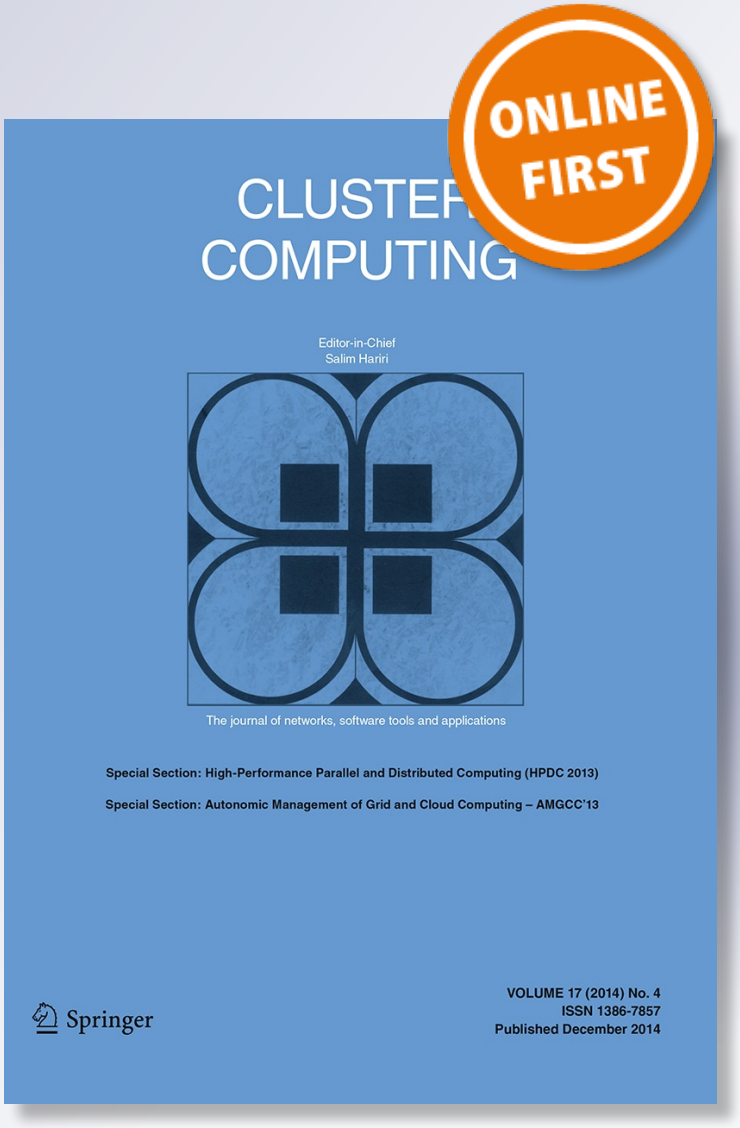
CRFs based parallel biomedical named entity recognition algorithm employing MapReduce framework

Zhuo Tang, Lingang Jiang, Li Yang, Kenli Li & Keqin Li

Cluster Computing
The Journal of Networks, Software Tools and Applications

ISSN 1386-7857

Cluster Comput
DOI 10.1007/s10586-015-0426-z



Springer

Cluster Comput
DOI 10.1007/s10586-015-0426-z

CRFs based parallel biomedical named entity recognition algorithm employing MapReduce framework

Zhuo Tang · Lingang Jiang · Li Yang · Kenli Li · Keqin Li

Received: 6 October 2014 / Revised: 4 January 2015 / Accepted: 10 January 2015

Abstract As the rapid growth of the biomedical literature, the model training time in biomedical named entity recognition increases sharply when dealing with large-scale training samples. How to increase the efficiency of named entity recognition in biomedical big data becomes one of the key problems in biomedical text mining. For the purposes of improving the recognition performance and reducing the training time, this paper proposes an optimization method for two-phase recognition using conditional random fields. In the first stage, each named entity boundary is detected to distinguish all real entities. In the second stage, we label the semantic class of the entity detected. To expedite the training speed, in these two phases, we implement the model training process on a parallel optimization program framework based on MapReduce. Through dividing the training set into several parts, the iterations in the training algorithm are designed as map tasks which can be executed simultaneously in a cluster, where each map function is designed to complete the calculation of a gradient vector component for each part in the training set. Our experiments show that the proposed method in this paper can achieve high performance with short training time, which has important implications for the current biological big data processing.

Z. Tang (✉) · L. Jiang · K. Li
College of Information Science and Engineering,
Hunan University, Changsha 410082, China
e-mail: ztang@hnu.edu.cn

L. Yang
College of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410004, China

K. Li
Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

Published online: 22 January 2015

Keywords Biomedical big data · Conditional random fields · MapReduce · Named entity recognition · Parallel algorithm

1 Introduction

With all kinds of wide applications of electronic texts in this era of information explosion, vast data have brought serious challenges for people to efficiently obtain useful information. People urgently need some automatic tools instead of using labor intensive artificial term lookup methods, which has given rise to the emergence of text mining technologies [1].

Text mining, which is recognized as the key to automatic acquisition of information, refers to the process of deriving high-quality information that is implicit, previously unknown, and potentially useful from massive data. It is an interdisciplinary field that is related to natural language processing (NLP), information retrieval, information extraction, data mining, computational linguistics, machine translation, block analysis, and so on.

Named entity recognition (NER) is a critical step for text mining. It is a subtask that seeks to locate and classify atomic elements with some special significance in text into predefined categories. The process of NER systems is structured as taking an unannotated block of text, and then producing an annotated block of text which highlights where the named entities are. The annotations are usually marked by XML ENAMEX elements, following the format developed for the Message Understanding Conference in the 1990s [2].

At present, biomedical literature has an enormous quantity and continues to increase at high speed. As one of the most concerned areas, papers on biomedicine have been published in a huge amount, reaching an average of 600,000 or

more per year. The currently most authoritative biomedical literature database Medical Literature Analysis and Retrieval System Online (MEDLINE) in American National Library of Medical (NLM) has included the information of more than 7,000 kinds of important biomedical journals published in over 70 countries and regions since 1966, including more than 18 million articles [3]. However, keyword searching in MEDLINE or Internet can only find relevant files list, so that a lot of valuable information contained in the text can not be directly shown to the user. Therefore, effective text mining for vast biomedical data analysis is an imminent task.

Research emphasis on biomedical text mining is mainly composed of two aspects, i.e., information extraction and data mining. Specifically, it includes biological named entity recognition (Bio-NER), synonyms and abbreviations identification, relation extraction, hypothesis generation using the reasoning relationship, text classification and integration framework of above work, and so on. This paper focuses on the study of Bio-NER that aims to help molecular biologists recognize and classify professional instances and terms, such as protein, DNA, RNA, cell_line, and cell_type.

Bio-NER is the precondition of subsequent work, such as extracting implicit semantic relation and background of biological process. However, because of its unique properties, unstable quantity, unified naming rules, complex form, and the existence of ambiguity, NER in biomedical field is not mature enough, short of effect as in other areas.

On the other hand, most Bio-NER systems are based on machine learning which need multiple iterative calculation from training data to create the model. Therefore, it is computationally intensive and seriously increases training time. In particular, when facing large-scale training samples in biomedical big data, the model training time will increase sharply [4]. It is natural to look into distributed systems and parallel methods to speed up the recognition process.

This paper presents a two-phase approach based on conditional random fields (CRFs) to improve the performance, and proposes an optimal algorithm for the model training process based on MapReduce to reduce the time consumption. Our approach divides the Bio-NER task into two sub-tasks, i.e., entity boundary detection and semantic class labeling. In the first phase, boundary of each entity is detected, in order to locate the real entities. In the second phase, entities detected are labeled into five classes: protein, DNA, RNA, cell_line, and cell_type. Meanwhile, in order to cope with the slow convergence speed of the training algorithm which contributes largely to the training time, we optimize the model training process based on MapReduce. In the training algorithm Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS), the first main step of each iteration is to calculate the gradient vector ∇L_i. Through analyzing the formula, we have found that the calculation of each component of ∇L_i can be linked with an ordered pair in the training data which is mutually independent. So we can split the calculation process into several Map tasks and summarize the results by a Reduce task. In addition, performance in the cases of different sizes of training samples, different numbers of map and compute nodes are studied, respectively. Experiments carried out on JNLPBA 2004 datasets show that the method outperforms most of the state-of-the-art systems and has short training time, especially for large-scale training samples with appropriate Maps and nodes.

The remaining sections of this paper are organized as follows. Section 2 reviews the related work. Section 3 describes our two-phase model based on CRFs and the useful feature sets in detail. Section 4 develops the MapReduce algorithm for the training process. Section 5 presents our experiments and the results. Finally, we conclude the paper in Sect. 6.

2 Related work

Current methods for Bio-NER can be divided into three categories, i.e., dictionary-based methods, heuristic rule-based methods [5], and statistical machine learning methods [6]. Compared with the first two methods, machine learning has achieved better performance in text mining. There have been many attempts to develop machine learning techniques to identify named entities in molecular biology, including Hidden Markov Model (HMM), Support Vector Machine (SVM), Maximum Entropy Markov Model (MEMM) [7], and CRFs [8,9]. However, one-phase approaches are widely adopted in most of the systems. According to their methods, an output label is represented by combining a region information B/I/O with a semantic class C (such as protein, DNA, RNA, cell_line, cell_type), which increases the number of features due to the increased number of labels. Especially, the training time will also be substantially longer.

Some researchers have tried to improve this situation. They have studied the efficiency of Bio-NER by using two-phase methods or exploring more helpful features [10,11]. Lee et al. proposed a method of two-phase Bio-NER based on SVMs and post-processing with dictionary-lookup in 2003 [12]. They used a SVM classifier and a simple dictionary in the first phase and chose the SVMs in the second phase. According to their experiments on GENIA corpus, the accuracy of the identification and the classification are about 79.9 and 66.5 %, respectively. Kim et al. used CRFs and ME for their two subtasks [13]. They selected separately a discriminative feature set for each subtask. Furthermore, a rule-based post-processing which was implemented using a finite state transducer was used to refine the results from the model based on machine learning. Their experiments [14] compared the results of CRFs and ME models and drew the conclusion that the CRFs-based model outperforms the ME-based model. The best performance they have obtained was 71.19 % by

CRFs model with post-processing. Settles [9] presented a framework for simultaneously recognizing occurrences of protein, DNA, RNA, cell_line, and cell_type entity classes using CRFs with a variety of traditional and useful features, and showed that this approach can achieve an overall F_1 measure around 70. However, from the research of Li et al. [15], we have known that most one phase approaches need 4–5 h in training process, while two-phase approaches reduce 3 h around, facing about only 15 MB data.

Biomedical literatures are typical big data, a large and complex collection of datasets characterized by four V's (volume, variety, veracity, and velocity), is difficult to deal with using traditional data processing algorithms and models. Wang et al. proposed a dictionary learning algorithm, which extends the classical method that uses the K-means and Singular Value Decomposition (K-SVD) algorithm by incrementally updating atoms, will ably represent the spatiotemporal remote sensing of Big Data and do so both efficiently and sparsely [16]. Recently, the experimental strategy for two-phase Bio-NER [14] and computational requirements for large-scale data-intensive analysis of big data have grown significantly. There are two effective parallel implementations for CRFs currently, the CRFs based on Message Passing Interface (MPI) and the CRFs based on Graphics Processing Units (GPU). However, they are not suitable for large volumes of biomedical big data in data-intensive applications. The strongest weakness of MPI is communication latency in a big data environment, and it is programming complex method, so that MPI for CRFs shows low performance for big data parallel processing. Due to the capacity limits of global memory and the bottleneck of data transmission for big data applications, GPU for CRFs based on CUDA programing also shows low performance. Hence, they are not good choices for current parallel CRFs with large-sale of Bio-NER in this paper's research plot.

At present, MapReduce, introduced by Google in 2004, is an excellent model for distributed computing. It has emerged as an important and widely used programming model for distributed and parallel computing, due to its ease of use, generality, and scalability. MapReduce Framework divides big data set into several parts called split block and divides the calculation problem of large amount of data into two stages, Map and Reduce. In Map stage, many map tasks are parallel executed, and every map process one split block and parses it as ⟨key, value⟩ sets. In Redcue stage, reduce tasks will get all the processing ⟨key, value⟩ sets from map tasks, and then put all the combination results and returns to the output. It is worth noting that each map operation is relatively independent, and all of the map tasks are run in parallel [17]. Meanwhile MapReduce processing can provide part of the function of fault tolerance and error recovery. When a map task or reduce task fails, the corresponding work will be rearranged, thus it will not affect the continuity of work. So MapReduce is able to handle large amount of data processing problem which is difficult to use general servers.

Now it is popular in text mining of various applications [18], especially Natural Language Processing (NLP) and Machine Learning (ML), as the MapReduce paradigm has emerged as a highly successful programing model for large-scale data-intensive computing applications [19]. Laclavik et al. presented a pattern of annotation tool based on MapReduce architecture to process large amount of text data [20]. Lin and Dyer discussed the processing method of data intensive text based on MapReduce, such as parallelization of EM algorithm and HMM model [4]. Wang et al. designed a MapReduce framework G-hadoop that aims to enable largescale distributed computing across multiple clusters[21]. Whitney et al. implemented a distributed perceptron algorithm training on a HPC cluster, and examined two topologies for the combination of separately trained weight vectors, and found that the choice which duplicates computation leads to shorter runtime [22]. However, little research on CRFs based on MapRedcue has been launched, especially for the Bio-NER application. That is because highly iterative of CRFs algorithm leading to difficulties in parallel and the complex semantics of biomedical named entity itself. In this paper, we give the solution to fill the void.

3 Two-phase recognition based on CRFs

3.1 Conditional random fields (CRFs)

Bio-NER can be thought of as a sequence segmentation problem, in which each word is a token in a sequence to be assigned a label such as protein, DNA, RNA, cell_line, cell_type, or others. Conditional random fields (CRFs) is a sequence data labeling model based on statistical approaches. As an undirected graphical model which encodes the conditional probability distribution, it represents a conditional model $P(y|x)$, which uses a Markov random field, with nodes corresponding to elements of the structured object y, and potential functions that are conditional on (features of) X. CRFs is well suited for sequence analysis, and in particular, has been shown to be useful in part-of-speech tagging and named entity recognition task [23].

More often than not, linear chain CRFs that corresponds to a conditionally trained finite-state machine is used in Bio-NER application [24], shown in Fig. 1.

Following Lafferty et al. [23], the conditional probability of the state output sequence y for a given input sequence x like above is

$$P(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{i=1}^{n}\sum_{k=1}^{K} \lambda_k f_k(x, i, y_{i-1}, y_i)\right), \quad (1)$$

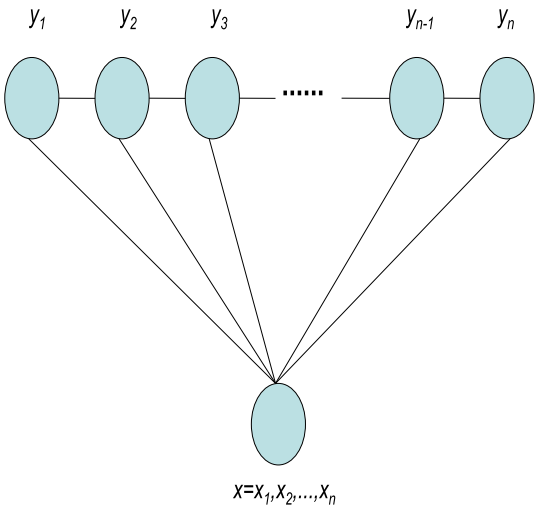


Fig. 1 Linear chain CRFs model

$$Z(x) = \sum_{y} \exp\left(\sum_{i=1}^{n}\sum_{k=1}^{K} \lambda_k f_k(x, i, y_{i-1}, y_i)\right), \qquad (2)$$

where $Z(x)$ is the normalization factor, $f_k(x, i, y_{i-1}, y_i)_{k=1}^{K}$ are feature functions, λ_k is a weight of a feature, and y_{i-1}, y_i denote the previous and the current states, respectively.

The process of model training is to find the maximum likelihood of ordered pairs (x_i, y_i) in training set $T = (x_i, y_i)_{i=1}^{M}$, which is in pursuit of the decent value of λ in fact. And the labeling for a new sequence in test set can be done by Viterbi algorithm [25].

3.2 Features of entity boundary detection

Entity boundary detection, which is the first phase of our Bio-NER task, is to label terms inside entities with C and outside entities with O, respectively. Similar to previous NER work [26], we use the features which are very easy to derive for the first phase. The features are listed below.

1. *Word features* The current word and its adjacent words have been proved to play an important role in most NER work. We use a window of size 5 which consists of the current word, the two preceding words, and the two following words. From experiments carried out on window sizes 3, 5, and 7, respectively, a window size more than 5 not only increases the dimension of features, but also degrades the value of F_1 score. When window size is 3, the F_1 score also degrades. This demonstrates that the second preceding word and the second following word are helpful features. To reduce the number of features, all the word features are lower-cased. Another feature which has a binary value for capitalization will be compensated.

2. *Capitalization and digit features* For capitalization, a few binary features are defined corresponding to initial capital, all capital, capital in inner and initial capital then mix. For digit, a few binary features are defined corresponding to only digit, digit with special character, initial digit then alphabetic, and digit in inner.

3. *Prefix and suffix features* Prefixes and suffixes are indicative for identifying NEs. The length of the prefixes and suffixes is very significant. If the length is too long or too short, then the feature would not be useful. During the experiments, we have used all suffixes and prefixes with length up to 5.

4. *Part-of-speech features* Part-of-speech features are also vital in the NER task. In our experiments, GENIA tagger v3.0 is applied to get POS features. The tagger is specifically tuned for biomedical documents such as MEDLINE abstracts. The GENIA tagger is trained not only on the Wall Street Journal corpus but also on the GENIA corpus and the PennBioIE corpus, so the tagger works well on various types of biomedical literature. The reported POS tagging accuracy of the tagger is 98.26 % on GENIA corpus. The POS values of the current, the previous two, and the next two words have been used in our experiments.

5. *Word normalization features* In the Bio-NER task, there are many words which are different from the words in the newswire domain. For example, the word "CD28" is a name of protein. The word "CD28" is composed of the two capitals and two digits. Words similar to the word "CD28" tend to be NEs. In our experiments, we have two types of word normalization. Capitalized characters, small characters, and consecutive digits are replaced by A, a, and 0, respectively. The word "CD28" will be normalized as AA00. To classify the similar words better, we also use a short normalization. The AA00 is shortened as A0.

6. *Previous NE tags* The previous NE tags are helpful for the target NEs. We have used two previous tags as features.

7. *Trigger word features* Some verbs preceding to NEs will deliver useful information about the NE class. For example, the word "activate" often occurs previous to NEs. We extract these trigger words automatically from the training corpus based on their frequency of occurrences.

8. *Keywords features* Many words which occur frequently as entities contribute to the NER task. We automatically extract the top 200 frequent ones as the keywords list from the training data. If the entity is in the keywords list, the binary feature is set to 1; otherwise, it is set to 0.

3.3 Features of semantic class labeling

Semantic class labeling is the second phase. The task is to label the entities detected in the first phase into 5 NE categories: protein, RNA, DNA, cell_line, and cell_type. In addition to the features described in Sect. 3.2, we also use the following four features for CRFs.

followed	O
by	O
the	O
activation	O
of	O
phospholipase	B-protein
A2	I-protein
and	O
5-lipoxygenase	B-protein
.	O

Fig. 2 An example of a segment of a sentence

1. *Input features* The input features are the results of entity boundary detection.

2. *Category-noun features* Nouns are very crucial to the category of the NEs because NEs are nouns. Some nouns are not frequent, but they can recognize the category directly. That is, all the occurrences of the noun are belonging to the same category. For example, "receptor" was classified into protein or DNA. The category of the noun "receptor" was according to the context information. The noun "phospholipase" was classified as protein almost at every occurrence and it is a category noun. All entities in our experiment are classified into five types: protein, RNA, DNA, cell_line, and cell_type. For each class, the category nouns which can decide the class of themselves are selected. For a particular word w_i, the following equation measures the fraction $p_j(w_i)$ of the word belonging to a class C_j:

$$p_j(w_i) = \frac{\text{occurrences of } w_i \text{ is a NE of class } C_j}{\text{total occurrences of } w_i \text{ in corpus}}. \tag{3}$$

Whether a word is selected for a class is decided according to the value $p_j(w_i)$. If the value $p_j(w_i)$ is greater than 0.9, the word w_i will be considered as a category noun and be added to the category-noun list of class C_j. Five binary features of the word are defined for these five classes. For class C_j, the binary feature is defined as having value 1 if the word belongs to its category-noun list; otherwise, 0 is given.

3. *Adjacent-noun features* Some words which occur as adjacent nouns appear to be helpful for the recognition of the category for the entities. For example, the NEs which have the word "phospholipase" as their adjacent nouns are classified into protein. For each class, the adjacent nouns are selected as features. In our experiments, the adjacent nouns are the two preceding or two following nouns of the target word. A segment of a sentence is shown in Fig. 2. The word "A2" has an adjacent noun "phospholipase", and this "A2" is classified into a protein. If a particular word w_i has an adjacent noun as a category noun, the word w_i will be added to the adjacent-noun list of the corresponding class. Binary features are defined for each class.

4. *Combined features* Some words are not category nouns, but they are indeed helpful with the aid of the other features. If the value of $p_j(w_i)$ is lower than 0.9, the word w_i will not be added to the category-noun list of class C_j. For example, the value of "enhancer" belonging to the class DNA is 0.66. It is recognized as an entity for the class DNA or protein. Sometimes, "enhancer" is not recognized as an entity. But if the digit feature of the preceding words of "enhancer" is 1 (e.g., HIV-1 enhancer), it is an entity for DNA. The combined features supplement category-noun features.

4 MapReduce for the training process

4.1 Parameter estimation

Parameter estimation of the model which is to calculate the parameter λ is the major part of the training process, and the most commonly used method is maximum likelihood estimation. However, on the other hand, when confronting large-scale training data, the parameter estimation process is complicated and slow, which will tremendously increase the time consumption. If we speed up the process of the parameter λ calculation, we would cut down the training time [27,28].

Presume that every $(x_i, y_i) \in T = (x_i, y_i)_{i=1}^{M}$ is independently and identically distributed. Then the log-likelihood function of the training data T will be shown below:

$$L(\lambda) = \sum_{T} \log P(y|x). \tag{4}$$

For linear chain CRFs as

$$P(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{i=1}^{n}\sum_{k=1}^{K} \lambda_k f_k(x, i, y_{i-1}, y_i)\right),$$

the log-likelihood function should be

$$\begin{aligned} L(\lambda) &= \sum_{T} \log\left(\frac{1}{Z(x)} \exp\left(\sum_{i=1}^{n}\sum_{k=1}^{K} \lambda_k f_k(x, i, y_{i-1}, y_i)\right)\right) \\ &= \sum_{T}\left(\sum_{i=1}^{n}\sum_{k=1}^{K} \lambda_k f_k(x, i, y_{i-1}, y_i) - \log Z(x)\right). \end{aligned} \tag{5}$$

To find the parameter λ to make convex function $L(\lambda)$ reach the maximum, we should make its gradient vector

$$\nabla L = \left(\frac{\partial L}{\partial \lambda_1}, \frac{\partial L}{\partial \lambda_2}, \ldots, \frac{\partial L}{\partial \lambda_k}\right)$$

to be $\vec{0}$, and then try to seek the parameter λ.

The partial derivative of each parameter λ_k will be calculated like this:

$$\begin{aligned}\frac{\partial L(\lambda)}{\partial \lambda_k} &= \sum_T \left(\sum_{i=1}^{n} f_k(x, i, y_{i-1}, y_i) - \frac{\partial}{\partial \lambda_k} \log Z(x) \right) \\ &= \sum_T \left(\sum_{i=1}^{n} f_k(x, i, y_{i-1}, y_i) - \frac{1}{Z(x)} \cdot \frac{\partial Z(x)}{\partial \lambda_k} \right) \\ &= \sum_T \Bigg(\sum_{i=1}^{n} f_k(x, i, y_{i-1}, y_i) \\ &\quad - \frac{1}{Z(x)} \sum_y \exp\left(\sum_{i=1}^{n} \sum_{k=1}^{K} \lambda_k f_k(x, i, y'_{i-1}, y'_i) \right) \\ &\quad \sum_{i=1}^{n} f_k(x, i, y'_{i-1}, y'_i) \Bigg) \\ &= \sum_T \Bigg(\sum_{i=1}^{n} f_k(x, i, y_{i-1}, y_i) \\ &\quad - \sum_y \frac{1}{Z(x)} \exp\left(\sum_{i=1}^{n} \sum_{k=1}^{K} \lambda_k f_k(x, i, y'_{i-1}, y'_i) \right) \\ &\quad \sum_{i=1}^{n} f_k(x, i, y'_{i-1}, y'_i) \Bigg) \\ &= \sum_T \Bigg(\sum_{i=1}^{n} f_k(x, i, y_{i-1}, y_i) \\ &\quad - \sum_y P(y'|x) \sum_{i=1}^{n} f_k(x, i, y'_{i-1}, y'_i) \Bigg). \end{aligned} \tag{6}$$

Meanwhile, we introduce a penalty function to solve overfitting:

$$L'(\lambda) = L(\lambda) - \sum_k \frac{\lambda_k^2}{2\sigma^2}. \tag{7}$$

And the partial derivative of each parameter λ_k for the log-likelihood function is:

$$\begin{aligned}\frac{\partial L'(\lambda)}{\partial \lambda_k} &= \sum_T \Bigg(\sum_{i=1}^{n} f_k(x, i, y_{i-1}, y_i) \\ &\quad - \sum_y P(y'|x) \sum_{i=1}^{n} f_k(x, i, y'_{i-1}, y'_i) \Bigg) - \frac{\lambda_k}{\sigma^2}. \end{aligned} \tag{8}$$

Unfortunately, there is no closed-form solution for searching the target parameter λ by making the above equation to be zero. As a result, the parameter selection process needs to use some numerical iteration techniques, such as Generalized Iterative Scaling Algorithm (GIS), Improved Iterative Scaling Algorithm (IIS), Limited-memory Broyden–Fletcher–Goldfarb–Shanno Algorithm (L-BFGS) [29], and so on.

4.2 L-BFGS

In the field of training phase in CRFs machine learning algorithm, one of the most versatile, effective, and widely used methods is the L-BFGS. Comparing with GIS, IIS and other training algorithm, L-BFGS [30–32] uses the change of the gradient to simulate the Hessian matrix at each iteration, so that it can achieve good results. Each iteration in L-BFGS can be mainly divided into three steps, and the full description of the L-BFGS algorithm is described detailedly by Jorge Nocedal in [33,34]:

- Calculate ∇L_i which is on behalf of iteration i of the gradient vector;
- Select the search direction $\overrightarrow{p_i} = H_i \nabla L_i$;
- Select the step length $\overrightarrow{a_i}$ which satisfies strong Wolfe conditions, and finally calculate the estimated parameters of the next.

4.3 MapReduce scheme

Although MapReduce is an easy-programming parallel framework for big data [35], it is not easy to comprehensively parallelize the whole iterative process of L-BFGS. Considering the iterative nature of overall algorithm L-BFGS, we choose to optimize each step of the iteration. We propose an optimization method for calculating the gradient vector ∇L_i. The main idea is to use MapReduce to calculate each component of ∇L_i.

For the training data $T = (x_i, y_i)_{i=1}^{M}$, the gradient vector of the model:

$$\nabla L = \left(\frac{\partial L}{\partial \lambda_1}, \frac{\partial L}{\partial \lambda_2}, \ldots, \frac{\partial L}{\partial \lambda_k} \right)$$

can be gradually obtained by calculating the partial derivative of each λ_k iteratively:

$$\begin{aligned}\lambda_0 &= 0; \\ \frac{\partial L'(\lambda)}{\partial \lambda_k} &= \sum_T \Bigg(\sum_{i=1}^{n} f_k(x, i, y_{i-1}, y_i) \\ &\quad - \sum_{y'} P(y'|x) \sum_{i=1}^{n} f_k(x, i, y'_{i-1}, y'_i) \Bigg) - \frac{\lambda_k}{\sigma^2}. \end{aligned} \tag{9}$$

Notice that:

$$\sum_{i=1}^{n} f_k(x, i, y_{i-1}, y_i)$$

is the expectation of characteristics f_k under the empirical distribution with a given vector $\overrightarrow{x}$ in one ordered pair. Its result can be simply calculated by knowing about the number of occurrences of f_k in the ordered pair.

$$\sum_{y'} P(y'|x) \sum_{i=1}^{n} f_k(x, i, y'_{i-1}, y'_i)$$

is the expectation of characteristics f_k under the model distribution with a given vector $\overrightarrow{x}$ in one ordered pair. We use a dynamic programming method called forward-backward algorithm to get its result indirectly like employing in HMM.

So we may safely draw the conclusion that it is feasible to calculate the Eq. (9). It is observed that we can get the difference between the two expectations for each of the input sequence in the training set with the condition that ordered pairs are mutually independent, and then accumulate all results. That means we can use MapReduce structure to get $\frac{\partial L'(\lambda)}{\partial \lambda_k}$ rapidly.

For MapReduce, the training set $T = (x_i, y_i)_{i=1}^{M}$ is firstly divided into several parts. Calculating the expectations of f_k on each part in the training set can be accomplished by a Mapper() in Hadoop, and the accumulation for the whole training set by Reducer(). For the post-processing, we add the difference between the accumulated value and the penalty term.

The MapReduce plan is divided into four main steps:

- Divide the training set into several parts (see Algorithm 1);
- Map function: Complete the parallel calculation for the expectations of f_k on each part in the training set, and map all of the samples in the training data to an ordered pair of (*key*, *value*) with keys value being the name of the training set (see Algorithm 2);
- Reduce function: Use the method of summation of values for all the ordered pairs with the same *key* value (see Algorithm 3);
- Post-processing: Calculate the difference between the summation and the penalty term (see Algorithm 1).

Algorithm 1 MapReduce Processing for the Parameter Estimation in L-BFGS

Require:
T:the training set $(x_i, y_i)_{i=1}^{M}$.
M:the predefined number.
D:the intermediate data.
EV:the difference of two expected value.
Ensure:
CL_i:component of the goal gradient vector ∇L_i.
1: Divide T into m parts: $T_1, T_2, ..., T_m$;
2: //Initial intermediate data
3: $D \leftarrow \emptyset$;
4: MapTask_GetExpectedValue();
5: ReduceTask_GetExpectedValue();
6: //Add the penalty term to correct data
7: $CL_i \leftarrow EV - \lambda_k/\sigma^2$. //We use $\sigma^2 = 10$

Algorithm 2 SubTask1: Map

Require:
T:the training set $(x_i, y_i)_{i=1}^{M}$.
D:the intermediate data.
1: **MapTask_GetExpectedValue()** {
2: **for** each part of T **do**
3: $key \leftarrow$ name of T;
4: $value \leftarrow \sum_{T_m}(\sum_{i=1}^{n} f_k(x, i, y_{i-1}, y_i) - \sum_{y'} P(y'|x) \sum_{i=1}^{n} f_k(x, i, y'_{i-1}, y'_i))$;
5: //store $(key, value)$ in the intermediate data D
6: $D \leftarrow D \cup \{(key, value)\}$;
7: **end for**
8: }

Algorithm 3 SubTask2: Reduce

Require:
D:the intermediate data.
Ensure:
EV:the difference of two expected value.
1: **ReduceTask_GetExpectedValue()** {
2: $EV \leftarrow 0$;
3: **for** each $(key, value)$ in D **do**
4: $EV \leftarrow EV + value$;
5: **end for**
6: return EV.
7: }

5 Experiments

5.1 Corpus

The JNLPBA 2004 datasets has been used for our experiments. The training set is GENIA corpus v.3.02, which consists of 2000 abstracts that are searched out by MEDLINE database with using MeSH terms human, blood cells and transcription factors as the keywords. 404 abstracts of the test set are also from MEDLINE database. Half of them are obtained by the same way as the training set, and the other half are searched out by using MeSH terms blood cells and transcription factors as the keywords. In our experiments, we only use the five classes: protein, DNA, RNA, cell_line, and cell_type as the task does, and the corpus was transformed into B/I/O format to express entities.

The experiment results are measured with F_1 score. As one of the commonly used evaluation standards, the F_1 score combines precision rate P and recall rate R:

$$P = \frac{TP}{TP + FP}, \tag{10}$$

$$R = \frac{TP}{TP + FN}, \tag{11}$$

where TP is the number of correctly identified named entities, FP is the number of non-named entities which are

Table 1 Results of the two phases

Phase	Class	Precision	Recall	F_1 score
Entity boundary detection	All	75.22	78.32	76.74
Semantic class labeling	Protein	73.92	77.23	75.54
	DNA	73.52	71.29	72.39
	RNA	64.61	77.69	70.50
	Cell_line	60.41	64.29	62.29
	Cell_type	77.34	71.27	74.18
	All	70.79	76.01	73.31

marked as named entities, and FN is the number of named entities that the system does not recognize. In many cases, we need to put them together as F_1 score value given below:

$$F_1(P, R) = \frac{2PR}{P + R}. \tag{12}$$

5.2 Results of the two-phase recognition approach

As described in Sects. 3.2 and 3.3 in this paper, experiments for the two-phase tasks are carried out based on CRFs. In Table 1, we have summarized the results of entity boundary detection and semantic class labeling, where all the features described in Sects. 3.3 and 3.3 are used to help increase the performance.

Next, we make comparisons between our system and the top-ranked systems on the JNLPBA 2004 shared task in Table 2. In both phases, we get higher precision and recall rates. Because of no post-processing, our two-phase CRFs model obtains a lower F_1 score of 73.31 % than the CRFs model with post-processing algorithms which has 74.31 % in F_1 score. Compared to the F_1 score 73.06 % of the CRFs model without post-processing algorithms, our CRFs model has higher F_1 score due to the useful features. In a word, Table 2 shows our system outperforms most systems.

5.3 Results of the optimization approach based on MapReduce

Now there are many Google's MapReduce open source implement versions, for example Apache Hadoop, Cloudera distribution including Hadoop (CDH), Hortonworks Data Platform (HDP), G-Hadoop [38] etc. As presented in Sect. 3.3, we adopt Apache Hadoop framework which has been widely used in industry of the whole world [39]. All the features described above are used in the two phases. In the experiments, Hadoop 1.2 is deployed in a cluster with five nodes, and the size of its data block is limited to 64 MB. The hardware configuration is listed in Table 3, and the configurations of the Hadoop cluster are in Table 4.

It is hard to get existing large-scale corpus with labeled information, so we use our own production. We copy from the GENIA corpus several times to the appropriate test scale. This method does not change the accuracy, which is important and necessary for our time research.

Table 2 Comparisons with other top-ranked experiments

Method	Precision	Recall	F_1 score
Our two-phase method	70.79	76.01	73.31
Li et al. [6]	70.59	75.71	73.06
Zhou and Su [36]	69.42	75.99	72.55
Okanohara et al. [37]	70.35	72.65	71.48
Kim and Yoon [14]	72.77	69.68	71.19
Finkel et al. [7]	71.62	68.56	70.06

Table 3 The hardware configuration

Level	CPU	Memory	Amount
Slave	4-core, 3.07 GHz	4G	4
Master	4-core, 2.7 GHz	4G	1

Table 4 The hardware configuration

Configuration items	Cofiguration properties	Value
Map slots	mapred.tasktracker.map.tasks.maximum	4
Reduce slot	mapred.tasktracker.Reduce.tasks.maximum	2
Copy thread	mapred.reduce.parallel.copies	5
HDFS replications	dfs.replication	3
Input split size	dfs.block.size	64 MB

Our experiments were carried out on a single machine and a Hadoop cluster with different scale of training samples, respectively. For clearly observing the results, we divide our experimental results into several charts to show different measures. In Fig. 3, we compare the training time in the two phases of the unoptimized and optimized methods. Then, in Fig. 4, we calculate the training speed and make comparison between them. Finally, the parallel speedup is given in Fig. 5.

It is obvious from Fig. 3 that the training time of the two phases increases with the expansion of training samples in both unoptimized and optimized methods. In addition, in the second stage, the model training process has more training time than it does in the first stage. It is because the other four useful features described in Sect. 3.3 played roles.

For small training samples, there is no much difference between the training times of the unoptimized and optimized methods. When the training samples are limited within 120 MB, the optimized method takes more time to establish the model. It is mainly due to the communication processing. Specifically, in the implementation process of MapReduce

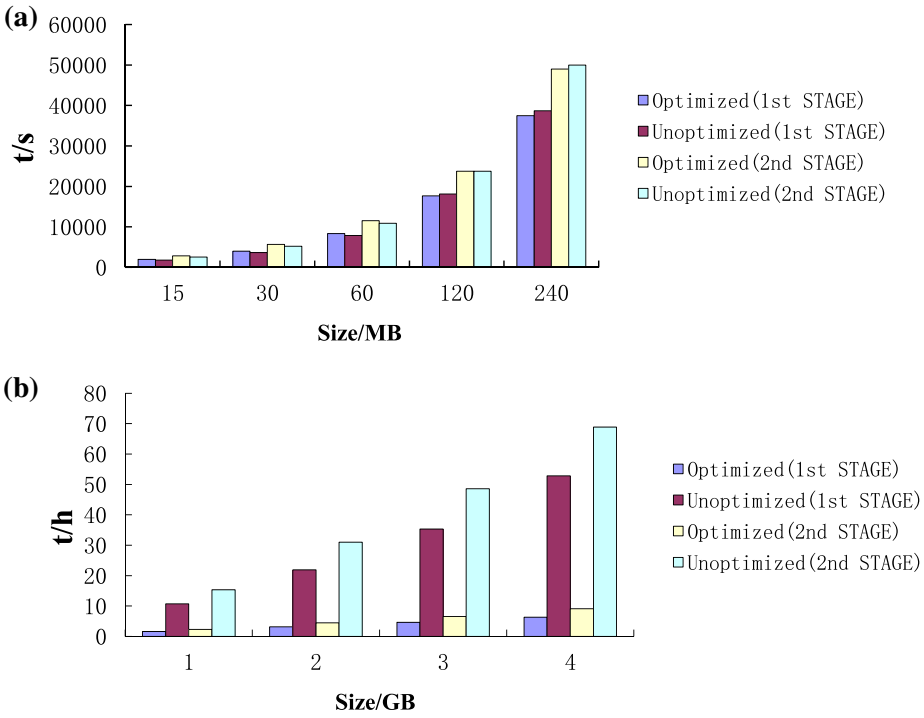


Fig. 3 Comparison between unoptimized and optimized recognitions (time). **a** Training time under small training samples. **b** Training time under large training samples

framework, each input data is output to the map-task local node when it is split and distributed to the corresponding map task. And then the reduce task will copy output data which have the same key value from map nodes. The process of data transmission leads to certain time delay. Moreover, there is a shuffle processing for data internal sorting after copying.

For large training samples, there is significant difference between the training times of the unoptimized and optimized methods. The time of the unoptimized method in the two phases keeps growing, and the extent of growth is more and more significant. While using the optimized method, the time has modest growth especially for large-scale samples. As we can expect, the increase rate tends to be gentle, and will be leveled off when samples are enough. Overall, the optimized method consumed less training time than the unoptimized method, but it is not obvious when the scale of training samples is small.

In order to observe and study the relation between the training speed (MB/s or GB/s) and the size of samples, we give the corresponding results in Fig. 4 obtained from Fig. 3.

For small data, the training speed decreases with the expansion of training samples for both optimized or unoptimized methods. However, it is gradually stable in the optimized method especially in its second stage, while dropping in the unoptimized method. On the other hand, when confronting large-scale data, the speed is higher and higher with training samples enlarging in the optimized method, while flatten out at a low speed in the unoptimized method. However, the speed starts decreasing when the data enlarge from 3 to 4 GB. This is largely restricted to the Hadoop cluster performance and the amount of Map.

In a word, the optimized method has higher training speed distinctly, especially in large-scale training samples. As the training samples continue to grow, the restriction of the cluster performance and Maps has show its own role in the parallel optimization process.

Figure 5 gives the speedup, which is defined as the ratio of the sequential processing time to the parallel processing time, i.e., the ratio of the unoptimized training time to the optimized training time.

As explained previously, our optimization method does not show obvious advantage for small data. However, when facing a large amount of data in parallel, the speed of our algorithm can make up the time waste caused by the internal communication delay, and bring larger parallel speedup.

From the above results, it can be seen that when the training samples are small, the parallel optimization method has no obvious influence on training time and speed. When the training sample reaches certain size, the optimized method has less training time and higher training speed. However, when facing training samples with larger sizes, the effect of the optimized method will not be better. So in the next step, we conduct two other experiments using our methods for large-scale training samples to explore the influence of the number of compute nodes and Map in a Hadoop cluster on the training efficiency.

The following experiments are designed to assess the impact on training time in the cases of different number of

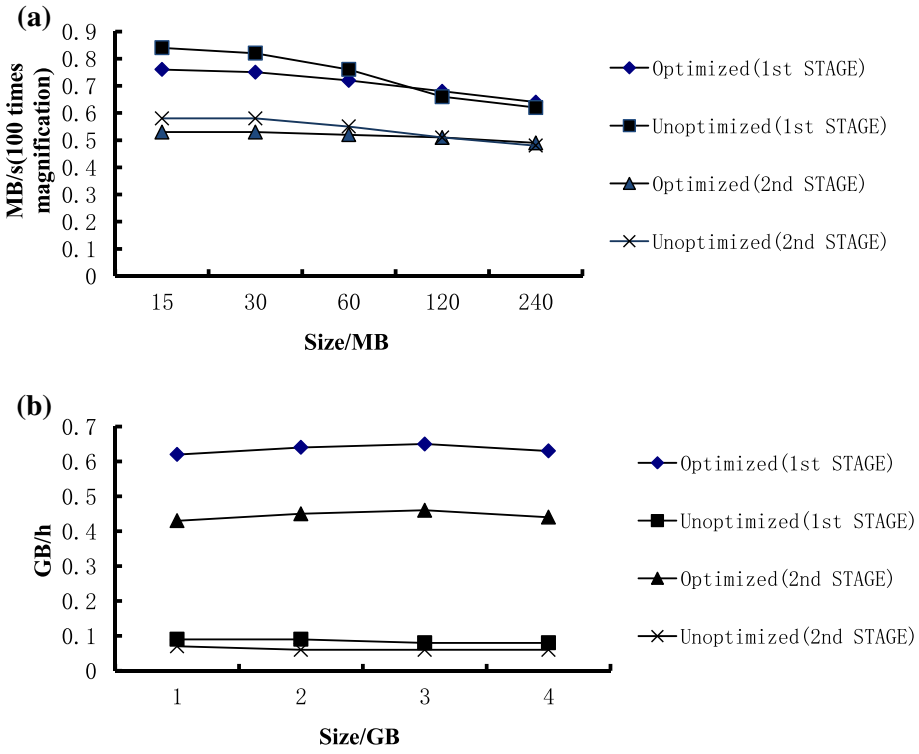


Fig. 4 Comparison between unoptimized and optimized recognitions (speed). **a** Training speed under small training samples. **b** Training speed under large training samples

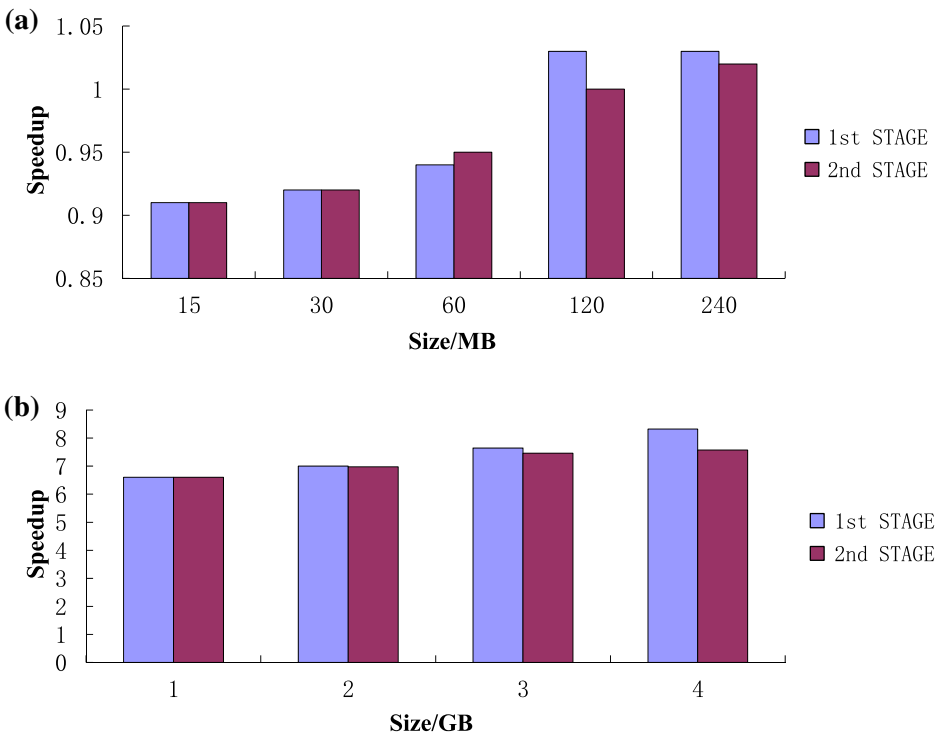


Fig. 5 Parallel speedup. **a** Speedup under small training samples. **b** Speedup under large training samples

compute nodes and Map, with the size of training data being 4GB. In Sect. 3.3, we divide $T = (x_i, y_i)_{i=1}^{M}$ into m parts, and each part is put on a map task. So the parameter m can be used to denote the number of map tasks in this example. Figure 6 gives the results.

It can be seen from the above experiments that for large-scale training samples, the training time is cut down again after increasing the number of cluster nodes and map tasks m. This demonstrates that Maps and performance of the cluster do affect the training efficiency of the model. When the

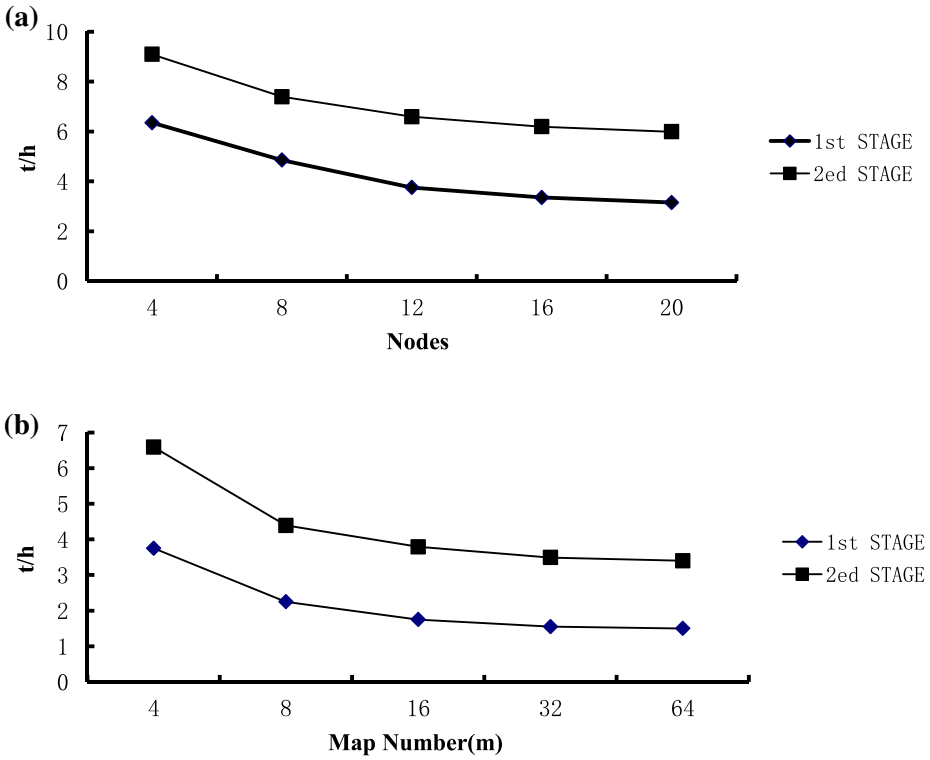


Fig. 6 Influence of the number of Map and compute nodes. **a** Different number of nodes. **b** Different compute Map

number reaches a certain value and continue increasing, the time will almost certainly be moderate.

6 Conclusion

In this paper, we have proposed a two-phase biomedical named entity recognition method using CRFs with some useful features and gave a parallel optimization based on MapReduce for the training algorithm in detail. Our experiments have shown that the useful feature sets could enhance the value of F_1 score and the MapReduce scheme could reduce the training time greatly, especially for large-scale training corpus. However, named entity recognition in biomedical field remains lots of challenges and there is still much room for improvement. For follow-up work, we will focus on enhanced parallel models and methods for large-scale data and try to improve the feature sets to get better performance.

Acknowledgments The authors are grateful to the three anonymous reviewers for their criticism and comments which have helped to improve the presentation and quality of the paper. This work is supported by the Key Program of National Natural Science Foundation of China (Grant No. 61133005), and National Natural Science Foundation of China (Grant Nos. 61370095,61432005).

References

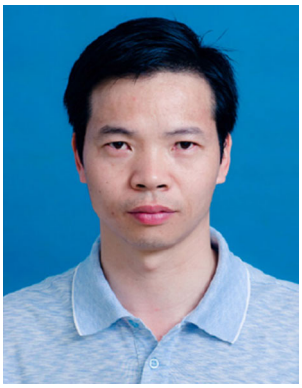
1. Wikipedia, Text mining [EB/OL]. http://en.wikipedia.org/wiki/Text_mining. 24 Oct 2013
2. Wikipedia, Named-entity recognition [EB/OL]. http://en.wikipedia.org/wiki/Named_entity_recognition. 22 Aug 2013
3. Wikipedia, MEDLINE [EB/OL]. http://en.wikipedia.org/wiki/MEDLINE. 14 Sep 2013
4. Lin, J., Dyer, C.: Data-Intensive Text Processing with MapReduce. Morgan and Claypool Publishers, San Francisco (2010). doi:10.2200/S00274ED1V01Y201006HLT007
5. Shen, L., Shen, H., Cheng, L.: New algorithms for efficient mining of association rules. In: The Seventh Symposium on the Frontiers of Massively Parallel Computation, pp. 234–241 (1999)
6. Li, L., Zhou, R., Huang, D.: Two-phase biomedical named entity recognition using CRFs. Comput. Biol. Chem. **33**(4), 334–338 (2009)
7. Finkel, J., Dingare, S., Nguyen, H.: Exploiting context for biomedical entity recognition: from syntax to the web. In: Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications (JNLPBA), pp. 88–91 (2004)
8. Wang, H., Zhao, T., Li, S., Yu, H.: A conditional random fields approach to biomedical named entity recognition. J. Electron. **6**(24), 838–844 (2007)
9. Settles, B.: Biomedical named entity recognition using conditional random fields and rich feature sets. In: Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA), pp. 104–107 (2004)
10. Li, L., Fan, W., Huang, D.: A two-phase bio-NER system based on integrated classifiers and multi-agent strategy. IEEE/ACM Trans. Comput. Biol. Bioinform. (2013). doi:10.1109/TCBB.2013.106
11. Yang, L., Zhou, Y.: Exploring feature sets for two-phase biomedical named entity recognition using semi-CRFs. Knowl. Inf. Syst. (2013). doi:10.1007/s10115-013-0637-7
12. Lee, K.-J., Hwang, Y.-S., Rim, H.-C.: Two-phase biomedical NE recognition based on SVMs. In: Proceedings of the ACL Workshop on Natural Language Processing in Biomedicine (BioMed), pp. 33–40 (2003)

13. Kim, S., Yoon, J., Park, K.-M., Rim, H.-C.: Two-phase biomedical named entity recognition using a hybrid method. In: Proceedings of the 2nd International Joint Conference (IJCNLP), pp. 646–657 (2005)
14. Kim, S., Yoon, J.: Experimental study on a two phase method for biomedical named entity recognition. IEICE Trans. Inf. Syst. **7**(E90–D), 1103–1110 (2007)
15. Li, Lishuang, Zhou, Rongpeng, Huang, Degen: Two-phase biomedical named entity recognition using CRFs. Comput. Biol. Chem. **33**, 334–338 (2009)
16. Wang, L., Ke, L., Liu, P., Ranjan, R., Chen, L.: IK-SVD: dictionary learning for spatial big data via incremental atom update. Comput. Sci. Eng. **16**(4), 41–52 (2014)
17. Wang, L., von Laszewski, G., Younge, A.J., He, X., Kunze, M., Tao, J.: Cloud computing: a perspective study. New Gener. Comput. **28**(2), 137–146 (2010)
18. Wittek, P., Darányi, S.: Accelerating text mining workloads in a MapReduce-based distributed GPU environment. J. Parallel Distrib. Comput. **2**(73), 98–206 (2013)
19. Wang, L., Tao, J., Marten, H., Streit, A., Khan, S.U., Kolodziej, J., Chen, D.: MapReduce across distributed clusters for data-intensive applications. In: The 26th IEEE International Parallel & Distributed Processing Symposium (IPDPS) Workshops 2012: 2004–2011
20. Laclavik, M., Seleng, M., Hluchy, L.: Towards large scale semantic annotation built on MapReduce architecture. Lecture Notes in Computer Science **3**(5103), 331–338 (2008)
21. Wang, L., Tao, J., Ranjan, R., Marten, H., Streit, A., Chen, J., Chen, D.: G-Hadoop: MapReduce across distributed data centers for data-intensive computing. Future Gener. Comput. Syst. **29**(3), 739–750 (2013)
22. Whitney, M., Clifton, A., Sarkar, A., Fedorova, A.: Making the most of a distributed perceptron for NLP. In: Pacific Northwest Regional NLP Workshop, Redmond, Washington, USA (2012)
23. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: 27th Proceedings of the International Conference on Machine Learning (ICML), pp. 282–289 (2010)
24. Atkinson, J., Bull, V.: A multi-strategy approach to biological named entity recognition. Expert Syst. Appl. **39**(17), 12968–12974 (2012)
25. Forney, G.D. Jr.: The viterbi algorithm. In: Proceedings of the IEEE, vol. 3(61), pp. 268–278. Codex Corporation. Newton, MA (2005)
26. Vijay Sundar Ram, R., Akilandeswari, A., Lalitha Devi, S.: Linguistic features for named entity recognition using CRFs. In: International Conference on Asian Language Processing (IALP), pp. 158–161 (2010)
27. Langford, J.: Parallel machine learning on big data, XRDS: crossroads. ACM Mag. Stud. **1**(19), 60–62 (2012)
28. Meraji, S., Tropper, C.: A machine learning approach for optimizing parallel logic simulation. In: 39th International Conference on Parallel Processing (ICPP), pp. 545–554 (2010)
29. Livieris, I.E., Apostolopoulou, M.S., Sotiropoulos, D.G., Sioutas, S., Pintelas, P.: Classification of large biomedical data using ANNs based on BFGS method. In: 13th Panhellenic Conference on Informatics (PCI), pp. 87–91 (2009)
30. Munkhdalai, T., Li, M., Kim, T., Namsrai, O.-E., Jeong, S.-p., Shin, J., Ryu, K.H.: Bio named entity recognition based on co-training algorithm. In: 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 857–862 (2012)
31. Zhang, J., Shen, D., Zhou, G., Tan, C.-L.: Enhancing HMM-based biomedical named entity recognition by studying special phenomena. J. Biomed. Inform. **6**(37), 411–422 (2004)
32. Mathur, A., Chakrabarti, S.: Accelerating newton optimization for log-linear models through feature redundancy. In: 6th International Conference on Data Mining, pp. 404–413 (2006)
33. Nocedal, J.: Updating quasi-Newton matrices with limited storage. Math. Comput. **35**, 773–782 (1980)
34. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. J. Math. Program. B **3**(45), 503–528 (1989)
35. Wang, L., Chen, D., Ranjan, R., Khan, S.U., Kolodziej, J., Wang, J.: Parallel processing of massive EEG data with MapReduce. In: The 18th IEEE International Conference on Parallel and Distributed Systems (ICPADS), pp. 164–171 (2012)
36. Guodong, Z., Jian, S.: Exploring deep knowledge resources in biomedical name recognition. In: Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications (JNLPBA), pp. 96–99 (2004)
37. Okanohara, D., Miyao, Y., Tsuruoka, Y., Tsujii, J.: Improving the scalability of semi-Markov conditional random fields for named entity recognition. In: Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pp. 465–472 (2006)
38. Zhao, Jiaqi, Wang, Lizhe, Tao, Jie, Chen, Jinjun, Sun, Weiye, Ranjan, Rajiv, Kolodziej, Joanna, Streit, Achim, Georgakopoulos, Dimitrios: A security framework in G-Hadoop for big data computing across distributed cloud data centres. J. Comput. Syst. Sci. **80**(5), 994–1007 (2014)
39. Xie, J., Yin, S., Ruan, X., Ding, Z., Tian, Y., Majors, J., Qin, X.: Improving MapReduce performance through data placement in heterogeneous Hadoop clusters. In: IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum (IPDPSW), pp. 1–9 (2010)

Zhuo Tang received the Ph.D. in computer science from Huazhong University of Science and Technology, China, in 2008. He is currently an associate professor of computer science and technology at Hunan University. His research interests include security model, parallel algorithms and resources scheduling for distributed computing systems, grid and cloud computing. He is a member of CCF.

Lingang Jiang is working towards the master degree at the College of Information Science and Engineering, Hunan University of China. His research interests include modeling and scheduling for distributed computing systems, parallel algorithms.

Li Yang received the Ph.D. in computer science from Huazhong University of Science and Technology, China, in 2013. She is currently an assistant professor of Computer and Communication Engineering at Changsha University of Science and Technology. Her research interests include bioinformatics, artificial intelligence, biomedical text mining.



Kenli Li received the Ph.D. in computer science from Huazhong University of Science and Technology, China, in 2003. He was a visiting scholar at University of Illinois at Champaign and Urbana from 2004 to 2005. Now, he is a professor of Computer science and Technology at Hunan University, associate director of National Supercomputing Center in Changsha. His major research includes parallel computing, grid and cloud computing, and DNA computer. He has published more than 70 papers in international conferences and journals, such as IEEE TC, IEEE TPDS, JPDC, ICPP, CCGrid. He is an outstanding member of CCF.

Keqin Li received the B.S. degree in computer science from Tsinghua University, Beijing, in 1985 and the Ph.D. degree in computer science from the University of Houston in 1990. He is currently a full professor of computer science in the Department of Computer Science, State University of New York, New Paltz. His research interests are mainly in the design and analysis of algorithms, parallel and distributed computing, and computer networking, with particular interests in approximation algorithms, parallel algorithms, job scheduling, task dispatching, load balancing, performance evaluation, dynamic tree embedding, scalability analysis, parallel computing using optical interconnects, optical networks, and wireless networks. He has more than 200 research publications. He is a fellow of the IEEE, and a member of the IEEE Computer Society, and the ACM.