



The Privacy Law Compass

An AI powered tool for legal exploration



Members:

Clayton Brock, Hua-Hsing Huang, and Srimant Mishra



OVERVIEW OF TPLC

- An interactive web app introducing data privacy law, built with Streamlit
- Provides educational materials about data privacy laws
- Allows users to query and summarize laws in simple language
 - Google's GenAI models for working with the data
 - A Vector database (FAISS) of legal documents for LLM functionality



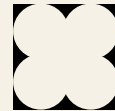


TABLE OF CONTENTS



01

BACKGROUND

Describes the motivation of the project

02

DATA USED

Goes over which data sets were used to build the app

03

USE CASES

Explains the interactions that different types of users have with the app

04

DESIGN

Explains the pipeline that we implemented

05

DEMO

Contains screenshots of various interactions with the app

06

LESSONS & FUTURE WORK

What we learnt and future scope of work





01

BACKGROUND

Motivations behind this project



IMPORTANCE OF DATA PRIVACY



Data privacy has become one of the central concerns of government and the tech industry these days. Several developments have also led to enhanced interest in the field of data privacy, such as:

- ◆ The rise of AI and the ethics of data collection
- ◆ Political fallout of privacy breaches (Cambridge Analytica)
- ◆ Growing interest from the people about privacy.

Thus, there is **clearly a need to understand data privacy protections better**, regardless of backgrounds.





ACCESSING LEGAL DOCUMENTS IS HARD



- Often, our queries are simple, like 'How does the law protect our health data?' However, we can't easily answer them just by reading the laws.



SCATTEREDNESS

Legal documents are created at all levels of the government and even internationally, causing information to be scattered across various websites



OVERLAP

Often, there are multiple laws on the same topic. Moreover, regulations can also be conflicting with each other.



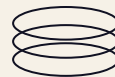
VERBOSITY

The legal language can be archaic or unclear, with much jargon that is unfamiliar to people without domain knowledge.



DENSITY

Legal documents are often huge in size and often come with various references to other equally voluminous laws





02

DATA USED

The foundation of our app





DATA SOURCES



- **Bill PDFs**

Provided the PDFs of state and federal level bills through their APIs



LegiScan



Open States

- **Education Materials**

Provided materials for the static content of our app



**International Association of
Privacy Professionals (IAPP)**





0

3

USE CASES

Different users and their interactions with the app



THREE PRIMARY USE CASES



1. THE INTERESTED CITIZEN

2. THE LEGAL EXPERT

3. THE WEBSITE MAINTAINER



O

4

DESIGN

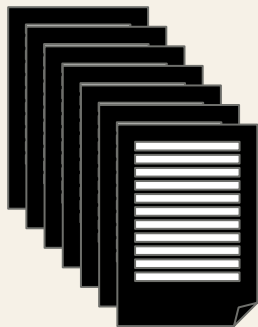
The app's implementation of its functionality



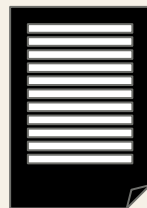
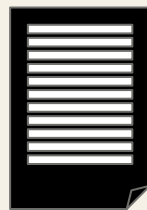


TASK 1: PREP THE DATA

1. Store PDF



2. Generate Doc Metadata



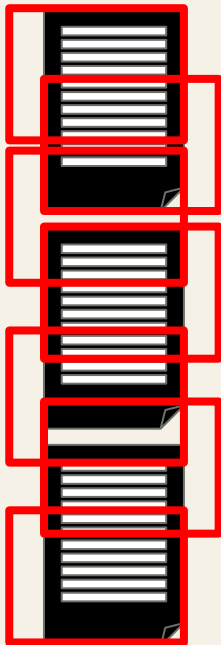
{"Title": " ",
"Sector": "",
...,
"Topics": []}



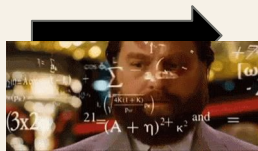


TASK 1: PREP THE DATA

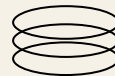
3. Chunk Doc



4. Vectorize Chunks



[300,
272,
...,
734,
11]

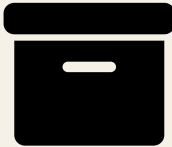
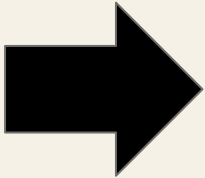




TASK 1: PREP THE DATA

5. Store in Vector DB

Chunk_ID : [[300, {“Title:” “,
272, “Sector:”},
..., ...,
734, “Topics”:[]}
11]



TASK 2: ANSWER USER QUERY



STEP 01

Question

STEP 02

Find Relevant Document
Chunks

STEP 03

Generate context relevant
answer to question

STEP 04

Generate relevant page
summaries



05

DEMO

A demonstration of the working product





06

LESSONS & FUTURE WORK

What we learnt and where we want to go from here





LESSONS



- ◆ Getting the data for bills of any nature is hard.
- ◆ API's for OpenStates and LegiScan are quite poor in performing filtered searches.
- ◆ The quality of the embedding function (converting words to vectors) is crucial for good results. For text embeddings, dimensions above 1000 are unnecessary.
- ◆ AI results can be unreliable. Chaining the results is useful in result validation and improvement.





LESSONS



- ◆ Creating custom styles in streamlit is a skill in itself
- ◆ Langchain documentation is hard to understand. Cannot exclusively rely on it.
- ◆ The functionality (and speed) of vector databases at large volumes is unreliable at the free tier.
- ◆ Finding the correct relevant information to use as context is difficult and sophisticated RAG approaches are necessary





FUTURE WORK



- ◆ Expanding the functionality of asking questions to other pages (with AI explainability)
- ◆ Explore integration testing - Test the interaction of functions and not just the function output
- ◆ Introducing “memory” functionality for the query section. New queries should reference knowledge of what was asked in the past, as well as former responses generated.
- ◆ Refine quality of responses and improve flexibility of type of queries the model can accurately handle
- ◆ Have multiple file inputs allowed in the Add documents page

