**Newsvendor LLM Experiment Codebase Organization**

**🧠 Core Experiment Engine**

**Purpose**: The heart of the negotiation system that orchestrates LLM negotiations

* **src/core/negotiation\_engine.py** - Main engine for conducting negotiations between buyer/supplier agents
* **src/core/model\_manager.py** - Optimized LLM model loading, unloading, and generation with memory management
* **src/core/conversation\_tracker.py** - Tracks negotiation rounds, speaker alternation, and termination conditions

**🤖 Negotiation Agents**

**Purpose**: LLM agents that represent buyers and suppliers in negotiations

* **src/agents/buyer\_agent.py** - Retailer agent that wants the LOWEST possible wholesale price
* **src/agents/supplier\_agent.py** - Supplier agent that wants the HIGHEST possible price above costs
* **src/agents/reflection\_mixin.py** - Self-reflection capabilities for strategic thinking and self-monitoring

**🔍 Text Processing & Parsing**

**Purpose**: Extract prices and detect agreements from LLM responses

* **src/parsing/price\_extractor.py** - Robust price extraction with multiple strategies and fallback logic
* **src/parsing/acceptance\_detector.py** - Detects explicit acceptance statements and implicit convergence
* **src/parsing/validation.py** - Output validation components

**🧪 Experiment Runners**

**Purpose**: Execute the complete experimental protocol across model pairings

* **src/experiments/run\_full\_experiment.py** - Complete three-phase experiment (validation → statistical power → full dataset)
* **src/experiments/run\_validation\_suite.py** - Validates setup, model availability, and basic functionality
* **src/experiments/run\_single\_negotiation.py** - Individual negotiation test runner
* **src/experiments/analyze\_results.py** - Post-experiment analysis

**📊 Analysis & Statistics**

**Purpose**: Calculate metrics, run statistical tests, and generate insights

* **src/analysis/metrics\_calculator.py** - Convergence rates, price optimality, efficiency, and reflection benefits
* **src/analysis/statistical\_tests.py** - ANOVA, t-tests, chi-squared tests, and effect size calculations
* **src/analysis/complete\_analysis\_runner.py** - Orchestrates comprehensive analysis pipeline
* **src/analysis/safe\_analysis\_runner.py** - Safe analysis runner for data structure validation
* **src/analysis/conversation\_analyzer.py** - Turn-by-turn behavioral analysis from conversation transcripts

**📈 Visualizations**

**Purpose**: Create comprehensive charts, dashboards, and publication-quality figures

**Core Visualization System**

* **src/analysis/visualizations.py** - Main visualization suite with auto-detection and comprehensive dashboards

**Specialized Visualizations**

* **src/analysis/extra\_visualizations/3d\_reflection\_advantage.py** - 3D "sweet spot" analysis for reflection benefits
* **src/analysis/extra\_visualizations/conversation\_flow.py** - Sankey diagrams showing negotiation paths
* **src/analysis/extra\_visualizations/llm\_personality\_fingerprint.py** - Unique "negotiation DNA" radar charts
* **src/analysis/extra\_visualizations/tournament.py** - March Madness style tournament bracket visualization

**⚙️ Configuration & Utilities**

**Purpose**: System configuration, data management, and helper functions

**Configuration Management**

* **src/utils/config\_loader.py** - YAML configuration loading with intelligent defaults
* **src/utils/data\_exporter.py** - Multi-format data export (CSV, JSON, Parquet) with compression

**Data & Debugging**

* **src/utils/file\_finder.py** - Auto-detection of latest experiment data files
* **src/utils/debug\_data.py** - Debug data issues and create sample datasets
* **src/utils/error\_recovery.py** - Failure handling utilities
* **src/utils/logging\_config.py** - Structured logging configuration
* **src/utils/reproducibility.py** - Seeding and determinism utilities

**🧪 Testing Framework**

**Purpose**: Ensure code quality and system reliability

**Unit Tests**

* **tests/unit/test\_price\_extractor.py** - Price extraction validation
* **tests/unit/test\_acceptance\_detector.py** - Agreement detection testing
* **tests/unit/test\_conversation\_tracker.py** - Conversation state management
* **tests/unit/test\_model\_manager.py** - Model loading and generation testing
* **tests/unit/test\_negotiation\_engine.py** - Core engine functionality

**Integration Tests**

* **tests/integration/test\_full\_negotiation.py** - End-to-end negotiation testing
* **tests/integration/test\_experiment\_pipeline.py** - Complete experimental pipeline validation
* **test\_auto\_detection.py** - Auto-detection functionality verification

**📋 Documentation & Specification**

**Purpose**: Experimental design, parameters, and implementation details

* **2LLMAgentNewsvendor v0.5.docx** - Complete experimental specification with:
  + 8 model selection (637MB → 5.2GB range)
  + Adaptive replication strategy (50-40-30-20 reps by model tier)
  + Corrected demand distribution: Normal(μ=40, σ=10)
  + Statistical power analysis
  + Progressive execution strategy
  + Quality assurance protocols

**🎯 Key Experiment Parameters**

Based on the specification document:

* **Problem**: Classical newsvendor with optimal price = $65
* **Models**: 8 LLMs from ultra-compact (637MB) to large (5.2GB)
* **Reflection Patterns**: 4 conditions (00, 01, 10, 11)
* **Total Negotiations**: ~1,940 with adaptive replication
* **Game Setup**: Selling price $100, production cost $30, demand Normal(40,10)
* **Success Metrics**: >95% completion, >85% convergence, ≤$8 price optimality

**🔄 Typical Workflow**

1. **Setup**: run\_validation\_suite.py validates models and configuration
2. **Execute**: run\_full\_experiment.py runs the three-phase experiment
3. **Analyze**: complete\_analysis\_runner.py calculates comprehensive metrics
4. **Visualize**: visualizations.py creates dashboards and publication figures
5. **Debug**: Various debugging tools help troubleshoot issues throughout