Information Sciences 607 (2022) 1023–1040

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Statute recommendation: Re-ranking statutes by modeling case-statute relation with interpretable hand-crafted features

Chuanyi Li [a], Jidong Ge [a,*], Kun Cheng [a], Bin Luo [a], Victor Chang [b,*]

[a] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[b] Department of Operations and Information Management, Aston Business School, Aston University, Birmingham, UK

ARTICLE INFO

Article history:
Received 27 August 2020
Received in revised form 8 June 2022
Accepted 11 June 2022
Available online 14 June 2022

Keywords:
Statute recommender
Learning to rank
Text matching
Text relation modeling

ABSTRACT

In the continental law system, it is appropriate for judges to find relevant laws and consider rules defined in them when dealing with legal cases. Therefore, recommending relevant laws quickly and accurately based on case content is crucial in improving the efficiency of case processing. There have been researched works of recommender systems in various fields, but few of them lucubrates systems that recommend statutes for cases. To the best of our knowledge, there is no research on recommending statutes by modeling the relationship between case content and law content with interpretable hand-crafted features. In this paper, we define five novel types of features for calculating relevance between a case and a statute for resorting all statutes retrieved through collaborative filtering for the input case. Both pair-wise and list-wise ranking models are trained based on all these features for re-ranking the statutes list. Besides, we also test the combinations of different learning algorithms and popular pre-trained language models. Experimental results show that adopting the proposed novel features in pair-wise ranking achieves the best performance. It improves the recommendation recall of the Top 1 statute by almost 5% compared with the collaborative filtering approach.

1. Introduction

With the rapid development of all aspects of Chinese society, the need for legal guidance is increasing in many fields, one of which is the divorce between husband and wife. Nowadays, there turn to be more and more divorce cases in China [1]. This means that lawyers and judges need to process more and more cases every day and more and more ordinary people are getting involved in divorce cases. Since divorce has a large short and long impact on both adults and children[2], it is essential for both the judges and the litigant to handle divorce cases properly. On the one hand, lawyers and judges need the support of an automatic and intelligent tool greatly to improve work efficiency. On the other hand, common people need a tool to help them handle their cases and give them pieces of advice on how to defend their rights. Hence, a statute recommender system that could suggest proper potential statutes is needed in seeking an optimal result for divorce cases. It would help the judge search-relevant statutes and the parties know their situation better. However, problems in the judicial domain always involve professional and complicated domain knowledge. There are few automatic and intelligent solutions

* Corresponding author.
E-mail addresses: lcy@nju.edu.cn (C. Li), gjd@nju.edu.cn (J. Ge), v.chang1@aston.ac.uk, victorchang.research@gmail.com (V. Chang).

https://doi.org/10.1016/j.ins.2022.06.042
0020-0255/

for these problems yet. Most solutions rely heavily on the participation of people. Therefore, how to automate and intelligent processes in these problems is an issue of practical significance in this field.

Approaches for recommending statutes automatically have been shifted from manually generated rules based ones [3,4] to machine learning based ones [5–7]. As we all know, eliminating the effects of data imbalances in classification tasks is a very tricky issue. For machine approaches, the most reasonable ones are to construct a model that maps information of cases to all statutes, which is usually a multi-label classification model [5,6]. But the problem is still under-solved in three aspects. First, due to a large number of statutes, it is a big challenge to train a classifier with machine learning algorithms to provide accurate classification results by taking all statutes as data labels. Second, the classifier cannot handle unregistered statutes, such as most recently modified or published ones. Last, it always trends to recommend frequently used statutes than infrequently used ones.

Collaborative filtering (CF) is a common technique utilized in recommender systems and it could overcome the pre-mentioned problems well to some extent. Few published works of statute recommendation adopt a collaborative filtering approach to the best of our knowledge. The key problem in CF is how to retrieve legal cases that would refer to the same statutes as the input case. However, just like the classification approaches, only case content is considered in the recommending process. Besides, more unsuitable statutes would also be retrieved through CF. Effective operation is needed for resorting to retrieved statutes by putting suitable statutes ahead of those unsuitable ones. I order to solve the problem, a deep-learning-based approach is proposed in [8] for judging the suitableness of a statute to a case. However, it is very difficult to construct a proper training dataset for building a high-performance classifier. Besides, under the CF scenario, the problem of unbalanced suitable and unsuitable statutes is still serious.

In this paper, we propose a novel ranking approach for resorting to statutes retrieved through collaborative filtering for recommendation and make three contributions concretely:

(1) We design and construct a complete and applicable divorce-related statutes recommender system, as is shown in Fig. 1. First, similar cases of the input one are derived. Then top-k1 statutes frequently cited in similar cases are retrieved as candidates for recommendation. This greatly reduces the size of the candidate set. Next, the relation between each candidate and the input case is calculated. Eventually, all relation vectors are fed to the pre-trained ranker for re-sorting statutes and top-k2 are returned as ultimately recommended ones. Besides, the system is applicable for data scaling up.

(2) We propose a weighted keywords-based method for searching for similar cases. Weights of keywords are calculated by integrating their TF-IDF values, types of POS tags, and their importance in determining suitable statutes for cases. Similar cases are ranked according to the weights of keywords that they contain.

(3) We propose a group of novel interpretable hand-crafted features to build the relationship between a case and a statute (named **Relational Features**), i.e., calculating relation vectors for ranking statute candidates. Unlike other statute recommendation methods, the content of statutes is utilized by the relational features in our approach.

Experimental results show that our collaborative filtering approach increases the performance of retrieving similar cases in the statute recommendation scenario compared with the baselines. This provides a good foundation for further ranking statutes. Ultimately, our Case-Statute relational feature-based ranking approach optimizes the results by reducing the impact of frequently referred statutes, i.e., the most frequently used statutes would not be recommended as the top 1 statute. To prove the effectiveness of the relational features, we compare the ranking approach with baseline methods like suitable binary classification, multi-label classification, and ranking based on text features generated by BERT [9] (the version of BERT adopted in this paper is *BERT-base, Chinese* [1]) and LBERT [10] (a pre-trained BERT on a large scale number of Chinese legal judgment documents [2]). Besides, feature ablation experiments show that different part of the relational features has different effectiveness in ranking statutes. It provides guidance on how to write a description of cases to make it easier to retrieve suitable statutes. Since there have been much-advanced learning to rank approaches proposed, our proposed Case-Statute relational features can be easily ported to and utlized by them.

The remainder of this paper is laid out as follows. Section 2 talks about related work. Section 3 introduces the prepared corpus for developing the approach. Section 4 describes details of the proposed statute recommending approach, including the collaborative filtering part and the Case-Statute relational features. Section 5 illustrates evaluation metrics to present and discuss the experimental results. Section 6 concludes our paper.

2. Related work

2.1. AI and Law

The history of Artificial Intelligence (AI) and Law started about four decades ago. Law is a rich testbed and important application field for logic-based AI research [11]. For most of its existence, this discipline has been primarily concerned with attempts to formalize legal reasoning and create expert systems [12]. In recent years, research on AI applications in the legal domain has slightly shifted from developing rule-based, argument-based, and case-based models for representing legal

[1] The github webpage is: https://github.com/google-research/bert, and the version we use can be downloaded from: https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip

[2] The github webpage is: https://github.com/thunlp/OpenCLaP, and the version we use can be downloaded from: https://thunlp.oss-cn-qingdao.aliyuncs.com/bert/ms.zip

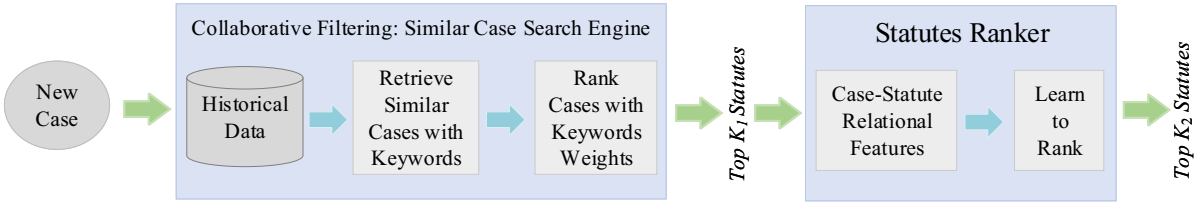

Fig. 1. Framework of the proposed statute recommending approach.

knowledge and reasoning to adopting machine learning and text analytics to accomplish tasks in legal domain effectively and efficiently, e.g., legal judgment prediction [13,14], legal case retrieval [15,16], and court view generation [17,18], etc. Most recently, the methods based on pre-training language models have gradually become mainstream in the Legal AI field. For example, LBERT is [10] a pre-trained BERT[9] model based on a large scale number of Chinese legal judgment documents. Legal-RoBERTa [19] [3] is the legal version of the general RoBERTa [20] pre-trained on Chinese legal long documents. Since Legal-RoBERTa is designed for large inputs, i.e., max-length equals 4096, it is not suitable for texts in our dataset whose average length is just 348. So, in this paper, we take LBERT as the baseline method of the SOTA pre-trained language model.

2.2. Recommender Systems and Statute Recommendation

In recommender systems, multiple efforts have been made for better results. Introducing and modeling new information useful for solving the problem is a common idea. For example, while recommending the preferred type for shops, [21] takes the location and commercial features of a shop into consideration for ranking the (shop, type) pairs. [22] tries to recommend product bundles by modeling relationships between product items and extracting consumers' preferences from this model with a ranking approach. [23] leverages the past click records and the purchase records of users to perform purchase prediction and recommendation. However, although most works focus on finding and utilizing heterogeneous data like the above examples, few of them try to model relationships between users/queries and recommend items, i.e., the direct connection between either a shop and a type or a user and a product is modeled by a feature vector for ranking. It is the same condition as statute recommendation approaches. Multiple ways and techniques have been applied in statute recommendations in recent years. [24] tries to use text mining techniques to find relevant cases and [5] uses text mining to find pertinent statutes. [25] attempts to improve statute recommendation with Latent Semantic Analysis. [6] tries to recommend statutes with relationships between statutes by Convolutional Neural Network. But none of the studies uses relationships between descriptions of statutes and cases for recommending statutes for cases. In our paper, we utilize texts of both input case and statute candidates to recommend modeling the connection between them and providing it to the ranking model.

2.3. Collaborative Filtering

Collaborative Filtering (CF) is a widely-used algorithm in the recommended system and has achieved good performance in many problems. For example, [26] predicts a user's rating for a product according to his/her social network connections and products he/she has bought with CF idea. [27] tries to model user-user and app-app similarities through a kernel function in MF (Matrix Factorization) in-app recommendation, where the connections between an input user and recommended apps are also constructed through CF theory. [28] enhances memory-based CF for personalised recommendation, computing similarity by representing users through their distances to preselected users. [29] proposes a sub-one quasi-norm-based similarity measure for CF in a recommender system of an online store, which makes good use of rating values and deemphasizes the dissimilarity between users. However, non of the existing automated statute recommendation approaches discusses how to retrieve statute candidates in detail. In this paper, we adopt the CF technique to search statute candidates for user input according to *UserInput* → *ExistingCases* → *CitedStatutes* relations.

2.4. Learn to rank

Learn to rank is a widely used solution in information retrieval and recommendation. [30] focuses on content similarity-based methods and integrates different similarity measures under the learning to the rank framework in news citation recommendation. [31] introduces three types of content information and utilizes a convolutional neural network as the foundation of a unified POI recommendation framework. However, [32] finds that learning-to-rank algorithms are based on convex proxies that lead to poor approximations and try to use re-ranking algorithms based on mathematical programming. [33] employs a neural network to re-rank and improve the results of the existing recommender system. [34] proposes to use LSTM in re-ranking documents for a query. Recently, neural network language models, such as BERT, are also adopted in re-

[3] https://github.com/thunlp/LegalPLMs

ranking documents for queries. In these approaches (e.g., [35,36]), a (query, document) pair is encoded by the language model into input embedding vectors for subsequent models. Besides, the adopted language models were also fine-tuned through their tasks. The process of our statute recommendation approach is more like the one proposed in [37] for retrieving basketball games. First, cluster similar historical games and then rank those similar to the query based on features extracted from basketball games and users' preference feedback. In our approach, statute candidates that are potentially suitable for the query are retrieved through CF first and then they are ranked by relation features extracted using the Case-Statute relational features.

3. Corpus

In the field of AI and Law, existing works often experimented on the high quality, of large size, and widely used Chinese legal dataset CAIL2018 [38] or extracted task-specific texts from the published judgment documents to construct their own dataset [39,13]. However, these datasets have been through a series of pre-processing, making it difficult to map each case to the original legal judgment document to find all required fields for calculating the relational features proposed in this paper. So for conducting our work, we collected a data set consisting of divorce cases that happened in 2015 and 2016 from Chinese Judgement Online (http://wenshu.court.gov.cn) by ourselves. We download the first 30000 case records presented in the system and only those of first-round civil instances are kept in the dataset, which is 19860 cases. Each case consists of descriptions from both plaintiffs and defendants, courts findings of the case, analysis of the case, the result of judgment and statutes that the case actually cited (i.e., Fig. 2). When judges or someone involved in a case try to search for statutes, they usually input descriptions about their cases and then the search engine will show the relevant statutes of input cases. In other words, statute recommendation is a problem that constructs a system to find relevant statutes and sort them in some ways based on the input descriptions of cases. So, in the statute recommendation scenario, only statements of the plaintiff and defendant can be used as inputs, as shown in Fig. 2. But other parts of historical cases can also provide advice on modeling case-statute relations and we utilize them in the training process.

Divorce cases in our data set actually cite 456 distinct statutes in total and each case cites four statutes on average. Besides, the maximum and the minimum numbers of cited statutes in a single case are 16 and 1. The maximum and minimum length of descriptions from both plaintiffs and defendants in case records are 4889 and 24 words, respectively, and the average length is 347.6 words. Table 1 concludes statistics of our dataset.

In order to apply a ranking model, there should be a list of potential citable statutes for each case. Hence, first, we build a strategy for searching citable statutes for cases based on collaborative filtering theory. That is, recommending statutes cited by similar cases of the input case. For each case, we derive top 30 (sorted according to their citation frequency) statutes from its top 50 similar cases among the other 19859 cases. The plaintiffs' and defendants' descriptions of each case and the corresponding 30 statutes are the inputs of the statute ranking model. There is a list of potential citable statutes for each of the

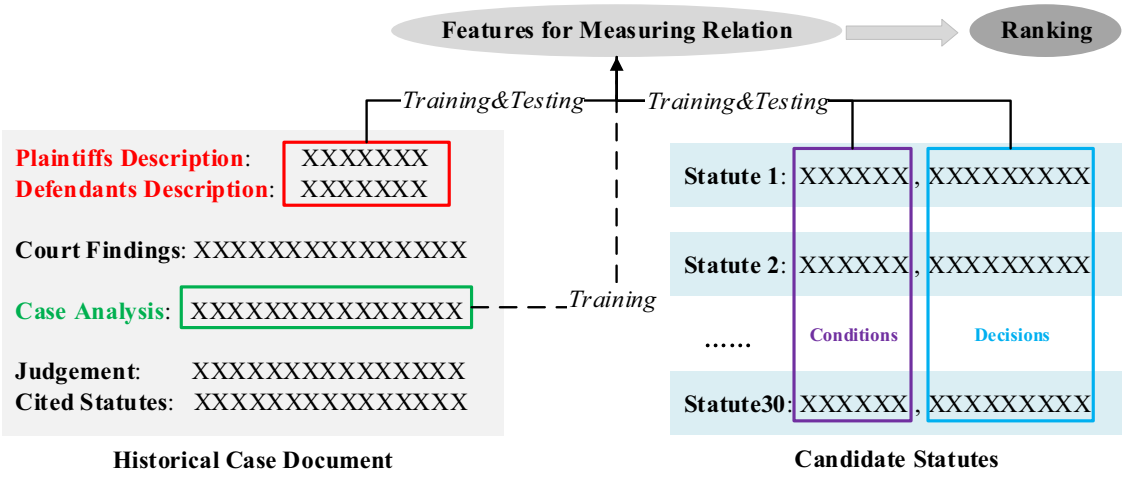

Fig. 2. Case Document, Statutes Content and the relational features.

Table 1
Statistics on the corpus.

# of Case	Maxlen of Case	Minlen of Case	Avg.len of Case	# of Statutes	Max Citation	Min Citation	Avg. Citation
19860	4889	24	347.6	456	16	1	4.01

19,860 cases in our data set. Since we try similar case searching methods, there is a corresponding list of candidate statutes for each case for each method.

4. Statute Recommending Approach

Our statute recommending approach consists of two steps: (1) searching similar cases for recommending potential citable statutes for the input divorce case according to collaborative filtering technique and rank the statutes by their frequencies of being cited by similar cases, and (2) re-ranking these statutes by measuring case-statute relations based on the newly proposed relational features and adopting learning-to-rank methods.

In the first step, we search for cases that are similar to the input one, e.g., A, and put in the set C_A with a similarity algorithm and then, based on cases in C_A we can generate a list of statutes that are cited by cases in C_A, mark with S_A, as potential recommendations for A. The similarity algorithm and retrieval mode in this step can be defined with plenty of flexibility.

In the second step, we first train a ranking model for sorting statutes in S_A to put those truly cited by the case A at the top places in the list. Upon the candidate statutes of a new input case are retrieved by the searching engine, the ranking model can be applied to sorting them. A group of novel case-statute relational features utilizing the content of both cases and statutes is proposed for calculating text features for ranking.

In the next two subsections, we introduce strategy of searching similar cases and the relational features, respectively.

4.1. Derive Statute Candidates by Retrieving Similar Cases

We try different text representation approach for divorce cases in the first step, i.e., transforming the case description into a document vector consisting of float values. For example, we try text topic model Latent Dirichlet Allocation (LDA) [40], neural network based Doc2vec [41], and Term Frequency-Inverse Document Frequency (TF-IDF). However, considering the practicality, we also try different case description indexing methods based on these document vectors to improve the searching efficiency and strategy scalability as the number of historical cases increases. Through experiments, we find that combining traditional inverse index with TF-IDF vectors, as well as cases' keywords retrieved through feature selection methods would achieve the best similar cases results under the statute recommendation scenario, i.e., the curve of average statute recall values of all cases in the data set converges fastest at the highest value.

4.1.1. Construct

Segmentation, TF-IDF, and Keywords. First, for each case, we generate a segmentation for its description text with the tool provided by Jieba [4]. In order to reduce the impact of common words on results, we generate a list of unimportant common words, i.e., stopwords, and ignore their existence in the case description while conducting further calculations. Upon the word list of each case is prepared, the TF-IDF vectors of them are calculated. At the same time, the Part-of-Speech (POS) Tag of each word is also derived.

Since we want to retrieve similar cases for improving statute recommendation performance, cases with more referred statutes should be more similar to each other. Therefore, it is necessary to measure the similarity between cases through the factors hidden in the case description that determine its cited statutes. Keywords in cases that determine the referred statutes are one of those readable and interpretable factors. Feature selection is an efficient and effective method for finding those keywords. We try three different algorithms to determine keywords of cases, namely, Information Gain (IG) [42], Weighted Log-Likelihood Ratio (WLLR) [43], and Weighted Frequency and Odds (WFO).

A divorce case classification system using statutes as text labels is established for applying these feature-selecting algorithms. In our dataset, there are 19860 data instances and 456 data labels. Each data instance has one or multiple labels. Eventually, for each algorithm, a list of keywords that are important to the classification system is retrieved, as well as the weights of all keywords. For example, for each keyword in the list derived by IG, its weight is the sum of its IG values of all statutes.

Eventually, each case is represented by a set of non-stop words and each word has three properties, namely, POS tag, TF-IDF value and keyword's weights (if it is not a keyword, the weight is 0). Let C be a divorce case, then $C = \{W|W = (word, POS, TFIDF, W_{key})\}$. Then we generate the inverse index for searching for similar cases.

Inverse Index. While generating an inverse index for historical divorce cases, we retrieve all words in cases for building the indexing list. For each indexing word, both ids of cases containing it and its corresponding three properties in the cases are recorded. Each indexing word could be expressed by $Word = \{(C_{ID1}, POS_1, TF-IDF_1, W_{key1}), \ldots, (C_{IDn}, POS_n, TFIDF_n, W_{keyn})\}$, where C represents a divorce case and n represents the number of cases that containing the word $Word$.

While searching similar cases for a new one, the words of the new one would be used for retrieving historical cases according to the constructed inverse index. There would be a list of cases derived for each word and the same case may be shown in different lists. So, we should merge all cases according to their IDs. The recorded properties of words in each case should also be merged for ranking all retrieved cases, i.e., calculating the weights of cases.

[4] https://pypi.org/project/jieba/

Weights of Cases. Let C_{new} be the input case and C (id of which is ID_C) be one of the similar cases of C_{new}, then the weight of C is calculated by formula 1:

$$Weight(C) = \alpha\sum_{i=1}^{m} f(POS_i) + \sum_{i=1}^{m} TFIDF_i + \beta\sum_{i=1}^{m} W_{keyi} \tag{1}$$

$$f(POS) = \begin{cases} w_s, & \text{if } POS \in \mathbb{S}; \\ w_o, & \text{if } POS \notin \mathbb{S}. \end{cases} \tag{2}$$

where m is the number of searching keys that shared by C_{new} and C, i.e., $C_{new} \cap C = \{Word_1, Word_2, \ldots, Word_m\}$, and $POS_i, TFIDF_i, W_{keyi}$ are corresponding properties of $Word_i$ ($i \in \{1, 2, \ldots, m\}$) in case C. While measuring the weight of cases, we considering all the three properties of indexing words in the case and allocating different weights for them. Further more, we set different weights for different POS tag. If the word has a *Special* tag in the set $\mathbb{S}$, the weight is w_s, otherwise is w_o. In our approach, we set $verb, adverb$ and *adjective* as special tags, and w_s equals 0.6 while w_o equals 0.4. α and β are chosen according to average value of different properties, as well as their importances. We set them 0.1 and 0.5 respectively in our experiments.

Upon weights of all retrieved cases are calculated, they could be sorted and a top K list of similar cases would be derived. However, ranking cases is not the ultimate goal. The weights of cases are used for weighting the statutes they cited. Let S be one statute cited by cases $\{C_1, C_2, \ldots, C_l\}$ in the top K list, then weight of S is:

$$StatuteWeight(S) = \sum_{i=1}^{l} Weight(C_i) \tag{3}$$

According to weights of all statutes cited by cases in the top K list, all statutes can also be ranked and a top K statutes can be retrieved for a first-round recommendation, i.e., statute candidates. Each statute candidate S has three properties, namely, $StatuteWeight, NumberofCitingCases$, and *textContent*. The text content of a statute is a list of words that are also derived by *Segmentation* and $RemovingStop - words$. These candidates would be re-sorted by the designed ranker.

4.1.2. Comparison

Baselines. To evaluate the proposed similar case searching strategy, we compare it with several baselines used for similar document searching. They are:

(1) Tfidf: In this method, we weight cases with TFIDF values of words only while ranking cases. That means only the TFIDF part in formula 1 is used.
(2) WightTfidf: Both POS tag type and TFIDF values of words are used in weighting retrieved cases, i.e., the first two parts of formula 1 are used.
(3) LDA-related: We generate topic vectors for cases with LDA and retrieve similar cases in two different ways. The first one is ranking cases by calculating the Euclidean Distance between the topic vectors of the input case and the historical ones. We name it LdaVec. The second one is similar to the second one but uses all topics as a search key. The difference is that each key is a topic name with a probability scope of the topic. For example, $Topic_1$ is a topic, then $Topic_1[0, 0.02)$ and $Topic_1[0.02, 0.04)$ are two different searching keys related to this topic. While handling a new case, we first calculate its topic vector and derive search key projections of all topics according to their topic values. Next, collecting cases by each searching key. Eventually, ranking cases by the number of common searching keys they share with the input case. For those with the same number of shared keys, rank them with the sum of probability gaps between them with the input case on each topic. The smaller the sum of the gap, the higher the ranking. We name this method LdaAll.
(4) Doc2Vec: Generate document vectors for cases using the Doc2Vec tool [5] and measuring similar between cases by Euclidean Distance.

Proposed Methods. Besides, for methods represented by formula 1, we try three different methods for selecting keywords and there are corresponding three distinct approaches for searching similar cases, namely, Tfidf_IG, Tfidf_WLLR and Tfidf_WFO.

4.1.3. Results

Metrics To evaluate the effect of statute recommendation, we use Recall, Precision and MAP as evaluation metrics. For each case in our dataset, its truly cited statutes are recorded. Hence, no matter where to recommend statutes for a case, as if a list of recommended statutes is generated, we could check each statute as cited by the input case. Let tp_i be the number of cited statutes in the first ith recommended statutes for an input case and g be the number of all cited statutes of the case, then the recall, precision and average precision (AP) at position i for this case is calculated with formula 4 and 5.

[5] https://radimrehurek.com/gensim/models/doc2vec.html

$$Recall@i = \frac{tp_i}{g}, Precision@i = \frac{tp_i}{i}, AP@i = \frac{1}{i}\sum_{j=1}^{i} Precision@j \times Cited@j, where \tag{4}$$

$$Cited@i = \begin{cases} 0, & \text{if } statute@i \text{ is not cited by the case;} \\ 1, & \text{if } statute@i \text{ is cited by the case.} \end{cases} \tag{5}$$

Recall, precision and AP at position i for the entire data set are the corresponding average values of all cases. While comparing the strategies for searching similar cases, we only adopt the average recall. For evaluating the ranking approaches, we use the average recall, precision and AP (i.e., MAP) of all five folds as the final evaluating value (i.e., adopted in Section 5).

How to Retrieve Cases. Fig. 3 shows the comparisons among TFIDF-related approaches, regarding the recall values of truly cited statutes by retrieving different top Kstatutes. We set the K in top K cases to 50 to make the comparison and the largest Kequals 50 too.

It proves that considering the POS tag of words in weighting cases would achieve better results than using TFIDF values as weights only. This is reasonable. In a divorce case, the most important part is the husband and wife's daily life, where a lot of verbs and nouns are used to describe. However, different cases would be more likely to have similar nouns, such as appellation of a related person, location, time and daily supplies. But husbands and wives of different cases would have different actions and even varying degrees of the same behavior. The behavior and the degrees are one of the key factors determining the statutes to be applied. So, cases with the same verbs or adverbs are more likely to cite the same statutes. It is the same reason for putting adjectives in our $\mathbb{S}$ set. However, Fig. 3 also tells that there are no explicit differences among the three candidate feature selection methods. So, in the rest of our approach, we choose the more widely used Information Gain theory for choosing statute candidates to be ranked.

In the following parts of this paper, statutes retrieved by Tfidf_IG approach are used for re-ranking them for the recommendation. In Fig. 3, the Tfidf_IG approach is also compared with other baseline approaches. The conclusion is that the LDA-

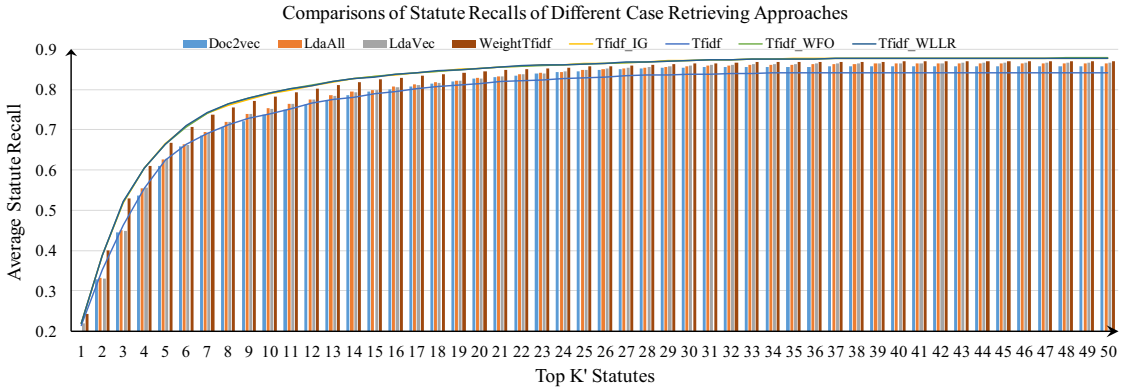

Fig. 3. Comparison of statutes recalls derived by different Case Searching strategies.

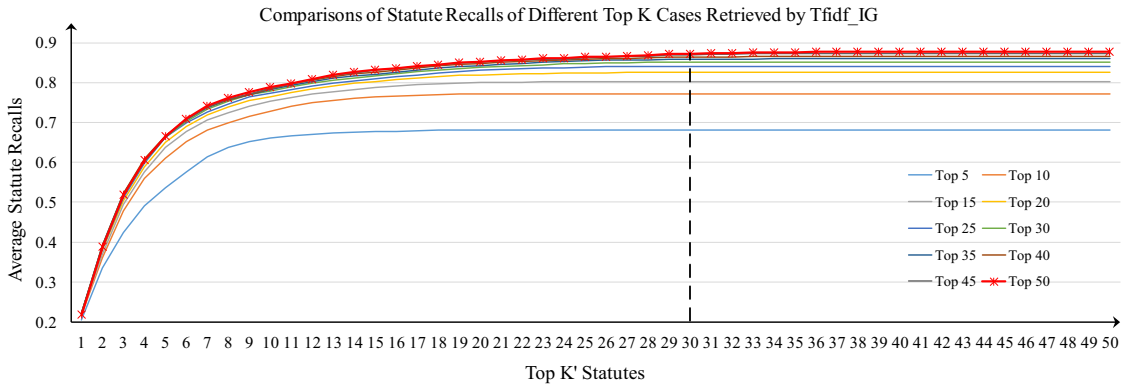

Fig. 4. Comparison of statutes recalls of different number of similar cases derived by TFIDF_IG.

related and Doc2vec methods have similar performance in retrieving similar cases for statute recommendation, while our Tfidf_IG outperforms them by about 0.05 in the average recall.

What are Proper K and K′. In a real application, the values of K and K'would affect the efficiency of the recommending system. They should be set with proper values. We hope they are as small as possible but achieve the best recall values at the same time. So, we try different K values and collect the recall values at different K's for the Tfidg_IG cases ranking strategy. The result is as shown in Fig. 4. It tells that there would be no notable differences between collecting statutes from 50 and 45 cases. Besides, deriving the Top 30 statutes would almost achieve the same suitable statute recall as the Top 50 ones. So, in our approach, we set K to 50 and $K'$to 30. However, please note that our main purpose is to verify that such a scheme is effective and we set a fixed set of values for all parameters in the experiment. These values do not ensure achieving the best performance. In different application scenarios, you can set different values for all parameters to seek the best performance.

4.2. Modeling Case-Statute Relation

Upon deriving all statute candidates, re-ranking is applied. There has not yet been any research on resorting statutes to the best of our knowledge. In this paper, we propose a learning-to-rank-based approach to conduct a secondary sort on statutes retrieved by CF methods. The novelty of our approach is twofold: adopting the text of statutes and defining a group of hand-crafted case-statute relational features. Among the most existing statute recommendation methods, the text of statutes is not used. For the others, they either use Bag-of-Words as text features or treat statutes text independently from cases.

The proposed case-statute relational features consist of basic features and novel features. Basic features are selected Unigram Pairs generated from case-statute pairs. Novel features contain information on some factors often considered by judges and lawyers when they handle divorce cases.

For each candidate statute of the input legal case, the values of each feature would be calculated according to the following definitions of each feature. Once the relation features of all statutes were prepared, they would be applied to the re-ranking model for ranking candidate statutes. Here is the definition of hand-carfted relational features.

4.2.1. Basic Feature: Unigram Pairs

Initial Unigram Pair: While TF-IDF is widely used in text representation, it can only represent two texts independently. In our system, we need to process the text of the case and its statutes together and generate the representation of the relations of the case and each statute. Therefore, we use Unigram Pair to represent cases and statutes. A Unigram Pair consists of a word from cases and a word from statutes. For each pair of case and statute, if the first word of Unigram Pair appears in case text and the second word appears in statute text, we set the value of the Unigram Pair to 1 or else we set it to 0.

Pre-conducted experiments show that Unigram Pair is better than TF-IDF in both classification and ranking approaches. The reason is that Unigram Pair features to model the relationship between a case and a statute directly. But when using TF-IDF features, case text and statute text are still considered independent items. Therefore, we think it is important to add other features representing the relation between cases and statutes to our case-statute relational features.

Selected Unigram Pairs: However, the size of the initial unigram pairs is too large to train the ranking model. There are more than 1.5 million unigram pairs in the prepared dataset. The learning algorithm would also get confused if we gave it many features. It would be better to select those pairs that are more likely to help soften statutes. If a pair could be used in distinguishing suitable statutes from unsuitable ones, it must be able to perform well in ranking statutes. So, we generate unigram pairs from cases with their cited statutes and cases with those not cited ones separately. Unigram pairs generated from cases with cited statutes are represented by $UP_{CiteTrue}$ and the others are represented by $UP_{CiteFalse}$. For each unigram pair up, its frequency of occurrence in both $UP_{CiteTrue}$ and $UP_{CiteFalse}$ in the case-statute pairs are counted, marked with $freqTrue(up)$ and $freqFalse(up)$ respectively. We select unigram pairs according to the ratio of $freqFalse()$ and $freqTrue()$ values as is shown in Fig. 5.

The rule for selecting up is that the more imbalanced the distribution of the word pair between cited case-statute pairs and not cited pairs, the more likely it is to help distinguish the suitableness of statutes. For $freqFalse(up) \neq 0$ and $freqTrue(up) \neq 0$, we select the pairs whose $freqTrue$ is bigger than f_1 and the ratio is less than $Threshold_1$ or more than

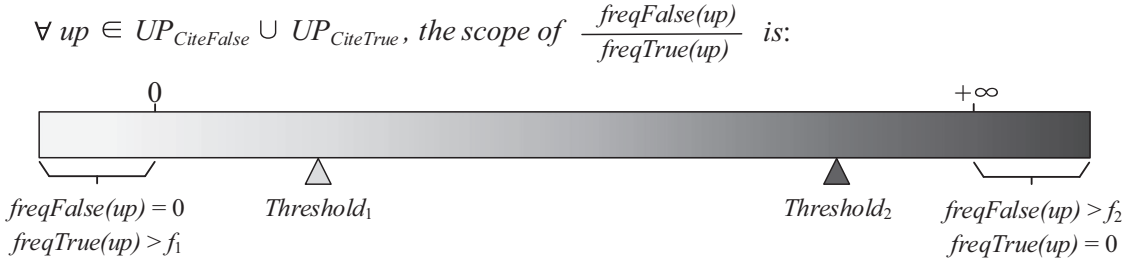

Fig. 5. The details for selecting Unigram Pairs for the case-statute relational features.

$Threshold_2$. If the $freqFalse(up) = 0$, we select those whose $feqTrue > f_1$. If the $feqTrue(up) = 0$, the pairs whose $freqFase > f_2$ are selected. All pairs are only selected from training data and for each training process in the cross-validation, we conduct the experiments for selecting unigram pairs separately. The bigger the f_1, f_2 and $Threshold_2$ are and the smaller the $Threshold_1$ is, the fewer pairs would be left as features. In the experiments, we set several groups of values for these thresholds, i.e., ($f_1, f_2, Threshold_1$ and $Threshold_2$), such as (5,10,10,50), (10,20,5,100), (10,150,1,150) and (15,200,1,150) etc, to test the performance of the relational features. Eventually, we set f_1 to 15, f_2 to 200, $Threshold_1$ to 1 and $Threshold_2$ to 100, and the average number of selected unigram pairs of all experiments are 3967. With these settings, the training efficiencies are proper and the performance is better. However, since we did not conduct a complete experiment for searching the best combinations of these parameters, the performance of the relational features reported in the Experiment section must not be the best. But it is enough to prove that our method is effective.

4.2.2. Words Overlap

match: The same words of case text and statute text can show correlations between cases and statutes in some way. Compared to statutes with few same words with the case, statutes with more words that appeared in a case are more likely to be cited in this case. We manually read documents of cases and find that some important words often appear in both case and statute texts. Obviously, these words help associate statutes to cases. It is hard to collect all these specific words and construct a global one-hot vector. Therefore, we use the number of words shared by the case and the statute of the concrete case-statute pair as the target feature (stop words are removed first).

Keywords and Key Unigram Pairs: We extract and filter keywords and key Unigram Pairs from documents by their frequency. We select words that appear in both cases and statutes first. Then reserve top k words appearing in both a case and its cited statutes as keywords. Key Unigram Pairs are selected in the same way. These keywords and key Unigram Pairs represent the citing principle between cases and statutes. We set k to 100 after several attempts.

Connection: Judge analysis tells us how judges summarize the descriptions of cases and how they make decisions or judgments. Therefore, we should pay more attention to words in judge analysis text. But we cannot use words to judge analysis directly since they are not the original case description. We make use of the judge analysis text of the training set and collect Unigram Pairs that both words appear in the judge analysis text from all Unigram Pairs. The value of these Unigram Pairs will be multiplied by 3 for emphasis.

4.2.3. Roles and Special Groups Consistency

Role: The Role features represent relations of roles that appeared in cases and statutes. Statutes usually explain what kind of people is relevant to them, e.g., couple, parents, brothers, sisters, etc. If a statute is applicable to a case, the roles mentioned in the case description are consistent with the roles in the statute text. On this basis, we design features that match if roles appeared in cases and statutes.

We find that statutes always express relevant people in unified ways. For example, if a statute is about parents and children, it will use written language instead of oral language. But in the case of texts, it will use various colloquial words in different cases because plaintiffs and defendants may come from different regions. In order to match roles in cases and statutes, we summarize all roles that may appear in cases and statutes. We cannot match roles simply by words because of the ambiguity of the text. Therefore, in addition to commonly used words of these roles, we need to estimate relations between roles and the plaintiff/defendant. If two roles have the same relation to the plaintiff/defendant, we consider they are matched. We extract features of matching roles in cases and statutes with information about roles or relations to the plaintiff/defendant. Since our dataset consists of divorce cases, we summarize all roles into five categories: plaintiff/defendant, parents of plaintiff/defendant, children of plaintiff/defendant, brothers/sisters of plaintiff/defendant and the third party in the marriage.

Special Group: We also found that sometimes statutes and cases describe a role with certain descriptions. E.g., children are often described by their ages and health conditions. For people with diseases or disabilities, special descriptions always appear in statutes or case texts in certain ways. In order to include this information in our system, we summarize some special groups and text rules of them and extract these features as well.

Eventually, if a role appears, we set the value of this role to 1 and otherwise 0. Then two vectors consists of 0 and 1 representing roles and special groups appearing in the case and statute are derived respectively. Afterward, we execute a join operation on two vectors to get a final representation of this feature.

4.2.4. Judgement Consistency

Approval vs. disapproval: Statutes cited by cases with the judgment of divorce may differ from those cited by cases with the judgment of disapproval. If a judge supported their divorce in a case, he or she would select statutes associated with follow-up issues of divorce like custody of children and division of property. But if he or she did not support their divorce, these statutes will not be used in the case. Therefore, we can divide all statutes into two categories: applicable and not applicable in a divorce. In general, there are words like "after divorce" or "in divorce" in statutes, if they are designed for follow-up issues of divorce. We also have statistics on how many divorced cases and other cases each statute is used in. The ratio of divorced cases and other cases will be a feature of a statute.

Meanwhile, we need to extract features from cases that can indicate if the couple would divorce. After analyzing training data, we discover that defendant's attitude has a major impact on the result of the divorce case. If the defendant agrees to the divorce or does not reply to a charge, a judge is more likely to support their divorce. The probability of no divorce increases if the defendant disagrees with their divorce. We also should take the plaintiff's statement into account. If the plaintiff describes the situation in detail, makes clear claims about their divorce and expresses negative feelings about their marriage, the judge may tend to agree with their divorce. What's worse, if there are conflicts between facts described by the plaintiff and defendant, the judge has to consider whose description is true. Additionally, we assume that possibility of divorce increases if the couple had filed for divorce before this case, which is usually mentioned in case descriptions.

Concrete Features: All factors we mentioned above have different impacts on applicable statutes of cases and can't determine if they will divorce. We represent each factor by a number to construct a feature vector. The feature vector indicates relations between cases and statutes in terms of judge results. Does the feature vector contain the following information: (1) If the statute contains words like "in divorce" or "after divorce"? (2) Ratio of divorced cases that cited the statute to all cases that cited the statute; (3) Ratio of undivorced cases that cited the statute to all cases that cited the statute; (4) Defendant's attitude towards the divorce, agree to 1, disagree to -1, unknown to 0; (5) Length of the content of the plaintiff's statements; (6) Number of plaintiff's claims; (7) If they had filed for divorce before, (8) Length of the content of the defendant's statements.

4.2.5. Relationship Consistency

Marriage Breach: In general, a judge usually grants a divorce by marriage breach, which is a significant factor in divorce cases. Some relevant statutes contain judgment criteria of marriage breach and these criteria are also mentioned in other statutes about divorce.

Vector for statute: Different statutes pay attention to different aspects of marriage breach, so we summarize some features about their marriage from statutes to describe their relationships. These features contain the following information: (1) if one couple has diseases that can be a reason for prohibiting marriages, (2) if one couple has a mental illness, (3) fraud in marriage, (4) arranged marriage, (5) if one couple commit crimes, (6) no affection in marriage, (7) if the couple didn't live together in marriage, (8) separation in marriage, (9) derailment in marriage, (10) bigamy, (11) difficulty in living together (e.g., one couple of gamble or domestic violence), (12) disappearance of one couple. We can judge if statutes contain information about these features by keywords and represent these features of statutes with a vector of 0/1, i.e., V_{sr}.

Vectors for case: We represent cases with two vectors in a similar way since case descriptions consist of descriptions from plaintiffs and defendants, i.e., V_{pr} and V_{dr}. In most cases, plaintiffs describe their relationships in more detail, so we can get more information from the plaintiff's statements. While features of fraud, gambling and bigamy can be extracted simply by keywords, other features are extracted with human summarized rules because they have multiple expression forms.

Integration: As mentioned in the previous section, if there is a conflict between the contents described by the plaintiff and defendant, the judge has to consider whose description is true. For example, if the plaintiff accuses the defendant of the derailment, but the defendant shows a denial of it, we should show questions about this factor in the value of the feature. Therefore, we use V_{sr} & V_{pr} * V_{dr} as final value of features of relationship consistency.

4.2.6. Features Based on LDA

Latent Dirichlet Allocation: Most cases focus on several fixed issues. Since we use divorce cases as our original data, most cases focus on topics like love breaks, children and property. Although we can annotate cases with common topic tags artificially, we do not want to omit some potential undiscovered topics. Therefore, we introduce Latent Dirichlet Allocation [44] features to simulate how humans find topics in documents.

In order to construct LDA features, we train LDA models with different types of texts and produce a topic distribution of each text. When training all LDA models, we adjust the number of topics based on topic coherence produced by the model and select topic number with the highest average topic coherence.

Simple shared model: In the first step, we train an LDA model with all documents of cases and statutes and then we can map cases and statutes to the same vector space. We can calculate the topic similarity of statute text and case text based on their LDA vectors.

Independent models: Due to differences between case text and statute text, we train LDA models for cases and statutes, respectively. If the topic distribution of case is L_{case} and topic distribution of statute is $L_{statute}$, we use $L_{case} \otimes L_{statute}$ as the feature vector of this case-statute pair. We need not map statutes and cases to the same vector space in this way and thus can train more accurate LDA models for both cases and statutes.

Shared model based on analysis paragraph: Finally, we can train an LDA model with the text of judge analysis in cases from the training set to represent relations of cases and statutes. Text of judge analysis tells us how judges analyze facts in case descriptions and make a decision on the judgment. The topic distribution of judge analysis may have associations with relations between cases and their applicable statutes. For test data, we will combine the text of statutes and case descriptions as input to the LDA model to get the topic distribution.

4.3. Ranker

Most ranking approaches can be categorized into point-wise ranking, pair-wise ranking and list-wise ranking. Pair-wise ranking formulates ranking task as a classification problem and focuses on relative preference between two items. List-wise

ranking tries to directly optimize the value of evaluation measures (i.e., loss), averaged over all queries in the training data. In this paper, we adopt both pair-wise and list-wise ranking methods to train the ranker. For pair-wise ranking algorithms, we tried SVM Rank[45], RankBoost[46], and RankNet[47]. However, SVM Rank in SVM Light performs best by adopting the relational features in this task. For the list-wise ranking method, we tried the state-of-the-art method DLCM (Deep Listwise Context Model) [34], taking the relational features as inputs. Details of comparisons among rankers are shown in the next section.

Please note that the ranking algorithm is not one of the contributions of this paper, but the relational features and the best way to utilize the relational features are.

While preparing the training data for rankers, the score for truly cited statutes is set to 5 and the others are set to 1. The inputs are encoded with different methods before transferring them to DLCM. For pair-wise rankers, each query is a list of case-statute pairs and each case-statute pair is a vector representing the relation between the case and the statute. Each data sample is a query for list-wise rankers and its corresponding list of candidate statutes. For ranking, all experimental results are obtained via fivefold cross-validation experiments. In each experiment, we use 3/5 cases in the data set for model training, another 1/5 for parameter tuning, and the final 1/5 for testing.

5. Experiments and Evaluation

5.1. Setup

For checking the effectiveness of the proposed relational features in re-ranking statutes, we conduct experiments for comparing our approach with other statute recommendation approaches, including binary classification, multi-label classification, collaborative filtering (i.e., search engine), and the state-of-the-art neural network-based list-wise ranking approach DLCM, as well as different ways to utilize relational features. The compared methods are listed as follows:

(1) None-learn-to-rank: they are **Searching Engine** (i.e., collaborative filtering, ranking statutes according to formula 3), and **C-MultiLabel** (i.e., viewing the suitable statutes as labels of cases and treating it as a multi-label classification problem of cases). Concretely, we tried both LibSVM [48] and BERT for training the classifiers. For LibSVM, we use the TF-IDF of cases for multi-label classification. While using BERT, we first encode texts of cases with a pre-trained BERT model, then use the [CLS] tokens' embedding as text representative, and eventually transfer them to a softmax output layer with multiple neurons. We find that utilizing BERT derives better results and we only report the results of using BERT in the experimental results, i.e., **C-MultiLabel-BERT** in Table 2. We firstly set the BERT model untrainable to train the multi-labels classifier, i.e., *weights* are fixed. For comprehensively comparing, we also conduct experiments by setting the BERT model trainable, i.e., fine-tuning BERT weights during training the classifier. The fine-tuning approach is marked by **C-MultiLabel-BERT-Tuning** in Table 2.

(2) Point-wise ranking: it is treating the case-statute suitableness as a 0/1 classification problem, i.e., viewing each case-statute pair as a to be classified data instance. The classification output can be viewed as a ranking score for each statute. For LibSVM, we use the relational features for binary classification. While using BERT, we firstly encode texts of

Table 2
Recall and precision of different learn to rank approaches.

Approaches	Recalls (%)				Precisions (%)			
	@1	@3	@5	@10	@1	@3	@5	@10
Search Engine	21.86	51.84	66.56	78.89	59.25	48.58	37.80	22.80
C-MultiLabel-BERT	18.38	42.32	62.73	77.05	47.53	40.54	34.82	22.15
C-MultiLabel-BERT-Tuning	17.72	42.46	62.88	77.26	47.31	40.55	34.83	22.17
C-MultiLabel-LBERT	17.26	42.43	62.67	77.13	47.25	40.55	34.82	22.16
C-MultiLabel-LBERT-Tuning	17.95	42.61	62.84	77.22	47.48	40.56	34.83	22.17
C-Binary-BERT	22.12	51.04	64.43	73.82	60.04	48.35	37.03	21.14
C-Binary-BERT-Tuning	22.86	52.41	66.25	74.63	60.63	49.14	38.35	21.53
C-Binary-LBERT	23.26	51.47	65.83	74.60	61.87	48.37	37.62	21.53
C-Binary-LBERT-Tuning	23.53	53.00	66.41	74.96	62.01	51.04	38.11	21.67
DLCM-BERT	26.05	54.25	67.81	79.74	73.42	53.17	38.45	23.52
DLCM-BERT-Tuning	26.11	54.28	67.80	79.45	73.44	53.18	38.45	23.51
DLCM-LBERT	26.10	54.22	67.80	79.46	73.44	53.16	38.31	23.50
DLCM-LBERT-Tuning	26.12	54.64	67.85	79.76	73.45	53.17	38.49	23.54
DLCM-Basic	23.70	50.45	65.55	79.17	62.16	47.02	37.57	22.96
DLCM-Basic + Novel	23.71	52.01	66.89	78.84	62.20	48.38	38.12	22.83
R-SVM-BERT	23.79	52.56	66.58	79.04	64.74	49.30	37.89	22.94
R-SVM-LBERT	23.81	53.12	66.23	79.16	65.52	51.74	37.75	22.98
R-SVM-Basic	25.46	54.22	67.18	79.42	70.38	52.90	38.22	23.03
R-SVM-Basic + Novel	**26.52**	**55.43**	**68.14**	**79.94**	**74.83**	**55.03**	**38.59**	**23.67**
Gold Standard	*38.13*	*82.38*	*86.70*	*87.14*	*99.98*	*76.98*	*50.56*	*25.66*

cases and statutes with pre-trained BERT model, then concate the case's [CLS] embedding and the statute's [CLS] embedding, and eventually transfer the entire embedding vector to an output layer with only one neuron. Just like marking models of multi-labels classifiers, we use **C-Binary-BERT** and **C-Binary-BERT-Tuning** to mark approaches of setting the BERT model untrainable and trainable, respectively, as in Table 2. Please note that our **C-Binary-BERT-Tuning** model has the same principle and structure as the state-of-the-art point-wise learn-to-rank models using BERT or Transformer, such as the ones in [35,49,36].

(3) List-wise ranking: we only check the performance of the the-state-of-the-art neural network-based list-wise ranking approach DLCM in ranking statute candidates with different textual embeddings, including BERT, (**DLCM-BERT** and **DLCM-BERT-Tuning** for fixed and improved BERT respectively), and the proposed case-statute relational features, i.e., **DLCM-Basic** and **DLCM-Basic + Novel**. While using BERT to encode cases and statutes, embeddings of [CLS] tokens are retrieved and concatenated as input of DLCM.

(4) Pair-wise ranking: we use SVM Rank to represent pair-wise ranking algorithms. For comparing, we use three different types of features as the input of SVM Rank, namely, BERT (i.e., **R-SVM-BERT**, utilizing BERT the same way as in **C-Binary-BERT**), basic features (i.e., **R-SVM-Basic**) and both basic and novel relational features (i.e., **R-SVM-Basic + Novel**).

In addition to the base version of BERT model, we also adopt the version pre-trained on a large amount of Chinese legal judgment documents, i.e., LBERT [10] for comparison and the experimental results are corresponding to **C-MultiLabel-LBERTX(-Tuning)**, **C-Binary-LBERT(-Tuning)**, **DLCM-LBERT(-Tuning)** and **R-SVM-LBERT** in Table 2. The ways to utilize LBERT are the same as using BERT (The differences lie in the model checkpoints and tokenizers.).

We conduct feature ablation experiments to compare the effectiveness of different novel features. The same fivefold cross-validation strategy is adopted in training classifiers used in training rankers. For SVM Rank, we tuned the parameter *c* from 30 to 30000. For each fold of the training DLCM model, the input list is 30 according to the number of candidate statutes and dimensions of input documents set to a concrete size of different features. We only tuned the hidden unit number from 10 to 100 and the batch size from 8 to 32. For the proposed case-statute relational features, average dimensions of each type of feature are shown in Table 3.

5.2. Results and Discussion

The metrics for evaluation are those illustrated in Section 4.1.3. We want to find answers to the following three Research Questions (i.e., **RQ**s) according to the experimental results.

RQ1 How much can the case-statute relational features improve recommending results comparing with other approaches?.

Results. Table 2 shows recalls and precisions of different statute recommendation approaches of different Top K recommended statutes. It shows that utilizing the proposed case-statute relational features for SVM Rank (i.e., **R-SVM-Basic + Novel**) would achieve the best statute recommendation performance for divorce cases. The recall and precision at the top 1 statute achieve 26.52% and 74.83%, respectively. They are 4.66 and 15.58 percent higher than those of the first recommending stage, i.e., retrieving statutes by collaborative filtering and ranking by their weights (**Search Enging**). Besides, it also outperforms the combination of **DLCM** and **BERT**/**LBERT** (no matter whether fine-tuned or not), which are the state-of-the-art methods of list-wise ranking and neural network-based language models. It proves that the proposed secondary sort strategy works for re-ranking candidate statutes of divorce cases. The performances of **C-Binary-X** and **C-MultiLabel-X** in Table 2 show that modeling the statute ranking issue as a classification problem is not a proper strategy for recommending statutes for divorce cases, even if the state-of-the-art language models BERT/LBERT are used to encode texts, no matter fine-tuned or not. We also draw the MAP curves of three competitive approaches (i.e., **DLCM-LBERT**, **R-SVM-Basic**, and **R-SVM-Basic + Novel**), as well as the **Searching Engine**, in Fig. 6. All approaches achieve the highest MAP at K = 3. However, the average number of truly cited statutes by each case is 4; the theoretical highest MAP should be achieved at K = 4. This also indicates that there is still space for improvement.

Discussion. It is reasonable that the performance of multi-label classification is the worst. It does not take the content of statutes into consideration and there are 456 labels for 19860 data instances. Besides, the distributions of all labels are unbalanced. Almost 10 k cases cite the most frequently cited statute and the least frequently one is cited by only one case, while the average cited frequency of all statutes is 154. So, it is really difficult for a multi-label classification model to solve this problem. Although the binary classification achieves a better result than multi-label classification, it does not utilize the

Table 3
Number of dimensions of each type of relational features.

Feature	Unigram Pairs	Words Overlap	Role and Special Group Consistency	Judgement Consistency	Relationship Consistency	LDA
# of Dimensions	3967	250	10	7	11	163

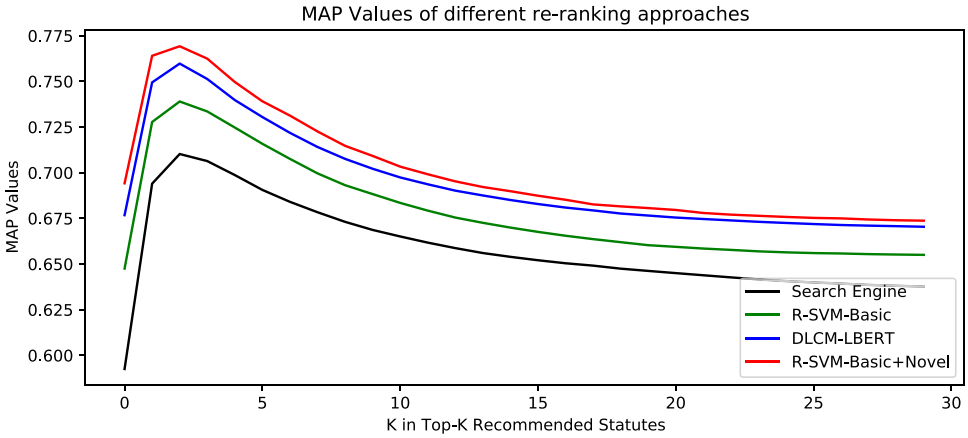

Fig. 6. MAP@Top K of four different re-ranking approaches.

differences between two statutes cited by the same case, which means it is more like a point-wise sorting method. It is no wonder that the binary classification outperforms the search engine only at the top 1 statute. There is another reason for the bad performance of binary classification and it is an unbalanced 0/1 distribution of data instances. On average, for each case, there are four positive case-statute pairs and twenty-six negative pairs.

We also tried two methods to balance 0/1 distribution for the binary classification model. The statute weights are to reserve the same number of negative pairs as the positive pairs for each case and those negative pairs with high statute weights (i.e., formula 3). The second method is to calculate a weight for each non-cited statute candidate for each case, similar to calculating TF-IDF, which is named SF-ICF (i.e., Statute Frequency-Inverse Case Frequency). Thus, statutes with high SF-ICF would be reserved as negative data instances. Statute Frequency is the frequency of the candidate cited by similar cases and Case Frequency is the frequency of the statute being a candidate of cases. Table 4 shows the recalls and precisions of the binary classification on 0/1 balanced data sets. Although there are slight improvements by balancing the data set, it still cannot achieve as good performance as list-wise ranking approaches.

We tested the performance for list-wise learning to rank algorithm DLCM by feeding it with different features, i.e., text embedding derived by neural network language model BERT/LBERT and the proposed case-statute relational features. It turns out that BERT/LBERT is more suitable for a neural network-based list-wise ranking approach, while hand-crafted relational features are more suitable for pair-wise SVM Rank. The results are reasonable. The various dimensions of features acquired by BERT/LBERT are interrelated and they should be viewed as a whole. Exactly, DLCM also treats each input vector as a whole and treats each dimension fairly. However, there are several different types of relational features. There are no strong connections among different features. It is not a proper way to connect between different dimensions of different data instances. But SVM Rank would focus on learning from the differences of the same dimension among different data instances. So, the proposed relational features help SVM Rank achieves better results. Besides, the size of relational features(i.e., 4408 dimensions on average) is too large for DLCM to handle.

One interesting observation is that, although adapting and improving BERT/LBERT would not make a big difference for any model that uses BERT/LBERT, the concrete impacts made by fine-tuning were slightly different on different models. For example, fine-tuned BERT/LBERT would increase results for both **C-Binary-(L) BERT** and **DLCM-(L) BERT** while decreasing the result of **C-MulitLabel-(L) BERT**. One possible interpretation may be that much more parameters in BERT/LBERT would make the pitiful multi-label classification model lose the direction of optimization compared with the much easier binary classification task. As to DLCM, the reason may be that the performance of the model is much more reliant on DLCM but not BERT/LBERT. This may also be the reason that the improvements made by fine-tuning BERT/LBERT for **C-Binary-(L) BERT** is also slightly higher than that for **DLCM-(L) BERT**.

Since we use two different pre-trained language models, i.e., BERT and LBERT, it is necessary to discuss the effects of the two on different models. In theory, using LBERT, a model pre-trained with more legal documents, should achieve better

Table 4
Recall and precision of binary classification models.

Balanced Strategy		Recalls (%)				Precisions (%)			
		@1	@3	@5	@10	@1	@3	@5	@10
C-Binary-BERT	Unbalanced	22.12	51.04	64.43	73.82	60.04	48.35	37.03	21.14
	Simple	22.84	50.73	64.80	77.22	61.31	47.99	37.09	22.19
	SF-ICF	23.55	51.56	65.51	78.13	62.00	48.40	37.45	22.57

results than basic BERT in this task. However, the experimental results show that this assumption is not always true in any condition. For example, for approaches of **C-Binary-X**, using LBERT can get better results than using BERT obviously. But for **C-MultiLabel-X** and **DLCM-X**, the effectiveness of LBERT and BERT are comparable. The reason is the same as discussed in the previous paragraph, i.e., (1) the pre-trained language models are not suitable for **C-MultiLabel-X**, and (2) models of **C-Binary-X** depend more on the pre-trained language models than models of **DLCM-X** (The better the language model is, the higher performance will the **C-Binary-X** classifiers achieve.).

The last row of Table 2 are the highest recalls and precisions at different places (i.e., Golden Standard) that the ranker could achieve theoretically by re-ranking statute candidates retrieved by the Search Engine. They are calculated by putting all truly cited candidates at the top of the list. It is obvious that there is still a big performance gap between the **R-SVM-Basic + Novel** and the Golden Standard. It implies that the statute recommendation is still a problem to be solved with much room for improvement.

RQ2 To what extent do the novel relational features contribute to the re-ranking approach?

Results. In Table 2, we compare the performances of using basic features independently and utilizing them with novel features for both DLCM and SVM Rank. After adding novel features, there is nearly no improvement of DLCM at the top 1 place. But there is a slight improvement at the top 3 places, i.e., 1.56% and 1.36% for recall and precision, respectively. For SVM Rank, the improvement happens at the top 1 place and both recall and precision achieve the highest values among all different approaches.

Discussion. As discussed in *RQ1*, the relational features are more suitable for SVM Rank than DLCM, so the novel features utilized by SVM would achieve greater improvements than those utilized by DLCM. However, the novel features do not bring a very big improvement to SVM Rank compared with basic features. The main reason is that there is a big difference between the number of their feature dimensions, i.e., 3967, on average for basic *VS*. 441 for a novel. Besides, while the novel features are used single input of SVM Rank, the results are not as good as those of using basic ones alone. In other words, the amount of information on novel features is not as great as that on basic features, but there are also certain different effective information.

Although the contribution of distinct information contained by novel features to improve Recall and Precision is not very obvious, it is outstanding in improving the recommendation effectiveness of low-frequently cited statutes. Fig. 7 shows the sums of citing frequencies of top 10 statutes recommended by **Search Engine**, **R-SVM-Basic**, and **R-SVM-Basic + Novel** on average to each testing fold. The searching engine trends recommend frequently cited statutes for all cases, but it has a poor performance. After applying the ranker considering basic features, the citing frequencies of the top three most frequently recommended statutes have suddenly dropped. This leads to those cases that did not cite or only cite one commonly used the statute to get the correct citation in the top three recommendations. Then the average recalls and precisions improve significantly. After adopting novel features, the citation frequency impact is reduced again. It proves that the proposed novel features successfully constructed the semantic association between cases and statutes and made the ranking model improve the prediction effect.

However, the decreasing recall improvements of the top five statutes (i.e., 0.96%) made by adding novel features (compared with the improvements of top 3 ones, i.e., 1.21%) indicates that there must be some truly cited frequently used statutes put outside the top five by mistake. So, a better way to integrate the citation frequency of statutes and semantic connections between cases and statutes needs to be investigated to improve the re-ranking performance.

RQ3 What kinds of human interpretable information in cases and statutes play a more critical role in ranking statutes than others?.

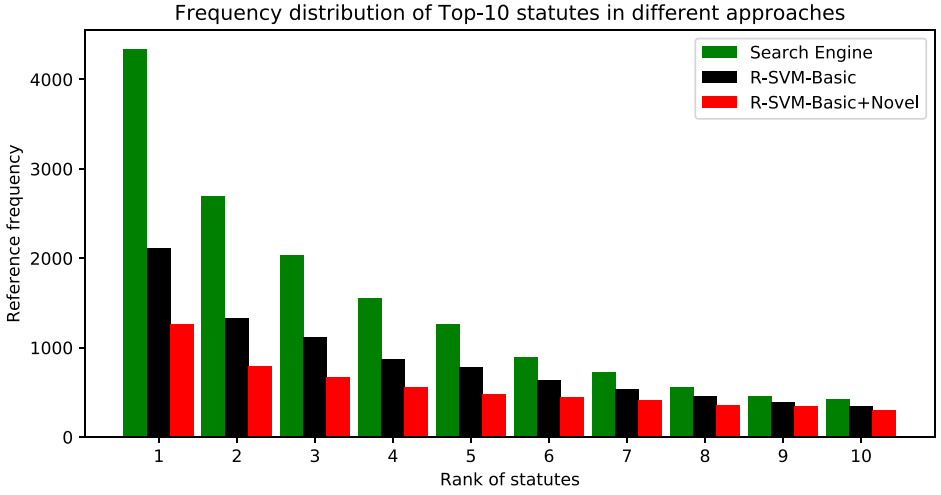

Fig. 7. Comparison of reference frequencies of top 10 recommended statutes among different approaches.

Table 5
Feature ablation results by using MAP@1 measuring performance.

Words Overlap	LDA	Role and Special Group Consistency	Judgement Consistency	Relationship Consistency
0.6810	0.6886	0.6893	0.6893	0.6908
0.6791	0.6878	0.6880	0.6904	–
0.6782	0.6867	0.6891	–	–
0.6774	0.6865	–	–	–

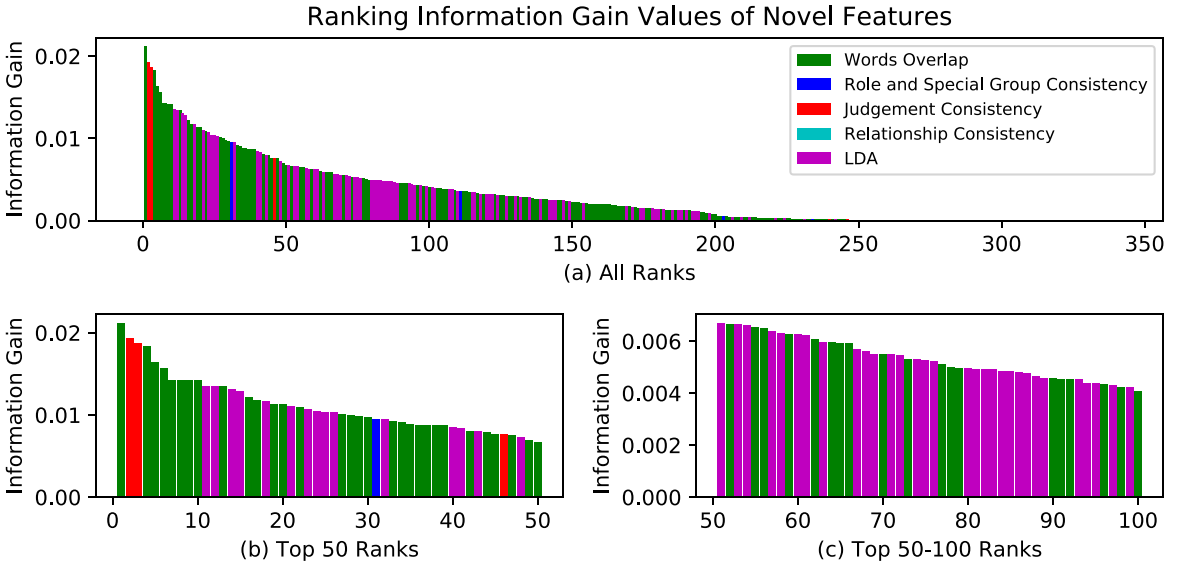

Fig. 8. Information gains of novel features in the binary classification of case-statute pairs.

Results. There are five types of interpretable features for ranking statute candidates in the case-statute relational features. The feature ablation experiments are conducted to investigate the effectiveness of different features. Table 5 shows the feature ablation results where MAP@1 is used for measuring the model (trained by R-SVM-Basic with selected novel features) performance. It tells that the words overlap and LDA are the most effective features in measuring case-statute relations for re-ranking statutes. Besides, we calculate the information gain of each dimension of the novel features in classifying the case-statute pairs into 0/1 classes, where 0 represents that the statute is not suitable for the case and one represents it is suitable. Fig. 8 shows the overall comparison of all dimensions and details of the top 50 and top 50–100 dimensions separately. Overall, it is obvious that words overlap and LDA are also the most important features in determining the suitableness of statutes.

Discussion. The re-ranking goal is to put truly suitable statutes at the top of the recommendation list. The goal of suitableness classification is to distinguish between suitable statutes and unsuitable ones. The two goals are essentially the same. So, the main conclusion derived from Table 5 and Fig. 8 is the same: overall, words overlap and LDA are the most effective features.

However, the IG values help us dig deeper into details about which dimensions are more critical and there are new discoveries. For example, there are several dimensions of **Role** and **Judgement** consistency features that are more effective than the others. This implies that the conclusion drawn from Table 5 is mainly caused by a large number of dimensions of words overlap and LDA features. In order to further compare the importance of different features, we tracked the concrete definitions of every single dimension whose IG value is in the top 50 list and they are shown in Table 6.

According to these definitions, we can sort out the guidelines for writing case descriptions and judgment documents that help recommend suitable statutes. For example, we find that dimensions of words that overlap with high IG values are mainly nouns and verbs frequently used in statute contents. This indicates that when describing the basic information of a case, adopt noun or verb terms used to define the statutes as much as possible, helping to locate the applicable statutes. Furthermore, it shows the importance of finding legal expertise to help write case descriptions and the necessity of improving the legal knowledge of ordinary people. Besides, for LDAs, features calculated through the *Shared model based on analysis paragraph* have higher IG values. This implies that when the judge writes the judgment document, he or she should explain the relationship between the case description and the applicable statute in the case analysis section and construct a reason-

Table 6
Definitions of feature dimensions with high IG values and their indicated instructions.

Feature Type	Rank of Dimension	Definition of the Dimension	Indicated Instructions
Words Overlap	Top 50	Frequently used nouns and verbs in statutes	(1) Invite legal professionals to help write case description. (2) Improve the general public's legal knowledge.
LDA	Top 50	*Shared model based on analysis paragraph*	Explain the relationship between the case description and the applicable statutes in detial in the case analysis section while writing Judgement Documents.
Judgement Consistency	No. 2	Defendant's Attitude	The defendant should express his/her attitude clearly, sin-ce it may affect the judge's judgment and thus the suitab-leness of statutes.
	No. 3	The first time filed for divorce	Whether to file a divorce lawsuit multiple times is critical. Never hide the previous divorce lawsuit nor make up ones in order to win the sympathy of the judge.
	No. 46	"during/after the divorce" in the statute	Describe clearly the performance of the two parties during the divorce and the claims after the divorce.
Role and Special Group Consistency	No. 31	Children of plaintiff/ defendant	Describe clearly the situation of children the plan of raising children after divorce.

ing process in detail to provide valuable information for recommending the applicable statutes to new cases in the future. Table 6 also shows more details about the indicated instructions on writing judgment documents of each dimension.

5.3. Scalability

The scalability of the proposed divorce cases' statutes recommending approach is reflected in handling the increasing scale of historical cases, applicable statutes, similar cases, and statute candidates.

With the increasing of historical cases, the search base for retrieving similar cases and the training data set would expand. As to the searching strategy, i.e., based on TFIDF_IG, the TF_IDF vectors of all cases should be reconstructed and the keywords derived through calculating the IG of each word in determining the applicable statutes should be re-generated. However, the search base could not be updated until the number of newly collected cases reached a certain threshold. As to the re-ranking strategy, more cases could be used to train the ranker. The case-statute relational features could be easily re-calculated.

As the applicable statutes are increasing, the only thing that needs to be updated is to collect the specific content of these statutes to re-calculate the features for training the ranker. The search base and search strategy for similar cases would not be influenced by the number of applicable statutes.

In this paper, we retrieve the Top 50 similar cases and the Top 30 statute candidates. However, as the number of cases for building the online approach, the two values may change. But the process of building the approach would not be affected.

6. Conclusion and Future Work

In this paper, we propose an approach for recommending statutes for divorce cases. The approach consists of two steps: retrieving similar cases to collect statute candidates and re-ranking statute candidates for recommendation. The candidate statutes preparation is essentially a collaborative filtering strategy. Different concrete algorithms for measuring the similarity of divorce cases can be adopted in the strategy. Besides, the re-sorting goal can be implemented through training rankers using different Learn to Rank algorithms. The innovation of the proposed approach mainly lies in defining and generating the hand-crafted case-statute relational features for rankers. The advantage of the relational features is that they are interpretable by human beings. According to the effectiveness of different features in recommending statutes, different written instructions for judicial cases could be concluded. In the future, cases written following these instructions would receive more accurate, suitable statute recommendations. Another contribution of this paper is the proposed similar cases retrieving method, which combines the TF_IDF vector of the document, POS tag of words, and the Information Gain algorithm.

Experimental results show that the proposed similar case retrieving methods outperform the others. Combining the relational features with SVM Rank would achieve the best statute recommendation results compared with baseline re-ranging approaches. By digging into the analysis of the importance of each dimension of the relational features, we find that the best way to improve statute recommendation performance is to describe the case with legal terms used in statutes, including nouns and verbs. Besides, suppose the judge could make it clear the connection between the facts and suitable statutes while writing judgment documents for historical cases. In that case, they will provide very valuable information for recommending statutes to newly issued cases and promote the recommending performance.

However, there is still a big gap between our recommendation result and the gold standard. The main reason is that the proposed relational features and existing methods do not set up a real reasoning model for constructing the connection between the case and the statutes. For setting up the reasoning process, deep semantic information of both cases and statutes should be understood. So, defining an abstract reasoning model manually and mining a knowledge base for reasoning are the future works of this paper.

CRediT authorship contribution statement

Chuanyi Li: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing - original draft, Writing - review & editing, Formal analysis, Funding acquisition. **Jidong Ge:** Conceptualization, Validation, Project administration, Writing - review & editing, Supervision, Funding acquisition. **Kun Cheng:** Methodology, Software, Validation, Writing - review & editing. **Bin Luo:** Validation, Funding acquisition, Writing - review & editing. **Victor Chang:** Validation, Funding acquisition, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Key R&D Program of China (2016YFC0800803), the National Natural Science Foundation of China (No. 61802167), and VC Research (VCR 0000070) for Prof. Chang. Jidong Ge and Victor Chang are the corresponding authors.

References

[1] G. Mu, What social problems are reflected behind the increase in the divorce rate"improve the ability of the new generation of chinese to love, People's Forum 23 (2019) 62–64.
[2] P.L. Chase-Lansdale, E. Mavis, The impact of divorce on life-span development: Short and long term effects, Journal of Marriage and Family 10 (1990) 105–150.
[3] S.S. Nagel, Applying correlation analysis to case prediction, Tex. L. Rev. 42 (1963) 1006.
[4] J.A. Segal, Predicting supreme court cases probabilistically: The search and seizure cases, 1962–1981, American Political Science Review 78 (4) (1984) 891–900.
[5] Y. Liu, Y. Chen, W. Ho, Predicting associated statutes for legal problems, Inf. Process. Manage. 51 (1) (2015) 194–211, https://doi.org/10.1016/j.ipm.2014.07.003, url:https://doi.org/10.1016/j.ipm.2014.07.003.
[6] Y. Feng, J. Ge, C. Li, L. Kong, F. Zhang, B. Luo, Statutes recommendation using classification and co-occurrence between statutes, in: X. Geng, B. Kang (Eds.), PRICAI 2018: Trends in Artificial Intelligence - 15th Pacific Rim International Conference on Artificial Intelligence, Nanjing, China, August 28–31, 2018, Proceedings, Part II, Vol. 11013 of Lecture Notes in Computer Science, Springer, 2018, pp. 326–334. doi:10.1007/978-3-319-97310-4_37. url:https://doi.org/10.1007/978-3-319-97310-4_37..
[7] Y. Feng, C. Li, J. Ge, B. Luo, http://proceedings.mlr.press/v101/feng19a.htmlImproving statute prediction via mining correlations between statutes, in: W.S. Lee, T. Suzuki (Eds.), Proceedings of The 11th Asian Conference on Machine Learning, ACML 2019, 17-19 November 2019, Nagoya, Japan, Vol. 101 of Proceedings of Machine Learning Research, PMLR, 2019, pp. 710–725. url:http://proceedings.mlr.press/v101/feng19a.html..
[8] C. Li, J. Ye, J. Ge, L. Kong, H. Hu, B. Luo, A novel convolutional neural network for statutes recommendation, in: X. Geng, B. Kang (Eds.), PRICAI 2018: Trends in Artificial Intelligence - 15th Pacific Rim International Conference on Artificial Intelligence, Nanjing, China, August 28–31, 2018, Proceedings, Part I, Vol. 11012 of Lecture Notes in Computer Science, Springer, 2018, pp. 851–863. doi:10.1007/978-3-319-97304-3_65. url:https://doi.org/10.1007/978-3-319-97304-3_65..
[9] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018). arXiv:1810.04805. url:http://arxiv.org/abs/1810.04805.
[10] H. Zhong, Z. Zhang, Z. Liu, M. Sun, Open Chinese Language Pre-trained Model Zoo, Tech. rep. (2019). url:https://github.com/thunlp/openclap.
[11] H. Prakken, G. Sartor, Law and logic: A review from an argumentation perspective, Artif. Intell. 227 (2015) 214–245.
[12] G. Contissa, F. Lagioia, M. Lippi, H.-W. Micklitz, P. Palka, G. Sartor, P. Torroni, Towards consumer-empowering artificial intelligence, in: International Joint Conference on Artificial Intelligence, 2018, pp. 5150–5157..
[13] H. Zhong, Y. Wang, C. Tu, T. Zhang, Z. Liu, M. Sun, Iteratively Questioning and Answering for Interpretable Legal Judgment Prediction, Proceedings of the AAAI Conference on Artificial Intelligence 34 (01) (2020) 1250–1257, number: 01..
[14] N. Xu, P. Wang, L. Chen, L. Pan, X. Wang, J. Zhao, Distinguish Confusing Law Articles for Legal Judgment Prediction, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 3086–3095..
[15] Y. Shao, J. Mao, Y. Liu, W. Ma, K. Satoh, M. Zhang, S. Ma, Bert-pli: Modeling paragraph-level interactions for legal case retrieval., in: IJCAI, 2020, pp. 3501–3507..
[16] Y. Ma, Y. Shao, Y. Wu, Y. Liu, R. Zhang, M. Zhang, S. Ma, Lecard: A legal case retrieval dataset for chinese law system, Information Retrieval (IR) 2 (2021) 22.
[17] Y. Wu, K. Kuang, Y. Zhang, X. Liu, C. Sun, J. Xiao, Y. Zhuang, L. Si, F. Wu, De-biased court's view generation with causality, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 763–780.
[18] Q. Li, Q. Zhang, Court opinion generation from case fact description with legal basis, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 14840–14848..
[19] C. Xiao, X. Hu, Z. Liu, C. Tu, M. Sun, Lawformer: A pre-trained language model for chinese legal long documents, AI Open (2021)..
[20] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019)..
[21] Z. Yu, M. Tian, Z. Wang, B. Guo, T. Mei, Shop-type recommendation leveraging the data from social media and location-based services, ACM Trans. Knowl. Discov. Data 11 (1) (2016) 1:1–1:21. doi:10.1145/2930671. url:https://doi.org/10.1145/2930671..
[22] G. Liu, Y. Fu, G. Chen, H. Xiong, C. Chen, Modeling buying motives for personalized product bundle recommendation, ACM Trans. Knowl. Discov. Data 11 (3) (2017) 28:1–28:26. doi:10.1145/3022185. url:https://doi.org/10.1145/3022185..
[23] C. Park, D. Kim, M.-C. Yang, J.-T. Lee, H. Yu, Click-aware purchase prediction with push at the top, Inf. Sci. 521 (2020) 350–364, https://doi.org/10.1016/j.ins.2020.02.062, url:http://www.sciencedirect.com/science/article/pii/S0020025520301456.
[24] S. Chou, T. Hsing, Text mining technique for chinese written judgment of criminal case, in: H. Chen, M. Chau, S. Li, S.R. Urs, S. Srinivasa, G.A. Wang (Eds.), Intelligence and Security Informatics, Pacific Asia Workshop, PAISI 2010, Hyderabad, India, June 21, 2010. Proceedings, Vol. 6122 of Lecture Notes in Computer Science, Springer, 2010, pp. 113–125. doi:10.1007/978-3-642-13601-6_14. url:https://doi.org/10.1007/978-3-642-13601-6_14..

[25] J. Zeng, J. Ge, Y. Zhou, Y. Feng, C. Li, Z. Li, B. Luo, Statutes recommendation based on text similarity, in: 14th Web Information Systems and Applications Conference, WISA 2017, Liuzhou, Guangxi Province, China, November 11–12, 2017, IEEE, 2017, pp. 201–204. doi:10.1109/WISA.2017.52. url:https://doi.org/10.1109/WISA.2017.52..
[26] G. Hu, X. Dai, F. Qiu, R. Xia, T. Li, S. Huang, J. Chen, Collaborative filtering with topic and social latent factors incorporating implicit feedback, ACM Trans. Knowl. Discov. Data 12 (2) (2018) 23:1–23:30. doi:10.1145/3127873. url:https://doi.org/10.1145/3127873..
[27] C. Liu, J. Cao, S. Feng, Leveraging kernel-incorporated matrix factorization for app recommendation, ACM Trans. Knowl. Discov. Data 13 (3) (2019) 31:1–31:27. doi:10.1145/3320482. url:https://doi.org/10.1145/3320482..
[28] G.R. Lima, C.E. Mello, A. Lyra, G. Zimbrao, Applying landmarks to enhance memory-based collaborative filtering, Inf. Sci. 513 (2020) 412–428, https://doi.org/10.1016/j.ins.2019.10.041, url:http://www.sciencedirect.com/science/article/pii/S0020025519310096.
[29] S. Jiang, S.-C. Fang, Q. An, J.E. Lavery, A sub-one quasi-norm-based similarity measure for collaborative filtering in recommender systems, Inf. Sci. 487 (2019) 142–155, https://doi.org/10.1016/j.ins.2019.03.011, url:http://www.sciencedirect.com/science/article/pii/S0020025519302002.
[30] H. Peng, J. Liu, C. Lin, News citation recommendation with implicit and explicit semantics, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7–12, 2016, Berlin, Germany, Volume 1: Long Papers, The Association for Computer Linguistics, 2016. url:http://aclweb.org/anthology/P/P16/P16-1037.pdf..
[31] S. Xing, F. Liu, Q. Wang, X. Zhao, T. Li, Content-aware point-of-interest recommendation based on convolutional neural network, Appl. Intell. 49 (3) (2019) 858–871, https://doi.org/10.1007/s10489-018-1276-1, url:https://doi.org/10.1007/s10489-018-1276-1.
[32] C. Rudin, Y. Wang, Direct learning to rank and rerank, in: A.J. Storkey, F. Pérez-Cruz (Eds.), International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9–11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain, Vol. 84 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 775–783. url:http://proceedings.mlr.press/v84/rudin18a.html..
[33] A. Saleh, F. Mai, C. Nishioka, A. Scherp, Reranking-based recommender system with deep learning, in: M. Eibl, M. Gaedke (Eds.), 47. Jahrestagung der Gesellschaft für Informatik, Informatik 2017, Chemnitz, Germany, September 25–29, 2017, Vol. P-275 of LNI, GI, 2017, pp. 2169–2175. doi:10.18420/in2017_216. url:https://doi.org/10.18420/in2017_216..
[34] Q. Ai, K. Bi, J. Guo, W.B. Croft, Learning a deep listwise context model for ranking refinement, in: K. Collins-Thompson, Q. Mei, B.D. Davison, Y. Liu, E. Yilmaz (Eds.), The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08–12, 2018, ACM, 2018, pp. 135–144. doi:10.1145/3209978.3209985. url:https://doi.org/10.1145/3209978.3209985..
[35] S. MacAvaney, F.M. Nardini, R. Perego, N. Tonellotto, N. Goharian, O. Frieder, Efficient document re-ranking for transformers by precomputing term representations, in: J. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. Wen, Y. Liu (Eds.), Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25–30, 2020, ACM, 2020, pp. 49–58. doi:10.1145/3397271.3401093. url:https://doi.org/10.1145/3397271.3401093..
[36] L. Sanchez, J. He, J. Manotumruksa, D. Albakour, M. Martinez, A. Lipani, Easing legal news monitoring with learning to rank and BERT, in: J.M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M.J. Silva, F. Martins (Eds.), Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II, Vol. 12036 of Lecture Notes in Computer Science, Springer, 2020, pp. 336–343. doi:10.1007/978-3-030-45442-5_42. url:https://doi.org/10.1007/978-3-030-45442-5_42..
[37] M. Di, D. Klabjan, L. Sha, P. Lucey, Large-scale adversarial sports play retrieval with learning to rank, ACM Trans. Knowl. Discov. Data 12 (6) (2018) 69:1–69:18. doi:10.1145/3230667. url:https://doi.org/10.1145/3230667..
[38] C. Xiao, H. Zhong, Z. Guo, C. Tu, Z. Liu, M. Sun, Y. Feng, X. Han, Z. Hu, H. Wang, J. Xu, CAIL2018: A Large-Scale Legal Dataset for Judgment Prediction, arXiv:1807.02478 [cs]ArXiv: 1807.02478 (Jul. 2018)..
[39] B. Luo, Y. Feng, J. Xu, X. Zhang, D. Zhao, Learning to Predict Charges for Criminal Cases with Legal Basis, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 2727–2736.
[40] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation, Journal of machine Learning research 3 (Jan) (2003) 993–1022.
[41] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119..
[42] J.T. Kent, Information gain and a general measure of correlation, Biometrika 70 (1) (1983) 163–173.
[43] J.M. Dickey, B. Lientz, The weighted likelihood ratio, sharp hypotheses about chances, the order of a markov chain, Ann. Math. Stat. (1970) 214–226.
[44] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022. url:http://jmlr.org/papers/v3/blei03a.html.
[45] T. Joachims, Optimizing search engines using clickthrough data, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23–26, 2002, Edmonton, Alberta, Canada, ACM, 2002, pp. 133–142. doi:10.1145/775047.775067. url:https://doi.org/10.1145/775047.775067..
[46] Y. Freund, R.D. Iyer, R.E. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences, J. Mach. Learn. Res. 4 (2003) 933–969. url:http://jmlr.org/papers/v4/freund03a.html.
[47] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to rank using gradient descent, in: Proceedings of the 22Nd International Conference on Machine Learning, ACM, New York, NY, USA, 2005, pp. 89–96, https://doi.org/10.1145/1102351.1102363.
[48] L.C.-J. Chang C-C, Libsvm: a library for support vector machines, url:http://www.csie.ntu.edu.tw/ cjlin/libsvm..
[49] S. MacAvaney, A. Yates, A. Cohan, N. Goharian, CEDR: contextualized embeddings for document ranking, in: B. Piwowarski, M. Chevalier, É. Gaussier, Y. Maarek, J. Nie, F. Scholer (Eds.), Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21–25, 2019, ACM, 2019, pp. 1101–1104. doi:10.1145/3331184.3331317. url:https://doi.org/10.1145/3331184.3331317..