

```
src > pages > DocumentPage.jsx
1 import React, { useState, useEffect } from 'react';
2 import { useLocation, useNavigate } from 'react-router-dom';
3 import PdfViewer from '../components/PdfViewer';
4 import SummaryBox from '../components/SummaryBox';
5 import QnABox from '../components/QnABox';
6 import { extractTextFromPdf, generateSummary, askQuestion } from '../services/googleCloudService';
7
8 const DocumentPage = () => {
9   const location = useLocation();
10   const navigate = useNavigate();
11   const [file, setFile] = useState(null);
12   const [extractedText, setExtractedText] = useState('');
13   const [summary, setSummary] = useState('');
14   const [isLoadingSummary, setIsLoadingSummary] = useState(false);
15   const [error, setError] = useState('');
16
17   useEffect(() => {
18     // Debug log
19     console.log('File in location.state:', location.state?.file);
20     if (location.state?.file) {
21       setFile(location.state.file);
22       processDocument(location.state.file);
23     } else {
24       // Try to restore from localStorage
25       const base64 = localStorage.getItem('uploadedPdfBase64');
26       const name = localStorage.getItem('uploadedPdfName');
27       const size = localStorage.getItem('uploadedPdfSize');
28       if (base64 && name && size) {
29         // Convert base64 back to Blob
30         const byteString = atob(base64.split('').join('') || base64);
31         const ab = new ArrayBuffer(byteString.length);
32         const ia = new Uint8Array(ab);
33         for (let i = 0; i < byteString.length; i++) {
34           ia[i] = byteString.charCodeAt(i);
35         }
36         const blob = new Blob([ab], { type: 'application/pdf' });
37         blob.name = name;
38         blob.size = size;
39         setFile(blob);
40       }
41     }
42   }, []);
43
44   const processDocument = (file) => {
45     extractTextFromPdf(file)
46       .then((text) => {
47         setExtractedText(text);
48         generateSummary(text)
49           .then((summary) => {
50             setSummary(summary);
51             askQuestion(summary, text)
52               .then((questions) => {
53                 // Handle questions
54               })
55               .catch((error) => {
56                 setError(error);
57             });
58           })
59           .catch((error) => {
60             setError(error);
61           });
62       })
63       .catch((error) => {
64         setError(error);
65       });
66   };
67
68   return (
69     <div>
70       <PdfViewer file={file} />
71       <SummaryBox text={extractedText} summary={summary} />
72       <QnABox text={extractedText} />
73     </div>
74   );
75 }
```

```
src > pages > LandingPage.jsx > ...
1 import React, { useState } from 'react';
2 import { useNavigate } from 'react-router-dom';
3
4 const LandingPage = () => {
5   const [selectedFile, setSelectedFile] = useState(null);
6   const [isUploading, setIsUploading] = useState(false);
7   const navigate = useNavigate();
8
9   const handleFileSelect = (event) => {
10     const file = event.target.files[0];
11     if (file && file.type === 'application/pdf') {
12       setSelectedFile(file);
13     } else {
14       alert('Please select a PDF file.');
```

```
PdfWorker.js pdf.min.mjs pdf.worker.min.mjs pdf.worker.mjs.map DocumentPage.jsx Document.d.ts App.jsx
src > App.jsx > ...
1 import React from 'react';
2 import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
3 import LandingPage from './pages/LandingPage';
4 import DocumentPage from './pages/DocumentPage';
5 import './App.css';
6
7 function App() {
8   return (
9     <Router>
10       <div className="App min-h-screen bg-gray-50">
11         <Routes>
12           <Route path="/" element={<LandingPage />} />
13           <Route path="/document" element={<DocumentPage />} />
14         </Routes>
15
16         { /* Disclaimer */ }
17         <footer className="fixed bottom-0 left-0 right-0 bg-yellow-50 border-t border-yellow-200 p-2">
18           <p className="text-center text-sm text-yellow-800">
19             ⚠ This is an AI prototype. Not legal advice. Please consult a qualified attorney for legal matters.
20           </p>
21         </footer>
22       </div>
23     </Router>
24   );
25 }
26
27 export default App;
28
```

```
.env.example .env.local googleCloudService.js requirements.md QnABox.jsx SummaryBox.jsx PdfViewer.jsx
src > components > PdfViewer.jsx > ...
1 import React, { useState, useEffect, useRef, useLayoutEffect } from 'react';
2 import { Document, Page, pdfjs } from 'react-pdf';
3 pdfjs.GlobalWorkerOptions.workerSrc = '/pdf.worker.min.js';
4
5 const PdfViewer = ({ file }) => {
6   const [numPages, setNumPages] = useState(null);
7   const [pageNumber, setPageNumber] = useState(1);
8   const [loading, setLoading] = useState(true);
9   const [error, setError] = useState(null);
10  const viewerRef = useRef(null);
11  const [pageWidth, setPageWidth] = useState(null);
12
13  useEffect(() => {
14    console.log('PdfViewer received file:', file);
15    if (file) {
16      console.log('File type:', typeof file);
17      if (file instanceof Blob) {
18        console.log('File is a Blob. Size:', file.size, 'Type:', file.type);
19      }
20      if (file instanceof File) {
21        console.log('File is a File. Name:', file.name, 'Size:', file.size, 'Type:', file.type);
22      }
23    } else {
24      console.warn('No file received by PdfViewer.');
```

Overview

Deployments

Analytics

Speed Insights

Logs

Observability

Firewall

Storage

Flags

Setting

legal-ease

Repository

Usage

Domains

Visit

Production Deployment

Build Logs

Runtime Logs

Instant Rollback

Legal Document Simplifier

Upload your legal document. Get clear answers.

Transform complex legal language into plain English with AI-powered analysis.

Upload your PDF document

Choose File

Text Extraction

Advanced OCR

Recognize text

AI Summarization

Get clear, concise

summaries of

Q&A Support

Ask specific

questions about your

Deployment

legal-ease-dd9lkvgb-chinmays-projects-fef69bf2.vercelApp

Domains

legal-ease-delta.vercelApp

Status

Created

Ready

Sep 21 by reno Schubert

Source

main

3f274ae changes

Deployment Settings

3 Recommendations

To update your Production Deployment, push to the main branch.

Deployments

Firewall 24h

Enable Bot Protection

Observability 6h

Edge Requests

0

Analytics

Track visitors and page views

https://legal-ease-delta.vercel.app

Legal Document Simplifier

Upload your legal document. Get clear answers.

Transform complex legal language into plain English with AI-powered analysis

Upload your PDF document

Select a PDF file to get started with AI analysis

Choose File

Text Extraction

Advanced OCR technology extracts

AI Summarization

Get clear, concise summaries of complex

Q&A Support

Ask specific questions about your document

This is an AI prototype. Not legal advice. Please consult a qualified attorney for legal matters.

https://legal-ease-delta.vercel.app/document

Document Analysis

+ New Document

Document Preview

1706.03762v7.pdf • 2.11 MB

1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state-of-the-art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t , as a function of the previous hidden state h_{t-1} and the input for position t . This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 39]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network.

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state-of-the-art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

2 Background

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building blocks, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions [12]. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 27, 28, 23].

End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [34].

To the best of our knowledge, however, the Transformer is the first transduction model relying

Document Summary

This summary has been generated by AI and simplified for better understanding.

Policy Overview

Insured: Mr. Suresh Bhanudas Kale

Vehicle: SKODA Rapid Elegance 1.6 MPI (MH01BY4385)

Policy Period: 18 April 2024 – 17 April 2025

Premium Paid: ₹18,614.50

Insured Declared Value (IDV): ₹3,35,826

No Claim Bonus: 35% applied (based on prior clean history)

Risks & Limitations

PUC Certificate Requirement: Must be valid at all times. Failure to maintain it can lead to policy cancellation.

Usage Restrictions: Vehicle not covered for:

This is an AI prototype. Not legal advice. Please consult a qualified attorney for legal matters.

[link to the websites](#)