Technical Challenge

Name: Sritha Hadassah Duddu

Objective:

The aim is to review both files attentively, concentrating on Rules 7 and 19. Please note any instances where the rules are similar or intersect in their application or interpretation.

My Solution:

1) Use BERT for word embeddings and then compute the cosine similarity (this only gives the score)
2) Use SBERT for sentence embeddings and then compute the cosine similarity and **extract similar sentences for both rules**.
3) TF-IDF + Cosine Similarity

Background:

First I start by identifying what kind of task this is. It is text similarity analysis. There are various ways to solve this problem. The general outline starts with preprocessing the text, tokenizing the text, embedding the tokens, computing a similarity score/distance measure, and then extracting the sentences that have a similarity score of more than a certain threshold we set. I went through a lot of resources and blogs before figuring out what kind of preprocessing and embedding to be involved because the method of embedding is what decides to capture the context of the text to an extent. Capturing the context of the text is important for text similarity analysis. No model captures 100% context of texts but attempts to capture context have been improving over the years. Traditional text similarity measures like Jaccard Index, Euclidean Distance, and Cosine Similarity do not capture any context. Then came text embedding which aims to capture the words' meaning, context, and semantic relationships. There are two kinds of text embedding, which are word embedding and sentence embedding. [1] explains how the method of embedding and similarity check improved over the years; [2] explains how BERT is better than Word2Vec because BERT can capture the multiple meanings of the same word in different situations, for instance, consider “river bank” and “bank deposit”, the word “bank” has 2 different meanings here. BERT captures that, while Word2Vec fails to do so. Similarly, by going through [3], [4], [5], [6], and [7], I have decided that I want to try both word embedding and sentence embedding using variations of BERT and then compute the cosine similarity score. SBERT is the variation of BERT that does sentence embedding which allows us to then find out the similarity score and also extract the very similar sentences.

Methodology:

Pre-processing:

I extracted Rule 7, and Rule 19 from file 1 and file 2, and copied and pasted them into 4 different pdfs. I used the “fitz” module from “PyMuPDF" library to read the pdfs. Then I transformed bullet points into full stops if they follow a semicolon or are at the start of the text. I used a regex to identify and keep numeric references and legal terms unchanged. Lowercase the entire text except for the preserved terms. Tokenize the text into sentences and words, remove punctuation (excluding intra-word dashes), and filter out stopwords. Applied lemmatization to reduce words to their base forms. Lastly, the original casing of specific terms is reinstated, ensuring the final text is clean yet retains essential information, ready for further evaluation.

BERT + cosine similarity score:

The idea is to use BERT embeddings from the Hugging Face Transformers library. Initially, it initializes a BERT tokenizer and model with the 'bert-base-uncased' configuration. The preprocessed texts are then tokenized and encoded into PyTorch tensors, taking care to pad and truncate them to ensure consistent length. By feeding these inputs into the BERT model and disabling gradient computations for efficiency, we obtain embeddings for each text. We average the embeddings of all content words in each document to obtain a single vector representing the whole document. It excludes the embeddings for the [CLS] and [SEP] tokens to focus solely on the content words. This averaging process condenses the information contained in the entire document into a single fixed-size vector, making it possible to compare documents of different lengths. The cosine similarity between these sentence embeddings is calculated and printed, providing a quantitative measure of how semantically similar the two rules are.

Note that, since we are using BERT for word embedding here, we can't extract similar sentences between the rules, directly, in this step. In the next section, I have used SBERT which does sentence embedding that will help with extracting similar sentences from the rules.

SBERT+cosine similarity and Extracting Similar Sentences:

I used the Sentence Transformers library, specifically leveraging the 'all-MiniLM-L6-v2' model [8], to compute semantic similarities between sentences from two preprocessed texts. I split the texts into individual sentences using a basic period-based segmentation approach. These sentences are then encoded into high-dimensional vectors (embeddings) that capture their semantic content. And then I used a pairwise cosine similarity measure to assess the semantic similarity between every sentence in the first text and every sentence in the second text, resulting in a comprehensive similarity matrix. To focus on the most relevant comparisons, I establish a similarity threshold of 0.7, above which sentence pairs are considered to exhibit high semantic similarity. These pairs, along with their similarity scores, are then extracted and organized into a data frame, which is sorted in descending order of similarity scores.

Tools used:

- fitz (PyMuPDF)
- nltk and its submodules (nltk.corpus, nltk.stem, nltk.tokenize)
- numpy
- Pandas
- re (regular expressions)
- sentence_transformers

- torch (PyTorch)
- transformers (Hugging Face's Transformers library)

The initial script is taken from the blogs/references that I went through and then modified as per requirements.

<u>RESULTS:</u>

I created two ipynbs, one for SEC-Rule File 1 and the other for NSCC_Disclosure_Framework File 2, just for neat code purposes. In each ipynb, I compare Rule 7 and Rule 19, i.e,

- SEC-Rule File 1.ipynb: “rule 7 doc 1.pdf” vs “rule 19 doc 1.pdf”
- NSCC_Disclosure_Framework File 2.ipynb: “rule 7 doc 2.pdf” vs “rule 19 doc 2.pdf”

<u>SEC-Rule File 1.ipynb:</u> “rule 7 doc 1.pdf” vs “rule 19 doc 1.pdf”

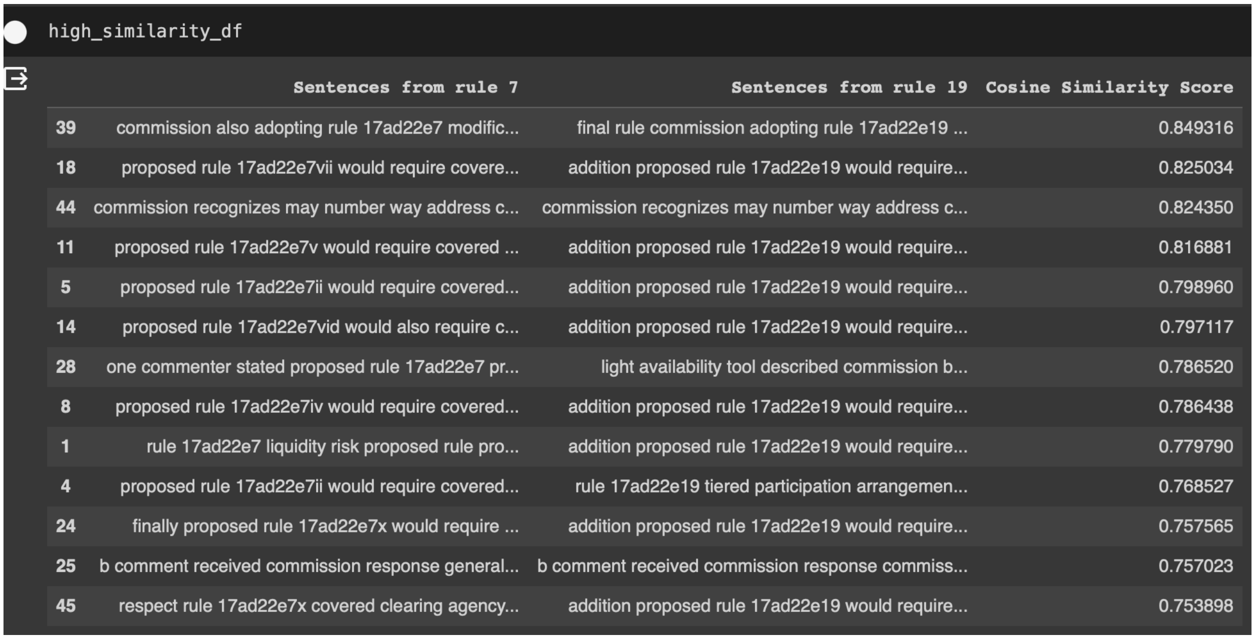
In this file, I followed the methodology mentioned above and came to the results:

Cosine similarity scores:

BERT(word embeddings)	SBERT(sentence embeddings)	TF-IDF
0.9339447021484375	0.8473458886146545	0.6324852109477889

The BERT and SBERT scores are close to each other but TF-IDF score is way lower (showing us that it doesn’t take context into account which resulted in a way off score compared to the other setups). This means that according to the methods I used, both the rules are highly similar. As the supporting statements, I have extracted the similar sentences between both the rules with their cosine similarity scores using SBERT. I set the threshold to 0.7, meaning that all the sentences that have a similarity score above 0.7 will be in the extracted sentences and I made a data frame. I then downloaded the CSV “Similar_sentences_file1.csv" from it.

There were a total of 48 rows. The below code snippet shows the data frame with sentences and their similarity scores in descending order.



```
high_similarity_df
```

	Sentences from rule 7	Sentences from rule 19	Cosine Similarity Score
39	commission also adopting rule 17ad22e7 modific...	final rule commission adopting rule 17ad22e19 ...	0.849316
18	proposed rule 17ad22e7vii would require covere...	addition proposed rule 17ad22e19 would require...	0.825034
44	commission recognizes may number way address c...	commission recognizes may number way address c...	0.824350
11	proposed rule 17ad22e7v would require covered ...	addition proposed rule 17ad22e19 would require...	0.816881
5	proposed rule 17ad22e7ii would require covered...	addition proposed rule 17ad22e19 would require...	0.798960
14	proposed rule 17ad22e7vid would also require c...	addition proposed rule 17ad22e19 would require...	0.797117
28	one commenter stated proposed rule 17ad22e7 pr...	light availability tool described commission b...	0.786520
8	proposed rule 17ad22e7iv would require covered...	addition proposed rule 17ad22e19 would require...	0.786438
1	rule 17ad22e7 liquidity risk proposed rule pro...	addition proposed rule 17ad22e19 would require...	0.779790
4	proposed rule 17ad22e7ii would require covered...	rule 17ad22e19 tiered participation arrangemen...	0.768527
24	finally proposed rule 17ad22e7x would require ...	addition proposed rule 17ad22e19 would require...	0.757565
25	b comment received commission response general...	b comment received commission response commiss...	0.757023
45	respect rule 17ad22e7x covered clearing agency...	addition proposed rule 17ad22e19 would require...	0.753898

Important observation: Since the preprocessed text is a result of the removal of stop words and lemmatization, the sentences extracted will not have those stop words and are still lemmatized and hence look ALMOST like a sentence but not a perfect one.

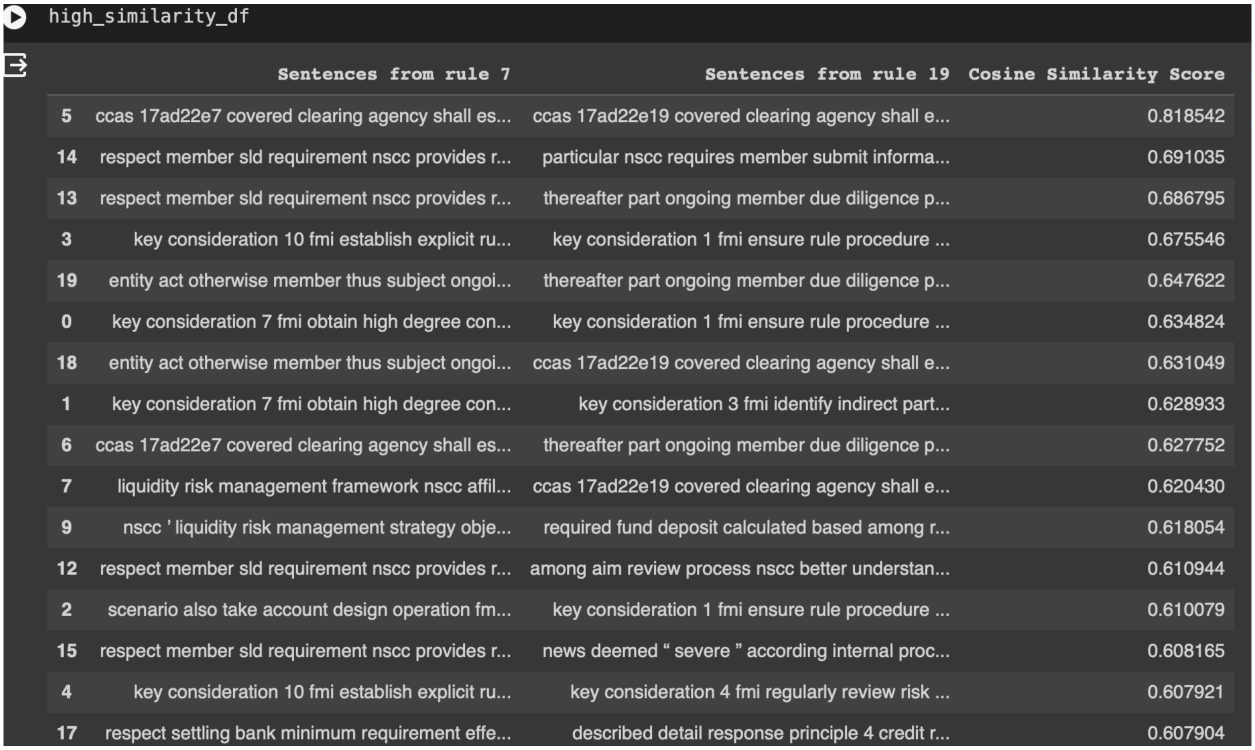
Example: “proposed rule 17ad22e7vii would require covered clearing agency establish implement maintain enforce written policy procedure reasonably designed result performing annual frequent conforming model validation liquidity risk model”

<u>NSCC_Disclosure_Framework File 2.ipynb:</u>
“rule 7 doc 2.pdf” vs “rule 19 doc 2.pdf”
After following the methodology, these are the results that I got:

Cosine similarity scores:

BERT (word embeddings)	SBERT(sentence embeddings)	TF-IDF
0.9217327237129211	0.7677137851715088	0.360892005728255

The scores, in this case, BERT vs SBERT are not very close to each other but still closer compared to TF-IDF score (showing us that it doesn’t take context into account which resulted in a way off score compared to the other setups). I firmly believe that word embedding (BERT) vs sentence embedding (SBERT) is causing this change in scores and the dip in the SBERT score could be that the semantical meaning with the sentence embeddings was considered not to match as much as the previous file. I have extracted the similar sentences between both the rules with their cosine similarity scores using SBERT. I set the threshold to 0.6, meaning that all the sentences that have a similarity score above 0.6 will be in the extracted sentences and I made a data frame. I set a lower threshold for file 2 than file 1 because the threshold of 0.7 resulted in only a single sentence from both rules with a similarity (obviously because of the lower score of similarity in this case). So, I set the threshold to 0.6. I then downloaded the csv
“Similar_sentences_file2.csv” from it. There were a total of 20 rows. The below code snippet shows the data frame with sentences and their similarity scores in descending order.



```
high_similarity_df

                             Sentences from rule 7                             Sentences from rule 19  Cosine Similarity Score
5    ccas 17ad22e7 covered clearing agency shall es...  ccas 17ad22e19 covered clearing agency shall e...                 0.818542
14   respect member sld requirement nscc provides r...   particular nscc requires member submit informa...                 0.691035
13   respect member sld requirement nscc provides r...   thereafter part ongoing member due diligence p...                 0.686795
3         key consideration 10 fmi establish explicit ru...    key consideration 1 fmi ensure rule procedure ...                 0.675546
19     entity act otherwise member thus subject ongoi...   thereafter part ongoing member due diligence p...                 0.647622
0       key consideration 7 fmi obtain high degree con...    key consideration 1 fmi ensure rule procedure ...                 0.634824
18     entity act otherwise member thus subject ongoi...  ccas 17ad22e19 covered clearing agency shall e...                 0.631049
1       key consideration 7 fmi obtain high degree con...       key consideration 3 fmi identify indirect part...                 0.628933
6    ccas 17ad22e7 covered clearing agency shall es...   thereafter part ongoing member due diligence p...                 0.627752
7        liquidity risk management framework nscc affil...  ccas 17ad22e19 covered clearing agency shall e...                 0.620430
9          nscc ’ liquidity risk management strategy obje...   required fund deposit calculated based among r...                 0.618054
12   respect member sld requirement nscc provides r...  among aim review process nscc better understan...                 0.610944
2      scenario also take account design operation fm...    key consideration 1 fmi ensure rule procedure ...                 0.610079
15   respect member sld requirement nscc provides r...   news deemed “ severe ” according internal proc...                 0.608165
4         key consideration 10 fmi establish explicit ru...     key consideration 4 fmi regularly review risk ...                 0.607921
17    respect settling bank minimum requirement effe...      described detail response principle 4 credit r...                 0.607904
```

Overall Comparison:

Rule 7 vs Rule 19	BERT (word embeddings) + Cosine Similarity	TF-IDF + Cosine Similarity	SBERT (sentence embeddings)+ Cosine Similarity	Similar sentences rows (SBERT)
SEC-Rule File 1.ipynb	0.9338878393173218	0.6324852109477889	0.8473458886146545	48
NSCC_Disclosure_Framework File 2.ipynb	0.9216710329055786	0.360892005728255	0.7677137851715088	20

When I read the data from the CSV files, there is a noticeable similarity between sentences.

Conclusion:

To conclude, based on the results from BERT and SBERT, there is an overlap of content and context of rules 7 and 19. But since we are still in an era of figuring out new concepts and models for text similarity analysis tasks, especially the models that capture context to a great extent must still be explored.

Additional Potential Solutions:

These are a couple of ideas that I haven't implemented due to limited resources.

1) One good way would be to compare the scores of using different models for embedding and using cosine similarity to determine the score. Example: GPT for embedding + cosine similarity. Unfortunately, the OpenAI API to use GPT is not free.
2) Another idea that I have is to use GPT and with prompt engineering ask it if the documents are similar, how much the score is, and what kind of sentences the model thinks match and doesn't, but again, API is not free.

References:

1. https://www.newscatcherapi.com/blog/ultimate-guide-to-text-similarity-with-python#topics-covered-in-this-article
2. https://spotintelligence.com/2022/12/17/sentence-embedding/#What_are_the_top_five_tools_for_implementing_sentence_embedding
3. https://www.youtube.com/watch?v=A8HEPBdKVMA
4. https://medium.com/@claude.feldges/word-embeddings-and-text-embeddings-a-comparison-9547f8f0c9a3
5. https://medium.com/@tam.tamanna18/exploring-the-power-of-nlp-why-embeddings-usually-outperform-tf-idf-98742e7b0bce
6. https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_615#:~:text=An%20index%20can%20facilitate%20the,between%20a%20pair%20of%20objects

7. https://medium.com/@ankiit/word2vec-vs-bert-d04ab3ade4c9#:~:text=Application%3A%20BERT%20architecture%20can%20be,vector%20representation%20of%20the%20vocabulary.
8. https://www.graft.com/blog/text-embeddings-for-search-semantic
9. https://medium.com/@khiljidanial/cosine-similarity-using-gpt-models-35b6b9685d70
10. https://towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use-3d5ab82ba5f8
11. https://www.scaler.com/topics/nlp/text-similarity-nlp/
12. https://spotintelligence.com/2022/12/19/text-similarity-python/#4_Text_similarity_with_RoBERTahttps://spotintelligence.com/2022/12/19/text-similarity-python/#4_Text_similarity_with_RoBERTahttps://platform.openai.com/docs/guides/embeddings/what-are-embeddings