www.ijecs.in
International Journal Of Engineering And Computer Science
Volume 9 Issue 05 May 2020, Page No. 25039-25046
ISSN: 2319-7242 DOI: 10.18535/ijecs/v9i05.4488

Legal Document Summarization Using Nlp and Ml Techniques

[1]Rahul C Kore[*], [2]Prachi Ray, [3]Priyanka Lade, [4]Amit Nerurkar

[1,2,3,4]Vidyalankar Institute of Technology, Mumbai University, Mumbai, India

Abstract:
Reading legal documents are tedious and sometimes it requires domain knowledge related to that document. It is hard to read the full legal document without missing the key important sentences. With 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. The purpose of this study is to understand a large legal document within a short duration of time. Summarization gives flexibility and convenience to the reader. Using vector representation of words, text ranking algorithms, similarity techniques, this study gives a way to produce the highest ranked sentences. Summarization produces the result in such a way that it covers the most vital information of the document in a concise manner. The paper proposes how the different natural language processing concepts can be used to produce the desired result and give readers the relief from going through the whole complex document. This study definitively presents the steps that are required to achieve the aim and elaborates all the algorithms used at each and every step in the process.

Keywords: Natural Language Processing, Word Embeddings, Page Rank Algorithm, Text Rank Algorithm.

1. Introduction

Document Summarization is one of those applications of natural language processing which is definitely going to have a great impact on everyone's lives. Nowadays who has the time to go through the entire document and understand the purpose of the same. The human summarization is the process of taking a document, understanding it, interpreting it and finally generating a new document as a summary, but this can be a time consuming process. So comes the concept of automatic text summarization. Automatic Summarization is the process of shortening a large document computationally to create a summary that represents the most important and relevant information within the original content or document. There are two general approaches to automatic summarization which are extractive summarization and abstractive summarization.

In extractive summarization, the sentences are extracted from the original document but the extracted sentences are not modified in any way. Abstractive summarization constructs an internal semantic representation for the original sentences and then use this particular representation to obtain a summary that is closer to how a human being might express. Abstractive summarization is computationally much more complex and challenging than extractive summarization, It requires both natural language processing and a deep understanding of the domain of the original document. Most of the existing methods uses statistical methods such as frequency of occurrence, inverse document frequency or linguistic information such as term distribution, sentence position to extract the most relevant sentences from the document.

However these methods ignore the relationship between different granularity information such as the relationships between the sentences. Hence the proposed system takes into consideration the similarities between different sentences before calculating the ranks of individual sentences in the document. Many researches are ongoing in the field of document summarization because text summarization becomes different and unique problem for each domain of research.

2. Background and Related Work

This part of the paper illustrates different work carried out by others in areas which are relevant to our research. The sub-parts below are the most important key areas in our study.

2.1 Semantic Similarity Measures

There are various applications of text similarity measures [2] which includes automatic text summarization, relevance feedback classification, automatic evaluation of machine translation and determining text coherence. There are various approaches which are used to calculate the similarity measure which are based on statistical methods, vector representation of words in the given document, string or corpus based approach and hybrid similarity measures. Some applications like TF-IDF uses inverse document frequency for calculating the frequencies of terms where it does not take into consideration the surrounding context amongst that term in the text. The mapping in such method is simply done using count and probabilistic measures.

Our study proposes a method in which we first convert all the words present inside the document into it's equivalent vector from by using an appropriate word embedding model which takes into consideration the semantic value of each word, where each word is located in an n-dimensional (n=10 or 20) space. All the words with similar semantics are placed closer to each other in this space. After converting the words in the vector form we use vector similarity measure to calculate the likeness. In this way, we also consider the semantic of all the terms present in the document which gives a better result.

2.2 Keyword Extraction

The smallest unit to express the core meaning of a document is known as a keyword [3]. By extracting several keywords from a document to summarize document theme content, helps users to quickly understand whether the document is of their interest or not. Keyword extraction can be classified in two categories which are supervised and unsupervised. In supervised we have two different methods which are Two classification problem and Multiple classification problem. In unsupervised we have three different methods which are word frequency, Model based and Graph methods.

Supervised methods are those which requires human intervention whereas Unsupervised methods are without human intervention which extracts keywords directly through the information of the text which in turn improves the efficiency greatly. In unsupervised methods, the word graph model treats the document as a network composed of words which is based on the theory of PageRank link analysis to calculate the importance of words.

Similarity and co-occurrence frequency between words are used as the weight for extracting keywords and Word2vec is used to calculate the close degree between the words. Word2vec makes use of deep learning to map each word into a vector of k-dimension.

2.3 Graph Based Ranking Algorithm

Using graph based ranking algorithm [1], we can find the importance of a vertex present within the graph based on the information which are drawn from the graph structure. In this section, a graph based algorithm-HITS which were previously found to be useful on a large range of documents is presented. This graph based ranking algorithm (HITS) can be used for unidirectional or weighted graphs.

HITS(Hyperlink-Induced Topic Search) [1] is a link analysis algorithm that ranks web nodes/pages. It estimates two types of values for a page which are hubs and authorities. Authority estimates the value of content of the page whereas hub estimates the value of its links to the other pages. HITS is an iterative algorithm based on linkage of documents like PageRank. The algorithm performs a series of iterations which consists of the following two steps:

- Authority update
- Hub update

The above two score for a node is calculated using the following algorithm:

- Start
- Initialize hub score and authority score to 1 for each node.
- Update authority score.
- Update hub score.

- Normalize hub score by dividing each hub score by square root of the sum of squares of all the hub scores.
- Normalize authority score by dividing each authority score by square root of the sum of squares of all the authority scores.
- Goto step 3 and repeat if necessary.

Formula for calculating authority and hub score:

$$HITSA(Vi) = \sum_{Vj \in In(Vi)} HITSH(Vj)$$

$$HITSH(Vi) = \sum_{Vj \in Out(Vi)} HITSA(Vj)$$

3. Proposed System

As stated earlier, the proposed system would be focusing on generating extractive summary from the given document using different natural language processing techniques such as word embedding, similarity measures, and ranking algorithm.

So before getting into specifications of the proposed system, let us understand the overall flow of the system which is given below in the figure.

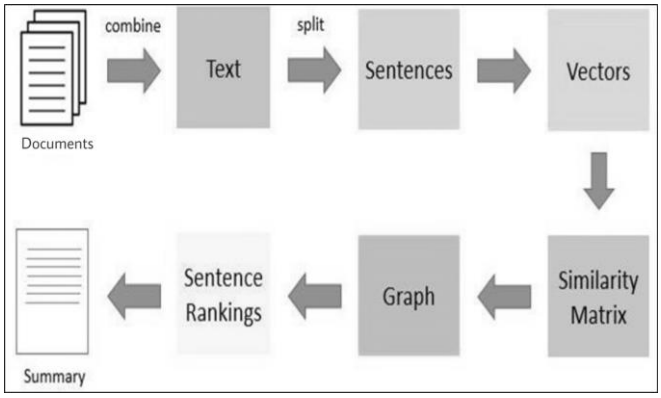

Figure 1: Flow of the system

The overall flow of the system can be explained with the help of following steps:

1) Initially we will concatenate all the text present in the document.

2) Then we would split the text into individual sentences

This can be done using the tokenizer of the natural language tool kit package of python.

3) Then we would remove all the punctuations, numbers and special characters from all the individual sentences.

This can be achieved with the help of regular expression and python packages.

4) Then all the alphabets are converted into lower case alphabets

This is done so there would not be any problem because of character case sensitivity in the sentences.

5) Then we would remove all the stop words from the sentences because stop words do not contribute any meaningful context to the sentences and would only waste processing time in the next step of vector conversion.

After all the above steps we would get clean sentences which are free from stop words and all other unwanted punctuations, numbers and special characters.

6) Now, we will fetch vectors for the constituent words in a sentence and then take mean/average of those vectors to obtain a consolidated vector for all the sentences in the document.

The above step is done using the word embedding model known as Law2Vec, which was developed by the Department of Informatics of the University of Athens.

After the above step we would get vector representation of all the sentences which would be carry forwarded in the later steps.

7) Now, we would create an empty similarity matrix of nxn size where n is the number of sentences present in the document.

8) Now, we would calculate cosine similarity for all the sentences present in the document using the vector representation of the sentences and not the original sentences.

Right after calculating the similarities between the sentences using their vector form we would be inserting all these in the similarity matrix created in the above steps.

9) After all the above steps, now we would take this similarity matrix and apply a ranking algorithm on this similarity matrix obtained in the above step.

In our case, we would be using page rank algorithm to calculate the ranking of all the individual sentences. After ranking all the sentences now we can display top ranked sentences from the document.

4. Overview of the System

In this section, we will be focusing more on all the concepts such as tokenization, stop words elimination, word embedding, similarity measure, ranking algorithm which are used in the above proposed system.

4.1 Dataset

Dataset was downloaded from UCI-Machine Learning Repository [10]. The downloaded dataset contains Australian Legal Cases from the Federal Court of Australia (FCA). It contains almost 4000 Legal cases.

4.2 Tokenization

Tokenization in a defined document unit is basically the task of chopping up the sentences into pieces, called tokens, perhaps it also means at the same time throwing away certain characters, such as punctuation.

One can think of tokens as words as a token in sentence, and sentences as tokens in a paragraph.

Below is an example of tokenization:

Input:

Friends	**Romans**	**Countrymen**	**lend me your ears;**

Output:

The above token are more often referred to as terms or sometimes words.

We would be using sentence tokenizer of natural language tool kit package of python which is already trained and thus it very well knows how to mark the end and the beginning of the sentences at what characters and at what punctuations.

4.3 Stop Words Elimination

In the below section, we would be discussing on why stop word elimination is an important step in natural language processing.

One of the important forms of pre-processing in natural language processing is to filter out all the useless data present in the document. In natural language processing, such useless words that we filter out in the pre-processing step are referred to as stop words. So a stop word is a commonly used word such as “the”. “a”, “an”, “in”, etc which are always ignored by all the applications using natural language processing or search engines as a matter of fact.

We do not want stop words to waste any space inside the database or increase any processing time in our application so it is better to eliminate such words. we can stop words easily as the natural language tool kit package in python has a list of stopwords stored in 16 different languages. So we just need to download the corpus and start eliminating all the stop words from all the sentences in the document.

4.4 Word Embedding

In natural language processing, when working with text, the first thing that we must do is come up with a strategy to convert strings to numbers or to vectorize the text before feeding it to any model. There were many techniques that came before word embedding to convert strings to text or vectorize the text but no technique was as good as word embedding [9]. Two such techniques that came before word embedding are as follows:

1) One-hot encodings [9]

In this, we might “one-hot” encode each word present in the vocabulary. To represent each of these words we would create a zero vector having length equal to the vocabulary and then we can place a one in the index that will correspond to the word. This approach is inefficient because if there are 1,000 words so to one-hot code each word we would have to create vector almost all the elements are zero.

2) Encode each word with a unique number [9].

In this approach we might encode each word with a unique number. This method is also inefficient because instead of sparse vector we have a dense vector where all the elements are full. There are two downsides to this method, the first one is that the integer encoding is arbitrary and it does not capture any relationship between the words, and the second one is that integer encoding can be challenging for a model to interpret.

Hence, word embedding [9] is the most efficient one which gives us a way to use an efficient, dense

representation in which all the similar words have a similar encoding and the main part is that we do not have to specify the encoding by hand. Here an embedding is a dense vector of floating point values. And instead of specifying these values manually we can train these parameters. It is very much common to see 8-dimensional word embedding. There are word embeddings with 1024-dimensions also when we work with very large datasets. It takes more data to learn but gives very fine grained relationship between the words. The graphical representation of word embedding can be visualized as follows:

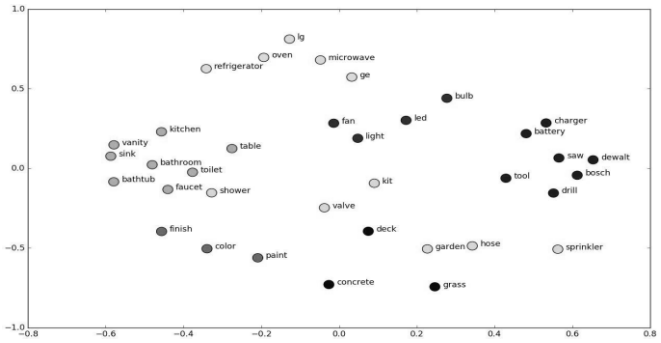

Figure 2: Graphical representation of word embedding

Thus we will be using Law2Vec [8] word embedding model which was developed by the department of Informatics of the University of Athens. This model contains millions of legal words already trained and are ready to use.\

4.5 Similarity Measure

Similarity measures [2] are used to calculate similarity between various documents, or different sentences present inside the document. It defines how much alike two objects are. It has various applications in natural language processing such as in automatic text summarization. It also has its application in computer vision. It can be used in many real world applications, one important application in the business world would be to use the similarity technique to match the resumes with the job description which saves a considerable amount of time for the job recruiters in the company. Another important application would be to use similarity measure to segment customers for marketing campaigns using some clustering algorithm which also uses similarity measures.There are many similarity metric that can be used for calculating similarity measure such as Euclidean metric, Jaccard similarity, cosine similarity, etc. We would be focusing on cosine similarity as we have used the same in our proposed system.

Cosine similarity [2] is the measure of similarity between two non zero vectors that calculates the cosine of the angle between these two non zero vectors. It is used to calculate how similar two documents are irrespective of their sizes. So we are using to measure how similar two sentences in our document are. The formula of cosine similarity can be given as follows in the image below:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 3: Cosine similarity formula

4.6 Page Rank

PageRank [1] is an algorithm which is used by google search engine to rank different web pages in their search engine results. It was named after Larry Page who is one of the founders of Google. It is way of measuring the importance of web pages and accordingly the results are shown to the users. The main component which is used in calculating the rank of the web pages are the number of links to that page, So by counting the number and quality of such links to the page the algorithm estimates how important the website is.

The following figure shows a graph of web pages A, B, C, D having certain links to each other.

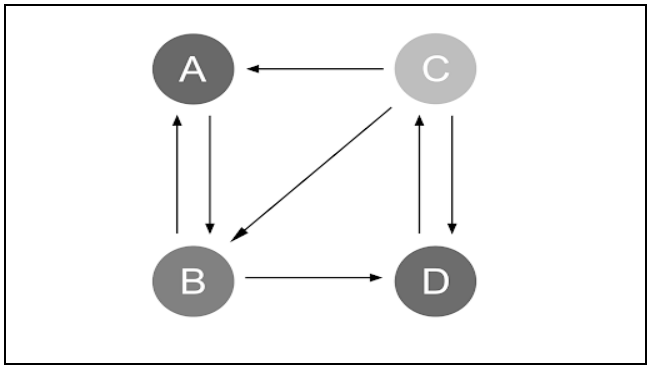

Figure 4: Graph of 4 web pages

The formula for calculating the page rank score is as given below:

Generally page rank value for any page u can be expressed as

$$PR(\upsilon) = \sum_{v \in Bu} \frac{PR(v)}{L(v)}$$

that means the PageRank value for a web page u is dependent on the page rank values for each web page v out of the set Bu (this set contains all the web pages which are linked to page u), divided by the number L(v) of links from web page v. So the page rank algorithm outputs a probability distribution which will be used to represent the likelihood that a person clicking on a web page A will arrive at another web page B. The page rank computations require several iterations to compute score for each web page.

Now that we have understood page rank algorithm, we can dive into understanding text rank algorithm [6]. There are certain similarities between these algorithms which are listed below:

1) We use sentences in place of web pages.

2) Here the similarity between two sentences is used as the web page transition probability.

3) Similarly to the page rank the similarity scores are stored in square matrix.

The text rank score of sentences can be visualized as follows:

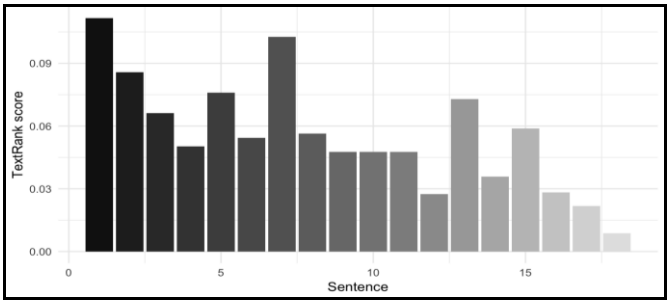

Figure 5: Text Rank score visualization

5. Conclusion and Future Scope

In this paper, we have introduced the basic idea of text rank algorithm which is based on page rank algorithm to calculate the rank of individual sentences in our document. We started by simply tokenizing the sentences and then removing various unwanted punctuations, numbers and other special characters from the sentences. Later we eliminated stop words and obtained clean sentences from the original sentences. Then we started calculating the equivalent vector representation of all the words in the cleaned sentences produced and we took the mean of all these vectors to obtain the vector representation of all the sentences. These vector representation then was used for calculating the similarity between sentences and fed into page rank algorithm which gives the score/rank of all the sentences and the top ranked sentences forms the summary of the document.

Currently the proposed system focuses only on extractive summarization and not on abstractive summarization. But the same can be extended for implementing abstractive summarization. All the sentences that ranked highest can then be used to do abstractive summarization and the sentences given to the user can be more simplified than the sentences which are given to the user now. Since the processing time and complexity of abstractive summarization is very much higher than extractive summarization, hence we were not able to dive into abstractive summarization. But with high processing power the same can be extended for abstractive summarization.

6. References

[1.] Khushboo S Thakkar, Dr. R. V. Dharaskar, M. B. Chandak, "Graph-Based Algorithms for Text Summarization", Third International Conference on Emerging Trends in Engineering and Technology, 2010.

[2.] Keet Sugathadasa, buddhi Ayesha, Nisansa de silve, Amal Shehan Perera, Vindula Jayawardjana, Dimuthu Lakmal, Madhavi Perera, "Legal Document Retrieval using Document Vector Embeddings and Deep Learning", Computing Conference -London, UK, 2018.

[3.] Yujun Wen, Hui Yuan, Pengzhou Zhang, "Research on Keyword Extraction Based on Word2Vec Weighted TextRank", 2nd IEEE International Conference on Computer and Communications, 2016

[4.] Md. Nizam Uddin, Shakil Akter Khan, "A Study on Text Summarization Techniques and Implement Few of Them for Bangla Language", 1-4244-1551-9/07IEEE, 2007.

[5.] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean,"Efficient Estimation of

Word Representations in Vector Space", In Proceedings of Workshop at ICLR, 2013.
[6.] Mihalcea R, Tarau P,"TextRank: Bringing Order into Texts", In: Proceedings of Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 2004, pp.404-411.
[7.] K. Sugathadasa, B. Ayesha, N. de Silva, A. S. Perera, V. Jayawardana, D. Lakmal, and M. Perera., "Synergistic union of word2vec and lexicon for domain specific semantic similarity", University of London International Programmes, 2017.
[8.] "Law2Vec: Legal Word Embeddings by Ilias Chalkidis", Available https://archive.org/details/Law2Vec
[9.] "Word Embeddings tutorials by Tensorflow Core", Available: https://www.tensorflow.org/tutorials/text/word_embeddings?source=post_page.
[10.] "Legal Case Reports Dataset by UCI Machine Learning Repository", Available: https://archive.ics.uci.edu/ml/datasets/Legal+Case+Reports.