Self-Assessment Report

Multi-Agent Turing Machine Translation System
LLMs and Multi-Agent Systems Course

Group Code:	OmerAndYogever
Students:	Omer Caplan, ID: 208753665 Yogev Cuperman, ID: 207540550
Project Name:	Multi-Agent Turing Machine Translation System
Repository:	https://github.com/OmerCaplan/turing-translation
Assessment Date:	November 26, 2025
Self-Grade:	85/100

Category-by-Category Assessment

Category	Weight	My Score	Final Score
Project Documentation	20%	17/20	17
README & Code Documentation	15%	14/15	14
Project Structure & Code Quality	15%	13/15	13
Configuration & Security	10%	10/10	10
Testing & QA	15%	12/15	12
Research & Analysis	15%	12/15	12
UI/UX & Extensibility	10%	7/10	7
TOTAL	**100%**		**85**

Grade Level: 80-89 (■■■■ - Very Good)

With a self-grade of 85, we expect **thorough and meticulous** scrutiny. The evaluators will check full compliance with all criteria and expect high quality standards throughout. This grade reflects excellent work at a high academic level with comprehensive documentation, real research with sensitivity analysis, and quality visualizations.

Justification for Self-Assessment

Strengths

The project demonstrates strong technical implementation and comprehensive documentation across multiple dimensions. The formal PRD provides clear project requirements, success metrics (KPIs), detailed functional and non-functional requirements, system architecture diagrams (C4-style), sequence diagrams, and API documentation. The comprehensive README offers detailed step-by-step installation instructions, troubleshooting guidance, technical details with code examples, and screenshots of results including the semantic drift graph.

Code quality is excellent with proper documentation through docstrings in all functions, clear module separation (experiment.py, claude_wrapper.py, prompts.py, sentences.py), and consistent naming conventions. The modular structure follows best practices with src/, tests/, results/, and assets/ directories, with all files under 150 lines. Security and configuration management are exemplary with proper use of .env files, no hardcoded credentials, and comprehensive .gitignore coverage.

The testing suite is comprehensive, covering 41 different test scenarios with a 100% pass rate. Tests cover sentences validation, prompts verification, typo injection functionality, semantic distance calculation, pipeline execution, graph generation, and edge cases like unicode handling and very long text.

The research component is strong with systematic experiments across 5 error rates (0%, 15%, 25%, 35%, 50%) and 10 test sentences meeting the 15-word minimum requirement. The results graph with confidence intervals demonstrates clear linear correlation between typo rate and semantic drift, providing quantitative insights into translation robustness.

The documentation of the AI interaction process adds educational value and transparency to the development methodology, showing the iterative development approach and key design decisions.

Weaknesses

The UI/UX component is the weakest area as this is a CLI-based application without a graphical user interface. While the command-line interface is clear and provides good progress indicators, there is no GUI or web interface for easier user interaction. Extension points and plugin architecture are not explicitly implemented.

The test coverage percentage (48% as reported by pytest-cov) is lower than optimal because the core code paths require actual API calls and ML model loading which are mocked in tests. While all 41 tests pass, achieving higher coverage would require integration tests with actual API credentials.

The research component, while solid, could be enhanced with additional experiments such as comparing different Claude models, testing different language chains, or including statistical significance testing. A Jupyter notebook for interactive analysis would add value.

Investment

Significant effort was invested in creating a complete multi-agent translation system with three specialized AI agents (Pierre, David, Emily), implementing proper Unix-style CLI piping for agent communication, and integrating sentence-transformers for semantic analysis. The work includes comprehensive documentation (PRD with 500+ lines, README with 400+ lines), a robust test suite with 41 test cases across multiple test classes, and proper project structure following Python best practices.

The development process involved debugging correlation issues, ensuring positive correlation between error rates and semantic drift, implementing proper sentence filtering for the 15-word minimum requirement, and creating professional visualizations with confidence intervals. This represents substantial work beyond a minimal implementation and demonstrates commitment to delivering a polished, research-quality application.

Innovation

The project demonstrates several innovative aspects. The multi-agent "Turing Machine" concept for analyzing semantic drift through translation chains is a creative approach to understanding translation quality degradation. The use of vector embeddings (all-MiniLM-L6-v2) combined with cosine similarity for quantifying semantic drift provides rigorous mathematical foundations.

The agent persona design (Pierre for French, David for Hebrew, Emily for English) with detailed prompts demonstrates understanding of prompt engineering best practices. The typo injection system that preserves spaces and handles edge cases shows attention to experimental rigor. The CLI piping architecture demonstrates Unix philosophy principles applied to multi-agent systems.

Learning

The project demonstrates learning in multiple areas: multi-agent system coordination using Claude API, semantic analysis using transformer-based embeddings, experimental methodology with controlled noise injection, and professional software engineering practices including comprehensive testing and documentation.

Key learnings include understanding the importance of positive correlation in semantic distance measurements (higher error rates should lead to greater drift), the challenges of multi-hop translation where meaning degrades through each step, and the practical application of vector embeddings for semantic comparison. The iterative debugging process to achieve 0.955 correlation demonstrates empirical research methodology.

Academic Integrity Declaration

We hereby declare that:

• Our self-assessment is honest and truthful

• We checked our work against all criteria before determining the grade

• We are aware that a high self-grade will lead to more rigorous review

• We accept that the final grade may differ from our self-assessment

• This work is the product of our work (of the team) and we are responsible for all software

Student 1: Omer Caplan (ID: 208753665)

Signature: Omer

Date: November 26, 2025

Student 2: Yogev Cuperman (ID: 207540550)

Signature: Yogev

Date: November 26, 2025