bast.ai Technical Explanation

01 Hallucination Prevention

Core Mechanism: We control what reaches the LLM and constrain what it can generate through grounded guardrails and structured retrieval.

The hallucination guarantee comes from three architectural layers:

Layer 1: Intent Classification (Pre-LLM)

- User input → entity extraction against uploaded knowledge graph
- Pattern matching determines routing: greeting, knowledge base (KB) query, out-of-scope, redirect
- If no entity overlap with ontology = automatic "no KB available" path
- **Critical point**: We use a hybrid approach combining NLP techniques with generative AI for intent understanding

Layer 2: Threshold Gating (LLM-Adjacent)

- When KB query proceeds, we use pre-extracted knowledge combined with data annotations plus vector similarity to make retrieval more accurate
- We score against:
  - Extracted intent entities
  - Available knowledge graph nodes
  - Below threshold = route to "cannot answer" agent
  - Above threshold = proceed to KB retrieval
- The LLM never sees a prompt unless we've already confirmed relevant knowledge exists

Layer 3: Structured Retrieval (Post-Classification)

- KB Answer Agent receives:
- Specific page references (OCR-indexed)
- Pre-validated entity context
- Structured formatting instructions
- Output must cite exact page/source or fail validation
- No open-ended generation permitted in response path

Why this works: The LLM only generates formatting/tone for pre-verified factual content. It cannot fabricate because it never receives an open-ended "answer this question" prompt, it receives "format this specific retrieved content for this classified intent."

Composability note: Every step is a discrete micro-service. Clients can add validation layers, adjust thresholds, or insert human-in-loop anywhere in the chain.

02 Compute Reduction

Baseline comparison: Token-to-token cost against standard RAG architecture

Three sources of reduction:

1. Hybrid Symbolic-Neural Execution (40-50% reduction)

- Traditional: Every query → embedding → vector search → LLM generation
- Bast: Intent classification + ontology traversal happen on CPU before LLM involvement
- Greetings/redirects = zero LLM calls
- Simple KB lookups = single targeted call with pre-constructed context
- Reduced embedding model runs and re-ranking iterations

2. Minimal Context Windows (20-30% reduction)

- Traditional RAG: Stuff 4k-8k tokens of retrieved context into every prompt
- Bast: Ontology traversal identifies exact nodes → send only relevant page segments
- Example: "meal times for paraplegics" routes to 2 specific pages, not entire nutrition corpus
- Shorter prompts = exponentially lower token costs (context is quadratic in transformers)

3. Reduced Retry Overhead (10-15% reduction)

- Traditional: Hallucination → validation fail → retry with corrections
- Bast: Grounded guardrails and pre-validation prevent malformed queries from reaching LLM
- Fewer correction loops, reduced safety re-prompts, less human-flagged regeneration

Measured comparison:

- Standard RAG system: ~5k-8k tokens/query (embedding + retrieval + generation)
- Bast: ~500-1.5k tokens/query (classification + targeted generation only)
- Net reduction: 70-85% token usage per interaction

Note on comparison: This isn't *"we're 70% better than GPT-4"*…it's *"we use 70% fewer API tokens than a standard RAG architecture to achieve the same explainable outcome."* Fair comparison would be: our hybrid approach vs. a competitor building guardrails through prompt engineering alone.

Bottom line for both: Hallucination prevention = **grounded guardrails + structured retrieval**, not statistical confidence thresholds

- Compute reduction = hybrid symbolic-neural execution, not model compression

The system is designed so the LLM handles the narrowest, most controlled part of the workflow, not the broadest.