

21 April 2023

Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)



CeylonLawMate

Bringing Data into the Sri Lankan Courtroom

22 23 - J 25

21 April 2023

Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)



TEAM

Supervisors



**Dr. Nuwan Kodagoda
Selvakkumaran**

Research Supervisor
External Supervisor



Mr. Kavinga Abeywardena

Research Co-Supervisor



Prof. N.

21 April 2023

Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)



Team Members



RANODYA M. J.C
IT19987644
DS



Bandara I. M. R.
H IT18013474
IT



De Silva E.A.
A IT19991054
ISE



WHY ?

Despite the rising availability of data on people's experiences with justice, institutions around the world still have a long way to go to ensure justice for all.

As a recent report by the **World Justice Project** estimates, five billion people have unmet justice needs globally.

- 1.5 billion people cannot obtain justice for civil, administrative, or criminal justice problems.
- 4.5 billion people are excluded from the opportunities the law provides.
- 253 million people live in extreme conditions of injustice.



Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)

RESEARCH PROBLEM

Sri Lanka is in a dire need of a more efficient, accurate, and transparent judicial system.



Sri Lanka

The WJP Rule of Law Index 2021 provides scores and rankings based on eight factors:

- Restraints on Government Powers
- Absence of Corruption
- Open Government
- Fundamental Rights
- Order and Security
- Regulatory Enforcement
- Civil Justice
- Criminal Justice


It paints a picture of the rule of law in 139 countries and jurisdictions. This research places Sri Lanka 76th out of 139 nations.





Sri Lanka

Sri Lanka's law index value is 0.50, which is lower than the global law index average value of 0.56.







	Sri Lanka	0.50	-0.02	76	3 ▼
Overall Score	Regional Rank	Income Rank	Global Rank		
0.50	2/6	8/35	76/139		
Score Change	Rank Change				
-0.02 ▼	-3 ▼				



Sri Lanka

Sri Lanka's average values for all the sub-sectors considered for determining this overall average value are lower than the global average values for that.

Also, the overall average value for Sri Lanka is decreasing from 2015 to 2021.

	Factor Score	Score Change	Regional Rank	Income Rank	Global Rank
 Constraints on Government Powers	0.54	-0.02	3/6	7/35	69/139
 Absence of Corruption	0.49	0.02	1/6	2/35	61/139
 Open Government	0.52	0.01	2/6	5/35	64/139
 Fundamental Rights	0.53	-0.01	1/6	7/35	76/139
 Order and Security	0.56	-0.15*	4/6	31/35	127/139
 Regulatory Enforcement	0.50	0.02	2/6	9/35	72/139
 Civil Justice	0.45	0.00	2/6	21/35	103/139
 Criminal Justice	0.43	-0.01	2/6	7/35	73/139

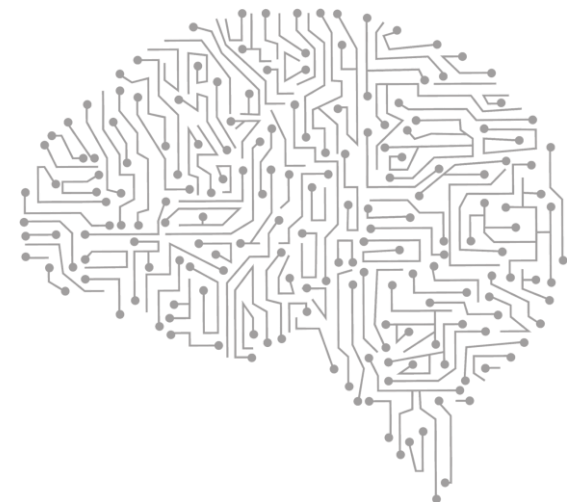
* Indicates statistically significant change at the 10 percent level

Low Medium High



HOW ?

While AI's potential in fields such as **healthcare, retail and e-commerce, food technology, banking and financial services, logistics and transportation, travel, real estate, entertainment and gaming, and manufacturing** is already widely explored, we started to look at what data has in store for jurisdictional institutions.





OBJECTIVES

Main Objective

- The main objective of this is to propose a well-functioning and accessible justice system, which is a prerequisite for sustainable development in the country, using new dimensions of AI technologies and thereby creating a more accessible, accountable, efficient, and impartial justice system for all citizens of the island.



OBJECTIVES

Sub Objectives

- To create and propose a system where people can get an idea of their legal problems before going to a lawyer or get the advice, they need by knowing other people's related legal problems and their legal background and experience.
- Assisting attorneys to automatically analyze their cases and generate quick cross-borders. Thereby creating opportunities for lawyers to provide better service to their clients and thereby helping to move the case for a more correct and fair decision in the trials.
- Assisting judges to automatically analyze their cases and study the facts of other previous trials similar to those trials. thereby helping the judge to take the case for a more accurate and fair decision.
- Propose and create an automated letter analysis system that enables court employees and the public to perform more accurate and proactive legal actions in filing lawsuits and other legal processes.



Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)

OUR SOLUTION



Our aim is to develop a web platform where all these identified stakeholders can get the necessary support.



The website will be a common platform for all these stakeholders to get the legal assistance they need and enhance the existing justice system.



A local platform to bring data to the Sri Lanka courtroom.

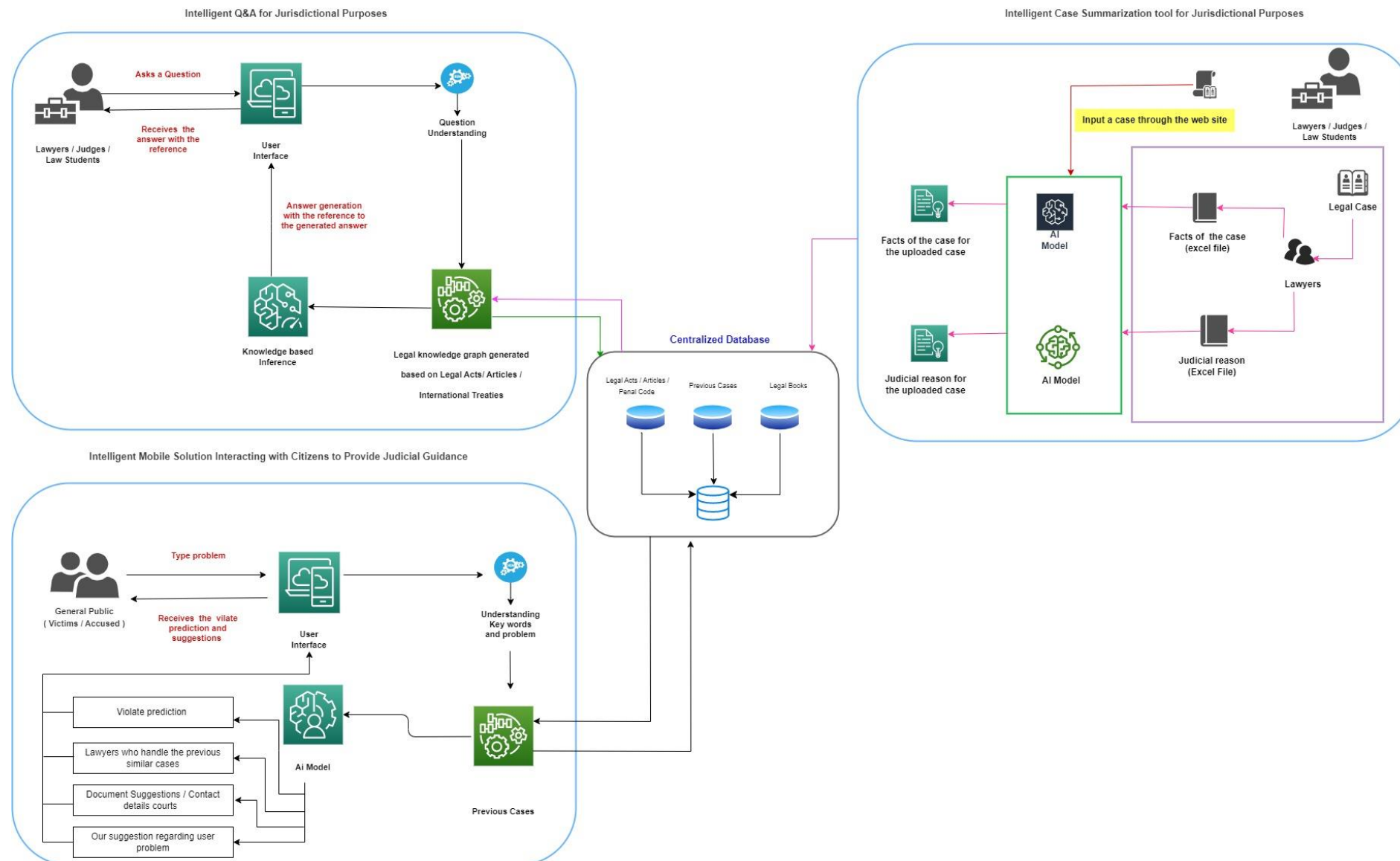


21 April 2023

Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)

Proposed System



21 April 2023

Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)



AI FOR JUSTICE – An Intelligent Question Answering System for Jurisdictional Purposes

RANODYA M. J. C. (IT19987644)



RESEARCH PROBLEM

- When preparing for a trial, attorneys and judges must review several materials. In addition, they must memorize or refer to another document to determine which documents they must review to find the information they seek. This takes up a lot of time, causing delays in case hearings because they need so much time to prepare for the trial, which has a negative impact on the administration of justice. As a result, an immediate emergency remedy should be implemented, as the situation poses significant social and corporate concerns.



SOLUTION

- The solution to the above research gap in the Sri Lankan context could be to design and implement an intelligent tool specifically aimed at improving access to, management of, and retrieval of relevant information from legal and jurisdictional resources according to the needs of lawyers and judges. A legal Q&A system easily eliminates the arduous effort of memorizing or referring to another document to determine which documents lawyers and judges must review to obtain the desired information. It also saves time, allowing the court to hear cases more rapidly because it takes less time to prepare for a trial. This tool could be optimized to make the legal process more efficient and less time-consuming for lawyers and judges, thus improving the administration of justice in the country.



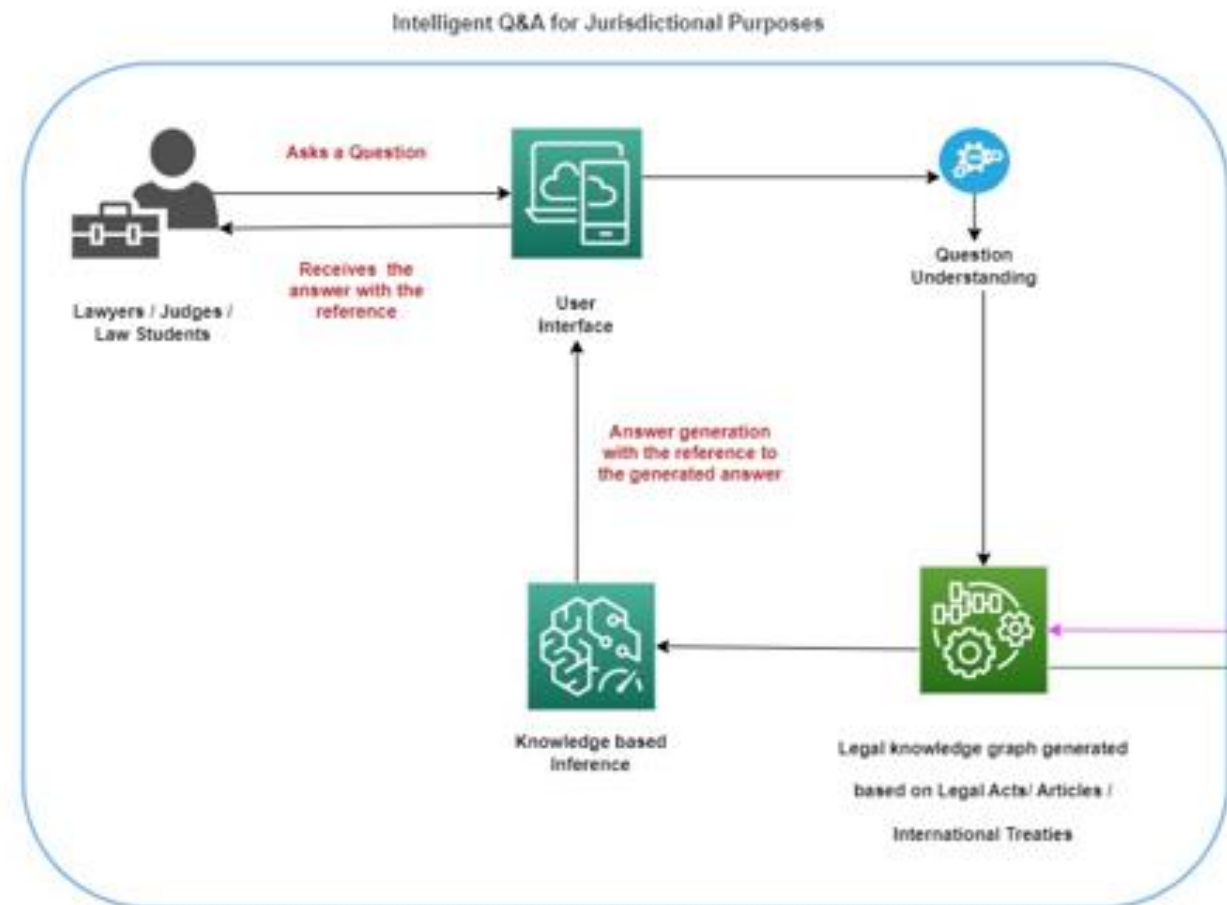
OBJECTIVES

Main Objective

- The main objective of this is to propose a well-functioning and accessible legal information retrieval system for lawyers and judges, using new dimensions of AI technologies, and thereby creating a more accessible, accountable, efficient, and impartial justice system for all citizens of the island.



SYSTEM DESIGN & METHODOLOGY





TECHNOLOGIES, LIBRARIES, TECHNIQUES

TECHNOLOGIES

- Python
- Anaconda
- VSCode
- React Native
- Google Colab
- Jupyter Notebook



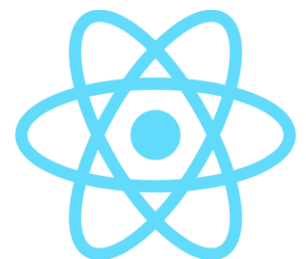
LIBRARIES

- Torch
- Numpy
- Pandas



TECHNIQUES

- AI
- NLP





RESEARCH PROGRESS

TASK	STATUS	COMPLETION
1.Data collection	Comple	100%
2.Build the model	Comple	100%
3.Checking the accuracy of the mod	Comple	100%
4.UI implementation	Comple	85%
5.Improving processing speed	In progress	70%





IMPLEMENTATION

Importing libraries and tokenizing

```
import math
import torch
import wikipedia
import PyPDF2, re
from pyvis.network import Network
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer
import IPython

tokenizer = AutoTokenizer.from_pretrained("Babelscape/rebel-large")
model = AutoModelForSeq2SeqLM.from_pretrained("Babelscape/rebel-large")
```



21 April 2023

Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)

IMPLEMENTATION

Data Loading

```
def read_pdf_data(
    pdf_file,
    pdf_dir = 'data/references/'
):
    pdf_path = pdf_dir + pdf_file + '.pdf' if pdf_file[-1] != '.' else pdf_dir + pdf_file + '.pdf'
    pdf_file = open(pdf_path, 'rb')
    pdf_reader = PyPDF2.PdfFileReader(pdf_file)
    num_pages = pdf_reader.getNumPages()

    whole_text = ''
    for page in range(num_pages):
        page_obj = pdf_reader.getPage(page)
        text = page_obj.extractText()
        whole_text += f"{text}"
    pdf_file.close()

    whole_text = whole_text.replace('\n', ' ')
    whole_text = re.sub(' +', ' ', whole_text)
    whole_text = whole_text.strip().lower()
    return whole_text
```

Python

```
text = read_pdf_data('Convention on the Rights of the Child-3')
# write to a file
with open('1.txt', 'w') as f:
    f.write(text)
```

Python



21 April 2023

Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)

IMPLEMENTATION

Creating the Knowledge Graph

```
class KB():
    def __init__(self):
        self.relations = []

    def are_relations_equal(self, r1, r2):
        return all(r1[attr] == r2[attr] for attr in ["head", "type", "tail"])

    def exists_relation(self, r1):
        return any(self.are_relations_equal(r1, r2) for r2 in self.relations)

    def print(self):
        print("Relations:")
        for r in self.relations:
            print(f" {r}")

    def merge_relations(self, r1):
        r2 = [r for r in self.relations
              if self.are_relations_equal(r1, r)][0]
        spans_to_add = [span for span in r1["meta"]["spans"]
                        if span not in r2["meta"]["spans"]]
        r2["meta"]["spans"] += spans_to_add

    def add_relation(self, r):
        if not self.exists_relation(r):
            self.relations.append(r)
        else:
            self.merge_relations(r)
```

Python



21 April 2023

Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)

IMPLEMENTATION

```
def extract_relations_from_model_output(text):
    relations = []
    relation, subject, relation, object_ = '', '', '', ''
    text = text.strip()
    current = 'x'
    text_replaced = text.replace("<s>", "").replace("<pad>", "").replace("</s>", "")
    for token in text_replaced.split():
        if token == "<triplet>":
            current = 't'
            if relation != '':
                relations.append({
                    'head': subject.strip(),
                    'type': relation.strip(),
                    'tail': object_.strip()
                })
            relation = ''
            subject = ''
        elif token == "<subj>":
            current = 's'
            if relation != '':
                relations.append({
                    'head': subject.strip(),
                    'type': relation.strip(),
                    'tail': object_.strip()
                })
            object_ = ''
        elif token == "<obj>":
            current = 'o'
            relation = ''
        else:
            if current == 't':
                subject += ' ' + token
            elif current == 's':
                object_ += ' ' + token
            elif current == 'o':
                relation += ' ' + token
        if subject != '' and relation != '' and object_ != '':
            relations.append({
                'head': subject.strip(),
                'type': relation.strip(),
                'tail': object_.strip()
            })
    return relations
```

NER



21 April 2023

Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)

IMPLEMENTATION

```
def from_text_to_kb(text, span_length=128, verbose=False):
    # tokenize whole text
    inputs = tokenizer([text], return_tensors="pt")

    # compute span boundaries
    num_tokens = len(inputs["input_ids"][0])
    if verbose:
        print(f"Input has {num_tokens} tokens")
    num_spans = math.ceil(num_tokens / span_length)
    if verbose:
        print(f"Input has {num_spans} spans")
    overlap = math.ceil((num_spans * span_length - num_tokens) /
                        max(num_spans - 1, 1))
    spans_boundaries = []
    start = 0
    for i in range(num_spans):
        spans_boundaries.append([start + span_length * i,
                                start + span_length * (i + 1)])
        start += overlap
    if verbose:
        print(f"Span boundaries are {spans_boundaries}")

    # transform input with spans
    tensor_ids = [inputs["input_ids"][0][boundary[0]:boundary[1]]
                  for boundary in spans_boundaries]
    tensor_masks = [inputs["attention_mask"][0][boundary[0]:boundary[1]]
                    for boundary in spans_boundaries]
    inputs = {
        "input_ids": torch.stack(tensor_ids),
        "attention_mask": torch.stack(tensor_masks)
    }
```

**Adding values to
Knowledge graph**



21 April 2023

Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)

IMPLEMENTATION

Q&A model

```
class QnA(object):
    def __init__(self):
        super(QnA, self).__init__()
        self.qna = QuestionAnswer()
        self.getEntity = GetEntity()
        self.export = exportToJson()
        self.graph = GraphEnt()
        self.pdf_dir = 'data/references/'

    def read_pdf_data(self, pdf_file):
        pdf_path = self.pdf_dir + pdf_file + '.pdf' if pdf_file[-1] != '.' else self.pdf_dir + pdf_file + '.pdf'
        pdf_file = open(pdf_path, 'rb')
        pdf_reader = PyPDF2.PdfFileReader(pdf_file)
        num_pages = pdf_reader.getNumPages()

        whole_text = ''
        for page in range(num_pages):
            page_obj = pdf_reader.getPage(page)
            text = page_obj.extractText()
            whole_text += f"{text}"
        pdf_file.close()

        whole_text = whole_text.replace('\n', ' ')
        whole_text = re.sub(' +', ' ', whole_text)
        whole_text = whole_text.strip().lower()
        return whole_text

    def extract_answers(self, question):
        all_files = os.listdir(self.pdf_dir)
        all_files = [file[:-3] for file in all_files if file[-3:] == '.pdf']
        all_outputs = []
        for idx, file in enumerate(all_files):
            context = self.read_pdf_data(file)
            refined_context = self.getEntity.preprocess_text(context)
            try:
                outputs = self.qna.findanswer(question, con=context)
            except:
                _, numberOfPairs = self.getEntity.get_entity(refined_context)
                outputs = self.qna.findanswer(question, numberOfPairs)
            all_outputs.append(outputs)

        print("Processing file {} of {}".format(idx + 1, len(all_files)))

        answers = [output['answer'] for output in all_outputs]
        scores = [output['score'] for output in all_outputs]
```

```
# get the best answer
best_answer = answers[scores.index(max(scores))]
reference = all_files[scores.index(max(scores))]
return best_answer, reference
```




REFERENCES

Automated Question Answering for Improved Understanding of Compliance Requirements: A Multi-Document Study

Sallam Abualhaija*[§], Chetan Arora[‡]*, Amin Sleimi*, Lionel C. Briand*[†]
*SnT Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg
[‡]Deakin University, Geelong, Australia

Towards Intelligent Legal Advisors for Document Retrieval and Question-Answering in German Legal Documents

Christoph Hoppe¹, David Pelkmann¹, Nico Migenda¹, Daniel Hötte², Wolfram Schenck¹
¹ Center for Applied Data Science, Bielefeld University of Applied Sciences, Gütersloh, Germany
² Faculty of Business, Bielefeld University of Applied Sciences, Bielefeld, Germany
{christoph.hoppe, david.pelkman, nico.migenda, daniel.hoette, wolfram.schenck}@fh-bielefeld.de

2020 7th NAFOSTED Conference on Information and
Computer Science (NICS)

VNLawBERT: A Vietnamese Legal Answer Selection Approach Using BERT Language Model

Chieu-Nguyen Chau
University of Science, Ho Chi Minh
city, Vietnam.
chauchieunguyen@gmail.com

Truong-Son Nguyen
Faculty of Information Technology,
University of Science, Ho Chi Minh
city, Vietnam.
Vietnam National University, Ho Chi
Minh city, Vietnam.

Le-Minh Nguyen
Japan Advanced Institute of Science
and Technology, Japan.
nguyenml@jaist.ac.jp

21 April 2023

Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)



AI FOR JUSTICE – An Intelligent Mobile Solution for Interacting with Citizens to Provide Jurisdictional / Judicial Guidance

Bandara I. M. R. H (IT18013474)



RESEARCH QUESTION

- People face many difficulties, at their environments. They often do not have any knowledge regarding their rights. Many of them are silent because they are either incapable of seeking legal advice or are afraid to do so. Sometimes they do not know how to contact a good lawyer to get support?
- Most of the systems are in as a desktop system.
- No solution to analyze and provide support.



21 April 2023

Artificial Intelligence (AI)

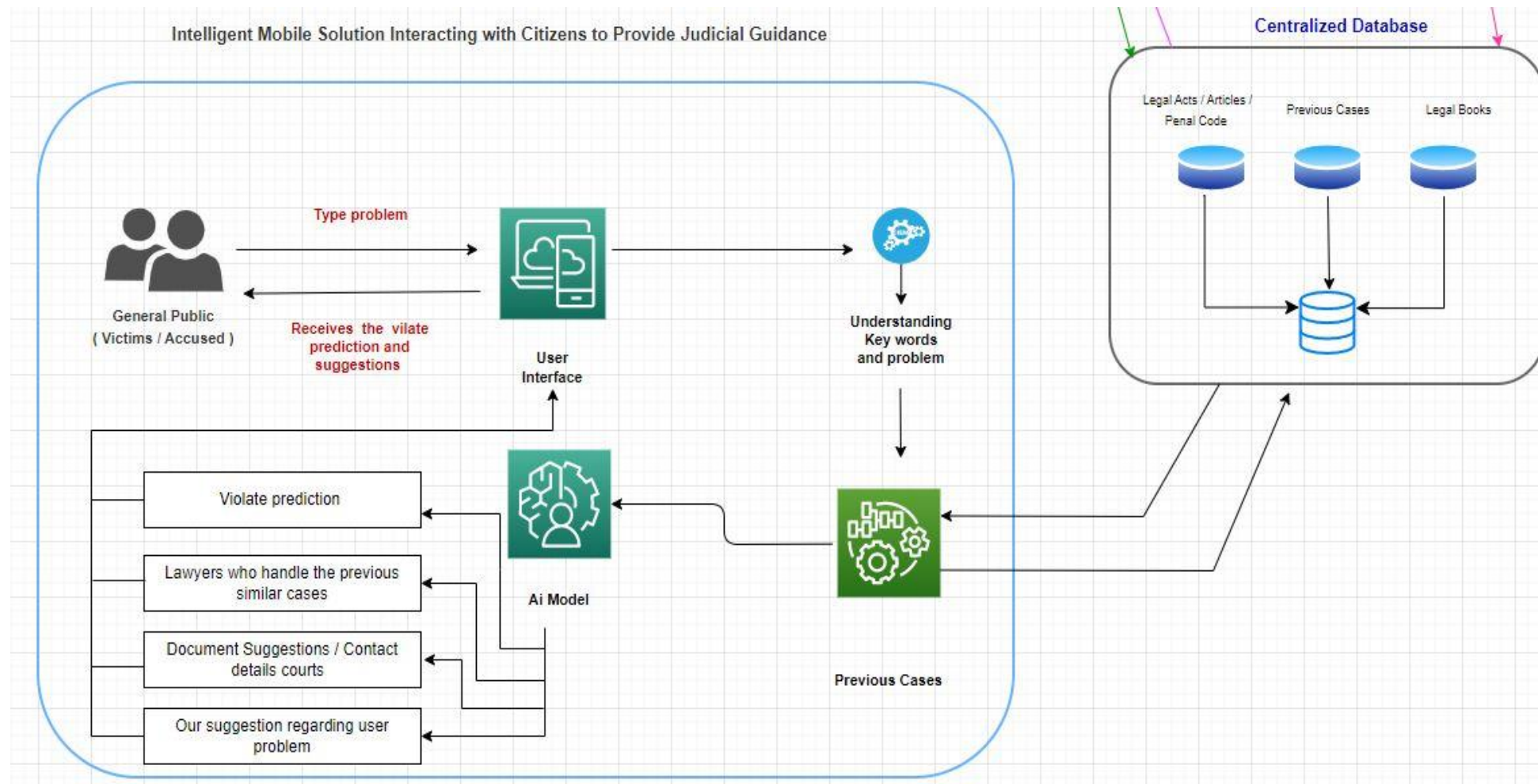
Autonomous Intelligent Machines and Systems (AIMS)

OBJECTIVES

Guide	Guide the public to understand their rights and responsibilities, provide them legal support when necessary, and connect them to a suitable lawyer.
Reduce	Reduce the percentage of person that unaware of their rights and that are not willing to take legal actions.
Save	Save the time and money of clients that are spending on unqualified lawyers.
Encourage	Encourage the percentage of the workforce to stand for their rights.



SYSTEM DESIGN





Requirements

Functional Requirements	Personal Requirements	Non - Functional Requirements
Predict user violate or not and guidance for user	All the materials to which law practitioners refer when handling a fundamental rights-related cases.	User-friendly interface
The system should be able to display the documents and previous cases handle lawyers		Performance and Memory
User authentication.		Scalability



Benefits of the System and how help to solve a problem

- The system will determine the problem by analyzing the information entered by the users to forecast the most pertinent solutions and simplify laws for their problems
- The best lawyers who have previously handled cases similar to the one at hand will handle the case.
- When members of the public type their problems into a specific field in our mobile app, the app suggests whether or not they are violating and if so, what the violation is
- And suggest what the process wants to do for those people. And suggest what documents they should bring with them. And contact details of the courts



Differentiate solution from other similar solutions available

- Many of the systems developed for the law domain are computer-based.
- The mobile apps have limited quantities available.
- But in public, most people don't have laptops or desktop computers. But all people have smart mobile phones, or at least one household member has one.
- So, with this mobile app, we can decrease the gap between the public and justice.
- And other mobile apps regarding justice are not predicted violated or not and do not suggest layers and the violated law and the documents



TECHNOLOGIES

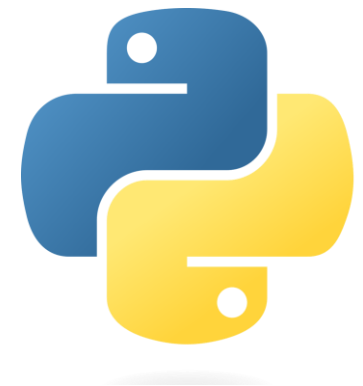
- We are using artificial intelligence (AI) and Machine learning (ML) Multi Class - Multi Output Text Classification and LSTM - Long Short-term memory
- And an artificial neural network used in the field of artificial intelligence and deep learning and includes libraries and packages for document analysis and generating predictive texts use to make a model
- And build a mobile application using react native with user friendly to predict the most relevant answers and simplify laws for their issues

- Python & Python libraries

- Jupyter Notebook

- Anaconda

- GitLab





Turned into a commercial product/project

- There is currently no such mobile application in Sri Lanka.
- Because people can easily identify their rights
- The app can be placed on the App Store
- And advertised through social media to gain commercial value.



IMPLEMENTATION

Import need Libraries

```
In [1]: import pickle
import PyPDF2, re
import numpy as np
import pandas as pd
import seaborn as sns
from datetime import datetime
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from tensorflow.keras.utils import plot_model

import re, os
os.environ["SM_FRAMEWORK"] = "tf.keras"

from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import RegexpTokenizer

from sklearn.utils import shuffle
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

import tensorflow as tf
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.activations import relu
from tensorflow.keras.models import load_model, Model
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Input, Dropout, LSTM, Bidirectional, Dense, Embedding, BatchNormalization, GRU

np.random.seed(1234)
tf.random.set_seed(1234)

print('Tensorflow version : ', tf.__version__)
```



IMPLEMENTATION

Reading data file

DATA LOADING

In [4]:

```
def read_pdf_data(
    pdf_file,
    pdf_dir = 'data/judgments/public cases/'
):
    pdf_path = pdf_dir + pdf_file + '.pdf' if pdf_file[-1] != '.' else pdf_dir + pdf_file + '.pdf'
    pdf_file = open(pdf_path, 'rb')
    pdf_reader = PyPDF2.PdfFileReader(pdf_file)
    num_pages = pdf_reader.getNumPages()

    whole_text = ''
    for page in range(num_pages):
        page_obj = pdf_reader.getPage(page)
        text = page_obj.extractText()
        whole_text += f"{text}"
    pdf_file.close()

    whole_text = whole_text.replace('\n', ' ')
    whole_text = re.sub(' +', ' ', whole_text)
    whole_text = whole_text.strip().lower()
    return whole_text
```




IMPLEMENTATION

Preprocessing data

DATA PREPROCESSING

In [10]:

```
def lemmatization(lemmatizer,sentence):
    lem = [lemmatizer.lemmatize(k) for k in sentence]
    return [k for k in lem if k]

def remove_stop_words(stopwords_list,sentence):
    return [k for k in sentence if k not in stopwords_list]

def preprocess_one(description):
    description = description.lower()
    remove_punc = re_tokenizer.tokenize(description) # Remove punctuations
    remove_num = [re.sub('[0-9]', '', i) for i in remove_punc] # Remove Numbers
    remove_num = [i for i in remove_num if len(i)>0] # Remove empty strings
    lemmatized = lemmatization(lemmatizer,remove_num) # Word Lemmatization
    remove_stop = remove_stop_words(stopwords_list,lemmatized) # remove stop words
    updated_description = ' '.join(remove_stop)
    return updated_description

def preprocessed_data(descriptions):
    updated_descriptions = []
    if isinstance(descriptions, np.ndarray) or isinstance(descriptions, list):
        updated_descriptions = [preprocess_one(description) for description in descriptions]
    elif isinstance(descriptions, np.str_) or isinstance(descriptions, str):
        updated_descriptions = [preprocess_one(descriptions)]

    return np.array(updated_descriptions)
```



IMPLEMENTATION

Making custom dataset

In [52]:

```
class PublicViolationDataset(Dataset):
    def __init__(self, cases_simple, ViolateFlags):
        self.cases_simple = cases_simple
        self.ViolateFlags = ViolateFlags
        self.tokenizer = tokenizer
        self.encodings = tokenizer(cases_simple, truncation=True, padding=True)

    def __len__(self):
        return len(self.cases_simple)

    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
        item['labels'] = torch.tensor(self.ViolateFlags[idx])
        return item
```



IMPLEMENTATION

Training the model

MODELING / TRAINING

In [19]:

```
inputs = Input(shape=(max_length,))
x = Embedding(vocab_size, embedding_dim, input_length=max_length)(inputs)
x = Bidirectional(LSTM(128, return_sequences=True))(x)
x = Bidirectional(LSTM(64, return_sequences=False))(x)

x = Dense(64, activation='relu')(x)
x = Dropout(0.5)(x)

x = Dense(32, activation='relu')(x)
x = Dropout(0.5)(x)

x1 = Dense(2, activation='softmax', name='ViolationFlag')(x)
x2 = Dense(5, activation='softmax', name='ViolationType')(x)

model = Model(
    inputs=inputs,
    outputs=[x1, x2],
    name='PublicViolationDetector')
model.summary()
```

Out [19]:

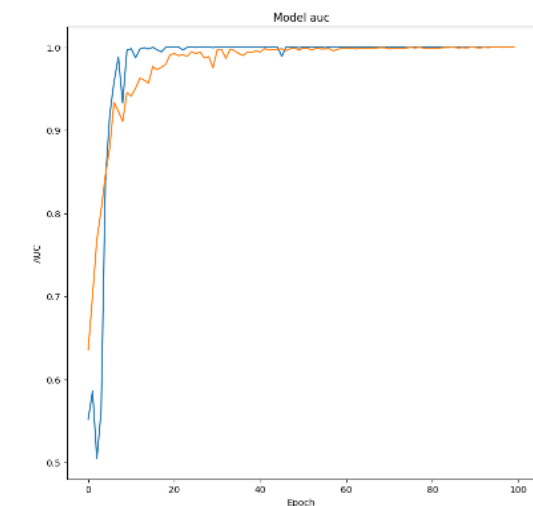
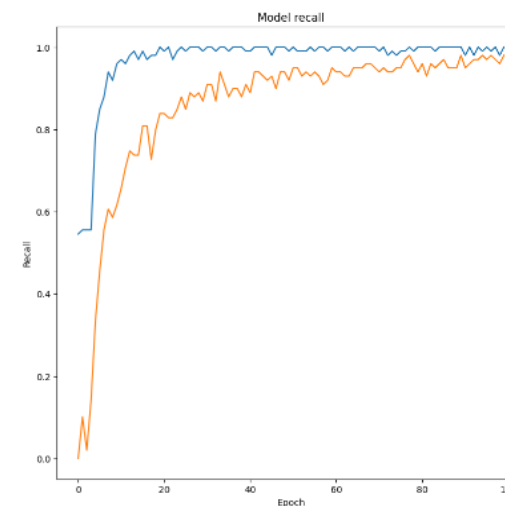
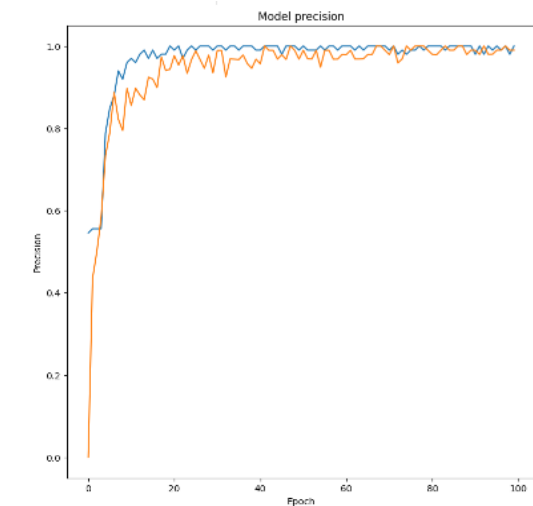
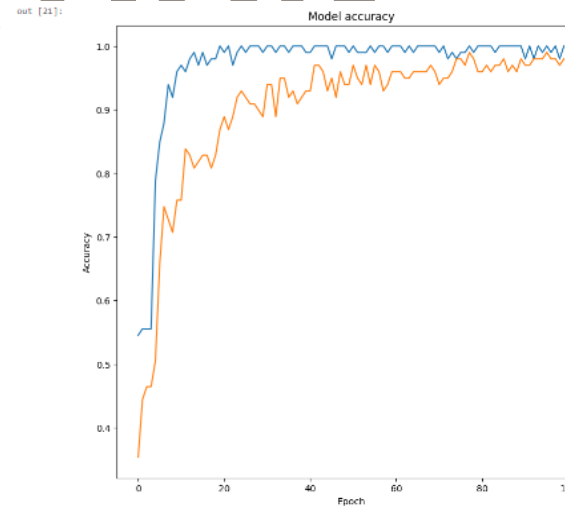
Model: "PublicViolationDetector"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 500)]	0	[]
embedding (Embedding)	(None, 500, 100)	433300	['input_1[0][0]']
bidirectional (Bidirectional)	(None, 500, 256)	234496	['embedding[0][0]']
bidirectional_1 (Bidirectional)	(None, 128)	164352	['bidirectional[0][0]']
dense (Dense)	(None, 64)	8256	['bidirectional_1[0][0]']
dropout (Dropout)	(None, 64)	0	['dense[0][0]']
dense_1 (Dense)	(None, 32)	2080	['dropout[0][0]']
dropout_1 (Dropout)	(None, 32)	0	['dense_1[0][0]']



IMPLEMENTATION

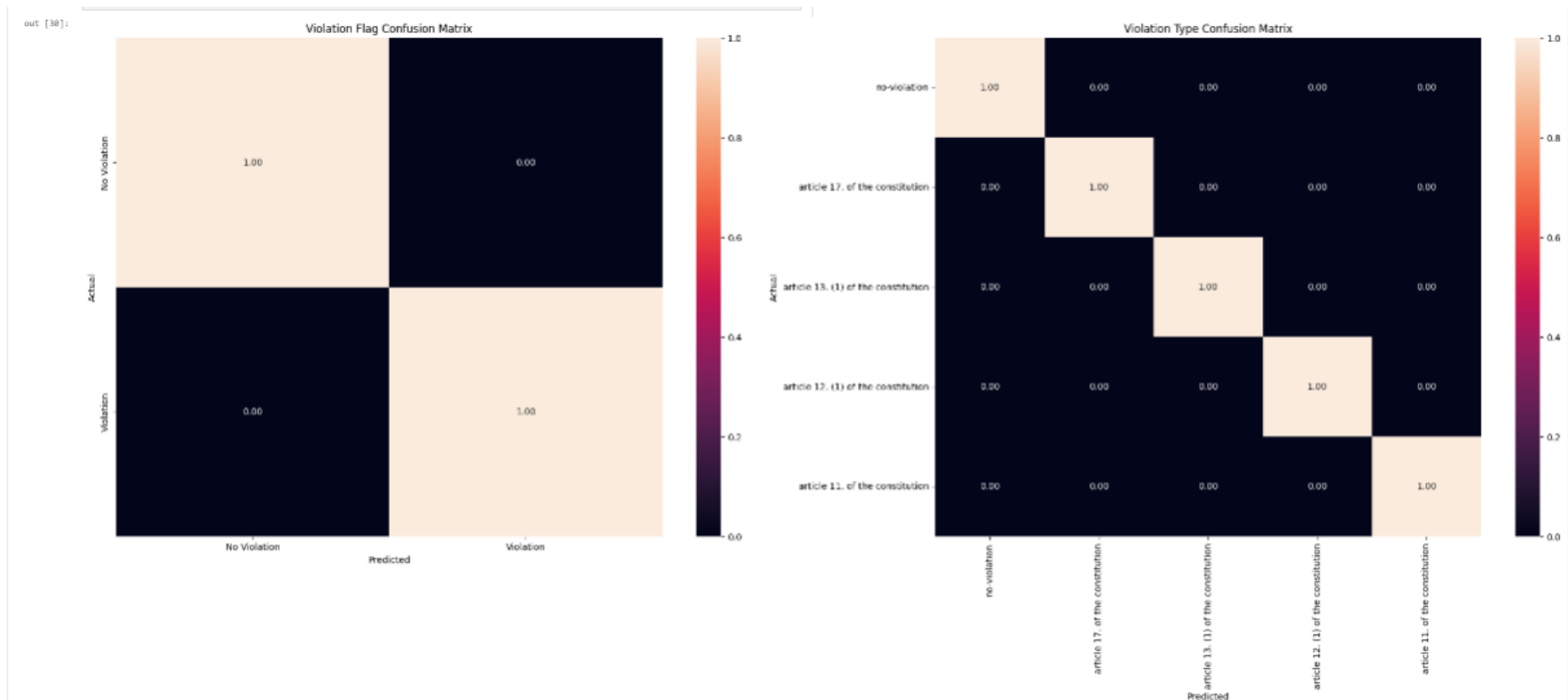
Result of trained model





IMPLEMENTATION

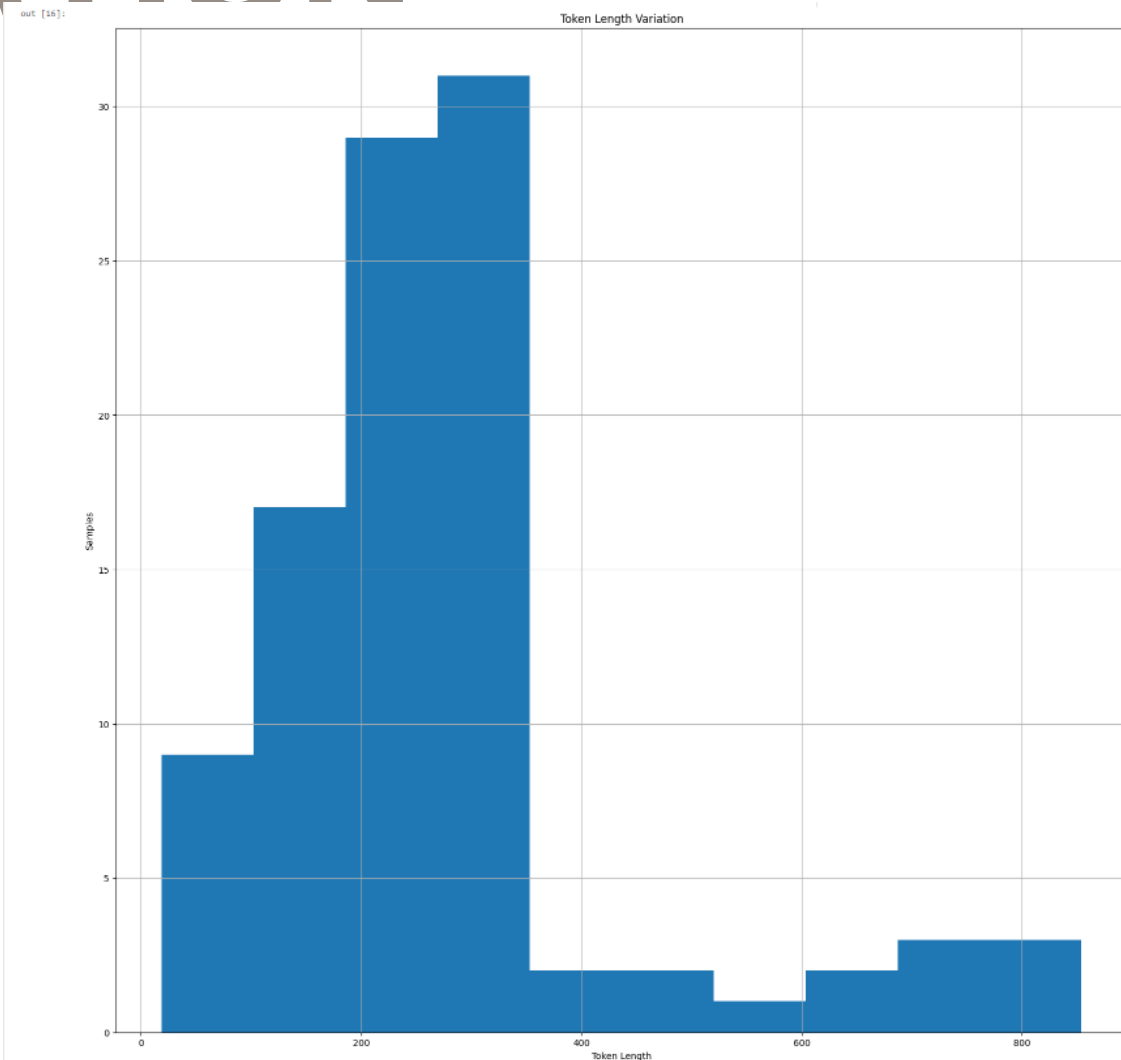
Result of trained model





IMPLEMENTATION

Result of trained model





IMPLEMENTATION

Result of trained mode





21 April 2023

Artificial Intelligence (AI)

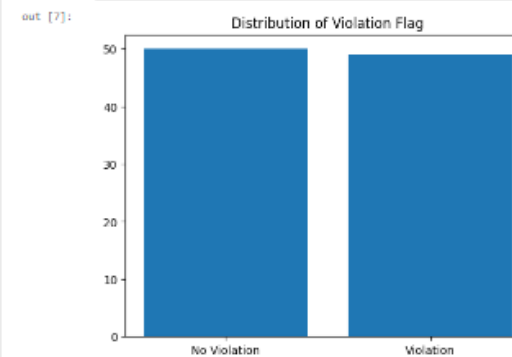
Autonomous Intelligent Machines and Systems (AIMS)

IMPLEMENTATION

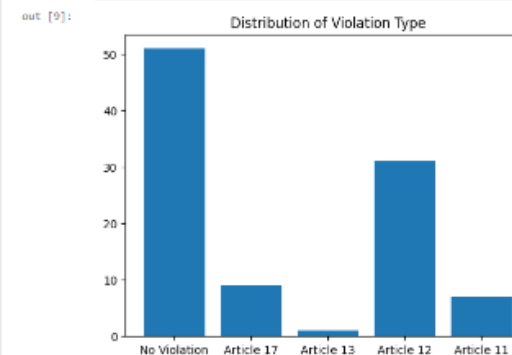
Result of trained model

EDA

```
In [7]: # Distribution of Violation Flag
plt.bar(['No Violation', 'Violation'], [ViolateFlags.count(0), ViolateFlags.count(1)])
plt.title('Distribution of Violation Flag')
plt.show()
```



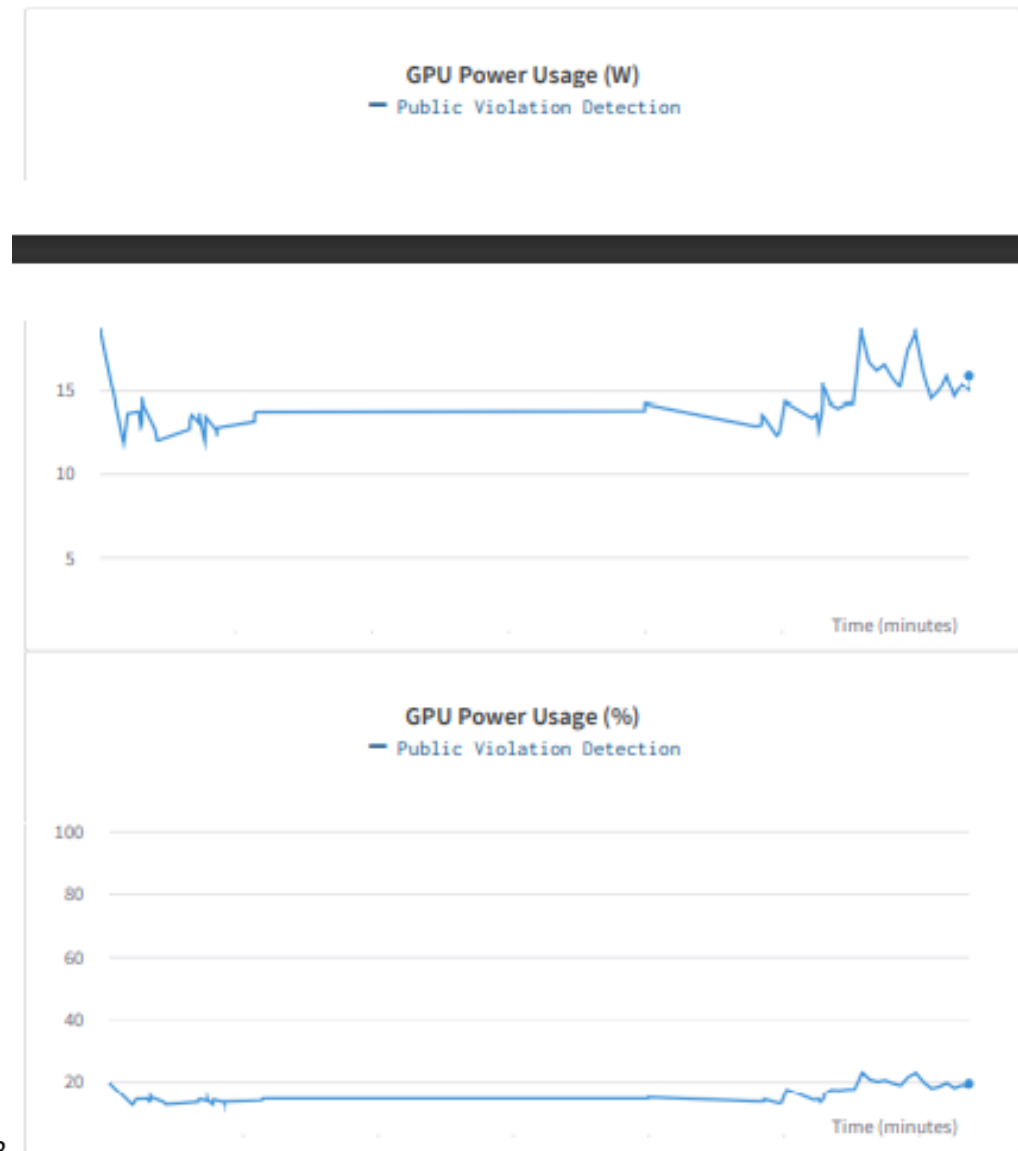
```
In [9]: # Distribution of Violation Type
plt.bar(['No Violation', 'Article 17', 'Article 13', 'Article 12', 'Article 11'], [ViolationTypes.count(0), ViolationTypes.count(1), ViolationTypes.count(2), ViolationTypes.count(3), ViolationTypes.count(4)])
plt.title('Distribution of Violation Type')
plt.show()
```





IMPLEMENTATION

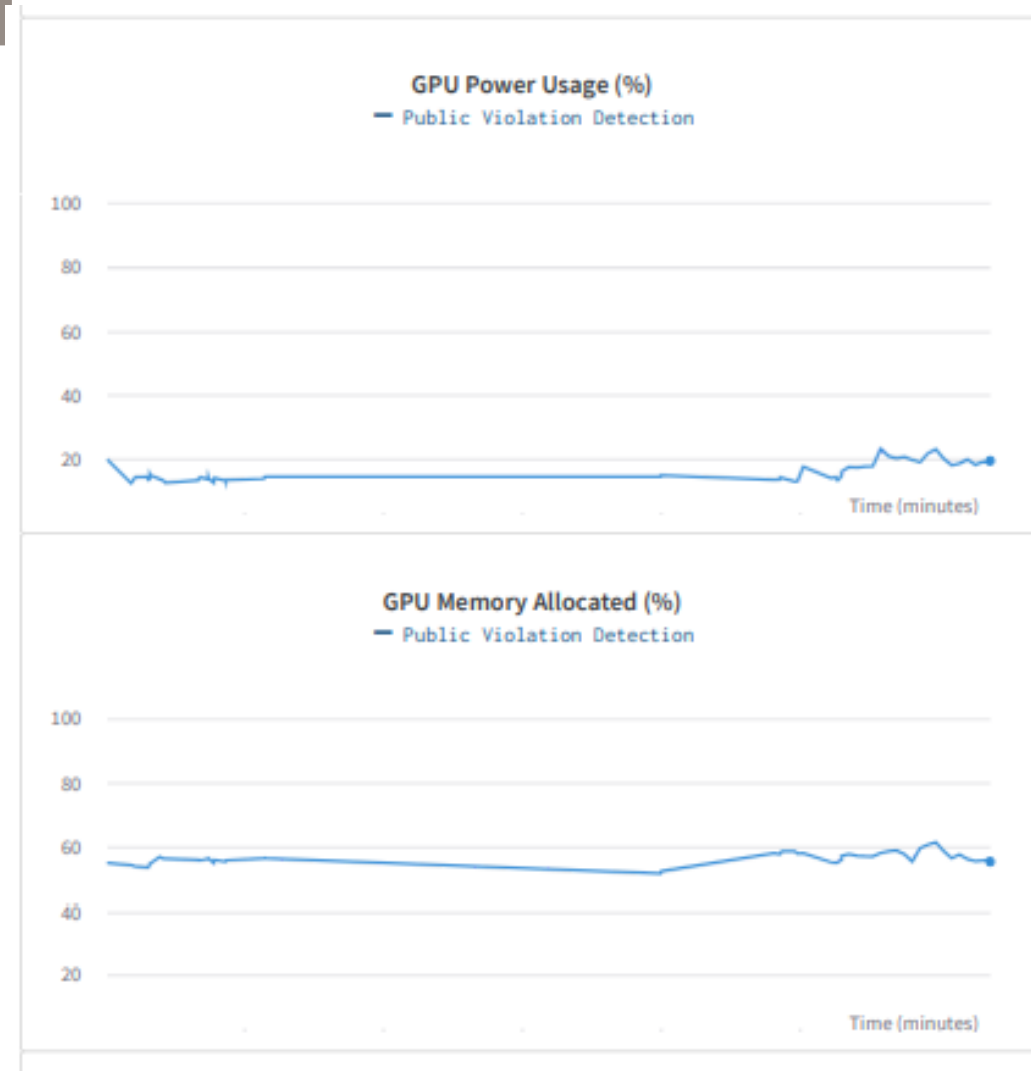
Requirements for training the model





IMPLEMENTATION

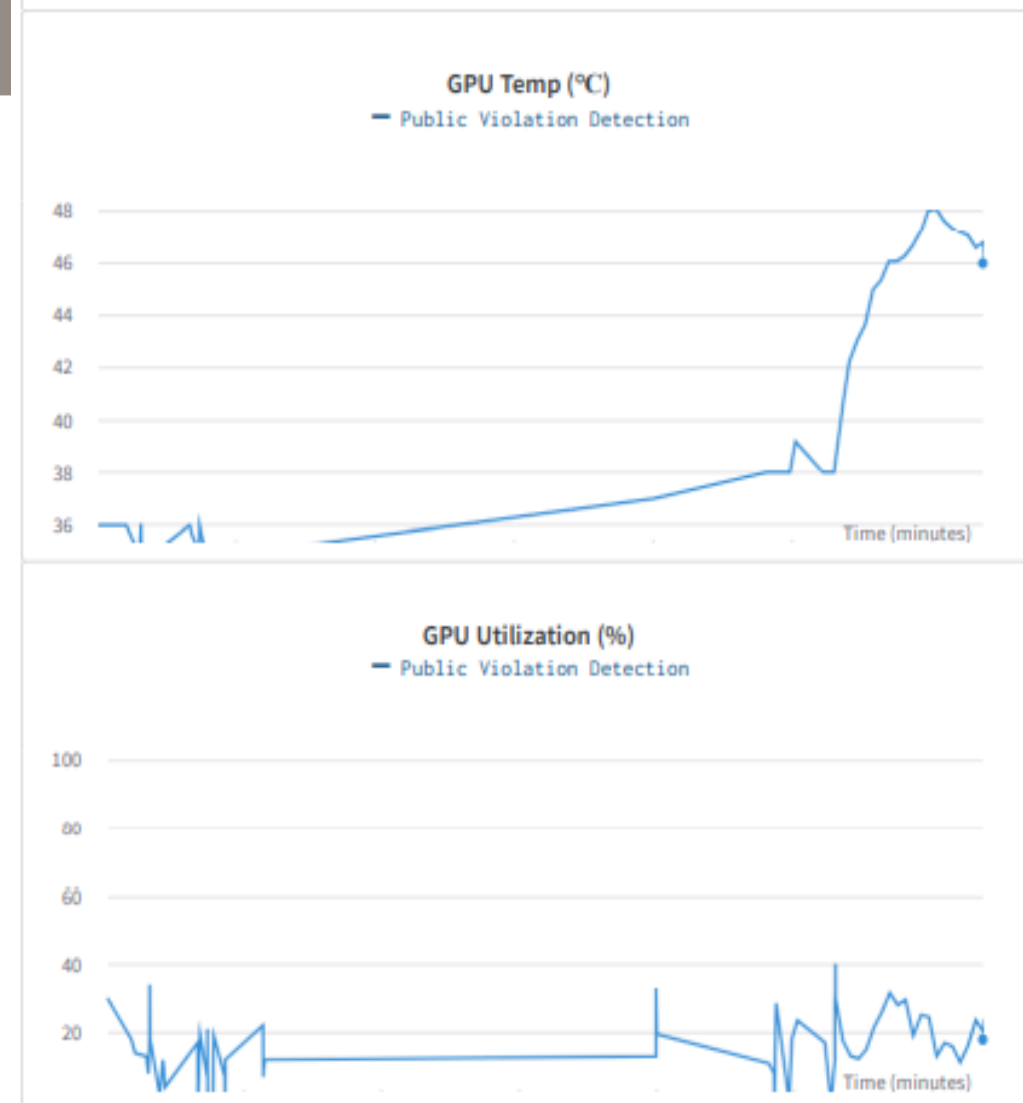
Requirements for training the model





IMPLEMENTATION

Requirements for training the model



21 April 2023

Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)



AI FOR JUSTICE – An Intelligent Content Summarizing tool for Jurisdictional Purposes.

De Silva E.A. A (IT19991054)



RESEARCH PROBLEM

- The judicial system in Sri Lanka faces several challenges that affect its ability to function effectively and efficiently.
- These include the shortage of lawyers and judges, as well as the lack of legal and educational training, leading to a rise in the number of pending cases year after year .
- Additionally, the preparation process for trials requires attorneys and judges to review multiple documents and rely on their memory to recall which documents are relevant to the case.
- This process is time-consuming and can lead to delays in case hearings, resulting in a negative impact on the administration of justice.



OBJECTIVES

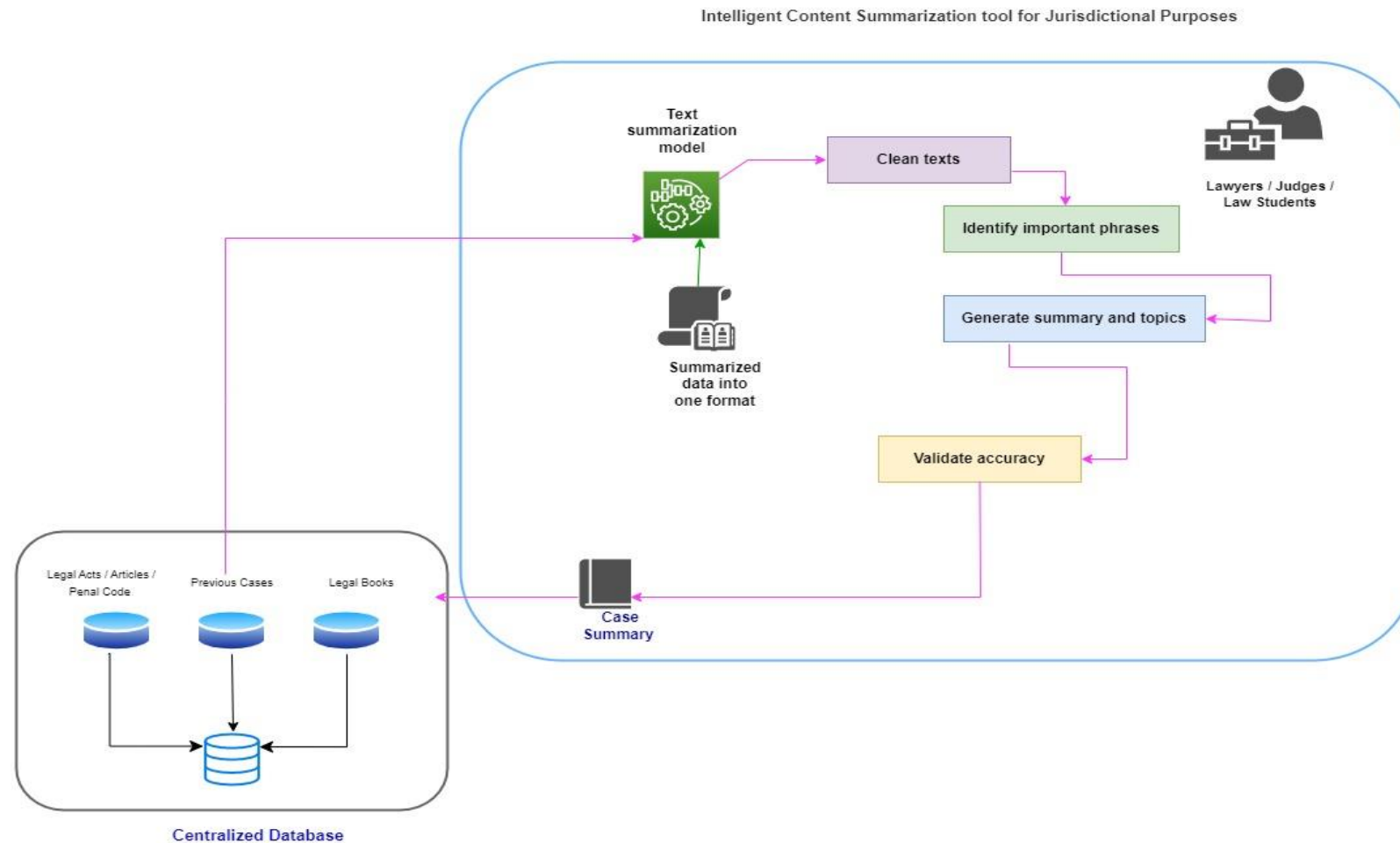
- The main objective of this component is to summarize the information from all related resources in different formats and aid lawyers and judges in understanding the necessary details in a more efficient way.

Specific Objectives

- Reduce the paper cost and paperwork.
- Decrease the paper cost and paperwork.
- Minimize the cost of case summarization



SYSTEM DESIGN





TECHNOLOGIES, LIBRARIES, TECHNIQUES

TECHNOLOGIES

- Python
- Anaconda
- VSCode
- React Native
- Google Colab
- Jupyter Notebook



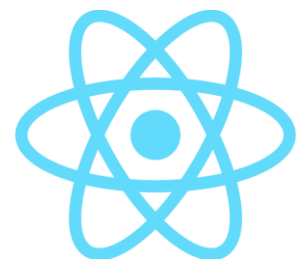
LIBRARIES

- Torch
- Numpy
- Pandas



TECHNIQUES

- AI
- NLP





IMPLIMANTATION

```
import re, pickle
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
```

Importing by default python libraries.

Tensorflow for model tracking, performance monitoring, and model retraining.
Matplotlib.pyplot to create and visualize line plots, scatter plots and histograms.



IMPLIMANTATION

```
def decontracted(phrase):  
    phrase = re.sub(r"won't", "will not", phrase)  
    phrase = re.sub(r"can't", "can not", phrase)  
    phrase = re.sub(r"n't", " not", phrase)  
    phrase = re.sub(r"\ 're", " are", phrase)  
    phrase = re.sub(r"\ 's", " is", phrase)  
    phrase = re.sub(r"\ 'd", " would", phrase)  
    phrase = re.sub(r"\ 'll", " will", phrase)  
    phrase = re.sub(r"\ 't", " not", phrase)  
    phrase = re.sub(r"\ 've", " have", phrase)  
    phrase = re.sub(r"\ 'm", " am", phrase)  
    return phrase
```

- This code defines a function called decontracted, which takes a single parameter phrase as input.

The purpose of the function is to expand English contractions into their full form.

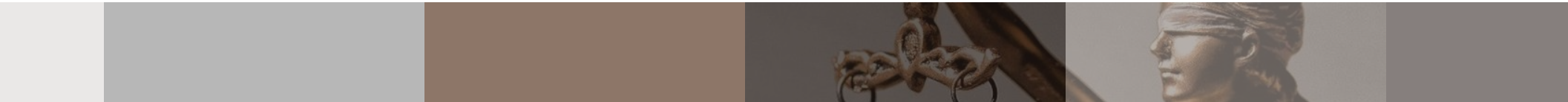
Eg : can't → can not



IMPLIMANTATION

```
def clean_dataset(file_path):  
    df = pd.read_excel(file_path)  
    df['Case'] = df['Case'].apply(clean_case)  
    df['Facts'] = df['Facts'].apply(lambda x : 'sostok ' + x + ' eostok')  
  
    cases = df['Case'].tolist()  
    summaries = df['Facts'].tolist()  
  
    return cases, summaries
```

This code defines a function called `clean_dataset`, which takes a single parameter `file_path` as input. The purpose of the function is to clean and preprocess a dataset of legal cases stored in an Excel file.



RESEARCH PROGRESS

TASK	STATUS	COMPLETION
1.Data collection	Comple	100%
2.Build the model	Comple	100%
3.Checking the accuracy of the mod	Comple	100%
4.UI implementation	Comple	85%
5.Improving processing speed	In progress	60%





METHOD

eg: Maintaining a custom data set

```
tokenizer = AutoTokenizer.from_pretrained("google/pegasus-xsum")
tokenized_answers = tokenizer(Answers, truncation=True, padding=True)
tokenized_summarized = tokenizer(Summarized, truncation=True, padding=True)

class SummarizationDataset(Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):
        return len(self.labels)
```

✓ 24.4s



Training Arguments:

```
training_args = TrainingArguments(  
    output_dir='Answer Summarization',  
    num_train_epochs=1,  
    per_device_train_batch_size=100,  
    per_device_eval_batch_size=100,  
    warmup_steps=500,  
    weight_decay=0.01,  
    logging_dir='Answer Summarization/logs',  
    logging_steps=10  
)
```



RISKS AND MITIGATION

RISK	THE WAY MITIGATED
Data gathering	<ul style="list-style-type: none">● Contact relevant parties(Lawyers and judges) in collect data
Economical risks	<ul style="list-style-type: none">● Identify the exact procedures that we have to spend money● money Collect certain amount of money from all members and use it when needed
Difficulty to manage time	<ul style="list-style-type: none">● Share tasks among members and act accordingly



Further improvements can make to this product/project

- Our application is currently based solely on the fundamental rights doctrine.
- We hope to improve the system in other law domains.
- Currently the system responds only for the English language. We hope to implement our system in Sinhala and Tamil languages

21 April 2023

Artificial Intelligence (AI)

Autonomous Intelligent Machines and Systems (AIMS)



Thank You!!