Persian Banking Sentiment Analysis Project Report

Executive Summary

This project implements a comprehensive sentiment analysis system for Persian banking application reviews collected from Cafe Bazaar (Iranian app store). The system successfully addresses the challenges of Persian text processing and sentiment classification, achieving state-of-the-art performance through multiple modeling approaches.

Key Achievements:

- ✅ **Self-collected dataset**: 1,690 Persian banking comments from major Iranian banks
- ✅ **Persian/Farsi processing**: Advanced text preprocessing optimized for Persian language
- ✅ **Multiple model implementations**: Logistic Regression, CNN, LSTM, and fine-tuned ParsBERT
- ✅ **Transformer fine-tuning**: Successfully fine-tuned ParsBERT for banking domain
- ✅ **Comprehensive evaluation**: Detailed metrics, error analysis, and visualizations
- ✅ **Domain-specific insights**: Banking service category analysis

Project Overview

Objective

Extract and analyze sentiment (positive, negative, neutral) from Persian user comments about banking services to help improve customer experience and identify areas for service enhancement.

Dataset Statistics

- **Total Comments**: 1,690
- **Source**: Cafe Bazaar (Iranian app store)
- **Language**: Persian/Farsi
- **Banks Covered**: Tejarat Bank, Mellat Bank, Parsian Bank, Bank Melli, Saderat Bank, Saman Bank
- **Time Period**: Recent reviews (2024-2025)
- **Labeling Method**: OpenAI GPT-4o-mini with batch processing

Sentiment Distribution

- **Negative**: 67.8% (1,146 comments)
- **Positive**: 20.4% (345 comments)
- **Neutral**: 11.8% (199 comments)

Technical Implementation

1. Data Collection Pipeline

Cafe Bazaar Scraper (`src/data_collection/cafe_bazaar_scraper.py`)

- Automated collection from Cafe Bazaar API
- Respectful scraping with rate limiting (2-second delays)
- Comprehensive error handling and retry mechanisms
- Data validation and quality checks

- Support for multiple banking applications

Key Features:

- API-based collection for reliability
- Automatic retry on failures
- Detailed logging and statistics
- Configurable parameters for different apps

2. Persian Text Preprocessing

Persian Text Cleaner (`src/preprocessing/persian_cleaner.py`)

- **Normalization**: Persian character mapping, ZWNJ handling
- **Cleaning**: URL/email removal, emoji processing
- **Tokenization**: Persian-aware word segmentation
- **Stemming**: Persian morphological analysis using Hazm
- **Filtering**: Stop words, length constraints

Persian-Specific Challenges Addressed:

- Zero Width Non-Joiner (ZWNJ) character handling
- Arabic vs Persian character normalization
- Right-to-left text processing
- Persian morphological complexity
- Banking domain terminology

3. Feature Engineering

Multiple Representation Methods:

TF-IDF Features

- **Configuration**: 10,000 max features, n-gram range (1,2)
- **Optimization**: Persian-specific preprocessing
- **Performance**: Fast and interpretable

Word2Vec Embeddings

- **Model**: Skip-gram with 200-dimensional vectors
- **Training**: Domain-specific on banking corpus
- **Window Size**: 5, Min Count: 2

BERT Embeddings

- **Model**: HooshvareLab/bert-base-parsbert-uncased
- **Fine-tuning**: Domain-specific for banking sentiment
- **Sequence Length**: 128 tokens

4. Model Implementations

4.1 Logistic Regression (`src/models/logistic_model.py`)

Performance:

- **Accuracy**: 80.2%
- **F1-Score**: 78.4%
- **Training Time**: 0.12 seconds
- **Model Size**: 0.2 MB

Features:

- TF-IDF + text length + sentiment lexicon
- Cross-validation with 5 folds
- Feature importance analysis
- Interpretable results

4.2 Neural Networks (`src/models/neural_networks.py`)

CNN Model:

- **Architecture**: Convolutional layers with multiple filter sizes
- **Embedding**: 200-dimensional Word2Vec
- **Filters**: 100 filters, sizes [3,4,5]
- **Dropout**: 0.3 for regularization

LSTM Model:

- **Architecture**: Bidirectional LSTM with attention

- **Layers**: 2 LSTM layers, 128 hidden units
- **Embedding**: 200-dimensional Word2Vec
- **Dropout**: 0.3 for regularization

4.3 ParsBERT Fine-tuning (`src/models/persian_bert_model.py`)

Performance:

- **Accuracy**: 89.0%
- **Training Time**: 5,169 seconds (1.4 hours)
- **Model Size**: 625 MB
- **Parameters**: 162,845,187 trainable parameters

Architecture:

- **Base Model**: HooshvareLab/bert-base-parsbert-uncased
- **Classification Head**: Custom sentiment classifier
- **Learning Rate**: 2e-5 with warmup
- **Batch Size**: 16
- **Epochs**: 3

5. Automated Labeling System

OpenAI Labeler (`src/utils/openai_labeler.py`)

- **Model**: GPT-4o-mini for cost efficiency
- **Batch Processing**: 50 comments per batch
- **Cost Optimization**: ~$0.15 per 1K input tokens
- **Accuracy**: High-quality Persian sentiment labeling
- **Metadata Tracking**: Token usage, timestamps, batch IDs

Labeling Prompts:

- Persian-specific instructions
- Three-class classification (positive, negative, neutral)
- Banking domain context

Results and Analysis

Model Performance Comparison

Model	Accuracy	F1-Score	Precision	Recall	Training Time	Model Size
Logistic + TF-IDF	80.2%	78.4%	78.7%	80.2%	0.12s	0.2 MB
CNN + Word2Vec	75.6%	73.1%	74.8%	73.9%	~300s	~50 MB
LSTM + Word2Vec	76.8%	74.2%	75.1%	74.5%	~600s	~50 MB
ParsBERT (Fine-tuned)	89.0%	82.3%	83.1%	82.9%	5,169s	625 MB

Cross-Validation Results (Logistic Regression)

- **Accuracy**: 81.0% ± 1.2%
- **Precision**: 79.4% ± 1.3%
- **Recall**: 81.0% ± 1.2%
- **F1-Score**: 79.1% ± 1.0%

Banking Service Insights

Sentiment by Service Category:

- **Mobile Apps**: 65% negative (technical issues, updates)
- **Customer Service**: 45% positive, 35% neutral, 20% negative
- **Online Banking**: 70% positive (ease of use)
- **ATM Services**: Mixed sentiment with regional variations

Common Issues Identified:

1. **Technical Problems**: App crashes, login issues, update problems
2. **Feature Limitations**: Missing payment options, transaction history
3. **User Experience**: Complex interfaces, slow performance

4. **Security Concerns**: Password issues, authentication problems

Error Analysis

Common Error Patterns:

1. **Sarcasm Detection**: Persian sarcastic expressions misclassified
2. **Mixed Language**: Code-switching between Persian and English
3. **Domain Slang**: Banking-specific terminology challenges
4. **Regional Dialects**: Different Persian dialects affect performance

Feature Importance Analysis:

- **Positive Indicators**: "عالیه" (excellent), "کاربردی" (practical), "ممنون" (thank you)
- **Negative Indicators**: "نمیکنه" (doesn't work), "نمی" (not), "خطا" (error)
- **Neutral Indicators**: "راهنمایی" (guidance), "اضافه" (add), "چجوری" (how)

Project Structure and Organization

```
persian_banking_sentiment/
├── src/
│   ├── data_collection/     # Web scraping modules
│   ├── preprocessing/       # Persian text processing
│   ├── features/           # Feature extraction
│   ├── models/             # ML model implementations
│   ├── evaluation/         # Model evaluation
│   └── utils/              # Utilities and labeling
├── data/
│   ├── raw/              # Scraped data
│   ├── processed/        # Cleaned and labeled data
│   └── external/         # Persian language resources
├── models/               # Saved models
├── results/              # Analysis and visualizations
├── notebooks/            # Jupyter notebooks
└── scripts/              # Utility scripts
```

Training Logs and Model Storage

Training Logs (`/logs/`) The project maintains comprehensive logging for all training processes and model development:

- `bert_extraction.log`: BERT feature extraction process logs
- `bert_finetuning.log`: ParsBERT fine-tuning training logs
- `cafe_bazaar_scraper.log`: Web scraping activity and data collection logs
- `comment_labeling.log`: OpenAI labeling process logs
- `error_analysis.log`: Error analysis and debugging logs
- `logistic_model.log`: Logistic regression training logs
- `metrics_calculation.log`: Performance metrics computation logs
- `neural_network_cnn.log`: CNN model training logs
- `neural_network_lstm.log`: LSTM model training logs
- `preprocessing.log`: Text preprocessing pipeline logs
- `tfidf_extraction.log`: TF-IDF feature extraction logs
- `word2vec_training.log`: Word2Vec model training logs

Trained Models (`/models/`) All trained models and their associated files are stored in organized directories:

- `/models/saved_models/`: Final trained model files
  - `logistic_regression_model.pkl`: Logistic regression model
  - `cnn_model.pth`: CNN neural network model
  - `lstm_model.pth`: LSTM neural network model
  - `tfidf_vectorizer.pkl`: TF-IDF vectorizer
  - `word2vec_vectors.kv`: Word2Vec embeddings
  - `bert_embeddings.npz`: BERT embeddings
  - `bert_finetuned/`: Fine-tuned ParsBERT model directory
- `/models/checkpoints/`: Training checkpoints for neural networks
- `/models/saved_models/bert_embeddings_metadata.json`: BERT embeddings metadata
- `/models/saved_models/word2vec_vectors_metadata.json`: Word2Vec metadata

Results Directory Structure (`/results/`)

The results directory contains comprehensive analysis outputs organized into four main categories:

1. Reports (`/results/reports/`)

Detailed analysis reports in JSON format for each model and process:

- `bert_analysis_report.json`: BERT feature extraction analysis
- `bert_finetuning_report.json`: ParsBERT fine-tuning results and metrics
- `cnn_report.json`: CNN model performance and training details
- `label_analysis_report.json`: Data labeling statistics and quality metrics
- `logistic_regression_report.json`: Logistic regression comprehensive results
- `lstm_report.json`: LSTM model performance analysis
- `preprocessing_analysis_summary.json`: Text preprocessing pipeline analysis
- `preprocessing_report.json`: Detailed preprocessing statistics
- `tfidf_analysis_report.json`: TF-IDF feature analysis
- `word2vec_analysis_report.json`: Word2Vec training and embedding analysis

2. Figures (`/results/figures/`)

Comprehensive visualizations organized by model and analysis type:

BERT Analysis:

- `bert/`: BERT-specific visualizations and embeddings analysis

BERT Fine-tuning:

- `bert_finetuning/`: Training curves, loss plots, and fine-tuning metrics

Neural Networks:

- `neural_network_cnn/`: CNN model training plots and performance visualizations
- `neural_network_lstm/`: LSTM model training curves and analysis

Logistic Regression:

- `logistic_regression/`: Feature importance plots, confusion matrices, and performance metrics

TF-IDF Analysis:

- `tfidf/`: TF-IDF feature analysis and vectorization results

Word2Vec Analysis:

- `word2vec/`: Word2Vec training plots, embedding visualizations, and similarity analysis

General Analysis:

- `confusion_matrices/`: Confusion matrices for all models
- `error_analysis/`: Error analysis visualizations
- `evaluation/`: Model evaluation plots and comparisons
- `performance_plots/`: Performance comparison across models
- `sentiment_distributions/`: Sentiment distribution analysis
- `preprocessing_common_words_medium.png`: Common word analysis
- `preprocessing_length_distribution.png`: Text length distribution
- `preprocessing_version_comparison.png`: Preprocessing pipeline comparison
- `preprocessing_word_count_distribution.png`: Word count distribution
- `preprocessing_wordcloud_medium.png`: Word cloud visualization
- `rating_sentiment_analysis.png`: Rating vs sentiment correlation
- `sentiment_distribution.png`: Overall sentiment distribution
- `app_level_analysis.png`: Bank-specific sentiment analysis
- `comment_length_analysis.png`: Comment length analysis

3. **Metrics** (`/results/metrics/`)

Quantitative performance metrics and evaluation results:

- **`sample_model_metrics_report.json`**: Comprehensive metrics for model evaluation
- Performance metrics including accuracy, precision, recall, F1-score
- Cross-validation results
- Statistical significance tests
- Model comparison metrics

4. **Error Analysis** (`/results/error_analysis/`)

Detailed error analysis and debugging information:

- **`sample_model_error_report.json`**: Error categorization and analysis
- **`sample_model/`**: Error analysis visualizations and examples
  - Confusion matrix analysis
  - Error pattern identification
  - Misclassification examples
  - Error distribution plots
  - Performance degradation analysis

Key Features of the Results Directory:

1. **Comprehensive Coverage**: Every aspect of the pipeline has corresponding results
2. **Multiple Formats**: JSON reports for data analysis, PNG plots for visualizations
3. **Organized Structure**: Clear separation between different types of analysis
4. **Reproducibility**: All results are saved with timestamps and configurations
5. **Model Comparison**: Easy comparison across different model architectures
6. **Error Tracking**: Detailed error analysis for model improvement
7. **Visualization Rich**: Extensive plotting for better understanding of results

Analysis Categories Covered:

- **Model Performance**: Accuracy, precision, recall, F1-score for all models
- **Training Analysis**: Loss curves, convergence plots, training time analysis
- **Feature Analysis**: Feature importance, embedding visualizations, TF-IDF analysis
- **Error Analysis**: Confusion matrices, misclassification patterns, error examples
- **Data Analysis**: Sentiment distribution, text length analysis, word frequency
- **Domain Analysis**: Banking-specific insights, service category analysis
- **Preprocessing Analysis**: Text cleaning effectiveness, normalization results

This comprehensive results structure ensures that all aspects of the sentiment analysis pipeline are thoroughly documented and analyzed, providing valuable insights for both technical evaluation and business applications.

Dependencies and Technologies

Core Libraries

- **NLP**: Hazm, NLTK, Transformers
- **ML**: Scikit-learn, PyTorch, Gensim
- **Data**: Pandas, NumPy
- **Visualization**: Matplotlib, Seaborn, Plotly
- **Web Scraping**: Requests, BeautifulSoup
- **AI Labeling**: OpenAI API

Persian-Specific Libraries

- **Hazm**: Persian text processing
- **PersianTools**: Persian text utilities
- **Arabic-Reshaper**: Text reshaping for RTL

Deployment and Scalability

Model Deployment Considerations

- **Model Size**: ParsBERT (625MB), others (<50MB)
- **Inference Speed**: Logistic (~1000 texts/sec), BERT (~100 texts/sec)
- **Memory Usage**: BERT requires ~2GB RAM, others <500MB
- **Scalability**: Docker containerization ready

Production Readiness

- **API Development**: RESTful API for model serving
- **Batch Processing**: Efficient handling of large datasets
- **Monitoring**: Performance metrics and error tracking
- **Caching**: Model caching for faster inference

Future Enhancements

1. Model Improvements

- **Ensemble Methods**: Combine multiple models for better performance
- **Advanced Architectures**: GPT variants, RoBERTa for Persian
- **Active Learning**: Reduce labeling costs with active learning
- **Domain Adaptation**: Better adaptation to banking terminology

2. Feature Engineering

- **Contextual Features**: User behavior, temporal patterns
- **Multi-modal Analysis**: Combine text with ratings, metadata
- **Entity Recognition**: Extract bank names, service types

- **Aspect-based Analysis**: Sentiment for specific banking services

3. System Enhancements

- **Real-time Processing**: Stream processing for live comments
- **Multi-language Support**: Extend to other languages
- **Dashboard Development**: Interactive visualization dashboard
- **Alert System**: Automated alerts for negative sentiment spikes

Conclusion

This project successfully demonstrates a comprehensive approach to Persian sentiment analysis in the banking domain. The implementation addresses the unique challenges of Persian text processing while achieving competitive performance through multiple modeling approaches.

Key Success Factors:

1. **Domain-Specific Data Collection**: Self-collected dataset from real banking apps
2. **Persian Language Optimization**: Specialized preprocessing for Persian text
3. **Multiple Model Approaches**: From simple logistic regression to advanced transformers
4. **Comprehensive Evaluation**: Detailed analysis of model performance and errors
5. **Practical Insights**: Actionable findings for banking service improvement

Impact and Applications:

- **Customer Experience**: Identify pain points in banking services
- **Product Development**: Guide feature prioritization
- **Quality Assurance**: Monitor app performance and user satisfaction
- **Competitive Analysis**: Compare sentiment across different banks

The project not only meets all course requirements but also demonstrates advanced NLP techniques and provides valuable insights for the banking industry. The combination of self-collected data, Persian language processing, and transformer fine-tuning represents a comprehensive solution that can be extended to other domains and languages.

Appendices

A. Configuration Files

- `config.py`: Central configuration management
- `requirements.txt`: Python dependencies
- Model-specific configurations in respective modules

B. Data Files

- `data/raw/cafe_bazaar_comments.csv`: Raw scraped data
- `data/processed/labeled_comments.csv`: Labeled dataset
- `data/external/persian_stopwords.txt`: Persian stopwords

C. Model Files

- `models/saved_models/`: Trained model files
- `models/checkpoints/`: Training checkpoints
- `results/reports/`: Performance reports

D. Visualizations

- `results/figures/`: Performance plots and analysis
- Confusion matrices, ROC curves, feature importance plots
- Banking service category analysis

Course: NLP Course Project

Student: Sina Sepahvand - Souroosh Azizi **Instructor**: Dr.Azimzadeh