# **Research Overview: 2D Contextual Transformer for Spreadsheet Analysis**

## **1. Project Foundation and Motivation**

### **1.1 Inspiration and Core Challenge**

The project emerged from a fundamental observation about transformer architectures. While transformers excel at processing sequences in parallel and bidirectionally, making them faster and more efficient than traditional RNNs, they primarily work in a 1D structure suited for language tasks (left-to-right or right-to-left processing). This creates a mismatch with spreadsheet data, which inherently presents a 2D contextual structure where each cell's content depends on both row and column relationships.

The hypothesis driving this research is that building a model with native 2D contextual understanding would not only enhance spreadsheet comprehension for complex tasks but also improve training efficiency compared to traditional language models. This approach represents a fundamental shift from conventional methods that treat spreadsheets as flattened sequences.

### **1.2 Vision for Financial Analysis**

The long-term vision extends beyond simple metadata prediction to develop an AI model that can analyze various types of spreadsheets with the sophistication of a financial consultant. The envisioned system would:

1. Process multiple financial document types simultaneously:
   * Balance sheets
   * P&L statements
   * Expense reports
   * Inventory lists
2. Generate comprehensive financial insights:
   * Analyze P&L statements and balance sheets in conjunction
   * Provide detailed comments on profitability, liquidity, and debt levels
   * Reference specific cells or rows in expense/inventory sheets to justify insights
   * Function as a comprehensive, spreadsheet-based financial consultant

### **1.3 Initial Scope and Task Definition**

To make the research tractable, we initially focused on a fundamental yet crucial task: metadata prediction. This involved:

1. Primary Task: Predicting 17 different types of metadata (e.g., bold, italic, underline, alignment, font size) for each cell based on its content
2. Simplified Version: Binary classification of boldness (1 for bold, 0 for non-bold) as a proof of concept
3. Data Structure: Each spreadsheet limited to 100x100 cells for computational feasibility

## **2. Technical Implementation**

### **2.1 Data Processing Infrastructure**

#### **Dataset Creation and Expansion**

1. Initial Dataset:
   * Started with Enron dataset (622 spreadsheets in .xls, .xlsx, and .csv formats)
   * Identified limitations in older files: outdated formats, insufficient metadata, parsing errors
2. Custom Web Parser Development:
   * Built using BeautifulSoup and asyncio with async and ClientSession()
   * Implemented 5 concurrent requests for efficient data.gov scraping
   * Validation criteria:
     + File type: .xls and .xlsx only
     + Response time: Skip URLs with >1 second response
     + Size: Limited to under 2MB
     + Parsing compatibility: Immediate post-download validation
3. Final Dataset Composition:
   * 800 training files
   * 100 validation files
   * 100 test files
   * Created subsets (teeny, micro, tiny, small, medium, big) for structured experimentation

### **2.2 Data Representation and Preprocessing**

#### **Vocabulary and Token Management**

The project implements a sophisticated token management system through a custom vocabulary class:

1. Special Token Integration:
   * <cls>: Classification token
   * <eos>: End of sequence marker
   * <unk>: Unknown token handler
   * <pad>: Padding token
2. Embedding Creation:
   * Utilizes GloVe-50 model
   * Produces 50-dimensional vectors for each token
   * Handles out-of-vocabulary words through random normal initialization

#### **Data Structuring**

1. Basic Tensor Structure:
   * Each spreadsheet: PyTorch tensor
   * Dimensions: 100 rows x 100 columns
   * Cell content: Tokenized and padded/truncated to 32 tokens
   * Final content tensor (x\_tok): 100x100x32
2. Metadata Representation:
   * Metadata tensor (y\_tok): 100x100x17
   * Each metadata type assigned specific positions
   * Binary encoding (e.g., position 6 for boldness: 1=bold, 0=non-bold)

#### **Processing Pipeline**

1. File Parsing:
   * Utilizes pandas, numpy, openpyxl, xlrd, and csv libraries
   * Extracts content and metadata into x\_tok and y\_tok tensors
   * Automatic exclusion of unparseable files
2. Batch Processing:
   * Custom SpreadsheetDataLoader for handling multiple tensors and file paths
   * Parallel processing using Parallel and joblib
   * Optimal CPU utilization: os.cpu\_count() // 4 jobs

### **2.3 Model Architecture Evolution**

#### **Approach Philosophy**

Two distinct approaches were developed and tested:

1. Global Context (Approach A):
   * Processes each cell's content with context from all surrounding cells
   * Comprehensive but computationally intensive
2. Local Context (Approach B):
   * Focuses on row and column context only
   * Based on the hypothesis that a cell's content is primarily influenced by its immediate row and column values

#### **Model Implementations**

##### **1. SimpleGeluEmbedAvg**

A lightweight neural model optimized for binary classification:

* Input: Token embeddings from custom vocabulary
* Process: Averages embeddings and applies GeLU activation
* Output: Binary prediction through feed-forward layer
* Performance:
  + Speed: 40 sheets/epoch in 2-3 seconds
  + Memory: 2-3GB GPU RAM
  + Accuracy: 0.99, 1.00, 0.98 F1-score on train/val/test

##### **2. TestRNN (Based on Approach A)**

Detailed Forward Pass Algorithm for batch size 8:

1. Global Context Calculation:
   * Initialize H\_local tensor (8 x 10000 x 100)
   * Loop over cells using 1D indexing
   * For each cell:
     + Retrieve 32 tokens across batches (8 x 32 tensor)
     + Apply embedding layer (8 x 32 x 50 tensor)
     + Apply dropout (rate 0.05)
     + Process through RNN layer
     + Extract final hidden state
     + Store in H\_local
2. Context Integration:
   * Calculate total context by summing hidden states
   * Subtract each cell's state for exclusive global context
   * Create final tensor (8 x 10000 x 100)
3. Prediction Computation:
   * Initialize S\_cube (8 x 100 x 100)
   * For each cell:
     + Generate local context
     + Combine with global context
     + Produce final prediction

Performance:

* Training time: 2:30 hours per epoch
* Memory usage: 40GB GPU
* Results: Tendency to predict entire text areas as bold

##### **3. BERT Adaptations**

Custom architecture with configurable parameters:

* No pre-existing weights
* Customizable hidden\_size, intermediate\_size, num\_hidden\_layers, num\_attention\_heads
* Performance:
  + Memory: 36GB RAM
  + Time: 2:45 minutes per epoch
  + Accuracy: >0.98 F1 scores on manual dataset

##### **4. 2D Positional Enhancement**

Recent improvements through 2D positional encoding:

* Custom implementation for explicit row/column position embedding
* Position encoding added after cell content processing
* Significant improvements:
  + Perfect F1 scores on baseline dataset
  + 0.68 F1 on generalized training/validation
  + 0.27 F1 on test dataset

### **2.4 Training Infrastructure**

#### **Training Loop Design**

1. Initial Configuration:
   * Dynamic batch\_size, learning rate, and device selection
   * Positive weight calculation for class imbalance handling
   * BCEWithLogitsLoss implementation with class weighting
2. Loop Structure:
   * Configurable epochs, patience, and save intervals
   * Early stopping mechanism based on validation performance
   * State saving at specified intervals
3. Training Phase:
   * Forward pass computation
   * Loss calculation with class weighting
   * Gradient clipping to prevent explosion
   * Optimizer updates (Adagrad)
   * Memory optimization through variable clearing
4. Validation Phase:
   * Model evaluation without gradient computation
   * Perplexity calculation for both datasets
   * Best model tracking and early stopping implementation
5. Logging and Model Preservation:
   * Comprehensive metric logging
   * State dictionary saving at intervals
   * Final model selection based on validation perplexity

## **3. Current State and Results**

### **3.1 Model Performance Analysis**

Detailed evaluation across different architectures:

1. SimpleGeluEmbedAvg:
   * Consistent high performance on controlled datasets
   * Excellent efficiency metrics
   * Limited capability for complex patterns
2. RNN Implementation:
   * Substantial computational overhead
   * Pattern recognition limitations
   * Valuable insights for architecture improvement
3. BERT Variants:
   * Promising results on structured data
   * Resource intensity challenges
   * Foundation for positional encoding advancements

### **3.2 Recent Advancements**

#### **SAFFU Integration**

Development of Self-Attentive Feed-Forward Units:

* Explicit optimization techniques
* Reduced training costs
* Maintained accuracy levels
* Custom byte-pair encoding tokenizer

#### **Graph Neural Network Exploration**

Collaboration with Dr. Edward Kim's Sparse Lab:

* Natural modeling of cell relationships
* Enhanced contextual understanding
* Potential for improved metadata handling

## **4. Future Directions**

### **4.1 Technical Roadmap**

1. 2D Positional Embeddings:
   * Extended sinusoidal function implementation
   * Enhanced spatial context modeling
   * Integration with existing architectures
2. SAFFU Layer Integration:
   * Complete integration of optimization techniques
   * Performance optimization
   * Scalability improvements
3. Dataset Enhancement:
   * Expansion to more diverse spreadsheet types
   * Improved preprocessing techniques
   * Robust validation methods

### **4.2 Collaboration Opportunities**

#### **SpreadsheetLLM Integration**

Potential collaboration areas with Microsoft:

1. Technical Integration:
   * SHEETCOMPRESSOR framework adoption
   * Structural-anchor-based compression
   * Inverted-index translation
   * Data-format-aware aggregation
2. Mutual Benefits:
   * Shared focus on 2D context preservation
   * Efficiency optimization techniques
   * Joint development opportunities
   * Enhanced interpretability methods

### **4.3 Research Impact**

The project contributes to the field through:

1. Novel 2D context implementation in transformers
2. Efficient processing techniques
3. Advanced optimization methods
4. Practical applications in financial analysis

The ongoing development continues to push boundaries in spreadsheet processing technology while maintaining a focus on practical applications and computational efficiency.