BoE Project Assignment 3: Final report

Team STOMPA

30th June 2025

1 Background and Context

The Bank of England's Prudential Regulation Authority (PRA) is responsible for ensuring the safety and soundness of critical financial institutions, including global systemically important banks (G-SIBs). While these institutions regularly publish quantitative financial data, a significant amount of qualitative insight is conveyed during earnings calls and analyst Q&A sessions, with transcripts often containing nuanced discussions of strategy, operations, and market concerns. However, their unstructured and technical nature limits use of traditional analytical methods.

This project seeks to apply advanced natural language processing (NLP) techniques to extract and interpret relevant information from these transcripts. The goal is to enhance the PRA's ability to detect early signals of risk and institutional change that may not be apparent in standard financial disclosures.

The central challenge is to determine whether these transcripts contain early warning signals that can inform supervisory judgment and decision-making. Specifically, we aim to identify:

- **Financial or operational risks** not evident in headline metrics
- **Strategic shifts** that may indicate changes in risk appetite or business model
- **Market concerns** raised by analysts that may point to reputational or systemic issues

Transforming unstructured transcripts into structured insights allows the PRA to strengthen its oversight and improve responses to emerging risks.

2 Project Development Process

The process flow shows the full pipeline for the analysis of the quarterly earnings calls. It allows for multiple agents simultaneously to analyse the data for indicators of risk and will display the key insights and findings through graphs and summaries comparing banks, quarters, sentiments and topics.

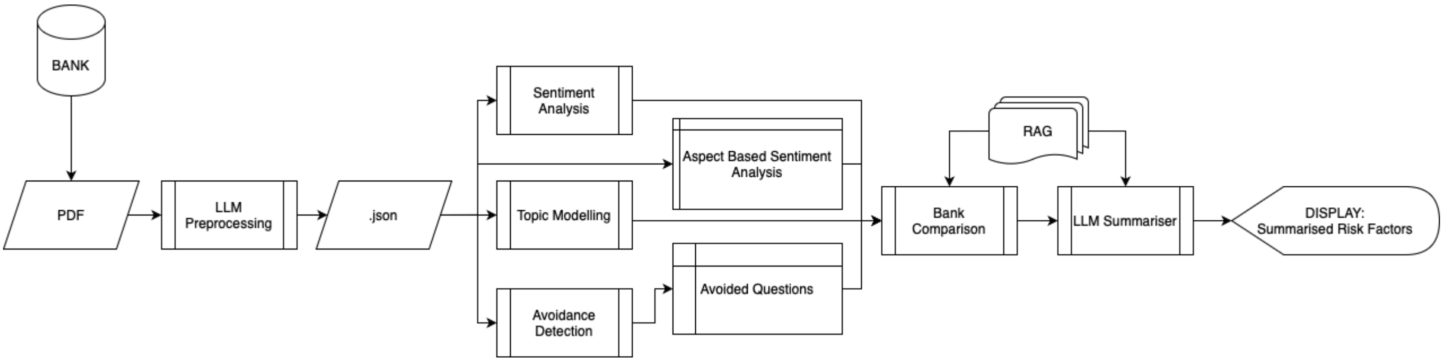


Figure 1 - Process flow of the full pipeline

2.1 Sentiment Analysis

2.1.1 Risk-Based Sentiment Analysis

The sentiment analysis component of this project followed a structured, iterative process encompassing data acquisition, preprocessing, modelling, and evaluation. Quarterly earnings call transcripts were collected for three major G-SIBs: Citi, JPMorgan and UBS, spanning 2010 to 2022. These were processed using PyMuPDF and segmented into Presentation and Q&A sections for more granular analysis.

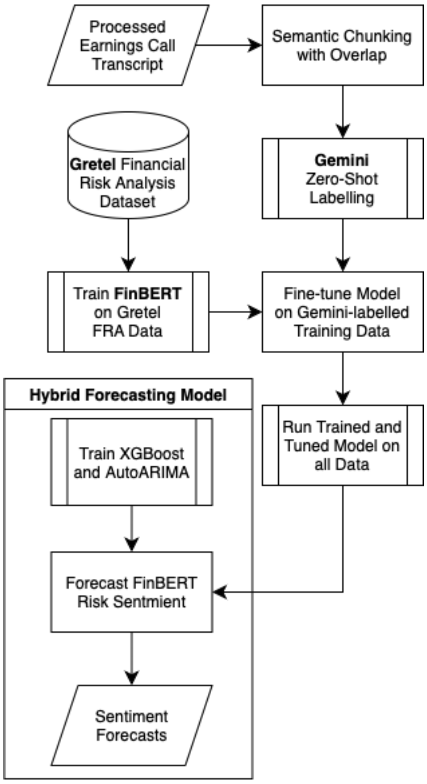

Processed Earnings Call Transcript
Semantic Chunking with Overlap
Gretel Financial Risk Analysis Dataset
Gemini Zero-Shot Labelling
Train FinBERT on Gretel FRA Data
Fine-tune Model on Gemini-labelled Training Data
Hybrid Forecasting Model
Train XGBoost and AutoARIMA
Run Trained and Tuned Model on all Data
Forecast FinBERT Risk Sentmient
Sentiment Forecasts

Figure 2: Process flow of the Sentiment Analysis pipeline

Risk severity labelling employed a three-stage strategy. The initial phase used a rule-based heuristic labeller, applying lemmatised unigrams, bigrams, and trigrams matched against curated keyword sets to classify sentences as LOW, MEDIUM, or HIGH risk. To improve semantic flexibility and scalability, Gemini was then used for zero-shot classification, enabling context-aware annotation at scale. These outputs, combined with the GretelAI Financial Risk Analysis dataset, were used to fine-tune FinBERT, producing a domain-specific model for consistent and automated risk tagging during inference.

For forecasting, a hybrid modelling approach was implemented by combining SARIMA and XGBoost. SARIMA parameters were selected via auto_arima. XGBoost was trained on lagged features with early stopping. Outputs from both models were integrated using a weighted average.

Final forecasts were visualised in a Streamlit-based dashboard aligned to future quarter-end dates. Key challenges included inconsistent transcript formats and initial FinBERT misclassification of MEDIUM risk, both mitigated through refined preprocessing scripts and data augmentation.

2.1.2 Aspect-Based Sentiment Analysis

Aspect-based sentiment analysis (ABSA) was used to gain insight into the sentiment expressed by G-SIBs towards topics identified in the earnings call transcripts. This enabled nuanced interpretation of how institutions communicated around key financial concepts such as liquidity and interest rates with the aim of unveiling shifts in tone that may reflect emerging risks or changing strategic priorities.

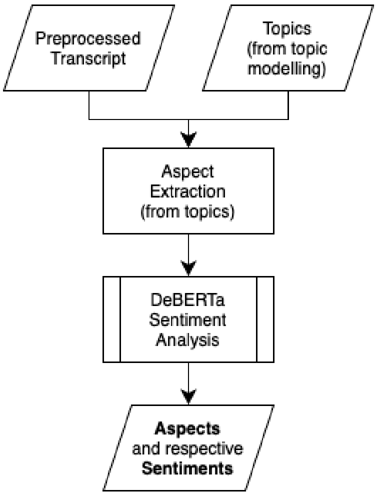

Preprocessed Transcript
Topics (from topic modelling)
Aspect Extraction (from topics)
DeBERTa Sentiment Analysis
Aspects and respective **Sentiments**

Figure 3: Process flow of the Aspect-Based Sentiment Analysis pipeline

A DeBERTa model was applied for this task (*deberta-v3-base-absa-v1.1*). This is a transformer-based model that has been fine tuned specifically for ABSA and was largely chosen for its ability to capture subtle contextual cues. This ability is important for analysing the formal language used in financial presentations and publishings.

The aspects to be evaluated were derived directly from the keywords identified through the topic modelling stage of the pipeline described in the next section. This integration aims to enable consistent tracking of how each G-SIB expresses sentiment towards their most frequently discussed themes across quarters.

2.2 Topic Modelling

Two complementary methodologies were employed for topic extraction from banking earnings calls: BERTopic and LLM-based topic modelling. The analysis incorporated transcripts from three major banks, with Silicon Valley Bank serving as a baseline for identifying risk-associated topics in failing institutions.

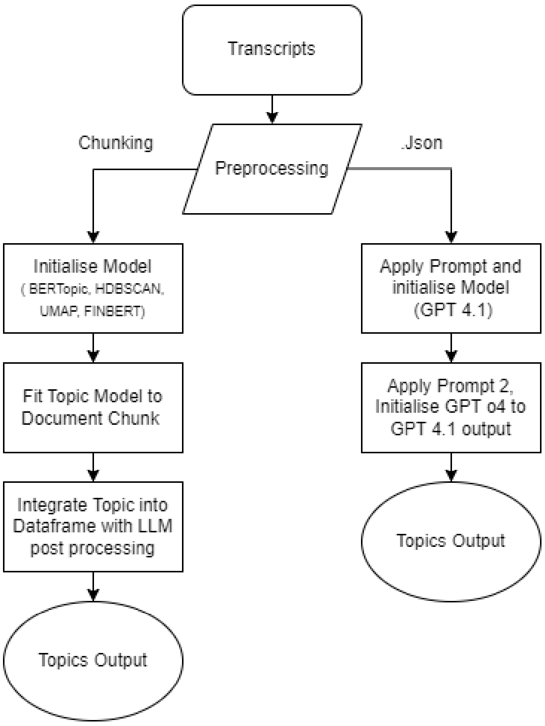

Transcripts
Chunking
Preprocessing
.Json
Initialise Model (BERTopic, HDBSCAN, UMAP, FINBERT)
Apply Prompt and initialise Model (GPT 4.1)
Fit Topic Model to Document Chunk
Apply Prompt 2, Initialise GPT o4 to GPT 4.1 output
Integrate Topic into Dataframe with LLM post processing
Topics Output
Topics Output

Figure 4: Topic Modelling, Process Flowchart

BERTopic utilised UMAP, HDBSCAN, and FinBERT for financial-context embeddings, requiring custom preprocessing of raw transcripts. However, several limitations emerged: spaCy misclassified proper names, n-grams could be generic, and topic interpretability remained challenging. To address some of these constraints, a Gemini LLM post-processing step was integrated to generate contextualized topic summaries with frequency analysis.

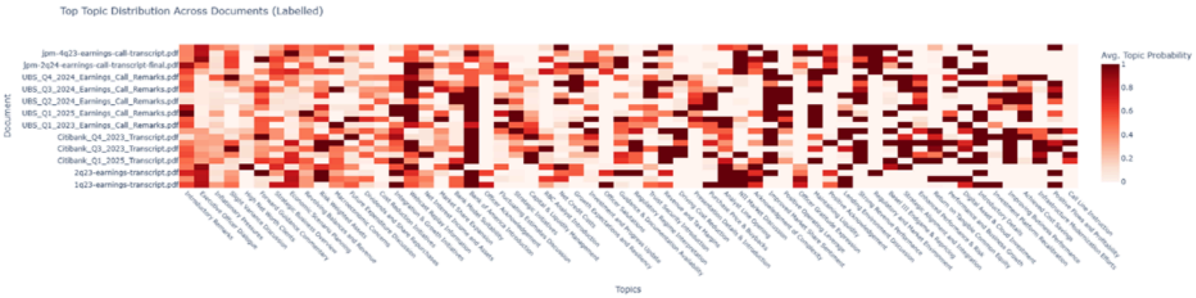


Figure 5: Frequency distribution of key topics and their occurrence across the document corpus

The LLM approach processed JSON files through OpenAI's GPT-4.1 API using carefully designed prompts, with results refined through GPT-4o-mini due to enhanced token capacity. This method identified the five leading topics per transcript, keywords and summarisations. Providing granular, actionable insights compared to BERTopic's broad categorizations.

Key advantages of the LLM methodology included highly specific thematic clustering, problem-focused identification of challenges e.g Net Interest Margin pressure and Commercial Real Estate exposure. This context and justification enhances the RAG's capacity to deliver accurate, relevant responses.

Validation against Silicon Valley Bank suggests the approach's effectiveness in identifying high-risk positions, though temporal context limitations are acknowledged.

2.2.1 Future Development

Cost optimization is essential as LLM processing expenses escalate with scale, requiring exploration of hybrid approaches and quantitative performance metrics to objectively compare methodologies. Expanding validation beyond Silicon Valley Bank to include multiple distressed institutions would enable statistical analysis of correlations between topic patterns and actual bank risk levels.

2.3 Question Avoidance Detection

This section assesses whether executives evaded direct answers to analyst questions with the aim of revealing notable trends in transparency and executive communication strategy. This process is based on Google’s gemini-2.0-flash LLM which analyses transcripts and highlights any avoided questions and its relevant theme.

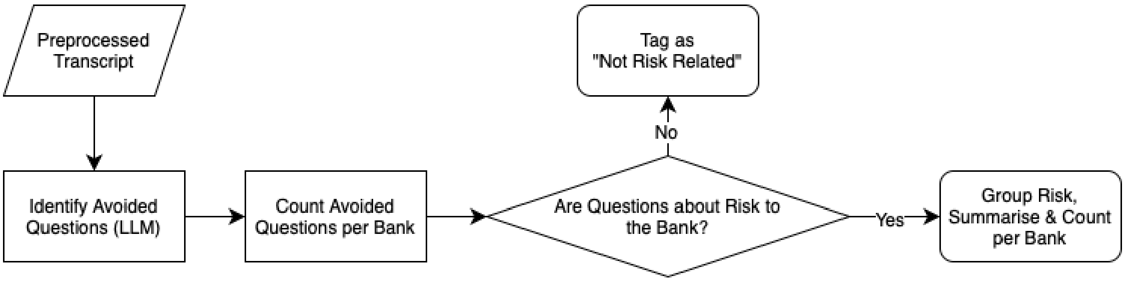

Preprocessed Transcript
Identify Avoided Questions (LLM)
Count Avoided Questions per Bank
Are Questions about Risk to the Bank?
No
Tag as "Not Risk Related"
Yes
Group Risk, Summarise & Count per Bank

Figure 6: Question Avoidance Detection, Process Flowchart

For each Q&A pair, the model evaluates whether an answer can be considered *evasive*. The prompt is engineered to classify responses with clarity and objectivity, marking them as either “avoided” or “answered.” Avoided questions are then noted and stored for analysis of any overarching themes.

2.4 Retrieval Augmented Generation (RAG)

LangChain has been utilised to develop a RAG pipeline. A vector database has been populated using two types of information:

- Earnings call transcripts
- Basel Framework Docs.

The previous steps of the pipeline will be used to inform prompts based on identified topics and other indicators associated with risk. The retriever performs a semantic search to the vector database to find additional context that is relevant to the query. The prompt along with the retrieved data is then fed into the LLM to produce a more contextualised response.

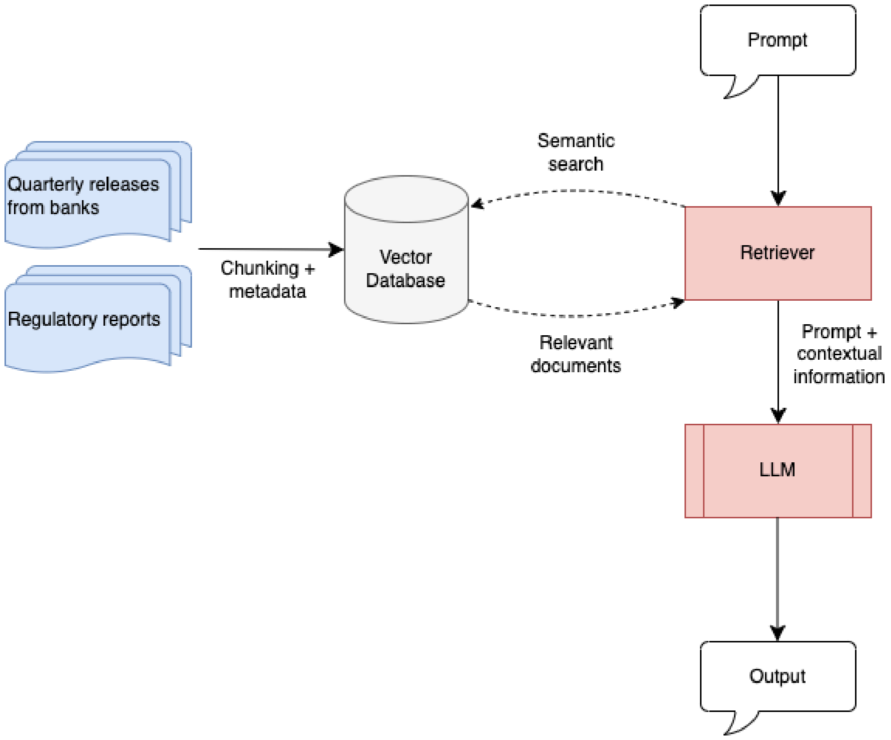
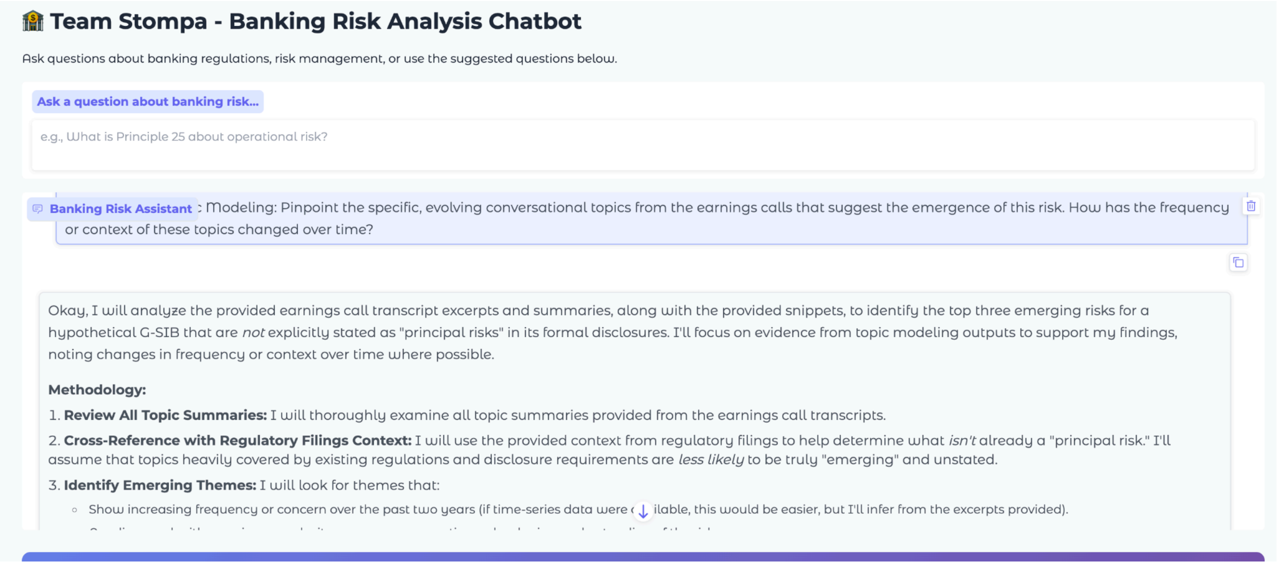


Figure 7: Process flow of the RAG pipeline

Initially, for some queries the RAG was failing to accurately find context and reference accurate sources. A chunking strategy was then developed to increase the effectiveness of the semantic search to the vector database. Metadata has been added to each chunk to ensure they all contain key information about the source document. This has increased the ability of the pipeline to provide contextual responses for some prompts.

The final RAG has been integrated into a virtual assistant chatbot that can be queried and provide contextual responses including references to the relevant regulatory framework. The virtual assistant can also suggest questions that could provide relevant insight on financial risk. We also found that different LLMs performed differently on some tasks, as a result the chatbot contains functionality to query three different LLMs, which may provide different insights.



🏦 **Team Stompa - Banking Risk Analysis Chatbot**

Ask questions about banking regulations, risk management, or use the suggested questions below.

Ask a question about banking risk...
e.g., What is Principle 25 about operational risk?
Banking Risk Assistant

: Modeling: Pinpoint the specific, evolving conversational topics from the earnings calls that suggest the emergence of this risk. How has the frequency or context of these topics changed over time?

Okay, I will analyze the provided earnings call transcript excerpts and summaries, along with the provided snippets, to identify the top three emerging risks for a hypothetical G-SIB that are *not* explicitly stated as "principal risks" in its formal disclosures. I'll focus on evidence from topic modeling outputs to support my findings, noting changes in frequency or context over time where possible.

Methodology:

1. **Review All Topic Summaries:** I will thoroughly examine all topic summaries provided from the earnings call transcripts.
2. **Cross-Reference with Regulatory Filings Context:** I will use the provided context from regulatory filings to help determine what *isn't* already a "principal risk." I'll assume that topics heavily covered by existing regulations and disclosure requirements are *less likely* to be truly "emerging" and unstated.
3. **Identify Emerging Themes:** I will look for themes that:
   - Show increasing frequency or concern over the past two years (if time-series data were ilable, this would be easier, but I'll infer from the excerpts provided).

Figure 8: Screenshot of the virtual assistant chatbot

3 Results

Topic modelling revealed consistent areas of concern across G-SIBs. Net interest margin pressure, exposure to commercial real estate, capital adequacy, and digital transformation appeared as dominant themes. UBS and JPMorgan emphasised loan book quality and capital resilience, while Citibank focused on operational efficiency and liquidity management. Across banks, commercial real estate and cybersecurity risks also appeared frequently, often in connection with digital infrastructure investments and regulatory scrutiny.

Aspect based sentiment analysis found the most frequently discussed aspects to be Loan Book Quality, Interest Rates, and Capital Adequacy. Overall sentiment was mostly neutral, though Loan Book Quality stood out with more negative than positive sentiment. Interest Rates showed a gradual rise in negativity over the past two years, while sentiment around Capital Adequacy has brightened slightly. In the last three quarters, concerns about Loan Book Quality have lessened, although it remains more negative than positive. Graphs of quarterly counts for the top three aspects can be found in appendix A.

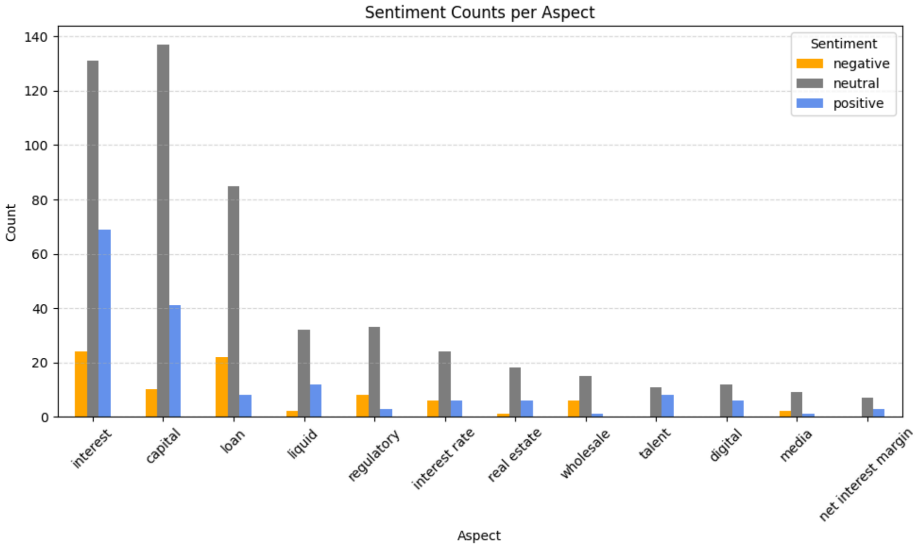


Figure 9: Number of sentiment occurrences per aspect

Risk-Based Sentiment analysis shows that UBS exhibits the highest proportion of high-risk sentences during Quarterly Earnings Calls and maintains elevated levels through 2026 despite a slight decline. JP Morgan shows strong volatility with a projected peak in mid-2025, followed by a gradual decline. Citigroup demonstrates a steady recovery from a mid-year dip, trending upward into next year. Overall, the hybrid forecast model highlights UBS and JPMorgan as key institutions to monitor due to their elevated or fluctuating high-risk outlooks.

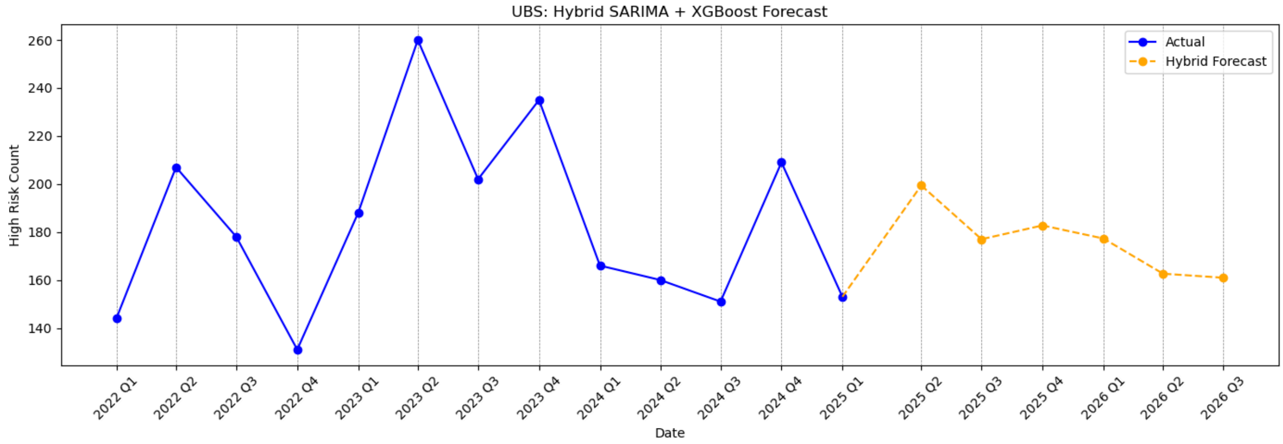
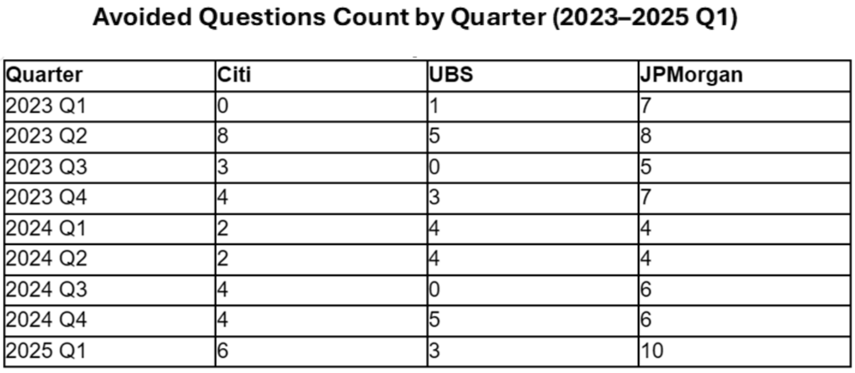


Figure 10: Hybrid sentiment forecast model results for UBS

Question avoidance detection identified several instances where questions appeared to be deflected. These occurred most often during periods of external pressure or internal transition. The avoidance frequency fluctuates over time with an uptick in Q2 of 2023 and generally higher avoidance at JPMorgan.



Avoided Questions Count by Quarter (2023–2025 Q1)

Quarter	Citi	UBS	JPMorgan
2023 Q1	0	1	7
2023 Q2	8	5	8
2023 Q3	3	0	5
2023 Q4	4	3	7
2024 Q1	2	4	4
2024 Q2	2	4	4
2024 Q3	4	0	6
2024 Q4	4	5	6
2025 Q1	6	3	10

Figure 11: Tabulated results of Avoided Questions per Bank, per Quarter (2023 Q1 -2025 Q1)

Avoided questions primarily focused on:

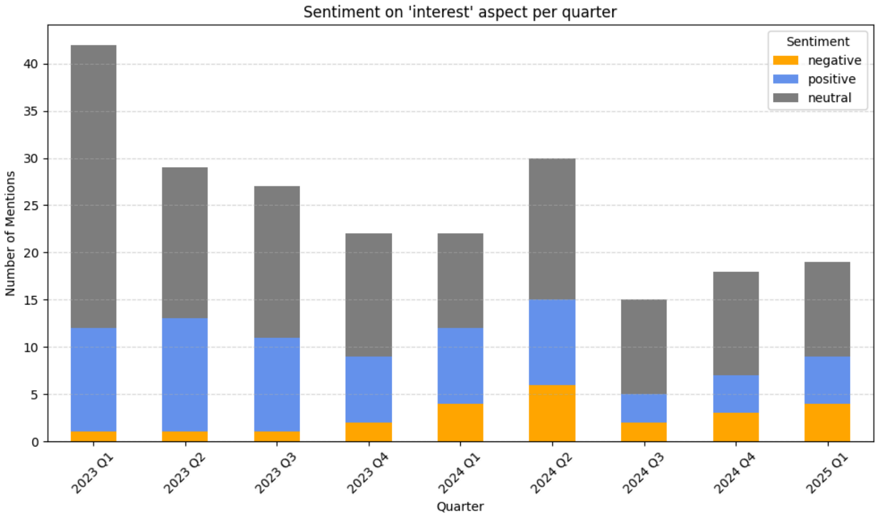
- **Citibank**: regulatory compliance and cost management
- **JPMorgan**: capital planning and macroeconomic resilience
- **UBS**: post-merger integration and capital return constraints

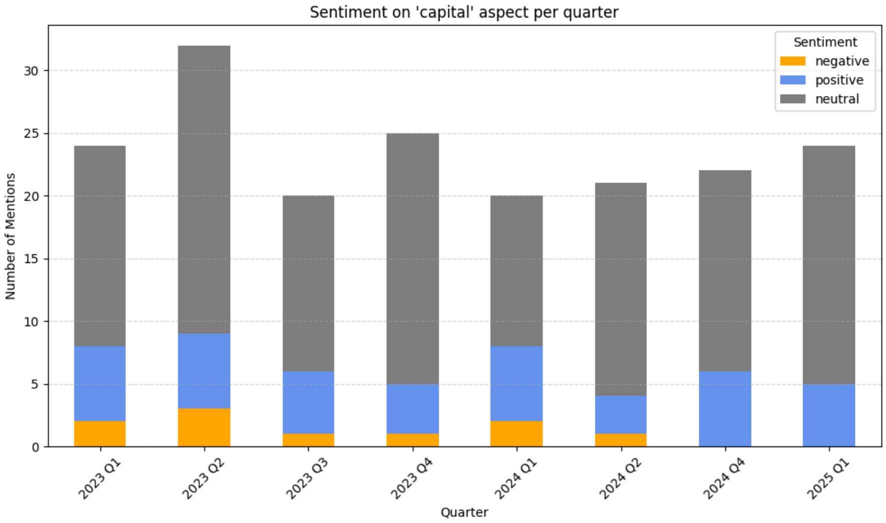
While these patterns stress a few common points, no single theme was consistently avoided across all banks.

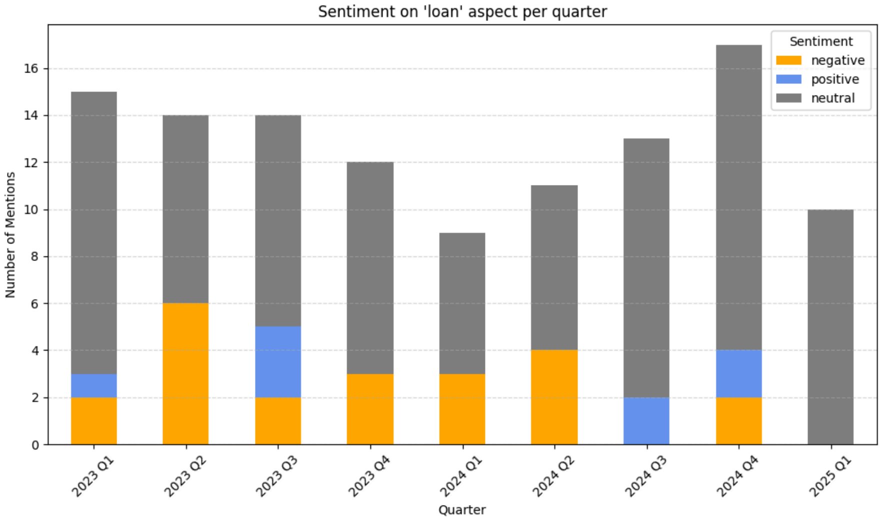
The RAG chatbot provides a helpful tool for providing context to the user, meaning that specific outputs from the prior steps of the process could be analysed by someone without a specific finance background.

Appendices

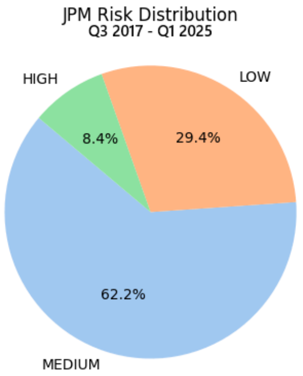
Appendix A

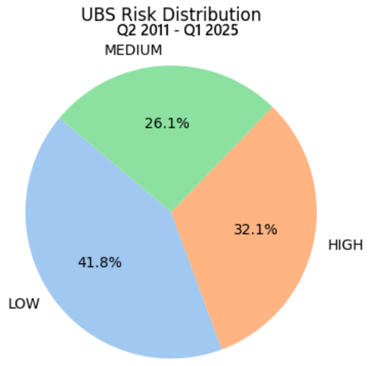


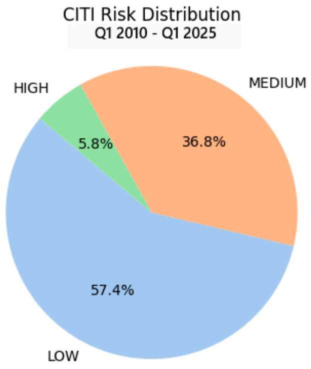


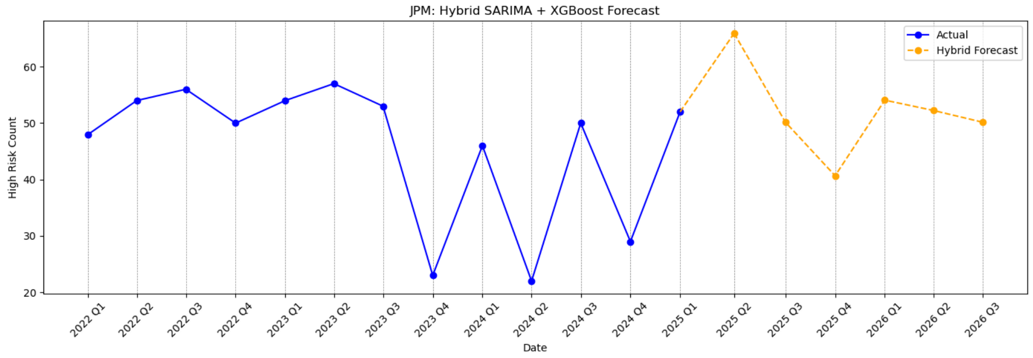


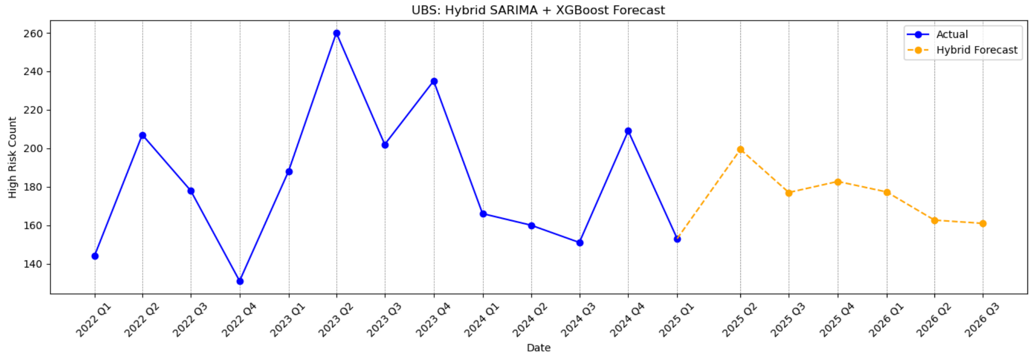
Appendix B

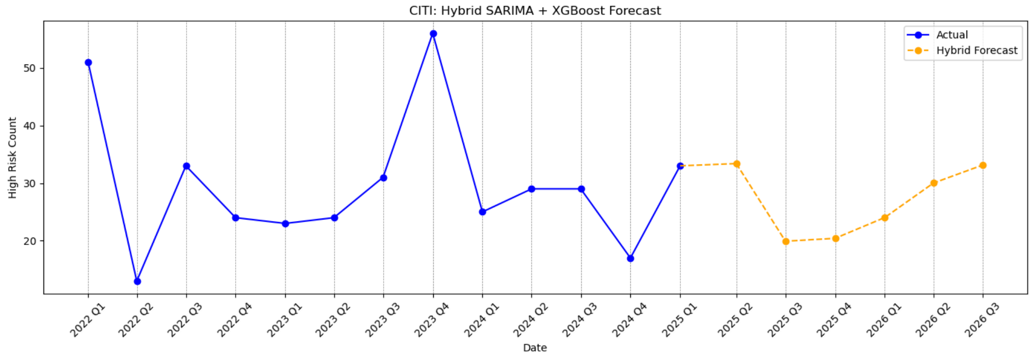

JPM Risk Distribution
Q3 2017 - Q1 2025
HIGH
LOW
8.4%
29.4%
62.2%
MEDIUM


UBS Risk Distribution
Q2 2011 - Q1 2025
MEDIUM
26.1%
32.1%
HIGH
41.8%
LOW


CITI Risk Distribution
Q1 2010 - Q1 2025
HIGH
MEDIUM
5.8%
36.8%
57.4%
LOW

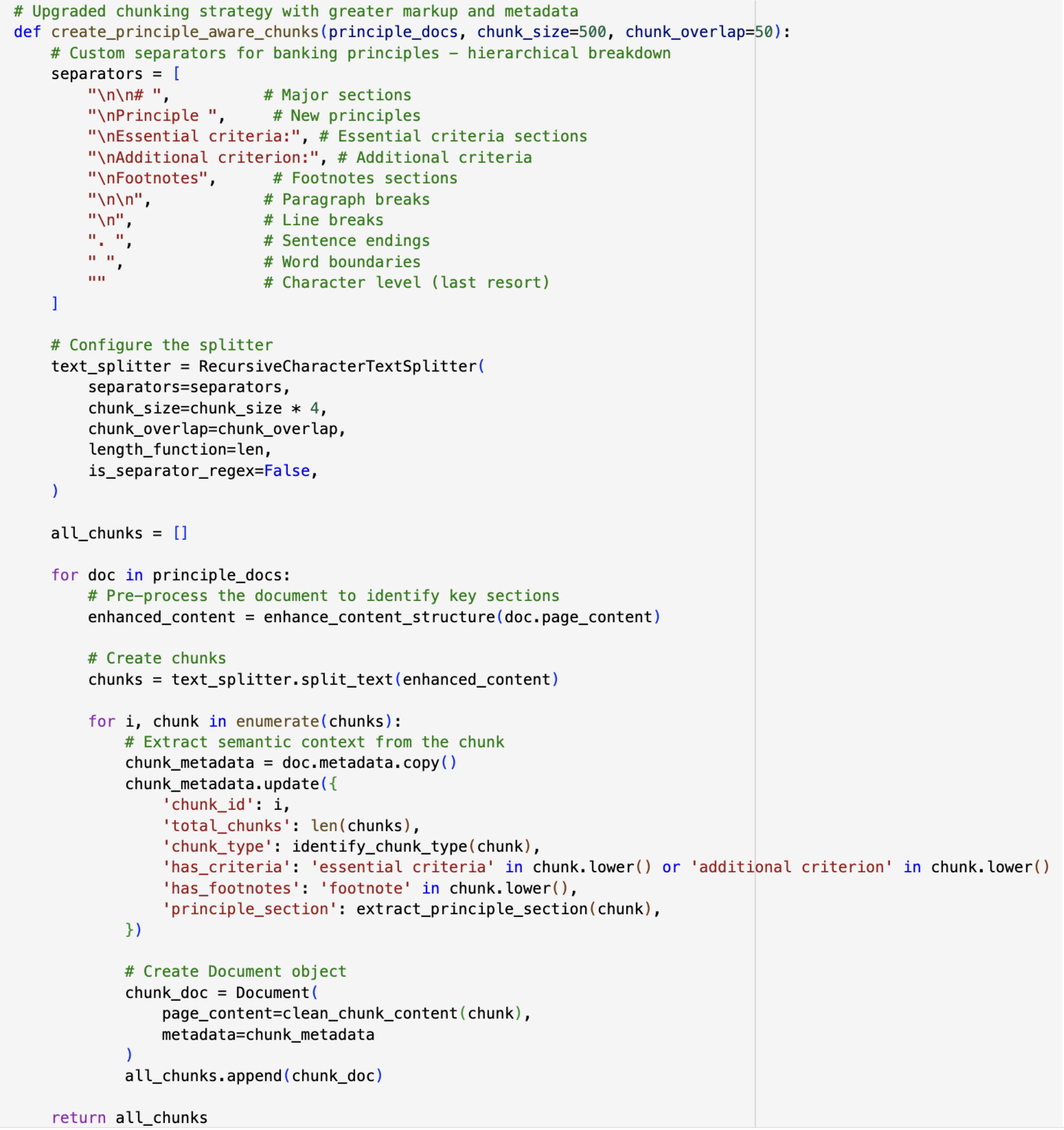

JPM: Hybrid SARIMA + XGBoost Forecast
Actual
Hybrid Forecast
High Risk Count
20
30
40
50
60
2022 Q1
2022 Q2
2022 Q3
2022 Q4
2023 Q1
2023 Q2
2023 Q3
2023 Q4
2024 Q1
2024 Q2
2024 Q3
2024 Q4
2025 Q1
2025 Q2
2025 Q3
2025 Q4
2026 Q1
2026 Q2
2026 Q3
Date


UBS: Hybrid SARIMA + XGBoost Forecast
Actual
Hybrid Forecast
High Risk Count
140
160
180
200
220
240
260
2022 Q1
2022 Q2
2022 Q3
2022 Q4
2023 Q1
2023 Q2
2023 Q3
2023 Q4
2024 Q1
2024 Q2
2024 Q3
2024 Q4
2025 Q1
2025 Q2
2025 Q3
2025 Q4
2026 Q1
2026 Q2
2026 Q3
Date


CITI: Hybrid SARIMA + XGBoost Forecast
Actual
Hybrid Forecast
High Risk Count
20
30
40
50
2022 Q1
2022 Q2
2022 Q3
2022 Q4
2023 Q1
2023 Q2
2023 Q3
2023 Q4
2024 Q1
2024 Q2
2024 Q3
2024 Q4
2025 Q1
2025 Q2
2025 Q3
2025 Q4
2026 Q1
2026 Q2
2026 Q3
Date

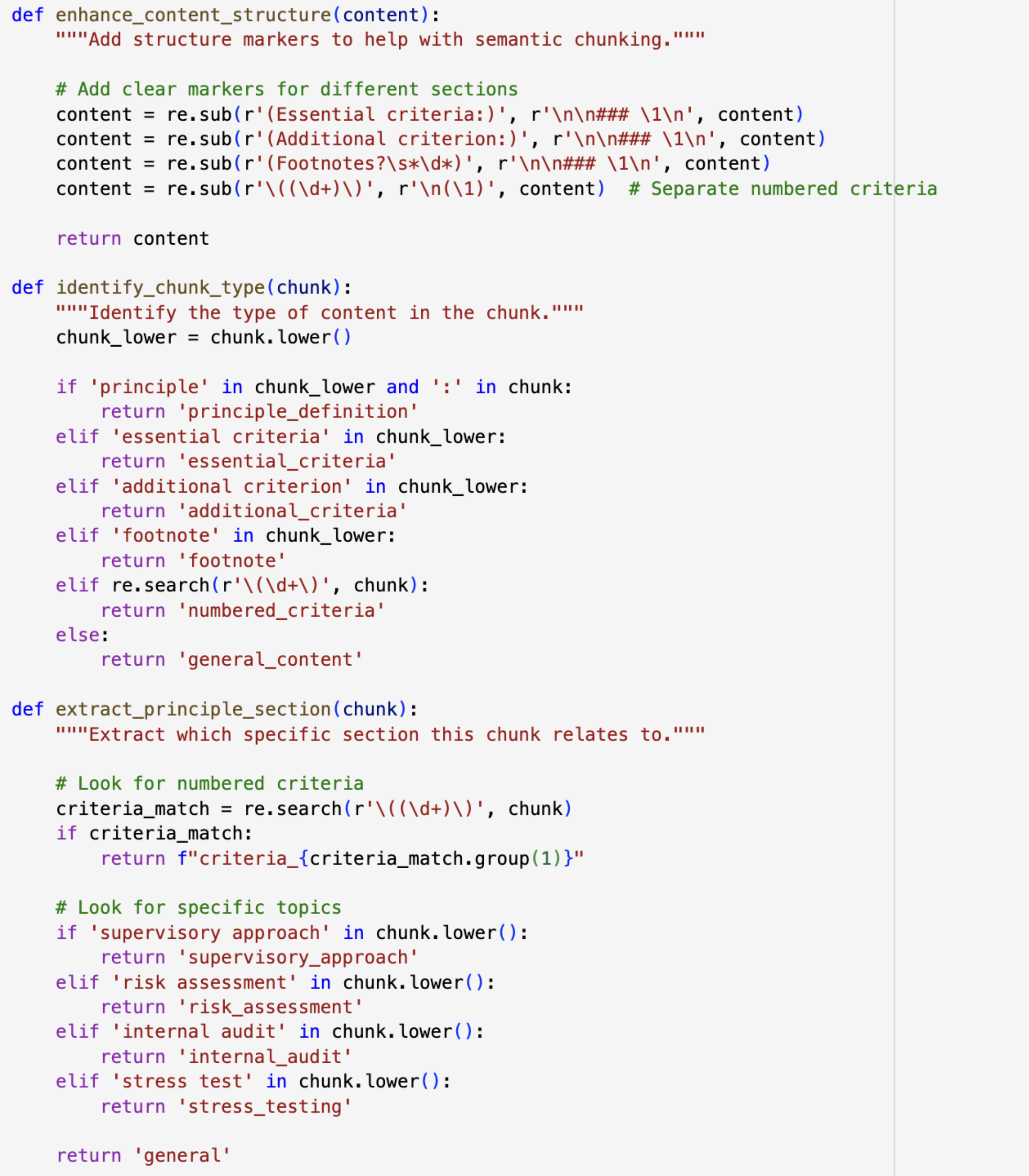
Appendix C

RAG Chunking Code - Block 1



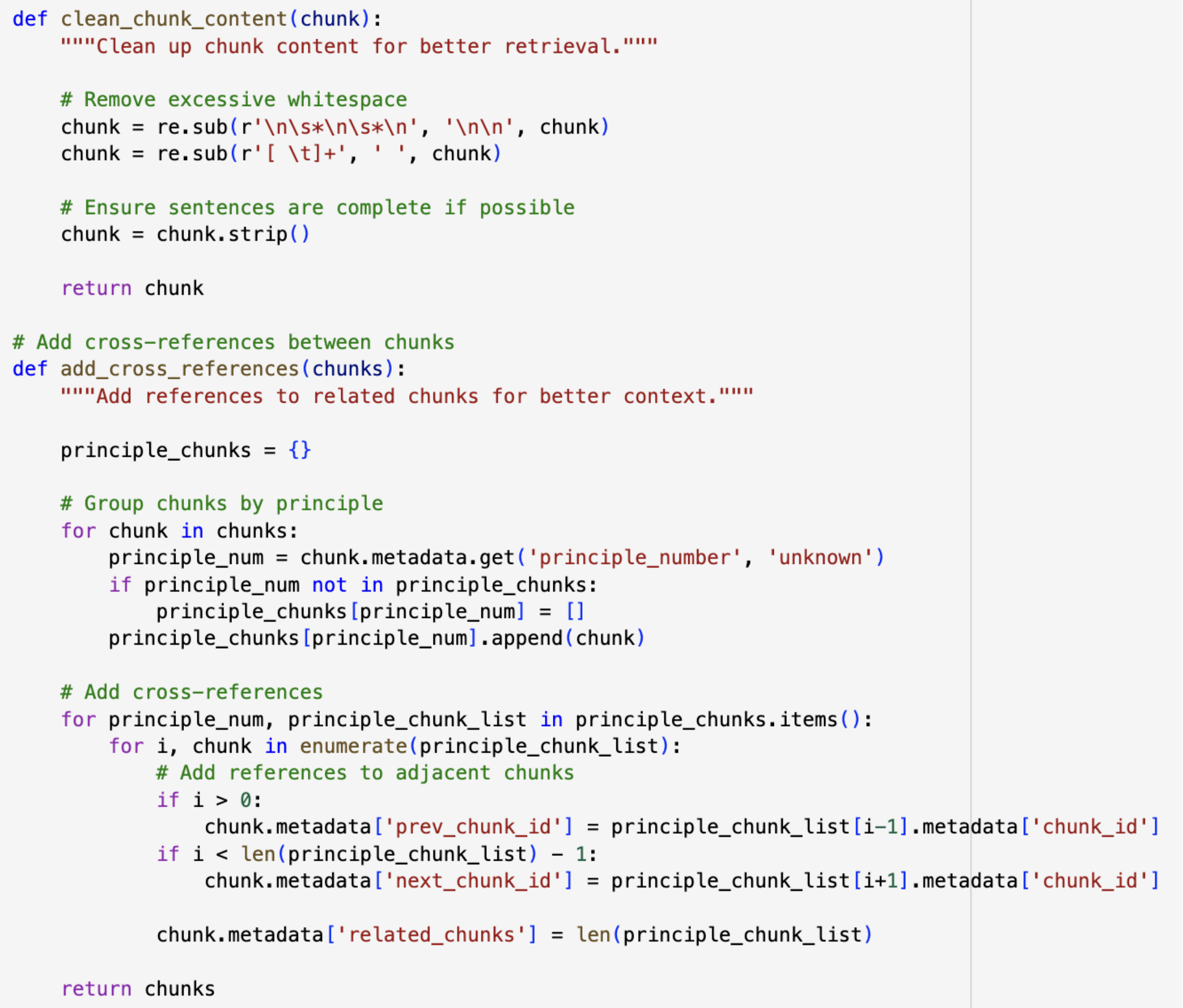
```
# Upgraded chunking strategy with greater markup and metadata
def create_principle_aware_chunks(principle_docs, chunk_size=500, chunk_overlap=50):
    # Custom separators for banking principles - hierarchical breakdown
    separators = [
        "\n\n# ",          # Major sections
        "\nPrinciple ",     # New principles
        "\nEssential criteria:", # Essential criteria sections
        "\nAdditional criterion:", # Additional criteria
        "\nFootnotes",      # Footnotes sections
        "\n\n",            # Paragraph breaks
        "\n",              # Line breaks
        ". ",              # Sentence endings
        " ",               # Word boundaries
        ""                 # Character level (last resort)
    ]

    # Configure the splitter
    text_splitter = RecursiveCharacterTextSplitter(
        separators=separators,
        chunk_size=chunk_size * 4,
        chunk_overlap=chunk_overlap,
        length_function=len,
        is_separator_regex=False,
    )

    all_chunks = []

    for doc in principle_docs:
        # Pre-process the document to identify key sections
        enhanced_content = enhance_content_structure(doc.page_content)

        # Create chunks
        chunks = text_splitter.split_text(enhanced_content)

        for i, chunk in enumerate(chunks):
            # Extract semantic context from the chunk
            chunk_metadata = doc.metadata.copy()
            chunk_metadata.update({
                'chunk_id': i,
                'total_chunks': len(chunks),
                'chunk_type': identify_chunk_type(chunk),
                'has_criteria': 'essential criteria' in chunk.lower() or 'additional criterion' in chunk.lower()
                'has_footnotes': 'footnote' in chunk.lower(),
                'principle_section': extract_principle_section(chunk),
            })

            # Create Document object
            chunk_doc = Document(
                page_content=clean_chunk_content(chunk),
                metadata=chunk_metadata
            )
            all_chunks.append(chunk_doc)

    return all_chunks
```

RAG Chunking Code - Block 2



```python
def enhance_content_structure(content):
    """Add structure markers to help with semantic chunking."""

    # Add clear markers for different sections
    content = re.sub(r'(Essential criteria:)', r'\n\n### \1\n', content)
    content = re.sub(r'(Additional criterion:)', r'\n\n### \1\n', content)
    content = re.sub(r'(Footnotes?\s*\d*)', r'\n\n### \1\n', content)
    content = re.sub(r'\((\d+)\)', r'\n(\1)', content)  # Separate numbered criteria

    return content

def identify_chunk_type(chunk):
    """Identify the type of content in the chunk."""
    chunk_lower = chunk.lower()

    if 'principle' in chunk_lower and ':' in chunk:
        return 'principle_definition'
    elif 'essential criteria' in chunk_lower:
        return 'essential_criteria'
    elif 'additional criterion' in chunk_lower:
        return 'additional_criteria'
    elif 'footnote' in chunk_lower:
        return 'footnote'
    elif re.search(r'\(\d+\)', chunk):
        return 'numbered_criteria'
    else:
        return 'general_content'

def extract_principle_section(chunk):
    """Extract which specific section this chunk relates to."""

    # Look for numbered criteria
    criteria_match = re.search(r'\((\d+)\)', chunk)
    if criteria_match:
        return f"criteria_{criteria_match.group(1)}"

    # Look for specific topics
    if 'supervisory approach' in chunk.lower():
        return 'supervisory_approach'
    elif 'risk assessment' in chunk.lower():
        return 'risk_assessment'
    elif 'internal audit' in chunk.lower():
        return 'internal_audit'
    elif 'stress test' in chunk.lower():
        return 'stress_testing'

    return 'general'
```

RAG Chunking Code - Block 3



```python
def clean_chunk_content(chunk):
    """Clean up chunk content for better retrieval."""

    # Remove excessive whitespace
    chunk = re.sub(r'\n\s*\n\s*\n', '\n\n', chunk)
    chunk = re.sub(r'[ \t]+', ' ', chunk)

    # Ensure sentences are complete if possible
    chunk = chunk.strip()

    return chunk

# Add cross-references between chunks
def add_cross_references(chunks):
    """Add references to related chunks for better context."""

    principle_chunks = {}

    # Group chunks by principle
    for chunk in chunks:
        principle_num = chunk.metadata.get('principle_number', 'unknown')
        if principle_num not in principle_chunks:
            principle_chunks[principle_num] = []
        principle_chunks[principle_num].append(chunk)

    # Add cross-references
    for principle_num, principle_chunk_list in principle_chunks.items():
        for i, chunk in enumerate(principle_chunk_list):
            # Add references to adjacent chunks
            if i > 0:
                chunk.metadata['prev_chunk_id'] = principle_chunk_list[i-1].metadata['chunk_id']
            if i < len(principle_chunk_list) - 1:
                chunk.metadata['next_chunk_id'] = principle_chunk_list[i+1].metadata['chunk_id']

            chunk.metadata['related_chunks'] = len(principle_chunk_list)

    return chunks
```