

Article

CPS-LSTM: Privacy-Sensitive Entity Adaptive Recognition Model for Power Systems

Hao Zhang ¹, Jing Wang ², Xuanyuan Wang ¹, Xuhui Lü ², Zhenzhi Guan ³, Zhenghua Cai ^{3,*} and Hua Zhang ^{3,*} 

¹ State Grid Jibei Electric Power Company Limited, Beijing 100054, China; ab123h@126.com (H.Z.); wang_xuanyuan_1979@126.com (X.W.)

² Beijing Kedong Electric Power Control System Co., Ltd., Beijing 100192, China; wangjing19891110@126.com (J.W.); xhuiii@foxmail.com (X.L.)

³ State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; a550279039@gmail.com

* Correspondence: zhenghua.cai@bupt.edu.cn (Z.C.); zhanghua_288@bupt.edu.cn (H.Z.)

Abstract: With the widespread application of Android devices in the energy sector, an increasing number of applications rely on SDKs to access privacy-sensitive data, such as device identifiers, location information, energy consumption, and user behavior. However, these data are often stored in different formats and naming conventions, which poses challenges for consistent extraction and identification. Traditional taint analysis methods are inefficient in identifying these entities, hindering the realization of accurate identification. To address this issue, we first propose a high-quality data construction method based on privacy protocols, which includes sentence segmentation, compression encoding, and entity annotation. We then introduce CPS-LSTM (Character-level Privacy-sensitive Entity Adaptive Recognition Model), which enhances the recognition capability of privacy-sensitive entities in mixed Chinese and English text through character-level embedding and word vector fusion. The model features a streamlined architecture, accelerating convergence and enabling parallel sentence processing. Our experimental results demonstrate that CPS-LSTM significantly outperforms the baseline methods in terms of accuracy and recall. The accuracy of CPS-LSTM is 0.09 higher than Lattice LSTM, 0.14 higher than WC-LSTM, and 0.05 higher than FLAT. In terms of recall, CPS-LSTM is 0.07 higher than Lattice LSTM, 0.12 higher than WC-LSTM, and 0.02 higher than FLAT.



Academic Editor: Ramesh Rayudu

Received: 13 February 2025

Revised: 24 March 2025

Accepted: 11 April 2025

Published: 14 April 2025

Citation: Zhang, H.; Wang, J.; Wang, X.; Lü, X.; Guan, Z.; Cai, Z.; Zhang, H. CPS-LSTM: Privacy-Sensitive Entity Adaptive Recognition Model for Power Systems. *Energies* **2025**, *18*, 2013. <https://doi.org/10.3390/en18082013>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: computer application technology; name entity recognition; privacy data; vocabulary enhancement

1. Introduction

In the power system, a large number of device terminals, such as cameras, temperature and humidity sensors, voltage and current sensors, are widely deployed. These devices collect and transmit real-time data to support power monitoring, dispatching, and operation management [1]. However, with the deepening application of information technology in the power sector, especially the continuous development of technologies like the Internet of Things (IoT), big data analytics, and cloud computing, the intelligence and automation levels of power systems are steadily increasing. To meet the real-time, stability, and intelligent analysis requirements of power monitoring centers, industrial Android all-in-one machines have gradually gained widespread use in these centers. These Android all-in-one machines [2] typically integrate advanced technologies such as the Android operating system, high-performance processors, and touch-screen displays, which not only

enable centralized management and data processing of device terminals but also provide a more intuitive and efficient user experience through the touch interface. Through the openness and strong compatibility of the Android operating system, these devices can flexibly support various power monitoring applications and data interaction with other devices. In this process, the demand for privacy and data security in the power system has become increasingly important. Especially with the large-scale data collection from sensors and monitoring devices, ensuring the security of user data and transmitted data within the power system has become a critical issue [3].

Android all-in-one manufacturers accompany the production of devices with corresponding privacy agreements, declaring the user data collected and used internally by the device to assist regulatory authorities in auditing, thus addressing users' concerns about the security of privacy data. These privacy agreements are written in natural language rather than formatted text, making it impossible to use rule-based matching for identification tasks. Currently, the academic community generally uses natural language processing (NLP) technologies to identify sensitive behaviors in privacy agreements, detecting privacy leakage or misuse behaviors. In studies on identifying sensitive behaviors in privacy agreements based on NLP [4], it is generally necessary to first determine the structural features of privacy-sensitive entities, and then analyze their relationships, further mining the privacy leakage paths. Therefore, identifying potential privacy-sensitive entities in sensitive agreements is crucial. The privacy-sensitive entities in Android privacy protocols consist of system-level and platform-level entities. System-level entities refer to user data stored by the system, which are typically well-defined and easy to identify. Platform-level entities refer to user data stored by applications (e.g., device identifiers, location data, power consumption). There are two main challenges in these privacy protocols: first, platform-specific entities (like "AAID" and "OID") lack standardized naming across manufacturers; second, data sparsity occurs, where simple sentence segmentation results in over 40% of sentences exceeding 300 characters, leading to inefficiencies in model training. Additionally, existing named entity recognition (NER) methods, such as Lattice LSTM and FLAT, despite achieving F1 scores of 82–89% on general Chinese text, struggle with mixed-language expressions in technical documents, lowering NER accuracy.

To address these challenges, this paper proposes a high-quality privacy protocol data construction method to fill the gap in Android privacy-sensitive datasets for power equipment. It also introduces an innovative model for the adaptive recognition of sensitive entities in power equipment, which integrates word vectors and character representations. This fusion significantly enhances character boundary perception by incorporating lexical information into character embeddings. The main contributions of this paper are as follows:

1. We propose a method to build a sensitive entity recognition dataset in three steps: sentence segmentation, compression encoding, and entity annotation. First, privacy protocol text is segmented into sentences of about 100 characters. Next, a character encoding mapping compresses the character space. Finally, entities are annotated using the BIOES method, with annotations fixed to 100 dimensions. The sentence and entity vectors are combined and stored in binary format, providing an efficient and compact dataset for model training.
2. We introduce the CPS-LSTM model for privacy-sensitive entity recognition. The model first receives compressed sentence vectors and uses a BiLSTM network to extract contextual features. Privacy-sensitive entities are identified by matching characters with a predefined word library, and lexical information is enhanced with multi-head attention. These enhanced vectors are expanded to include semantic information through BiLSTM and passed through a fully connected layer for character-level

label prediction. The final output is decoded using a CRF layer for optimization. This approach achieves accurate entity recognition.

3. Experimental results show that the proposed dataset construction method generates data samples with at least 85% accuracy, making it effective for generating high-quality data in various privacy protocol scenarios. The CPS-LSTM model outperforms baseline methods in both accuracy and recall, with accuracy improvements of 0.09 over Lattice LSTM, 0.14 over WC-LSTM, and 0.05 over FLAT. In terms of recall, CPS-LSTM outperforms Lattice LSTM by 0.07, WC-LSTM by 0.12, and FLAT by 0.02. These results demonstrate CPS-LSTM's effectiveness in accurately identifying privacy-sensitive entities in new texts.

The structure of this paper is as follows: Section 1 introduces the research background and objectives, providing a brief overview of the study's content and key contributions. Section 2 reviews the existing literature on named entity recognition (NER) and privacy-sensitive entity detection, with a focus on key studies and their limitations. Section 3 discusses the necessity and challenges of constructing datasets for sensitive entity recognition. Section 4 proposes a method for constructing such datasets. Section 5 presents the CPS-LSTM model, a character-based adaptive model for privacy-sensitive entity recognition, aimed at improving the accuracy of named entity recognition in the power industry. Section 6 provides a detailed explanation of the experimental setup, evaluation metrics, and results from comparative and ablation studies, validating the effectiveness of the proposed approach. Finally, Section 7 summarizes the key findings of the paper, discusses its limitations, and outlines potential directions for future research.

2. Related Works

In the task of Chinese named entity recognition (NER) for Android all-in-one devices, the target entities are essentially strings, and since there are no explicit delimiters between characters in Chinese sentences, word segmentation tools are typically used to tokenize the sentences before entity recognition. In this pipeline structure, the accuracy of the word segmentation tool directly impacts the entity recognition accuracy. Currently, accurate cross-domain word segmentation remains an unresolved issue. Even with the most advanced segmentation tools [4], precise segmentation of sentences from different domains is still not achievable. Therefore, Chinese named entity recognition tasks often employ character-level prediction approaches. However, these prediction methods fail to effectively leverage the latent relationships between characters and words, leading to errors in predictions, especially when identifying the boundary characters of sensitive entities, as the word-level information is not available during the prediction process.

In 2015, Huang et al. [5] proposed LSTM-CRF, a model that combines long short-term memory networks (LSTM) and conditional random fields (CRF) to perform well in natural language processing tasks, especially named entity recognition (NER) tasks. In 2018, Zhang et al. [6] proposed the Lattice LSTM model, which was the first to fuse word-level information with characters. Since words contain rich prior knowledge, this fusion provides more boundary information for entities, and the model demonstrates good entity recognition capabilities in Chinese datasets. However, in this model, only the last character of a word can access word-level information, causing context characters to not fully benefit from the word-level knowledge. To address the low utilization of word-level information, Sui et al. [7] proposed a Chinese named entity recognition model, CGN, which utilizes a collaborative graph network to integrate word knowledge. The model builds a relationship graph between words to jointly analyze the context, incorporating word-level information into each character, which leads to more effective utilization of word boundary information. However, during training, this model often encounters issues like gradi-

ent vanishing and explosion. To tackle this issue, Li et al. [8] proposed the FLAT model, which employs the Transformer architecture, known for its excellent stability, to capture long-distance dependencies in sentence features. Furthermore, by incorporating relative position encodings in the LSTM network based on the lattice grid structure, the model further improves entity recognition. However, the Transformer architecture requires substantial computational resources, making the training and inference process of the FLAT model highly time-consuming.

Despite the advancements in traditional named entity recognition (NER) models, existing methods still struggle with the unique challenges posed by power equipment, especially in the context of privacy-sensitive entities. The complexity of cross-lingual text and the stringent operational constraints in power systems further highlight the limitations of general NER approaches. For instance, the mix of Chinese and English expressions in power equipment Android all-in-one machine protocols, such as “OpenID configuration exception” and “AAID collection cycle”, presents difficulties for traditional models to effectively recognize entities. Additionally, the need for fast inference speed and low memory usage, typically requiring less than 50 milliseconds per instance and memory consumption below 2 GB, makes it challenging for deep learning-based models like Lattice LSTM and FLAT to perform optimally.

In the context of privacy data leakage in power systems, existing research has proposed several solutions. Platform-level privacy leakage in the power industry refers to the malicious leakage of user privacy data held by third-party power platforms or service providers. Due to the diverse nature of data interfaces and the lack of standardized naming conventions in power devices and applications, automated extraction of privacy data becomes extremely difficult. Currently, research on privacy leakage in power systems is limited, but both domestic and international studies have been conducted on text analysis of Android privacy policies. For example, Harkous et al. [9] proposed the Poisis method, which transforms fine-grained privacy data phrases and sharing policies into a sequence labeling problem, solved using an entity recognition model. However, Poisis heavily depends on the format of privacy policy clauses and cannot handle the varied interface behavior descriptions in power systems. Yu et al. [10] proposed the PPChecker tool, which extracts privacy data sharing policies from privacy agreements using syntactic analysis and combines it with taint analysis to detect privacy leakage behaviors. Although this method can accurately assess the content and intent of privacy agreements, it still cannot effectively perform consistency checks between privacy policies and the actual conditions in power systems. Wang et al. [11] proposed the XFinder framework, which combines machine learning and natural language processing techniques to identify privacy data items in SDK service agreements to detect privacy data theft behaviors. However, this tool only studies a small number of third-party component libraries and still requires manual parsing of numerous privacy-sensitive data interfaces in third-party component libraries, resulting in significant labor consumption when processing large-scale documents.

In summary, while existing research provides some effective solutions for named entity recognition and privacy leakage detection in power systems, further improvement in model flexibility and efficiency is required when facing the challenges of cross-domain applications, particularly in the power device field. To address these challenges, this paper proposes an adaptive Chinese privacy policy entity recognition model specifically designed for Android all-in-one devices in the power equipment domain. By leveraging word-level information, the proposed model not only enhances entity boundary recognition accuracy but also meets the operational constraints by achieving lower memory consumption and faster model training times, especially when processing large datasets.

3. Problem Analysis

This paper analyzes the privacy data declarations in privacy policies across different platforms for Android all-in-one devices and concludes that user privacy data possesses specific attributes. By examining the challenges arising from the lack of privacy-sensitive datasets and the issue of overly dispersed clause length distributions in naive sentence segmentation methods, the paper emphasizes the need for a targeted approach to construct a privacy-sensitive dataset for Android all-in-one devices.

3.1. Analysis of the Specificity of Privacy-Sensitive Entities

In Android all-in-one devices, applications typically operate in collaboration with multiple SDKs alongside the main business code. User privacy data are often retrieved through these SDKs. For instance, data such as device ID, electricity usage, address, and device status can be indirectly accessed from the operating system through data retrieval interfaces provided by the Android SDK. In addition, privacy data specific to certain platforms can be obtained through SDK libraries provided by the manufacturers of power equipment. However, the user privacy data from these specific platforms varies in format and lacks a unified and clear naming convention [12]. As shown in Figure 1, different platforms contain different privacy-sensitive entities. For example, on Platform A, the privacy-sensitive entities include Moments, OpenID, etc., while on Platform B, the privacy-sensitive entities include AAID, Push Token, OID, and GUID, among others. In power equipment scenarios, user data such as location, electricity usage patterns, and the relationships between devices are involved. When these data are combined with privacy-sensitive entities like social media Moments or OpenID, it may expose personal aspects of the user's life and behavioral patterns, increasing the risk of privacy breaches.

Platform A Privacy Agreement

1. When users forward posts to their Moments, the platform will store the user's **Moments information** to ensure the proper functioning of the Moments feature. When users delete their Moments information, the platform will delete the corresponding data.
2. When users use the login feature on websites or applications operated by third parties, this platform will provide the user's personal identifier information, **OpenID**, to the third-party entity. The specific scope of external sharing is based on what you confirm in the interaction page.

Platform B Privacy Agreement

1. To support pushing message content to designated users, we need to use the in-app device identifier **AAID** and **Push Token**.
2. When using our products and services, we will collect device information and details about how the user and the user's device interact with our products and services. This includes identifiers such as **OID**, **GUID**, and others.

Figure 1. Privacy-sensitive entities on different platforms.

Due to the platform-specific nature of user privacy data, a universal character-matching rule cannot be used to filter and extract interfaces related to such specific privacy data from the interface documentation.

3.2. Analysis of the Distribution of Privacy-Sensitive Data

Practical investigations reveal that these specific privacy-sensitive entity declarations are not only present in component interface documentation but are also widely found in the associated privacy agreements. However, the distribution of sentence lengths in the privacy agreements is excessively scattered, and a significant number of sentences are unrelated to privacy-sensitive data. This results in a large number of irrelevant data samples in the dataset, significantly reducing the training efficiency of the model. Moreover, because sentences are stored using variable-length encoding, each character in a sentence does not

have a fixed storage size. This requires additional consideration of the actual encoding of each character when constructing data samples, which further decreases the efficiency of sample construction and leads to a lower information density in the constructed data samples. In this paper, 200 privacy agreements are divided into four groups, and naive sentence segmentation is applied to automatically split the sentence content. The distribution of sentence lengths and the information density of the protocol content in each group are statistically analyzed, as shown in Table 1.

Table 1. Privacy agreement clause statistics.

Group Number	<100 Characters	100~200 Characters	200~300 Characters	>300 Characters	Original Content Length (Bytes)	Compressed Encoded Content Length (Bytes)
Group 1	20.10%	27.20%	12.20%	40.50%	2421	1820
Group 2	10.50%	42.60%	22.40%	24.40%	2955	2032
Group 3	38.20%	26.70%	14.40%	30.70%	3323	2096
Group 4	28.20%	34.30%	26.50%	11%	1922	1372

Based on the data in the table, it can be inferred that the sentence length distribution after applying the naive sentence delimiter segmentation is overly scattered, which may result in high bias or high variance during model training, preventing convergence to the optimal solution. Additionally, because sentences are stored using variable-length encoding, some of the storage space is utilized for saving international character encodings.

4. Construction of Sensitive Entity Recognition Dataset

To address issues such as sparse sentence length distribution and low information density in privacy policies of Android all-in-one machines, this paper proposes a dataset construction method aimed at improving the training performance of sensitive entity recognition models. The process involves several steps: sentence segmentation, compression encoding, and entity annotation.

- (1) **Sentence Segmentation:** the original privacy policy text is segmented using sentence delimiters to ensure that sentence lengths are concentrated within a specific range. Initially, common delimiters (such as periods, exclamation marks, and question marks) are used for segmentation. For sentences exceeding 100 characters, further segmentation is performed using commas, colons, and semicolons to ensure sentence length is close to 100 characters. For special sentences, Stanford CoreNLP is used for syntactic analysis to extract the longest sub-clause (as shown in Figure 2).
- (2) **Compression Encoding:** the segmented sentences are mapped into compact sentence vectors using short integers, preserving the core semantics. A variable-length character encoding scheme is employed to compress the data, and a character encoding mapping dictionary is constructed using the THUCNews dataset to enhance the efficiency of the encoding process (as shown in Figure 3).
- (3) **Entity Annotation:** based on a predefined list of sensitive privacy entities, each sentence vector is annotated using the BIOES tagging scheme. The annotation dimension is fixed at 100, with short sentences padded with zeros and long sentences truncated (as shown in Figure 4).

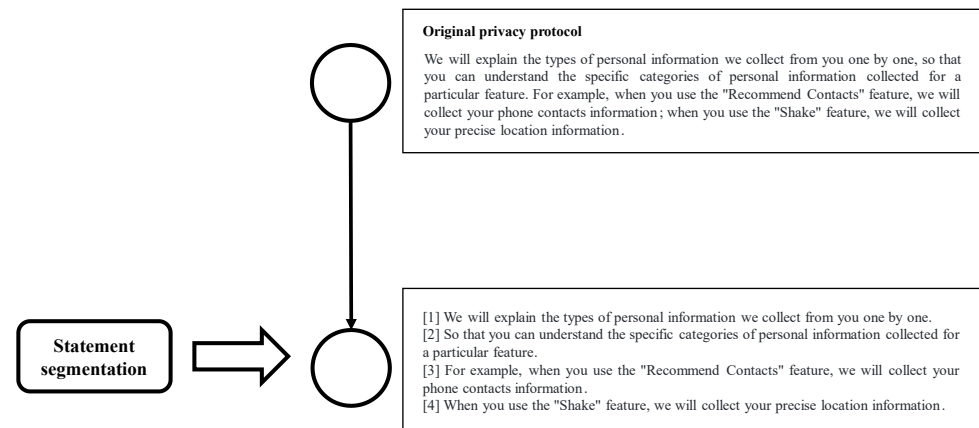


Figure 2. Statement segmentation.

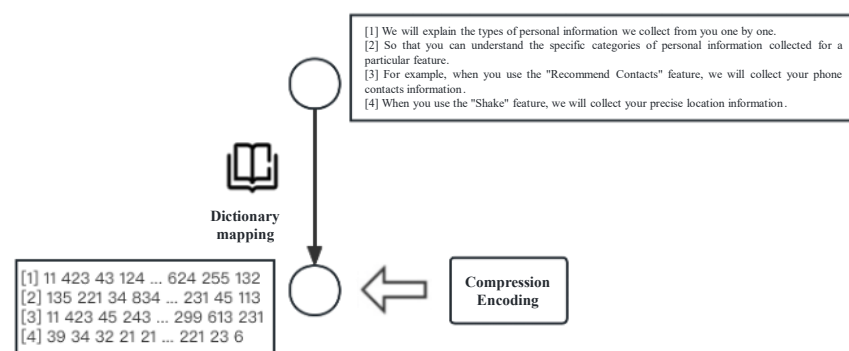


Figure 3. Compressed encoding.

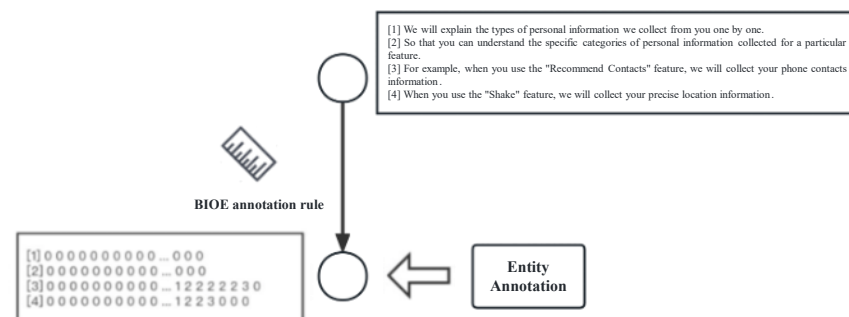


Figure 4. Entity labeling.

Finally, the generated sentence vectors and annotation vectors are combined to form data samples, which are serialized using pickle to reduce storage space and provide data for subsequent training. The detailed process is outlined in Algorithm 1.

Step 1: Sentence Segmentation: first, traverse the collection of privacy agreement paragraphs, referred to as *InitSectionSet*, and determine the length of each paragraph, *Section*. If the length is less than 100 characters, the paragraph is directly added to the sentence collection *AvailableSenSet*. If the length exceeds 100, the paragraph is split into multiple sentences using delimiters such as ".", "!", and "?". Sentences with lengths less than 100 are added to the sentence collection *AvailableSenSet*, while sentences longer than 100 characters are placed in the *NeedCommaSplitSenSet* for further splitting. Next, traverse the *NeedCommaSplitSenSet* collection, and for each sentence, use delimiters like commas, colons, and semicolons for further segmentation. Sentences with sub-clause lengths shorter than 100 characters are added to the *AvailableSenSet*, while those longer than 100 are placed in the *NeedSyntaxSplitSenSet* for final splitting. Finally, traverse the *NeedSyn-*

taxSplitSenSet collection, and for each sub-clause, apply the splitBySyntax() function to perform the final segmentation using syntactic analysis. If the sub-clause is shorter than 100 characters, it is added to the AvailableSenSet; otherwise, it is discarded.

Step 2: Compression Encoding: traverse the AvailableSenSet collection of segmented sentences. For each sentence Sen, map each character in the sentence to a short integer representation using the Mapping() function. The dictionary CharDict contains a mapping of each character to a short integer, for example, CharDict = {"I": 11, "individually": 124, "we": 423}, and when a character is not found in the dictionary, it defaults to 0. As a result, each character in the sentence corresponds to a short integer representation. The sentence Sen can, thus, be represented as a sentence vector consisting of short integers. To ensure that the sentence vector SenVec has a fixed dimension of 100, compute the difference between the sentence length and 100, padding zeros to the end of the vector. Finally, add the mapping between the sentence Sen and the sentence vector SenVec to the hash table VecMapper.

Step 3: Entity Annotation: traverse the AvailableSenSet collection and construct the privacy-sensitive entity annotation vector for each sentence. For any given sentence Sen in the collection, when reaching the i -th character in the sentence, call the findRelatedWord() function to search for privacy-sensitive words starting from the i -th character and determine their start (Start) and end (End) positions within the sentence. The function findRelatedWord() iterates through the privacy-sensitive entity list, and if a matching entity PSE is found (i.e., $\text{match}(\text{PSE}, \text{Sen}[i \dots | \text{PSE}|])$), then $\text{Start} = i$ and $\text{End} = i + |\text{PSE}| - 1$. If no match is found, then $\text{End} = \text{Start}$. If matching fails, set $\text{LabelVec}[i] = 0$; when matching succeeds, set $\text{LabelVec}[\text{Start}] = 1$, $\text{LabelVec}[\text{End}] = 3$, and set all positions between $\text{Start} + 1$ and $\text{End} - 1$ to 2. Then, set $i = \text{End} + 1$ and begin a new round of entity annotation from the next character. This process continues until all characters in the sentence have been annotated. As a result, each segmented sentence Sen corresponds to an annotation vector LabelVec, which is combined with its sentence vector by looking up the hash table VecMapper to form training data samples, which are added to the training dataset TrainableSamples.

Step 4: Output: The function returns the training dataset TrainableSamples.

Algorithm 1: Privacy-sensitive Entity Recognition Dataset Construction Algorithm

Input: Privacy policy statement paragraph set InitSectionSet; character hash table charDict

Output: Training dataset TrainableSamples

```

1  For Section in InitSectionSet then
    If Len(Section) < 100 then
        AvailableSenSet += Section; //Directly add to the construction set if length is less
        than 100
    End If
    startIndex = 0;
    For i in 0 ... Len(section)
        If section[i] = '!' or section[i] = '!' or section[i] = '?' then //Split by period,
        exclamation mark, or question mark
            CurrentSen = section.split(startIndex, i);
            StartIndex = i;
            If Len(CurrentSen) < 100 then
                AvailableSenSet += CurrentSen; //Add to AvailableSenSet if length is less
                than 100
            Else NeedCommaSplitSenSet += CurrentSen; //Otherwise, add to
            NeedCommaSplitSenSet

```

```

        End If
    End If
End For
For Sen in NeedCommaSplitSenSet then
    Sentences = Sen.split(",",";",".");//Split using comma, colon, and semicolon
    For CurrentSen in Sentences then
        If Len(CurrentSen) < 100 then//Add to AvailableSenSet if length is less than
100
            AvailableSenSet += CurrentSen;
            Else NeedSynaxSplitSenSet += CurrentSen;
        End If
    End For
End For
For Sen in NeedSynaxSplitSenSet then
    Sentences = splitBySyntax(Sen);//Split by syntax
    For CurrentSen in Sentences then
        If Len(CurrentSen) < 100 then//Add to AvailableSenSet if length is less than
100
            AvailableSenSet += CurrentSen;
        End If
    End For
End For
2 For AvailableSen in AvailableSenSet then
    For i in 0 ... Len(AvailableSen) then
        SenVec[i] = Mapping(charDict, AvailableSen [i]);//Query the character's
mapping value in the dictionary
    End For
    For I in Len(SenVec) ... 100 then
        SenVec += 0;//Pad zeros at the end
    End For
    addVecMapping(VecMapper, Sen, SenVec);//Establish sentence and vector mapping
relationship
End For
3 For AvailableSen in AvailableSenSet then
    i = 0
    While i < Len(Sen) do
        Start, End = findRelatedWord(AvailableSen[i]);//Find the starting and ending
positions of sensitive words starting with the current character in the sentence
        If Start == -1 and End == -1 then//Matching succeeded
            LabelVec[i] = 0; I += 1;continue;
            LabelVec[i] = 1; LabelVec[i] = 3;
        For i in Start +1 ... End -1
            LebelVec[i] = 2;
        End If
        i = end + 1
        TrainableSamples += [Mapping(VecMapper, AvailableSen),LabelVec];//Add to
training set
    End For
4 Return TrainableSamples

```

5. CPS-LSTM: Privacy-Sensitive Entity Adaptive Recognition Model for Power Systems

This section presents the Chinese privacy-sensitive entity adaptive recognition model CPS-LSTM, which is designed for automatically generating privacy policy labeling data in large batches. The CPS-LSTM model employs a BiLSTM network as the semantic context feature extractor and incorporates a multi-head attention mechanism to integrate word vector information for lexical augmentation of sentence vector representations. Ultimately, the model uses a CRF Viterbi decoder to learn the latent feature sequence of entity labeling relationships.

5.1. Model Architecture

The complete network architecture of the privacy-sensitive entity adaptive perception model is shown in Figure 5. The model receives sentence vector information obtained through character compression encoding and utilizes a BiLSTM network that retains contextual semantic information to extract features. For each character in the sentence, the model looks up all known privacy-sensitive entities related to the character in a predefined lexicon and uses a multi-head attention-based word embedding technique to enhance lexical information for the character. The BiLSTM feature extractor is then used to further expand the semantic information. Subsequently, the model feeds the sentence vector representation containing lexical information into a fully connected layer for character-level labeling prediction. The predicted results are compared with the actual labeling vectors, and the CRF decoder is used to evaluate and compute the loss value. This process gradually fits the model parameters to achieve the optimal labeling prediction performance.

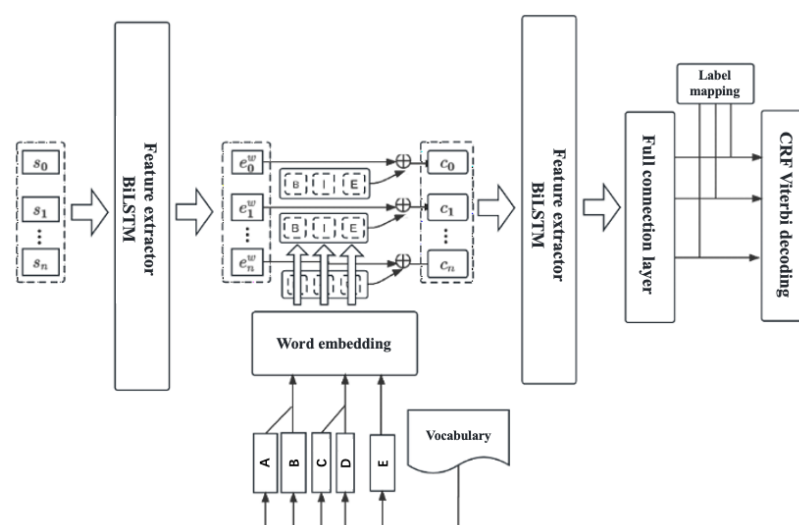


Figure 5. CPS-LSTM network architecture.

5.2. BiLSTM-Based Character Embedding Feature Extraction Method with Contextual Semantic Retention

This section employs the BiLSTM model structure [13] as the feature extractor, which processes the binary serialized data of privacy policy sentences in two directions. One model processes the sequence from left to right, while another model processes it from right to left. By merging the hidden state vectors from both directions, a character embedding vector that retains contextual semantic features is obtained. The resulting character embedding vector contains word features, syntactic features, and grammatical features from the sentence, offering a richer semantic representation that facilitates the model's target fitting and training.

The BiLSTM uses a dual-layer LSTM network structure to extract both forward semantic dependencies and backward semantic dependencies from the clauses in the privacy policy. These two directional feature sets are merged and stacked to generate a feature vector that retains contextual semantic dependencies. LSTM computes the hidden layer output of the current neuron based on the previous semantic features, effectively addressing the issues of gradient explosion and gradient vanishing commonly encountered in traditional RNN networks when handling long-term dependencies. This method shows particularly notable effectiveness with privacy policy clauses, which typically have a length distribution of around 100 words, resulting in small errors. The formal description of the state transition of the LSTM neuron's hidden layer is given in Equation (1) [14,15].

$$\begin{aligned} \begin{bmatrix} I_i \\ O_i \\ F_i \\ \tilde{C} \end{bmatrix} &= \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} (W_p[X_i H_{i-1}] + B_p), \\ C_i &= \tilde{C} \otimes I_i + \tilde{C}_{i-1} \otimes F_i, \\ H_i &= O_i \otimes \tanh(C_i), \\ e_i^w &= \vec{e}_i^w \oplus \overleftarrow{e}_i^w. \end{aligned} \quad (1)$$

In this section, $\tilde{C}$ represents the candidate memory cell, i_i denotes the input gate, o_i represents the output gate, and f_i denotes the forget gate, while $\otimes$ indicates the Hadamard product. The hidden layer output and memory cell features of each neuron depend on the computation from the previous neuron's cell. $\vec{e}_i^w$ is represented as the final hidden layer output of the LSTM for the i -th character of the privacy policy sentence. The forward semantic representation of the privacy policy clause can be denoted as $\vec{s}_w = \{\vec{e}_0^w, \vec{e}_1^w, \dots, \vec{e}_n^w\}$, and correspondingly, the backward semantic representation can be denoted as $\overleftarrow{s}_w = \{\overleftarrow{e}_0^w, \overleftarrow{e}_1^w, \dots, \overleftarrow{e}_n^w\}$. By merging and stacking the semantic representations from both directions, the complete semantic representation of the privacy policy clause can be denoted as $\leftrightarrow s_w = \{\leftrightarrow e_0^w, \leftrightarrow e_1^w, \dots, \leftrightarrow e_n^w\}$.

5.3. Method for Constructing Character Representations with Integrated Word Embeddings

This section proposes a word embedding fusion method based on the Multi-Head Attention mechanism, which incorporates predefined sensitive entity vocabulary information to enrich the character embedding representation. This enables the model to more accurately learn the feature information of privacy-sensitive entities in the target text during the training process, thus improving the model's recognition accuracy.

The absence of vocabulary information can lead to the model's inability to accurately understand the text, resulting in incorrect entity labeling predictions. Therefore, the feature construction method in this section introduces vocabulary information enhancement learning to increase the information content of sentence vectors. Based on Lattice LSTM, this section applies the Multi-Head Attention mechanism [8] to fuse the word embedding representations under different labels. After concatenating and stacking the fused word vectors B_i^w, I_i^w , and E_i^w under different labels, they are added to the character embedding representation e_i^w to generate a character embedding representation c_i^w enriched with integrated vocabulary information, thus achieving the goal of vocabulary enhancement. The feature extraction network structure for integrating word vector information is shown in Figure 6.

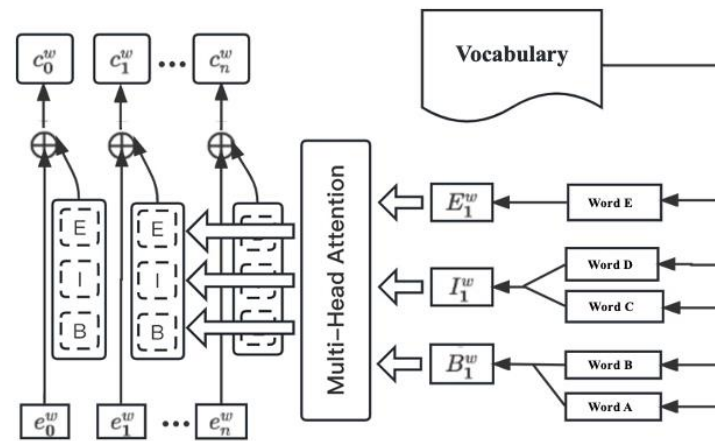


Figure 6. Construction process of word embedding representation based on multi-word fusion.

For each character in the sentence segmentation of the protocol, contextual information from both the preceding and following context is used to retrieve the sensitive entity vocabulary association list for that character under all possible labels from the predefined seed privacy-sensitive list. In this paper, $B(e_i^w)$, $I(e_i^w)$, and $E(e_i^w)$ represent the sets of words matched by the character e_i^w under different labels. The mathematical description of the sensitive entity dictionary matching is shown in Equation (2).

$$\begin{aligned} B(c_i) &= \{w_{ik} | \forall w_{ik} \in \text{Lexicon}, i < k < n\} \\ I(c_i) &= \{w_{jk} | \forall w_{jk} \in \text{Lexicon}, j < i < k\} \\ E(c_i) &= \{w_{ki} | \forall w_{ki} \in \text{Lexicon}, 0 \leq k < i\} \end{aligned} \quad (2)$$

All matched vocabulary must be converted into word embedding representations. The vocabulary can be represented as a sequence of characters, i.e., $= \{e_i^w, \dots, e_j^w\}$. This section employs the Self-Attention mechanism to identify the contrastive weight relationships between character embeddings in privacy-sensitive entities for information aggregation representation. The formalized description is given by Equation (3) [16], where $a_{m,n}$ denotes the influence weight coefficient of character e_n^w on character e_m^w .

$$\begin{aligned} e_{ij}^l &= \text{SelfAttention}(e_i^w, \dots, e_j^w) \\ \text{SelfAttention}(e_i^w, \dots, e_j^w) &= \sum_{n=i}^j a_{m,n} e_n^w \end{aligned} \quad (3)$$

The influence weight coefficients are determined by the impact scoring function provided in this paper, and the computation process is shown in Equation (4). Here, w_1 and w_2 are trainable model parameters, which are continuously refined through backpropagation and gradient descent to minimize the loss function.

$$\begin{aligned} a_{m,n} &= \text{Softmax}(\text{effect_score}(e_m^w, e_n^w)) \\ \text{effect}(e_m^w, e_n^w) &= \text{LeakyRelu}(w_1 * e_m^w + w_2 * e_n^w) \end{aligned} \quad (4)$$

The word vectors for different label classifications are represented as B_i^w , I_i^w , and E_i^w , as shown in Equation (5).

$$B_i^w = \{B_0^i, \dots, B_n^i\}, I_i^w = \{I_0^i, \dots, I_n^i\}, E_i^w = \{E_0^i, \dots, E_n^i\} \quad (5)$$

Since the vocabulary retrieved for each label may differ, this paper employs the Multi-Head Attention mechanism to integrate multiple word embedding representations under each label classification. As each word embedding has a dimensionality of 512, eight atten-

tion heads are used to split the word embedding vectors into sub-feature spaces of length 64 for separate fusion. Finally, the results are concatenated to form the complete word embedding representation. The word embedding fusion process is illustrated in Figure 7.

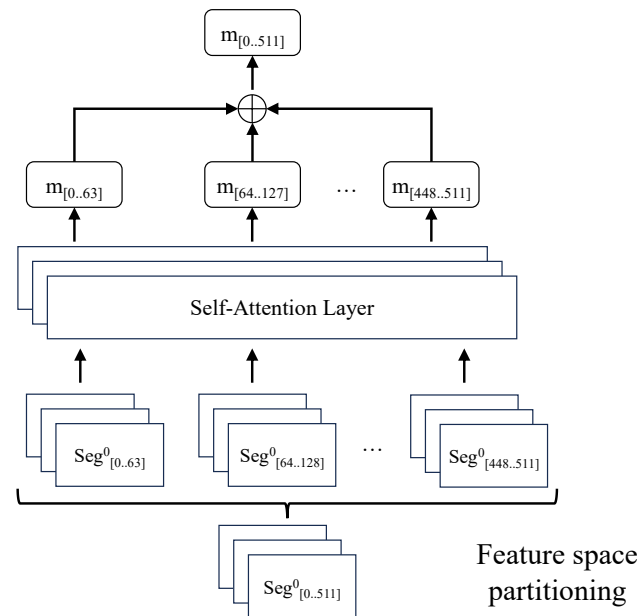


Figure 7. Fusion process of multi-word embedding.

For each sub-feature space of length 64, the Attention weight calculation formula is shown in Equation (6). Here, w_1 and w_2 are trainable model parameters, and information fusion is performed for each word's sub-feature space based on the weights. At this point, each label classification of a character corresponds to an embedding vector that mixes the word meanings.

$$a_{[j..k]}^i = \text{softmax}\left(w_1 * \text{Relu}\left(w_2 * \text{seg}_{[j..k]}^i\right)\right) \quad (6)$$

In this paper, the fused word embeddings B_i^w, I_i^w , and E_i^w from different label classifications are concatenated and then stacked onto the character embedding representation e_i^w , resulting in the character embedding representation c_i^w that helps the model understand lexical information, as shown in Equation (7). Through lexical enhancement, the model is better able to adapt to variations in unseen text, such as the emergence of potential privacy-related entities, thereby significantly improving the model's accuracy.

$$c_i^w = e_i^w \oplus [B_i^w; I_i^w; E_i^w] \quad (7)$$

5.4. Position Label Prediction Optimization Algorithm Based on CRF Viterbi Decoder

This section utilizes the CRF Viterbi decoder to optimize the prediction results obtained through the fully connected layer, leveraging the contextual information of privacy agreement statements to more accurately identify privacy-sensitive entities.

First, the Viterbi decoder analyzes the contextual information of the privacy agreement statement, including the characters before and after each character in the sentence. Then, the Viterbi decoder recursively labels each character in the sequence based on the contextual information of the sentence. It starts from the first character in the sequence, predicting the label of the current character based on the label of the previous character. Next, it predicts the label of the next character based on the current character's label. This recursive process continues until the labels for all characters in the sentence are predicted. Finally, the Viterbi decoder calculates the prediction accuracy and returns the most accurate sequence of sentence labels.

Let there be a sentence embedding representation vector $X = (X_1, X_2, \dots, X_n)$ and its corresponding label vector $Y = (Y_1, Y_2, \dots, Y_n)$. Given the sentence embedding representation vector X , the corresponding label vector Y forms a Conditional Random Field $P(Y|X)$ that satisfies the Markov property. The Bayesian probability representation is shown in Equation (8).

$$P(Y_i|X, Y_1, \dots, Y_{i-1}, Y_{i+1}, Y_n) = P(Y_i|X, Y_{i-1}, Y_{i+1}) \quad (8)$$

Under the condition that the sentence embedding representation vector X takes any value X_i , the conditional probability of the corresponding label values Y_i is calculated as shown in Equation (9). Here, t_k and s_l are feature functions, λ_k and u_l are weight parameters, and $Z(x)$ is the normalization factor. The label prediction result with the highest probability Y_i^* for each character X_i is denoted as the maximum probability label.

$$\begin{aligned} P(Y_i|X_i) &= \frac{1}{Z(X_i)} \exp(\sum_{ik} \lambda_k t_k(Y_{i-1}, Y_i, X_i, i) + \sum_{il} u_l s_l(Y_i, X_i, i)) \\ Z(X_i) &= \sum_{Y_i} \exp(\sum_{ik} \lambda_k t_k(Y_{i-1}, Y_i, X_i, i) + \sum_{il} u_l s_l(Y_i, X_i, i)) \\ Y_i^* &= \arg\max(P(Y_i|X_i)) \end{aligned} \quad (9)$$

For the named entity recognition task, the squared loss function is commonly used to measure the prediction accuracy. The squared loss function is shown in Equation (10).

$$L = \sum_{i=0} (Y_i^* - Y_i)^2 \quad (10)$$

Here, L represents the value of the loss function, y_i denotes the value of the i -th element in the true label sequence, and $\bar{y}_i$ represents the value of the i -th element in the predicted label sequence. When the predicted label sequence exactly matches the true label sequence, the value of the squared loss function is 0. However, when the predicted label sequence differs from the true label sequence, the value of the squared loss function increases. Therefore, the prediction accuracy can be measured by controlling the value of the squared loss function. Additionally, the Viterbi decoder can help optimize the loss function. The Viterbi decoder can be used to find a label sequence that minimizes the loss function value. This allows for the maximization of prediction accuracy and minimization of prediction error.

In this section, the loss function optimized by the CRF decoder is used, which is capable of learning the transition features of potential relationships in character labeling. The loss function optimized by the CRF decoder consists of two parts: the score of the true label sequence and the score of all possible label sequences. The score of the true label sequence should be the highest among all paths, and the objective of model training is to minimize the loss function. Therefore, a negative sign is introduced here, and the loss function is defined as shown in Equation (11).

$$loss = - \left(\sum_{i=1}^N x_i y_i + \sum_{i=1}^N t y_i y_{i+1} - \log(e^{s_1} + e^{s_2} + \dots + e^{s_n}) \right) \quad (11)$$

5.5. Model Construction

The detailed network design of the adaptive recognition model for privacy-sensitive entities, CPS-LSTM, is shown in Figure 8. It processes and recognizes privacy protocol statements through the following four steps.

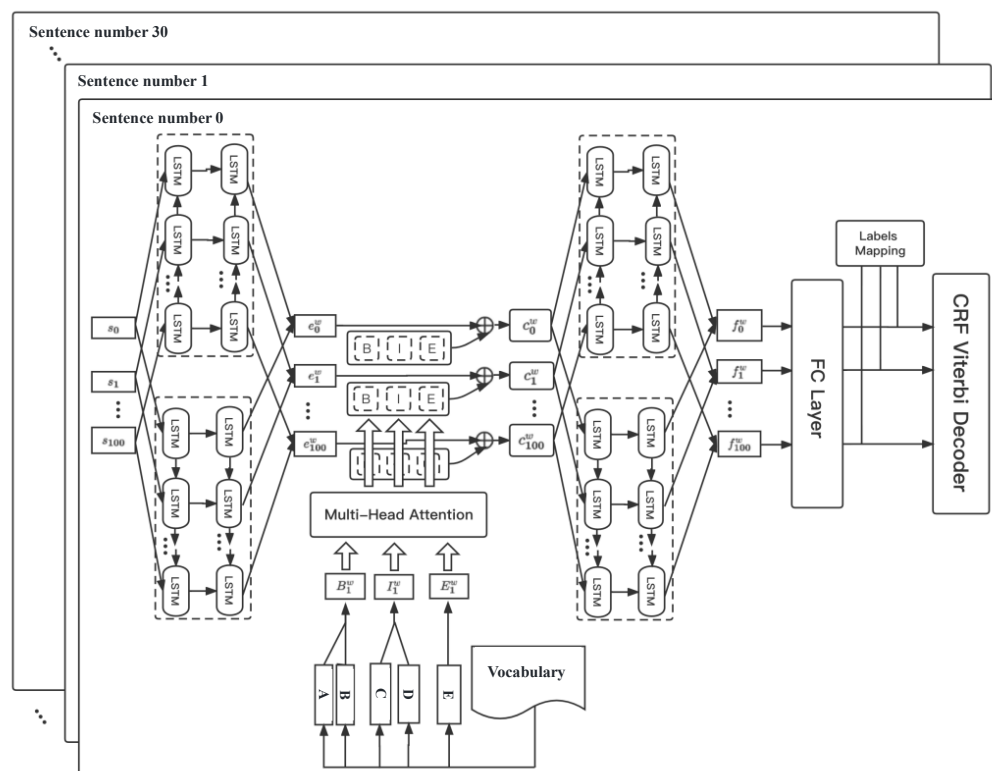


Figure 8. Detailed network design of CPS-LSTM.

Step 1: Constructing Sentence Vector Representations with Contextual Semantics. The input neurons of the input layer receive initial sentence vector representations $[s_i, \dots, s_{100}]$, which are constructed based on a dictionary. To enable parallel computation of protocol sentences, the batch size is set to 30, meaning that the model can process 30 protocol sentences in parallel. The initial sentence vector representations $[s_i, \dots, s_{100}]$ are then fed into a BiLSTM (Bidirectional Long Short-Term Memory) network to extract embedded representations $[e_0^w, \dots, e_{100}^w]$ that retain contextual semantics.

Step 2: Identifying Sensitive Terms in the Input Sentence and Constructing Word Vector Representations. For instance, in the phrase “Network IP belonging region,” the character “地” (meaning “region”) can be matched with a term starting with “地” in the lexicon, such as “地区” (region). This can be referred to as a B word sequence. Similarly, I and E word sequences can also be identified. The relative self-attention mechanism is then applied to calculate corresponding word embedding representations for all the identified terms in the lexicon.

Step 3: Fusion of Word Embeddings for Identified Labels and Word Vector Enhancement. All word embeddings for terms in the B label category are fused using a multi-head attention mechanism. The same applies to the I and E label categories. At this point, the fused word embeddings for the different labels are obtained, denoted as B_i^w, I_i^w, E_i^w . These fused word embeddings are then concatenated with the character embeddings to enhance the word semantics in the final character embeddings, resulting in the enriched character embedding c_i^w .

Step 4: Expanding Word Semantics and Initial Prediction via Fully Connected Layer, Combined with Provided Label Vectors for Viterbi Decoder Optimization. Within the current sentence vector, only characters within the entity are capable of perceiving the word information, whereas external characters do not receive word-level information, leading to insufficient semantic smoothness. To address this, BiLSTM is employed to further extract features, resulting in the final sentence representation vector $[f_0^w, \dots, f_{100}^w]$. The prediction

model uses a fully connected layer to evaluate the prediction scores for each label, and the provided label vectors are combined with the actual predictions as inputs to the CRF (Conditional Random Field) Viterbi decoder, calculating the loss value. The loss value is then backpropagated to update the gradient information for parameter adjustments. The Adam optimizer is used for adaptive learning rates, allowing the model to converge faster and achieve better learning results.

6. Experiment and Discussions

This section validates the feasibility and effectiveness of the privacy-sensitive identification model based on the character representation method fused with word vector information. To ensure the fairness of the experimental results, all models in the experiment use the same dataset and hyperparameter combinations, with standardized model evaluation metrics for the experimental data.

6.1. Experimental Environment and Dataset

The model training for this experiment was deployed on Kaggle, a globally renowned machine learning and data science platform, utilizing the free and open GPU computing resources to accelerate model training in parallel. The neural network structure of the model was implemented using PyTorch1.7.1, leveraging its built-in Tensor acceleration capabilities to enable rapid deployment on the CUDA computing platform for hardware acceleration. Since the model validation process does not require high computational power, the validation process was set up in the local environment, using the available CPU resources and graphical visualization to demonstrate the model's recognition performance. The specific hardware and software parameters of the experimental platform are shown in Table 2.

Table 2. Hardware and software parameters of the experimental platform.

Experimental Platform	Hardware Parameters	Software Parameters
Kaggle	CPU: Intel Xeon 2-core GPU: Two Navida Tesla T4 Memory: 13 GB	Python 3.8 Pytorch 1.7.1
Local Environment	CPU: Intel Core i9 2.3 GHz (Intel Corporation, Suzhou, China) GPU: AMD Radeon Pro 5500 M (Advanced Micro Devices, Shanghai, China) Memory: 32 GB	Python 3.7 Pytorch 1.12.1

PyTorch, as a globally recognized deep learning framework, provides advanced tools for model deployment and is widely used in the deep learning community. Numerous researchers and developers continuously contribute a variety of base model architectures, making the framework rich with useful code libraries and tools. This makes PyTorch particularly suitable for exploratory deep learning research. Additionally, PyTorch supports CUDA parallel GPU acceleration, which significantly enhances the training and inference speed of deep learning models.

The data samples were collected using web crawlers from the internet, randomly sampled by category, totaling 1000 application privacy policies. The categories include Video (50 samples, 5%), Music (50 samples, 5%), Lifestyle (100 samples, 10%), News (100 samples, 10%), Learning (100 samples, 10%), Social (200 samples, 20%), Tools (200 samples, 20%), and Games (200 samples, 20%). The collected privacy policies were then split into three parts in an approximate 8:1:1 ratio for the training set (Tr), test set (Te), and validation

set (Va). Due to some privacy policies containing text that was obfuscated or encrypted, which hindered natural language processing, all documents were filtered using the UTF-8 character encoding standard. Any documents not adhering to this encoding rule were deemed invalid. After the filtering process, 982 valid privacy policy samples remained. Of these, 750 were used to automatically construct the training set (Tr), 100 were used to manually construct the test set (Te), and the remaining 122 were used to manually construct the validation set (Va), with the distribution roughly as shown in Table 3.

Table 3. Distribution of Dataset Categories.

Type	Video	Music	LifeStyle	News	Learning	Social	Tools	Games
Train	37	38	75	75	75	150	150	150
Test	6	5	11	11	11	22	22	22
Val	5	6	11	11	11	22	22	22

This section evaluates the model's recognition performance using metrics such as Precision, Recall, False Positive Rate (FPR), False Negative Rate (FNR), F1 score, and AUC. A complete description of these metrics can be found in reference [17].

6.2. Validation of Dataset Construction Effectiveness

To demonstrate that the dataset construction method proposed in this paper can generate high-quality data samples in a short period of time across different privacy policy scenarios and can be effectively used for training tasks related to privacy-sensitive entity recognition, this section employs a randomized controlled experimental design combined with manual evaluation methods to assess the experimental results. In this experiment, a total of 30 application privacy policies were randomly collected from different application markets, then randomly divided into three distinct groups. These were sequentially input into the data sample generator to construct data samples. The experiment used a questionnaire survey method for manually generated samples, which was employed to verify the quality of samples generated by the dataset construction method based on privacy policies. The results are shown in Table 4.

Table 4. Validation of dataset construction effect.

Group Number	Privacy Sentences Constructed by Our Method	Privacy Sentences Manually Constructed	Precision	FPR	FNR
Group 1	353	392	0.86	0.11	0.23
Group 2	347	374	0.89	0.08	0.16
Group 3	351	362	0.94	0.04	0.09

The experimental results indicate that the dataset construction method proposed in this paper leads to an improvement in the accuracy of the generated data samples as the experimental groups are adjusted, with accuracy consistently above 85%. This suggests that the method can better identify sensitive keywords within privacy policy statements, which can, in turn, assist downstream entity recognition models in more effectively identifying potential privacy-sensitive entities.

6.3. Comparison Validation of the Vocabulary-Augmented Model Performance

To validate the effectiveness of the privacy-sensitive entity recognition model proposed in this paper in adaptively mining privacy-sensitive entities from unknown target texts, this section compares it with other named entity recognition models based on the

vocabulary augmentation idea, including Lattice LSTM6, WC-LSTM7, and FLAT8. To ensure fairness and accuracy, all models were trained and evaluated using the training set, test set, and validation set generated during the experimental setup phase. The networks were implemented using PyTorch, and the code logic was manually reviewed multiple times to ensure optimal network performance at the logical level. All models used the Adam optimizer to adjust model parameters and the CRF Viterbi decoder to calculate the loss. Additionally, all models employed the same hyperparameter combination (learning rate = 1×10^{-3} , batch size = 30, epochs = 16) to avoid unintended influences from irrelevant external factors on the experimental results.

This section evaluates the recognition performance of all models using the validation dataset, which includes 122 privacy policy statements, based on metrics such as Precision, Recall, F1, and AUC. The experimental results are presented in Table 5.

Table 5. Validation of vocabulary enhancement model effect.

Entity Recognition Model	Precision	Recall	F1	AUC
Lattice LSTM [6]	0.83	0.78	0.81	0.91
WC-LSTM [7]	0.78	0.73	0.76	0.89
FLAT [8]	0.87	0.83	0.85	0.91
CPS-LSTM	0.92	0.85	0.88	0.93

The experimental results indicate that the word embedding fusion-based character representation method proposed in this paper outperforms other vocabulary-augmented entity recognition models in terms of privacy-sensitive entity recognition capability. It is able to identify a greater number of potential sensitive entity nouns within the given target text.

6.4. Model Ablation Experiment Validation

The model architecture proposed in this paper consists of three main components: the BiLSTM-based text feature extractor, which constructs the contextual semantics of privacy policy statements; the character embedding construction method that integrates word vectors based on a multi-head Attention mechanism; and the CRF Viterbi decoder, which determines the optimal annotation prediction path to compute the loss value. To further demonstrate the necessity of each module in the network structure, this section designs an ablation study to compare and validate the recognition performance.

6.4.1. Feature Extractor Comparison Experiment

To illustrate the importance of the BiLSTM module in the entity recognition task of this model, this subsection conducts a comparison experiment with three other commonly used text feature extractors: word2vec, BERT, and TextCNN. PyTorch provides implementations of these three neural network structures, which minimizes the impact of additional human factors on model performance. In this subsection, the sentence representation construction module is replaced with one of the four feature extractors, including BiLSTM. To ensure fairness and consistency, the same dataset and hyperparameter combination were used for training and validation, and the experimental results are shown in Table 6.

The experimental results show that BiLSTM, as a feature extractor, effectively handles sequential data such as text, making it particularly suitable for adaptive recognition tasks like named entity recognition. Word2vec, as a commonly used pre-trained word embedding model, does not consider the contextual semantics of privacy policy statements, and, therefore, performs poorly in named entity recognition tasks. BERT is primarily designed to address semantic similarity issues and has broad applications in natural language pro-

cessing. However, in named entity recognition tasks, it still lags behind sequence-based semantic models like BiLSTM. TextCNN extracts text features through convolutional layers and is typically used for text classification tasks. The experimental results demonstrate that its performance in extracting semantic features is suboptimal.

Table 6. Necessity verification of feature extractor.

Feature Extractor	Precision	Recall	F1	AUC	FPR	FNR
word2vec	0.67	0.72	0.69	0.85	0.23	0.28
BERT	0.92	0.88	0.90	0.96	0.092	0.122
TextCNN	0.74	0.69	0.71	0.87	0.12	0.151
BiLSTM	0.95	0.91	0.93	0.97	0.048	0.029

6.4.2. Comparison of Vocabulary Fusion Methods

Next, we compare the impact of different vocabulary fusion methods on the model's actual privacy-sensitive entity recognition ability. To prove the effectiveness of vocabulary augmentation, this section sets up a group of experimental models that do not incorporate vocabulary information into the character embeddings. Then, a comparison experiment is conducted using the sum and average vocabulary fusion methods to construct entity recognition models. Finally, the model based on the multi-head attention mechanism for fusion of word vectors, as proposed in this paper, is used for experimental comparison. The experimental results are shown in Table 7.

Table 7. Necessity verification of vocabulary enhancement methods.

Enhancement Methods	Precision	Recall	F1	AUC	FPR	FNR
No enhancement	0.54	0.42	0.47	0.63	0.21	0.58
sum and average	0.76	0.64	0.69	0.77	0.08	0.36
multi-head attention	0.94	0.92	0.93	0.96	0.051	0.03

The experimental results demonstrate that the character embedding representation method based on the multi-head attention mechanism for word vector fusion, as employed in this study, enriches the semantic representation of the sentence vectors in the protocol. This enhancement is beneficial for the named entity recognition annotation and prediction tasks.

6.4.3. Loss Function Comparison Experiment

To further prove the superiority of the CRF-based Viterbi decoder employed in this model for calculating the actual annotation prediction loss value, we compare several loss functions commonly used in named entity recognition tasks. These include the cross-entropy loss function, information gain loss function, hinge loss function, and the CRF conditional probability loss function used in this model. The experimental results are presented in Table 8.

Table 8. Necessity verification of loss functions.

Loss Function	Precision	Recall	F1	AUC	FPR	FNR
Cross-entropy	0.65	0.54	0.59	0.71	0.14	0.46
Information Gain	0.63	0.52	0.57	0.69	0.16	0.48
Hinge	0.67	0.57	0.61	0.73	0.13	0.43
CRF Conditional Probability	0.93	0.92	0.92	0.97	0.046	0.12

The experimental results indicate that the CRF conditional probability loss function outperforms the other loss functions. The CRF conditional probability loss function can reflect the sequential labeling relationships, effectively capturing the features of sequence data. This makes the conditional probability loss function perform better in sequence labeling tasks such as named entity recognition. Additionally, the CRF conditional probability loss function is more flexible during model training. Depending on the data, it can adaptively select different transition probability matrices, which helps the model better accommodate various annotation tasks.

The experimental results comprehensively validate the effectiveness of the proposed privacy-sensitive entity recognition framework. By constructing a high-quality dataset (98% precision in Group 3) and integrating a BiLSTM-based feature extractor with multi-head attention-driven word vector fusion, the model achieves superior performance ($F1 = 0.93$, $AUC = 0.97$) compared to baseline methods (e.g., Lattice LSTM, BERT). Ablation studies confirm the necessity of each component: BiLSTM outperforms TextCNN and BERT in capturing sequential context ($F1 = 0.93$ vs. $0.71/0.90$), multi-head attention significantly enhances entity recognition over naive fusion methods ($F1 = 0.93$ vs. 0.69), and the CRF loss function optimizes sequence labeling ($F1 = 0.92$ vs. 0.59 – 0.61 for other losses). These results demonstrate the model's ability to precisely identify privacy-sensitive entities (e.g., "personal information") while maintaining low error rates ($FPR = 0.046$, $FNR = 0.03$), making it robust for real-world privacy governance tasks.

7. Conclusions

This paper proposes an innovative data construction method and model for the adaptive recognition of privacy-sensitive entities in Android privacy protocols within power equipment. First, addressing the issues of the lack of standardized naming for platform-specific entities and the data sparsity problem caused by simple sentence segmentation methods, a high-quality data construction method for privacy protocols is introduced. This method uses sentence segmentation based on specific characters, character compression mapping, and a BIOES-style annotation scheme to provide an efficient and compact dataset for model training. The samples generated by this data construction method achieve an accuracy of at least 85%, effectively solving the problem of insufficient privacy protocol datasets in the power equipment domain.

Next, to address the low accuracy of existing named entity recognition (NER) methods in the case of mixed-language expressions, an adaptive sensitive entity recognition model, CPS-LSTM, is proposed. During the construction of the sensitive entity recognition model, a novel word embedding-character representation fusion method is introduced. By incorporating lexical information into word embedding features, the model significantly enhances its ability to perceive character boundaries. Furthermore, the use of a CRF Viterbi decoder and a position-based label prediction optimization algorithm enables the model to effectively learn the sequential features of entity labeling relationships, solving the issue of poor accuracy in independent label predictions. Compared with existing baseline methods such as Lattice LSTM, WC-LSTM, and FLAT, the CPS-LSTM model shows improvements in both accuracy and recall, with accuracy increasing by 0.09, 0.14, and 0.05, and recall increasing by 0.07, 0.12, and 0.02, respectively. These results demonstrate that the proposed multi-word fusion lexical enhancement model achieves faster convergence and higher recognition accuracy, enabling it to effectively extract all potential privacy-sensitive entities from the target text.

Although the proposed method for privacy-sensitive entity recognition shows improvements in data construction and model accuracy, it still has limitations. The model may struggle with complex mixed-language expressions, particularly in low-resource lan-

guages. Additionally, the data construction method, which relies on specific character segmentation and compression, may not be flexible enough for more complex privacy protocols. Future research could focus on improving multilingual capabilities, diversifying datasets, and exploring more efficient training methods like adaptive labeling and online learning.

Author Contributions: Conceptualization, H.Z. (Hao Zhang); Methodology, J.W.; Software, H.Z.; Validation, X.W. and J.W.; Formal analysis, J.W., H.Z. and X.L.; Investigation, H.Z., J.W., X.W. and X.L.; Resources, H.Z., J.W., X.W. and X.L.; Data curation, H.Z. and J.W.; Writing—original draft, Z.G., H.Z. (Hua Zhang) and Z.C.; Visualization, J.W., Z.G. and X.W.; Supervision, H.Z.; Project administration H.Z. and J.W.; Funding acquisition, H.Z. (Hua Zhang). All authors have read and agreed to the published version of the manuscript.

Funding: The paper was supported by the Science and Technology Project of State Grid Corporation of China “Research on Key Technologies for Targeted Network Attack Detection of Power Monitoring System Based on Machine Learning(52010124000A)”.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: Authors Hao Zhang and Xuanyuan Wang were employed by the company State Grid Jibei Electric Power Co., Ltd. Authors Jing Wang and Xuhui Lü were employed by the company Beijing Kedong Electric Power Control System Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. The funders had no role in the design of the study, in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Su, S.; Wang, G.; Liu, L.; Chen, Q.; Wang, K. A Review on the Security Protection of Power IoT Terminals. *High Volt. Eng.* **2022**, *48*, 513–525.
2. Hou, W.J. Implementation of Modbus Protocol Stack Based on Android. *Mod. Eng. Proj. Manag.* **2023**, *2*, 239–242.
3. Guo, S.Y.; Liu, Y.; Shao, S.; Zang, Z.; Yang, C.; Qi, F. New Technologies for Cross-Domain Data Flow Security Boundary Protection in New Power Systems. *Power Syst. Autom.* **2024**, *48*, 96–111.
4. stanfordnlp. StandCoreNLP. 2020. Available online: <https://github.com/stanfordnlp/CoreNLP> (accessed on 11 February 2025).
5. Huang, Z.; Xu, W.; Yu, K. Bidirectional LSTM-CRF models for sequence tagging. *arXiv* **2015**, arXiv:1508.01991.
6. Zhang, Y.; Yang, J. Chinese NER using lattice LSTM. *arXiv* **2018**, arXiv:1805.02023.
7. Sui, D.; Chen, Y.; Liu, K.; Zhao, J.; Liu, S. Leverage lexical knowledge for Chinese named entity recognition via collaborative graph network. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; pp. 3830–3840.
8. Li, X.; Yan, H.; Qiu, X.; Huang, X. FLAT: Chinese NER using flat-lattice transformer. *arXiv* **2020**, arXiv:2004.11795.
9. Harkous, H.; Fawaz, K.; Lebre, R.; Schaub, F.; Shin, K.G.; Aberer, K. Polisis: Automated analysis and presentation of privacy policies using deep learning. In Proceedings of the 27th USENIX Security Symposium, Baltimore, MD, USA, 15–17 August 2018; pp. 531–548.
10. Yu, L.; Luo, X.; Chen, J.; Zhou, H.; Zhang, T.; Chang, H.; Leung, H.K.N. Ppchecker: Towards accessing the trustworthiness of android apps’ privacy policies. *IEEE Trans. Softw. Eng.* **2018**, *47*, 221–242. [[CrossRef](#)]
11. Wang, J.; Xiao, Y.; Wang, X.; Nan, Y.; Xing, L.; Liao, X.; Dong, J.; Serranom, N.; Lu, H.; Wang, X.; et al. Understanding Malicious Cross-library Data Harvesting on Android. In Proceedings of the 30th USENIX Security Symposium, Vancouver, BC, Canada, 11–13 August 2021; pp. 4133–4150.
12. Yu, L.; Luo, X.; Liu, X.; Zhang, T. Can we trust the privacy policies of android apps? In Proceedings of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Toulouse, France, 28 June–1 July 2016; pp. 538–549.
13. Wenzhen, J.; Hong, Z.; Guocai, Y. An efficient character-level and word-level feature fusion method for Chinese text classification. *J. Phys. Conf. Ser.* **2019**, *1229*, 012057. [[CrossRef](#)]
14. Peters, M.E.; Neumann, M.; Iyyer, M.; Grander, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. In Proceedings of the NAACL, New Orleans, LA, USA, 1–6 June 2018; p. 5.

15. Xu, G.; Zhang, Z.; Zhang, T.; Yu, S.; Meng, Y.; Chen, S. Aspect-level sentiment classification based on attention-BiLSTM model and transfer learning. *Knowl.-Based Syst.* **2022**, *245*, 108586. [[CrossRef](#)]
16. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. In *Neural Computation*; MIT Press: Cambridge, MA, USA, 1997; Volume 9, pp. 1735–1780. [[CrossRef](#)]
17. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6000–6010.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.