Contract Review & Risk Analysis System

Comprehensive Methodology & Implementation Guide

Author: Mohammad Babaie
Email: mj.babaie@gmail.com
LinkedIn: https://www.linkedin.com/in/mohammadbabaie/
GitHub: https://github.com/Muh76

Project Overview

Production-ready legal AI system combining CUAD dataset, risk scoring, RAG, and MLOps for maximum employment impact.

Core Value Proposition

• **Risk-First Approach:** Prioritize risk scoring over pure classification

• **Actionable Intelligence:** Provide specific recommendations, not just analysis

• **Production Ready:** Full MLOps pipeline with monitoring and deployment

• **Business Focus:** ROI metrics and compliance tracking

Market Opportunity

• **Legal Tech Market:** $25B+ growing at 15% CAGR

• **Contract Review:** $3B+ segment with 80% manual processes

• **AI Adoption:** 60% of law firms planning AI investment in next 2 years

Competitive Advantages

1. **Real Dataset:** CUAD v1 with 510 contracts and 13,000+ labels
2. **Risk Quantification:** Numerical risk scores with business impact
3. **RAG Integration:** Precedent analysis and alternative suggestions
4. **Production Deployment:** Full-stack solution, not just research

Implementation Phases

Phase 1: MVP Foundation (2 weeks)

Goal: Working demo with core risk analysis

Week 1: Core Analysis Engine

- ■ Set up development environment
- ■ Implement CUAD data preprocessing
- ■ Build basic clause extraction model
- ■ Create risk scoring algorithm
- ■ Develop clause highlighting system

Week 2: User Interface & RAG

- ■ Build Streamlit dashboard
- ■ Implement vector database (ChromaDB)
- ■ Add similar clause retrieval
- ■ Integrate SHAP explanations
- ■ Deploy MVP to Streamlit Cloud

Phase 2: Production Features (1 month)

Goal: Production-ready API with MLOps

Week 3-4: Backend & API

- ■ Develop FastAPI backend
- ■ Implement authentication system
- ■ Add file upload/processing
- ■ Create RESTful endpoints
- ■ Add request/response validation

Week 5-6: MLOps & Monitoring

- ■ Set up MLflow tracking
- ■ Implement model versioning
- ■ Add performance monitoring
- ■ Create automated retraining pipeline
- ■ Set up alerting system

Phase 3: Advanced Features (2 months)

Goal: Enterprise-ready solution

Month 2: Innovation & RAG

■ Advanced RAG with LLM integration

■ Multi-modal document processing

■ Alternative clause suggestions

■ Precedent analysis system

■ Risk trend analysis

Month 3: Scale & Polish

■ Cloud deployment (AWS/GCP)

■ Load testing & optimization

■ Security hardening

■ Comprehensive documentation

■ Demo preparation

Core Methodologies

Multi-Task Learning Model

```
class LegalContractModel(nn.Module): def __init__(self, num_categories=41):
super().__init__() self.encoder = AutoModel.from_pretrained("roberta-base")
self.classifiers = nn.ModuleDict({ 'binary': nn.Linear(768, 2), # Yes/No categories
'extractive': nn.Linear(768, 2), # Span extraction 'regression': nn.Linear(768, 1)
# Dates, amounts }) def forward(self, input_ids, attention_mask, task_type):
outputs = self.encoder(input_ids, attention_mask) return
self.classifiers[task_type](outputs.pooler_output)
```

Risk Scoring Algorithm

```
class RiskScoringEngine: def __init__(self): self.risk_weights = {
'uncapped_liability': 0.25, 'non_compete': 0.20, 'ip_assignment': 0.15,
'termination_convenience': 0.10, 'audit_rights': 0.05 } def
calculate_risk_score(self, extracted_clauses): risk_score = 0 for clause_type,
weight in self.risk_weights.items(): if clause_type in extracted_clauses:
risk_score += weight * self._clause_risk_value(extracted_clauses[clause_type])
return min(risk_score, 1.0)
```

RAG Implementation

Retrieval-Augmented Generation for precedent analysis and alternative clause suggestions

```
class LegalRAGSystem: def __init__(self): self.embedder =
SentenceTransformer('all-MiniLM-L6-v2') self.vector_db = chromadb.Client()
self.collection = self.vector_db.create_collection("legal_clauses") self.llm =
openai.OpenAI() def find_similar_clauses(self, query_clause, clause_type):
query_embedding = self.embedder.encode(query_clause) results =
self.collection.query( query_embeddings=[query_embedding], n_results=5,
where={'clause_type': clause_type} ) return results def
suggest_alternative_wording(self, risky_clause, clause_type): similar_clauses =
self.find_similar_clauses(risky_clause, clause_type) # Generate safer alternative
wording return self.llm.generate_suggestion(risky_clause, similar_clauses)
```

Business Metrics & ROI

Key Performance Indicators

• **Time Savings:** 80% reduction in contract review time

• **Cost Reduction:** $500-2000 per contract reviewed

• **Risk Mitigation:** 95% detection rate for critical clauses

• **Scalability:** Process 1000+ contracts/day

• **User Satisfaction:** >90% positive feedback

Employment Strategy

Target Companies

1. **Legal Tech Startups:** DocuSign, LegalZoom, Clio
2. **Law Firms:** Big Law firms with tech initiatives
3. **Enterprise:** Fortune 500 companies with legal departments
4. **Consulting:** McKinsey, BCG, Deloitte
5. **Tech Companies:** Google, Microsoft, Amazon

Success Metrics

Technical Success Metrics

• **Model Accuracy:** >85% on CUAD test set
• **Risk Scoring Correlation:** >0.8 with expert assessment
• **API Response Time:** <2 seconds
• **System Uptime:** >99.5%
• **Code Coverage:** >90%

Employment Success Metrics

• **Portfolio Visits:** 10x more than typical projects
• **Interview Requests:** 5x higher response rate
• **Salary Negotiation:** 20-30% higher offers
• **Role Level:** Senior/Lead positions
• **Company Tier:** Top-tier companies

Key Success Factors

1. **Focus on Risk:** Prioritize risk scoring over pure classification
2. **RAG Integration:** Add precedent analysis and suggestions
3. **Production Ready:** Full MLOps pipeline with monitoring
4. **Business Focus:** ROI metrics and compliance tracking
5. **Portfolio Quality:** Polished demo and documentation

Next Steps

1. Set up development environment
2. Begin Phase 1 implementation
3. Track progress weekly
4. Iterate based on feedback
5. Prepare for deployment and demo

This document should be reviewed and updated weekly to track progress and ensure alignment with project goals.