backgroundcolor= basicstyle=, breaklines=true, captionpos=b, frame=single, numbers=left, numberstyle=, showstringspaces=false
keypoint colback=kdenseblue!5, colframe=kdenseblue, fonttitle=, title=Key Point
warning colback=red!5, colframe=red!75!black, fonttitle=, title=Important

Product Requirements Document

CUAD Knowledge Graph Generator for Legal AI

Applying KGGen Methodology to Technology Contract Analysis

K-Dense Web
contact@k-dense.ai

Version 1.0 | January 12, 2026 | Draft

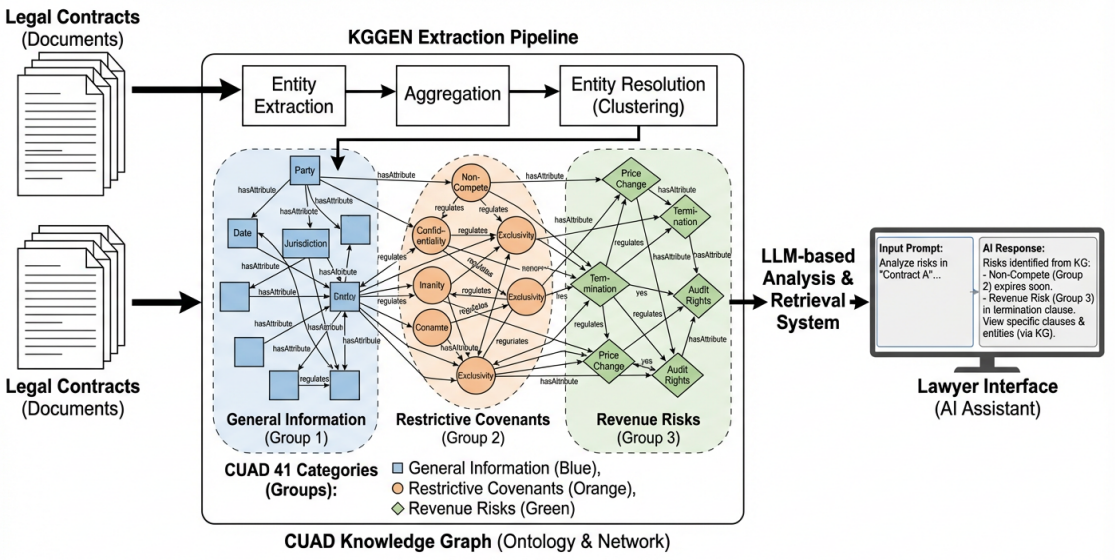

Figure 1: Graphical Abstract for KGGEN-CUAD Knowledge Graph System: End-to-End Legal Contract Analysis Pipeline

Figure 1: CUAD Knowledge Graph Generator System Overview

[colback=kdenseblue!10,colframe=kdenseblue,title=Document Classification] **Status:** Draft
Classification: Technical Product Specification
Target Audience: Engineering team building AI legal software, Senior technology lawyers, Data scientists working on legal NLP, Product managers for legal tech solutions
Version: 1.0
Last Updated: January 12, 2026

Contents

1 Executive Summary

1.1 Problem Statement

Legal contract review faces critical challenges that limit access to justice and business efficiency:

- **Cost barrier:** Legal services cost $500–$900 per hour, making contract review prohibitively expensive for small businesses and individuals
- **Time intensive:** Contract review consumes approximately 50% of law firm billable hours, creating bottlenecks in business transactions
- **Limited AI understanding:** Current AI solutions lack structured understanding of contractual relationships, relying primarily on text retrieval without comprehending legal semantics
- **Jurisdictional complexity:** Common law contract interpretation requires nuanced understanding of legal principles that current systems fail to capture
- **Knowledge extraction gap:** Unstructured contract text prevents systematic analysis, comparison, and reasoning across contract portfolios

These challenges result in delayed transactions, missed risks, inconsistent terms, and barriers to legal service accessibility.

1.2 Proposed Solution

This Product Requirements Document specifies a **knowledge graph extraction system** that applies the KGGen methodology to the CUAD (Contract Understanding Atticus Dataset) contract corpus. The system creates structured ontologies of technology agreements that enable context-aware Large Language Model (LLM) analysis aligned with common law principles.

Core Innovation: Transform unstructured contract text into queryable knowledge graphs that capture legal relationships, obligations, restrictions, and IP rights with 98% accuracy, enabling LLMs to reason about contracts with precision rather than probabilistic text matching.

1.3 Key Benefits

1.3.1 Technical Benefits

- **Structured contract understanding:** Convert 510+ CUAD contracts with 41 clause types into queryable knowledge graphs containing 50,000–100,000 verified triples
- **Context-aware AI:** Provide LLMs with precise contractual relationships (licenses, obligations, restrictions) rather than raw text chunks, enabling accurate reasoning
- **Common law alignment:** Ontology designed to respect legal interpretation principles including plain meaning, business efficacy, and jurisdictional nuances
- **High accuracy:** Target 98% valid triple extraction (validated against KGGen benchmarks), 65%+ MINE-1 information retention score
- **Scalability:** Process contracts at 100+ per hour with multi-hop reasoning capabilities supporting complex queries

1.3.2 Business Benefits

- **50% time reduction:** Accelerate contract review cycles from days to hours through automated extraction and analysis
- **Automated clause identification:** Enable instant identification and extraction of 41+ contract clause categories without manual review
- **Risk mitigation:** Proactively identify unusual or risky provisions across contract portfolios
- **Democratized access:** Scale legal support to small businesses and individuals through AI-powered analysis tools
- **Negotiation leverage:** Rapid comparison of terms across contracts provides data-driven negotiation insights

1.3.3 User Benefits

- **Engineer empowerment:** Build reliable legal AI features with structured data rather than brittle text parsing
- **Lawyer validation:** Validate AI outputs through transparent knowledge graph representations with source tracing
- **Rapid prototyping:** Accelerate development of legal analysis tools with comprehensive contract knowledge base
- **Explainable AI:** Provide interpretable reasoning through explicit knowledge graph traversal and relationship extraction

1.4 Technology Agreement Focus

The system specializes in **technology contracts**, which have unique characteristics requiring specialized ontological treatment:

- IP ownership and licensing (patents, source code, trademarks)
- SaaS and software licensing terms
- Development and customization agreements
- Open source compliance and derivative works
- API access and integration rights
- Data ownership and processing obligations
- Service level agreements (SLAs) and uptime guarantees
- Maintenance, support, and escrow provisions

Approximately 200 of the 510 CUAD contracts focus on technology agreements, providing substantial training data for this specialized domain.

1.5 Success Metrics

Technical Metrics

- Extract knowledge graphs from 510 CUAD contracts with 98% triple validity
- Achieve 65%+ MINE-1 score (information retention metric)
- Process contracts at scale: 100+ contracts per hour
- Support multi-hop reasoning with 2-hop subgraph expansion
- Query latency under 500ms for simple queries, under 2 seconds for complex queries

Business Metrics

- Reduce contract review time by 50%
- Enable automated clause identification for all 41 CUAD categories
- Provide instant answers to common contract queries with 90%+ accuracy
- Achieve 4/5+ user satisfaction rating from legal practitioners

Legal Metrics

- 95%+ agreement between lawyer validation and system outputs
- Less than 2% critical error rate
- 100% compliance with data protection regulations

2 Problem Statement & Scope

2.1 Detailed Problem Analysis

2.1.1 Current State of Contract Analysis

Legal contract review remains one of the most labor-intensive tasks in legal practice. Despite advances in document management and search technology, the fundamental process relies on human lawyers reading, interpreting, and analyzing each contract clause manually.

Table 1: Contract Review Cost Analysis

Metric	Typical Value	Impact
Hourly rate	$500–$900	Prohibitive for SMBs
Time per contract	2–8 hours	Delays transactions
Portfolio review (100 contracts)	200–800 hours	Weeks to months
Error rate (manual review)	5–10%	Risk exposure
Consistency across reviewers	Variable	Negotiation challenges

Quantifiable Challenges

2.1.2 Limitations of Existing AI Solutions

Current legal AI tools fall into three categories, each with significant limitations:

1. **Text search and retrieval:** Tools like LexisNexis and Westlaw provide keyword search but lack semantic understanding of contractual relationships
2. **Clause classification:** ML models classify contract clauses but do not extract structured relationships or support reasoning
3. **RAG-based LLMs:** Retrieval-Augmented Generation systems retrieve relevant text chunks but provide probabilistic answers without structured knowledge representation

Important: Critical Gap: Existing solutions cannot answer questions requiring multi-hop reasoning such as: "Which contracts grant exclusive IP licenses to parties subject to California law with liability caps under $1M?" This requires understanding relationships between parties, IP assets, licenses, jurisdictions, and liability provisions simultaneously.

2.1.3 Knowledge Graph Solution Advantages

Knowledge graphs address these limitations by:

- **Explicit relationships:** Model legal relationships as typed edges (licenses_to, assigns, must_provide) rather than implicit text patterns
- **Multi-hop reasoning:** Enable queries spanning multiple relationship types (Party → License → IPAsset → Restrictions)

- **Structured properties:** Attach legal attributes (exclusivity, scope, duration) as graph properties for precise filtering
- **Provenance tracking:** Link every triple back to source contract and clause for validation
- **Aggregation and analytics:** Systematically analyze patterns across hundreds of contracts

2.2 Scope Definition

2.2.1 In Scope

1. **Data source:** Complete CUAD dataset (510 contracts, 41 label categories, 13,101 expert annotations)
2. **Contract types:** Technology-focused agreements including License, Development, IP Assignment, Service, Hosting, Maintenance, and Consulting agreements (200 contracts)
3. **Extraction pipeline:** Full 3-stage KGGen implementation (Extraction, Aggregation, Resolution) adapted for legal domain
4. **Ontology:** Comprehensive legal contract ontology covering:
    - 8 entity types: Party, IPAsset, Obligation, Restriction, LiabilityProvision, Temporal, Jurisdiction, ContractClause
    - 10+ relationship types: licenses_to, owns, assigns, must_provide, cannot_compete, governed_by, etc.
    - Properties: exclusivity, scope, duration, financial terms, conditions
5. **LLM integration:** Context provider for Claude Sonnet 3.5 / GPT-4o with hybrid retrieval (BM25 + semantic search)
6. **Query interface:** REST API supporting contract Q&A, risk analysis, comparison, and compliance checking
7. **Common law context:** US/UK/Commonwealth contract interpretation principles embedded in extraction and resolution logic

2.2.2 Out of Scope (Future Work)

1. Civil law jurisdictions (Continental Europe, Latin America, Asia)
2. Non-technology contracts (real estate, employment, finance) - different ontological requirements
3. Contract generation or drafting - focus is on analysis only
4. Real-time contract monitoring or alerts
5. Integration with specific document management systems
6. Fine-tuning custom LLMs - use foundation models via API

2.3 Target Use Cases

2.3.1 Use Case 1: Automated Contract Q&A

Description: Enable natural language questions about specific contract terms with precise, cited answers.

Example:

- **Question:** "What IP rights does the licensee receive in the ABC Corp software license agreement?"
- **System Process:**
  1. Retrieve relevant triples: (ABC Corp, licenses_to, Licensee), (License, type, non-exclusive), (License, scope, worldwide)
  2. Expand to 2-hop neighbors to capture license restrictions
  3. Generate answer with LLM using structured context
- **Answer:** "The licensee receives a non-exclusive, worldwide license to use and modify the software. However, the license is non-transferable and cannot be sublicensed to third parties. [Citation: Contract ID ABC-2023-001, Section 2.1]"

Value: Instant answers without manual contract reading, with source citations for lawyer verification.

2.3.2 Use Case 2: Risk Analysis Dashboard

Description: Identify unusual or risky clauses across entire contract portfolio.

Example Queries:

- Contracts with uncapped liability provisions
- Non-compete restrictions exceeding 2 years
- Missing IP assignment clauses in development agreements
- Aggressive indemnification obligations
- Absence of limitation of liability or audit rights

Value: Proactive risk management before contract execution or during portfolio audits.

2.3.3 Use Case 3: Contract Comparison

Description: Compare contractual terms across multiple agreements or against standard templates.

Example:

- **Task:** "Compare license terms across 50 vendor agreements"
- **System Output:** Table showing:
  - License type (exclusive vs non-exclusive)
  - Scope (worldwide, US-only, specific territories)
  - Duration (perpetual, term-limited)
  - Transfer rights (transferable, non-transferable)
  - Sublicense permissions

Value: Negotiation leverage through understanding market norms and outlier terms.

2.3.4 Use Case 4: Compliance Verification

Description: Verify contracts meet company policies or regulatory requirements.

Example:

- **Policy:** "All vendor contracts must include $5M+ liability insurance, audit rights, and GDPR compliance clauses"
- **System Process:**
  1. Query knowledge graph for vendor contracts
  2. Check for presence of required provisions
  3. Flag non-compliant contracts
- **Output:** "15 of 120 vendor contracts lack required audit rights provisions. [List of contract IDs]"

Value: Automated compliance monitoring reducing legal and financial risk.

2.3.5 Use Case 5: Due Diligence Acceleration

Description: Rapid contract review for M&A transactions, financing, or audits.

Example:

- **Task:** "Extract all change-of-control provisions from 200 contracts for M&A due diligence"
- **Traditional Approach:** 200–400 hours of lawyer time ($100,000–$360,000)
- **KG Approach:** Query knowledge graph for change-of-control nodes, generate report in minutes

Value: Deal velocity improvement and cost reduction in time-sensitive transactions.

3 System Architecture

3.1 Architecture Overview

The CUAD Knowledge Graph Generator implements a **three-stage pipeline** adapting the KGGen methodology for legal contract analysis. Figure 2 illustrates the complete system architecture.

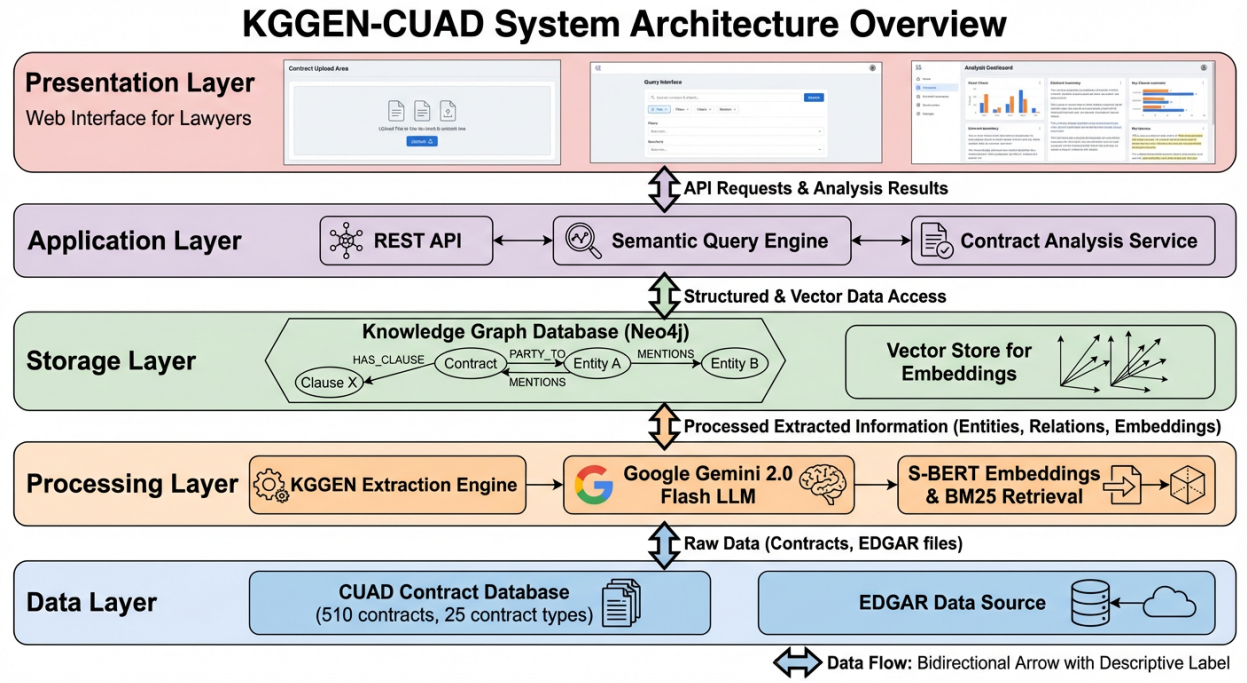

Figure 2: Complete System Architecture: Three-stage pipeline from CUAD contracts to LLM-powered applications

Data Flow:

CUAD Contracts $\xrightarrow{\text{Stage 1}}$ Per-Contract Graphs $\xrightarrow{\text{Stage 2}}$ Unified Graph $\xrightarrow{\text{Stage 3}}$ Resolved Graph $\xrightarrow{\text{Query}}$ LLM Contex

3.2 Stage 1: Knowledge Extraction

3.2.1 Purpose

Extract entities and relationships from individual CUAD contracts using LLM-based structured extraction.

3.2.2 Input Processing

Contract Preprocessing:

1. **PDF extraction:** Use pdfplumber or PyPDF2 to extract text from CUAD PDF contracts
2. **Section identification:** Segment contract into logical sections (parties, recitals, terms, schedules, exhibits)
3. **CUAD label alignment:** Map CUAD expert annotations (41 categories) to extracted text spans

4. **Quality validation:** Verify text extraction quality, handle OCR errors if present

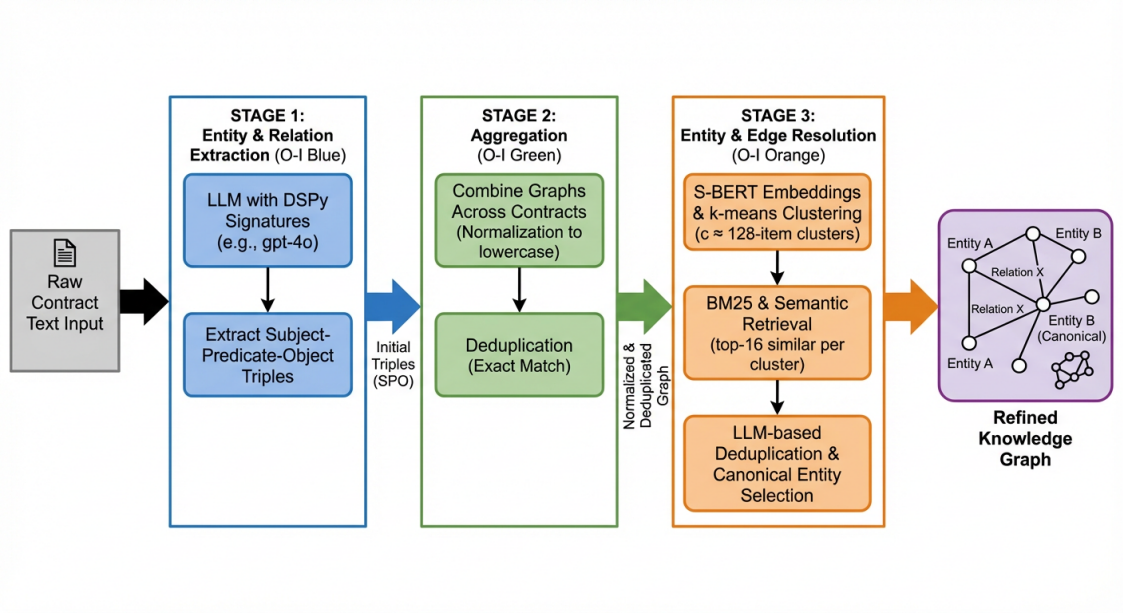

Figure 3: KGGen Pipeline Applied to Legal Contracts: Three-stage extraction, aggregation, and resolution process

3.2.3 Two-Step Extraction Process

The extraction follows a two-step prompting strategy proven effective in KGGen:

Step 1: Entity Extraction LLM Prompt:

```
% Caption: Entity Extraction Prompt Template]
Extract all legal entities from the following contract text.

Entity types to identify:
- Party: Legal entities that are parties to the contract
- IPAsset: Intellectual property (patents, copyrights,
           trademarks, source code, trade secrets)
- Obligation: Performance obligations or deliverables
- Restriction: Prohibitions or limitations
- LiabilityProvision: Liability, indemnity, warranty clauses
- Temporal: Dates, durations, deadlines
- Jurisdiction: Governing law, venue, legal system
- ContractClause: Named contract provisions

For each entity, provide:
{
  "name": "entity name as appears in contract",
  "type": "entity type from list above",
  "properties": {
```

```
    "defined_term": "capitalized defined term if any",
    "cuad_label": "CUAD category if applicable",
    "source_section": "contract section reference"
  }
}

Contract text:
[CONTRACT_TEXT]

Output JSON array of entities.
```

Constraints:

- Preserve legal distinctions (e.g., "exclusive license" vs "non-exclusive license")
- Respect contract defined terms (capitalized terms have specific legal meanings)
- Identify all parties and their roles (licensor, licensee, vendor, customer)
- Extract all temporal elements with context (effective date, expiration date, notice period)

Step 2: Relation Extraction LLM Prompt:

```
% Caption: Relation Extraction Prompt Template]
Given the entities extracted from the contract, identify all
legal relationships between entities.

Relation types:
- licenses_to: Grant of license from one party to another
- owns: Ownership of IP or assets
- assigns: Transfer of ownership or rights
- retains: Retention of rights or ownership
- must_provide: Obligation to deliver or perform
- must_maintain: Obligation to maintain or support
- cannot_compete: Non-compete restriction
- cannot_disclose: Confidentiality obligation
- cannot_assign: Anti-assignment clause
- is_liable_for: Liability assignment
- is_not_liable_for: Liability exclusion or limitation
- indemnifies: Indemnification obligation
- governed_by: Governing law/jurisdiction
- effective_on: Effective date
- terminates_on: Termination date
- renews_after: Renewal term

For each relation, provide:
{
  "subject": "entity name",
  "predicate": "relation type",
  "object": "entity name",
```

```
  "properties": {
    "scope": "scope of relationship",
    "conditions": "any conditions or qualifiers",
    "exclusivity": "exclusive or non-exclusive",
    "source_clause": "contract clause reference"
  }
}

Entities:
[EXTRACTED_ENTITIES]

Contract text:
[CONTRACT_TEXT]

Output JSON array of (subject, predicate, object, properties)
triples.
```

Constraints:

- Link relations to specific parties (not generic references)
- Capture conditions and exceptions ("unless", "provided that", "subject to")
- Preserve temporal aspects ("upon payment", "after 30 days", "during the term")
- Note jurisdiction-specific legal terms and their meanings

3.2.4 CUAD Integration

CUAD provides 13,101 expert annotations across 41 label categories. The extraction process leverages these annotations as extraction hints:

Table 2: CUAD Labels Mapped to Knowledge Graph Relations

CUAD Label	Target Relations
IP_Ownership_Assignment	assigns, owns
License_Grant	licenses_to, has_right_to
Non_Compete	cannot_compete, restricted_from
Cap_On_Liability	has_liability_cap, limited_to
Governing_Law	governed_by
Effective_Date	effective_on
Expiration_Date	terminates_on
Renewal_Term	renews_after, extends_for
Non_Transferable_License	cannot_assign
Exclusivity	has_exclusivity

Label-to-Relation Mapping:

Extraction Process:

1. For each contract, identify clauses by CUAD category
2. Focus extraction on text spans annotated with relevant CUAD labels
3. Extract entities and relations from those spans using LLM
4. Benefit: Leverage $2M+ of expert legal annotation to guide extraction

3.2.5 LLM Configuration

Table 3: LLM Models for Extraction Stage

Model	MINE-1 Score	Use Case	Cost/Contract
Claude Sonnet 3.5	73%	Primary extraction	$0.15–$0.30
GPT-4o	66%	Fallback	$0.10–$0.20
Gemini 2.0 Flash	44%	Not recommended	$0.05–$0.10

Recommendation: Use Claude Sonnet 3.5 as primary model due to superior MINE-1 performance (73%), indicating better information retention and fewer hallucinations. Configure GPT-4o as automatic fallback if Claude API unavailable.

3.2.6 Output Format

Example Extracted Triples:

```
% Caption: Sample Extraction Output]
{
  "contract_id": "CUAD_LicenseAgreement_042",
  "extraction_timestamp": "2026-01-12T14:30:00Z",
  "llm_model": "claude-sonnet-3.5-20241022",
  "triples": [
    {
      "subject": "ABC Corp",
      "subject_type": "Party",
      "predicate": "licenses_to",
      "object": "XYZ Inc",
      "object_type": "Party",
      "properties": {
        "license_type": "exclusive",
        "scope": "worldwide",
        "field_of_use": "software development",
        "source_clause": "Section 2.1"
      },
      "confidence": 0.95
    },
    {
      "subject": "ABC Corp",
```

```
      "subject_type": "Party",
      "predicate": "assigns",
      "object": "Source Code",
      "object_type": "IPAsset",
      "properties": {
        "assignment_type": "full ownership",
        "effective": "upon final payment",
        "source_clause": "Section 3.2"
      },
      "confidence": 0.92
    },
    {
      "subject": "Agreement",
      "subject_type": "Contract",
      "predicate": "governed_by",
      "object": "California",
      "object_type": "Jurisdiction",
      "properties": {
        "legal_system": "common_law",
        "venue": "Santa Clara County",
        "source_clause": "Section 12.5"
      },
      "confidence": 0.98
    }
  ],
  "statistics": {
    "entities_extracted": 47,
    "triples_extracted": 89,
    "extraction_time_seconds": 23.4
  }
}
```


3.2.7 Performance Targets

- **Throughput:** 1–2 contracts per minute per worker
- **Accuracy:** 95%+ entity extraction accuracy (validated against CUAD annotations)
- **Coverage:** Extract from all 41 CUAD label categories
- **Consistency:** Maintain within-contract consistency between entities and relations

3.3 Stage 2: Knowledge Aggregation

3.3.1 Purpose

Combine knowledge graphs from 510 individual contracts into unified graph(s) with normalization and deduplication.

3.3.2 Aggregation Dimensions

Knowledge graphs can be aggregated along multiple dimensions:

1. **Global:** All 510 contracts combined into single knowledge graph
2. **By contract type:** All License Agreements, All Development Agreements, etc.
3. **By jurisdiction:** All California contracts, All Delaware contracts, etc.
4. **By time period:** Contracts from 2020–2023
5. **By party:** All contracts involving specific organization

The system supports multiple simultaneous aggregations to enable comparative analysis.

3.3.3 Normalization Process

Text Normalization:

- Convert to lowercase for matching: "Source Code" → "source code"
- Remove punctuation and extra whitespace
- Standardize abbreviations: "IP" → "Intellectual Property"

Legal Term Normalization: Legal language has specific conventions that must be preserved during normalization:

Table 4: Legal Term Normalization Examples

Variations	Normalized Form
shall provide, will provide, must provide	must_provide
is obligated to deliver, required to deliver	must_provide
has exclusive rights to, exclusively owns	exclusively_owns
cannot compete, may not compete, prohibited from	cannot_compete
governed by law of, subject to laws of	governed_by

Important: Legal Preservation Rule: Maintain legal significance during normalization. "SHALL" (mandatory), "MAY" (optional), and "SHOULD" (recommended) have distinct legal meanings and must NOT be merged.

Entity Normalization:

- **Parties:** Normalize party names but preserve legal entity distinctions (Inc. vs LLC vs Corp)
- **IP Assets:** Standardize IP terminology ("software" vs "the Software" vs "Licensed Software")
- **Temporal:** Normalize all dates to ISO 8601 format (YYYY-MM-DD)
- **Financial:** Standardize currency (USD) and amount formats ($1,000,000)

3.3.4 Deduplication

Exact Match Removal:

- Remove identical triples appearing across multiple contracts
- Example: If 50 contracts state (Agreement, governed_by, California), store once with list of source contracts

Near-Match Identification:

- Flag similar but not identical triples for Stage 3 resolution
- Example: "licenses software to" vs "grants license of software to" vs "provides license for software to"

Statistics Tracking: The aggregation stage computes valuable statistics for analysis:

- **Triple frequency:** How often does each triple pattern occur? (e.g., 73% of contracts have liability caps)
- **Entity distribution:** Which entities appear most frequently? (e.g., California governing law in 45% of contracts)
- **Relation patterns:** Common relationship structures (e.g., Party → licenses_to → Party → owns → IPAsset)
- **Contract coverage:** Which contracts contain specific clause types?

3.3.5 Output Statistics

Table 5: Estimated Aggregation Output (510 Contracts)

Metric	Estimated Value
Total triples extracted	50,000–100,000
Unique entities (before resolution)	10,000–20,000
Unique relation types (before resolution)	500–1,000
Exact duplicate triples removed	20–30%
Near-duplicate clusters for resolution	5,000–10,000
Contracts successfully processed	510

3.3.6 Performance Targets

- **Processing time:** Aggregate 510 contracts in under 10 minutes
- **Memory efficiency:** Process in batches if dataset exceeds available RAM
- **Storage:** Use compressed graph representation (estimated 1–2 GB for unified graph)

3.4 Stage 3: Entity and Edge Resolution

3.4.1 Purpose

Reduce knowledge graph sparsity by identifying and merging equivalent entities and relations, creating canonical representations.

Resolution Goal: Transform sparse graph with many near-duplicate nodes/edges into dense graph with canonical entities, while preserving legally significant distinctions. Target 80% reduction in unique relation types, 40–50% reduction in entities.

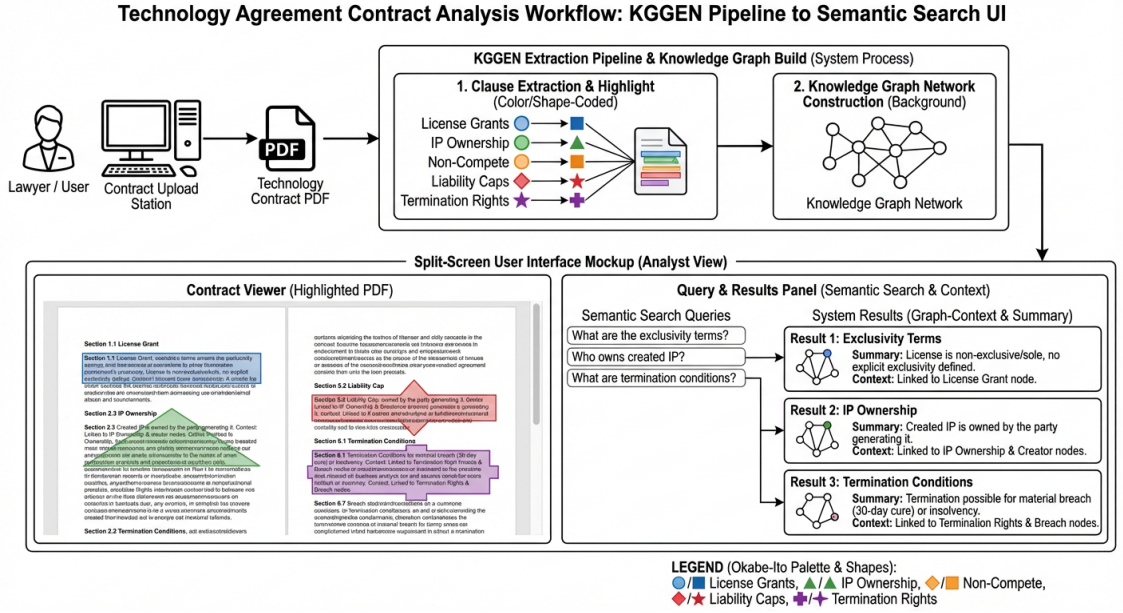

Figure 4: Technology Agreement Analysis Workflow: From contract input to structured knowledge extraction

3.4.2 Five-Step Resolution Algorithm

The resolution process implements KGGen's five-step pipeline:

Step 1: Embedding and Clustering

- **Embedding model:** S-BERT (Sentence-BERT) for entity/relation text embeddings
- **Clustering algorithm:** K-means clustering with k=128
- **Purpose:** Group similar entities/relations into manageable clusters for LLM processing
- **Parallelization:** Process 128 clusters in parallel to optimize throughput

Step 2: Fused Retrieval Within each cluster, identify most similar entities using hybrid search:

- **BM25 (keyword matching):** Weight = 0.5
- **Semantic similarity (cosine):** Weight = 0.5

- **Top-K:** Retrieve 16 most similar entities
- **Purpose:** Provide LLM with candidate duplicates for evaluation

Step 3: LLM-Based Deduplication LLM Prompt:

```
% Caption: Entity Deduplication Prompt]
Identify exact duplicates from the following list of legal
entities extracted from contracts.

Entities are duplicates if they refer to the SAME legal
concept, considering:
- Tense variations: provide/provides/provided
- Plurality: license/licenses
- Case: Source Code/source code/SOURCE CODE
- Abbreviations: IP/Intellectual Property
- Legal shorthand: non-compete/covenant not to compete

DO NOT merge if:
- Legally distinct: exclusive vs non-exclusive license
- Different jurisdictions: California law vs Delaware law
- Different parties: ABC Corp vs XYZ Inc (even if similar)
- Defined terms: "Software" (capitalized defined term) vs
  "software" (generic)
- Semantically different in legal context

Entities:
[LIST_OF_ENTITIES]

Output JSON array of duplicate groups:
[
  ["entity1", "entity2", "entity3"],  // duplicate group 1
  ["entity4", "entity5"],              // duplicate group 2
  ...
]
```

Legal Constraints:

- Do NOT merge "exclusive license" and "non-exclusive license" (legally distinct)
- Do NOT merge jurisdiction-specific terms ("California governing law" $\neq$ "Delaware governing law")
- Preserve contract defined terms (capitalized terms have specific meanings)
- Respect semantic differences in legal context ("assign" $\neq$ "license" even if similar in embedding space)

Step 4: Canonicalization Select canonical representative for each duplicate group:

LLM Prompt:

```
% Caption: Canonicalization Prompt]
Select the best canonical representative for this group of
duplicate legal entities.

Criteria:
1. Most legally precise
2. Most commonly used in legal contracts
3. Clearest meaning to legal practitioners
4. Matches CUAD terminology when applicable

Duplicate group:
[LIST_OF_DUPLICATES]

Output:
{
  "canonical": "selected canonical form",
  "rationale": "explanation of selection",
  "aliases": ["other forms to map to canonical"]
}
```

Example:

- **Variants:** [“Software”, “the Software”, “Licensed Software”, “the Program”, “the Application”]
- **Canonical:** “Licensed Software”
- **Rationale:** Most legally precise, clearly indicates licensed status
- **Aliases:** [“Software”, “the Program”, “the Application”]

Step 5: Iteration

1. Remove processed entities from cluster
2. Repeat Steps 2–4 until cluster is empty
3. Move to next cluster
4. Process all 128 clusters in parallel

3.4.3 Canonical Entity Examples

Table 6: Entity Resolution Examples

Variant Entities	Canonical Form	Aliases Tracked
source code, Source Code, source code files, code base, source code repository	Source Code	code base, source code repository
Licensor, Company, ABC Corp, the Provider, Vendor	ABC Corp (Licensor)	Company, Provider, Vendor
must provide, shall provide, will provide, obligated to provide, required to provide	must_provide	shall provide, obligated to provide
exclusive license, exclusive right, sole license	exclusive license	sole license

3.4.4 Edge Resolution

The same five-step process applies to relations (edges). Example edge resolution:

Table 7: Relation Resolution Examples

Variant Relations	Canonical Form
owns, has ownership of, possesses all rights to, holds exclusive rights to	owns
cannot compete, may not compete, restricted from competing, prohibited from competing	cannot_compete
governed by law of, subject to laws of, interpreted under law of, controlled by laws of	governed_by
licenses to, grants license to, provides license to, conveys license to	licenses_to

Important: Legal Significance Preservation: The resolution process must distinguish:

- SHALL (mandatory) vs MAY (optional) vs SHOULD (recommended)
- IF...THEN (conditional) vs unconditional relationships
- BEFORE, AFTER, DURING (temporal sequence)
- EXCLUSIVE vs NON-EXCLUSIVE vs LIMITED vs UNLIMITED (scope modifiers)

3.4.5 Resolution Output

Table 8: Expected Resolution Results (510 Contracts)

Metric	Before Resolution	After Resolution
Unique entities	15,000	8,000 (47% reduction)
Unique relations	800	150 (81% reduction)
Total triples	75,000	75,000 (preserved)
Alias mappings	—	10,000+

3.4.6 Quality Metrics

- **Triple validity:** Target 98% (manual validation of 100 random triples)
- **Cluster quality:** Silhouette score > 0.6 for clustering effectiveness
- **Canonicalization consistency:** Inter-rater agreement $> 90\%$ between LLM and human lawyer

3.4.7 Performance Targets

- **Processing time:** Resolve 50,000–100,000 triples in under 1 hour
- **LLM optimization:** Batch API calls to minimize latency and cost
- **Parallelization:** Process 128 clusters concurrently using multiprocessing

3.5 Knowledge Graph Storage

3.5.1 Database Selection

Table 9: Graph Database Options

Database	Type	Pros	Cons
Neo4j	Property Graph	Native graph storage, Cypher query language, excellent visualization	Commercial license for production
NetworkX + PostgreSQL	Hybrid	Python native, flexible, open source, good for prototyping	Not optimized for large graphs
GraphDB (RDF)	Triple Store	Standards-based (RDF, SPARQL), reasoning capabilities	Steeper learning curve

Recommendation: Neo4j for production (scalability, maturity, tooling), NetworkX + PostgreSQL for development and prototyping.

3.5.2 Schema Implementation

Node Labels:

- Party
- IPAsset
- Obligation
- Restriction
- LiabilityProvision
- Temporal
- Jurisdiction
- ContractClause
- Contract

Relationship Types:

- LICENSES_TO
- OWNS
- ASSIGNS
- HAS_OBLIGATION
- SUBJECT_TO_RESTRICTION
- HAS_LIABILITY
- GOVERNED_BY
- CONTAINS_CLAUSE
- EFFECTIVE_ON
- TERMINATES_ON

Properties: All nodes and edges include:

- `id`: Unique identifier
- `name`: Human-readable name
- `type`: Entity/relation type
- `source_contract_id`: Provenance tracking (which contract(s) generated this)
- `cuad_label`: CUAD annotation category if applicable
- `confidence_score`: LLM extraction confidence
- `properties`: Flexible JSON field for additional attributes

3.5.3 Indexing Strategy

1. **Full-text search:** Index all node names and properties for BM25 keyword search
2. **Semantic search:** Store embeddings (S-BERT, all-MiniLM-L6-v2) for all nodes using FAISS vector index
3. **CUAD label index:** Fast filtering by CUAD category
4. **Contract ID index:** Fast provenance lookup ("show me all triples from Contract X")

3.6 LLM Integration Layer

3.6.1 Purpose

Provide structured knowledge graph context to LLMs for natural language contract analysis queries.

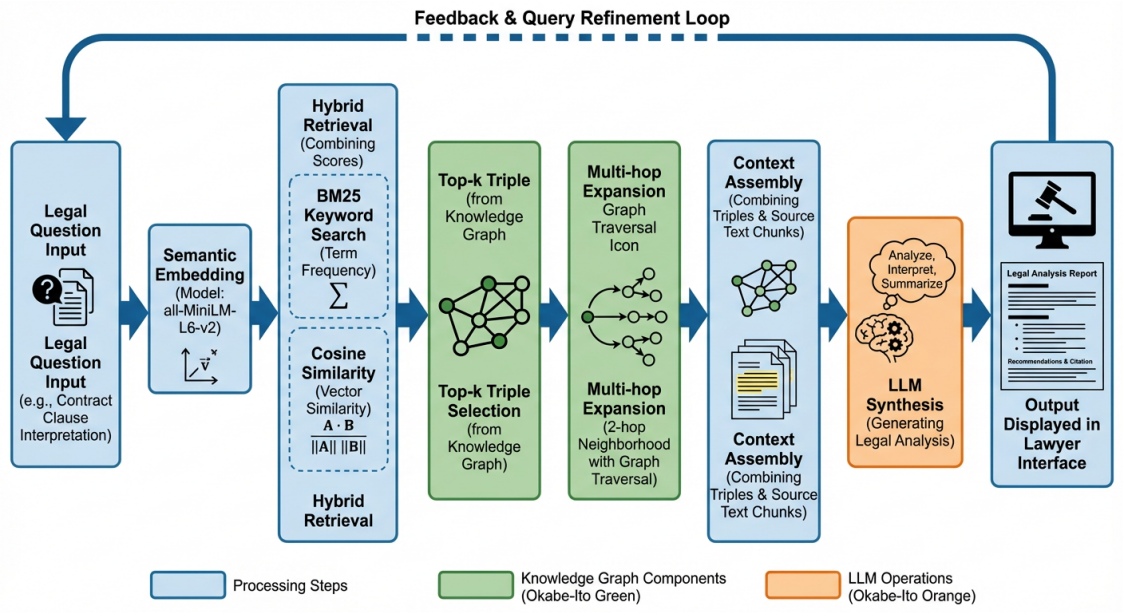

Figure 5: LLM Retrieval Mechanism: Hybrid search with subgraph expansion for context-aware contract analysis

3.6.2 Query Processing Pipeline

Step 1: Query Embedding

- Model: all-MiniLM-L6-v2 (same as KGGen evaluation for consistency)
- Input: User's natural language question
- Output: 384-dimensional embedding vector

Step 2: Hybrid Retrieval Combine keyword and semantic search:

$$\text{score}(q, t) = 0.5 \times \text{BM25}(q, t) + 0.5 \times \text{cosine\_sim}(\text{embed}(q), \text{embed}(t)) \quad (1)$$

where:

- q = user query
- t = knowledge graph triple
- BM25 = keyword matching score
- cosine_sim = semantic similarity score
- **Top-K:** Retrieve 10 most relevant triples
- **Search space:** All nodes and edges in knowledge graph
- **Weighting:** Equal weight (0.5 BM25, 0.5 semantic) as validated in KGGen paper

Step 3: Subgraph Expansion Enable multi-hop reasoning by expanding to neighboring nodes:

- **Expansion depth:** 2-hop neighbors
- **Rationale:** Most legal queries require 1–2 relationship hops (e.g., Party → License → IPAsset → Restrictions)
- **Expansion size:** Add approximately 10 additional triples (total 20 triples)
- **Pruning:** Remove low-relevance nodes outside main subgraph to maintain focus

Step 4: Context Enrichment Augment knowledge graph triples with additional context:

- **Original contract text:** Retrieve original text chunks from source contracts for each triple
- **CUAD labels:** Include CUAD annotation category for each triple
- **Metadata:** Add contract type, jurisdiction, dates, parties
- **Statistics:** Include triple frequency ("This clause appears in 73% of similar contracts")

Step 5: Context Formatting Format retrieved information for LLM consumption:

```
% Caption: LLM Context Template]
# Contract Knowledge Graph Context

## Relevant Entities and Relationships

- (ABC Corp) --[licenses_to]--> (XYZ Inc)
  Properties: {type: exclusive, scope: worldwide,
               field: software development}
  Source: Contract ID ABC-2023-001, Section 2.1
  CUAD Label: License_Grant

- (ABC Corp) --[assigns]--> (Source Code)
  Properties: {assignment: full ownership,
               effective: upon final payment}
  Source: Contract ID ABC-2023-001, Section 3.2
```

```
  CUAD Label: IP_Ownership_Assignment

- (Source Code) --[subject_to_restriction]-->
    (No Open Source Usage)
  Properties: {scope: development,
               prohibition: GPL/copyleft licenses}
  Source: Contract ID ABC-2023-001, Section 3.4
  CUAD Label: License_Grant

## Original Contract Text

Section 2.1 License Grant: "ABC Corp hereby grants to
XYZ Inc an exclusive, worldwide license to use, modify,
and distribute the Licensed Software for purposes of
software development..."

Section 3.2 IP Assignment: "Upon receipt of final payment,
ABC Corp assigns all rights, title, and interest in the
Source Code to XYZ Inc..."

## Query

What IP rights does the licensee receive in the ABC Corp
software license agreement?

## Instructions

Answer the query based on the knowledge graph and contract
text. Be precise and cite specific clauses. If the
information is not in the provided context, state that
clearly.
```

Step 6: LLM Generation

- **Model:** Claude Sonnet 3.5 (primary) or GPT-4o
- **Temperature:** 0.0 (deterministic for legal analysis)
- **Max tokens:** 1000
- **System prompt:** "You are a legal contract analysis assistant. Provide accurate, precise answers based on the knowledge graph and contract text. Always cite specific clauses and explain your reasoning. If information is not in the context, say so explicitly."

3.6.3 API Endpoints

Table 10: REST API Endpoints

Endpoint	Method	Description
/api/v1/query	POST	Natural language Q&A with knowledge graph context
/api/v1/graph/search	POST	Search knowledge graph, return relevant subgraph
/api/v1/graph/entity/{id}	GET	Get entity details with all relationships
/api/v1/contracts/{id}/graph	GET	Get complete knowledge graph for specific contract

4 Ontology Specifications

4.1 Overview

The legal contract ontology defines the structure of entities and relationships extracted from CUAD technology agreements. The ontology design balances three competing requirements:

1. **Legal precision:** Capture legally significant distinctions and nuances
2. **Computational tractability:** Enable efficient graph operations and queries
3. **Common law alignment:** Respect contract interpretation principles from US/UK/Commonwealth jurisdictions

4.2 Entity Types (Nodes)

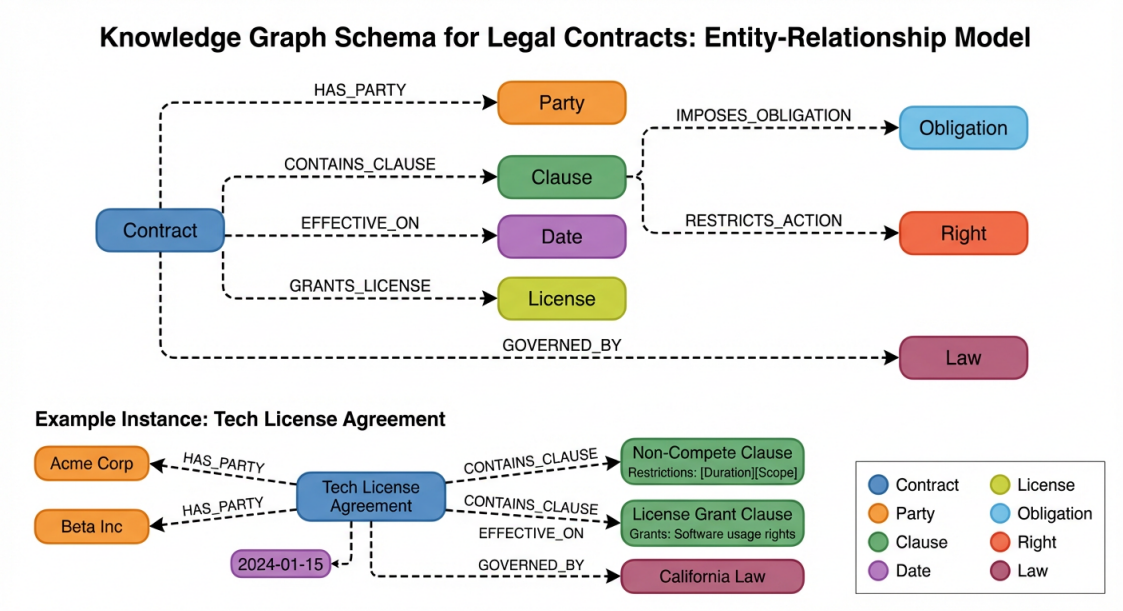

Figure 6: Contract Knowledge Graph Schema: Entity types and relationship structure

4.2.1 1. Party

Definition: Legal entities that are parties to the contract.

Attributes:

- `legal_name`: Official registered name
- `role`: licensor, licensee, vendor, customer, contractor, client, etc.
- `entity_type`: corporation, LLC, partnership, individual
- `jurisdiction`: State/country of incorporation
- `address`: Legal address

Examples:

- ABC Corporation (Licensor)
- XYZ Inc (Licensee)
- John Doe (Independent Contractor)

4.2.2 2. IPAsset

Definition: Intellectual property assets subject to contract terms.

Subtypes:

- Patent
- Copyright (Software, Documentation, Content)
- Trademark
- Trade Secret
- Source Code
- Know-How

Attributes:

- `ip_type`: patent, copyright, trademark, trade_secret, source_code
- `description`: Detailed description of IP asset
- `registration_number`: Patent/trademark number if registered
- `jurisdiction`: Country/region of IP protection

Examples:

- Source Code for Customer Relationship Management System
- US Patent 10,123,456 for Data Encryption Method
- ABC Software trademark

4.2.3 3. Obligation

Definition: Performance obligations or deliverables required under the contract.

Attributes:

- `description`: What must be provided/performed
- `deadline`: Due date or timeline
- `conditions`: Any conditions triggering obligation
- `modality`: SHALL (mandatory), SHOULD (recommended), MAY (optional)

Examples:

- Provide technical support within 24 hours
- Deliver source code within 30 days of final payment
- Maintain 99.9% uptime

4.2.4 4. Restriction

Definition: Prohibitions or limitations on party actions.

Subtypes:

- Non-compete
- Non-disclosure (confidentiality)
- Non-solicitation (employees/customers)
- Usage restrictions
- Transfer restrictions

Attributes:

- `restriction_type`: non_compete, non_disclosure, non_solicitation, usage, transfer
- `scope`: What is restricted
- `duration`: How long restriction applies
- `geography`: Geographic limitation if any
- `exceptions`: Any exceptions to restriction

Examples:

- Non-compete in same market for 2 years in California
- Confidentiality obligation for 5 years post-termination
- No reverse engineering of software

4.2.5 5. LiabilityProvision

Definition: Clauses allocating liability, providing indemnity, or limiting damages.

Subtypes:

- Liability cap
- Liability exclusion
- Indemnification
- Warranty
- Disclaimer

Attributes:

- `provision_type`: cap, exclusion, indemnity, warranty, disclaimer
- `amount`: Financial cap if applicable
- `scope`: What is covered (direct damages, consequential damages, etc.)
- `exceptions`: Exclusions from cap (IP infringement, willful misconduct, etc.)

Examples:

- Liability capped at $1,000,000 for direct damages
- No liability for consequential damages
- Indemnification for third-party IP claims

4.2.6 6. Temporal

Definition: Time-related terms including dates, durations, and deadlines.

Attributes:

- `temporal_type`: effective_date, expiration_date, renewal_date, notice_period, deadline
- `date`: Specific date (ISO 8601 format)
- `duration`: Time period (e.g., "2 years", "30 days")
- `trigger`: Event triggering temporal element

Examples:

- Effective Date: January 1, 2024
- Expiration Date: December 31, 2026
- Renewal Term: 1 year, automatic unless 60 days notice

4.2.7 7. Jurisdiction

Definition: Governing law and venue provisions.

Attributes:

- `governing_law`: State/country law that governs contract
- `legal_system`: common_law, civil_law
- `venue`: Court location for disputes
- `arbitration`: Whether arbitration required

Examples:

- Governed by laws of California, US (common law system)
- Venue: Santa Clara County Superior Court
- Arbitration in San Francisco under AAA rules

4.2.8 8. ContractClause

Definition: Named contract provisions or sections.

Attributes:

- `clause_name`: Title of clause
- `clause_number`: Section number
- `cuad_label`: CUAD category
- `text`: Full text of clause

Examples:

- Section 2.1: License Grant
- Section 8.3: Limitation of Liability
- Section 12: Governing Law

4.3 Relationship Types (Edges)

4.3.1 1. LICENSES_TO

Definition: Grant of license from licensor to licensee.

Domain: Party → Party or Party → IPAsset

Properties:

- `license_type`: exclusive, non-exclusive, sole
- `scope`: worldwide, limited territory, specific use
- `transferable`: true/false
- `sublicensable`: true/false
- `field_of_use`: Permitted uses

Example: (ABC Corp) –[LICENSES_TO: {type: exclusive, scope: worldwide}]–> (XYZ Inc)

4.3.2 2. OWNS

Definition: Ownership of intellectual property or assets.

Domain: Party → IPAsset

Properties:

- `ownership_type`: full, joint, partial
- `rights_included`: reproduction, distribution, modification, etc.

Example: (ABC Corp) –[OWNS: {type: full}]–> (Source Code)

4.3.3 3. ASSIGNS

Definition: Transfer of ownership or rights from one party to another.

Domain: Party → IPAsset or Party → Contract

Properties:

- `assignment_type`: full, partial, conditional
- `effective_trigger`: upon_payment, upon_delivery, immediately
- `consideration`: Payment or other consideration

Example: (Contractor) –[ASSIGNS: {type: full, effective: upon_final_payment}]–> (Work Product)

4.3.4 4. HAS_OBLIGATION

Definition: Party has obligation to perform or deliver.

Domain: Party → Obligation

Properties:

- `modality`: SHALL, SHOULD, MAY
- `deadline`: Due date
- `conditions`: Triggering conditions

Example: (Vendor) –[HAS_OBLIGATION: {modality: SHALL, deadline: 2024-03-01}]–> (Deliver Software)

4.3.5 5. SUBJECT_TO_RESTRICTION

Definition: Party is subject to prohibition or limitation.

Domain: Party → Restriction

Properties:

- `duration`: Time period
- `geography`: Geographic scope
- `scope`: What is restricted

Example: (Employee) –[SUBJECT_TO_RESTRICTION: {duration: 2_years, geography: California}]–> (Non-Compete)

4.3.6 6. HAS_LIABILITY

Definition: Party has liability under specified provision.

Domain: Party → LiabilityProvision

Properties:

- `cap_amount`: Financial limit if any
- `scope`: direct_damages, indirect_damages, all_damages
- `exceptions`: Exclusions from cap

Example: (Vendor) –[HAS_LIABILITY: {cap: $1000000, scope: direct_damages, exceptions: [IP_infringement]}]–> (Liability Cap)

4.3.7 7. GOVERNED_BY

Definition: Contract is governed by specified jurisdiction's law.

Domain: Contract → Jurisdiction

Properties:

- `legal_system`: common_law, civil_law
- `venue`: Court location
- `conflict_of_laws`: Choice of law rules

Example: (License Agreement) –[GOVERNED_BY: {system: common_law, venue: Santa Clara County}]–> (California)

4.3.8 8. EFFECTIVE_ON

Definition: Contract or provision becomes effective on specified date.

Domain: Contract → Temporal or ContractClause → Temporal

Properties:

- `date`: Effective date
- `trigger`: Triggering event if conditional

Example: (Agreement) –[EFFECTIVE_ON: {date: 2024-01-01}]–> (Effective Date)

4.3.9 9. TERMINATES_ON

Definition: Contract or provision terminates on specified date or condition.

Domain: Contract → Temporal

Properties:

- `date`: Expiration date if fixed
- `trigger`: Termination trigger (breach, convenience, etc.)
- `notice_period`: Required notice for termination

Example: (Agreement) –[TERMINATES_ON: {date: 2026-12-31, notice: 60_days}]–> (Expiration Date)

4.3.10 10. CONTAINS_CLAUSE

Definition: Contract contains specific clause or provision.

Domain: Contract → ContractClause

Properties:

- `clause_number`: Section number
- `cuad_label`: CUAD category

Example: (License Agreement) –[CONTAINS_CLAUSE: {number: "2.1", cuad_label: "License_Grant"}]–> (License Grant Clause)

4.4 Common Law Alignment

4.4.1 Key Principles Embedded in Ontology

1. **Plain Meaning Rule:** Canonical entity names use clear, plain legal terminology as understood by reasonable legal practitioners
2. **Business Efficacy:** Relations model business relationships (licenses, obligations, assignments) rather than just textual patterns
3. **Contextual Interpretation:** Extraction uses full contract context, not isolated clauses, to determine entity and relation semantics

4. **Defined Terms:** Preserve contract-specific defined terms (capitalized terms) as distinct entities when legally significant

5. **Legal Modality:** Distinguish SHALL (mandatory) vs MAY (optional) vs SHOULD (recommended) in obligations

4.4.2 Jurisdictional Tagging

All contracts are tagged with jurisdiction metadata:

- `governing_law`: California, Delaware, New York, UK, etc.
- `legal_system`: common_law or civil_law
- `jurisdiction_notes`: Special legal considerations

This enables jurisdiction-specific analysis and prevents inappropriate cross-jurisdiction reasoning.

5 Technical Implementation Plan

5.1 Development Roadmap

The implementation follows a four-phase approach spanning 16–20 weeks from prototype to production deployment.

5.2 Phase 1: Prototype (4–6 weeks)

5.2.1 Goals

- Validate KGGen methodology on CUAD dataset
- Build minimal viable pipeline (Extraction → Aggregation → Resolution)
- Demonstrate value with sample use cases
- Establish quality baselines

5.2.2 Week-by-Week Plan

Week 1: Environment Setup and Data Exploration

- Set up Python 3.11+ development environment
- Install dependencies: DSPy, anthropic, openai, sentence-transformers, networkx, pandas
- Download CUAD dataset (510 contracts + annotations)
- Explore contract structure and CUAD label distribution
- Extract text from 10 sample contracts

Week 2: Stage 1 Extraction Implementation

- Implement two-step extraction pipeline (entities → relations)
- Configure Claude Sonnet 3.5 API with DSPy
- Create extraction prompts with legal constraints
- Process 10 sample contracts
- Validate output against CUAD annotations

Week 3: Stage 2 & 3 Implementation

- Implement aggregation logic (normalization, deduplication)
- Implement basic resolution (clustering + LLM canonicalization)
- Process 10-contract knowledge graph through full pipeline
- Store in NetworkX + JSON for prototyping

Week 4: Query Interface Development

- Implement hybrid retrieval (BM25 + semantic search)
- Build subgraph expansion logic (2-hop neighbors)
- Create LLM context formatting
- Develop simple CLI for contract Q&A
- Test on 10 sample queries

Week 5: Evaluation and Quality Assessment

- Manual validation: Review 100 random triples for validity (target 90%+)
- MINE-1 evaluation: Run on subset of contracts
- Accuracy assessment: Compare entity extraction against CUAD annotations
- Document quality metrics and identify failure modes

Week 6: Iteration and Refinement

- Refine extraction prompts based on errors
- Improve resolution logic for legal term handling
- Expand to 50 contracts
- Prepare demonstration for stakeholders

5.2.3 Phase 1 Deliverables

- Working extraction pipeline processing CUAD contracts
- Knowledge graph with 50+ contracts (5,000–10,000 triples)
- Basic query interface (CLI or simple web UI)
- Quality metrics report: triple validity, MINE-1 score, extraction accuracy
- Technical documentation

5.2.4 Success Criteria

- Extract knowledge graphs from 50+ contracts successfully
- Achieve 90%+ triple validity
- Answer 10 sample queries correctly with proper citations
- Demonstrate value to stakeholders (engineers + lawyers)

5.3 Phase 2: Scale and Optimize (6–8 weeks)

5.3.1 Goals

- Process full CUAD dataset (510 contracts)
- Optimize performance and reduce costs
- Enhance extraction quality to 98% target
- Build production-grade REST API

5.3.2 Milestones

Weeks 1–2: Full Dataset Processing

- Scale extraction to all 510 CUAD contracts
- Implement parallel processing for throughput
- Handle edge cases and extraction failures gracefully
- Estimated output: 50,000–100,000 triples

Weeks 3–4: Resolution Optimization

- Optimize clustering algorithm (parallelization, caching)
- Batch LLM API calls to reduce latency
- Implement incremental resolution for new contracts
- Target: Resolve 50K–100K triples in under 1 hour

Week 5: Advanced Query Features

- Implement filters (by contract type, jurisdiction, date range)
- Add aggregation queries ("Show all non-compete durations")
- Support comparison queries ("Compare license terms across 50 contracts")
- Enhance context enrichment with statistics

Week 6: REST API Development

- Build FastAPI-based REST API
- Implement 4 core endpoints: /query, /graph/search, /graph/entity, /contracts/{id}/graph
- Add authentication (API key-based)
- Implement rate limiting and caching

Week 7: Comprehensive Evaluation

- Manual validation: 100 random triples, target 98% validity
- MINE-1 evaluation on full dataset, target 65%+
- MINE-2 evaluation (multi-hop reasoning)
- Query accuracy: 50 test queries evaluated by lawyers
- Performance benchmarks: extraction throughput, query latency

Week 8: Bug Fixes and Performance Tuning

- Address issues identified in evaluation
- Optimize query latency (target <500ms)
- Reduce LLM API costs through caching and batching
- Prepare for production deployment

5.3.3 Phase 2 Deliverables

- Complete knowledge graph: 510 contracts, 50K–100K triples
- Production-grade REST API with documentation
- Comprehensive evaluation report with quality metrics
- Performance benchmarks and optimization recommendations
- Deployment-ready codebase

5.3.4 Success Criteria

- Process all 510 contracts successfully (100% coverage)
- Achieve 98% triple validity (validated on random sample)
- MINE-1 score 65%+ (information retention)
- Query latency <500ms for 90% of queries
- API throughput 100+ requests per second

5.4 Phase 3: Product Integration (4–6 weeks)

5.4.1 Goals

- Integrate with user-facing application
- Develop 5 target use cases
- User testing with legal practitioners
- Production deployment with monitoring

5.4.2 Milestones

Week 1: User Interface Design

- Design web application for contract analysis
- Create mockups for Q&A, risk dashboard, comparison views
- User flow design for common tasks
- Collaborate with lawyers on UI requirements

Week 2: Contract Q&A Feature

- Implement natural language query interface
- Display answers with knowledge graph visualizations
- Show source citations with contract links
- Add confidence scores and explanations

Week 3: Risk Analysis Dashboard

- Implement risk detection queries (uncapped liability, long non-competes, etc.)
- Create visualization of risks across portfolio
- Generate risk reports with recommendations

Week 4: Comparison and Compliance Features

- Implement side-by-side contract comparison
- Build compliance verification against policies
- Create export functionality (PDF reports, CSV data)

Week 5: User Testing

- Recruit 5–10 legal practitioners for user testing
- Collect feedback on accuracy, usability, value
- Measure time savings on test contract review tasks
- Iterate based on feedback

Week 6: Production Deployment

- Deploy to production infrastructure (cloud VM or Kubernetes)
- Set up Neo4j production database
- Configure monitoring (Prometheus + Grafana)
- Implement alerting for critical failures
- Create deployment documentation

5.4.3 Phase 3 Deliverables

- Web application for contract analysis
- User documentation and video tutorials
- Deployment guide for production infrastructure
- Monitoring and alerting system
- User testing report with feedback

5.4.4 Success Criteria

- Positive user feedback from legal practitioners (4/5+ satisfaction)
- Demonstrate 50% time savings in contract review tasks
- Successful production deployment
- 99.9% uptime in first month of operation

5.5 Phase 4: Enhancement and Expansion (Ongoing)

5.5.1 Areas for Continuous Improvement

1. **Ontology Expansion**
   - Add more contract types beyond technology focus
   - Expand clause categories beyond 41 CUAD labels
   - Include financial terms, payment schedules
2. **Common Law Reasoning**
   - Incorporate case law and legal precedents
   - Link contract clauses to relevant court decisions
   - Enable reasoning about legal interpretations
3. **Multi-Jurisdiction Support**
   - Extend beyond US to UK, Canada, Australia
   - Add civil law systems (EU, Latin America)
   - Jurisdiction-specific interpretation rules
4. **Advanced Analytics**
   - Trend analysis: How contract terms evolve over time
   - Risk scoring: Quantitative risk assessment
   - Negotiation insights: Market norms and outliers
5. **System Integration**
   - Connect to document management systems (SharePoint, Box)

- Integrate with CRMs (Salesforce, HubSpot)
- API integration with legal research tools

6. **Model Improvement**
   - Fine-tune LLMs on legal contract corpus
   - Train custom entity extraction models
   - Develop contract-specific embeddings
7. **User Feedback Loop**
   - Collect corrections from lawyers
   - Active learning to improve extraction
   - Continuous quality monitoring

5.6 Technology Stack

Table 11: Complete Technology Stack

Component	**Technology**	**Rationale**
Programming Language	Python 3.11+	Ecosystem for NLP, ML, graph processing
LLM Orchestration	DSPy	Structured outputs, multi-model support
LLM Provider (Primary)	Claude Sonnet 3.5	73% MINE-1 score, best for legal reasoning
LLM Provider (Fallback)	GPT-4o	66% MINE-1, reliable fallback
Embedding Model	all-MiniLM-L6-v2	Fast, good quality, matches KGGen benchmark
Clustering	S-BERT + K-means	Effective semantic clustering
Graph Database (Prod)	Neo4j 5.x	Industry standard, Cypher, visualization
Graph Database (Dev)	NetworkX + PostgreSQL	Python native, prototyping
Search Engine	rank-bm25 + FAISS	Hybrid keyword + semantic search
Web Framework	FastAPI	Async support, auto-documentation, modern
Background Jobs	Celery + Redis	Long-running extraction tasks
Monitoring	Prometheus + Grafana	Metrics, alerting, visualization
Deployment	Docker + Kubernetes	Containerization, orchestration, scaling

5.7 Infrastructure Requirements

5.7.1 Compute

- **Extraction:** GPU optional (speeds up large-scale processing), CPU sufficient for small batches
- **Resolution:** CPU-intensive (clustering), can parallelize across cores
- **Query:** Low latency required (<500ms), CPU sufficient

5.7.2 Storage

- **Contracts:** 500 MB (S3 or local filesystem)
- **Knowledge graph:** 1–2 GB (Neo4j database with 50K–100K triples)
- **Embeddings:** 500 MB (FAISS index, in-memory or mmap)
- **Cache:** Redis for query caching (1–2 GB RAM)
- **Total:** 3–4 GB for complete system

5.7.3 Deployment Options

1. **Development:** Local machine (Mac/Linux with 16 GB RAM)
2. **Staging:** Single cloud VM (AWS EC2 t3.xlarge: 4 vCPU, 16 GB RAM)
3. **Production:** Kubernetes cluster with horizontal scaling
    - API pods: 2–4 replicas
    - Worker pods (extraction/resolution): 4–8 replicas
    - Neo4j: 1 instance (4–8 GB RAM)
    - Redis: 1 instance (2 GB RAM)

6 Success Metrics & Risk Analysis

6.1 Success Metrics

Success is measured across three dimensions: technical performance, business value, and legal accuracy.

6.1.1 Technical Metrics

Table 12: Technical Performance Metrics

Metric	Target	Measurement Method
Triple validity	98%	Manual review of 100 random triples by lawyer
MINE-1 score	65%+	Run KGGen MINE-1 benchmark on sample contracts
Entity extraction accuracy	95%+	Compare against CUAD expert annotations
Query latency (simple)	<500ms	P95 latency for single-hop queries
Query latency (complex)	<2s	P95 latency for multi-hop queries
Extraction throughput	100+ contracts/hour	Avg processing rate with parallelization
Graph density improvement	80% reduction in unique relations	Before/after resolution comparison
System uptime	99.9%	Production monitoring over 30 days

6.1.2 Business Metrics

Table 13: Business Value Metrics

Metric	Target	Measurement Method
Time savings	50% reduction	User studies: time to complete contract review tasks
Query accuracy	90%+ correct answers	Human evaluation of 50 sample queries
User satisfaction	4/5+ rating	Post-task user surveys with legal practitioners
Contract coverage	510 contracts	System logs: successful processing
Use case demonstration	All 5 use cases	Case studies for Q&A, risk, comparison, compliance, due diligence
Cost per contract	<$1	LLM API costs + infrastructure costs

6.1.3 Legal Metrics

Table 14: Legal Accuracy Metrics

Metric	Target	Measurement Method
Lawyer validation	95%+ agreement	Senior lawyer reviews 100 system outputs
Critical error rate	<2%	Track critical errors (incorrect legal interpretations)
Compliance	100%	Audit against data protection regulations (GDPR, CCPA)
Provenance accuracy	100%	Verify all triples link back to correct source contract/clause

6.2 Risk Analysis and Mitigation

6.2.1 Technical Risks

Table 15: Technical Risk Matrix

Risk	Impact	Likelihood	Mitigation Strategies
LLM hallucination (incorrect extractions)	High	Medium	Use Claude Sonnet 3.5 (73% MINE-1); strong prompt constraints; confidence scores; human review of low-confidence triples; continuous evaluation
Resolution errors (merging distinct entities)	High	Medium	Conservative resolution approach; LLM prompted with legal constraints; manual review of samples; lawyer validation; maintain aliases
Scalability bottlenecks	Medium	Low	Parallel processing; incremental updates; query caching; efficient graph database; optimize resolution algorithm
High LLM API costs	Medium	Medium	Batch processing; cache LLM outputs; use cheaper models for non-critical tasks; optimize prompts; consider self-hosted models
Graph database performance degradation	Medium	Low	Use Neo4j with proper indexing; optimize Cypher queries; implement caching layer; horizontal scaling if needed

6.2.2 Legal Risks

Table 16: Legal Risk Matrix

Risk	Impact	Likelihood	Mitigation Strategies
Incorrect legal advice	Critical	Medium	Clear disclaimer: assistive tool, not legal advice; lawyer review of outputs; confidence scores; explain reasoning with citations; E&O insurance
Confidentiality breach	Critical	Low	Secure storage with encryption; access control and authentication; audit logs; compliance with GDPR/CCPA; self-hosted option for sensitive contracts
Jurisdictional limitations	Medium	High	Clear scope: focus on US/UK/Commonwealth common law; jurisdiction tagging; future expansion to civil law; partner with local legal experts
Unauthorized practice of law	High	Low	Position as software tool for legal professionals, not legal service; require lawyer oversight; do not provide legal advice or recommendations

6.2.3 Business Risks

Table 17: Business Risk Matrix

Risk	Impact	Likelihood	Mitigation Strategies
Low user adoption	High	Medium	Involve lawyers in design and testing; emphasize augmentation not replacement; transparent, explainable system; strong accuracy and reliability; education and training materials
Market competition	Medium	High	Unique value: structured knowledge graph vs text search; common law specialization; technology contract focus; open source components; partnerships with law firms
Regulatory changes	Medium	Low	Monitor legal tech regulations; maintain compliance documentation; flexible architecture for adaptation; legal counsel consultation
Data quality issues (CUAD)	Medium	Low	Validate CUAD annotations during extraction; identify and flag inconsistencies; human review of problematic contracts; augment with additional datasets

6.3 Quality Assurance Process

6.3.1 Testing Strategy

1. **Unit Tests**
   - Target: 90%+ code coverage
   - Test extraction logic, normalization, resolution algorithms
   - Use pytest framework
2. **Integration Tests**
   - End-to-end pipeline testing
   - Test with sample contracts through full workflow
   - Validate API endpoints
3. **Validation Tests**
   - Triple validity: Manual review of 100 random triples, target 98%
   - MINE-1 evaluation: Run KGGen benchmark, target 65%+
   - Entity extraction: Compare against CUAD annotations, target 95%+
   - Resolution quality: Manual review of entity clusters, target 95% correct
4. **User Acceptance Testing**
   - Recruit 5–10 legal practitioners
   - Test on real contract review tasks
   - Collect qualitative and quantitative feedback

6.3.2 Continuous Monitoring

Production system monitoring includes:

- **Error tracking:** Extraction failures, LLM API errors, database errors
- **Performance metrics:** Query latency (P50, P95, P99), throughput (queries/second)
- **Quality metrics:** User feedback, correction rate, confidence scores
- **Cost metrics:** LLM API costs per contract, infrastructure costs
- **Usage metrics:** Active users, queries per user, most common query types
- **Alerting:** Slack/email alerts for critical failures, performance degradation, anomalies

6.4 Ethical Considerations

6.4.1 Transparency

- All system outputs include source citations linking back to contract text
- Confidence scores displayed for LLM-generated content
- Knowledge graph provenance: every triple traceable to source
- Clear disclosure that system uses AI/LLM technology

6.4.2 Bias and Fairness

- CUAD dataset represents real-world contracts (potential bias toward larger companies)
- Monitor for systematic errors favoring one party type over another
- Validate across diverse contract types and jurisdictions
- Human lawyer oversight to catch biased interpretations

6.4.3 Data Privacy

- CUAD contracts are publicly available (no confidentiality issues)
- User contracts must be encrypted at rest and in transit
- Access control: users can only access their own contracts
- Audit logs: track all data access for compliance
- Self-hosted deployment option for sensitive contracts

6.4.4 Professional Responsibility

- System is assistive tool for legal professionals, not replacement
- Lawyers retain responsibility for final decisions
- Encourage lawyer review and validation of outputs
- Training materials emphasize appropriate use and limitations

7 Conclusion

7.1 Summary

This Product Requirements Document specifies a comprehensive knowledge graph extraction system that applies the KGGen methodology to the CUAD contract dataset. The system addresses critical gaps in legal contract analysis by:

1. **Structured Knowledge Representation:** Converting 510 unstructured contracts into queryable knowledge graphs containing 50,000–100,000 verified triples
2. **Context-Aware LLM Integration:** Providing Large Language Models with precise contractual relationships rather than raw text, enabling accurate multi-hop reasoning
3. **Common Law Alignment:** Embedding contract interpretation principles from US/UK/Commonwealth jurisdictions throughout extraction and resolution logic
4. **Technology Contract Specialization:** Focusing on IP licensing, software development, and SaaS agreements with specialized ontology
5. **High Accuracy:** Targeting 98% triple validity and 65%+ MINE-1 information retention scores

7.2 Value Proposition

The CUAD Knowledge Graph Generator enables:

- **50% time reduction** in contract review cycles
- **Automated analysis** of 41+ clause categories without manual reading
- **Proactive risk identification** across entire contract portfolios
- **Democratized legal access** for small businesses and individuals
- **Engineer-lawyer collaboration** through structured, explainable AI

7.3 Path to Production

The four-phase implementation roadmap provides clear path from prototype to production:

1. **Phase 1 (4–6 weeks):** Validate methodology with 50-contract prototype
2. **Phase 2 (6–8 weeks):** Scale to full 510-contract dataset with optimization
3. **Phase 3 (4–6 weeks):** Build user-facing application and deploy to production
4. **Phase 4 (Ongoing):** Expand ontology, add jurisdictions, improve models

Total timeline: 16–20 weeks to production deployment.

7.4 Competitive Advantages

This system differentiates from existing legal AI tools through:

1. **Knowledge graph foundation:** Structured relationships vs probabilistic text retrieval
2. **Multi-hop reasoning:** Answer complex queries spanning multiple relationship types
3. **Common law expertise:** Designed by and for legal practitioners in common law jurisdictions
4. **Technology focus:** Specialized for IP, software, and technology agreements
5. **Explainability:** Transparent reasoning with source citations and provenance tracking

7.5 Next Steps

Immediate Actions:

1. Secure approval from stakeholders (engineering team + senior lawyer)
2. Allocate resources: 2 engineers + 1 lawyer (part-time) for Phase 1
3. Set up development environment and access to LLM APIs
4. Download CUAD dataset and begin Phase 1 implementation

Success Criteria for Approval:

- Technical feasibility validated through Phase 1 prototype
- Legal accuracy confirmed by lawyer review (95%+ agreement)
- User value demonstrated through time savings measurement
- Cost model validated (target <$1 per contract processing cost)

7.6 Contact Information

For questions, clarifications, or feedback on this PRD:

- **Technical Lead:** K-Dense Web (contact@k-dense.ai)
- **Documentation:** k-dense.ai

This document was generated using K-Dense Web to support the development of next-generation legal AI systems.

k-dense.ai

A CUAD Label Categories

The CUAD dataset includes 13,101 expert annotations across 41 label categories:

1. Document Name
2. Parties
3. Agreement Date
4. Effective Date
5. Expiration Date
6. Renewal Term
7. Notice Period To Terminate Renewal
8. Governing Law
9. Most Favored Nation
10. Non-Compete
11. Exclusivity
12. No-Solicit Of Customers
13. No-Solicit Of Employees
14. Non-Disparagement
15. Termination For Convenience
16. ROFR/ROFO/ROFN
17. Change Of Control
18. Anti-Assignment
19. Revenue/Profit Sharing
20. Cap On Liability
21. Uncapped Liability
22. Liquidated Damages
23. Warranty Duration
24. Insurance
25. Covenant Not To Sue
26. Third Party Beneficiary
27. Irrevocable Or Perpetual License
28. Source Code Escrow
29. Post-Termination Services
30. Audit Rights
31. Volume Restriction
32. IP Ownership Assignment
33. Joint IP Ownership
34. License Grant
35. Non-Transferable License
36. Affiliate License-Licensor
37. Affiliate License-Licensee
38. Unlimited/All-You-Can-Eat-License
39. Minimum Commitment
40. Competitive Restriction Exception
41. Price Restrictions

B Common Law Contract Interpretation Principles

B.1 Key Principles Guiding Ontology Design

1. **Plain Meaning Rule (Literal Interpretation)**
   - Terms interpreted by their ordinary, plain meaning as understood by reasonable person
   - **Application:** Canonical entity names use clear, plain legal terminology
2. **Contra Proferentem (Against the Drafter)**

- Ambiguous terms construed against party who drafted the contract
- **Application:** Flag ambiguous terms in knowledge graph for human review

3. **Business Efficacy Test**
   - Interpret contract to give commercial effect and make business sense
   - **Application:** Relations model business relationships, not just textual patterns
4. **Entire Agreement Clause**
   - Contract is complete expression of parties' agreement, excludes external evidence
   - **Application:** Extract from full contract text, don't rely on external sources
5. **Contextual Interpretation (Purposive Approach)**
   - Consider contract as whole, commercial context, and parties' intentions
   - **Application:** Use full contract context for extraction, not isolated clauses

C Example Queries and Expected Outputs

C.1 Query 1: IP Rights Inquiry

Question: "What IP rights does the licensee receive in the software license agreement?"

Relevant Triples:

- (ABC Corp, licenses_to, XYZ Inc)
- (License, is_non_exclusive, True)
- (License, scope, worldwide)
- (License, includes_right_to, use)
- (License, includes_right_to, modify)
- (XYZ Inc, cannot_transfer_license_to, third_party)

Answer: "The licensee (XYZ Inc) receives a non-exclusive, worldwide license to use and modify the software. However, the license is non-transferable and cannot be sublicensed to third parties. [Citation: Contract ID ABC-2023-001, Section 2.1]"

C.2 Query 2: Non-Compete Restriction

Question: "Are there any non-compete restrictions, and if so, what is the scope?"

Relevant Triples:

- (ABC Corp, subject_to_restriction, Non-Compete)
- (Non-Compete, scope, same_market)
- (Non-Compete, duration, 2_years)
- (Non-Compete, geography, California)
- (Non-Compete, effective_from, termination_date)

Answer: "Yes, ABC Corp is subject to a non-compete restriction. The restriction prohibits competing in the same market within California for 2 years following termination of the agreement. [Citation: Contract ID ABC-2023-001, Section 7.2]"

C.3 Query 3: Liability Cap

Question: "What is the liability cap in this agreement?"

Relevant Triples:

- (Agreement, has_liability_provision, Cap On Liability)
- (Cap On Liability, amount, $1,000,000)
- (Cap On Liability, scope, direct_damages)
- (Cap On Liability, exceptions, [IP_infringement, willful_misconduct])

Answer: "The agreement includes a liability cap of $1,000,000 for direct damages. However, this cap does not apply to IP infringement claims or willful misconduct, which remain uncapped. [Citation: Contract ID ABC-2023-001, Section 8.3]"

D Technology Stack Details

D.1 LLM Providers

Table 18: LLM Model Comparison

Model	Version	MINE-1	Use Case
Claude Sonnet 3.5	20241022	73%	Primary extraction
GPT-4o	20241120	66%	Fallback extraction
Gemini 2.0 Flash	Latest	44%	Not recommended

D.2 Python Libraries

Core Dependencies:

- `dspy-ai>=2.0`: LLM orchestration

- `anthropic`: Claude API client
- `openai`: GPT-4o API client
- `sentence-transformers`: Embeddings
- `neo4j`: Graph database driver
- `networkx`: Graph algorithms
- `rank-bm25`: BM25 search
- `faiss-cpu`: Vector similarity search
- `fastapi`: Web framework
- `pdfplumber`: PDF text extraction
- `pydantic`: Data validation
- `pytest`: Testing framework