HAAG NLP Summarization Week 14

Michael Bock

November 2024

1 Slack Questions

What did you accomplish this week?

- Found a bug where I accidentally trained on validation data, so will need new results
- Thought I had tfidf working well in PyTorch
- Started on end of semester report outline
- Provided data to undergrads trying to validate labels

What are you planning on working on next?

- Make a single issue training pipeline that is less prone to bugs.

What is blocking you from progressing?

- None

2 Abstract

Learning to store information over extended time intervals by recurrent backpropagation takes a very long time, mostly because of insufficient, decaying error backflow. We briefly review Hochreiter's (1991) analysis of this problem, then address it by introducing a novel, efficient, gradient based method called long short-term memory (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete-time steps by enforcing constant error flow through constant error carousels within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is O. 1. Our experiments with artificial data involve local, distributed, real-valued, and noisy pattern representations. In comparisons with real-time recurrent learning, back propagation through time, recurrent cascade correlation, Elman nets, and neural sequence chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long-time-lag tasks that have never been solved by previous recurrent network algorithms.

2.1 Brief Analysis

LSTMs are a sequence network like RNNs. They accumulate an infinite sequence into a hidden state for inference. This can be useful for our model because TFIDF's main attribute was that it took a full legal document to a fixed vector.

3 Scripts and Code Blocks

```
import os
import ast
import pandas as pd
from labels import DNO_ISSUES, PARENTS
from datasets import Dataset
from transformers import AutoTokenizer
from peft import prepare_model_for_kbit_training, AutoPeftModel
from peft import LoraConfig, get_peft_model
import datetime
import os
from transformers import AutoTokenizer
from transformers import AutoModelForSequenceClassification, AutoModel
from transformers import TrainingArguments, Trainer, BitsAndBytesConfig
import numpy as np
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score
from sklearn.model_selection import StratifiedKFold
from torch.utils.data import Dataset, DataLoader
from torch import nn
from torch import optim
import torch
from sklearn.utils.class_weight import compute_class_weight
from accelerate import Accelerator
from sklearn.feature_extraction.text import TfidfVectorizer
from transformers import PreTrainedModel, PretrainedConfig
from tqdm import tqdm
from torch.utils.tensorboard import SummaryWriter

torch.manual_seed(0)

target_label = 'Exclusion issues'

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

df = pd.read_csv('dno_labels.csv').dropna()

df['word_count'] = df['Text'].apply(lambda x: len(x))
df = df[df['word_count'] >= 2000]

df = df.drop(columns=['word_count'])

class TfIdf(Dataset):
    def __init__(self, vectorizer, df, target_class):
        self.df = df
        self.vectorizer = vectorizer
        self.X = self.vectorizer.fit_transform(self.df['Text']).toarray().astype(np.
    float32)
        self.target_label = target_class
```

```
48
49      def __len__(self):
50          return len(self.df)
51
52      def __getitem__(self, idx):
53          return self.X[idx], self.df[self.target_label]
54
55  class MLP(nn.Module):
56      def __init__(self, input_size, hidden_size, output_size, num_hidden_layers):
57          super(MLP, self).__init__()
58
59          layers = []
60          # Input layer
61          layers.append(nn.Linear(input_size, hidden_size))  # First hidden layer
62
63          # Hidden layers
64          for _ in range(num_hidden_layers - 1):
65              layers.append(nn.ReLU())  # Activation function
66              layers.append(nn.Linear(hidden_size, hidden_size))  # Next hidden layer
67
68          layers.append(nn.ReLU())  # Activation function for last hidden layer
69          layers.append(nn.Linear(hidden_size, output_size))  # Output layer
70          layers.append(nn.Sigmoid())  # Softmax activation for multi-class
      classification
71
72
73
74          # Create the model using nn.Sequential
75          self.model = nn.Sequential(*layers)
76
77      def forward(self, x):
78          return self.model(x)
79
80  import torch
81  from torch.utils.data import random_split, DataLoader
82
83  def train_single_class(target_label):
84      vectorizer = TfidfVectorizer(sublinear_tf=True, max_df=0.5, min_df=5, stop_words
      ="english")
85
86      dataset = TfIdf(vectorizer, df, target_class = target_label)
87
88      # Assuming you have your dataset loaded as 'dataset'
89      dataset_size = len(dataset)
90      train_size = int(0.8 * dataset_size)
91      test_size = dataset_size - train_size
92
93      train_dataset, test_dataset = random_split(dataset, [train_size, test_size])
94
95      model = MLP(input_size = train_dataset[0][0].shape[-1], hidden_size = 128,
      output_size = 1, num_hidden_layers = 3)
96      model.to(device)
97
98      criterion = nn.BCELoss()  # Binary Cross-Entropy Loss
99      optimizer = optim.Adam(model.parameters())
100
101     train_loader = DataLoader(train_dataset, batch_size=64)
102
```

```
    epochs = 10
    model.train()

    for epoch in range(epochs):
        train_preds = []
        train_trues = []
        for batch_x, batch_y in tqdm(train_loader, total = len(train_loader)):

            batch_x = batch_x.to(device)
            batch_y = batch_y.to(device)

            optimizer.zero_grad()
            outputs = model(batch_x)
            loss = criterion(outputs, batch_y.unsqueeze(1))
            loss.backward()
            optimizer.step()

            for output, y in zip(outputs, batch_y):
                print(output.item(), y.item())

    model.eval()
    for batch_x, batch_y in tqdm(train_loader, total = len(train_loader)):
        outputs = model(batch_x).detach()



if __name__ == '__main__':
    train_single_class('Exclusion issues')
```

4 Documentation

I know that the model was training on validation data because when I removed all the layers the F1 score was 1. That's really suspicious, then I found the exact lines (which I have now removed) where the training was happening. However, I have a notebook with tfidf running with sklearn that I presented last week, so I know that TFIDF provides a good baseline. I think this time I will train the same way I trained on the sklearn version, which was to have separate networks for each issue. This is just less prone to bugs, in reality I bet the features learned in each network are similar.

```
    ...

    model.eval()
    val_loss = 0.0
    val_cm = {k: np.zeros((2, 2)) for k in DNO_ISSUES}
    for inputs, labels in tqdm(val_loader, total = len(val_loader)):

        outputs = model(inputs.to(device))

        loss = criterion(outputs, labels.to(device))

        optimizer.zero_grad() #<--- Problem lines
        loss.backward() #<--- Problem lines
        optimizer.step() #<--- Problem lines
        for output, label in zip(outputs, labels):
            for i, (cls, issue) in enumerate(zip(output, label)):
                pred = int(torch.nn.functional.sigmoid(cls).item() > 0.5)
```

```
                true = int(issue.item())
                val_cm[DNO_ISSUES[i]][true, pred] += 1
        val_loss += loss.item()

    ...
```

5 Script Validation(Optional)

N/A, not working yet

6 Results Visualization

```
    ...

    model.eval()
    val_loss = 0.0
    val_cm = {k: np.zeros((2, 2)) for k in DNO_ISSUES}
    for inputs, labels in tqdm(val_loader, total = len(val_loader)):

        outputs = model(inputs.to(device))

        loss = criterion(outputs, labels.to(device))

        optimizer.zero_grad() #<--- Problem lines
        loss.backward() #<--- Problem lines
        optimizer.step() #<--- Problem lines
        for output, label in zip(outputs, labels):
            for i, (cls, issue) in enumerate(zip(output, label)):
                pred = int(torch.nn.functional.sigmoid(cls).item() > 0.5)
                true = int(issue.item())
                val_cm[DNO_ISSUES[i]][true, pred] += 1
        val_loss += loss.item()

    ...
```

```
/var/lib/slurm/slurmd/job965255/slurm_script: line 11: pip: command not found
/usr/bin/python: No module named pip
  0%|          | 0/43 [00:00<?, ?it/s]
Traceback (most recent call last):
  File "/storage/ice1/5/9/mbock9/law-data-design-vip/ner/llamas/tfidf_single_class.
    py", line 130, in <module>
    train_single_class('Exclusion issues')
  File "/storage/ice1/5/9/mbock9/law-data-design-vip/ner/llamas/tfidf_single_class.
    py", line 109, in train_single_class
    for batch_x, batch_y in tqdm(train_loader, total = len(train_loader)):
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/tqdm/std.py", line
    1181, in __iter__
    for obj in iterable:
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/utils/data/
    dataloader.py", line 631, in __next__
    data = self._next_data()
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/utils/data/
    dataloader.py", line 675, in _next_data
    data = self._dataset_fetcher.fetch(index)  # may raise StopIteration
```

```
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/utils/data/
    _utils/fetch.py", line 54, in fetch
    return self.collate_fn(data)
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/utils/data/
    _utils/collate.py", line 277, in default_collate
    return collate(batch, collate_fn_map=default_collate_fn_map)
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/utils/data/
    _utils/collate.py", line 144, in collate
    return [collate(samples, collate_fn_map=collate_fn_map) for samples in
    transposed]  # Backwards compatibility.
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/utils/data/
    _utils/collate.py", line 144, in <listcomp>
    return [collate(samples, collate_fn_map=collate_fn_map) for samples in
    transposed]  # Backwards compatibility.
  File "/home/hice1/mbock9/.local/lib/python3.9/site-packages/torch/utils/data/
    _utils/collate.py", line 152, in collate
    raise TypeError(default_collate_err_msg_format.format(elem_type))
TypeError: default_collate: batch must contain tensors, numpy arrays, numbers, dicts
    or lists; found <class 'pandas.core.series.Series'>
```

7 Proof of work

It's currently not liking me trying to store a pandas object in a pytorch dataset I think? I don't know why this is a problem yet. But basically when I want to do is put the tfidf vectorizer into the dataset object so that I don't accidentally train on validation data.

8 Next Week's proposal

- Make a single issue training pipeline that is less prone to bugs.

HAAG Research Report
NLP - Sentencias / NLP - Gen Team
Week 14

Víctor C. Fernández
November 2024

1 WEEKLY PROJECT UPDATES

What progress did you make in the last week?

- Created scaffolding for conference paper.
- Adapted code to extract dates and context for existing sentencias
- Prepared data structure for labeling dates for further model tuning.
- Worked on a pipeline to orchestrate the whole process using prefect.

What progress are you making next?

- Completing the content of the paper.
- Complete full implementation of a simple pipeline to run all processes end to end.
- Document and upload all code to the Sentencias private folder.

Is there anything blocking you from making progress?

No, no blockers right now.

2 ABSTRACTS

1. **Title:** Incorporating domain knowledge for extractive summarization of legal case documents
   - **URL:** https://dl.acm.org/doi/10.1145/3462757.3466092 (may require accessing with GaTech credentials to view actual paper)

   - **Abstract:** Automatic summarization of legal case documents is an important and practical challenge. Apart from many domain-independent text summarization algorithms that can be used for this purpose, several algorithms have been developed specifically for summarizing legal case documents. However, most of the existing algorithms do not systematically incorporate domain knowledge that specifies what information should ideally be present in a legal case document summary. To address this gap, we propose an unsupervised summarization algorithm DELSumm which is designed to systematically incorporate guidelines from legal experts into an optimization setup. We conduct detailed experiments over case documents from the Indian Supreme Court. The experiments show that our proposed unsupervised method outperforms several strong baselines in terms of ROUGE scores, including both general summarization algorithms and legal-specific ones. In fact, though our proposed algorithm is unsupervised, it outperforms several supervised summarization models that are trained over thousands of document-summary pairs.

   - **Summary:** The paper introduces DELSumm, an unsupervised extractive summarization algorithm for legal case documents. DELSumm integrates legal domain knowledge from experts into an Integer Linear Programming (ILP) framework to create concise, balanced summaries of legal documents. It emphasizes segmenting and including key rhetorical sections such as facts, legal issues, statutes, and final judgments while minimizing redundancy. Tested on Indian Supreme Court case documents, the model demonstrated superior performance compared to baseline methods, including supervised deep learning models, in terms of ROUGE scores. The algorithm's robustness allows it to handle noisy rhetorical labels effectively, making it adaptable for various jurisdictions with appropriate legal guidelines.

- **Relevance:**This paper is directly applicable to the project on extracting dates and critical information from Dominican judicial decisions (sentencias). The emphasis on leveraging domain-specific knowledge to structure and summarize legal texts parallels the challenges in processing sentencias, where understanding the rhetorical structure is essential for extracting temporal and procedural information. The methods proposed in DELSumm, such as segment prioritization and the use of optimization frameworks, can inspire enhancements to date extraction workflows by ensuring critical segments containing dates are emphasized.

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.

For this week I've been focusing on extracting dates from the existing sentencias so we can define our own labels to be used with the last model to classify the dates from the sentencias. We are initially going to focus on labor cases.

1. Updated NER text extraction function to include context around dates from the chunks here.

```
def process_text(self, text):
        """Runs NER model on text and returns JSONL string."""
        try:
            chunks = self.split_text_with_overlap(text)
            all_predictions = []
            for chunk in chunks:
                preds = self.deployed_ner_pipeline(chunk)
                all_predictions.extend(preds)
                # Now we add the context to each prediction
                for pred in preds:
                    pred['context'] = chunk
            all_predictions =
            ↪  self.deduplicate_entities(all_predictions)

            formatted_output = {
                # "sentence": text,
                "entities": self.run_predictions(all_predictions)
            }

            return json.dumps(formatted_output)
```

Code 1—process_text function for NER class

2. Code for using the NER model to extract dates with their context into a csv file for later manual classification here.

```
import json
import os
import pandas as pd
from ner_processor import NerProcessor

cleaned_files_folder =
↪  "../../Documents/Generated/1.sentencias_cleaned"

ner_processor = NerProcessor()
df = pd.DataFrame(columns=['filename', 'date', 'score', 'start',
↪  'end', 'context'])
# We are going to iterate over all the cleaned files and extract
↪  the dates in each of them.
# Then we will store these in a file called filename_dates.json

for filename in os.listdir(cleaned_files_folder):
    with open(os.path.join(cleaned_files_folder, filename), 'r',
    ↪  encoding='utf-8') as file:
        # Print in green the filename being processed
        print(f'\033[92mProcessing {filename}\033[0m')
        cleaned_text = file.read()
        ner_output = ner_processor.process_text(cleaned_text)
        ner_json = json.loads(ner_output)
        dates = []
        for entity in ner_json['entities']:
            # We add one row to the dataframe for each date found
            ↪  without using the append method since it's
            ↪  deprecated
            df.loc[len(df)] = [filename, entity['word'],
            ↪  entity['score'], entity['start'], entity['end'],
            ↪  entity['context']]
```

```
# We now save the results into a csv file called dates.csv (if it
↪  exists, we append a new number to the filename as dates_1.csv,
↪  dates_2.csv, etc.)
if os.path.exists('dates.csv'):
    i = 1
    while os.path.exists(f'dates_{i}.csv'):
        i += 1
    df.to_csv(f'dates_{i}.csv', index=False)
else:
    df.to_csv('dates.csv', index=False)
```

Code 2—date extraction from NER model to csv file

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:

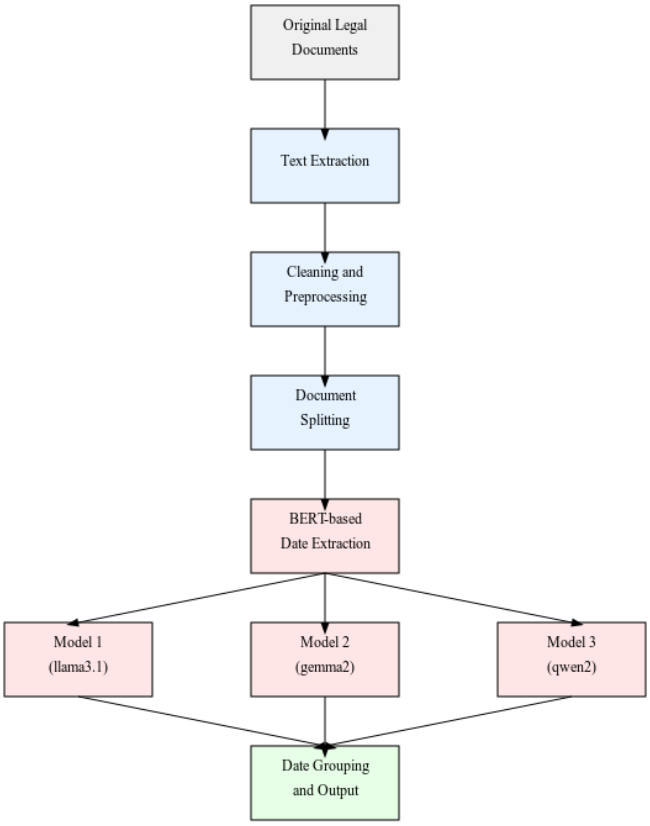
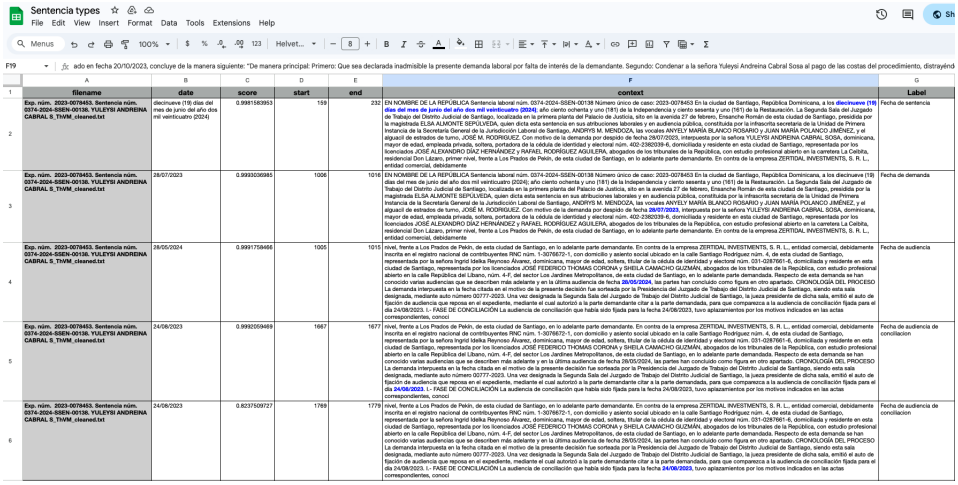


Figure 1—Full date extraction process

This week, my focus has been on generating adequate labels to be identified by the LLM when prompting with a specific date and the context for classifying such date. For this, I have extracted the dates in csv format allowing us to then manually label each of the dates. Link to the current spreadsheet is here.

Once we have successfully identified the labels based on the chunks of text retrieved for each date, we can then re-run the prompting with the existing models to verify the results and see if this labeling has made a significant difference.



	filename	date	score	start	end	context	Label
2	Exp. núm. 2023-0079453. Sentencia núm. 0374-2024-SSEN-00136. YULEYSI ANDREINA CABRAL S_ThVM_cleaned.txt	diecinueve (19) días del mes de junio del año dos mil veinticuatro (2024)	0.998158353	159	232	EN NOMBRE DE LA REPUBLICA Sentencia laboral núm. 0374-2024-SSEN-00136 Número único de caso: 2023-0078453 En la ciudad de Santiago, República Dominicana, a los diecinueve (19) días del mes de junio del año dos mil veinticuatro (2024); año ciento ochenta y uno (181) de la Independencia y ciento sesenta y uno (161) de la Restauración. La Segunda Sala del Juzgado de Trabajo del Distrito Judicial de Santiago, localizada en la primera planta del Palacio de Justicia, sito en la avenida 27 de febrero, Ensanche Román de esta ciudad de Santiago, presidida por la magistrada ELSA ALMONTE SEPÚLVEDA, quien dicta esta sentencia en sus atribuciones laborales y en audiencia pública, constituida por la infrascrita secretaria de la Unidad de Primera Instancia de la Secretaría General de la Jurisdicción Laboral de Santiago, ANDRYS M. MENDOZA, las vocales ANYELY MARÍA BLANCO ROSARIO y JUAN MARÍA POLANCO JIMÉNEZ, y el alguacil de estrados de turno, JOSÉ M. RODRIGUEZ, Con motivo de la demanda por despido de fecha 28/07/2023, interpuesta por la señora YULEYSI ANDREINA CABRAL SOSA, dominicana, mayor de edad, empleada privada, soltera, portadora de la cédula de identidad y electoral núm. 402-2360339-6, domiciliada y residente en esta ciudad de Santiago, representada por los licenciados JOSÉ ALEXANDRO DÍAZ HERNÁNDEZ y RAFAEL RODRÍGUEZ AGUILERA, abogados de los tribunales de la República, con estudio profesional abierto en la carretera La Cebiba, residencial Don Lázaro, primer nivel, frente a Los Prados de Pekín, de esta ciudad de Santiago, en lo adelante parte demandante. En contra de la empresa ZERTIDAL INVESTMENTS, S. R. L., entidad comercial, debidamente	Fecha de sentencia
3	Exp. núm. 2023-0079453. Sentencia núm. 0374-2024-SSEN-00136. YULEYSI ANDREINA CABRAL S_ThVM_cleaned.txt	28/07/2023	0.9993036985	1006	1016	EN NOMBRE DE LA REPUBLICA Sentencia laboral núm. 0374-2024-SSEN-00136 Número único de caso: 2023-0078453 En la ciudad de Santiago, República Dominicana, a los diecinueve (19) días del mes de junio del año dos mil veinticuatro (2024); año ciento ochenta y uno (181) de la Independencia y ciento sesenta y uno (161) de la Restauración. La Segunda Sala del Juzgado de Trabajo del Distrito Judicial de Santiago, localizada en la primera planta del Palacio de Justicia, sito en la avenida 27 de febrero, Ensanche Román de esta ciudad de Santiago, presidida por la magistrada ELSA ALMONTE SEPÚLVEDA, quien dicta esta sentencia en sus atribuciones laborales y en audiencia pública, constituida por la infrascrita secretaria de la Unidad de Primera Instancia de la Secretaría General de la Jurisdicción Laboral de Santiago, ANDRYS M. MENDOZA, las vocales ANYELY MARÍA BLANCO ROSARIO y JUAN MARÍA POLANCO JIMÉNEZ, y el alguacil de estrados de turno, JOSÉ M. RODRIGUEZ, Con motivo de la demanda por despido de fecha 28/07/2023, interpuesta por la señora YULEYSI ANDREINA CABRAL SOSA, dominicana, mayor de edad, empleada privada, soltera, portadora de la cédula de identidad y electoral núm. 402-2360339-6, domiciliada y residente en esta ciudad de Santiago, representada por los licenciados JOSÉ ALEXANDRO DÍAZ HERNÁNDEZ y RAFAEL RODRÍGUEZ AGUILERA, abogados de los tribunales de la República, con estudio profesional abierto en la carretera La Cebiba, residencial Don Lázaro, primer nivel, frente a Los Prados de Pekín, de esta ciudad de Santiago, en lo adelante parte demandante. En contra de la empresa ZERTIDAL INVESTMENTS, S. R. L., entidad comercial, debidamente	Fecha de demanda
4	Exp. núm. 2023-0079453. Sentencia núm. 0374-2024-SSEN-00136. YULEYSI ANDREINA CABRAL S_ThVM_cleaned.txt	28/05/2024	0.9991758466	1005	1015	nivel, frente a Los Prados de Pekín, de esta ciudad de Santiago, en lo adelante parte demandante. En contra de la empresa ZERTIDAL INVESTMENTS, S. R. L., entidad comercial, debidamente inscrita en el registro nacional de contribuyentes RNC núm. 1-3076672-1, con domicilio y asiento social ubicado en la calle Santiago Rodríguez núm. 4, de esta ciudad de Santiago, representada por la señora Ingrid Idelka Reynoso Álvarez, dominicana, mayor de edad, soltera, titular de la cédula de identidad y electoral núm. 031-0287661-6, domiciliada y residente en esta ciudad de Santiago, representada por los licenciados JOSÉ FEDERICO THOMAS CORONA y SHEILA CAMACHO GUZMÁN, abogados de los tribunales de la República, con estudio profesional abierto en la calle República del Líbano, núm. 4-F, del sector Los Jardines Metropolitanos, de esta ciudad de Santiago, en lo adelante parte demandada. Respecto de esta demanda se han conocido varias audiencias que se describen más adelante y en la última audiencia de fecha 28/05/2024, las partes han concluido como figura en otro apartado. CRONOLOGÍA DEL PROCESO La demanda interpuesta en la fecha citada en el motivo de la presente decisión fue sorteada por la Presidencia del Juzgado de Trabajo del Distrito Judicial de Santiago, siendo esta sala designada, mediante auto número 00777-2023. Una vez designada la Segunda Sala del Juzgado de Trabajo del Distrito Judicial de Santiago, la jueza presidente de dicha sala, emitió el auto de fijación de audiencia que reposa en el expediente, mediante el cual autorizó a la parte demandante citar a la parte demandada, para que comparezca a la audiencia de conciliación fijada para el día 24/08/2023, l.- FASE DE CONCILIACIÓN La audiencia de conciliación que había sido fijada para la fecha 24/08/2023, tuvo aplazamientos por los motivos indicados en las actas correspondientes, consci	Fecha de audiencia
5	Exp. núm. 2023-0079453. Sentencia núm. 0374-2024-SSEN-00136. YULEYSI ANDREINA CABRAL S_ThVM_cleaned.txt	24/06/2023	0.9992059469	1667	1677	nivel, frente a Los Prados de Pekín, de esta ciudad de Santiago, en lo adelante parte demandante. En contra de la empresa ZERTIDAL INVESTMENTS, S. R. L., entidad comercial, debidamente inscrita en el registro nacional de contribuyentes RNC núm. 1-3076672-1, con domicilio y asiento social ubicado en la calle Santiago Rodríguez núm. 4, de esta ciudad de Santiago, representada por la señora Ingrid Idelka Reynoso Álvarez, dominicana, mayor de edad, soltera, titular de la cédula de identidad y electoral núm. 031-0287661-6, domiciliada y residente en esta ciudad de Santiago, representada por los licenciados JOSÉ FEDERICO THOMAS CORONA y SHEILA CAMACHO GUZMÁN, abogados de los tribunales de la República, con estudio profesional abierto en la calle República del Líbano, núm. 4-F, del sector Los Jardines Metropolitanos, de esta ciudad de Santiago, en lo adelante parte demandada. Respecto de esta demanda se han conocido varias audiencias que se describen más adelante y en la última audiencia de fecha 28/05/2024, las partes han concluido como figura en otro apartado. CRONOLOGÍA DEL PROCESO La demanda interpuesta en la fecha citada en el motivo de la presente decisión fue sorteada por la Presidencia del Juzgado de Trabajo del Distrito Judicial de Santiago, siendo esta sala designada, mediante auto número 00777-2023. Una vez designada la Segunda Sala del Juzgado de Trabajo del Distrito Judicial de Santiago, la jueza presidente de dicha sala, emitió el auto de fijación de audiencia que reposa en el expediente, mediante el cual autorizó a la parte demandante citar a la parte demandada, para que comparezca a la audiencia de conciliación fijada para el día 24/08/2023. l.- FASE DE CONCILIACIÓN La audiencia de conciliación que había sido fijada para la fecha 24/08/2023, tuvo aplazamientos por los motivos indicados en las actas correspondientes, consci	Fecha de audiencia de conciliacion
6	Exp. núm. 2023-0079453. Sentencia núm. 0374-2024-SSEN-00136. YULEYSI ANDREINA CABRAL S_ThVM_cleaned.txt	24/08/2023	0.8237509727	1769	1779	nivel, frente a Los Prados de Pekín, de esta ciudad de Santiago, en lo adelante parte demandante. En contra de la empresa ZERTIDAL INVESTMENTS, S. R. L., entidad comercial, debidamente inscrita en el registro nacional de contribuyentes RNC núm. 1-3076672-1, con domicilio y asiento social ubicado en la calle Santiago Rodríguez núm. 4, de esta ciudad de Santiago, representada por la señora Ingrid Idelka Reynoso Álvarez, dominicana, mayor de edad, soltera, titular de la cédula de identidad y electoral núm. 031-0287661-6, domiciliada y residente en esta ciudad de Santiago, representada por los licenciados JOSÉ FEDERICO THOMAS CORONA y SHEILA CAMACHO GUZMÁN, abogados de los tribunales de la República, con estudio profesional abierto en la calle República del Líbano, núm. 4-F, del sector Los Jardines Metropolitanos, de esta ciudad de Santiago, en lo adelante parte demandada. Respecto de esta demanda se han conocido varias audiencias que se describen más adelante y en la última audiencia de fecha 28/05/2024, las partes han concluido como figura en otro apartado. CRONOLOGÍA DEL PROCESO La demanda interpuesta en la fecha citada en el motivo de la presente decisión fue sorteada por la Presidencia del Juzgado de Trabajo del Distrito Judicial de Santiago, siendo esta sala designada, mediante auto número 00777-2023. Una vez designada la Segunda Sala del Juzgado de Trabajo del Distrito Judicial de Santiago, la jueza presidente de dicha sala, emitió el auto de fijación de audiencia que reposa en el expediente, mediante el cual autorizó a la parte demandante citar a la parte demandada, para que comparezca a la audiencia de conciliación fijada para el día 24/08/2023. l.- FASE DE CONCILIACIÓN La audiencia de conciliación que había sido fijada para la fecha 24/08/2023, tuvo aplazamientos por los motivos indicados en las actas correspondientes, consci	Fecha de audiencia de conciliacion

Figure 2—Labeling example within the shared spreadsheet

Additionally, began scaffolding the paper to be presented to the ICAIL 2025 conference. Document may be found here

5 SCRIPT VALIDATION

The NER model fine-tuned over Spanish legal documents was used for extracting the dates, as that has already been proved to work efficiently. The mentioned script uses pre-cleaned text and passes that on to the model code, which takes care of chunking the text to comply with the 512 token limit and then returns the identified date entities.

Results were then saved into a csv file. Below is a sample of these results:

```
   filename,date,score,start,end,context
Levantamiento de embargo retentivo-0406-2022_cleaned.txt,treinta
↪  (30) días del mes de marzo del año dos mil veintidós
↪  (2022);,0.9946591258049011,646,714,"PRESIDENCIA DE LA CÁMARA
↪  CIVIL Y COMERCIAL DEL JUZGADO DE PRIMERA INSTANCIA DEL
↪  DISTRITO NACIONAL Ordenanza civil núm. 504-2022-SORD-0406
↪  Número único de caso (NUC): 2022-0015525 Yo,
↪  _____, secretaria de la
↪  Presidencia de la Cámara Civil y Comercial del Juzgado de
↪  Primera Instancia del Distrito Nacional, CERTIFICO Y DOY FE:
↪  Que en los archivos de esta cámara hay un expediente de
↪  carácter civil marcado con el número 2022-0015525, que
↪  contiene una ordenanza cuyo texto es el siguiente: EN NOMBRE
↪  DE LA REPÚBLICA En la ciudad de Santo Domingo de Guzmán,
↪  Distrito Nacional, capital de la República Dominicana, a los
↪  treinta (30) días del mes de marzo del año dos mil veintidós
↪  (2022); años ciento setenta y nueve (179) de la Independencia
↪  y ciento cincuenta y nueve (159) de la Restauración.
↪  Presidencia de la Cámara Civil y Comercial del Juzgado de
↪  Primera Instancia del Distrito Nacional, localizada en el
↪  primer piso del Palacio de Justicia del Centro de los Héroes
↪  de Constanza, Maimón y Estero Hondo, en el Distrito Nacional,
↪  República Dominicana, presidida por XXXXXXXXXX, quien dicta
↪  esta ordenanza en sus atribuciones de juez presidente de los
↪  referimientos y en audiencia pública constituida por la
↪  secretaria YYYYYYY. ZZZZZZZZ, y el alguacil de estrados de
↪  turno. Con motivo de la demanda en referimiento sobre
↪  levantamiento de embargo retentivo u oposición interpuesta por
↪  la entidad XXXXXXX (YYYYYYY), constituida conforme a las leyes
↪  de la República Dominicana, con el Registro Nacional de
↪  Contribuyente núm. 1-11-11111-1, con su domicilio social en la
↪  calle BBBBBBB, esquina avenida JJJJJJJ, de esta ciudad,
↪  debidamente representada por su Gerente"
```

Code 3—Sample of retrieved date with context

6 RESULTS VISUALIZATION

Results for this week can be found in the shared document for the paper site which the NLP-DR team will be filling in along the week.

Additionally, in terms of the code, the sample provided in the script validation section would represent the output for the recently implemented code.

7 PROOF OF WORK

Results from the NER model have been verified manually, as the results also include a part of the original text. With these, dates are being identified within the text and then manually labeled to later use these labels to retrieve the classes using one of the Llama models we have been working with so far.

No additional analysis has been made so far on such results.

8 NEXT WEEK'S PROPOSAL

1. Complete the content of the paper.
2. Complete full implementation of a simple pipeline to run all processes end to end.
3. Document and upload all code to the Sentencias private folder.

Week 14 | HAAG - NLP | Fall 2024

Alejandro Gomez

November 22nd, 2024

1 Time-log

1.1 What progress did you make in the last week?

- This week was more of a logistic and planning week. Last week was a major effort to bring code to completion, create a deployed demo, and present to multiple stake holders, so this week was cleaning up loose ends. This week I added documentation for the deployed model and discussed further documentation efforts needed as well as visuals and how they will be connected and created.

1.2 What are you planning on working on next?

- I need to run the model with a different base model, i.e. the facebook bert model where the MMG xlm model I used is a derivative of this. Need to benchmark this for the paper and visualize the data returned so Ill have to have a metric in place.
- Need to work on flow charts, graphs, and other visuals to include in the research paper.
- Meeting with the team to work all of next week to write as much as possible for the publication during the holiday week.

1.3 Is anything blocking you from getting work done?

N/A

2 Article Review

2.1 Abstract

The term Natural Language Processing (NLP) defines the area of research and applications represented by statistical models and algorithms responsible for the analysis and representation of natural language, both phonetic and written. In several countries, the applications of NLP methods in Law are becoming more and more present and represents an increasingly promising future. Classification of legal documents, named entity recognition in legal texts or predictions of legal decisions are just some examples of possible applications. arXiv:2110.15709v1 [cs.CL] 5 Oct 2021 The first objective of this work is to present and make available pre-trained language models for the Brazilian legal language in addition to a Python package with functions to facilitate the use of these models. The second objective of this work is to present and make available demonstrations1, which are accessible to the general public, on how to use language models to analyze Brazilian legal texts in real situations using our models. The organization of this work is as follows: in Sections 2 and 3, we give an overview on NLP methods and mention some related works, besides discussing the importance of this work for the current Brazilian legal context; in Section 4, we briefly present the functions that are available in the first version of our package; in Section, 5 we detail the datasets used for training the language models; in Section 6 we present each of the pre-trained language models provided by us and, finally, in Section 7, we present two demonstrations involving our models. doi[PMP+21]

2.2 Summary

This paper was shared by my teammate because of it's similarity to the domain we are working on. These researchers finetuned models to gather clusters of legal terms as well as to predict legal status with a combination of models. This is parallel to our research because our main objective is to understand delays via timeline information extracted and then make predictions on cases in the future.

3 Scripts and Code Blocks

3.1 Code

```python
from transformers import AutoTokenizer, AutoModelForTokenClassification, pipeline

REPO = "agomez302/nlp-dr-ner"

class NerProcessor:
    def __init__(self):
        self.deployed_tokenizer = AutoTokenizer.from_pretrained(REPO)
        self.deployed_model = AutoModelForTokenClassification.from_pretrained(REPO)
        self.deployed_ner_pipeline = pipeline(
                "ner",
                model=self.deployed_model,
                tokenizer=self.deployed_tokenizer,
                aggregation_strategy="simple"
        )

    def process_text(self, text):
        """Runs NER model on text and returns JSONL string."""
        try:
            chunks = self.split_text_with_overlap(text)
            all_predictions = []
            for chunk in chunks:
                preds = self.deployed_ner_pipeline(chunk)
                all_predictions.extend(preds)
            all_predictions = self.deduplicate_entities(all_predictions)

            formatted_output = {
                "entities": self.run_predictions(all_predictions)
            }

            return json.dumps(formatted_output)

        except Exception as e:
            logger.error(f"Failed to run NER model on extracted text: {e}")

    def split_text_with_overlap(self, text, max_tokens=450, overlap=50):
        """Split text into chunks with overlap to handle long sequences."""
        if not text:
            return []
        max_tokens = min(max_tokens, 512)

        tokenizer = self.deployed_tokenizer
        tokens = tokenizer.encode(text, truncation=False)

        if len(tokens) <= max_tokens:
            return [text]

        chunks = []
        i = 0
        while i < len(tokens):
            chunk = tokenizer.decode(tokens[i:i + max_tokens], skip_special_tokens=
    True)
            chunks.append(chunk)
            i += max_tokens - overlap
        return chunks

    def deduplicate_entities(self, predictions):
        """Remove duplicate entities from overlapping chunks."""
```

```
        unique = []
        seen = set()
        for entity in predictions:
            key = (entity['entity_group'], entity['word'], entity['start'], entity['
    end'])
            if key not in seen:
                unique.append(entity)
                seen.add(key)
        return unique

    def run_predictions(self, predictions: list):
        """Format predictions for output, converting float32 to regular float."""
        try:
            processed_predictions = []
            for pred in predictions:
                pred_dict = dict(pred)
                pred_dict['score'] = float(pred_dict['score'])
                processed_predictions.append(pred_dict)

            return processed_predictions

        except Exception as e:
            logging.error(f"Failed to process predictions: {e}")
            raise


def main():
  text = "SENTENCIA DEL 31 DE ENERO DE 2024 ... 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"
  ner_processor = NerProcessor()
  ner_output = ner_processor.process_text(text)
  print(ner_output)

if __name__ = '__main__':
  main()
```

Listing 1: Example use of NER model on Hugging Face Model Card

This is the script from the documentation that allows users to try out the NER model locally in similar fashion to the deployed demo on HuggingFace Spaces.

3.2 List of Scripts

- Full NER pipeline
  - The NER model documentation oh HuggingFace
  - The finetuning source code for conference submission. Tenatively pending open source publication, too

3.3 Documentation

Following an internal touchbase with the NLP-DR team, I will have a final understanding of the code flow to create a diagram that we can include in the publication. We will also be adding data visualization to support our publication - I'll follow up on these reports.

3.4 Script Validation (optional)

N/A

3.5 Results Visualization

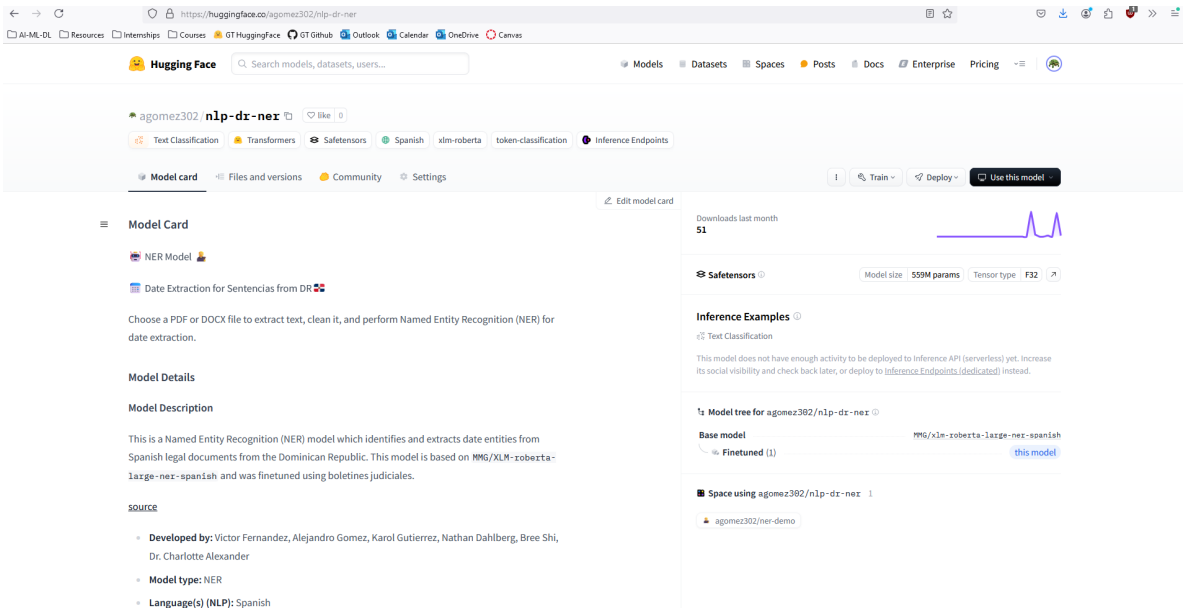


Figure 1: NER Model Documentation

3.6 Results Visualization Summary

This screenshot shows the NER model card which will hopefully serve to boost the AI-Law community by allowing public access and use.

3.7 Proof of Work

Scripts in GitHub Repo

4 Next Week's Proposal

- (See first section for full list. Brief summary below)
- Documentation for NER model including diagrams
- Make larger contributions toward the publication
- Update current documentation, e.g. NLP website with larger changes request by Bri this week.

References

[PMP+21] Felipe Maia Polo, Gabriel Caiaffa Floriano Mendonça, Kauê Capellato J. Parreira, Lucka Gianvechio, Peterson Cordeiro, Jonathan Batista Ferreira, Leticia Maria Paz de Lima, Antônio Carlos do Amaral Maia, and Renato Vicente. Legalnlp – natural language processing methods for the brazilian legal language, 2021.

HAAG NLP Sentencias — Week 14 Report
NLP-Gen Team

Karol Gutierrez

November 22, 2024

1 Weekly Project Update

1.1 What progress did you make in the last week?

- Work based on feedback from call with specialists regarding the classification of sentencias.
- Add code for better classifier using Azure AI Studio API.
- 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. This week on Monday and Friday.

1.2 What are you planning on working on next?

- Wrap up code for project.
- Work on paper with team.
- 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 but by the end of this week we were supposed to receive more sentencias documents. This can help us improve our model.

2 Literature Review

Paper: Using NLP to Model U.S. Supreme Court Cases [LSS23].

2.1 Abstract

The advantages of employing text analysis to uncover policy positions, generate legal predictions, and inform or evaluate reform practices are multifold. Given the far-reaching effects of legislation at all levels of society these insights and their continued improvement are impactful. This research explores the use of natural language processing (NLP) and machine learning to predictively model U.S. Supreme Court case outcomes based on textual case facts. The final model achieved an F1-score of .324 and an AUC of .68. This suggests that the model can distinguish between the two target classes; however, further research is needed before machine learning models are used in the Supreme Court.

2.2 Summary

The authors apply a variety of NLP techniques to model U.S. Supreme Court cases, emphasizing the interpretative potential of computational methods. The methodology includes:

- **Dataset Preparation**: Collecting and preprocessing a dataset of Supreme Court opinions, tagged with metadata such as case topics, justices, and voting outcomes.
- **NLP Techniques**: Employing text classification to predict case outcomes, topic modeling to uncover thematic trends, and sentiment analysis to gauge the ideological tone of opinions.
- **Evaluation and Insights**: Assessing model performance using accuracy and other metrics, and providing insights into shifts in judicial ideology and voting patterns over time.

The results demonstrate that NLP methods can effectively capture the nuances of legal reasoning and contribute to the study of judicial behavior.

2.3 Relevance

This paper is relevant to our Sentencias project as it shows how NLP can be used to analyze judicial opinions, which aligns with our goals. While the focus is on U.S. Supreme Court cases, their methods, such as classification, topic modeling, and metadata tagging, are in line with many of the goals of our project (even beyond). We can apply these techniques to the Spanish language and also leverage that we have access to the Sentencias from the Suprema Corte (Supreme Court) in Dominican Republic to feed our models. We could also incorporate elements of sentiment analysis into our project.

3 Scripts and code blocks

The code is in the private repository repository. The progress for this week is in `./karol/week14/`.

3.1 Code developed

The current workflow of the code is shown in Figure 1.

- I used the existing approach of classifying the code into existing five buckets, but here I used the Azure Open AI API to use gpt-4o as the model to classify the context of the sentencias. Figure 2.
- Code to show compare results with test set in Fig 3. This is an adaptation of the previous script that generates the plots shown in this report.

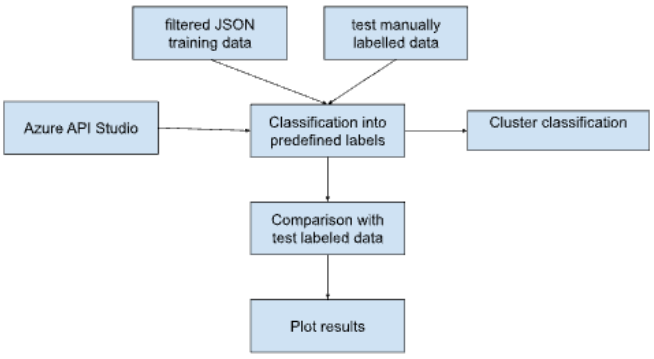


Figure 1: Code logic workflow to process data in Week 14.

4 Documentation

The documentation is present in the README.md file in the repository. No new dependencies were added for this week, all the scripts can be run in Python.

5 Script Validation

Figure 4 shows the generated contexts from the sentencias in Spanish. This document was used to validate the results and provide performance numbers.

6 Results Visualization

I used this different approach to let LLM select the categories for the provided contexts, and it even return "unclassified" or "NA" in some cases, which is useful to identify outliers. The results can be seen in Figure 5. The results are significantly better than previous results since now accuracy is 65%.

Accuracy: 0.65 Precision: 0.52 Recall: 0.65 F1 Score: 0.57

7 Proof of Work

Figures 7 and Figure 8 show the new distribution of the data and samples of the classification during runtime using the LLM, thus proving the work.

8 Next Week's Proposal

Refer to section 1.2 for details (avoid repetition).

References

[LSS23] Katherine Lockard, Robert Slater, and Brandon Sucrese. Using nlp to model u.s. supreme court cases. *SMU Data Science Review*, 7(1), 2023.

Figure 2: Clustering of categories for context using gpt4o

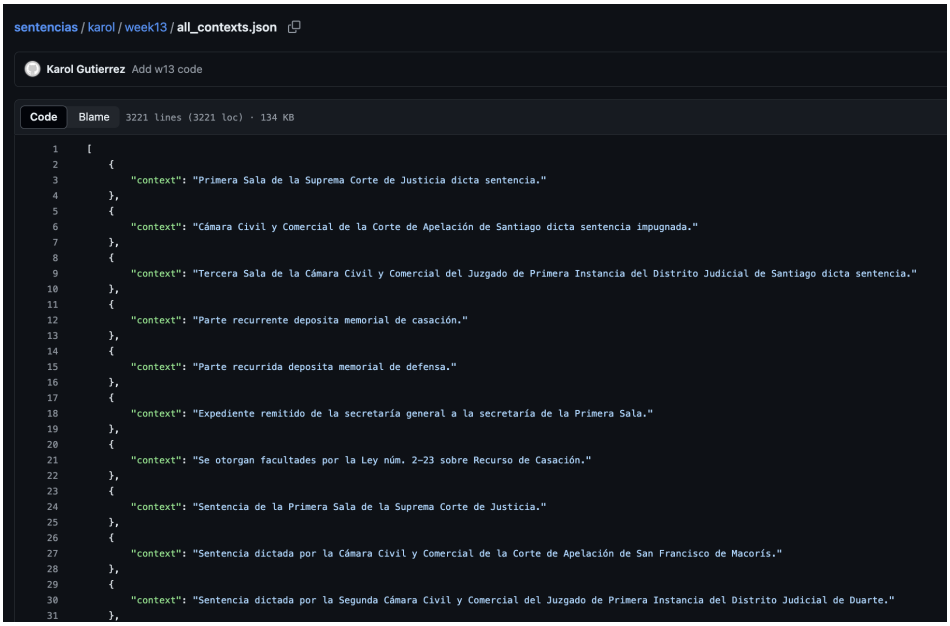


Figure 3: Sentencias context data

sentencias / karol / week13 / classified_contexts_test.json

Karol Gutierrez w13

Code Blame 210 lines (210 loc) · 8.61 KB


```
[
    {
        "context": "Primera Sala de la Suprema Corte de Justicia dicta sentencia.",
        "category": "Supreme Court Judgments"
    },
    {
        "context": "Cámara Civil y Comercial de la Corte de Apelación de Santiago dicta sentencia impugnada.",
        "category": "Appeal Court Decisions"
    },
    {
        "context": "Tercera Sala de la Cámara Civil y Comercial del Juzgado de Primera Instancia del Distrito Judicial de Santiago dicta sentencia.",
        "category": "Initial Trial Judgments"
    },
    {
        "context": "Parte recurrente deposita memorial de casación.",
        "category": "Filing and Appeal Actions"
    },
    {
        "context": "Parte recurrida deposita memorial de defensa.",
        "category": "Filing and Appeal Actions"
    },
    {
        "context": "Expediente remitido de la secretaría general a la secretaría de la Primera Sala.",
        "category": "Supreme Court Judgments"
    },
    {
        "context": "Se otorgan facultades por la Ley núm. 2-23 sobre Recurso de Casación.",
        "category": "Scheduled Dates and Readings"
    },
    {
        "context": "Sentencia de la Primera Sala de la Suprema Corte de Justicia.",
        "category": "Supreme Court Judgments"
    },
    {
        "context": "Sentencia dictada por la Cámara Civil y Comercial de la Corte de Apelación de San Francisco de Macorís.",
        "category": "Appeal Court Decisions"
    },
    {
        "context": "Sentencia dictada por la Segunda Cámara Civil y Comercial del Juzgado de Primera Instancia del Distrito Judicial de Duarte.",
        "category": "Initial Trial Judgments"
    },
    {
        "context": "Memorial de casación depositado por la parte recurrente.",
        "category": "Unclassified"
    },
    {
        "context": "Memorial de defensa presentado por la parte recurrida.",
        "category": "Unclassified"
    },
    {
        "context": "Dictamen emitido por la procuradora adjunta relativo a la solución del recurso de casación.",
        "category": "Unclassified"
    },
    {
        "context": "Expediente remitido de la secretaría general a la secretaría de la sala.",
        "category": "Unclassified"
    },
    {
        "context": "Ocurrió un accidente de tránsito que causó la muerte de Virgilio de Jesús Hernández García.",
        "category": "Unclassified"
    },
    {
        "context": "Primera Sala de la Suprema Corte de Justicia dicta sentencia.",
        "category": "Supreme Court Judgments"
```


Figure 4: Sentencias used for final testing

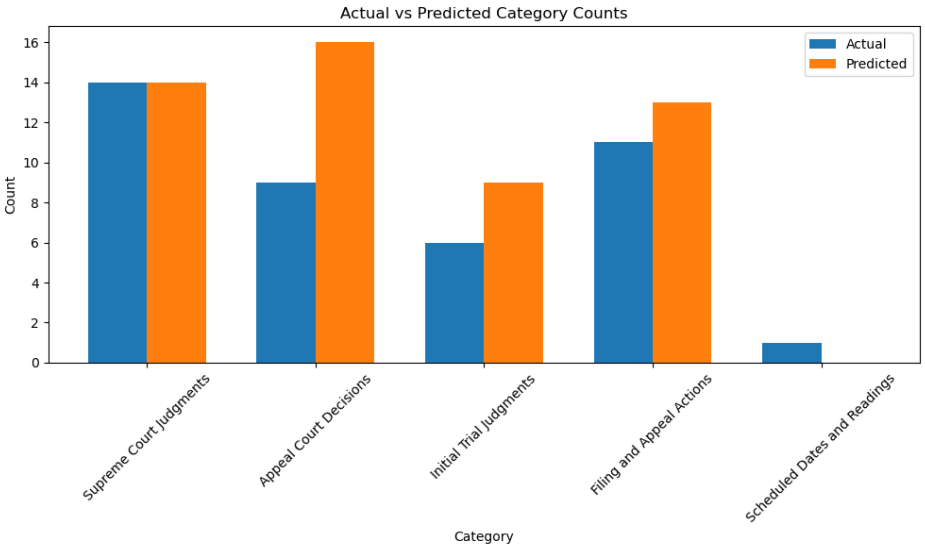


Figure 5: Comparison of results with test data

Figure 6: Improved accuracy results using gpt4o

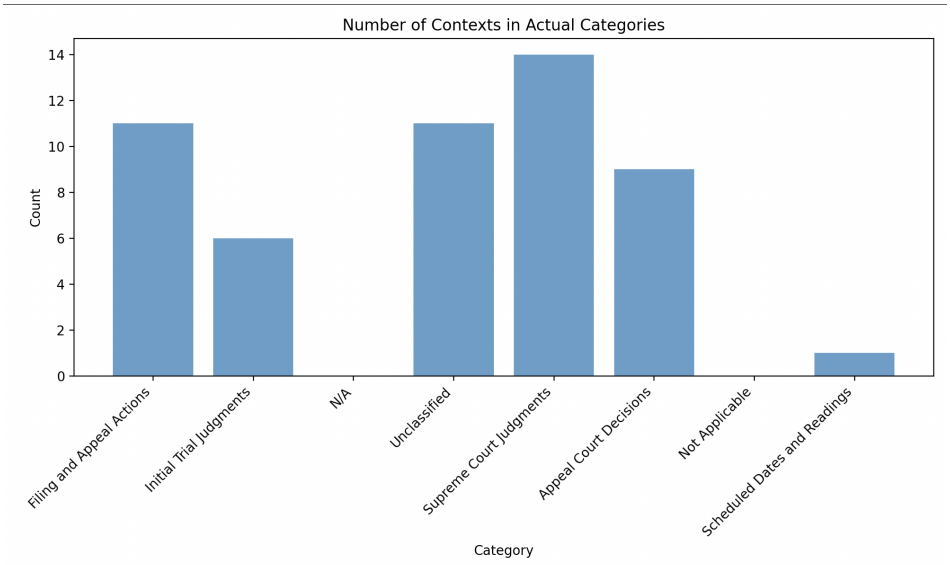


Figure 7: Distribution of categories in available updated data using gpt4o, showing broader categories as result of LLM

Figure 8: Code working and showing the classified contexts with updated data using gpt4o

Week 14 Report

Thuan Nguyen – Clearinghouse Summarization project

Friday, November 22, 2024

Summary

What progress did you make in the last week?

- I started following some langgraph tutorials to learn this tool. Set Up LangGraph: Learned how to configure LangGraph with SQLite for state persistence, including using AsyncSqliteSaver with aiosqlite.
- Understood Event System: Explored LangGraph's event-driven workflow, including lifecycle events like on_chain_start, on_chain_stream, and on_chain_end.
- Worked with Streaming Logic: Practiced handling asynchronous event streams to capture real-time outputs from LangGraph workflows.

What are you planning on working on next?

Dive Deeper into Multi-Agent Workflows: Explore how to design and implement multi-agent workflows using LangGraph's branching and looping capabilities.
Experiment with Human-in-the-Loop Features: Practice integrating human approval and state modification into LangGraph workflows for more interactive applications.

Is anything blocking you from getting work done?

- Nothing at the moment.

Abstract

Translating Legalese - Enhancing Public Understanding of Court Opinions with Legal Summarizers

https://dl.acm.org/doi/10.1145/3614407.3643700

- **Goal**: AI-generated summaries (via GPT-4) simplify Supreme Court opinions, making them accessible to non-experts and improving public understanding.
- **Findings**: AI summaries are clearer, easier to read (7th-grade level), and preferred over expert-written ones, especially by less-educated participants.

- **Impact**: Participants exposed to AI summaries demonstrated better comprehension of case decisions (+11%) and rated them as more shareable and detailed.
- **Challenges**: Simplified summaries may omit legal nuances and context, requiring prompt engineering and expert oversight for accuracy.
- **Future Directions**: Research will explore persuasive effects of summaries, factual validation methods, and their influence on public trust in legal institutions.

Work done this week – further details

- I started following some langgraph tutorials to learn this tool. Set Up LangGraph: Learned how to configure LangGraph with SQLite for state persistence, including using AsyncSqliteSaver with aiosqlite.
- Understood Event System: Explored LangGraph's event-driven workflow, including lifecycle events like on_chain_start, on_chain_stream, and on_chain_end.
- Worked with Streaming Logic: Practiced handling asynchronous event streams to capture real-time outputs from LangGraph workflows.

My code to experiment with LangGraph:

had to do these steps:

git clone https://github.com/langchain-ai/langgraph.git

cd langgraph

2. Navigate to the Correct Submodule

The langgraph library itself and its submodules (e.g., checkpoint-sqlite) are located in libs/. To install the specific submodules you need:

For the main langgraph package:

bash

Copy code

pip install ./libs/langgraph

For the checkpoint-sqlite feature:

bash

Copy code

pip install ./libs/checkpoint-sqlite

You can install both in one step if desired:

bash

Copy code

pip install ./libs/langgraph ./libs/checkpoint-sqlite

also note:

You can create a SQLite connection with check_same_thread=False to allow the connection to be shared across multiple threads:

python

Copy code

conn = sqlite3.connect(":memory:", check_same_thread=False)

memory = SqliteSaver(conn)

1. Properly Use AsyncSqliteSaver

The AsyncSqliteSaver is designed to work with an asynchronous connection, which SQLite does not provide out of the box. You need to use aiosqlite, a library that provides an async wrapper for SQLite.

from langgraph.graph import StateGraph, END

from typing import TypedDict, Annotated

import operator

from langchain_core.messages import AnyMessage, SystemMessage, HumanMessage, ToolMessage

from langchain_openai import ChatOpenAI

```
from langchain_community.tools.tavily_search import TavilySearchResults
import asyncio

import os
os.environ["TAVILY_API_KEY"] = "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"
tool = TavilySearchResults(max_results=4)
print(type(tool))
print(tool.name)

class AgentState(TypedDict):
   messages: Annotated[list[AnyMessage], operator.add]

from langgraph.checkpoint.sqlite import SqliteSaver
import sqlite3
conn = sqlite3.connect(":memory:")
memory = SqliteSaver(conn)

class Agent:
   def __init__(self, model, tools, checkpointer, system=""):
      self.system = system
      graph = StateGraph(AgentState)
      graph.add_node("llm", self.call_openai)
      graph.add_node("action", self.take_action)
      graph.add_conditional_edges("llm", self.exists_action, {True: "action", False: END})
      graph.add_edge("action", "llm")
      graph.set_entry_point("llm")
      self.graph = graph.compile(checkpointer=checkpointer)
      self.tools = {t.name: t for t in tools}
      self.model = model.bind_tools(tools)
```

```
    def call_openai(self, state: AgentState):
        messages = state['messages']
        if self.system:
            messages = [SystemMessage(content=self.system)] + messages
            message = self.model.invoke(messages)
            return {'messages': [message]}

    def exists_action(self, state: AgentState):
        result = state['messages'][-1]
        return len(result.tool_calls) > 0

    def take_action(self, state: AgentState):
        tool_calls = state['messages'][-1].tool_calls
        results = []
        for t in tool_calls:
            print(f"Calling: {t}")
            result = self.tools[t['name']].invoke(t['args'])
            results.append(ToolMessage(tool_call_id=t['id'], name=t['name'], content=str(result)))
            print("Back to the model!")
            return {'messages': results}


prompt = """You are a smart research assistant. Use the search engine to look up information. \
You are allowed to make multiple calls (either together or in sequence). \
Only look up information when you are sure of what you want. \
If you need to look up some information before asking a follow up question, you are allowed to do that!
"""
```

```
model = ChatOpenAI(model="gpt-4o")
# conn = sqlite3.connect(":memory:", check_same_thread=False)
# memory = SqliteSaver(conn)

# messages = [HumanMessage(content="What's the weather in HCM City?")]
# thread = {"configurable": {"thread_id": "1"}}
# # with SqliteSaver.from_conn_string(":memory:") as memory:
# #     abot = Agent(model, [tool], system=prompt, checkpointer=memory)

# #     for event in abot.graph.stream({"messages": messages}, thread):
# #         for v in event.values():
# #             print(v["messages"])

# abot = Agent(model, [tool], system=prompt, checkpointer=memory)

# for event in abot.graph.stream({"messages": messages}, thread):
#     for v in event.values():
#         print(v["messages"])

# messages = [HumanMessage(content="What's weather in Hanoi?")]
# thread = {"configurable": {"thread_id": "1"}}
# for event in abot.graph.stream({"messages": messages}, thread):
#     for v in event.values():
#         print(v)

# messages = [HumanMessage(content="Which one is colder??")]
# thread = {"configurable": {"thread_id": "1"}}
# for event in abot.graph.stream({"messages": messages}, thread):
#     for v in event.values():
```

```
#       print(v)

# messages = [HumanMessage(content="Which one is colder?")]
# thread = {"configurable": {"thread_id": "2"}}
# for event in abot.graph.stream({"messages": messages}, thread):
#     for v in event.values():
#       print(v)

async def main():
    import aiosqlite
    from langgraph.checkpoint.sqlite.aio import AsyncSqliteSaver
    conn = aiosqlite.connect(":memory:", check_same_thread=False)
    memory = AsyncSqliteSaver(conn)
    abot = Agent(model, [tool], system=prompt, checkpointer=memory)

    messages = [HumanMessage(content="What is the weather in SF?")]
    thread = {"configurable": {"thread_id": "4"}}
    async for event in abot.graph.astream_events({"messages": messages}, thread, version="v1"):
        kind = event["event"]
        if kind == "on_chain_stream":
            content = event["data"]["chunk"].content
            if content:
                # Empty content in the context of OpenAI means
                # that the model is asking for a tool to be invoked.
                # So we only print non-empty content
                print(content, end="|")

asyncio.run(main())
```

Scripts - Documentation - Script Validation - Results Visualization - Proof of Work

For further details, please refer to my scripts posted on GitHub or the information above.

Next week's proposal

- Dive Deeper into Multi-Agent Workflows: Explore how to design and implement multi-agent workflows using LangGraph’s branching and looping capabilities.
- Experiment with Human-in-the-Loop Features: Practice integrating human approval and state modification into LangGraph workflows for more interactive applications.

Week 14 Research Report

Thomas Orth (NLP Summarization / NLP Gen Team)

November 2024

0.1 What did you work on this week?

1. Generate summaries using Gemini for settlements
2. Sent summaries to interview team
3. Started reviewing Langgraph, Autogen, and Huggingface Agents for Agentic Workflows

0.2 What are you planning on working on next?

1. Continue working with interview team
2. Work on final semester report for MS Project

0.3 Is anything blocking you from getting work done?

1. None currently

1 Abstracts

- Title: Translating Legalese: Enhancing Public Understanding of Court Opinions with Legal Summarizers. Conference / Venue: CSLAW '24: Proceedings of the Symposium on Computer Science and Law. Link: https://dl.acm.org/doi/10.1145/3614407.3643700
- Abstract: Judicial opinions are written to be persuasive and could build public trust in court decisions, yet they can be difficult for non-experts to understand. We present a pipeline for using an AI assistant to generate simplified summaries of judicial opinions. Compared to existing expert-written summaries, these AI-generated simple summaries are more accessible to the public and more easily understood by non-experts. We show in a survey experiment that the AI summaries help respondents understand the key features of a ruling, and have higher perceived quality, especially for respondents with less formal education.

- Summary: This paper is doing summarization of legal documents with a focus on translating to plain english. The propose a pipeline, centered around GPT-4 for for this and look at different types of cases.
- Relevance: While this work is largely focused on .

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.
- Gemini is Google's LLM product.

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/foejfx8hly8diex99m5smldvnh7ly4by`
3. Thomas-Orth/gemini/settlements/domain_specific_scot_chunked_gemini.py
   - Brief Description: Run a domain specific version of Summary Chain-of-thought (CoT) on settlements 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: No screenshots provided due to the code being largely the same as previous weeks, just with different prompts. Prompts will be pasted at the bottom of the report.
4. Flow Diagram:

Figure 1: Flow diagram

5. Running scripts:
   (a) Download the scripts, the csv from the box link and llm.requirements.txt
   (b) Run: python -m pip install -r llm.requirements.txt
   (c) Sign up for a Gemini API account via Google's AI studio, generate an API Key, and set "GOOGLE_API_KEY" in your environment.
   (d) Run: python (chosen python script)

4 Documentation

1. Download CSV file and
2. Update scripts to point to CSV file
3. Run script to output generated summaries as CSVs
4. Manually evaluate summary

5 Results

5.1 Example Settlement Summary

Below is the example of a settlement summary from Claude Sonnet:

"On November 21, 1996, a class action settlement was reached resolving a lawsuit concerning the educational rights of homeless children in Chicago. The Chicago Public Schools (CPS) and the Illinois State Board of Education agreed to implement policies ensuring equal access to education for homeless children, including immediate enrollment regardless of missing records, transportation to their school of origin (with options for CTA tokens and alternative transportation), a choice of schools, prevention of discouragement from attending their school of origin, access to food programs, tutoring services, and a dispute resolution process for enrollment and transportation issues. They also agreed to provide annual in-service training for school personnel, develop a model for coordinating services with other agencies, designate school liaisons, and provide semi-annual written notices to students and parents, along with reports to plaintiffs' counsel and the Illinois State Board of Education. Attorney's fees and costs were to be negotiated, with a court petition allowed if no agreement was reached. The court retained jurisdiction to enforce the agreement, which included data collection and reporting provisions extending through the 1998-99 school year, with potential for extension. The settlement was contingent upon court approval."

5.1.1 Difference from before

I switched to Gemini based on our interview team's feedback on which LLM to go with.

5.1.2 Evaluation

The summaries currently are evaluated mainly on the information points I added to the prompts. Our interview team will review the summaries to ensure factual correctness, the data points in the prompts are done as well as compare to any additional clearinghouse criteria.

5.2 Prompts

Below are the prompts used by the Gemini model. First prompt will extract key details. The second will take that information to make a summary.

First prompt:

You are a law student tasked with extracting key information from a chunk of a settlement agreement. Your goal is to identify and summarize specific elements of the agreement. Here is the settlement chunk you will analyze:

```
<settlement_chunk>
{document}
</settlement_chunk>
```

Please extract the following information from the settlement chunk:

1. **Actions to be Taken by Defendants:** Describe who has agreed to do what. Be very detailed in providing this information.
2. **Damages (Money):** Identify who is paying for what, including attorney fees. For the money to be paid to plaintiffs, do not name the plaintiffs and report the total sum to be paid to plaintiffs.
3. **Implementation and Enforcement:** Note if there's a court-appointed "monitor" or other oversight.
4. **Duration:** How long the settlement is in effect.
5. **Conditional Agreements:** Mention any conditions for the settlement (e.g., "will only agree IF ...").
6. **Policy Adoptions:** Note any agreement to adopt policies and provide any relevant details about those policies. Do not omit important information and describe in detail.
7. **The Date of the Settlement:** This is typically the document's filing date, the date the document is dated, or the date of execution.
8. **The Type of Settlement:** This is the type of settlement that was entered by this document.

For each piece of information you extract, include a citation of the text from the settlement chunk that supports your conclusion. Use the following format:

<citation>[Exact quote from the text]</citation>

If any of the requested information is not present in the settlement chunk, state *"Not Specified"* for that item.

If any acronyms are present and their definitions are defined, please spell out the acronym the first time it is used.

After extracting the information, provide a brief summary of your findings.

Important: Do not extract or include the following types of information:

- Introductory and Boilerplate Information
- Reporting Information (how parties must report progress)
- Notice for Class Actions (how parties must give notice to consumers for class action suits)
- Giving Up Claims or Admitting Fault (it's a given that settling parties must give up claims)

Present your findings in the following format:

```
<extracted_information>
1. Actions to be Taken by Defendants:
[Your summary]
[Citation if applicable]

2. Damages (Money):
[Your summary]
[Citation if applicable]

3. Implementation and Enforcement:
[Your summary]
[Citation if applicable]

4. Duration:
[Your summary]
[Citation if applicable]

5. Conditional Agreements:
[Your summary]
[Citation if applicable]

6. Policy Adoptions:
[Your summary]
[Citation if applicable]

7. Date of the Settlement:
[Your info]
```

```
[Citation if applicable]

8. Type of Settlement:
[Your info]
[Citation if applicable]
</extracted_information>

<summary>
[Your brief summary of the key points found in the settlement chunk]
</summary>
```

Second Prompt:

You are a law student skilled at distilling sets of extracted information and partial summaries into informative summaries. You will be provided with a set of extracted information and a partial summary about a legal settlement. Your task is to create a concise, one-paragraph summary of the settlement.

Here is the set of extracted information and partial summary:

```
<extracted_info_and_summary>
{chunks}
</extracted_info_and_summary>
```

Using the provided information, create a summary of the settlement following these guidelines:

1. Begin with a sentence describing when the settlement was entered, including the specific date and the type of settlement that was entered.
2. If the case was not dismissed in the settlement, include information on the following aspects, if available:
    - Actions to be Taken by Defendants
    - Damages (Money)
    - Implementation and Enforcement
    - Duration
    - Conditional Agreements
    - Policy Adoptions
3. If the settlement was dismissed, talk about why it was dismissed and what the outcome was.
4. Keep the summary to one paragraph.
5. If any information provides a citation, do not use that information in your summary.
6. Do not omit any of the actions or policy adoptions noted.

7. Write the summary in past tense.

8. If for the requested information, all of the chunks say *"Not Specified"*, do not include that information in the summary.

Carefully review the extracted information and partial summary to ensure you capture all relevant details. Focus on presenting the most important aspects of the settlement in a clear and concise manner.

Please provide your summary within the following tags:

```
<summary>
[Your concise one-paragraph summary here]
</summary>
```

6 Proof of work

The prompts were generated using Anthropic Workbench and ran using Gemini LLMs, so the results are relatively reliable.

6.1 Known Limitations

The prompting could be improved potentially.