REPRESENTATION LEARNING AND SIMILARITY OF LEGAL JUDGEMENTS USING CITATION NETWORKS

Harshit Jain and Naveen Pundir

Department of Computer Science and Engineering, IIT Kanpur, India

ABSTRACT

India and many other countries like UK, Australia, Canada follow the 'common law system' which gives substantial importance to prior related cases in determining the outcome of the current case. Better similarity methods can help in finding earlier similar cases, which can help lawyers searching for precedents.

Prior approaches in computing similarity of legal judgements use a basic representation which is either abag-of-words or dense embedding which is learned by only using the words present in the document. They, however, either neglect or do not emphasize the vital 'legal' information in the judgements, e.g. citations to prior cases, act and article numbers or names etc.

In this paper, we propose a novel approach to learn the embeddings of legal documents using the citationnetwork of documents. Experimental results demonstrate that the learned embedding is at par with the state-of-the-art methods for document similarity on a standard legal dataset.

KEYWORDS

Representation Learning, Similarity, Citation Network, Graph Embedding, Legal Judgements.

1. INTRODUCTION

It is hard to imagine modern human civilization without the 'rule of law'. Strangely, not enough effort has gone into using digital information processing and retrieval techniques in the legal domain. While digital archiving of legal information has indeed happened, the rich inter-relationships and dependencies between legal documents have not been adequately captured. The size of the information base precludes using manual methods to do this. It has to be done algorithmically.

Legal information can be put into two broad categories that are actually intimately connected. First, we have the laws that are codified in the form of articles of a constitution, acts, statutes, rules, procedures etc. passed or laid down by various bodies (like parliaments, legislatures, panchayats, local bodies, etc.) who are empowered to do so. And second, we have the vast body of judgements or decisions that interpret and apply the law or rules when two or more partiesare on opposing sides and approach an adjudicating body - typically a court. While laws, rules and procedures change relatively slowly the body of judgements grows much more rapidly.

Most countries like the Commonwealth countries (UK, Australia, Canada, India, etc.), USA and several others follow the *common law system*. In this system substantial importance is given to prior related cases to determine the outcome for the current case, a principle known as *stare*

David C. Wyld et al. (Eds): NLP, MLTEC, CLBD, SEAPP, NeTIoT, VLSIE, ITCS, ARIA, DMS - 2021
pp. 23-30, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.112302

decisis - literally 'let the decision stand'. This means judgements in cases similar to the current case are a critical component for decision making. So, it is important to invent robust similarity metrics and ways to organize documents such that similar documents cluster together and are easily accessible.

In this paper, our main aim is to calculate the similarity between legal documents by learning *suitable embedding vector space*, which captures their semantic meaning and the relationships among them. We solve this problem by transforming each legal document to a vector R^d (d isthe dimension of a suitably chosen vector space) and then calculate the cosine similarity between the vectors.

2. CHALLENGES

Measuring the similarity of legal documents is non-trivial as legal documents are quite complex in nature. They contain different kinds of information like facts of the case, citations, mention and interpretations of laws/statutes, legal reasoning, opinions on social and economic matters all of this often in a single judgement. Clearly, different similarity metrics will apply depending on the kind of similarity that is sought by the end user. In this paper the similarity we are concernedwith is guided by the *common law system*. So, we look for earlier judgements that are pertinent (i.e. similar) to the current case.

Legal documents differ from other documents in the following ways:

- Legal documents are usually long and contain complex sentences and often the required information is embedded in a mass of text making it difficult to find.
- Legal documents contain legal jargon and citations to prior cases, laws, articles, acts, etc. which is important information that is external to the document itself.
- A legal document may pertain to more than one legal issue and thus may be useful in cases that are not similar on the surface.

Standard methods like: bag-of-words vectors, document vectors constructed using word embeddings in vector spaces, latent Dirichlet allocation (LDA) to assign topic words etc. ignores or does not give the importance to the specific legal information in the judgement (like acts, articles and citations to prior cases). So, we tried to capture this legal information in the form of a citation network (weighted graph) and learned the embedding of legal judgements using that graph.

3. RELATED WORK

KUMAR ET AL. [1] proposed multiple approaches to find similar legal judgements by using the techniques of information retrieval (IR) and search engines. They analyzed four approaches to find similarity between documents and found that *legal term cosine similarity* performed well. They analysed the following four approaches: **(1) All term cosine similarity**: In this approach, they represent each document as a vector using tf-idf and the similarity is computed using cosine similarity. The tf–idf is the product of two statistics, term frequency and inverse document frequency. **(2) Legal term cosine similarity**: They only consider 'legal' words in the documents - those that occur in a legal dictionary. Then represent each document as a vector using tf-idf and the legal dictionary and similarity is again computed using cosine similarity. **(3) Bibliographic coupling (BC) similarity**: They made a citation graph of the documents by finding the citations using regular expressions and the similarity is measured using Bibliographic coupling (BC). The bibliographic coupling similarity score between two legal documents is equal to the number of

common out-citations. Out-citations for document *d* are the documents that are cited by document *d*. **(4) Co-citation (CC) similarity**: Co-citation (CC) similarity is the number of common in-citations in the citation graph. In-citations for document *d* are the documents that cite document *d*.

KUMAR ET AL. [2] presented a hybrid approach, based on the text and the citation network. They proposed an improved approach to find the similarity between the legal judgements using link-based similarity. Through the experiments, they observed that it is possible to find similar judgements by using citations. For this, they introduced the notion of paragraph-links to improve the efficiency of link-based similarity methods. To calculate paragraph-links, as each document constitutes many paragraphs and these paragraphs are considered as a separate entity. Now a similarity score across all pairs of paragraphs is measured. For measuring similarity between a pair of paragraphs, they first represented the paragraph into the tf-idf embedding and then computed the cosine similarity between them. A paragraph-link is inserted between judgements A and B if any similarity score between paragraphs is greater than a threshold.

THENMOZHI ET AL. [3] proposed an approach to retrieve precedent cases that are relevant to the current case. They used three methods to find the similarity score. **(1) With concepts and tf-idf scores**: In this approach, they extracted all forms of nouns (NN (singular noun), NNS (plural noun), and NNP (proper noun)) and considered only these nouns to get the tf-idf representation of the document. The similarity score was calculated using the cosine similarity between the tf-idf representation of the documents. **(2) With concepts, relations, and tf-idf scores**: In this approach, they extracted all forms of nouns (NN, NNS, and NNP) and verb (VB (verb, base), VBD (verb, past tense), VBZ (verb, 3rd person), VBN (verb, past participle)) as well to get the tf-idf representation of the documents. The similarity score was the cosine similarity between the tf-idf representation of documents. **(3) With concepts, relations, and Word2Vec**: In this approach, they extracted all forms of nouns (NN, NNS, and NNP) and verb (VB, VBD, VBZ, VBN) and used Word2Vec [4] to get a vector for each extracted word and represented each document as vector $\in R^{300}$ by taking the average of all the vectors of the vocabulary words present in the document.

MANDAL ET AL. [5] performed extensive experiments on legal documents using tf-idf, similarity measures (such as topic modeling), and neural network models (such as the average of word embeddings Word2Vec and document embeddings Doc2Vec [6]. They have shown thatthe Doc2Vec based technique significantly out-performs baseline techniques which utilize both text-based (TF-IDF) and network-based similarity measures (Bibliographic coupling similarity).

4. PROPOSED METHOD

4.1. Dataset Preparation

For the experiments, we collected a set of 30,016 legal judgements of the Supreme Court of India over a period from 1950 to 2012. These documents were crawled from the Legal Information Institute of India[1] that provides free access to various databases related to the Indianlegal system. An example of a legal court case (document) is shown in Figure 4.1.

In addition, we crawled the dataset of Articles defined in the Constitution of India and important Acts like IPC (Indian Penal Code), CrPC(Code of Criminal Procedure), and CPC(Code of Civil Procedure)[2]. This dataset contains the information against each section's number of acts (IPC, CrPC, CPC) and articles.

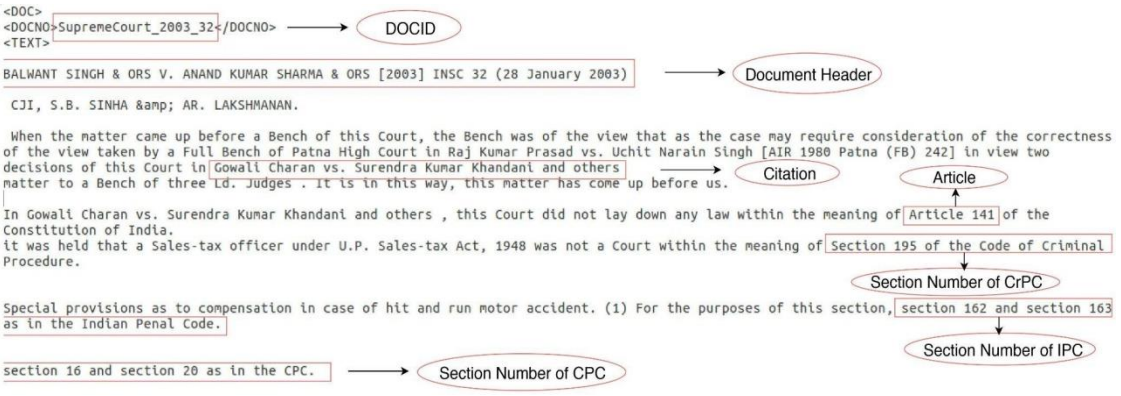

Figure 4.1. Example of a legal judgement marked with DOCID, Document Header, andCitation Information.

4.2. Our Approach

Our goal is to compute the similarity between legal documents. In order to find the similarity, we map each document in our corpus to a vector in R^d (where *d* is the dimension of the vector space) and compute their cosine similarity. The useful legal information is very sparse in the document and is completely dominated by details of the case and other kinds of information that are not useful in deciding whether or not this judgement can be used as a precedent. Since court judgements give us useful information in the form of citations, section numbers of acts and articles, rule numbers of procedures, etc, we exploit this information along with the important words of the document to learn the embeddings. These embeddings are learned using the following sub-steps: (1) Information Extraction (2) Graph Construction (3) Graph Embedding.

4.2.1. Information Extraction

The text of a judgement apart from the judgement itself contains other useful information like references to acts and article numbers and citations to similar prior cases. An example is shown in Figure 4.1. This information can be used to compute similarity among documents. Specifically, we find the article, section numbers, and citations (e.g. Oma Ram v. State Of Rajasthan and ors.) mentioned in the document using regular expressions. All this information isstored corresponding to each document in a table where every document is given a unique DOCID.

Each citation extracted from a document is mapped to the corresponding document title e.g. the extracted citation “Atma Ram vs. the State of Punjab” will be mapped with the document header “Atma Ram Kumar vs. State of Punjab and ors”. Since the citation and title cannot be matched directly, to solve this problem without manual intervention, we compute the score (given below) for each citation with respect to all the document titles. The score is calculated by

$$score = \frac{len(lcs(\text{cite}, \text{doc-header}))}{len(\text{doc-header})}$$

where *lcs* is the longest common subsequence and *len* is the number of characters in the string.

The values will be in the range [0-1].

For titles that have a score at least equal to a threshold, the citation is mapped to the document title with the highest score. If there is no such title then the citation is not mapped and is ignored. We don't want to include more false positives and also at the same time do not want to miss the true positives so we choose the threshold value as 0.6 based on our manual checks.

With this approach, we were able to map more than 55% of all the citations in all the documents.

4.2.2. Graph Construction

Citation information can be nicely captured if we convert our set of documents into an undirected weighted graph where nodes are the individual legal judgements and an edge defines the similarity between the two judgements. An example of a citation graph is shown in Figure

The resulting graph has 20,469 nodes and 109,892 edges. Some nodes are dropped since they contain no citations and also no judgement cites them. To further augment the graph, we added another type of edge called the tf-idf edge into our graph. The idea is to increase the number of edges and reduce sparsity. To add these edges, the following steps are performed sequentially:

1. Convert each document into a list of words using a tokenizer where each word is processed through a lemmatizer and excludes stopwords.
2. Since legal documents also contain some unnecessary information that is not useful in measuring the similarity, we need to remove this information. Intuitively, for any similar pair of documents, this unnecessary information will generally be specific to one document and will not be present in another document. Therefore to get the useful words in our corpus, we perform the following operation. For every pair of documents in which one document cites another, find the words that are common to both, called intersection words, then take the union of all intersection words for all citation pairs in our dataset. This final set of words are the catchwords and are found to be the most relevant words in our corpus.
3. Since our document is stored as a list of words, iterate over this list and exclude all words that are not present in the catchwords computed above.
4. Acts and articles present in a legal document are very important in capturing the similarity. To capture this information, we extended each document word list with the words that are mentioned in the definition of acts and articles which we collected in dataset preparation (section 4.1 above).
5. Now compute the tf-idf vector representation of each document.
6. Finally, compute cosine similarity (*sim*) for every possible pair of documents and if the *sim* is more than the threshold then this edge (tf-idf edge) is included in the graph with weight *sim* otherwise not. The value of threshold would affect the quality of the citation network, lower value of threshold would add more false (unrelated) edges while higher value would discard more good (related) edges. Through manual checks on a subset of document pairs, we used the value of threshold as 0.6.

Since any particular pair of documents in the graph can have both citation edge and tf-idf edge, in that case, we consider only the tf-idf edge. Finally, we get an undirected weighted graph of all the documents in our corpus. As new nodes are also included by introducing tf-idf edges. The final graph has 23,420 nodes and 355,426 edges.

4.2.3. Graph Embedding

Several graph embedding techniques have been proposed such as DeepWalk [7], LINE [8], BigGraph [9], Node2Vec [10] etc. In this paper, we used Node2Vec for feature learning. It takes as input an edge list with optional weights for edges and outputs a dense representation of the

nodes in the network. It learns a feature representation that maximizes the likelihood of preserving network neighborhoods of nodes in a *d* dimensional feature space. We ran the Node2Vec algorithm on our graph obtained in the previous step. We chose the following parameter values: *dimensions*=300, *walk length*=16, *number of walks*= 300 and *p*, *q*= 1. After Node2Vec we have dense representations for all documents in our corpus.

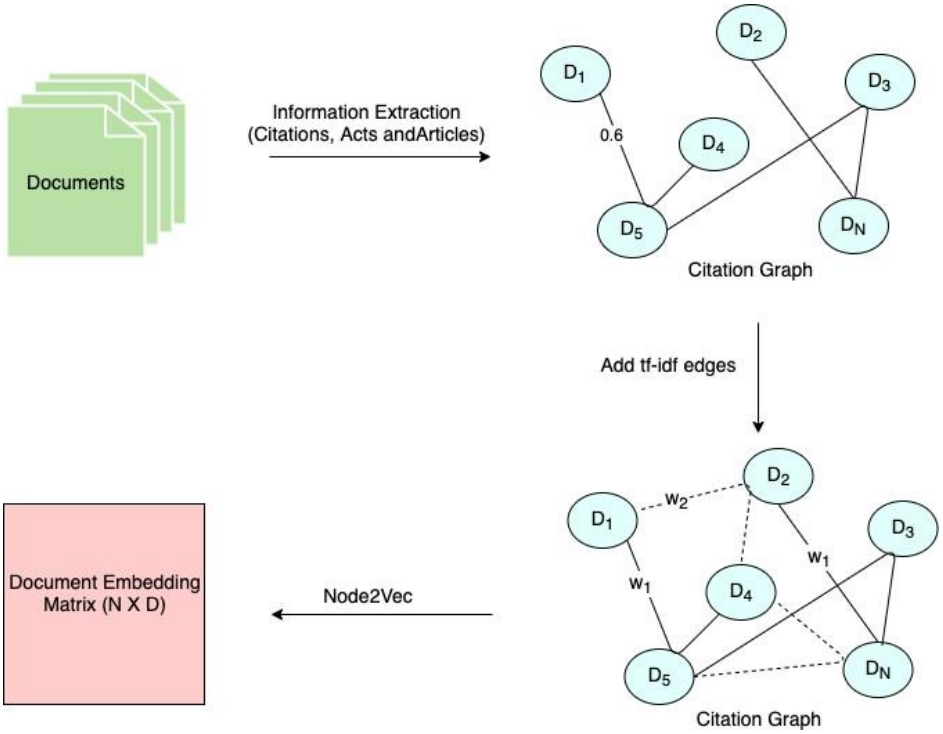

Figure 4.2. Complete system overview

5. EXPERIMENTS AND RESULTS

As discussed above, our embeddings are derived from the citation graph, so the quality of learned embeddings depends on the quality of the citation graph. We conducted experiments to evaluate the quality of (i) Citation Graph (ii) Learned Embeddings using this citation graph.

Dataset: We used the dataset consisting of 47 pairs of documents from prior work by KUMAR ET AL. [1] and MANDAL ET AL. [5] . The original dataset were tagged by experts on a scale of 0 to 10 based on the similarity of the document pair.

Baseline: For baseline performance we consider the methods proposed in KUMAR ET AL. [1] and MANDAL ET AL. [5]. They studied different methods to learn document similarity. Specifically, [1] considered the co-citation and bibliographic coupling-based similarity methods to find similar judgements and [5] performed the experiments using (1) tf-idfrepresentation of the document (2) Word2Vec, where each document vector is computed by taking the weighted average of word vectors in the document with weights being the tf-idf of theword (3) Doc2Vec [6].

Results: To compare our approach with previous benchmarks, we used the same evaluation metric as in [5] i.e. the Pearson correlation coefficient between the similarity scores computed by different methods and the expert scores. The Pearson correlation coefficient of two variables is

given by

$$\rho = \frac{cov(X, Y)}{\sigma_X . \sigma_Y}$$

where *cov(A, B)* is the covariance of variables *A* and *B*, and σ_A is the standard deviation of the variable *A*. The correlation coefficient has a value in the range [−1,1] and will indicate how similar are variables *A* and *B*.

To evaluate the Citation Graph, we computed the correlation coefficient between the expert's scores and similarity of documents (inverse of the distance between the nodes (representing the documents) in the graph). The distance between the nodes is the minimum path length between the nodes and documents that are more similar will have less distance in the graph.

To evaluate the learned embeddings we computed the correlation coefficient between the expert's scores and similarity of documents (using cosine similarity of learned embeddings of the document-pair). A higher value of cosine similarity will signify document-pair is similar and dissimilar otherwise.

The correlation coefficients of the methods are given in Table 5.1. Our learned embeddings outperforms MANDAL ET AL. [5] and other document embedding techniques. Also, we obtained 0.64 correlation (ρ) for Citation Graph which is at par with other benchmarks. The Citation Graph can further be improved using better extraction techniques or manually extracted citation pairs and may further improve our learned embeddings.

Table 5.1. Correlation Coefficient comparisons using different document representations

	KUMAR ET AL. [1]	**Word2Vec**	**tf-idf**	**Our Approach (Citation Graph)**	**MANDALET AL. [5]**	**Our Approach (Learned Embeddings)**
ρ	0.33	0.60	0.62	**0.64**	0.69	**0.73**

We also performed a binary classification task by modifying the dataset using similar settings as done in [5]. The expert scores (in the range [0, 10]) are converted into two labels '0' and '1' i.e. scores in range [0, 5] are considered dissimilar pairs and given label '0' while scores in range [6, 10] are considered similar pairs and given label '1'. At the time of prediction, a document-pair is given label '1' if the cosine similarity between the embedding of the documents is greater than 0.5, otherwise given label '0'. The confusion matrix for our approach and [5] is shown in figure 5.1.

		Our Approach	
		Similar	**Dissimilar**
Expert Score	**Similar**	12 / 17	5 / 17
	Dissimilar	1 / 30	29 / 30

		MANDAL ET AL. [3]	
		Similar	**Dissimilar**
Expert Score	**Similar**	13 / 17	4 / 17
	Dissimilar	5 / 30	25 / 30

Figure 5.1. Confusion Matrix for evaluating the performance of our approach and [5]. Here, theproblem of identifying similar documents is modeled as a two-class classification problem, where document-pair is to be classified as either similar or not similar. Our Approach achieves better accuracies (87.23%) as compared to the baseline method [5] (80.85%)

6. CONCLUSION

In this paper, we have proposed a novel approach to algorithmically learn the representations of legal judgments using the citation graph. We explored the different types of information present in legal judgements. For example: facts of the case, citations to earlier cases, acts, articles, judicial reasoning, general observations. We started with the hypothesis that for common law systems the important information in the judgement is present in a) the citations of cases, articles, acts, etc. and b) the legally important words in the text. We represented this information in the form of a citation graph and learned the low dimensional representation of legal judgements on this graph. Our experiments and results show that the learned embeddings are capable of capturing the similarity between the legal judgements and outperform previous works on legal judgement similarity tasks. These learned representations of judgements can further be used in other downstream tasks like taxonomy construction, question-answering (document retrieval for user query), summarization, etc.

We believe that our results can be further improved by using better extraction methods for determining the cited judgements, acts, articles, and other important references or using the hand-curated dataset. This will help in constructing a dense citation graph which will result in learning better embeddings. Currently, we are not capturing amendments to existing laws that happen over time, so in future work, we can capture this information and may improve our results further.

REFERENCES

[1] Kumar, S., Reddy, P. K., Reddy, V. B., and Singh, A. (2011). Similarity analysis of legal judgments. In *Proceedings of the Fourth Annual ACM Bangalore Conference*, page 17. ACM

[2] Sushanta Kumar, P. Krishna Reddy, V. Balakista Reddy, and Malti Suri. 2013. Finding Similar Legal Judgements under the Common Law System. 103–116

[3] Thenmozhi, D., Kannan, K., and Aravindan, C. (2017). A text similarity approach for precedence retrieval from legal documents. In *FIRE (Working Notes)*, pages 90–91

[4] Mikolov, T., Chen, K., Corrado, G. S., and Dean, J. (2013b). *Efficient estimation of word representations in vector space*. CoRR, abs/1301.3781

[5] Mandal, A., Chaki, R., Saha, S., Ghosh, K., Pal, A., and Ghosh, S. (2017). Measuring similarity among legal court case documents. In *Proceedings of the 10th Annual ACM India Compute Conference on ZZZ*, pages 1–9. ACM

[6] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196

[7] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena.2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM

[8] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, JunYan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World WideWeb Conferences Steering Committee

[9] Adam Lerer, Ledell Wu, Jiajun Shen, TimotheeLacroix, Luca Wehrstedt, Abhijit Bose, and AlexPeysakhovich. 2019. PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of the 2nd SysML Conference, Palo Alto, CA, USA*

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Con-ference on Knowledge Discovery and Data Mining*