applied sciences

MDPI

Article

Multi-Label Classification of Chinese Rural Poverty Governance Texts Based on XLNet and Bi-LSTM Fused Hierarchical Attention Mechanism

Xin Wang and Leifeng Guo *

Agricultural Information Institute of Chinese Academy of Agricultural Sciences, Beijing 100081, China; 82101181269@caas.cn

* Correspondence: guoleifeng@caas.cn

Abstract: Hierarchical multi-label text classification (HMTC) is a highly relevant and widely discussed topic in the era of big data, particularly for efficiently classifying extensive amounts of text data. This study proposes the HTMC-PGT framework for poverty governance's single-path hierarchical multi-label classification problem. The framework simplifies the HMTC problem into training and combination problems of multi-class classifiers in the classifier tree. Each independent classifier in this framework uses an XLNet pretrained model to extract char-level semantic embeddings of text and employs a hierarchical attention mechanism integrated with Bi-LSTM (BiLSTM + HA) to extract semantic embeddings at the document level for classification purposes. Simultaneously, this study proposes that the structure uses transfer learning (TL) between classifiers in the classifier tree. The experimental results show that the proposed XLNet + BiLSTM + HA + FC + TL model achieves micro-P, micro-R, and micro-F1 values of 96.1%, which is 7.5~38.1% higher than those of other baseline models. The HTMC-PGT framework based on XLNet, BiLSTM + HA, and transfer learning (TL) between classifier tree nodes proposed in this study solves the hierarchical multi-label classification problem of poverty governance text (PGT). It provides a new idea for solving the traditional HMTC problem.

Keywords: HMTC; XLNet; hierarchical attention mechanism; Bi-LSTM; transfer learning; rural poverty governance; NLP; BERT; text classification

Citation: Wang, X.; Guo, L. Multi-Label Classification of Chinese Rural Poverty Governance Texts Based on XLNet and Bi-LSTM Fused Hierarchical Attention Mechanism. *Appl. Sci.* **2023**, *13*, 7377. https://doi.org/10.3390/app13137377

Academic Editor: Valentino Santucci

Received: 30 May 2023
Revised: 16 June 2023
Accepted: 19 June 2023
Published: 21 June 2023

1. Introduction

Machine intelligence is increasingly important in global poverty governance [1,2]. By using algorithms and models, social scientists and policymakers can embrace more substantial insights into the essence of poverty, using complex phenomena and vast amounts of data to prevent vulnerable populations from falling into the cycle of poverty [3]. From the viewpoint of the digital governance of poverty [4], unstructured text data have become a significant part of poverty governance data. These documents contain policies, news announcements, working papers, etc. [5]. At the same time, each class includes a tree of subclasses, adding much information related to certain aspects of the poverty governance lifecycle. Usually, researchers give these texts multiple labels in line with the above hierarchical classes via time-consuming manual efforts before analyzing this information and extracting knowledge about poverty governance [6,7]. Therefore, a classification model based on natural language processing is the primary method for automatic multi-label classification [8].

According to the existing literature, a multi-label classification model can be divided into two parts based on the number of classifiers in the model. First, a single-classifier model transforms the multi-label classification into a multi-class classification problem [9], which is concise and considers labels' correlation as a whole, but usually has an unsatisfactory effect because of the imbalanced training dataset [10]. As the number of labels increases, the parameter size and complexity of the model also significantly increase. In contrast,

combining multiple classifiers [11], namely one or a subset of labels corresponding to a classifier, could reduce the complexity of a single model and avoid being influenced by the dataset's quality, yielding a better holistic classification result. Nevertheless, the number of classifiers will increase while the labels are abundant. In the classifier set, the correlation of each classifier can be independent or dependent on the relationship between related labels. The former transforms the multi-label problem into several single-label issues, ignoring label correlation. The latter regards a combination of classifiers as a chain of classifiers. The advantage of classifier chains is that they construct topological relationships between independent classifiers based on the semantic topological relationships of labels and form a chain structure. However, excessively long classifier chains can lead to a systematic accumulation of classification errors, affecting the end labels' classification accuracy [12]. It is necessary to ensure that each classifier in the classifier chain can achieve the best accuracy to remedy this problem.

From a process perspective, each text classifier includes four modules: data preprocessing, feature extraction, classification algorithms, and result evaluation [13]. To improve the performance of the classifier chain, researchers typically perform unified preprocessing on all classifier inputs and a unified assessment of the classification results [11]. Feature extraction and classification are two kernel processes that determine the performance of a classifier.

The methods for obtaining text features are usually divided into metadata-based and content-based methods [7]. The metadata-based approaches utilize explicit information such as titles, authors, keywords, categories, and other existing labels in the text as text features. This method is standard for the scientific literature with complete metadata information [11], but for text in poverty governance, metadata information is often not a mandatory configuration. Therefore, content-based feature extraction methods are more valuable in PGT text classification. In the existing literature, content-based text feature extraction mainly includes traditional statistical methods such as word bags and TF-IDF, neural network methods such as Word2Vec, and pretrained models such as BERT, ERNIE, and XLNet [14]. Since Google proposed BERT in 2018, with the continuous introduction of new models and methods, the performance of pretrained models has significantly improved owing to their unsupervised training on massive corpora [15]. As a result, pretrained models have gradually become mainstream compared with other feature extraction methods. The basic structure of the BERT model consists of multiple layers of Transformers, including the "masked language model" (MLM) and "next sentence prediction" (NSP) as pretrained tasks. The most prominent feature of the ERNIE model is the introduction of a knowledge map as information outside the text to supplement the semantics of words. The state-of-the-art version of ERNIE has reached 3.0 [16]. Unlike the BERT and ERNIE models, based on the BERT model, the XLNet model replaces the traditional Transformer with Transformer-XL [17] and introduces two improved methods, namely the "permutation language model" and the "two-stream self-attention" mechanism, to achieve better performance [18]. Text is a type of temporal dataset. Before the emergence of attention mechanisms, models such as LSTM and RNN were widely used for the semantic extraction of temporal data [19]. However, these models always suffer from the problem of gradient disappearance as the text becomes too long. Attention mechanisms can avoid the problem of gradient disappearance and better understand text semantics [20].

However, the labels of poverty governance texts (PGTs) have a hierarchically fixed relationship. Moreover, for a specific training dataset, the label that reaches the relation tree leaves has less simple data [10]. Therefore, an unbalanced training dataset results in poor holistic classification accuracy. Because the current research results cannot solve the multi-label classification problem in the above situations better, this study focuses on hierarchical labels and limited training datasets to conduct multi-label classification research on PGTs. The research innovation of this article mainly lies in the following four aspects: (1) designing a hierarchical label tree for PGTs and constructing a standardized HTMC-PGT corpus through preprocessing and data labeling; (2) proposing an overall model to solve the hierarchical and transferable multi-label classification of PGTs (HTMC-PGT); (3) verifying

the optimality of feature extraction based on XLNet for poverty alleviation governance text classification; and (4) proposing a transfer learning mechanism between hierarchical label classifiers based on an attention mechanism integrated with Bi-LSTM. Based on the results of this study, HTMC-PGT can achieve better hierarchical multi-label classification effects with limited training datasets and provide data support for further research on the policy analysis and knowledge graph of poverty governance [5,21,22]. The rest of the paper is organized as follows: Section 2 highlights the hierarchical label tree of PGTs and the formal definition of the HTMC-PGT problem; Section 3 presents the primary models and methods utilized in this study; Section 4 introduces and discusses the process, methods, and results of the experiments; the final section concludes the paper.

2. Preliminaries

Compared with traditional multi-label classification problems, the HTMC-PGT proposed in this paper has a hierarchy in the label system. This hierarchical and multi-label feature comes from PGT itself and the internal correlation logic between PGTs (see Section 2.1). Unlike traditional HTMC problems that focus on solving single-classifier parameters, HTMC-PGT focuses on solving the parameter set (partial order set) of the classifier tree. This change undoubtedly increases the difficulty of studying the problem (see Section 2.2).

2.1. Hierarchical Label Tree of PGTs

With the openness and transparency of information in the internet era, PGTs can accurately reflect the dynamic process of poverty governance. The government's publicly available PGTs include crucial elements, such as poverty issues, governance policies, working methods, and governance effectiveness, which are presented in four main types of text: policy documents, dynamic news, working papers, and public announcements [23]. In contrast to the analysis methods of traditional structured data surrounding these four types of unstructured texts of poverty governance, it is easy to gain insights into the life cycle processes of poverty governance policy formation, implementation, and refinement, which contribute to the formation of poverty governance knowledge that can be accumulated or shared globally (see Figure 1).

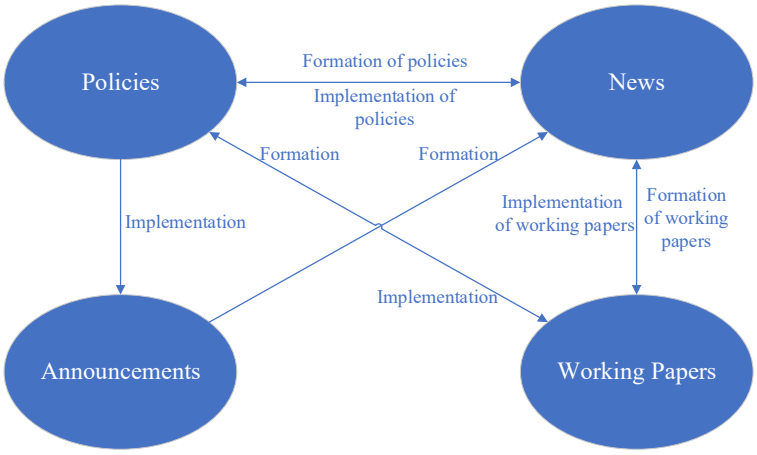

Figure 1. The interclass correlation of PGTs.

The classification of texts is a prerequisite for extracting and mining the entities and relations of PGTs. On the one hand, different text categories are heterogeneous in the structure of semantic information. Therefore, the corresponding ontology model must be used according to the text classification results. On the other hand, the relationship between text classes is hierarchical; therefore, only a hierarchical multi-label classification of text can fully reflect the ontological relationship between texts based on categories. Therefore, depending on the need for knowledge extraction, PGTs should be subdivided

into multilevel and multidimensional collections. These collections of subclasses with hierarchical relationships constitute a hierarchical label tree rooted in the PGTs. The label trees of different subclasses have different labels and depths (Figure 2).

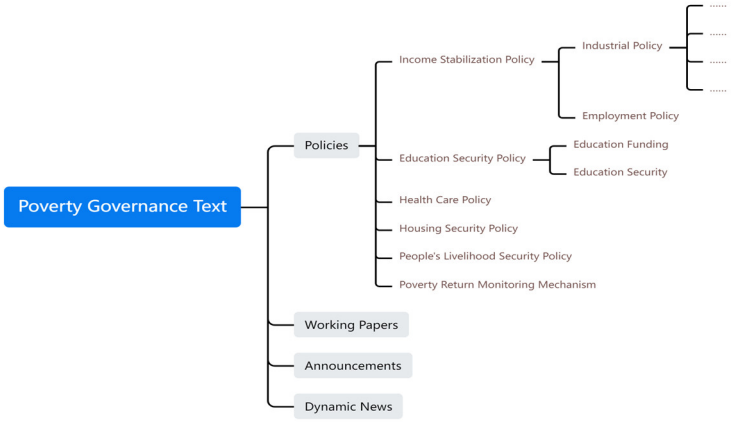

Figure 2. Hierarchical label tree of PGTs.

2.2. Problem Definition

In the HTMC-PGT problem, a set of PGTs exists. Each text in this set contains a fixed number of words that together constitute the semantics of the text. Based on semantic features, these texts also correspond to multiple categories with a hierarchical structure. Unlike the nonlinear structure of multiple labels in traditional HMTC (hierarchical multi-label text classification) problems [24], the multiple categories corresponding to text in HTMC-PGT are linear structures. To clearly express the focus of this article, this section first defines the hierarchical structure of multiple labels, text semantics, and the linear structure of multiple target labels.

Definition 1 (**Hierarchical Label Tree of PGT**). *The hierarchical label tree of PGT (HLT-PGT)* $\mathcal{T}$ *introduced in Section 2.1 can be defined as a partial order set* $(C_T, \prec)$ *where* $C_T = \{c_1, c_2, \cdots, c_N\}$ *is a finite set of all categories of PGT and the size of the set is* N. $\prec$ *is a partial order that represents the parent–child relationship between nodes in* $\mathcal{T}$, *which is antireflective and transitive [25]. On the other hand, the node elements in the tree* C_T *can be demarcated into three parts: root, stem, and leaf, corresponding to three non-overlapping subsets* R, S, *and* L; $C_T = R \bigcup S \bigcup L$. *Because each tree only contains one root node,* $|R| = 1$:

- $\forall c_x \in C_T, c_x \nprec c_x$;
- $\forall c_x, c_y, c_z \in C_T$, if $c_x \prec c_y$ and $c_y \prec c_z$ then $c_x \prec c_z$;
- $\forall c_x \in L, \exists c_y \in R \bigcup S, c_x \prec c_y$ then $c_y \nprec c_x$;
- $\forall c_x \in S, \exists c_y \in R \bigcup S, c_x \prec c_y$ then $c_y \nprec c_x$.

Therefore, compared with the nonlinearity of HMTC, the elements in the classification result set of HTMC-PGT form a directed acyclic graph connected by $\prec$ relationships, and the arrangement of nodes is a linear sequence.

Definition 2 (**Linear Structure of Target Multiple Labels**). *In the HTMC-PGT problem, each document* D_i *contains multiple labels* C_{Ti}. *One and only one of these labels* C_{Ti} *belongs to the leaf node set* L *(denoted as* l_i*), and* C_{Ti} *must contain elements from the set* R *(i.e., root nodes, denoted as* γ*), while the other node sets* $S_i = \{c_1, c_2, \cdots, c_k\}$ *, are subsets of the set* S. *If* S_i *is not an empty set, a sequence exists* $\left\{c'_1, c'_2, \cdots, c'_k\right\}$ *such that all elements in* S_i *satisfy the following arrangement* $l_i \prec c'_1 \prec c'_2 \prec \cdots \prec c'_k \prec \gamma$. *Therefore, all elements of the leaf node set* L *correspond one-to-one to the sequence of the multi-label classification results in the HTMC-PGT problem. That is, the set of classification results for the HTMC-PGT problem is the set of paths from all leaf nodes in set* L *to*

the root node. Therefore, in the HTMC-PGT problem, the set of all the classification result sequences can be denoted as follows, where $i = 1, 2, \cdots, |L|$*:*

$$C_S = \{ C_{Ti} | i = 1, 2, \cdots, |L| \} \tag{1}$$

The HTMC-PGT problem exhibits three typical characteristics. First, the classified text has multiple labels, which differs from the traditional single-label text classification. Second, there is a hierarchical semantic relationship between labels, which differs from independent labels. Finally, the hierarchical multi-label classification has strict temporal dependencies among label classifiers. Unlike the binary relevance (BR) method [26], hierarchical multi-label classification decomposes the classification problem with n labels into an $n - m$ multi-class classifiers problem where $n = |C_T|$ is the total number of labels including the root node of the label tree and $m = |L|$ is the total number of label leaf nodes. According to Definition 2, HTMC-PGT aims to have text with a unique leaf node label. Therefore, from the perspective of leaf node sets alone, HTMC-PGT can be considered a single-label classification, also called the "flat classification approach" [27]. However, owing to the imbalanced distribution of label semantics, single-label classification methods often fail to achieve satisfactory classification results. At the same time, it is also impossible to model the problem as a holistic solution that integrates hierarchical label trees and document semantics, as proposed in [24,28,29]. Therefore, it is necessary to start from the root node of the label tree and construct a multi-class classifier to gradually reduce the target leaf node label set until the final target label is obtained (Figure 3).

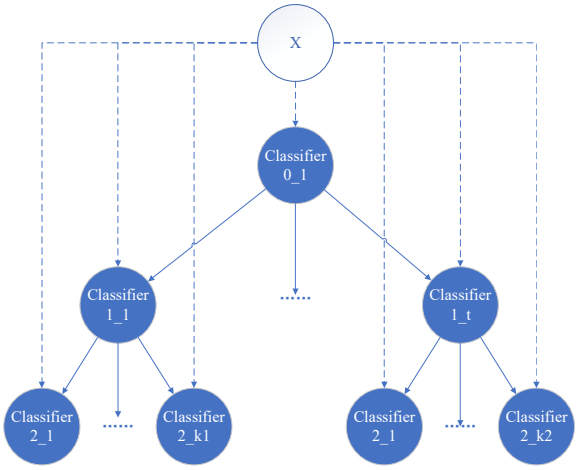

Figure 3. Multi-class classifier tree.

Definition 3 (HTMC-PGT Problem). *Let X be the feature vector of a given poverty governance text. Each classifier in the classifier tree shares X as the input parameter.*

$$\hat{y}_j = C_j(x) = argmin\, H_j(x, y) \quad for\, index\, j = 1, \ldots, \left| n - m \right| \tag{2}$$

In Equation (2), $\hat{y}_j$ is the classification result of feature vector x in the multi-class classifier $C_j(\cdot)$. H_j is the cost function of the classifier $C_j(\cdot)$. By transforming the classification problem into a minimum cost function problem using $H_j(\cdot)$, the optimal classification result can be obtained. Based on the classification results of the previous classifier, the next classifier can be determined and used to continue classifying text feature vector x.

Given a poverty governance text set D and the corresponding hierarchical label tree structure $\mathcal{T} = (C_T, \prec)$, first, text set $\mathcal{D}$ is divided into multiple text sets $\mathcal{D}_i$ based on the classifier set, whereas C_T is divided into several corresponding label sets C_{Ti}. The goal of

HTMC-PGT is to learn the parameter Θ_i of the classifier model Ω_i based on the text set $\mathcal{D}_i$ and corresponding multiple labels C_{Ti}, and ultimately predict the label sequence of a document using the model in the hierarchical classifier tree. To improve the performance of classifiers with a small sample number $|\mathcal{D}_i|$, this study adopts the method of transfer learning between classifiers, that is, the model parameter Θ_i of the parent node in set S is taken as the initial value of the model parameter of the child node, and the training effect of the child node is improved through incremental calculations:

$$\Omega_i(\mathcal{D}_i, \Theta_i, \Theta_{ip}) \longrightarrow C_{Ti} \tag{3}$$

where Θ_{ip} is the parameter set of the parent node of this classifier in the classifier tree and $i = 1, \cdots, |R \bigcup S|$.

The HTMC-PGT problem requires solving Equations (2) and (3) and forming a classifier tree $Y = (\Theta, \prec)$, which can be used to classify any PGT:

$$\Omega(D, Y) \longrightarrow C_S \tag{4}$$

3. Materials and Methods

This study aims to build a classifier tree for the problem defined in Section 2, in which transfer learning can occur among classifiers. Thus, the classifier tree can achieve optimal classification accuracy for the HTMC-PGT problem. From a data flow perspective, the PGT must undergo unified data preprocessing before entering the classifier tree to avoid redundant calculations at all levels of classifiers. In Definition 3 of Section 2, the classifier tree is defined as $Y = (\Theta, \prec)$, which is isomorphic to HTMC-PGT without leaf nodes, noted as $(R \bigcup S, \prec)$. The classifier tree can determine whether transfer learning can be carried out between two classifiers according to the semantic relationship between the parent and child nodes; in other words, some of the parameters of the parent classifier are used as the initial parameters of the child classifier in the training, to achieve the goal of incremental training of the child label classification model when the number of training samples of the child classifier is relatively small (Figure 4).

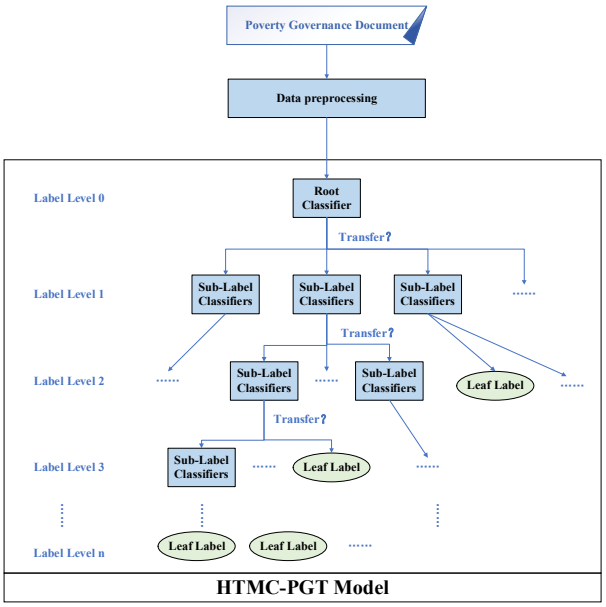

Figure 4. HTMC-PGT model.

All classifiers adopt the same architecture model based on XLNet and the attention mechanism to facilitate knowledge transfer between classifiers. In addition, this study utilizes a pretrained XLNet model to extract features from PGT and obtain char-embeddings for PGT sentence segmentation, thus completing the data preprocessing process. Then,

starting from the root node of the classifier tree, a hierarchical attention mechanism based on MLP is implemented for char-level embeddings in PGT-segmented sentences to obtain PGT semantic feature vectors corresponding to the text and category, which can be used for the final classification. Finally, when training the sub-classifier, the parameters of the attention mechanism in the parent classifier are transferred to the initial values of the corresponding parameters in the sub-classifier. Then, the classification effect of the sub-classifier is improved through the transfer learning method.

3.1. Features Extracted through XLNet Model

XLNet is an improved model for BERT proposed by scholars from Carnegie Mellon University and the Google AI Brain Team in 2019 [18]. Unlike the self-coding language model method based on BERT dependency masks, XLNet adopts an autoregressive language model to avoid the problem of ignoring dependencies between mask positions in BERT and the negative impact of artificially introducing "[MASK]" markers. Meanwhile, XLNet overcomes the drawbacks of traditional autoregressive models, which can only utilize unidirectional information, by introducing a bidirectional context learning method based on self-coding language models. In addition, it replaces the Transformer in BERT with Transformer XL. This new neural network framework has a segment-level recurrence mechanism (segment-level recurrence with state reuse) and a new location coding strategy (relative positional encoding), which can not only break through the limitation of fixed context length in language modeling of Transformer, thus learning long-distance dependencies in text sequences, but also solve the problem of context fragmentation [17] and improve the overall performance of the model.

The most prominent feature of XLNet compared to BERT and other models is that it introduces the Transformer XL framework to solve the problem of long text dependencies. This is of great significance for the research presented in this paper: PGTs are long, structurally complex, and diverse. However, owing to fixed length limitations, the traditional Transformer framework cannot effectively capture the semantic connections between segments, leading to BERT's inability in terms of long-term text dependency.

Regarding specific methods, Transformer-XL introduces a recurrence mechanism between segments so that each segment does not need to start entirely from scratch during calculation but can learn from the information of the previous segment [17]. The output of the hidden layer in the previous section enables the model to accumulate long-term dependencies during the training phase. Furthermore, the model can learn longer distance dependencies in the text during testing.

For the HTMC-PGT problem (see Definition 3 in Section 2 of this paper), each classifier in the classifier tree must use XLNet for feature extraction, which is time-consuming. Therefore, this study separates the feature extraction layers that commonly exist in a single classifier in other studies and places them in the data preprocessing stage to maintain the simplicity of each classifier in the next stage, thereby improving the overall training and testing efficiency of the model.

In the data preprocessing stage, the PGT is first split into paragraphs of consistent length. Then, the words of each paragraph are input into XLNet's pretraining model to extract the corresponding char-level semantic features, forming a char-level semantic feature matrix based on segments. Meanwhile, based on the algorithmic advantages of XLNet for solving long-distance dependencies, the hidden layer information of the previous paragraph is also input into XLNet's pretraining model when calculating each paragraph, thus preserving the long-distance dependencies between the long-text paragraphs [17] (Figure 5).

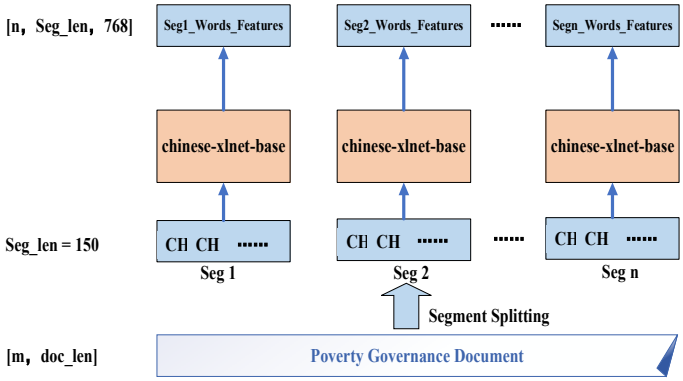

Figure 5. Data preprocessing based on XLNet.

The preprocessing first involves cropping all texts in the PGT set to a fixed length and dividing them into multiple segments. If the length of the text is insufficient, it is filled with placeholders. Then, the fixed length document set is denoted as D and the individual documents in the set are denoted as D_i, where the number of PGT set D is noted as m, and $i = 1, \cdots, m$. Therefore, the PGT set D can be represented as a matrix form of D_i:

$$D = |D_1 D_2 \cdots D_m| \tag{5}$$

Each PGT D_i can be divided into several fixed-length segments, the number of which is n. Therefore, text D_i can be represented as a vector form of paragraph s_{il}, where $l = 1, \cdots, n$:

$$D_i = |s_{i1} s_{i2} \cdots s_{in}| \tag{6}$$

and each segment of s_{il} can be represented as a vector of word tokens:

$$s_{il} = \left|w^t{}_{il1} w^t{}_{il2} \cdots w^t{}_{ilk}\right| \tag{7}$$

where k is the length of a segment.

Then, taking a single PGT D_i as a unit, all segments in the text D_i are input into the XLNet pretrained model as parameters, and the char-embedding vectors of all word tokens in each segment and the hidden layer information m_{il} are obtained using transfer learning. This information can be input into the XLNet pretrained model as parameters when solving the char-embedding vector of the next segment, thus preserving the semantic association among the segments.

$$m_{il}, s^{'}_{il} \leftarrow XLNet\_WordEmbedding\left(s_{il}, m_{i(l-1)}\right) \tag{8}$$

$$s^{'}_{il} = \left|w^e{}_{il1} w^e{}_{il2} \cdots w^e{}_{ilk}\right| \tag{9}$$

Set the number of dimensions of the char-embeddings ($w^e{}_{ilt}, t = 1, \cdots, k$) to d. The most commonly used d in XLNet is 768.

3.2. Hierarchical Attention Mechanism

This study builds a basic model of a single classifier in the classifier tree based on the improved multilayer perception mechanism to complete the mapping from the segmented char-embeddings to categories after data preprocessing (Figure 6). From a functional perspective, the model can be divided into three parts. First, a hierarchical attention mechanism layer solves the problem of transitioning from char-level to document-level embeddings [30]. The second layer is the fully connected layer, which converts document-level embeddings into category probabilities. Finally, an activation layer uses the function Argmax to find the target category with the highest probability.

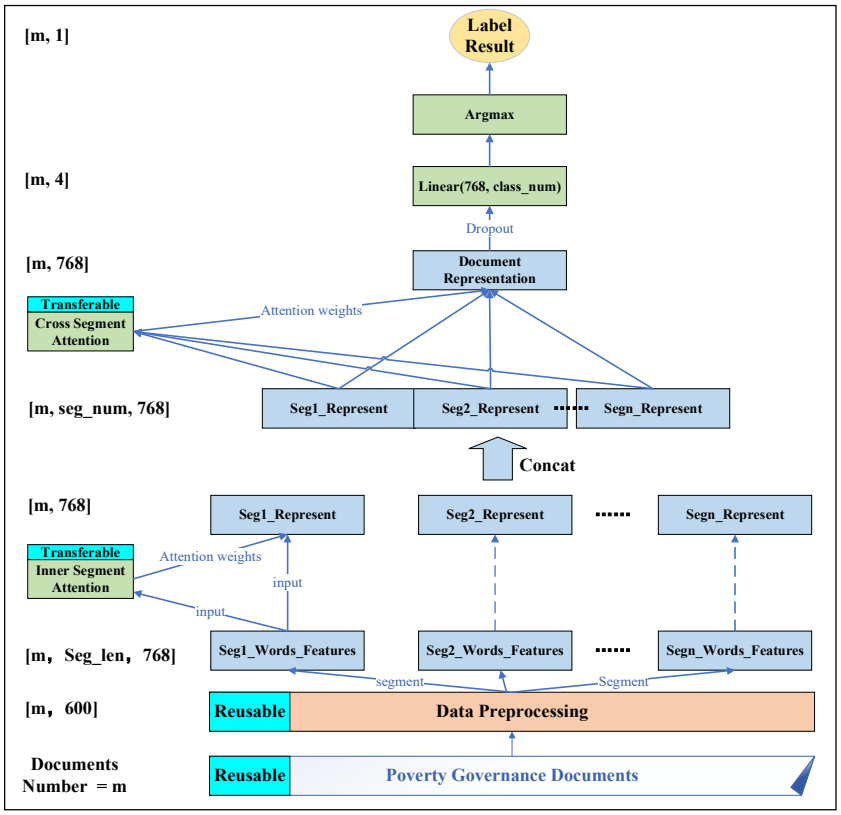

Figure 6. Basic transferable classifier model for classifier tree.

The hierarchical attention mechanism layer utilizes hierarchical MLP to learn and sequentially obtain (1) attention weights from char-level embeddings to segment-level embeddings mapping and (2) attention weights from segment-level embeddings to document-level embeddings [31]. Using this hierarchical calculation method, vectors that represent the semantic features of documents are obtained. Furthermore, this hierarchical attention mechanism layer can be further divided into a lower-order representation layer and a higher-order representation layer. Of these, the lower-order representation layer represents the semantics of words, whereas the higher-order representation layer represents the semantics in paragraphs. This method of hierarchical attention mechanism can extract document-level embeddings from the text on the one hand and shorten the length of input text in each processing unit through text segmentation and layering on the other hand, thus avoiding the vanishing gradient problem common in long text processing [32].

In the low-order representation layer, the char-embeddings $s^{'}_{il}$ of each segment are converted into a hidden layer whose dimension number is half of the char-embeddings size through the linear function. Formally, let $X \in R^{\mathrm{m}\times\mathrm{n}\times\mathrm{k}\times d}$ be the input char-embeddings, where m is the amount of PGT, n is the number of segments in a text, k is the number of words contained in each segment, and d is the dimension of the char-embeddings. Moreover, let $x_{il} \in R^{\mathrm{k}\times\mathrm{d}}$ be the char-embeddings matrix of segment l in document i. $A_{il} \in R^{\mathrm{k}\times(\mathrm{d}//2)}$ denotes the activations and is given by

$$A_{il} = \phi\left(W^1{}_l x_{il}{}^T + b^1{}_l\right)^T \tag{10}$$

where $W^1{}_l \in R^{(\mathrm{d}//2)\times d}$ is a weight matrix and $b^1{}_l \in R^{(\mathrm{d}//2)\times 1}$ is the bias vector in the low-order layer. Meanwhile, ϕ is a nonlinear activation function, which is taken as the function *Tanh* in this study, that is,

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{11}$$

Let $O_{il} \in R^{1\times d}$ be the output of the low-order representation layer, given by

$$O_{il} = \sigma\left(W^2{}_l A_{il}{}^T\right) x_{il} \tag{12}$$

where $W^2{}_l \in R^{1\times(d//2)}$ is a weight matrix that can convert A_{il} into a vector and σ denotes the function $softmax(\cdot)$ function, which can calculate the weights of each word for the semantic vectors of a segment.

After obtaining segment-level embeddings in the higher-order representation layer, it is necessary to concatenate the embeddings of all segments in the same document in the order of the corresponding segments in the text. Therefore, let $O_i \in R^{n\times d}$ be the segment-level embeddings matrix of text i and $\bigodot$ denote vector concatenation. Then, O_i can be calculated as

$$O_i = O_{i1} \bigodot O_{i2} \bigodot \cdots \bigodot O_{in} \tag{13}$$

Subsequently, the higher-order representation layer uses a method like the low-level representation layer to calculate document-level embeddings. Let $A_i \in R^{n\times(d//2)}$ denote the activations given by

$$A_i = \phi\left(W^1{}_h O_i{}^T + b^1{}_h\right)^T \tag{14}$$

where $W^1{}_h \in R^{(d//2)\times d}$ is a weight matrix and $b^1{}_h \in R^{(d//2)\times 1}$ is the bias vector in the high-order layer.

Let $H_i \in R^{m\times d}$ be the output of the high-order representation layer, which is given by

$$H_i = \sigma\left(W^2{}_h A_i{}^T\right) O_i \tag{15}$$

where $W^2{}_h \in R^{1\times(d//2)}$ is a weight matrix that can convert A_i into a vector.

At the end of the classifier model, let C be the target category set that corresponds to the current classifier. Therefore, $|C|$ denotes the number of categories. Therefore, document-level embeddings can be implicitly classified as category probabilities through a fully connected layer, forming a document category probability matrix. Let $I \in R^{m\times 1}$ be the final classification result of the classifier, which can be given by

$$I = Argmax\left(\left(W_f H^T + b_f\right)^T\right) \tag{16}$$

where $W_f \in R^{|C|\times d}$ is a weight matrix of the fully connected layer, $b_f \in R^{|C|\times m}$, and the function $Argmax$ takes the category with the highest probability corresponding to the PGT.

3.3. Hierarchical Attention Mechanism Fused with Bi-LSTM

Reference [28] proposed a method for extracting contextual information from char-embeddings sequences using Bi-LSTM networks prior to hierarchical attention mechanisms. Although the performance of Bi-LSTM in the semantic learning of long-distance dependencies outweighs traditional RNN and LSTM [33], this method is unsuitable for long texts. It also introduces too many parameters that are inappropriate for an unbalanced dataset. In contrast, reference [27] proposed an algorithm structure for Bi-LSTM based on an attention mechanism. However, this attention mechanism does not consider the inherent hierarchical characteristics of text data and cannot effectively achieve deep fusion of HA and Bi-LSTM to extract semantic features suitable for classification problems. Therefore, this paper improves on these deficiencies: based on the hierarchical attention mechanism above, a Bi-LSTM network layer is added between the low-order and higher-order representation layers to extract the contextual semantic information of the segment-level embeddings.

The specific algorithm adds a group of calculation formulas for the Bi-LSTM network between formulas (13) and (14) of HA in the previous text. That is, taking the output result sequence of Equation (13) as the input of the Bi-LSTM network, a new result sequence

with the same structure is calculated and finally is input into Equation (14). The detailed calculation process is as follows [28,33]:

$$\begin{aligned} \vec{h}_{in} &= LSTM\left(\vec{h}_{i(n-1)}, O_{in}\right), \\ \overleftarrow{h}_{in} &= LSTM\left(\overleftarrow{h}_{i(n-1)}, O_{in}\right), \\ h_{in} &= \left[\vec{h}_{in}, \overleftarrow{h}_{in}\right], \\ O_i = h_i &= h_{i1} \odot h_{i2} \odot \cdots \odot h_{in} \end{aligned} \tag{17}$$

where $\overleftarrow{h}_{in}, \vec{h}_{in} \in R^{1\times(d//2)}$, $\overleftarrow{h}_{in} \in R^{1\times d}$.

3.4. Transfer Learning with Classifier Tree

This study used two main transfer learning methods [34]. One is feature-based transfer, in which the output of the coaching model is used as input to the student model, thereby improving the training effectiveness of the student model. The char-embeddings extracted by XLNet mentioned above are precisely the transfer learning method in this form [35]. Another type is parameter-based transfer [36], which means that some parameters in the coaching model are directly transferred to the student model, and these parameters can be fixed or updated later through post-propagation. This section uses the latter method to transfer the parameters across the classifier tree. Then, based on the parent classifier parameters, the training samples of the child classifier are further used for incremental training to accelerate the training convergence speed and classification accuracy of the child classifier model.

In any classifier model in the tree, the parameters that can be used for migration are the HA and Bi-LSTM + HA (HAs). In other words, the parameters related to the extraction process of document-level embeddings are transferred. Because the parameters of FC are unsuitable for transferring due to the different target categories of each classifier, formally, set $\Theta_i \in Y$ is the parameters set of the classifier i in the classifier tree $Y = (\Theta, \prec)$. Therefore, the set of weights ${W^1}_{l|h} \in R^{(d//2)\times d}$ and ${W^2}_{l|h} \in R^{1\times(d//2)}$ and bias ${b^1}_{l|h} \in R^{(d//2)\times 1}$ and parameters of the Bi-LSTM network in Equations (11), (13)–(15) and (17), respectively, can be denoted as $\mathcal{E}_i$. For other classifier nodes j that satisfy the $\Theta_j \prec \Theta_i$ in tree Y, namely the child nodes of classifier i, the parameter set $\mathcal{E}_i$ can be used as the initial value of the parameter set $\mathcal{E}_j$ for the training of the classifier j.

Theoretically, the effectiveness of this parameter-level transfer learning method is to improve the learning effect of parameters with a specific loss function and parameter update strategy. In the training phase, each classifier uses cross-entropy as a loss function [24] as follows:

$$L = -\sum_{i=1}^{m}\left[y^{(i)} log\hat{y}^{(i)} + \left(1 - y^{(i)}\right) log\left(1 - \hat{y}^{(i)}\right)\right] \tag{18}$$

Simultaneously, Adam [37] is used as the optimizer algorithm to optimize and update the parameters, which is given by

$$\begin{aligned} W+ &= -LearningRate \times \frac{m}{\sqrt{v}} \\ m &= b_1 \times m + (1 - b_1) \times dx \\ v &= b_2 \times v + (1 - b_1) \times (dx)^2 \end{aligned} \tag{19}$$

The transfer learning method proposed in this section can theoretically make the classifier model have an initial value closer to the optimal solution, thereby reducing the dependence of the child classifier on the amount of training sample data and improving the convergence efficiency of the training process.

4. Results and Discussion

This section briefly introduces the datasets used in this study and the experimental environment configuration and parameter settings. Then, baseline models, such as BERT and ERNIE, are proposed, and the evaluation metrics for the experimental results are determined. Finally, the performance of the XLNet model, the effect of HAs, and the effect of HA-based transfer learning are evaluated from multiple perspectives. The purposes of these experiments are to answer the following four questions.

RQ1: How does the XLNet pretrained model perform in extracting text semantic features?
RQ2: Which types of HAs have the best performance in learning document-level embeddings?
RQ3: Can HA-based transfer learning improve the training and testing performance of the classifier trees?
RQ4: Can the XLNet + Has + TL model achieve the best performance for the classifier tree?

4.1. Dataset Description

Owing to the lack of publicly available textual datasets on poverty governance, the authors' team created the dataset used in the experiments. Its source is the webpage text searched and extracted by the authors using a web crawler program on the Internet with poverty governance-related keywords in the Chinese style, such as "targeted poverty alleviation", "poverty alleviation", "returning to poverty", etc. Then, we cleaned the webpage text data, filtered the HTML tags and characters unrelated to the main text, and unified the encoding of all texts. To eliminate the influence of geographical names, time, and people's names on the classification results of nouns, we replaced these words in the original texts with unified category markers. Next, based on the standardized format of the texts, industry experts on our team were organized to annotate the text using HLT-PGT. To eliminate as many non-objective factors as possible from the labeling, these experts were divided into three groups, two of which conducted their work back to back. Finally, texts with inconsistent markings from these two groups of experts were taken out and handed over to the third group for independent evaluation. We obtained a labeled dataset with this organizational structure and process (Table 1).

4.2. Experimental Setups

To verify the effectiveness of HTMC-PGT, this study uses a 7:3 ratio to partition the dataset, forming a training set (70%) and a validation set (30%), with the testing set being the same as the validation set. We do not plan to optimize the hyperparameters (mentioned later) in the experiments, aiming to simplify constructing the classifier tree. Therefore, preserving an independent testing set is unnecessary, which can also avoid inconsistent data distribution between datasets (for training, validation, and testing) when randomly extracted from the original dataset.

Document Model Settings. We set the word count of PGT documents to 600, which means that documents exceeding this limit will be truncated, whereas documents below this length will be filled with the placeholder "[PAD]". Each document is divided into 150 segments, each containing four words.

XLNet Settings. We downloaded the pretrained model "chinese-xlnet-base" from Hugface, which has 13 hidden layers, each with a dimension of 768 [18]. Because the last hidden layer extracts the semantic information of words in the original text, we use the output of this layer as the char-level embeddings to input the classification model of PGT [35].

Training Details [28,38]. We use Adam as the optimizer and set the learning rate to 0.0005, the weight rate to 0.0001, and the batch size to 32. In addition, for the sub classifier i whose parent node j in the classifier tree Y has already obtained the optimal training parameters, the parameters' values ε_j of HAs in the parent node classifier j can be directly used as the initial value of the corresponding parameters ε_i of the sub classifier i. All

parameters in HAs and FC can be automatically tuned during training. At the same time, to avoid overfitting, we use dropout technology [39] and set the dropout rate to 0.5.

Platform Settings. These experiments are implemented on the Windows Server 2020 platform with Intel Core processors i7-7800X@3.5 GHz and a memory of 32 GB (DDR4 2666 MHz 16 GB * 2). The program is encoded in Python 3.6 and uses Pytorch 1.7.0 as its machine learning library. To improve the efficiency of the tensor calculation, the machine is equipped with a graphics card (NVIDIA GeForce GTX 1080 Ti) with a graphics memory of 11 GB, and the CUDA (Compute Unified Device Architecture) version is 10.2.

Table 1. Summary of datasets for each classifier in the classifier tree *.

Index	Classifier No.	Parent Classifier	|C|	|D|	Training Set		Valid Set	
					Total	PL	Total	PL
1	C00000	—	4	13,979	9785	2446	4194	1049
2	C01000	C00000	6	4620	3233	539	1387	231
3	C01100	C01000	2	1383	967	484	416	208
4	C01110	C01100	5	1017	711	142	306	61
5	C01120	C01100	4	365	255	64	110	28
6	C01200	C01000	2	1673	1170	585	503	252
7	C01300	C01000	2	475	332	166	143	72
8	C01310	C01300	2	194	135	68	59	30
9	C01320	C01300	7	280	195	28	85	12
10	C01400	C01000	2	228	159	80	69	35
11	C01410	C01400	3	156	108	36	48	16
12	C01420	C01400	4	71	49	12	22	6
13	C01500	C01000	2	659	460	230	199	100
14	C01510	C01500	2	440	307	154	133	67
15	C01600	C01000	2	197	137	69	60	30
16	C02000	C00000	3	4890	3422	1141	1468	489
17	C02100	C02000	2	3711	2597	1299	1114	557
18	C02110	C02100	4	2788	1951	488	837	209
19	C02120	C02100	2	922	645	323	277	139
20	C02200	C02000	4	652	456	114	196	49
21	C02300	C02000	2	525	367	184	158	79
22	C02310	C02300	2	456	318	159	138	69
23	C03000	C00000	4	3202	2241	560	961	240
24	C04000	C00000	3	1264	884	295	380	127
25	C04100	C04000	2	268	187	94	81	41
26	C04200	C04000	2	511	357	179	154	77
27	C04300	C04000	2	483	337	169	146	73

* Classifier number (Classifier No.), number of labels (|C|), number of documents (|D|), the total number of instances (Total), the total number of instances per label (PL).

4.3. Baseline Models

We construct a baseline model from three perspectives for the experimental evaluation: (1) an extraction method for char-level embeddings. We use pretraining models based on BERT, such as XLNet [18], BERT-BASE (BERT), BERT-WWM (WWM) [40], and ERNIE3.0 [16], to extract char-level embeddings representations during the preprocessing. (2) An extraction method for document-level embeddings. We use models such as HAs (HA or Bi-LSTM + HA) and FastText (FT) [41] to extract document-level embeddings representations for classification based on char-level embeddings. (3) Whether to enable transfer learning between classifiers. That is, whether transfer learning (TL) is enabled for classifiers using HA models in the classifier tree. Therefore, we design 13 models based on these elements from these three perspectives. Of these, the first one, XLNet + HAs + FC + TF, is the HTMC-PGT model proposed in this paper, and the remaining 12 models are the baseline models (see Table 2).

Table 2. Baseline models in each experiment.

Ex.	Model	W-Embeddings	D-Embeddings	Transfer Learning
Ex. I	XLNet + HA + FC	XLNet	HA	/
	BERT + HA + FC	BERT	HA	/
	WWM + HA + FC	BERT-WWM	HA	/
	ERNIE3.0 + HA + FC	ERNIE3.0	HA	/
Ex. II	XLNet + HA + FC	XLNet	HA	/
	XLNet + FT + FC	XLNet	FT	/
	XLNet + BiLSTM + HA + FC	XLNet	Bi-LSTM + HA	/
Ex. III	XLNet + HA + FC	XLNet	HA	/
	XLNet − HA − FC − TL	XLNet	HA	TL
	XLNet + BiLSTM + HA + FC	XLNet	BiLSTM + HA	/
	XLNet + BiLSTM + HA + FC + TL	XLNet	BiLSTM + HA	TL
Ex. IV	XLNet + BiLSTM + HA + FC + TL	XLNet	BiLSTM + HA	TL
	XLNet + BiLSTM + HA + FC	XLNet	BiLSTM + HA	/
	XLNet + HA + FC + TL	XLNet	HA	TL
	XLNet + HA + FC	XLNet	HA	/
	WWM + HA + FC	BERT-WWM	HA	/
	BERT + HA + FC	BERT	HA	/
	ERNIE3.0 + HA + FC	ERNIE3.0	HA	/
	XLNet + FT + FC	XLNet	FT	/

4.4. Evaluation Metrics

This study assesses the effectiveness of specific methods by utilizing precision (P), recall (R), and F-measure (F1) as metrics [42]. The classifier tree models used in this study differ in their training and testing stages, leading to the division of these metrics into three categories based on specific algorithms: micro-label, macro-averaging [11,43], and micro-tree. These categories are used to evaluate the performance of the model during the training and testing stages. We compare these three types of metrics and use the micro-label and macro-averaging of the precision P, recall R, and F1 to determine the overall average performance of all classifiers in the classifier tree in each class. In contrast, we use the micro-tree of precision P, recall R, and F1 to observe the performance of the classifier tree in the classification results.

Given a classifier $i \in \mathbb{C}$ ($\mathbb{C}$ is the classifier set for a classifier tree) and category $j \in \mathcal{L}_i$ ($\mathcal{L}_i$ is the category set of classifier i), let TP_i^j, FP_i^j, and FN_i^j be the values of true positives, false positives, and false negatives, respectively. Then, the P_i, R_i, and $F1_i$ metrics for classifier i are defined as follows [11,28]:

$$\begin{aligned} P_i &= \frac{\sum_{j\in\mathcal{L}_i} TP_i^j}{\sum_{j\in\mathcal{L}_i} TP_i^j + \sum_{j\in\mathcal{L}_i} FP_i^j}, \\ R_i &= \frac{\sum_{j\in\mathcal{L}_i} TP_i^j}{\sum_{j\in\mathcal{L}_i} TP_i^j + \sum_{j\in\mathcal{L}_i} FN_i^j}, \\ F1_i &= \frac{2\times P_i \times R_i}{P_i + R_i} \end{aligned} \tag{20}$$

To combine the results measured by all classifiers in the classifier tree, we use the macro-averaging of the precision P_{macro}, recall R_{macro}, and $F1_{macro}$ [11]:

$$\begin{aligned} P_{macro-train} &= \frac{\sum_{i=1}^{n} P_i}{n} \\ R_{macro-train} &= \frac{\sum_{i=1}^{n} R_i}{n} \\ F1_{macro-train} &= \frac{\sum_{i=1}^{n} F1_i}{n} \end{aligned} \tag{21}$$

Given a label $k \in$ K (K is the set of leaf nodes on the multi-label tree), let TP_k, FP_k, and FN_k, be the values of true positives, false positives, and false negatives, respectively, in the test stage. Subsequently, the P_k, R_k, and $F1_k$ metrics are defined as

$$P_k = \frac{TP_k}{TP_k + FP_k}, R_k = \frac{TP_k}{TP_k + FN_k}, F1_k = \frac{2 \times P_k \times R_k}{P_k + R_k} \tag{22}$$

$$P_{macro-test} = \frac{\sum_{i=1}^{m} P_i}{m}, R_{macro-test} = \frac{\sum_{i=1}^{m} R_i}{m}, F1_{macro-test} = \frac{\sum_{i=1}^{m} F1_i}{m} \tag{23}$$

$$\begin{aligned} P_{micro-test} &= \frac{\sum_{k \in K} TP_k}{\sum_{k \in K} TP_k + \sum_{k \in K} FP_k}, \\ R_{micro-test} &= \frac{\sum_{k \in K} TP_k}{\sum_{k \in K} TP_k + \sum_{k \in K} FN_k}, \\ F1_{micro-test} &= \frac{2 \times P_{micro} \times R_{micro}}{P_{micro} + R_{micro}} \end{aligned} \tag{24}$$

4.5. Experiment I: The Performance of the XLNet Model

The primary purpose of Experiment I is to verify the effectiveness of using XLNet to extract text semantic features using a combination of pretrained models and hierarchical attention. We select pretrained models in the experiment, such as BERT base, BERT-WWM, and ERNIE3.0, as baseline models. First, we train 27 classifiers for each of the four classifier trees, totaling 108 classifiers. The corresponding P, R, and F1 values are then calculated for each classifier based on the metrics described in Section 4.4. The detailed results are presented in Table 3. Among the four models listed in Table 4, no pretrained model has a 100% overwhelming absolute advantage for the 27 classifiers. However, from a relative perspective, the advantage of XLNet's training effect is still significant (Figure 7). In Figure 7, the horizontal axis represents each model's performance metrics P, R, and F1. In contrast, the vertical axis represents the proportion of classifiers with the best training performance compared to other models for each metric. Among each group of the 27 classifiers for the four classifier trees, from the perspective of P, the group using XLNet has 16 (59%) classifiers outperform the groups using the other three baseline models in terms of training performance. The best classifiers using BERT, WWM, and ERNIE3.0 are 10 (37%), 4 (19%), and 2 (7%), respectively.

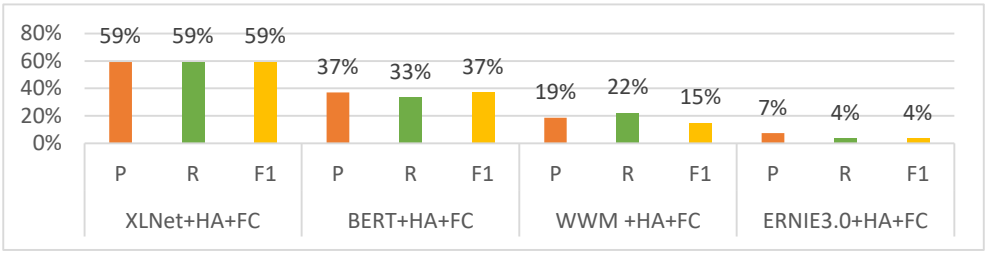

Figure 7. Comparison of training advantages between XLNet and baseline models such as BERT, WWM, and ERNIE3.0.

Based on Table 3, we calculated the mean values of P, R, and F1 for the 27 classifiers in each model. The model using XLNet demonstrates the best performance in the training of the classifier tree, with an average of P value as 93.7%, R value as 93.3%, and F1 value as 93.2%. Moreover, the results show that the performance of XLNet is 2.7%, 5%, and 15% higher than the other baseline models on P values, 4.4%, 5.9%, and 18.1% higher on R values, and 4.3%, 6%, and 18.8% higher on F1 values, respectively. Therefore, for RQ1 "How does the XLNet pretrained model perform in extracting text semantic features?", we can conclude that the training results using the XLNet pretrained model in the classifier tree are significantly better than those of the other three models.

Table 3. The performance of the XLNet model.

Classifier No.	XLNet + HA + FC			BERT + HA + FC			WWM + HA + FC			ERNIE3.0 + HA + FC		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
C00000	**0.970** *	**0.967**	**0.969**	0.963	0.960	0.961	0.965	0.964	0.964	0.948	0.957	0.952
C01000	**0.956**	**0.934**	**0.944**	0.940	0.919	0.928	0.951	0.936	0.943	0.923	0.913	0.917
C01100	0.977	0.990	0.983	**0.987**	**0.984**	**0.986**	0.979	0.988	0.983	0.958	0.955	0.957
C01110	**0.974**	**0.892**	**0.926**	0.965	0.744	0.781	0.731	0.598	0.594	0.303	0.394	0.335
C01120	**0.971**	**0.989**	**0.979**	0.924	0.953	0.936	0.972	0.988	0.979	0.610	0.502	0.483
C01200	**0.955**	**0.952**	**0.953**	0.893	0.884	0.889	0.952	0.955	0.954	0.889	0.900	0.895
C01300	0.826	0.816	0.821	**0.918**	**0.905**	**0.910**	0.884	0.878	0.881	0.928	0.886	0.900
C01310	**0.962**	**0.972**	**0.966**	**0.962**	**0.972**	**0.966**	0.951	0.944	0.947	0.821	0.763	0.773
C01320	**0.953**	**0.833**	**0.863**	0.638	0.512	0.532	0.427	0.326	0.314	0.340	0.329	0.310
C01400	0.809	0.830	0.819	0.809	0.848	0.824	0.812	0.866	0.829	**0.826**	**0.858**	**0.839**
C01410	**0.802**	**0.844**	**0.808**	0.563	0.637	0.554	0.649	0.692	0.653	0.489	0.389	0.308
C01420	**0.969**	**0.917**	**0.933**	0.659	0.727	0.683	0.384	0.464	0.420	0.125	0.250	0.167
C01500	0.780	0.772	0.775	**0.862**	**0.846**	**0.853**	0.826	0.816	0.820	0.791	0.765	0.775
C01510	0.977	0.970	0.974	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.951	0.878	0.905
C01600	**0.978**	**0.941**	**0.958**	0.825	0.665	0.688	0.958	0.883	0.912	0.922	0.765	0.804
C02000	**0.989**	**0.982**	**0.986**	0.978	0.976	0.977	0.983	0.970	0.976	0.970	0.969	0.969
C02100	0.996	0.996	0.996	**1.000**	**0.998**	**0.999**	0.996	0.994	0.996	0.985	0.989	0.987
C02110	**0.986**	**0.980**	**0.983**	0.981	0.970	0.975	0.980	0.967	0.973	0.709	0.633	0.648
C02120	**0.995**	**0.997**	**0.996**	0.987	0.990	0.989	0.990	0.987	0.988	0.940	0.952	0.945
C02200	0.670	0.784	0.704	**0.936**	**0.833**	**0.839**	0.935	0.818	0.830	0.531	0.603	0.533
C02300	**0.979**	**0.997**	**0.988**	0.930	0.946	0.938	0.997	0.979	0.987	0.831	0.920	0.866
C02310	**1.000**	**1.000**	**1.000**	0.996	0.965	0.980	**1.000**	**1.000**	**1.000**	0.901	0.849	0.873
C03000	0.919	0.914	0.916	**0.926**	**0.920**	**0.923**	0.926	0.914	0.919	0.914	0.906	0.909
C04000	**0.986**	**0.987**	**0.987**	0.963	0.945	0.953	0.946	0.954	0.950	0.861	0.879	0.869
C04100	0.975	0.975	0.975	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.807	0.516	0.411
C04200	0.974	0.974	0.974	**0.981**	**0.979**	**0.980**	0.968	0.966	0.967	0.960	0.948	0.953
C04300	**0.984**	**0.996**	**0.990**	0.984	0.934	0.956	0.801	0.754	0.773	0.924	0.650	0.690
Avg(C)	**0.937**	**0.933**	**0.932**	0.910	0.889	0.889	0.887	0.874	0.872	0.783	0.752	0.740

* The bold numbers in the table indicate that in the horizontal direction, the performance of this classifier using this model could achieve the best.

4.6. Experiment II: Performance of Hierarchical Attention Mechanism

The experiment in this section aims to compare the performance of various hierarchical attention mechanisms based on semantic features extracted from XLNet pretraining models. The HA, FT, and Bi-LSTM + HA selected in Experiment II are hierarchical attention mechanisms. These three structures aim to solve the problem of mapping char-level embeddings of the text to document-level embeddings. Unlike HA, FT uses maximum pooling to address the mapping from char-level embedding to segment-level embeddings and finally to document-level embeddings. Bi-LSTM + HA, on the other hand, adds a bidirectional LSTM (Bi-LSTM) on top of the lower-order representation layer of HA and then maps segment-level embeddings to document-level embeddings using the higher-order representation layer of HA.

In Experiment II, we train 27 classifiers in the classifier tree using the XLNet-based HA, FT, and Bi-LSTM models and then obtain a total of 3 classifier trees and 81 classifiers, as well as the P, R, and F1 values of each classifier (see Table 4 for detailed results). Compared with the other two models, the XLNet + BiLSTM + HA + FC model has a significant advantage in various metrics, with the average values of P, R, and F1 reaching 95.6%, 95%, and 94.9%, respectively. Furthermore, from the micro perspective of each classifier, for the three metrics of P, R, and F1, the number and proportion of classifiers that used BiLSTM + HA and achieved optimal performance in the three models are significantly higher than those of the other two, namely 22 (81%), 21 (78%), and 21 (78%), as shown in Figure 8.

Table 4. The performance of the hierarchical attention mechanism.

Classifier No.	XLNet + HA + FC			XLNet + FT + FC			XLNet + BiLSTM + HA + FC		
	P	R	F1	P	R	F1	P	R	F1
C00000	0.970	**0.967** *	**0.969**	0.844	0.819	0.827	**0.971**	0.966	0.968
C01000	0.956	0.934	0.944	0.875	0.792	0.826	**0.961**	**0.946**	**0.953**
C01100	0.977	0.990	0.983	0.919	0.868	0.889	**0.988**	**0.995**	**0.992**
C01110	**0.974**	**0.892**	**0.926**	0.804	0.452	0.527	0.949	0.890	0.915
C01120	0.971	0.989	0.979	0.567	0.473	0.476	**1.000**	**1.000**	**1.000**
C01200	0.955	**0.952**	**0.953**	0.870	0.802	0.828	**0.964**	0.936	0.949
C01300	0.826	0.816	0.821	0.683	0.684	0.684	**0.882**	**0.882**	**0.882**
C01310	0.962	0.972	0.966	0.737	0.638	0.629	**0.980**	**0.986**	**0.983**
C01320	**0.953**	0.833	0.863	0.320	0.283	0.239	0.950	**0.926**	**0.924**
C01400	0.809	0.830	0.819	0.723	0.592	0.597	**0.846**	**0.922**	**0.867**
C01410	**0.802**	**0.844**	**0.808**	0.605	0.585	0.552	0.740	0.797	0.752
C01420	0.969	0.917	0.933	0.162	0.250	0.197	**1.000**	**1.000**	**1.000**
C01500	0.780	0.772	0.775	0.669	0.654	0.659	**0.847**	**0.814**	**0.826**
C01510	0.977	0.970	0.974	0.797	0.801	0.799	**1.000**	**1.000**	**1.000**
C01600	**0.978**	**0.941**	**0.958**	0.865	0.530	0.477	**0.978**	**0.941**	**0.958**
C02000	0.989	0.982	0.986	0.885	0.845	0.859	**0.994**	**0.987**	**0.990**
C02100	0.996	0.996	0.996	0.817	0.887	0.839	**1.000**	**0.998**	**0.999**
C02110	0.986	**0.980**	0.983	0.862	0.653	0.693	**0.990**	**0.980**	**0.985**
C02120	**0.995**	**0.997**	**0.996**	0.698	0.715	0.676	0.990	0.995	0.992
C02200	0.670	0.784	0.704	0.371	0.347	0.312	**0.900**	**0.833**	**0.813**
C02300	**0.979**	**0.997**	**0.988**	0.577	0.635	0.420	**0.979**	**0.997**	**0.988**
C02310	**1.000**	**1.000**	**1.000**	0.555	0.541	0.173	0.996	0.965	0.980
C03000	0.919	0.914	0.916	0.805	0.692	0.685	**0.924**	**0.932**	**0.928**
C04000	0.986	0.987	0.987	0.808	0.812	0.810	**0.995**	**0.994**	**0.994**
C04100	0.975	0.975	0.975	0.666	0.638	0.639	**1.000**	**1.000**	**1.000**
C04200	0.974	0.974	0.974	0.900	0.886	0.892	**0.987**	**0.987**	**0.987**
C04300	0.984	0.996	0.990	0.497	0.499	0.479	**0.996**	**0.984**	**0.990**
Avg(C)	0.937	0.933	0.932	0.699	0.643	0.618	**0.956**	**0.950**	**0.949**

* The bold numbers in the table indicate that in the horizontal direction, the performance of this classifier using this model could achieve the best.

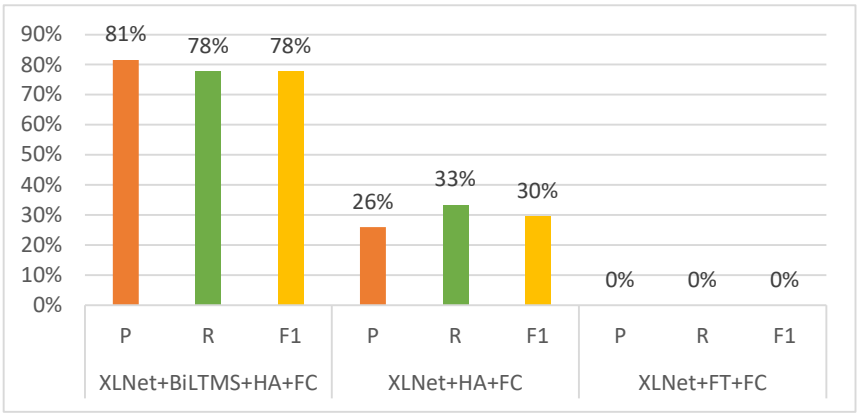

Figure 8. Comparison of training advantages of HA, FT and BiLSTM + HA (design concept is identical to Figure 7).

Moreover, the results show that BiLSTM + HA was 1.8% and 25.1% higher than the other two models for the P values, 1.7% and 30.7% higher for the R values, and 1.7% and 33.1% higher for the F1 values, respectively. Therefore, for RQ2 "Which types of HAs have the best performance in learning document-level embeddings?", we can conclude that the training results of the XLNet + BiLSTM + HA + FC model in the classifier tree are significantly better than those of the other two models.

4.7. Experiment III: The Availability of Transfer Learning Method Based on HAs

Experiment III is designed to verify whether transfer learning between classifiers in a tree could improve the training performance of the classifier tree. Therefore, based on the results of the first two experiments, this experiment further modified the XLNet + HA + FC and XLNet + BiLSTM + HA + FC models. Therefore, when training the classifier tree to obtain each node in sequence, the current classifier can use the HA parameters of its parent node classifier (the model that has completed the training and has the best accuracy in the validation set) as the initial values of its own HA parameters.

In Experiment III, we train all classifiers in the classifier tree using XLNet + HA + FC and XLNet + BiLSTM + HA + FC models. We then obtain 2 classifier trees and 54 classifiers for each classifier's P, R, and F1 values. Simultaneously, we directly reuse the training results of model XLNet + HA + FC in Experiment I and the results of model XLNet + BiLSTM + HA + FC in Experiment II. Comparing the results shown in Table 5, the transfer learning of the nodes in the classifier tree can improve the classifier classification effect from the perspective of P, R, and F1. The XLNet + BiLSTM + HA + FC + TL model has significant performance advantages in various metrics, with the average values of P, R, and F1 reaching 96.5%, 96.1%, and 96%, respectively.

Table 5. The availability of transfer learning method based on HAs.

Classifier No.	XLNet + HA + FC + TL			XLNet + BiLSTM + HA + FC			XLNet + BiLSTM + HA + FC + TL		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
C00000	0.970	0.967	0.969	0.971	0.966	0.968	0.971	0.966	0.968
C01000	0.956	0.934	0.944	0.961	0.946	0.953	0.961	0.946	0.953
C01100	0.984	0.993	0.989	0.988	0.995	0.992	**0.995**	**0.995**	**0.995**
C01110	0.898	0.912	0.896	**0.949** *	0.890	0.915	0.937	**0.925**	**0.928**
C01120	0.989	0.969	0.978	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
C01200	0.956	0.945	0.950	**0.964**	0.936	0.949	0.953	**0.948**	**0.951**
C01300	0.929	0.923	0.926	0.882	0.882	0.882	**0.938**	**0.929**	**0.933**
C01310	0.962	0.972	0.966	**0.980**	**0.986**	**0.983**	**0.980**	**0.986**	**0.983**
C01320	0.967	0.948	0.954	0.950	0.926	0.924	**0.968**	**0.931**	**0.943**
C01400	0.915	0.943	0.928	0.846	0.922	0.867	**0.909**	**0.961**	**0.930**
C01410	0.841	0.785	0.804	0.740	0.797	0.752	**0.764**	**0.789**	**0.769**
C01420	0.550	0.727	0.611	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
C01500	0.880	0.875	0.877	0.847	0.814	0.826	**0.940**	**0.916**	**0.926**
C01510	1.000	1.000	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
C01600	0.978	0.941	0.958	**0.978**	**0.941**	**0.958**	**0.978**	**0.941**	**0.958**
C02000	0.989	0.982	0.986	0.994	0.987	0.990	0.994	0.987	0.990
C02100	0.996	0.992	0.994	**1.000**	**0.998**	**0.999**	**1.000**	**0.998**	**0.999**
C02110	0.984	0.973	0.978	**0.990**	**0.980**	**0.985**	0.987	0.979	0.983
C02120	0.992	0.992	0.992	0.990	0.995	0.992	**1.000**	**1.000**	**1.000**
C02200	0.859	0.829	0.781	**0.900**	**0.833**	**0.813**	0.885	0.832	0.804
C02300	1.000	1.000	1.000	0.979	0.997	0.988	**1.000**	**1.000**	**1.000**
C02310	1.000	1.000	1.000	0.996	0.965	0.980	**1.000**	**1.000**	**1.000**
C03000	0.919	0.914	0.916	0.924	0.932	0.928	0.924	0.932	0.928
C04000	0.986	0.987	0.987	0.995	0.994	0.994	0.995	0.994	0.994
C04100	1.000	1.000	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
C04200	0.995	0.992	0.993	0.987	0.987	0.987	**1.000**	**1.000**	**1.000**
C04300	0.857	0.882	0.869	**0.996**	0.984	0.990	0.984	**0.996**	**0.990**
Avg(C)	0.939	0.940	0.935	0.956	0.950	0.949	**0.965**	**0.961**	**0.960**

* The bold numbers in the table indicate that in the horizontal direction, the performance of this classifier using this model could achieve the best.

Note that 5 of the 27 classifiers in the classifier tree do not use transfer learning during the training. This is because (1) the classifier C00000 is the root node of the tree, and there is no parent node available to it for transfer learning and (2) the target categories of these four classifiers, C01000, C02000, C03000, and C04000, are not semantically related to the target categories of the root node classifier C00000, which results in the poor performance of the classifier using transfer learning. Therefore, these four classifiers do not use transfer

learning for training to avoid affecting the learning effect of subsequent nodes. The number of classifiers involved in the transfer learning is 22 instead of 27.

Furthermore, from the micro perspective of each classifier, for the three metrics of P, R, and F1, the number and proportion of classifiers that use TL and achieve optimal performance in the three models are significantly higher than those of the other two, namely 22 (81%), 21 (78%), and 21 (78%), as shown in Figure 9.

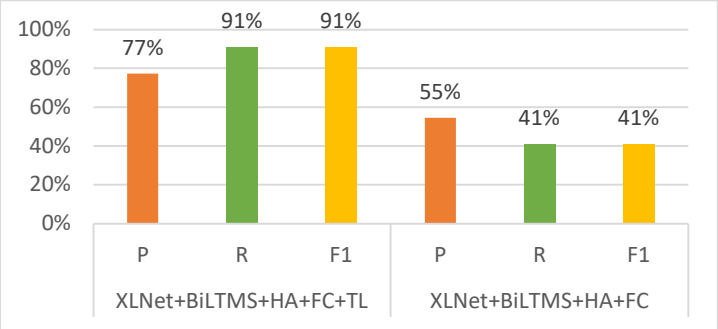

Figure 9. Comparison of training advantages of the model XLNet + BiLSTM + HA + FC using TL or not (design concept is identical to Figure 7).

Moreover, the results show that the XLNet + BiLSTM + HA + FC + TL model is 0.9% higher than that without TL for the P value, 1.1% higher for the R value, and 1% higher for the F1 value. Therefore, for RQ3 "Can HA-based transfer learning improve the training and testing performance of classifier trees?", we can conclude that the training results of the model with TL in the classifier tree perform better than those without TL.

4.8. Experiment IV: The Performance of Classifier Trees Based on Various Models

The three experiments conducted above aim to train each classifier in the classifier tree from different perspectives, enabling the identification of the best model and method for each direction. Experiment IV in this section, however, focuses on the classification scenario of PGT by treating the classifier tree, whose purpose is to evaluate various models and methods using the final classification results to determine the optimal one. This experiment is based on the results of the training stages of all classifiers obtained from the previous three experiments. It requires all 27 classifiers to be trained independently using the model as a unit to be assembled according to a multi-label tree, which associates labels with classifiers and converts the label tree into an instance of the classifier tree. The detailed process is shown in Algorithm 1.

Algorithm 1: The construction of the classifier tree

N # the number of classifiers in a classifier tree.
Input: The classifier information set of classifier tree $ClassifierTree$ is composed of classifiers instances set $Classifiers$ and the class–classifier relationship collection of all classifiers *RCs* (Note: All classifiers are named as classifier no. above).
Output: The root classifier $rootClassifier$ of the classifier tree contains the structure of a complete classifier tree instance, which can be used to calculate the leaf node labels for PGT.
1: *Classifiers = ClassifierTree.Classifiers*
2: *RCs = ClassifierTree. RCs*
3: *For j = 1 to N Do*
4: *classifier = Classifiers[j]*
5: *classifier.loadcheckpoint() # load model checkpoint from .tar file.*
6: *rc = RCs[classifier.name] # get class–classifier relationship for this classifier.*
7: *classifier.rc = rc # rc is a dictionary, the keys of which are classes of the classifier, and the values are the collection of the corresponding classifier no*
8: *Return rootClassifier = ClassifierTree.getClassifer ("C00000") # C00000 is the name of the root node classifier in the classifier tree.*

In the classifier tree, the test samples start from the root node classifier individually and turn to the corresponding classifier based on the classification results of each classifier they experience until they reach the leaf node of the label tree (this process is shown in Algorithm 2). Theoretically, in the prediction process of all test samples, a maximum of four classifiers are loaded for each test sample. However, the prediction of all samples requires the loading of 27 classifiers. The memory required for each classifier's checkpoint after loading is 632 MB; therefore, the total memory required for the testing program is approximately 17 GB, much larger than the graphics memory size used in this experiment. We preload all models into memory during the experimental process to solve this problem and wait until a specific classifier is needed for prediction before loading the classifier and data into the graphics memory. After use, we promptly migrate them to memory. In this manner, the time cost of the model's spatial migration in graphics and memory is exchanged for the computational performance of the classifier model in graphics, and the average prediction time for each sample is reduced to 0.8 s.

Algorithm 2: The prediction of the classifier tree for PGT

M **#** the number of examples.
Input: *rootClassifier* is the root node of the classifier tree; P is the set of examples; L is the set of labels.
Output: Y is the leaf label predicted.
1: *For i = 1 to M Do*
2: *p = P[i]*
3: *l = L[i]*
4: *y = rootClassifier.predict(text = p, truelabel = l)*
5: *Y.append(y)*
6: *Return Y*

We use the above algorithm to construct the corresponding classifier trees for the eight models involved in this study and conduct classification performance tests using the same set of test samples. Various indicators of the test results are presented in Table 6. These metrics are divided into micro- and macro-average types. The former measures the classification performance of a classifier tree. By contrast, the latter measures the average classification performance of all classifiers on each label in the classifier tree.

Table 6. The performance of the classifier tree based on various models.

Model	Micro			Macro Avg.		
	P	**R**	**F1**	**P**	**R**	**F1**
XLNet + BiLSTM + HA + FC + TL	0.961	0.961	0.961	0.946	0.920	0.929
XLNet + BiLSTM + HA + FC	0.957	0.957	0.957	0.939	0.914	0.919
XLNet + HA + FC + TL	0.882	0.882	0.882	0.778	0.836	0.789
XLNet + HA + FC	0.851	0.851	0.851	0.788	0.817	0.766
WWM + HA + FC	0.857	0.857	0.857	0.737	0.693	0.692
BERT + HA + FC	0.825	0.825	0.825	0.741	0.727	0.705
ERNIE3.0 + HA + FC	0.673	0.673	0.673	0.535	0.491	0.467
XLNet + FT + FC	0.580	0.580	0.580	0.392	0.346	0.340

The results in Table 6 indicate that the proposed model "XLNet + BiLSTM + HA + FC + TL" achieved the best classification performance for all metrics. The conclusions from the first three experiments are obtained and verified. Clearly, (1) XLNet performs significantly better than other pretrained models in extracting semantic embeddings, and (2) BiLSTM + HA has significantly improved performance compared to HA. At the same time, (3) models using TL can slightly improve the classification performance on their original basis, and this degree of improvement weakens the overall performance improvement.

5. Conclusions

To solve the hierarchical multi-label classification problem of PGTs, this study proposes an HTMC-PGT framework. The most prominent feature of this framework is transforming the traditional HMTC problem into a parameter solving problem for various multi-class classifiers in a classifier tree. In this way, the HTMC-PGT framework brings more flexibility to problem solving. In particular, (1) gradually training classifiers based on the classifier tree structure can effectively solve the problems of sample scarcity and imbalance in leaf node labels on hierarchical label trees. In addition, to improve the overall performance of the classifier tree, relevant experiments are conducted on the pretrained extraction model of character semantic embedding, refining the model of document-level semantic embedding and transfer learning between the parent and child classifiers in the classifier tree. It is found that (2) XLNet performs better than other pretrained models such as BERT in HTMC-PGT; (3) BiLSTM + HA can extract document-level embedding vectors that are more conducive to text classification; and (4) on the whole classifier tree scale, the transfer learning (TL) between parent and child node classifiers can increase the overall classification performance of the classifier tree to a certain extent. The above conclusions obtained during the training phase of independent learning of each classifier are equally valid during the testing phase of the classifier tree as a whole experiment. Moreover, the micro-precision, recall, and F1 values of XLNet + BiLSTM + HA + FC + (TL) proposed in this paper can reach values over 95%, 7.5%, and 38.1% higher than the other baseline models used in this study. The method proposed in this study has some shortcomings. First, the time required for extracting char-level semantic embeddings using XLNet is much longer than that required for models such as BERT. However, by optimizing the program code, the experimental results (1 s per example) in the final testing stage can also satisfy practical requirements. Second, the proposed transfer learning between classifiers requires a semantic parent–child inclusion relationship between classifiers. Only in this way can TL be used to improve the training performance of the sub-classifiers. Overall, the HTMC-PGT framework proposed in this study can better meet the requirements of solving the hierarchical multi-label classification problem of PGT with various evaluation indicators.

Author Contributions: Conceptualization, X.W. and L.G.; methodology, X.W.; software, X.W.; validation, L.G.; formal analysis, X.W. and L.G.; data preparation, X.W.; writing—original draft preparation, X.W.; writing—review and editing, X.W. and L.G.; funding acquisition, L.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China (2021ZD0110901) and the Science and Technology Innovation Project of the Chinese Academy of Agricultural Sciences, grant number CAAS-ASTIP-2023-AII.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available on request from the authors.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wahl, B.; Cossy-Gantner, A.; Germann, S.; Schwalbe, N.R. Artificial intelligence (AI) and global health: How can AI contribute to health in resource-poor settings? *BMJ Glob. Health* **2018**, *3*, e000798. [CrossRef] [PubMed]
2. Hall, O.; Ohlsson, M.; Rognvaldsson, T. A review of explainable AI in the satellite data, deep machine learning, and human poverty domain. *Patterns* **2022**, *3*, 100600. [CrossRef] [PubMed]
3. Usmanova, A.; Aziz, A.; Rakhmonov, D.; Osamy, W. Utilities of Artificial Intelligence in Poverty Prediction: A Review. *Sustainability* **2022**, *14*, 14238. [CrossRef]
4. Wang, X.; Wang, W.; Guo, L. Mechanism and Path to Improve Digital Governance Capacity for Prevention of Relapse into Poverty Based on the Integration of New IT. *J. Huazhong Agric. Univ. Soc. Sci. Ed.* **2023**, *1*, 58–70.
5. Loftis, M.W.; Mortensen, P.B. Collaborating with the Machines: A Hybrid Method for Classifying Policy Documents. *Policy Stud. J.* **2020**, *48*, 184–206. [CrossRef]

6. Zhao, F.; Li, P.; Li, Y.; Hou, J.; Li, Y. Semi-supervised convolutional neural network for law advice online. *Appl. Sci.* **2019**, *9*, 3617. [CrossRef]
7. Sajid, N.A.; Rahman, A.; Ahmad, M.; Musleh, D.; Basheer Ahmed, M.I.; Alassaf, R.; Chabani, S.; Ahmed, M.S.; Salam, A.A.; AlKhulaifi, D. Single vs. Multi-Label: The Issues, Challenges and Insights of Contemporary Classification Schemes. *Appl. Sci.* **2023**, *13*, 6804. [CrossRef]
8. Erlich, A.; Dantas, S.G.; Bagozzi, B.E.; Berliner, D.; Palmer-Rubin, B. Multi-Label Prediction for Political Text-as-Data. *Polit. Anal.* **2022**, *30*, 463–480. [CrossRef]
9. Maltoudoglou, L.; Paisios, A.; Lenc, L.; Martinek, J.; Kral, P.; Papadopoulos, H. Well-calibrated confidence measures for multi-label text classification with a large number of labels. *Pattern Recognit.* **2022**, *122*, 21. [CrossRef]
10. Bennett, P.N.; Nguyen, N. Refined experts: Improving classification in large taxonomies. In Proceedings of the 32nd International ACM SIGIR Conference on RESEARCH and Development in Information Retrieval, Boston, MA, USA, 19–23 July 2009; Association for Computing Machinery: Boston, MA, USA, 2009; pp. 11–18.
11. Traore, Y.; Bassole, D.; Malo, S.; Sere, A. Multi-Label Classification using an Ontology. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 472–476. [CrossRef]
12. McCallum, A.; Ng, A. Improving text classification by shrinkage in a hierarchy of classes. In Proceedings of the 15th International Conference on Machine Learning, Madison, WI, USA, 24–27 July 1998; pp. 359–367.
13. Du, J.C.; Chen, Q.Y.; Peng, Y.F.; Xiang, Y.; Tao, C.; Lu, Z.Y. ML-Net: Multi-label classification of biomedical texts with deep neural networks. *J. Am. Med. Inf. Assoc.* **2019**, *26*, 1279–1285. [CrossRef]
14. Ding, J.Q.; Li, B.; Xu, C.; Qiao, Y.; Zhang, L.X. Diagnosing crop diseases based on domain-adaptive pre-training BERT of electronic medical records. *Appl. Intell.* **2022**, *53*, 15979–15992. [CrossRef]
15. Cui, Y.M.; Che, W.X.; Liu, T.; Qin, B.; Yang, Z.Q. Pre-Training With Whole Word Masking for Chinese BERT. *IEEE-Acm Trans. Audio Speech Lang. Process.* **2021**, *29*, 3504–3514. [CrossRef]
16. Sun, Y.; Wang, S.; Feng, S.; Ding, S.; Pang, C.; Shang, J.; Liu, J.; Chen, X.; Zhao, Y.; Lu, Y.; et al. ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation. *arXiv* **2021**, arXiv:2107.02137.
17. Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q.V.; Salakhutdinov, R. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *arXiv* **2019**, arXiv:arXiv:1901.02860.
18. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; Le, Q.V. Quoc, XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv* **2020**, arXiv:1906.08237.
19. Topal, M.O.; Bas, A.; van Heerden, I. Exploring Transformers in Natural Language Generation: GPT, BERT, and XLNet. *arXiv* **2021**, arXiv:2102.08036.
20. Liang, Y.J.; Li, H.H.; Guo, B.; Yu, Z.W.; Zheng, X.L.; Samtani, S.; Zeng, D.D. Fusion of heterogeneous attention mechanisms in multi-view convolutional neural network for text classification. *Inf. Sci.* **2021**, *548*, 295–312. [CrossRef]
21. Liang, D.; Yi, B. Two-stage three-way enhanced technique for ensemble learning in inclusive policy text classification. *Inf. Sci.* **2021**, *547*, 271–288. [CrossRef]
22. Li, X.; Yao, Y.; Zhu, M. The Temporal Spatial Dynamic of Land Policy in China: Evidence from Policy Analysis Based on Machine Learning. *Math. Probl. Eng.* **2022**, *2022*, 8500272. [CrossRef]
23. Liu, Y.S.; Guo, Y.Z.; Zhou, Y. Poverty alleviation in rural China: Policy changes, future challenges and policy implications. *China Agric. Econ. Rev.* **2018**, *10*, 241–259. [CrossRef]
24. Wehrmann, J.; Cerri, R.; Barros, R. Hierarchical multi-label classification networks. In *International Conference on Machine Learning*; PMLR: New York, NY, USA, 2018; pp. 5075–5084.
25. Wu, F.; Zhang, J.; Honavar, V. Learning classifiers using hierarchically structured class taxonomies. In Proceedings of the Abstraction, Reformulation and Approximation: 6th International Symposium, SARA 2005, Airth Castle, Scotland, UK, 26–29 July 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 313–320.
26. Ali, T.; Asghar, S. Efficient label ordering for improving multi-label classifier chain accuracy. *J. Natl. Sci. Found. Sri Lanka* **2019**, *47*, 175–184. [CrossRef]
27. Silla, C.N.; Freitas, A.A. A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.* **2011**, *22*, 31–72. [CrossRef]
28. Huang, W.; Chen, E.; Liu, Q.; Chen, Y.; Huang, Z.; Liu, Y.; Zhao, Z.; Zhang, D.; Wang, S. Hierarchical Multi-label Text Classification: An Attention-based Recurrent Network Approach. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; Association for Computing Machinery: Beijing, China, 2019; pp. 1051–1060.
29. Cerri, R.; Barros, R.C.; PLF de Carvalho, A.C.; Jin, Y. Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC Bioinform.* **2016**, *17*, 373. [CrossRef]
30. Zhu, K.F.; Ma, B.S.; Huang, T.H.; Li, Z.Q.; Ma, H.Y.; Li, Y.J. Sequence Generation Network Based on Hierarchical Attention for Multi-Charge Prediction. *IEEE Access* **2020**, *8*, 109315–109324. [CrossRef]
31. Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; Hovy, E. Hierarchical attention networks for document classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016; pp. 1480–1489.

32. Abuqaddom, I.; Mahafzah, B.A.; Faris, H. Oriented stochastic loss descent algorithm to train very deep multi-layer neural networks without vanishing gradients. *Knowl.-Based Syst.* **2021**, *230*, 107391. [CrossRef]
33. Deng, J.F.; Cheng, L.L.; Wang, Z.W. Attention-based BiLSTM fused CNN with gating mechanism model for Chinese long text classi fi cation. *Comput. Speech Lang.* **2021**, *68*, 12. [CrossRef]
34. Pan, S.J.; Yang, Q.A. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
35. Yan, R.E.; Jiang, X.; Dang, D.P. Named Entity Recognition by Using XLNet-BiLSTM-CRF. *Neural Process. Lett.* **2021**, *53*, 3339–3356. [CrossRef]
36. Pinto, G.; Messina, R.; Li, H.; Hong, T.Z.; Piscitelli, M.S.; Capozzoli, A. Sharing is caring: An extensive analysis of parameter-based transfer learning for the prediction of building thermal dynamics. *Energy Build.* **2022**, *276*, 112530. [CrossRef]
37. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
38. Bin, L. GCN-BERT and Memory Network Based Multi-Label Classification for Event Text of the Chinese Government Hotline. *IEEE Access* **2022**, *10*, 109267–109276.
39. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
40. Sun, Z.; Li, X.; Sun, X.; Meng, Y.; Ao, X.; He, Q.; Wu, F.; Li, J. Chinesebert: Chinese pretraining enhanced by glyph and pinyin information. *arXiv* **2021**, arXiv:2106.16038.
41. Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of tricks for efficient text classification. *arXiv* **2016**, arXiv:1607.01759.
42. Borges, H.B.; Nievola, J.C. Multi-label hierarchical classification using a competitive neural network for protein function prediction. In Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, 10–15 June 2012; IEEE: New York, NY, USA, 2012; pp. 1–8.
43. Vens, C.; Struyf, J.; Schietgat, L.; Džeroski, S.; Blockeel, H. Decision trees for hierarchical multi-label classification. *Mach. Learn.* **2008**, *73*, 185–214. [CrossRef]