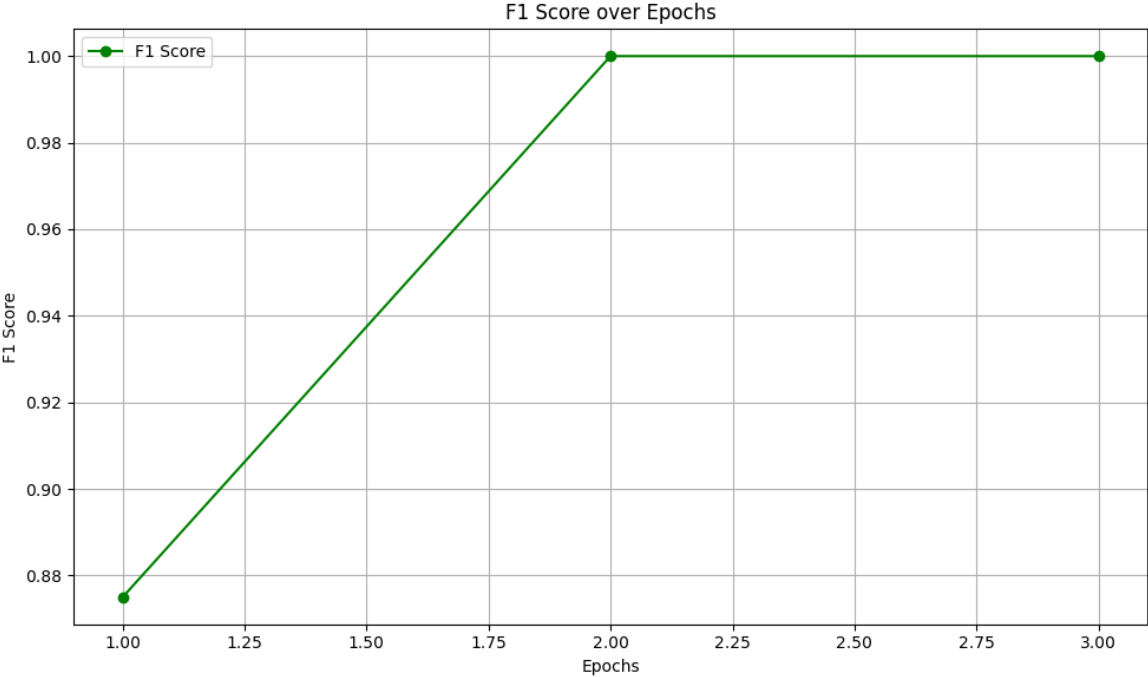


Figure 4: f1 score

It may be too early to tell but this model can be seen to increase and converge as it reaches the

balance between recall and precision, i.e. the f1 score.

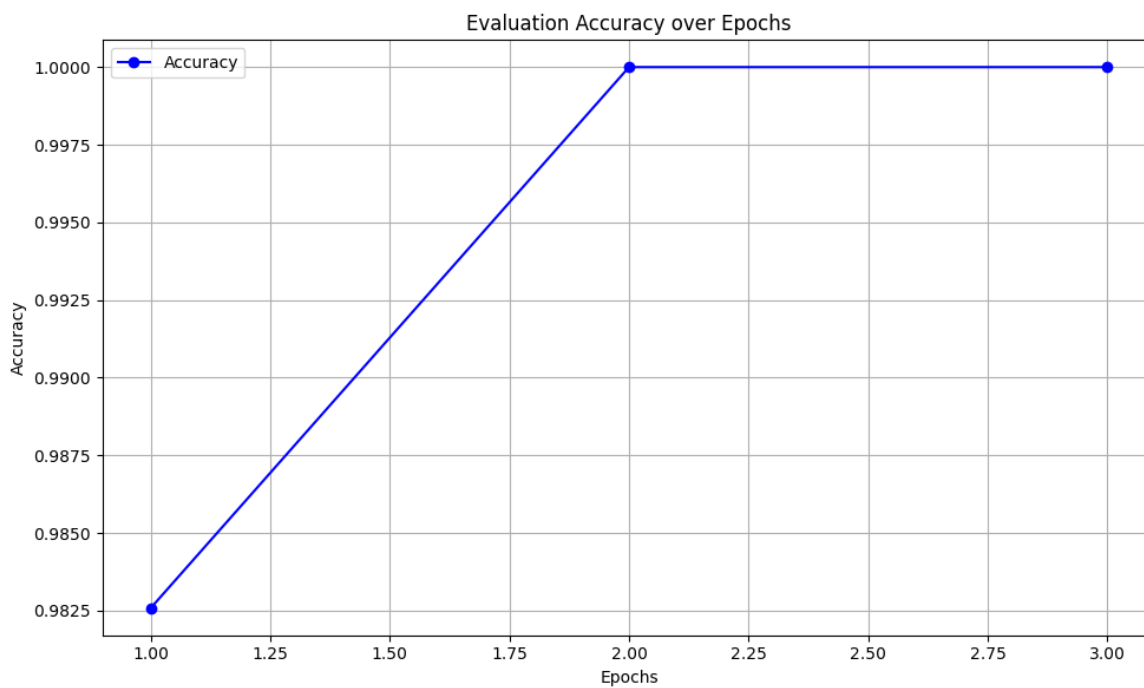


Figure 5: eval accuracy

This graphs shows an increase in accuracy over training epochs with a slight dip. With more epochs, this might not be significant but it is questionable for future reproducung of steps.

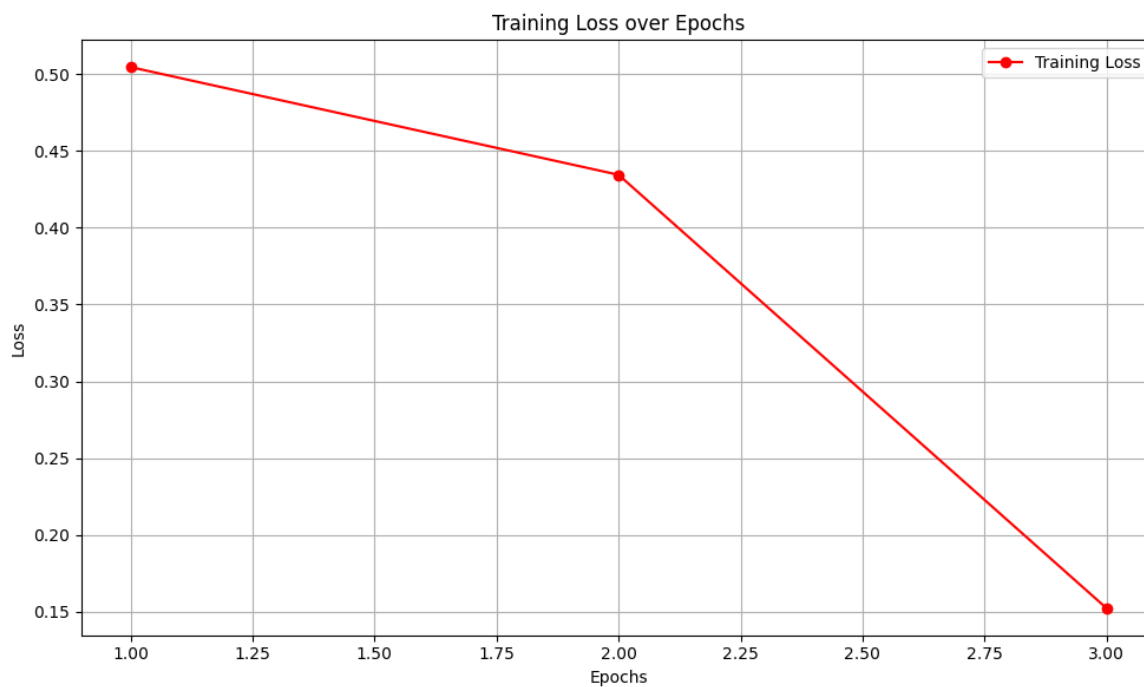


Figure 6: loss

Training loss trends downward which shows the model is reducing its errors.

SUMMARY: Generally, the graph shapes look good but given I was using decent resources, I shouldve added more data and more epochs - I would've liked to see these trends level out for the most part, save for training loss which I believe is good to dive toward the x-axis. These metrics came from the model training found in the week 8 repo where I defined a standard hugging face metric for evaluation using [segeval](#) which is common for NER models. With a Python function, I was able to print out the computations of the methods so that I could later compare them.

3.5 Proof of Work

[Scripts in GitHub Repo](#) This report focused on displaying the data processing, ML workflow, and model training outputs, but the scripts are avialable on the GH repo as usual. Many scripts were used for the data cleanse and preprocessing, training, and visualization.

4 Next Week's Proposal

- I'll be meeting with the NLP DR team this weekend again to discuss our results and plan of action for this week. We need to start making small, but critical moves toward publication and begin converging our efforts in our shared ML pipeline work.
- Finetune the model with more data and modifications to the hyper parameters. I want to ensure it is becoming accurate by having a separate test set aside from the validation set as well.
- Work on creating a draft abstract to start the efforts toward a publication.
- Update current documentation, e.g. NLP website

References

- [MWZ⁺23] Ruotian Ma, Xiaolei Wang, Xin Zhou, Qi Zhang, and Xuanjing Huang. Towards building more robust NER datasets: An empirical study on NER dataset bias from a dataset difficulty view. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4616–4630, Singapore, December 2023. Association for Computational Linguistics.

HAAG NLP Sentencias — Week 8 Report

NLP-Gen Team

Karol Gutierrez

October 11, 2024

1 Weekly Project Update

1.1 What progress did you make in the last week?

- Adjusted scripts to use ChatGPT4 to generate JSON files for individual sentencias, including dates, ranges within the document and context.
- Scripts to validate date ranges in generated files.
- Generate more training data using new 5k pages document.
- 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?

- Use SpaCy and the generated data to train model.
- Add performance for training.
- 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: LEGAL-BERT: The Muppets Straight Out of Law School [CFM+20].

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 LEGAL-BERT, a family of specialized BERT models tailored for the legal domain. The authors systematically explore adaptation strategies for BERT to effectively handle legal texts, which differ from generic corpora in vocabulary, syntax, and semantics. Key contributions of this study include:

- **Training Corpora:** The authors assembled a comprehensive dataset of 12 GB, containing diverse English legal texts from legislation, court cases, and contracts. The investigation considers three primary strategies for BERT adaptation: (i) further pre-training on legal corpora, (ii) pre-training from scratch with a custom vocabulary, and (iii) utilizing the standard BERT model without modifications.
- **Performance Insights:** Results demonstrate that both further pre-training and pre-training from scratch lead to significant performance improvements compared to the original BERT model for legal tasks. The strategies yield comparable results across multiple legal datasets, underscoring the necessity of domain-specific adaptations.
- **Hyper-Parameter Optimization:** An expanded hyper-parameter tuning strategy resulted in substantial performance enhancements in downstream tasks, indicating that traditional guidelines may not suffice. Notably, the research reveals that smaller BERT-based models can compete effectively with larger counterparts in specialized domains, with LEGAL-BERT-SMALL showcasing impressive efficiency.
- **Legal NLP Applications:** The evaluation of the models included various legal tasks, such as multi-label text classification, binary classification of court cases, and named entity recognition in contracts. The findings highlighted the superior performance of LEGAL-BERT, particularly in challenging multi-label classification tasks where domain knowledge plays a crucial role.

2.3 Relevance

This paper is useful to our project on NLP for extracting procedural history from legal documents (sentencias). Similar to our work, it emphasizes the need to adapt and fine-tune models to meet the specific linguistic and contextual needs of legal texts. The exploration of specialized pre-training methods, such as additional training on legal corpora and the creation of a custom vocabulary, directly aligns with our goal of optimizing model performance for accurate information extraction. Moreover, the emphasis on hyper-parameter tuning resonates with our understanding that a customized approach can significantly enhance model efficacy. The findings of this study provide valuable guidance and inspiration.

3 Scripts and code blocks

The code is in the private repository [repository](#). The progress for this week is in `./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 verify the dates and positions within the text, shown in Figure 2
- Generation of new cleaned data with a different original source file.

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`. No more libraries were added from the previous changes of Week 7, which appear in `./karol/week7/readme.md`.

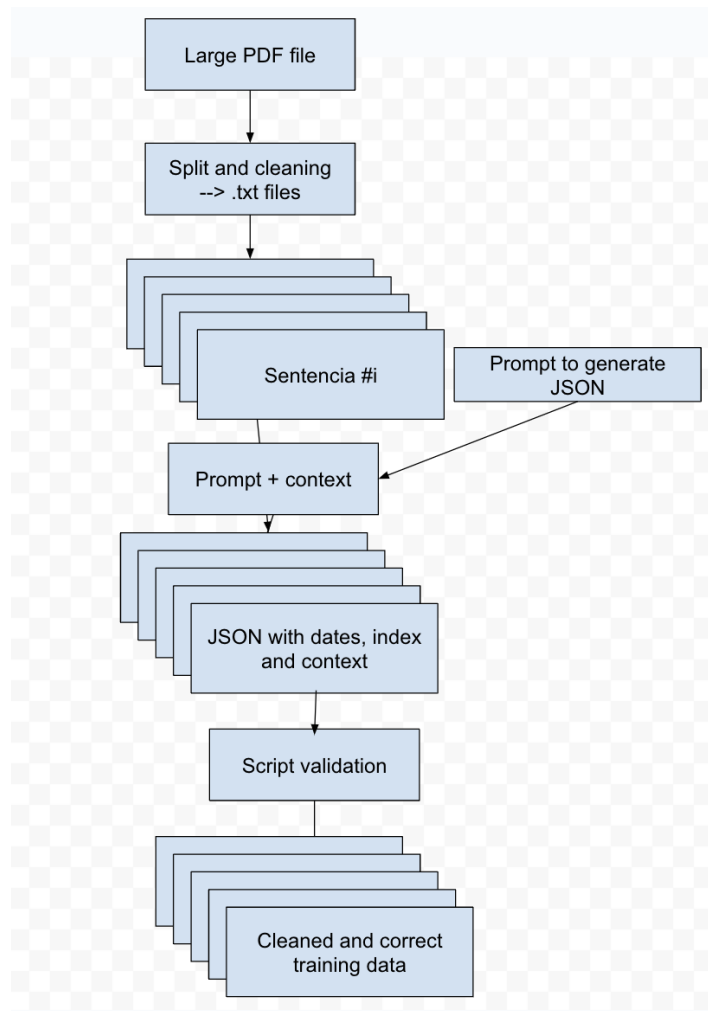


Figure 1: Code logic workflow to process file including filtering.

5 Script Validation

The scripts are validated by analyzing the final JSON results. The running of the scripts is shown in Figure 3. This script cleans the dates adjust the new indices and creates a new folder with the corrected files in Figure 4.

6 Results Visualization

As in previous submission, the initial part of the pipeline shows a sentencia, as in Figure 5. The final resulting files are generated in a new folder, and one example of such file is Figure 6 .

Once we have a training model, we will be able to provide a visual representation of the performance of our model, so far this process only generates training data.

7 Proof of Work

Figure 7 shows that the filtering process effectively detects errors in the format of the JSON files, this can be compared to the warnings detected from Figure 3 .

```

example.py  ner.py  split.py U  correct_dates.py A X  section_64-70_cleaned_json  .gitignore
karol > week8 > correct_dates.py > ...
2  import json
3
4  # Define directories
5  dates_normalized_dir = '.././dates_normalized' # Updated path
6  cleaned_dir = '.././week7/cleaned' # Updated path
7  verified_json_dir = 'verified_json' # This remains the same
8
9  # Create verified_json directory if it doesn't exist
10 os.makedirs(verified_json_dir, exist_ok=True)
11
12 # Process each JSON file in the dates_normalized directory
13 for filename in os.listdir(dates_normalized_dir):
14     if filename.endswith('.json'):
15         json_file_path = os.path.join(dates_normalized_dir, filename)
16
17         # Read the JSON file
18         with open(json_file_path, 'r', encoding='utf-8') as json_file:
19             date_entries = json.load(json_file)
20
21         # Corresponding cleaned text file
22         cleaned_file_path = os.path.join(cleaned_dir, filename.replace('.json', '.txt'))
23
24         # Read the cleaned text file
25         with open(cleaned_file_path, 'r', encoding='utf-8') as cleaned_file:
26             text_content = cleaned_file.read()
27
28         # Prepare a list to hold updated date entries
29         updated_entries = []
30
31         for entry in date_entries:
32             original_format = entry['original_format']
33             indices = entry['indices']
34
35             # Determine start and end indices based on the structure
36             if isinstance(indices, dict):
37                 start_index = indices.get('start')
38                 end_index = indices.get('end')
39             elif isinstance(indices, list) and len(indices) == 2:
40                 start_index = indices[0]
41                 end_index = indices[1]
42             else:
43                 print(f"Warning: Invalid indices format in file {filename}: {indices}")
44                 continue # Skip this entry if indices are invalid
45
46             # Find the correct indices of the original_format in the text_content
47             start_index = text_content.find(original_format)
48             end_index = start_index + len(original_format)
49
50             # Only proceed if start_index is valid (non-negative)
51             if start_index != -1:
52                 # Update indices in the entry
53                 entry['indices'] = [start_index, end_index]
54             else:
55                 print(f"Warning: '{original_format}' not found in {cleaned_file_path}")
56
57             updated_entries.append(entry)
58
59         # Save the updated entries to a new JSON file in the verified_json directory
60         verified_json_file_path = os.path.join(verified_json_dir, filename)
61         with open(verified_json_file_path, 'w', encoding='utf-8') as verified_file:
62             json.dump(updated_entries, verified_file, ensure_ascii=False, indent=4)
63

```

Figure 2: Code to filter only the correct dates and fix the indices.

8 Next Week’s Proposal

Refer to section 1.2 for details (avoid repetition).

References

[CFM⁺20] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. Legal-bert: The muppets straight out of law school, 2020.

```
(haag-nlp) → week8 git:(main) x open verified_json
(haag-nlp) → week8 git:(main) x python correct_dates.py
Warning: '24 de febrero de 2023' not found in ../week7/cleaned/section_577-593_cleaned.txt
Warning: '18, 19 de mayo de 2023' not found in ../week7/cleaned/section_626-638_cleaned.txt
Warning: '1 de abril de 2023' not found in ../week7/cleaned/section_626-638_cleaned.txt
Warning: 'quince (15) del mes de enero del año dos mil veinte (2020)' not found in ../week7/cleaned/section_107-114_cleaned.txt
Warning: 'cuatro (4) del mes de septiembre del año dos mil veinte (2020)' not found in ../week7/cleaned/section_1282-1295_cleaned.txt
Warning: '6 de diciembre de 2022' not found in ../week7/cleaned/section_1282-1295_cleaned.txt
Warning: '7 de marzo de 2017' not found in ../week7/cleaned/section_1182-1193_cleaned.txt
Warning: '17 de agosto del 2023' not found in ../week7/cleaned/section_1008-1015_cleaned.txt
Warning: 'once del mes de julio del año dos mil veintidós (11/07/2022)' not found in ../week7/cleaned/section_1371-1377_cleaned.txt
Warning: '30-9-2021' not found in ../week7/cleaned/section_594-598_cleaned.txt
Warning: '21 de octubre de 2022' not found in ../week7/cleaned/section_219-225_cleaned.txt
Warning: '23 de mayo de 2022' not found in ../week7/cleaned/section_473-482_cleaned.txt
Warning: 'dieciséis (16) de enero del dos mil dieciocho (2018)' not found in ../week7/cleaned/section_500-512_cleaned.txt
Warning: '23 de junio de 2023' not found in ../week7/cleaned/section_522-532_cleaned.txt
Warning: '12 de septiembre del año 2017' not found in ../week7/cleaned/section_213-218_cleaned.txt
Warning: '25/09/2022' not found in ../week7/cleaned/section_680-687_cleaned.txt
Warning: 'veintiuno (21) del mes de noviembre del año dos mil veintidós (2022)' not found in ../week7/cleaned/section_1163-1167_cleaned.txt
Warning: '7 de septiembre de 2023' not found in ../week7/cleaned/section_688-696_cleaned.txt
Warning: 'veinticinco (25) del mes de septiembre del año dos mil dieciocho (2018)' not found in ../week7/cleaned/section_51-56_cleaned.txt
Warning: '31 de ENERO DE 2024' not found in ../week7/cleaned/section_666-675_cleaned.txt
Warning: 'treinta y uno (31) del mes de julio del año dos mil diecisiete (2017)' not found in ../week7/cleaned/section_115-120_cleaned.txt
Warning: 'veinte y uno (21) del mes de diciembre del año dos mil diez y ocho (2018)' not found in ../week7/cleaned/section_533-544_cleaned.txt
Warning: '29 de septiembre de 2021' not found in ../week7/cleaned/section_533-544_cleaned.txt
Warning: '8 de noviembre de 2023' not found in ../week7/cleaned/section_236-241_cleaned.txt
Warning: '12 de septiembre del año 2017' not found in ../week7/cleaned/section_207-212_cleaned.txt
Warning: '10 de julio del año 2015' not found in ../week7/cleaned/section_1452-1457_cleaned.txt
Warning: '15 de julio del año 2015' not found in ../week7/cleaned/section_1452-1457_cleaned.txt
Warning: 'treinta (30) del mes de noviembre del año dos mil veintiuno (2021)' not found in ../week7/cleaned/section_1472-1477_cleaned.txt
Warning: '26 de noviembre de 2019' not found in ../week7/cleaned/section_347-357_cleaned.txt
Warning: '17 de enero de 2022' not found in ../week7/cleaned/section_1328-1334_cleaned.txt
Warning: 'veinte (20) del mes de septiembre del año dos mil veintidós (2022)' not found in ../week7/cleaned/section_1347-1352_cleaned.txt
Warning: 'veintidós (22) del mes de abril del año dos mil veintidós (2022)' not found in ../week7/cleaned/section_1347-1352_cleaned.txt
Warning: '31 de enero de 2024' not found in ../week7/cleaned/section_1077-1084_cleaned.txt
Warning: '31 de julio de 2018' not found in ../week7/cleaned/section_1060-1067_cleaned.txt
Warning: Invalid indices format in file section_970-978_cleaned.json: 54
Warning: Invalid indices format in file section_970-978_cleaned.json: 245
Warning: Invalid indices format in file section_970-978_cleaned.json: 1406
Warning: Invalid indices format in file section_970-978_cleaned.json: 1535
Warning: Invalid indices format in file section_970-978_cleaned.json: 2465
Warning: Invalid indices format in file section_970-978_cleaned.json: 2576
Warning: Invalid indices format in file section_970-978_cleaned.json: 2664
Warning: Invalid indices format in file section_970-978_cleaned.json: 2721
Warning: Invalid indices format in file section_970-978_cleaned.json: 2842
Warning: Invalid indices format in file section_970-978_cleaned.json: 3788
Warning: '3 de diciembre de 2017' not found in ../week7/cleaned/section_570-576_cleaned.txt
Warning: '31 de enero de 2024' not found in ../week7/cleaned/section_1496-1501_cleaned.txt
Warning: 'veintiocho (28) de septiembre del año dos mil veintidós (2022)' not found in ../week7/cleaned/section_1496-1501_cleaned.txt
Warning: '10 de mayo de 2023' not found in ../week7/cleaned/section_1353-1365_cleaned.txt
Warning: '20 de septiembre de 2014' not found in ../week7/cleaned/section_1118-1128_cleaned.txt
Warning: '21 de septiembre del 2023' not found in ../week7/cleaned/section_823-828_cleaned.txt
Warning: 'veintiuno (21) de marzo del año dos mil veintidós (2022)' not found in ../week7/cleaned/section_936-944_cleaned.txt
Warning: 'dos (02) de mayo del año dos mil tres (2003)' not found in ../week7/cleaned/section_936-944_cleaned.txt
Warning: 'ocho (08) de enero de dos mil veintiuno (2021)' not found in ../week7/cleaned/section_847-852_cleaned.txt
Warning: '31 de ENERO de 2024' not found in ../week7/cleaned/section_483-491_cleaned.txt
Warning: 'veintidós (22) de agosto del año dos mil veintidós (2022)' not found in ../week7/cleaned/section_1534-1540_cleaned.txt
Warning: 'dieciséis (16) del mes de julio del año dos mil diecinueve (2019)' not found in ../week7/cleaned/section_697-705_cleaned.txt
Warning: '30 del mes de Mayo del año Dos Mil Veintitrés (2023)' not found in ../week7/cleaned/section_829-840_cleaned.txt
Warning: '27 de octubre de 2023' not found in ../week7/cleaned/section_1208-1217_cleaned.txt
Warning: '31 de ENERO DE 2024' not found in ../week7/cleaned/section_455-464_cleaned.txt
Warning: '13 de noviembre de 2009' not found in ../week7/cleaned/section_1136-1146_cleaned.txt
Warning: '15 de junio del año 2022' not found in ../week7/cleaned/section_545-550_cleaned.txt
```

Figure 3: Execution of code processing sentencias texts

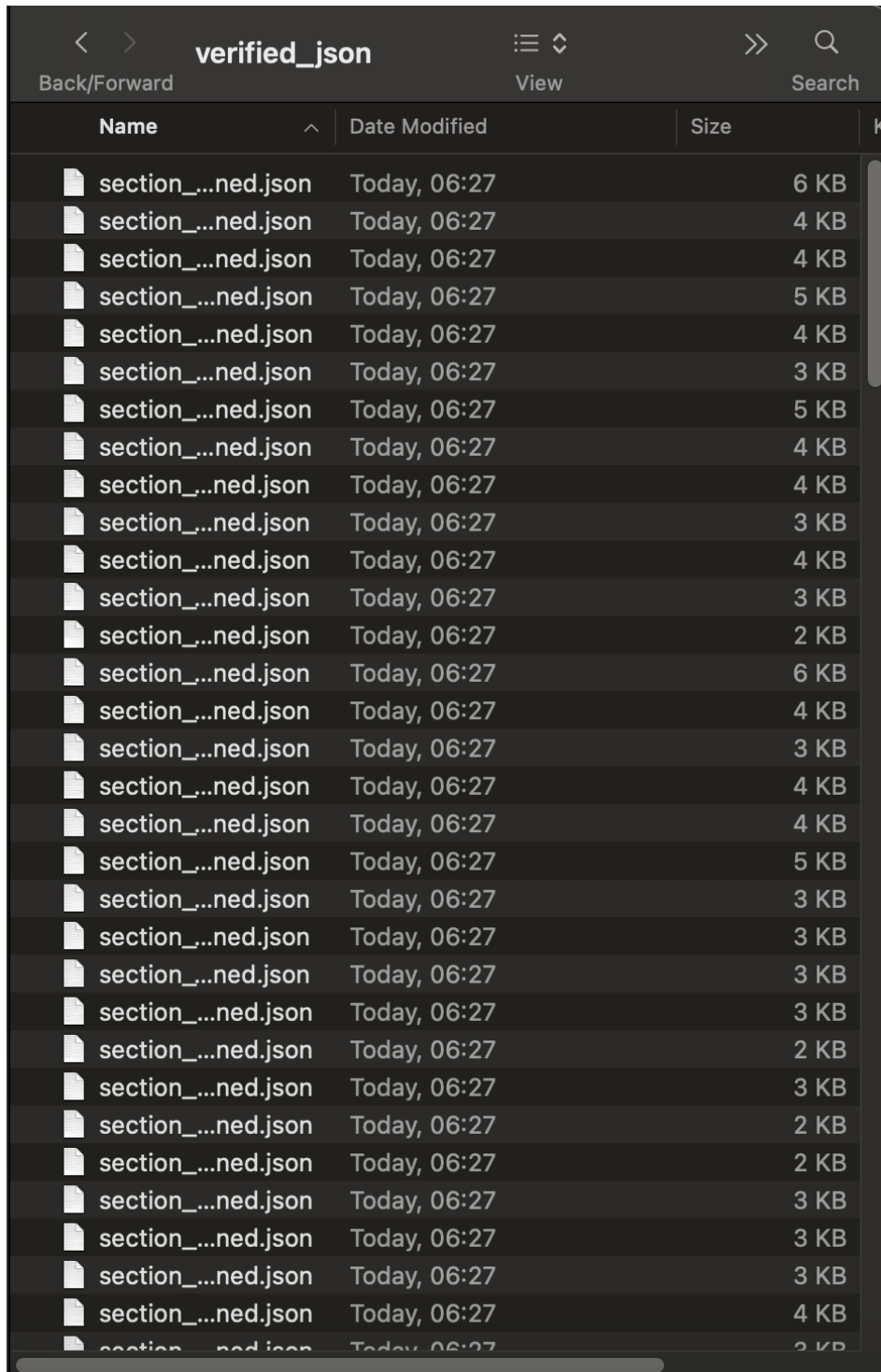


Figure 4: Resulting folder with JSON files

SENTENCIA DEL 31 DE ENERO DE 2024, NÚM. SCI-PS-24-0001

Sentencia impugnada:	Tercera Sala de la Cámara Civil y Comercial de la Corte de Apelación del Distrito Nacional, del 31 de marzo de 2023.
Materia:	Civil.
Recurrente:	Aimé Josefina Grand.
Abogado:	Lic. Juan F. De Jesús M.
Recurridos:	Asociación Cibao de Ahorros y Préstamos y compañías.
Abogados:	Licda. Olga María Veras L. y Lic. Nardo Augusto Matos Beltré.

Jueza ponente: *Pilar Jiménez Ortiz.*

Decisión: Declara Caducidad.



EN NOMBRE DE LA REPÚBLICA

La PRIMERA SALA DE LA SUPREMA CORTE DE JUSTICIA, competente para conocer de los recursos de casación en materia civil y comercial, regularmente constituida por los jueces Pilar Jiménez Ortiz, presidente, Justliniano Montero Montero, Samuel Arias Arzeno y Vanessa Acosta Peralta, miembros, asistidos del secretario general, en la sede de la Suprema Corte de Justicia, ubicada en Santo Domingo de Guzmán, Distrito Nacional, en fecha **31 de enero de 2024**, año 180º de la Independencia y año 161º de la Restauración, dicta la siguiente sentencia:

En ocasión del recurso de casación interpuesto por la señora Aimé Josefina Grand, quien tiene como abogado apoderado al Lcdo. Juan F. De Jesús M.; de generales que constan en el expediente.

3

www.poderjudicial.gob.do

BOLETÍN JUDICIAL NÚM. 1358 • PRIMERA SALA • SUPREMA CORTE DE JUSTICIA

En este proceso figuran como partes recurridas **a)** Asociación Cibao de Ahorros y Préstamos, debidamente representada por su presidente ejecutivo, José Luis Ventura Castañón, quien tiene como abogado

Figure 5: Original Sentencia sample file


```
sentencias / karol / week8 / verified_json / section_1008-1015_cleaned.json 🔗

Karol Gutierrez Add w8 code

Code Blame 164 lines (164 loc) · 4.77 KB

1  [
2  {
3      "standard_format": "2024/01/31",
4      "original_format": "31 de enero de 2024",
5      "indices": [
6          1078,
7          1097
8      ],
9      "context": "SENTENCIA DEL 31 DE ENERO DE 2024, NÚM. SCJ-PS-24-0118"
10 },
11 {
12     "standard_format": "2023/08/17",
13     "original_format": "17 de agosto de 2023",
14     "indices": [
15         257,
16         277
17     ],
18     "context": "sentencia impugnada de Cámara Civil y Comercial de la Corte de Apelación de San Pedro de Macorís"
19 },
20 {
21     "standard_format": "2023/09/13",
22     "original_format": "13 de septiembre de 2023",
23     "indices": [
24         3121,
25         3145
26     ],
27     "context": "el memorial de casación depositado el 13 de septiembre de 2023"
28 },
29 {
30     "standard_format": "2023/09/14",
31     "original_format": "14 de septiembre de 2023",
32     "indices": [
33         3299,
34         3323
35     ],
36     "context": "el acto de emplazamiento núm. 480/2023, de fecha 14 de septiembre de 2023"
37 },
38 ]
```

Figure 6: Cleaned and corrected version of JSON file

```
sentencias / dates_normalized / section_970-978_cleaned.json
Code Blame 62 lines (62 loc) · 1.49 KB
23     "indices": 1535,
24     "context": 1553
25   },
26   {
27     "standard_format": "2023/03/28",
28     "original_format": "28 de marzo de 2023",
29     "indices": 2465,
30     "context": 2483
31   },
32   {
33     "standard_format": "2023/03/31",
34     "original_format": "31 de marzo de 2023",
35     "indices": 2576,
36     "context": 2594
37   },
38   {
39     "standard_format": "2023/05/10",
40     "original_format": "10 de mayo de 2023",
41     "indices": 2664,
42     "context": 2681
43   },
44   {
45     "standard_format": "2023/01/17",
46     "original_format": "17 de enero de 2023",
47     "indices": 2721,
48     "context": 2739
49   },
50   {
51     "standard_format": "2023/01/17",
52     "original_format": "17 de enero de 2023",
53     "indices": 2842,
54     "context": 2860
55   },
56   {
57     "standard_format": "2023/03/31",
58     "original_format": "31 de marzo de 2023",
59     "indices": 3788,
60     "context": 3806
61   }
62 ]
```

Figure 7: Effective filtering by detecting that the indexes are not correct for this generated file

Week 8 Research Report

Thomas Orth (NLP Summarization / NLP Gen Team)

October 2024

0.1 What did you work on this week?

1. Presented midterm presentation to Clearinghouse group which received positive results.
2. Finished domain specific chain of thought approach and began examining summaries.
3. Sent summaries to interview team for validation.

0.2 What are you planning on working on next?

1. Continue reviewing summaries from domain specific approach
2. Start working with commercial LLM models
3. Look into scaling LLM experiments with text chunking

0.3 Is anything blocking you from getting work done?

1. None

1 Abstracts

- Title: LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models. Conference / Venue: Arxiv. Link: <https://arxiv.org/abs/2308.11462>
- Abstract: The advent of large language models (LLMs) and their adoption by the legal community has given rise to the question: what types of legal reasoning can LLMs perform? To enable greater study of this question, we present LegalBench: a collaboratively constructed legal reasoning benchmark consisting of 162 tasks covering six different types of legal reasoning. LegalBench was built through an interdisciplinary process, in which we collected tasks designed and hand-crafted by legal professionals. Because these subject matter experts took a leading role in construction, tasks

either measure legal reasoning capabilities that are practically useful, or measure reasoning skills that lawyers find interesting. To enable cross-disciplinary conversations about LLMs in the law, we additionally show how popular legal frameworks for describing legal reasoning – which distinguish between its many forms – correspond to LegalBench tasks, thus giving lawyers and LLM developers a common vocabulary. This paper describes LegalBench, presents an empirical evaluation of 20 open-source and commercial LLMs, and illustrates the types of research explorations LegalBench enables.

- Summary: This paper goes over an open approach to evaluating the effectiveness of LLMs in the Legal domain. At the time of this technical report, GPT-4 was the best model across all domains
- Relevance: LegalBench may be a way to understand the extent finetuning for summarization has on the legal understanding of an LLM.

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

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/g3hepr11vzamua0gwdkhz5k2r34ocgwt>
3. Thomas-Orth/domain_specific_scot.py
 - Brief Description: Run a domain specific version of Summary Chain-of-thought (CoT) on complaints.
 - 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

(c) Third Block: Evaluate via manual inspection

- Screenshot of code:

```
df = pd.read_csv("/Users/thomasorth/law-clearinghouse-ocr/parsed_documents.csv", sep="|")
doc = df["document"].iloc[0]

def generate_prompt(document):
    prompt = """
(CASE: "{document}")
You are a law student, skilled in retrieving case information, who is tasked with identifying major entities in the above complaint case. You are
1. The filing date. If no explicit date is given but a year is, provide the year only.
2. Full name of the court where the case was filed e.g. "U.S. District Court for the District of New York". This should include the state district
3. The name and title of the Judge. Example: District Judge J. Paul Oetken.
4. Type of counsel (private, legal services, state protection & advocacy system, ACLU, etc.). Please identify legal service organizations by name
5. Indicate if this is a class action lawsuit or if it involves individual plaintiffs. Do not name any attorneys as plaintiffs.
6. Who are the defendants?
7. Who are the plaintiffs? If plaintiffs are not an organization, just describe them.
8. The plaintiffs legal claims which includes: The Statutory or constitutional basis for claim. If there is a state law claim, note the state.
9. As part of the remedies for the case, was injunctive relief sought? If so, describe the injunctive relief sought in relation to any judgement.
10. As part of the remedies for the case, was Declaratory relief sought?
11. As part of the remedies for the case, was money damages sought?
Provide only the requested entities in this list above.
"""
    return prompt

def generate_reduce_cot_prompt(context):
    prompt = """
CONTEXT: "{context}"
By using the above context, summarize the case in paragraph format into a concise by informative summary.
The format of the paragraph should be the following:
1. The first sentence should be: "This is a case about <factual info> where <factual info> is the factual background of the case, from the context
2. The second sentence should be: "On <date>, <plaintiffs> filed this lawsuit in <courts> where <date> is the date provided by the context, <plaintiff
3. The third sentence should be: "<plaintiffs> sued <defendants> under <statute> where <plaintiffs> are the plaintiffs in the case from the case
4. The fourth sentence should be: "Represented by <counsel>, <plaintiffs> sought <remedy> where <counsel> is the counsel representing the plaintiff
5. The last sentence should be: "They claim that <claim> where claim is the legal claim that the plaintiffs are making.
Please follow the above format when making the paragraph. Only provide the requested summary, no other text:
"""
    return prompt

prompt = generate_prompt(doc)
llm = ChatOllama(model="llama3.2", temperature=0.000000001)
a1_msg = llm.invoke(prompt)
a1_msg_sum = llm.invoke(generate_reduce_cot_prompt(a1_msg.content))
print(" ".join(a1_msg_sum.content.split("\n")))
```

Figure 1: Domain Summary CoT

4. Flow Diagram:



Figure 2: Flow diagram

5. Running scripts:

- (a) Download the script, the csv from the box link and llm.requirements.txt
- (b) Install ollama: <https://ollama.com/download>
- (c) To pull and run llama 3.2, run: ollama run llama3.2
- (d) Run: python -m pip install -r llm.requirements.txt
- (e) 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:

This is a case about relatives and visitors to those incarcerated at the Tennessee Correctional Complex (TCDC) filing a class action lawsuit against the TCDC in 1987. On November 18, 1987, relatives and visitors to those incarcerated at the TCDC filed this lawsuit in the U.S. District Court for the District of Maryland. Represented by ACLU, National Prison Project, plaintiffs sought permanent injunctions against Defendants' policies, practices, acts, and omissions, as well as declaratory relief stating that these policies violate their rights guaranteed by the First, Sixth, Eighth, and Fourteenth Amendments. They claim that the TCDC's denial of access to courts in violation of the Sixth and Fourteenth Amendments violates their constitutional rights.

This summary provides more of the information that the clearinghouse would ask for and look for in a summary from a first year law student.

5.2 Known Limitations

Llama 3.2 3b seems to have trouble with acronyms through testing. Commercial models or larger LLMs may deal with this better. So investigation is needed.

6 Answers to Higher Ed Feedback

6.1 Clarifying Questions Response

For the question about accuracy evaluation, primarily we've tried a metric approach with things such as ROUGE, BLEU, METEOR and BERT SCORE but such metrics focus on checking on overlapping n-grams for the most part. Given that LLM based summarization is zero-shot, it can capture the facts of a complaint case but won't have the same writing style. We might more directly be able use metrics if we finetune LLMs for this task.

Right now, we're doing an expert evaluation process by providing summaries to a group of students trained in legal summarization to get an evaluation of the summaries made.

6.2 Data Visualization

So I don't have additional data visualizations for the processes but I can include the prompts that were used for each approach.

6.2.1 Summary Chain-of-thought Prompts

```
CASE: ```{document}```
For the above court case:
What are the important acronyms? If the document says what the acronym means, include that information. Otherwise, include only the acronym.
What are the important entities in this document?
What are the important dates in this document?
What events are happening in this document?
What is the result of these events?
Please answer the above questions. Use only the document as context for answering:
"""
return prompt
```

Figure 3: 1st Prompt used

```
prompt = f"""
CASE: ```{document}```
CONTEXT: ```{context}```
By using the above context, summarize the case in paragraph format into a concise by informative summary.
If a date is mentioned, mention that in the summary.
If no date is found, do not mention it. Do not make note of any content you decided to exclude.
Only use the information provided in the case and the content to create the summary.
Only provide the summary, do not include any text outside the summary.
Provide the summary in past tense, do not use present tense.
If the document does not spell out an acronym, do not spell it out yourself or make an inference about what the acronym could mean:
"""
return prompt
```

Figure 4: 2nd Prompt used

```
prompt = f"""
SUMMARY: ```{summary}```
Based on the provided summary, please change the tense to past tense. Please only provide back the corrected summary, no other text:
"""
return prompt
```

Figure 5: 3rd Prompt used

Here is a more detailed flow chart about Summary Chain-of-thought

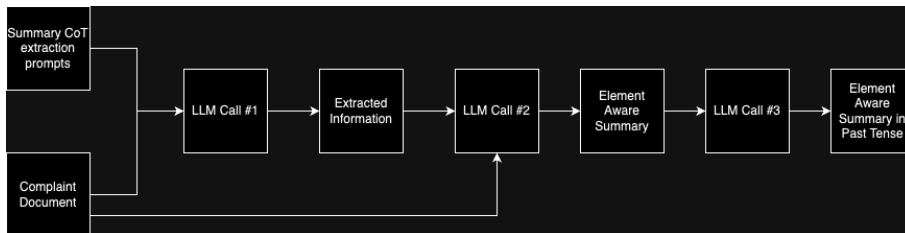


Figure 6: Data flow for Summary Chain-of-thought

6.2.2 Entity Extraction

```
prompt = f"""
CASE: ``{document}``
You are a law student, skilled in retrieving case information, who is tasked with identifying major entities in the above complaint case.
You are to find the listed entities below:
1. The filing date. If no explicit date is given but a year is, provide the year only.
2. Full name of the court where the case was filed e.g. "U.S. District Court for the District of New York"
This should include the state district that this course is taking place in.
3. The name and title of the Judge. Example: District Judge J. Paul Oetken.
4. Type of counsel. Such as: private, legal services, state protection & advocacy system, ACLU, etc. Do not list organizations or names.
5. Indicate if this is a class action lawsuit or if it involves individual plaintiffs. Do not name any attorneys as plaintiffs.
6. Who are the defendants?
7. Who are the plaintiffs? If plaintiffs are not an organization, just describe them.
8. The plaintiffs legal claims which includes: The Statutory or constitutional basis for claim. If there is a state law claim, note the state.
9. As part of the remedies for the case, was injunctive relief sought? If so, describe the injunctive relief sought in relation to any judgement.
10. As part of the remedies for the case, was Declaratory relief sought?
11. As part of the remedies for the case, was Attorney fees sought and how much?
12. As part of the remedies for the case, was money damages sought? If so, what kind?
Provide only the requested entities in this list above.
"""
```

Figure 7: Prompt used

6.3 Understanding as a Non-Expert

So Summary Chain-of-thought is a general approach to creating more element aware summaries to include facts in the output. However, these general questions don't include everything that the Clearinghouse is looking for.

That was where the extract relevant info flow comes in. This was an attempt to ask more pointed questions that the clearinghouse is interested in. Once this was validated, this would evolve into a domain specific version of Summary Chain-of-thought.

Validation of accuracy for these preliminary results have been reviewing the document and human generated summaries to compare to AI outputs to check for factual accuracy.