spec

SPEC-PPP-001: Dynamic Proactivity & Vagueness Detection

Status: Research **Created**: 2025-11-16 **Category**: PPP Framework - Proactivity Dimension **Priority**: High (Foundation for R_{Proact} calculation)

Overview

This SPEC researches methods for detecting vague user prompts and classifying question effort to enable dynamic proactivity in PPP framework agents. The goal is to determine when agents should ask clarifying questions (proactive) vs when they should proceed directly (productive).

Core Challenge: Balance productivity (don’t ask unnecessary questions) with quality (ask when truly ambiguous).

Background

PPP Proactivity Formula

From the PPP framework paper (arXiv:2511.02208):

```
R_Proact = {
    +0.05  if no questions asked
    +0.05  if all questions are low-effort
    -0.1 × (medium-effort questions)  if any medium/high-effort
    -0.5 × (high-effort questions)
}
```

Effort Levels: - **Low**: Selection from options, accessible context (e.g., “Which provider: A, B, or C?”) - **Medium**: Some research needed, not blocking (e.g., “What format do you prefer?”) - **High**: Deep investigation or blocking decision (e.g., “Should I investigate distributed caching strategies?”)

Research Gap

Current State: - Agents ask questions inconsistently (model-dependent) - No systematic vagueness detection - No effort classification (all questions treated equally)

Target State: - Detect genuinely vague prompts requiring clarification - Classify questions by user effort (low/medium/high) - Penalize agents for high-effort blocking questions

---

Research Questions

RQ1: Vagueness Detection Methods

Question: What are the state-of-the-art methods for detecting ambiguous/vague natural language prompts?

Sub-questions: 1. **RQ1.1**: How do NLP models detect ambiguity in questions? 2. **RQ1.2**: Can heuristic methods (keyword matching, pattern detection) achieve acceptable accuracy? 3. **RQ1.3**: What is the accuracy of LLM-based vagueness detection? 4. **RQ1.4**: What are common indicators of vague prompts in coding tasks?

Success Criteria: - Identify 3+ vagueness detection methods - Benchmark accuracy: Target >75% for Phase 1 (heuristic), >90% for Phase 3 (LLM-based) - Document trade-offs: accuracy vs latency vs cost

---

RQ2: Question Effort Classification

Question: How can we classify questions into low/medium/high effort categories?

Sub-questions: 1. **RQ2.1**: What linguistic features correlate with user effort? (question length, complexity, domain knowledge required) 2. **RQ2.2**: Can we build a heuristic classifier using regex patterns? 3. **RQ2.3**: What accuracy can LLM-based classification achieve? 4. **RQ2.4**: How do we handle edge cases (rhetorical questions, multi-part questions)?

Success Criteria: - Define clear criteria for each effort level - Achieve >75% accuracy with heuristics (Phase 1) - Achieve >90% accuracy with LLM (Phase 3, optional) - Validate against labeled dataset (create or source)

RQ3: Rust NLP Ecosystem

Question: What Rust libraries exist for NLP tasks relevant to vagueness detection?

Sub-questions: 1. **RQ3.1**: Text processing: tokenization, sentence splitting, POS tagging 2. **RQ3.2**: Pattern matching: regex, keyword extraction 3. **RQ3.3**: Similarity/semantic analysis: embedding libraries 4. **RQ3.4**: ML inference: ONNX, TensorFlow Lite for Rust

Success Criteria: - Identify 5+ relevant Rust crates - Evaluate maturity, performance, ease of use - Recommend libraries for Phase 1 implementation

---

RQ4: Integration with Existing /reasoning

Question: How should vagueness detection integrate with existing `/reasoning` command patterns?

Sub-questions: 1. **RQ4.1**: Should vagueness detection trigger automatic `/reasoning` invocation? 2. **RQ4.2**: How do OpenAI `reasoning_effort` and Anthropic "extended thinking" relate to PPP proactivity? 3. **RQ4.3**: Can we leverage existing reasoning patterns for vagueness handling?

Success Criteria: - Document integration points - Identify synergies with existing features - Propose design that enhances both systems

---

Deliverables

1. Literature Review (findings.md)

Content: - Academic papers on ambiguity detection, question classification - NLP techniques: keyword extraction, dependency parsing, semantic similarity - Rust crate evaluation (text processing, pattern matching) - Existing vagueness detection systems (chatbots, assistants)

Format: Annotated bibliography with relevance ratings

---

2. Comparative Analysis (comparison.md)

Comparison Tables: - Vagueness detection methods (heuristic vs ML vs LLM) - Question effort classifiers (rule-based vs learned) - Rust NLP crates (regex, rust-bert, lingua, etc.) - Performance trade-offs (accuracy vs latency vs cost)

Decision Matrices: Method selection with scoring

---

3. Proof of Concept (evidence/vagueness_detector_poc.rs)

Implementation:

```rust
pub struct VaguenessDetector {
    // Heuristic-based (Phase 1)
}

impl VaguenessDetector {
    pub fn is_vague(&self, prompt: &str) -> bool;
    pub fn vagueness_score(&self, prompt: &str) -> f32;  // 0.0-1.0
}

pub struct QuestionEffortClassifier {
    // Pattern-based (Phase 1)
}

impl QuestionEffortClassifier {
    pub fn classify_effort(&self, question: &str) -> EffortLevel;
}
```

Test Cases: - Vague prompts: “Implement OAuth” → vague (no provider specified) - Specific prompts: “Implement OAuth2 with Google provider using PKCE” → clear - Question types: Selection (low), research (medium), blocking (high)

---

4. Architectural Decision Records (adr/)

ADR-001-001: Heuristic vs LLM-based Vagueness Detection - **Decision**: Start with heuristics (Phase 1), optional LLM upgrade (Phase 3) - **Rationale**: Cost, latency, 75%+ accuracy achievable with patterns

ADR-001-002: Question Effort Classification Strategy - **Decision**: Keyword matching + length heuristics (Phase 1) - **Rationale**: Simple, fast, no external dependencies

ADR-001-003: Integration with /reasoning Command - **Decision**: Separate but complementary systems - **Rationale**: Vagueness ≠ complexity, different triggers

---

5. Implementation Recommendations (recommendations.md)

Phase 1 (Heuristic-Based): - Pattern matching for vague indicators - Keyword lists for effort classification - Integration with trajectory logging (SPEC-PPP-004)

Phase 2 (Enhanced Heuristics): - Dependency parsing (if Rust library available) - Domain-specific patterns (coding task vocabulary)

Phase 3 (Optional LLM): - LLM-based vagueness scoring - Fine-tuned question effort classifier

---

Acceptance Criteria

Research Coverage

- [ ] 10+ academic papers reviewed (ambiguity detection, NLP)
- [ ] 5+ Rust NLP crates evaluated
- [ ] 3+ vagueness detection methods compared
- [ ] 2+ question effort classification approaches benchmarked

Deliverables

- [ ] findings.md: Comprehensive literature review
- [ ] comparison.md: Detailed method comparisons with matrices
- [ ] evidence/vagueness_detector_poc.rs: Working PoC (compiles, tests pass)
- [ ] adr/: 3+ ADRs documenting key decisions
- [ ] recommendations.md: Phased implementation plan

Quality Gates

- ☐ Heuristic vagueness detection: >75% accuracy (on test set)
- ☐ Heuristic effort classification: >75% accuracy (on test set)
- ☐ All code examples compile and run
- ☐ Integration plan with SPEC-PPP-004 (trajectory logging)

Dependencies

Upstream

- PPP Framework paper (arXiv:2511.02208) - Proactivity formula definition
- SPEC-PPP-004 - Trajectory logging (stores questions for R_{Proact} calculation)

Downstream

- SPEC-PPP-003 - Weighted consensus (uses R_{Proact} scores)
- Future: Agent prompt engineering (when to trigger vagueness detection)

Success Metrics

Research Quality

- Papers reviewed: 10+ (target), 5+ (minimum)
- Methods compared: 3+ (heuristic, ML, LLM)
- Rust crates evaluated: 5+

Technical Feasibility

- Heuristic accuracy: >75% (validates Phase 1 approach)
- Latency: <10ms (vagueness detection per prompt)
- Cost: $0 (heuristic-based, no API calls)

Deliverable Completeness

- All 5 deliverables created (findings, comparison, PoC, ADRs, recommendations)
- PoC demonstrates working vagueness detection and effort classification
- ADRs provide clear decision rationale

Open Questions

1. **Ambiguity Definition**: What constitutes “vague” for coding tasks? (lack of specificity, missing context, multiple interpretations?)

2. **Question Extraction**: How to reliably extract questions from agent responses? (regex, dependency parsing, LLM?)

3. **Multi-turn Context**: Should vagueness detection consider conversation history? (e.g., “use the same approach” refers to previous turn)

4. **Domain Adaptation**: Do coding tasks have different vagueness patterns than general NLP? (technical vocabulary, specific requirements?)

5. **Calibration**: How to validate effort classification without user feedback? (create labeled dataset, expert annotation?)

Timeline Estimate

Research: 12 hours (1.5 days) - Literature review: 4 hours - Rust crate evaluation: 3 hours - Method comparison: 2 hours - Integration design: 3 hours

Implementation (PoC + Recommendations): 8 hours (1 day) - Vagueness detector PoC: 4 hours - Effort classifier PoC: 3 hours - Testing + documentation: 1 hour

Documentation (ADRs + Guides): 4 hours (0.5 days) - ADRs: 2 hours (3 ADRs × 40 min each) - Recommendations: 2 hours

Total: ~24 hours (3 days, 1 engineer)

Related Work

- SPEC-PPP-000: Master coordination SPEC
- SPEC-PPP-002: Personalization (preference violations)
- SPEC-PPP-003: Weighted consensus (R_{Proact} integration)
- SPEC-PPP-004: Trajectory logging (stores questions)

References

1. Sun, W., et al. (2025). “Training Proactive and Personalized LLM Agents.” arXiv:2511.02208
2. TrustNLP (2025). “Ambiguity Detection and Uncertainty Calibration for Question Answering”
3. Brosnan, K., et al. (2021). “Cognitive Load Reduction in Questionnaires” (Paas scale for effort)