Overview of Judicial Text Summarization Method

Hongtao Zhao *

Beijing University of Posts and Telecommunications, Beijing, China

* Corresponding Author Email: Zhaohongtao@bupt.ecu.cn

Abstract. This article delves deep into the core aspects of the task of generating judicial text summaries. Through a systematic review and distillation of existing relevant literature, the article primarily focuses on extractive text summarization techniques in both unsupervised and supervised learning contexts, conducting a multidimensional and comprehensive analysis. To begin with, the article traces the evolution of text summarization techniques and dissects the differences between extractive and generative text summarization methods, along with a comparison of various algorithms. Furthermore, it provides a detailed introduction to a pipeline judicial summary generation model that combines both extractive and generative approaches. The article also conducts an in-depth analysis of the impact of transfer learning, using three different models, on judicial text summary generation. Lastly, while acknowledging significant progress in the field, the article points out the main issues and challenges in current judicial text summarization research. It also suggests potential solutions and outlines future development trends.

Keywords: Judicial Text Summarization, Natural Language Processing, Deep Learning.

1. Introduction

In the past few years, the rise of artificial intelligence (AI) has sparked a technological and societal revolution on a global scale. From self-driving cars to smart assistants, from medical diagnoses to financial analysis, AI is permeating nearly every field. However, one area stands out for its significant potential and impact, and that is the field of law.

In the legal domain, AI has demonstrated tremendous potential and practical value. Law is an information-intensive field, and lawyers, judges, and legal professionals deal with vast volumes of documents, cases, and regulations. AI leverages natural language processing (NLP) technology to transform this massive textual data into useful information, providing support for legal work. Legal documents are typically lengthy and filled with legal terminology, posing challenges for both laypersons and legal practitioners. Through NLP and summarization techniques, AI can convert complex legal documents into concise and understandable summaries, making it easier for people to comprehend and access key information.

AI's applications in the legal field can be divided into several aspects. Office automation has become a reality in the legal sector. Lawyers and legal assistants can use AI tools to generate legal documents, contracts, and legal paperwork. This not only improves efficiency but also reduces errors. Lawyers can utilize natural language processing techniques to search and analyze numerous case laws and legal literature to extract crucial information and precedents, facilitating legal research and case preparation. Additionally, AI can be employed for evidence analysis in the courtroom. It can assist lawyers and judges in quickly locating and analyzing evidence, supporting investigations and trials.

China's smart justice development covers various aspects. Deep applications of AI technologies such as speech recognition, image and text recognition, and semantic analysis have become a reality in China's legal domain. These technologies facilitate lawyers and judges in processing legal documents and cases more efficiently, expediting judicial processes. The introduction of intelligent case-handling systems has yielded significant results. For instance, Collaborations between the Jiangsu Provincial People's Procuratorate and KUKA Robotics Corporation have resulted in legal service robots that incorporate facial recognition and voice interaction technologies, enhancing the interactivity and user-friendliness of judicial information service platforms.

The application of artificial intelligence in the legal field holds immense potential and significance. It not only enhances the efficiency of legal work but also helps address the issue of information overload in the legal domain, making law more accessible and understandable to a wider audience.

2. Development History of Text Summarization

2.1. Research Status of Extractive Summarization

In the early stages of automatic summarization development, extractive methods were predominantly used. The extraction process typically involves two main steps: sentence scoring and sentence selection. In sentence scoring, various aspects such as word choice, position, length, and semantics are comprehensively considered to provide suggested scores [1]. Sentence selection takes into account factors like information content, redundancy, and coherence. Extractive summarization is practical and efficient, and it remains an active and widely recognized branch in the field, enjoying widespread acceptance and usage within the academic community.

Traditional extractive summarization methods often employ unsupervised approaches, unlike translation tasks that require parallel corpora for training, eliminating the need for corpus calibration and labor-intensive manual labeling. The Lead-N algorithm selects the first N sentences of an article to construct a summary, known for its simplicity and efficiency [2]. However, its limitation lies in its heavy reliance on the dataset itself and its suitability primarily for news articles that provide the main idea at the beginning. Subsequently, there emerged keyword frequency-based methods guided by statistical probabilities as a guiding principle. These methods often focus on sentences rich in entity-related semantic words for summarization. Edmundson, for instance, eliminates non-entity words with low relevance to the article's main points, such as adverbs and prepositions [3]. Sarkar, building upon this idea, considers subsets of words in the text as keywords, creating text summaries by maximizing the importance of single words or multi-word concepts [4]. The aforementioned rule-based or word frequency-based methods filter similar text based on surface-level semantics to generate summaries. However, they lack in-depth features at the word or sentence level and contain a considerable amount of redundant information, leaving ample room for improvement in the quality of summaries.

To address the issue of extensive redundancy in extractive summarization, some researchers have explored the use of graph-based ranking or topic clustering methods to consolidate clusters of sentences and segments with similar meanings. Among these methods, the Text Rank algorithm has gained significant attention [5]. It effectively leverages similar segments within the text to construct a directed graph, where sentences are treated as nodes, and directed edges with weights represent the similarity between sentences, with direction indicating their adjacent order in the text. Additionally, the algorithm can build a word co-occurrence network using extracted keywords and iteratively calculate sentence similarity until the edge weights stabilize within an acceptable range. This process helps extract important sentences for generating the summary. On the other hand, the Latent Dirichlet Allocation (LDA) algorithm, based on topic distribution, computes the topic distribution for each sentence and selects those sentences that highly summarize the topics for summary generation [6]. However, this method may lead to summaries with relatively singular topics or topics that do not align with the actual content.

In recent years, Researchers like Rush introduced the sequence-to-sequence architecture to automatic summarization tasks, inspiring subsequent research in the field [7]. However, their neural network language model did not effectively address the issue of long-range dependencies caused by temporal state transitions. Additionally, treating the entire article as a single sequence input to the encoder presented challenges in capturing distinctive features that represent differences at the sentence and document levels, resulting in suboptimal performance. To tackle these challenges, Cheng employed Long Short-Term Memory (LSTM) as the encoder and combined it with a single-layer Convolutional Neural Network (CNN) for encoding sentence representations [8]. This approach resolved the long-range dependency issue but did not directly address the differentiation of features

between sentences and documents. Subsequently, Nallapati adopted a tagging approach to extract summary sentences and used hierarchical encoding, splitting the encoding into two synchronized directions [9]. They first employed a sentence encoder to encode words into sentences and then used a document encoder to encode sentences into documents. This hierarchical encoding method effectively captured distinct features at the sentence and document levels. Furthermore, their document encoder utilized Bidirectional Gated Recurrent Unit (BiGRU) as the encoding unit, which has fewer gating units and computational parameters compared to Bidirectional Long Short-Term Memory (BiLSTM), resulting in improved performance.

With the rise of legal artificial intelligence, there has been a notable increase in interest in judicial summarization, which is a subfield of automatic summarization. Developed countries have made steady progress in judicial summarization research. Chieze, for instance, focused on summarizing Canadian legal documents [10]. They defined these documents as having four sections: introduction, context, legal analysis, and conclusion. Their algorithm systematically identified relevant sentences for each section and concatenated them to form the summary. Similarly, Bhattacharya conducted research on summarizing Australian legal documents [11]. They considered domain-specific features such as acronyms and legal entities to determine sentence scores. Sentences with scores above a certain threshold were selected to compose the summary.

2.2. Research Status of Abstractive Summarization

In the early days of generative summarization, there was a strong focus on entity semantics, with the goal of obtaining summaries by compressing sentences while preserving their core structure. This approach was relatively straightforward and aimed to maintain the overall framework of the original text [12]. Compared to simple sentence extraction, it provided some control over redundancy. However, the resulting compressed sentences often removed many descriptive elements, leading to less richness and completeness in the generated summaries. These summaries still fell significantly short of human-authored summaries. To address this challenge, Nenkova and others built upon compression-based methods by introducing fusion techniques for multi-document summarization [13]. They sought to combine content from similar texts while preserving certain syntactic structures, attempting to strike a balance between redundancy and richness. Lebanoff explored potential factors for selecting and fusing sentences from multiple sources, aiming to bridge the gap by ranking single sentences and pairs of sentences in a new space [14]. However, these approaches primarily relied on shallow syntactic structure analysis, lacking a deep understanding of the articles' underlying semantic content, leaving ample room for improvement. These early generative methods provided a foundation but struggled to capture the essence of the articles due to their limited understanding of deep semantic meaning. As a result, there was significant room for improvement in the field of generative summarization.

In 2014, Generative summarization models, primarily based on sequence-to-sequence neural networks, focus on transforming the original text sequence into an intermediate state vector during the encoding phase and then generate candidate summaries during the decoding phase. Nallapati employed Recurrent Neural Networks (RNNs) as encoding and decoding units, complemented by embeddings that go beyond individual words, such as part-of-speech tags and named entity recognition [15]. This approach allowed them to model abstract text summaries and achieved some success. However, these models were limited in their effectiveness when applied to longer original texts. Subsequent improvements were made by introducing attention mechanisms on top of the Seq2Seq framework [16]. By combining the decoder with attention vectors that calculate attention scores for the entire text, models could focus on the most relevant textual information, thereby enhancing prediction capabilities [17]. Duan extended attention mechanisms by proposing a contrasting attention mechanism, which considered both regular attention to relevant parts of the source sentences and contrasting attention to irrelevant parts [18]. Experiments on benchmark datasets confirmed the effectiveness of this approach. Furthermore, Song incorporated Convolutional Neural Networks (CNNs) as sentence encoders, exploring the reconstruction of new sentences from

finer-grained units, known as semantic phrases, rather than entire sentences [19]. Leveraging text convolution, they achieved promising results on short-text news datasets such as Daily Mail.

To further enhance the quality of summaries, some researchers have turned their attention to strategies involving pre-trained word embeddings and fine-tuning of neural network language models. Devlin introduced the language representation model DBERT (Dual Bidirectional Encoder Representations from Transformers), which combines the advantages of bidirectional encoding in Transformers to achieve multi-level contextual adjustments, obtaining comprehensive semantic representations through fine-tuning [20]. Cai and others extended the Transformer with an additional RNN in their RC-Transformer model, allowing for the filtration of locally important contextual sequences and improving the semantic representation of long sequences in summary models [21]. Pre-training models and fine-tuning them for downstream generative tasks have shown good performance in the field of short text summarization. However, due to the limitations of pre-trained models like BERT (Bidirectional Encoder Representations from Transformers) and Transformer in handling complex semantics in long texts, they may struggle to adapt to the domain of lengthy documents. Some scholars have explored embedding semantic representations of deep text using a thematic approach. Gao used extracted keywords to learn latent discriminative Gaussian themes in the embedded vector space, closely integrating themes with sentence embeddings to enhance semantic relationships between adjacent sentences, demonstrating the effectiveness of this method in query-based summarization [22]. However, thematic representation based on keywords can be somewhat discrete and lack continuity. Therefore, Liu and others considered the progressive variation of themes in dialogue summarization, simulating changing themes in a thematic-aware manner to consolidate scattered dialogue information using a global thematic approach, expanding the scope of summarization in the dialogue domain [23]. To obtain more diverse themes, Cui and colleagues utilized a neural topic model to discover latent themes that extend across documents, achieving significant improvements in multi-document summarization guided by cross-document thematic distribution [24]. By introducing thematic elements, generative summarization has gained further development opportunities.

Whether it's extractive or generative summarization, advancements in summarization models depend on gaining a deeper understanding of the text. This is particularly relevant in the realm of legal text summarization research. In our study, which focuses on summarizing legal documents, we've noticed that existing topic models, whether they're based on keywords or neural networks, tend to aim for a broad coverage of global topics. However, legal texts, being highly specialized, often revolve around more concentrated and closely related themes, even featuring similar subtopics. Remarkably, there hasn't been much research dedicated to addressing the relevance of these themes in legal document summarization. Hence, to further enhance legal summarization, a key aspect of our generative summarization research involves exploring better techniques for modeling these themes at a more granular level. Our goal is to collectively improve the quality of the generated summaries by leveraging the concentrated thematic information found within the specialized legal domain.

3. General Process of Text Summarization

As a subfield of automatic summarization, legal summarization involves summarizing the content of legal documents, extracting, and generating summaries based on the original legal texts. It holds significant practical value for effectively dealing with the overload of legal information. In this chapter, we introduce the general process of extractive summarization methods, particularly suitable for summarizing legal documents characterized by complex sentence structures, rigorous reasoning, and well-organized structures. Some research divides the summarization process into two stages: extraction and generation. In recent years, some scholars have focused on separating content selection and summary generation. For example, Lebanoff proposed a method for selecting appropriate single sentences and sentence pairs to generate single-sentence summaries [25]. Subramanian, for long document summarization, added a simple extraction step before generating summaries [26].

Intuitively, summarizing important paragraphs or sentences aligns more closely with the human thought process of summarization. Content selection, relative to summary generation, emphasizes information identification and domain-specific characteristics. For instance, sports news focuses on scores and information about participating athletes, while legal texts emphasize case details and legal provisions. Summary generation, on the other hand, focuses on semantic trimming and abstraction. Separating these two processes allows for better testing of sentence compression and fusion models' performance and provides a degree of interpretability. However, it may also result in some information loss during the extraction process. In this section, we further process the corpus using an algorithm that maximizes the sum of the similarity between important sentences and the summary.

3.1. Extractive Summarization

The architecture of the extractive summarization model consists of sentence embedding and document-level encoders. The BERT model is used to create sentence-level encoding vectors, and the document-level encoder employs a multi-layer Transformer structure [38]. Finally, a fully connected layer is applied to the sentence-level encodings to predict the label category for each sentence. BERT, through self-supervised learning on a large corpus, acquires excellent word representations, often using the output vector from the "[CLS]" position as the text representation for downstream specific tasks in text classification. The way BERT is used for extractive summarization is by inserting "[CLS]" and "[SEP]" before and after each sentence, and then concatenating all the sentences as input [27].

However, legal datasets often contain documents with significantly longer text than what a single BERT model can handle in one pass. Additionally, considerations such as word tokenization, adding special characters, and the memory capacity of typical GPUs can pose challenges. To address this, a strategy is used where each sentence is processed one at a time, and the output vector at the "[CLS]" position, denoted as $Tj[CLS]$ is cached. The L sentence vectors for a document form the sequence $T[CLS] = \{T\,1[CLS], T2[CLS], \ldots\, TL[CLS]\}$. This approach significantly reduces the memory usage at any given time but has the drawback that the sentence encoder cannot be fine-tuned.

In sentence-level extractive summarization algorithms, it is common to represent sentences using a document-level encoding approach. After encoding with the BERT model, the obtained vectors are sentence-level encoding vectors, which only contain information about the sentences themselves and lack sentence context information. Therefore, it is a common practice to input the sentence-level encoding vectors into a document-level encoder. In the document-level encoder, multiple layers of Transformer structures are used, and multi-head attention is calculated at each layer. This allows for interactions between sentences, resulting in the final output, which is the document-level encoding vector for each sentence. Finally, a fully connected network classifier is used for classification. In the "collection-full text" model, binary classification is used, while in the "collection-sentence" segmented summarization model, it outputs three entity labels: "B", "I", and "O". The label sequence is used to obtain a subset of sentence indices in the extractive summary. For example, based on the label sequence "$\{B, O, I, B, O, B\}$", the extractive summary set is obtained as "$\{\{0, 2\}, \{3\}, \{5\}\}$". The probability of each sentence label is calculated using Formula (1), where Trm represents the computation process of the multi-layer Transformer, and W_0 is an updatable parameter matrix.

$$Y = soft\,max(W_0 Trm(T[CLS])) \tag{1}$$

3.2. Generative Summarization Models

On the encoding side, the word embedding representation vectors of a sequence $S = \{s1, s2, s3\ldots, sL\}$ with a length of L are input into a Bidirectional Long Short-Term Memory network (BiLSTM). As shown in equation (2), the hidden layer vectors at the i-th position are concatenated to serve as the output of the encoder $H = \{h1enc, h2enc, h3enc\ \ldots, hLenc\}$:

$$h_{enc}^{i} = [\overrightarrow{h_i^T}, \overleftarrow{h_i^T}]^T \tag{2}$$

In the decoding phase, as expressed in equation (3), the decoder, primarily composed of LSTM (with the function g denoting the decoding process), receives inputs at time t. These inputs include the previous time step's hidden layer outputs, $h_{t-1}$and c_{t-1}, the word embedding of the previous output result, y_{t-1}, as well as the attention vector of the encoder's hidden layer at the previous time step, $h_{t-1}{}^{*}$. This decoder computes the current time step's hidden layer outputs, ht and ct. Equations (4), (5), and (6) respectively represent the attention score u_i^t, weight a_i^t for the i-th word in the input sequence, and the computation of the encoder's hidden layer attention $h_t{}^{*}$. at the current time step. In these equations, W_1 and W_2 are parameter-updatable matrices, while V_0 is a parameter-updatable vector.

$$h_t, C_t = g(h_{t-1}^{e*}, y_{t-1}, h_{t-1}, c_{t-1}) \tag{3}$$

$$u_i^t = V_0 tanh(W_1 h_i^{enc} + W_2[h_t^T, C_t^T]^T) \tag{4}$$

$$a_i^t = softmax(u_i^t) \tag{5}$$

$$h_t^{e*} = \sum_L^{i=1} a_i^t * h_i^{enc} \tag{6}$$

Although the Intra-temporal Attention mechanism can control the decoder's tendency to focus on different parts of the input sequence at different time steps, it is still possible to generate duplicate sequence units if the decoder's hidden layer output at the previous time step is like the current one. Therefore, an Intra-decoder Attention mechanism has been employed, which calculates attention for all hidden layer output vectors of the decoder prior to time step t, to generate the output vector at time step t [28]. The Intra-decoder Attention mechanism allows the decoder to output not only the current hidden layer vector h_t^d but also calculates the attention h_t^{d*} for all previous hidden layer output vectors up to time t. The calculation process is described by equations (12), (13), and (14), where W' and We are matrices with updatable parameters, and V' is a row vector with updatable parameters:

$$e_{tt'}^d = (V'(W'h_{t'}^d + Wh_t^d)) \tag{7}$$

$$a_{tt'}^d = \frac{exp(e_{tt'}^d)}{\sum_{j=1}^{t-1} exp(e_{tj}^d)} \tag{8}$$

$$h_t^{d*} = \sum_{j=1}^{t-1} a_{tj}^d h_j^d \tag{9}$$

4. Application of Transfer Learning in Judicial Text Summarization

When generating summaries for legal texts, there are many unique characteristics to consider. Legal texts are typically filled with legal terminology, regulations, and legal logic, making them quite complex and specialized. This complexity can pose challenges for non-legal professionals who may find it difficult to understand the content. Generating summaries for legal texts requires a deep understanding of legal concepts and principles to accurately explain and summarize the content of the legal text. It also involves the transformation of formal legal language into more accessible and comprehensible language so that a broader audience can understand the summary's content. Legal texts are often lengthy and contain a substantial amount of detail and information, including legal terms and the distinctive writing style commonly found in legal documents. To create meaningful summaries, summary generation systems must be capable of extracting the most important and central information from these lengthy texts.

Additionally, legal texts include legal clauses and legal reasoning, which form the foundation of the documents. Summary generation systems need to comprehend these legal concepts and ensure that the summaries they generate maintain logical consistency, allowing readers to understand the legal requirements and conclusions presented in the legal documents. In response to the characteristics of legal texts mentioned above, the subsequent sections introduce three deep learning models and compare them in terms of entity recognition performance and model transferability. The analysis assesses the impact of different text representation methods and feature generation approaches on the recognition of various types of entities in the legal domain. The study summarizes models suitable for identifying both general and specific entities in criminal judgment text and concludes by discussing the transferability of models generated for judicial decision texts to other types of legal texts.

4.1. The CRF++ tool

The CRF++ tool is utilized to perform named entity recognition based on Conditional Random Fields (CRF) [29]. The process involves the following steps (1) Constructing Training and Testing Data in CRF++ Format. (2) Manual Feature Construction: Part-of-speech (POS) features and boundary features are manually crafted. POS features are automatically labeled using the Jieba segmentation tool [30]. Feature templates are created based on the constructed features, and a character window of length 5 is set. High-order features are not used. (3) Model Optimization for Entity Recognition in Criminal Judgments.

The data is annotated using both the BIO (Begin, Inside, Outside) and BIEOS (Begin, Inside, End, Outside, Single) tagging systems, and experiments are conducted separately by inputting them into the CRF++ tool. The training parameters for CRF++ use conventional values with f set to 3 and c set to 44. A comparison is made between the results obtained with different tagging systems. Although the BIEOS system extends the BIO tagging system, it does not lead to an improvement in entity recognition performance. The F1 score using the BIO system is significantly higher than that of the BIEOS system for most entity categories. When comparing the training time and considering both recognition performance and time expenditure, it becomes evident that the BIO tagging system outperforms the BIEOS system. Therefore, all subsequent experiments in this paper will utilize the BIO tagging system.

4.2. The IDCNN-CRFs model

Iterated Dilated CNN with CRFs (IDCNN-CRFs model), convert the input composed of text and labels into vector form, using the labeling system that yielded better results in CRFs model experiments [31]. Word Embedding: Utilize a pre-trained model based on a large-scale Chinese corpus from Wikipedia with a word vector dimension of 100. Prior to entering the dilated convolutional layer, incorporate a Dropout layer. This layer determines, based on a preset probability, whether the weights of a hidden layer neuron in the network should be transmitted to the next layer [32]. This step, to some extent, helps prevent the introduction of features that only function under specific circumstances, enhancing the model's generalization capability [33]. Utilize dilated convolutional layers to extract features from the input data. Employ four identical Dilated CNN modules combined. Each module consists of three layers of dilated convolutional layers with dilation widths of 1, 1, and 2. Convert the output vectors to match the dimension of the label vectors. The dimension of the output vectors from the dilated convolutional layers does not match the dimension of the label vectors used when calculating the loss function. Therefore, a conversion is required. Compute the loss using the CRFs layer. The CRFs layer employs the Viterbi algorithm to decode the output data from the neural network into predicted labels for each character. The IDCNN-CRFs model has default parameters, including Batch Size 32, Epoch 100, dropout keep 0.5, Word Embedding Dimension 100, Number of Filters 100, Gradient Clip 5, Learning Rate 0.001, and Optimizer Adam. Subsequent experiments involve adjusting the Batch Size parameter to 8 and the Epoch to 10, while keeping other parameters at their default values.

4.2.1. The batch size and epoch combination experiments in the IDCNN-CRFs model

In the IDCNN-CRFs model, the same BIO tagging system that yielded better results in the CRFs model experiments is employed. Due to the relatively small dataset, the tuning process begins by setting the batch size to 16. Subsequently, the batch size is adjusted by either doubling it or increasing it by 50%, and the experimental results are compared. If doubling the batch size yields better results, it is further doubled. Similarly, if increasing it by 50% yields better results, it is increased accordingly. To save time, the epoch is set to 1 during this process. Upon comprehensive analysis, it is observed that when the batch size is set to 8, the F1 scores for various entity classes are more balanced compared to other values. Considering that with a batch size of 8, the F1 scores for entities in the "Crime" and "Time" classes are already high, increasing the number of training epochs may not necessarily lead to improved performance. However, to explore whether results are stable when epoch = 10, an additional experiment is conducted with epoch = 20 and batch size = 8. It is found that with a batch size of 8, epoch = 10 already achieves the overall best results. Therefore, it is evident that when utilizing the IDCNN neural network for entity recognition, it is not necessary to blindly increase the number of training epochs. Instead, a balance should be struck between batch size and epoch to achieve satisfactory experimental results while conserving training time and reducing memory consumption.

4.2.2. Dropout Rate Experiment Results

When Batch Size = 8 and Epoch = 10, adjustments were made to the dropout layer's neuron dropout probability. The default value is 0.5. For each experiment, the dropout rate was increased or decreased by 0.1, and the experimental results were compared. Based on the direction that yielded better results, the dropout rate was either increased or decreased until the results stabilized. Overall, it was found that maintaining the default dropout rate of 0.5 is more appropriate. In addition to the adjustments made to the Batch Size, Epoch, and Dropout parameters mentioned above, tuning other parameters in the IDCNN-CRFs model had a relatively small impact on the experimental results. Therefore, all other parameters were kept at their default values.

4.3. The BiLSTM-CRFs model

In constructing the BiLSTM-CRFs model for entity recognition in judgment text [34], experiments were conducted in the word embedding layer using three different approaches: random numbers, BERT pre-trained word vectors [35], and ALBERT (A Lite BERT) pre-trained word vectors [36]. For models using random numbers and BERT pre-trained word vectors for word embedding, the model's architecture for word embedding was first established. The training dataset was divided into several batches based on the longest sentence to determine the vector length, with padding applied to other sentences to match this length. The BiLSTM layer was utilized to extract data features, followed by two fully connected layers for nonlinear transformations to capture relationships between features. Finally, the output was mapped to the output space, and the predicted label sequence was obtained using the CRFs layer. This paper did not employ an optimizer or early stopping mechanism in the model, so there was no need to set parameters such as learning rate and optimizer type.

In the BiLSTM-CRFs model, different word embedding vectors were experimented with, including random number vectors, BERT word vectors pretrained on a large-scale Chinese corpus released by Google, and ALBERT word vectors improved and released on the BERT model. The experiments were conducted using the Kashgari tool, which is a modular natural language processing toolkit [37]. Micro-average evaluation metrics were added to the BiLSTM-CRFs model to assess the overall performance. Micro-average refers to the arithmetic average of the test results of all samples [38].

4.3.1. Random Number Embedding Experiment

After converting tokens into vectors composed of random numbers for word embedding, the process involved setting the Batch Size initially to 16. Then, the Batch Size was successively doubled and increased by 50%, following the direction that yielded better results. Eventually, the Batch Size

was set to 4. Once the Batch Size was determined to be 4, the next step was to vary the default Embedding Size of 100 in the range [50, 100, 200, 300]. The highest F1 micro-average, reaching 88.38%, was achieved when the Embedding Size was set to 200. Therefore, the Embedding Size was set to 200. The paper focused on finding the optimal values for four hyperparameters of the BiLSTM-CRFs model. Firstly, there were two hyperparameters related to the BiLSTM layer: BiLSTM Layer Units and return_sequences. The default value for BiLSTM Layer Units was 128. Experimentation involved varying it in the range [64, 128, 256], and it was found that keeping it at the default value yielded the highest F1 micro-average. The return_sequences parameter, which defaults to True, was changed to False, resulting in a decrease in the F1 micro-average from 88.38% to 84.91%. There were two hyperparameters associated with the two fully connected layers following the BiLSTM layer: Fully Connected Layer Units and the activation function. The default value for Fully Connected Layer Units was 64. Experimentation involved varying it in the range [32, 64, 128], and it was found that keeping it at the default value yielded the highest F1 micro-average. Regarding the activation function, the default tanh function was compared with sigmoid and ReLU. Sigmoid and tanh functions have high computational complexity as they require exponential calculations, while ReLU only requires a threshold to obtain the activation value. Additionally, the tanh function can suffer from the vanishing gradient problem, which is not an issue with ReLU. As a result, the paper chose to compare ReLU and tanh, and it was found that the F1 micro-average was higher when using the default tanh activation function. Further experiments were conducted by increasing the number of training epochs, with experiments conducted for both Epoch = 3 and Epoch = 5. From the experimental results, it can be observed that after 3 training iterations, the model did not overfit. When compared to the experiment with Epoch = 5, after 5 training iterations, the model exhibited a higher degree of overfitting to the training dataset but a lower degree of fit to the validation dataset. This indicates that after 5 training iterations, the model overfitted the training dataset excessively, resulting in suboptimal performance on the validation dataset. Therefore, in this paper, for the BiLSTM-CRFs experiment using random number embeddings, the value of Epoch was set to 3.

4.3.2. BERT and ALBERT Embedding Experiment

When converting input data into the required formats for BERT and ALBERT, additional special tokens were added, including the [sep] token to represent sentence separation, the [unk] token for characters not found in the model's vocabulary, the [cls] token to represent the beginning of a sentence, and the [pad] token for padding. For model tuning, in the BERT embedding model, Batch Size was set to 8, and four hyperparameters were kept at their default values with Epoch set to 3. In the ALBERT embedding model, Batch Size was set to 4, Embedding Size was set to 100, Dropout rate was set to 0.5, and, like the IDCNN-CRFs model, Epoch was set to 10, while other parameters were left at their default values. The F1 values for different word embedding experiments with the BiLSTM-CRFs model were obtained. From the results, models using pre-trained word vectors for word embedding significantly outperformed models using random number embeddings. While random number embeddings had shorter training times, they exhibited a performance gap in entity recognition compared to other models. The F1 micro-average of the BERT-BiLSTM-CRFs model was like that of the ALBERT-BiLSTM-CRFs model, but the training time for the latter was longer. ALBERT is a simplified pre-trained model based on BERT with improved computational efficiency and maintained its ability to capture various types of information in text, as observed from the results.

5. Conclusion

Experiments were conducted with the IDCNN-CRFs and BiLSTM-CRFs models that automatically generate features, as well as the CRFs model with manually constructed features. In the BiLSTM-CRFs model, three types of word embeddings were experimented with: random numbers, BERT, and ALBERT. The goal was to identify the best-performing models within these three categories, and their transferability was assessed using new data. On the original dataset, the ALBERT-BiLSTM-CRFs model exhibited the best overall entity recognition capability, followed by

the IDCNN-CRFs model, with both outperforming the CRFs model. In terms of training time, the IDCNN-CRFs model required the shortest training time.

When it comes to recognizing new data, the IDCNN-CRFs model and the ALBERT-BiLSTM-CRFs model each had their strengths, with varying performance for different entity recognition tasks. Overall, the differences between them were not significant. The CRFs model showed slightly lower recognition capabilities for new data and required the application of model features before making predictions. In summary, for entity recognition in criminal judgment texts, deep learning models that automatically generate features outperformed traditional statistical learning models with manually constructed features. Additionally, they demonstrated stronger transferability to new data.

Transfer learning has a key application area in the automated generation of legal text summaries. In the legal domain, legal documents and decisions are often presented in extensive textual formats, such as judgments, regulations, and legal cases. Lawyers, judges, and legal researchers invest a significant amount of time in reading and comprehending these texts to extract essential information. Transfer learning models can expedite this process by learning from a vast corpus of existing legal texts and transferring this knowledge to generate concise and informative summaries for new texts. This can greatly enhance the efficiency of legal professionals, allowing them to access the required information more rapidly. Furthermore, transfer learning can be employed for generating summaries of legal texts in multiple languages. Legal systems in different countries and regions utilize various languages, necessitating the handling of multilingual legal texts in cross-border legal affairs. Transfer learning aids in transferring models trained in one language to work effectively in another language for legal text summarization. This facilitates international collaboration and legal communication, enabling professionals in cross-border legal fields to better understand legal texts from different countries and regions.

These application areas illustrate the potential value of transfer learning in judicial text summarization. It holds the promise of delivering more efficient and intelligent solutions to the legal domain, fostering progress and development in the field of law.

References

[1] Yadav A.K, Maurya A.K, Ranvijay, et al. Extractive Text Summarization Using Recent Approaches: A Survey [J]. Ingénierie des Systèmes d Inf, 2021, 26 (1): 109 - 121.

[2] Dalal V, Malik L.G. A Survey of Extractive and Abstractive Text Summarization Techniques [J]. 6th International Conference on Emerging Trends in Engineering and Technology. IEEE, 2013.

[3] Edmundson H.P. New Methods in Automatic Extracting [J]. Journal of the ACM, 1969, 16 (2): 264 - 285.

[4] Sarkar K. Automatic Single Document Text Summarization Using Key Concepts in Documents[J]. J Inf Process Syst, 2013, 9 (4): 602 - 620.

[5] Mihalcea R, Tarau P. TextRank: Bringing Order into Text [C]. Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP). Barcelona, Spain: ACL, 2004: 404 - 411.

[6] Gao D, Li W, You O, et al. LDA-Based Topic Formation and Topic-Sentence Reinforcement for Graph-Based Multi-document Summarization [C]. 8th Asia Information Retrieval Societies Conference (AIRS). Tianjin, China: Springer, 2012.

[7] Rush A.M, Chopra S, Weston J. A Neural Attention Model for Abstractive Sentence Summarization[C]. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP). Lisbon, Portugal: ACL, 2015: 379 - 389.

[8] Cheng J, Lapata M. Neural Summarization by Extracting Sentences and Words [C]. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL). Berlin, Germany: ACL, 2016: 484 - 494.

[9] Nallapati R, Zhai F, Zhou B. SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents [C]. Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. San Francisco, USA: AAAI, 2017.

[10] Chieze E, Farzindar A, Lapalme G. An Automatic System for Summarization and Information Extraction of Legal Information [J]. Lecture Notes in Computer Science, Springer, 2010.

[11] Bhattacharya P, Poddar S, Rudra K, et al. Incorporating domain knowledge for extractive summarization of legal case documents [C]. 18th International Conference for Artificial Intelligence and Law (ICAIL). Paulo, Brazil: ACM, 2021: 22 - 31.

[12] Jing H. Sentence Reduction for Automatic Text Summarization [C]. Proceedings of the 6 th Applied Natural Language Processing Conference (ANLP). Washington, USA: ACL, 2000: 310 - 315.

[13] Nenkova A, Siddharthan A, McKeown K.R. Automatically Learning Cognitive Status for Multi-Document Summarization of Newswire [C]. Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing. Vancouver, Canada: ACL, 2005: 241 - 248.

[14] Lebanoff L, Song K, Dernoncourt F, et al. Scoring Sentence Singletons and Pairs for Abstractive Summarization [C]. Proceedings of the 57 th Conference of the Association for Computational Linguistics (ACL). Florence, Italy: ACL, 2019: 2175 - 2189.

[15] Nallapati R, Zhou B, Santos C.N, et al. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond [C]. Proceedings of the 20 th SIGNLL Conference on Computational Natural Language Learning. Berlin, Germany: ACL, 2016: 280 - 290.

[16] Vaswani A, Shazeer N, Parmar N, et al. Attention is All you Need [C]. Proceeding of the 31 st International Conference on Neural Information Processing Systems. Long Beach, USA: Curran Associates Inc, 2017.

[17] Wang Y, Liu X, Gao Z. Neural Related Work Summarization with a Joint Context-driven Attention Mechanism [C]. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Brussels, Belgiun: ACL, 2018.

[18] Duan X, Yu H, Yin M, et al. Contrastive Attention Mechanism for Abstractive Sentence Summarization [C]. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9 th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: ACL, 2019.

[19] Song S, Huang H, Ruan T. Abstractive text summarization using LSTM-CNN based deep learning [J]. Multimed Tools Appl. 2019, 78 (1): 857 - 875.

[20] Devlin J, Chang M.W, Lee K, et al. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding [C]. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). Minnesota, USA: ACL, 2019.

[21] Cai T, Shen M, Peng H, et al. Improving Transformer with Sequential Context Representations for Abstractive Text Summarization[C]. Natural Language Processing and Chinese Computing 8 th CCF International Conference (NLPCC). Dunhuang, China: Springer, 2019.

[22] Gao Y, Xu Y, Huang H, et al. Jointly Learning Topics in Sentence Embedding for Document Summarization [J]. IEEE Trans. Knowl. Data Eng, 2020, 32 (4): 688 - 699.

[23] Liu J, Zou Y, Zhang H, et al. Topic-Aware Contrastive Learning for Abstractive Dialogue Summarization [J]. Findings of the Association for Computational Linguistics, ACL, 2021: 1229 - 1243.

[24] Cui P, Hu L. Topic-Guided Abstractive Multi-Document Summarization [J]. Findings of the Association for Computational Linguistics, ACL, 2021.

[25] LEBANOFF L, SONG K, DERNONCOURT F et al. Scoring Sentence Singletons and Pairs for Abstractive Summarization [C]//Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, 2019: 2175 – 2189.

[26] PILAULT J, LI R, SUBRAMANIAN S et al. On Extractive and Abstractive Neural Document Summarization with Transformer Language Models [C]//Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Online: Association for Computational Linguistics, 2020: 9308 – 9319.

[27] LIU Y, LAPATA M. Text Summarization with Pretrained Encoders [C]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019: 3721 - 3731.

[28] SANKARAN B, MI H, AL-ONAIZAN Y et al. Temporal Attention Model for Neural Machine Translation [J]. 2016.

[29] CRF++: Yet Another CRF toolkit [EB/OL]. [2021-01-20]. https:// taku910.github.io/crfpp/.

[30] Jieba tokenization tool [EB/OL]. [2021-01-20]. https://github.com/fxsjy/ jieba. (Chinese Text Segmentation"Jieba"[EB/OL]. [2021-01- 20]. https://github.com/fxsjy/jieba.)

[31] Strubell E, Verga P, Belanger D, et al. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions [OL]. arXiv Preprint, arXiv: 1702. 02098.

[32] Hinton G E, Srivastava N, Krizhevsky A, et al. Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors [OL]. arXiv Preprint, arXiv: 1207. 0580.

[33] Krizhevsky A, Sutskever I, Hinton G E. Imagenet Classification with Deep Convolutional Neural Networks[J]. Communications of the ACM, 2017, 60 (6): 84 - 90.

[34] Huang Z H, Xu W, Yu K. Bidirectional LSTM-CRF Models for Sequence Tagging [OL]. arXiv Preprint, arXiv: 1508.01991.

[35] Devlin J, Chang M-W, Lee K, et al. Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding [OL]. arXiv Preprint, arXiv: 1810. 04805.

[36] Lan Z Z, Chen M D, Goodman S, et al. ALBERT: A Lite Bert for Self-Supervised Learning of Language Representations [OL]. arXiv Preprint, arXiv: 1909. 11942.

[37] Eziz E. Kashgari [EB/OL]. [2021-01-20]. https://github. com/ BrikerMan/Kashgari.

[38] Maoran, Wang Yilei, Gao Song, et al. A Deep-Learning Model Based on Attention Mechanism for Chinese Comparative Relation Detection [J]. Journal of the China Society for Scientific and Technical Information, 2019, 38 (6): 612 - 621.)