**Hierarchical Semantic Reasoning (HSR) System**

**Novel Methodology Overview**

The HSR system introduces **Multi-Modal Semantic Graphs** combined with **Contextual Chain Reasoning** to create a self-improving document understanding system.

**Core Innovation: Triple-Layer Architecture**

**Layer 1: Intelligent Document Preprocessing**

Raw Documents → Semantic Chunking → Context-Aware Embedding → Knowledge Graph Construction

**Novel Techniques:**

* **Semantic Boundary Detection**: Instead of fixed-size chunks, use sentence transformers to identify semantic boundaries
* **Multi-Modal Context Preservation**: Maintain document structure (tables, headers, lists) as metadata
* **Dynamic Entity Relationship Mapping**: Build real-time knowledge graphs from document entities

**Layer 2: Query Understanding & Intent Reasoning**

Natural Query → Multi-Dimensional NER → Intent Classification → Constraint Extraction

**Breakthrough Approach:**

* **Contextual NER with Domain Adaptation**: Fine-tuned NER models for insurance/legal domains
* **Intent Hierarchy Mapping**: Map queries to decision trees (eligibility → coverage → amount → conditions)
* **Fuzzy Constraint Matching**: Handle vague inputs like "recent policy" or "major surgery"

**Layer 3: Hierarchical Reasoning Engine**

Structured Query → Multi-Hop Retrieval → Logic Chain Construction → Decision Synthesis

**Key Innovations**

**1. Semantic Document Fingerprinting**

* Create unique semantic signatures for each document section
* Enable lightning-fast retrieval even from massive document collections
* Use locality-sensitive hashing for similar clause detection

**2. Chain-of-Reasoning (CoR) with Evidence Tracking**

Query: "46M, knee surgery, Pune, 3-month policy"

Reasoning Chain:

1. Age eligibility check → Find age-related clauses

2. Procedure coverage → Search medical procedure sections

3. Geographic restrictions → Check location-based rules

4. Policy duration requirements → Validate waiting periods

5. Cross-reference conflicts → Resolve contradictory clauses

**3. Confidence-Weighted Decision Fusion**

* Each reasoning step gets a confidence score
* Multiple evidence sources are weighted and combined
* Uncertainty quantification for ambiguous cases

**Technical Architecture**

**Component 1: Smart Document Ingestion**

class DocumentProcessor:

def \_\_init\_\_(self):

self.semantic\_chunker = SemanticChunker()

self.entity\_extractor = DomainNER()

self.knowledge\_graph = Neo4jGraph()

def process\_document(self, doc\_path):

# Extract text while preserving structure

content = self.extract\_with\_structure(doc\_path)

# Semantic chunking instead of fixed-size

chunks = self.semantic\_chunker.chunk\_by\_topics(content)

# Extract entities and relationships

entities = self.entity\_extractor.extract\_domain\_entities(chunks)

# Build knowledge graph

self.knowledge\_graph.add\_document\_entities(entities)

return chunks, entities

**Component 2: Query Intelligence Module**

class QueryIntelligence:

def \_\_init\_\_(self):

self.ner\_model = AutoModelForTokenClassification.from\_pretrained("domain-ner")

self.intent\_classifier = IntentClassifier()

self.constraint\_parser = ConstraintParser()

def understand\_query(self, query):

# Multi-dimensional NER

entities = self.extract\_structured\_entities(query)

# Intent classification

intent\_hierarchy = self.intent\_classifier.classify(query)

# Constraint extraction

constraints = self.constraint\_parser.parse(query, entities)

return QueryStructure(entities, intent\_hierarchy, constraints)

**Component 3: Hierarchical Reasoning Engine**

class ReasoningEngine:

def \_\_init\_\_(self):

self.retriever = HybridRetriever()

self.logic\_chains = LogicChainBuilder()

self.decision\_synthesizer = DecisionSynthesizer()

def reason(self, structured\_query, document\_store):

# Multi-hop retrieval

evidence\_chains = []

for constraint in structured\_query.constraints:

chain = self.retriever.multi\_hop\_retrieve(

constraint, document\_store, max\_hops=3

)

evidence\_chains.append(chain)

# Build logic chains

reasoning\_paths = self.logic\_chains.build\_paths(evidence\_chains)

# Synthesize decision

decision = self.decision\_synthesizer.synthesize(

reasoning\_paths, confidence\_threshold=0.75

)

return decision

**Optimization Strategies**

**1. Hybrid Retrieval System**

* **Dense Retrieval**: Sentence-BERT for semantic similarity
* **Sparse Retrieval**: BM25 for exact keyword matching
* **Graph Traversal**: Neo4j for relationship-based retrieval
* **Fusion Scoring**: Combine all three with learned weights

**2. Caching & Performance**

* **Semantic Cache**: Cache similar queries using embedding similarity
* **Incremental Learning**: Update embeddings as new documents are added
* **Async Processing**: Parallel processing of multiple reasoning chains

**3. Model Optimization**

* **Quantization**: Use INT8 quantized models for faster inference
* **Knowledge Distillation**: Distill large models into smaller, faster ones
* **Dynamic Batching**: Batch similar queries for efficient processing

**Sample Implementation Flow**

**Input Processing**

{

"query": "46M, knee surgery, Pune, 3-month policy",

"structured\_entities": {

"age": 46,

"gender": "male",

"procedure": "knee surgery",

"location": "Pune",

"policy\_duration": "3 months"

},

"intent\_hierarchy": ["eligibility\_check", "coverage\_verification", "amount\_calculation"],

"constraints": ["age\_limit", "procedure\_coverage", "waiting\_period", "geographic\_coverage"]

}

**Reasoning Process**

{

"reasoning\_chains": [

{

"constraint": "age\_limit",

"evidence": [

{

"clause": "Policy covers individuals aged 18-65",

"document": "policy\_terms.pdf",

"confidence": 0.95,

"page": 5

}

],

"decision": "ELIGIBLE"

},

{

"constraint": "procedure\_coverage",

"evidence": [

{

"clause": "Orthopedic surgeries including knee procedures are covered",

"document": "coverage\_details.pdf",

"confidence": 0.88,

"page": 12

}

],

"decision": "COVERED"

},

{

"constraint": "waiting\_period",

"evidence": [

{

"clause": "30-day waiting period for elective surgeries",

"document": "policy\_terms.pdf",

"confidence": 0.92,

"page": 8

}

],

"decision": "WAITING\_PERIOD\_APPLIES"

}

]

}

**Final Output**

{

"decision": "CONDITIONALLY\_APPROVED",

"amount": 150000,

"conditions": ["Must wait 30 days from policy start date"],

"confidence": 0.87,

"justification": {

"eligible\_clauses": [

"Age criteria met (18-65 years) - policy\_terms.pdf:5",

"Knee surgery covered under orthopedic procedures - coverage\_details.pdf:12"

],

"restricting\_clauses": [

"30-day waiting period for elective surgeries - policy\_terms.pdf:8"

]

},

"reasoning\_trace": "Chain of reasoning available for audit"

}

**Competitive Advantages**

1. **Explainable AI**: Complete reasoning trace with clause-level citations
2. **Handles Ambiguity**: Fuzzy matching and confidence scoring
3. **Scalable**: Graph-based architecture scales to millions of documents
4. **Self-Improving**: Learns from user feedback and corrections
5. **Multi-Modal**: Handles PDFs, Word docs, emails seamlessly
6. **Real-Time**: Optimized for sub-second response times

**Implementation Stack**

* **LLM**: Fine-tuned Llama-2 7B for domain-specific reasoning
* **Embeddings**: Sentence-BERT + domain-adapted embeddings
* **Vector DB**: Qdrant for fast similarity search
* **Graph DB**: Neo4j for relationship modeling
* **NER**: spaCy + custom insurance/legal models
* **Backend**: FastAPI with async processing
* **Caching**: Redis for semantic query caching

**Success Metrics**

* **Accuracy**: >95% on standard insurance claim datasets
* **Speed**: <2 seconds average response time
* **Explainability**: 100% decisions traceable to source clauses
* **Handling Ambiguity**: >80% accuracy on vague queries
* **Scalability**: Handle 10K+ concurrent queries

This HSR methodology combines cutting-edge NLP with graph reasoning to create a system that doesn't just retrieve information—it truly understands and reasons about document content to make intelligent decisions.