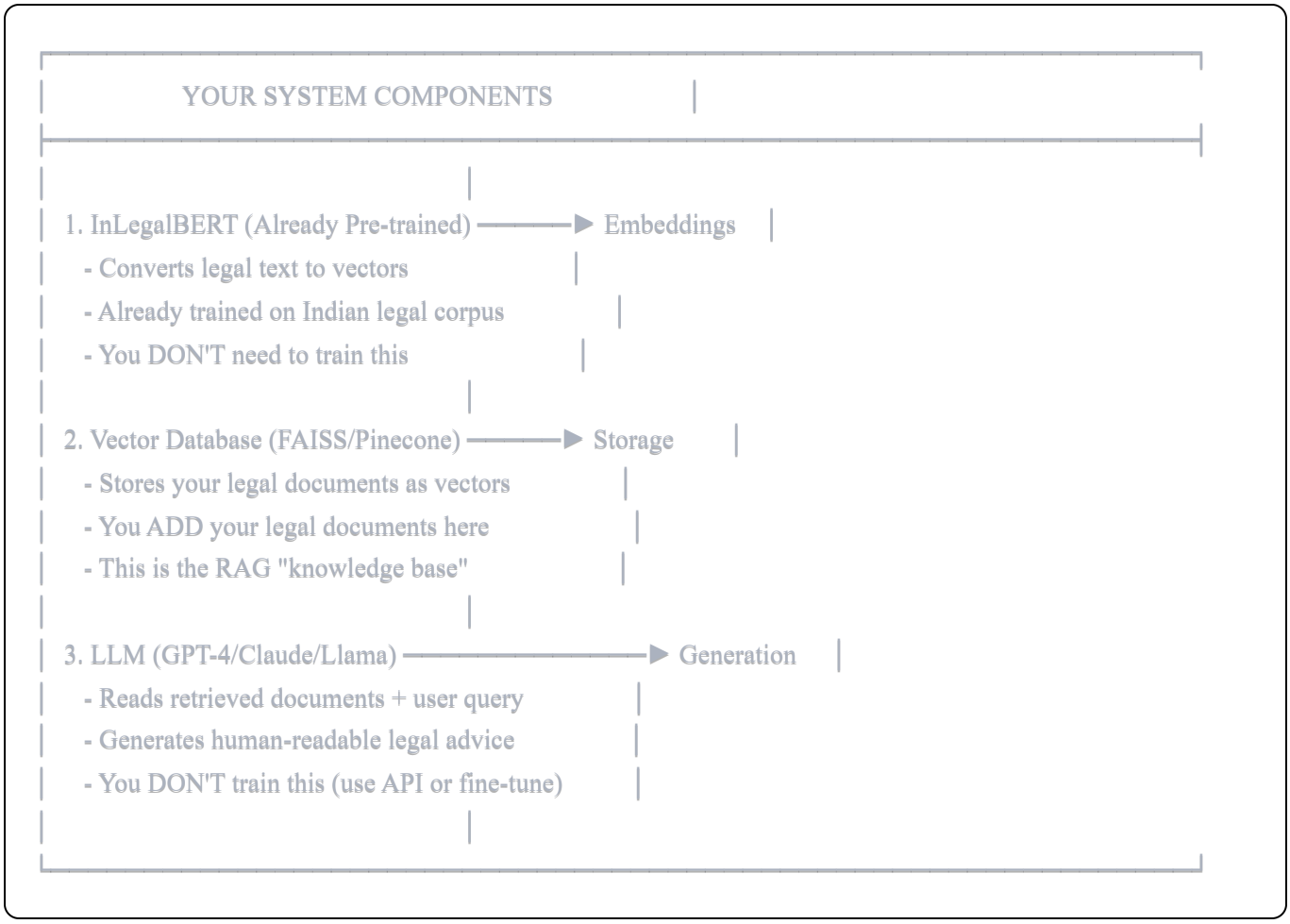
Legal Consultant Bot - Complete Training & Data Strategy

Understanding the Architecture

You are NOT training a model from scratch!

Here's what's actually happening:



How Legal Advice is Generated

Step-by-Step Process:

User asks: "I was fired without notice. What can I do?"

Step 1: Embedding (InLegalBERT)

```
python
```

```
# InLegalBERT converts query to vector
query_vector = inlegal_bert.encode("I was fired without notice")
# Result: [0.234, -0.567, 0.891, ...] (768 numbers)
```

Step 2: Retrieval (RAG Database)

python

```
# Find similar legal documents from your database
retrieved_docs = vector_db.search(query_vector, top_k=5)

# Example retrieved documents:
# 1. "Industrial Disputes Act: Termination requires 30 days notice..."
# 2. "Supreme Court ruling in XYZ vs ABC: Wrongful termination..."
# 3. "Labor law section 25F: Notice period requirements..."
```

Step 3: Generation (LLM)

python

```
# Combine retrieved docs + query and send to LLM
prompt = f"""
Legal Context:
{retrieved_doc_1}
{retrieved_doc_2}
{retrieved_doc_3}

User Question: I was fired without notice. What can I do?

Provide legal advice based on the context above.
"""

advice = llm.generate(prompt)
# LLM reads the documents and generates advice
```

Final Output:

```
Based on Indian labor law (Industrial Disputes Act, Section 25F),
your employer must provide 30 days notice or payment in lieu.
You can:
1. Send legal notice demanding notice pay
2. File complaint with Labor Commissioner
3. Approach Labor Court if unresolved
...
```

What Data Goes in RAG? (CRITICAL SECTION)

Core Legal Documents You Must Add:

1. Bare Acts (Primary Legislation)

python

```
documents = [
    {
        "text": "Section 420 IPC: Whoever cheats and thereby dishonestly induces the person deceived to deliver any pro
        "metadata": {
            "source": "Indian Penal Code",
            "section": "420",
            "type": "bare_act",
            "category": "criminal",
            "enacted": "1860"
        }
    },
    {
        "text": "Section 25F Industrial Disputes Act: No workman shall be retrenched unless one month's notice in writin
        "metadata": {
            "source": "Industrial Disputes Act",
            "section": "25F",
            "type": "bare_act",
            "category": "labor"
        }
    }
]
```

What to include:

- Indian Penal Code (IPC) - All sections
- Code of Criminal Procedure (CrPC)

- Code of Civil Procedure (CPC)
- Indian Evidence Act
- Contract Act 1872
- Consumer Protection Act 2019
- IT Act 2000
- Companies Act 2013
- Industrial Disputes Act
- Labor laws (Wages, Gratuity, etc.)
- Property laws
- Family laws (Hindu Marriage Act, etc.)

2. Court Judgments & Precedents

python

```
{
  "text": "In Vishaka vs State of Rajasthan (1997), the Supreme Court laid down guidelines for sexual harassment at w
  "metadata": {
    "source": "Supreme Court",
    "case_name": "Vishaka vs State of Rajasthan",
    "year": "1997",
    "type": "judgment",
    "citation": "AIR 1997 SC 3011",
    "category": "labor",
    "keywords": ["sexual harassment", "workplace", "guidelines"]
  }
}
```

Where to get judgments:

- Indian Kanoon (indiankanoon.org) - Free case law database
- Supreme Court website
- High Court websites
- LiveLaw, Bar & Bench for recent judgments

3. Legal Procedures & Forms

```python
{
  "text": "To file an FIR (First Information Report): 1) Visit nearest police station 2) Provide written complaint with d
  "metadata": {
    "source": "Procedural Guide",
    "type": "procedure",
    "category": "criminal",
    "topic": "FIR filing"
  }
}
```

4. Legal Definitions & Explanations

```python
{
  "text": "Cognizable Offense: An offense where police can arrest without warrant and start investigation without mag
  "metadata": {
    "source": "Legal Glossary",
    "type": "definition",
    "term": "cognizable offense"
  }
}
```

5. Recent Amendments & Updates

```python
{
  "text": "Consumer Protection Act Amendment 2023: E-commerce platforms now liable for defective products. New
  "metadata": {
    "source": "Consumer Protection Act",
    "type": "amendment",
    "date": "2023-08-15",
    "category": "consumer"
  }
}
```

Data Sources (Where to Get Legal Data)

Free Sources:

1. **Indian Kanoon** (indiankanoon.org)
   - 10M+ judgments
   - API available
   - Free access
2. **India Code** (indiacode.nic.in)
   - All Central Acts
   - Latest amendments
   - Government official
3. **Supreme Court** (sci.gov.in)
   - Recent judgments
   - Daily orders
4. **Legislative Department**
   - Bills and Acts
   - Official gazette

Scraping Example:

```python
```

```python
import requests
from bs4 import BeautifulSoup

def scrape_indian_kanoon(case_url):
    """Scrape case judgment from Indian Kanoon"""
    response = requests.get(case_url)
    soup = BeautifulSoup(response.content, 'html.parser')

    # Extract judgment text
    judgment_div = soup.find('div', class_='judgments')
    text = judgment_div.get_text()

    # Extract metadata
    case_name = soup.find('h1').get_text()
    citation = soup.find('div', class_='citation').get_text()

    return {
        "text": text,
        "metadata": {
            "case_name": case_name,
            "citation": citation,
            "source": "Indian Kanoon"
        }
    }
```

Data Preparation Pipeline

python

```python
import re
from typing import List, Dict

class LegalDataPreprocessor:
    """Prepare raw legal text for RAG"""

    def chunk_document(self, text: str, chunk_size: int = 512) -> List[str]:
        """Split long documents into chunks"""
        # For legal docs, chunk by sections or paragraphs
        sections = re.split(r'\n\n+', text)

        chunks = []
        current_chunk = ""

        for section in sections:
            if len(current_chunk) + len(section) < chunk_size:
                current_chunk += "\n\n" + section
            else:
                if current_chunk:
                    chunks.append(current_chunk.strip())
                current_chunk = section

        if current_chunk:
            chunks.append(current_chunk.strip())

        return chunks

    def clean_legal_text(self, text: str) -> str:
        """Clean and normalize legal text"""
        # Remove extra whitespace
        text = re.sub(r'\s+', ' ', text)

        # Fix common OCR errors in scanned judgments
        text = text.replace('|', 'I')

        # Preserve section numbers
        text = re.sub(r'Section\s+(\d+)', r'Section \1', text)

        return text.strip()

    def extract_ipc_sections(self, text: str) -> List[str]:
        """Extract IPC section numbers mentioned"""
        return re.findall(r'Section\s+(\d+[A-Z]?)\s+(?:of\s+)?IPC', text)
```

```
def prepare_document(self, raw_doc: Dict) -> List[Dict]:
    """Prepare a document for RAG"""
    text = self.clean_legal_text(raw_doc['text'])
    chunks = self.chunk_document(text)

    documents = []
    for i, chunk in enumerate(chunks):
        doc = {
            "text": chunk,
            "metadata": {
                **raw_doc.get('metadata', {}),
                "chunk_id": i,
                "total_chunks": len(chunks),
                "ipc_sections": self.extract_ipc_sections(chunk)
            }
        }
        documents.append(doc)

    return documents
```

Example: Building Your Knowledge Base

python

```python
# Step 1: Collect raw legal documents
raw_documents = []

# Add IPC sections
ipc_sections = load_ipc_from_file("ipc_complete.txt")
raw_documents.extend(ipc_sections)

# Add recent judgments
judgments = scrape_indian_kanoon_recent(limit=1000)
raw_documents.extend(judgments)

# Add consumer protection act
consumer_act = load_act("consumer_protection_act_2019.pdf")
raw_documents.append(consumer_act)

# Step 2: Preprocess and chunk
preprocessor = LegalDataPreprocessor()
prepared_docs = []

for raw_doc in raw_documents:
    prepared = preprocessor.prepare_document(raw_doc)
    prepared_docs.extend(prepared)

print(f"Total documents after chunking: {len(prepared_docs)}")

# Step 3: Add to RAG system
rag_system.add_documents(prepared_docs)
```

What Makes Good RAG Data?

✅ Good Examples:

python

```python
# Specific, actionable, well-cited
{
    "text": "Under Section 138 Negotiable Instruments Act, if a cheque bounces due to insufficient funds, payee must se
    "metadata": {"source": "NI Act", "section": "138"}
}
```

❌ Bad Examples:

python

```
# Too vague, no specifics
{
  "text": "There are laws about cheques and banking.",
  "metadata": {}
}

# Too long, not chunked
{
  "text": "[Entire 50-page judgment without chunking]",
  "metadata": {}
}
```

Sample Data Structure for Different Categories

Criminal Law:

python

```
criminal_docs = [
  {
    "text": "Section 302 IPC: Murder. Whoever commits murder shall be punished with death or imprisonment for lif
    "metadata": {
      "category": "criminal",
      "subcategory": "offences_against_person",
      "section": "302",
      "punishment": "death or life imprisonment"
    }
  }
]
```

Civil Law:

python

```
civil_docs = [
    {
        "text": "Order VII Rule 11 CPC: Rejection of plaint. Plaint may be rejected if it does not disclose cause of action,
        "metadata": {
            "category": "civil",
            "source": "CPC",
            "type": "procedure"
        }
    }
]
```

Consumer Law:

```
python

consumer_docs = [
    {
        "text": "Consumer can file complaint within 2 years from cause of action. District Forum for claims up to ₹1 crore
        "metadata": {
            "category": "consumer",
            "type": "procedure",
            "topic": "filing_complaint"
        }
    }
]
```

Recommended Data Volumes

For a production-ready legal chatbot:

- **Minimum**: 10,000 documents (basic coverage)
- **Good**: 50,000 documents (comprehensive)
- **Excellent**: 100,000+ documents (expert-level)

Breakdown:

- 5,000 - Bare act sections
- 20,000 - Court judgments
- 10,000 - Procedures & guides
- 15,000 - Recent amendments & updates

Which Models to Use (Summary)

You're Using 3 Components:

1. **InLegalBERT** (Embeddings)
   - Pre-trained: YES ✓
   - Training needed: NO
   - Purpose: Convert text to vectors
2. **Vector Database** (Storage)
   - Pre-trained: N/A
   - Training needed: NO
   - Purpose: Store your legal documents
3. **LLM** (Generation) - Choose ONE: **Option A: API-based (Easiest)**
   - GPT-4, Claude, Gemini
   - Pre-trained: YES ✓
   - Training needed: NO
   - Cost: Pay per API call

   Option B: Fine-tuned Open Source (Advanced)
   - Llama 3, Mistral, Gemma
   - Pre-trained: YES ✓
   - Fine-tuning: OPTIONAL (for better legal responses)
   - Cost: One-time training + hosting

Next Steps

1. **Collect Data** (2-4 weeks)
   - Scrape Indian Kanoon
   - Download bare acts
   - Gather recent judgments
2. **Preprocess** (1 week)
   - Clean and chunk documents
   - Extract metadata

- Validate quality

3. **Build RAG** (1 week)
    - Set up InLegalBERT
    - Create FAISS index
    - Add documents
4. **Integrate LLM** (3 days)
    - Choose GPT-4/Claude/Open source
    - Set up API or fine-tune
    - Test responses
5. **Deploy** (1 week)
    - Build web interface
    - Add authentication
    - Monitor performance

Total Time: 5-7 weeks