A distributed event extraction framework for large-scale unstructured text

Zhigang Kan, Haibo Mi, Sen Yang, Linbo Qiao‡, Dawei Feng‡, Dongsheng Li‡
College of Computer, National University of Defense Technology ChangSha 410073, China
kanzhigang5206@163.com, haibo_mihb@126.com, {yangsen.nudt, linboqiao, davyfeng.c}@gmail.com,
lds1201@163.com

***Abstract*—Event extraction is an important subtask of information extraction. The goal of event extraction is to quickly extract events of a specified type from a large amount of textual information. Many excellent models and algorithms have been proposed since ACE released the event extraction task in 2005. Most of them are based on the dataset published by ACE and have contributed to the accuracy of event extraction to a certain extent. In real-world applications, the processing object of the event extraction task is large-scale text data. However, as far as we know, there is currently no adequate model for using multiple computers for event extraction. In this paper, we propose a framework for event extraction based on inter-cloud computing technology, which aims to extract events from huge-amount of unstructured text data in the wild. The experimental results demonstrate that our method could improve the throughput, reduce time consumption of the event extraction process, and further gets better accuracy than advanced models.**

***Index Terms*—event extraction, massive data, inter-cloud**

I. Introduction

The rapid development of network technology has brought great convenience to human society. In particular, the emergence of the World Wide Web has provided a fast and straightforward way for humans to exchange information and understand current events. With the widespread popularity of mobile terminals, the amount of data generated on the Internet every day is tremendous. Humans need machines to process these massive amounts of data. However, much of this information is text data composed of human language, which is unstructured and complicated for machines to understand. Many data mining techniques and algorithms have been proposed to obtain information from text data. Among these technologies, one of the most effective ways to transform unstructured information into structured information is information extraction(IE).

Information extraction is a text processing technology that extracts specified types of entities, relationships, events, and other information from natural language text and forms structured data output. Event extraction(EE) is one of the main tasks of information extraction [1]. ACE defined it in 2005. The purpose of event extraction is to determine the type of event and automatically extract arguments with different roles from unstructured text.

Early event extraction systems were based on pattern-matching methods, relying on artificially written rules about syntactic and semantic constraints. When using these methods for event extraction, researchers only need to find information that meets the constraints of the pattern through various pattern matching algorithms. These pattern matching methods can achieve excellent results in specific fields, but the system is not portable. When the system is applied from one domain to another, the model needs to be rebuilt.

With the development of machine learning, researchers have borrowed ideas from text classification and tried to use machine learning algorithms for event extraction [2]. The machine learning-based event extraction systems do not depend on the field of the corpus and the form of documents. These systems are portable to a certain extent. The core problems of this type of EE system are the construction of the classifier and the representation of text features. The training of classifier requires a large amount of labeled data. In addition, triggers of events only account for a small part of the text, which leads to an imbalance in the proportion of positive and negative samples during the training process. Therefore, event extraction systems based on machine learning face the problem of sparse data. The text feature representation of such systems usually relies on traditional external NLP tools such as dependency analysis, syntax analysis, and part-of-speech tagging, which will cause error accumulation.

The emergence of neural networks has brought new ways of thinking to the field of event extraction. The neural network-based event extraction systems use word vectors as input. More abstract features can be learned through different neural network models [3]. Therefore, the feature representation of words can contain more semantic information. Besides, this type of system reduces the reliance on external NLP tools and can build an end-to-end system.

Since the release of the event extraction task by ACE [1], many technologies and algorithms have been applied to the field of event extraction. The development of event extraction technology has attracted academic attention, and event extraction has been widely used in the industry. The structured information obtained from massive text data by

‡Corresponding author.

Project supported by the National Key R&D Program of China (No. 2018YFB0204301) and the National Natural Science Foundation of China (No. 61806216, and 61702533).

978-1-7281-6981-1/20/$31.00 ©2020 IEEE
DOI 10.1109/JCC49151.2020.00024

event extraction technology brings significant benefits to some information retrieval systems. Information retrieval systems (such as GDELT) in which event extraction technologies are applied can store event information by type. However, there is still much room for improvement in event extraction. Faced with a large amount of text data, event extraction models using a single machine are time-consuming. The parallelism of multiple computers is an inevitable trend for processing large amounts of text [4]–[6].

In this paper, we propose a method for extracting events in parallel, which is based on a pre-trained language model. In particular, our work contributes to the following:

- A new inter-cloud parallel event extraction method based on pre-trained language models is proposed. Our model can divide the data to be processed according to the characteristics of the data and dynamically adjust the division of labor in the cluster.
- The ability of the GPU and CPU to process each subtask of event extraction is discussed. A configuration recommendation is given to maximize the computing power of the cluster.
- Contrast experiments were carried out, and the experiments proved that the proposed model significantly shortened the event extraction time without affecting intensive reading.

II. Related Works

ACE defined the event as a specific occurrence involving participants, something that happens, and a change of state [1]. In order to facilitate the understanding of this article, we first introduce some concepts of event extraction used in this article as follows:

- Event Mention: a phrase/sentence/sentence group is describing the event, including a trigger and any number of arguments.
- Trigger: the word in the event description that best represents the event or important characteristics that determine the category of an event. It's usually a verb or a noun.
- Argument: participants of the event. It consists of words or phrases that express complete semantics, such as entities and attribute values.
- Argument Role: the role arguments played in events.

Event extraction has two tasks. The first is the identification and classification of event types, which can also be regarded as the identification and classification of triggers. The second is the identification of event arguments and the classification of argument roles. Early machine learning-based event extraction systems usually completed these two tasks step by step. The structure of such systems is called the pipeline structure. The event extraction methods using the pipeline structure usually first predict the trigger of the event and determine the event type, and then predict the event's argument based on the result [7]–[10]. However, there is an inherent disadvantage to event extraction models with pipeline structures: error propagation. In the pipeline structure, the stage of predicting the argument will use the results of the previous stage. For this reason, errors in the trigger stage will also affect argument prediction.

In order to avoid this drawback, Li et al. proposed a joint event extraction framework based on global features, which is used to extract the trigger word and argument of the event at the same time [11]. This joint event extraction framework can not only avoid error propagation but also effectively utilize the potential relationship between triggers and arguments [12]–[14].

With the rapid development of neural networks, researchers have applied neural networks to event extraction. First, Chen et al. proposed a Dynamic Multi-Pooling CNN(DMCNN) model based on the pipeline structure [15]. After performing dynamic segmentation on the convolved features, they perform max-pooling on each part separately, which enables CNN to learn more from the sentence. Subsequently, Nguyen et al. proposed a JRNN model, which is a model for event extraction using RNN based on a joint framework [16]. As far as we know, DMCNN is the state-of-the-art system with pipeline structure, and JRNN is the most advanced system using the joint framework.

III. Method

A. Event Extraction Based on Pipeline Structure

In this paper, we choose the pipeline structure to implement event extraction. We divide event extraction into two subtasks: trigger extraction and argument role assignment. The goal of the first subtask is to classify each word in the sentence and determine its event type (if the word is not a trigger, the event type is "NONE"). The goal of the second subtask is to assign an argument role to each word in the sentence according to the trigger extracted in the first stage. Because the structure of these two subtasks is similar, and the model of the second stage is more complicated than the first stage, we will introduce the model of the second subtask in this chapter. The differences between the two subtask models will be explained at the end of this subsection.

As shown in Fig 1, our event extraction model consists of three components:

- Word Embedding: Use external word embedding tools to encode the input text.
- Embedding Connect: Concatenate the word embeddings in a window to obtain a new feature representation.
- Classify: Assign the argument role to the word, according to the trigger predicted in the first stage.

1) Word Embedding: Context information plays a very important role in event extraction. Both DMCNN [15] and JRNN [16] use neural networks to capture word features from sentences. This paper uses external word embedding tools instead of neural networks to obtain word embeddings. Traditional word embedding methods use a fixed vector to express a word [17]. Those methods ignore the semantic information of the sentence in which the word is located. In order to better capture the characteristics of the context, the Google team proposed a model called Bidirectional Encoder Representations from Transformers(BERT) in 2018 [18].

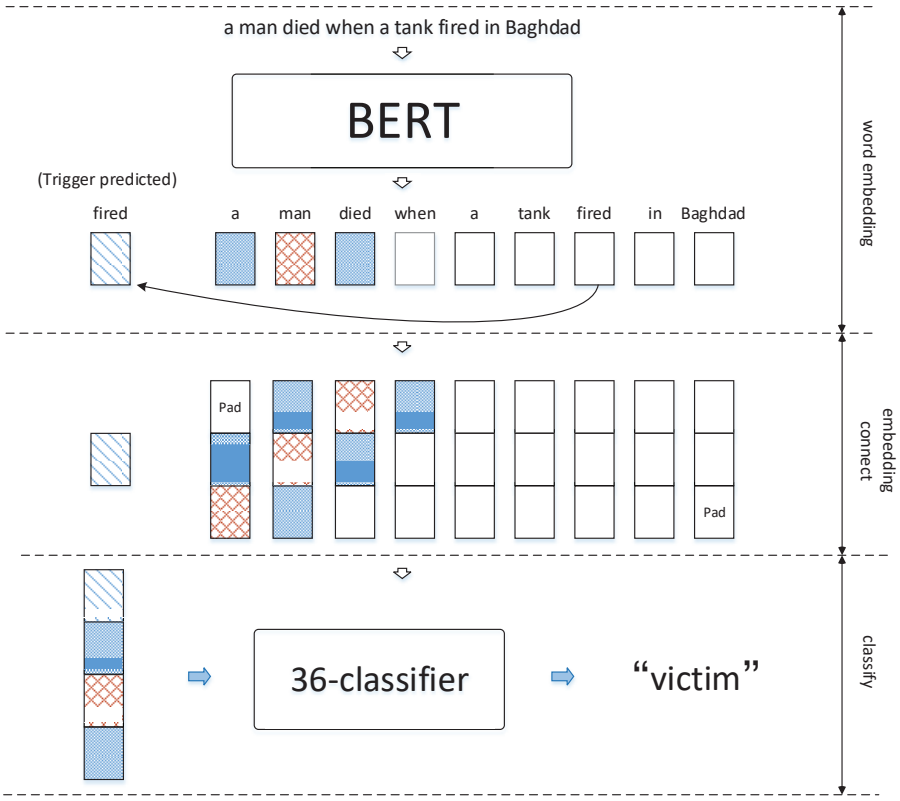

Fig. 1. Overview of the proposed model for argument role assignment subtask. At this time, ”fried” is known as the result of the first stage. This picture describes the process of assigning roles to the candidate argument ”man”..

Different from ELMo [19] and OpenAI GPT [20], BERT is based on the bidirectional transformer, which is a bidirectional language preprocessing model in the true sense. Thanks to the advantages of the transformer, this model can capture the potential relationships between words without being limited by the distance between words.

We recommend using the BERT model as the word embedding tool to provide word embeddings for the event extraction system in this paper.

2) Embedding Connect: In the corpus released by ACE, a trigger of an event is usually a word, but the argument of an event is a phrase composed of multiple words in most cases. In an argument consisting of multiple words, the prefix of a noun (such as an article or an adjective) is easily misjudged when classified. For example, in the event mention, ”a man died when a tank fired in Baghdad,” the word ”a” can only be considered as the argument of the event when combined with its neighbor ”man.” If the word ”a” is processed separately, the model may think that it is not an event argument. In order to strengthen the relationship between a word and its left and right neighbors, when we assign an argument role to a word, we will use a ”window” to combine it with its neighbors. When we deal with the word w_i with the window size of 3, the feature of w_i should be the combination of the word embeddings of these three words: w_{i-1}, w_i, w_{i+1}.

3) Classify: A sentence may contain multiple events, and a word may play different roles in different events. In order to take advantage of the potential relationship between trigger and arguments and distinguish the role that an argument plays in different events. We concatenate the word embedding corresponding to the trigger predicted by the first subtask with the features of the candidate argument, and the result is used as the input of the multi-classifier. Since the number of argument roles in the corpus released by ACE is 36 (including ”NONE”), the classifier at this stage should classify words into 36 categories.

4) Difference Between Two Subtasks: Since most triggers in the ACE corpus consist of only one word, the candidate trigger does not need to be connected with its neighbors. That is, in the first subtask, the window size should be set to 1. In addition, since there is no known information when the first subtask is executed, the feature of the candidate trigger does not need to be combined with other vectors. We directly use the output of BERT as the input of the multi-classifier. Considering there are 34 types of events (including ”NONE”) in the ACE2005 corpus, the classifier for the first subtask is

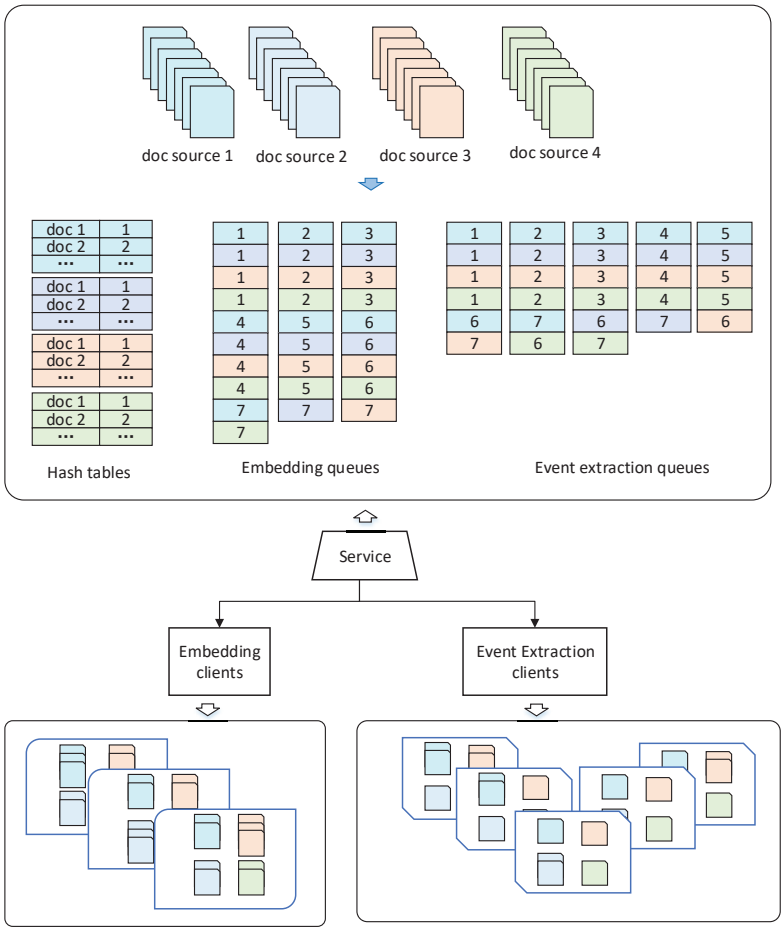

Fig. 2. Division of Computers in the Cluster.

34-classifier.

B. Event Extraction Via Inter-cloud Computing Technology

As shown in Fig 2, we divide the computers in the cluster into three categories according to their functions:

- Server: All pending documents are stored on the server. After receiving the task request, the server creates an independent hash table for each type of pending document. Depending on the source of the document data, the server builds an embedding queue or event extraction queue for every computer in the cluster. Whenever an embedding client finishes processing a document, the server receives the embedding results from the Embedding Clients and stores them locally for access by the Event Extraction Clients.
- Embedding Clients: Visit the server and obtain the pending documents against the hash table after receiving the embedding queues from the server. For each document, the Embedding Clients use BERT to process its content sentence by sentence, and package the results corresponding to the sentences in the same document into a file and return it to the server.
- Event Extraction Clients: Access the server and obtain the embedding file against the hash table after receiving the event extraction queues from the server. For a word embedding file, Event Extraction Clients parse it to obtain embeddings corresponding to multiple sentences. And then perform the event extraction sentence by sentence.

1) Method of Document Division: Data from different sources in the ACE2005 corpus have different characteristics. For example, the average size of documents from Newswire(NW) is less than 5k. And the sentences in these documents are mostly from news, which is relatively long. However, the average size of documents from Conversational Telephone Speech(CTS) is more than 8k, and the sentences in these documents are mostly human conversations, which are very short. This shows that documents from CTS have more sentences than documents from NW. Therefore, both

the Embedding Clients and the Event Extraction Clients will take a longer time to process a document from CTS. In order to make the number of tasks for computers in each cluster approximately the same, we establish embedding queues and event extraction queues to specify the number of tasks for each computer. After receiving the request, the server traverses the documents to be processed and builds a hash table for each source document, according to the number of computers in the initial embedding cluster. The structure of embedding queues is shown in Fig 3. The server takes turns filling the index numbers of unallocated pending documents in each hash table into the embedding queues orderly so that the amount of tasks for each Embedding Client is about the same. Event extraction queues are constructed in the same way as embedding queues.

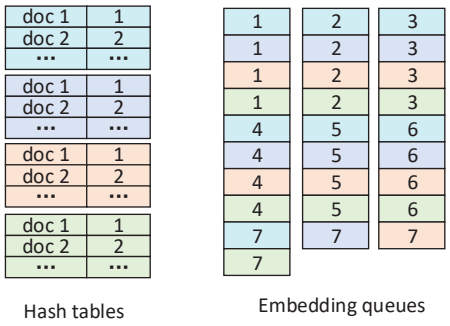

Fig. 3. The structure of embedding queues.

2) Dynamically Adjust the Tasks of the Computers in the Cluster: The event extraction model mentioned above has three components. The word embedding step is completed by Embedding Clients, and the steps of embedding connect and classify are performed by Event Extraction Clients. Because Event Extraction Clients need to combine features and classify each word in the document twice times (the first is the trigger classification and the second is the argument role assignment), an Event Extraction Client takes much longer to process a document than an Embedding Client. Therefore, the number of Event Extraction Clients must be greater than the number of Embedding Clients, so that the processing speed of the Event Extraction Cluster can keep up with the processing speed of the Embedding Cluster. However, when the number of Embedding Clients is too small, the Embedding Cluster cannot provide enough word embedding files, which will cause the computers in the Embedding Cluster to be idle and cause a waste of resources.

In this paper, we use a dynamic scheduling method to dynamically adjust the number of computers in the two clusters based on the work completion of the computers in the Embedding Cluster and Event Extraction Cluster. When the server accepts the request, it establishes two clusters according to the default number. At this time, the number of Embedding Clients is small. During task execution, when a computer in the Event Extraction Cluster is idle, the server will send an embedding instruction and an embedding queue to it. After receiving the embedding instruction, the idle Event Extraction Client will work as an Embedding Client for a short time. Similarly, when the server finds that computers in the Embedding Cluster have completed the tasks in the embedding queue, the server will send event extraction instructions and queues to them. Unlike the strategy of dealing with idle Event Extraction Clients, this idle Embedding Client will continue to work as Event Extraction Client until the end.

3) Make Good Use of CPUs and GPUs in the Cluster: The task of the Embedding Clients is to encode the words in the document into word vectors via BERT. The BERT model released by Google is based on a multilayer bidirectional transformer. Each layer of those transformers contains multi-head attention. Therefore, the BERT model contains a large number of matrix operations. We recommend using computers with a GPU as the Embedding Clients because the GPU has a stronger matrix processing capability than the CPU.

The task of Event Extraction Clients in the first stage is to multi-classify each word in the document, find out the trigger of the event, and determine the event type. Then, in the second stage, Event Extraction Clients combine the prediction results of the first stage to assign an argument role to each word in the document. For these two-stage tasks, the computer needs to make repeated predictions for the same batch of data multiple times, so we recommend using computers with a strong CPU as the Event Extraction Clients.

IV. Experiment

A. Dataset

We evaluate our model with the ACE2005 corpus. The ACE2005 corpus data come from six sources: Newswire (NW), Broadcast News (BN), Broadcast Conversation (BC), Weblog (WL), Usenet Newsgroups (UN), and Conversational Telephone Speech (CTS). In order to make different types of documents have more distinctive features, we combine the documents from NW, BN, BC, WL into a new document type called "NEWS." The two remaining source documents become separate document types called "CHAT" and "DISCUSSION." Documents from "NEWS" account for 70% of the experimental data, and the other two types of documents account for 15% of the experimental data.

B. Details

The BERT model used in this paper is the "BERT-Base, Uncased"[1] version released by Google. The initial number of computers for Embedding Clients is three, and for Event Extraction Clients is five. We do not consider the impact of communication bottlenecks on the system, because the amount of data processed by the Embedding Clients and Event Extraction Clients per unit time is much smaller than the communication bottleneck between the computers in the cluster.

[1]https://github.com/google-research/bert

We use a 3-layer fully connected layer to implement classifiers for the two subtasks of event extraction. When performing the trigger extraction subtask, considering the 33 subtypes of events in the ACE2005 corpus, we set the output dimension of the last fully connected layer to 34. Then, a softmax operation is used to calculate the probability of classifying words into different event types. Model training is performed on a single machine, and the loss function is the categorical cross-entropy loss function. Batch size is set to 32 during training, 16 during prediction. The learning rate is set to 1e-3. The model construction in the argument role assignment phase is the same as the trigger extraction phase. The final output dimension of the full connection layer is set to 36, the learning rate is set to 1e-5, and the window size is set to 3.

C. Evaluation of Inter-cloud Systems

We follow the evaluation methods in previous works [15], [16], using precision(P), recall(R), and F-Measure as the evaluation values. We name the system that deploys our event extraction model on a single machine as EE-single. The system that deploys the algorithm to the computer cluster mentioned above will be referred to as EE-cloud. To prove the effectiveness of our proposed system, we also compare the results with other event extraction systems using neural networks. They are DMCNN model with pipeline structure and JRNN model using joint structure. The performance of the four systems is shown in Table I:

TABLE I
OVERALL PERFORMANCE ON BLIND TEST DATA

Model	Trigger Identification(%)			Trigger Classification(%)		
	P	R	F	P	R	F
Chen's DMCNN	80.4	67.7	73.5	75.6	63.6	69.1
Nguyen's JRNN	68.5	75.7	71.9	66.0	73.0	69.3
EE-single	77.7	82.3	**79.9**	74.2	78.7	76.4
EE-cloud	77	82.7	79.7	74	79.5	**76.7**

Table I shows that the event extraction system proposed achieves the best performance during the trigger extraction phase. Compared with the advanced event extraction system JRNN, our model has an 8% and 7.1% improvement in identification and classification. This shows that the richer semantic information contained in the word embedding provided by the BERT model is very useful for event extraction. In particular, compared with the DMCNN model, which is also a pipeline structure model, the F1 value of our model in the trigger extraction phase has increased by 6.4% and 7.3%. The EE-single and EE-cloud models are very close in F1 value, which shows that our proposed inter-cloud event extraction model has substantially no impact on accuracy while greatly reducing the time overhead.

D. Time-consuming Analysis of Embedding Client and Even Extraction Client

We calculate the time cost of word embedding tasks on three types of clients with different computing resources, and the results are shown in Fig 4. The computing resources of these clients are Intel Core i7-6700HQ CPU, NVIDIA GTX 960M GPU, and 1080Ti GPU. From the histogram, we can intuitively see that for word embedding tasks, the time consumption of the client with the GPU is much less than the client using the CPU. From the histogram, we can intuitively see that when processing word embedding tasks, the time consumption of a client with a GPU is much less than a client with a CPU. The time consumption of a GPU with higher performance should be further reduced. Further, as the length of the sentence increases, the time consumed by the client to process the sentence also increases. In particular, when the sentence length exceeds a threshold, the time consumption of word embedding is directly proportional to the sentence length. For the client with a CPU, the threshold is about 64, while the threshold is about 256 for the client with a GPU.

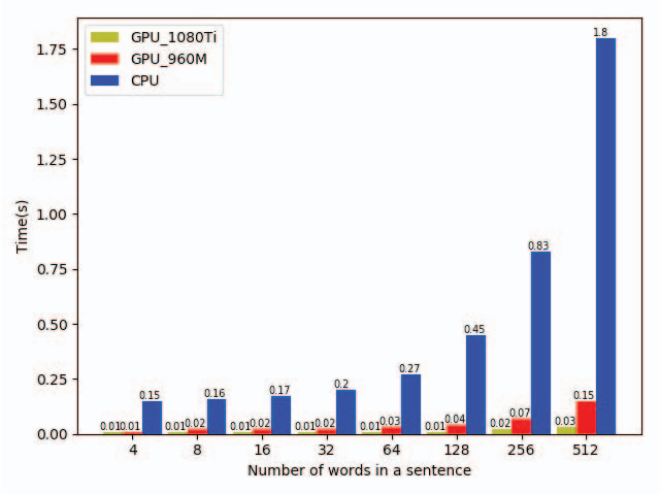

Fig. 4. The time consumption of the word embedding task in different computing environments.

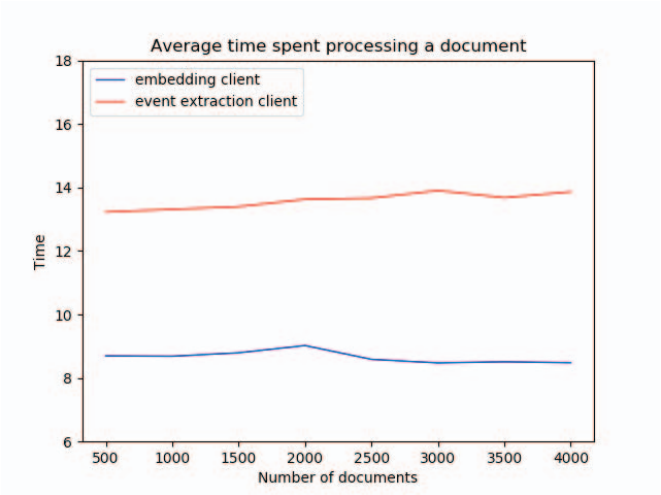

Fig. 5. Average time spent processing a document.

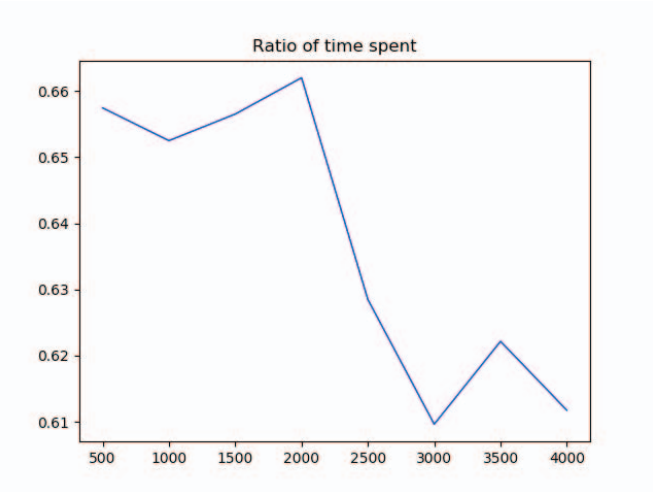

Fig. 6. The ratio of the time consumption of the Embedding Client to the Event Extraction Client.

In order to balance the processing rate of the Embedding cluster and the Event Extraction Cluster, we compare the time required for an Embedding Client and an Event Extraction Client for the same amount of data.

As shown in Fig 5, as the number of data increases, the average processing time of the Embedding Client tends to decrease, while the processing time of the Event Extraction Client tends to increase. It can be seen from Fig 6 that as the data size increases, the ratio of the time spent in Embedding to the time spent in event extraction decreases. This proves that when the amount of data processed by the cluster increases, the processing rate of the Embedding Cluster cannot meet the processing requirements of the Event Extraction Cluster. Therefore, it is necessary to adjust the number of computers in the two clusters dynamically.

V. Conclusion

This paper proposes an event extraction system based on inter-cloud computing technology, which can dynamically assign tasks based on the state of the computers in the cluster. On the one hand, the proposed event extraction model has greatly improved the F1 value compared to other advanced models. On the other hand, the use of inter-cloud computing technology reduces the time consumption of processing massive text data. Experiments prove that our event extraction system is effective in large-scale processing data.

References

[1] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, and R. M. Weischedel, "The automatic content extraction (ace) program - tasks, data, and evaluation," in *LREC*, 2004.

[2] D. Ahn, "The stages of event extraction," in *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*. Association for Computational Linguistics, 2006, pp. 1–8.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[4] D. Li, *Research on Peer-to-Peer Resource Location in Large-scale Distributed Systems*, 2009.

[5] Y. Fu, N. Xiao, X. Liao, and F. Liu, "Application-aware client-side data reduction and encryption of personal data in cloud backup services.doi:10.1007/s11390-013-1394-5.sci:000326953200007," *Journal of Computer Science and Technology*, vol. 28, pp. 1012–1024, 08 2013.

[6] X. Liao, C. Yang, T. Tang, H. Yi, F. Wang, Q. Wu, and J. Xue, "Openmc: Towards simplifying programming for tianhe supercomputers.doi:10.1007/s11390-014-1447-4.sci:000335936300013," *Journal of Computer Science and Technology*, vol. 29, pp. 532–546, May 2014.

[7] H. Ji and R. Grishman, "Refining event extraction through cross-document inference," *Proceedings of ACL-08: HLT*, pp. 254–262, 2008.

[8] S. Liao and R. Grishman, "Using document level cross-event inference to improve event extraction," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 789–797.

[9] Y. Hong, J. Zhang, B. Ma, J. Yao, G. Zhou, and Q. Zhu, "Using cross-entity inference to improve event extraction," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 1127–1136.

[10] D. McClosky, M. Surdeanu, and C. D. Manning, "Event extraction as dependency parsing," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 1626–1635.

[11] Q. Li, H. Ji, and L. Huang, "Joint event extraction via structured prediction with global features," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2013, pp. 73–82.

[12] H. Poon and L. Vanderwende, "Joint inference for knowledge extraction from biomedical literature," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 813–821.

[13] S. Riedel, H.-W. Chun, T. Takagi, and J. Tsujii, "A Markov logic approach to bio-molecular event extraction," in *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*. Boulder, Colorado: Association for Computational Linguistics, Jun. 2009, pp. 41–49. [Online]. Available: https://www.aclweb.org/anthology/W09-1406

[14] S. Riedel and A. McCallum, "Fast and robust joint models for biomedical event extraction," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, Jul. 2011, pp. 1–12. [Online]. Available: https://www.aclweb.org/anthology/D11-1001

[15] Y. Chen, L. Xu, K. Liu, D. Zeng, and J. Zhao, "Event extraction via dynamic multi-pooling convolutional neural networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015, pp. 167–176.

[16] T. H. Nguyen, K. Cho, and R. Grishman, "Joint event extraction via recurrent neural networks," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 300–309.

[17] J. Turian, L.-A. Ratinov, and Y. Bengio, "Word representations: A simple and general method for semi-supervised learning," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, Jul. 2010, pp. 384–394. [Online]. Available: https://www.aclweb.org/anthology/P10-1040

[18] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[19] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *CoRR*, vol. abs/1802.05365, 2018.

[20] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.