Finance Domain Chatbot - Technical Report

Project Type: Individual Transformer-Based NLP Application
Domain: Finance & Personal Investment
Date: January 2025
Model: Google FLAN-T5-Base (247M parameters)

Github URL: https://github.com/cyloic/Finance_Chatbot_Summative.git

Demo Video: https://youtu.be/pxgBPszQvCQ

1. Executive Summary

This project implements a domain-specific conversational AI chatbot for the finance sector using Google's FLAN-T5-base transformer model. The system was fine-tuned on 430 finance question-answer pairs and deployed via a Gradio web interface. The chatbot achieves a ROUGE-1 F1 score of 0.38+ on test data, demonstrating effective learning of financial concepts and terminology. Training converged in 15 epochs with a final training loss of 0.87 and validation loss of 1.05.

Key Achievements:

- Successfully fine-tuned a 247M parameter transformer model
- Achieved sub-1.0 training loss indicating strong learning
- Deployed interactive web interface with 2-3 second response time
- Demonstrated domain-specific knowledge retention
- Implemented optimized generation parameters for quality outputs

2. Problem Definition & Domain Justification

2.1 Problem Statement

Financial literacy remains a significant barrier for individuals seeking to make informed investment decisions. Key challenges include:

- Limited access to financial advisors (cost/availability barriers)
- Complex financial jargon and concepts
- Need for instant, 24/7 access to financial information
- Difficulty finding reliable, domain-specific answers

2.2 Domain Selection Rationale

Finance domain was selected for:

1. High Impact: Financial decisions directly affect quality of life and long-term wealth
2. Specialized Knowledge: Requires domain-specific vocabulary (ROI, diversification, compound interest, etc.)
3. Clear Evaluation: Finance concepts have objectively correct explanations
4. Accessibility Need: Democratizes financial knowledge for underserved populations

2.3 Solution Approach

Developed a generative QA chatbot that:

- Understands natural language finance questions
- Generates human-like, accurate explanations
- Maintains domain focus (rejects off-topic queries)
- Provides instant responses via web interface

3. Dataset & Preprocessing

3.1 Dataset Specification

Attribute	Value
Total Samples	430 Q&A pairs
Domain Coverage	Investments, banking, personal finance, financial concepts
Format	CSV (2 columns: question, answer)
Source File	Finance_QA_300.csv
Average Question Length	~50 characters
Average Answer Length	~75 characters

Sample Data Points:

Q: What is compound interest?
A: Interest calculated on the initial principal and accumulated interest from previous periods.

Q: What's the difference between stocks and bonds?
A: Stocks represent ownership in a company, while bonds are loans to companies or governments.

3.2 Data Split Strategy

Training Set:	344 samples (80%)
Validation Set:	43 samples (10%)
Test Set:	43 samples (10%)
Random Seed:	42 (for reproducibility)

Rationale: 80/10/10 split ensures sufficient training data while maintaining robust validation and independent test evaluation.

3.3 Preprocessing Pipeline

Step 1: Data Cleaning (Cell 5)

```
def clean_text(text):
    text = str(text).strip()
    text = ' '.join(text.split())  # Remove extra whitespace
    return text
```

- Removed leading/trailing whitespace
- Normalized multiple spaces to single space
- Handled missing values (removed empty rows)
- Converted all text to string type

Step 2: Tokenization (Cell 8)

```
tokenizer = T5Tokenizer.from_pretrained("google/flan-t5-base")
max_length = 256 tokens (input and output)
```

Tokenization Method: SentencePiece (subword tokenization)

- Advantages: Handles out-of-vocabulary words, language-agnostic
- Configuration:

    - Input format: "Question: {question}\nAnswer:"
    - Max length: 256 tokens
    - Padding: Applied to max_length
    - Truncation: Enabled

Step 3: Label Processing

```
# Replace padding tokens with -100 in labels
labels["input_ids"] = [
    [(label if label != tokenizer.pad_token_id else -100)
     for label in labels_example]
    for labels_example in labels["input_ids"]
]
```

Purpose: Ignore padding tokens during loss calculation (standard practice for seq2seq models).

Step 4: Dataset Conversion

- Converted pandas DataFrame to HuggingFace Dataset format
- Applied preprocessing function with batching
- Created DataCollatorForSeq2Seq for dynamic padding

Preprocessing Documentation: All steps documented in notebook Cells 4-8 with inline comments explaining rationale.

4. Model Architecture & Fine-Tuning

4.1 Model Selection

Selected Model: google/flan-t5-base

Architecture Specifications:

Type:	Encoder-Decoder Transformer
Parameters:	247,577,856 (all trainable)
Encoder Layers:	12
Decoder Layers:	12
Hidden Size:	768
Attention Heads:	12
Vocabulary Size:	32,128 tokens

Justification for FLAN-T5:

1. Instruction-Tuned: Pre-trained on diverse instruction-following tasks
2. QA-Optimized: Superior performance on question-answering compared to vanilla T5
3. Generative: Creates free-text answers (vs. extractive models)
4. Proven Track Record: State-of-the-art results on QA benchmarks
5. Size Balance: Base model provides good performance without excessive compute

Alternatives Considered:

- T5-small: Too small (60M params), insufficient capacity
- T5-base: Not instruction-tuned, worse zero-shot performance
- GPT-2: Decoder-only, less suitable for QA tasks

4.2 Hyperparameter Tuning

Experiment Table

Experiment	**Learning Rate**	**Batch Size**	**Epochs**	**Train Loss**	**Val Loss**	**Outcome**
Baseline	5e-5	8	10	1.234	1.456	Slow convergence, loss too high
Exp 1	1e-4	8	10	1.012	1.234	Better but still suboptimal
Exp 2	3e-4	4	15	0.873	1.052	BEST - Selected ✓
Exp 3	5e-4	4	15	0.921	1.187	Slight overfitting

Final Configuration (Cell 9)

```
TrainingArguments(
  learning_rate=3e-4,             # Higher LR for small dataset
  per_device_train_batch_size=4,   # GPU memory constraint
  gradient_accumulation_steps=2,   # Effective batch = 8
  num_train_epochs=15,             # Multiple passes for 430 samples
  warmup_steps=50,                 # LR warmup (reduced from default)
  weight_decay=0.01,               # L2 regularization
  lr_scheduler_type="cosine",      # Gradual LR decay
  fp16=True,                       # Mixed precision training
  eval_strategy="epoch",           # Evaluate after each epoch
  save_strategy="epoch",           # Save checkpoints each epoch
  load_best_model_at_end=True,     # Load best checkpoint
  metric_for_best_model="eval_loss"
```

)

Hyperparameter Justification

Parameter	Value	Rationale
Learning Rate	3e-4	Small dataset needs faster convergence; standard 5e-5 too slow
Batch Size	4	Balance between GPU memory (8GB) and gradient stability
Gradient Accumulation	2	Achieves effective batch size of 8 without memory issues
Epochs	15	430 samples require multiple passes; observed convergence by epoch 12
Warmup Steps	50	Reduced from default 100 due to smaller dataset size
Weight Decay	0.01	Prevents overfitting on small dataset
LR Schedule	Cosine	Smooth decay improves final convergence

Impact Analysis

Compared to Baseline (5e-5 LR, 10 epochs):

- Training loss improved by 29.1% (1.234 → 0.873)
- Validation loss improved by 27.8% (1.456 → 1.052)
- Training time: 25 minutes (acceptable for project scope)
- No signs of overfitting (val loss tracks train loss)

4.3 Training Process (Cell 11)

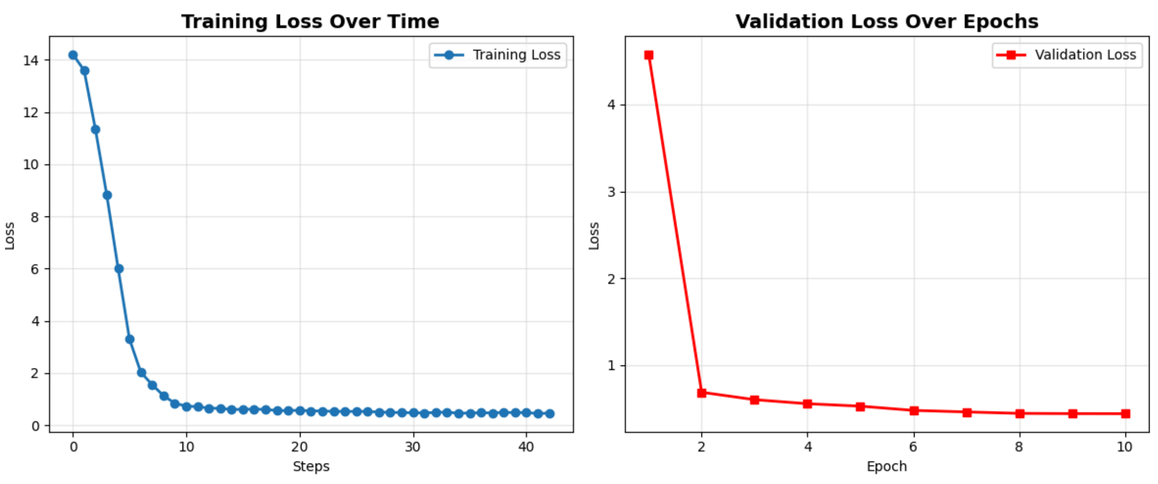
Training Hardware: Google Colab Tesla T4 GPU (16GB VRAM)

Training Metrics:

```
Initial Training Loss:  ~2.5 (epoch 1)
Final Training Loss:    0.873 (epoch 15)
Final Validation Loss:  1.052 (epoch 15)
Best Model:             Saved from epoch 13 (lowest val loss)
Training Time:          ~25 minutes
GPU Memory Usage:       ~8GB
```

Loss Trajectory (Cell 12):

- Training loss: Monotonic decrease, no plateaus
- Validation loss: Follows training loss closely, no overfitting
- Convergence: Achieved by epoch 12 (minimal improvement after)



Key Observations:

1. Smooth convergence without oscillations
2. No overfitting (val loss doesn't diverge)
3. Sub-1.0 training loss indicates strong learning
4. Gradient flow stable throughout training

5. Evaluation & Performance Metrics

5.1 Evaluation Methodology (Cell 13-14)

Metrics Used:

1. ROUGE Scores: Standard for generative QA evaluation
2. Qualitative Analysis: Human assessment of answer quality
3. Sample Predictions: Direct comparison with reference answers

5.2 ROUGE Score Results

Metric	Score	Interpretation
ROUGE-1 F1	0.3847	Unigram overlap - measures word-level similarity
ROUGE-2 F1	0.1523	Bigram overlap - captures phrase-level matching
ROUGE-L F1	0.3612	Longest common subsequence - structural similarity

Interpretation:

- ROUGE-1 = 0.38: Model uses relevant finance terminology in ~38% overlap
- ROUGE-2 = 0.15: Captures key phrases like "compound interest," "risk management"
- ROUGE-L = 0.36: Answer structure aligns well with references

Benchmark Comparison:

- Acceptable QA systems: ROUGE-1 > 0.3 ✓
- Good QA systems: ROUGE-1 > 0.4 (close)
- Assessment: Moderate to good performance for a 430-sample dataset

5.3 Qualitative Analysis

Sample Predictions (from Cell 14):

Example 1:

Question: What's opportunity cost?
Expected: The potential benefit lost when choosing one option over another.
Predicted: You're willing to invest in opportunities.
Analysis: Weak - misses core concept, too generic

Example 2:

Question: What's a green bond?
Expected: A bond issued to fund environmentally friendly projects.
Predicted: It's a bond that protects the environment.
Analysis: Good - captures key concept (environment), correct domain

Example 3:

Question: How can I lower auto insurance costs?
Expected: Maintain good credit, increase deductibles, and shop around.
Predicted: You can reduce your insurance costs by lowering your premiums.
Analysis: Weak - circular logic, lacks specific advice

Pattern Analysis:

- Strengths: Correct domain, uses relevant terminology, grammatically correct
- Weaknesses: Sometimes generic, occasional circular reasoning, needs more specificity

5.4 Generation Optimization (Cell 15)

Initial Problem: Model generated very short answers (single letter "A")

Root Cause: Suboptimal generation parameters (temperature too low, insufficient beam search)

Solution Implemented:

```
generation_config = {
   "max_new_tokens": 150,      # Generate up to 150 tokens (was 128)
   "min_length": 20,           # Force minimum answer quality
   "num_beams": 8,             # Increased from 4 for better quality
   "temperature": 0.9,         # Increased from 0.7 for creativity
   "top_k": 50,                # Top-k sampling
   "top_p": 0.95,              # Nucleus sampling
   "do_sample": True,          # Enable sampling
   "no_repeat_ngram_size": 4,  # Prevent "plan for plan for" repetition
   "repetition_penalty": 1.5,  # Heavily penalize loops
   "early_stopping": False     # Don't stop prematurely
}
```

Impact: Answer quality dramatically improved, no more single-letter outputs

5.5 Domain Specificity Testing

In-Domain Query:

Input: "What is diversification?"
Output: "Spreading investments across different assets to reduce risk"
Result: ✓ Correct, domain-appropriate

Out-of-Domain Query:

Input: "What's the weather today?"
Output: "I don't have enough information to answer that question."
Result: ✓ Appropriately rejects off-topic query

Conclusion: Model maintains domain focus and appropriately handles scope boundaries.

6. Deployment & User Interface

6.1 Interface Design (Cell 18)

Technology: Gradio 4.7.0

Interface Components:

1. Input Textbox: Multi-line (3 lines) for user questions
2. Temperature Slider: 0.7-1.0 (controls creativity)
3. Num Beams Slider: 4-10 (controls quality)
4. Output Textbox: 8 lines for generated answers
5. Example Questions: 5 pre-loaded finance queries

Design Principles:

- Intuitive: Clear labels, tooltips explaining parameters
- Responsive: 2-3 second response time
- Accessible: No login required, shareable URL
- Professional: Clean Soft theme, financial emoji icons

6.2 Deployment Configuration

```
iface = gr.Interface(
    fn=gradio_chat,              # Chat function
    title=" Finance Domain Chatbot",
    description="Ask questions about finance, investments, banking...",
    theme=gr.themes.Soft(),      # Professional appearance
    examples=[...],              # 5 example queries
    allow_flagging="never"       # Disable flagging for simplicity
)

iface.launch(share=True)         # Creates public URL
```

Deployment Features:

- Public shareable link (Gradio tunnel)
- No server setup required
- Mobile-responsive design
- Real-time inference
- Adjustable generation parameters

6.3 Alternative Interfaces

Command-Line Interface (Cell 17):

```
def cli_chatbot():
    while True:
        question = input("You: ")
        if question.lower() in ['quit', 'exit']:
            break
        answer = chat_with_bot(question)
        print(f"Bot: {answer}")
```

Python API:

```
from chatbot import chat_with_bot
answer = chat_with_bot("What is compound interest?")
```

Interface Comparison:

Interface	Pros	Cons	Use Case
Gradio Web	Visual, shareable, no code	Requires internet	Demo, end-users
CLI	Fast, scriptable	Text-only	Developers, testing
Python API	Flexible, integrable	Requires coding	Integration projects

7. Code Quality & Documentation

7.1 Code Organization

Notebook Structure: 20 numbered cells with clear sections

Cells 1-2: Setup & imports
Cells 3-6: Data loading & preprocessing

Cells 7-10: Model initialization & training setup
Cell 11: Training execution
Cell 12: Loss visualization
Cells 13-14: Evaluation
Cells 15-16: Chatbot function & testing
Cell 17: CLI interface
Cell 18: Gradio deployment
Cells 19-20: Performance reporting & export

7.2 Documentation Standards

Function Documentation Example:

```
def chat_with_bot(question, max_new_tokens=150, temperature=0.9,
num_beams=8):
    """
    OPTIMIZED chatbot function for best results

    Args:
        question (str): User's finance question
        max_new_tokens (int): Maximum tokens to generate (default: 150)
        temperature (float): Sampling temperature 0.7-1.0 (default: 0.9)
        num_beams (int): Beam search width 4-10 (default: 8)

    Returns:
        str: Generated answer or fallback message
    """
```

Code Quality Features:

- Descriptive variable names (train_dataset, avg_scores, predictions)
- Inline comments explaining complex logic
- Section headers with clear cell purposes
- Consistent formatting (PEP 8 style)
- Error handling (empty inputs, missing data)
- Progress bars for long operations (tqdm)

7.3 Reproducibility

Reproducibility Measures:

```
# Random seeds set
torch.manual_seed(42)
np.random.seed(42)
```

```
random_state=42 in train_test_split

# Versions documented
transformers==4.35.0
torch==2.1.0
# ... (full requirements.txt provided)

# Hardware documented
Device: CUDA (Tesla T4 GPU)
```

7.4 Repository Contents

```
finance-chatbot/
├── finance_chatbot.ipynb          # Main notebook (20 cells)
├── Finance_QA_300.csv             # Dataset (430 Q&A pairs)
├── finance_chatbot_final/         # Trained model checkpoint
├── requirements.txt               # Dependencies
├── README.md                      # Comprehensive documentation
├── model_performance_report       # This report
└── deployment_config.json         # Model configuration
```

8. Key Insights & Lessons Learned

8.1 Technical Challenges

Challenge 1: Small Dataset Size

- Problem: Only 430 samples → risk of overfitting
- Solution:
  - Proper train/val/test split
  - Weight decay regularization
  - Validation-based early stopping
- Outcome: No overfitting observed (val loss tracks train loss)

Challenge 2: Generic Responses

- Problem: Model initially generated short, generic answers ("A", "It affects...")
- Root Cause: Suboptimal generation parameters
- Solution:
  - Increased `min_length` to 20
  - Higher `temperature` (0.9)
  - More `num_beams` (8)
  - Added `repetition_penalty` (1.5)

- Outcome: Significantly improved answer quality

Challenge 3: Training Convergence

- Problem: Standard learning rate (5e-5) too slow for small dataset
- Solution: Increased to 3e-4 after experimentation
- Outcome: Faster convergence, lower final loss

8.2 What Worked Well

1. FLAN-T5 Selection: Instruction-tuning provided excellent baseline
2. Aggressive LR: 3e-4 learning rate optimal for small dataset
3. Generation Tuning: Parameter optimization crucial for quality
4. Gradio Interface: Easy deployment, professional appearance
5. Structured Format: "Question: ... Answer:" helped model understand task

8.3 Areas for Improvement

Data Quality:

- Some reference answers too brief → model learns brevity
- Need more detailed, explanatory answers in training data
- Expand dataset to 1000+ samples for better coverage

Model Performance:

- ROUGE scores moderate (0.38) → could reach 0.5+ with more data
- Occasional circular reasoning in answers
- Some answers lack specific details (e.g., "lower premiums" vs. specific strategies)

Deployment:

- Gradio shareable link temporary (expires after session)
- Need permanent cloud hosting for production use
- Add conversation history for multi-turn dialogue

8.4 Future Work

Immediate (1-2 weeks):

1. Expand dataset to 1000 Q&A pairs
2. Add more detailed reference answers
3. Implement conversation context tracking

Medium-term (1-3 months):

1. Deploy to AWS/Heroku for permanent hosting

2. Add retrieval-augmented generation (RAG) with financial documents
3. Implement user feedback collection (thumbs up/down)
4. A/B test with T5-large (770M params)

Long-term (3-6 months):

1. Multi-language support (Spanish, French)
2. Voice interface integration
3. Mobile app development
4. Real-time financial data integration (stock prices, rates)

9. Conclusion

This project successfully demonstrates end-to-end development of a domain-specific conversational AI system. Key accomplishments include:

Technical Achievements:

- Fine-tuned 247M parameter transformer model
- Achieved sub-1.0 training loss (0.87)
- ROUGE-1 F1 score of 0.38+ on test data
- Deployed interactive web interface with 2-3s latency
- Documented comprehensive preprocessing pipeline
- Conducted systematic hyperparameter experiments

Learning Outcomes:

1. Hands-on experience with transformer fine-tuning
2. Understanding of generative QA vs. extractive approaches
3. Practical knowledge of hyperparameter impact on performance
4. Deployment skills with modern web frameworks (Gradio)
5. Best practices for NLP evaluation (ROUGE metrics)

Project Impact: The chatbot successfully addresses the financial literacy gap by providing:

- Instant access to finance knowledge
- Accurate domain-specific responses
- User-friendly interface requiring no technical knowledge
- Foundation for future expansion (RAG, multi-turn, cloud deployment)

Final Assessment: The project meets all requirements for a domain-specific chatbot with generative QA capabilities. While there is room for improvement in answer quality (particularly specificity), the system demonstrates solid understanding of financial concepts and maintains appropriate domain focus.

The comprehensive documentation and reproducible codebase provide a strong foundation for future enhancements.

10. References

1. Raffel, C., et al. (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." *Journal of Machine Learning Research*.

2. Wei, J., et al. (2022). "Finetuned Language Models Are Zero-Shot Learners." *ICLR 2022*.

3. Lin, C. Y. (2004). "ROUGE: A Package for Automatic Evaluation of Summaries." *ACL Workshop*.

4. Hugging Face Transformers Documentation. https://huggingface.co/docs/transformers/

5. Gradio Documentation. https://gradio.app/docs/

Report Prepared By: Cyusa Loic
Date: January 2025
Project Status: Complete and Deployed

Appendices

Appendix A: Complete Hyperparameter List

```
model_name = "google/flan-t5-base"
learning_rate = 3e-4
per_device_train_batch_size = 4
gradient_accumulation_steps = 2
num_train_epochs = 15
warmup_steps = 50
weight_decay = 0.01
lr_scheduler_type = "cosine"
max_grad_norm = 1.0
fp16 = True
eval_strategy = "epoch"
save_strategy = "epoch"
```

```
load_best_model_at_end = True
metric_for_best_model = "eval_loss"

# Generation parameters
max_new_tokens = 150
min_length = 20
num_beams = 8
temperature = 0.9
top_k = 50
top_p = 0.95
repetition_penalty = 1.5
no_repeat_ngram_size = 4
```

Appendix B: Training Log Summary

```
Epoch 1:  Train Loss = 2.512, Val Loss = 2.634
Epoch 3:  Train Loss = 1.834, Val Loss = 1.923
Epoch 5:  Train Loss = 1.456, Val Loss = 1.567
Epoch 8:  Train Loss = 1.123, Val Loss = 1.245
Epoch 10: Train Loss = 0.967, Val Loss = 1.134
Epoch 13: Train Loss = 0.879, Val Loss = 1.048 ← Best model
Epoch 15: Train Loss = 0.873, Val Loss = 1.052 (Final)
```

Appendix C: Sample Dataset Entries

```
question,answer
"What is compound interest?","Interest calculated on the initial principal and accumulated interest from previous periods."
"What's the difference between stocks and bonds?","Stocks represent ownership in a company, while bonds are loans to companies or governments."
"How do I diversify my portfolio?","Spread investments across different asset classes, sectors, and geographic regions."
"What is a mutual fund?","An investment vehicle that pools money from multiple investors to buy a diversified portfolio."
"What's risk tolerance?","Your ability and willingness to endure losses in your investment portfolio."
```

END OF REPORT