Complex & Intelligent Systems (2022) 8:3199–3213
https://doi.org/10.1007/s40747-022-00673-1

ORIGINAL ARTICLE

Effective and scalable legal judgment recommendation using pre-learned word embedding

Jenish Dhanani[1] · Rupa Mehta[1] · Dipti Rana[1]

Received: 28 March 2021 / Accepted: 28 January 2022 / Published online: 17 February 2022

Abstract

Legal professionals strongly demand an automatic and convenient legal document recommendation system (LDRS) to identify similar judgments for preparing the advantageous and strategic arguments in the Court. Doc2Vec excellently learns semantically rich embedding (i.e., vector) space from the textual information of judgment corpus. During Doc2Vec learning, the practice of prior domain-specific knowledge can potentially enhance the embedding representation. This research thus proposes a pre-learned word embedding based LDRS (P-LDRS) that learns the Doc2Vec embedding using Legal domain-specific pre-learned word embedding possessing the Legal semantic knowledge. However, learning the judgment embedding from existing substantial Legal documents turns out to be a scalability issue for Doc2Vec. The proposed P-LDRS also provides additional functionality to learn the judgment embedding distributedly over the cluster of computing nodes using frameworks like MapReduce and Spark to address the scalability issue. The empirical analysis is performed with a non-distributed and a distributed variant of the proposed P-LDRS to validate the effectiveness and scalability. Experiment results showcase that proposed non-distributed P-LDRS perform significantly better than traditional Doc2Vec based LDRS with an Accuracy of 0.88, F1-Score of 0.82 and MCC Score of 0.73. They also demonstrate that the proposed distributed P-LDRS improves the time efficiency and achieves stable Accuracy of ≈0.88, F1-Score of ≈0.83 and MCC Score of ≈0.72, with an increasing number of nodes.

Keywords Legal judgment recommendation · Similarity analysis · Doc2Vec embedding · Distributed framework · MapReduce · Spark

Introduction

Advancement in information systems has steered the legal domain towards digitizing various types of legal documents such as precedent (judgment), constitutions, various codes, laws, acts, rules, and regulations. Legal systems are primarily categorized into (1) Civil Law System and (2) Common Law System, based on the importance of the source of law (types of legal documents) in judicial decisions. The Civil Law System outweighs codified Statutes and Laws, which assists in deriving judicial decisions as a key source of law. In contrast, the Common Law System emphasises previously delivered judgments as a crucial source of law.

A nation like India follows the Common Law System, which outweighs previously delivered judgments (also known as precedent) as an essential source of law to support judicial decision making [14,15,20,29]. It has been adopted by the Indian judiciary system, which follows the belief of "stare decisis" in that "similar facts, and circumstances should be treated in a similar way". Judicial decision-makers are bound to consider and follow prior judgment(s) interpretations as per concern, if the present case and prior judgment(s) have comparable arguments, facts, circumstances, and issues.

To form strategic, strong, and supportive arguments against the opponent, Legal professionals have to recognize the relevant judgments and associated Legal issues from a massive amount of previous judgments. The manual practice of identifying similar judgments is very challenging as

✉ Jenish Dhanani
jenishdhanani26@gmail.com

Rupa Mehta
rgm@coed.svnit.ac.in

Dipti Rana
dpr@coed.svnit.ac.in

[1] Sardar Vallabhbhai National Institute of Technology, Surat, India

it is labour, time, and domain knowledge-intensive. These challenges urge a strong demand to automate the process by constructing the Legal Document Recommender System (LDRS).

LDRS measures the relevance by computing the similarity scores among judgments. There are two major sources to compute the similarity score, *Referential information* like associated previous judgments (i.e., precedents), laws and codified statutes cited in judgments, and *Textual information* like facts, arguments, issues, circumstances, and associated decisions of the judgments. In recent works, Reference-based approaches [13,14,19] constructed a network of citations to apply network-based methods such as *co-citation*, *bibliographic coupling* [14] and *Node2Vec*, [10]. However, the citation network is sparse as several judgments may not refer to sufficient citations. This practice limits the scope in the field of Legal document recommendations. In contrast, Text-based approaches [14,20,26] construct the embedding space (also known as vector) using Term Frequency - Inverse Document Frequency (TF-IDF) [14], Latent Dirichlet Allocation (LDA) [1], Word2Vec [23,24], and Doc2Vec [17] methods to compute the Cosine Similarity (CS) score between Legal documents. Empirically, Mandal et al. [20] demonstrated that Doc2vec could obtain superior performance in terms of Accuracy and Correlation, referenced to the human expert's similarity scores. Moreover, many recent frameworks [15,18,29–31] also utilized both Text- and Reference-based approaches to compute the similarity score.

Motivation and contribution

Doc2vec is a prominent document embedding approach that transforms each document (i.e., judgment) into continuous, dense, and real-valued vectors, which effectively preserves semantic information. Hence, vectors of semantically related documents are in close proximity in the learned document embedding space. Doc2vec is an advanced variant of Word2Vec (Word Embedding (WE)), representing each word as a vector using contextual information (i.e., context words) and shallow Neural Network (NN). In Doc2Vec, document embedding is co-trained with word embedding (i.e., Word2Vec) where the document is considered as another context word together with actual context words.

The word embedding thus plays a prominent role in learning qualitative judgment embedding. However, Doc2Vec arbitrarily initializes word embedding, which leads to slower convergence and hard to optimize the document embedding and significantly affects the embedding quality. Instead, using domain-specific prior knowledge such as pre-learned word embedding can potentially help to enrich the semantics among the judgment embedding. Vectors of words appearing in the particular document certainly contribute to learning specific document vectors [17]. Therefore, the prior domain-specific semantic knowledge can help to enrich the document embedding during the learning. Usually, pre-learned WE owns the prior semantic knowledge as it was learned from the massive domain-specific data.

This research proposes a Pre-learned WE based LDRS (P-LDRS) that initializes the word vectors of Doc2Vec with the Legal domain-specific pre-learned WE instead of random values to enrich the judgment embedding. This practice allows the use of prior semantic knowledge (i.e., learned from large domain-specific corpora) of pre-learned WE to enhance performance. However, massive and growing judgments turn out in a voluminous corpus and large vocabulary. Traditionally utilized Doc2Vec is the memory and compute-intensive embedding approach for such voluminous data due to the in-memory computation of associated vocabulary and documents vectors. Commodity machines cannot efficiently administer such a substantial amount of judgments due to limited computational resources, which limits the scalability of P-LDRS. Hence, this research also emphasizes mitigating the scalability issue of the proposed P-LDRS by adopting distributed frameworks like MapReduce [6] and Spark [32]. Accordingly, the proposed P-LDRS provides additional functionality to learn the pre-learned WE based judgment embedding in the distributed environment of multiple computing nodes.

To the best of the authors' knowledge, the proposed work is one of the initial efforts to enrich the document representation using Legal domain-specific pre-learned WE in a distributed environment, specific to the Indian judiciary system. The key contributions of this research are highlighted as follows:

- This research proposes the P-LDRS that utilizes the prior Legal domain-specific knowledge (i.e., semantic) preserved in the pre-learned WE to enhance the performance. The empirical analysis is performed to validate the effectiveness of the proposed work using a vast amount of judgments from the Supreme Court of India. The proposed P-LDRS considered different Legal domain-specific pre-learned WEs namely, *Law2Vec* [3], *LeGlove* [9], and *Legal W2V*. Experiment results demonstrate that the proposed P-LDRS with *Legal W2V* yields superior performance in terms of Accuracy, F1-Score, and MCC Score.
- This research also proposes a distributed P-LDRS that learns the judgment embedding in the distributed environment like MapReduce and Spark to address scalability issues. Experiment results illustrate that the proposed distributed P-LDRS outcomes considerably better time efficiency than non-distributed P-LDRS without compromising the performance.

The rest of the paper is structured as follows: "Related work" presents the related work. "Proposed pre-learned word embedding based LDRS (P-LDRS)" presents the methodology of the proposed P-LDRS with distributed functionality. "Experiment analysis" explains the empirical analysis of the proposed P-LDRS. Finally, "Conclusion" summarizes the research work including challenges and future work.

Related work

This section briefly discusses the state-of-the-art works in document similarity analysis in the Legal domain. Doc2Vec and distributed frameworks like MapReduce and Spark are also described in this section. A brief discussion is also presented on the existing distributed word embedding techniques.

Similarity analysis for recommendation

Judgments include textual details such as arguments, facts, circumstances, issues, and decisions; along with referential data such as statutes, acts, and previous judgments. Existing works used Textual and /or Referential information to measure similarity among Legal documents (i.e., judgments). These approaches are mainly categorized in *Network-based* [13,14,19], *Text-based* [14,20,26], and *Hybrid* [15,18,29–31] approaches.

Network-based approaches

Network-based approaches primarily construct a citation network (i.e., referential network) by utilizing referential information. Subsequently, the similarity score is computed by analyzing the direct or indirect citations. Recent studies have considered network-based methods from the analogous domain such as Scholarly Article citation network [4]. Kumar et al. [14] analyzed the similarity scores among judgments considering the *co-citation analysis* and *bibliographic coupling* for the precedent citation network. Koniaris et al. [13] used the network statistical and structural information (i.e., degree) to capture the similarity among Legal documents from the European Union. However, the connectivity strength of the network is one of the essential aspects of the effectiveness of network-based approaches. Most precedents refer to limited precedents and/or statutes, laws, among others, which construct sparse connectivity in the citation network [20].

Text-based approaches

Text-based approaches try to capture the lexical or semantic similarity among the judgments. Mainstream approaches practice the Cosine similarity score as a similarity measure among the document vectors. Recent attempts in the field of Legal document similarity analysis have adopted several vectorization techniques such as TF-IDF, LDA, Word2Vec, and Doc2Vec, to transform the textual information into fixed-length real-valued vectors efficiently. Kumar et al. [14] considered TF-IDF to construct the judgment vector space from Indian judgments. However, TF-IDF results in sparse and high-dimensional vectors [21], failing to preserve semantic meaning. Hence, many recent works used semantic vector space modelling to enable effective similarity analysis. Nanda et al. [26] analyzed Legal document similarity considering LDA based topic modelling that may not be efficient for long textual documents such as judgments [11]. Shallow NN based embedding like Word2Vec [23,24] and Doc2Vec [17] learns the semantic vector space considering the contextual information which can prominently help to preserve the semantic relationships among the words or documents. Sagathadasa et al. [30] employed Word2Vec and lexical relevance to capture domain-specific semantic similarity. Dipankar et al. [2] designed a "risk o meter" framework to analyze the risk associated with the Legal contracts using supervised machine learning and Doc2Vec. In the empirical analysis of Mandal et al. [20], Doc2vec has superior performance in terms of Accuracy and Correlation referenced to the human expert similarity score, compared to TF-IDF, LDA, and Word2Vec.

Hybrid approaches

Kumar et al. [15] attempted the hybrid approach using textual and referential information to enhance the performance. The notion of "paragraph links" and previous judgments forms a citation network. The paragraph link exists between two judgments if the Cosine similarity score between TF-IDF vectors of two paragraphs from different judgments is above the threshold. A significant enhancement in performance can be observed when compared to standalone approaches. Raghav et al. [29] performed the cluster analysis using paragraph links and a citation network for Indian Legal documents. Leibon et al. [18] have also employed the network-based method and the text representation approach such as LDA for opinions from the US Supreme Court. An approach proposed by Sugathadasa et al. [31] utilized a text-based approach like TextRank [22] for sentence similarity and a network-based approach like Node2Vec [10] for node embedding to find relevant Legal documents.

Doc2Vec embedding

Doc2vec maps the arbitrary length of textual data (i.e., sentences, paragraphs, documents) into the semantically rich, low-dimensioned, continuous, and real-valued vectors. There

are two classical variations of Doc2Vec 1) *Distributed Memory:* which learns the document embedding along with word vectors, and 2) In *Distributed Bag of Words,* learning of word vectors is optional with document embedding. This research emphasises the Distributed Memory variant due to its admissible performance with most tasks [17]. As discussed earlier, it is the extended variant of the profound word embedding Word2Vec that learns semantically rich word vectors. It follows the belief of the Distributional hypothesis, summarized as "semantically similar words share a similar kind of contextual information". Word2Vec uses contextual information (Neighboring words of the target word) for learning the word vector (target word) to preserve the semantic meanings. In Doc2Vec, the document vector matrix is allied with the word vectors matrix of Word2Vec. Each document can be considered as an additional adjacent word along with context words (i.e., contextual information). This practice allows to co-learn the document vectors together with the word vectors, aiming to preserve the semantics among documents. Hence, semantically similar documents can be placed in close proximity in the document vector space.

NN can also be trained in a transferable way because it has an incremental learning nature [25] which allows the network to be initialized with the pre-learned parameters (i.e., weights or vectors). Hence, the practice of pre-learned word embedding instead of random initialization can be an effective feature in the NN based model like Word2vec and Doc2vec. Patel et al. [28] fine-tuned the pre-learned word embedding for the domain-specific medical coding data to enhance the performance. Lau et al. [16] performed an extensive empirical assessment of the Doc2Vec on various tasks. The work has also used pre-trained word embedding to learn the document vectors (i.e., word vectors and Distributed Bag of Words based on Doc2Vec). The authors have observed a significant improvement in the performance and model convergence speed. This research focused on the other variant of Doc2Vec, namely Distributed Memory, due to its better performance in our preliminary experiments.

Distributed framework

Distributed frameworks like MapReduce[1] [6] and Spark[2] [32] are prominently used to address the issue of scalability using the computational resources from multiple computing nodes. Usually, distributed frameworks use the Distributed File System to distribute data partitions (also known as data blocks or chunks) over multiple computing nodes. MapReduce comprises of two essential sequential Phases namely Map Phase (i.e., Mapper class) and Reduce Phase (i.e., Reducer class). Map Phase initiates multiple Map functions over the computing nodes to parallelly apply user-defined logic to the available partition. Intermediate results are outputted by each Map function which is locally preserved in the form of <*Key, Value*> pairs. The Reducer invokes the Reduce functions that collect the intermediate results to produce the final outputs based on the user-defined logic. Similarly, Spark provides several transformations like *mappartition*, *map*, etc. to perform user-defined logic over partitions. In addition, it also provides *reducebykey*, *groupbykey*, etc. transformation to aggregate the values based on the *Key*, as per the user-defined logic. These transformations are specifically applicable for the *Key-Value* paired dataset (i.e., intermediate output).

Ji et al. [12] attempted distributed learning of word embedding from the large volume of textual data consisting of a large vocabulary. Each node locally learns the word embedding, which is periodically updated by synchronizing with embedding from other nodes. Further, Ordentlich et al. [27] proposed a parameter-server based framework to learn word embedding using a column-wise distribution of vectors. Dhanani et al. [7] proposed a MapReduce based word embedding consisting of two phases, Map and Reduce. Map phase constructs the local word embedding from each partition and Reduce phase performs element-wise averaging on local word embeddings to prepare the final word embedding.

Research gap

In the field of Legal information systems, previous works centred on analyzing the direct or indirect citations (i.e., referential information) and/or measuring the semantic similarity among the Legal documents (i.e., textual information), capturing the relevance [13–15,18–20,29–31]. In the LDRS, the performance of Network-based approaches is highly affected by the network density, which in the case of Indian judgments is very sparse due to limited citations [20]. Alternatively, Doc2Vec has shown admirable performance due to its excellent ability to preserve the semantics among Legal documents. There is still scope to improve the performance using pre-learned WE consisting of prior Legal domain-specific semantic knowledge. However, the substantial amount of Legal documents turns out to be a scalability issue for document embedding. The in-memory computation using Neural Network for the vocabulary, related word and document vector space makes Doc2Vec and Word2Vec [27] memory and compute-intensive for such a large number of Legal documents [8]. This problem has received insufficient consideration in existing systems. Recent related attempts [13–15,18–20] are centred on qualitatively capturing the semantic relevance for enhancing the prediction correctness. To mitigate the mentioned challenge, this research proposed a

[1] The Apache Hadoop software library is a framework for the distributed processing: https://hadoop.apache.org/.

[2] Apache Spark is a framework for the in-memory distributed processing: https://spark.apache.org/.

distributed framework that learns the document embedding in the distributed environment of MapReduce and Spark. To the authors' knowledge, these efforts have not been previously investigated in the literature, specifically the Legal document similarity analysis. The following section presents a detailed discussion of the proposed approaches.

Proposed pre-learned word embedding based LDRS (P-LDRS)

The identification of relevant judgments is a significant demand for Legal professionals. Doc2Vec effectively captures the semantic relevance among Legal documents using contextual information. As discussed earlier, Doc2Vec trains the conventional word vectors (i.e., Word2Vec) and document embedding. Therefore, the word vectors significantly affect the learning process of document embedding. This research proposes P-LDRS that initializes the word vectors with the Legal domain-specific pre-learned WE instead of random values to enrich the judgment embedding. This practice prevents learning the word vectors from scratch. Instead, it uses the already learned Legal domain-specific knowledge (i.e., semantic meanings of words) preserved in a pre-learned WE. This research also proposes a distributed framework (i.e., Distributed P-LDRS) to learn the judgment embedding from the large volume of judgments over multiple computing nodes. The architecture of the proposed P-LDRS is shown in Fig. 1, which can be divided into three significant stages (1) Data preparation, (2) Document Embedding, and (3) Legal Judgment Recommendation.

Data preparation

This stage extracts the associated textual and referential information from the given judgment corpus. Judgments are transcripted in natural language by different humans, which causes linguistic and structural heterogeneity. The raw judgment file consists of paragraph wise judgment text like arguments, facts, issues, verdicts of the court, and various meta-data like judge name, time-based information, and headnotes. This research extracts the judgment text and eliminates irrelevant meta-data by crafting contextual rules. Standard pre-processing techniques are adopted to structure the judgment text, such as lowercasing the text, removing stop-words, white spaces, line breaks, tiny words (smaller than three characters), punctuations, numbers, and noisy data like proper names. To use referential information along with textual information, we assigned unique IDs to the text of all the cited references for uniformity throughout the corpus. This research also performs domain-specific pre-processing by converting the Legal domain-specific phrases like 'high courts', 'district court', 'state courts', etc. into single tokens to prevent irregularity in vocabulary. Additionally, the inconsistency is observed in paragraph length having few or thousands of words. This research eliminates those tiny (< 10 words) and extra-large (> 100 words) paragraphs as the content of the large paragraphs dominates small paragraphs in the vector representation.

Document embedding

The *Document Embedding* stage aims to efficiently learn the semantically rich, continuous, low-dimensional, and real-valued vectors for each judgment. Therefore, vectors of semantically related judgments are in close proximity in the judgment embedding space. An input to this stage is pre-processed judgments and Legal domain-specific pre-learned WE to initialize the word vectors. The proposed P-LDRS can learn the judgment embedding using pre-learned WE either on the standalone computing node (i.e., Non-distributed learning) or in a distributed environment of multiple computing nodes (i.e., Distributed learning), as follows.

Non-distributed learning of judgment embedding

Assume the Judgment Corpus (JC) consists of n pre-processed judgments ($Judgment$), and Legal domain-specific pre-Learned WE (LPWE) contains vocabulary (V_{LPWE}) which is a set of m unique words (W_{LPWE}) and vectors ($Vector_{LPWE}$) comprising of respective vector representation ($Vec_{(LPWE,i)}$) of vocabulary word ($W_{(LPWE,i)}$), as follows:

$$JC = \{Judgment_1, Judgment_2, ..., Judgment_n\} \quad (1)$$

$$V_{LPWE} = \{W_{(LPWE,1)}, W_{(LPWE,2)}, ..., W_{(LPWE,i)}, W_{(LPWE,m)}\} \quad (2)$$

$$Vector_{LPWE} = \{Vec_{(LPWE,1)}, Vec_{(LPWE,2)}, ..., Vec_{(LPWE,i)}, Vec_{(LPWE,m)}\} \quad (3)$$

Vocabulary words and their respective vectors (i.e., prepared from the JC) are used during the learning of Doc2Vec. Therefore, the proposed LDRS builds the vocabulary (V_{JC}) from the Judgment Corpus consisting of k unique words ($W_{JC)}$), as shown in Eq. (4). Respective vectors ($Vector_{JC}$) of vocabulary (V_{JC}) are initialized with the help of pre-learned WE (V_{LPWE} and $Vector_{LPWE}$) instead of only random values. If the word $W_{(JC,j)}$ from vocabulary (V_{JC}) is present in the vocabulary (V_{LPWE}) then the respective vector $Vec_{(JC,j)}$ of $Vector_{JC}$ is initialized with the vector $Vec_{(LPWE,i)}$ of $Vector_{LPWE}$ (i.e., $W_{(LPWE,i)} = W_{(JC,j)}$), otherwise initialized with random values, as shown in Eqs. (5) and (6). There is a possibility that pre-learned WE may not contain all the vocabulary words of (V_{JC}) as both are built from a different corpus.

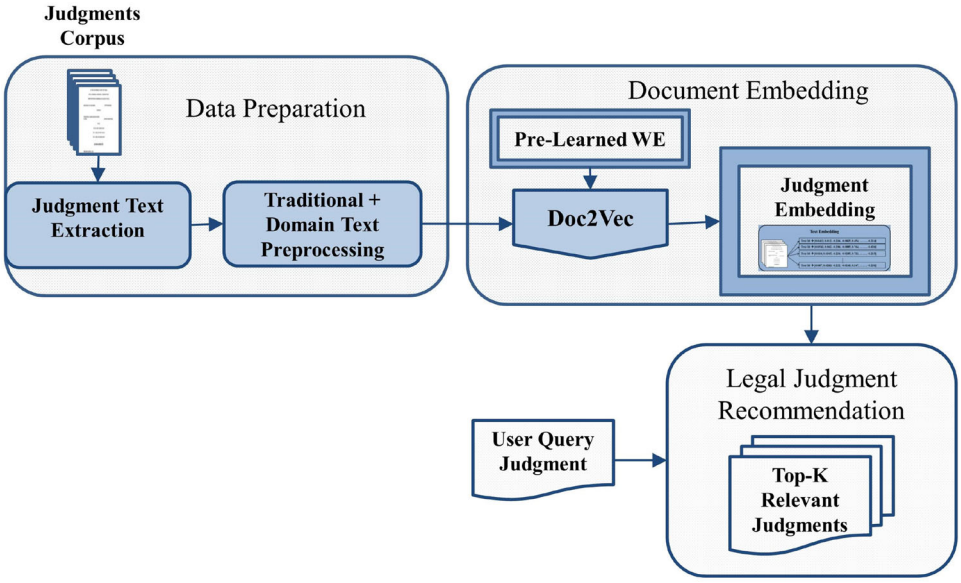


Fig. 1 Architecture of the proposed P-LDRS

$$V_{JC} = \{W_{(JC,1)}, W_{(JC,2)}, ..., W_{(JC,j)}, W_{(JC,k)}\} \quad (4)$$

$$Vec_{(JC,j)} = \begin{cases} Respactive\ Vec_{(LPWE,i)}\ of\ W_{(JC,j)} & \text{if } W_{(JC,j)} \in V_{LPWE} \\ RandomVector & \text{Otherwise} \end{cases} \quad (5)$$

$$Vector_{JC} = \{Vec_{(JC,1)}, Vec_{(JC,2)}, ..., Vec_{(JC,j)}, Vec_{(JC,k)}\} \quad (6)$$

Once word vectors are initialized, Doc2Vec starts learning judgment embedding with user-defined parameters and pre-learned word embedding. The final result of this stage is Judgments (J) and their respective vectors ($JVector$), as follows:

$$J = \{J_1, J_2, ..., J_i, J_n\} \quad (7)$$

$$JVector = \{JVec_1, JVec_2, ..., JVec_i, JVec_n\} \quad (8)$$

Here, J is the set of n judgments, and $JVector$ is the set of respective vector representations ($JVec_i$) of judgment (J_i).

Distributed learning of judgment embedding

Doc2Vec based LDRS has demonstrated admirable performance due to its excellent ability to capture the semantics. However, the substantial amount of Legal documents urge a strong demand for a scalable framework as a commodity machine contains limited computational resources. To enhance the scalability, the proposed P-LDRS can also provide a functionality to learn the judgment embedding distributedly using Legal domain-specific Pre-Learned WE over multiple computing nodes. The proposed distributed framework is logically divided into two phases: *Distribution Phase* and *Collection Phase*, as illustrated in Fig. 2. Initially, the judgment corpus is primarily divided into several partitions distributed among multiple computing nodes (i.e., using Distributed file system). The Distribution Phase aims to learn the Local Judgment Embedding (LJE) using the pre-learned WE from individual partitions, over the multiple nodes. The Collection Phase accumulates the learned LJEs from all nodes to produce the final judgment embedding. The algorithmic representation of the proposed approach is depicted in Algorithm 1.

Each node is primarily loaded with the Legal domain-specific pre-learned WE consisting of vocabulary (V_{LPWE}) and associated vectors ($Vector_{LPWE}$), as shown in Eqs. (5) and (6). Let us assume, Judgment Corpus (JC) contains n judgments which are divided into D partitions (P) (i.e., the number of nodes) as shown in Eq. (9). On each node, the Distribution Phase builds the vocabulary (V_D) from locally available partition (i.e., D^{th} partition) consisting of k_D unique words, as shown in Eq. (10) (i.e., In Algorithm 1, Step No: 2).

$$JC = \{P_1, P_2, ..., P_D\} \quad (9)$$

$$V_D = \{W_{(D,1)}, W_{(D,2)}, ..., W_{(D,j)}, W_{(D,k)}\}\ and\ |V_D| = k_D \quad (10)$$

To initialize the word vectors on each node, the proposed framework follows the same process as discussed in Sect. "Non-distributed learning of judgment embedding". If the word ($W_{(D,j)}$) from vocabulary (V_D) is present in the vocabulary (V_{LPWE}) then the respective vector ($Vec_{(D,j)}$) of $Vector_D$ is initialized with the vec-

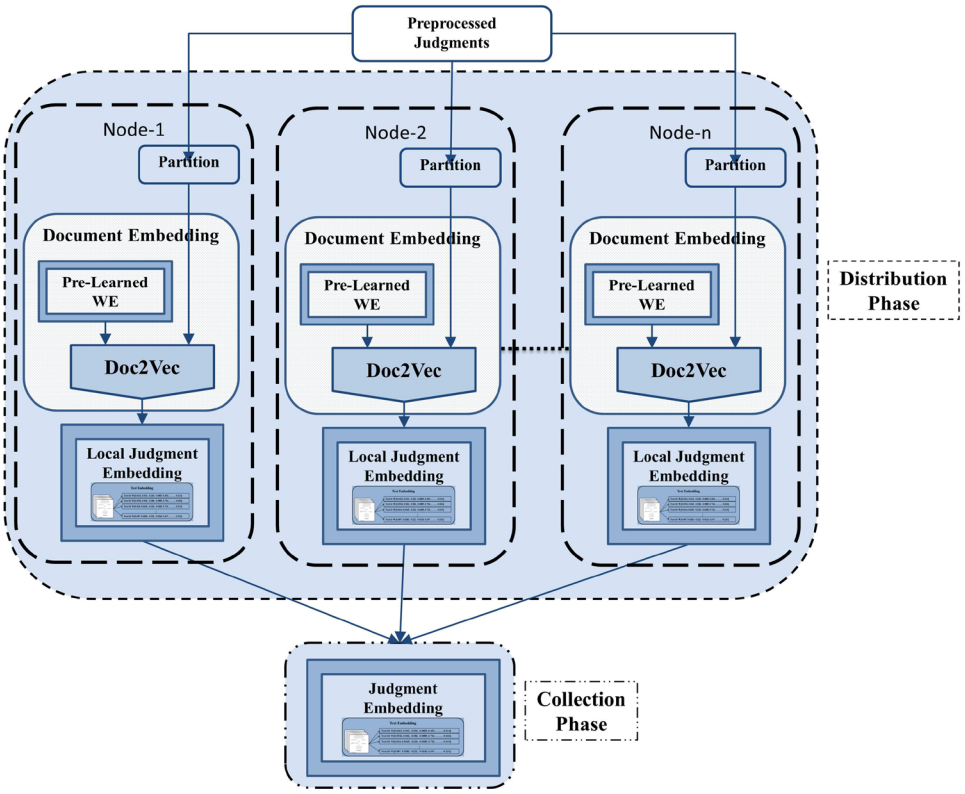


Fig. 2 Proposed framework for the distributed learning of judgment embedding using Pre-learned WE

Algorithm 1: DISTRIBUTED LEARNING OF JUDGMENT EMBEDDING USING PRE- LEARNED WE

Input: Judgment Corpus (JC): a set of D Partitions $\{P_1, P_2, ..., P_i, P_D\}$ which are distributed over D nodes, and Pre-Learned WE: Vocabulary (V_{LPWE}) and respactive Vector ($Vector_{LPWE}$)

Output: Judgment Embedding, consisting of Judgments $\{J_1, J_2, ...J_i, J_n\}$ and their respactive $\{JVec_1, JVec_2, ..., JVec_i, JVec_n\}$

1 **Distribution Phase** $V_i \leftarrow$ Build Vocabulary (P_i)
2 **for** *each Word* (W_j) ***in*** V_i, **do**
3 **if** *Word* (W_j) *is* ***in*** *Vocabulary* (V_{LPWE}) **then**
4 $Vec_j \leftarrow$ Respactive Vector of W_j from $Vector_{LPWE}$
5 **end**
6 **else**
7 $Vec_j \leftarrow$ Random Vector
8 **end**
9 **end**
10 $LJE_i \leftarrow Doc2Vec(P_i)$
11 **Distribution Phase Output:** $\{LJE_1, LJE_2, ..., LJE_i, LJE_D\}$
12 **Collection Phase**
13 **for** *each* LJE_i *in* ***Distribution Phase Output*** **do**
14 $JE \leftarrow JE \bigcup LJE_i$
15 **end**
16 **Collection Phase Output:** JE consist of Judgments $\{J_1, J_2, ...J_i, J_n\}$ and their respactive vectors $\{JVec_1, JVec_2, ..., JVec_i, JVec_n\}$

tor ($Vec(LPWE, i)$) of $Vector_{LPWE}$ (i.e., $W_{(LPWE,i)} = W_{(D,j)}$), otherwise initialized with random values (i.e., In Algorithm 1, Step No: 3 to 10), as follows:

$$Vec_{(D,J)} = \begin{cases} Vec_{(LPWE,i)}\ of\ word\ W_{(D,j)} & \text{if } W_{(D,j)} \in V_{LPWE} \\ RandomVector & \text{Otherwise} \end{cases} \quad (11)$$

$$Vector_D = \left\{vec_{(D,1)}, vec_{(D,2)}, ..., vec_{(D,j)}, vec_{(D,k)}\right\} \quad (12)$$

Once the word vectors are initialized, Doc2Vec learns the local judgment embedding (LJE) from the available partition over each node (i.e., In Algorithm 1, Step No: 11). Local judgment embedding comprises of judgment (J_D) and their associated learned vectors, as follows:

$$J_D = \left\{J_{(D,1)}, J_{(D,2)}, ..., J_{(D,j)}, J_{(D,r)}\right\} \quad (13)$$

$$JVector_D = \left\{JVec_{(D,1)}, JVec_{(D,2)}, ..., JVec_{(D,j)}, JVec_{(D,r)}\right\} \quad (14)$$

Here, J_D is a set of all judgments ($J_{(D,j)}$), r is the number of judgments, $JVector_D$ is a vector set of the corresponding vector ($JVec_{(D,j)}$) of Judgment ($J_{(D,j)}$) from D^th partition. In the Collection Phase, all the local judgment embedding are merged to form the final judgment embedding consisting of

all the judgment and their respective vectors (i.e., In Algorithm 1, Step No: 14 to 16), as follows:

$$J = \bigcup_{i=1}^{i=D} J_i = \{J_1, J_2, ...J_i, J_n\} \tag{15}$$

$$JVector = \bigcup_{i=1}^{i=D} JVector_i = \{JVec_1, JVec_2, ..., JVec_i, JVec_n\} \tag{16}$$

Legal judgment recommendation

The learned judgment embedding is referred to recommending similar judgments given a query judgment. In this sense, to measure the relevance between two judgments (i.e., J_a and J_b), this research computes the profound **C**osine **S**imilarity **score** (CS score) between the vectors of those judgments, as computed in Eq. (17).

$$CS\ Score(J_a, J_b) = \frac{JVec_a \cdot JVec_b}{|JVec_a| \cdot |JVec_b|} \tag{17}$$

$JVec_a \cdot JVec_b$ specifies the dot product of the judgment vectors $JVec_a$ and $JVec_b$, and $|JVec_a|$ and $|JVec_b|$ specifies the length of the vector $JVec_a$ and $JVec_b$, respectively. Then, given a query judgment, Top-k similar judgments can be recommended by identifying judgments with the highest k CS scores.

Experiment analysis

An empirical analysis is performed to evaluate and validate the proposed P-LDRS. This section presents the dataset employed and various performance measures used throughout the experimentation. The proposed P-LDRS provides an elective functionality for learning the pre-learned WE based judgment embedding in a non-distributed or distributed environment. So, this research performed two separate experiments to validate the proposed Non-Distributed and Distributed P-LDRS empirically.

Dataset and performance measures

Experimentation is performed on the training dataset containing real judgments from the Supreme Court of India. The dataset was crawled from the famous Indian Legal repository[3], including more than 48,000 judgments delivered from 1950 to 2016 (76 Years). In the absence of a standard testing dataset, Kumar et al. [14] collected expert similarity scores

[3] Indian Kanoon Legal Document Repository: https://indiankanoon.org/.

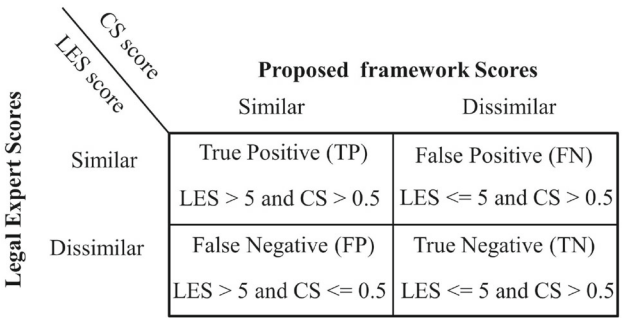


Fig. 3 Confusion matrix to evaluate the performance

for 50 pairs of judgments from the Supreme Court of India. **L**egal domain **E**xperts assigned the **S**imilarity score (LES score) in the range of 0 (Lowest Similarity) to 10 (Highest Similarity), where a pair is considered to be similar if LES score >= 5, otherwise dissimilar. For the same pairs, if the proposed LDRS predicts the CS score > 0.5 a pair is considered to be predicted similar, otherwise dissimilar.

The performance is evaluated by considering the binary classification measures like Accuracy, F1-Score, and Matthews Correlation Coefficient score (MCC score), computed from the confusion matrix presented in Fig. 3. Accuracy measures complete correctness by calculating the ratio of correctly predicted similar and dissimilar judgment pairs (TN+TP) to all judgment pairs (TN+TP+FP+FN). However, the disproportion is observed in test pair's classes (i.e., Similar and Dissimilar). So, this research also computes F1-Score, which represent the harmonic mean of Precision and Recall. Precision is the ratio of the correctly predicted similar judgment pairs (TP) to all the predicted similar judgment pairs (TP+FP). Recall is the ratio of correctly predicted similar judgment pairs (TP) to all actual similar judgment pairs (i.e., judgment pairs with LSE > 5) (TP+FN). MCC score is also computed, as it provides significant performance insights for imbalance data by considering TP, TN, FP and FN in balanced proportion [5]. As shown in Eq. 18, it can be computed having the range on the scale of -1 (Worst) to 1 (Best).

$$MCC\ Score = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TP + FP)(TN + FN)}} \tag{18}$$

Non-distributed P-LDRS

This subsection evaluates the effectiveness of the proposed non-distributed P-LDRS compared to the traditional Doc2Vec (D2V) based LDRS. Based on this literature, this research found two publicly available Legal domain-specific pre-learned WEs, namely *Law2Vec* [3] and *LeGloVe* [9]. However, Indian Legal documents were not considered in the

training of above WEs. So, this research also constructs the WE namely *Legal W2V* by considering the dataset mentioned above of Indian judgments (i.e., discussed in Sect. "Dataset and performance measures"). The detailed description of these pre-learned WEs is as follows:

- **Law2Vec:** It was trained using more than 123,000 Legal documents from various legislations like UK, European, Canadian, Australian, USA, etc. Law2Vec contains the 160,439 vocabulary words and their respective 100-dimensional vectors, which were learned using the Skip-Gram variant of Word2Vec.
- **LeGlove:** It was learned from the 63,981 opinions of the Supreme Court of the United States. The word co-occurrence based Glove model was employed to construct LeGlove having more than 417,000 vocabulary words and their respective 100-dimensional vectors.
- **Legal W2V:** This research utilized Word2Vec (i.e., CBOW variant) to learn the Legal W2V from more than 48,000 Indian judgments. It contains 34,931 vocabulary words and their respective 100-dimensional vectors[4].

Doc2Vec includes various hyper-parameters like vector size (V), window size (W), and iterations (ITR), having a significant impact on the performance and learning time. Window size is the range of context words (i.e., previous and next words) from the target word. Vector size describes the dimensionality of the word and document vector. Iteration indicates the number of times it iterates through the corpus during the learning of embedding. This research performed rigorous experiments on the intuitively chosen vector sizes (i.e., 100 to 300 with the interval of 100), window sizes (i.e., 5–20 with the interval of 5) and the number of iterations (i.e., 5, 10, 15) to obtain the best-performing parameters for D2V based LDRS and proposed P-LDRS (i.e., D2V+Law2Vec, D2V+LeGlove, and D2V+Legal W2V). Table 1 shows the values of the best performing parameters, which are considered for further experimentation.

Experiments were performed over the system configured with 16GBs of RAM, Ubuntu 14.04 operating system, and Intel i7-7700 processor with 8 CPUs (i.e., four cores). The Python based Gensim Machine Learning framework[5] was employed to learn the Doc2Vec and Word2Vec embedding.

[4] Extensive experimentation was performed with different Word2Vec parameters viz. vector size, window size and iteration to obtain the most promising performing one. The best performance was observed with the vector size of 100, the window size of 10 and the number of iterations is 10.

[5] The Machine Learning (ML) library for training of vector embeddings: https://radimrehurek.com/gensim/.

Table 1 The best performing parameters for various judgement Embeddings

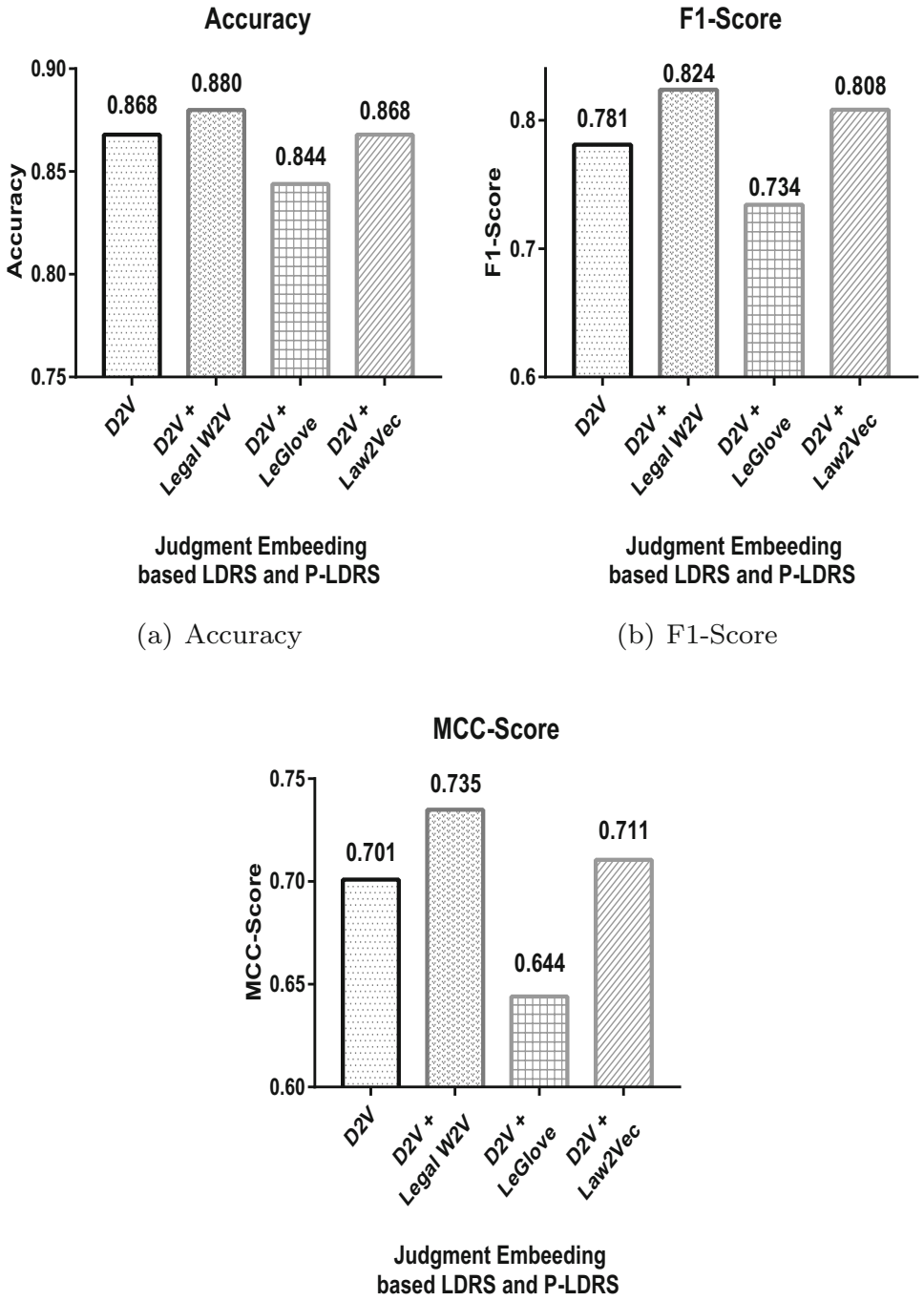
Judgment Embedding	V	W	ITR
D2V	100	20	10
D2V+Law2Vec	100	20	5
D2V+LeGlove	100	15	10
D2V+Legal W2V	100	10	5

Result analysis

Figure 4 showcases the performance of standard Doc2Vec (D2V) based LDRS and the proposed P-LDRS with the mentioned pre-learned WEs (i.e., Legal W2V (D2V+Legal W2V), LeGlove (D2V+LeGlove) and Law2Vec (D2V+Law 2Vec)). As shown in Fig. 4a–c, the proposed D2V+Legal W2V based P-LDRS demonstrate the best performance in terms of Accuracy, F1-Score and MCC-Score. It showcases a significant improvement in the performance of D2V, as the proposed P-LDRS initializes the word vectors of D2V using pre-learned WE (i.e., Legal W2V) instead of random values. Here, the Legal W2V possesses essential semantic knowledge that can significantly enrich the judgment embedding. Also, Legal W2V based P-LDRS outperforms LeGlove and Law2Vec based P-LDRS. The underlying reason is that both Legal W2V and judgment embedding are trained from the Indian judgments, which steered toward a better match of vocabulary and other linguistic features compared to LeGlove and Law2Vec. Due to such encouraging performance, this research considered the Legal W2V pre-learned WE for experimentation of distributed P-LDRS.

Distributed P-LDRS

This experimentation evaluates the effectiveness and efficiency of the proposed distributed P-LDRS compared to non-distributed P-LDRS and Doc2Vec based LDRS. For better comparison, Doc2Vec is also implemented in the distributed environment by adopting the proposed distributed learning process but without using pre-learned WEs. In addition to the mentioned performance measures (i.e., Accuracy, F1-Score and MCC Score), *Time efficiency* is also computed to evaluate the proposed distributed approach's efficiency over the existing non-distributed approaches. Time efficiency is considered as the amount of time necessary to learn the judgment embedding space from the given pre-processed corpus. The proposed distributed approach is implemented using two state-of-the-art distributed platforms such as MapReduce and Spark, discussed in the following subsections.



(a) Accuracy (b) F1-Score

(c) MCC Score

Fig. 4 Performance of D2V based LDRS and proposed P-LDRS

MapReduce based approach

The *Distribution Phase* of the proposed distributed approach is analogous to the *Map Phase* of MapReduce, which inputs and outputs the <*Key*-*Value*> pairs. The input to the Map Phase is an instance (i.e., record) which is considered to be individual judgment (i.e., text) from the locally available partition. Here, *Key* is the byte offset, and *Value* is the judgment text. In this sense, all judgments are collected from the partition to learn the Local Judgment Embedding (LJE) by applying the proposed approach. The output of the *Map Phase* (i.e., intermediate output) is $LJEs$, where *Key* is the judgment (J_i), and *Value* is the respective vector ($JVec_i$), as shown in Eqs. (13) and (14). The *Collection Phase* amalgamates the multiple locally generated intermediate outputs ($LJEs$) to build the final judgment embedding, as shown

in Eqs. (15) and (16). Consequently, the proposed implementation does not need to initialize the *Reduce Phase* of MapReduce, as each LJE contains unique judgments (i.e., available in the individual partition) and their respective vectors. This practice prevents excessive data transfer and redundant computational efforts.

Spark based approach

For better assessment, this research also implements the proposed distributed approach using Spark which supports in-memory processing. In the Spark implementation, *Distribution Phase* uses *mappartition* transformation to learn the LJE from individual RDD partition (i.e., partition). The output is new RDD partitions containing the intermediate output in the form of $<Key\text{-}Value>$ pairs, where, *Key* is the judgment (J_i) and *Value* is the respective vector ($JVec_i$), as illustrated in Eqs. (13) and (14). Similar to MapReduce, *Collection Phase* straightaway amalgamates RDD partitions to build the final judgment embedding without the use of additional aggregate transformations, as illustrated in Eqs. (15) and (16).

Hadoop and spark cluster configuration

Experiments are performed using a cluster of three computing nodes, where each node contains 16GBs of RAM, Ubuntu 14.04 operating system and Intel i7-7700 processor with 8 CPUs (i.e., four cores). One node acts as a Master and Slave (i.e., Worker) in the cluster, while the remaining two nodes entirely act as Slaves(i.e., Workers). Each node is configured with MapReduce (Hadoop 2.7.2 with YARN Resource Manager), Spark (Spart-1.2.1 with Standalone Cluster Manager), and Gensim to develop the proposed approach. This research used the Hadoop Distributed File System for a distributed storage of judgment corpus over the cluster.

Results analysis

In the proposed P-LDRS, Legal W2V is considered for further experimentation as it has demonstrated promising performance in Sect. "Result analysis". Figure 5 illustrates the time efficiency and performance of the MapReduce implementation of proposed distributed P-LDRS (MRP-LDRS) and distributed Doc2Vec based LDRS (MRD2V-LDRS). Likewise, the time efficiency and performance of the Spark implementation of proposed distributed P-LDRS (SparkP-LDRS) and distributed Doc2Vec based LDRS (SparkD2V-LDRS) are depicted in Fig. 6. Non-distributed variants of P-LDRS and D2V based LDRS[6] (Number of Nodes is equal to 1) are compared with the proposed distributed variants of P-LDRS and D2V based LDRS (Number of Nodes are equal to 2 and 3). To evaluate the impact of multiple nodes on the performance and time efficiency, we performed the experiments on an increasing number of nodes (i.e., 1, 2, and 3).

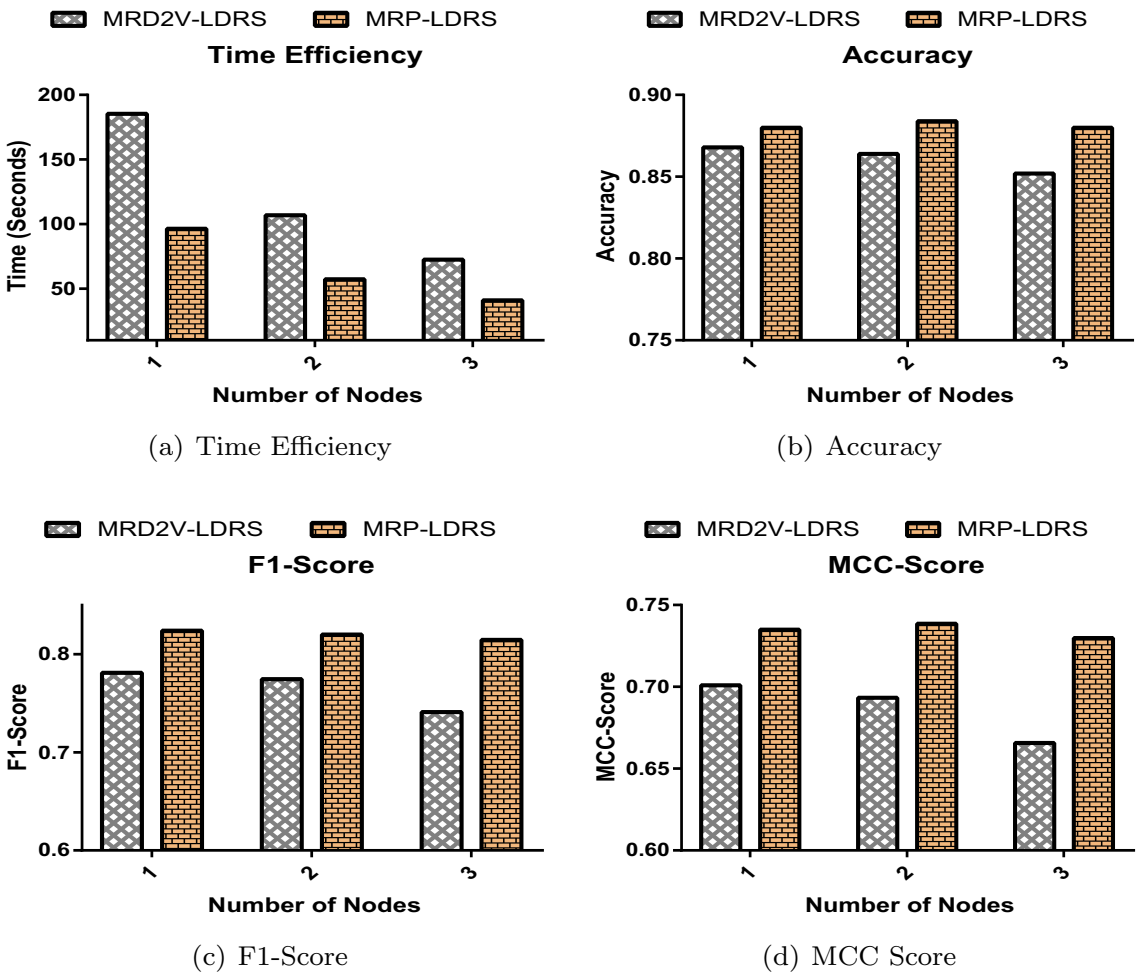
Figures 5a, 6a demonstrate that proposed distributed approaches yield significantly better time efficiency than non-distributed approaches. In addition, the required time is also declining with the increasing number of nodes, as the corpus is divided into a smaller size of partitions, which decreases the computational load on an individual node. It is also observed that the proposed P-LDRS is time efficient than the D2V based LDRS due to fewer iterations are required by the P-LDRS than the D2V based LDRS, as shown in Table 1.

It is observed that non-distributed D2V based LDRS yields better performance in terms of Accuracy, F1-Scores and MCC-Score compared to the distributed variants, as depicted in Figs. 5b–d and 6b–d. The underlying reason is that embedding of non-distributed D2V uses the entire corpus during the learning process. Whereas, the embedding of distributed D2V has limited coverage to the respective partition (of corpus) only, resulting in a lack of semantic knowledge. Therefore, slight performance deterioration can be observed with the increasing number of nodes. In contrast, the proposed non-distributed and distributed P-LDRS (i.e., MRP-LDRS and SparkP-LDRS) outcomes the competent performance, as judgment embedding space is learned using the pre-learned WE that owns Legal domain-specific semantics knowledge. This practice thus considerably helps to capture the essential semantics even from a lesser amount of data (i.e., partitions). Also, the proposed distributed P-LDRS showcases noticeably better performance compared to D2V based LDRS due to the usage of pre-learned WE that enriches the judgment embedding during the learning.

Discussion

Doc2Vec has demonstrated admirable performance in the field of Legal document similarity analysis. To enhance the performance of Doc2Vec, the proposed P-LDRS initializes word vectors with Legal domain-specific pre-learned WE instead of random values. The proposed P-LDRS showcases a significantly better performance than the LDRS due to the usage of pre-learned word embedding. Here, the pre-learned word embedding possesses essential semantic knowledge that can significantly enrich the judgment embedding. This research experimented with three different pre-learned WEs, namely Law2Vec, LeGlove, and Legal

[6] The number of nodes is equivalent to the number of partitions utilized to learn the Local judgment embedding on an individual node. Thus, approaches with 1 partition (also a number of the node) are considered as non-distributed.



(a) Time Efficiency (b) Accuracy

(c) F1-Score (d) MCC Score

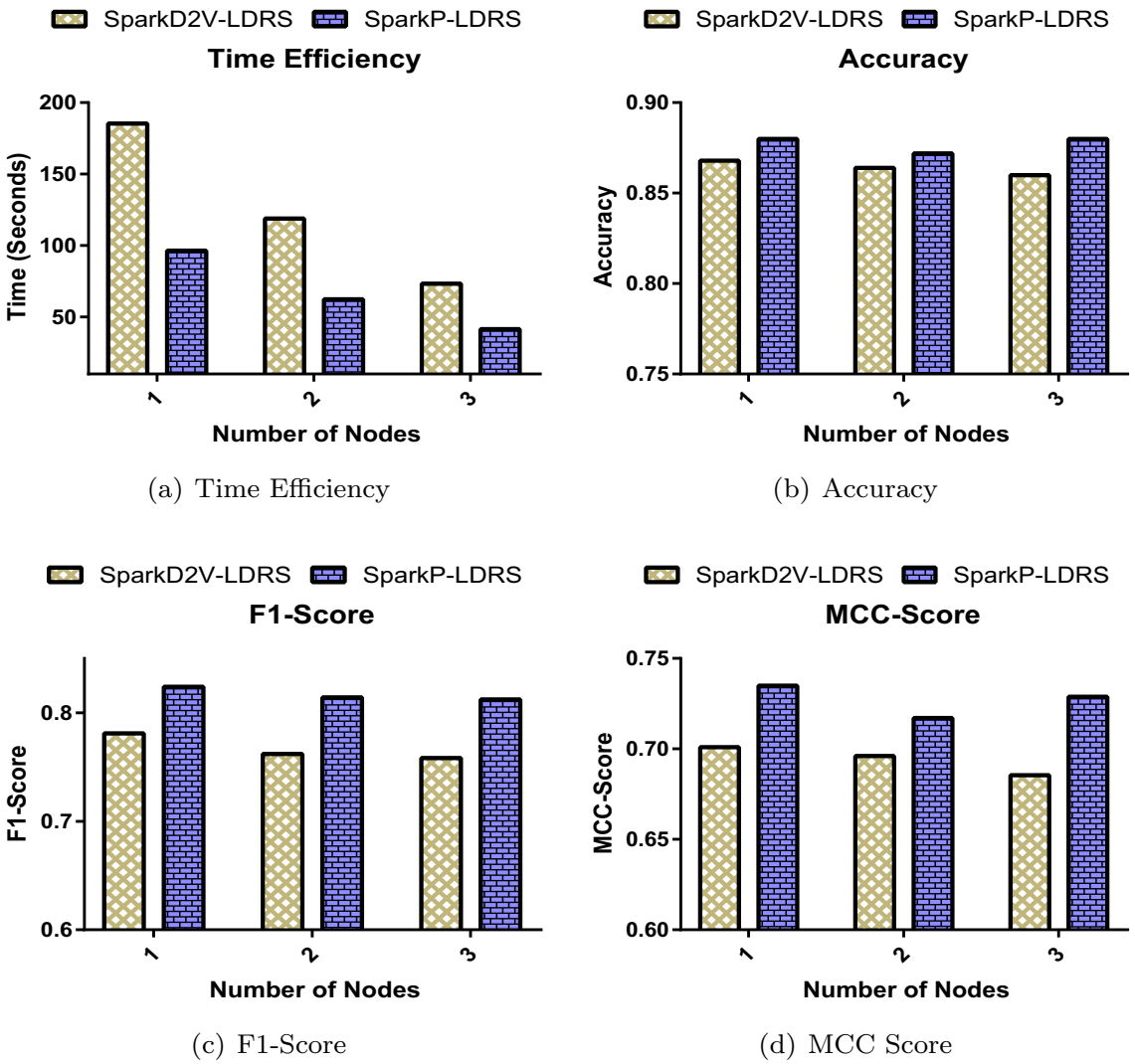
Fig. 5 Time Efficiency and performance of the proposed MapReduce based approaches

W2V. The above experimental results show that the proposed P-LDRS achieves admirable performance using Legal W2V in terms of Accuracy, F1-Scores, and MCC Score of 0.88, 0.73, and 0.73, respectively. Legal W2V has more similar vocabulary and linguistic features compared to Law2Vec and LeGlove, for Indian judgment embedding. The proposed LDRS leverages the priorly learned semantic knowledge extracted from massive Legal data to initialize the word vectors rather than random values.

It is also observed that the Doc2Vec based LDRS possesses the scalability issue due to the existence of an extensive amount of Legal documents, which are continuously increasing in nature. So, massive computational resources are demanded to learn the judgment embedding efficiently. The proposed P-LDRS provides additional functionality to learn the judgment embedding in the distributed environment for mitigating this issue.

The experimental results confirm that the proposed distributed approaches are more time efficient than non-distributed variants. The efficiency is also improving with the increasing number of nodes, as the corpus is divided into smaller partitions with more nodes. This practice reduces the computational load on an individual node, resulting in improved efficiency. The proposed P-LDRS is more time efficient than the D2V-LDRS due to the reduced number of iterations required in the P-LDRS, as shown in Table 1.

Furthermore, it yields stable performance with the increasing number of nodes due to the utilization of Legal domain-specific pre-learned WE, which captures sufficient semantic knowledge even from small data like an individual partition. These promising experimental results imply that the proposed P-LDRS can potentially administrate a large number of judgments with admissible performance. In general, the proposed distributed P-LDRS achieves promising time efficiency, without compromising other performance measures like Accuracy, F1-Scores, and MCC Score. Also, the proposed distributed P-LDRS showcases the significant improvement in time efficiency and scalability without negotiating the performance compared to non-distributed P-LDRS.



(a) Time Efficiency (b) Accuracy

(c) F1-Score (d) MCC Score

Fig. 6 **a** Time efficiency **b** accuracy **c** F1-Score **d** MCC score time efficiency and performance of the proposed spark based approaches

The proposed P-LDRS potentially enhances the semantic quality of the text embedding considering prior domain-specific knowledge such as pre-learned word embedding. Additionally, the proposed distributed P-LDRS can efficiently handle large volumes of textual data over the cluster of multiple computing machines, considering the distributed platforms like MapReduce and Spark. Thus, the proposed distributed approaches are featured with excellent scalability and performance due to practice domain-specific prior knowledge in the distributed environment.

Conclusion

LDRS aims to identify the most relevant judgments to support the Legal professionals, for formulating strategic and beneficial arguments. State-of-the-art existing works have employed the Doc2Vec that effectively learns the semantically rich vector representations for judgments. It has demonstrated superior performance due to an excellent ability to capture semantic relevance. This research thus proposed P-LDRS that enriches the Doc2Vec embedding by leveraging pre-learned WE, which possess the Legal domain-specific semantic knowledge learned from the massive Legal corpus. However, an enormous number of Legal documents result in a scalability issue for P-LDRS, which is addressed by proposed MapReduce and Spark based distributed approaches. Empirical analysis demonstrates the potential superiority of the proposed non-distributed P-LDRS over the traditional Doc2Vec based LDRS by achieving an accuracy of 0.88, F1-Score of 0.83, and MCC-Score of 0.73. Also, the proposed distributed P-LDRS improves the noticeable time efficiency of $\approx$ 60s on two nodes and $\approx$ 41s on three nodes, compared to non-distributed P-LDRS. The proposed distributed

approach achieves a stable Accuracy of ≈ 0.88, F1-Score of ≈ 0.82, and MCC-Score of ≈ 0.72 with an increasing number of nodes. The efficacy and scalability are the prime requirements for the Legal search engines, which require to govern a large number of Legal documents. The outperforming results and time efficiency of the proposed approaches can lead to potential employment in large-scale Legal search engines.

- **Challenges and future work**

Due to the availability of limited resources, the experimentation performed on the small cluster (i.e., three nodes) with the dataset consisted of only judgments from the Supreme Court of India. In a real scenario, other important legislative institutions are producing a large number of Legal documents. A large number of Legal documents from important legislative can be considered over the huge sized cluster of nodes in the future. Moreover, a limited number of Legal domain-specific pre-learned WEs are publicly available. Building effective WEs from the massive amount of Legal documents would also motivate research to support the Legal and AI-based research community. Recent advancements in parallel processing hardware (i.e., GPUs) also encourage NLP growth to empower the application with Deep Neural Networks. In Legal data analytics, considering such Deep Neural network-based transfer learning would be interesting. Furthermore, the present work has not benefited from a hybrid approach (Text- and Citation- based approaches) that can be investigated in the future.

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest

References

1. Blei D, Ng A, Jordan M (2003) Latent dirichlet allocation. J Mach Learn Res 3(Jan):993–1022
2. Chakrabarti D, Patodia N, Bhattacharya U, Mitra I, Roy S, Mandi J, Roy N, Nandy P (2018) Use of artificial intelligence to analyse risk in legal documents for a better decision support. In: TENCON 2018-2018 IEEE region 10 conference, IEEE, pp 683–688
3. Chalkidis I, Kampas D (2019) Deep learning in law: early adaptation and legal word embeddings trained on large corpora. Artificial Intell Law 27(2):171–198
4. Chang LLH, Phoa FKH, Nakano J (2019) A new metric for the analysis of the scientific article citation network. IEEE Access 7:132027–132032
5. Chicco D (2017) Ten quick tips for machine learning in computational biology. BioData Min 10(35):1–17
6. Dean J, Ghemawat S (2008) Mapreduce: simplified data processing on large clusters. Commun ACM 51(1):107–113
7. Dhanani J, Mehta R, Rana D, Tidke B (2018) Sentiment analysis using novel distributed word embedding for movie reviews. In: proceedings of 10th International Conference on Advanced Computing (ICoAC), IEEE, pp 138–145
8. Dhanani J, Mehta R, Rana D (2021) Legal document recommendation system: a cluster based pairwise similarity computation. J Intell Fuzzy Syst 41(5):5497–5509
9. Farhangi A (2018) Legal domain-specific pre-trained word vectors. https://github.com/ashkonf/LeGloVe
10. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp 855–864
11. Guo C, Lu M, Wei W (2019) An improved lda topic modeling method based on partition for medium and long texts. Ann Data Sci pp 1–14
12. Ji S, Satish N, Li S, Dubey P (2016) Parallelizing word2vec in shared and distributed memory. arXiv preprint arXiv:1604.04661
13. Koniaris M, Anagnostopoulos I, Vassiliou Y (2017) Network analysis in the legal domain: a complex model for European Union legal sources. J Complex Netw 6(2):243–268
14. Kumar S, Reddy PK, Reddy VB, Singh A (2011) Similarity analysis of legal judgments. In: Proceedings of the fourth annual ACM Bangalore conference, pp 1–4
15. Kumar S, Reddy PK, Reddy VB, Suri M (2013) Finding similar legal judgements under common law system. In: International Workshop on Databases in Networked Information Systems, Springer, pp 103–116
16. Lau JH, Baldwin T (2016) An empirical evaluation of doc2vec with practical insights into document embedding generation. arXiv preprint arXiv:1607.05368
17. Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: International conference on machine learning, pp 1188–1196
18. Leibon G, Livermore M, Harder R, Riddell A, Rockmore D (2018) Bending the law: geometric tools for quantifying influence in the multinetwork of legal opinions. Artificial Intell Law 26(2):145–167
19. Lodha S, Wagh R (2019) Exploratory analysis of legal case citation data using node embedding. ICIC Express Lett 13(10):883–889
20. Mandal A, Chaki R, Saha S, Ghosh K, Pal A, Ghosh S (2017) Measuring similarity among legal court case documents. In: Proceedings of the 10th annual ACM India compute conference, ACM, pp 1–9
21. Martinčić-Ipšić S, Miličić T, Todorovski L (2019) The influence of feature representation of text on the performance of document classification. Appl Sci 9(4):1–27
22. Mihalcea R, Tarau P (2004) Textrank: Bringing order into text. In: Proceedings of the 2004 conference on empirical methods in natural language processing, pp 404–411
23. Mikolov T, Chen K, Corrado G, Dean J (2013a) Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781
24. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013b) Distributed representations of words and phrases and their compo-

sitionality. In: Advances in neural information processing systems, pp 3111–3119
25. Mou L, Meng Z, Yan R, Li G, Xu Y, Zhang L, Jin Z (2016) How transferable are neural networks in nlp applications? In: Proceedings of the 2016 conference on empirical methods in natural language processing, pp 479–489
26. Nanda R, Adebayo KJ, Di Caro L, Boella G, Robaldo L (2017) Legal information retrieval using topic clustering and neural networks. In: COLIEE@ ICAIL, pp 68–78
27. Ordentlich E, Yang L, Feng A, Cnudde P, Grbovic M, Djuric N, Radosavljevic V, Owens G (2016) Network-efficient distributed word2vec training system for large vocabularies. In: Proceedings of the 25th ACM international on conference on information and knowledge management, pp 1139–1148
28. Patel K, Patel D, Golakiya M, Bhattacharyya P, Birari N (2017) Adapting pre-trained word embeddings for use in medical coding. BioNLP 2017:302–306
29. Raghav K, Reddy PB, Reddy VB, Reddy PK (2015) Text and citations based analysis of legal judgments. In: International Conference on Mining Intelligence and Knowledge Exploration, Springer, pp 449–459
30. Sugathadasa K, Ayesha et al B (2017) Synergistic union of word2vec and lexicon for domain specific semantic similarity. In: 2017 IEEE International conference on industrial and information systems (ICIIS), IEEE, pp 1–6
31. Sugathadasa K, Ayesha B, de Silva N, Perera AS, Jayawardana V, Lakmal D, Perera M (2018) Legal document retrieval using document vector embeddings and deep learning. In: Science and Information Conference, Springer, pp 160–175
32. Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I et al (2010) Spark: Cluster computing with working sets. HotCloud 10(10–10):95