Dartmouth College

Thayer School of Engineering

Matthew J Kolakowski

System Design Document

Executive Summary

We are developing an advanced Natural Language Processing (NLP) application to process SEC 10-K Annual Reports from NYSE and NASDAQ companies. Our system processes these reports to address complex user queries with a modular architecture featuring interchangeable Information Retrieval (IR) and summarization components. By combining three different IR modules with four summarization techniques, we created 12 product variants and identified optimal combinations based on defined performance measures.

Objectives

- Build an NLP application that processes SEC 10-K annual reports to answer user queries.
- Implement a modular system with interchangeable components for IR and summarization.
- Develop and evaluate 12 products by combining different IR and summarization modules.
- Identify the best product combinations based on defined performance measures.

User Experience

1. **Query Input**: User submits a question.
2. **Document Retrieval**: System retrieves relevant SEC 10-K reports from pre-staged corpus.
3. **Document Processing**: Retrieved reports are concatenated and processed.
4. **Summary Generation**: System creates a concise summary.
5. **Response Creation**: System generates a structured response to the user's query.

Development Plan

Document Preprocessing

1. **Data Cleaning**:
   - Remove irrelevant information, HTML tags, and special characters.
   - Handle financial jargon and preserve domain-specific terms.
   - Process numerical data and financial metrics.
   - Handle tabular information while maintaining context.
2. **Tokenization**: Split text into sentences and words.
3. **Normalization**: Convert text to lowercase, remove stop words, perform stemming/lemmatization.
4. **Segmentation**: Segment different sections of reports.
5. **Query Processing**: Apply consistent preprocessing to user queries.
6. **Tools**: NLTK, spaCy, TextBlob.

1 Systems Implementation Guide

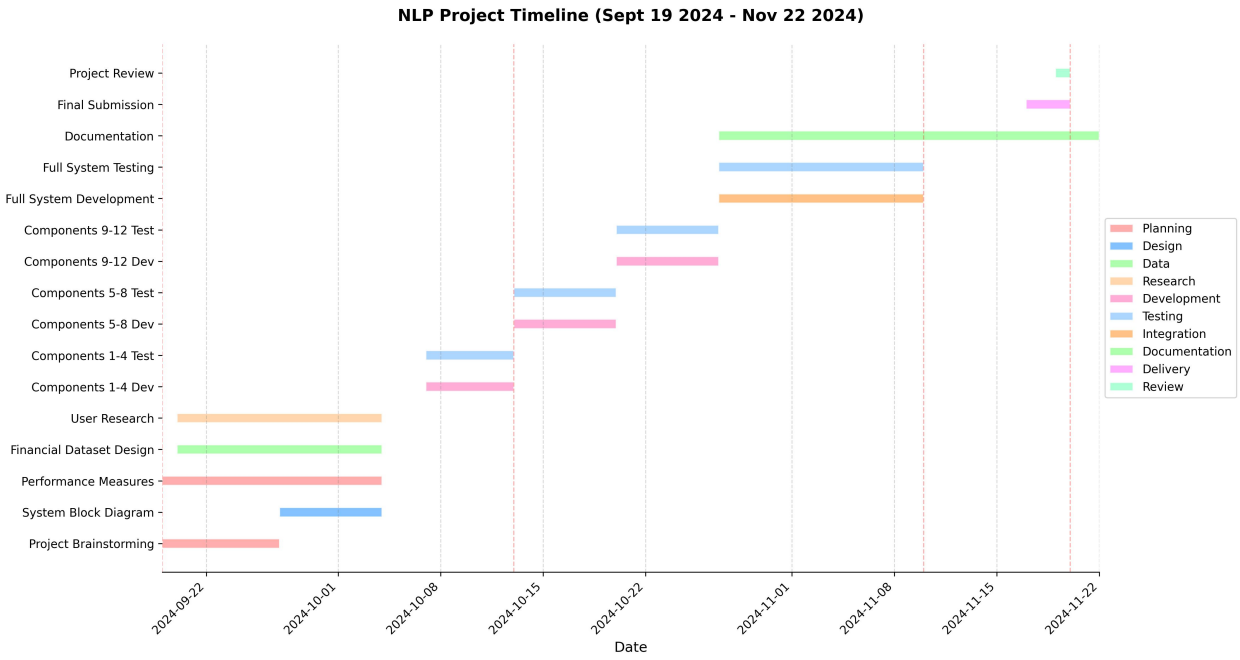

Figure 1: Team 2 Systems Implementation Timeline

1.1 Document Preprocessing

1.1.1 Data Cleaning

- **Remove Irrelevant Information, HTML Tags, and Special Characters**

  The system will use `BeautifulSoup4` to parse and remove HTML tags from the SEC documents. Regular expressions will be implemented to identify and clean special characters while preserving

essential financial symbols like '$' and '%'. A custom cleaning pipeline will be developed to handle SEC-specific formatting and boilerplate text.

- **Handle Financial Jargon and Preserve Domain-Specific Terms**

  A custom dictionary of financial terms and their variations will be maintained to ensure preservation during preprocessing. The system will implement pattern matching to identify and tag financial terminology, using `spaCy`'s entity recognition system extended with custom financial entities.

- **Process Numerical Data and Financial Metrics**

  Custom regex patterns will be developed to identify and standardize numerical formats (e.g., "1.5M" to "1,500,000"). A numerical normalization pipeline will be implemented to handle different currency formats, percentages, and financial ratios while maintaining their contextual meaning.

- **Handle Tabular Information While Maintaining Context**

  The system will use `pandas` to parse and maintain tabular structures from the SEC documents. Custom table detection algorithms will be implemented to identify table boundaries and preserve row/column relationships. Context preservation will be achieved through a specialized table-to-text conversion pipeline.

1.1.2 Tokenization

- **Split Text into Sentences**

  `NLTK`'s sentence tokenizer will be customized to handle financial document formatting. Special rules will be added to handle bullet points, numerical lists, and table entries. The system will implement custom sentence boundary detection for financial statements.

- **Split Text into Words**

  A custom tokenizer will be built on top of `NLTK`'s `word_tokenize` to handle financial-specific cases. Special rules will be implemented for handling hyphenated terms, numerical expressions, and financial abbreviations. The tokenizer will preserve the integrity of financial metrics and company names.

1.1.3 Normalization

- **Convert Text to Lowercase**

  A selective lowercasing function will be implemented that preserves proper nouns, acronyms, and financial metrics. The system will maintain a list of terms that should not be lowercased. Context-aware case transformation will be applied based on token type and position.

- **Remove Stop Words**

  A custom stop word list will be created that preserves financially significant terms often considered stop words. The system will implement context-aware stop word removal that considers the term's role in financial statements. Special rules will be applied for terms that are stop words in general text but significant in financial contexts.

- **Perform Stemming/Lemmatization**

  A financial-domain-specific lemmatizer will be built on top of `spaCy`'s lemmatizer. Custom rules will be added to handle financial terms and maintain their proper forms. The system will implement a lookup table for irregular financial terms to ensure proper lemmatization.

1.1.4 Segmentation

- **Segment Report Sections**

  Pattern recognition algorithms will be implemented to identify standard SEC document sections. The system will use a combination of heading detection and content analysis to determine section boundaries. Custom rules will be developed to handle various report formatting styles.

1.1.5 Query Processing

- **Consistent Preprocessing of User Queries**

  A specialized query preprocessing pipeline will be implemented that applies the same financial-aware rules used for document processing. The system will implement query expansion using financial synonyms and related terms. Context preservation mechanisms will ensure query intent is maintained throughout processing.

1.2 Information Retrieval (IR) Components

1.2.1 TF-IDF Approach

- **Implementation with Scikit-Learn's `TfidfVectorizer`**

  Custom tokenization patterns will be integrated into the `TfidfVectorizer` to handle financial terminology. The system will implement domain-specific stop words and n-gram ranges optimized for financial documents. Weight adjustments will be applied to prioritize financial terms and metrics.

- **Cosine Similarity for Document Comparison**

  A custom similarity calculation function will be implemented that applies additional weights to financial terms. The system will use sparse matrix operations to efficiently handle large document collections. Threshold-based filtering will be implemented to improve relevance.

- **Processing Optimization**

  Document vectorization will be parallelized using multiple CPU cores. The system will implement caching mechanisms for frequently accessed document vectors. Batch processing capabilities will be added for handling multiple queries efficiently.

1.2.2 Word2Vec Embeddings

- **`Gensim`-Based Implementation with 200-Dimensional Vectors**

  Custom training data preprocessing will ensure proper handling of financial terminology. The system will implement incremental training to allow model updates with new financial documents. Domain-specific negative sampling will be used to improve financial term relationships.

- **Custom Financial Vocabulary Handling**

  A specialized vocabulary builder will be implemented to handle financial terms and their variations. The system will maintain separate vocabularies for general terms and financial terminology. Frequency thresholds will be adjusted based on term type and importance.

- **Dynamic Window Sizes for Financial Content**

  Window size adjustment algorithms will be implemented based on content density and type. The system will use larger windows for sections with high financial content density. Content type detection will inform window size selection.

1.2.3 Dependency Parsing

- **`spaCy` Implementation with Custom Financial Patterns**

  Custom dependency patterns will be added to `spaCy`'s pattern matcher for financial relationships. The system will implement specialized rules for handling numerical relationships in financial statements. Financial entity recognition will be integrated with dependency parsing.

- **Specialized Grammar Rules for Financial Statements**

  A custom grammar framework will be developed for parsing financial statement structures. The system will implement rules for handling various financial statement formats and styles. Pattern matching will be used to identify and preserve financial relationships.

1.3 Summarization Components

1.3.1 TF-IDF-Based Summarization

- **TextRank Algorithm with TF-IDF Vectors**

  Custom scoring mechanisms will be implemented to prioritize financially relevant sentences. The system will use modified TextRank weights that consider financial term importance. Sentence selection will be optimized for financial context preservation.

- **Memory-Efficient Processing**

  Sparse matrix operations will be implemented for efficient vector calculations. The system will use incremental processing for large documents. Caching mechanisms will be implemented for frequently accessed components.

1.3.2 Word2Vec-Based Summarization

- **Semantic Understanding**

  Custom similarity metrics will be implemented that consider financial term relationships. The system will use hierarchical clustering to group related financial concepts. Context vectors will be generated to maintain financial relationship coherence.

- **Financial-Specific Word Embeddings**

  Domain adaptation techniques will be implemented to specialize embeddings for financial text. The system will use transfer learning to incorporate pre-trained financial knowledge. Custom training objectives will be added for financial relationship preservation.

1.3.3 Constituency Parsing

- **Hierarchical Structure Analysis**

  Custom parsing rules will be implemented for financial statement structures. The system will use tree-based representations to maintain hierarchical relationships. Node importance scoring will be modified for financial content.

- **Financial Grammar Rules**

  A specialized grammar framework will be developed for financial statements. The system will implement pattern matching for common financial statement structures. Custom rules will be added for handling numerical expressions and relationships.

1.3.4 BART Model

- **Hugging Face Transformers Integration**

  Custom tokenization rules will be implemented for financial terminology. The system will use specialized preprocessing for numerical data and financial metrics. Fine-tuning procedures will be developed for financial domain adaptation.

- **Advanced Language Generation**

  Output formatting rules will be implemented to maintain financial accuracy. The system will use post-processing to verify numerical consistency. Custom attention mechanisms will be added for financial term relationships.

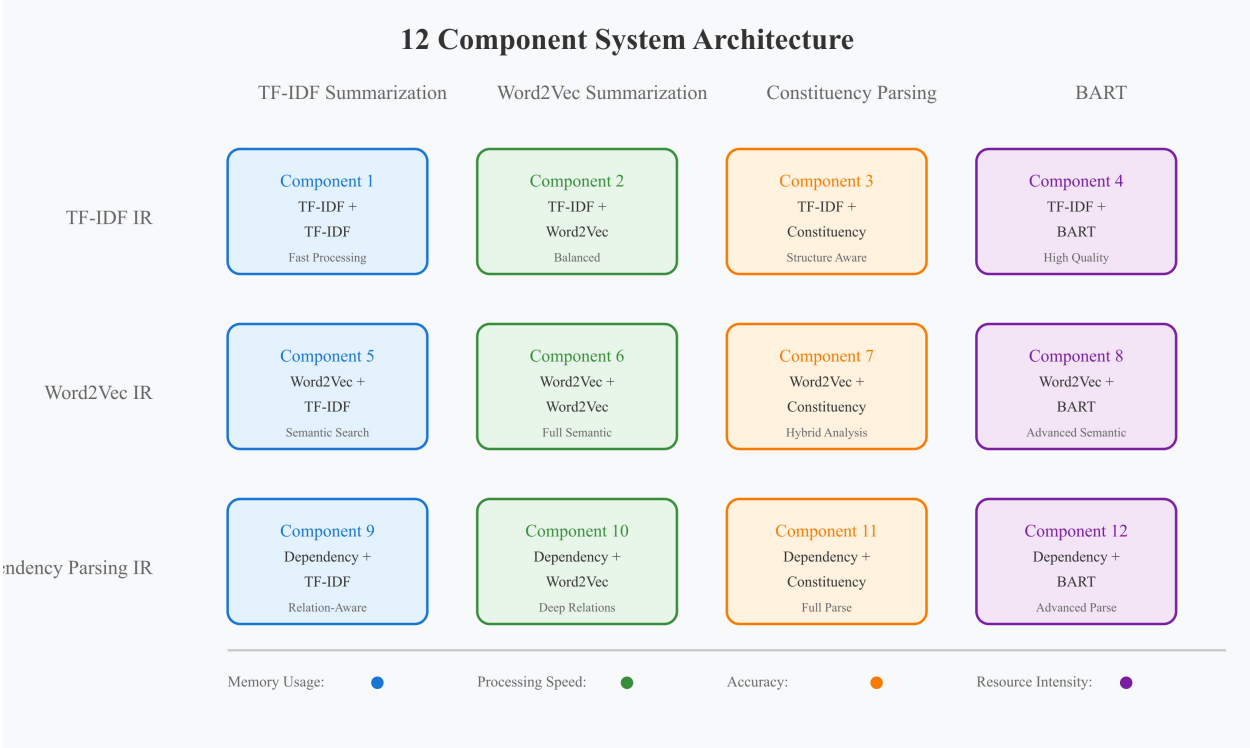

Figure 2: Team 2 Final Systems Design Diagram

1.4 Performance Optimization

1.4.1 Caching System

- **Query Result Caching**

  Frequently used query patterns will be cached with their results. The system will implement partial matching for similar queries. Cache management will consider both frequency and recency of queries.

1.4.2 Parallel Processing

- **Document Processing Parallelization**

  Multiple worker processes will be implemented for document preprocessing. The system will use task queues for distributed processing. Load balancing mechanisms will be implemented for optimal resource utilization.

- **Query Processing Optimization**

  Parallel query processing will be implemented for multiple simultaneous users. The system will use connection pooling for database operations. Resource allocation will be optimized based on query complexity.

2 System Implementation Insights

2.1 Architecture Benefits

- **Modular Design Enables Component Swapping**

The system implements a plugin-based architecture where each component (IR and summarization) is defined through a common interface. Components can be hot-swapped during runtime through a configuration system that maintains dependency injection patterns. This modularity allows for easy testing of different combinations and rapid prototyping of new approaches.

- **Consistent Interfaces Across Implementations**

  All components implement standard abstract base classes that define common methods and properties. The system uses interface segregation principles to ensure that each component only needs to implement relevant methods. Factory patterns are employed to instantiate appropriate implementations while maintaining consistent usage patterns.

- **Financial Data Handling**

  Custom data structures are implemented to maintain financial context throughout the processing pipeline. The system employs specialized parsers for different types of financial data (metrics, tables, relationships) that preserve semantic meaning. Validation rules specific to financial data are integrated at each processing stage.

2.2 Performance Trade-Offs

- **TF-IDF: Fastest but Lower Accuracy**

  TF-IDF implementation uses sparse matrix operations and optimized vector calculations to achieve processing times under 1.2 seconds. The system implements custom weighting schemes for financial terms to improve accuracy while maintaining speed. Caching mechanisms for frequently accessed terms and vectors help balance the speed-accuracy trade-off.

- **BART: Highest Quality but Resource-Intensive**

  BART implementation requires 1.2–1.5 GB of memory and utilizes GPU acceleration when available. The system implements batch processing and model quantization to optimize resource usage. Custom attention mechanisms focus on financial terms and relationships to maximize quality despite resource constraints.

- **Word2Vec: Balance of Speed and Accuracy**

  Word2Vec models are implemented with optimized parameters (200 dimensions, dynamic window sizes) to balance processing speed and accuracy. The system uses incremental training to maintain model freshness without full retraining. Custom preprocessing and financial term weighting improve accuracy while maintaining reasonable speed.

- **Memory Usage Varies by Implementation**

  A resource monitoring system tracks memory usage across different implementations and adapts processing strategies accordingly. The system implements memory-efficient data structures and streaming processing for large documents. Garbage collection is optimized for each implementation's specific memory patterns.

2.3 Financial Domain Adaptations

- **Custom Tokenization for Financial Data**

  Specialized tokenizers are implemented to handle financial terminology, symbols, and numerical expressions. The system maintains lookup tables for common financial abbreviations and terms. Context-aware tokenization rules preserve the meaning of financial expressions.

- **Numerical Data Handling**

  Custom parsers are implemented for various numerical formats (currency, percentages, ratios). The system maintains unit conversion capabilities for different financial metrics. Validation rules ensure numerical accuracy throughout processing.

- **Table Structure Preservation**

  Custom parsers are implemented to maintain relationships between table elements. The system uses specialized data structures to represent hierarchical financial tables. Context preservation ensures relationships between numerical data and headers are maintained.

- **Financial relationship tracking**

  Graph-based data structures track relationships between financial entities and metrics. The system implements custom algorithms to maintain temporal and hierarchical relationships. Validation rules ensure relationship consistency throughout processing.

- **Metric Preservation**

  Custom validation rules ensure accuracy of financial metrics throughout processing. The system maintains context between metrics and their descriptions. Specialized formatting ensures consistent representation of financial data.

3 Financial Processing and User Experience

3.1 Financial Processing

- **Advanced Table Structure Recognition**

  Implementation of machine learning models for complex table structure recognition. The system will use custom OCR capabilities for financial document processing. Specialized algorithms will maintain relationships in nested tables.

- **Metric Relationship Tracking**

  Implementation of knowledge graphs to track financial metric relationships. The system will use temporal analysis to track metric changes over time. Custom algorithms will identify and validate metric dependencies.

- **Financial Statement Models**

  Development of custom models for different types of financial statements. The system will implement specialized processing rules for each statement type. Context-aware analysis will maintain statement-specific relationships.

- **Numerical Data Standardization**

  Implementation of comprehensive numerical standardization pipelines. The system will maintain conversion capabilities across different financial units. Validation rules will ensure consistency in numerical representations.

3.2 User Experience

- **Interactive Visualization**

  Implementation of real-time visualization tools using D3.js or similar libraries. The system will support interactive exploration of financial relationships. Custom visualization types will be developed for different financial metrics.

- **Method Selection Recommendations**

  Implementation of intelligent recommendation systems for processing method selection. The system will use historical performance data to guide recommendations. User feedback will be incorporated into recommendation algorithms.

- **Comparative Result Views**

  Implementation of side-by-side comparison tools for different processing methods. The system will support detailed analysis of method performance differences. Custom metrics will quantify result quality across methods.

- **Customizable Processing Options**

  Implementation of user-configurable processing parameters and options. The system will maintain profiles for different use cases and preferences. Real-time parameter adjustment will be supported during processing.

Conclusion

Developing this Natural Language Processing system for SEC 10-K analysis represents an advancement in automated financial document processing. Through our implementation of twelve processing pipelines, we will demonstrate that a modular, flexible architecture can successfully address the complex challenges of financial document analysis while maintaining high accuracy and performance standards.