Six Steps Question Algorithm

How the Question Algorithm Works

Core Principle: The system reflects the user's own words back to them—it never introduces new concepts.

The Flow

1. **Opening Question** (iteration 1)
   - Each space has a tailored first question, e.g.:
     - *Here*: "And what do you notice in this present moment?"
     - *Before*: "And what was life like before?"
   - These are "cleanish"—adapted for self-guided use without a facilitator
2. **Subsequent Questions** (iterations 2-6)
   - The algorithm extracts key words from the user's response
   - These words are inserted into Clean Language templates:
     - Step 2: "And when [**their words**], what do you notice?"
     - Step 3: "And what kind of [**their words**] is that [**their words**]?"
     - Step 4: "And is there anything else about [**their words**]?"
     - Step 5: "And what does [**their words**] know?"
     - Step 6: "And when [**their words**], what do you know now?"
3. **Integration** (after 6 iterations or safety trigger)
   - "And what do you know now that you didn't know before?"
   - This question is deliberately untethered from the reflected words [X]
   - The phrase "that you didn't know before" invites recognition of *emergence*—new knowledge that arose through the iterative process
   - This aligns with Emergent Knowledge (EK) principles: knowledge emerges through iteration, and naming it helps integration

Word Extraction Logic

The `extract_key_concept` method: - Strips punctuation and normalises whitespace - Removes filler words (articles, pronouns, common verbs like "feel", "think", "notice") - Takes the **last 1-2 significant words** from their response - Falls back to "that" if nothing meaningful remains

Example: If user writes *"I'm feeling a heaviness in my chest"*, it extracts **"heaviness chest"** and generates: *"And when heaviness chest, what do you notice?"*

What It Doesn't Do

- No interpretation or analysis of meaning
- No suggestions or advice

- No probing into trauma or relationships
- No AI/LLM generating content—pure pattern matching

Why Not Use an LLM?

Using an LLM like Claude to generate Clean Language questions would undermine the approach in several ways:

1. LLMs introduce content

Even when instructed to "only reflect the user's words," LLMs interpret, summarise, and rephrase. They're trained to be helpful—to add value. Clean Language requires the opposite: adding *nothing*. A model might turn "heaviness in my chest" into "that weight you're carrying" or "the burden you feel"—subtle interpretations that contaminate the client's symbolic landscape.

2. Unpredictable outputs

The same input can produce different questions across runs. Clean Language relies on predictable, formulaic questions ("And what kind of X is that X?"). An LLM might generate creative variations that sound good but violate the structure.

3. The "helpful" instinct

LLMs want to move things forward, offer insight, or gently guide. They might: - Ask "Why do you think you feel that way?" (probing) - Suggest "Perhaps that heaviness is grief?" (interpretation) - Say "That sounds difficult" (evaluation)

All of these break Clean Language principles.

4. Safety unpredictability

With pattern-matching, you know exactly what words trigger safety responses. An LLM might miss crisis signals, or conversely, over-react to benign content.

5. The "clean" is the point

The therapeutic value comes from the *absence* of the facilitator's model of the world. An LLM has a massive implicit model—its entire training corpus. It can't truly be "clean."

The Six Steps approach uses deterministic pattern matching: extract words → slot into templates. No intelligence, no interpretation. That's a feature, not a limitation.

Can the Pattern Matching Be Improved?

Yes, there are several potential improvements worth considering:

Current Limitation: Losing Context

The current algorithm extracts just the last 1-2 significant words, which can lose important context.

Example: - Input: *"I'm feeling a heaviness in my chest"* - Current extraction: **"heaviness chest"** - Result: *"And when heaviness chest, what do you notice?"*

This loses the relational phrase "in my" which matters for Clean Language.

Potential Improvements

1. Preserve prepositional phrases Instead of just nouns, keep phrases like "heaviness in my chest" intact: - Detect patterns like `[noun] + [preposition] + [possessive] + [noun]` - Result: *"And when heaviness in your chest, what do you notice?"*

2. Preserve the whole noun phrase Keep adjectives and determiners that modify the key noun: - "a dark cloud" → keep "dark cloud" not just "cloud" - "this strange feeling" → keep "strange feeling"

3. Mirror exact phrasing for short responses If the user's response is brief (under 10 words), use more of it verbatim rather than extracting fragments.

4. Handle metaphors specially The code already looks for "like a…" patterns. When found, the metaphor could be preserved more carefully: - "It's like a stone in my stomach" - Preserve: "stone in your stomach" for reflection

5. Grammatical tidying Apply minimal grammar fixes when inserting into templates: - Change "my" to "your" (possessive shift) - Ensure verb agreement

Trade-offs to Consider

Improvement	Benefit	Risk
Longer extractions	More context preserved	Questions become unwieldy
Grammar tidying	More natural reading	Introduces facilitator's voice

Improvement	Benefit	Risk
Metaphor preservation	Honours symbolic content	May over-complicate extraction

Recommendation

A middle-ground approach: 1. Keep extractions to 3-5 words maximum (currently 1-2) 2. Preserve prepositional phrases when they follow key nouns 3. Swap "my" → "your" for grammatical flow 4. For very short inputs (<6 words), use the whole response

This would turn: - *"heaviness in my chest"* → *"And when heaviness in your chest, what do you notice?"*

Rather than the current: - *"And when heaviness chest, what do you notice?"*

NLP Algorithms for Future Improvement

Several Natural Language Processing techniques could improve word extraction while preserving the deterministic, non-generative approach required for Clean Language.

Recommended Techniques

1. Part-of-Speech (POS) Tagging

- Identifies nouns, verbs, adjectives, prepositions
- The original Slackbot used EngTagger for this
- **Ruby library:** EngTagger (rule-based Brill tagger)

2. Noun Phrase Chunking

- Extracts complete noun phrases as units
- "a heaviness in my chest" identified as one chunk
- Preserves meaningful phrase structures
- **Ruby libraries:** Treat, ruby-spacy

3. Keyword Extraction Algorithms

- **RAKE** (Rapid Automatic Keyword Extraction) - scores phrases by word frequency and co-occurrence
- **TextRank** - graph-based ranking, like PageRank for words
- **TF-IDF** - identifies distinctive terms
- These find "important" words rather than just grammatical categories

4. Dependency Parsing

- Maps grammatical relationships between words
- Shows that "in my chest" modifies "heaviness"
- Could preserve meaningful phrase structures

5. Coreference Resolution

- Resolves pronouns to their referents
- "I felt it again" → knows "it" refers to "the heaviness" from earlier
- Could maintain continuity across iterations

Techniques to Avoid

Word Embeddings / Semantic Similarity

- Finds related words ("heaviness" "weight" "burden")
- **Breaks Clean Language** - we must use their exact words, not synonyms

Summarization / Paraphrasing

- Explicitly introduces interpretation
- Antithetical to "clean" principles

LLMs / Generative Models

- Non-deterministic, content-generating
- See "Why Not Use an LLM?" section above

Ruby NLP Libraries

Library	Capabilities	Type
EngTagger	POS tagging	Rule-based
Treat	NER, chunking, keywords	Mixed
ruby-spacy	Full spaCy pipeline	Neural (deterministic)
Pragmatic Tokenizer	Sentence/word segmentation	Rule-based
Composable Operations	RAKE implementation	Statistical

Why These Are Safe

These classical NLP algorithms maintain our safety and compliance requirements:

1. **No content generation** - They *extract* words already present, never rephrase or interpret
2. **Deterministic** - Same input always produces same output, fully testable

3. **Not "AI" in regulatory terms** - EU AI Act targets generative systems; text parsing is data processing
4. **Safety features unchanged** - SafetyMonitor's crisis detection (regex patterns) remains separate and unaffected

Recommendation: Stick with **rule-based NLP** (EngTagger, RAKE, regex for prepositional phrases) for maximum compliance confidence. These give improved extraction while maintaining full auditability and no EU AI Act concerns.

Relationship to Emergent Knowledge (EK)

The Six Steps question sequence is inspired by David Grove's Emergent Knowledge approach:

Key EK Principles Applied

1. **Iterative questioning** - EK uses approximately 6 iterations to allow knowledge to emerge
2. **No metaphor requirement** - Unlike Symbolic Modelling, EK doesn't require metaphors. Step 3 uses "What kind of X is that X?" rather than "X is like what?" because metaphors emerge naturally if relevant—they shouldn't be forced
3. **Knowledge emerges from any space except A** - In the ABCD model, A is the client's current position. The Six Spaces (Here, There, Before, After, Inside, Outside) are exploratory entry points, not direct mappings to ABCD
4. **The closing question invites emergence** - "What do you know now that you didn't know before?" helps users recognise that new knowledge has emerged through the process

What Six Steps Is NOT

- **Not A→B coaching** - We're not moving someone from A to B
- **Not therapy** - This is structured self-reflection with safety guardrails
- **Not Symbolic Modelling** - We don't develop elaborate metaphor landscapes

---

Historical Context

The Six Steps question algorithm evolved from an earlier project developed by **Simon Coles** at Amphora Research Systems—a Slackbot designed to help people formulate Clean Language questions.

The Original Slackbot (c. 2018)

The Slackbot offered two modes:

Random Mode: When triggered on a Slack message, it would: - Use EngTagger (a part-of-speech tagger) to extract candidate words - Take the first 5 words (based on the principle that earlier words often carry more significance) - Randomly select one and generate a Clean question from a small set: - "What kind of 'X' is that 'X'?" - "Is there anything else about that 'X'?" - "What happens just before 'X'?"

Thoughtful Mode: A dialog-based approach that let users: - See the top 10 extracted words from a message - Choose which word was most interesting to them - Select a question type (What Kind Of, Just Before, Size/Shape, Anything Else)

Evolution to Six Steps

The Slackbot demonstrated that Clean Language questions could be generated programmatically without an LLM. Six Steps built on this foundation, adding: - A structured 6-iteration sequence aligned with Emergent Knowledge principles - Safety monitoring for unsupervised use - The Six Spaces as exploratory entry points - An integration question to help users recognise emergence

The word extraction was simplified from POS tagging to filler word removal, and the interactive "choose your word" approach was replaced with a deterministic sequence—better suited to guided self-reflection than ad-hoc question formulation.

Summary

The deterministic pattern-matching approach is intentionally simple because Clean Language requires predictability and non-interpretation. The question sequence aligns with Emergent Knowledge principles: iterative "clean" questions allow knowledge to emerge, and the closing question helps users recognise what has emerged. While the word extraction could be refined to preserve more context (especially prepositional phrases), any improvements must maintain the core principle: **reflect, don't interpret**.