LegalDocNLP: Contract Review and Legal Document Analysis Using NLP

By- Kartik Vijay Badkas, Satheesh M K, Sarvesh Kulkarni, Guzzu Aditya, Saket Pitale

BITS Pilani - MBA in Business Analytics

Problem Statement & Objective

Problem Statement

- Legal contracts complex, dense, hard to analyze
- Manual review slow, error-prone, inconsistent
- Risky clauses and obligations difficult to track

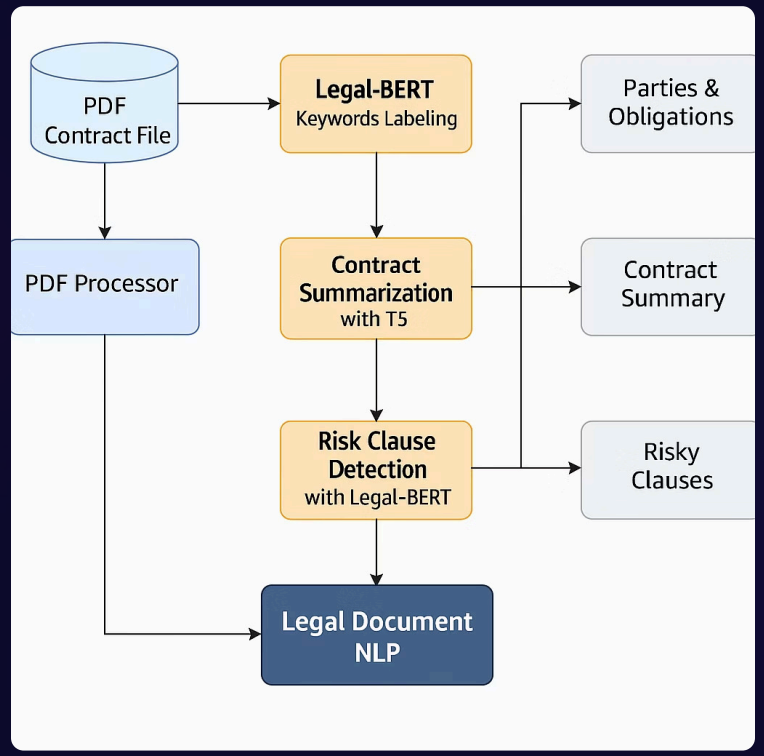
Objective

- Automate clause classification using Legal-BERT
- Summarize clauses with LegalT5
- Visualize risky clauses and keyword contexts
- Develop intuitive Flask-based web interface

System Architecture

Core Components

- PDF Input → Text Extraction
- Sentence Segmentation (NLTK)
- Clause Classification (Legal-BERT)
- Clause Labeling (Rule-based)
- Summarization (LegalT5)
- Keyword Context Extraction
- Flask Web Interface



Models Used & NLP Workflow

Legal-BERT

Fine-tuned on legal corpora

Binary classification: risky vs. not risky

LegalT5

Text-to-text model for clause summarization

NLP Pipeline

- Tokenization
- Classification
- Labeling
- Summarization

```
# Load models and tokenizers
summarizer_tokenizer = T5Tokenizer.from_pretrained( pretrained_model_name_or_path: "SEBIS/legal_t5_small_summ_en", use_fast=False)
summarizer_model = T5ForConditionalGeneration.from_pretrained("SEBIS/legal_t5_small_summ_en")

bert_tokenizer = BertTokenizer.from_pretrained("nlpaueb/legal-bert-base-uncased")
bert_model = BertForSequenceClassification.from_pretrained( pretrained_model_name_or_path: "nlpaueb/legal-bert-base-uncased", num_labels=2)
```

```
from transformers import T5Tokenizer, T5ForConditionalGeneration
from transformers import BertTokenizer, BertForSequenceClassification
import torch
import nltk
import fitz  # PyMuPDF

nltk.download('punkt')

# Load models and tokenizers
summarizer_tokenizer = T5Tokenizer.from_pretrained( pretrained_model_name_or_path: "SEBIS/legal_t5_small_summ_en", use_fast=False)
summarizer_model = T5ForConditionalGeneration.from_pretrained("SEBIS/legal_t5_small_summ_en")
```

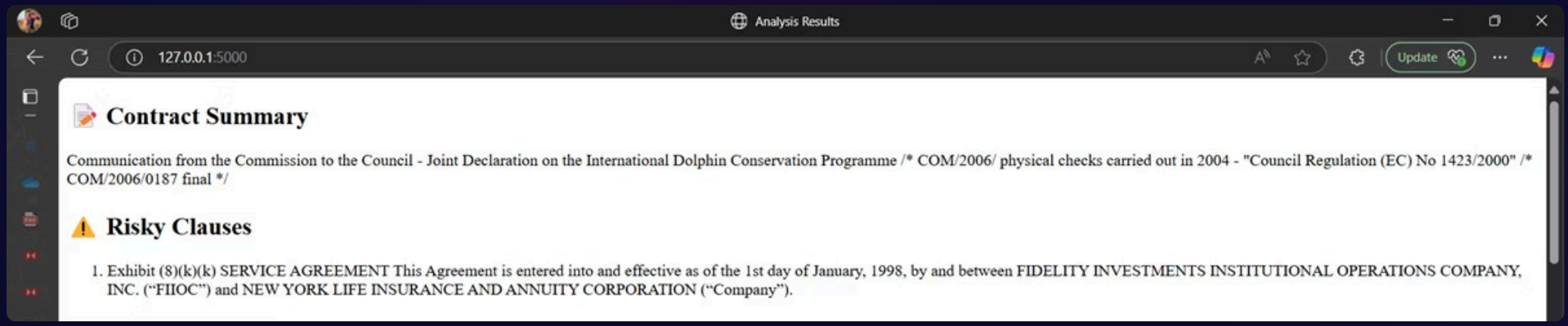
Clause Labeling & Keyword Contexts

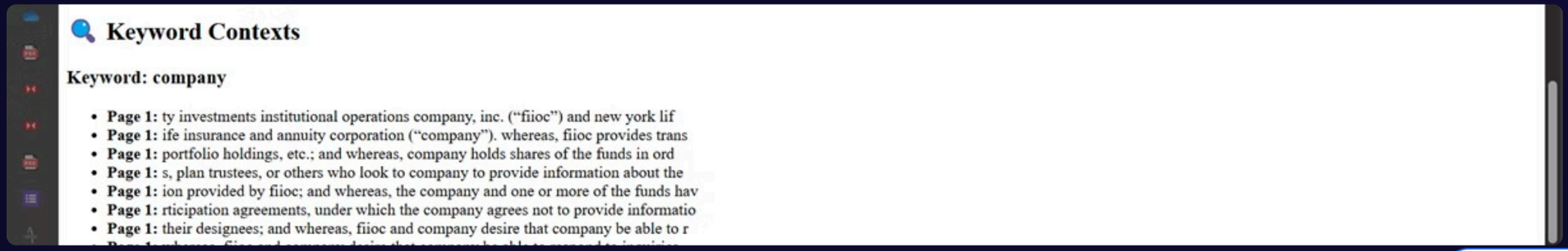
Clause Labeling

- “termination” → Termination
- “indemnity” → Indemnification
- “confidential” → Confidentiality
- “jurisdiction” → Governing Law

Keyword Context Extraction

- Top keywords detected
- Context window for each occurrence





```python
from flask import Flask, request, render_template, redirect, url_for
import os
from werkzeug.utils import secure_filename
from pdf_processor import extract_keywords_contexts
import sys
sys.path.append(os.path.join(os.path.dirname(__file__), "scripts"))

from summarize_contract import generate

app = Flask(__name__)
UPLOAD_FOLDER = 'uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```

Implementation Stack

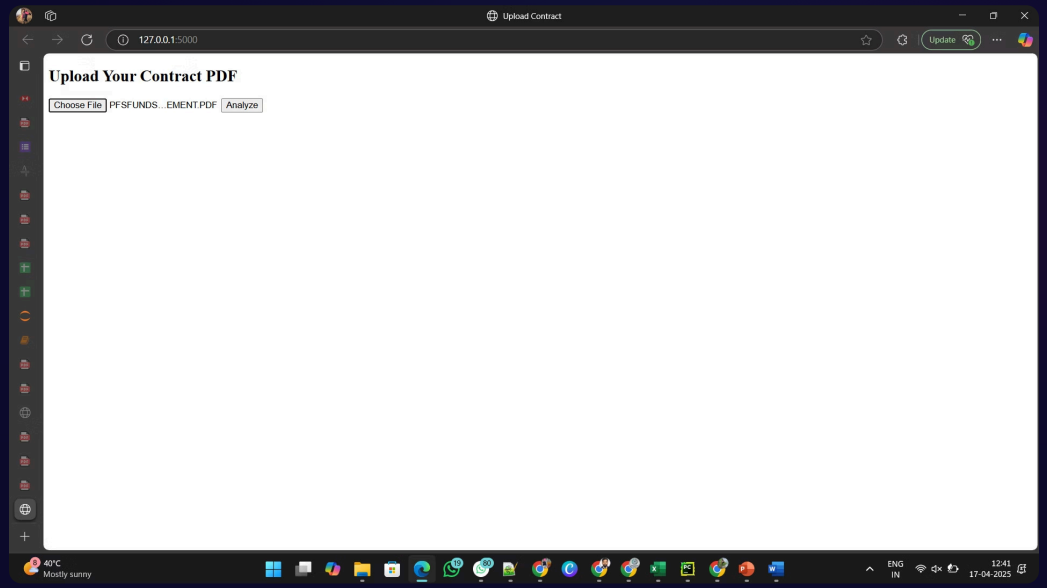
Tools & Technologies

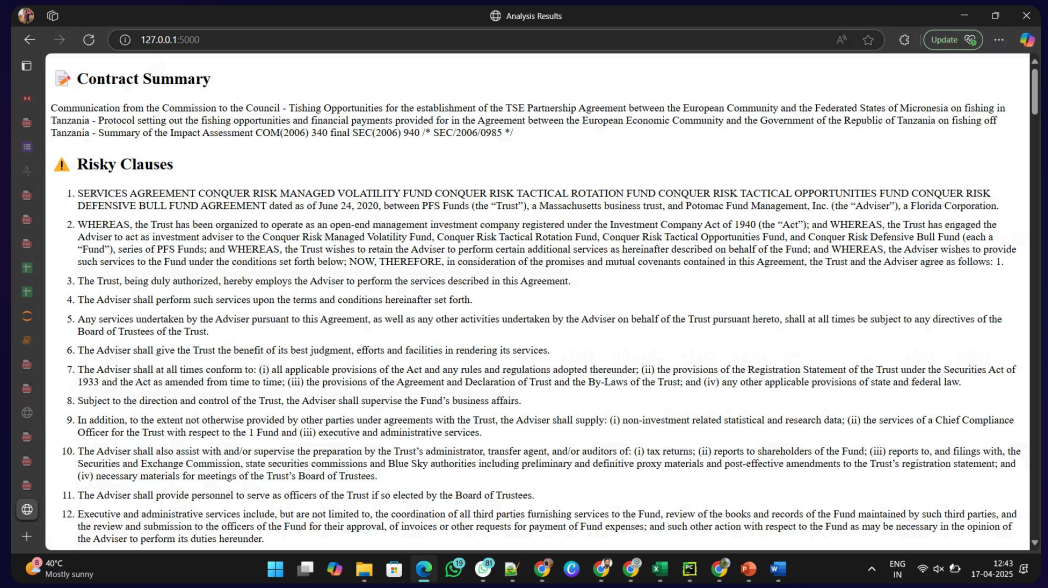
- Python 3.10+
- Flask for UI
- HuggingFace Transformers
- PyMuPDF for PDF parsing
- NLTK for tokenization
- Torch for inference

Backend Flow

- Upload PDF → Process Text
- Extract information → Render frontend

Results & Output Showcase





Conclusion & Future Work

1

Conclusion

- Automates contract comprehension
- Accelerates legal workflows
- Supports compliance and audits

2

Future Work

- OCR for scanned contracts
- Clause severity scoring
- Multi-language support
- Clause comparison views
- Legal clause database integration

Thank you for your attention!