AI/ML based Methods for Judicial Analytics and Efficiency

Project-Part II (CS57004) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Masters in Technology - Dual 5Y

in

Computer Science and Engineering

by

Aayushi Vidyanta

(18CS30003)

Under the supervision of

Professor Partha P. Chakrabarti

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

Spring Semester, 2022-23

April 26, 2023

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: April 26, 2023 (Aayushi Vidyanta)

Place: Kharagpur (18CS30003)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**AI/ML based Methods for Judicial Analytics and Efficiency**" submitted by **Aayushi Vidyanta** (Roll No. 18CS30003) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Masters in Technology - Dual 5Y in Computer Science and Engineering is a record of bona fide work carried out by him under my supervision and guidance during Spring Semester, 2022-23.

Date: April 26, 2023

Place: Kharagpur

Professor Partha P. Chakrabarti
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Name of the student: **Aayushi Vidyanta** Roll No: **18CS30003**

Degree for which submitted: **Masters in Technology - Dual 5Y**

Department: **Department of Computer Science and Engineering**

Thesis title: **AI/ML based Methods for Judicial Analytics and Efficiency**

Thesis supervisor: **Professor Partha P. Chakrabarti**

Month and year of thesis submission: **April 26, 2023**

A lot of judicial time in courts, especially in Indian courts, is spend on administrative work which can be done much more faster by the use of AI/ML in this field. However, till date, the use of AI/ML has been very limited, if at all, especially for a complex problem like court scheduling, which has a lot of factors that affect it. The objective of this project is to analyse judicial data using AI/ML which will allow us to develop next generation scheduling algorithms for courts that combine the unique scheduling algorithms with data-derived learning algorithms to optimise a variety of parameters covering time, fairness, urgency, etc. We have collected large amounts of judicial data from CHC, MCCC and SCoI. We perform various statistical analyses on this data. We also introduce a novel representation of a case as a Case Trace and it summary by a Reduced Trace. We introduce various Edit Distance based algorithms for computing similarity between traces. Furthermore, we propose a metric based on Top-K analysis which helps us in analysing our similarity formulations. Finally, we also perform an ablation study on the effect of presence of senior advocates on pendency of cases.

Contents

List of Figures

List of Tables

Abbreviations

AI	**A**rtificial **I**ntelligence
ML	**M**achine **L**earning
IST	**I**ntelligent **S**cheduling **T**ool
CIS	**C**ase **I**nformation **S**ystem
CNR	**C**ase **N**umber **R**ecord
CHC	**C**alcutta **H**igh **C**ourt
MCCC	**M**umbai **C**ity **C**ivil **C**ourt
SCoI	**S**upreme **C**ourt **o**f **I**ndia
SAS	**S**tandard **A**bstraction **L**evel
DAS	**D**etailed **A**bstraction **L**evel

Chapter 1

Introduction

1.1 Background

The legal judiciary system of most countries, including the USA, UK, and especially India, have a lot of administrative work done in courts. This quite often leads to a lot judicial time being wasted for work which can be handled much faster by the use of state-of-the-art technology in the domain of AI/ML. This, in turn, can cause a an efficiency in court hearings, a substantial reduction in pendency of court cases, an equitable distribution of case hearings of all types of cases and a lot of time being saved by the parties and legal representatives involved in them. Our work is based on the legal and judicial systems in place in India.

Indian Legal System Indian law refers to the system of law which operates in India. It is largely based on the practices of English common law. Various Acts introduced by the British are still in effect in modified form today. Much of contemporary Indian law shows substantial European and American influence.

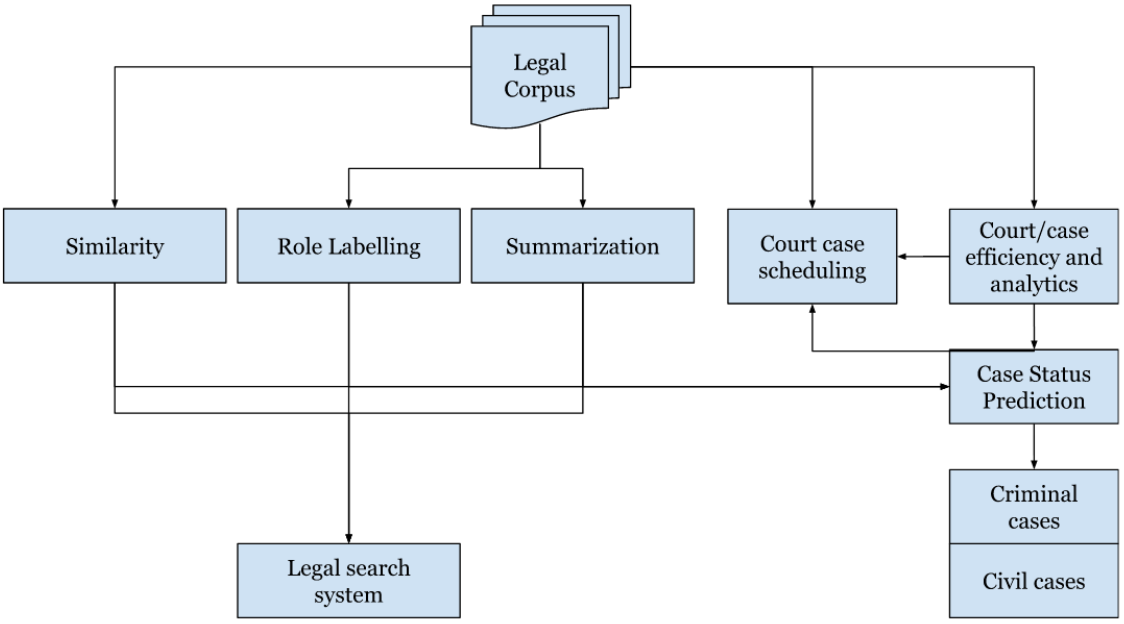


FIGURE 1.1: The umbrella project on Legal Analytics

Indian Judicial System The three-tiered system of Indian judiciary comprises of the: Supreme Court (New Delhi) at its helm, High Courts standing at the head of the state judicial system and is followed by district and sessions courts in the judicial districts, into which the states are divided.

The lower rung of the system comprises courts of civil (civil judges) and criminal (judicial/metropolitan magistrates) jurisdiction.

1.2 Motivation and Objectives

Recent trends in the Indian Legal System Whereas the use of computers and other electronic devices in the legal profession has been advocated in the West for the last forty years or so, hardly any worthwhile attention seems to have been paid in India until recently towards highlighting their use and significance in this area. The consensus that has emerged now in India is that in view of increasing globalization of the legal profession, the use of computers in this profession too has

become a need of the day. Hence, the Indian Judiciary has taken significant efforts in computerising the Indian Legal System.

However, our legal system is far from removing its vices. A lot of judicial time is still wasted in Indian Courts on matters which could be handled with technology much more rapidly. The law is still inaccessible to the common public. Most of the common public do not know when their rights are being violated and are vulnerable to unintentional law violations. Most people do not take action in court for any wrongdoing committed against them because the legal proceedings take years and years to complete, and even then, the verdict can quite possibly not be in their favor. Our project aims to target these issues.

The 2 main parts to this umbrella project as shown in 1.1 are:

1. **Procedural Efficiency**
2. **Behavioral Analytics**

Procedural Efficiency It aims to design a **Smart Legal Consultant** (as shown in 1.2) which is low-cost automated text processing system for prior case retrieval and summarization of long legal documents.

Behavioral Analytics This part (on which this paper focuses) concerns itself with developing an **Intelligent Scheduling Tool (IST)** which will be user-centric, adaptable and based on responsive technology to schedule and allocate cases in courts in order to ensure optimal utilisation of judicial time. The highlighted portion in 1.3 shows the court scheduling part which is the focus of this report.

Required specifications of the IST Output for Users

1. **Judges**

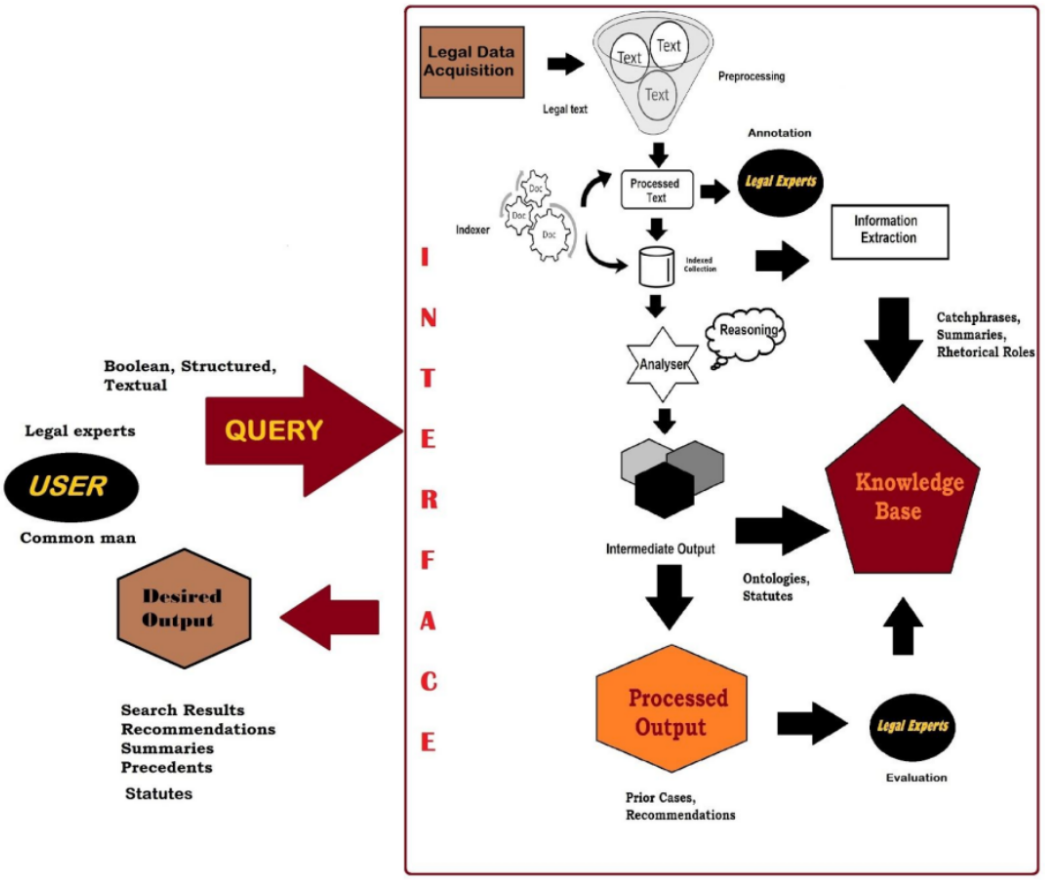


FIGURE 1.2: An overview of the system proposed under the Smart Legal Consultant

(a) Automatic scheduler that saves judicial time spent on allocating dates for cases

(b) Smart assistance in adjourning matters to a date when it can be heard without conflict in schedule of the advocates and judges

(c) Assist in balancing judicial time between fresh and pending matters, urgent (both fresh and pending) and non urgent matters

2. **Registry and Court Officials**

(a) Assist in preparation of automated cause lists

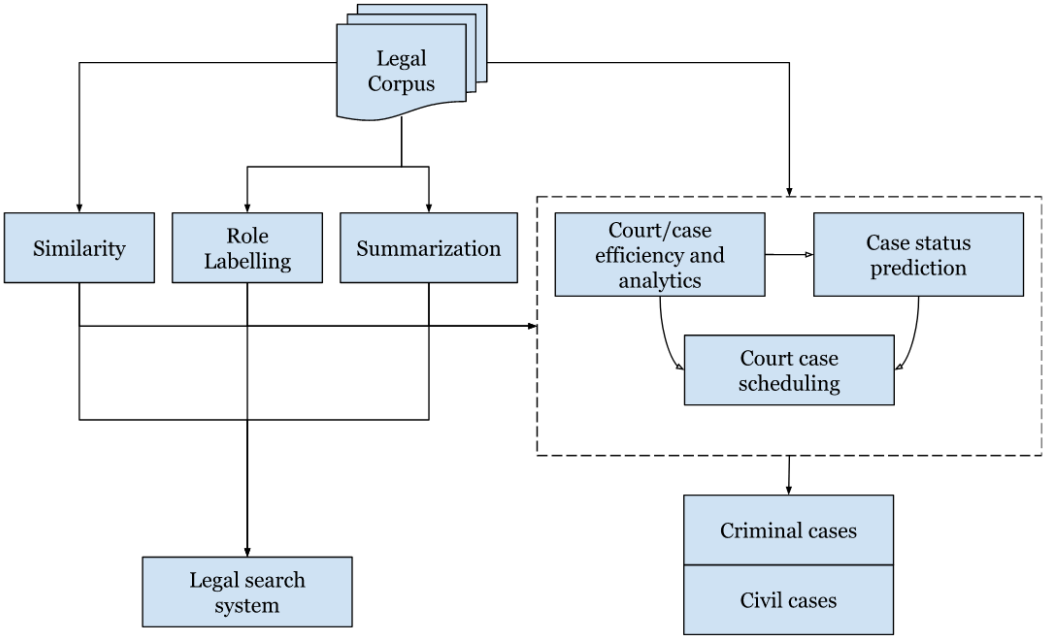


FIGURE 1.3: Behavioral in the umbrella project

(b) The chosen court will have a smart centralised digital calendar which will, after learning from the data feed to it, predict the capacity of the court of a particular date to hear a matter

3. **Advocates and Litigants**

(a) Advocates and litigants who will be notified of judges' leaves in advance with an option, within the IST, for advocates to indicate an adjournment request paving the way for online adjournments

(b) Notify advocates and parties in case the court cannot sit in session or is discharged (i.e, matters which a judge declares adjourned because the court is busy) due to unforeseen circumstances.

(c) Speedy and efficient disposal of cases

Outcome from IST

1. Allocation of cases equitably based on roster distribution

2. Decrease chances unattended case types
3. Decrease shelf life of all categories of cases
4. Continuously track the qualitative and quantitative metrics that can improve the adoption and impact of the tool

Features of the IST

1. **Judges**
   (a) To schedule other internal/external meetings and be given reminders or notifications at a fixed time prior to these meetings
   (b) To identify categories or case stages that are not being heard despite being listed
   (c) To remove overlaps automatically ensuring no conflict in the judges' and advocates' schedule to the extent possible
2. **Registry and Court Officials**
   (a) To generate consolidated roster for the court complex
   (b) To allocate cases based on court listing rules/regulations/orders
   (c) To prioritise cases based on listing rules (cases that are groups/clubbed, involving senior citizens/women as parties to be prioritised even if not mentioned specifically in the rules)
   (d) To create and control schedule for the judges based on their roster and availability
   (e) To generate weekly/monthly cause-lists
   (f) To have in-built analytics reports to improve the tool
3. **Advocates and Litigants**

(a) To remove overlaps automatically ensuring no conflict in the judges' and advocates' schedule to the extent possible

(b) To be able to see all cases of a particular advocate listed across courtrooms from a particular day/week

(c) To receive SMS/email notification to advocates and litigants with the schedule

Input Data Points for IST

1. **Data for Prototype**

   (a) Exact number of pending cases and age of these cases

   (b) Number of judges in the court complex and vacancy

   (c) Average Number of fresh filings per day/week/month

   (d) Average number of urgent hearings granted by a particular court

   (e) Roster of every judge

   (f) Past assignments of a judge

   (g) Case categories

   (h) Number of days/months/years since a case has been pending

   (i) Number of times the case has been transferred

2. **Data captured by the tool on an Ongoing Basis**

   (a) Average number of fresh filings per day/week/month

   (b) Average number of urgent hearings granted by a particular court

   (c) Date of filing for every case

   (d) Remedy/prayer/action sought from the court

   (e) Case stage

(f) Name of the advocate appearing in the matter

(g) Other cases that the advocate is hearing for

(h) Details from the courtroom during hearing - for example, what order/decision was taken to be recorded for subsequent scheduling

1.3 Related Work

Precedence retrieval There has been some work on precedence retrieval. (Al-Kofahi et al., 2001) applied simple machine learning techniques like Support Vector Machine (SVM) to retrieve precedents. (Jackson et al., 2003) presented an information extraction and retrieval system, named History Assistant, to retrieve relevant prior cases. The information retrieval component in this system generated queries which were submitted to a citator database to identify prior cases. Prior research on citation networks and patent retrieval (Lupu and Hanbury, 2013) can also be useful in legal document retrieval. However, there has been comparatively little research by the data science community on legal text mining, such as on applying network analysis, machine learning, etc. on legal text (Al-Kofahi et al., 2001). Advanced deep learning models have also been applied in prior case retrieval (Shao et al., 2020).

Summarization of legal documents There have been a few prior efforts towards summarization of legal documents (Moens, 2007) (Galgani et al., 2012), and the SALOMON project on summarizing Belgian legal documents. More recently an extensive analysis of the state of the art summarization algorithms on legal documents has been published which established the wide scope in the development of novel algorithms to this end (Bhattacharya et al., 2019).

Legal judgment prediction/charge prediction/violation prediction This paradigm refers to the task of predicting the violated statutes, charges (e.g., theft), terms of penalty etc. of the case, given the facts of a case (Chalkidis et al., 2019) (Zhong et al., 2020a) (Chen et al., 2019) (Yang et al., 2019). Advanced machine learning algorithms e.g. Hierarchical BERT (Chalkidis et al., 2019), topological multi-task learning, deep gating network (Chen et al., 2019) etc. has been used to this end. Zhong et al. (Zhong et al., 2020a) proposed to design a QA based task for explainable judgement prediction. However, most of the methods preclude legal-expert level explainability.

Crime Classification Wang et al. (Wang et al., 2018) (Wang et al., 2019) employed advanced neural network models using Multi-task learning, multi-label classification etc. for classification of crimes as represented by Articles of Chinese law in legal documents.

Legal Question Answering This task refers to automatic professional-level catering to several legal questions usually asked by laymen. Many algorithms (Bach et al., 2017) (Kim and Goebel, 2017) have been proposed. However, experiments by (Zhong et al., 2020b) show that there is ample room for considerable improvement in the state of the art.

Commercial legal search systems There are some existing proprietary legal search systems that allow search for prior cases, as well as provide summaries of the prior cases, e.g., LexisNexis, WestLaw. LexisNexis[1] also provides technology-assisted legal services. WestLaw[2], a product of Thomson Reuters, is a similar proprietary legal assistance system. IBM Watson has recently designed ROSS[3] which is supposed

[1] https://www.lexisnexis.com/en-us/products/analyzer.page

[2] https://legalsolutions.thomsonreuters.com/law-products/westlaw-legal-research/

[3] https://rossintelligence.com/

to assist legal experts. However, these legal systems are proprietary software, and come with very high subscription costs which are often beyond the budget of law experts and the common man alike. Additionally, these systems are meant for use by law practitioners, and are not suitable for use by common people without domain knowledge.

1.4 Summary of Work Done

1. **Literature Survey**: A comprehensive survey of the current legal system in place was done to understand the improvements that we needed to focus on. The survey showed that while the past decade has seen many technological improvements in the legal system in India, court scheduling is still a novel problem on which no substantial work has been done till date.

2. **Dataset collection**: We have downloaded causelists (a list of cases which will be heard on any particular court day in a particular court establishment) and orders (a summary of what happened in the hearing written by the for a period of 10 years from the CHC, MCCC and SCoI websites by web scraping. The dataset consists of about 5000 causelists, 90,000 orders/judgements and 40,000 case histories spanning multiple courts.

3. **Statistical Analyses on dataset**: We have extracted the cases from the causelists and performed various analyses on them. Further study showed that a lot of these cases were listed in the causelist without any actual hearing pertaining to them because each causelist contained an enormous number of cases. Most of the cases listed in a causelist cannot be heard due to time being insufficient or parties being absent, which leads to the cases being rolled over to a later date. Also, the number of days between consecutive listings of a case in causelists vastly depended on its case type, year of institution and various other factors. This shows that the current system of scheduling court cases is

very inefficient, and leads to high pendency of cases, inequitable distribution of judicial time among case types and a lot of time wasted for the parties and advocates involved.

4. **Next State Predictor**: Using bigram and n-gram probabilities, we have built simple next state predictors for the MCCC cases. These predictors show that the probability of a state to repeat itself in the next state is almost 1 in most of the cases.

5. **Encoding a case as a chain of nodes**: We have proposed the concept of case traces which enable us to succinctly store the data pertaining to any case in the causelists. We have also proposed a way to summarize a case trace to form a reduced trace which holds all the previous info but is 3-4 times shorter in length.

6. **Disambiguation of case states**: There were many similar case states collected from the causelists that were very similar to each other in spelling and type. We have manually mapped all the obtained case states to certain number of general case states at 2 different abstraction levels: **Standard Abstraction Level** and **Detailed Abstraction Level**. All the case and reduced traces were recomputed at both these abstraction levels.

7. **Comparing case histories and reduced traces**: We have compared the length of a case history (list of orders for the case) with its corresponding case trace and reduced trace and observed that the ratio of length of case history to the length of reduced trace is lies between 0.7 and 1.3 which substantiates our hope of the reduced trace being similar to the case history

8. **Computing Similarity of Cases**: We have used edit distance between traces to compute similarity between any two cases. We have computed similarity using both case traces and reduced traces and compared these 2. These similarity scores has been used to cluster the cases.

9. **Integrating Orders into computing similarity**: We have taken the text in orders corresponding to a node in the case/reduced trace and used cosine similarity on it. This cosine similarity of orders has also been integrated into the edit distance calculation between two traces. These similarities were also used to cluster the cases.

10. **Comparing Top-K similar cases obtained from case and reduced trace similarities**: We have obtained Top-K similar cases for a case using their case trace similarity. We have also used the reduced trace similarity to find another list of Top-K similar cases. We have compared both these Top-K lists to show that reduced trace similarity gives as good results as case trace similarity for our case similarity formulation.

11. **Ablation Study on the effect on presence of Senior Advocates on the pendency of cases**: The type of advocates (Advocate/Senior Advocate) of both parties of the case were extracted from orders of case and we studied the effect of the presence of senior advocates on the number of years a case has been pendent for each case type.

1.5 Layout of the report

The Introduction (Chapter 1) of the report outlines the Background for the project, the Goals of the project and a Summary of the work done under this project.

The Review of Current Legal Systems in India (Chapter 2) of the report outlines a summary of the ICT enabling legal systems currently in India including eCourts Project, CIS and JustIS mobile app.

The Data Collection (Chapter 3) of the report describes all the data collected from the CHC, MCCC and SCoI including causelists, orders, judgements, case histories and a proposed database structure to store all this information.

The Statistical Analyses on Dataset (Chapter 4) of the report explains the various preliminary statistical analyses done on the data collected.

The Case Trace and Reduced Trace (Chapter 5) of the report proposes a way of encoding a case as a chain of nodes from the information collected from the causelists of CHC.

The Similarity of Cases (Chapter 6) of the report dives deep into the similarity formulations between case/reduced traces and also integrates orders into the formulated similarity metric.

The Top-K Analysis Metric (Chapter 7) of the report introduces the Top - K Analysis as a metric of evaluation of similarity scores computed.

The Ablation Study of the effect of presence of Senior Advocates on Pendency of Cases (Chapter 8) of the report deals with the results of the ablation study on the presence of senior advocates on the pendency of cases.

The Conclusion and Future Work (Chapter 9) of the report concludes the results of the project and proposes future work to be done.

Chapter 2

Review of Current Legal Systems in India

2.1 eCourts Project

The eCourts Project was conceptualized on the basis of the "National Policy and Action Plan for Implementation of Information and Communication Technology (ICT) in the Indian Judiciary – 2005" submitted by eCommittee, Supreme Court of India with a vision to transform the Indian Judiciary by ICT enablement of Courts.

Ecommittee is a body constituted by the Government of India in pursuance of a proposal received from Hon'ble the Chief Justice of India to constitute a eCommittee to assist him in formulating a National policy on computerization of Indian Judiciary and advise on technological communication and management related changes.

The eCourts Mission Mode Project, is a Pan-India Project, monitored and funded by the Department of Justice, Ministry of Law and Justice, Government of India for the District Courts across the country.

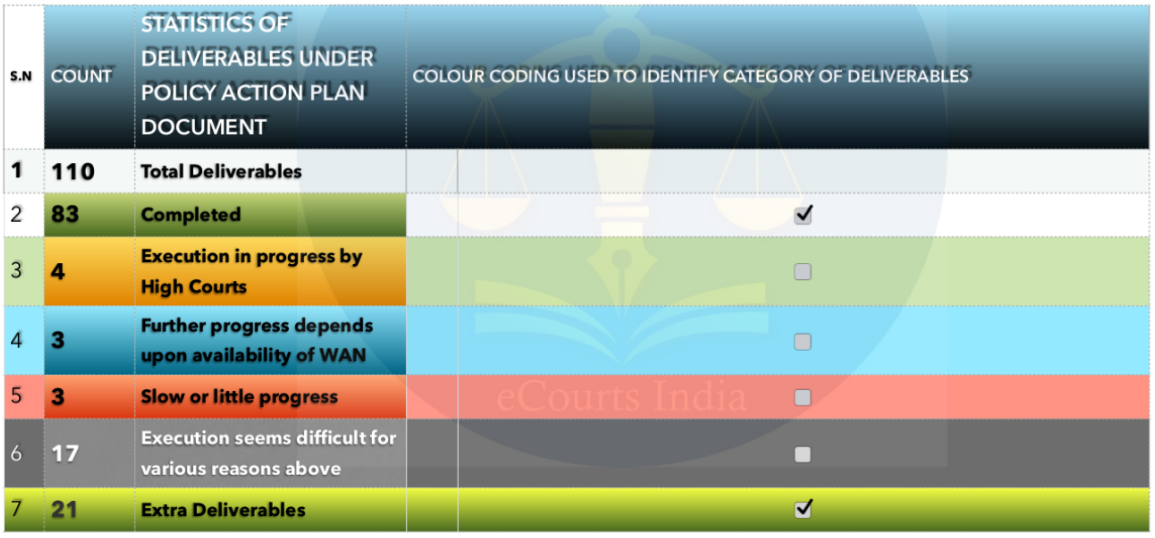
Features of the eCourts Project It is conceptualized and implemented in Free and Open Source Software (FOSS). The project has resulted in an estimated saving of 340 crores. It was implemented using the core-periphery model, which means that the core is sacrosanct, decided by the eCommittee and contains data available for policy and decision making at the national level – Supreme Court, Parliament and Central Government, while the periphery modules developed by each High Court based on its Rules, the Civil and Criminal Court Manuals and the available data in the core.

The project is citizen-centric, keeping the litigant in mind, resulting in remarkable coordination and teamwork between judicial officers and court staff.

Future Available technology has been fully utilized in Phase II of the eCourts Project and several innovations have been made as the Project has progressed. In Phase III of the eCourts Project, consolidation and growth using technological advancements are envisioned including (for example) migration to the cloud (tested and partly implemented already), big data mining and processing through blockchain technology and artificial intelligence. The focus will remain to be affordable and expeditious justice delivery.

Important deliverables accomplished in Phase II

1. Use of Computers by Section of Registry for day to day progresses
2. Unified CIS (Case Information System) for all Courts - It aims to horizontally and vertically integrate the case data of all courts all over India
3. NJDG (National Judicial Data Grid) - It gives the consolidated figures of cases instituted, disposed and the pendency of cases in all courts across the country
4. Unique Identification Codes to Judicial Officers and Court Establishments in the country



S.N	COUNT	STATISTICS OF DELIVERABLES UNDER POLICY ACTION PLAN DOCUMENT	COLOUR CODING USED TO IDENTIFY CATEGORY OF DELIVERABLES
1	110	Total Deliverables	
2	83	Completed	☑
3	4	Execution in progress by High Courts	☐
4	3	Further progress depends upon availability of WAN	☐
5	3	Slow or little progress	☐
6	17	Execution seems difficult for various reasons above	☐
7	21	Extra Deliverables	☑

FIGURE 2.1: Deliverables under the eCourts Project as per Policy

5. eFiling Portals for High Courts and District Courts

6. eCourt Unified Portal - This is one of the most used portals in the country

7. ePayment Gateway - It facilitates accepting payment of Court Fees or Judicial Deposits in the eCourts portal

2.2 Case Management through CIS 3.0

CIS Case Information System software is a giant move under the initiative of the eCommittee to make the Indian Judiciary more transparent and more litigant friendly. The CIS versions are available for District Judiciary and High court exclusively. The CIS Software for District Judiciary is created under the guidance of the e-committee, Supreme Court of India through the software team at National Informatics Center (NIC), Pune.

The whole idea of CIS to put it in a nutshell is that the litigant should be able to view the daily status of his case, to view the orders of the case, hearing date of his

case, the progress of the case on any particular date etc on-line from any part of the world.

Milestones of CIS 2.0

1. Case details/court orders available 24 X 7 mobile app / web
2. Unique CNR number for all the cases
3. QR code can be generated for each and every cases under eCourts.
4. eCourts Transaction raised from 2 crores in 2014 to 42 crores in 2017
5. NJDG houses 7 crore case details and 4 crore Orders/Judgments. eCourts mobile app installation went upto 5 lakhs as on Oct 2017 as per google play store data
6. Number of court orders accessed increased from 64 in 2014 to 3.56 crores in 2017
7. Case details through SMS
8. Getting case status through eCourts automated email services

Need for CIS 3.0 CIS 3.0 has also been developed by NIC Pune to further opens the doors of district judiciary digitalization towards the much awaited e filing, e-payments, and e process; to enhance the existing CIS 2.0 based on the suggestions received for CIS 2.0 from all over the country through all the High courts; and to move a step forward towards the ICJ visionary project of integrating the court, police station, prison.

There are two types of enhancement made by the NIC Pune in CIS 3.0 :

- Technological enhancement - better speed than CIS 2.0
- Functionalities enhancement - more user friendly

Features of CIS 3.0

1. **Bilingual Toggling** - It allows the user to switch over to the bilingual language without logging out
2. **Improved Query-options**
3. **Report generation** - This lets the user generate reports in PDF, Excel, CSV formats.
4. **Kiosk** (Automated information provider) - Any person can get details about the case status using CNR number/Case number/FIR number/Registration number/Party name. Cause lists can be viewed in KIOSK. It is bilingual and comes with regional language support.
5. **CIS 3.0 Core and Periphery Modules** - CIS software created based on the core and periphery model. Under CIS 3.0, more meaningful master modules provided at National, State, Local, periphery level which results in more customization according to the needs of the state and district, but, at the same time, maintaining the standardized features nationwide.
6. **View QR code/Scan QR code** - For every case, can get the QR code from the www. eCourts .gov.in website. With the scan QR code option in the eCourts mobile app or using any other scan QR code app, one can get the details of the case
7. **eFiling module** - Registered users i.e. advocate/party in person can file their digital case content through the e-filing web portal.

2.3 Court Management through JustIS Mobile App

JustIS mobile app is the latest digital court management tool gifted by the eCommittee, Supreme court of India, empowering the judicial officers of the District and

FIGURE 2.2: A comprehensive view of the features of the JustIS Mobile App

Taluk level courts for efficient court management and for speedy administration of justice at District level. This JustIS app gives the exclusive data of a particular court where the judicial officer is working, with data analysis of the entire court data of that court. JustIs app throws more light into further drill down of the disposal and pending case data like Case type wise, Year wise, stage wise disposal and pendency etc from the judicial officer's perspective

Chapter 3

Data Collection

The IST is being developed primarily to be tested and used in the MCCC. However, due to the absence of a lot of data in the MCCC website[1], we have collected large amount of data from the CHC website[2] and the SCoI website[3]. The CHC was chosen for this study primarily because of it applicability to IIT Kharagpur's location as well as for its abundance of causelist and case order data. We collected data for the SCoI because of judgements being available and the because of the hierarchical structure of the Indian Judicial System.

The summary of all the data collection done can be seen in 3.1

3.1 Daily Causelists

A causelist is a legal term for a list of cases awaiting a hearing. Before the court day begins, every court establishment (CHC, MCCC, etc) releases a daily causelist which is the list of cases which will be heard on that court day in that establishment. A typical causelist has a list of cases where for each case, there is a case identity.

[1] https://districts.ecourts.gov.in/mumbai-citycivil-court
[2] https://www.calcuttahighcourt.gov.in/
[3] https://main.sci.gov.in/

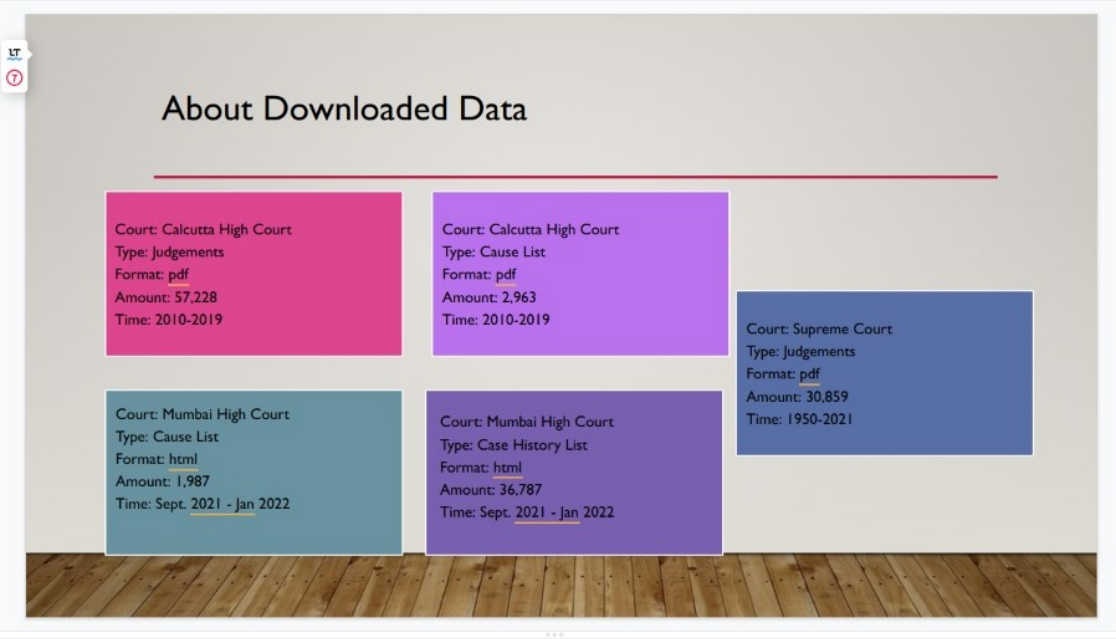


FIGURE 3.1: Summary of the data collected

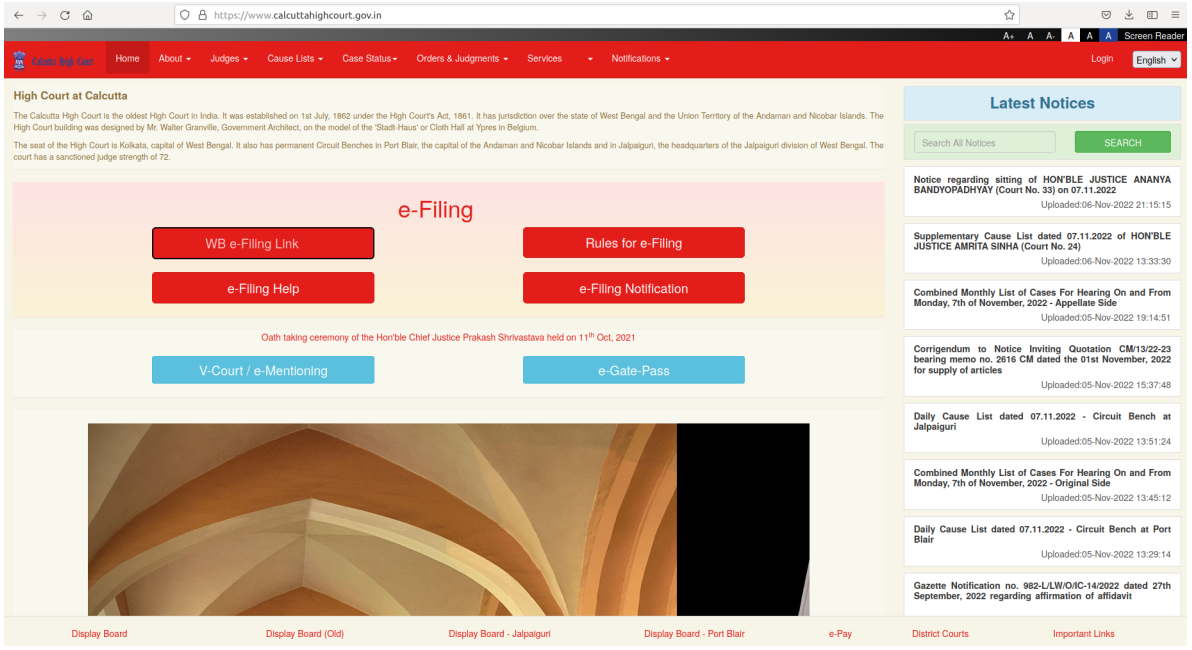


FIGURE 3.2: A snapshot of the data available on the Calcutta High Court website

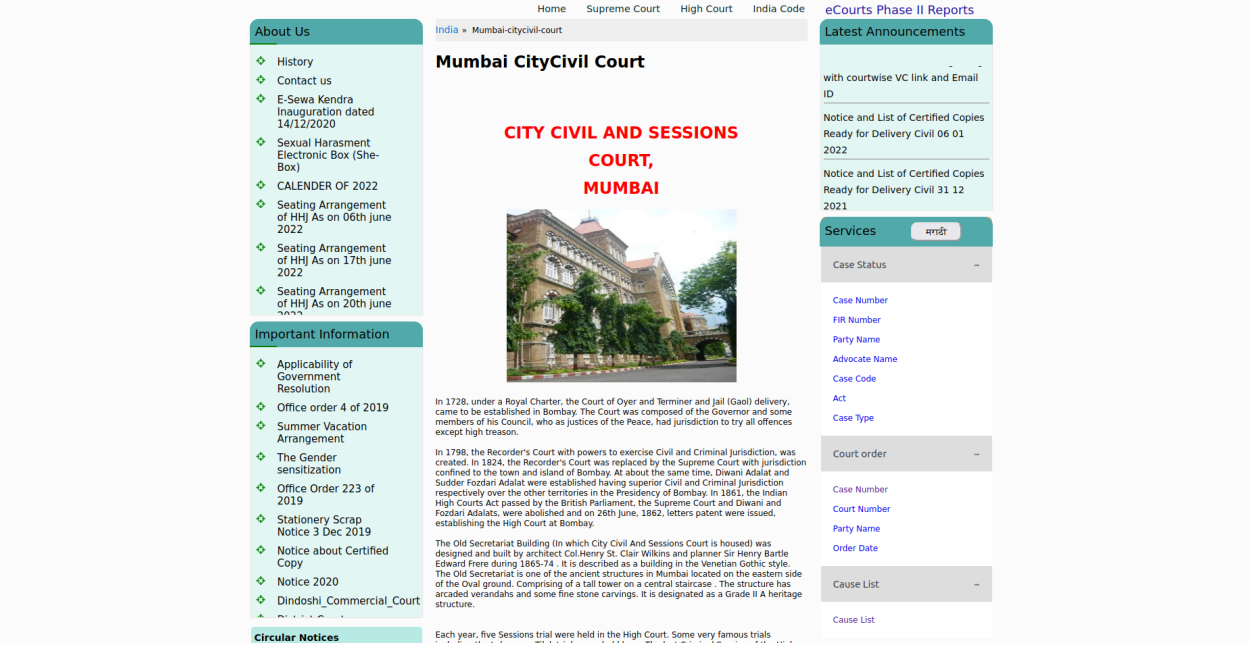


FIGURE 3.3: A snapshot of the data available on the Mumbai City Civil Court website

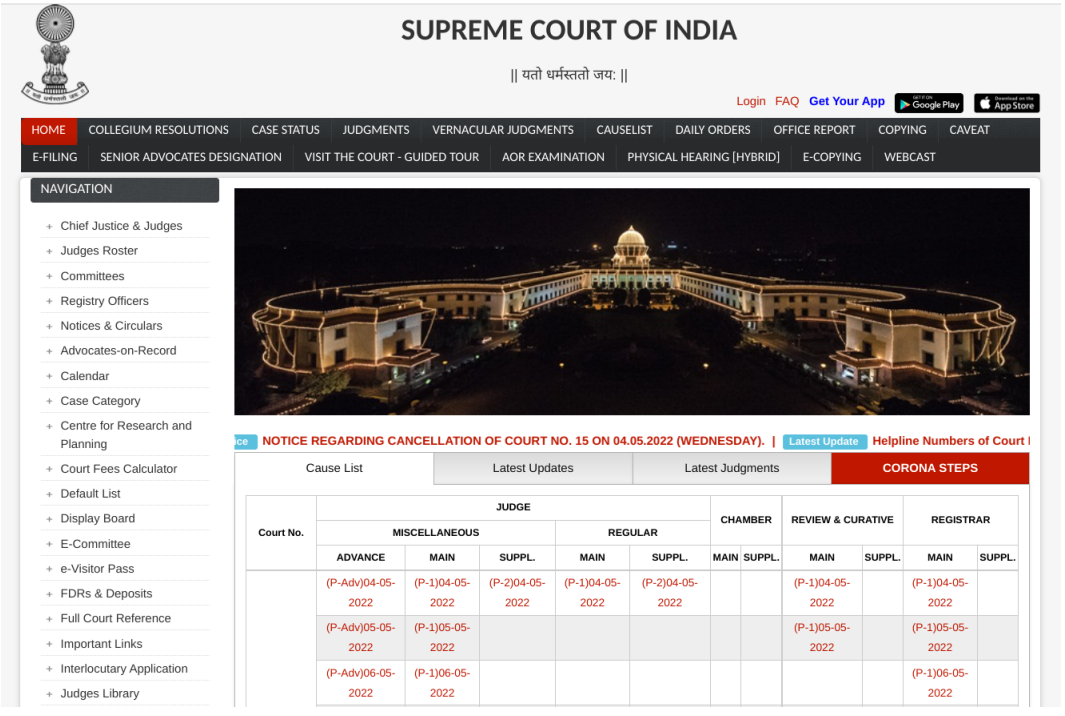


FIGURE 3.4: An snapshot of the data available of the Supreme Court on India website

The case identity is a trifecta of the: [1] **case type**, which is typically a 2-4 letter capital code signifying the type (for example, Civil Case, Appeal, Bail Application, etc); [2] **case number**, which is natural number; and [3] **year of institution**, which is the year in which the case was registered into the legal system. Other than the case identity, every case also has its parties listed in the form of Party 1 vs Party 2 in the causelist.

Calcutta High Court CHC releases daily causelists for every working court day which are available on its website. There are two sides here: [1] **Appellate**, and [2] **Original**. Each side has its own daily causelist which comprises of cases belonging to the particular side. Appellate side cases are the cases which have been passed from lower level courts (for example, District and Sessions Court, etc) to the High Court (here, CHC). Original side cases refers to those cases which have been directly registered under the High Court.

The causelists are typically in the PDF format. CHC comprises of many court hearing halls, in each of which hearings can take place simultaneously. Every daily causelist, here, has a list of court hearing halls and the judge(s) presiding over each hearing hall. Under each hearing hall, there are a list of case states. A case state is the state of the hearing (for example, application, arguments, summon, etc). Under each case state, there is a list of cases which are cases which are in that particular state and are to be heard in that hall (under the judge presiding over the hall). The typical hierarchy of a causelist can be seen in figure 3.5

A causelist does not mention when every case listed is to be heard and typically lists tens or even hundreds of cases under every hearing hall. In a court day, a judge typically begins from the beginning of the causelist (of that hearing hall), hearing cases one by one, in order. Thus, if the case is mentioned in the causelist, the parties and advocates involved need to be present when the judge reaches their case, otherwise the judge simply skips their case and the case is pushed back to a later date (typically the next court day). Also, since the causelist lists hundreds of cases,

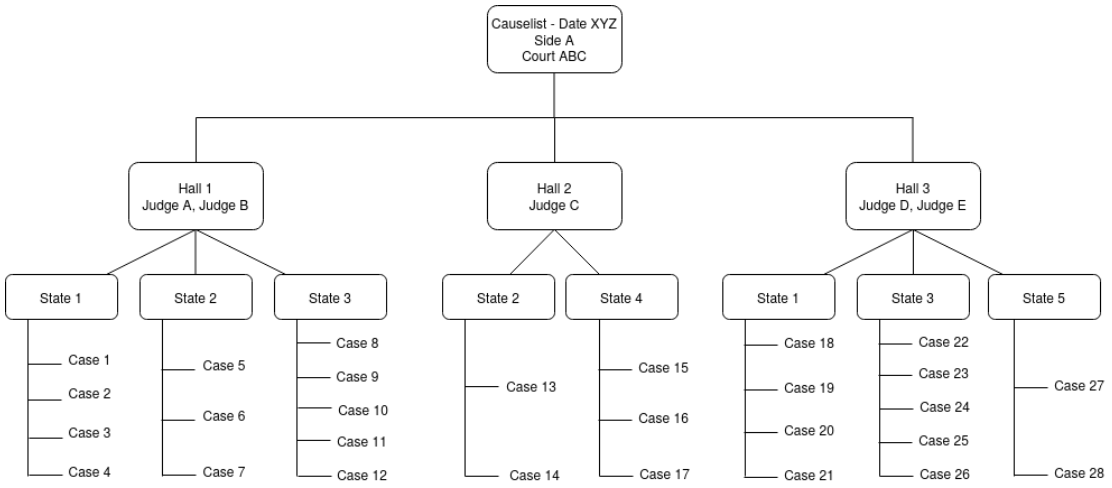


FIGURE 3.5: The hierarchical structure of a typical daily causelist of the CHC

the cases towards the end of the list are not even called and just get rolled over to the next court day. This is extremely wasteful for the parties involved in the case and a waste of judicial resources.

We have collected a data set[4] of all the available daily causelists for the period of 2010-2019 for both Appellate and Original sides. No causelist prior to 2010 is present on the CHC website. Beyond 2019, the data was bound to be affected by the worldwide COVID-19 pandemic, due to which this particular 10 year period was chosen. The algorithm uses a simple **web scraper** implemented using the Selenium web driver. For downloading a causelist, the date has to be selected from a date dropdown menu and the side has to be selected (Appellate/Original). On clicking the download button, the causelist is downloaded. Each of these manual actions can be easily replicated using web scraping.

Mumbai City Civil Court Causelist collection from the MCCC causelist website[5] is much more complicated because of the use of captcha recognition for downloading any causelist data. Moreover, unlike the CHC website, downloading a single

[4] `https://drive.google.com/drive/folders/1L8C1RcCSzq-hMkKfbC508BN5M1fx9gqp?usp=sharing`

[5] `https://services.ecourts.gov.in/ecourtindia_v4_bilingual/cases/dailyboard.php?state=D&state_cd=1&dist_cd=37`

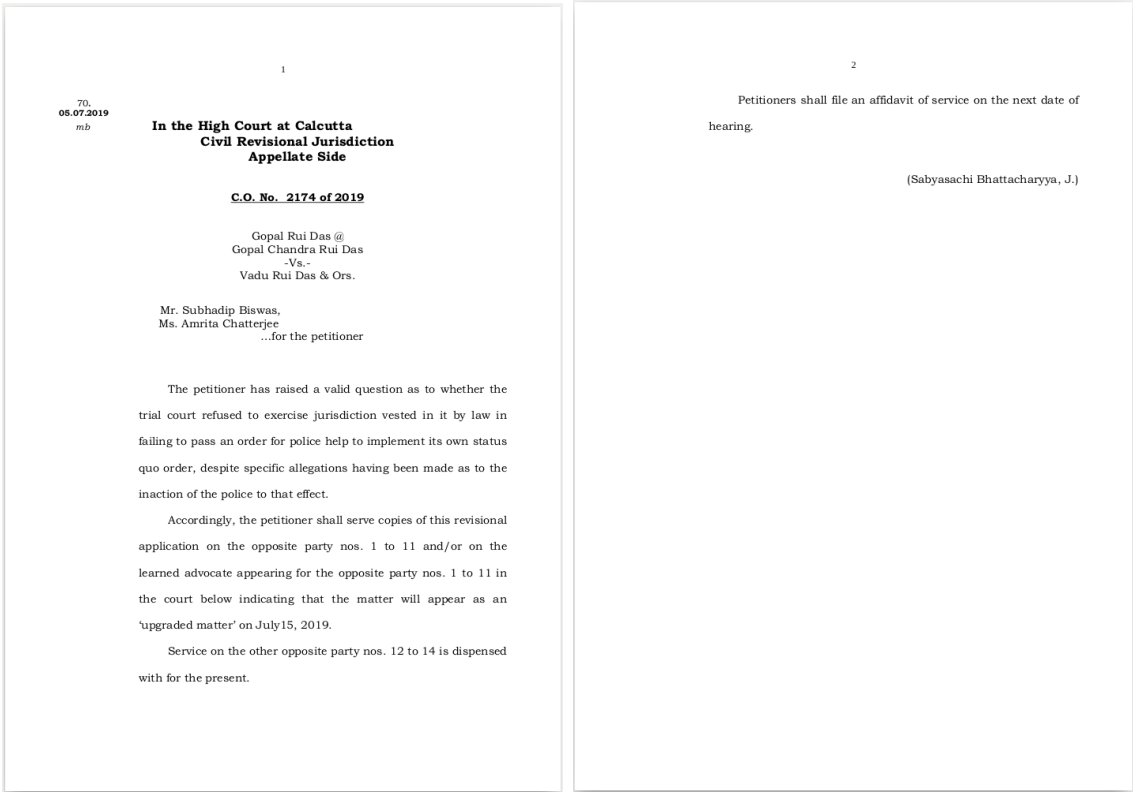
daily causelist for the entire MCCC under all the judges is not possible, the cases under each judge in any court complex have to be downloaded separately.

The daily causelists per court complex per judge have to be downloaded using web scraping which uses captcha recognition software. Each of this is an HTML file, unlike the causelists collected from the CHC website. We have developed a captcha recognition software which uses the PIL python package to capture the captcha image and uses the pytesseract python package to extract the text from the image. The algorithm is fine tuned to be have an accuracy of about 70 %. The algorithm works such that it keeps retrying the same download until the captcha output given by the captcha recognizer is correct and the download succeeds. A dataset has been downloaded consisting of 400 daily causelists (each causelist a culmination of all the the sub causelists per court complex per judge) from the period of 21-08-2021 to 18-12-2021. Much more causelists could not be collected because the MCCC website only has the causelist data of the past 7 days on its server on any given day.

3.2 Orders/Judgements

An order is an official record of all that happens in a particular hearing of a case. It is a summary of the outcome of the hearing. Orders are very important, since, frequently, a case is listed in the causelist but it does not have any order corresponding to that day. This typically means that the case was not heard on that day. While the presence of a case order in itself, is a big indicator as to the case hearing, the actual content can sometimes point out that the hearing was fruitless. Many case orders simply state that the hearing was adjourned to a later date due to absence of one or both parties/advocates or the failure to produce certain required documents, proof, etc. This tells us that while the case hearing happened, nothing tangible happened during the hearing. We hope to analyse this and develop a system that can handle these issue online so that judicial time is utilised more usefully.

Calcutta High Court An order exists for every case hearing. The order is usually a long text document which lists the case name, the parties involved and the judge(s) presiding over the hearing. The rest of the order vastly varies in length as well as actual content, because it largely depends on the judge as well as the case type and the case state. Many judges prefer to write concise orders while others prefer much detailed orders. A typical case order for an original side case can be seen in figures 3.6(a) and 3.6(b) below.



1

70.
05.07.2019
mb

In the High Court at Calcutta
Civil Revisional Jurisdiction
Appellate Side

C.O. No. 2174 of 2019

Gopal Rui Das @
Gopal Chandra Rui Das
-Vs.-
Vadu Rui Das & Ors.

Mr. Subhadip Biswas,
Ms. Amrita Chatterjee
...for the petitioner

The petitioner has raised a valid question as to whether the trial court refused to exercise jurisdiction vested in it by law in failing to pass an order for police help to implement its own status quo order, despite specific allegations having been made as to the inaction of the police to that effect.

Accordingly, the petitioner shall serve copies of this revisional application on the opposite party nos. 1 to 11 and/or on the learned advocate appearing for the opposite party nos. 1 to 11 in the court below indicating that the matter will appear as an 'upgraded matter' on July15, 2019.

Service on the other opposite party nos. 12 to 14 is dispensed with for the present.

2

Petitioners shall file an affidavit of service on the next date of hearing.

(Sabyasachi Bhattacharyya, J.)

(a) Page 1 (b) Page 2

FIGURE 3.6: An example court order obtained from the CHC website

We have designed an algorithm which uses web scraping (implemented using Selenium) to download the case orders for every case. For downloading an order from the CHC website, we need to choose the side (Appellate/Original) as well the case identity (see section 3.1). Each of the side, case type and year of institution are chosen from their respective dropdown menus. The case number is simple input box. Since, we have considered the time period of 2010 to 2019 for the causelist

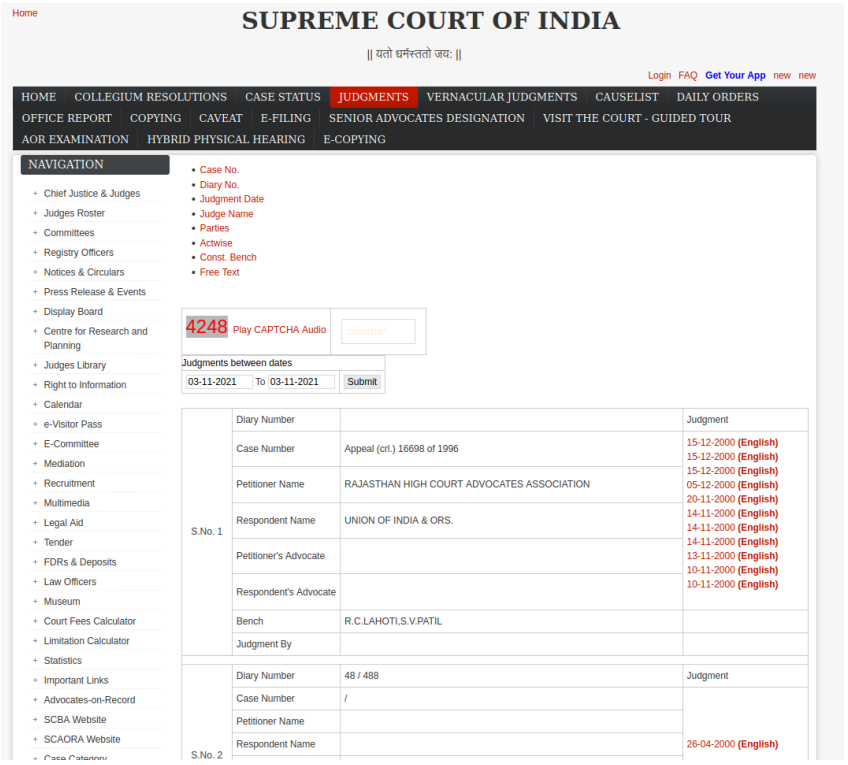


FIGURE 3.7: A sample HTML page downloaded from SCoI website with the judgements

collection, we also limit ourselves to orders pertaining to this time period. Consequently, we set the from and to date in the date dropdown menus as 01-01-2010 and 31-12-2019 respectively. Finally, on clicking the search button, we get all the case orders for that case in our 10 year time period. A thing to keep in mind is that the side selected should match the actual case side for the orders to be displayed, otherwise, we get an empty list.

For the order collection, we need the list of case identities which is obtained from the causelists. The causelists contain about 600,000 unique appellate cases and 100,000 original cases which have been extracted after analysing the causelists using a PDF reader in python. The order collection for the original cases is currently in progress. We have collected 57,228 orders for the original cases.[6] The order collection of the appellate cases has yet to be formally started.

[6]The complete data can be viewed here: `https://drive.google.com/drive/folders/1_3i62Z8Gws2QRz_FhuaB8Mf3TblUfM6S?usp=sharing`

Supreme Court of India Daily order collection from the SCoI website is very similar to the MCCC causelist collection because it also requires a captcha. In addition, we need the case type, case number and the year of institution. This will give us a list of all the judgements of the case available on the SCoI server. We have developed an algorithm to download these judgements given an individual case and have downloaded 30,859 judgements as of now. These judgements are in the PDF format and span a large number of year from 1950-2021. The data collected is in the form of HTML pages as shown in 3.7

3.3 Case History

Mumbai City Civil Court The MCCC website has a unique feature which the CHC website does not have. It can show the entire case history of any given case. The case history is basically the list of hearings the case has had from its institution till its disposition (when the case's final verdict is given), or till date, if the case has not been disposed yet. This data is collected while collecting the causelists from the MCCC website. If a certain daily causelist (per court complex per judge) is present, the case histories of all the cases listed in the causelist can be extracted by clicking on each case present in the causelist, which leads to an HTML page that is downloaded. There is captcha recognition required again to get the case history downloaded.

A listing of a hearing in the case history is different from a case listing in a causelist. If there is a case hearing on a certain day for a case, that case will also have been listed in the daily causelist. However, as explained already in the section 3.2, if a case is listed in the causelist, it does not mean it was actually heard. Thus, a listing of a hearing in the case history is analogous to the presence of an order for the case on that particular hearing date. Every listing of a hearing consists the case registration number, judge name (i.e, the judge presiding over the hearing), the hearing date,

Case History

Registration Number	Judge	Business on Date	Hearing Date	Purpose of Hearing
100773/2018	COURT 1 ADDL SESSIONS JUDGE	24-04-2018	31-07-2018	CHAMBER SUMMONS
100773/2018	COURT 1 ADDL SESSIONS JUDGE	31-07-2018	21-09-2018	CHAMBER SUMMONS
100773/2018	COURT 1 ADDL SESSIONS JUDGE	21-09-2018	18-12-2018	CHAMBER SUMMONS
100773/2018	COURT 1 ADDL SESSIONS JUDGE	18-12-2018	21-02-2019	CHAMBER SUMMONS
100773/2018	COURT 1 ADDL SESSIONS JUDGE	21-02-2019	16-04-2019	C/S ARGUMENTS
100773/2018	COURT 1 ADDL SESSIONS JUDGE	16-04-2019	12-07-2019	C/S ARGUMENTS
100773/2018	COURT 1 ADDL SESSIONS JUDGE	12-07-2019	16-09-2019	C/S HEARING
100773/2018	COURT 1 ADDL SESSIONS JUDGE	12-07-2019	16-09-2019	C/S HEARING
100773/2018	COURT 1 ADDL SESSIONS JUDGE	16-09-2019	05-11-2019	NM FOR HEARING
100773/2018	COURT 1 ADDL SESSIONS JUDGE	05-11-2019	20-12-2019	NM FOR HEARING
100773/2018	COURT 1 ADDL SESSIONS JUDGE	20-12-2019	13-01-2020	NM FOR HEARING
100773/2018	COURT 1 ADDL SESSIONS JUDGE	13-01-2020	19-03-2020	C/S HEARING
100773/2018	COURT 1 ADDL SESSIONS JUDGE	13-01-2020	19-03-2020	C/S HEARING
100773/2018	COURT 1 ADDL SESSIONS JUDGE	19-03-2020	26-06-2020	C/S HEARING
100773/2018	COURT 1 ADDL SESSIONS JUDGE	26-06-2020	01-10-2020	C/S HEARING
100773/2018	COURT 1 ADDL SESSIONS JUDGE	01-10-2020	18-12-2020	C/S HEARING
100773/2018	COURT 1 ADDL SESSIONS JUDGE	18-12-2020	24-02-2021	C/S HEARING
100773/2018	COURT 1 ADDL SESSIONS JUDGE	24-02-2021	15-04-2021	C/S HEARING
100773/2018	COURT 1 ADDL SESSIONS JUDGE	15-04-2021	23-07-2021	C/S HEARING

FIGURE 3.8: An example case history obtained from the MCCC website

link to the order of the earlier hearing date and the purpose/state of the hearing. A typical case history obtained from the MCCC website is shown in figure 3.8

3.4 Database Structure

Calcutta High Court We have designed a first level database for the causelist and order data collected from the CHC, MCCC and SCoI websites. The ER Diagram

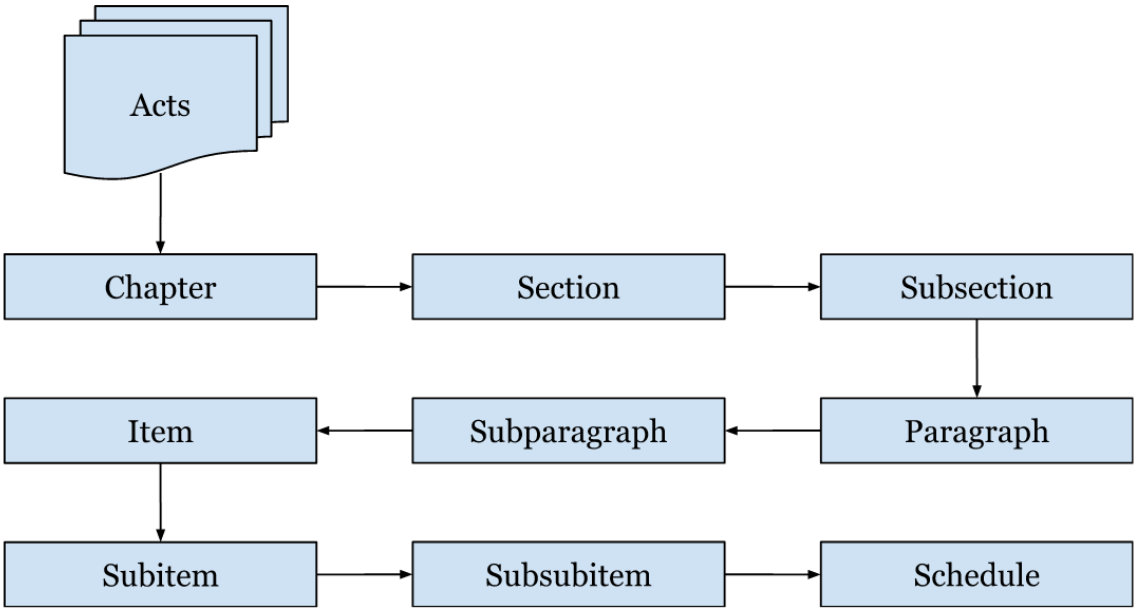
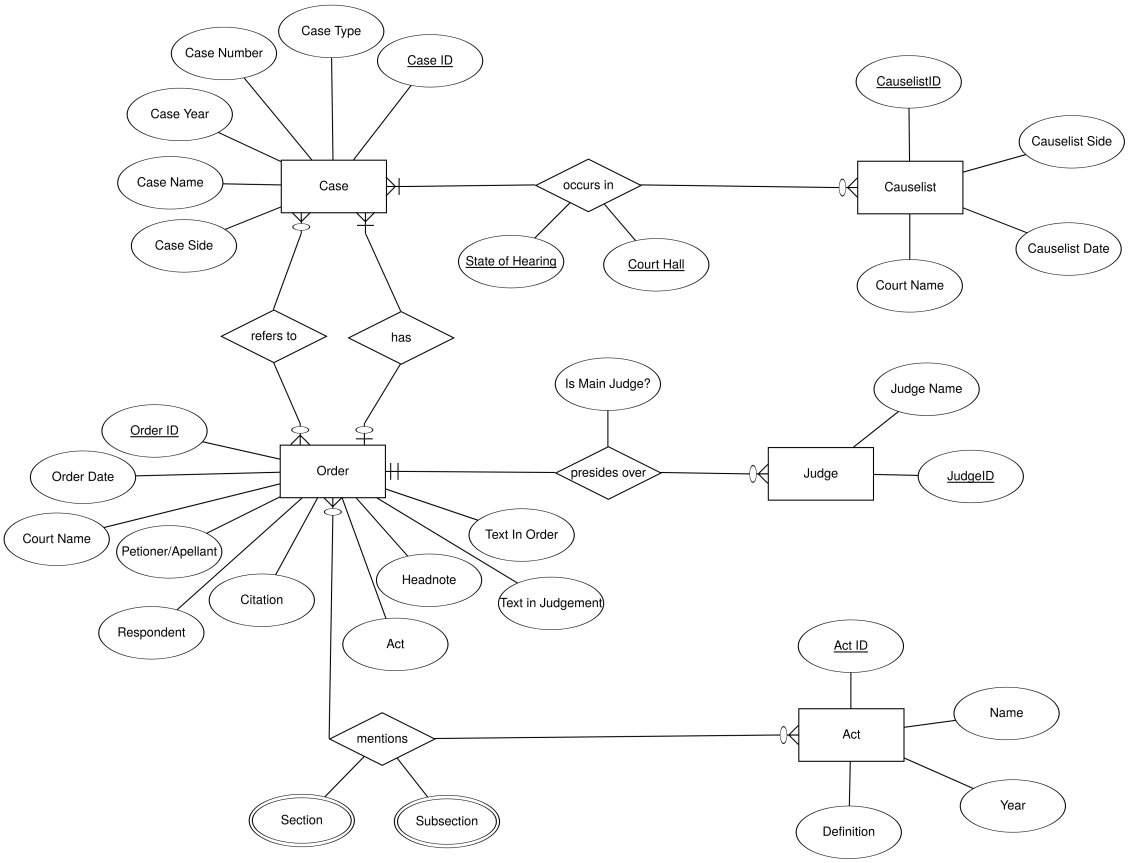


FIGURE 3.9: The hierarchy of an Act

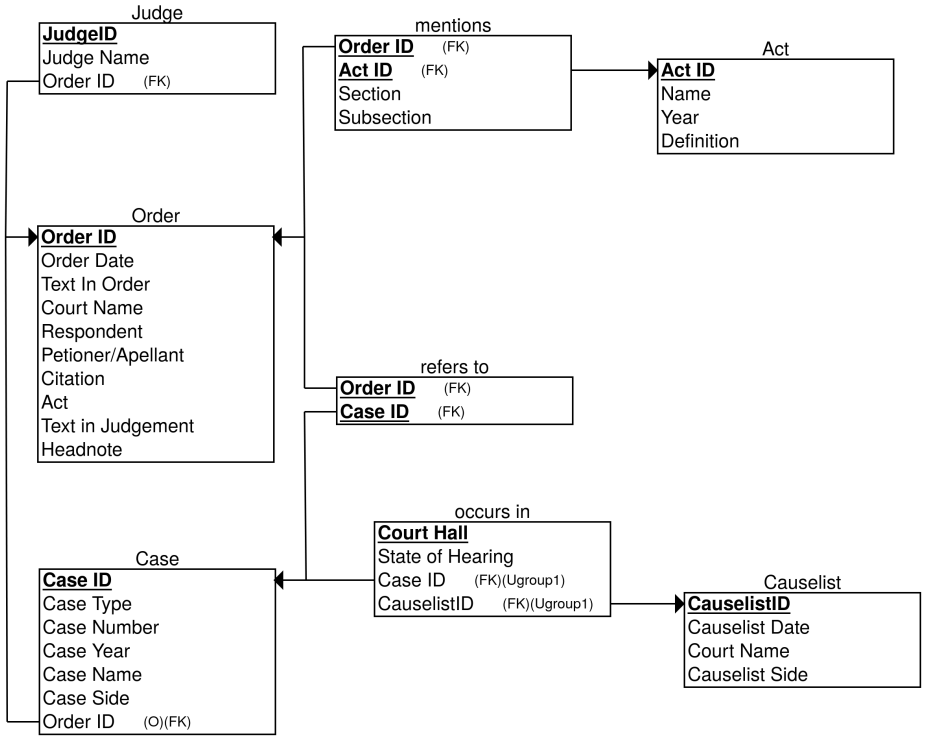
and DB schema can be seen in figure 3.10(a) and 3.10(b) below. The entities present in this schema are Causelists, Orders, Cases, Judges and Acts. For each of these, we have introduced an alphanumeric ID to uniquely identity each instance of them.

An Act is a decree that is approved by the respective legislature, i.e. in India's case, the State Legislative Assembly or Parliament of India. It is a sub-set of a law. Most of the orders/judgements in CHC, MCCC and SCoI mention acts in them while declaring a verdict, stating a precedent, etc. The hierarchical parts of an act is shown in 3.9. The rest of the schema can be explained by the explanations of causelist and orders given in sections 3.1 and 3.2.

The original database schemas which was built primarily for CHC data would not be able to store the MCCC and the SCoI data as well. The modified database schema will work to store the data from the MCCC and SCoI as well.



(a) ER Diagram



(b) DB Schema

FIGURE 3.10: First level database schema for court data

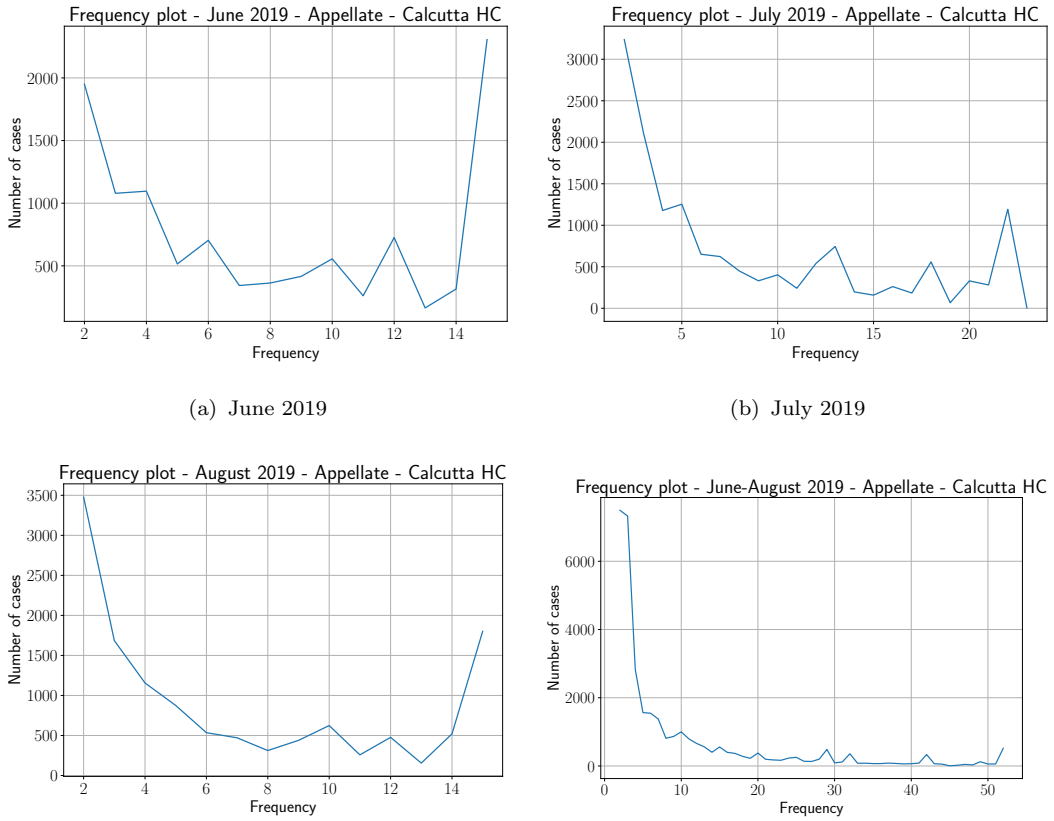
Chapter 4

Statistical Analyses on Dataset

4.1 Analyses on Causelists collected from CHC

Analysis on Jun - Aug 2019 The causelists of both the appellate and the original side were initially manually downloaded for a period of three months (June 2019 - August 2019). The case repetitions in this period were analysed and it showed many cases being repeated as many as 40-45 times in period of about 60 working court days. The case listings based on case types also vary vastly. This clearly shows that the current system of designing the causelists is very wasteful and inequitable since most of the cases listed are simply not heard and are rolled over to the next date. The frequency plots for this data can be seen in figures 4.1 and 4.2 below.

Analysis on Original Causelists of 2010 - 2019 The causelists of the original side of the period 2010 - 2019 (about 1500 causelists) was used for this analysis. A similar analysis as the June 2019 - July 2019 analysis was done and it still shows many cases being repeated many times in this period when it was not heard for that many days. The case listings based on case types also vary vastly. This plot reinforces our belief that the current system of allocating cases to a causelist needs



(a) June 2019

(b) July 2019

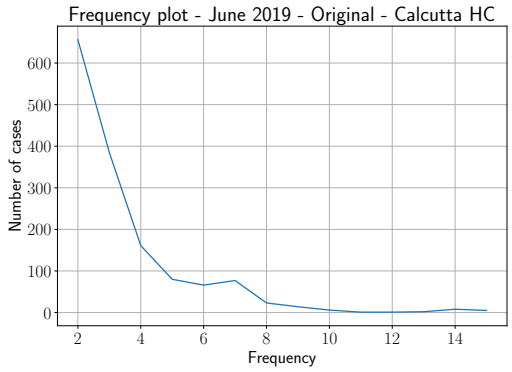
(c) August 2019

(d) June 2019 - August 2019

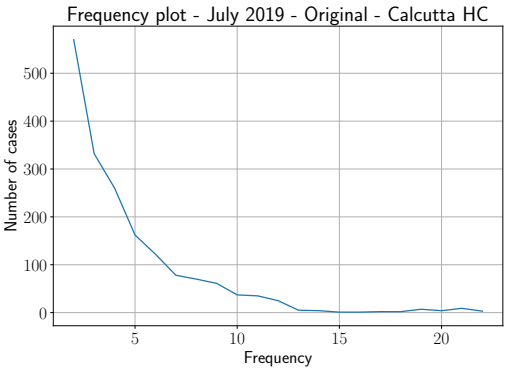
FIGURE 4.1: Frequency plots for Appellate cases in CHC for the period Jun - Aug 2019

a lot of improvement. The frequency plot for this data can be seen in figures 4.3(a) and 4.3(b) below.

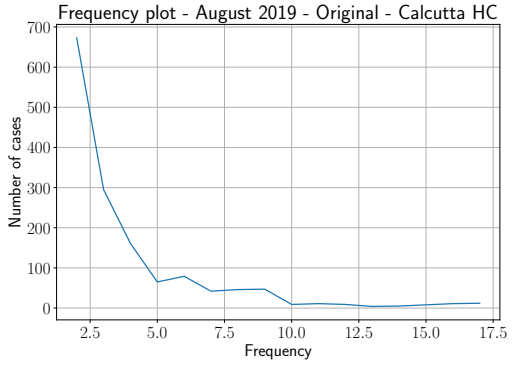
Extracting the cases for the period 2010 - 2019 The causelists were analysed to extract all the unique case identities. The appellate and original side causelists are handled separately since a case belongs to only one side. Since the number of cases in this 10 year period on both the Appellate and Original sides would be quite large, it was not possible to directly extract the case identities using python because the data structure storing the case identities (either a set or a dictionary) would require much more memory than is present in the RAM of a typical personal computer. To overcome this problem, we directly stored every case extracted (unique



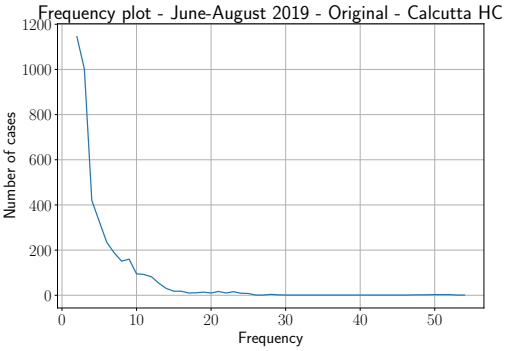
(a) June 2019



(b) July 2019

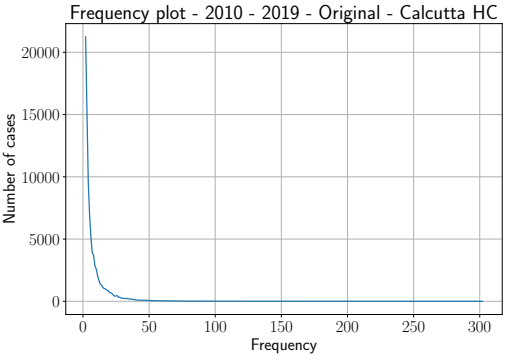


(c) August 2019

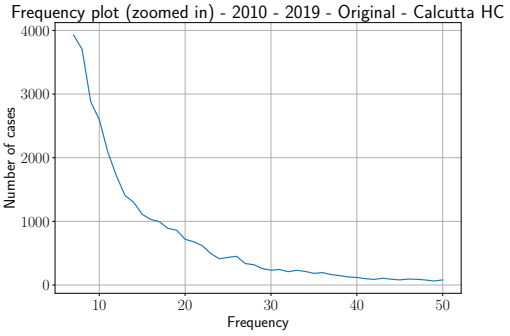


(d) June 2019 - August 2019

FIGURE 4.2: Frequency plots for Original cases in CHC for the period Jun - Aug 2019



(a) 2010 - 2019



(b) 2010 - 2019 (zoomed in)

or otherwise) from any causelist in a text file instead of a data structure. This text file for the appellate side on completion of the algorithm had about 9,000,000 cases (with repetitions), while the text file for the original side had about 900,000 cases (with repetitions). The unique cases were then obtained from these text files by using the shell script. Without repetitions, the causelists amounted to about 600,000 unique appellate cases and 100,000 unique original cases which were listed in the causelists in our 10 year period. These results can be viewed in table 4.1.

We observe that the number of unique appellate cases is about 6 times larger than the number of unique original cases, which tells us that there are far more cases which are passed on from lower courts to the CHC than the cases directly instituted into the CHC. This is intuitive because the CHC allows only certain types of cases (relating to higher importance/relevance) to be directly instituted into it.

We also observe that average number of repetitions per appellate case is about 15, while for an original case, it is about 9. This shows that appellate cases appear more frequently in the causelists, when compared to original cases. This probably means that appellate cases appear very often in the causelists, are not heard most of the times and rolled over to the next court date leading to high inefficiency in court case scheduling.

TABLE 4.1: Counts of cases in CHC causelists (2010-2019)

Side	Total number of cases (with repetitions)	Number of unique cases (without repetitions)	Average number of repetitions per case
Appellate	8,927,705	585,419	15.25
Original	873,168	95,968	9.10

4.2 Next Case State Predictor

Mumbai City Civil Court We analysed the case histories obtained and found that 6 acts (case types) were most prevalent among the cases. We build a simple

Table 4.2: Next State Predictor using Bigram Probabilities

Act	State	Most Probable State (with prob and avg no. of days b/w hearings)	Second Most Probable State (with prob and avg no. of days b/w hearings)
Hindu Marriage Act	Hearing	Hearing 0.978891 830	Hearing on Preliminary Issue 0.059490 959
Specific Relief Act	Order	Order 0.971429 43	Arguments 0.014286 111
Protection of Women from Domestic Violence Act	Arguments	Arguments 0.984984 955	Order 0.002691 968
Code of Civil Procedure	Written Statement	Written Statement 0.906250 77	Reply 0.046875 86
Public Premises Act	Framing Issues	Framing Issues 0.978429 1043	Framing Issues P.H. 0.027269 1975

Table 4.3: Next State Predictor using N-gram Probabilities

Act	State	Most Probable State (with prob and avg no. of days b/w hearings)	Second Most Probable State (with prob and avg no. of days b/w hearings)
Bombay Tenancy and Agricultural Land Act	Hearing	Hearing 0.970820 755	NM Hearing 0.004370 770
Bombay Tenancy and Agricultural Land Act	NM Hearing	NM Hearing 0.965405 770	NM Orders 0.006990 770

case state predictor which analyses the case histories of the cases of a particular type/act among the 6 most prevalent ones. On each of these acts, for each case state, we can find the probabilities that a given state is the next state using the bigram counts of the these case states in the case histories. This idea is leveraged to form the case state predictor, which essentially takes as input a case state and gives the output as a list of states with their probabilities of being the next state arranged in descending order based on the probabilities. An extract of the bigram

case state predictor data is shown in table 4.2[1].

We have also built a next state predictor using the n-gram counts of these case states in the MCCC case histories. An extract of the n-gram case state predictor is shown in the table 4.3.

From the data, we observe that for all the states in each act, using both the bigram and the n-gram probabilities, the most probable next state is the state itself. This means that the case is very likely to keep looping in the same state for a very long time. The average number of days between 2 consecutive hearings of the same state is also very large (almost 2-5 years) which shows that the progress in these cases in MCCC is extremely slow. This in turn leads to very large pendencies of cases (time between institution and disposition of a case) which is a matter of concern and further solidifies the need for intelligent court scheduling.

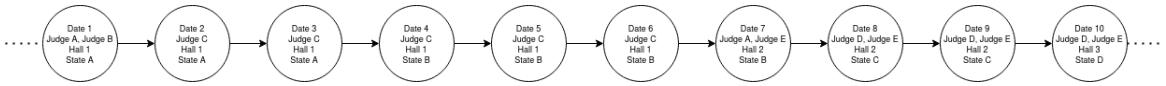
[1]The full bigram data can be viewed here: `https://docs.google.com/spreadsheets/d/1kfYEIOyUB2Pyi_BYglZS_4sxJXd4f60WX01XJ9AEkdc/edit?usp=sharing`

Chapter 5

Case Trace and Reduced Trace

5.1 Building Case Traces and Reduced Traces from Causelists of CHC

A case trace is analogous to a case history of an MCCC case, except that instead of a list of hearings, it is a list of appearances of a particular case in the daily causelists. Let us call an appearance of a case in a daily causelist to be a node. Every node has the date, court hall, judge name(s) and the state of the hearing associated with it. A case trace is a list of nodes where the dates associated with the nodes are in sorted order. The algorithm to build the case trace is shown in algorithm 1.



(a) Case trace

(b) Reduced trace

FIGURE 5.1: An example case trace and corresponding reduced trace

Algorithm 1 Build all case traces of a given case side

Require: Side, Causelists of given side, List of unique cases of given side

1: Initialize empty traces for all the cases of the given side from the list of unique case identities
2: Sort all causelists in ascending order based on the dates
3: **for** each causelist C **do**
4: $currDate \leftarrow$ Date of causelist
5: $currJudge \leftarrow NULL$
6: $currHall \leftarrow NULL$
7: $currState \leftarrow NULL$
8: **for** each text line L in C **do**
9: **if** L contains new court hall **then**
10: $currHall \leftarrow$ new court hall
11: $currJudge \leftarrow NULL$
12: **else**
13: **if** L contains new judge **then**
14: Append new judge to $currJudge$
15: **else**
16: **if** L contains new state **then**
17: $currState \leftarrow$ new state
18: **else**
19: **if** L contains new case and case trace present for new case **then**
20: Append to the case trace of the new case a new node ($currDate$, $currHall$, $currJudge$, $currState$)
21: **else**
22: Go to next line
23: **end if**
24: **end if**
25: **end if**
26: **end if**
27: **end for**
28: **end for**

Reduced Case Traces As we mentioned earlier, many a times, cases get rolled over to the next court day without any actual hearing. So in the case trace, there will be no change from one node to the next node, except that the date associated changes (becomes a later date) for this condition. Due to this, the case trace becomes unnecessarily long. We can compress consecutive nodes with every attribute (other than the date) the same, to form a **reduced trace** of the case. Every node in the reduced trace has a list of dates associated with it. A certain node can have the list

with just one date in it as well. Building the reduced trace from any given case trace is a straightforward algorithm. An example case trace and its corresponding reduced trace is shown in figures 5.1(a) and 5.1(b). The algorithm to build the reduced case trace is shown in 2

Algorithm 2 Build a reduced trace of a given case trace

Require: Case trace
1: Initialize empty reduced traces corresponding to the given case trace
2: $currDate \leftarrow NULL$
3: $currJudge \leftarrow NULL$
4: $currHall \leftarrow NULL$
5: $currState \leftarrow NULL$
6: **for** each node n **do**
7: **if** (n[date] == n[court hall] == n[judge] == $n[state] == NULL$) **then**
8: $currDate \leftarrow n$[date]
9: $currHall \leftarrow n$[court hall]
10: $currJudge \leftarrow n$[judge]
11: $currState \leftarrow n$[state]
12: **else**
13: **if** ((n[court hall], n[judge], $n[state]$) == ($currHall$, $currJudge$, $currState$)) **then**
14: Append n[date] to $currDate$
15: **else**
16: Append to the reduced trace a new node ($currDate$, $currHall$, $currJudge$, $currState$)
17: $currDate \leftarrow NULL$
18: $currJudge \leftarrow NULL$
19: $currHall \leftarrow NULL$
20: $currState \leftarrow NULL$
21: **end if**
22: **end if**
23: **end for**

Work done and Observations The case traces of all the original cases[1] (about 100,000 cases) from the CHC 10 year causelist dataset has been built. The case traces of some appellate cases (about 20,000 cases) from a 3 month period of March - May 2011 have also been built. The average case trace length for the appellate

[1] `https://drive.google.com/drive/folders/1-557x6YlE7VJgS2td4K3ytwKIOsZYeXF?usp=sharing`

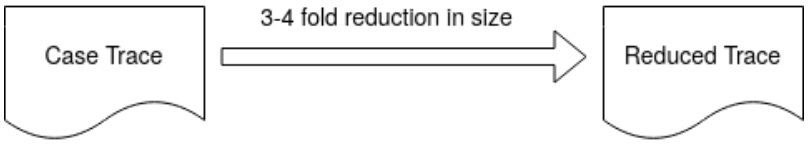


FIGURE 5.2: Reduction in length from case trace to reduced trace

cases is lesser than the original cases because only a part of the CHC dataset was considered for the appellate traces (March - May 2011). The reduced case traces[2] have also been built for all the appellate and original case traces we built.

TABLE 5.1: Counts of case traces built from CHC causelists (2010-2019)

Side	Number of case traces	Average size of a case trace	Number of reduced traces	Average size of a reduced trace
Appellate	20,579	6.4179	20,579	1.7926
Original	95,968	8.9761	95,968	2.7375

$$\frac{\text{Avg appellate case trace length}}{\text{Avg appellate reduced trace length}} = \frac{6.4179}{1.7926} = 3.5802$$

$$\frac{\text{Avg original case trace length}}{\text{Avg original reduced trace length}} = \frac{8.9761}{2.7375} = 3.2789$$

For the original cases, we see that the average case trace length reduces by more than 3-fold when reduced traces are considered. For the appellate cases, this reduction in average trace length is almost 4-fold. This observation further substantiates that most of the case listings in a causelist are simply rolled over to a later date, instead of having an actual hearing which shows that the current way of making the causelists for the CHC is very wasteful.

The results above are summarized in table 5.1.

[2] `https://drive.google.com/drive/folders/1nbb7A_8q40xVmY7b073eURPieAMG-pCf?usp=share_link`

Adjourned motion
ADJOURNED MOTION
ADJOURNED MOTION
ADJOURNED MOTION (2)
ADJOURNED MOTION (ADJ.) SEC.9
ADJOURNED MOTION AT 10.30 A.M.
ADJOURNED MOTION (ENTRY TAX)

FIGURE 5.3: Example of similar case states disambiguated to the same case state

5.2 Disambiguation of case states

The case states obtained from the causelists which were used to build our case trace and reduced trace had many similar case states which were essentially the same state but with different spellings, slightly different type, etc. To give a good measure of comparison between case states, we had to disambiguate these case states.

Extraction of case states from traces We extracted all the case states from the case traces of Original cases of CHC (about 1200 case states). We observed there were a lot of similar case states extracted which essentially meant the same case state. An example of such similar case states is shown in figure 5.3.

Clustering the case states using K-Means Clustering We computed the edit distance between 2 case states (essentially 2 strings) using algorithm 3 to compute the similarity between all the 1236 unique case states extracted from the case traces as shown below:

$$sim(state1, state2) \leftarrow 1 - \frac{dist(state1,state2)}{max(length(state1),length(state2))}$$

Algorithm 3 Calculate edit distance between 2 case states

Require: 2 case states - $l1$, $l2$
1: $m \leftarrow$length of $l1$
2: $n \leftarrow$length of $l2$
3: Initialize a 2-d array $dist$ of size $(m+1, n+1)$ with all zeroes
4: **for** each i in $[0, m]$ **do**
5: **for** each j in $[0, n]$ **do**
6: **if** $i == 0$ **then**
7: $dist[i][j] \leftarrow j$
8: **else**
9: **if** $j == 0$ **then**
10: $dist[i][j] \leftarrow i$
11: **else**
12: **if** $l1[i-1] == l2[j-1]$ **then**
13: $dist[i][j] \leftarrow dist[i-1][j-1]$
14: **else**
15: $dist[i][j] = 1 + min(dist[i-1][j-1], dist[i-1][j], dist[i][j-1])$
16: **end if**
17: **end if**
18: **end if**
19: **end for**
20: **end for**
21: **return** $dist[m][n]$

We used this above notion of similarity to compute 1236x1236 size similarity matrix. This matrix was used cluster the case states using simple K-Means clustering. We used the Elbow Method to identify the number of optimal clusters as 25 as shown in figure 5.4 (the elbow of the graph is approximately at 25). We then used K-Means Clustering to generate 25 clusters[3]. The extract of the clusters formed is shown in figure 5.5. As the figure clearly shows, the clusters generated are not good because very different case states are get mapped to the same cluster.

Manually disambiguating the case states As the K-Means Clustering did not give good results, we decided to manually disambiguate the case states. We observed that there were case states that were similar but of slightly different types

[3]The clusters formed can be viewed here: `https://drive.google.com/file/d/110XtCt_rRuyv3I4o9U6cAaVG-ybHDWaN/view?usp=share_link`

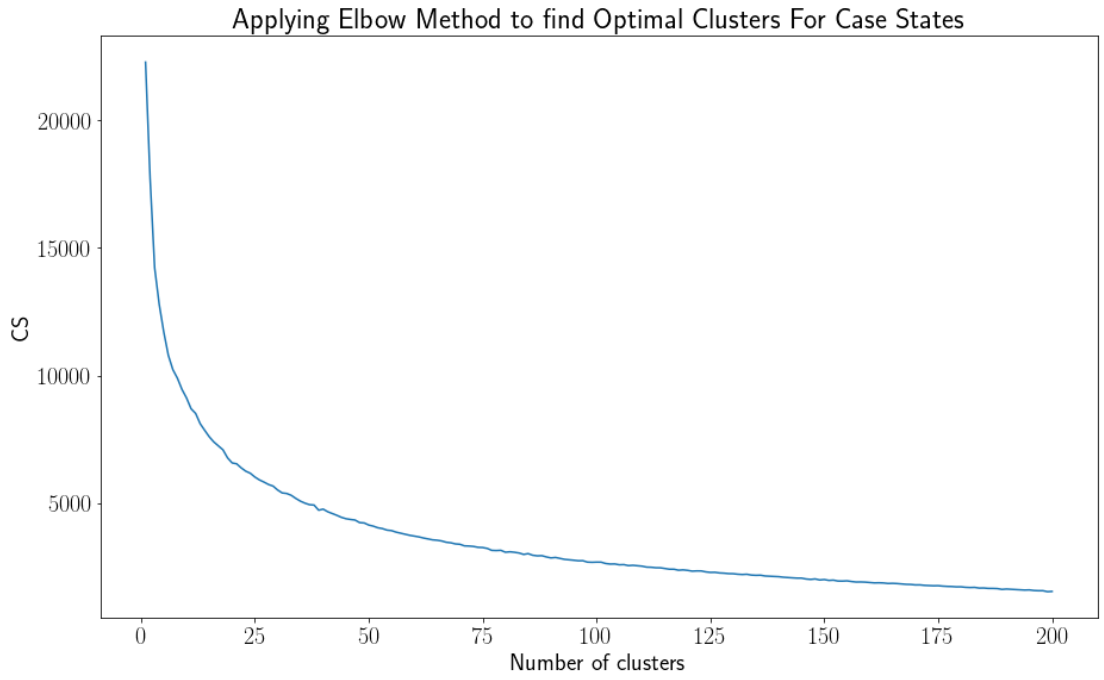


FIGURE 5.4: Using elbow method to find the number of Optimal Clusters as 25 for Clustering Case States

Cluster 0

Adjourned Motion
Application
Election Petition
New Application
New Chamber Application
New Motion
Restoration Application
Review Application

Cluster 3

Company Law Board Appeal
Confirmation of Sale
Company Matter Adjourned
Examination of Judgement Debtors
Non Contentious Business
Receivers Account
Testamentary Matters
Undefended Suits

FIGURE 5.5: An extract of the clusters formed

(like ADJOURNED MOTION (UNOPPOSED), ADJOURNED MOTION (WRIT), ADJOURNED MOTION (WRIT MATTERS), ADJOURNED MOTOIN (ENTRY TAX MATTERS), etc). So we decided to disambiguate at two different abstraction levels: **Standard Abstraction Level (SAS)** and **Detailed Abstraction Level (DAS)**. An example of how these 2 abstraction levels differ is shown in table 5.2.

TABLE 5.2: An example of the 2 abstraction levels)

Case State (extracted from causelist)	Case State (SAS)	Case State (DAS)
Adjourned Motions	Adjourned Motion	Adjourned Motion
Adjourned Motion (Unopposed)	Adjourned Motion	Adjourned Motion - Unopposed
Adjourned Motion (Writ)	Adjourned Motion	Adjourned Motion - Writ Matter
Adjourned Motion (Writ Matters)	Adjourned Motion	Adjourned Motion - Writ Matter
Adjourned Motion (Entry Tax Matters)	Adjourned Motion	Adjourned Motion - Entry Tax Matters

Thus, we manually disambiguated the case states at SAS and DAS[4]. This mapping gave us 175 case states under SAS and 403 case states under DAS. An extract of the case states under SAS and DAS is shown in figure 5.6. We recomputed the case traces using this mapping. We also recomputed the reduced traces under SAS and DAS. A comparison of the trace lengths under these abstraction levels can be seen in table 5.3.

TABLE 5.3: Comparison of trace lengths under SAS and DAS

Side	Abstraction level	Number of case traces	Avg size of a case trace	Number of reduced traces	Avg size of a reduced trace	Ratio of size
Original	No Abstraction	95,968	8.9761	95,968	2.7375	3.2789
Original	SAS	95,968	8.9761	95,968	2.3773	3.7758
Original	DAS	95,968	8.9761	95,968	2.4993	3.5914

5.3 Comparison of Case History and Trace

Comparison of Lengths of Case histories and Case/Reduced traces of Original cases of CHC A case history as discussed previously is the list of

[4]The mapping and the case states under SAS and DAS can be viewed here: `https://docs.google.com/spreadsheets/d/1kxfFnNbf8Wq5X82UxFxNPldispr1R3iHDtRhVL3bwXQ/edit?usp=sharing`

Unique Case States - State Abstraction Level	Unique Case States - Detailed Abstraction Level
ADJOURNED MOTION	ADJOURNED MOTION
ADJOURNED APPLICATION	ADJOURNED MOTION - ELECTION PETITION
ADMIRALTY MATTER	ADJOURNED MOTION - ENTRY TAX MATTERS
ADMIRALTY SUIT	ADJOURNED MOTION - FROM WARNING
ANNUAL STATEMENT RETURN	ADJOURNED MOTION - GROUP 4
APPLICATION FOR RULE IN CONTEMPT	ADJOURNED MOTION - GROUP "A"
APPEALS FROM DECREE	ADJOURNED MOTION - GROUP "B"
APPEALS FROM ORDERS	ADJOURNED MOTION - GROUP "C"
APPEAL	ADJOURNED MOTION - GROUP 6
APPEALS UNDER COMMERCIAL APPELLATE DIVISION	ADJOURNED MOTION - GROUP 5, 6
APPEAL UNDER SECTION 37	ADJOURNED MOTION - GROUP 1
APPLICATION	ADJOURNED MOTION - GROUP 2
APPLICATION ADJOURNED	ADJOURNED MOTION - OLD MATTER
APPLICATION FOR EXECUTION	ADJOURNED MOTION - SUIT
APPLICATIONS UNDER COMMERCIAL DIVISION	ADJOURNED MOTION - UNDER COMMERCIAL DIVISION
ARBITRATION MOTION	ADJOURNED MOTION - UNOPPOSED
ARBITRATION NEW MOTION	ADJOURNED MOTION - WRIT MATTER
AS APPLICATION	ADMIRALTY MATTER
ASSIGNED MATTER	ADMIRALTY SUIT
BUSINESS	ADMIRALTY SUITS AND APPLICATIONS
CENTRAL EXCISE AND CUSTOMS APPEALS	ADMIRALTY SUITS - MOTION
CHAMBER APPLICATION ADJOURNED	ANNUAL STATEMENT RETURN
CHAMBER APPLICATION FOR FINAL DISPOSAL	APPLICATION FOR RULE IN CONTEMPT
CHAMBER FOR FINAL DISPOSAL	APPEAL FROM DECREE
CHAPTER X RULE 35 OF ORIGINAL SIDE RULES	APPEAL FROM ORDER
COMPANY APPEAL	APPEAL FROM ORDER - ARBITRATION
COMPANY APPLICATION ADJOURNED	APPEAL FROM ORDER - COMPANY
COMPANY LAW BOARD APPEAL	APPEAL
COMPANY MATTER	APPEALS FROM DECREE

FIGURE 5.6: An extract of case states under SAS and DAS

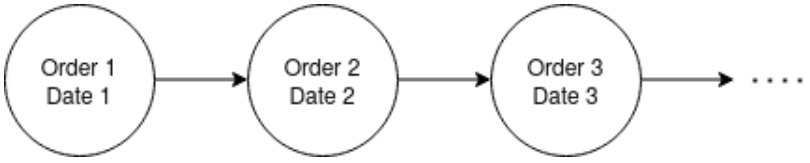
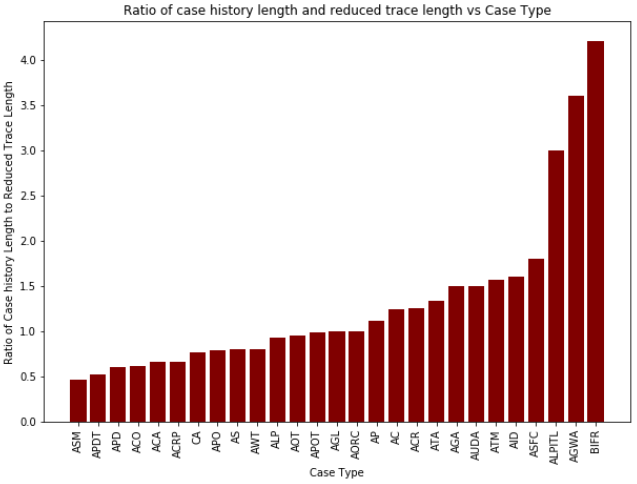


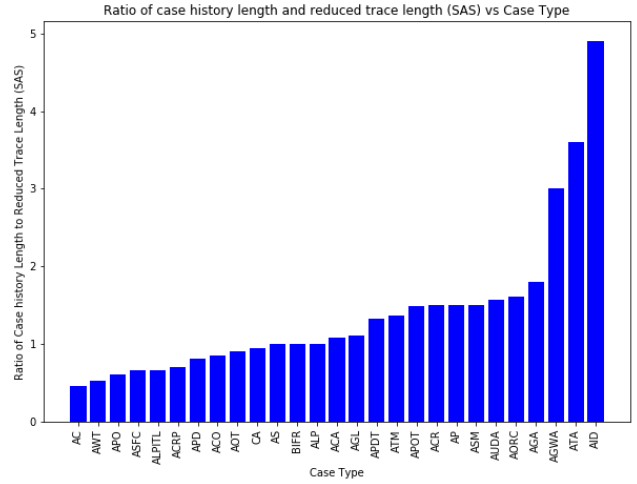
FIGURE 5.7: An example Case History

orders of a case, i.e it is list of actual hearings of a case as shown in figure 5.7. In comparison, a case trace is a list of the times the case was listed in the causelist. By our hypothesis, a node in a reduced trace should correspond to an order entry in the case history. We have compared the case history length with the lengths of the case traces and reduced traces for a few case types of Original cases from the CHC. An extract of these comaprisons can be seen in table 5.4[5]. We see that for most of the case types, the case history lengths are somewhat similar to the reduced trace lengths of No abstraction/SAS/DAS when compared to the case trace length. The figures 5.8(a), 5.8(b) and 5.8(c) shows the variation of case history length with

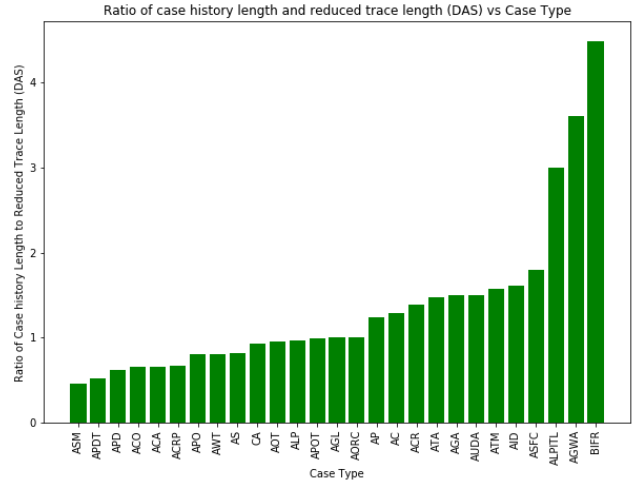
[5]The full table can be viewed at: `https://docs.google.com/spreadsheets/d/1aHQLVkvqZaq85GuLKI3jL910LFk8-4qKnVA1dxwOLSQ/edit?usp=sharing`



(a) No Abstraction



(b) SAS



(c) DAS

FIGURE 5.8: Variation of case history length with various case types

TABLE 5.4: Comparison of Case History length with Trace Lengths for various Case Types

Case Type	Avg case trace length	Avg reduced trace length	Avg reduced trace length (SAS)	Avg reduced trace length (DAS)	Avg case history length
ATA	7.3151	2.1781	1.9315	1.9726	2.9041
AP	8.6786	2.2729	1.9216	2.0584	2.5534
APO	17.695	3.4694	3.3654	3.3997	2.7412
AS	9.0075	4.0902	3.8421	4.0301	3.2932
APOT	4.1856	1.8214	1.647	1.8113	1.7925
CA	6.6871	2.3512	1.9100	1.9547	1.8045
AORC	2.7143	1.2857	1.2857	1.2857	1.2857
ALP	7.2759	2.2759	2.0966	2.1793	2.1103
AWT	13.4286	3.2381	2.9048	3.2381	2.6190
ACR	11.6216	3.7027	3.1351	3.3514	4.6486
AOT	12	3.3077	2.8462	3.3077	3.1538
AID	13.8551	3.1159	3.1159	3.1159	5
AC	8.6071	2.8214	2.5714	2.7143	3.5

various case types for Original Cases in the CHC dataset. For most of the case types, the ratio of case history length to various reduced trace length lies between 0.7 to 1.3 which shows that they are comparable.

Chapter 6

Similarity of Cases

6.1 Computing Similarity between Case Traces

Similarity between cases is a very important notion which is needed to find similar cases and find patterns between similar cases and the time they take from institution to disposal. This would help us in eventually developing a court scheduling algorithm. We also wanted to see the effect of summarization on similarity computation. A case trace corresponds to a document and the reduced trace corresponds to the summarization of the case trace (document). To show this, we needed a to define a notion of similarity between traces. We have used **Edit distance** based similarity as an indicator for this.

Edit Distance Edit distance is generally used as metric to find similarity of 2 strings or even 2 graphs. We have used this metric as an indicator of similarity between any two case traces or any two reduced traces. For our purpose, we are using the **Levenshtein distance** as edit distance, i.e, we are considering the **Insert**, **Remove** and **Replace** operations.

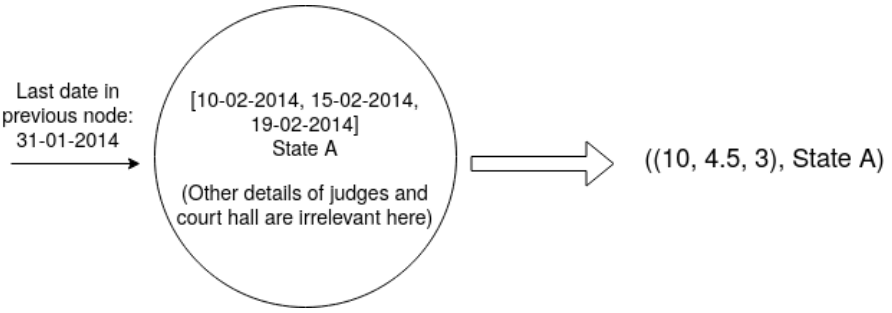


FIGURE 6.1: Example pre-processing of a node of a trace

To calculate the edit distance, we need to pre-process the traces first. We have considered only the dates and the case state of each node in the trace for edit distance computation for now. We convert each node of a trace into a tuple ((no. of days after last date in previous node, avg no. of days between 2 consecutive dates in this node, no. of dates in this node), state). This pre-processing is explained with an example in figure 6.1.

TABLE 6.1: Costs of operations in calculating Edit Distance using DP

Operation	Cost
Replacing case type	5
Replace the no. of days after last date in previous node	2
Replace the avg no. of days between 2 consecutive dates in the node	2
Replace the no. of dates in the node	1
Replace the state	5
Insert/Removing a node	10

The following notation is used:

$A \leftarrow$ No. of days after last date in previous node

$B \leftarrow$ Avg. no. of days between 2 consecutive nodes in the node

$C \leftarrow$ No. of dates in the node

After this pre-processing of every node, we obtain a list of the tuples obtained from each node. We use these lists to compute the edit

Algorithm 4 Calculate edit distance between 2 pre-processed traces

Require: 2 pre-processed traces - $l1$, $l2$ and their case types - $caseType1$, $caseType2$
3 dictionaries: Max_A, Max_B, Max_C mapping case types to the maximum of A, B and C values wrt case type

$caseTypeCost \leftarrow 0$
if $caseType1 == caseType2$ **then**
  $caseTypeCost \leftarrow 5$
end if
$m \leftarrow$length of $l1$
$n \leftarrow$length of $l2$
Initialize a 2-d array $dist$ of size $(m+1, n+1)$ with all zeroes
for each i in $[0, m]$ **do**
  for each j in $[0, n]$ **do**
    if $i == 0$ **then**
      $dist[i][j] \leftarrow 10 \times j$
    else
      if $j == 0$ **then**
        $dist[i][j] \leftarrow 10 \times i$
      else
        if $l1[i-1] == l2[j-1]$ **then**
          $dist[i][j] \leftarrow dist[i-1][j-1]$
        else
          $dist[i][j] \leftarrow 10 + min(dist[i-1][j], dist[i][j-1])$
          $replaceCost \leftarrow 0$
          if $l1[i-1][0][0] \neq l2[j-1][0][0]$ **then**
            $replaceCost \leftarrow replaceCost + 2 \times \frac{abs(l1[0][0]-l2[0][0])}{max(Max_A[caseType1], Max_A[caseType2])}$
          end if
          if $l1[i-1][0][1] \neq l2[j-1][0][1]$ **then**
            $replaceCost \leftarrow replaceCost + 2 \times \frac{abs(l1[0][1]-l2[0][1])}{max(Max_B[caseType1], Max_B[caseType2])}$
          end if
          if $l1[i-1][0][2] \neq l2[j-1][0][2]$ **then**
            $replaceCost \leftarrow replaceCost + 1 \times \frac{abs(l1[0][2]-l2[0][2])}{max(Max_C[caseType1], Max_C[caseType2])}$
          end if
          if $l1[i-1][1] \neq l2[j-1][1]$ **then**
            $replaceCost \leftarrow replaceCost + 5$
          end if
          $dist[i][j] = min(dist[i][j], replaceCost + dist[i-1][j-1])$
        end if
      end if
    end if
  end for
end for
return $dist[m][n] + caseTypeCost$

distances. Dynamic programming is used to calculate the edit distances. We need to define costs of each of the replacement operations for this. We assign a cost of 5 if the 2 case types are not the same to $caseTypeCost$. This $caseTypeCost$ is added to the final edit distance calculated between the 2 traces. We have defined costs for each operation as shown in table 6.1. These costs are chosen based on some domain knowledge. We wanted to give a higher cost to nodes that differ by their states when compared to nodes that differ by their dates. The number of times a case loops in a particular state is given even lesser weight because this mostly means that the case is being rolled over and not actually being heard.

The replacement costs of replacing A, B, or C in a node is normalized by multiplying with a normalization factor. Let $n1$ be a node of a trace $t1$ with case type $caseType1$ and $n2$ be a node of trace $t2$ with case type $caseType2$. Let Max_A, Max_B, Max_C be a dictionary mapping each case type to the maximum value of A, B and C for that case type respectively. Then the normalization factor for A, B and C are defined as:

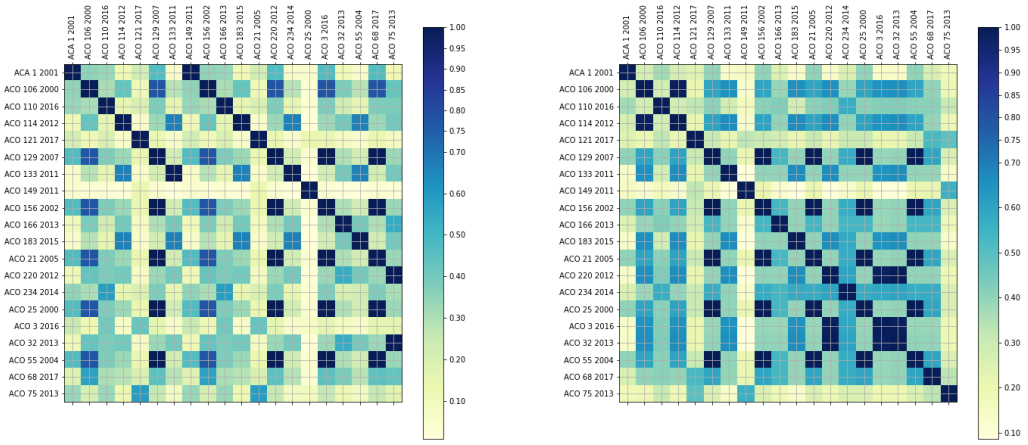
$$nf_A(n1, n2) \leftarrow \frac{|n1_A - n2_A|}{max(Max_A[caseType1], Max_A[caseType2])}$$
$$nf_B(n1, n2) \leftarrow \frac{|n1_B - n2_B|}{max(Max_B[caseType1], Max_B[caseType2])}$$
$$nf_C(n1, n2) \leftarrow \frac{|n1_C - n2_C|}{max(Max_C[caseType1], Max_C[caseType2])}$$

The algorithm to calculate edit distance based on dynamic programming is shown in algorithm 4. This algorithm will work both for case traces and reduced traces. The recurrence used in this algorithm is:

$$dist(i,j) \leftarrow min(insertCost+dist(i,j-1), removeCost+dist(i-1,j), replaceCost + dist(i-1,j-1))$$

Let $dist(t1,t2)$ denote the edit distances between 2 traces $t1$ and $t2$. The similarity $sim(t1,t2)$ is calculated as follows:

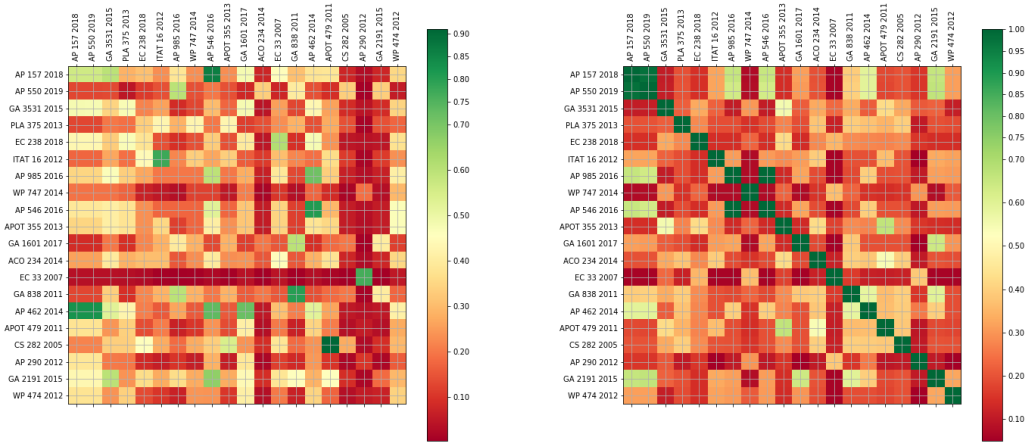
$$sim(t1,t2) \leftarrow 1 - \frac{dist(t1,t2)}{max(length(t1),length(t2))\times 10+caseTypeCost}$$



(a) Case Trace (b) Reduced Trace

FIGURE 6.2: Similarity Matrices for first 20 samples of 2000 Original case subset

Similarity Matrix We used the above formulation to calculate the similarity matrix for case traces and reduced traces of the original



(a) Case Trace (b) Reduced Trace

FIGURE 6.3: Similarity Matrices for a 20-sized random sample of the 2000 Original case subset

cases. We considered a **random sample of the Original cases of size 2000** for this experiment. The time taken to compute these matrices is shown in table 6.2. As is expected, a computation of reduced trace similarity takes lesser time than a computation of case trace similarity. The visualization of the similarity matrices for case traces[1] is shown in figures 6.2(a) and 6.3(a). The similar visualization of the similarity matrices for the reduced traces[2] is also shown in figures 6.2(b) and 6.3(b).

In the figure 6.2, we see that the similarity matrices obtained from the case traces and the reduced traces are quite comparable. The points in the case trace similarity matrix which are blue, i.e similarity = 1 are also blue in the reduced trace similarity matrix. A closer look of

[1]The complete similarity matrix for case traces can be viewed here:
[2]The complete similarity matrix for reduced traces can be viewed here:

the two shows that the reduced traces give lower similarity scores than case traces. This shows that reduced trace is much better in showing dissimilarity of cases.

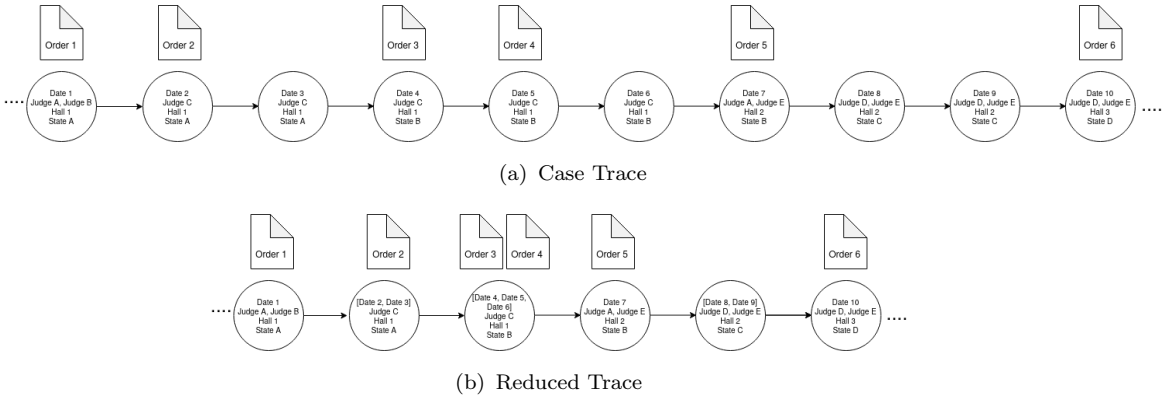
In figure 6.3, we see that the similarity matrices obtained from the case traces and the reduced traces are a little different. The points in the reduced trace similarity matrix which are near green, i.e similarity = 1 are almost red in the case trace similarity matrix. A lot of the points on case trace similarity matrix are yellowish while those are mostly red in the reduced trace similarity matrix. The reduced trace matrix colors most of the points as red which again substantiates our claim that reduced trace helps find dissimilarity among cases.

TABLE 6.2: Time taken for 2000x2000 Similarity Matrix Computation (without orders)

Type of Similarity	Total Time taken	Average Time Taken Per Similarity Computation
Case Trace Similarity	6 hr 32 min	5.8833 milliseconds
Reduced Trace Similarity	6 hr 2 min	5.431 milliseconds

6.2 Integrating Orders into computing similarity between Traces

The case trace encompasses only the information of the case present in the causelist. We can include the order information present into the case trace and reduced trace as well as shown in figures 6.4(a) and 6.4(b).



(a) Case Trace

(b) Reduced Trace

FIGURE 6.4: An example case trace with orders and corresponding reduced trace

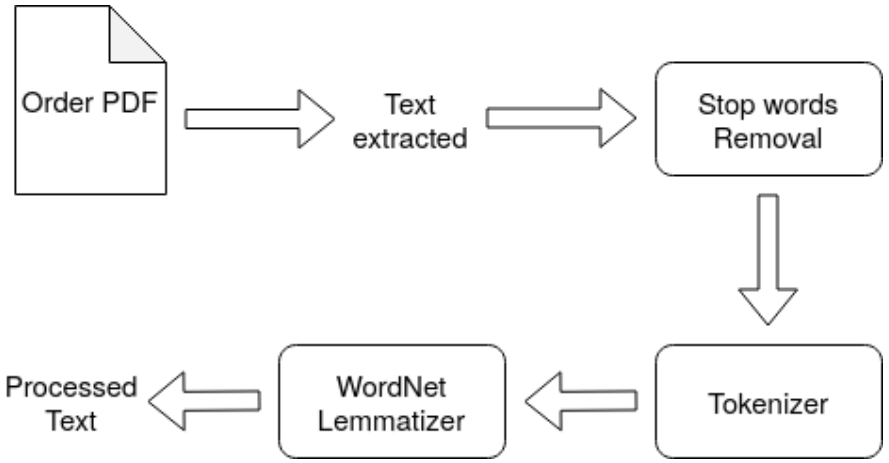


FIGURE 6.5: Pre-processing of orders

Preprocessing Orders We have the orders available to us in the form of PDF documents. We processed these PDFs and obtained the text from them. This text was then tokenized and lemmatized using WordNet and all stop words were removed. We have then converted the text into TF-IDF vectors and used cosine similarity between the TF-IDF vectors of 2 orders to compare them.

Preprocessing Traces Just as before, we need to pre-process the traces first. We convert each node of a trace into a tuple ((no. of days after

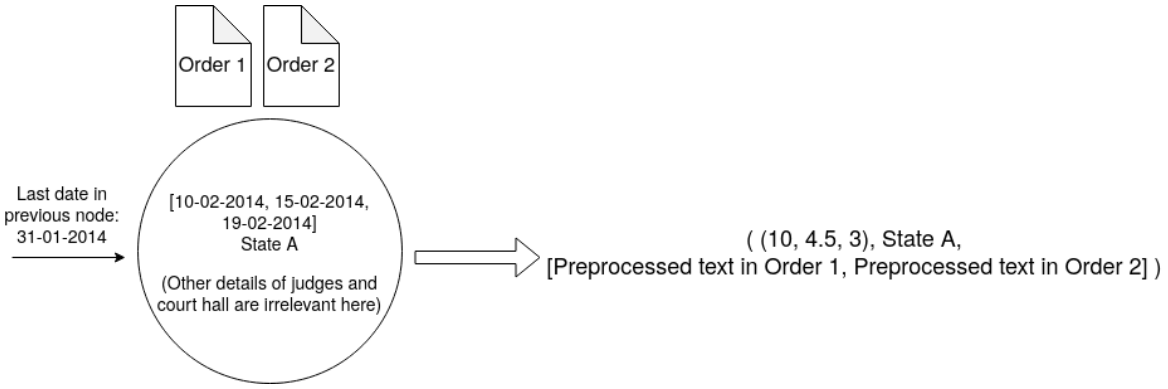


FIGURE 6.6: Example pre-processing of a node of a trace with orders

last date in previous node, avg no. of days between 2 consecutive dates in this node, no. of dates in this node), state, list of processed text of orders). An example pre-processing is shown in figure 6.6.

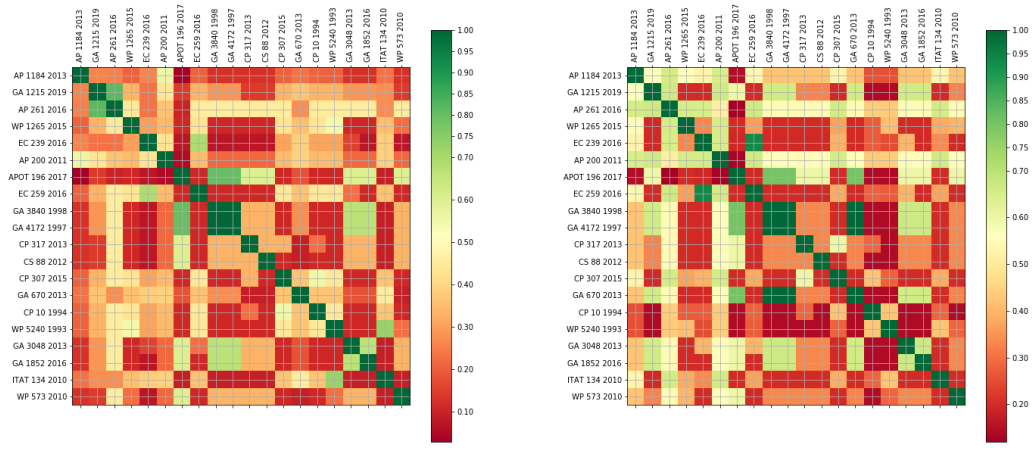
Edit Distance with Orders The edit distance is calculated with 2 preprocessed traces. The costs for each operation is kept the same as before, and an additional cost of 10 is assigned to replacing the cosine similarity of Orders. We have given highest cost to this as a lot of important information is present in orders. This is shown in table 6.3. Let $avgCosSim$ be the average cosine similarity between each pair of orders: one from node $n1$ of trace $t1$ and another from node $n2$ of trace $t2$ The replacement cost for this is defined as follows:

$$replaceCost_{cosSim}(n1, n2) \leftarrow 10 \times (1 - avgCosSim)$$

TABLE 6.3: Costs of operations in calculating Edit Distance With Orders

Operation	Cost
Replacing case type	5
Replace the no. of days after last date in previous node	2
Replace the avg no. of days between 2 consecutive dates in the node	2
Replace the no. of dates in the node	1
Replace the state	5
Replace the cosine similarity of orders	10
Insert/Removing a node	20

The replacement/insert/remove cost of a node is 20 if the node has orders associated with it, otherwise, it is 10. The algorithm for computing this edit distance with orders is shown in algorithms 5 and 6.



(a) Case Trace (b) Reduced Trace

FIGURE 6.7: Similarity Matrices (with orders) for random 20 samples of 5000 Original cases subset

Similarity Matrix We have constructed the similarity matrix for a **random sample of the Original cases of size 5000**. The time

Algorithm 5 Calculate edit distance between 2 pre-processed traces with orders

Require: 2 pre-processed traces - $l1$, $l2$ and their case types - $caseType1$, $caseType2$
3 dictionaries: Max_A, Max_B, Max_C mapping case types to the maximum of A, B and C values wrt case type

$caseTypeCost \leftarrow 0$
if $caseType1 == caseType2$ **then**
  $caseTypeCost \leftarrow 5$
end if
$m \leftarrow$length of $l1$
$n \leftarrow$length of $l2$
Initialize a 1-d array $cummulativeMaxCost1$ of size m with all zeroes
for each i in $[0, m-1]$ **do**
  if $len(l1[i][2]) == 0$ **then**
    $cummulativeMaxCost1[i] = 10$
  else
    $cummulativeMaxCost1[i] = 20$
  end if
  if $i \neq 0$ **then**
    $cummulativeMaxCost1[i] += cummulativeMaxCost1[i-1]$
  end if
end for
Initialize a 1-d array $cummulativeMaxCost2$ of size n with all zeroes
for each j in $[0, n-1]$ **do**
  if $len(l2[j][2]) == 0$ **then**
    $cummulativeMaxCost2[i] = 10$
  else
    $cummulativeMaxCost2[i] = 20$
  end if
  if $j \neq 0$ **then**
    $cummulativeMaxCost2[j] += cummulativeMaxCost2[j-1]$
  end if
end for
$dist \leftarrow$ EditDistanceWithOrders($l1, l2, caseType1, caseType2$, $cummulativeMaxCost1, cummulativeMaxCost2, Max_A, Max_B, Max_C$)
return $dist[m][n] + caseTypeCost$, $max(cummulativeMaxCost1[m-1], cummulativeMaxCost[n-1])$

Algorithm 6 EditDistanceWithOrders

Require: $l1$, $l2$, $caseType1$, $caseType2$, $cummulativeMaxCost1$, $cummulativeMaxCost2$, Max_A, Max_B, Max_C

$m \leftarrow$length of $l1$
$n \leftarrow$length of $l2$
Initialize a 2-d array $dist$ of size $(m+1, n+1)$ with all zeroes
for each i in $[0, m]$ **do**
  for each j in $[0, n]$ **do**
    if $i == 0$ **then**
      $dist[i][j] \leftarrow 10 \times j$
    else
      if $j == 0$ **then**
        $dist[i][j] \leftarrow 10 \times i$
      else
        if $l1[i-1] == l2[j-1]$ **then**
          $dist[i][j] \leftarrow dist[i-1][j-1]$
        else
          $replaceCost \leftarrow 0$
          if $len(l1[i-1][2]) == 0$ or $len(l2[j-1][2]) == 0$ **then**
            $dist[i][j] \leftarrow 10 + min(dist[i-1][j], dist[i][j-1])$
          else
            $dist[i][j] \leftarrow 20 + min(dist[i-1][j], dist[i][j-1])$
            $avgCosSim \leftarrow$ Average Cosine Similarity between any order of $l1[i-1]$ and $l2[j-1]$
            $replaceCost \leftarrow 10 \times (1 - avgCosSim)$
          end if
          if $l1[i-1][0][0] \neq l2[j-1][0][0]$ **then**
            $replaceCost \leftarrow replaceCost + 2 \times \frac{abs(l1[0][0] - l2[0][0])}{max(Max_A[caseType1], Max_A[caseType2])}$
          end if
          if $l1[i-1][0][1] \neq l2[j-1][0][1]$ **then**
            $replaceCost \leftarrow replaceCost + 2 \times \frac{abs(l1[0][1] - l2[0][1])}{max(Max_B[caseType1], Max_B[caseType2])}$
          end if
          if $l1[i-1][0][2] \neq l2[j-1][0][2]$ **then**
            $replaceCost \leftarrow replaceCost + 1 \times \frac{abs(l1[0][2] - l2[0][2])}{max(Max_C[caseType1], Max_C[caseType2])}$
          end if
          if $l1[i-1][1] \neq l2[j-1][1]$ **then**
            $replaceCost \leftarrow replaceCost + 5$
          end if
          $dist[i][j] = min(dist[i][j], replaceCost + dist[i-1][j-1])$
        end if
      end if
    end if
  end for
end for
return $dist[m][n]$

taken to compute these matrices is shown in table 6.4. As is expected, a computation of reduced trace similarity takes lesser time than a computation of case trace similarity. The visualization of the similarity matrix for case traces[3] is shown in figure 6.7(a). The similar visualization of the similarity matrix for the reduced traces[4] is also shown in figure 6.7(b).

TABLE 6.4: Time taken for 5000x5000 Similarity Matrix Computation (with orders)

Type of Similarity	Total Time taken	Average Time Taken Per Similarity Computation
Case Trace Similarity	34 hr 6 min	4.9112 milliseconds
Reduced Trace Similarity	29 hr 57 min	4.312 milliseconds

In the figure 6.7, we see that the similarity matrices obtained from the case traces and the reduced traces are quite different. The points in the case trace similarity matrix which are green, i.e similarity = 1 are also green in the reduced trace similarity matrix. Howvever, there are a lot of points which are greenish is the reduced trace matrix but are reddish in the case trace matrix. This shows that unlike the similarity without orders, reduced trace similarity with orders is much better in capturing similar cases.

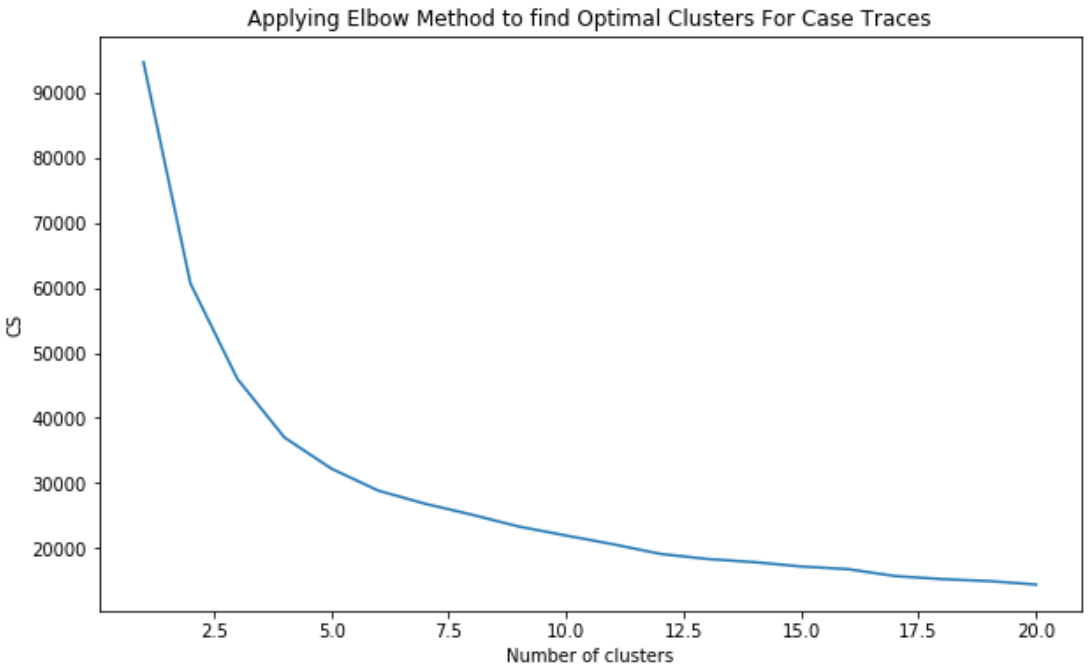


FIGURE 6.8: Using elbow method to find the number of Optimal Clusters as 4 for Clustering with Case Traces Similarity Without Orders

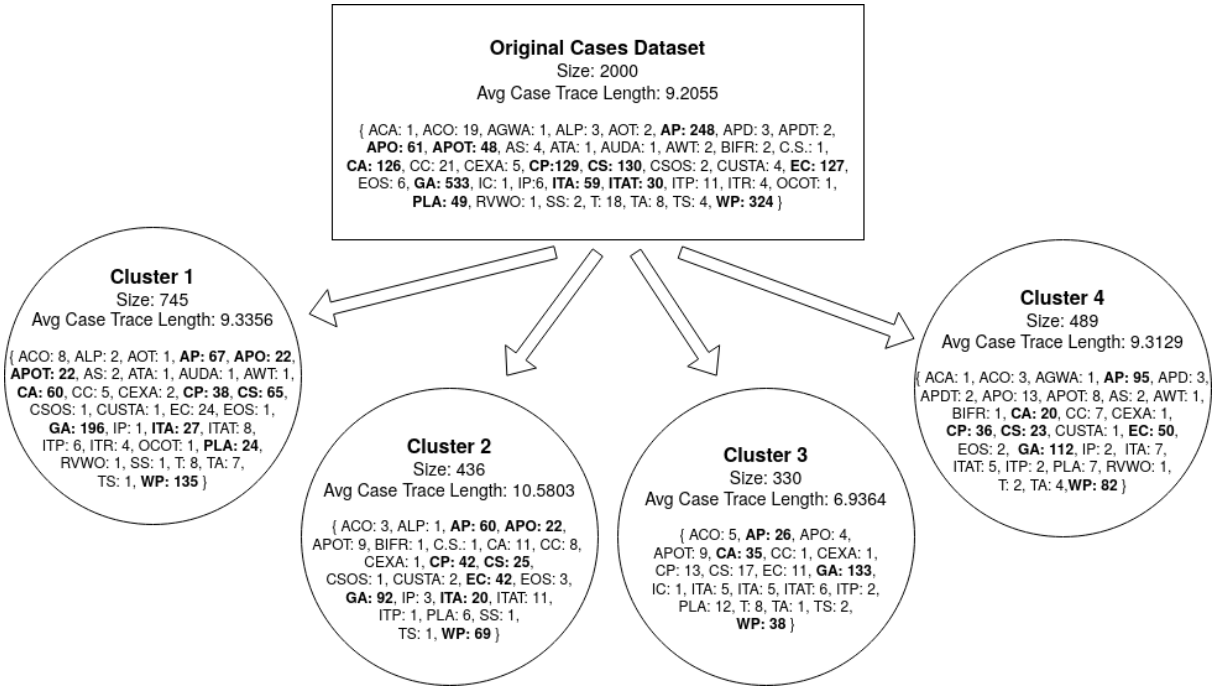


FIGURE 6.9: A summary of the clusters obtained by K Means Clustering with Case Trace Similarity Matrix without Orders

6.3 Clustering Cases based on Similarity of Traces

Clustering based on Case Trace Similarity Without Orders We used the 2000x2000 similarity matrix that we generated using edit distances between **case traces** to cluster the cases using simple K-Means Clustering. We used the **Elbow Method** to identify the number of optimal clusters as 4 as shown in figure 6.8 (the elbow of the graph is approximately at 4). We then used K-Means Clustering to generate 4 clusters[5]. The clusters generated are summarized in figure 6.9. The average case trace lengths of each cluster is also shown in the figure. We see that the clusters differ slightly in the average case trace length and do not separate out different case types very well, which means that cases of different case types are also similar.

Clustering based on Reduced Trace Similarity Without Orders We performed the similar analysis using the 2000x2000 similarity matrix that we generated using edit distances between **reduced traces** to cluster the cases using simple K-Means Clustering. The elbow method gives the number of optimal clusters as 5 this time as shown in figure 6.10 (the elbow of the graph is approximately at 5). We saw that the similarity matrix of reduced traces is more helpful is finding dissimilarity

[3]The complete similarity matrix for case traces can be viewed here: `https://drive.google.com/file/d/1GKlklJkW_DORVDubyt_-k4mp9nfLgvzm/view?usp=sharing`

[4]The complete similarity matrix for reduced traces can be viewed here: `https://drive.google.com/file/d/1myfVVQbTfh2EVX1z74C23ZcTFKoziUIk/view?usp=sharing`

[5]The cluster assignments can be viewed here: `https://drive.google.com/file/d/1cdkms0nlZubt3YDQKM6kTxkoEQLNg1J_/view?usp=sharing`

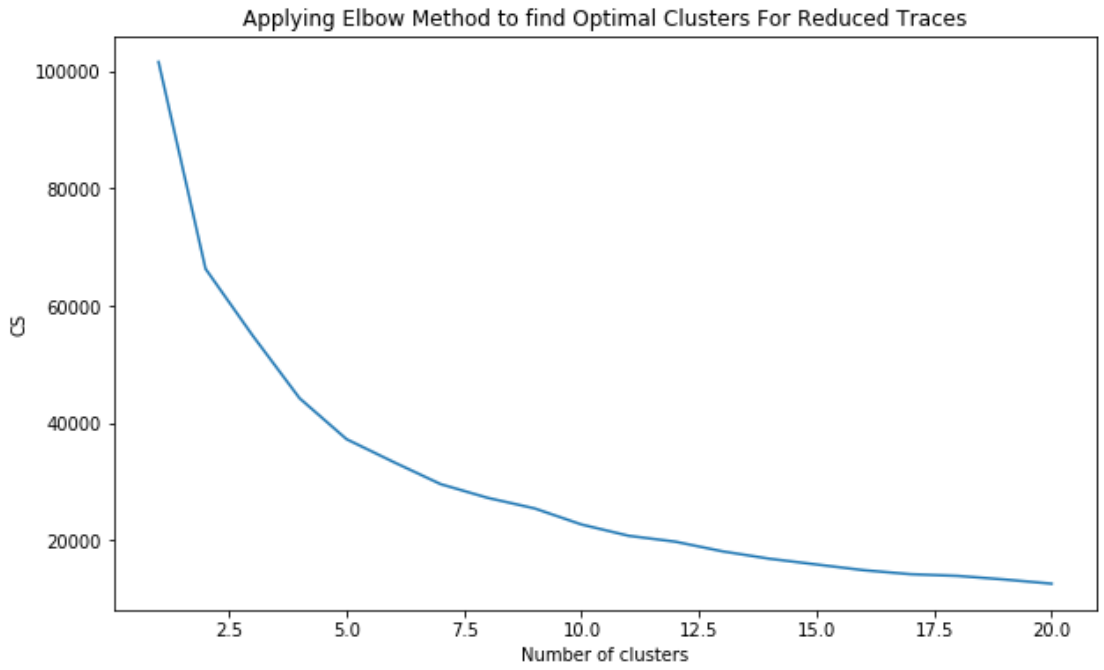


FIGURE 6.10: Using elbow method to find the number of Optimal Clusters as 5 for Clustering with Reduced Trace Similarity Without Orders

between cases and hence, the number of optimal clusters for this case is larger.

K-Means Clustering is used again to generate 5 clusters[6] using the reduced trace similarity matrix. The clusters generated are summarized in figure 4.1(b). The average reduced trace lengths of each cluster is also shown in the figure. We see that the clusters differ much lesser in the average reduced trace length when compared to the clusters formed using case traces. This indicates that the importance of trace length as one of the underlying factors influencing the formation of the clusters is much lesser.

[6]The cluster assignments can be viewed here: `https://drive.google.com/file/d/1p9OWYViGLL6p2Dyv_SRy-gsRBrxzw0tC/view?usp=share_link`

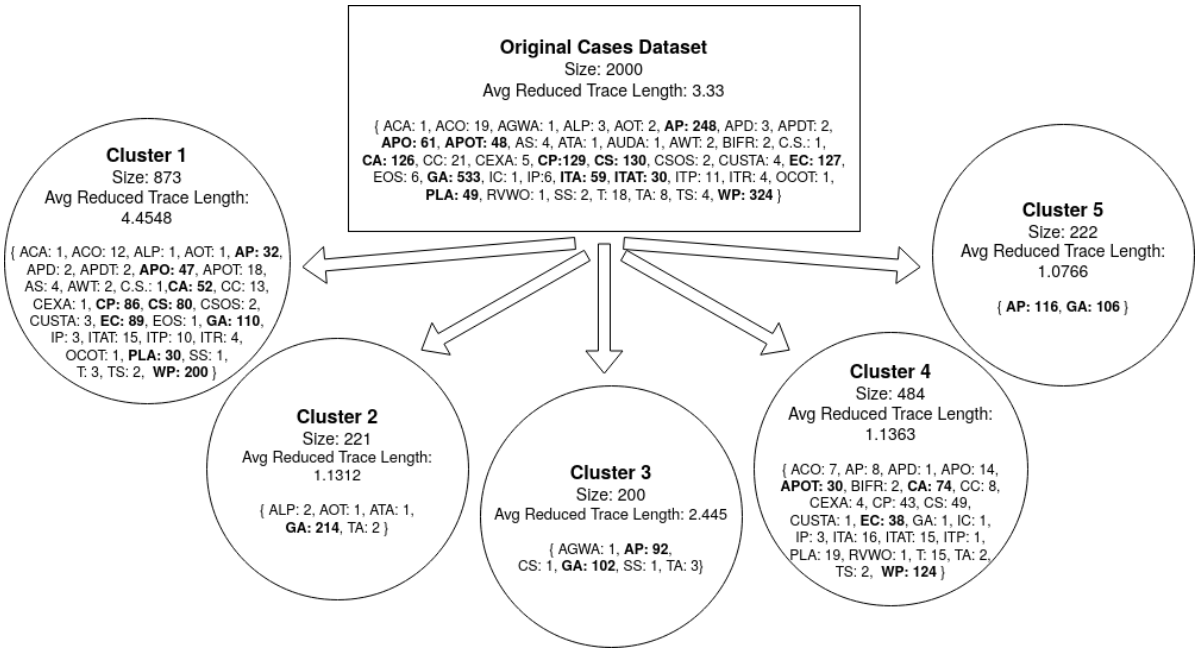


FIGURE 6.11: A summary of the clusters obtained by K Means Clustering with Reduced Trace Similarity Matrix without orders

The clusters obtained from both the case trace similarity matrix does not separate out different case types but the clusters obtained reduced trace similarity matrix achieves this to some degree as Cluster 2, 3 and 5 are majorly dominated by only certain case types. This shows that reduced trace is better at capturing similarity between cases.

Clustering based on Case Trace Similarity With Orders We used the top leftmost 2000x2000 matrix from the 5000x5000 case trace similarity matrix that we generated using edit distances between **case traces** to cluster the (2000) cases using K-Means Clustering. We used the **Elbow Method** to identify the number of optimal clusters as 5 as shown in figure 6.12 (the elbow of the graph is approximately at 5). We then

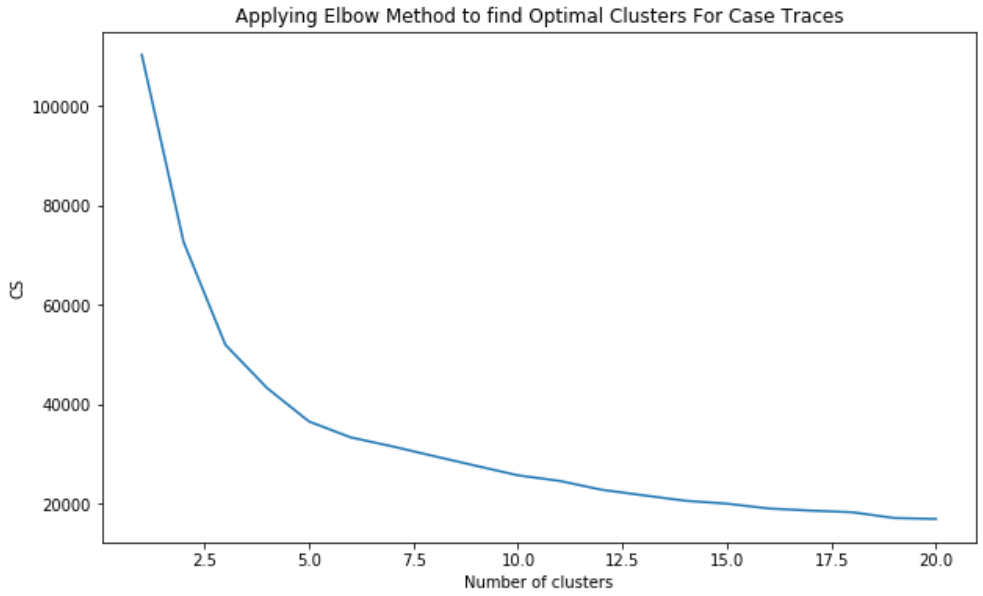


FIGURE 6.12: Using elbow method to find the number of Optimal Clusters as 5 for Clustering with Case Traces Similarity With Orders

used K-Means Clustering to generate 5 clusters[7]. The clusters generated are summarized in figure 6.13. The average case trace lengths of each cluster is also shown in the figure. We see that the clusters differ highly in the average case trace length which indicates that it is one of the underlying factors influencing the formation of these clusters.

Clustering based on Reduced Trace Similarity With Orders We performed the similar analysis using the top leftmost 2000x2000 matrix from the 5000x5000 reduced trace similarity matrix that we generated using edit distances between **reduced traces** to cluster the (2000) cases using K-Means Clustering. The elbow method gives the number of optimal clusters as 5 this time as shown in figure 6.14 (the elbow of

[7]The cluster assignments can be viewed here: `https://drive.google.com/file/d/1AUPFMU_zdRuWgd5UwYyF_fPlkBuIja1E/view?usp=sharing`

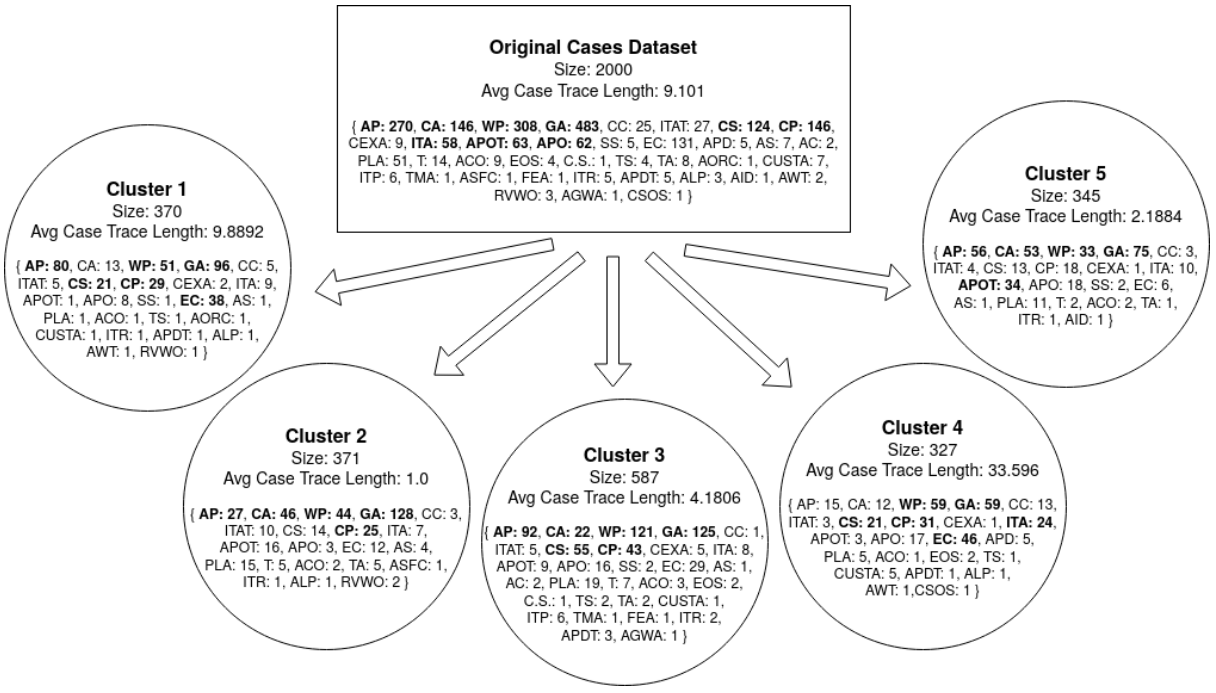


FIGURE 6.13: A summary of the clusters obtained by K Means Clustering with Case Trace Similarity Matrix with Orders

the graph is approximately at 5). We saw that the similarity matrix of reduced traces is more helpful is finding dissimilarity between cases and hence, the number of optimal clusters for this case is larger.

K-Means Clustering is used again to generate 5 clusters[8] using the reduced trace similarity matrix. The clusters generated are summarized in figure 6.15. The average reduced trace lengths of each cluster is also shown in the figure. We see that the clusters differ much lesser in the average reduced trace length when compared to the clusters formed using case traces. This indicates that the importance of trace length as one of the underlying factors influencing the formation of the clusters is much lesser.

[8]The cluster assignments can be viewed here: `https://drive.google.com/file/d/1kEUnfkqjiygpxVd-vDMMeQRpjxOGNPjW/view?usp=sharing`

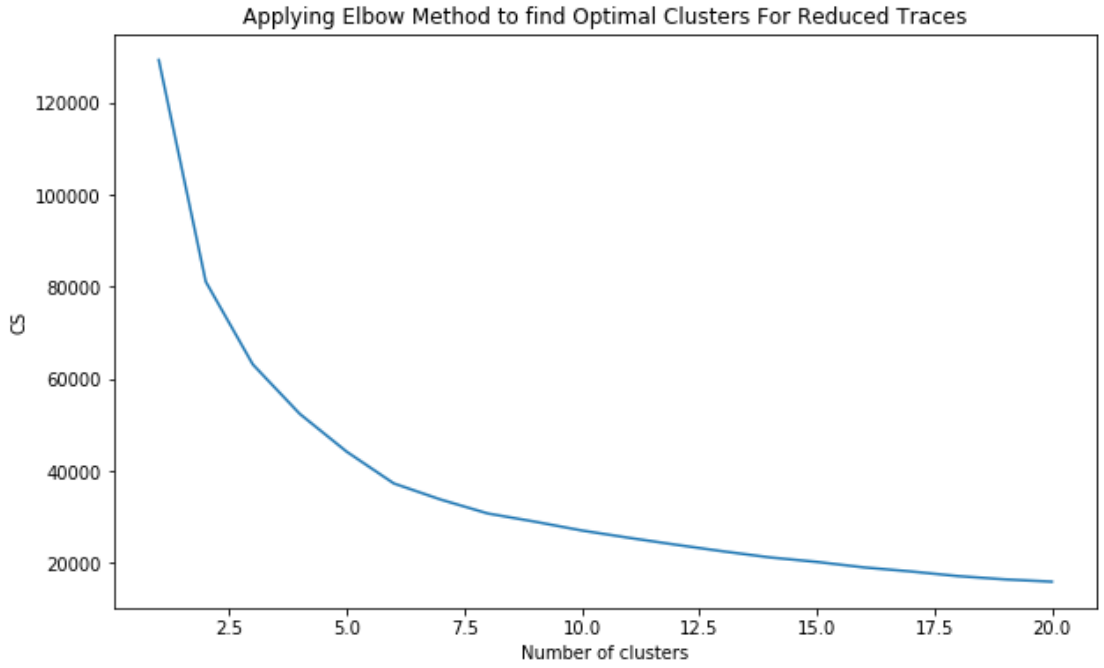


FIGURE 6.14: Using elbow method to find the number of Optimal Clusters as 5 for Clustering with Reduced Trace Similarity With Orders

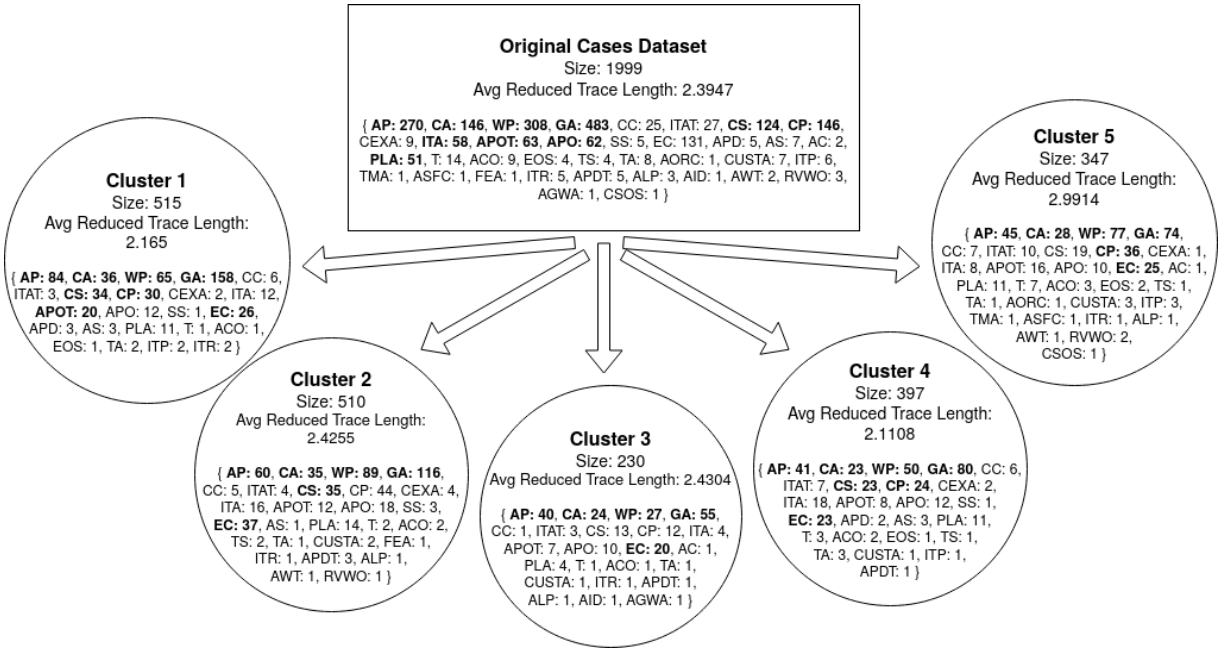


FIGURE 6.15: A summary of the clusters obtained by K Means Clustering with Reduced Trace Similarity Matrix with orders

The clusters obtained from both the case trace similarity matrix separates out different case types to a certain degree but the clusters obtained reduced trace similarity matrix does not achieve this. This shows that case trace is better at capturing similarity with orders between cases.

Chapter 7

Top-K Analysis Metric

7.1 Formulation

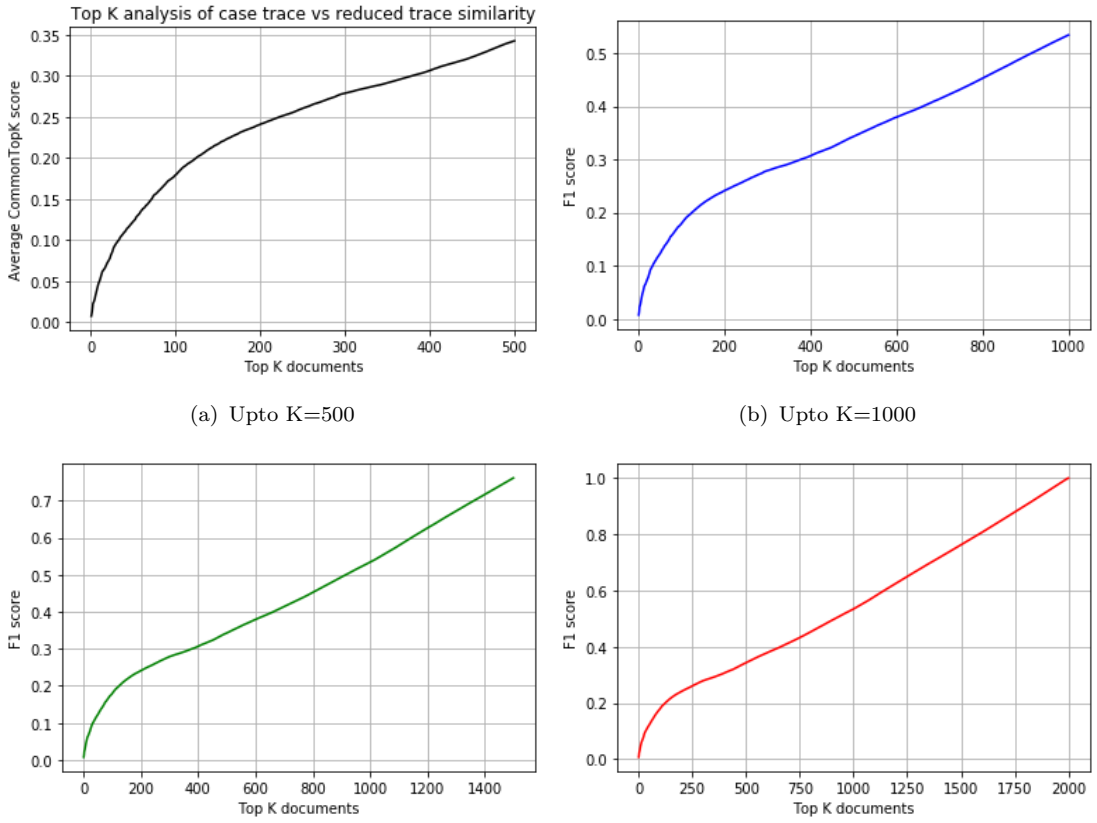
We want to see the effect of summarization on similarity computation. A case trace corresponds to a document and the reduced trace corresponds to the summarization of the case trace (document) as mentioned before. A widely used application of similarity between cases is top-K cases which are similar to a query case. This metric aims to compare the number of common cases between the top-K cases found using case trace similarity and the top-K documents found using reduced trace similarity.

Let $K1$ be list of top-K similar cases using case trace similarity for a query case and $K2$ be the top-K similar cases using reduced trace similarity for the same query case. Then, the metric $CommonTopK$ is defined as follows:

$$CommonTopK \leftarrow \frac{\text{Number of common cases in } L1 \text{ and } L2}{K}$$

This metric can be a measure of how good out similarity metric is.

7.2 Experiments



(a) Upto K=500

(b) Upto K=1000

(c) Upto K=1500

(d) Upto K=2000

FIGURE 7.1: Variation of *CommonTopK* with K for similarity without orders

Top-K Analysis using Similarity Without Orders We have built 2000x2000 similarity matrices, one using case trace similarity and the other using reduced trace similarity as mentioned in chapter 6. We use these matrices to compute the top-K similar cases using both the similarities

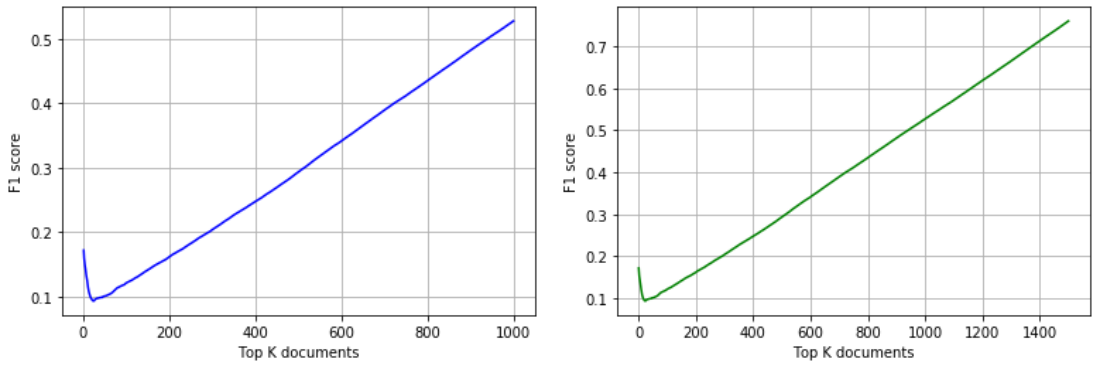
for various values of K. Figure 7.1 and table 7.1 shows how the metric $CommonTopK$ varies with K. We see that, the $CommonTopK$ increases faster at lower values of K and starts varying linearly at higher values of K in figure 7.1. Table 7.1 shows that $CommonTopK$ reaches a high value (greater than 0.7) only for high values of K, which means that there is scope of improvement in our similarity measure without orders.

TABLE 7.1: Variation of $CommomTopK$ with K for similarity without orders

K	$CommonTopK$	Average number of common cases
10	0.0487	0.49
20	0.0726	1.45
50	0.1222	6.11
100	0.1792	17.92
200	0.2411	48.22
500	0.3428	171.4
1000	0.534	534.02
1500	0.7618	1142.64
2000	1	2000

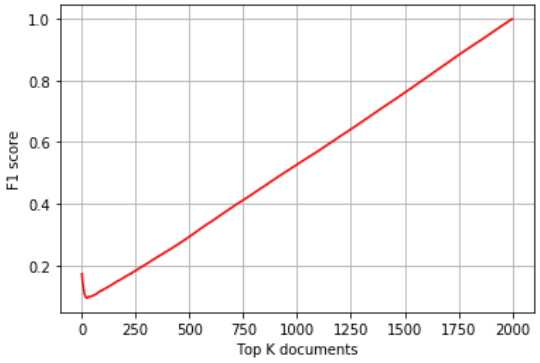
TABLE 7.2: Variation of $CommomTopK$ with K for similarity with orders

K	$CommonTopK$	Average number of common cases
10	0.1198	1.2
20	0.0958	1.92
50	0.1011	5.05
100	0.1217	12.17
200	0.1619	32.38
500	0.2934	146.7
1000	0.5273	527.3
1500	0.7605	1140.75
2000	1	2000



(a) Upto K=1000

(b) Upto K=1500



(c) Upto K=2000

FIGURE 7.2: Variation of *CommonTopK* with K for similarity with orders

Top-K Analysis using Similarity With Orders We have built 5000x5000 similarity matrices, one using case trace similarity and the other using reduced trace similarity as mentioned in chapter 6. We have extracted the topmost 2000x2000 matrices from each of these matrices for our experiment. We use these matrices to compute the top-K similar cases using both the similarities for various values of K. Figure 7.2 and table 7.2 shows how the metric *CommonTopK* varies with K. We see that after an initial kink, the *CommonTopK* varies linearly with K in figure 7.2.

On comparing tables 7.1 and 7.2, we see that *CommonTopK* is higher

for similarity with orders, than for similarity without orders, upto $K = 20$. After $K = 20$, $CommonTopK$ is consistently higher for similarity without orders (when compared to similarity with orders). This shows that integrating similarity into our similarity metric has worsened the $CommonTopK$ score. This is probably due to the fact that the costs for various operations are not fine-tuned in both the similarity metrics.

Table 7.2 shows that $CommonTopK$ reaches a high value (greater than 0.7) only for high values of K. These observations means that there is scope of improvement in our similarity measure with orders as this metric should outperform the similarity without orders.

Chapter 8

Ablation Study of the effect of presence of Senior Advocates on Pendency of Cases

8.1 Pendency of a Case

Pendency of court cases[1] in India is the delay in the disposal of cases (lawsuits) to provide justice to the aggrieved person or organisation by judicial courts at all levels. The judiciary in India works in hierarchy at three levels - federal or supreme court, state or high courts, and district courts. The court cases is categorised into two types - civil and criminal. In 2022, the total number of pending cases of all types and at all levels rose to 50 million or 5 crores, including over 169,000 court cases pending for more than 30 years in district and high courts.

[1] https://en.wikipedia.org/wiki/Pendency_of_court_cases_in_India

4.3 crore out of 5 crore cases, i.e more than 85% cases, are pending in district courts as of December 2022. Government itself is the biggest litigant, having 50% of the pending cases being sponsored by the state.

India has the largest number of pending court cases in the world. Many judges and government officials have said that the pendency of cases is the biggest challenge before Indian judiciary. According to a 2018 Niti Aayog strategy paper, at the then-prevailing rate of disposal of cases in our courts, it would take more than 324 years to clear the backlog. At that time in 2018, the pending cases stood at 29 million. With the cases taking time in courts, it leads to delays in the delivery of justice for both victim and accused. In April 2022, a court in Bihar state acquitted a man of murder for lack of evidence after he spent 28 years in jail. He was arrested at age 28, was freed at 58.

Pendency of cases cost 1.5%-2% of India's GDP.

8.2 Analysis of presence of Senior Advocates on Pendency of cases

Extracting senior advocates from Orders Extracting the senior advocates from Orders and which party they belong is a non trivial task that cannot be performed by simple regular expressions since orders follow no specific pattern. We have taken the presence of the words

'Senior Advocate' or **'Senior Adv.'** in the text of the order as the presence of (at least one) senior advocate in the hearing.

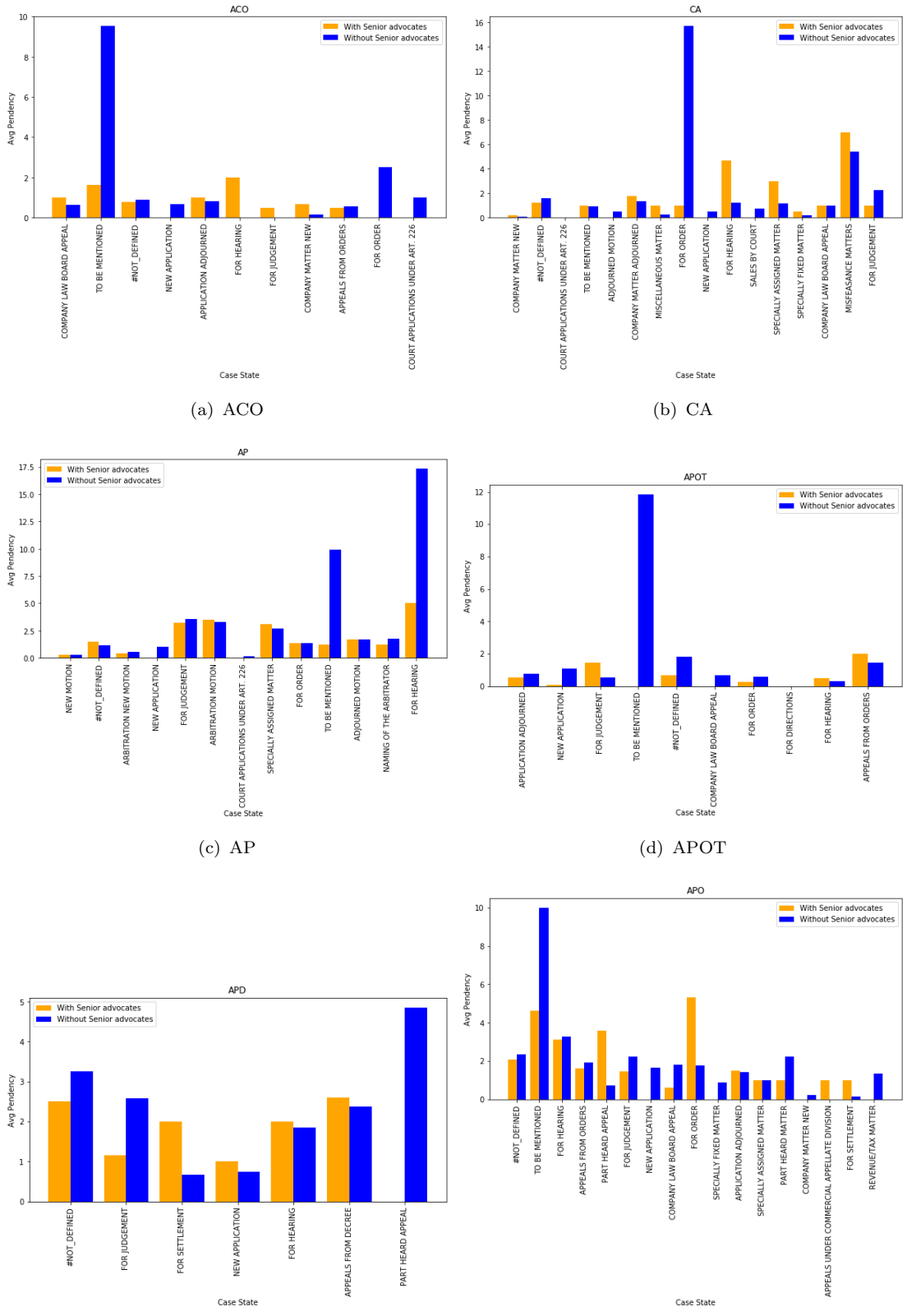
The following plots in figures 8.1 show the effect of presence of senior advocates in at least one of the parties on the pendency of cases for various case types.[2]. The $\#NOT\_DEFINED$ case state in the plots 8.1 and 8.2 corresponds to anomalous orders that do not occur in the causelist of the same date as the order.

We hypothesize that one party will most likely have only one senior advocate. Thus, orders where we detect more than one occurrence of the words 'Senior Advocate' or 'Senior Adv.' are likely to be cases where there is a senior advocate on each parties' side. The following plots in figures 8.2 show the effect of presence of senior advocates in both parties on the pendency of cases for various case types.

On comparing figures 8.1 and 8.2, we see that presence of advocates in both parties does bring down the pendency over all case types. This shows that the number of senior advocates in the hearing has a negative correlation with the pendency of the case (when number of senior advocates in the hearing increases, the pendency of the case goes down).

This experiment shows that the pendency of cases depends on a lot of factors like case type, presence of senior advocates, etc. The results

[2]The plots for other case types can be viewed here: `https://docs.google.com/document/d/1VOWoHm9QoAODN4a5MAatBxSRZBkphyFKvA1eD-eNRtE/edit?usp=sharing`

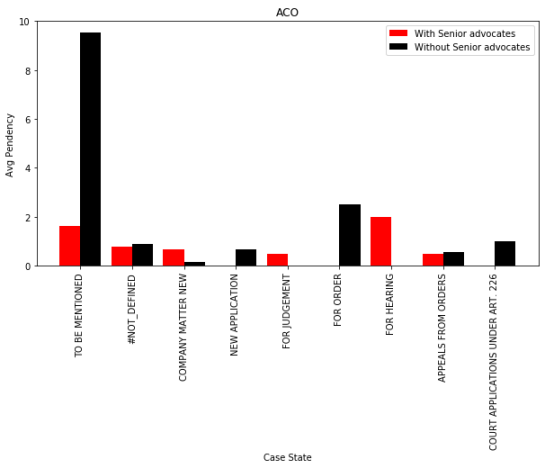


(a) ACO (b) CA

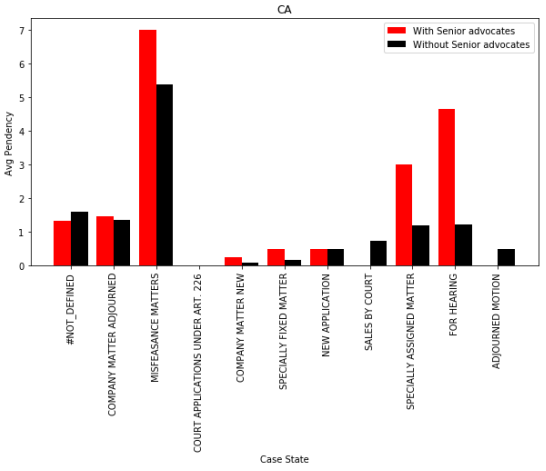
(c) AP (d) APOT

(e) APD (f) APO

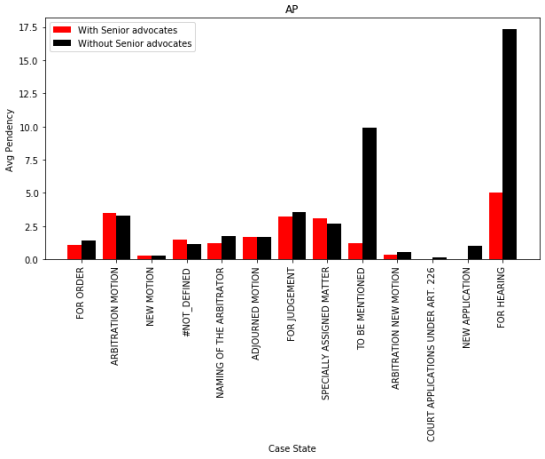
FIGURE 8.1: Effect of presence of Senior Advocate(s) in at least one of the parties on the Pendency of cases for various case types



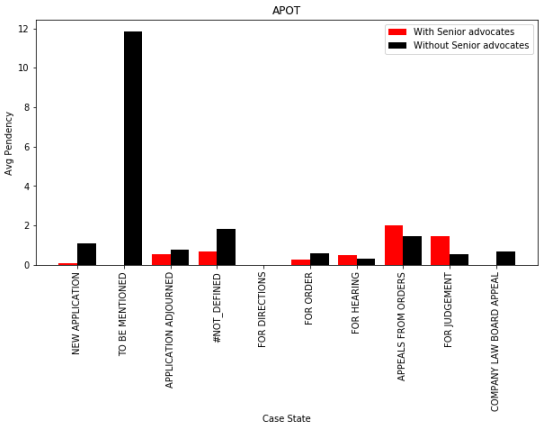
(a) ACO



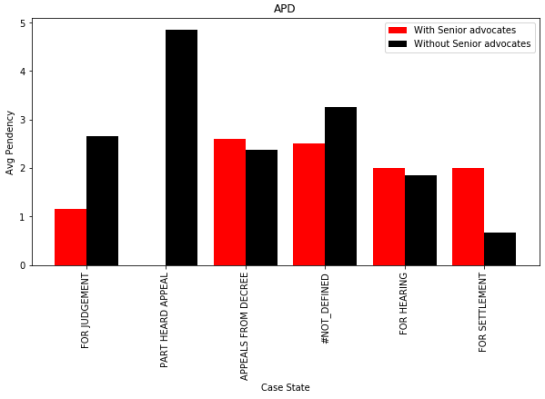
(b) CA



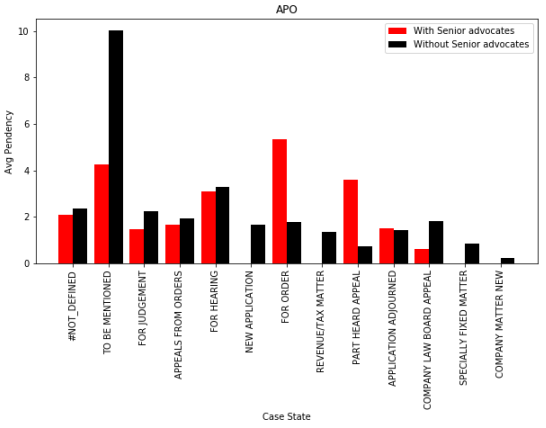
(c) AP



(d) APOT



(e) APD



(f) APO

FIGURE 8.2: Effect of presence of Senior Advocate(s) in both parties on the Pendency of cases for various case types

of this experiment were somewhat inconclusive as we see that the pendency of cases depends of various factors and the we cannot find a direct correlation between the presence/absence of a senior advocate and the number of years a case is pendent for each case type. We do however see that in most cases, the pendency with the presence of a senior advocate is either comparable or much lesser than the pendency without any senior advocate.

Chapter 9

Conclusion and Future Work

9.1 Conclusion

In this paper, we have introduced the problem of court scheduling in Indian Courts and discussed why it is necessary to develop algorithms to schedule cases such that the realised causelists are feasible, which the current system vastly lacks. We have also collected a large amount of data of causelists, case orders and case histories from the CHC, MCCC and SCoI websites. We have also analysed this data to support our claims for the need of the IST. We have also built simple next state predictors for the MCCC cases based on their case histories using bigram and n-gram probabilities.

Further, we have introduced a novel concept of case traces and reduced traces. We compared the lengths of the case traces and reduced traces and saw that there is a 3-4 fold reduction in their lengths, which means

that a reduced traces is a fairly good summary of its corresponding case trace. We have also disambiguated all the case states in the case traces of Original cases of CHC. These case states are mapped to more generalized case states at 2 different abstraction levels.

We have also used these traces to compute edit distances between cases which is then used to find the similarity of any 2 cases. We have used this similarity to cluster the cases and have observed that better clusters are being formed using similarity on reduced traces as compared to using similarity on case traces. We have also integrated orders into the similarity computation, and done similar experiments on the similarity matrix computed using this similarity. Similarity with orders highlights more similarity as compared to similarity without orders. This supports our claim that summarization gives better similarity. We have also developed a $CommonTopK$ metric to evaluate how good our similarity measure is. The metric shows that both the similarity metrics have scope of improvements.

Finally, we have done an ablation study on the effect of presence of senior advocates on the pendency of cases. The effect of this study were largely inconclusive because pendency is very complex and has a variety of causes.

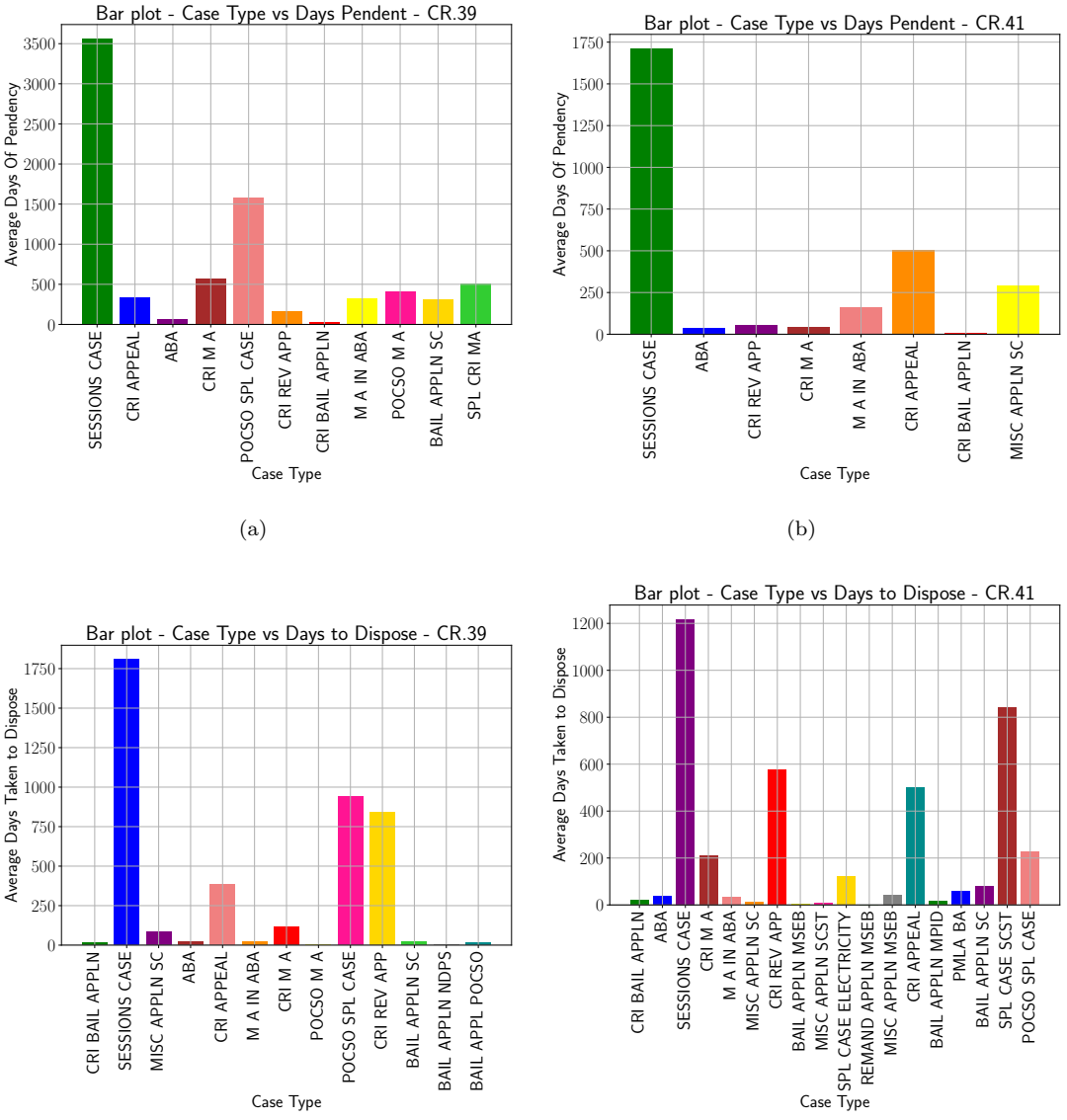
9.2 Future Work

We plan to store all the data collected in the proposed database structure. We also plan to use natural language processing to obtain the state of the hearing, judges presiding over the hearing, court hall, etc from the text in the order so that we can build the case histories for the CHC dataset. Comparing the complete case histories and case/reduced traces should give us interesting results. We plan to define a similarity based on edit distances between the case history and reduced trace. Our hope is that the reduced trace of a case will be more or less indistinguishable from its case history can be tested further with this similarity. Further, the case traces and case histories will be used for building a finite state automaton which mimics the progress of a case from one state to another, which will in turn give us a robust case state predictor. This state predictor will lead us to understand how much time a case spend on each state, when it is likely to move to the next state, etc, and thereby pave our way towards developing the IST.

Appendix A

Pendency of various case types from a MCCC case dataset

We have also obtained other datasets of the cases pendent and disposed under C.R.39 and C.R.41 under the MCCC. C.R.39 and C.R.41 refers to the sections 39 and 41 of the **CRPC** (The Code of Criminal Procedure). The dataset was analysed to understand how the pendency (time passed from institution) and lifetime (from institution to disposition) varies for the cases. We constructed various bar plots to study how these varies with the case types. We conclude by these bar plots that there is not an equitable distribution of judicial time between various case types, which leads to vastly varying pendencies and lifetimes of the cases in our dataset.



(a) (b)

(c) (d)

FIGURE A.1: Bar plots for an MCCC dataset

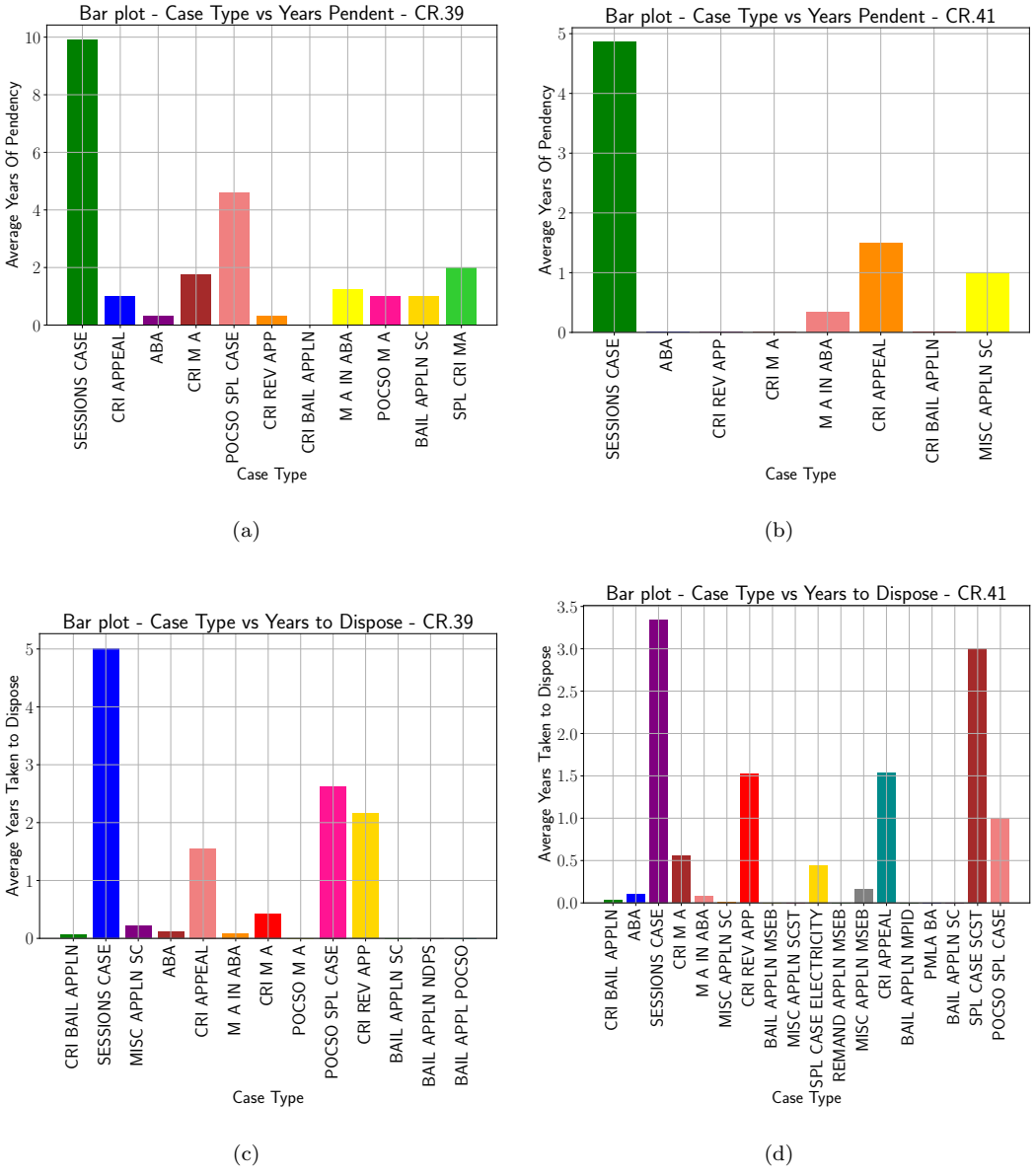


FIGURE A.2: Bar plots for an MCCC dataset

Bibliography

Al-Kofahi, K., Tyrrell, A., Vachher, A., and Jackson, P. (2001). A machine learning approach to prior case retrieval. In *Proceedings of the 8th International Conference on Artificial Intelligence and Law*, ICAIL '01, page 88–93, New York, NY, USA. Association for Computing Machinery.

Bach, N. X., Ngoc Cham, L. T., Ngoc Thien, T. H., and Phuong, T. M. (2017). Question analysis for vietnamese legal question answering. In *2017 9th International Conference on Knowledge and Systems Engineering (KSE)*, pages 154–159.

Bhattacharya, P., Paul, S., Ghosh, K., Ghosh, S., and Wyner, A. (2019). Identification of rhetorical roles of sentences in indian legal judgments.

Chalkidis, I., Androutsopoulos, I., and Aletras, N. (2019). Neural legal judgment prediction in English. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4317–4323, Florence, Italy. Association for Computational Linguistics.

Chen, H., Cai, D., Dai, W., Dai, Z., and Ding, Y. (2019). Charge-based prison term prediction with deep gating network.

Galgani, F., Compton, P., and Hoffmann, A. (2012). Towards automatic generation of catchphrases for legal case reports. In *Proceedings of the 13th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part II*, CICLing'12, page 414–425, Berlin, Heidelberg. Springer-Verlag.

Jackson, P., Al-Kofahi, K., Tyrrell, A., and Vachher, A. (2003). Information extraction from case law and retrieval of prior cases. *Artificial Intelligence*, 150(1):239–290. AI and Law.

Kim, M.-Y. and Goebel, R. (2017). Two-step cascaded textual entailment for legal bar exam question answering. In *Proceedings of the 16th Edition of the International Conference on Articial Intelligence and Law*, ICAIL '17, page 283–290, New York, NY, USA. Association for Computing Machinery.

Lupu, M. and Hanbury, A. (2013). Patent retrieval. *Found. Trends Inf. Retr.*, 7(1):1–97.

Moens, M.-F. (2007). Summarizing court decisions. *Inf. Process. Manage.*, 43(6):1748–1764.

Shao, Y., Mao, J., Liu, Y., Ma, W., Satoh, K., Zhang, M., and Ma, S. (2020). Bert-pli: Modeling paragraph-level interactions for legal case retrieval. In Bessiere, C., editor, *Proceedings of the Twenty-Ninth*

International Joint Conference on Artificial Intelligence, IJCAI-20, pages 3501–3507. International Joint Conferences on Artificial Intelligence Organization. Main track.

Wang, P., Fan, Y., Niu, S., Yang, Z., Zhang, Y., and Guo, J. (2019). Hierarchical matching network for crime classification. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 325–334, New York, NY, USA. Association for Computing Machinery.

Wang, P., Ze, Y., Niu, S., Zhang, Y., Zhang, L., and Niu, S. (2018). Modeling dynamic pairwise attention for crime classification over legal articles. pages 485–494.

Yang, W., Jia, W., Zhou, X., and Luo, Y. (2019). Legal judgment prediction via multi-perspective bi-feedback network. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4085–4091. International Joint Conferences on Artificial Intelligence Organization.

Zhong, H., Wang, Y., Tu, C., Zhang, T., Liu, Z., and Sun, M. (2020a). Iteratively questioning and answering for interpretable legal judgment prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):1250–1257.

Zhong, H., Xiao, C., Tu, C., Zhang, T., Liu, Z., and Sun, M. (2020b). How does nlp benefit legal system: A summary of legal artificial intelligence.