Publishing and Retrieval System for Traffic Court Cases

Wei Kit Shiu
School of Computer Science and Engineering
Nanyang Technological University
Singapore
Email: shiu0003@e.ntu.edu.sg

Chai Kiat Yeo
School of Computer Science and Engineering
Nanyang Technological University
Singapore
Email: asckyeo@ntu.edu.sg

Abstract—**This paper details the design and development of a web-based publishing and retrieval system for traffic court cases. This proof-of-concept is meant to complement and in time to come, replace, existing manual processes of doing legal research for traffic court cases. Currently, legal staff have to manually browse through practitioners' library and motor accident guide books to look at precedents for the assessment of damages in personal injuries and fatal accidents. The system automatically extracts key information of the court cases to allow retrieving of relevant court cases from a search query term by professionals such as judges, lawyers, insurers, as well as the public for their research and references.**

Keywords-traffic court cases; intelligent document retrieval system; natural language processing; automated text extraction.

I. Introduction

With the general improvement in road safety over the years, the number of accidents resulting in injuries has dropped slightly [1]. Nevertheless, it still amounts to more than 7000 cases per year in Singapore, a dense city state with 5.8 million population and 9.5 million motor vehicles. This naturally leads to a huge number of traffic accident cases reaching the courts as well as claims for injuries suffered and deaths during the accidents.

Accident victims will naturally seek compensation for injuries incurred. However, depending on how co-operative the offender is, the process to seek compensation may be difficult. The situation may involve an investigation by the related insurers and may even escalate into legal cases to be settled in the courts. This can be a long and expensive process in which compensations that are eventually awarded may not even be sufficient to cover the legal expenses of the disputes.

Typically, if the claim is heard before the courts, it will involve a detailed re-accounting of the accident as well as medical reports of the injuries incurred by the plaintiffs. The judge will then consider all details together with relevant past cases to decide on a quantum of the damages to be awarded to the plaintiff [2] [3].

Every year, there are up to 12,000 accident claims that are heard in the courts and they are important precedents for the judges to use for future references. Since 2001, a book named "Practitioners' Library Assessment of Damages: Personal Injuries and Fatal Accidents", commonly known as the Blue Book has been published with the 3rd edition launched in Feb 2017 [4]. It is written by judges and serves as a reference for judges, lawyers and insurers when it comes to assessing the amount of damages that the court may award in cases involving personal injuries and death. It also gives road users an idea of the damage awards in an accident. The Blue Book is also used by practitioners and members of the insurance industry to negotiate and expedite the settlement of accident cases without escalating the case to the courts. The Blue Book is almost 800 pages and referencing it is a tedious task, let alone revising it to keep it as up-to-date as possible.

Another book, the Motor Accident Guide [5], written in simple English and illustrated by dozens of diagrams picked from past court cases serve to provide readers, especially layman, an idea of where they stand should they take an accident claim to court. The guide aims to keep a lid on claims arising from motor accidents. Similar to the Blue Book, the readers will have to go through the entire guide to look for the scenario that is most applicable to his/her case.

The motivation of this proof-of-concept (POC) is therefore to introduce digitalization of court documents and facilitate the search for precedent motor accident cases. The project will facilitate judges to publish past cases efficiently and in a timely manner and also enable others concerned to efficiently retrieve and review information of the past cases without the need to laboriously go through the physical Blue Book.

The rest of the paper is organized as follows: Section II details the design and development of the system. Section III shows the outputs from the system and discussion on its performance. Section IV concludes this paper.

II. System Design and Development

The project is divided into two main parts namely Case Publishing System and Case Retrieval System. The former requires the uploading of pdf versions of case documents as well as conversion of pdf to text for further processing. It also entails the extraction of key information such as plaintiff's name, age, gender, date of assessment, injuries, claims and amount awarded. The extracted information is also stored and serves as search engine index to provide results to the search queries. The Case Retrieval System involves retrieving of the relevant case documents based on the queries made to the system. Figure 1 shows the use case diagram for the system.

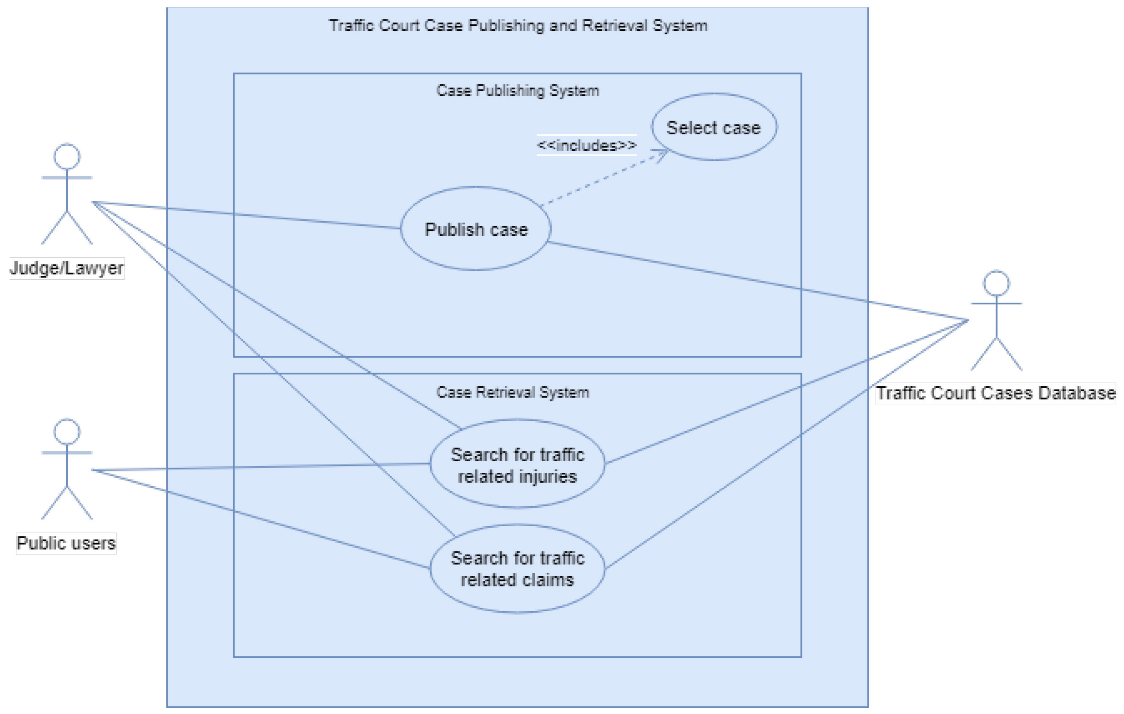

Traffic Court Case Publishing and Retrieval System
Case Publishing System
Select case
<<includes>>
Publish case
Judge/Lawyer
Traffic Court Cases Database
Case Retrieval System
Search for traffic related injuries
Public users
Search for traffic related claims

Figure 1. Use case diagram of the system.

The overall system comprises the Case Publishing System and the Case Retrieval System. The former allows the judges to upload past cases into the database and to edit or update published cases. The latter allows all concerned, namely, judges, lawyers and the public to access the databse and search for precedents matching the search terms entered such as the type of injuries, the award quantum.

A. Database

PyMySQL [6] is used to implement the system's database and the tools used to manage the database are Cross-Platform Apache, MariaDB, PHP and Perl (XAMPP) Control Panel [7] and phpMyAdmin [8]. The database is designed such that "cases" table holds a one-to-many relationship with the "injuries" table. Each case in the "cases" table is associated with one or more injury/claim in the "injuries" table. Each injury/claim in the "injuries" table uses its foreign key "case_ID" to identify its case mapping in the "cases" table. This design thus prevents data duplication.

B. Implementation

The system is fully written using Python.

PDF to text conversion: This function allows the uploaded PDF file to be converted to text format so that further processing can be done. A third-party library named "pdfminer.six" [9] is used here as it gives the best performance. It takes in an argument called [pdfname] where [pdfname] is the directory of the PDF file that is being uploaded and returns the text after the conversion is done. Many other libraries such as "PyPDF2" [10] have been used but the results are unsatisfactory as the converted text are either concatenated wrongly or there are missing text. However, "pffminer.six" is also not perfect and manual checking on the converted has to be performed. This is a big problem in the digitisation of past court cases and a one-off exercise will thus be needed to convert all the hard copies into digital form. Note that Natural Language Toolkit (NLTK) [11] is used to tokenize the converted text into sentences which are then parsed for the various extraction algorithms.

1) Extraction of plaintiff's name: Heuristic rule is applied in extracting the plaintiff's name after an analysis of the sample court cases on hand. The plaintiff's name will always appear at the top of every page in the document, in the form of "[Plaintiff Name] v [Defendant name]" and it is similar throughout all the cases. Therefore, the approach to this algorithm is to use regular expressions to extract the name. The *re.search* function takes [text] as a huge string and returns any substring that matches the pattern [r'(?P<PName>\b.*)\sv\s.*\b'], a regular expression created to match the format of the name given in the court document. Subsequently, symbolic group name *PName* is used to extract only the plaintiff's name. The code segment for the extraction is given in Figure 2.

We have explored the use of NLTK and pyenchant [9] for the plaintiff's name extraction. The algorithm is as follows: The converted text is tokenized into sentences and the sentences are parsed to extract those that contain the word "victim" or "plaintiff". The continuous name chunks are extracted using Name Entity Recognition with Regular Expressions and checked against those in the dictionary library. Name chunks with at least one word that is not a valid English word will be treated as a valid name but the result is not as good as the heuristic described above. Moreover, it is vulnerable to other word chunks that contain non-English words like national identity number and company name.

2) *Extraction of plaintiff's age:* The algorithm starts by tokenising the converted text into sentences. Next, it filters and extracts the sentences that contain keywords like "plaintiff" or "victim" and key phrases like "years old", "at the time", "accident happens", "when" etc. The reason of such key phrases is to improve the accuracy of extracting the plaintiff's age from neighbouring context. For example, "the plaintiff was 72 years old at the time of hearing". After obtaining the sentences that contain the keyword and key phrases, re.findall function is used to extract all the age numbers. The maximum of all the age numbers extracted will be set as the plaintiff's age at the time of assessment. An issue with this method is that the court documents may sometimes contain the age of more than one person. This will significantly increase the chances of extracting a wrong age number. Therefore, to reduce the odds of extracting a wrong age, a layer of filter is added to priortize the age number extracted from sentences that contain keywords like "plaintiff" or "victim" over others.

```
import re

#Algorithm starts here---
PlaintiffName = ""
#variable name [text] contains the text
converted from PDF
SearchPlaintiffName =
re.search(r'(?P<PName>\b.*)\sv\s.*\b',text
)
if(SearchPlaintiffName):
    PlaintiffName =
str(SearchPlaintiffName.group('PName'))
```

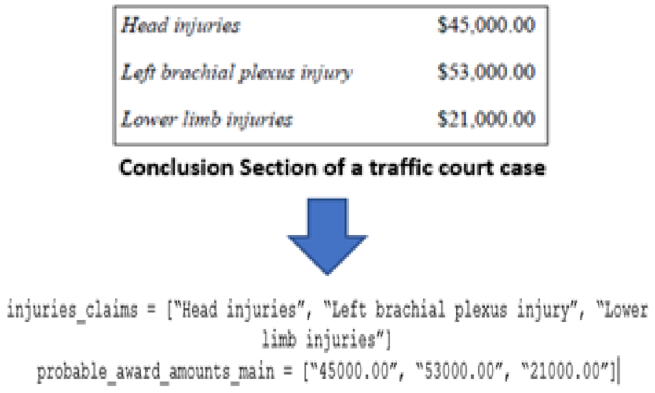
Figure 2. Code segment for extraction of plaintiff's name.

3) *Extraction of plaintiff's gender:*The main approach here is to identify the number of he/his and she/her pronouns that appears near to the keyword "plaintiff" or "victim" throughout the entire document. The higher count will be taken as the plaintiff's gender. First the converted text is tokenised into sentences. Next, for every sentence that contains the keyword "plaintiff" or "victim", the algorithm will count the number of times he/his and she/her appears. Finally, the gender with the higher count frequency will be taken as the plaintiff's gender.

4) *Extraction of date of assessment:* After analysing the cases on hand, it was found that the latest date shown in the traffic court cases is always the date of assessment. Therefore, the approach is to use re.findall function to extract every single date string that appears in the document and subsequently, extract the latest date out of all the date strings obtained. As the dates are extracted in the string format, an additional step is required to convert the date strings to numerical form so that the algorithm is able to compare every single date and recognise the latest date.

5) *Extraction of injuries, claims and amount awarded:* It is observed that every traffic court cases will have a section at the end of the document called "Conclusion". In the section, the injuries, claims and amount awarded will be listed out as a summary as shown in Figure 3. Therefore, the approach is to create two lists that store injuries/claims and amount awarded respectively, which is also shown in Figure 3. To implement the algorithm, a bag of words is created with all the relevant injuries/claims stored in it. With the help of the bag of words, the algorithm can identify and store the injuries/claims into the list "injuries_claims" while re.findall function is used to identify and store the amount awarded into the list "probable_award_amounts_main".

6) *Case Retrieval:* This function receives an input from the user and queries the database for matching results. User can searcj for traffic accident information by entering either the name of an injury or the name of a traffic accident claim. This function also allows substring search. For example, if the user enters "hand", cases that involved "left hand" injuries or "right hand" injuries will also be retrieved.



Head injuries	$45,000.00
Left brachial plexus injury	$53,000.00
Lower limb injuries	$21,000.00

Conclusion Section of a traffic court case

```
injuries_claims = ["Head injuries", "Left brachial plexus injury", "Lower
limb injuries"]
probable_award_amounts_main = ["45000.00", "53000.00", "21000.00"]
```

Figure 3. Illustration of the Conclusion section of the case summary.

III. RESULTS AND DISCUSSION

The capabilities of the system are evaluated against the requirements specified. Functional testing is adopted to examine the functions of the system to ensure that it performs as required. Four available traffic court cases are used as test dataset [12] – [15] and they are all past judgements made by the Supreme Court of Singapore. Unfortunately, only four cases are made available online.

Table I shows the test results. Basically, extraction of plaintiff's name, age, gender and date of assessments works well for the 4 test dataset with the exception of the extraction of injuries, claims and amount awarded. This extraction is the most challenging as it requires much more sophisticated natural language processing techniques such as topic modelling to extract the different types of injuries and the medical terms. The simple technique adopted in this POC proves to be inadequate to address the entire spectrum of possible injuries.

TABLE I. SYSTEM TEST RESULTS

Functionality Test	Accuracy
Extraction of plaintiff's name	100%
Extraction of plaintiff's age	100%
Extraction of plaintiff' gender	100%
Extraction of date of assessment	100%
Extraction of injuries, claims and amount awarded	25%

Figure 4 shows the key information extracted from the test case in [15] while Figure 5 shows the retrieval results when a user types in the search term '*fracture*'.

This POC has set the trail in the digitalization of the legal domain. It is very useful in facilitating the search for precedent court cases of traffic injuries and the amount of damages awarded to reduce expensive law suits and court time. It also shows that automation of text extraction from voluminous case files is feasible. The premise for this POC is that the court cases have already been digitized and exist in pdf form. This is not the case as most, if not all, case documents exist in hard copies and have to be manually digitized and checked before being published in a system like the proposed system. Only then can accurate extraction of critical information be performed and retrieval of case documents be accurate.

A qualitative comparison is made against existing related work such as [16], [17] and [18]. Wyner et. al. [16] detail the use of text mining to automatically profile and extract arguments from legal cases and shows how context-free grammar can be used to extract arguments, and how ontologies and NLP can identify complex information such as case factors and participant roles. The approach applies linguistic analysis and stereotypical pattern of reasoning called argument schemes to identify argument sentences and semantically relevant sentences from a legal corpus. The arguments in the legal corpus need to be first analysed and represented in XML format for later mining. Compared to our POC, we do not need manual labelling of the legal documents. We extract precise entities such as injuries, plaintiff's details and damage awards while [16]'s extraction is very coarse-grained in the form of sentences of arguments. [16] also does not lend itself to retrieve cases based on search queries.

Wagh [17] merely proposes a study to group legal documents based on the contents using unsupervised text mining techniques. It only describes what the authors intend to do with no actual design and implementation. Andrew and Tannier [18] use a combination of both statistical and rule based techniques to enable journalists to automatically identify and annotate entities such as names of people, organizations, role and functions of people in legal documents. They also try to explore the relationship between these entities. The statistical method used is Conditional Random Fields while document and language specific regular expressions are used for the rule based technique. It is focused on extraction of specific entities from the documents but do not include the more complicated entities such as injuries, damages awarded and age. It also does not support search and retrieval of precedent cases based on input query terms unlike our POC.

In summary, in comparison with existing work, our POC supports more precise and fine-grained extraction of plaintiff's details, injuries and damage awards based on the search string input thereby greatly facilitates users of the system to easily extract and compare precedent cases closest to their query of interest. Another merit of our POC is we do not require labelled dataset.

There are however limitations in this POC which need to be addressed before a fully functional system can be deployed as it relies heavily on heuristic algorithms for the unstructured text mining. Much more sophisticated natural language processing techniques, namely, topic modelling using Latent Dirichlet Allocation (LDA) is needed not just to extract the injuries but also in the extraction of other plaintiff's details. The test cases used here are considered simple as they only involve a single plaintiff and a single defendant. Hence, extraction of plaintiff's details is very accurate as shown in Table I, which will not be the case for multiple plaintiffs. Another challenge is when the search query comprises a long sentence instead of a single word. In this case, the key words have to be extracted from the search string as well. Moreover, there is a lack of readily available court cases, preferably in the hundreds, to adequately stress test the POC.

IV. CONCLUSION

A POC for a web-based publishing and retrieval system for traffic court cases has been successfully developed. The system automatically extracts key information of the court cases to allow retrieving of relevant cases from a search query term by professionals such as judges, lawyers, insurers and the public. Such a system not only renders the legal research process for traffic court cases to be much more efficient but also relieves the judges of the laborious manual compilation and update of the Practitioners' Library Assessment of Damages (the Blue Book). Judges can publish past cases much more efficiently and keep the publication up to date compared to the manually compiled Blue Book which is published once after a few years.

A limitation of the POC is the adoption of heuristics in the text mining. Future work shall involve the introduction of topic modeling in NLP processing to handle the extraction of plaintiff's details and injuries for more complex cases than those shown in this paper as well as use of deep learning.

ACKNOWLEDGMENT

This work is supported in part by NTU Grant no. M4081329.020.

REFERENCES

[1] Public Affairs Department Singapore Police Force. *Annual Road Traffic Sitatuon 2018*, Singapore, 2019.

[2] State Courts Practice Directions, Section 40. Singapore: State Courts.

[3] Supreme Court of Judicature Act, Chapter 322, Section 80, Order 37. Singapore, 2014 edition.

[4] C. Chan et al., Practitioners' Library Assessment of Damages: Personal Injuries and Fatal Accidents, 3rd ed., LexisNexis, Feb 2017.

[5] States Court, Motor Accident Guide, Tusitala (RLS) Pte Ltd, Feb 2017.

[6] I. Naoki, PyMySQL. 2017. [Online]. Available from: https://pypi.python.org/pypi/PyMySQL. [retrieved: Dec 2019].

[7] Apache Friends, XAMPP,. [Online]. Available from: https://www.apachefriends.org/index.html. [retrieved: Dec 2019].

[8] Software Freedom Conservancy, phpMyAdmin. [Online]. Available from: https://www.phpmyadmin.net/. [retrieved: Dec 2019].

[9] Y. Shinyama, pdfminer.six, 2014. [Online]. Available from: https://pypi.python.org/pypi/pdfminer.six/20140915. M. Fenniak, PyPDF2. 2011. [Online]. Available from:, https://pypi.python.org/pypi/PyPDF2. [retrieved: Dec 2019].

[10] M. Fenniak, PyPDF2. 2011. Available from: https://pypi.python.org/pypi/PyPDF2. [retrieved: Dec 2019].

[11] S. Bird, E. Loper and E. Klein, Natural Language Toolkit. 2014. Available from: https://www.nltk.org/. [retrieved: Dec 2019].

[12] K. Ramesh, High Court Judgement: Lee Mui Yeng v Ng Tong Yoo. 2016. [Online]. Available from: https://www.supremecourt.gov.sg/docs/default-source/module-document/judgement/-2016-sghc-46-pdf.pdf. [retrieved: Dec 2019].

[13] K. C. Pang, High Court Judgement: Tan Hun Boon v Rui Feng Travel Pte Ltd and another. 2017. [Online]. Available from : https://www.supremecourt.gov.sg/docs/default-source/module-document/judgement/judgment-s662-2014-v2-pdf.pdf. [retrieved: Dec 2019].

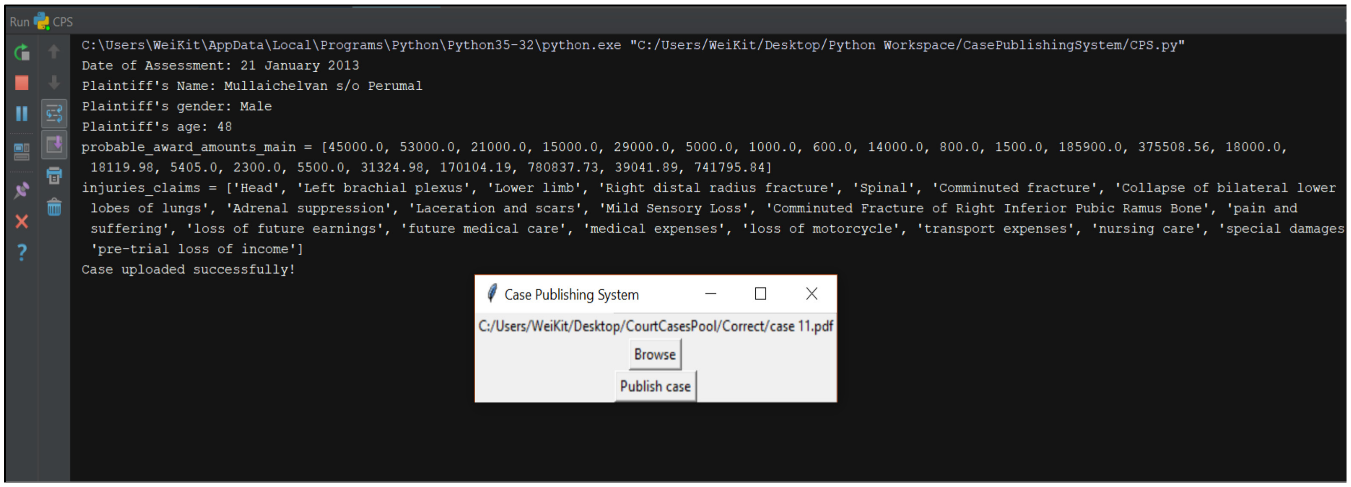
[14] J. Y. Lee, Supreme Court Judgement: Ng Hua Bak v Eu Kok Thai. 2016. [Online]. Available from: https://www.supremecourt.gov.sg/docs/default-source/module-document/judgement/s1351-14-ad43-15-sghcr-12-by-jaylee-2nov2016-pdf.pdf. [retrieved: Dec 2019].

[15] C. Seow, High Court Judgement: Mullaichelvan s/o Perumal v Lee Heng Kah. 2013. [Online]. Available from: https://www.supremecourt.gov.sg/docs/default-source/module-document/judgement/2013-sghcr-3.pdf [retrieved: Dec 2019].

[16] A. Wyner, R. Mochales-Palau, M. F. Moens, D. Milward, Approaches to Text Mining Arguments from Legal Cases. In: E. Francesconi, S. Montemagni, W. Peters, D. Tiscornia (eds) Semantic Processing of Legal Texts. Lecture Notes in Computer Science, vol 6036. Springer, Berlin, Heidelberg, 2010.

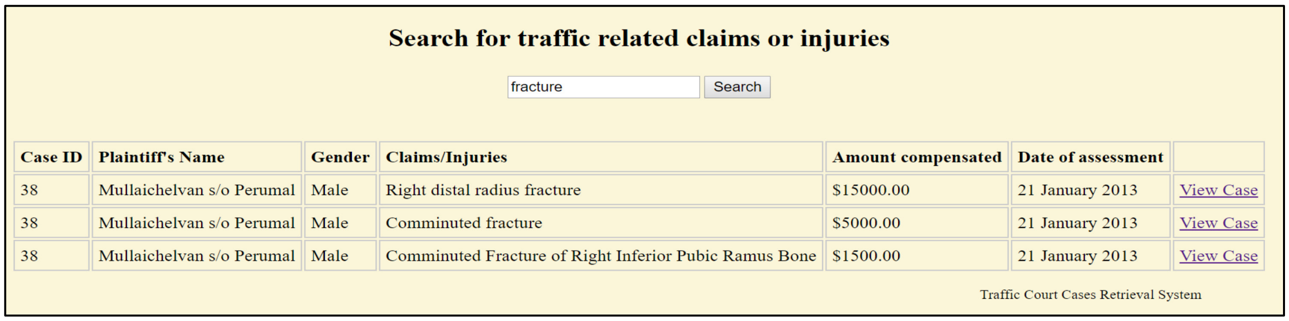
[17] R. S. Wagh, Knowledge Discovery from Legal Documents Dataset using Text Mining Techniques, International Journal of Computer Applications 66(23):32-34, 2013.

[18] J. J. Andrew and X. Tannier, Automatic Extraction of Entities and Relation from Legal Documents, Proceedings of the Seventh Named Entities Workshop, Melbourne, pp. 1-8, Jul 2018.



```
C:\Users\WeiKit\AppData\Local\Programs\Python\Python35-32\python.exe "C:/Users/WeiKit/Desktop/Python Workspace/CasePublishingSystem/CPS.py"
Date of Assessment: 21 January 2013
Plaintiff's Name: Mullaichelvan s/o Perumal
Plaintiff's gender: Male
Plaintiff's age: 48
probable_award_amounts_main = [45000.0, 53000.0, 21000.0, 15000.0, 29000.0, 5000.0, 1000.0, 600.0, 14000.0, 800.0, 1500.0, 185900.0, 375508.56, 18000.0,
 18119.98, 5405.0, 2300.0, 5500.0, 31324.98, 170104.19, 780837.73, 39041.89, 741795.84]
injuries_claims = ['Head', 'Left brachial plexus', 'Lower limb', 'Right distal radius fracture', 'Spinal', 'Comminuted fracture', 'Collapse of bilateral lower
 lobes of lungs', 'Adrenal suppression', 'Laceration and scars', 'Mild Sensory Loss', 'Comminuted Fracture of Right Inferior Pubic Ramus Bone', 'pain and
 suffering', 'loss of future earnings', 'future medical care', 'medical expenses', 'loss of motorcycle', 'transport expenses', 'nursing care', 'special damages
 'pre-trial loss of income']
Case uploaded successfully!
```

Figure 4. Extraction of key data from Test Case in [15].



Search for traffic related claims or injuries

Case ID	Plaintiff's Name	Gender	Claims/Injuries	Amount compensated	Date of assessment	
38	Mullaichelvan s/o Perumal	Male	Right distal radius fracture	$15000.00	21 January 2013	View Case
38	Mullaichelvan s/o Perumal	Male	Comminuted fracture	$5000.00	21 January 2013	View Case
38	Mullaichelvan s/o Perumal	Male	Comminuted Fracture of Right Inferior Pubic Ramus Bone	$1500.00	21 January 2013	View Case

Figure 5. Case retrieval results for the search term ‘fracture’.