Artificial Intelligence and Law (2022) 30:359–390
https://doi.org/10.1007/s10506-021-09297-1

ORIGINAL RESEARCH

PRILJ: an efficient two-step method based on embedding and clustering for the identification of regularities in legal case judgments

Graziella De Martino[1] · **Gianvito Pio**[1,2] · **Michelangelo Ceci**[1,2,3]

Accepted: 10 July 2021 / Published online: 4 August 2021

Abstract

In an era characterized by fast technological progress that introduces new unpredictable scenarios every day, working in the law field may appear very difficult, if not supported by the right tools. In this respect, some systems based on Artificial Intelligence methods have been proposed in the literature, to support several tasks in the legal sector. Following this line of research, in this paper we propose a novel method, called PRILJ, that identifies paragraph regularities in legal case judgments, to support legal experts during the redaction of legal documents. Methodologically, PRILJ adopts a two-step approach that first groups documents into clusters, according to their semantic content, and then identifies regularities in the paragraphs for each cluster. Embedding-based methods are adopted to properly represent documents and paragraphs into a semantic numerical feature space, and an Approximated Nearest Neighbor Search method is adopted to efficiently retrieve the most similar paragraphs with respect to the paragraphs of a document under preparation. Our extensive experimental evaluation, performed on a real-world dataset provided by EUR-Lex, proves the effectiveness and the efficiency of the proposed method. In particular, its ability of modeling different topics of legal documents, as well as of capturing the semantics of the textual content, appear very beneficial for the considered task, and make PRILJ very robust to the possible presence of noise in the data.

Keywords Legal information retrieval · Embedding · Clustering · Approximate nearest neighbor search

✉ Gianvito Pio
gianvito.pio@uniba.it

Extended author information available on the last page of the article

1 Introduction

The actions of members within a community are usually regulated by an enforced system of rules, that aims to ensure equality, fairness, and justice within the community. When the community is actually a Nation, these binding rules of conduct are usually referred to as the *law*.

Contrary to most of the fields where Computer Science can be considered as a boost for daily activities, the law may appear as a *static* system, that often responds and adapts too slowly. To alleviate this issue, researchers are putting significant effort in designing advanced (also automated) solutions to improve the efficiency of the processes in the legal sector. In this context, a strong contribution may come from the Artificial Intelligence (AI) field. Among the few attempts made in this direction, we can mention the work by Mandal et al. (2017), where the authors applied AI techniques to measure the similarity among legal case documents, which can be useful to speed up the identification and analysis of judicial precedents. Another relevant example is the work by Medvedeva et al. (2020), where the authors consider the semi-automation of some legal tasks, such as the prediction of judicial decisions of the European Court of Human Rights.

Following this line of research, in this paper, we propose an AI method that can support human legal experts during their activity of writing legal case judgments, by exploiting lexical and semantic similarity at two different degrees of granularity. Indeed, legal case judgments can be considered, represented, and analyzed as whole documents, or according to their summaries, paragraphs, sentences, or reasons for citation. In this work, we analyze them as whole documents as well as according to their paragraphs, with the goal of identifying paragraph *regularities* among legal case judgments. In particular, given a (possibly incomplete or under preparation) document, henceforth called *target document*, our system will support the retrieval of paragraphs semantically similar to those of the target document, appearing in the set of reference documents related to previous transcribed legal case judgments. Therefore, paragraph *regularities* refer to the set of retrieved paragraphs, that can significantly support and facilitate the redaction of the target document at hand. Indeed, such paragraphs may provide useful indications about aspects, clauses, or citations that have been reported in contexts similar to that of the target document, and that are expected to be reported also in the target document. In other words, accessing a set of similar paragraphs would provide the legal expert with possible clues on missing pieces of information in the target document, that would possibly deserve to be added to a document under preparation.

Although in the literature we can find several document similarity measures implemented through (a) network-based approaches (Kumar et al. 2011; Minocha et al. 2015), (b) text-based methods (Kumar et al. 2013; Mandal et al. 2017) or (c) hybrid approaches (Kumar et al. 2013), the estimation of the similarity between two legal case documents is still considered a challenging task. Indeed, different themes in legal case documents form different networks of rules that, if considered as a single collection of documents, may lead to inaccurate estimations.

In order to overcome this issue, in this paper we combine the embedding of legal case judgments with a clustering approach, to effectively identify regularities among paragraphs. In particular, we (i) pre-process documents through standard Natural Language Processing (NLP) approaches; *ii)* represent them into a multidimensional semantic feature space, through a document embedding approach based on Neural Networks (NN); (iii) group them through a clustering method, in order to capture similar documents; (iv) learn an embedding model for each cluster, from the paragraphs belonging to their documents. The specific paragraph embedding model, learned from a subset of documents falling into a given cluster, can then be adopted to represent paragraphs (belonging to reference documents or to the target document) into a semantic feature space. Finally, we exploit an efficient strategy to identify paragraph regularities based on Approximated Nearest Neighbor Search (ANNS).

Our two-step approach has the main advantage of learning a different semantic representation for each group of documents (rather than one single model), that allows us to capture peculiarities of paragraphs according to the specific topic. We argue that such peculiarities would not be easily identifiable through a unique representation learned from the whole set of paragraphs. This aspect also allows the proposed two-step approach to be robust to the presence of noise in the data. Note that noise can be in the form of misleading words, e.g., homonyms, or single words that are strongly related to a topic, appearing in paragraphs that are related to a totally different topic. Therefore, it is of utmost importance to be able to capture the right topics of paragraphs, without being affected by the presence of such words.

The rest of the paper is structured as follows. In Sect. 2 we describe previous work related to the present paper, while in Sect. 3 we describe in detail the proposed method. In Sect. 4 we describe our experimental evaluation, and we show and discuss the results. Finally, in Sect. 5 we draw some conclusions and outline possible future work.

2 Related work

In the following subsections, we briefly discuss existing works that support the retrieval of legal information as well as the identification of regularities in legal case judgments by exploiting clustering-based approaches.

2.1 Retrieval of legal information

The retrieval of legal information from existing collections of legal documents can be supported by Information Retrieval (IR) techniques. Indeed, in the literature they have been already profitably used for different tasks. Existing approaches can mainly be categorized into methods that adopt manual knowledge engineering procedures (Brüninghaus and Ashley 2001; Silveira and Ribeiro-neto 2004) and methods that exploit NLP (Biagioli et al. 2005; Tomlinson et al. 2007). In general, NLP-based approaches applied to the legal sector are deemed to be

superior, since they combine data-driven methods and embedding models when analyzing legal case judgments to directly identify legal concepts or different representations, such as tagged feature-value pairs or logical predicates (Maxwell and Schafer 2008; Zhong et al. 2020).

More recently, the big data paradigm, as well as the availability of big data analytics tools, is influencing legal authorities towards the publication of legal case documents through their online databases. On the other hand, researchers in the AI field are seizing this opportunity to enhance existing studies and contribute to the legal informatics field. For example, the work by Kumar et al. (2011); Trompper and Winkels (2016); Shulayeva et al. (2017); Mandal et al. (2017) prove the effort devoted to the development of complex intelligent solutions, to solve tasks such as legal document review, precedent analysis, or document similarity evaluation. Specifically, since various judgments lack semantic annotations, Trompper and Winkels (2016) proposed a method to assign a section hierarchy to Dutch legal case judgments: given a set of unstructured legal case judgments, the authors apply tokenization with linear-chain condition random fields (Sutton and McCallum 2012) to identify and label the roles of textual elements in a legal case judgment. Probabilistic context-free grammars are finally used to organize the text elements into a section hierarchy. Shulayeva et al. (2017) applied a machine learning method to automatically annotate sentences containing legal facts and principles in common law reports. The proposed approach relies on a feature selection step and on a Naïve Bayes multinomial classifier, to classify a given sentence as a principle, a fact or a neutral text. Although satisfactory results were achieved, experiments were limited to a corpus of only 50 reports.

Kumar et al. (2011) and Mandal et al. (2017) also contributed to the enhancement of the precedent analysis task. Both the proposed approaches were based on measuring the similarity among legal case judgments. In particular, Kumar et al. (2011) applied TF-IDF to represent documents, considering all the terms or only legal terms. Moreover, they also adopted the so-called *bibliographic coupling* similarity (that considers common out-citations), and *co-citation similarity* (that considers common in-citations). The authors reported that only the cosine similarity based on legal terms and the bibliographic coupling similarity provided accurate results when identifying similar legal case judgments, but generally observed a limited scalability of the proposed approach. On the other hand, Mandal et al. (2017) applied four different methodologies, namely, TF-IDF, Word2Vec, Doc2Vec, and Latent Dirichlet Allocation, to represent legal case judgments at different levels of granularity, such as summaries, paragraphs, sentences, or reasons for citation.

The adoption of embedding techniques to represent the textual content of legal documents has also been explored in LEGAL-BERT proposed by Chalkidis et al. (2020). LEGAL-BERT uses Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al. 2019), to obtain contextual representations from legal documents. BERT uses a Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. Chalkidis et al. (2020) compared the performance of a general purpose pre-trained BERT model (Devlin et al. 2019), a fine-tuned BERT model with domain-specific corpora, and a BERT model totally learned from scratch from domain-specific corpora. The experiments performed by

Chalkidis et al. (2020) confirmed that the fine-tuned version and the model trained from scratch from domain-specific documents show the best performance.

The same line of research has been explored in the Competition On Legal Information Extraction/Entailment (COLIEE), during which several tasks related to the legal domain have been solved with the support of embedding techniques. Among the approaches related to the present paper, it is worth mentioning BERT-PLI (Shao et al. 2020), that adopts BERT to capture the semantic relationships at the paragraph-level and then infers the relevance between two cases by aggregating paragraph-level interactions. Analogously to LEGAL-BERT, the BERT model in BERT-PLI is fine-tuned with a dataset related to the legal field.

It is noteworthy that, although the exploitation of embedding techniques has already been explored in the literature, to the best of the authors knowledge, the method proposed in this paper is innovative in the following aspects: *i)* it is the first method in the literature that exploits a two-step approach, based on clustering, to analyze textual content at document and paragraph levels in the legal domain; *ii)* it exploits an *efficient* Approximated Nearest Neighbor Search (ANNS) method to identify paragraph regularities; *iii)* it is much more robust to the presence of noise in the data, with respect to both baseline and state-of-the-art solutions, as we will show in our empirical evaluation (see Sect. 4.3).

2.2 Clustering-based approaches in the legal sector

Clustering generally refers to an unsupervised task consisting in grouping similar objects into clusters. More specifically, similar objects should fall into the same cluster, while dissimilar objects should fall into different clusters. Together with the design of advanced clustering algorithms (Berkhin 2002; Ester et al. 1996; Pio et al. 2012; Corizzo et al. 2019), the most critical research aspect of clustering is in the design of a proper representation of the objects/items at hand (Mikolov et al. 2013; Le and Mikolov 2014), as well as of similarity measures which allow the algorithms to understand how much two objects are similar/dissimilar.

Clustering techniques have also been adopted in the legal field. Despite the fact that legal documents are highly abstract, researchers managed to group similar legal documents on the basis of their topics, or on the basis of case citations and legal citations (Conrad et al. 2005; Lu et al. 2011; Raghav et al. 2015; Kachappilly and Wagh 2018).

Conrad et al. (2005) adopted a clustering tool (Zhao et al. 2005) to apply both hard and soft clustering on three large heterogeneous datasets to effectively generate a taxonomy while supporting legal firms in their knowledge management processes. Particularly promising results were achieved when adopting hierarchical clustering methods, where clusters are organized in a hierarchy, or overlapping clustering methods, where each document can possibly belong to multiple clusters.

Lu et al. (2011) reports a successful and scalable implementation of a soft clustering algorithm that is based on topic segmentation. Contrary to typical approaches based on lexical similarity, the authors exploit topics, document citations, and

click-stream data from user behavior databases, to obtain a high-quality classification similar to that achieved by human legal experts.

Raghav et al. (2015) exploited citations and paragraph links to cluster legal case judgments from the Supreme Court of India, aiming to build efficient search engines. The authors used regular expressions to extract citations. Moreover, they define links between pairs of paragraphs belonging to different judgments showing a cosine similarity, computed on the basis of their TF-IDF representation, higher than a given threshold. A new clustering algorithm was also proposed, based on the Jaccard coefficient, and cluster prototypes were defined by selecting the legal case judgment exhibiting the highest similarity with the other legal case judgments within the cluster. Analogously, also Kachappilly and Wagh (2018) used case citations when clustering legal case judgments from the Indian Constitution. The proposed approach transforms the dataset into a binary matrix, indicating the presence or the absence of a citation of each case. Subsequently, the dataset is partitioned into groups through the classical k-means algorithm, using the Euclidean distance.

Although there are several works in the literature that considered the task of measuring the similarity among legal documents, and possibly identifying clusters thereof, the method proposed in this paper can be considered the first that simultaneously exploits advanced embedding techniques to capture the semantics and the context from the text. As mentioned in Sect. 1, these techniques are then used both in a two-step model to analyze the textual content at both document and paragraph level, and in an ANNS method to efficiently retrieve the most similar paragraphs with respect to a document at hand.

3 Methodology

In this section, we describe our method, called PRILJ (Paragraph Regularities Identification in Legal Judgments), that combines a clustering technique applied at a document level with an embedding approach applied at both document and paragraph levels. Embedding techniques have found several applications in the literature (Grover and Leskovec 2016; Corizzo et al. 2020; Pio et al. 2020; Ceci et al. 2020), and actually aim at representing any kind of structured and unstructured data as a numerical feature vector, so that existing retrieval, data mining and machine learning methods that work on classical feature vector representation can be adopted.

The methodological contribution of our approach comes from its ability to identify regularities by also exploiting common themes/topics of groups of legal case documents, as well as their possibly common/similar case citations or legal citations. Moreover, once we represent documents and paragraphs in a semantic feature space through embedding techniques, we exploit a smart strategy to identify the most similar paragraphs that overcomes the bottleneck usually introduced by cosine-based pairwise similarity comparisons (Mandal et al. 2017; Thenmozhi et al. 2017). This strategy makes our approach not only accurate but also very efficient.

Before describing PRILJ in detail, we provide some useful definitions to ease the understanding:

- *Training set* $\boldsymbol{D_T}$: a collection of legal case judgments, represented as textual documents, adopted to train our models;
- *Reference set* $\boldsymbol{D_R}$: a collection of legal case judgments, represented as textual documents, from which we are interested to identify paragraph regularities;
- *Target document* $\boldsymbol{d}$: a legal case judgment (possibly incomplete or under preparation) about which we are interested to identify paragraph regularities from the reference set.

The training set and the reference set may possibly fully (or partially) overlap i.e., $D_T = D_R$ (or $D_T \cap D_R \neq \emptyset$), namely, the set of documents adopted to train our models may be the same as (or overlap with) the collection from which we want to identify paragraph regularities with respect to the target document. Note that PRILJ is fully unsupervised and the target document d is never contained in either the training set or in the reference set (i.e., $d \notin (D_T \cup D_R)$).

Our method PRILJ consists of three main phases, namely:

Training phase (see Fig. 1 and Algorithm 1), during which PRILJ *i)* trains a document embedding model from D_T, that is able to represent **documents** into a semantic feature space; *ii)* identifies k groups of documents in D_T according to their semantic representation, through a clustering method; *iii)* learns k paragraph embedding models, one for each cluster, that are able to represent **paragraphs** into a semantic feature space.

Paragraph embedding of the reference set (see Fig. 2 and Algorithm 2), that exploits both the document embedding model and the k paragraph embedding models learned during the training phase, to identify a semantic representation of all the paragraphs of the reference set.

Identification of paragraph regularities (see Fig. 3 and Algorithm 3), that exploits the identified document clusters, the document embedding model and the k paragraph embedding models to evaluate, through an efficient strategy, the similarity among paragraphs. The purpose is to identify paragraphs from the reference set that appear related to those of the target document (possibly under preparation).

In the remainder of this section, we will describe these three phases.

Following Algorithm 1, we now provide the details of the training phase. The algorithm starts with the application of some pre-processing steps to the documents in D_t (line 2). In details, the pre-processing consists of: *i)* lowercasing the text, *ii)* removing punctuation and digits, *iii)* applying lemmatization, and *iv)* removing rare words. The pre-processed documents are then used to train a document embedding model M (line 3), that is subsequently exploited to represent each document of the training set D_T in the latent feature space, obtaining the set of embedded training documents E_T (lines 4–7). Such documents are then partitioned into k clusters $[C_1, C_2, ..., C_k]$ by adopting the k-means clustering algorithm (line 8). Each cluster of documents becomes the input for a further learning step at the paragraph level: documents falling in the same cluster will contribute to the learning of a specific paragraph embedding model. Algorithmically, for each

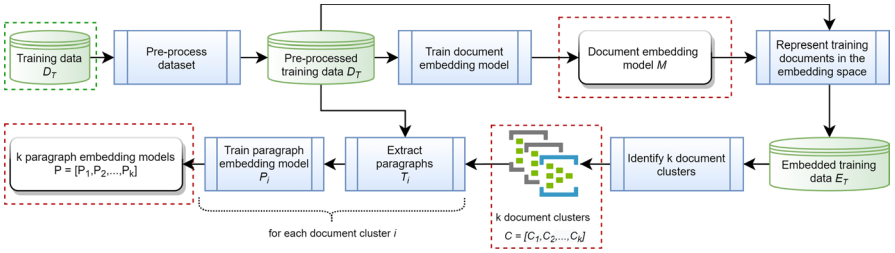

Fig. 1 Graphical overview of the training phase. Green- and red-dotted rectangles represent inputs and outputs, respectively. (Color figure online)

Algorithm 1: Training phase

Data:
- D_T: training documents of legal court cases;
- k: number of desired document clusters.

Result:
- M: document embedding model;
- $C = [C_1, C_2, \ldots, C_k]$: k document clusters;
- $P = [P_1, P_2, \ldots, P_k]$: k paragraph embedding models.

begin
  $D_T \leftarrow preprocessText(D_T)$;
  /* Train an embedding model for documents */
  $M \leftarrow trainEmbeddingModel(D_T)$;
  /* Represent training documents in the embedding space */
  $E_T \leftarrow \emptyset$;
  foreach $doc \in D_T$ **do**
    $E_T \leftarrow E_T \cup \{M.embed(doc)\}$;
  end
  /* Identify k document clusters through k-means */
  $C \leftarrow$ k-means(E_T, k);
  /* Build a paragraph embedding model for each document cluster */
  $P \leftarrow [\,]$;
  for $i \leftarrow 1$ **to** k **do**
    $T_i \leftarrow extractParagraphs(C_i)$;
    $P_i \leftarrow trainEmbeddingModel(T_i)$;
  end
  return $\langle M, C, P \rangle$;
end

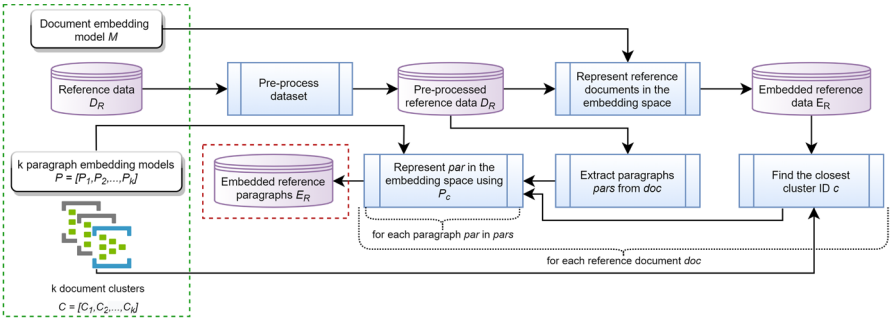

Fig. 2 Graphical overview of the paragraph embedding of the reference set. Green- and red-dotted rectangles represent inputs and outputs, respectively. (Color figure online)

Algorithm 2: Paragraph embedding of the reference set

Data:
- D_R: reference documents of legal court cases;
- M: document embedding model;
- $C = [C_1, C_2, \ldots, C_k]$: k document clusters;
- $P = [P_1, P_2, \ldots, P_k]$: k paragraph embedding models.

Result:
- E_R: set of embedded paragraphs from reference documents.

```
1  begin
2      D_R ← preprocessText(D_R);
       /* Set of embedded paragraphs from reference documents */
3      E_R ← ∅;
       /* For each reference document */
4      foreach doc ∈ D_R do
           /* Represent it in the document embedding space */
5          e_doc ← M.embed(doc);
           /* Find the id of the closest document cluster */
6          c ← getClosestClusterID(C, e_doc);
           /* Extract paragraphs from the document */
7          pars ← extractParagraphs(doc);
           /* Represent its paragraphs using paragraph embedding model P_c
              associated to the cluster c */
8          foreach par ∈ pars do
9              e_par ← P_c.embed(par);
10             E_R ← E_R ∪ {e_par}
11         end
12     end
13     return E_R;
14 end
```

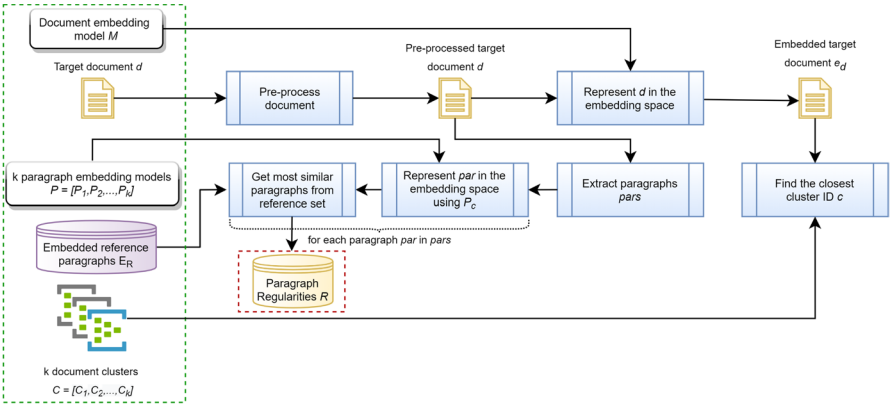

Fig. 3 Graphical overview of the identification of paragraph regularities. Green- and red-dotted rectangles represent inputs and outputs, respectively. (Color figure online)

Algorithm 3: Identification of paragraph regularities

Data:
- d: target document of a legal court case;
- M: document embedding model;
- $C = [C_1, C_2, \ldots, C_k]$: k document clusters;
- $P = [P_1, P_2, \ldots, P_k]$: k paragraph embedding models;
- E_R: set of embedded paragraphs from reference documents;
- n: desired number of most similar paragraphs for each paragraph of d.

Result:
- R: paragraph regularities identified for the target document d.

begin
  $d \leftarrow preprocessText(d)$;
  /* Represent d in the document embedding space */
  $e_d \leftarrow M.embed(d)$;
  /* Find the id of the closest document cluster */
  $c \leftarrow getClosestClusterID(C, e_d)$;
  /* Extract paragraphs from the target document */
  $pars \leftarrow extractParagraphs(d)$;
  $R \leftarrow \emptyset$;
  /* For each paragraph */
  foreach $par \in pars$ **do**
    /* Represent the paragraph using paragraph embedding model P_c associated to the cluster c */
    $e_{par} \leftarrow P_c.embed(par)$;
    /* Get the n most similar paragraphs from the reference set */
    $R \leftarrow R \cup getMostSimilarParagraphs(e_{par}, E_R, n)$;
  end
  return R;
end

document cluster $C_i, 1 \leq i \leq k$, we extract the paragraphs from the documents falling into C_i (line 11) and train a paragraph embedding model P_i (line 12).

The embedding models, both at the document level and at the paragraph level, are learned through neural network architectures based on Word2Vec (Mikolov et al. 2013) and Doc2Vec (Le and Mikolov 2014). Such approaches, originally proposed to embed words and documents, respectively, can fruitfully be adopted to represent legal court case documents and their paragraphs. Previous works demonstrated the superiority of Word2Vec and Doc2Vec over classical counting-based approaches, since they take into account both the syntax and semantics of the text (Donghwa et al. 2018; Mandal et al. 2017). In addition, their ability to catch the semantics and the context of single words and paragraphs allow them to properly represent new (previously unseen) documents which features have not been explicitly observed during the training phase. On the contrary, a purely counting-based syntactic approach would fail to represent a document which words have never been observed in the training collection. Further details on the Word2Vec and Doc2Vec architectures implemented in PRILJ are provided in Sects. 3.1 and 3.2.

Following Algorithm 2 we now describe the embedding of the reference set. Analogously to the training phase, we pre-process the documents of the reference set D_r (line 2). Then, each document of the reference set is embedded using the previously learned document embedding model M (line 5). The embedded representation of the document is then used to identify the closest document cluster id (i.e., c at line 6) that corresponds to the optimal paragraph embedding model (i.e., P_c) to adopt in the embedding of its paragraphs (lines 7-10). The set of all the embedded paragraphs E_R is finally returned by the algorithm, from which we are interested to identify regularities for a given target document d. We stress the fact that this two-step strategy allows us to model both general patterns at the document level and specific patterns at the paragraph level, possibly leading to an improved representation and, accordingly, to more accurate identification of paragraph regularities.

The identification of paragraph regularities, described in Algorithm 3, starts by following the same steps mentioned in Algorithm 2 to represent each paragraph of the target document d in the paragraph embedding space. Namely, the most proper paragraph embedding model is adopted to embed its paragraphs, selected according to the closest document cluster with respect to d. For each embedded paragraph, we finally identify the top-n most similar paragraphs from the set of embedded paragraphs E_R belonging to the reference set (line 9). Their identification could straightforwardly be based on the computation of vector-based similarity/distance measures (e.g., cosine similarity, Euclidean distance, etc.) between the embedded paragraphs of the target document d and all the embedded paragraphs of the reference set E_r. Such a pairwise comparison would be computationally intensive and would lead to inefficiencies during the adoption of the proposed system in a real-world scenario. To overcome this issue, we adopt a more advanced method for the identification of the top-n most similar paragraphs, based on random projections.

In the following subsections, we provide additional details about two different models that we adopt for document and paragraph embedding (Word2Vec and DocVec), as well as about the approach we propose to efficiently identify the top-n most similar paragraphs.

3.1 Learning document and paragraph embedding models through Word2Vec

Word2Vec (Mikolov et al. 2013) is a word embedding method, namely a method to represent words as numerical feature vectors. Word2Vec learns a model to embed words by analyzing a collection of documents and by exploiting two different neural network architectures: Continuous-Bag-of-Words (CBOW) and Skip-gram (SG). Both architectures can capture rich syntactic and semantic relationships between words, but they are based on two different techniques: CBOW adopts a feed-forward neural network to predict a target (central) word from a given (surrounding) context, while SG aims to predict the surrounding words of a given target word. The first is generally faster to train and usually provides slightly better accuracy for frequent words, while the second is more accurate in the representation of rare words, at the price of a generally higher running time. However, previous experiments provided even discordant conclusions, depending on the specific datasets (Jin and Schuler 2015; Miñarro-Giménez et al. 2015).

In PRILJ, we adopt the variant based on CBOW, also because its learning process is conceptually closer to our final goal. In fact, a feed-forward neural network which can predict a target word from a given (surrounding) context well adapts to the task of identifying words and paragraphs to suggest while writing a document (according to the current context). This is not the case SG, where the task is rather different (predicting the context given a word).

Methodologically, given a sequence of words $\langle w_{t-j}, ..., w_t, ..., w_{t+j} \rangle$, representing the target word w_t and its context of size $2j$, Word2Vec first maps each context word w_i to a one-hot vector representation $\mathbf{w_i}$ of size V, where V corresponds to the size of the vocabulary observed in the collection of documents. Each element of the vector corresponds to one word of the vocabulary and a generic word is then represented by a vector of 0s for all the vector values, except the value corresponding to the specific word, which is 1.

The neural network architecture aims to learn the optimal matrix $S \in \mathbb{R}^{V \times E}$, where E is the desired size of the embedding space. The one-hot vectors of the context words are then multiplied by S. The obtained $2j$ vectors in the space $\mathbb{R}^E$ are averaged by the hidden layer to obtain the embedding of the target word w_t. Formally, the hidden layer is computed as:

$$h = \sum_{w_i \in \{w_{t-j}, ..., w_{t-1}, w_{t+1}, ..., w_{t+j}\}} \mathbf{w_i} \cdot S \tag{1}$$

The output layer, obtained by multiplying the embedding of the target word w_t by $S^\top$, corresponds to the one-hot vector of w_t (see Fig. 4). This means that the neural network is learned so that it accurately reconstructs the one-hot vector of the target word w_t, given the one-hot vectors of the context words.

Once the neural network has been trained using the training set, the obtained matrix S can be used to embed a given word into a numerical feature space of size E.

In our case, we learn a document embedding model from the training documents D_T (Algorithm 1, line 3), as well as k paragraph embedding models, one for each group of documents identified through k-means (Algorithm 1, line 12). An

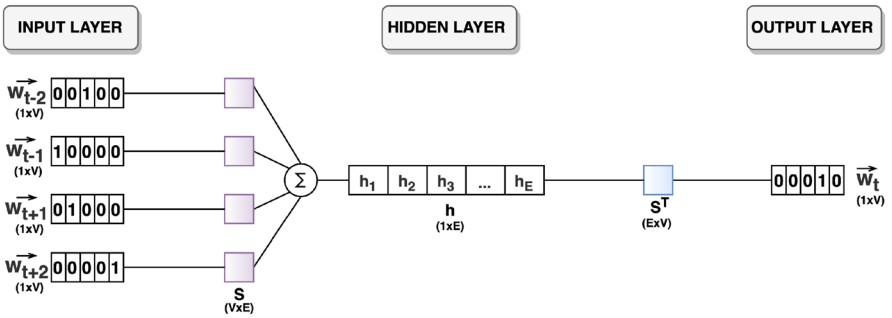

Fig. 4 Graphical representation of the Word2Vec CBOW neural network architecture implemented in PRILJ. Note that there is only one matrix *S*, that is repeated multiple times in the figure only for explanatory purposes. (Color figure online)

embedding model learned by Word2Vec can be queried for several purposes, but it natively provides an embedding for single words which represents the semantic of the word based on its context. In order to obtain an embedding for sequences of words (that may correspond to paragraphs or whole documents, in our case), different aggregation strategies can be adopted, including sum and mean, as suggested by Le and Mikolov (2014). In PRILJ, we obtain an embedding for the documents of the reference set (Algorithm 2, line 5) and for the target document (Algorithm 3, line 3), as well as for paragraphs of the reference set (Algorithm 2, line 9) and of the target document (Algorithm 3, line 8). For these purposes, we adopted the mean of the embeddings of the words.

3.2 Learning document and paragraph embedding models through Doc2Vec

Although Word2Vec can in principle be used to represent sequences of words, by adopting the mentioned aggregation strategies, it was not originally designed for this purpose. Consequently, Le and Mikolov (2014) proposed Doc2Vec, that is natively able to generate a vector representation of word sequences, where a sequence can be either a paragraph or a whole document.

Methodologically, Doc2Vec can exploit two different architectures, namely distributed memory (PV-DM) and distributed bag of words (PV-DBOW). Similar to CBOW, PV-DM aims to predict the word w_t, given its context. However, the context is represented not only by the one-hot vector representations of its surrounding words, but also by a *C*-dimensional one-hot vector representation (**seq**) of the unique sequence ID (*seq*), where *C* is the total number of sequences. This vector encapsulates the topic shared by words in the same sequence. Conversely, PV-DBOW makes use of the SG architecture, where the one-hot vector representation of the unique ID associated with the sequence is fed to the input layer instead of the one-hot vector of w_t.

For the same motivations for the adoption of CBOW in Word2Vec, in PRILJ, we adopt the PV-DM architecture in Doc2Vec, as shown in Fig. 5. The main differences with respect to the architecture shown in Fig. 4 are *i)* the presence of the

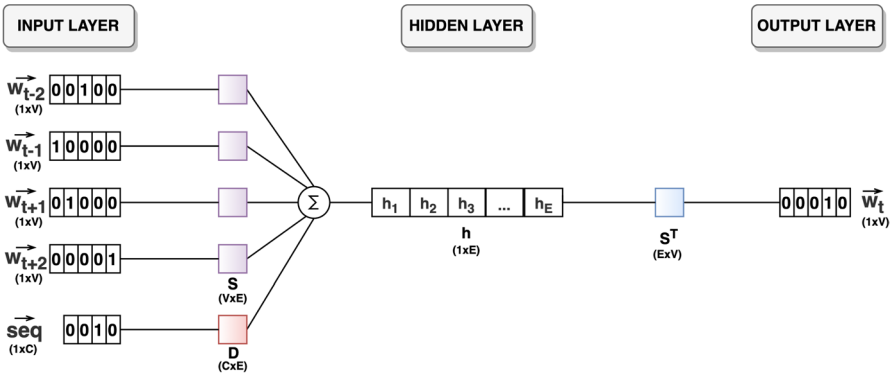

Fig. 5 Graphical representation of the Doc2Vec PV-DM neural network architecture implemented in PRILJ. Note that there is only one matrix *S*, that is repeated multiple times in the figure only for explanatory purposes. (Color figure online)

C-dimensional one-hot vector (**seq**) in the input layer associated to the sequence ID (*seq*), and *ii)* the additional matrix $D \in \mathbb{R}^{C \times E}$, which values are optimized together with those of the matrix *S*. Formally, in this case, the hidden layer is computed as:

$$h = \sum_{w_i \in \{w_{t-j},\ldots,w_{t-1},w_{t+1},\ldots,w_{t+j}\}} \mathbf{w_i} \cdot S + \mathbf{seq} \cdot D \tag{2}$$

Analogously to the adoption of Word2Vec, we learn a document embedding model and *k* paragraph embedding models, and exploit them to embed documents and paragraphs, respectively. The main difference is that, in this case, we do not need any aggregation step to obtain the embedding of sequences of words from the embedding of single words.

3.3 Approximated Nearest Neighbour Search (ANNS) for the identification of paragraph regularities

In this subsection, we describe the strategy adopted in PRILJ to efficiently identify the top-*n* most similar paragraphs of the reference set, with respect to the paragraphs of the target document. A straightforward approach would consist in the computation of the pairwise cosine similarity between the vector representation of the paragraphs. However, such an approach would be computationally intensive. Namely, its time complexity would be $O(n_r)$ for each paragraph of the target document, where n_r is the number of paragraphs of the reference set.

To deal with this computational issue, we propose an approach based on Annoy (Bernhardsson 2015), where the idea is to perform an approximated nearest neighbour search (ANNS). Methodologically, we perform two phases, i.e., *index*

construction on the paragraphs of the reference set, and *search*, that occurs when we actually need to identify the top-n most similar paragraphs with respect to a paragraph of the target document. During the index construction, we build T binary trees. Each tree is built by partitioning the input set of vectors recursively, by randomly selecting two vectors and defining a hyperplane that is equidistant from them (see Fig. 6). It is noteworthy that even if based on random partitioning, vectors that are close to each other in the feature space are more likely to appear close to each other in the tree. It can be proved that this indexing step has a computational cost of $O(T \times log_2(n_r)) = O(log_2(n_r))$.

During the search process, we traverse the binary trees by exploiting a priority queue. Specifically, each tree is recursively traversed, and the priority of each split node is defined according to the distance to the query vector (that is a paragraph of the target document, in our case). This process leads to the identification of T leaf nodes, where the query vector falls into. The distance between the query vector and the set of vectors falling into the identified leaves is finally exploited to return the top-n most similar paragraphs (Li et al. 2016). Computationally, also the search process takes $O(T \times log_2(n_r)) = O(log_2(n_r))$.

Despite the fact that the results may not be identical to those of the exact search, previous experiments showed that its ability to face the curse of dimensionality leads to high-quality approximations, together with much higher efficiency.

In PRILJ, the index construction is performed once, on the set of all the paragraphs belonging to the reference set E_R, while the search process is carried out for each paragraph of the target document (Algorithm 3, line 9).

4 Experiments

In this section, we describe the experimental evaluation we performed to assess the effectiveness of the proposed method PRILJ. Specifically, in Sect. 4.1, we describe the considered real-world dataset, while in Sect. 4.2 we describe in detail the experimental setting and the considered comparative evaluations. Finally, in Sect. 4.3, we show and discuss the obtained results.

4.1 Dataset

In our experiments, we use a dataset made available by EUR-Lex[1] which, excluding empty documents, consists of 4,181 official public EU legal documents, having an average length of 2,739 words, related to both unconsolidated and finalized legal

[1] https://eur-lex.europa.eu/homepage.html.

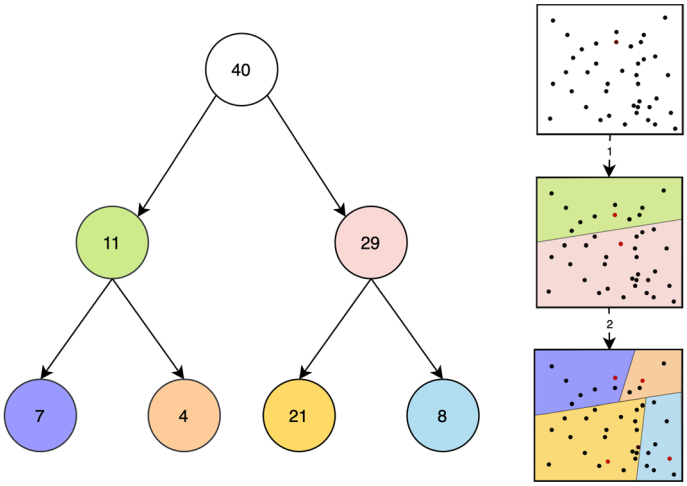

Fig. 6 Graphical representation of the construction of the index performed by the adopted ANNS approach. The value in each node represents the number of vectors falling in that node. Red dots on the right represent the randomly selected vectors for the identification of the separating hyperplane. (Color figure online)

case judgments from 2008 to 2018. The total number of paragraphs in the dataset is 530,744, with an average of 22 words per paragraph. Each legal case judgment has a unique CELEX number which components are the EUR-Lex sector, the year, the document type and the document number. The CELEX number was used to fetch the legal case judgments. We extracted the textual content by ignoring HTML elements, while the $< p >$ tag was used to identify the paragraphs. We applied the preprocessing steps mentioned in Sect. 3. We ignored words having a document frequency lower than 3, and we retained only paragraphs having at least 10 words.

Note that the considered dataset falls within the case-law sector and includes only legal case judgments by the Courts of Justice. Therefore, it does not include views delineated by the Advocate General and opinions on draft agreements given by the European Court.

4.2 Experimental setting

All the experiments were performed in a 10-fold cross-validation (10-fold CV) setting, where 90% of the dataset is considered as training set and the remaining 10% of the dataset is considered as testing set, alternatively for 10 times. All the documents of the testing set were considered as target documents, while the reference set was built by constructing 20 replicas of each paragraph of the documents in the testing set, perturbed by introducing a controlled amount of noise. In particular, the noise was introduced by replacing a given percentage of words of each paragraph with random words selected from the Oxford dictionary[2]. In our experiments, we considered different levels of noise, namely, 10%, 20%, 30%, 40%, 50%, and 60%,

[2] https://raw.githubusercontent.com/cduica/Oxford-Dictionary-Json/master/dicts.json.

in order to evaluate the robustness of the proposed approach to different amounts of noise. We stress the importance of specifically evaluating this aspect, since noise (e.g., homonyms or misleading words) can be easily present in textual documents, and a robust approach should provide accurate results also when input documents are affected by potentially high amounts of noise.

In order to assess the specific contribution of the adopted embedding strategies, we compared the results obtained through Word2Vec and Doc2Vec with those achieved using a baseline strategy, i.e., the classical TF-IDF approach. In all the cases, we adopted a 50-dimensional feature vector. For TF-IDF, we selected the top-50 words showing the highest frequency across the set of legal case judgments.

We evaluated the performance of the two-step model implemented in PRILJ with different numbers of clusters, i.e., $k \in \{\sqrt{|D_T|}/2, \sqrt{|D_T|}, \sqrt{|D_T|} \cdot 2\}$. Note that $k = \sqrt{|D_T|}$ is generally considered a default value for the number of clusters, when this has to be manually specified. In our analysis we also evaluate the sensitivity of PRILJ to the value of k.

Moreover, we compared the observed performance with that obtained by a baseline strategy that does not group training documents into clusters (henceforth denoted as *one-step model*).

We also performed an additional comparative analysis with state-of-the-art competitor systems. Specifically, we compared PRILJ with:

- **LEGAL-BERT-BASE**, that is the LEGAL-BERT model[3] fine-tuned by Chalkidis et al. (2020) using a wide set of legal documents related to EU, UK and US law;
- **LEGAL-BERT-SMALL**, that is the LEGAL-BERT model[3] fine-tuned by Chalkidis et al. (2020) using the same set of documents adopted for LEGAL-BERT-BASE, but in a lower-dimensional embedding space;
- **LEGAL-BERT-EURLEX**, that is the LEGAL-BERT model[3] fine-tuned by Chalkidis et al. (2020) using the EUR-LEX dataset;
- **BERT-PLI**, that is the system BERT-PLI[4] based on BERT, fine-tuned with a small set of legal documents, proposed by Shao et al. (2020) in the Competition On Legal Information Extraction/Entailment (COLIEE).

Note that the above-mentioned competitors are able to represent paragraphs as feature vectors (i.e., they are embedding models), taking into account the semantics and the context of the textual content. Specifically, LEGAL-BERT-BASE, LEGAL-BERT-EURLEX and BERT-PLI represent paragraphs in a 768-dimensional feature space, while LEGAL-BERT-SMALL represents paragraphs in a 512-dimensional feature space. The embedding of each paragraph was computed as the mean of the embedding of its tokens.

Finally, we evaluated the effectiveness and the efficiency of the approach implemented in PRILJ for the identification of the *top*-n most similar paragraphs based on

[3] https://huggingface.co/nlpaueb/legal-bert-base-uncased.
[4] https://github.com/sophiaalthammer/bert-pli.

ANNS, with $T = 100$ (number of trees). Specifically, we performed an additional comparative analysis against a non-approximated solution based on the cosine similarity, on a subset of 100 documents randomly selected from the dataset. This analysis was performed considering the best configuration in terms of the number of clusters k, and also focused on evaluating the advantages in terms of computational efficiency.

As evaluation measures, we collected precision@n, recall@n and f1-score@n, averaged over the paragraphs of target documents and over the 10 folds, with $n \in \{5, 10, 15, 20, 50, 100\}$. Specifically, for each paragraph of a target document in the testing set, we considered as True Positives the number of correctly retrieved (perturbed) replicas from the reference set.

In summary, our experimental evaluation was performed along multiple dimensions of analysis, i.e., on the evaluation of: (i) the effect of different amounts of noise in the data, to evaluate the robustness of PRILJ to the presence of noise; (ii) the contribution of the embedding approaches implemented in PRILJ that also catch the semantics, with respect to the adoption of TF-IDF; (iii) the contribution provided by the two-step model with different numbers of clusters, with respect to the one-step model that does not exploit document clustering; (iv) the effect of the approximated nearest neighbor approach implemented in PRILJ, both in terms of effectiveness and in terms of efficiency.

4.3 Results

In Tables 1, 2 and 3, we report the precision@n, the recall@n, and the f1-score@n results, respectively, measured with different embedding strategies and different levels of noise introduced in the dataset. The upper-left subtable shows the results obtained with the one-step model, while the other subtables show the results obtained by PRILJ with different numbers of clusters.

As expected, we can observe that in all the configurations the presence of noise negatively affects the results: the higher the amount of noise introduced, the lower the precision@n, the recall@n, and the f1-score@n. It is noteworthy that there is no difference in terms of conservativeness among all the considered approaches, namely, approaches obtaining a higher precision@n also obtain a higher recall@n and, accordingly, a higher f1-score@n. This interesting result allows us to outline clear conclusions (reported in the following of the section) about the most effective approaches (and their parameters) for the different phases implemented in PRILJ.

First, we can observe that, although the baseline based on TF-IDF obtained acceptable results, the adoption of the embedding methods implemented in PRILJ is significantly beneficial. Specifically, when adopting Doc2Vec, we observe an average improvement of 17.68% for the precision@n, 17.71% for the recall@n, and 17.70% for the f1-score@n. On the other hand, when adopting Word2Vec, we observe an average improvement of 24.22% for the precision@n, 24.27% for the recall@n, and 24.29% for the f1-score@n. This result confirms our initial intuition that catching the context and the semantics leads to significant improvements. Moreover, although Doc2Vec is natively able to work with word sequences, Word2Vec

Table 1 Precision@n results obtained with different embedding strategies (T=TF-IDF; D=Doc2Vec; W=Word2Vec) and different levels of noise

		One-step model					
		Noise %					
		10%	20%	30%	40%	50%	60%
p@5	**T**	0.753	0.722	0.660	0.540	0.338	0.114
	D	0.930	0.910	0.882	0.782	0.553	0.301
	W	**0.936**	**0.925**	**0.911**	**0.887**	**0.827**	**0.690**
p@10	**T**	0.758	0.701	0.604	0.450	0.244	0.075
	D	0.922	0.897	0.849	0.697	0.453	0.241
	W	**0.935**	**0.922**	**0.904**	**0.870**	**0.783**	**0.615**
p@15	**T**	0.742	0.657	0.534	0.368	0.189	0.058
	D	0.911	0.877	0.788	0.599	0.379	0.205
	W	**0.932**	**0.917**	**0.894**	**0.841**	**0.726**	**0.543**
p@20	**T**	0.696	0.587	0.456	0.305	0.154	0.047
	D	0.885	0.815	0.687	0.510	0.327	0.180
	W	**0.927**	**0.904**	**0.861**	**0.780**	**0.648**	**0.475**
p@50	**T**	0.317	0.278	0.221	0.149	0.077	0.024
	D	0.383	0.367	0.330	0.266	0.186	0.113
	W	**0.390**	**0.387**	**0.380**	**0.362**	**0.322**	**0.254**
p@100	**T**	0.167	0.149	0.121	0.084	0.044	0.015
	D	0.194	0.189	0.177	0.151	0.114	0.075
	W	**0.197**	**0.196**	**0.194**	**0.188**	**0.173**	**0.145**
		Two-step model - $k = \sqrt{\lvert D_T \rvert}/2$					
		Noise %					
		10%	20%	30%	40%	50%	60%
p@5	**T**	0.870	0.853	0.825	0.772	0.673	0.492
	D	0.946	0.933	0.921	0.905	0.873	0.786
	W	**0.952**	**0.945**	**0.935**	**0.920**	**0.891**	**0.829**
p@10	**T**	0.874	0.846	0.798	0.721	0.593	0.402
	D	0.940	0.925	0.910	0.886	0.831	0.703
	W	**0.951**	**0.942**	**0.930**	**0.911**	**0.871**	**0.784**
p@15	**T**	0.867	0.821	0.753	0.653	0.514	0.338
	D	0.933	0.915	0.893	0.853	0.765	0.615
	W	**0.949**	**0.938**	**0.924**	**0.897**	**0.839**	**0.726**
p@20	**T**	0.835	0.764	0.678	0.573	0.444	0.291
	D	0.918	0.889	0.847	0.779	0.675	0.533
	W	**0.944**	**0.930**	**0.905**	**0.859**	**0.776**	**0.652**
p@50	**T**	0.369	0.353	0.328	0.290	0.235	0.165
	D	0.392	0.388	0.381	0.366	0.334	0.280
	W	**0.395**	**0.393**	**0.391**	**0.385**	**0.369**	**0.334**
p@100	**T**	0.189	0.184	0.175	0.160	0.136	0.103
	D	0.198	0.197	0.195	0.190	0.179	0.158
	W	**0.199**	**0.199**	**0.198**	**0.197**	**0.193**	**0.182**

Table 1 (continued)

		Two-step model - $k = \sqrt{\lvert D_T \rvert}$					
		Noise %					
		10%	20%	30%	40%	50%	60%
p@5	**T**	0.886	0.871	0.846	0.800	0.715	0.559
	D	0.949	0.938	0.926	0.911	0.885	0.818
	W	**0.955**	**0.949**	**0.940**	**0.927**	**0.901**	**0.847**
p@10	**T**	0.890	0.866	0.824	0.755	0.642	0.470
	D	0.944	0.930	0.915	0.894	0.848	0.745
	W	**0.954**	**0.946**	**0.936**	**0.919**	**0.884**	**0.808**
p@15	**T**	0.884	0.843	0.782	0.692	0.566	0.402
	D	0.937	0.920	0.900	0.864	0.790	0.661
	W	**0.952**	**0.943**	**0.930**	**0.907**	**0.856**	**0.755**
p@20	**T**	0.854	0.789	0.709	0.611	0.492	0.349
	D	0.923	0.896	0.856	0.795	0.703	0.575
	W	**0.948**	**0.935**	**0.913**	**0.872**	**0.797**	**0.681**
p@50	**T**	0.375	0.361	0.340	0.308	0.262	0.199
	D	0.393	0.390	0.383	0.370	0.344	0.298
	W	**0.396**	**0.395**	**0.393**	**0.389**	**0.376**	**0.346**
p@100	**T**	0.192	0.187	0.180	0.168	0.149	0.121
	D	**0.199**	0.198	0.196	0.192	0.183	0.165
	W	**0.199**	**0.199**	**0.199**	**0.198**	**0.195**	**0.186**
		Two-step model - $k = \sqrt{\lvert D_T \rvert} \cdot 2$					
		Noise %					
		10%	20%	30%	40%	50%	60%
p@5	**T**	0.899	0.884	0.862	0.822	0.746	0.611
	D	0.953	0.943	0.931	0.916	0.891	0.835
	W	**0.959**	**0.953**	**0.945**	**0.934**	**0.912**	**0.866**
p@10	**T**	0.902	0.880	0.842	0.780	0.679	0.526
	D	0.948	0.935	0.920	0.899	0.857	0.768
	W	**0.958**	**0.951**	**0.942**	**0.928**	**0.898**	**0.831**
p@15	**T**	0.896	0.859	0.803	0.721	0.605	0.456
	D	0.942	0.925	0.904	0.870	0.803	0.688
	W	**0.956**	**0.948**	**0.937**	**0.917**	**0.873**	**0.782**
p@20	**T**	0.868	0.808	0.732	0.641	0.530	0.399
	D	0.928	0.901	0.862	0.804	0.718	0.601
	W	**0.952**	**0.941**	**0.922**	**0.885**	**0.817**	**0.710**
p@50	**T**	0.379	0.368	0.350	0.322	0.281	0.227
	D	0.395	0.391	0.385	0.373	0.349	0.309
	W	**0.397**	**0.397**	**0.395**	**0.392**	**0.382**	**0.357**
p@100	**T**	0.193	0.190	0.184	0.174	0.159	0.136
	D	0.199	0.198	0.196	0.193	0.185	0.170
	W	**0.200**	**0.200**	**0.200**	**0.199**	**0.197**	**0.190**

The upper-left subtable shows the results obtained with the one-step model, while the other subtables show the results obtained by PRILJ with different numbers of clusters. The best result observed for a given n of the precision@n in a given subtable is shown in boldface, while the absolute best result is underlined

Table 2 Recall@n results obtained with different embedding strategies (T=TF-IDF; D=Doc2Vec; W=Word2Vec) and different levels of noise

		One-step model					
		Noise %					
		10%	20%	30%	40%	50%	60%
r@5	**T**	0.188	0.181	0.165	0.135	0.085	0.028
	D	0.233	0.228	0.220	0.195	0.138	0.075
	W	**0.234**	**0.231**	**0.228**	**0.222**	**0.207**	**0.172**
r@10	**T**	0.379	0.350	0.302	0.225	0.122	0.038
	D	0.461	0.449	0.424	0.348	0.227	0.121
	W	**0.467**	**0.461**	**0.452**	**0.435**	**0.391**	**0.307**
r@15	**T**	0.557	0.493	0.401	0.276	0.142	0.043
	D	0.683	0.658	0.591	0.449	0.285	0.154
	W	**0.699**	**0.688**	**0.670**	**0.631**	**0.544**	**0.408**
r@20	**T**	0.696	0.587	0.456	0.305	0.154	0.047
	D	0.885	0.815	0.687	0.510	0.327	0.180
	W	**0.927**	**0.904**	**0.861**	**0.780**	**0.648**	**0.475**
r@50	**T**	0.794	0.695	0.552	0.373	0.191	0.061
	D	0.957	0.919	0.826	0.665	0.465	0.282
	W	**0.976**	**0.968**	**0.950**	**0.906**	**0.804**	**0.635**
r@100	**T**	0.833	0.743	0.603	0.418	0.220	0.074
	D	0.972	0.945	0.884	0.756	0.571	0.374
	W	**0.985**	**0.980**	**0.969**	**0.941**	**0.867**	**0.724**
		Two-step model - $k = \sqrt{\lvert D_T \rvert}/2$					
		Noise %					
		10%	20%	30%	40%	50%	60%
r@5	**T**	0.217	0.213	0.206	0.193	0.168	0.123
	D	0.236	0.233	0.230	0.226	0.218	0.197
	W	**0.238**	**0.236**	**0.234**	**0.230**	**0.223**	**0.207**
r@10	**T**	0.437	0.423	0.399	0.360	0.296	0.201
	D	0.470	0.463	0.455	0.443	0.415	0.352
	W	**0.475**	**0.471**	**0.465**	**0.456**	**0.435**	**0.392**
r@15	**T**	0.650	0.616	0.564	0.490	0.385	0.253
	D	0.700	0.686	0.670	0.640	0.574	0.461
	W	**0.711**	**0.704**	**0.693**	**0.673**	**0.629**	**0.545**
r@20	**T**	0.835	0.764	0.678	0.573	0.444	0.291
	D	0.918	0.889	0.847	0.779	0.675	0.533
	W	**0.944**	**0.930**	**0.905**	**0.859**	**0.776**	**0.652**
r@50	**T**	0.922	0.882	0.819	0.725	0.589	0.413
	D	0.979	0.970	0.952	0.914	0.835	0.701
	W	**0.987**	**0.984**	**0.978**	**0.964**	**0.923**	**0.835**
r@100	**T**	0.947	0.920	0.874	0.799	0.680	0.515
	D	0.990	0.985	0.974	0.951	0.895	0.789
	W	**0.994**	**0.993**	**0.990**	**0.984**	**0.964**	**0.908**

Table 2 (continued)

		Two-step model - $k = \sqrt{\lvert D_T \rvert}$					
		Noise %					
		10%	20%	30%	40%	50%	60%
r@5	**T**	0.222	0.218	0.212	0.200	0.179	0.140
	D	0.237	0.234	0.232	0.228	0.221	0.205
	W	**0.239**	**0.237**	**0.235**	**0.232**	**0.225**	**0.212**
r@10	**T**	0.445	0.433	0.412	0.377	0.321	0.235
	D	0.472	0.465	0.458	0.447	0.424	0.373
	W	**0.477**	**0.473**	**0.468**	**0.460**	**0.442**	**0.404**
r@15	**T**	0.663	0.632	0.587	0.519	0.424	0.301
	D	0.703	0.690	0.675	0.648	0.592	0.496
	W	**0.714**	**0.707**	**0.697**	**0.680**	**0.642**	**0.566**
r@20	**T**	0.854	0.789	0.709	0.611	0.492	0.349
	D	0.923	0.896	0.856	0.795	0.703	0.575
	W	**0.948**	**0.935**	**0.913**	**0.872**	**0.797**	**0.681**
r@50	**T**	0.937	0.903	0.851	0.771	0.654	0.498
	D	0.983	0.974	0.958	0.926	0.860	0.745
	W	**0.989**	**0.987**	**0.983**	**0.972**	**0.940**	**0.865**
r@100	**T**	0.958	0.937	0.901	0.841	0.746	0.607
	D	0.993	0.989	0.979	0.959	0.914	0.826
	W	**0.996**	**0.996**	**0.995**	**0.991**	**0.976**	**0.932**

		Two-step model - $k = \sqrt{\lvert D_T \rvert} \cdot 2$					
		Noise %					
		10%	20%	30%	40%	50%	60%
r@5	**T**	0.225	0.221	0.216	0.205	0.186	0.153
	D	0.238	0.236	0.233	0.229	0.223	0.209
	W	**0.240**	**0.238**	**0.236**	**0.234**	**0.228**	**0.216**
r@10	**T**	0.451	0.440	0.421	0.390	0.339	0.263
	D	0.474	0.467	0.460	0.449	0.428	0.384
	W	**0.479**	**0.476**	**0.471**	**0.464**	**0.449**	**0.416**
r@15	**T**	0.672	0.645	0.602	0.541	0.454	0.342
	D	0.706	0.694	0.678	0.652	0.602	0.516
	W	**0.717**	**0.711**	**0.703**	**0.688**	**0.655**	**0.587**
r@20	**T**	0.868	0.808	0.732	0.641	0.530	0.399
	D	0.928	0.901	0.862	0.804	0.718	0.601
	W	**0.952**	**0.941**	**0.922**	**0.885**	**0.817**	**0.710**
r@50	**T**	0.947	0.919	0.874	0.805	0.704	0.568
	D	0.987	0.978	0.962	0.933	0.874	0.772
	W	**0.993**	**0.992**	**0.989**	**0.981**	**0.956**	**0.893**
r@100	**T**	0.965	0.948	0.919	0.871	0.793	0.678
	D	0.996	0.991	0.982	0.964	0.925	0.850
	W	**0.999**	**0.999**	**0.998**	**0.996**	**0.986**	**0.952**

The upper-left subtable shows the results obtained with the one-step model, while the other subtables show the results obtained by PRILJ with different numbers of clusters. The best result observed for a given n of the recall@n in a given subtable is shown in boldface, while the absolute best result is underlined

Table 3 f1-score@n results obtained with different embedding strategies (T=TF-IDF; D=Doc2Vec; W=Word2Vec) and different levels of noise

		One-step model					
		Noise %					
		10%	20%	30%	40%	50%	60%
f@5	**T**	0.301	0.289	0.264	0.216	0.135	0.046
	D	0.372	0.364	0.353	0.313	0.221	0.121
	W	**0.374**	**0.370**	**0.365**	**0.355**	**0.331**	**0.276**
f@10	**T**	0.506	0.467	0.403	0.300	0.163	0.050
	D	0.614	0.598	0.566	0.464	0.302	0.161
	W	**0.623**	**0.615**	**0.603**	**0.580**	**0.522**	**0.410**
f@15	**T**	0.636	0.563	0.458	0.315	0.162	0.049
	D	0.781	0.752	0.676	0.514	0.325	0.176
	W	**0.799**	**0.786**	**0.766**	**0.721**	**0.622**	**0.466**
f@20	**T**	0.696	0.587	0.456	0.305	0.154	0.047
	D	0.885	0.815	0.687	0.510	0.327	0.180
	W	**0.927**	**0.904**	**0.861**	**0.780**	**0.648**	**0.475**
f@50	**T**	0.454	0.397	0.316	0.213	0.109	0.035
	D	0.547	0.525	0.472	0.380	0.266	0.161
	W	**0.558**	**0.553**	**0.543**	**0.518**	**0.460**	**0.363**
f@100	**T**	0.278	0.248	0.201	0.139	0.073	0.025
	D	0.324	0.315	0.295	0.252	0.190	0.125
	W	**0.328**	**0.327**	**0.323**	**0.314**	**0.289**	**0.241**
		Two-step model - $k = \sqrt{\lvert D_T \rvert}/2$					
		Noise %					
		10%	20%	30%	40%	50%	60%
f@5	**T**	0.348	0.341	0.330	0.309	0.269	0.197
	D	0.378	0.373	0.368	0.362	0.349	0.314
	W	**0.381**	**0.378**	**0.374**	**0.368**	**0.356**	**0.332**
f@10	**T**	0.583	0.564	0.532	0.480	0.395	0.268
	D	0.627	0.617	0.606	0.591	0.554	0.469
	W	**0.634**	**0.628**	**0.620**	**0.608**	**0.580**	**0.523**
f@15	**T**	0.743	0.704	0.645	0.560	0.440	0.289
	D	0.800	0.784	0.766	0.731	0.656	0.527
	W	**0.813**	**0.804**	**0.792**	**0.769**	**0.719**	**0.623**
f@20	**T**	0.835	0.764	0.678	0.573	0.444	0.291
	D	0.918	0.889	0.847	0.779	0.675	0.533
	W	**0.944**	**0.930**	**0.905**	**0.859**	**0.776**	**0.652**
f@50	**T**	0.527	0.504	0.468	0.414	0.336	0.236
	D	0.560	0.554	0.544	0.522	0.477	0.401
	W	**0.564**	**0.562**	**0.559**	**0.551**	**0.528**	**0.477**
f@100	**T**	0.316	0.307	0.291	0.266	0.227	0.172
	D	0.330	0.328	0.325	0.317	0.298	0.263
	W	**0.331**	**0.331**	**0.330**	**0.328**	**0.321**	**0.303**

Table 3 (continued)

		Two-step model - $k = \sqrt{\lvert D_T \rvert}$					
		Noise %					
		10%	20%	30%	40%	50%	60%
f@5	**T**	0.355	0.348	0.338	0.320	0.286	0.223
	D	0.380	0.375	0.370	0.365	0.354	0.327
	W	**0.382**	**0.380**	**0.376**	**0.371**	**0.360**	**0.339**
f@10	**T**	0.593	0.577	0.549	0.503	0.428	0.314
	D	0.629	0.620	0.610	0.596	0.565	0.497
	W	**0.636**	**0.631**	**0.624**	**0.613**	**0.589**	**0.538**
f@15	**T**	0.757	0.723	0.670	0.593	0.485	0.344
	D	0.803	0.789	0.771	0.740	0.677	0.566
	W	**0.816**	**0.808**	**0.797**	**0.777**	**0.734**	**0.647**
f@20	**T**	0.854	0.789	0.709	0.611	0.492	0.349
	D	0.923	0.896	0.856	0.795	0.703	0.575
	W	**0.948**	**0.935**	**0.913**	**0.872**	**0.797**	**0.681**
f@50	**T**	0.535	0.516	0.486	0.440	0.374	0.284
	D	0.562	0.557	0.548	0.529	0.491	0.425
	W	**0.565**	**0.564**	**0.562**	**0.555**	**0.537**	**0.494**
f@100	**T**	0.319	0.312	0.300	0.280	0.249	0.202
	D	0.331	0.330	0.326	0.320	0.305	0.275
	W	**0.332**	**0.332**	**0.332**	**0.330**	**0.325**	**0.311**
		Two-step model - $k = \sqrt{\lvert D_T \rvert} \cdot 2$					
		Noise %					
		10%	20%	30%	40%	50%	60%
f@5	**T**	0.359	0.354	0.345	0.329	0.298	0.244
	D	0.381	0.377	0.372	0.366	0.356	0.334
	W	**0.383**	**0.381**	**0.378**	**0.374**	**0.365**	**0.346**
f@10	**T**	0.601	0.587	0.561	0.520	0.452	0.351
	D	0.632	0.623	0.613	0.599	0.571	0.512
	W	**0.638**	**0.634**	**0.628**	**0.619**	**0.599**	**0.554**
f@15	**T**	0.768	0.737	0.689	0.618	0.518	0.391
	D	0.807	0.793	0.775	0.746	0.688	0.589
	W	**0.819**	**0.813**	**0.803**	**0.786**	**0.748**	**0.671**
f@20	**T**	0.868	0.808	0.732	0.641	0.530	0.399
	D	0.928	0.901	0.862	0.804	0.718	0.601
	W	**0.952**	**0.941**	**0.922**	**0.885**	**0.817**	**0.710**
f@50	**T**	0.541	0.525	0.499	0.460	0.402	0.324
	D	0.564	0.559	0.550	0.533	0.499	0.441
	W	**0.567**	**0.567**	**0.565**	**0.561**	**0.546**	**0.510**
f@100	**T**	0.322	0.316	0.306	0.290	0.264	0.226
	D	0.332	0.330	0.327	0.321	0.308	0.283
	W	**0.333**	**0.333**	**0.333**	**0.332**	**0.329**	**0.317**

The upper-left subtable shows the results obtained with the one-step model, while the other subtables show the results obtained by PRILJ with different numbers of clusters. The best result observed for a given n of the f1-score@n in a given subtable is shown in boldface, while the absolute best result is underlined

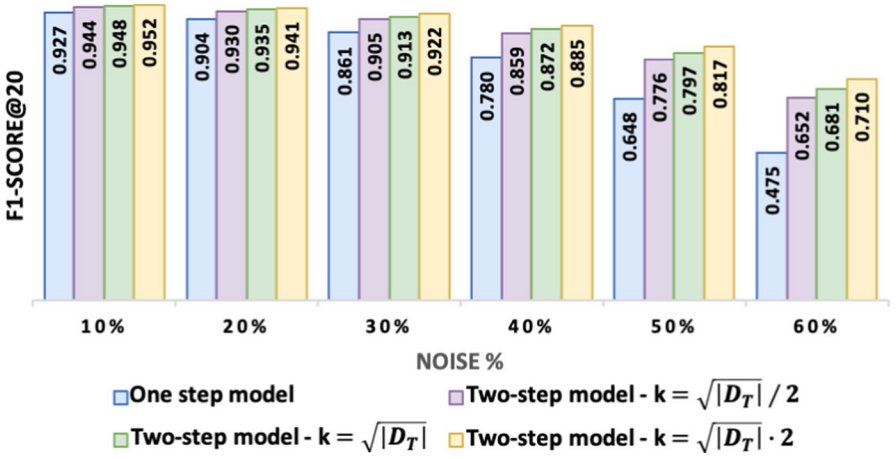

Fig. 7 A histogram showing the impact of the noise on the f1-score@20, with the two-step model (with different values of k) and with the one-step model. As embedding strategy, we considered Word2Vec. The one-step model appears much more sensitive to the presence of noise

always obtains better results. This may be due to the fact that several paragraphs of different legal documents may share a similar topic, and the introduction of the unique sequence ID (*seq*) to associate the context with the document, as done by Doc2Vec, may generate overfitting.

From the results, it is possible to clearly identify the contribution of the two-step architecture we propose. Indeed, the results show that the proposed two-step model outperforms the one-step model, in all situations and for all the considered evaluation measures. We can also observe that the two-step model is much more robust to the presence of noise: although we can still observe a decrease of precision@n, recall@n, and f1-score@n when the noise amount increases, its impact is much less evident. In Fig. 7, we report a histogram that graphically shows the impact of the noise on the f1-score@20, with the two-step model (with different values of k) and with the one-step model. From the results, we can also observe that in general, the number of extracted clusters k seems to not significantly affect the results, even if the best results are observed with $k = \sqrt{|D_T|} \cdot 2$. This means that the documents are distributed among several topics and that learning a different (more specialized) paragraph embedding model for each of them is helpful to retrieve significant paragraph regularities.

Focusing on the comparison with state-of-the-art systems, in Table 4 we report the f1-score@n results obtained by PRILJ (two-step model, $k = \sqrt{|D_T|} \cdot 2$, Word2Vec) and by the considered competitor systems, with different levels of noise. From the results, we can easily observe that PRILJ always outperforms all the competitors, independently on the value of n of the f1-score@n measure, and independently on the amount of noise in the data. Moreover, as we can also observe in Fig. 8, the impact of noise is very evident on competitor systems. On the contrary, PRILJ appears very robust to the noise and, thus, adoptable in real contexts even when the amount of noise in the data is high. The significantly lower f1-score@n results achieved by the competitors, when documents are affected by high levels of noise, can be mainly due to

Table 4 f1-score@n results obtained by PRILJ (two-step model, $k = \sqrt{|D_T|} \cdot 2$, Word2Vec) and by the competitors, with different levels of noise

		Noise %					
		10%	20%	30%	40%	50%	60%
f@5	PRILJ	**0.383**	**0.381**	**0.378**	**0.374**	**0.365**	**0.346**
	LEGAL-BERT-BASE	0.358	0.221	0.082	0.026	0.009	0.003
	LEGAL-BERT-SMALL	0.370	0.243	0.095	0.034	0.014	0.006
	LEGAL-BERT-EURLEX	0.365	0.235	0.099	0.038	0.015	0.006
	BERT-PLI	0.243	0.052	0.017	0.008	0.004	0.002
f@10	PRILJ	**0.638**	**0.634**	**0.628**	**0.619**	**0.599**	**0.554**
	LEGAL-BERT-BASE	0.592	0.347	0.122	0.037	0.013	0.005
	LEGAL-BERT-SMALL	0.611	0.375	0.136	0.047	0.019	0.008
	LEGAL-BERT-EURLEX	0.604	0.370	0.146	0.053	0.021	0.008
	BERT-PLI	0.373	0.072	0.022	0.010	0.006	0.003
f@15	PRILJ	**0.819**	**0.813**	**0.803**	**0.786**	**0.748**	**0.671**
	LEGAL-BERT-BASE	0.739	0.408	0.141	0.042	0.015	0.006
	LEGAL-BERT-SMALL	0.773	0.440	0.154	0.054	0.021	0.009
	LEGAL-BERT-EURLEX	0.757	0.433	0.166	0.060	0.024	0.010
	BERT-PLI	0.428	0.078	0.024	0.011	0.006	0.004
f@20	PRILJ	**0.952**	**0.941**	**0.922**	**0.885**	**0.817**	**0.710**
	LEGAL-BERT-BASE	0.798	0.422	0.147	0.044	0.016	0.006
	LEGAL-BERT-SMALL	0.856	0.459	0.161	0.056	0.023	0.010
	LEGAL-BERT-EURLEX	0.820	0.446	0.172	0.063	0.025	0.010
	BERT-PLI	0.432	0.078	0.024	0.011	0.007	0.004
f@50	PRILJ	**0.567**	**0.567**	**0.565**	**0.561**	**0.546**	**0.510**
	LEGAL-BERT-BASE	0.524	0.336	0.135	0.044	0.016	0.007
	LEGAL-BERT-SMALL	0.544	0.358	0.143	0.053	0.023	0.010
	LEGAL-BERT-EURLEX	0.532	0.350	0.154	0.060	0.025	0.011
	BERT-PLI	0.325	0.065	0.022	0.011	0.008	0.005
f@100	PRILJ	**0.333**	**0.333**	**0.333**	**0.332**	**0.329**	**0.317**
	LEGAL-BERT-BASE	0.318	0.233	0.107	0.037	0.014	0.006
	LEGAL-BERT-SMALL	0.325	0.245	0.110	0.044	0.020	0.010
	LEGAL-BERT-EURLEX	0.321	0.240	0.119	0.050	0.022	0.010
	BERT-PLI	0.218	0.047	0.018	0.010	0.007	0.005

The best result for a given n and with a given noise amount is shown in boldface

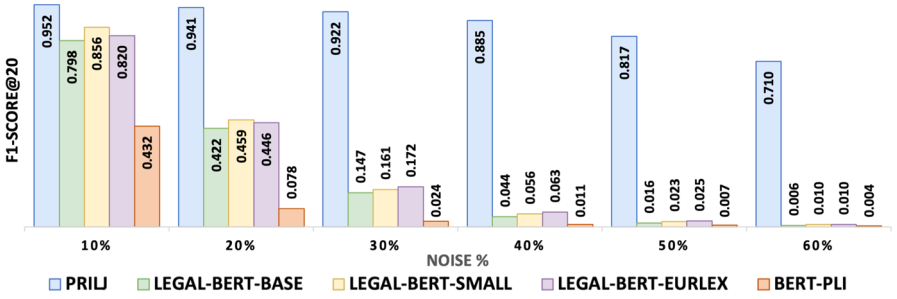

Fig. 8 A histogram showing the impact of the noise on the f1-score@20 achieved by PRILJ (two-step model, $k = \sqrt{|D_T|} \cdot 2$, Word2Vec) and by the competitors. PRILJ appears much more robust to the presence of noise

the higher dimensionality of their embedding space (768 for LEGAL-BERT-BASE, LEGAL-BERT-EURLEX and BERT-PLI, 512 for LEGAL-BERT-SMALL), with respect to that adopted in PRILJ (50). Indeed, although BERT-based models exhibit very interesting results in several NLP tasks, including sentiment analysis, question answering, language inference, and named entity recognition (Devlin et al. 2019), their high-dimensional feature space makes them more susceptible to the curse of dimensionality on tasks based on the computation of distance/similarity, like in the task at hand. This result is coherent with what observed by Kumar et al. (2020), where the sensitivity of BERT-based models to the presence of noise has been investigated.

Finally, we specifically analyzed the performance of the ASSN approach implemented in PRILJ. We recall that we adopt an approximated approach for the identification of paragraph regularities to overcome computational bottlenecks. Since approximated approaches may usually lead to a loss in terms of accuracy, it is important to show that the high efficiency achieved by PRILJ does not come at the price of significantly worse results than those achievable through an exact search. As anticipated in Sect. 4.2, for this purpose, we performed a comparison with the exact computation of the top-n most similar paragraphs using the cosine similarity on a subset of 100 documents, with the PRILJ configuration that provided the best results (i.e., two-step model with $k = \sqrt{|D_T|} \cdot 2$, see Tables 1, 2, 3). The f1-score results of this comparison are shown in Table 5, and graphically summarized in Fig. 9. The exact search based on cosine similarity leads to better results mainly when adopting TF-IDF with high levels of noise. On overall, the observed average improvement in terms of f1-score@n with respect to the adopted ANNS approach is 0.6%, which can be considered negligible. On the other hand, the advantage in terms of efficiency is significant: the exact search required up to 1000x the time took by the ASSN implemented in PRILJ (see Table 6). This advantage is empirically evident even with the small subset of documents that we used, but the difference between their theoretical computational complexity (i.e., $O(log_2(n_r))$ vs $O(n_r)$, for each paragraph of the target document) provides a clear win to ANNS for large document collections. Indeed, while we were able to complete one run of the experiments on the full dataset on average in 1.5 hours, the adoption of the cosine similarity would have required some months on our server equipped with a 6-cores CPU@3.2 Ghz and 64GB of RAM.

Table 5 Comparison between the ANNS implemented in PRILJ and cosine similarity on a subset of 100 documents in terms of the f1-score@n, with different embedding strategies (T=TF-IDF; D=Doc2Vec; W=Word2Vec), different levels of noise and $k = \sqrt{|D_T|} \cdot 2$

		ANNS					
		Noise %					
		10%	20%	30%	40%	50%	60%
f@5	T	0.361	0.355	0.346	0.330	0.301	0.253
	D	0.384	0.381	0.377	0.372	0.364	0.345
	W	0.385	0.384	0.382	0.379	0.372	0.358
f@10	T	0.604	0.589	0.562	0.521	0.454	0.364
	D	0.637	0.631	0.623	0.609	0.585	0.533
	W	0.641	0.639	0.635	0.627	0.610	0.571
f@15	T	0.771	0.738	0.687	0.617	0.522	0.407
	D	0.815	0.804	0.787	0.758	0.706	0.619
	W	0.823	0.819	0.811	0.797	0.760	0.686
f@20	T	0.869	0.805	0.727	0.639	0.534	0.417
	D	0.938	0.914	0.875	0.819	0.741	0.634
	W	0.958	0.949	0.928	0.890	0.822	0.719
f@50	T	0.544	0.527	0.500	0.462	0.408	0.341
	D	0.569	0.566	0.558	0.543	0.513	0.463
	W	0.570	0.570	0.568	0.563	0.548	0.512
f@100	T	0.322	0.317	0.307	0.292	0.269	0.236
	D	0.333	0.332	0.330	0.326	0.315	0.295
	W	0.333	0.333	0.333	0.332	0.328	0.316
		Cosine similarity					
		Noise %					
		10%	20%	30%	40%	50%	60%
f@5	T	*0.365*	*0.359*	*0.349*	*0.333*	*0.305*	*0.262*
	D	0.384	0.381	0.377	0.372	0.364	*0.346*
	W	0.385	0.384	0.382	0.379	0.372	*0.359*
f@10	T	*0.608*	*0.592*	*0.566*	*0.525*	*0.463*	*0.380*
	D	0.637	0.631	0.623	0.609	0.585	*0.536*
	W	0.641	0.639	0.635	0.627	0.610	*0.573*
f@15	T	*0.775*	*0.742*	*0.692*	*0.623*	*0.533*	*0.426*
	D	0.815	0.804	0.787	0.758	*0.708*	*0.622*
	W	*0.824*	0.819	*0.812*	0.797	*0.761*	*0.689*
f@20	T	*0.873*	*0.811*	*0.735*	*0.647*	*0.545*	*0.434*
	D	0.938	0.914	0.875	0.819	*0.743*	*0.637*
	W	0.958	0.949	*0.929*	*0.892*	*0.825*	*0.724*
f@50	T	*0.551*	*0.533*	*0.506*	*0.468*	*0.416*	*0.350*
	D	0.569	0.566	0.558	0.543	*0.514*	*0.464*
	W	*0.571*	0.570	0.568	*0.564*	*0.549*	*0.514*
f@100	T	*0.328*	*0.322*	*0.312*	*0.297*	*0.274*	*0.242*
	D	0.333	0.332	0.330	0.326	*0.316*	0.295
	W	0.333	0.333	0.333	0.332	0.328	*0.317*

The configurations in which the cosine similarity achieves better results are emphasized in italic

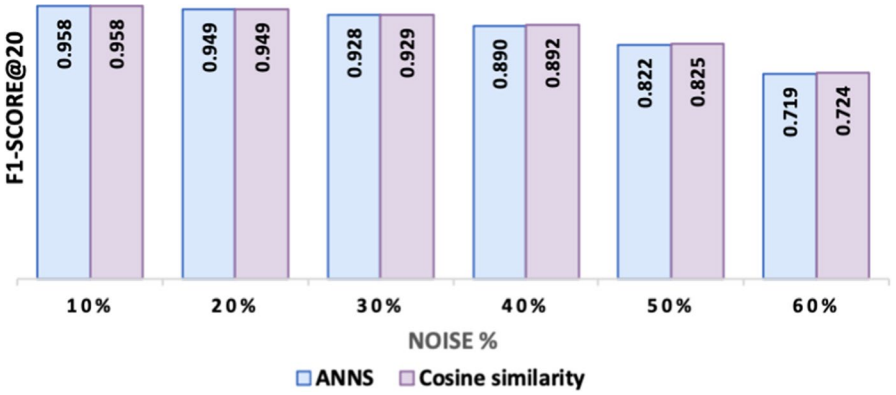

Fig. 9 A histogram showing the differences in terms of f1-score@20 achieved on a subset of 100 documents when using ANNS or the cosine similarity, with the two-step model, $k = \sqrt{|D_T|} \cdot 2$ and Word2Vec as embedding strategy

Table 6 Average running time (s) for the identification of the top-n most similar paragraphs, with the two-step model and $k = \sqrt{|D_T|} \cdot 2$

	ANNS	Cosine Similarity
TF-IDF	**0.513**	407.612
Doc2Vec	**0.551**	580.842
Word2Vec	**0.610**	668.040

Best results are emphasized in boldface

The obtained results allow us to conclude that: *i)* PRILJ based on Word2Vec, the two-step model and $k = \sqrt{|D_T|} \cdot 2$ provides the best overall results for the identification of paragraph regularities in legal case judgments; *ii)* the two-step model based on clustering implemented in PRILJ provides clear advantages, since it is able to properly model the different topics in the document collection and is very robust to the presence of noise; *iii)* the efficient ASSN strategy adopted by PRILJ provides results comparable to those achieved by an exact search, in a fraction of time. These conclusions make PRILJ a useful tool that can be adopted in real-world scenarios, for the accurate and efficient identification of paragraph regularities from large collections of legal case judgments, which can be profitably used in the redaction of similar legal documents.

5 Conclusions

In this work, we proposed PRILJ, a novel approach to identify paragraph regularities in legal case judgments. PRILJ represents documents and paragraphs thereof in a numerical feature space by exploiting embedding methods able to catch the context and the semantics. Moreover, PRILJ is based on a two-step model, that groups similar documents into clusters and, for each of them, learns a specific paragraph

embedding model. This approach allows us to properly catch peculiarities exhibited by paragraphs and documents of similar topics and to handle the presence of noise in a robust manner. Finally, PRILJ is able to identify paragraph regularities with respect to target documents very efficiently.

Our extensive experimental evaluation has proved the accuracy and the efficiency of the developed approach on real data, which can be considered a useful tool in real-world scenarios, also when large collections of documents have to be analyzed. PRILJ has also been able to outperform four existing state-of-the-art competitor systems, achieving significantly better performances when the amount of noise in the data increases.

For future work, we will extend the capabilities of PRILJ in providing, in addition to retrieval functionalities, also suggestions during the preparation of new legal documents. Specifically, we will exploit process mining methods to identify frequent patterns observed in the sequences of paragraphs of legal documents. This would allow us to suggest the next (type of) paragraph to include in a legal document under preparation, as well as to perform conformance checking on a legal document, i.e., to verify if it has been properly written in accordance with the patterns observed on other, similar legal documents.

Funding Information Open access funding provided by Università degli Studi di Bari Aldo Moro within the CRUI-CARE Agreement. GP acknowledges the support of Ministry of Universities and Research through the project "Big Data Analytics", AIM 1852414-1 (line 1).

Data Availability Statement The adopted dataset and all the detailed results are available at: https://osf.io/2jum9/?view_only=ea9c9294999746ccb2af62eddacb8d9a. The source code repository of the system will be made publicly available after the publication of the manuscript.

References

Berkhin P (2002) Survey of clustering data mining techniques. A Survey of Clustering Data Mining Techniques Grouping Multidimensional Data: Recent Advances in Clustering, vol 10

Bernhardsson E (2015) Annoy at github. https://github.com/spotify/annoy

Biagioli C, Francesconi E, Passerini A, Montemagni S, Soria C (2005) Automatic semantics extraction in law documents. In: The tenth international conference on artificial intelligence and law, proceedings of the conference, June 6-11, 2005, Bologna, Italy, ACM, pp 133–140

Brüninghaus S, Ashley K (2001) Improving the representation of legal case texts with information extraction methods. In: Proceedings of the international conference on artificial intelligence and law, pp 42–51

Ceci M, Corizzo R, Japkowicz N, Mignone P, Pio G (2020) ECHAD: embedding-based change detection from multivariate time series in smart grids. IEEE Access 8:156053–156066

Chalkidis I, Fergadiotis M, Malakasiotis P, Aletras N, Androutsopoulos I (2020) LEGAL-BERT: The muppets straight out of law school. In: Findings of the association for computational linguistics: EMNLP 2020, Association for Computational Linguistics, Online, pp 2898–2904

Conrad JG, Al-Kofahi K, Zhao Y, Karypis G (2005) Effective document clustering for large heterogeneous law firm collections. In: Sartor G (ed) The tenth international conference on artificial intelligence and law, proceedings of the conference, June 6-11, 2005, Bologna, Italy, ACM, pp 177–187, 10.1145/1165485.1165513, https://doi.org/10.1145/1165485.1165513

Corizzo R, Pio G, Ceci M, Malerba D (2019) DENCAST: distributed density-based clustering for multi-target regression. J Big Data 6:43

Corizzo R, Ceci M, Zdravevski E, Japkowicz N (2020) Scalable auto-encoders for gravitational waves detection from time series data. Expert Syst Appl 151:113378

Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, pp 4171–4186

Donghwa K, Seo D, Cho S, Kang P (2018) Multi-co-training for document classification using various document representations: Tf–idf, lda, and doc2vec. Information Sciences, vol 477

Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the second international conference on knowledge discovery and data mining, AAAI Press, KDD'96, pp 226–231

Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Krishnapuram B, Shah M, Smola AJ, Aggarwal CC, Shen D, Rastogi R (eds) Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, San Francisco, CA, USA, August 13-17, 2016, ACM, pp 855–864

Jin L, Schuler W (2015) A comparison of word similarity performance using explanatory and non-explanatory texts. In: Mihalcea R, Chai JY, Sarkar A (eds) NAACL HLT 2015, The 2015 conference of the north american chapter of the association for computational linguistics: human language technologies, Denver, Colorado, USA, May 31 - June 5, 2015, The Association for Computational Linguistics, pp 990–994

Kachappilly D, Wagh R (2018) Similarity analysis of court judgments using clustering of case citation data: a study. Int J Eng Technol 7:855

Kumar A, Makhija P, Gupta A (2020) Noisy text data: Achilles' heel of bert. In: Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020), pp 16–21

Kumar S, Reddy PK, Reddy VB, Singh A (2011) Similarity analysis of legal judgments. In: Proceedings of the 4th Bangalore Annual Compute Conference, Compute 2011, Bangalore, India, March 25-26, 2011, ACM, p 17

Kumar S, Reddy PK, Reddy VB, Suri M (2013) Finding similar legal judgements under common law system. In: Madaan A, Kikuchi S, Bhalla S (eds) Databases in networked information systems. Springer, Berlin Heidelberg, pp 103–116

Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: 31st International conference on machine learning, ICML 2014 4

Li W, Zhang Y, Sun Y, Wang W, Zhang W, Lin X (2016) Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement (v1.0). CoRR

Lu Q, Conrad JG, Al-Kofahi K, Keenan W (2011) Legal document clustering with built-in topic segmentation. In: Proceedings of the 20th ACM conference on information and knowledge management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011, ACM, pp 383–392

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, Association for Computing Machinery, Compute '17, pp 1–9

Maxwell KT, Schafer B (2008) Concept and context in legal information retrieval. In: Francesconi E, Sartor G, Tiscornia D (eds) Legal knowledge and information systems - JURIX 2008: the twenty-first annual conference on legal knowledge and information systems, Florence, Italy, 10-13 December 2008, IOS Press, Frontiers in Artificial Intelligence and Applications, vol 189, pp 63–72

Medvedeva M, Vols M, Wieling M (2020) Using machine learning to predict decisions of the european court of human rights. Artif Intell Law 28:237–266

Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems 26:3111–3119
Miñarro-Giménez JA, Marín-Alonso O, Samwald M (2015) Applying deep learning techniques on medical corpora from the world wide web: a prototypical system and evaluation. CoRR
Minocha A, Singh N, Srivastava A (2015) Finding relevant indian judgments using dispersion of citation network. In: Proceedings of the 24th International Conference on World Wide Web, Association for Computing Machinery, pp 1085–1088
Pio G, Ceci M, Loglisci C, D'Elia D, Malerba D (2012) Hierarchical and overlapping co-clustering of mrna: mirna interactions. In: Raedt LD, Bessiere C, Dubois D, Doherty P, Frasconi P, Heintz F, Lucas PJF (eds) ECAI 2012 - 20th European conference on artificial intelligence. Including prestigious applications of artificial intelligence (PAIS-2012) system demonstrations track, Montpellier, France, August 27-31 , 2012, IOS Press, Frontiers in Artificial Intelligence and Applications, vol 242, pp 654–659
Pio G, Ceci M, Prisciandaro F, Malerba D (2020) Exploiting causality in gene network reconstruction based on graph embedding. Mach Learn 109(6):1231–1279
Raghav K, Reddy P, Reddy V, Krishna RP (2015) Text and citations based cluster analysis of legal judgments. In: Mining Intelligence and Knowledge Exploration, Springer International Publishing, pp 449–459
Shao Y, Mao J, Liu Y, Ma W, Satoh K, Zhang M, Ma S (2020) Bert-pli: Modeling paragraph-level interactions for legal case retrieval. In: Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI-20, pp 3501–3507
Shulayeva O, Siddharthan A, Wyner A (2017) Recognizing cited facts and principles in legal judgements. Artif Intell Law 25(1):107–126
Silveira M, Ribeiro-neto B (2004) Concept-based ranking: A case study in the juridical domain. Inf Process Manage 40:791–805
Sutton C, McCallum A (2012) An introduction to conditional random fields. Found Trends Mach Learn 4:267–373
Thenmozhi D, Kannan K, Aravindan C (2017) A text similarity approach for precedence retrieval from legal documents. In: Working notes of FIRE 2017 - Forum for Information Retrieval Evaluation, Bangalore, India, December 8-10, 2017, CEUR-WS.org, CEUR Workshop Proceedings, vol 2036, pp 90–91
Tomlinson S, Oard DW, Baron JR, Thompson P (2007) Overview of the TREC 2007 legal track. In: Proceedings of The Sixteenth Text REtrieval Conference, TREC 2007, Gaithersburg, Maryland, USA, November 5-9, 2007, National Institute of Standards and Technology (NIST), NIST Special Publication, vol 500-274
Trompper M, Winkels R (2016) Automatic assignment of section structure to texts of dutch court judgments. In: Legal Knowledge and Information Systems - JURIX 2016: The Twenty-Ninth Annual Conference, IOS Press, Frontiers in Artificial Intelligence and Applications, vol 294, pp 167–172
Zhao Y, Karypis G, Fayyad U (2005) Hierarchical clustering algorithms for document datasets. Data Min Knowl Discov 10:141–168
Zhong H, Xiao C, Tu C, Zhang T, Liu Z, Sun M (2020) How does NLP benefit legal system: A summary of legal artificial intelligence. CoRR arXiv:2004.12158

Authors and Affiliations

Graziella De Martino[1] · Gianvito Pio[1,2] · Michelangelo Ceci[1,2,3]

Graziella De Martino
graziella.demartino@uniba.it

Michelangelo Ceci
michelangelo.ceci@uniba.it

[1] University of Bari Aldo Moro, Via E. Orabona 4, 70125 Bari, Italy

[2] Big Data Lab, National Interuniversity Consortium for Informatics (CINI), Rome, Italy

[3] Jozef Stefan Institute, Jamova 39, Ljubljana, Slovenia