Network Protocol Entity Extraction Based on Few-shot Learning

Zhiyuan Chang [2] and Shoubin Li [12]
[1]University of Chinese Academy of Sciences, Beijing, China
[2]The Institute of Software, Chinese Academy of Sciences, Beijing, China
{shoubin zhiyuan}@iscas.ac.cn

***Abstract*—Knowledge Graph is a new way of knowledge collections, and building a protocol knowledge graph on RFCs can help us study and analyse network protocol better. Protocol entity extraction is one of the keys to constructing the network protocol knowledge graph.Because RFC (Request For Comments) contains detailed descriptions of basic Internet communication protocols, the protocol entities of the network can be obtained from it. However, the document format and wording are not uniform, which leads to the inability to complete the extraction task of network protocol entities based on traditional rule information extraction methods. Therefore, this paper proposes a network protocol entity extraction method based on Few-Shot Learning. This method can use a very small amount of labeled samples to extract network protocol entities from a large number of unlabeled samples and maintain high recognition accuracy. This method firstly mines as many potential network protocol entities as possible in the RFC document, and secondly performs accurate re-identification of the identified potential network protocol entities. Experiments show that using 5 manually annotated RFC documents to train our model, the accuracy of network protocol entity extraction reaches 88.4%. Compared with the existing methods, this method has higher accuracy and better robustness in terms of network protocol entity extraction, and it also has better identification ability for network protocol entities that have not appeared in the training set.**

***Index Terms*—Few-Shot Learning, Entity Extraction , Knowledge Graph, Network Protocol,RFC**

I. Introduction

With the rapid development of the Internet, network security issues have attracted increasing attention. The network protocol is an essential factor that affects network security. For network protocol, automated fuzzy testing [14] can excavate protocol vulnerabilities and improve protocol security. Besides, the network protocol identification algorithm [3] can prevent network attacks in advance and ensure network security. Maxime et al. [13] also proposed an unsupervised anomaly-based intrusion detection solution focused on protocol header analysis. The above network protocol analysis methods require protocol knowledge as support.

Protocol knowledge is usually manually extracted by experts, and it lacks portability in different protocols, which dramatically raises the cost of acquiring protocol knowledge [9]. Considering that the knowledge graph has strong reasoning ability [10], the knowledge graph is used to construct protocol knowledge. Entity extraction is the first step in constructing a knowledge graph. Since RFC documents define all network protocols, RFC documents are used as the data set for extracting entities.

RFC documents have a long writing period, and the writing styles of documents vary in different periods. Therefore, it is impossible to define a set of rules to extract entities from all RFC documents. RFC documents are semi-structured, and various charts are embedded in the text, which increases the difficulty of information extraction. For semi-structured text, Luo et al. [16] proposed a domain-specific parser to extract information from images. CoLin et al. [15] proposed a remote supervised extraction method to automatically generate labels and train a classifier to predict new instances. While for RFCs, Siva et al. [8] proposed versatile system architecture for the Text Mining in RFCs that maintains structured and unstructured data components of the document. Due to the different representations of the same entity between different documents, it is not possible to correctly extract entities through character matching, as shown in Fig. 1.

Jero et al. proposed the ZSL framework to extract entities [9], which can automatically extract new entities that are not in the original document in the unknown RFC documents. However, the extraction method they use to build the ZSL framework can only be learned on a small sample; therefore, it can only achieve good entity extraction results on specific RFC documents. Once we expand the training sample size for learning more entity features, the accuracy of entity extraction is significantly reduced compared with before the expansion of the sample size. It is inferred that this method has high flexibility and low robustness, and the extraction effect on unknown RFC documents is not ideal. In order to solve the above problems, after expanding the sample size, the ZSL framework can still be used to extract entities accurately. We propose a cascade entity extraction model. The first stage is mining potential entities. After expanding the sample size, we use the method of Jero et al. to extract the unknown entities. Due to the expansion of the sample size, the model can be applied to more RFC documents, but the extraction accuracy is also significantly reduced. The second stage is the precise identification of entities. In order to improve the extraction accuracy, we train a classifier to predict the positive samples in the entities extracted in the previous stage. The positive sample set is considered as the final entity set. Compared with the benchmark, our method has achieved significant and continuous improvement.

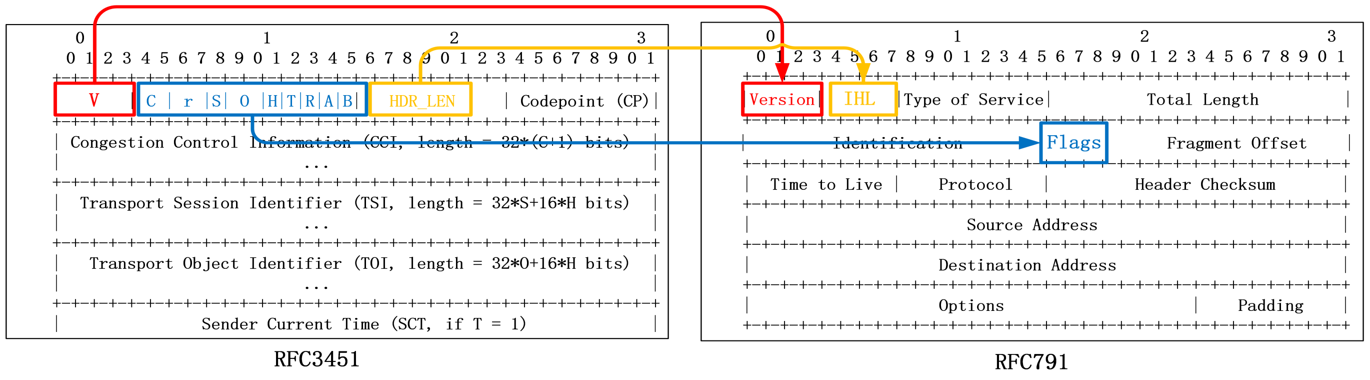
V
C | r |S| O |H|T|R|A|B|
HDR_LEN
Codepoint (CP)
Congestion Control Information (CCI, length = 32*(C+1) bits)
Transport Session Identifier (TSI, length = 32*S+16*H bits)
Transport Object Identifier (TOI, length = 32*O+16*H bits)
Sender Current Time (SCT, if T = 1)
RFC3451
Version
IHL
Type of Service
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
Options
Padding
RFC791

Fig. 1. Different writing styles of RFC documents.The header field "V" in RFC3451 has the same meaning as "Version" in RFC791 but is written differently. The same goes for the header fields in the other two boxes.

Our contributions are as follows:

1) We propose a network protocol entity extraction method based on small sample learning, which combines the advantages of traditional machine learning and deep learning and can make full use of semantic features to extract network protocol entities.
2) We evaluated the effect of the method on the RFC document and proved through experiments that the method proposed in this paper has obtained higher accuracy and F1 value, which are 88.4% and 70.4%, respectively.
3) We publish the extraction results on GitHub for researchers who are interested in network protocols.

Section 1 of this article introduces the related research on network protocol entity extraction. Section 2 introduces the entity extraction method's main ideas and design ideas based on small sample learning. Section 3 gives the experimental design scheme. Section 4 introduces and analyzes the results of the method. Finally, it summarizes the full text and makes a preliminary discussion on the research directions worthy of attention in the future.

II. Related work

A. Network protocol analysis

The protocol entity extracted in this article is the protocol header field in the RFC document. The extracted header fields can help researchers further analyze the network protocol. Calzarossa et al. [2] conducted a comprehensive analysis of the usage pattern of headers contained in the client's HTTP request message to detect potential security attacks. In addition, Bai et al. [1] used the characteristics of the jitter field in the RTCP header to set up a covert channel for secret transmission of information, thereby preventing network attacks. Similarly, Xu et al. [28] based on the structure of the HTTP protocol, embedding secret information into the header field of the HTTP protocol, and proposed a real-time information hiding algorithm to solve the problem of embedding large amounts of data into TCP/IP packets. And Ohta et al. [20] focused on the time to live (TTL) and identification field (IPID) of the IP header to understand the abnormal traffic behavior, and proposed a sequence based on the TTL and IPID fields. Separate IP addresses that may be deceptive from other addresses to extract more deceptive data packets. In addition to using network protocols to improve network security, network protocol research also includes the classification of data packets. Qu [22] and Varvello [23] and others have used the high parallelism and delay hiding function of the graphics processing unit (GPU) to quickly and effectively classify data packets. Dixit [7] proposed a fast data packet classification algorithm that uses recursive flow classification (RFC) and hierarchical spatial mapping (HSM) to process header fields, and determines the data packet classification of a single data packet according to the header field calculation table.

B. Few-Shot Learning based on small samples

Few-Shot Learning (FSL) is a type of machine learning problem, which is specified by E (experience), T (target), and P (performance index), where E only contains a small part of the supervision information for the target T [25]. FSL usually learns the projection function from the feature space to the semantic embedding space (such as the attribute space) [12]. The function is usually constructed by a classifier, which can predict data that is not in the training set [21]. Wan et al. [24] proposed the Visual Center Adaptation Method (VCAM) method to solve the domain-shift problem in FSL when mapping images to the semantic space. Berkan et al. [6] changed the traditional FSL method only to learn the class's semantic labelling method and increased the visual label of the class so that the label and the class had a higher-dimensional similarity. Jero et al. [9] applied the ZSL framework to the RFC data set for protocol syntax extraction. The ZSL framework obviated the need for building a classifier for each protocol and it made the extraction process more automated. Besides, when training a small number of samples, new entities that were not in the training set could be extracted from the new RFC.

C. Entity extraction

This article involves the application of SVM [4], TextCNN[11] and AttBi-LSTM[30] when extracting network protocol entities. Chiu et al. [5] proposed novel neural network architecture. The architecture uses a two-way LSTM and CNN hybrid architecture to automatically detect the word and character-level features. The named entity recognition task on the CoNLL-2003 data set has more outstanding performance than the latest. Zeng et al. [29] used two-way long short-term memory (LSTM) and conditional random field decoding (LSTM-CRF) recurrent neural networks to do the task of identifying drug names on the DDI2011 and DDI2013 data sets and achieved the best results at the time. Miwa et al. [19] proposed a novel end-to-end neural model. The model uses bidirectional LSTM-RNN and bidirectional tree LSTM-RNN to capture word sequences and dependency tree substructure information. It was tested on the data set SemEval-2010 Task 8, and the results were significantly improved compared to the latest model.

III. Methodology

In this section, we will introduce the details of the model. Our goal is to apply the model to extract the protocol entities on a large unlabeled corpus under the premise of training small samples. We first select a small amount of RFC documents as samples. After deleting headers and footers, and charts of each sample. We use the NLP tool in the CoreNLP package [17] to split the text into chunks. Each chunk represents a grammatical phrase, which may be a noun phrase or a verb phrase. In the first stage, we selected N RFC documents as the training set and M RFC documents as the test set. The training set text is divided into chunk sets, which are defined as $C_{train} = \{c_1, c_2, \cdots, c_n\}$, and the test set chunk sets are defined as $C_{test} = \{c_1, c_2, \cdots, c_k\}$. In the meanwhile, we use a rule-based method to extract the protocol entity in the training set. The entity set is defined as $E = \{e_1, e_2, \cdots, e_m\}$. We apply the model to a new RFC, and we first preprocess it into chunks. The chunk set for the new RFC is defined as $C_N = \{c_1, c_2, \cdots, c_q\}$. Our model filters out chunks representing entities in C_N, and the collection of entity chunks serves as the protocol entity for the new RFC document. Our model consists of two stages:

1) **Potential entity mining** SVM learns the similarity function $S_1(c_i, e_j)$ from the text chunk C_{train} and the entity set E to measure the likelihood that the chunk $c \in C_{train}$ contains the entity $e_j \in E$. Then it applies the similarity function S_1 to the C_{test} set. We mine potential entity reference chunks from C_{test}, and define it as C_{entity}, as the extracted entity in the first stage.
2) **Entity precise identification** We extract the entity result set E_{result} from the RFC documents corresponding to C_{test}. After comparing E_{result} with C_{entity}, C_{entity} is divided into positive and negative samples. Deep learning model learns the scoring function $S_2(P, N)$ from positive and negative samples, which judges the probability that the chunk is a positive or negative sample.

Fig. 2 shows the overall process of the cascade entity extraction model. First, according to the above two stages, the similarity function S_1 and the prediction function S_2 are generated from the training samples. Enter a new RFC document, the text is preprocessed and converted into a chunk set C_N. In the potential entity mining stage, the S_1 function is used to mine the chunk set E_{S1} in C_N that is similar to the entity characteristics. In the precise entity recognition stage, first use part of speech noise reduction to improve the accuracy of the method. The S_2 function extracts the positive sample set E_{S2} from E_{S1}, and E_{S2} is the network protocol entity set extracted from the new RFC document. In the first stage, the method excavates text chunks with entity characteristics as much as possible, and in the second stage, it further identifies whether the text chunks extracted in the first stage are valid values and recognize the truth labels in them. The method not only guarantees the recall rate of entity extraction in the first stage, but also improves the accuracy of extraction.

A. Potential entity mining

The details of the potential entity mining process in the first stage are shown in Fig. 3. The model's input is each word in a chunk, which is converted into a vector by TF-IDF, and then input into the SVM classifier for classification. The output is the label corresponding to this chunk. The chunk input in the figure is ”the sequence number the sender,” which is the entity reference of the ”Sequence Number” in the result entity set, so it is marked as a positive label. After all the chunks are classified by the classifier, the positive label set is used as the next stage input.

Fig. 4 shows the process in the first stage of training. Specifically, we define a binary classification problem on all (c_i, e_j), c_i is a chunk in the text, and e_j is an entity in the result set. Taking the two text chunks as an example, the entity set Entities does not contain the sub-chunk”communicates”, so this chunk is marked as a negative sample. In the entity set, an entity reference contains the chunk ”the sequence number”, which is marked as a positive sample, and so on. The same judgment is made on all training set chunks, and each chunk is labeled with positive and negative. From this, we can learn the similarity function $S_1(c_i, e_j)$ between c_i and e_j, and we trained an SVM classifier with a set of two-class features.

B. Entity precise identification

In order to learn more entity features, we expand the sample set. After expanding the sample, it is found that the accuracy of the entities extracted from the first stage is significantly reduced. In order to analyze the reasons for the decrease in accuracy, we compare the result set with the extracted entity and divide the extracted entity set into positive and negative samples. The specific operations are as follows: The ”**Chunked Text**” in the left part of Fig. 4 is replaced with the extracted entity set in stage one, the ”**Entities**” in the right part is replaced with the resulting entity set, and the middle part makes the same judgment. The extracted entities containing any entity in the result set are taken as positive samples, and the negative samples are taken otherwise. The goal of entity precise identification in the second stage is to

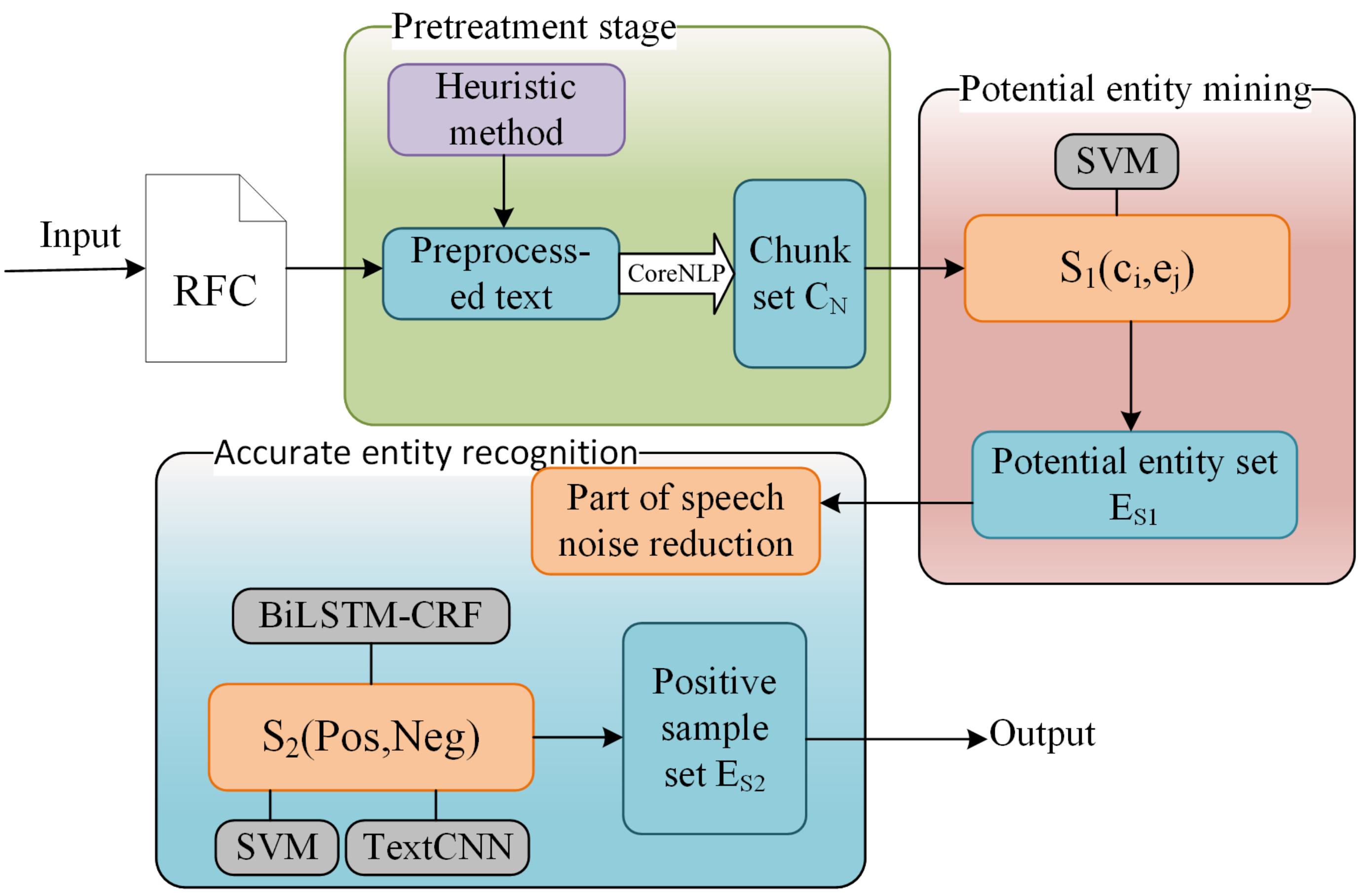

Fig. 2. An overview of network protocol entity extraction framework based on small sample learning. C_N is the chunk set after processing a new RFC document E_{S1} is the set of potential entities extracted from C_N in the first stage, and E_{S2} is the final entity set.

reduce the number of negative samples while maintaining the number of positive samples.

By analyzing the structure of positive and negative samples, it is found that the chunks in most negative samples contain verb phrases or adverbs, but the positive samples do not.

Part-of-speech tags	Meaning
RB	Adverb
TO	Infinitive
VBZ	Singular verb
UH	Interjection
CD	Quantifier
MD	Modal verbs
IN	Preposition
VBG	Verb noun
CC	Conditional conjunction
VBP	Non third person singular
VB	Verb prototype
POS	Possessive

Table I: Part of speech included in negative samples. The first column is the part-of-speech tag in the NLP tool; the second column is the corresponding interpretation.

So consider using part of speech to delete some negative samples In this experiment, the NLP tool in the CoreNLP package is used to extract the parts of speech in the chunks, and the chunks containing the parts of speech in the following table I are deleted.

After applying part-of-speech noise reduction to the entity set, half of the chunks in the original negative sample are deleted, while the number of positive samples does not change. The remaining negative samples are nominal phrases consistent with the positive sample, which cannot be deleted using part-of-speech.

In this experiment, SVM, TextCNN, and AttBi-LSTM are used as classifiers, and they all have different performance in classification performance. Take the positive and negative samples processed with part of speech as input. For deep learning models, the word2vec model [18] is used for word vector conversion, and for SVM, TF-IDF is used for vector conversion. The training classifier learns the score function $S_2(Pos, Neg)$, which calculates the probability that the chunk is a positive sample. Applying the S_2 function to the new chunk set can accurately predict the positive samples in it, and select the positive samples as the final entity set, thereby further reducing the number of negative samples. Taking AttBi-LSTM as an example, as shown in Fig. 5, the input is a chunk remaining after the part-of-speech denoising of the extracted entity in the first stage, which is composed of 5

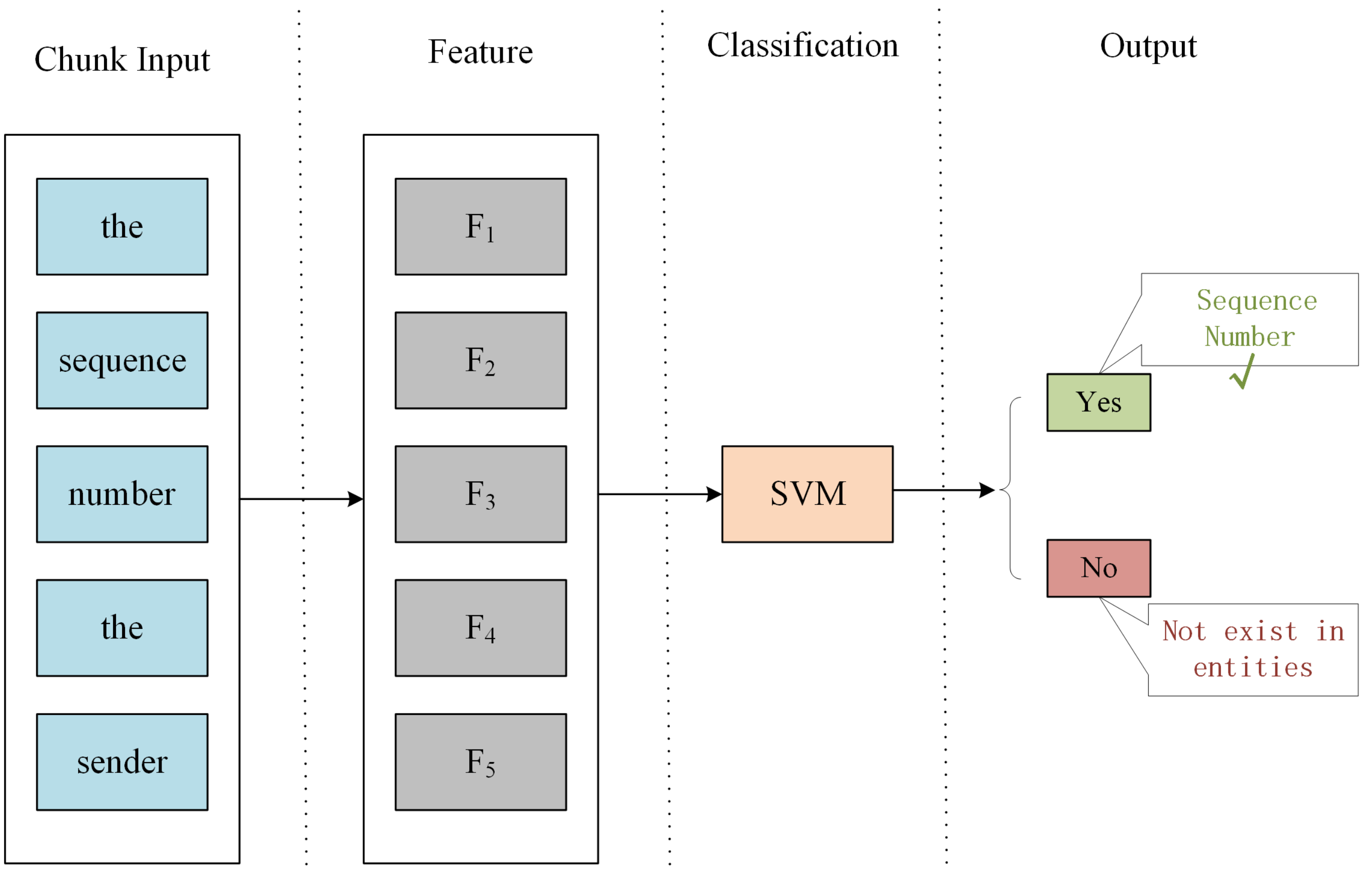

Fig. 3. The first phase of the model architecture. The input is each word in a chunk, and the output is the classification label for this chunk. Yes indicates that the input chunk contains a reference to an entity, and No indicates that the input chunk does not exist in the entity set.

words. After being classified by AttBi-LSTM, the output is the positive and negative labels corresponding to this chunk. The vector u in the figure represents the word's importance,, and a_t is used to normalize the word weight. The sum of all information in sentence v is the weighted sum of each h_t (formula 1), where a_t is the corresponding weight [26].

$$v = \sum_t a_t \overrightarrow{h_t} \tag{1}$$

IV. Experiment Setup

A. research problem

We use the following three questions to verify the method's effectiveness from three aspects: effectiveness, robustness, and model design.

RQ1: How effective is the two-stage entity extraction method?

RQ2: Does the network protocol entity extraction method based on small sample learning perform stably on large unlabeled test sets?

RQ3: Is there an optimal model design for the network protocol entity extraction method based on small sample learning?

B. Data

We choose RFC documents as our experiment data, and the data set is constructed as follows:

1) Data preparation: Firstly, we select RFC documents with two specific wordings: 1) The header field and its size are on the same line, separated by a colon, e.g. (Type of Service: 8 bits). 2) The detailed description of the header field is directly below the area. The upper box in Fig. 6 is an example that meets the above characteristics. The box below is an independent text. It is not possible to extract entities by heuristic methods. It is necessary to use a small sample-based network protocol entity method to extract. Among them, "the ACK control bit" and "sequence number" are header fields. Then, heuristic-based methods are used to extract entities from RFC documents that meet the above conditions, and the entities are classified in the result set. There are 8672 RFC documents, and there are 906 documents with the above-mentioned structural features. Therefore, it is impossible to extract entities from all RFC documents by heuristic methods.
2) Preprocess: We download the RFC texts from the official RFC website and then filter out the RFC documents with characteristics in step one. In this experiment, protocol

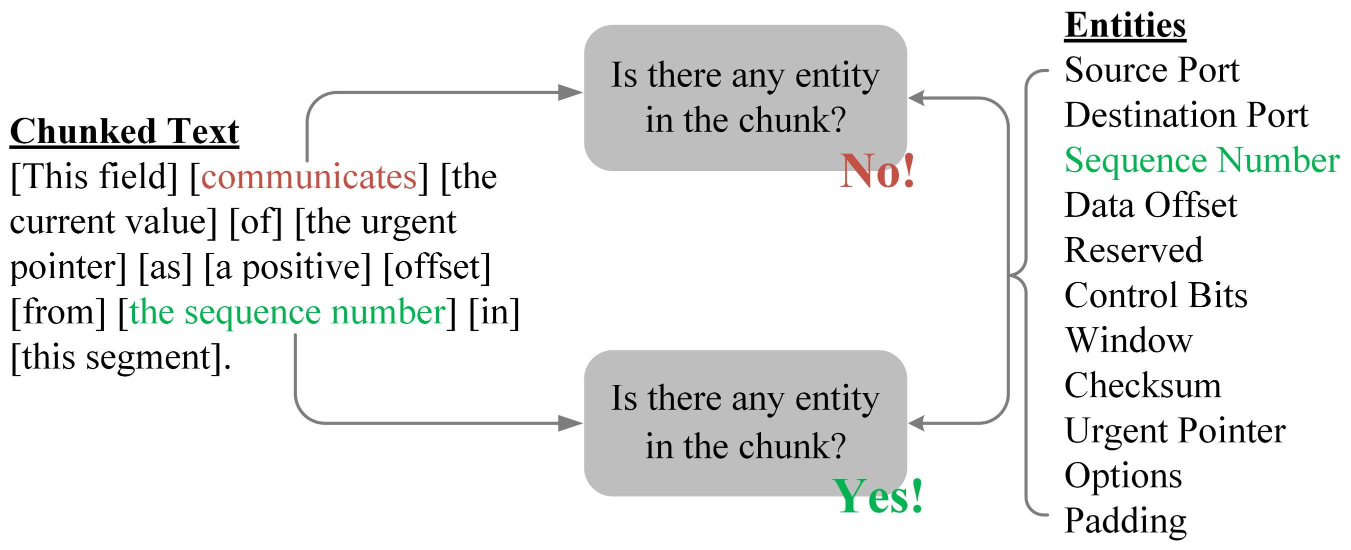
Chunked Text
[This field] [communicates] [the current value] [of] [the urgent pointer] [as] [a positive] [offset] [from] [the sequence number] [in] [this segment].
Is there any entity in the chunk?
No!
Is there any entity in the chunk?
Yes!
Entities
Source Port
Destination Port
Sequence Number
Data Offset
Reserved
Control Bits
Window
Checksum
Urgent Pointer
Options
Padding

Fig. 4. Example of dividing positive and negative samples.

entities are extracted from the RFC texts, so information that is not related to the text content should be deleted. Firstly, we delete the page header and page footer in the text. This part of the writing is fixed and can be removed by pattern matching. Secondly, we delete the charts in the document. Most charts are surrounded by the symbol ”+-” or other special characters. Therefore, we identify the position of the symbol in the text, and then delete the symbol from this position. Considering that the word sparsity in each line containing the chart is relatively low, we specify a threshold and delete symbols in a line until the word sparsity is higher than the threshold.

In the next step, We cut the text into chunks. The experiment's purpose is to extract protocol entities, so only the entity description text is segmented. The description part immediately follows the header and can be extracted using a rule-based method. For each sample RFC, the extracted description text is saved separately for subsequent segmentation. This experiment uses the NLP tool in the CoreNLP package to achieve text segmentation. The NLP tool can convert each sentence in the text into a grammar tree structure, and each sentence can be divided into multiple grammar phrases according to the grammar tree, as shown in Fig. 7.

3) Entity extraction: The first-stage and second-stage classifiers require the RFC document's protocol entity as input, so we extract the protocol entity from the sample documents in advance. The extracted documents have the same rules in the writing of the header. We use regular expressions to extract the header part. The comparison finds that extracting the entity based on the rule is the same as the actual result, so the extracted entity set is used as the result set of the sample document. Fig. 8 is an example of the entity types extracted by this method from an RFC in our dataset.

C. Evaluation Metrics

We use Precision, Recall, and F1 values as evaluation indicators. Taking RFC document a as an example, after comparing the entity set extracted by our model with the result set, the number of correctly extracted entities is $TP(a)$; the number of wrongly extracted entities is $FP(a)$. The number of unrecognized entities in the result set of document a is $FN(a)$.

Because entity writing is not uniform, and the extracted entities are grammatical phrases, which include text noise. It is impossible to accurately determine whether the extracted entities include result set entities directly using the character matching method. In this experiment, we convert the extraction entity and the result set entity to lowercase and divide the extraction entity into multiple words by spaces. Then determine whether each word in the extracted entity is in the result set entity. There will be some noise in word matching, such as ”it”, ”number”, ”data” and other common words. When we encounter such a word in the matching process, skip this match. We use the following formula to calculate the above three values:

$$Precison = \frac{TP(a)}{TP(a) + FP(a)} \tag{2}$$

$$Recall = \frac{TP(a)}{TP(a) + FN(a)} \tag{3}$$

$$F1 = \frac{2TP(a)}{2TP(a) + FP(a) + FN(a)} \tag{4}$$

D. Baseline

In this experiment, Jero et al. [9] used the ZSL framework to extract entities as the baseline of the evaluation method. For a fair comparison, we select the RFC files used in Jero et al. 's

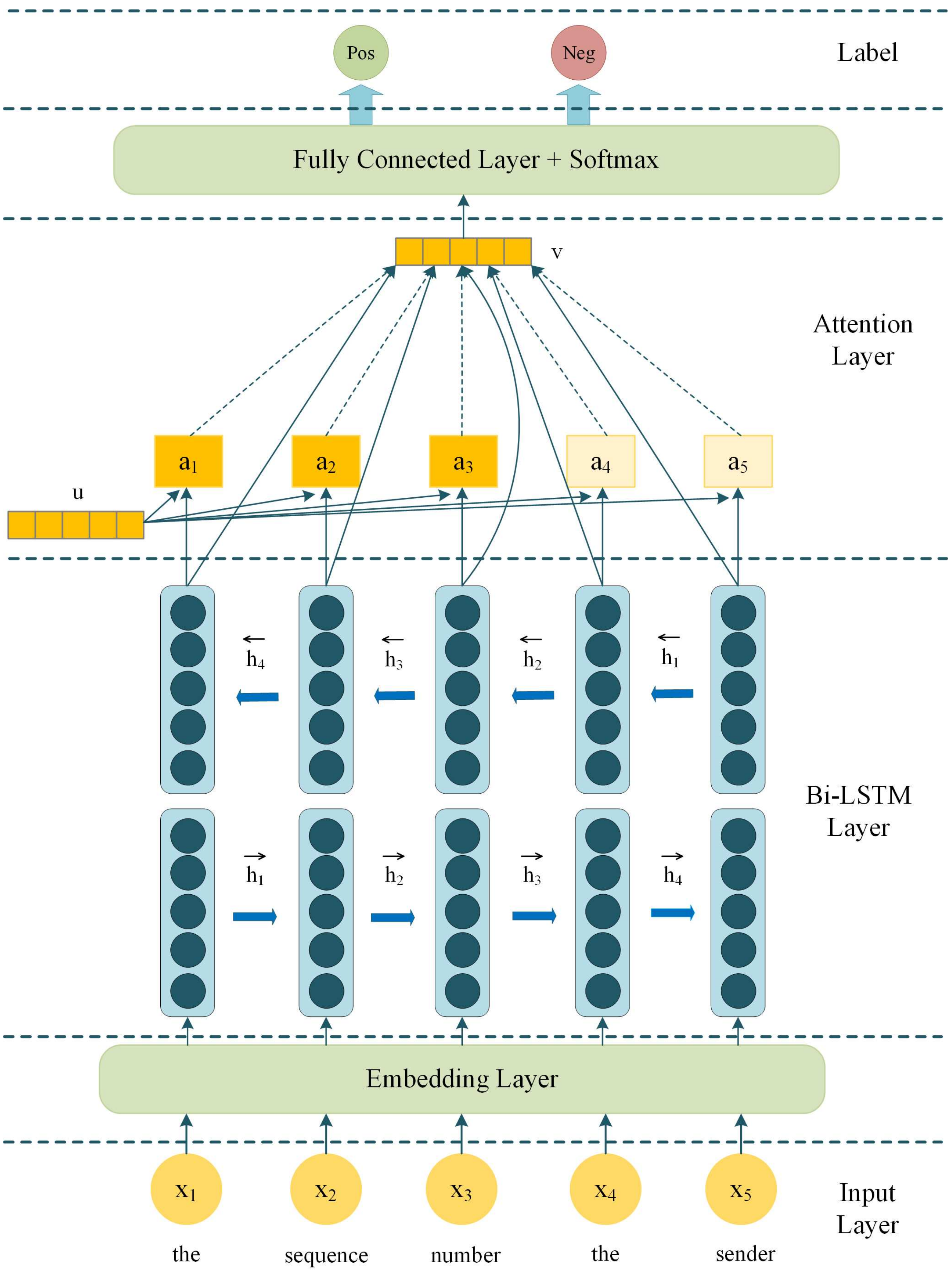

Fig. 5. AttBi-LSTM network protocol entity extraction model architecture. In the output layer, Pos is a positive label, and Neg is a negative label.

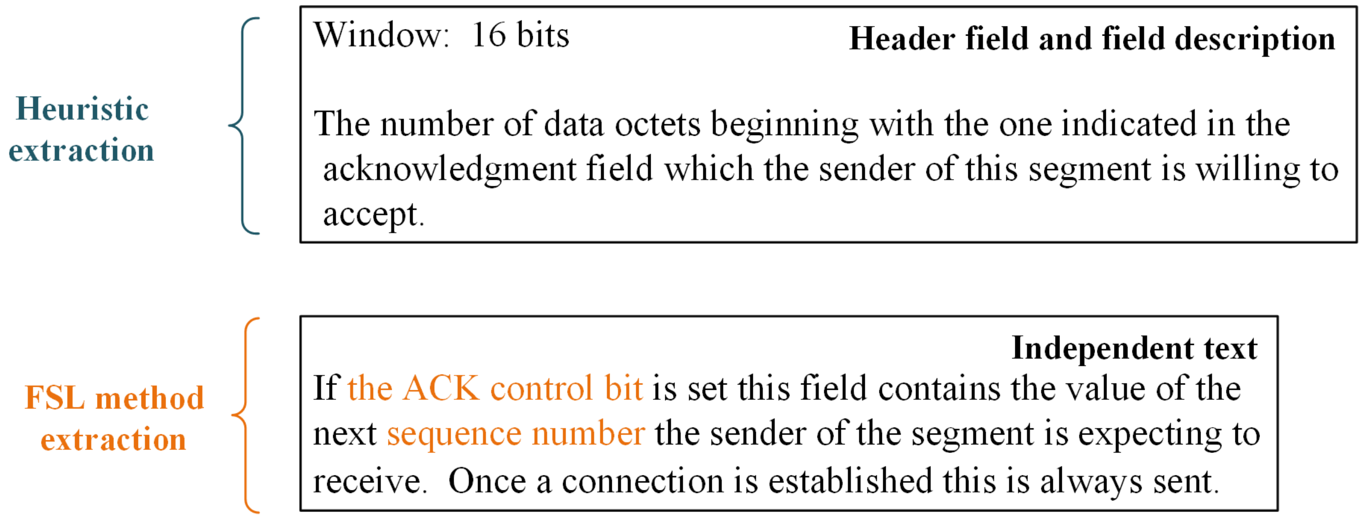
Heuristic extraction
Window: 16 bits
Header field and field description
The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.
FSL method extraction
Independent text
If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent.

Fig. 6. RFC text corresponding to different extraction methods.

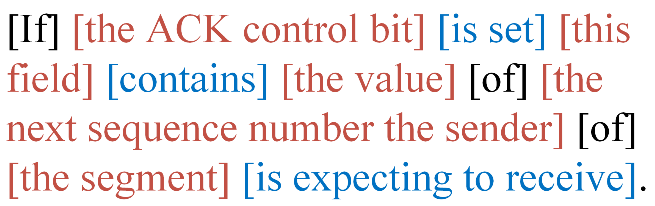
[If] [the ACK control bit] [is set] [this field] [contains] [the value] [of] [the next sequence number the sender] [of] [the segment] [is expecting to receive].

Fig. 7. Text chunk example.The red chunks correspond to noun phrases, the blue chunks correspond to verb phrases, and the other chunks correspond to phrases such as prepositions.

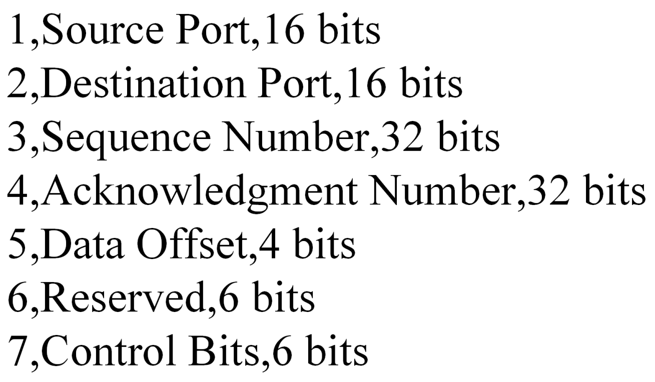
1,Source Port,16 bits
2,Destination Port,16 bits
3,Sequence Number,32 bits
4,Acknowledgment Number,32 bits
5,Data Offset,4 bits
6,Reserved,6 bits
7,Control Bits,6 bits

Fig. 8. Entity extraction example.

experiment as the samples. The file's corresponding protocols are GRE, IPV6, IP, DCCP, and SCTP, and then tested on the TCP protocol. After comparing the same samples, expand the sample set capacity to 20, and test on the TCP protocol to check the method's robustness.

The ZSL framework uses SVM as the classifier, the kernel function of SVM is set to "rbf", the kernel function parameter gamma is set to 10, and the penalty parameter C is set to 0.2, which enhances its generalization ability. Due to the small data set, overfitting needs to be avoided.

E. Implementation Details

Our experiment uses SVM as the classifier in the first stage, and its parameter settings are the same as the baseline. In the second stage, we use SVM, TextCNN or BiLSTM-CRF as classifiers. The parameters of the SVM classifier are consistent with the first stage. As for TextCNN or BiLSTM-CRF, before running the model, pre-train word vectors with word2vec, and the training data set is the text set of all sample RFCs. The word vector for each word is set to 100 dimensions. TextCNN uses three convolution kernels $(3, 4, 5)$ to extract features, and then the features extracted by different convolution kernels are stitched together in the maximum pooling as a feature vector input to softmax. When training the BiLSTM-CRF classifier, in order to prevent the over-fitting phenomenon in the training process, the dropout mechanism was introduced, and the dropout discarding rate was set to 0.5 [27].

Through extensive analysis of RFC entities, in the first stage, we select 5 RFCs as test sets, namely: RFC760, RFC761, RFC791, RFC1883, RFC1889, which provide data for the construction of positive and negative samples in the second stage.

V. Result and Analysis

We conduct two sets of comparative experiments. One group is the comparison between our method and the baseline method on different samples. The other group is a comparison between the three methods we used on different samples.

A. RQ1. Method effectiveness

For the 5 sample size experiments, we perform 6 iterations, use 5 samples for training, and test on the 6th sample. For experiments with 20 sample sizes, we use the above 6 sample tests as the test set for a fair comparison. Table II and Table III

Sample Size	Model	Precision	Recall	F1
20	SVM(Baseline)	29.2%	83.7%	43.3%
	AttBi-LSTM	36.9%	92.3%	52.7%
	SVM+AttBi-LSTM	**69.2%**	**77.2%**	**72.9%**
5	SVM(Baseline)	80.2%	53.5%	64.1%
	AttBi-LSTM	76.1%	54.7%	63.7%
	SVM+AttBi-LSTM	**88.4%**	**58.5%**	**70.4%**

Table II: Ablation study on the effectiveness of two-stage method.

Sample Size	Model	Precision	Recall	F1
20	SVM+SVM	61.3%	74.6%	67.3%
	SVM+TextCNN	68.2%	77.0%	72.2%
	SVM+Attbi-LSTM	**69.2%**	**77.2%**	**72.9%**
5	SVM+SVM	81.1%	60.0%	68.9%
	SVM+TextCNN	87.9%	58.4%	70.1%
	SVM+Attbi-LSTM	**88.4%**	**58.5%**	**70.4%**

Table III: Horizontal comparison of methods on different sample sizes. Compare the performance of the three methods we used on different sample sizes.

show the summary results of iterations under different sample sizes.

According to the information in Table II, both baseline and AttBi-LSTM are single-stage entity extraction methods, and our approach is a two-stage entity extraction method. When using 20 samples for training, our method's Precision value is higher than that of the single-stage entity extraction method, and the Recall value is slightly lower than that of the single-stage entity extraction method. The F1 value of our approach is much higher than the comparison method. When using 5 samples to train the model, the Precision value and Recall value of our method are higher than the single-stage entity extraction method, and the overall F1 value is also higher than the comparison method. After the single-stage entity extraction method expands the sample size, the Precision value decreases significantly, the Recall value increases, and the overall F1 value drops significantly. After our method expands the sample size, the Precision value decreases, the Recall value increases, and the overall F1 is basically unchanged.

According to the statistics in Table III, when using 20 samples for training, using AttBi-LSTM as the second stage of the classifier works best. When using 5 samples for training, the second stage uses SVM as the classifier with the highest Recall value but the lowest Precision value. AttBi-LSTM has the highest accuracy as a classifier, and the F1 value is also the highest. It can be inferred that using AttBi-LSTM as the second stage classifier can improve the method's performance. The effect of TextCNN as a classifier is basically the same as that of AttBi-LSTM, only the Precision value is slightly lower. In our three methods, after expanding the sample, the Precision value decreases, while the Recall value increases, and the F1 value remains unchanged.

According to the experimental results in Table II and Table III, we can observe:

1) On a small sample, the baseline method results are not much different from the results of our cascade model, but we can still find that our method is higher than the baseline in the accuracy. The reason is that the accurate identification of entities in the second stage is to further divide the entities extracted by the baseline method, extract the positive samples, and reduce the number of negative samples in the extracted entities, thereby the Precision value is improved. The reason for the decrease in the recall rate is that there are few sample data, which leads to the low accuracy of the positive sample model in the second stage, and there are cases of wrong scores.
2) After expanding the number of samples, the experimental results show that the baseline method can extract almost all of the result set entities, but its accuracy will be significantly reduced, and the F1 value will also be reduced. The reason is that after increasing the sample size, the entity features the model can learn have increased, and more accurate entities can be extracted from the unknown RFC documents. However, at the same time, more entities similar to the sample result set will be extracted. These entities are not in the result set of the unknown RFC, which reduces extraction accuracy. Expanding the sample size is to apply the algorithm to more unknown RFCs and extract entities from them accurately. However, the baseline method has a significant change in accuracy, which is not suitable for promoting large sample sets.
3) After expanding the sample set, our method's accuracy is reduced, the Recall value is increased, and the F1 value is unchanged. Compared with the baseline method, the accuracy is doubled, the Recall value drops slightly, the F1 value is much higher than the baseline. Our model's overall performance is better than the baseline method. Above proves that our method can accommodate more sample data and extend to more unknown RFC documents with high extraction accuracy.
4) Our model still performs well with only 20 training samples. It can be found from the experimental results that the model using the ZSL framework can achieve good extraction results on the test set after training on small samples. To further improve the stability of the ZSL framework, our model adds precise identification of entities in mining potential entities, and the experimental results verify the effectiveness of our model.

B. *RQ2. Robustness verification*

To further verify our model's robustness, besides testing on the protocol TCP (RFC793), we also select multiple other protocols for testing to avoid accidental events. Fig. 9 shows the average performance achieved by different cross-validation methods, and Figure 11 shows the detailed performance of each method. We select 20 RFCs as samples, and the training set and test set are divided by 8:2. The test sets are RFC793, RFC3332, RFC3451, and RFC5793. It can be found from Fig. 9 that the accuracy and F1 value of the two-stage method are higher than the baseline, and the Recall value is lower than the baseline, but the baseline accuracy value is too low, so it is not applicable. According to Fig. 10, we comprehensively compare the three methods we selected. In the second stage,

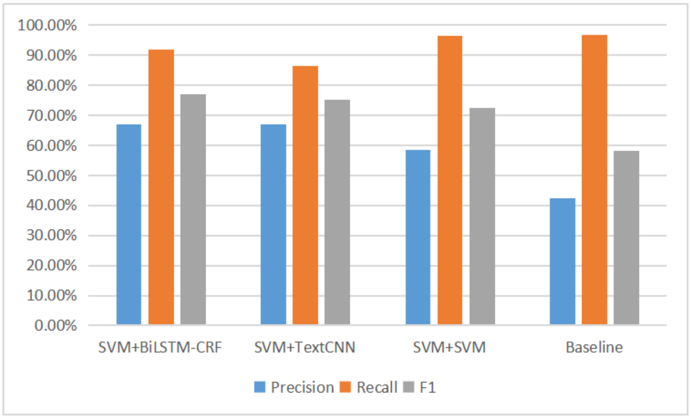

Fig. 9. The average performance in Cross-validation.

using AttBi-LSTM as the classifier has the best effect. The average scores in accuracy, recall, and F1 value is 66.9%, 91.9%, and 77.0%, respectively. While ensuring the recall rate, the extraction accuracy makes the F1 value the highest. Next is the TextCNN classifier, and finally, the SVM classifier. Experiments show that our method has strong stability and high robustness and can be widely used in other unlabeled RFCs.

C. RQ3. Optimal design of the model

In order to verify the optimality of the combined effect of the two-stage model, we select different combined models to compare. In the potential entity mining stage, we replace the original SVM classifier with the AttBi-LSTM classifier. In the stage of precise entity identification, we choose TextCNN or AttBi-LSTM as the classifier. According to our previous experimental results, the SVM+AttBi-LSTM combined model has the best classification effect. So it is chosen as the experimental baseline. From the sub-figure (a) in Fig. 11, we can find that our baseline has the lowest accuracy fluctuation range and high accuracy value on different RFCs. From the sub-figure (b), it finds that on different RFCs, the recall value of the baseline method is higher than the other two combined models. Therefore, the F1 value obtained by our baseline on different RFCs is the highest, and the performance of the model is the best. There are two main reasons why our model is better than the comparison models:

1) The first stage of the combined model for comparison uses AttBi-LSTM as the classifier. The sample feature is the word vector converted by word2vec in text chunks. In the second stage, TextCNN or AttBi-LSTM is used as the classifier. The sample feature is still the word vector of the text chunks. Due to repeated training samples, the two-stage model can be used separately to obtain good results, but the combined effect is not good.
2) Our baseline method uses SVM as the classifier in the first stage. The sample feature is the cosine similarity value between the text chunk and the entity. In the second stage, AttBi-LSTM is used as the classifier. The sample feature is the word vector of the text chunks. So that the two classifiers can complement each other, thereby improving the overall classification performance.

Using our model to extract entities on the RFCs is a process from coarse-grained mining to fine-grained recognition. The effect of SVM classifier alone is not as good as AttBi-LSTM or TextCNN, but it can be used in the first stage of our model to mine more potential entities. Using AttBi-LSTM in the second stage will make up for the loss of accuracy in the previous stage, thereby improving the performance of the entire model. Therefore, the model combination method we choose is the best.

VI. Conclusion

This paper proposes a network protocol entity extraction method based on small sample learning. This method uses a small number of labeled RFC document samples for training, which can achieve protocol entity extraction from unlabeled RFC documents and maintain high recognition accuracy. Experimental results show that our method is significantly better than the baseline method. The average precision, average recall, and average F1 value are 69.2%, 77.2%, and 72.9%, respectively.

We extracted 20 RFC documents from 906 RFC documents containing the protocol entities' structural features as experimental samples and used the entity extraction model trained with 20 RFCs as samples to extract on 906 RFCs. We are the first team to extract protocol entities on the RFC and publish the entity set results. The extracted entities are stored in the CSV file in ascending order of RFC serial number. The first column in the file is the RFC name, and the second column is the extracted entity. We publish the documents of the extracted entities on GitHub (https://github.com/czycurefun/RFCdocument) for researchers to continue to improve in the future. We will continue to optimize the model to obtain higher extraction accuracy and recall rate in future work. And use the extracted entities to construct a knowledge graph of network protocols and better analyze unlabeled protocols.

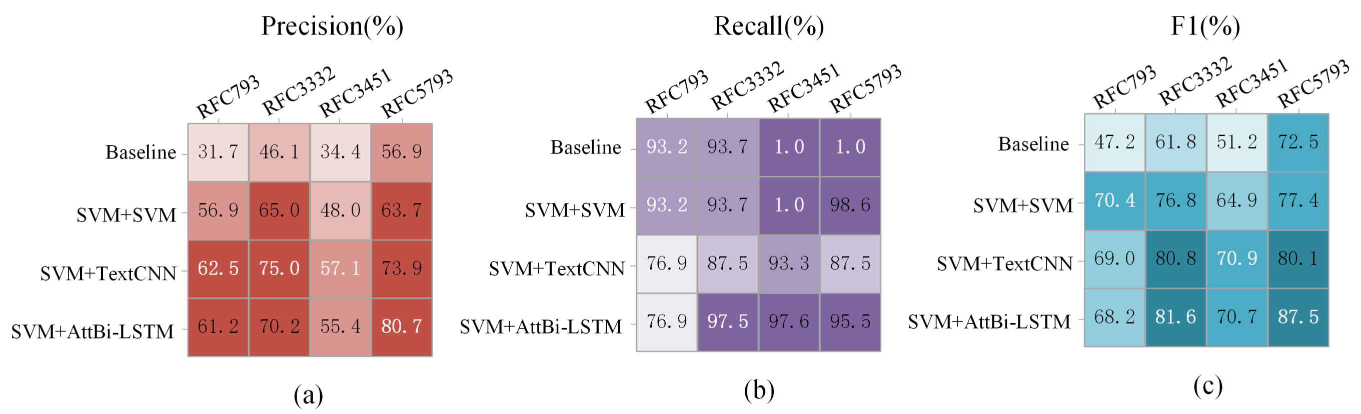

Fig. 10. Various index values on different RFCs. The indicator values tested on four RFCs are displayed, and the sub-graphs a, b, and c correspond to the Precision, Recall, and F1 values, respectively. The white values correspond to the best method for each RFC.

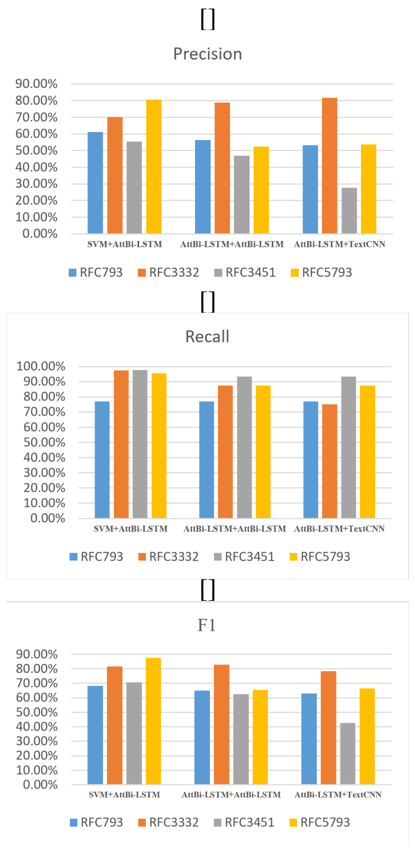

Fig. 11. Comparison of different combination models

References

[1] Linda Yunlu Bai, Yongfeng Huang, Guannan Hou, and Bo Xiao. 2008. Covert channels based on jitter field of the rtcp header. In2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pages 1388–1391. IEEE.

[2] Maria Carla Calzarossa and Luisa Massari. 2014. Analysis of header usage patterns of http request messages. In2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), pages 847–853. IEEE.

[3] Liang Chen, Jian Gong, and Xuan Xu. 2007. A survey of application-level protocol identification algorithm. Computer science, 34(7):73–75.

[4] Pai-Hsuen Chen, Chih-Jen Lin, and Bernhard Schölkopf. 2005. A tutorial on ν-support vector machines. Applied Stochastic Models in Business and Industry, 21(2):111–136.

[5] Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. Transactions of the Association for Computational Linguistics, 4:357–370.

[6] Berkan Demirel, Ramazan Gokberk Cinbis, and Nazli Ikizler-Cinbis. 2019. Learning visually consistent label embeddings for zero-shot learning. In2019 IEEE International Conference on Image Processing (ICIP), pages 3656–3660. IEEE.

[7] Mrudul Dixit, Anuja Kale, Madhavi Narote, Sneha Talwalkar, and BV Barbadekar. 2012. Fast packet classification algorithms. International Journal of Computer Theory and Engineering, 4(6):1030.

[8] Siva Gurusamy, D Manjula, and TV Geetha. 2002. Text mining in'request for comments document series'. InLanguage Engineering Conference, 2002. Proceedings, pages 147–155. IEEE.

[9] Samuel Jero, Maria Leonor Pacheco, Dan Goldwasser, and Cristina Nita-Rotaru. 2019. Leveraging textual specifications for grammar-based fuzzing of network protocols. InProceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 9478–9483.

[10] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2020. A survey on knowledge graphs: Representation, acquisition and applications. ArXiv, abs/2002.00388.

[11] Yoon Kim. 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

[12] Elyor Kodirov, Tao Xiang, and Shaogang Gong. 2017. Semantic autoencoder for zero-shot learning. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3174–3183.

[13] Maxime Labonne, Alexis Olivereau, Baptise Polve, and Djamal Zeghlache. 2020. Unsupervised protocol-based intrusion detection for real-world networks. In2020 International Conference on Computing, Networking and Communications (ICNC), pages 299–303. IEEE.

[14] Wei-Ming Li, Ai-Fang Zhang, Jian-Cai Liu, and Zhi-Tang Li. 2011. An automatic network protocol fuzz testing and vulnerability discovering method. Jisuanji Xuebao(Chinese Journal of Computers), 34(2):242–255.

[15] Colin Lockard, Xin Luna Dong, Arash Einolghozati, and Prashant Shiralkar. 2018. Ceres: Distantly supervised relation extraction from the semi-structured web. arXiv preprint arXiv:1804.04635.

[16] Kangqi Luo, Jinyi Lu, Kenny Q Zhu, Weiguo Gao, Jia Wei, and Meizhuo Zhang. 2019. Layout-aware information extraction from semi-structured medical images. Computers in biology and medicine, 107:235–247.

[17] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. InProceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations, pages 55–60.

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[19] Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. arXiv preprint arXiv:1601.00770.

[20] Masayuki Ohta, Yoshiki Kanda, Kensuke Fukuda, and Toshiharu Sugawara. 2011. Analysis of spoofed ip traffic using time-to-live and identification fields in ip headers. In2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications, pages 355–361. IEEE.

[21] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. 2009. Zero-shot learning with semantic output codes. InAdvances in neural information processing systems, pages 1410–1418.

[22] Yun R Qu, Hao H Zhang, Shijie Zhou, and Viktor K Prasanna. 2015. Optimizing many-field packet classification on fpga, multi-core general purpose processor, and gpu. In2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), pages 87–98. IEEE.

[23] Matteo Varvello, Rafael Laufer, Feixiong Zhang, and TV Lakshman. 2015. Multilayer packet classification with graphics processing units. IEEE/ACM Transactions on Networking, 24(5):2728–2741.

[24] Ziyu Wan, Yan Li, Min Yang, and Junge Zhang. 2019. Transductive zero-shot learning via visual center adaptation. InProceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 10059–10060.

[25] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. ACM Computing Surveys (CSUR), 53(3):1–34.

[26] Genta Indra Winata, Onno Pepijn Kampman, and Pascale Fung. 2018. Attention-based lstm for psychological

stress detection from spoken language using distant supervision. In2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 6204–6208. IEEE.

[27] Guixian Xu, Yueting Meng, Xiaoyu Qiu, Ziheng Yu, and Xu Wu. 2019. Sentiment analysis of comment texts based on bilstm. Ieee Access, 7:51522–51532.

[28] Tianling Xu, Kaiguo Yuan, Jingzhong Wang, Xinxin Niu, and Yixian Yang. 2009. A real-time information hiding algorithm based on http protocol. In2009 IEEE International Conference on Network Infrastructure and Digital Content, pages 618–622. IEEE.

[29] Donghuo Zeng, Chengjie Sun, Lei Lin, and Bingquan Liu. 2017. Lstm-crf for drug-named entity recognition. Entropy, 19(6):283.

[30] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. InProceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 207–212.