Identifying Keywords in Legal Articles using Machine Learning Techniques

- Final Report -

Kshithija Dilhani
Ishani Maduwanthi
Wathsari Samindani

Department of Computer Engineering
University of Peradeniya

Final Year Project (courses CO421 & CO425) report submitted as a requirement of the degree of
B.Sc.Eng. in Computer Engineering

March 2021

Supervisors: Dr. Janaka Alawatugoda (University of Peradeniya), Dr. Damayanthi Herath (University of Peradeniya) and Mr. Thidas Maduwalthanne (Lawyer)

We would like to dedicate this thesis to my loving parents and “teachers” ...

Declaration

We hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is our own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgments.

Kshithija Dilhani
Ishani Maduwanthi
Wathsari Samindani
March 2021

Acknowledgements

At first we would like to acknowledge our supervisors Dr.Janaka Alawathugoda, Dr. Damayanthi Herath and Mr. Thidas Herath Maduwalthanne for providing us with a solid background for this project. So far they have been with us to guide, encourage, motivate and support to complete these tasks. In every meeting, they checked our progress and gave us feedback to improve our quality of work. Not only that, they provided many references which were related to our project and tried to make it easy for us.

Next we would like to acknowledge the Department of Computer Engineering, Faculty of Engineering, University of Peradeniya. Specially to Department Head Dr. Kamalanath Samarakoon for providing us with necessary opportunities to succeed in our project and all the lectures, who support this project to make it true. Instructors of the computer engineering department also gave us the support. So we would like to give our special thanks to them too.

Furthermore, we would like to give our special thanks to our parents for the help and support that has been given us to succeed in every step. Next we would like to thank all our friends for thoughtful and valuable discussions that helped to the success of the project.

Finally, We would like to give our gratitude to each and every one who helped us in every possible way to succeed the project.

Abstract

This paper presents a survey of strategies and approaches for keyword extraction task. The paper provides an in depth review of existing analysis, additionally to the organisation of strategies. Related work on keyword extraction is concerned for supervised and unsupervised learning strategies.

Nowadays there are plenty of legal documents offered in electronic format. Therefore, legal scholars and professionals are in need of systems able to search and quantify connotative details of those documents. Legitimate information and customary laws are generally offered in raw form and onerous to know, since they are not in organized form. All legitimate information is nowadays processed since the legal information gets generated often in a large volume due to the rise of law courts. The objective of this analysis is to explore an associate economical way to implement an algorithm to identify keywords by predicting the connectedness of legal documents from an enormous classification system which is difficult to do manually.

The system to analyze this legal knowledge will serve effectively for lawyers and law students, which might address a lawyer's role and may even become powerful to unleash such a task in future. Designers of such systems face a key challenge that the bulk of those documents are in natural language streams which are lacking formal structure or different specific linguistics information. During this analysis, we tend to describe associate unsupervised learning approach for automatically distinguishing necessary details in each legal document.

The machine learning and deep learning algorithms based mostly analysis systems apply these strategies in the main for document classification. Legal document classification, translation, account, data obtention are part of the goals obtained from this research. During this study, we tend to review the various strategies of deep learning employed in legal tasks like Legal knowledge search, Legal document analytics, and Legal perspective interface. Through this review, we tend to instituted that machine learning models are giving advanced performance.

Table of contents

Chapter 1

Introduction

1.1 What is a keyword?

Keywords are considered as important parameters in a given context. For example people, places, words, or ideas that provides the idea of the relevant context. In this research, legal documents are used to obtain keywords, and thus the keyword is expected to convey a considerable idea of the legal document. Then, the key is the quality measuring parameter, which represents the importance of the given context.

Keywords can be single word or multi-word keywords, which is known as key phrases. Phrases can be made by combining words together , and they usually generate a new meaning which is not related to the meaning given by single keywords. Therefore, if we take only single words as keywords, then , it would sometimes miss the significant things in the document.

There are two factors considered in the process of identifying keywords. First, if a word is more frequently occurs in the document, then it can take as a keyword. And second, if a word is more frequently occurs in a speech,it has a less chance to take as a keyword of any document. According to the second factor the words that very frequently use in a speech, such as prepositions, conjunctions or common nouns, cannot be considered as keywords.

1.2 What is keyword extraction?

eyword extraction is one of the text analysis methods, extract the most important words in the document and express the idea of it. It helps to get a summarizing of the content of the document. A keyword is a important unique word that convey whole idea of a

document,or a word which is used to find information when studying legal cases. They can express approximately the overall idea of the document. Keywords are also called as 'Search Queries', since they are the words or phrases that people use when they are searching. Keywords are important since they provide the connection between what people search for and what system they have.

When there are thousands of documents, keyword extraction helps to find the best matching document for our purpose. Keywords may be considered as a summary for a document which lead to have information extraction, or to categorize a document collection. However, in our case , there are relatively few documents keywords are assigned. Therefore finding methods to automate the assignment is important thing in legal context.

1.3 Why keyword extraction for legal articles?

1.3.1 Problem Statement

When lawyers, attorneys or paralegals obtain a legal case to work on, one of the first steps is the searching case related documents. They need to search, clean, and organize the important facts in legal documents in order to study the legal concepts and find pieces of evidence in a legal case. With increasing amounts of information in the electronic form, lawyers put more emphasis on an efficient process of searching documents using keywords. It is common that a legal case consists of a single page to thousands of related documents, which a lawyer needs to scan through to understand a case and find useful information, potentially a shred of evidence.

The goal of this research is to study, design and evaluate a mechanism that is able to highlight legal documents with valuable information and filter out the irrelevant. Such a tool could save the time spent on reading the documents and thus, increase the lawyers' efficiency. This paper presents how text rank keyword extraction methods can be applicable to keyword extraction from legal articles.

1.3.2 Background

Reading legal documents is a very difficult task and sometimes it needs some domain knowledge related to that document. And also it is hard to read the full legal document without missing the key important sentences and it is a very time consuming task. With an increasing number of legal documents it would be convenient to get the essential

information from the document without having to go through the whole document. Hence manual extraction of keywords is slow, expensive and prone to mistakes.

Finding database and e-Resource that provide legal and legislative information is vital need for lawyers in Sri Lanka.There are current implementations but those systems does not come up with efficient solution.There also manual work is costly.We need to reduce man work from the beginning of the portal.To implement user friendly and a system which learn itself to categorize documents the keyword extraction is essential. Also part of this research important information is mined. Therefore, many algorithms and systems for automatic keyword extraction have been proposed in the recent past. Those experiments are the basic background for this paper.

This report is structured as follows. First chapter brings the introduction with problem statement and background for this research. Then second chapter does the literature survey recording the previous work has been done on keyword extraction. Third chapter will describe the methodology that we followed to mined the information from the legal documents and the methodology we followed for extracting the keywords for relevant document and implementation details . Forth chapter describes the results and evaluation of the study. Then fifth chapter analysis of our study. Finally sixth chapters brings the conclusion and future works to be done regarding this domain.

Chapter 2

Literature Review

2.1 Introduction

Early automatically keywords extraction approaches were focused on evaluating corpus-oriented statistics of extracting individual words. In 1972 Jones and in 1975 Salton described positive results of selecting for an index vocabulary the statistically discriminating words across a corpus. Later keyword extraction research applies these metrics to select different words as keywords for individual documents. As an example, in 1998, Andrade and Valencia used the comparison of word frequency distributions within a text against distributions from a reference collection. But corpus-oriented methods also typically operate only on single words.

Document-oriented methods of keyword extraction have combined natural language processing approaches to identify part-of-speech (POS) tags. They are combined with supervised learning, machine-learning algorithms, or statistical methods. Considering the existing methods that are used in automatic keyword extraction, can be divided into four categories: simple statistics, linguistics, machine learning and other approaches.

2.1.1 Simple Statistic Approaches

Statistical methods are simple and training data is not needed.To identify the keywords in a document statistics information of the words can be used.The statistical keyword extraction methods are based on a sentence-term matrix.In this matrix, each row corresponds to a sentence of a document and each term corresponds to a column.A term can be a individual word or a sequence/set of words.

2.1.2 Linguistics Approaches

These approaches use the linguistics feature of the words mainly, sentences and documents.The Linguistic Approach includes lexical analysis, syntactic analysis, discourse analysis and so on.

2.1.3 Machine Learning Approaches

Keyword extraction can be seen as both supervised learning and unsupervised learning.Machine learning approach can be used to train a model by using previously extracted keywords and applying the model to extract keywords from new documents. Machine learning technique is mainly categorized into two sections as supervised learning methods and unsupervised learning methods. This approach includes Naïve Bayes, SVM, Bagging, etc.Some keyword extraction tools, e.g. KEA [1], GenEx[2] have been developed. Following section will be described more on machine learning approaches.

2.2 Machine learning Approaches

In this research we followed machine learning techniques to develop the searching algorithm. Machine learning technique is mainly categorized into two sections as supervised learning methods and unsupervised learning methods.

2.2.1 Supervised Learning Methods

Supervised learning is a learning that is supervised by a set of examples with class assignments. The goal of this approach is to find a representation of the problem in some feature space that is used to build up profiles of the classes. The supervised approach problem is treated as a classification task. Well-known classification models for the supervised learning method are naïve Bayes classification, classification by the C4.5 decision tree induction and neural networks.

In this model, first documents should be trained which are already labeled with keyphrases assigned by humans because keywords are taken from unseen documents. Peter Turney [2]developed key-phrase extraction, there they considered as correct keyphrases if and only if they matches with human assigned keyphrases. KEA is another algorithm to extract keyphrases. A classifier supporting the Bayes theorem using training documents is built. LAKE is another system and it uses linguistic features for learning and used to identify candidates. Linguistic knowledge, i.e. part-of-speech (pos) tags was introduced

by Hulth in determining the candidate sets and uses 56 potential pos-patterns [3]. In A fuzzy classification method for websites is proposed by Zhang and Lu [4].

2.2.2 Unsupervised Learning Methods

In unsupervised learning, no need to training data. A general set of candidate phrases from the given document is selected and a ranking method is created to take most important words.Using simple heuristics nonun phrases were extracted and ranked them by Barker and Cornacchia [5]. Bracewell et al. could able to extract noun phrases from a document [6]. The clusters are ranked on the idea of term and phrase frequencies.

2.2.3 Graph-Based Methods

Following are some related works through the years. In 1998 an algorithm for automatic indexing by co-occurrence graphs constructed from metaphors was presented by Oshawa[7]. It is called KeyGraph. it is based on the segmenting of a graph and representing the co-occurrence between terms in an article. In 2013 Boudin compared various centrality measures for graph-based keyphrase extraction[8]. Experiments on standard data sets show that Easy degree centrality achieves results which corresponding to the TextRank algorithm obtains the most effective results on short documents.

In 2009 community detection techniques for key terms extraction on Wikipedia's texts were used by Grineva [9]. The results showed that the terms associated with the most topics of the document tend to create a community, thematically cohesive groups of terms.

A stochastic graph based method for computing the relative importance of textual units on the matter of text summarization was presented by Erkan and Radev. LexRank calculates sentence importance and supports the concept of eigenvector centrality in an exceedingly graph representation of sentences.

2.2.4 Other Methods

The textRank algorithm for the text summarization is introcuced by Mihalcea in 2004 and it is used in sentence extraction. A network based ranking algorithm called SemanticRank is introduced by Tsatsaronis in 2010. Semantic relatedness between linguistic units and keywords is the basic of linguistic relation. The keyword extraction from the Inspec abstracts' results reported a good performance of SemanticRank over state-of-the-art counterparts - weighted and unweighted variations of PageRank and HITS.

In 2014 a keyword extraction method to represent tweets as graphs is presented by Abilhoa and de Castro[10]. Further they have applied centrality measures. Mihalcea in 2004 presented an extension to earlier work. In text summarization task text rank can be applied.

Chapter 3

Methodology

3.1 Methodology

The software that is going to develop should be able to identify certain key data contained in the judgements in the website and extract the keywords for each document. Website will contain judgements, statutes and various other content. We need an intelligent system which can;

1. Identify,

-The names of the previous judgements
-The names of the statutes, section numbers
-The Key legal concepts discussed in the body of the legal document
-Court name
-The date and the year of the judgement
-Judges who are involved in the judgement and their roles
-Lawyers who are involved in the judgement and their roles

2. Extract the Keywords from each document. The experimental setup is as follows we

used NLR and unreported supreme court articles for this research that was downloaded from lawnet web site (https://www.lawnet.gov.lk/) which was Developed by Lanka Logistics Technologies Ltd. - Ministry of Defence

3.1.1 Identify the previous judgements referred to in the judgement and the surrounding sentence

Usually, most judgements will refer to various other previous judgements. These previous judgements can be found at several places of the document. In a judgement, the previous judgements can be found inside the judgement, within paragraphs.

These previous judgements can be found in different formats.

-In some cases, the previous judgments which are found inside the judgement could be given as a list, at the beginning or end of the current judgement.

3.1.2 Identify the key legal concepts

Legal concepts typically come across in the context of legal documents, and they are used in legal inferences. To Identify those key legal concepts we have used **Black's Law Dictionary** [11]because it presents how judgments are published and relevant key concepts used in. It's the most widely cited law book in the world.

Eg: -

- SECTION 190 OF **THE CEYLON PENAL CODE**
- SECTION 439 OF **THE CRIMINAL PROCEDURE CODE**
- SECTION 372 OF **THE PENAL CODE**
- SECTION 18 OF **THE RENT RESTRICTION ACT**
- SECTION 47 OF **THE RENT ACT**

3.1.3 Tf-Idf method for keyword extraction

Term frequency(Tf)–Inverse document frequency(Idf), is a formula that measures how important a word is to a text document. Term frequency means the number of times a word appears in a text and Inverse document frequency means how rare a word is in the entire text. After multiplying these two quantities it provides the $TF-IDF$ score of a specific word in a document. When the score is high,word is more relevant to the document. $TD-IDF$algorithms in machine learning are used for several applications.Most of the time search engines use various types of $TF-IDF$algorithms when searching queries.. This Tf-Idf algorithm metric helps to identify the most relevant words in a document which have the higher tf-idf scores in keyword extraction. In some cases, the words that appear frequently in a collection of documents are not necessarily the most relevant. And

also a word that appears in a single text but doesn't appear in the remaining text may be very important to understand the content of that text. In a text,words like and, if, the, this or what, will probably have the higher frequencies And there will be content related words which have higher frequencies but not provide much details about the document.But in this $TF-IDF$algorithm,we are able to weigh the importance of each term and extract the keywords that helps to get the overall idea of the document. [6]

If we write this in formal mathematical terms, $tf-idf$ for a word$t$in the document d from document collection$D$is calculated as,

$$tfidf(t,d,D) = tf(t,d).idf(t,D)$$

Where,

$$tf(t,d) = log(1 + freq(t,d)$$

and

$$idf(t,D) = log(N/count(d \subset D : t \subset d)$$

When machine learning is used to work with natural language processing(NLP) the major problem is its algorithms usually work with numbers.Therefore we need to transform text into numbers.This is also known as text vectorization.It is one of the fundamental steps for analyzing data in machine learning. After transforming text into numbers that machine learning algorithms can understand ,tf-idf scores can be fed to machine learning algorithms like Support Vector Machine(SVM) and Naive Bayes. In this method ,a word vector represents a text as a list of numbers,with one for each possible word of the corpus.Text vectorizing means that taking the text and creating one of these vectors and the vector numbers represents the content of the text.TF-IDF enables us to gives us a way to associate each word in a document with a number that shows how relevant each word is in that document. Then, documents with synonyms will have similar vectors, which is what we are looking for in a machine learning algorithm.

3.1.4 Text Rank method for keyword extraction

Text Rank is a graph based algorithm and it is derived from a popular algorithm which is used by Google called Page Rank. In this algorithm, it will be calculated the importance of a vertex using the global information extracted from the entire graph. The importance of a vertex is defined according to the linkage between two vertices. It is called voting. Having higher votes for vertex may become more important. Then a score will be calculated to build the ranking model. Mathematically. Let $G = (V, E)$ be a directed

graph where V denotes for the vertices and E denoted for the edges, for a given vertex Vi, let $In(V_i)$ be the predecessors and $Out(V_i)$ be successors. Then the score of vertex is defined as follows:

$$S(V_i) = (1-d) + d * \sum_{j \epsilon (V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

Here d is the damping factor and it is the probability of going from one vertex to another.d is set in between 0 and 1. As mentioned in Brain and Page (1998) damping factor is set as 0.85 and in our case also we are using this number[12].

3.2 Implementation

3.2.1 Keyword Extraction from Tf-Idf method

Extract content of the PDF and convert it into text format.

Here we used pdfminer library to convert pdf files to text files

```
import io
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
from pdfminer.pdfpage import PDFPage

def convert(fname,pages=None):
  infile = open(fname, 'rb')
   if not pages:
       pagenums = set()
   else:
       pagenums = set(pages)

   output = io.StringIO()
   manager = PDFResourceManager()
   converter = TextConverter(manager, output, laparams=LAParams())
   interpreter = PDFPageInterpreter(manager, converter)

   for page in PDFPage.get_pages(infile, pagenums):
       interpreter.process_page(page)
   infile.close()
```

```
    converter.close()
    text = output.getvalue()
    file1 = open("myfile.txt", "w")
    n = file1.write(text)
    file1.close()
    output.close
    return n
```

Data Pre-processing

```
import spacy

nlp = spacy.load("en_core_web_sm")
file = open("myfile.txt", "rt")
data = file.read()

def textProcessing(doc):
    '''Prepocessing of input text with
    1. tokenisation and Lemmatisation
    2. Removing stop words
    3. Creating and removing custom stop words.
    4. Generating required Vocabulary from input
    5. Preprocessing the input
    '''
    Nouns = []
    Noun_set = []
    trimmed_noun_set = []
    removing_duplicates = []
    arr = []
    vocab = []
    vocab_dict = {}

    doc = nlp(doc.upper())#Convert to upper case

    for possible_nouns in doc:
        if possible_nouns.pos_ in ["NOUN","PROPN"] : #PROPN ->Proper Nouns
            Nouns.append([possible_nouns , [child for child in possible_nouns.childre
```

```
    for i,j in Nouns:
        for k in j:
            Noun_set.append([k,i])

    for i , j in Noun_set:
        if i.pos_ in ['PROPN','NOUN','ADJ']:
            trimmed_noun_set.append([i ,j])

    for word in trimmed_noun_set:
        if word not in removing_duplicates:
            removing_duplicates.append(word)

    for i in removing_duplicates:
        strs = ''
        for j in i:
            strs += str(j)+" "
        arr.append(strs.strip())

    for word in Noun_set:
        string = ''
        for j in word:
            string+= str(j)+ " "
        vocab.append(string.strip())

    for word in vocab:
        vocab_dict[word]= 0

    for word in arr:
        vocab_dict[word]+= 1

    return vocab_dict,arr
```

Implementation of Tf-Idf method

```
def computeTF(wordDict,bow):
```

```
    '''Computing TF(Term Frequency of the vocab) '''
    tfDict = {}
    bowCount = len(bow)
    for word, count in wordDict.items():
        tfDict[word] = count/float(bowCount)
    return tfDict


def computeIDF(doclist):
    '''Computing IDF for the vocab '''
    import math
    count = 0
    idfDict = {}
    for element in doclist:
        for j in element:
            count+=1
    N = count

    # count no of documents that contain the word w
    idfDict = dict.fromkeys(doclist[0].keys(),0)

    for doc in doclist:
        for word,val in doc.items():
            if val>0:
                idfDict[word]+= 1

    # divide N by denominator above
    for word,val in idfDict.items():
        if val == 0:
            idfDict[word] = 0.0
        else:
            idfDict[word] = math.log(N / float(val))

    return idfDict

def computeTfidf(tf,idf):
```

```
    '''Computing TF-IDF for the words in text '''
    tfidf = {}
    sorted_list = []
    for word , val in tf.items():
        tfidf[word] = val * idf[word]

    ranking_list  = sorted(tfidf.items(),reverse=True, key = lambda kv:(kv[1], kv[0])
    for i, _ in ranking_list:
        sorted_list.append(i)

    return sorted_list


vocab_dict , arr = pre.textProcessing(data)
tf = f.computeTF(vocab_dict,arr)
idf = f.computeIDF([vocab_dict])
keywords = f.computeTfidf(tf,idf)

with open("keywords.txt", "w") as outfile:
    outfile.write("\n".join(keywords))

print('keywords:')
print(keywords)
```

Implementation of Text Rank method

In our implementation it follows Mihalcea and Tarau, 2004 implementation but we have chosen few design decisions according to our application[12]. There we have used undirected graph implementation for keyword extraction then the graphs are created using a co-occurrence matrix with co-occurrence window size two. Then the similarity measure is based on the normalized word overlap between adjacent sentences. Then their texts are normalized into lower case while some unicode characters are replaced by their proper unicode counterparts. Tokenization is done using NLTK's enhanced Treebank Word Tokenizer and to remove unnecessary words , NLTK's built-in stopword set is utilized while by only selecting nouns and adjectives Part-of-Speech tagging is used. For the ranking as previously mentioned, damping factor was selected as 0.85 and the delta rank score is set as 0.0001.

The implementation code consists of three python files. Mainly following are considered as prerequisites

- Python 2.7 or Python 3.*
- NumPy
- NLTK

data.py , model.py and textrank.py are the three python files. In data.py file the preprocessing data is done using NLTK libraries such as pos tagging, vocabulary building, stemming etc. Then in the model.py file by using numpy library build the co-occurrence matrix and ranking model is applied. Finally the keyword extraction is done under textrank.py python file.

3.2.2 Find Previous Judgements,Case name and Court name

```
previous_judgments = ["THE CASE OF", "THE JUDGMENT OF", "VIDE", "VS", "HELD IN"]
count = 0
for sentence in sentences_list:
    tokens = nltk.word_tokenize(sentence)
    for i in range(4):
        for token in tokens:
            if token == previous_judgments[i]:
                if (i == 3):
                    count = count + 1
                    if (count > 1):
                        previous_judgment_list.append(sentence)

                else:
                    previous_judgment_list.append(sentence)

if len(previous_judgment_list) > 0:
    print("The Previous Judgements:")
else:
    print("No previous judgments")
```

```
for p in previous_judgment_list:
    if (p != name):
        print(p)
```

3.2.3 Find Key Legal Concepts

```
pattern1 = "\bSECTION\s\d+\s\w+\s\w+\s\w+\s\w+\s+\w+\b"
pattern2 = "\bSECTION\s\d+\s\w+\s\w+\s\w+\s\w{4}\b"
pattern3 = "\bCOPYRIGHT\s\w+\s\w+\s\w+\s\w+\b"
pattern4 = "\bSECTION\s\d+\s\w{2}\s\w{3}\s\w+\s\w{2}\s+\w+\s\w+\s\w{3}\s\w+\s\d+\s\w{

match1 = re.findall(r"\bSECTION\s\d+\s\w{2}\s\w{3}\s\w+\s\w+\s+\w+\b", result)
match2 = re.findall(r"\bSECTION\s\d+\s\w{2}\s\w+\s\w+\s\w{4}\b",result)
match3 = re.findall(r"\bCOPYRIGHT\s\w+\s\w+\s\w+\s\w+\b",result)
match4 = re.findall(r"\bSECTION\s\d+\s\w{2}\s\w{3}\s\w+\s\w{2}\s+\w+\s\w+\s\w{3}\s\w+
```

matches = [ ]
matches = match1 + match2 + match3 + match4

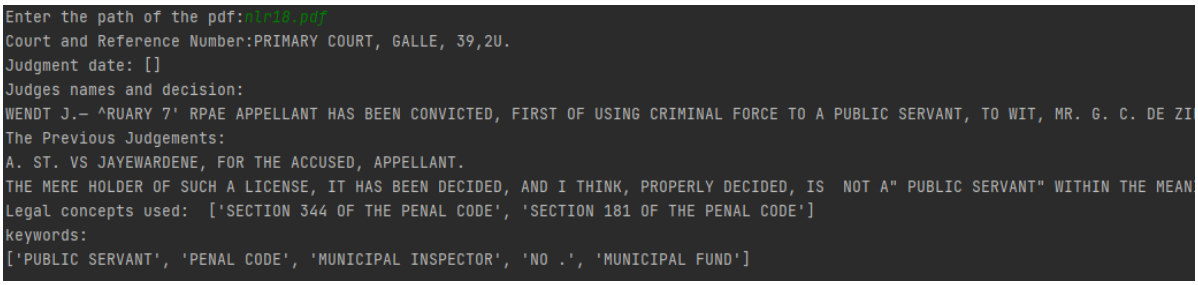
```
concepts = [ ]

for i in matches:

    if i not in concepts:

        concepts.append(i)

print('Legal concepts used: ',concepts)
```

Chapter 4

Results and Evaluation

4.1 Results

Fig. 4.1 Results



keyword Extraction

- TF-IDF Method

Document Type	Number of correct keywords	Number of wrong keywords
NLR	110	143
Supreme Court	51	39

Accuracy for NLR = 0.4347
Accuracy for Supreme court = 0.5666

- Text-Rank Method

Document Type	Number of correct keywords	Number of wrong keywords
NLR	186	311
Supreme Court	99	151

Accuracy for NLR = 0.3742
Accuracy for Supreme court = 0.396

4.2 Evaluation

4.2.1 Result Evaluation

So far we have Identified,

- The names of the previous judgements
- The names of the statutes, section numbers Key legal concepts discussed in the body of the legal document
- Court name
- The date and the year of the judgement
- Judges who are involved in the judgement and their roles
- Lawyers who are involved in the judgement and their roles
- Extracted the Keywords from each document.

We could identify the above mentioned key information for a given document. When considering keyword extraction results, TF-IDF Method table shows that TF-IDF methods shows 0.4347 accuracy for NLR documents and 0.5666 accuracy for supreme court documents. Text-Rank Method table results shows the number of correctly and wrongly identified keywords accordingly. With respect to that the NLR data-set has achieved 0. 3742 accuracy and supreme court data-set has achieved 0. 3960 accuracy.

To evaluate the results we had to use manual method because its not like assigning keywords to other documents, legal documents have different context and assigning keywords need prior knowledge for legal documents. When it comes to automate the keyword extraction, therefore we had to do evaluation by manually.

4.2.2 Difficulties and challenges

There are several challenges we have to face in this scenario as follows,

- The major issue is that there is no exact definition for keywords.We can not define a keyword exactly.As an example someone can say this word is a keyword for this document ,while another one can say it is not.Its mean that there are no baseline to identify a keyword.Therefore,in evaluation process we have to get support from a expert in this legal domain to check that whether the extracted keywords are relevant to the particular document.

- The judgments in our website will be HTML files and not PDF files.But Since the system is still in the developing stage there are no sufficient case law reports in HTML format.Therefore,we have to use scanned pdf files until HTML files will be available.When extracting the data from pdf there are several issues.When compare the pdf with extracted text there are some mismatching words .The reason for that is unclearness of the pdfs since they are scanned pdfs.

- There is not exactly one format in case law.Mainly there are four types.They are,
  1. NLR format
  2. SLR format
  3. Unreported Supreme Court cases format
  4. Unreported Court of Appeal cases format

  The way the content is arranged in each judgement will differ from one another.Therefore the system should understand all of the variations in all formats.

- There are a number of dates mentioned in these case law reports both past and future dates.So it is important to identify the date which is the exact date that case is conducted at the court.Also there are several formats in dates.As an example,23rd December 2020 can be mentioned as 2020-12-23 or 23-12-2020 and etc.Therefore System should identify all those formats.

Chapter 5

Conclusion and Future Works

5.1 Conclusion

The goal of this research is to discover answers on the questions of keyword identifying process of legal domain, especially, legal documents vary from others. It is considered about what are the things that make a legal document unique, which features important most in each document, if the formation is important in the applicable prediction, and what mechanisms work best for applicable prediction in the legal domain.

5.2 Future Works

During the experiments on data, some ideas on enhancement of the relevance prediction were proposed. Our plan was to implement a method searching documents using a single keyword or keyword phrase. There are still chances for additional improvements , which might rapidly accelerate and simplify lawyer's work.Expectantly, this research will help everyone who are involved in the legal domain and software developers in coming-up decisions to obtain these improvements.
Besides the analysis, a concurring result of the research was additionally the process of making a system that uses the discussed methods and is integrated with Lawciter which is an E-discovery system. The system conferred all documents of law cases. If the user testing , confirms quality and usability of the add-ons, it will come the finishing deliverance.

References

[1] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning, "Kea: Practical automated keyphrase extraction," in *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*, pp. 129–152, IGI global, 2005.

[2] P. D. Turney, "Learning to extract keyphrases from text. national research council," *Institute for Information Technology, technical report ERB-1057*, 1999.

[3] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pp. 216–223, 2003.

[4] K. Zhang, H. Xu, J. Tang, and J. Li, "Keyword extraction using support vector machine," in *international conference on web-age information management*, pp. 85–96, Springer, 2006.

[5] K. Barker and N. Cornacchia, "Using noun phrase heads to extract document keyphrases," in *conference of the canadian society for computational studies of intelligence*, pp. 40–52, Springer, 2000.

[6] D. B. Bracewell, F. Ren, and S. Kuriowa, "Multilingual single document keyword extraction for information retrieval," in *2005 International Conference on Natural Language Processing and Knowledge Engineering*, pp. 517–522, IEEE, 2005.

[7] S. Al_Janabi, M. A. Salman, and M. Mohammad, "Multi-level network construction based on intelligent big data analysis," in *International Conference on Big Data and Smart Digital Environment*, pp. 102–118, Springer, 2018.

[8] A. Bougouin, F. Boudin, and B. Daille, "Topicrank: Graph-based topic ranking for keyphrase extraction," in *International joint conference on natural language processing (IJCNLP)*, pp. 543–551, 2013.

[9] M. Grineva, M. Grinev, and D. Lizorkin, "Extracting key terms from noisy and multitheme documents," in *Proceedings of the 18th international conference on World wide web*, pp. 661–670, 2009.

[10] W. D. Abilhoa and L. N. De Castro, "A keyword extraction method from twitter messages represented as graphs," *Applied Mathematics and Computation*, vol. 240, pp. 308–325, 2014.

[11] H. C. Black, B. A. Garner, B. R. McDaniel, D. W. Schultz, and W. P. Company, *Black's law dictionary*, vol. 196. West Group St. Paul, MN, 1999.

[12] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, pp. 404–411, 2004.