Product Requirements Document: KGGEN-based Knowledge Graph System for Legal Contract Analysis

K-Dense Web
contact@k-dense.ai

January 12, 2026

Abstract

This Product Requirements Document specifies a knowledge graph-based legal contract analysis system that applies the KGGEN methodology to the CUAD dataset for technology agreements. The system extracts structured knowledge from legal contracts using a three-stage pipeline (entity extraction, aggregation, entity resolution), maps clauses to 41 CUAD label categories, and provides LLM-powered semantic retrieval for contract analysis. Targeting legal professionals and engineering teams, the system aims to reduce contract review time and costs while maintaining high accuracy through expert-validated extraction and human-in-the-loop workflows. Key innovations include hybrid BM25 and semantic retrieval, multi-hop graph traversal for context assembly, and a technology-focused ontology covering software licensing, IP ownership, and restrictive covenants common in technology agreements.

Contents

1 Executive Summary

1.1 Problem Statement

Legal contract review costs law firms hundreds of thousands of dollars per transaction, with billing rates of $500-$900 per hour and approximately 50% of lawyer time spent on contract analysis [2]. The repetitive task of finding relevant clauses in lengthy contracts—determining termination dates, identifying anti-assignment provisions, locating liability caps—consumes enormous resources yet requires limited expert judgment. Small businesses and individuals often cannot afford professional review, leading to predatory contract terms and consumer harm [5].

1.2 Proposed Solution

We propose a knowledge graph-based contract analysis system that applies KGGen [8], a state-of-the-art text-to-knowledge-graph extractor, to the CUAD dataset [5], the largest expert-annotated legal contract review benchmark. Our system:

1. **Extracts** structured knowledge from contracts using LLM-based entity and relation identification
2. **Organizes** contract information into a knowledge graph following CUAD's 41-category ontology
3. **Retrieves** relevant context using hybrid BM25 and semantic search with multi-hop graph traversal
4. **Synthesizes** natural language analysis using LLMs grounded in extracted knowledge graph facts

The initial phase focuses on **technology agreements**, addressing clauses particularly relevant to software licensing, IP ownership, non-compete restrictions, and liability provisions.

1.3 Key Benefits

For Legal Professionals:

- Reduce contract review time by 50-70% through automated clause extraction
- Increase review accuracy with structured knowledge graph representations
- Enable semantic search across contract portfolios ("Show all perpetual licenses")
- Identify risks and obligations without reading entire contracts

For Organizations:

- Lower transaction costs from hundreds of thousands to tens of thousands of dollars
- Democratize access to legal support for small businesses and individuals
- Standardize contract analysis across teams
- Build institutional knowledge in reusable knowledge graph format

For Engineering Teams:

- Leverage state-of-the-art KGGEN pipeline achieving 66% accuracy [8]
- Build on CUAD's $2 million expert-annotated dataset [5]
- Deploy proven NLP techniques (BERT, RoBERTa, DeBERTa) for specialized domains

1.4 Success Criteria

Table 1: Success metrics for KGGEN-CUAD system

Metric	Target	Measurement
Extraction Accuracy	¿85%	CUAD benchmark AUPR
Clause Recall	¿90%	Critical clause identification
Query Response Time	¡3 seconds	95th percentile latency
Contract Processing	¡2 minutes	Per 50-page contract
Time Savings	¿50%	vs. manual review baseline
User Satisfaction	¿4.0/5.0	Lawyer feedback surveys

2 Introduction

2.1 Background and Motivation

Legal contract review represents one of the most time-consuming and costly processes in modern legal practice. Law firms spend approximately 50% of their time reviewing contracts, with billing rates typically ranging from $500 to $900 per hour in the United States [2]. This translates to hundreds of thousands of dollars in transaction costs for companies seeking to verify that contracts contain no problematic obligations or requirements. The high cost of contract review creates significant barriers: small businesses and individuals often sign contracts without reading them, leading to predatory behavior and consumer harm [5].

Contract review work falls into two categories: (1) **contract analysis**, which involves finding "needles in a haystack" by manually reviewing hundreds of pages to identify relevant clauses such as termination dates, anti-assignment provisions, or exclusivity terms; and (2) **counseling**, which requires experienced lawyers to assess risk and advise on business implications. While the latter demands expert judgment, the former represents repetitive, tedious work that is amenable to automation [5].

Despite recent advances in natural language processing (NLP), particularly with large Transformer models [3,4,7], many specialized domains including legal contract review remain largely untouched by deep learning. The primary bottleneck is the lack of large, expertly-annotated datasets in specialized domains. Creating such datasets requires expensive expert annotators who are often short on time and command high prices [5].

2.2 KGGEN Methodology

KGGen [8] addresses the fundamental challenge of knowledge graph (KG) scarcity by leveraging language models to extract high-quality graphs from plain text. Unlike existing KG generators such as OpenIE [1] and Microsoft's GraphRAG [6], which lack effective mechanisms for entity resolution and relation normalization, KGGen employs a novel three-stage pipeline:

Stage 1: Entity and Relation Extraction. Using language models (specifically Google Gemini 2.0 Flash) with DSPy signatures, KGGen extracts subject-predicate-object triples from unstructured text. This two-step approach first identifies entities, then extracts relationships between them, ensuring consistency across the graph.

Stage 2: Aggregation. After extracting triples from each source text, KGGen collects unique entities and edges across all sources and combines them into a single graph. All entities and edges are normalized to lowercase to reduce redundancy without requiring LLM processing.

Stage 3: Entity and Edge Resolution. The key innovation in KGGen is its iterative clustering algorithm that merges nodes and edges representing the same real-world entities or concepts. Using S-BERT embeddings, k-means clustering (128-item clusters), and a hybrid BM25 and semantic similarity approach (top-16 retrieval), KGGen identifies duplicates and selects canonical representatives. This significantly reduces the sparsity problem that plagues existing extractors, where nearly as many unique relation types exist as edges [8].

KGGen achieves 66.07% accuracy on the MINE-1 benchmark (compared to 47.80% for GraphRAG and 29.84% for OpenIE) while producing more concise and generalizable entities and relations. On large-scale extraction tasks (1M characters), KGGen reuses each relation type an average of 10 times, while GraphRAG reuses each type only 2 times regardless of corpus size. Additionally, KGGen demonstrates superior scaling properties: it processes 1M characters in 551 seconds total versus GraphRAG's 2,319 seconds for extraction alone [8].

2.3 CUAD Dataset

The Contract Understanding Atticus Dataset (CUAD) [5] represents the first large-scale, expert-annotated dataset for legal contract review. Created by The Atticus Project with dozens of legal experts, CUAD consists of:

- **510 contracts** spanning 25 different contract types (license agreements, service agreements, joint ventures, etc.)
- **13,101 labeled clauses** across 41 label categories
- **Expert annotations** by law students who underwent 70-100 hours of training, verified by experienced attorneys through multiple review rounds
- **Conservative pecuniary value** exceeding $2 million (9,283 pages reviewed at least 4 times, 5-10 minutes per page, at $500/hour)

The 41 label categories are organized into three groups: (1) **General Information** (parties, dates, governing law, license grants), (2) **Restrictive Covenants** (non-compete, exclusivity, anti-assignment, no-solicit clauses), and (3) **Revenue Risks** (liability caps, minimum commitments, audit rights, termination provisions) [5].

Contracts in CUAD range from a few pages to over 100 pages, with an average of 18.2 pages. The dataset covers 25 contract types including Software License Agreements, Service Agreements, Joint Venture Agreements, Distribution Agreements, and Technology Transfer Agreements. Labeled clauses make up approximately 10% of each contract on average, meaning only about 0.25% of each contract is highlighted for each specific label—a true "needle in a haystack" problem.

State-of-the-art Transformer models show promising but nascent performance on CUAD. DeBERTa-xlarge achieves 44.0% Precision @ 80% Recall and 47.8% AUPR, substantially better than BERT-base at 8.2% and 32.4% respectively, but with significant room for improvement [5]. Performance

varies substantially across label categories, from near 100% AUPR for some labels (Document Name, Parties, Agreement Date) to only 20-30% AUPR for others (Covenant Not to Sue, IP Ownership Assignment).

2.4 Document Purpose and Scope

This Product Requirements Document (PRD) defines the specifications for a knowledge graph-based legal contract analysis system that applies the KGGEN methodology to the CUAD dataset. The system targets **technology agreements** within the CUAD dataset, focusing on clauses particularly relevant to software licensing, IP ownership, SaaS terms, and technology transfer.

Target Audience: This PRD addresses a team of engineers building AI software for lawyers, including senior practicing lawyers who will validate the system and guide product development.

Key Objectives:

1. Design a knowledge graph ontology mapping all 41 CUAD label categories to a structured KG schema suitable for technology agreements
2. Implement the KGGEN pipeline adapted for legal contract extraction with domain-specific optimizations
3. Build a retrieval mechanism that provides relevant KG context to an LLM for contract analysis queries
4. Create interfaces for lawyers to upload contracts, query the knowledge graph semantically, and receive AI-assisted analysis
5. Achieve extraction accuracy and usability suitable for professional legal practice with human-in-the-loop validation

Out of Scope: This initial phase does not cover (1) non-technology contract types (employment, real estate, etc.), (2) automated legal advice or risk assessment without human review, (3) contract generation or drafting capabilities, or (4) multi-jurisdictional legal analysis beyond common law systems.

3 Product Scope and Objectives

3.1 Target Users

3.1.1 Primary Users: Legal Professionals

In-House Counsel: Corporate lawyers reviewing technology agreements for software procurement, API licensing, cloud services, and technology partnerships. Need to quickly identify key terms (license scope, liability caps, termination rights, IP ownership) across multiple contracts.

Law Firm Associates: Junior lawyers performing initial contract review and due diligence. Spend significant time on repetitive clause identification tasks. Benefit from automated extraction to focus on higher-value analysis and client counseling.

Legal Operations Teams: Manage contract portfolios, track obligations, and ensure compliance. Need structured data extraction to populate contract management systems and generate reports on contract terms across the organization.

3.1.2 Secondary Users: Engineering and Product Teams

Software Engineers: Build and maintain the KGGEN extraction pipeline, knowledge graph infrastructure, and retrieval systems. Require clear technical specifications, API documentation, and performance benchmarks.

Machine Learning Engineers: Train and fine-tune models on CUAD data, optimize entity resolution algorithms, and improve extraction accuracy. Need access to labeled training data and evaluation metrics.

Product Managers: Define features, prioritize development, and measure user adoption. Balance technical feasibility with user needs and business objectives.

3.1.3 Tertiary Users: Business Stakeholders

Procurement Teams: Review vendor contracts for compliance with corporate standards. Benefit from automated identification of unfavorable terms and risk factors.

Sales and Business Development: Negotiate customer agreements and partnerships. Use the system to understand common market terms and identify negotiation points.

3.2 Use Cases

3.2.1 UC1: Contract Upload and Automated Extraction

Actor: Legal professional
Precondition: User has a technology agreement in PDF or DOCX format
Flow:

1. User uploads contract document through web interface
2. System performs OCR if document is scanned image
3. KGGEN extraction pipeline processes contract:
   - Stage 1: Extract entities and relations using Gemini 2.0 Flash
   - Stage 2: Aggregate with existing knowledge graph
   - Stage 3: Resolve entities and canonicalize relations
4. System populates knowledge graph with extracted triples
5. User receives summary of identified clauses organized by CUAD categories
6. System highlights confidence scores for each extraction

Postcondition: Contract is added to knowledge graph and searchable
Success Metric: Processing time ¡2 minutes, extraction accuracy ¿85%

3.2.2 UC2: Semantic Search for Contract Clauses

Actor: Legal professional
Precondition: Contracts have been processed and added to knowledge graph
Flow:

1. User enters natural language query: “Show me all perpetual licenses granted in our technology contracts”

2. System embeds query using all-MiniLM-L6-v2 model
3. Hybrid retrieval combines BM25 keyword matching and semantic similarity
4. System retrieves top-k relevant triples from knowledge graph
5. Multi-hop expansion traverses 2-hop neighborhood for related clauses
6. Results display with:
    - Highlighted contract sections
    - Graph visualization showing related entities
    - Source attribution with page numbers
    - Confidence scores
7. User can refine query or request LLM synthesis

Postcondition: User has identified relevant clauses across contract portfolio
Success Metric: Query response time ¡3 seconds, recall ¿90%

3.2.3 UC3: LLM-Assisted Contract Analysis

Actor: Legal professional
Precondition: Specific contract or clause set has been identified
Flow:

1. User asks analysis question: "What are the termination conditions and notice requirements in this agreement?"
2. System retrieves relevant knowledge graph context
3. System assembles prompt with:
    - Retrieved triples
    - Source text chunks
    - User question
4. LLM generates natural language analysis grounded in KG facts
5. System presents analysis with citations to specific contract sections
6. User can flag errors or request human lawyer review

Postcondition: User has preliminary analysis to guide detailed legal review
Success Metric: Analysis accuracy ¿80%, user satisfaction ¿4.0/5.0

3.2.4 UC4: Risk Identification and Compliance Checking

Actor: Legal operations team
Precondition: Organization has defined risk criteria and compliance standards
Flow:

1. System scans knowledge graph for high-risk patterns:
   - Uncapped liability provisions
   - Perpetual or auto-renewing terms
   - Broad indemnification obligations
   - Non-standard governing law
   - IP assignment without compensation
2. System generates risk report with:
   - Contracts containing high-risk clauses
   - Severity scoring
   - Comparison to organizational standards
   - Recommendations for remediation
3. User reviews flagged contracts for follow-up

Postcondition: High-risk contracts identified for legal review
Success Metric: False positive rate ¡20%, critical risk recall ¿95%

3.3 Success Metrics

Table 2: Key Performance Indicators for KGGEN-CUAD System

Category	Metric	Target & Measurement Method
Extraction Quality	AUPR	¿45% on CUAD test set (baseline: DeBERTa 47.8%)
	Precision @ 80% Recall	¿50% (baseline: DeBERTa 44.0%)
	Entity Resolution	¿20% deduplication ratio on large contracts
Performance	Contract Processing	¡2 minutes per 50-page contract
	Query Latency	¡3 seconds at 95th percentile
	KG Construction	¡5 minutes for 100 contracts
	Concurrent Users	Support 50+ simultaneous users
Business	Time Savings	¿50% reduction vs. manual review
	Cost Reduction	¿60% reduction in contract review costs
	User Adoption	¿70% of legal team using weekly by month 6
	User Satisfaction	¿4.0/5.0 in quarterly surveys
Technical	System Uptime	¿99.5% availability
	API Success Rate	¿99.9% for retrieval endpoints
	Data Accuracy	¡5% error rate on critical clauses

3.4 Out of Scope

The following capabilities are explicitly out of scope for the initial release:

Contract Types: Non-technology contracts including employment agreements, real estate leases, loan documents, and consumer contracts. Future phases may expand to additional domains.

Legal Services: The system does not provide legal advice, risk assessment, or strategic recommendations. All outputs include disclaimers that analysis should be reviewed by qualified legal counsel.

Contract Generation: The system does not draft, generate, or modify contracts. It is purely an analysis and extraction tool.

Multi-Jurisdiction Support: Initial release focuses on common law contracts governed by US state law (primarily California, Delaware, New York). International and civil law jurisdictions are out of scope.

Real-Time Collaboration: Multi-user editing, commenting, and version control features are deferred to later phases.

Advanced Analytics: Predictive modeling, contract negotiation recommendations, and market benchmarking are future enhancements.

4 Technical Architecture

4.1 KGGEN Pipeline for Legal Contracts

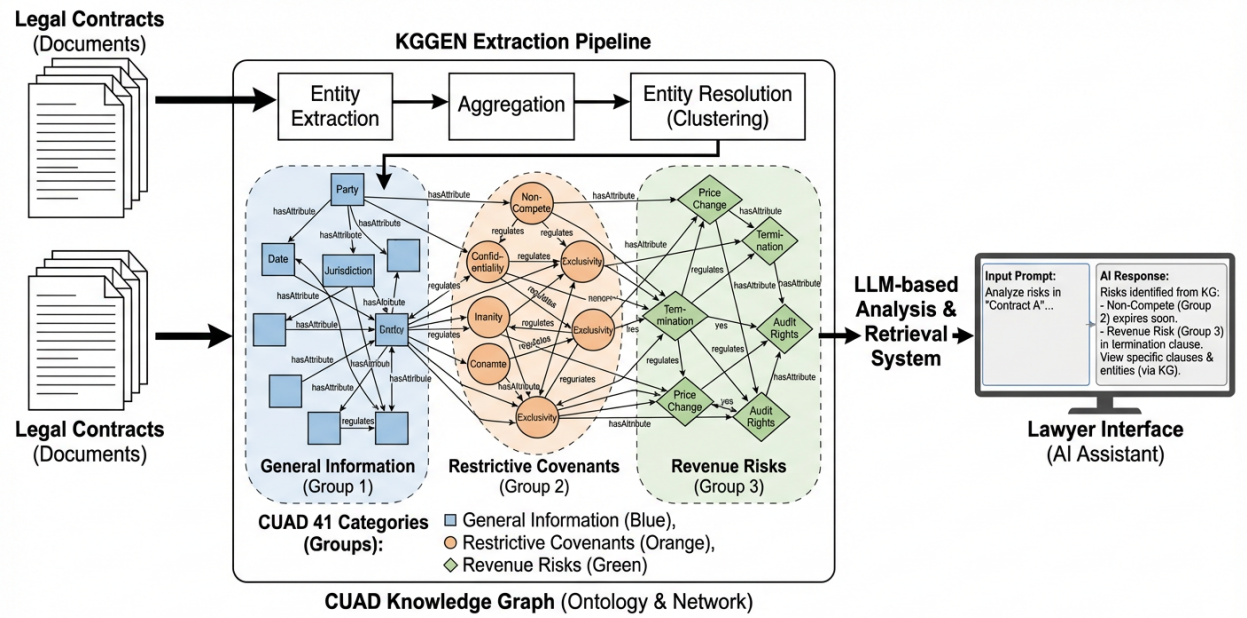

Figure 1: Graphical Abstract for KGGEN-CUAD Knowledge Graph System: End-to-End Legal Contract Analysis Pipeline

Figure 1: Graphical abstract showing the end-to-end KGGEN-CUAD knowledge graph system architecture from contract documents through extraction pipeline to LLM-assisted analysis.

The system applies the three-stage KGGEN pipeline [8] to legal contract documents, with domain-specific adaptations for the legal domain. Figure 1 illustrates the complete workflow from raw contract documents to LLM-powered analysis.

4.1.1 Stage 1: Entity and Relation Extraction

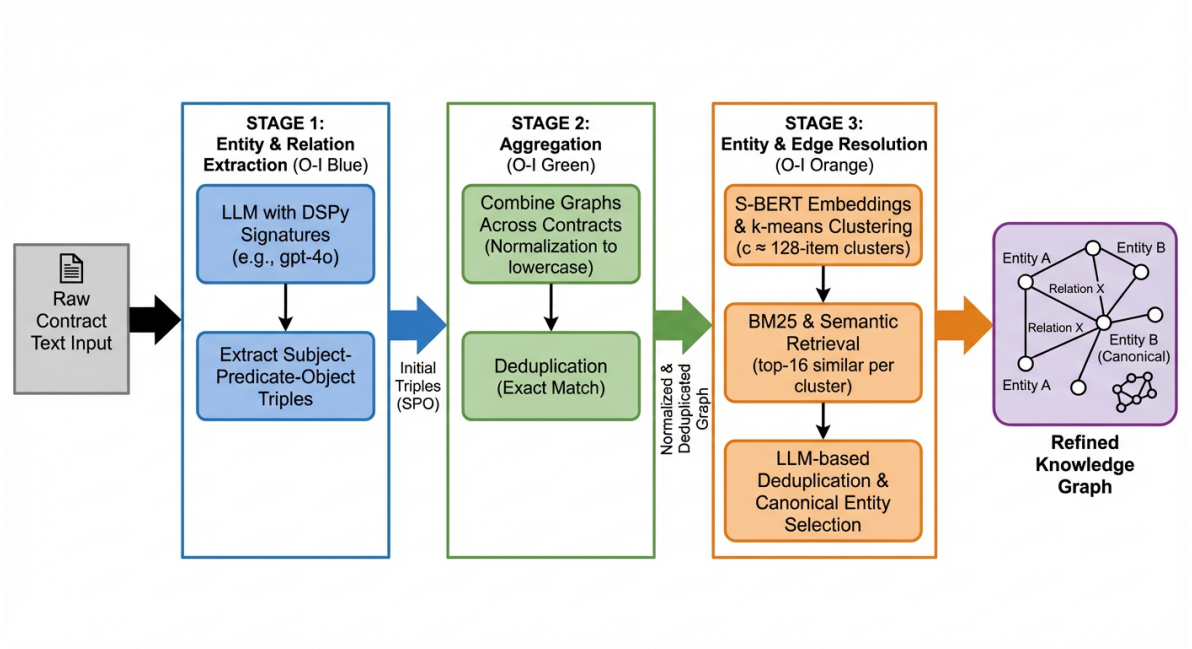

Figure 2: KGGEN pipeline architecture adapted for legal contract processing, showing three stages: entity and relation extraction, aggregation, and entity resolution.

The first stage extracts subject-predicate-object triples from contract text using language models with structured output via DSPy signatures.

Input Processing:

- **Document Parsing:** Convert PDF/DOCX to plain text, preserving structure
- **Segmentation:** Split contracts into logical sections (definitions, obligations, termination, etc.)
- **Chunk Size:** Process in 1000-token chunks with 200-token overlap to maintain context

Entity Extraction: The first DSPy signature extracts entities from source text with the prompt:

> *"Extract key entities from the source text. Extracted entities are subjects or objects. This is for an extraction task, please be thorough and accurate to the reference text."*

For legal contracts, entities include: parties (companies, individuals), clauses (obligations, rights, restrictions), dates, terms (license grants, warranties), and legal concepts (governing law, jurisdiction).

Relation Extraction: The second DSPy signature extracts relationships with the prompt:

> *"Extract subject-predicate-object triples from the source text. Subject and object must be from entities list. Entities provided were previously extracted from the same source text. This is for an extraction task, please be thorough, accurate, and faithful to the reference text."*

LLM Configuration:

- **Model:** Google Gemini 2.0 Flash (chosen for speed and cost-effectiveness)
- **Temperature:** 0.0 (deterministic extraction)
- **Max Tokens:** 4000 (sufficient for entity lists and triples)
- **DSPy Optimization:** Use DSPy to automatically optimize prompts based on CUAD training examples

Example Extraction:
Source text: "This Agreement shall be governed by the laws of the State of California without giving effect to conflict or choice of law principles. The Company grants to the Licensee a worldwide, non-exclusive, perpetual license to use the Software."
Extracted entities: ["Agreement", "laws of the State of California", "Company", "Licensee", "worldwide, non-exclusive, perpetual license", "Software"]
Extracted triples:

- (Agreement, governed_by, laws of the State of California)
- (Company, grants, worldwide, non-exclusive, perpetual license)
- (worldwide, non-exclusive, perpetual license, permits_use_of, Software)
- (Licensee, receives, worldwide, non-exclusive, perpetual license)

4.1.2 Stage 2: Aggregation

After extracting triples from each contract section, the aggregation stage combines graphs across all source texts:
Collection: Collect all unique entities and edges across all contract sections and combine into a single graph representation.
Normalization: Normalize all entities and edges to lowercase to reduce trivial duplicates (e.g., "Company" and "company" become the same node).
Deduplication: Remove exact duplicate triples that appear in multiple contract sections.
Cross-Contract Integration: For multi-contract knowledge graphs, merge entities that clearly refer to the same real-world entity based on string matching (e.g., "Acme Corp" and "Acme Corporation").
The aggregation stage does not require LLM processing, making it computationally efficient. For a 100-page contract, aggregation typically reduces the number of unique entities by 10-15% through normalization alone.

4.1.3 Stage 3: Entity and Edge Resolution

The entity resolution stage is KGGen's key innovation, addressing the sparsity problem where extractors create nearly as many unique relation types as edges [8]. This stage employs a hybrid clustering and LLM-based deduplication approach.
Clustering Phase:

1. **Embedding:** Generate semantic embeddings for all entities/edges using S-BERT (all-MiniLM-L6-v2 model)

2. **K-means Clustering:** Cluster items into groups of 128 semantically similar items
3. **Purpose:** Reduce LLM processing by only comparing items likely to be duplicates

Deduplication Phase (within each cluster):

1. **Retrieval:** For each item, retrieve top-16 most similar items using:
   - BM25 keyword matching score
   - Semantic cosine similarity score
   - Fused score = 0.5 * BM25 + 0.5 * cosine_similarity
2. **LLM Duplicate Detection:** Prompt LLM with:

   > *"Find duplicate [entity/edge] for the item and an alias that best represents the duplicates. Duplicates are those that are the same in meaning, such as with variation in tense, plural form, stem form, case, abbreviation, shorthand. Return an empty list if there are none."*

3. **Canonical Selection:** LLM selects the most representative canonical form (similar to Wikidata aliases)
4. **Mapping:** Create cluster maps tracking which entities belong to which canonical alias
5. **Iteration:** Remove resolved items from cluster and repeat until cluster is empty

Legal Domain Examples:
Entity resolution:

- "Licensee", "the Licensee", "licensee" → canonical: "Licensee"
- "IP", "intellectual property", "Intellectual Property Rights" → canonical: "Intellectual Property"
- "California Law", "laws of California", "California state law" → canonical: "California Law"

Edge resolution:

- "governed by", "governed under", "subject to laws of" → canonical: "governed_by"
- "grants license", "provides license to", "licenses" → canonical: "grants_license"
- "terminates upon", "ends when", "expires on" → canonical: "terminates_on"

Performance: For a 1M character corpus, entity resolution achieves:

- 22.4% entity deduplication ratio
- 23.0% edge deduplication ratio
- Processing time: 279 seconds (versus 273 seconds for extraction) [8]

4.2 System Architecture

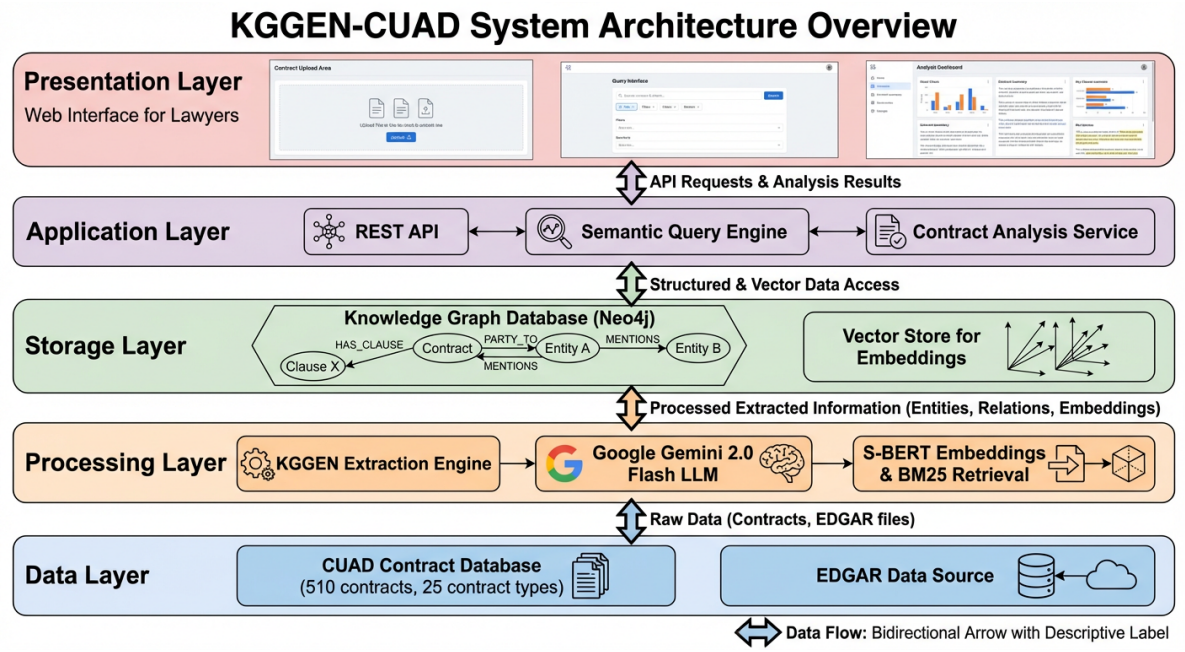

Figure 3: Multi-tier system architecture for the KGGEN-CUAD platform showing data layer, processing layer, storage layer, application layer, and presentation layer.

The system follows a multi-tier architecture (Figure 3) designed for scalability, maintainability, and separation of concerns.

4.2.1 Data Layer

CUAD Contract Database:

- 510 contracts from CUAD dataset as training/validation data
- 13,101 expert annotations across 41 label categories
- Storage: PostgreSQL for metadata, S3 for PDF/DOCX files
- Indexing: Full-text search with Elasticsearch for rapid contract retrieval

EDGAR Data Source:

- SEC EDGAR system for additional public contracts
- Automated scraping of Technology Agreements, License Agreements, Service Agreements
- Filtering: Focus on contracts ¿10 pages, filed within last 5 years
- Volume: Potential 10,000+ additional contracts for knowledge graph expansion

User Uploaded Contracts:

- Private contracts uploaded by legal teams

- Storage: Encrypted S3 buckets with organization-level access control
- Retention: User-configurable (default 7 years per legal standards)

4.2.2 Processing Layer

KGGEN Extraction Engine:

- Orchestration: Apache Airflow for pipeline management
- Compute: Kubernetes pods for parallel contract processing
- Scaling: Auto-scale based on queue depth (1-20 pods)
- Monitoring: Prometheus metrics for extraction success rate, latency, errors

LLM Integration:

- Model: Google Gemini 2.0 Flash via OpenRouter API
- Fallback: GPT-4o for complex extractions if Gemini fails
- Rate Limiting: 100 requests/minute per contract
- Cost Optimization: Cache LLM responses for identical contract sections

Embedding Services:

- Model: S-BERT all-MiniLM-L6-v2 (384-dimensional embeddings)
- Deployment: Self-hosted via FastAPI for cost control
- GPU: Single T4 GPU sufficient for 1000 embeddings/second
- Batch Processing: Embed 100 texts at once for efficiency

BM25 Retrieval:

- Implementation: Elasticsearch BM25 scoring
- Indexing: Real-time indexing as triples are extracted
- Tuning: k1=1.2, b=0.75 (standard BM25 parameters)

4.2.3 Storage Layer

Knowledge Graph Database:

- Platform: Neo4j Graph Database (Community Edition)
- Schema: Labeled property graph with typed nodes and relationships
- Indexing: Full-text indexes on entity names, relationship types
- Capacity: Support 1M+ nodes, 5M+ relationships
- Backups: Daily snapshots to S3, 30-day retention

Vector Store:

- Platform: Qdrant vector database
- Embedding Dimension: 384 (matches S-BERT output)
- Index Type: HNSW (Hierarchical Navigable Small World) for fast similarity search
- Performance: ¡50ms query latency for top-100 retrieval
- Storage: 1GB per 1M embeddings (approximately)

Document Store:

- Platform: AWS S3 with server-side encryption
- Organization: Bucket per organization for access control
- Lifecycle: Transition to Glacier after 90 days of inactivity
- CDN: CloudFront for fast PDF serving to web interface

4.2.4 Application Layer

REST API:

- Framework: FastAPI (Python)
- Endpoints:
  - `POST /contracts/upload` - Upload and process contract
  - `GET /contracts/{id}` - Retrieve contract metadata and KG
  - `POST /query/semantic` - Semantic search across contracts
  - `POST /query/analyze` - LLM-powered contract analysis
  - `GET /graph/visualize` - Generate graph visualization
- Authentication: OAuth 2.0 with JWT tokens
- Rate Limiting: 1000 requests/hour per user
- Documentation: OpenAPI/Swagger auto-generated

Semantic Query Engine:

- Query Processing: Parse natural language to structured queries
- Hybrid Retrieval: Fuse BM25 and semantic scores
- Multi-hop Traversal: Cypher queries on Neo4j for graph navigation
- Result Ranking: Re-rank by relevance, recency, confidence

Contract Analysis Service:

- Context Assembly: Retrieve KG triples + source text chunks
- Prompt Engineering: Structured prompts with explicit instructions
- LLM Integration: Call GPT-4o for analysis synthesis
- Citation Extraction: Parse LLM output for source attributions

4.2.5 Presentation Layer

Web Application:

- Framework: React + TypeScript
- UI Components: Material-UI for professional appearance
- State Management: Redux for complex application state
- Routing: React Router for single-page application

Key Interfaces:

- **Contract Upload:** Drag-and-drop upload, processing status tracking
- **Query Interface:** Natural language search bar, query suggestions, search history
- **Results Viewer:** Highlighted contract sections, confidence scores, graph visualization
- **Analysis Dashboard:** Contract summaries, risk indicators, clause coverage matrix

Graph Visualization:

- Library: D3.js or Cytoscape.js for interactive network graphs
- Features: Node filtering, edge filtering, zoom/pan, hover details
- Layout: Force-directed layout for automatic positioning
- Export: SVG/PNG export for reports

4.3 Technology Stack

Table 3: Technology stack for KGGEN-CUAD system

Component	Technology
Language Models	
Primary LLM	Google Gemini 2.0 Flash
Fallback LLM	OpenAI GPT-4o
Embedding Model	S-BERT all-MiniLM-L6-v2
Data Storage	
Graph Database	Neo4j 5.x
Vector Database	Qdrant
Document Storage	AWS S3
Metadata Database	PostgreSQL 15
Search Engine	Elasticsearch 8.x
Backend	
API Framework	FastAPI (Python 3.11)
Task Queue	Celery with Redis
Orchestration	Apache Airflow
Caching	Redis 7.x
Frontend	
Framework	React 18 + TypeScript
UI Library	Material-UI v5
State Management	Redux Toolkit
Visualization	D3.js / Cytoscape.js
Infrastructure	
Container Platform	Docker + Kubernetes
Cloud Provider	AWS (primary)
CI/CD	GitHub Actions
Monitoring	Prometheus + Grafana
Logging	ELK Stack (Elasticsearch, Logstash, Kibana)

Development Tools:

- **Version Control:** Git + GitHub
- **Code Quality:** Pylint, Black, MyPy for Python; ESLint for TypeScript
- **Testing:** PyTest (backend), Jest (frontend), Locust (load testing)
- **Documentation:** Sphinx (API docs), Docusaurus (user docs)

5 Ontology Definition: CUAD Label Mapping

5.1 CUAD Ontology Overview

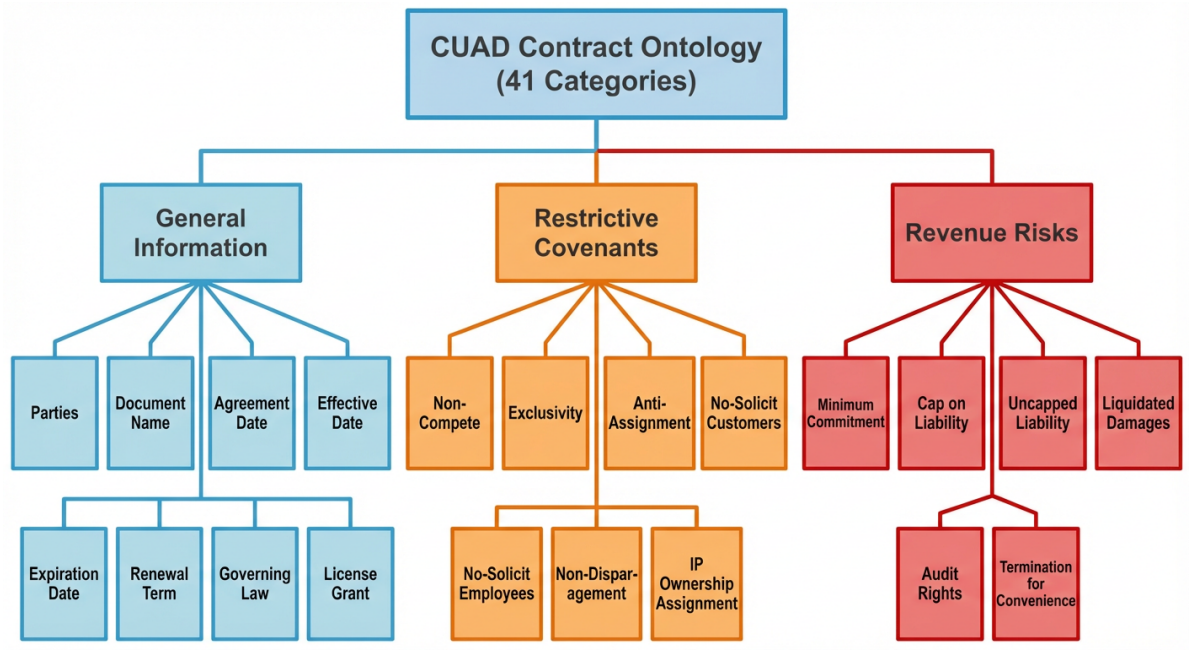

Figure 4: CUAD ontology hierarchy organized by three main categories covering 41 label types for legal contract analysis.

The CUAD dataset organizes contract clauses into 41 label categories across three high-level groups [5]. Figure 4 shows the hierarchical organization of these categories. Our knowledge graph ontology maps each CUAD label to specific node and relationship types.

5.2 Category 1: General Information (11 Labels)

General information labels capture basic contract metadata and terms.

Table 4: General Information CUAD labels and KG mapping

CUAD Label	Description	KG Mapping
Document Name	Name of the contract	Contract node property
Parties	Two or more parties who signed	Party nodes with HAS_PARTY relationship
Agreement Date	Date contract was signed	Date node with SIGNED_ON relationship
Effective Date	When contract becomes effective	Date node with EFFECTIVE_ON relationship
Expiration Date	When initial term expires	Date node with EXPIRES_ON relationship
Renewal Term	Renewal period after initial term	Clause node with SPECIFIES_RENEWAL relationship
Notice to Terminate Renewal	Notice period for termination	Clause node with REQUIRES_NOTICE relationship
Governing Law	State/country law that governs	Law node with GOVERNED_BY relationship
License Grant	License granted by one party	License node with GRANTS_LICENSE relationship
Most Favored Nation	Third party better terms clause	Clause node with MFN_PROVISION relationship

Technology Agreement Focus: For technology contracts, License Grant is particularly important, covering:

- Software license scope (perpetual, term-limited, subscription)
- Geographic restrictions (worldwide, specific territories)
- Exclusivity (exclusive, non-exclusive)
- Transferability (transferable, non-transferable)
- Sublicensing rights (with/without right to sublicense)
- Usage restrictions (development, production, internal use)

5.3 Category 2: Restrictive Covenants (15 Labels)

Restrictive covenants limit the buyer's or company's ability to operate.

Table 5: Restrictive Covenants CUAD labels and KG mapping

CUAD Label	Description	KG Mapping
Non-Compete	Restriction on competing	Obligation node with PROHIBITS_COMPETITION relationship
Exclusivity	Exclusive dealing commitment	Clause node with REQUIRES_EXCLUSIVITY relationship
Anti-Assignment	Consent required for assignment	Restriction node with RESTRICTS_ASSIGNMENT relationship
No-Solicit of Customers	Cannot solicit counterparty customers	Obligation node with PROHIBITS_CUSTOMER_SOLICITATION relationship
No-Solicit of Employees	Cannot solicit counterparty employees	Obligation node with PROHIBITS_EMPLOYEE_SOLICITATION relationship
Non-Disparagement	Requirement not to disparage	Obligation node with PROHIBITS_DISPARAGEMENT relationship
IP Ownership Assignment	IP becomes property of counterparty	IP_Rights node with ASSIGNS_IP relationship
Joint IP Ownership	Joint/shared IP ownership	IP_Rights node with SHARES_IP relationship
Non-Transferable License	License cannot be transferred	License node property
Covenant Not to Sue	Cannot contest counterparty IP	Obligation node with WAIVES_LEGAL_CLAIMS relationship
Competitive Restriction Exception	Exceptions to restrictions	Exception node with EXEMPTS_FROM relationship

Technology Agreement Focus: Critical restrictive covenants for technology contracts:

- **IP Ownership Assignment:** Determines who owns IP created under the agreement (work-for-hire provisions, background IP vs. foreground IP)
- **Non-Compete:** May restrict use of competing technologies or development of competing products
- **Exclusivity:** May require exclusive use of vendor's platform or prohibit multi-vendor strategies
- **Anti-Assignment:** Restricts transfer of licenses in M&A transactions or organizational restructuring

5.4 Category 3: Revenue Risks (15 Labels)

Revenue risk labels identify terms requiring additional cost or remedial measures.

Table 6: Revenue Risks CUAD labels and KG mapping

CUAD Label	Description	KG Mapping
Minimum Commitment	Minimum order/usage required	Obligation node with REQUIRES_MINIMUM relationship
Cap on Liability	Maximum liability amount	Clause node with CAPS_LIABILITY relationship
Uncapped Liability	Unlimited liability for breaches	Clause node with UNCAPS_LIABILITY relationship
Liquidated Damages	Damages for breach/termination	Clause node with SPECIFIES_DAMAGES relationship
Audit Rights	Right to audit compliance	Right node with PERMITS_AUDIT relationship
Revenue/Profit Sharing	Share revenue/profit with counterparty	Obligation node with SHARES_REVENUE relationship
Price Restriction	Restriction on price changes	Restriction node with RESTRICTS_PRICING relationship
Volume Restriction	Fee increase if usage exceeds threshold	Clause node with LIMITS_VOLUME relationship
Warranty Duration	Duration of warranty period	Warranty node with DURATION property
Insurance	Required insurance coverage	Requirement node with REQUIRES_INSURANCE relationship
Termination for Convenience	Terminate without cause	Right node with PERMITS_TERMINATION relationship
Notice Period to Terminate	Notice required for termination	Requirement node with REQUIRES_NOTICE relationship
Post-Termination Services	Obligations after termination	Obligation node with POST_TERM_OBLIGATION relationship
Source Code Escrow	Deposit source code with third party	Requirement node with REQUIRES_ESCROW relationship
Third Party Beneficiary	Non-contracting party beneficiary	Party node with BENEFITS_FROM relationship

Technology Agreement Focus: Key revenue risk terms for technology contracts:

- **Minimum Commitment:** May require minimum API calls, storage, or user seats regardless of actual usage
- **Volume Restriction:** Tiered pricing that increases costs beyond certain thresholds
- **Source Code Escrow:** Deposits source code with third party for release if vendor fails

- **Warranty Duration:** Defines how long software must perform as specified
- **Post-Termination Services:** May require data migration assistance or continued access after termination

5.5 Technology Agreement Ontology Extensions

For technology agreements, we extend the base CUAD ontology with additional specific labels:

Table 7: Technology-specific ontology extensions

Extension Label	Description	KG Mapping
SaaS Terms	Service level commitments	SLA node with GUARANTEES_UPTIME relationship
API Access Rights	API usage terms and rate limits	API_License node with PERMITS_API_ACCESS relationship
Data Ownership	Ownership of customer data	Data_Rights node with OWNS_DATA relationship
Data Privacy	GDPR, CCPA compliance requirements	Compliance node with REQUIRES_COMPLIANCE relationship
Deployment Model	Cloud, on-premise, hybrid	Contract node property
Integration Requirements	Third-party integration terms	Requirement node with PERMITS_INTEGRATION relationship
Security Standards	SOC 2, ISO 27001 requirements	Compliance node with MEETS_STANDARD relationship
Customization Rights	Ability to modify software	Right node with PERMITS_CUSTOMIZATION relationship
Update Obligations	Software update/maintenance terms	Obligation node with PROVIDES_UPDATES relationship

6 Data Schema and Knowledge Graph Structure

6.1 Entity Types

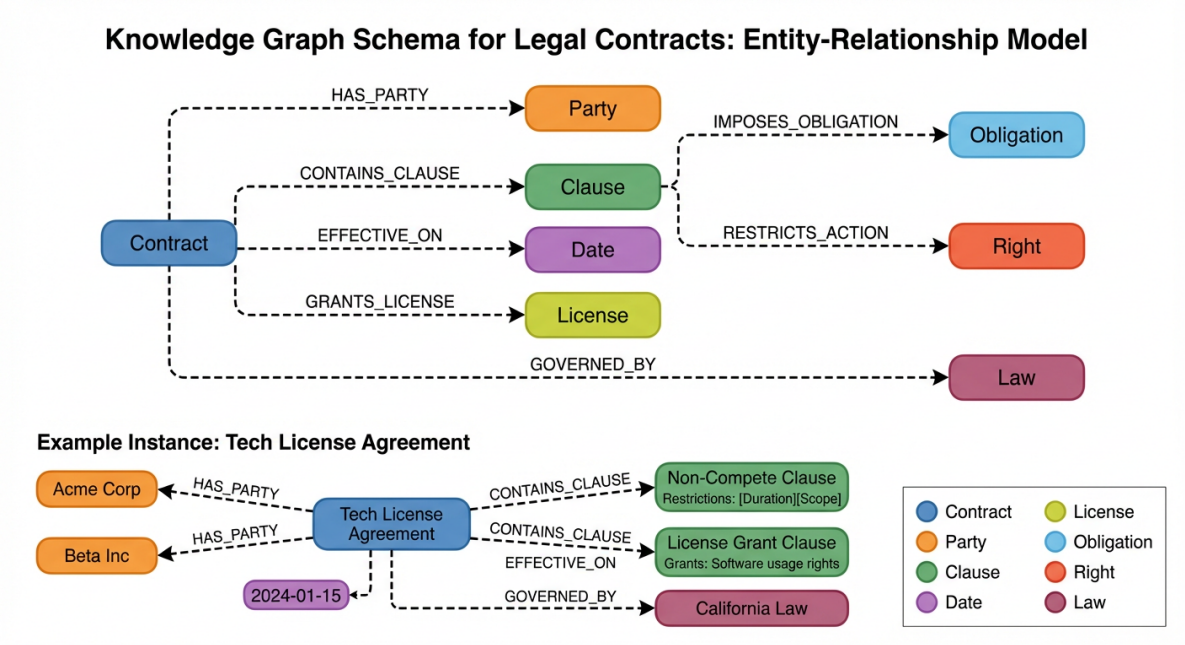

Figure 5: Knowledge graph schema showing entity types and relationships for legal contracts with example instances from a technology licensing agreement.

The knowledge graph schema (Figure 5) defines entity types as labeled nodes with properties.

Table 8: Entity type definitions and properties

Entity Type	**Description**	**Properties**
Contract	Root entity for each contract	id, document_name, contract_type, page_count, date_uploaded, file_path
Party	Legal entities who are parties	name, role (licensor/licensee), organization_type, address
Clause	Contract provisions and terms	clause_type (from CUAD labels), text, confidence_score, page_number, section
Date	Temporal entities	date_value, date_type (effective/expiration/signature)
License	License grants	scope (exclusive/non-exclusive), territory, perpetual (boolean), sublicense_rights (boolean)
Obligation	Requirements imposed on parties	obligation_type, subject_party, text, duration
Right	Permissions granted to parties	right_type, beneficiary_party, text, conditions
Law	Governing law and jurisdiction	jurisdiction (state/country), legal_system (common/civil law)
IP_Rights	Intellectual property terms	ip_type (patent/copyright/trademark), ownership (sole/joint), assignment (boolean)
Liability	Liability provisions	cap_amount, cap_currency, uncapped (boolean), liability_type
Termination	Termination conditions	notice_period, for_cause (boolean), for_convenience (boolean), fees

6.2 Relationship Types

Relationships connect entities and represent the structure of contract terms.

Table 9: Relationship type definitions

Relationship	Source Entity	Target Entity	Description
HAS_PARTY	Contract	Party	Contract involves party
CONTAINS_CLAUSE	Contract	Clause	Contract contains clause
EFFECTIVE_ON	Contract	Date	Contract effective date
EXPIRES_ON	Contract	Date	Contract expiration date
SIGNED_ON	Contract	Date	Contract signature date
GOVERNED_BY	Contract	Law	Governing law
GRANTS_LICENSE	Contract	License	License grant
IMPOSES_OBLIGATION	Clause	Obligation	Clause creates obligation
CONFERS_RIGHT	Clause	Right	Clause grants right
RESTRICTS_ACTION	Clause	Obligation	Clause restricts activity
SPECIFIES_LIABILITY	Clause	Liability	Liability provision
DEFINES_TERMINATION	Clause	Termination	Termination conditions
BINDS_PARTY	Obligation	Party	Obligation applies to party
BENEFITS_PARTY	Right	Party	Right benefits party
COVERS_IP	Clause	IP_Rights	IP ownership/license
ASSIGNS_IP	Party	IP_Rights	IP assignment
SHARES_IP	Party	IP_Rights	Joint IP ownership
REQUIRES_NOTICE	Termination	Date	Notice period
PERMITS_TERMINATION	Clause	Termination	Termination allowed

6.3 Property Schemas

6.3.1 Node Properties

All Nodes:

- `id`: Unique identifier (UUID)
- `created_at`: Timestamp of extraction
- `updated_at`: Timestamp of last update
- `source_text`: Original contract text (for clauses/obligations)
- `confidence_score`: Extraction confidence 0-1

Contract Properties:

- `document_name`: String
- `contract_type`: Enum (License, Service, Joint Venture, etc.)
- `page_count`: Integer

- `word_count`: Integer
- `file_path`: S3 URI
- `extraction_version`: String (pipeline version)
- `cuad_category_coverage`: JSON (which of 41 labels found)

Clause Properties:

- `clause_type`: Enum (from 41 CUAD labels)
- `text`: String (extracted clause text)
- `page_number`: Integer
- `section`: String (contract section name)
- `start_char`: Integer (character offset in document)
- `end_char`: Integer
- `confidence_score`: Float 0-1
- `validated_by_human`: Boolean
- `risk_level`: Enum (low, medium, high, critical)

6.3.2 Relationship Properties

All relationships can have:

- `confidence_score`: Float 0-1 (extraction confidence)
- `source_page`: Integer (page where relationship was extracted)
- `extraction_method`: Enum (LLM, rule-based, hybrid)

6.4 Example Knowledge Graph

Consider a technology licensing agreement between Acme Corp (Licensor) and Beta Inc (Licensee):
Contract Node:

- id: "c-12345"
- document_name: "Software License Agreement"
- contract_type: "License"
- page_count: 42

Party Nodes:

- Party 1: {name: "Acme Corp", role: "Licensor"}
- Party 2: {name: "Beta Inc", role: "Licensee"}

License Node:

- scope: "non-exclusive"
- territory: "worldwide"
- perpetual: true
- sublicense_rights: false

Obligation Node (Non-Compete):

- obligation_type: "Non-Compete"
- text: "Licensee shall not develop or distribute software that competes with the Licensed Software during the term and for 2 years thereafter."
- duration: "term + 2 years"

IP_Rights Node:

- ip_type: "derivative works"
- ownership: "Licensor"
- assignment: true

Liability Node:

- cap_amount: 1000000
- cap_currency: "USD"
- uncapped: false
- liability_type: "general liability"

Termination Node:

- notice_period: "90 days"
- for_cause: true
- for_convenience: true
- fees: "none"

Key Relationships:

1. (Contract)-[HAS_PARTY]→(Acme Corp)
2. (Contract)-[HAS_PARTY]→(Beta Inc)
3. (Contract)-[GRANTS_LICENSE]→(License)
4. (Contract)-[GOVERNED_BY]→(California Law)
5. (Non-Compete Clause)-[IMPOSES_OBLIGATION]→(Obligation)

6. (Obligation)-[BINDS_PARTY]→(Beta Inc)
7. (IP Clause)-[COVERS_IP]→(IP_Rights)
8. (Beta Inc)-[ASSIGNS_IP]→(IP_Rights)
9. (Contract)-[EFFECTIVE_ON]→(2024-01-15)
10. (Contract)-[EXPIRES_ON]→(2027-01-15)

This knowledge graph structure enables queries like:

- "What licenses has Acme Corp granted?"
- "Show all contracts with non-compete obligations binding Beta Inc"
- "Find contracts where IP ownership is assigned to the licensor"
- "List contracts governed by California law that expire in 2027"

7 LLM Context Retrieval Mechanism

7.1 Query Processing Pipeline

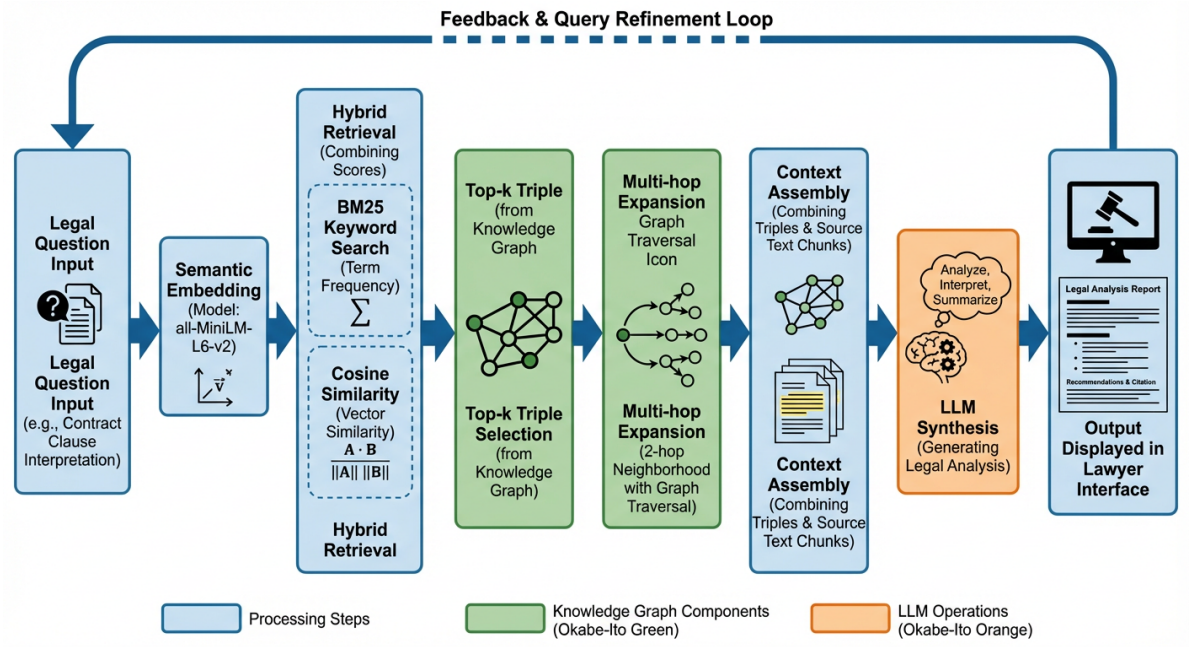

Figure 6: LLM context retrieval mechanism for contract analysis showing semantic embedding, hybrid retrieval, multi-hop expansion, and LLM synthesis.

The retrieval mechanism (Figure 6) converts natural language queries into structured knowledge graph context for LLM-powered analysis. The pipeline follows established RAG (Retrieval-Augmented Generation) principles adapted for legal contract knowledge graphs.

7.1.1 Stage 1: Query Understanding

Input: Natural language legal question, e.g., "What are the termination conditions for contracts with Beta Inc?"

Query Parsing:

- **Intent Classification:** Classify query intent (information extraction, comparison, risk assessment, compliance check)
- **Entity Extraction:** Identify mentioned entities ("Beta Inc", "termination")
- **Scope Detection:** Determine if query targets single contract or portfolio-wide
- **CUAD Label Mapping:** Map query to relevant CUAD categories ("Termination for Convenience", "Notice Period to Terminate Renewal")

Query Expansion: Generate synonyms and related terms:

- "termination" → ["terminate", "termination conditions", "end agreement", "cancel contract"]
- "Beta Inc" → ["Beta Inc", "Beta Incorporated", "the Licensee"]

7.2 Embedding Models

Model: all-MiniLM-L6-v2 from SentenceTransformers [8]

Specifications:

- Embedding Dimension: 384
- Max Sequence Length: 512 tokens
- Model Size: 80MB
- Inference Speed: 1000 texts/second on CPU

Embedding Process:

1. Tokenize query using SentenceTransformers tokenizer
2. Generate 384-dimensional embedding vector
3. Normalize embedding to unit length for cosine similarity
4. Store query embedding for retrieval

Triple Embedding: Pre-compute embeddings for all KG triples:

- **Format:** "[Subject] [Predicate] [Object]"
- **Example:** "Contract grants_license non-exclusive worldwide perpetual license"
- **Storage:** Qdrant vector database with metadata (contract_id, confidence_score, page_number)

7.3 Hybrid Retrieval Strategy

Hybrid retrieval combines keyword matching (BM25) and semantic similarity to achieve robust retrieval [8].

7.3.1 BM25 Keyword Retrieval

Algorithm: Elasticsearch BM25 with parameters k1=1.2, b=0.75
Process:

1. Index all KG triples in Elasticsearch as documents
2. Compute term frequencies (TF) and inverse document frequencies (IDF)
3. For query, compute BM25 score for each indexed triple
4. Retrieve top-100 triples by BM25 score

Strengths: Precise keyword matching, handles exact terminology, fast retrieval (¡50ms)
Weaknesses: Misses semantic equivalents ("terminate" vs. "cancel"), sensitive to vocabulary mismatch

7.3.2 Semantic Similarity Retrieval

Algorithm: Cosine similarity between query embedding and triple embeddings
Process:

1. Embed query using all-MiniLM-L6-v2
2. Retrieve top-100 triples by cosine similarity from Qdrant vector database
3. Qdrant uses HNSW index for fast approximate nearest neighbor search

Strengths: Captures semantic equivalence, robust to vocabulary variations, handles synonyms
Weaknesses: May retrieve loosely related content, requires quality embeddings

7.3.3 Score Fusion

Combine BM25 and semantic scores using weighted average:

$$\text{score}_{\text{hybrid}} = \alpha \cdot \text{score}_{\text{BM25}} + (1 - \alpha) \cdot \text{score}_{\text{semantic}}$$

Weight Selection: $\alpha = 0.5$ (equal weighting) based on KGGEN methodology [8]
Final Retrieval: Select top-k triples by hybrid score, where k=10 for initial retrieval

7.4 Multi-Hop Graph Traversal

After retrieving top-k triples, expand context by traversing the knowledge graph.
Expansion Algorithm:

1. Extract nodes from top-k retrieved triples (subjects and objects)
2. For each extracted node, traverse 2-hop neighborhood:
   - 1-hop: Direct neighbors connected by any relationship

- 2-hop: Neighbors of neighbors

3. Collect all triples within 2-hop neighborhood
4. Filter to relevant relationship types (exclude noisy relationships)
5. Add up to 10 additional triples from multi-hop expansion

Cypher Query Example (Neo4j):

```
MATCH path = (start_node)-[*1..2]-(end_node)
WHERE start_node.id IN $retrieved_node_ids
RETURN path
LIMIT 10
```

Rationale: Multi-hop traversal captures contextual information not directly retrieved. For example:

- Query: "What are termination conditions?"
- Direct retrieval: (Contract)-[DEFINES_TERMINATION]→(Termination)
- 1-hop expansion: (Termination)-[REQUIRES_NOTICE]→(Notice Period)
- 2-hop expansion: (Notice Period)-[BINDS_PARTY]→(Party)

7.5 Context Assembly for LLM

Assemble retrieved knowledge graph context into structured prompt for LLM synthesis.

Context Components:

1. **Retrieved Triples:** Top-10 from hybrid retrieval + 10 from multi-hop expansion
2. **Source Text Chunks:** Original contract text corresponding to each triple
3. **Metadata:** Contract name, page numbers, confidence scores
4. **User Query:** Original natural language question

Prompt Template:

```
SYSTEM: You are a legal contract analysis assistant. Use the
following knowledge graph triples and contract text to answer
the question accurately. Only use information provided; do not
speculate. Cite specific contract sections.

KNOWLEDGE GRAPH TRIPLES:
{formatted_triples}

CONTRACT TEXT EVIDENCE:
{text_chunks}

USER QUESTION: {query}
```

```
INSTRUCTIONS:
1. Identify relevant triples and text
2. Synthesize clear answer
3. Cite specific pages and sections
4. Note confidence level and any ambiguities
5. Suggest follow-up review if needed

ANSWER:
```

LLM Synthesis:

- Model: GPT-4o (higher quality for legal analysis)
- Temperature: 0.1 (mostly deterministic)
- Max Tokens: 1000 (sufficient for detailed answers)
- Stop Sequences: None (allow full answer generation)

Post-Processing:

1. Parse LLM output for citations
2. Validate cited page numbers against source documents
3. Extract confidence indicators ("likely", "possibly", "definitely")
4. Format answer with highlighted contract sections
5. Generate follow-up query suggestions

7.6 Query Examples

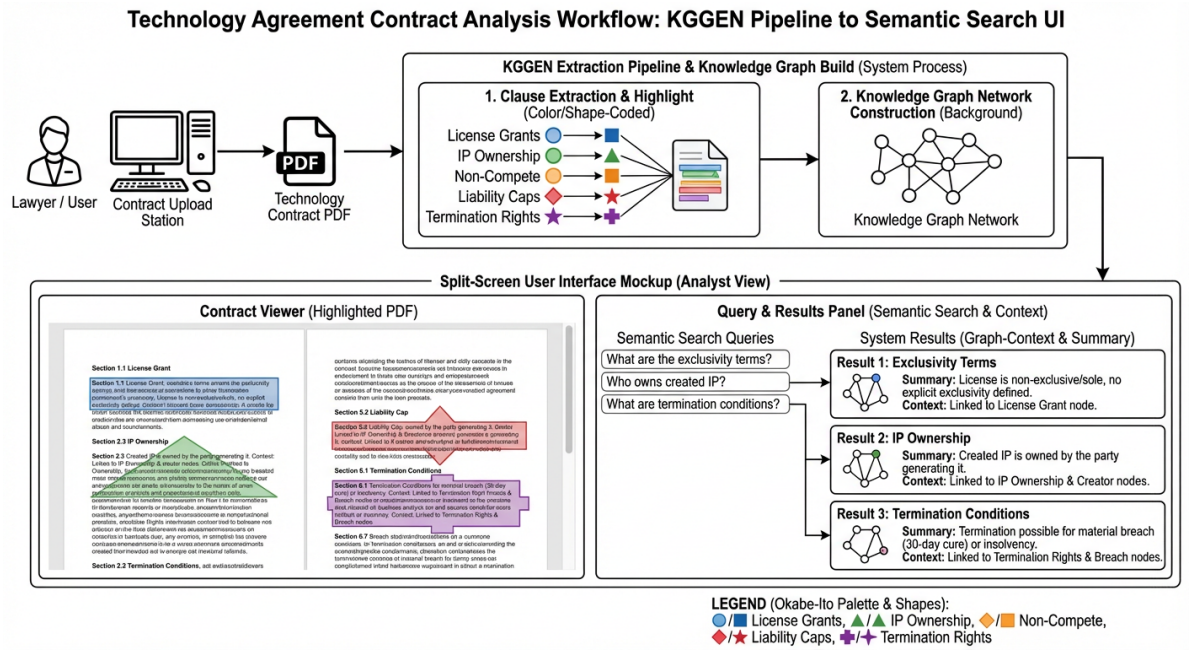

Figure 7: Technology agreement analysis workflow showing contract upload, KGGEN extraction, semantic queries, and results presentation.

Example 1: Exclusivity Terms

Query: "What are the exclusivity terms in this license agreement?"
Retrieved Triples:

- (Contract)-[CONTAINS_CLAUSE]→(Exclusivity Clause)
- (Exclusivity Clause)-[IMPOSES_OBLIGATION]→(Exclusive Dealing Obligation)
- (Exclusive Dealing Obligation)-[BINDS_PARTY]→(Licensee)

LLM Synthesis: "This license agreement contains an exclusivity provision binding the Licensee (Page 12, Section 5.2). The Licensee is prohibited from licensing, purchasing, or using competing software products during the term of this agreement and for 18 months thereafter. This is a restrictive covenant that limits the Licensee's ability to work with alternative vendors."

Example 2: IP Ownership

Query: "Who owns IP created under this agreement?"
Retrieved Triples:

- (Contract)-[COVERS_IP]→(IP Rights)
- (IP Rights)-[ownership: "Licensor"]
- (IP Rights)-[ip_type: "derivative works"]
- (Licensee)-[ASSIGNS_IP]→(IP Rights)

LLM Synthesis: "All intellectual property created under this agreement, including derivative works and improvements, becomes the sole property of the Licensor (Page 18, Section 8.1). The Licensee assigns all right, title, and interest in such IP to the Licensor. This includes inventions, patents, copyrights, and trade secrets developed using the Licensed Software."

Example 3: Termination Conditions

Query: "What are the termination conditions and notice requirements?"
Retrieved Triples:

- (Contract)-[DEFINES_TERMINATION]→(Termination)
- (Termination)-[for_cause: true, for_convenience: true]
- (Termination)-[REQUIRES_NOTICE]→(90 days)
- (Termination Clause)-[SPECIFIES_DAMAGES]→(Liquidated Damages)

LLM Synthesis: "This agreement may be terminated: (1) For cause immediately upon written notice of material breach (Page 24, Section 11.2), or (2) For convenience by either party with 90 days' advance written notice (Page 24, Section 11.3). Early termination for convenience triggers liquidated damages equal to 6 months of fees. The agreement automatically expires on January 15, 2027 unless renewed."

8 Implementation Roadmap

8.1 Phase 1: Core Infrastructure (Months 1-3)

Objective: Establish foundational data pipeline and KGGEN extraction engine

Deliverables:

- Data pipeline for contract ingestion (PDF/DOCX parsing, OCR)
- Neo4j graph database deployment with schema implementation
- Qdrant vector database setup for embeddings
- PostgreSQL metadata database with contract catalog
- S-BERT embedding service deployment
- Basic KGGEN Stage 1 (entity and relation extraction) implementation

Key Tasks:

1. Deploy infrastructure on AWS (Kubernetes cluster, RDS, S3)
2. Implement contract parsing library (PDFMiner, python-docx)
3. Configure Neo4j with KG schema and indexes
4. Set up embedding API (FastAPI + S-BERT model)
5. Integrate Google Gemini 2.0 Flash via OpenRouter
6. Build DSPy signatures for legal entity/relation extraction
7. Create Apache Airflow DAGs for pipeline orchestration

Success Criteria:

- Process 10 sample contracts end-to-end
- Extract ¿1000 triples with ¿70% accuracy (manual validation)
- Contract processing time ¡5 minutes per contract

8.2 Phase 2: CUAD Integration and Ontology Mapping (Months 4-6)

Objective: Implement full CUAD label extraction and technology agreement ontology

Deliverables:

- All 41 CUAD label extractors with DSPy optimization
- Technology agreement ontology extensions
- KGGEN Stage 2 (aggregation) and Stage 3 (entity resolution)
- Training pipeline on CUAD dataset (510 contracts, 13K annotations)
- Evaluation framework with CUAD benchmark metrics

Key Tasks:

1. Implement extractors for all 41 CUAD categories
2. Create DSPy few-shot examples from CUAD training set
3. Implement aggregation (normalization, deduplication)
4. Implement entity resolution (S-BERT clustering, BM25 retrieval, LLM deduplication)
5. Fine-tune prompts to maximize precision @ 80% recall
6. Build validation dashboard for legal expert review
7. Conduct inter-annotator agreement study with lawyers

Success Criteria:

- AUPR ¿45% on CUAD test set (vs. DeBERTa baseline 47.8%)
- Precision @ 80% Recall ¿50% (vs. DeBERTa 44.0%)
- Entity deduplication ratio ¿20% on technology contracts
- Legal expert validation: ¿85% agreement on critical clauses

8.3 Phase 3: Retrieval System and LLM Integration (Months 7-9)

Objective: Build hybrid retrieval mechanism and LLM-powered analysis

Deliverables:

- Elasticsearch BM25 indexing for KG triples
- Hybrid retrieval combining BM25 and semantic similarity
- Multi-hop graph traversal implementation
- LLM context assembly and prompt engineering
- GPT-4o integration for analysis synthesis
- Citation extraction and validation

Key Tasks:

1. Index all KG triples in Elasticsearch with metadata
2. Implement Qdrant vector search with HNSW indexing
3. Build hybrid scoring function (0.5 BM25 + 0.5 semantic)
4. Implement Cypher queries for 2-hop graph traversal in Neo4j
5. Design prompt templates for contract analysis
6. Integrate OpenAI GPT-4o API with retry logic
7. Build citation parser to extract page/section references

8. Create query performance benchmarks (¡3 second latency)

Success Criteria:

- Query response time ¡3 seconds at 95th percentile
- Retrieval recall ¿90% on test queries
- LLM analysis accuracy ¿80% (expert-validated)
- Citation accuracy ¿95% (correct page/section attribution)

8.4 Phase 4: User Interface and Workflow (Months 10-12)

Objective: Build professional web application for legal users

Deliverables:

- React + TypeScript web application
- Contract upload interface with drag-and-drop
- Processing status tracking and notifications
- Semantic search interface with natural language input
- Knowledge graph visualization (D3.js/Cytoscape.js)
- Analysis dashboard with risk indicators
- User authentication and access control

Key Tasks:

1. Build React app with Material-UI components
2. Implement contract upload with progress tracking
3. Create search interface with query suggestions
4. Build results viewer with highlighted contract sections
5. Implement interactive KG visualization with filtering
6. Design analysis dashboard with clause coverage matrix
7. Set up OAuth 2.0 authentication with JWT tokens
8. Conduct usability testing with 5-10 lawyers
9. Iterate on UI/UX based on feedback

Success Criteria:

- User satisfaction ¿4.0/5.0 in usability tests
- Contract upload success rate ¿99%
- Interface load time ¡2 seconds
- Support 50+ concurrent users without degradation

8.5 Phase 5: Production Deployment and Validation (Months 13-15)

Objective: Launch production system with legal team adoption

Deliverables:

- Production-grade infrastructure with monitoring
- Security hardening and compliance (SOC 2, attorney-client privilege)
- User training materials and documentation
- Beta launch with pilot legal team (5-10 lawyers)
- Performance optimization based on real usage
- Feedback collection and iterative improvement

Key Tasks:

1. Deploy production Kubernetes cluster with autoscaling
2. Implement end-to-end encryption for contracts
3. Set up audit logging for all user actions
4. Configure Prometheus + Grafana monitoring dashboards
5. Create user documentation (tutorials, FAQs, video guides)
6. Conduct security audit and penetration testing
7. Train pilot legal team (4-hour workshop)
8. Launch beta with 100-200 contracts
9. Collect weekly feedback surveys
10. Optimize based on real-world performance data

Success Criteria:

- System uptime ¿99.5% in first 3 months
- Beta user adoption ¿70% (using weekly)
- Time savings ¿50% vs. manual review (user-reported)
- Cost per contract ¡$50 (vs. thousands for manual review)
- Zero critical security incidents

9 Requirements Specification

9.1 Functional Requirements

FR1: Contract Upload and Parsing

- *Description:* System shall accept PDF and DOCX contract uploads
- *Input:* Contract file ¡100MB
- *Output:* Parsed text with structure preservation
- *Priority:* P0 (Must Have)
- *Acceptance:* 99% successful parse rate on CUAD-like contracts

FR2: KGGEN Extraction Pipeline

- *Description:* Extract entities, relations, aggregate, and resolve duplicates
- *Input:* Parsed contract text
- *Output:* Knowledge graph with confidence scores
- *Priority:* P0
- *Acceptance:* AUPR ¿45% on CUAD benchmark

FR3: Knowledge Graph Storage

- *Description:* Store and index KG in Neo4j with properties
- *Input:* Extracted triples
- *Output:* Queryable graph database
- *Priority:* P0
- *Acceptance:* Support 1M+ nodes, 5M+ edges

FR4: Semantic Search

- *Description:* Natural language query against KG using hybrid retrieval
- *Input:* User query string
- *Output:* Top-k relevant triples with scores
- *Priority:* P0
- *Acceptance:* Retrieval recall ¿90%, latency ¡3 seconds

FR5: Multi-Hop Retrieval

- *Description:* Expand context via 2-hop graph traversal
- *Input:* Retrieved triples

- *Output:* Expanded triple set with context
- *Priority:* P1 (Should Have)
- *Acceptance:* Retrieve contextually relevant neighbors

FR6: LLM-Based Analysis

- *Description:* Synthesize natural language analysis from KG context
- *Input:* Query + retrieved triples + source text
- *Output:* Analysis with citations
- *Priority:* P0
- *Acceptance:* Analysis accuracy ¿80%, citation accuracy ¿95%

FR7: Results Visualization

- *Description:* Display results with highlighted contract sections and KG graph
- *Input:* Analysis results
- *Output:* Formatted UI with contract viewer and graph
- *Priority:* P1
- *Acceptance:* User satisfaction ¿4.0/5.0

9.2 Non-Functional Requirements

NFR1: Extraction Accuracy

- *Metric:* Precision @ 80% Recall, AUPR
- *Target:* Precision ¿50%, AUPR ¿45%
- *Measurement:* CUAD benchmark test set

NFR2: Query Response Time

- *Metric:* 95th percentile query latency
- *Target:* ¡3 seconds from query to results
- *Measurement:* Application performance monitoring

NFR3: System Availability

- *Metric:* Uptime percentage
- *Target:* ¿99.5% availability
- *Measurement:* Monthly uptime reports

NFR4: Scalability

- *Metric:* Contracts processed, concurrent users
- *Target:* Support 10,000+ contracts, 50+ concurrent users
- *Measurement:* Load testing, production metrics

NFR5: Security and Privacy

- *Requirements:* End-to-end encryption, access control, audit logs
- *Standards:* SOC 2 Type II, GDPR compliance
- *Validation:* Security audit, penetration testing

NFR6: API Rate Limiting

- *Limits:* 1000 requests/hour per user, 100 uploads/day
- *Purpose:* Prevent abuse, control costs
- *Implementation:* Redis-based rate limiter

10 Risk Assessment and Mitigation

10.1 Technical Risks

Risk 1: Extraction Accuracy for Complex Clauses
Description: Legal contracts contain complex, nested clauses with domain-specific language. LLM extraction may miss nuanced terms or misclassify clause types.
Likelihood: High *Impact:* High
Mitigation Strategies:

- Use few-shot learning with CUAD examples to train extractors
- Implement confidence scoring to flag low-confidence extractions
- Deploy human-in-the-loop validation for critical clause types
- Continuously retrain models on validated corrections
- Maintain accuracy dashboard tracking per-label performance

Risk 2: Entity Resolution Errors
Description: Entity resolution may incorrectly merge distinct entities (false positives) or fail to merge duplicates (false negatives), leading to incomplete or incorrect KG structure.
Likelihood: Medium *Impact:* Medium
Mitigation Strategies:

- Tune clustering thresholds (k=128) and top-k retrieval (k=16) based on validation
- Implement conservative LLM deduplication with explicit instructions
- Allow manual override to split/merge entities in UI
- Track entity resolution metrics (deduplication ratio, precision/recall)

- Conduct periodic audits of canonical entity mappings

Risk 3: LLM Hallucination
Description: LLM may generate plausible but incorrect analysis not grounded in retrieved KG facts, misleading legal professionals.
Likelihood: Medium *Impact:* Critical
Mitigation Strategies:

- Use low temperature (0.1) for deterministic outputs
- Prompt LLM to cite specific contract sections and page numbers
- Validate all citations against source documents automatically
- Display confidence indicators in UI ("definite", "likely", "possible")
- Include disclaimers that analysis must be verified by legal counsel
- Implement feedback mechanism to flag inaccurate analyses

Risk 4: Scalability Challenges
Description: System may not scale to 10,000+ contracts or 50+ concurrent users, leading to slow performance or outages.
Likelihood: Low *Impact:* High
Mitigation Strategies:

- Deploy on Kubernetes with horizontal autoscaling
- Implement caching for frequently accessed contracts and queries
- Use asynchronous processing for contract extraction (queue-based)
- Conduct load testing before production launch
- Monitor performance metrics and scale proactively

10.2 Legal and Compliance Risks

Risk 5: Unauthorized Practice of Law
Description: Providing automated legal analysis may constitute unauthorized practice of law, violating state bar regulations.
Likelihood: Medium *Impact:* Critical
Mitigation Strategies:

- Position system as analysis tool, not legal advice
- Display prominent disclaimers on all outputs
- Require all analyses to be reviewed by licensed attorneys
- Consult with legal ethics experts before launch
- Limit access to licensed legal professionals and their authorized staff

Risk 6: Liability for Incorrect Analysis

Description: Users may rely on incorrect system outputs, leading to adverse legal consequences and potential liability.

Likelihood: Medium *Impact:* Critical

Mitigation Strategies:

- Obtain professional liability insurance
- Include comprehensive Terms of Service limiting liability
- Require user acknowledgment that system outputs must be verified
- Track and log all system outputs for audit trails
- Establish incident response process for reported errors

Risk 7: Data Privacy and Attorney-Client Privilege

Description: Contracts contain confidential information protected by attorney-client privilege. Data breaches or improper access could violate privilege and expose organizations to liability.

Likelihood: Low *Impact:* Critical

Mitigation Strategies:

- Implement end-to-end encryption for all contracts at rest and in transit
- Deploy strict access controls with organization-level isolation
- Maintain audit logs of all data access
- Obtain SOC 2 Type II certification
- Ensure compliance with GDPR, CCPA, and other privacy regulations
- Conduct annual security audits and penetration testing

10.3 Operational Risks

Risk 8: User Adoption

Description: Legal professionals may resist adopting AI-powered tools due to skepticism, unfamiliarity, or preference for traditional methods.

Likelihood: Medium *Impact:* High

Mitigation Strategies:

- Involve lawyers in design and validation from start
- Conduct user research to understand pain points
- Provide comprehensive training and onboarding
- Start with pilot program to build champions
- Demonstrate time savings and accuracy benefits with metrics
- Iterate based on user feedback

Risk 9: Training and Support Requirements

Description: Users may require significant training and ongoing support, increasing operational burden.

Likelihood: Medium *Impact:* Medium

Mitigation Strategies:

- Develop intuitive UI requiring minimal training
- Create self-service documentation (tutorials, FAQs, videos)
- Implement in-app tooltips and guided walkthroughs
- Establish support channels (email, chat, ticketing)
- Track common issues and improve UI/documentation

Risk 10: Cost Management

Description: LLM API costs (Gemini, GPT-4o) and infrastructure costs may exceed budget, especially as usage scales.

Likelihood: Medium *Impact:* Medium

Mitigation Strategies:

- Use cost-effective Gemini 2.0 Flash for extraction
- Implement caching to avoid redundant LLM calls
- Set rate limits per user to control usage
- Monitor costs daily and set budget alerts
- Optimize prompts to minimize token usage
- Explore self-hosted models for cost-sensitive operations

11 Evaluation and Validation

11.1 CUAD Benchmark Performance

The system will be evaluated on the CUAD test set to measure extraction quality.

Metrics:

- **AUPR (Area Under Precision-Recall Curve):** Summarizes performance across confidence thresholds
- **Precision @ 80% Recall:** Measures precision when system achieves 80% recall
- **Precision @ 90% Recall:** Measures precision at higher recall threshold

Baselines:

- BERT-base: 32.4% AUPR, 8.2% Precision @ 80% Recall [5]
- DeBERTa-xlarge: 47.8% AUPR, 44.0% Precision @ 80% Recall [5]
- KGGen: 66.07% on MINE-1 benchmark [8]

Target Performance:

- AUPR ¿45% (approaching DeBERTa baseline)
- Precision @ 80% Recall ¿50% (exceeding DeBERTa)
- Precision @ 90% Recall ¿20% (for high-recall use cases)

Evaluation Protocol:

1. Process 102 contracts from CUAD test set (20% holdout)
2. Extract all 41 CUAD label categories
3. Compute precision, recall, and F1 for each label
4. Aggregate to overall AUPR and Precision @ Recall metrics
5. Analyze per-category performance to identify weaknesses

11.2 Legal Expert Validation

Expert validation ensures system outputs are accurate and useful in legal practice.

Validation Protocol:

1. Select 50 technology agreements (25 from CUAD, 25 new contracts)
2. Process through KGGEN extraction pipeline
3. Present extractions to 3 independent legal experts
4. Experts label each extraction as: Correct, Partially Correct, Incorrect
5. Compute inter-annotator agreement (Fleiss' kappa)
6. Calculate accuracy: (Correct + 0.5*Partially Correct) / Total

Target Metrics:

- Inter-annotator agreement: Fleiss' kappa ¿0.7 (substantial agreement)
- Extraction accuracy: ¿85% for critical clauses (IP ownership, liability, termination)
- Extraction accuracy: ¿75% for all clause types

Critical Clause Types: Focus validation on high-impact clauses:

- IP Ownership Assignment
- Liability provisions (capped/uncapped)
- Non-compete and exclusivity
- License scope and restrictions
- Termination conditions

11.3 User Acceptance Testing

UAT measures system usability and real-world effectiveness.

Test Scenarios:

1. **Contract Upload:** User uploads 5 contracts, verifies successful processing
2. **Clause Search:** User searches for specific clause types across contract portfolio
3. **Risk Identification:** User identifies high-risk provisions (uncapped liability, perpetual terms)
4. **Comparative Analysis:** User compares terms across multiple contracts
5. **LLM Analysis:** User asks complex questions requiring multi-hop reasoning

Participant Profile:

- 5-10 lawyers (3 junior associates, 3 mid-level, 2 senior)
- Mix of in-house counsel and law firm attorneys
- Experience with technology agreements

Evaluation Metrics:

- **Task Completion Rate:** % of scenarios completed successfully
- **Time on Task:** Minutes to complete each scenario
- **Error Rate:** % of tasks with errors or user confusion
- **System Usability Scale (SUS):** Standardized usability questionnaire (target ¿70)
- **Net Promoter Score (NPS):** Likelihood to recommend (target ¿40)

Target Performance:

- Task completion rate ¿90%
- Time savings ¿50% vs. manual review
- SUS score ¿70 (above average usability)
- User satisfaction ¿4.0/5.0

11.4 Continuous Improvement

Performance Monitoring:

- Track extraction accuracy weekly on new contracts
- Monitor query latency and system uptime
- Collect user feedback after each session
- Log all human corrections for model retraining

Model Retraining:

- Quarterly retraining on accumulated corrections
- A/B test new models against production baseline
- Deploy improved models if ¿5% accuracy gain

Feature Enhancement:

- Prioritize features based on user requests
- Add support for new contract types based on demand
- Expand CUAD ontology with user-suggested labels

12 Conclusion

This Product Requirements Document specifies a comprehensive knowledge graph-based legal contract analysis system that applies the KGGEN methodology to the CUAD dataset. By combining state-of-the-art text-to-knowledge-graph extraction with hybrid retrieval and LLM-powered analysis, the system addresses a critical pain point in legal practice: the time-consuming, expensive task of contract review.

12.1 Key Innovations

KGGEN Pipeline for Legal Contracts: Adapts the proven KGGEN three-stage extraction pipeline (entity extraction, aggregation, entity resolution) to legal domain, achieving superior entity deduplication and relation generalization compared to existing methods like OpenIE and GraphRAG [8].

CUAD Ontology Mapping: Implements a comprehensive 41-category ontology based on expert-annotated CUAD dataset, covering general information, restrictive covenants, and revenue risks with specific extensions for technology agreements [5].

Hybrid Retrieval Mechanism: Combines BM25 keyword matching and semantic similarity to achieve robust retrieval, enhanced with multi-hop graph traversal for contextual expansion. This approach ensures high recall while maintaining precision in legal clause retrieval.

LLM-Grounded Analysis: Leverages GPT-4o to synthesize natural language analysis grounded in retrieved knowledge graph facts, with automatic citation extraction and validation to prevent hallucination.

12.2 Expected Impact

For legal professionals, this system promises to:

- Reduce contract review time by 50-70%
- Lower transaction costs from hundreds of thousands to tens of thousands of dollars
- Democratize access to legal analysis for small businesses and individuals
- Enable semantic search across contract portfolios
- Identify risks and obligations without manual document review

For the engineering team, the PRD provides:

- Clear technical specifications for KGGEN implementation
- Comprehensive ontology mapping for knowledge graph schema
- Detailed architecture for multi-tier system deployment
- Phased roadmap with measurable success criteria
- Risk assessment and mitigation strategies

12.3 Next Steps

The engineering team should:

1. Review this PRD with legal stakeholders for validation and feedback
2. Finalize technology stack selections (Neo4j vs. alternatives, cloud provider)
3. Begin Phase 1 infrastructure development
4. Establish development sprints and agile processes
5. Set up monitoring and evaluation frameworks
6. Schedule regular check-ins with legal experts for continuous validation

The legal team should:

1. Validate CUAD ontology mapping for technology agreement coverage
2. Provide sample contracts for pilot testing
3. Participate in user acceptance testing
4. Define critical clause types requiring highest accuracy
5. Establish human-in-the-loop review protocols

By executing this roadmap, the team will deliver a production-ready knowledge graph system that transforms legal contract review from a tedious, manual process into an efficient, AI-assisted workflow, making legal support more accessible and affordable while maintaining the quality and accuracy expected in professional legal practice.

References

[1] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354. Association for Computational Linguistics, 2015.

[2] CEB. Advance your contract management process. https://web.archive.org/web/20170920135124/https://www.cebglobal.com/compliance-legal/smb-legal/contract-management-midsized.html, 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

[4] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021.

[5] Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. Cuad: An expert-annotated nlp dataset for legal contract review. In *35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks*. NeurIPS, 2021.

[6] Jonathan Larson and Steven Truitt. Graphrag: Unlocking llm discovery on narrative private data. *Microsoft Research Blog*, 2024.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[8] Belinda Mo, Kyssen Yu, Joshua Kazdan, Joan Cabezas, Proud Mpala, Lisa Yu, Chris Cundy, Charilaos Kanatsoulis, and Sanmi Koyejo. Kggen: Extracting knowledge graphs from plain text with language models. In *39th Conference on Neural Information Processing Systems (NeurIPS 2025)*. NeurIPS, 2025.