Thomson Reuters Data Challenge

Question 1: Data Exploration

Data Contents

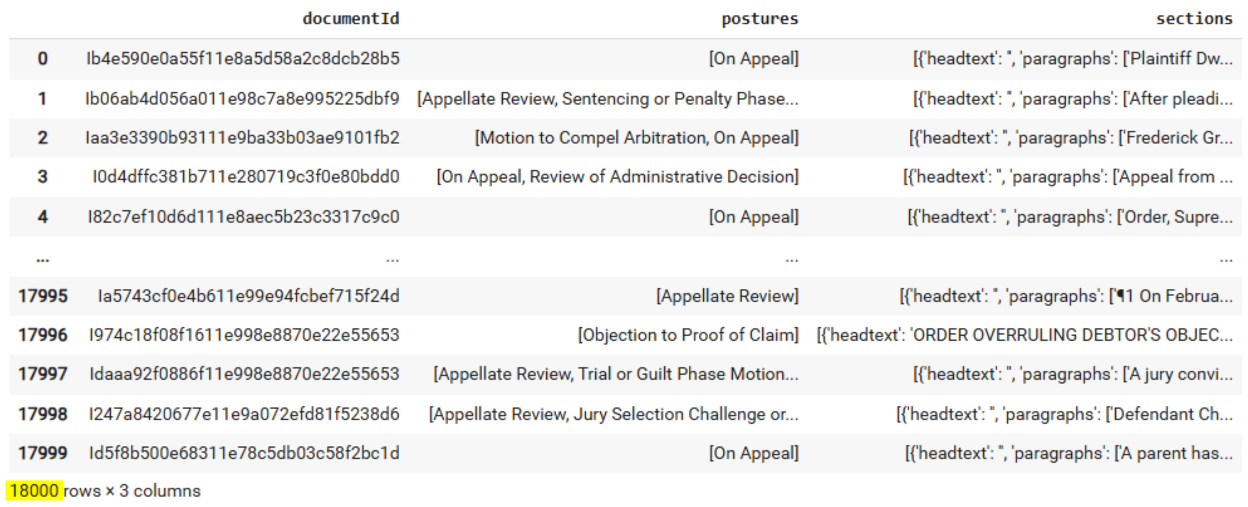
*See notebook 1.

Number of documents: 18,000

Number of postures: 27,659 total, 224 unique

Number of paragraphs: 542,915

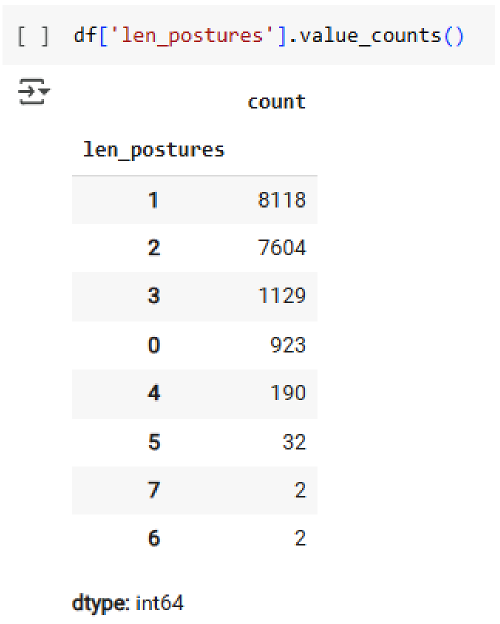
Describe the data: The data, in jsonl format, has 18,000 rows containing one document per row along with a list of postures and sections.



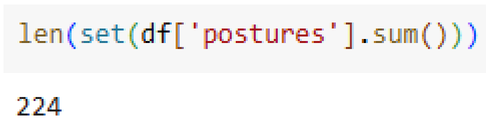
	documentId	postures	sections
0	Ib4e590e0a55f11e8a5d58a2c8dcb28b5	[On Appeal]	[{'headtext': '', 'paragraphs': ['Plaintiff Dw...
1	Ib06ab4d056a011e98c7a8e995225dbf9	[Appellate Review, Sentencing or Penalty Phase...	[{'headtext': '', 'paragraphs': ['After pleadi...
2	Iaa3e3390b93111e9ba33b03ae9101fb2	[Motion to Compel Arbitration, On Appeal]	[{'headtext': '', 'paragraphs': ['Frederick Gr...
3	I0d4dffc381b711e280719c3f0e80bdd0	[On Appeal, Review of Administrative Decision]	[{'headtext': '', 'paragraphs': ['Appeal from ...
4	I82c7ef10d6d111e8aec5b23c3317c9c0	[On Appeal]	[{'headtext': '', 'paragraphs': ['Order, Supre...
...	...	...	...
17995	Ia5743cf0e4b611e99e94fcbef715f24d	[Appellate Review]	[{'headtext': '', 'paragraphs': ['¶1 On Februa...
17996	I974c18f08f1611e998e8870e22e55653	[Objection to Proof of Claim]	[{'headtext': 'ORDER OVERRULING DEBTOR'S OBJEC...
17997	Idaaa92f0886f11e998e8870e22e55653	[Appellate Review, Trial or Guilt Phase Motion...	[{'headtext': '', 'paragraphs': ['A jury convi...
17998	I247a8420677e11e9a072efd81f5238d6	[Appellate Review, Jury Selection Challenge or...	[{'headtext': '', 'paragraphs': ['Defendant Ch...
17999	Id5f8b500e68311e78c5db03c58f2bc1d	[On Appeal]	[{'headtext': '', 'paragraphs': ['A parent has...

18000 rows × 3 columns

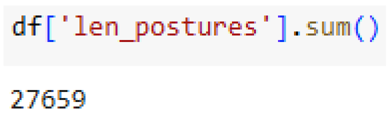
Each document is associated with a list of postures with a maximum length of seven postures per document. Most documents (>15,000) have either one or two associated postures. There are 27,659 total postures, with 224 being unique.



```
[ ] df['len_postures'].value_counts()

                  count
len_postures
           1       8118
           2       7604
           3       1129
           0        923
           4        190
           5         32
           7          2
           6          2

dtype: int64
```

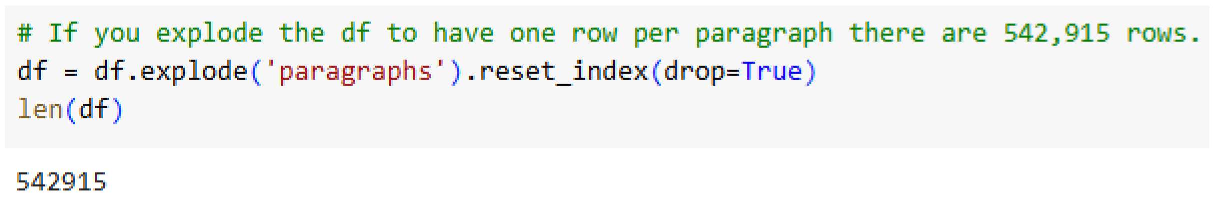


```
len(set(df['postures'].sum()))

224
```

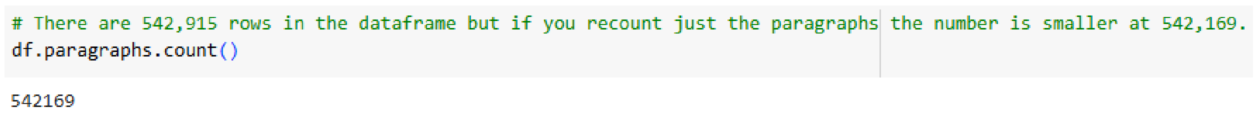


```
df['len_postures'].sum()

27659
```

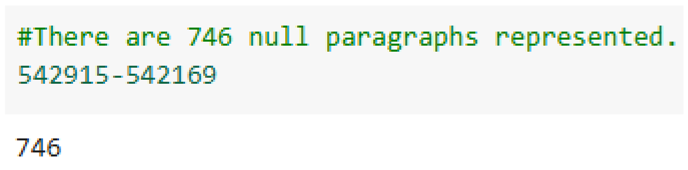
Each document also has a list of sections which take the form of a list of dicts with two keys, headtext and paragraphs. There is one headtext per dict, which may be empty string, and one list of paragraphs, which may occasionally be something like a single period, or null. Overall there are 542,169 paragraphs.



```
# If you explode the df to have one row per paragraph there are 542,915 rows.
df = df.explode('paragraphs').reset_index(drop=True)
len(df)

542915
```



```
# There are 542,915 rows in the dataframe but if you recount just the paragraphs the number is smaller at 542,169.
df.paragraphs.count()

542169
```



```
#There are 746 null paragraphs represented.
542915-542169

746
```

You could reasonably drop those paragraphs with little consequence or replace them with an empty string if your goal is to retain the header text. I dropped them. This eliminated three zero-paragraph count documents from the dataset.

Aspects of Data Relevant to Modeling Task

There are many possible ways to go about processing the text for classification. This text is presented in documents with sections containing headers and paragraphs. **The labeling granularity is at the document level, not the section or paragraph level.** Some options for the choice of what to classify include:

- Concatenating the entire text, including headers, into a single text, creating document embeddings, then classifying.
- Assuming (falsely) that each set of document classifications applies to all sections and paragraphs of a document broadly, creating a concatenated section or paragraph, embedding that text, then classifying.
- Trying to somehow bootstrap your way into determining which paragraphs motivated the labeler to apply certain labels, re-labeling the text with more granular labels, building paragraph level embeddings, then classifying on a more granular basis. This might be done via regex or through using another classifier built on more granularly labeled data that is then applied to the paragraphs. In either case, this process would be fraught and imprecise.
- Asking the data labeling team to label the text at a more granular level, then embedding and classifying at that level.

The first option is more difficult to accomplish using a typical model, which would have a token limit smaller than the length of a legal document, but the last three would allow much more of the text to be processed in its default format before truncating at 512 tokens (or similar).

Question 2: Model

Building the Model

*See notebooks 2 – 4.

Model Selection & Performance

It seems to me that the simplest way to classify these texts is to process them at the document level since the labeling is at the document level. For that purpose, I chose to start with an LLM trained on legal vocabulary, `nlpaueb/legal-bert-base-uncased` available via HuggingFace. I picked BERT, and this particular variant, because BERT is an LLM I have used in the past and because LegalBERT happens to be the only legal language-specific LLM I was aware of at the outset of this project.

My original plan was to follow one of several tutorials I found on multi-class, and multi-class, multi-label text classification using BERT (or its variants). Normally this is done using the following process:

- Acquire some labeled texts
- Tokenize those texts
- Produce embeddings on the tokenized text
- Pass the resulting data structure to BERT's classification head for classification

I did this, initially, and had a baseline accuracy (**not F1**) of somewhere between 0.65 and 0.70. However, BERT is limited to a maximum token length of 512 so using BERT for classification according to this process isn't useful on this dataset of long documents. This method truncated the text to a very short sample at the beginning of the document. There has been some research to suggest that the "head and tail" method is useful with longer documents but it would not be ideal for complex legal texts that contain text on varying subject matter. This method was not remarkably useful so it has been deleted from the notebooks but the poor results serve as a good baseline against which to compare better methods.

There are several options available to classify long documents. One simple and practical method is to use an LLM designed for longer documents. BigBird and PEGASUS-X have a maximum token length of 16,384. Of the documents in this corpus only 98 have a **word count** longer than that total. The token count will be significantly longer but even if all documents with a word count over 10k are excluded it is still the case that only 501 documents of the total 18k in the corpus are truncated, so a practical technique for texts of this length. However, it is **not optimized for legal texts.**

Another option, if you want to retain the benefit of the model trained on legal texts, is to use LegalBERT or similar to chunk, tokenize, embed, stride to an overlapping chunk (so as not to lose contextual information about words that occur at chunk borders) and repeat. Then those chunks can be classified and the classifications aggregated by a chosen method and checked for F1. However, this method runs into the same problem of label granularity as classifying at the paragraph level. We don't know that all labels apply equally to all parts of the text so this method will mean that you **make false assertions about the chunks when training**.

A third option, and the one I chose, is to use the method of chunking, striding, tokenizing and embedding each chunk, then extracting the final layer (the CLS token representation, the encapsulation of the information from the entire input sequence) and taking a mean across all chunks to get a full document embedding. The embedding creation step was successful. My intention with this embedding creation method was to take the full document embedding and feed it to the classification head of the LegalBERT classifier. However, that I am so far aware, there isn't a way to access that classifier head directly that will allow for a 768 element embedding to be processed directly by the BERT classifier head. In any case, the benefit of using this particular version of BERT to process this text comes from the embedding creation, not the classifier head, which is just a subclass of Pytorch's nn.module, so it isn't strictly necessary. Instead, I just used the embeddings along with the one-hot encoded postures as input to a Keras feed-forward network. This method achieved an F1 of about 58%.

Another idea the instructions document seems to hint at is examining only the top ten postures along with their rates of inter-annotator agreement. In notebook four, I changed the one hot encoding to take into account only the top ten postures and used the provided kappas as a class weight, then retrained the model. That version of the model achieves an F1 of about 68%.

Model Strengths & Weaknesses

The strength of feed-forward networks lies in their simplicity. They tend to work in generic situations because they learn to classify through transformations that assume that patterns can be captured in a static feature space rather than with sequential dependencies. Many modern deep-learning models use a simple feed-forward network to do their classification.

However, this model does nothing to take advantage of the way LegalBERT forms its embeddings. It treats the embeddings as static features and does not update them based on classification loss as training a BERT model end-to-end would do. In an ideal situation, a model trained on legal data that can handle long texts would be fine-tuned end-to-end toward the goal of classifying for this particular set of texts and associated postures to capture more domain-specific nuance.

Recommendation to the Business

This dataset has many postures and many occur very rarely. The current model can only provide the correct answer about 70% of the time and much less for rare postures. I would recommend that we do not pursue putting this model into production with the current dataset. However, to improve classification accuracy some steps we could take would be:

- Limit the dataset to fewer classes.
- Ensure that each class is well represented in the dataset.
- Ensure that each class has high inter-annotator agreement.
- Request that the annotators label with a more granular (paragraph level) labeling strategy.

Question 3: Next Steps to Communicate to the Business

In this situation I would conduct additional experiments to improve the model.

Primarily, I would like more time to experiment with models that are purpose-built for these types of long texts. It’s very likely that fine-tuning one of these models would do a better job than the simple baseline model tested during prototyping. Models such as BigBird and PEGASUS-X are capable of managing most of our texts, although they may not be tailored specifically for legal content. A relatively recent model, LexLM (also known as Legal Longformer) released in 2023 does focus on legal text, although it may not fully accommodate the length of our documents. Any of these models is likely to outperform the baseline model currently in place.

In addition, I recommend making modifications to the dataset based on the following considerations.

First, I suggest narrowing the dataset to include fewer classes, prioritizing those that are most beneficial for the user. This approach would allow us to deliver the most useful features immediately while maintaining model accuracy and reliability.

Second, I suggest that we look for more texts that are examples of rare postures. Providing more examples of rare postures will enhance the model's performance.

Third, I would request that the annotation team re-check those texts that have poor agreement between annotators. It is crucial to have dependable data to build an accurate model.

Lastly, I propose that the annotation team apply their labels to specific paragraphs rather than the document as a whole. Identifying which sections of a document prompted a particular annotation will enable more precise classification and facilitate the verification of each other's work among the annotators.

There are several potential challenges related to these recommendations. Collecting and labeling new data requires considerable time and effort, and determining which features to prioritize is a complex undertaking. These initiatives entail costs in terms of both time and budget resources. Additionally, while the models I intend to evaluate are designed to perform well with long texts, they may necessitate additional memory or GPU hardware resources. Allocating these resources would be a beneficial step toward ensuring the project's success.