An Extraction based approach to Keyword generation and Precedence Retrieval: BITS, Pilani - Hyderabad

G. V. Sandeep
Student, Department of Computer Science and Information Systems, BITS Hyderabad
Telangana, India
gvsandeep2647@gmail.com

Shikhar Bharadwaj
Student, Department of Computer Science and Information Systems, BITS Hyderabad
Telangana, India
shikhar.coder@gmail.com

ABSTRACT

Precedence Retrieval is an information retrieval task that involves ranking the given set of documents according to their relevance to a query document. It is used for finding prior cases in common law system. A prior or precedent case discussing the same issue can be used as a reference in the current case. With the increase in the digitalisation of legal documents it is imperative to develop systems for efficient precedence retrieval. This paper proposes a method for the same based on Keyword extraction and Nearest Neighbor algorithms. Keywords can be used to summarise a document. We have extracted the keywords for each document to be used in Precedence Retrieval using TF-IDF and other relevance scores. The keywords are then used to rank the documents.The dataset for experimentation was obtained from FIRE 2017 IRLeD track[6]. The results of keyword extraction task have been expressed in Precision@10, Precision@100, mean precision and mean recall metrics. For the second task of Precedence Retrieval Precision@10, Recall@10, Mean Average Precision and Mean reciprocal rank have been used to depict results.

KEYWORDS

Information Retrieval,Nearest Neighbor,Vector Space Model, Document Vector, POS Tagging, Keyword extraction, Precedence Retrieval, Legal Documents Retrieval System, Document Similarity

1 INTRODUCTION

Associating a document with a set of keywords can prove to be extremely useful in various domains. Especially in the domain of law, where the documents of previous similar cases are often used as references, tagging them with a set of keywords is essential. Keywords give a very high - level description of a document. The reader can save himself a lot of time by quickly evaluating the relevance of the document to him by having a look at these keywords. They play a crucial role in reducing the search space and also act as a tool to find similarity between two documents with drastically low cost. There are two ways to handle the task of generating keywords for a given document. One approach is to agree to a set of exhaustive keywords apriori and classify each document to one or more of these keywords. This approach is also known as text classification. Another approach is to find out words within the document which can represent the whole document.

In this paper, we accomplish two tasks:

(1) Generate keywords for a given document using extractive methods.
(2) Given a case document, find documents of similar cases using nearest neighbour approach.

The rest of the paper is organized as follows: In Section - 2 we first describe the tasks in detail which includes the assumptions, methodology (motivated by [2]) and limitations. We present our results in Section - 3 followed by possible enhancements in Section - 4. Finally, we conclude in Section - 5

2 TASK DESCRIPTION

The FIRE 2017-IRLed track, motivated by the need for an efficient legal document retrieval system, had the following subtasks :

2.1 Subtask - 1

2.1.1 Description. **Catchphrase Extraction** Given a set of documents, extraction of catchphrases that describe the content of the document. The data provided for the task consists of 100 documents and their corresponding gold standard catchphrases for training. The catchphrases were obtained from Manupatra which employs legal experts to annotate case documents with catchphrases. The test set consists of 300 separate documents whose catchphrases were to be found.

2.1.2 Assumptions. In the generation of the catchphrases, we assume that all catchphrase are single word. We have chosen an extractive method for catchphrase generation rather than an abstractive one. The method works under the assumptions that a word that is more frequent has a higher possibility of being a keyword, some parts of speech have higher a probability of being keywords and keywords tend to appear together in sentences. This is the central idea of our algorithm.

2.1.3 Methodology. The task is to summarise a document in the form of keywords in an extractive manner. This is done in two stages. First, we shorten the sentences of each document by throwing out unimportant words. Next, we work with shortened sentences to extract keywords.

For determining whether a word is important enough to retain in a sentence we have computed a metric for each word of the document. If this metric exceeds the threshold we retain the word for further processing. This metric is a linear combination of term frequency of the word and its part of speech(POS) tag weight. To compute the POS tag weight, we performed an initial analysis on the provided golden catchphrases and found the frequencies of all tags. The POS score for a word is then the normalized frequency of the POS the word is tagged as. More formally,

$word\_weight = t_f + \alpha * (POS\ Score)$

If this word_weight exceeds the threshold for the word we retain the word to shorten the document. The threshold is calculated for each word in the vocabulary(dictated by dataset corpora) separately. The threshold is the frequency which gives a minimum error on the already available catchphrase and document pair.

From the remaining words in the sentences, we count all frequencies in a given document and find the top k most frequent words. These are the *popular words* for the document. Then we construct a multiset S of all sentences that contain at least one popular word. Next, we consider all words of all sentences in S and remove those that appear only once. From this filtered set of candidate words, we keep only unique words.This forms our set of final keywords. Then we compute the score for each word by summing up the TF-IDF score, word frequency and the number of occurrences in S. This is the importance score of the word which determines how relevant the word is when describing the contents of the document. A sorted list of these words with normalized score is produced as output.

2.1.4 Limitations. The limitations of our model arise from the very assumptions that make the task simpler, namely:

(1) Only one-word keywords can be extracted from the document. The model ignores the fact that the catchphrase can contain more than one word that describes the document.
(2) Sometimes a word that occurs not so frequently in the document can be a keyword. Our model will definitely overlook this fact. Conversely, our model will predict that certain word is a keyword just because it occurs many times in the text. These are the corner cases we chose to ignore to keep our method simple.

2.2 Subtask - 2

2.2.1 Description. **Precedence Retrieval** Given two sets of documents A and B, rank those in A according to their relevance for each case in B. The data for precedence retrieval consists of: Query_docs - current cases, formed by removing the links to the prior cases and Object_docs - the prior cases which have been cited by the cases in Query_docs (links to which are removed from the Query_docs) ăalong with someărandom cases (not among Query_docs). There are 200 documents in the Query_docs and more than 2000 in Object_docs. For each case in the Query_docs, the task was to rank all the 2000+ cases in the Object_docs such that all the actually cited prior cases are ranked higher than other documents. The task is made challenging by adding cases in Object_docs that are not related to any Query_doc.

2.2.2 Assumptions. The approach presented here uses a slight variation of the Bag of words representation of a document to find the similarity among them. Thus, it assumes that similar cases will have similar word usage in their body. Thus, higher correlation between the words used and case's context would lead to better results.

2.2.3 Methodology. In the preprocessing phase, each case document from the `Current_Cases` folder was read line by line. Every line was then tokenized into a list of words using nltk packages' word tokenizer [1]. From this list, all those words which belong to the set of stop words defined for the English language as per nltk are removed. Also, all those words whose lengths are less than or equal to four were also removed. For all the remaining words their frequency in that particular document was recorded. This data was stored in a global dictionary which had its key as a word and value as a list of frequencies of that word in each document of the corpus. This list was then condensed to remove all zero entries.

After this, each word is given a weight which combines its IDF score with that of the POS Score. The POS Score is estimated on the basis of keywords given in Task - 1. POS tags of all the words were noted and based on the counts, the chances of a word being a keyword given its POS tag was estimated. For Eg.: if there are two keywords with POS tag[2] as *'VBG'* and three keywords with POS tag as *'JJ'* then given a word with POS tag of *'JJ'*, it has 0.6 probability of being a keyword. This was to ensure that proper nouns or other words which tend to have a higher IDF Score would be scaled properly.

$$word = IDF\ Score * POS\ Score$$
$$IDF\ Score = (1 + \log(\frac{N_d}{df}))$$

N_d = Number of documents
df = Number of documents in which the word occurs

Once each word had been associated with a weight, they were sorted in decreasing order of their weights and only the top 5000 words were retained. This effectively meant that each document would now be represented in this 5000 dimension space as a vector. Suppose the first element of the vector was associated with the word **abducts** which had a weight of 0.7. Now if in a given document, the word **abducts** occurred 4 times, then the vector representing the document would have its first element as $4 * 0.7 = 2.8$. In a similar manner, the entire vector was created for each document.

Similarly, all cases in `Prior_Cases` were also represented in this 5000-dimensional space. Each case in `Current_Cases` was then compared with each case in `Prior_Cases`. The similarity score was given by the dot product of the two vectors representing the cases and they were reported in the decreasing order of their similarity.

2.2.4 Limitations. The following are the limitations of the model:

(1) This model may not give a high similarity score to documents which use different words to convey the same meaning. Suppose a document extensively uses the word **abducts** and another document uses the word **kidnaps** instead. They may or may not be given a high similarity score although they cater to a similar category of cases.
(2) Other than the fact that the documents have similar word usage no other explanation can be given to the results obtained. In other words, results are not explainable.
(3) This approach can be very time-consuming. For each query document, it has to find its similarity with all the existing cases in the 5000-dimensional space which itself might change with the addition of new documents to the corpus.

[1] http://www.nltk.org/api/nltk.tokenize.html
[2] http://www.nltk.org/book/ch05.html

3 RESULTS

Our algorithm was run on the dataset provided by FIRE 2017-IRLeD track. The results produced by our model were compared with those tagged manually by legal experts and precision and recall values were calculated. For the second task, mean reciprocal rank was also calculated. The results are as follows:

Table 1: Subtask - 1 Results

Methods/ Evaluation Metrics	Mean R precision	Mean Precision at 10	Mean Recall at 100	Mean Average Precision	Overall Recall
bphc_with POS_1	0.065781 20144	0.102	0.13650 79757	0.16067 55473	0.16548 09155

For Subtask-1, table 1 shows the results. We were ranked fifth with a Mean Average Precision(MAP) of 16.1% and an overall recall of 16.5%. When compared with other systems submitted in the competition, it can be noticed that our system provided almost equal weight to precision and recall.

Table 2: Subtask - 2 Results

Team_ID/ Methods	Mean Average Precision	Mean Reciprocal Rank	Precision at 10	Recall at 10
bphcTASK2 IRLeD	0.0711	0.1975	0.06	0.28

For subtask-2, our system's results are depicted in table 2. We had the MAP of 7.11% and a mean reciprocal rank of 19.75%. For a system as basic as ours, it is a surprisingly good result.

4 POSSIBLE ENHANCEMENTS

The models proposed for both the tasks rely highly on the importance of individual words. Thus, one straightforward enhancement would be to generate keyphrases and extend it to extract catchphrases. One of the possible enhancement could be to exploit the semantic similarity between words as suggested in [3]. In this paper, the authors have used features of WordNet along with the standard preprocessing of stop words removal and stemming to group semantically similar words and replace them with the most commonly occurring term of that group. The basic idea is to find similarity on the basis of context as against to word usage. They also took help of the Michael Lesk Algorithm [4] to disambiguate words in the sentence context.

As our model takes advantage of the fact that words with a certain POS tags are more likely to be keywords, better POS Tagging techniques will obviously help the model achieve better results. As suggested in [8], POS tagging which considers the tags of the surrounding words as well as joint features of current word and surrounding words outperforms the traditional taggers. They have achieved better results in tagging by using this idea along with lexicalization and smoothing.

Another promising idea has been presented in [1] where the authors have used Naive Bayes Classifier on corpus pertaining to a single domain. They have used features like $TF\ XIDF\ Score$ and Distance (of the catchphrase from the beginning of the sentence) to build their model. The Kea catchphrase generation algorithm was used to generate set of candidate catchphrases.

Since the main purpose of keyphrases or keywords is to give a high-level description of the document, techniques which build a lattice of concepts for a given document will enhance results as well as explainability. This phenomenon has been used as a backbone in both [9]. Similarly, semantic chains *BioChain* has been used on the domain-specific corpus in [7]. Along with this they also proposed *FreqDist* - a frequency distribution based approach and a hybrid method which combines both *BioChain* and *FreqDist*.

All these methods have shown improvements on their traditional counterparts on the basis of ROGUE score. Details and formulation of ROGUE scores can be found in [5]

5 CONCLUSION

Keywords are a significant help to lawyers for determining precedents for a case. In this paper, we proposed a very simple keyword extraction algorithm and a precedence retrieval algorithm that uses our keyword extraction algorithm to work. The keyword extraction algorithm utilizes both the frequency of words and the POS tag of the word to determine its importance. The precedence retrieval algorithm builds a vector for each document and computes the similarities accordingly. Despite the simplicity of our model we achieved surprisingly good results.

6 ACKNOWLEDGEMENTS

We would like to express sincere thanks to our mentor Dr. Aruna Malapati[3], Department of Computer Science and Information Systems, BITS, Pilani - Hyderabad Campus for her constant guidance and also for the help in writing this paper.

REFERENCES

[1] Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *16th International Joint Conference on Artificial Intelligence (IJCAI 99)*, Vol. 2. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 668–673.

[2] Filippo Galgani, Paul Compton, and Achim Hoffmann. 2012. Towards automatic generation of catchphrases for legal case reports. *Computational Linguistics and Intelligent Text Processing* (2012), 414–425.

[3] Mohamed H Haggag. 2013. Keyword extraction using semantic analysis. *International Journal of Computer Applications* 61, 1 (2013).

[4] Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*. ACM, 24–26.

[5] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, Vol. 8. Barcelona, Spain.

[6] Arpan Mandal, Kripabandhu Ghosh, Arnab Bhattacharya, Arindam Pal, and Saptarshi Ghosh. 2017. Overview of the FIRE 2017 track: Information Retrieval from Legal Documents (IRLeD). In *Working notes of FIRE 2017 - Forum for Information Retrieval Evaluation (CEUR Workshop Proceedings)*. CEUR-WS.org.

[7] Lawrence H Reeve, Hyoil Han, and Ari D Brooks. 2007. The use of domain-specific concepts in biomedical text summarization. *Information Processing & Management* 43, 6 (2007), 1765–1776.

[8] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association*

[3] http://universe.bits-pilani.ac.in/hyderabad/arunamalapati/Profile

for Computational Linguistics on Human Language Technology-Volume 1. Association for Computational Linguistics, 173–180.

[9] Shiren Ye, Tat-Seng Chua, Min-Yen Kan, and Long Qiu. 2007. Document concept lattice for text understanding and summarization. *Information Processing & Management* 43, 6 (2007), 1643–1662.