Detection of Catchphrases and Precedence in Legal Documents

Yogesh H. Kulkarni
Consultant, RightSteps Consultancy, Pune, India.
yogeshkulkarni@yahoo.com

Rishabh Patil
Engineer, RightSteps Consultancy, Pune, India.
rishabh@rightstepsconsultancy.com

Srinivasan Shridharan
Founder, RightSteps Consultancy, Pune, India.
srini@rightstepsconsultancy.com

ABSTRACT

"Common Law System" practiced in India refers to statute as well as precedent to form judgments. As number of cases are increasing rapidly, automation becomes highly desirable. This paper presents two such systems viz. Automatic Catchphrase Detection and Automatic Precedence Detection.

Automatic Catchphrase Detection: One of the key requirements of such information retrieval system is to pre-populate database of prior cases with catchphrases for better indexing and faster, relevant retrieval. This paper proposes an automatic catchphrases prediction for cases for the same. The problem catchphrase detection has been modeled as "custom named entity recognition (NER) using conditional random fields (CRF)". CRF is trained with pairs of prior cases and their respective catchphrases, the gold standards. The model is, then used to predict catch-phases of unseen legal texts. End of the first section demonstrates efficacy of the proposed system using practical data-set.

Automatic Precedence Detection: Due to thousands of past cases it becomes tedious and error-prone to find relevant precedent, manually. An automatic precedent retrieval system is the need of the hour. One of the key requirements of such information system is to find cases which could be "similar" to the case in hand. The "similarity" used in this paper is about citations. The problem is of predicting prior cases which could potentially be cited by a particular case text. This paper proposes such association system using mixed approaches. It employs rule-based Regular Expressions based on references to statute and Articles. It finds cosine similarity between cases using vectors generated by popular word embedding called doc2vec. It also leverages topic modeling by finding matches between cases based on the number of common topic words. End of the second section demonstrates efficacy of the proposed system by generating cite-able documents from test data-set.

CCS CONCEPTS

• **Information systems → Probabilistic retrieval models**; **Document topic models**; • **Computing methodologies → Information extraction**;

KEYWORDS

Information Retrieval, Conditional Random Fields, Named Entity Recognition, Regular Expressions, Word Embedding, Topic Modeling, Legal, Word2Vec, Legal, Text Mining, Natural Language Processing.

1 INTRODUCTION

Indian judicial system, like many in the other parts of the world, is based on whats called "Common Law System" in which both, written law (called "statutes") and prior cases (called "precedent") are given equal importance while forming the judgment. Such system brings uniformity of the legal decisions across similar situations.

Court cases, judgments, legal texts are typically long and unstructured, making it hard to query relevant information from them, unless someone goes through them manually and vigilantly. Looking at the volume of legal text to be processed, it is desirable to have automatic system that detect key concepts, catchphrases in the legal texts.

With number of cases increasing day by day, it has become humanly impossible to search relevant past cases for a particular topic. Automatic Precedent Retrieval System (APRS) is the need of the hour. As more and more cases are coming in the digital form, text mining has found immense importance for developing APRS.

This paper is divided into two sections. The first dealing with the task of Automatic Catchphrase Detection and the second one of Automatic Precedence Detection.

2 AUTOMATIC CATCHPHRASE DETECTION

The aim of this section is to propose automatic catchphrase detection-prediction system for legal text. It uses training data comprising of pairs of text and respective catchphrases, the gold standard, prepared manually by legal experts. The proposed system builds probabilistic model based on this training data, which, in turn, can predict the catchphrases in the unseen legal texts.

The contributions made in this system are as follows:

(1) A novel method to prepare training data needed for CRF.
(2) Feature engineering for better results with CRF

The section has been structured as follows: in the following section 3.1 the catchphrases detection task has been described in details, as definition of the problem. In section 3.2, structure of the training data has been explained. Next, the proposed system is elaborated in Section 3.3. It describes preparation of CRF training data-set and feature engineering adapted for this custom named entity recognition (NER) methodology. Section 4 discusses the findings drawn from this work.

2.1 Task Definition

Catchphrases are short phrases from within the text of the document. Catchphrases can be extracted by selecting certain portions from the text of the document[2]. The data-set provided consists of legal texts and their respective catchphrases, along with test documents for which the catchphrase needs to be extracted-predicted.

2.2 Data-set

Fire-2017 [2] dataset contains following directories:

(1) Train_docs : contains 100 case statements, $case\_ < i > \_statement.txt$ where $i = 0 \rightarrow 99$. Sample document looks like "R.P. Sethi, J. 1. Aggrieved by the determination of annual . . . ultimate result.".
(2) Train_catches: contains the gold standard catchwords for each of these 100 case statements, $case\_ < i > \_catchwords.txt$ where $i = 0 \rightarrow 99$. Sample document looks like "Absence, Access, Accident, Account, . . . Vehicle, Vehicles".
(3) Test_docs: contains 300 test case statements, similar to Train_docs, $case\_ < i > \_statement.txt$ where $i = 100 \rightarrow 399$.

2.3 System Description

2.3.1 Preprocessing. Each of the training statements were tokenized into a list. Their Parts-of-Speech (POS) tags were generated using python nltk [1] library. Another sequence of custom NER tagging was made by referring to token list and given catchphrases. B-LEGAL and I-LEGAL tags were employed for Begin and Intermediate of the catchphrases respectively and O for other tokens. So training data file looked like:

in	IN	O
the	DT	O
year	NN	O
1987	CD	O
and	CC	O
that	IN	O
property	NN	B-LEGAL
had	VBD	O
extensive	JJ	O
national	JJ	B-LEGAL
highway	NN	I-LEGAL
frontage	NN	O

Table 1: Training data with primary features

There are 3 columns for each token.

(1) The word itself (e.g. property);
(2) POS associated with the word (e.g. NN);
(3) Custom NER tag (e.g. B-LEGAL);

Each test statement was also tokenized into a list and its POS tags were generated using nltk [1], so testing data file looked like:

appeals	NN
the	NNS
high	DT
court	JJ
accepted	NN
the	VBD
view	DT

Table 2: Testing data with primary feature

There are 2 columns for each token.

(1) The word itself (e.g. appeals);
(2) POS associated with the word (e.g. NN);

2.3.2 Modeling. The problem of detecting catchphrases was modeled as customized NER. POS and custom NER tagging performed during pre-processing stage were used to form secondary features. These were used in building CRF model. CRF++ [5] toolkit was used. Salient secondary features developed were:

(1) Unigrams:
  (a) Previous 3 tokens, current token and next 3 tokens
  (b) Previous 3 POS tags, current POS tag and next 3 POS tags
(2) Bigram tokens

CRF model was generated using:

crf_learn template_file train_file model_file

The generated model file was then used to predict from test data:

crf_test $-v1 -m$ *model_file test_files*

With $-v1$ option the highest probability is shown as:

Rockwell	NNP	B	B/0.992465
International	NNP	I	I/0.979089
Corp.	NNP	I	I/0.954883

Table 3: Sample results with probabilities

2.3.3 Results. The CRF++ model was used to predict custom NER tags from the given testing data as:

case_102_statement	notification:0.990733,tax:0.7341635
case_103_statement	prevention of corruption:0.9988746666666667
case_104_statement	natural justice:0.7491485,appeal:0.708494
case_105_statement	seniority:0.997623,legislation:0.994512,appointing

Table 4: Submission file

There far less catchphrase words compared to total number of words in the documents. Thus, accuracy is not a good metrics to measure the performance of this prediction. "Precision" and "Recall" values on the testing data-set came out to be as follows:

Mean Average Precision	Overall Recall
0.47923074	0.2476876075

Table 5: Results conveyed by FIRE[2]

3 AUTOMATIC PRECEDENCE DETECTION

Automatic Precedent Retrieval System (APRS) is desirable in such situation. One of the key functionality necessary in APRS is to find "similar" cases so that they can be cited or referred from the case being built. The notion of "similarity" has various connotations. In the context of the given problem, it is said to be the documents which share citations. In other words, the task is to find such prior cases which are potentially cite-able from the case in hand.

Legal texts are typically lengthy and unstructured in nature. It is challenging to find similarity score among two texts by just counting words or their frequency distributions or such preliminary statistical measures. Need to embed higher level constructs such as

word embedding to introduce semantic similarity as well as higher level clusters given by Topic Modeling.

The aim of this section is to propose automatic cite-able texts detection-prediction system for legal texts. It is unsupervised (no labeled training data) technique comprising of Regular Expressions, Word2Vec and Topic Modeling. All being employed to give similarity score based on different aspects of the texts. Final rank is arrived at using weighted sum of the individual scores.

The contributions made in this system are as follows:

(1) Detecting statute and Articles based on Regular Expressions.
(2) Proposing cosine similarity between texts based on vector generated by Word Embedding (word2vec/doc2vec).
(3) Proposing Document-Topics-Words distribution for all texts and then scoring similarity based on common topic-words.

The section has been structured as follows: in the following section 3.1 the precedence detection task has been described in details, as definition of the problem. In section 3.2, structure of the training data has been explained. Next, the proposed system is elaborated in section 3.3. It describes ranking method based on weighted-sum of scores from individual methods. Section 4 discusses the findings drawn from this work.

3.1 Task Definition

Legal cases typically cite statute, Articles and previous cases relevant to them. Thus, it is necessary to form association or similarity between documents based on citations, so that it can be leveraged for Precedent retrieval. Given sets of current and prior cases, the task is: *For each document in the first set, the participants are to form a list of documents from the second set in a way that the cited prior cases are ranked higher than the other (not cited) documents.*[2]

3.2 Data-set

Fire-2017 [2] data-set contains following directories:

(1) Current cases: A set of cases for which the prior cases have to be retrieved, $current\_case_{<}i > .txt$ where $i = 0001 \rightarrow 0200$. Sample document looks like "**Judgment** IN THE SUPREME COURT OF INDIA...".
(2) Prior cases: contains prior cases that were actually cited in the case decision along with other (not cited) documents, $prior\_case\_ < i > .txt$ where $i = 0001 \rightarrow 2000$. Sample document looks like " 551; AIR 1996 SC 463; 1995 (6) SCC 315; 1995 (7) JT 225; 1995 (5) SCALE 690 (11 October 1995) JEEVAN REDDY, B.P. (J) JEEVAN REDDY, ...the assessee. No costs".

3.3 System Description

The proposed APRS solution uses the following 3 distinct approaches to determine similarity. The final similarity score is computed as a weighed average of the scores generated by these 3 approaches.

3.3.1 Regular Expressions based. Cases refer to statue in the form of legal Articles, such as Article 270, 370, etc. Such references can be extracted using Regular Expressions. In this system, patterns like r' article (\d+)' are used for both, current as well as prior cases. For a current case, all the prior cases are collected which have same Articles.

3.3.2 Topic Modeling based. Basic premise of this approach is that if most of the topics extracted from documents match then they are similar or cite-able. In this system, Document-Topics-Words distribution is generated by Latent Dirichlet Allocation (LDA) algorithm in gensim library[4]. Score of similarity is calculated based on ratio of matching topic-words to the total.

3.3.3 Doc2Vec Similarity based. Word2vec has emerged one of the most popular vectorization based on semantic similarity [3]. Process to generate document vectors (based on word2vec) was:

(1) Got every case as cleaned text, split it to form list of words/tokens, for both, current and prior cases.
(2) Created gensim TaggedDocument for each case text, giving filename as tag.
(3) A Map of tag to the content i.e. word-list for each cases were generated and saved for reuse.
(4) LDA model was built and saved. It was used to generate document vectors for both current and prior cases.

A similarity matrix was generated where current cases are rows and prior cases as columns with values as cosine similarity between document vectors of the current-prior case pair (row-column). The values act as score for this particular approach.

3.3.4 Results. Each current-prior case pair has a final score based on weighted sum of scores from individual approaches mentioned above. Due to lack of labeled training data, the weights were decided heuristically. The results were presented as sorted list of prior case for each current case, and looked as follows:

Current case	Prior case	Rank	Score
current_case_0001	prior_case_0780	0	1.783
current_case_0001	prior_case_1256	1	1.743
current_case_0001	prior_case_0838	2	1.727
...			
current_case_0116	prior_case_1533	0	1.003
current_case_0116	prior_case_0411	1	0.929
current_case_0116	prior_case_1600	2	0.877
...			

Table 6: Submission file

Results on the test cases were evaluated by Fire-2017[2] and conveyed as below:

Mean Average Precision	Mean Reciprocal Rank
0.2017	0.45

Table 7: Results conveyed by FIRE[2]

4 CONCLUSIONS

This paper has been divided into two parts, elaborating two different tasks for legal documents.

The first was of Catchphrase detection which started with a brief overview of Automatic Catchphrases Prediction System to extract catchphrases from legal texts. Accuracy was impacted as some of the training samples had very few catchphrases. Various approaches/tool-kits were tried but it was found that the problem of catchphrase detection needs to be modeled as sequential probabilistic labeling problem rather than a simple linear classification

problem. CRF algorithm was chosen with primary features as POS and custom NER tags and numerous secondary features representing the context. As a future work, if sufficient gold standard data is available, one can explore more sophisticated techniques such as Long Short Term Memory networks (LSTM), where custom features need not be provided but get generated internally.

In the second part, a brief overview of Automatic Citation Prediction System was presented to discover cite-able prior cases. It was found that the problem of citation detection needs to be modeled as a mixed approach, employing rule based, machine learning and deep learning based approaches rather than a simple cosine similarity of tf-idf (term frequency inverse document frequency) approach. Weighted sum of scores by individual approaches was done. A threshold cut-off was decided to prune out irrelevant cite-able prior cases. Current cases and their predicted cite-able prior cases were presented along with corresponding ranking scores.

VITAE

Yogesh H. Kulkarni works as Data Science Consultant and Trainer. Profile: https://www.linkedin.com/in/yogeshkulkarni/
Rishabh Patil works as Data Engineer. His profile is at https://www.linkedin.com/in/rishabh-patil-256a25124/.
Srinivasan Shridharan is Data Scientist and entrepreneur. Profile: https://www.linkedin.com/in/srinivasan-shridharan-08a86a6/.

ACKNOWLEDGMENTS

Wish to thank Ankur Parikh, a keen researcher of Deep Learning and NLP, for discussions.

REFERENCES

[1] Edward Loper Bird, Steven and Ewan Klein. 2009. Natural Language Toolkit. (Sep 2009). http://www.nltk.org/
[2] IRSI. 2017. Forum for Information Retrieval Evaluation. Information Retrival Society of India. (Dec 2017). https://sites.google.com/view/fire2017irled/track-description
[3] Chris McCormick. 2016. Word2Vec Tutorial - The Skip-Gram Model. (Apr 2016). http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
[4] Radim Rehurek. 2017. Gensim: Topic Modeling for humans. (Sep 2017). https://radimrehurek.com/gensim/
[5] Taku. 2017. CRF++: Yet Another CRF toolkit. (Sep 2017). https://taku910.github.io/crfpp/