Multi-word-Agent Autonomy Learning Based on Adaptive Immune Theories

[1] Xishuang Dong, [2] Xinbo Lv, [3] Yi Guan, [4] Jinfeng Yang
[1,] *Web Intelligence Lab, Research Center of Language Technology,*
School of Computer Science and Technology,
Harbin Institute of Technology, 150001 Harbin, China, dongxishuang@gmail.com
[2,] *Web Intelligence Lab, Research Center of Language Technology,*
School of Computer Science and Technology,
Harbin Institute of Technology, 150001 Harbin, China, lvxinbo@gmail.com
[*3,] *Web Intelligence Lab, Research Center of Language Technology,*
School of Computer Science and Technology,
Harbin Institute of Technology, 150001 Harbin, China, guanyi@hit.edu.cn
[4,] *Web Intelligence Lab, Research Center of Language Technology,*
School of Computer Science and Technology,
Harbin Institute of Technology, 150001 Harbin, China, yangjinfeng2010@gmail.com

Abstract

A multi-word-agent autonomy learning model as a new online learning model is presented based on clonal selection theory and idiotypic immune network theory, where words simulate B cells and antigens involved in adaptive immune responses to construct autonomic immune word-agents under the framework of autonomy oriented computing. Moreover, relations between words simulate specific relations between B cells and antigens, where attributes of words simulate receptors, and strengths of these relations are defined as affinities calculated by matching receptors. The environment simulates the region in which the word-agents interact with each other and share information. The goal of the model is to optimize these relations iteratively by increasing affinities with hyper-mutation of receptors of B cell word-agents. Experimental results on dependency parsing show that not only accuracies are improved continually, but also, compared with MSTParser, high accuracies can be gained with less learning samples.

Keywords*: Word-Agent, Autonomy Learning, Adaptive Immune Theories*

1. Introduction

Online learning models can be taken to regulate weights of parameters constantly in a learning sequence of consecutive rounds [1]. On each round, a labeled sample, which is a pair of questions and answers, is inputted into the model, and an answer to this sample is required to predict by the model. For example, the model might receive an email and be required to predict whether the email is spam or not. The quality of prediction is assessed with a loss function that measures the discrepancy between the predicted answer and the correct one. Losses of an online learning model are measured by these cumulative losses suffered by the learning along its run on a sequence of labeled samples. The model's ultimate goal is to minimize these losses through updating the weights of parameters after each round. Moreover, online learning models allow the learning sequence to be deterministic or stochastic while classic statistical machine learning models enforces strong assumptions on the statistical properties of the learning sequence [2]. Online learning models have two advantages over statistical machine learning based models. One is that the model can regulate weights of parameters without large-scale labeled corpus. The other one is that weights of parameters of the model can be regulated further during usage with use's feedback. Many of online learning models [3-6] are presented. However, these models cannot handle the case of regulating strength of relations between words, which is the key to solving natural language processing (NLP) problems such as dependency parsing [7]. Therefore, we present a multi-word-agent autonomy learning model based on adaptive immune theories that can directly regulate strength of relations between words, which is inspired by similarities between the human language system (HLS) and the immune system (IS).

There are three perspectives of similarities between HLS and IS from microcosmic to macrocosmic levels. The first perspective is about consistencies between immune cells and words [8]: (1) specificity. Specific receptors on the surface of B cells are unique identifications that are different from those of other B cells. For words, each word also has a unique identification. For examples, character sequences of English words can be thought as their identifications; (2) diversity. There are more than 10^7 distinguished B cells in the immune system and 10^5 unique words in the language system. (3) combinability. Receptors on the surface of B cells can be bound specifically with those of antigens to eliminate these antigens. Words combine to form phrases in order to express semantics; (4) dynamics. The bone marrow can generate new B cells that differentiate as plasma and memory B cells and die. Words can also be created or die; (5) completeness. B cells can eliminate all antigens intruding immune systems. All semantics can be expressed by organizing words with syntax. The second perspective is about procedures of these two systems adapting to new elements, which are realized by selection, regulation, and memory [9]. The procedure of adapting new words by HLS is: (1) new words are created; (2) these new words are used on certain syntactic structures; (3) these structures are regulated based on syntax and contents; (4) the words and structures are rememorized. On the other hand, the procedure of eliminating antigens by IS is: (1) new antigens invade the immune system; (2) receptors on the surface of B cells can match those of antigens specifically; (3) these receptors matched are edited by hyper-mutation so as to improve affinities; (4) these cells with higher affinities differentiate plasmas and memory cells, where memory cells can differentiate masses of plasma when stimulated by the same antigens again, and plasmas can secret plant of antibodies to eliminate antigens. The final perspective is that the immune network of B cells and the dependency network of words are all complex networks. A network can be thought as a complex network if properties of the network include [10]: (1) the value of the clustering coefficient (CC), which is used to measure aggregation degree of nodes in the network, is large and stable; (2) degrees of nodes are exponentially distributed. Silva and Santos found that the procedure of constructing relations between nodes in the immune network showed stochastic when the CC was fixed and large. When the network was stable, degrees of nodes were exponentially distributed. Static results from dependency syntax corpus of Chinese [11-12], Czekh, Germany, and Rumania [13], showed that the dependency network of words has these two properties. In addition to these similarities, IS can be regarded as a reinforcement learning system because the ability of protecting bodies can be strengthen constantly by eliminating antigens in the environment. Inspired by these similarities and the reinforcement learning nature of IS [14], NLP problems might be solved with adaptive immune theories. Therefore, multi-word-agent autonomy learning model is presented based on adaptive immune theories.

As a complex system modeling method, autonomy oriented computing (AOC) [15] is employed to construct the autonomy learning model whose elements contain immune word-agents and the environment. Under the framework of AOC, elements of an AOC system are composed of agents and environments. Moreover, each of agents is an entity who realizes goals with autonomy behaviors through interacting with the environment and neighbor agents [16]. And characteristics of agents consist of autonomy, cooperation, learning, and adaptability. In detail, autonomy refers to that agents can select certain behaviors based on the self state and the state of local environment without guidance from a system center controller and achieve goals through cooperating with other agents. Meanwhile, agents can learn and reinforce experiences by interaction between agents and between agents and environments, which is to adapt changes of environments by regulating policies that are rules controlling state transaction. According to similarities between HLS and IS discussed previously, words are viewed as agents called word-agent who can optimize relations between words by interactions among agents. Furthermore, they simulate B cells and antigens to construct immune word-agents whose elements are composed of attributes, states, behaviors, policies, and evaluation functions. The goal of immune word-agents is to optimize their relations, which is achieved by interactions between agents and between agents and the environment that is the place in which agents interact with each other and gain feedbacks. Experimental results of dependency parsing on CoNLL06 data sets show weights of parameters of the model can be regulated constantly to improve prediction accuracies. Moreover, high accuracies can be gained with less learning samples.

The remainder of the paper is organized as follows. In section two, we summarize related work about online learning, AIS, and AOC. Autonomy learning model based on adaptive immune theories is introduced in detail and applied to dependency parsing in section three. In section four, experimental results of the model are presented and analyzed, and its advantages and disadvantages are discussed.

Finally, we conclude that the model can regulate weights of parameters to improve prediction accuracies constantly, and outline future work such as improving efficiency and prediction accuracies.

2. Background

An overview of online learning, AIS algorithms, and AOC, which are frameworks, theories, and modeling methods of the model respectively, is presented below.

2.1. Online learning algorithms

In online learning algorithms, weights of parameters are usually updated in a multiplicative way such as weighed majority algorithm [17] and winnow algorithm [18]. Weighted majority algorithm is designed based on the expert voting policy, which is just to punish the wrong experts though cutting their weights in half. Compared to weight majority algorithm, winnow algorithm is suitable for processing the problem that the number of related features is smaller than that of all features because weights of related features are doubled while weights of unrelated features are cut in half. However, there are two disadvantages in the above regulation procedure. First, no update is made if the predicted answer is correct. Secondly, all parameters are multiplied a fixed value. Therefore, only the current example is used to define the loss for updating parameters, and previous examples are ignored. To overcome these disadvantages, aggressive variants of the perceptron [3] are proposed, where parameters are updated in terms of the average value of k consecutive losses of learning samples so that updates are also performed if the prediction answer of current sample is correct. As for their applications, the usage of online learning models comprises regression [18], dependency parsing [19], ranking [20-21], and image search [22]. However, existing online learning models failed to regulate strength of relations between words directly when solving structure analysis problems in NLP. For example, in dependency parsing, dependency structures of a sentence can be optimized by regulating strengths of relations between words directly [19]. For regulating these relations theoretically, an autonomy learning model based on adaptive immune theories is presented, which is integrated by many kinds of AIS algorithms under the framework of AOC.

2.2. AIS algorithms

AIS algorithms including negative selection algorithms (NSA), idiotypic immune net-work algorithms (IINA), and clonal selection algorithms (CSA) are constructed by simulating functions of immune systems with theories such as negative selection (NS), idiotypic immune network (IIN), and clonal selection (CS). The overview of NSA, IINA, and CSA is organized as basic theories, related algorithms, and applications respectively.

NS theory refers to that T cells bound by self major histocompatibility complex (MHC) with high affinities are eliminated so as to protect self bodies from attracting by T cells [23-24]. In the light of the theory, we need two steps to construct NSA. The first step is generation of detectors on training sets, and then we evaluate these detectors on testing sets. In the first step, detectors are generated randomly after initializing related parameters, and some of them matched self samples with high affinities are eliminated. In step two, detectors left in the first step are used to match testing samples to distinguish self or non-self. If the testing samples match any of detectors, they are classified as non-self, otherwise self. And if the performance is larger than a threshold, constructing the model is accomplished, otherwise go to the first step. In recent years, people focused on detector generation [25] such as simple evolution NSA (SENSA) and basic evolution NSA (BENSA) which were presented based on binary coding of receptors [26]. The creative idea of SENSA is that new detectors generated by evolving iteratively are added into the set of detectors to strengthen the ability of recognizing self. In addition, in BENSA, creating detectors randomly is introduced to improve global the search ability. Moreover, complexity analysis [27] was presented. In the domain of applications, NSA can be used to resolve problems such as the partition of software and hardware in embedded systems [28], aircraft fault detection [29], motor abnormal detection [30], and information security [31].

In respect of immune network, IIN theory indicates that B cells can form a network by stimulus and suppression [32]. Through simulating IIN theory, the earliest IIN algorithm [33] is constructed through

simulating the bone marrow, the B cell network, and a population of antigens. From the bone marrow, B cells are generated to form IIN by matching paratopes with idiotopes, where if the matching degrees are larger than a threshold, we construct an edge between these B cells. Then antigens are injected into some nodes in the network, and B cells in a node clone and mutate to improve affinities if they can recognize the antigens. On the other hand, B cells around the node can bind redundant clones to reduce the number of clones. Therefore, the number proportion of B cells can be regulated to maintain a stable value dynamically in order to realize immune memory [34]. In recent years, structures of the immune network and regulation mechanisms were two points studied. In respect of structures of the immune network, local and directed circle structure of local networks was presented to realize memory [35]. In addition, it is suggested that B cells can form MSTs in local networks to maintain the maximum ability of eliminating antigens [36]. In the regard of regulation mechanism, Coelho presented the concentration based artificial immune network algorithms, where numbers of antibodies are regulated based on concentrations of neighbor antibodies [37]. Considering applications, immune network algorithms have been used in clustering, data visualization, automatic control, and optimization domains [38].

Clonal selection theory refers to that B cells stimulated by antigens adaptively can mutate to generate new B cells with higher affinities who can differentiate into memory B cells and plasma, where memory B cells are stimulated to proliferate as plasma in large quantities, and plasma can secret antibodies to eliminate antigens [39]. According to the theory, main steps to construct the clonal selection algorithm (CSA) include: (1) the population of B cells is initialized randomly; (2) given a set of patterns, each pattern is matched to B cells to compute affinities; (3) these B cells whose affinities are larger than a threshold will mutate to generate new B cells with higher affinities; (4) these new B cells are replicated to generate clones; (5) these clones are added into the population; (6) repeating (2) to (5) until it satisfies with the terminal conditions. Moreover, parallel matching, mutating, and cloning are realized in parallel clone selection algorithms inheriting the parallel characteristics of the immune system [40-41]. In particular, Tiwari combined clonal selection theory with the hierarchy of needs theory [42] including physiological needs, security needs, social needs, growth needs, and own actual needs to present psychoclonal algorithms [43], where these needs are corresponding to initializing of the populations, matching, selecting, mutating, and cloning. In the area of applications, CAS can be applied to optimization of functions [44-45], pattern recognition [46], job scheduling and project scheduling [47], and TSP [48]. Although immune systems are modeled as different AIS algorithms, these models are not constructed from the perspective of the multi-agent complex system (MCS) [49]. Therefore, multi-agent complex system modeling such as AOC should be the best method to model the immune system [50].

2.3. AOC

AOC [15] can be used to construct multi-agent systems consisting of an environment, autonomy agents, neighbors of the agents, and a system target function. It includes three steps to construct a multi-agent system with AOC [51]: (1) to define inputs, outputs, and global targets; (2) to design agents, which contains state spaces, local and global environments, sets of behaviors and policies, and evaluation functions. (3) to distribute agents and obtain optimum solutions by self-organized interactions. Problems are solved with the AOC models through cloning self-organized behaviors of real worlds, where their characteristics are [15]: (1) entities are homogeneous, but their policies can be different; (2) the population of entities is evolved, which means that new entities are generated, and ones without adaptive abilities are died; (3) interactions among entities are local, and there is no center controller to manage entity behaviors; (4) selections of behavior are not random, but satisfied with a distribution and driven by local goals; (5) global targets are defined as system-level object functions; (6) dynamic environments are medium of information shared by entities, where information are updated through interactions between environments and entities. AOC can be applied to complex system modeling such as extracting web user behaviors [52], analyzing dynamics of social networks [53], simulating and analyzing emergence behaviors in HIV responses [54] and self-organized behaviors of agents in multi-agent systems [55], and to solving complex problems such as distributed constraint satisfaction problems [56-57], distributed optimization [58-59], self-organized autonomy network proxy [60], data routing problems in sense networks [61], mining network communities [62-63], modeling robots in a complex system [64-65], and resource distribution in a dynamic network [66-

67]. With AOC, a multi-word-agent autonomy leaning model is presented, where words simulate B cells and antigens, and relations between words simulate specific relations between B cells and antigens.

3. Multi-word-agent autonomy learning model

We simulate words as B cells and antigens involved in adaptive immune responses to construct immune word-agents, and word relations simulate specific relations between B cells and antigens. Word relations can be optimized iteratively by interactions among agents involved continued immune responses in local environments. Immune word-agent is introduced first.

3.1. Immune word-agents

In general, an autonomy agent in the AOC framework is defined as a quintuple $< A, S, B, P, E >$ that donates attributes, states, behaviors, policies guiding the transition of states, and an evaluation function assessing whether a positive feedback can be gained when running a behavior. We define word-agent as follow:

Definition1:
Word-agents are constructed by simulating words as autonomy agents who can select behaviors according to self states and local information to achieve self goals.

Words are viewed as living agents who can decide self behaviors and reinforce their experiences by interacting with other agents and environments. Each word-agent has states and senses local information of environments, and word-agent goals can be realized by strengthening their experiences. We define immune word-agents as follow through simulating words as immune cellular and molecular.

Definition2:
Immune word-agents are word-agents whose attributes, states, behaviors, policies, and the evaluation function are built by simulating B cells and antigens.

Considering an immune word-agent, A contains attributes such as words, part-of-speech, and frequencies of words. S includes active, mutated, matured, and dead. B donates the set of behaviors containing moving, interacting, mutating, selecting, and cloning. P donates a rule set of state transactions. E donates the set of evaluation functions used to modified policies. There are two kinds of immune word-agents named B cell word-agents and antigen word-agents. We introduce B cell word-agents first in detail.

3.1.1. B cell word-agent

3.1.1.1 Attributes

Attributes are designed by simulating receptors on the surface of B cells. The receptors from the point of IIN can be viewed as Figure 1.

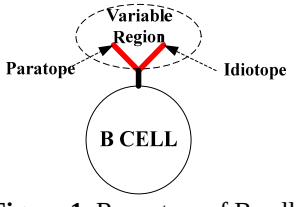

Figure 1. Receptors of B cells

A paratope is the unique set of sites that can match receptors of antigens. An idiotope as a mirror image of an antigen is another unique set of the site that attaches to B cell. Paratopes and idiotopes of

different B cells can recognize each other to stimulus or suppress B cells proliferation. Compared to B cells, attributes of B cell word-agent can be built as Figure 2.

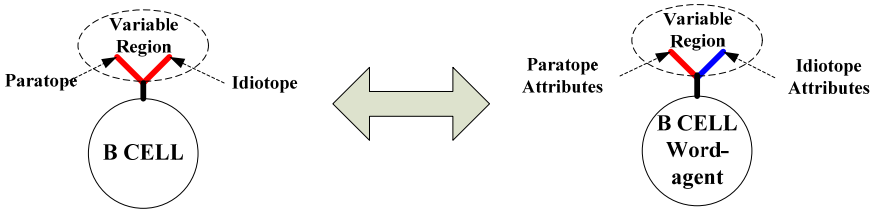

Figure 2. Attributes of B cell word-agent

The A of a B cell receptor consists of paratope attributes and idiotope attributes. Farmer presented that two binary strings donating the paratope and idiotope are used to represent receptors of antibodies, and degrees of matching between paratopes and idiotopes are affinities [34]. We design attributes based on the Farmer's idea as Figure 3.

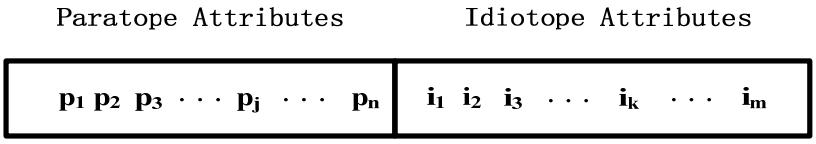

Figure 3. Paratope attributes and idiotope attributes

where n and m donates the number of bits of paratope attributes (PA) and idiotope attributes (IA). p_j and i_k are designed as Figure 4.

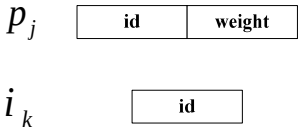

Figure 4. Bits of attributes

Weight in p_j can be used to evaluate the weights of the bit. And the id is the unique identification of a bit. Matching degrees between PA and IA can be defined as equation (1).

$$Affinity(PA, IA) = \sum_{j=1}^{n} \sum_{k=1}^{m} f(p_j, i_k) \tag{1}$$

where $f(p_j, i_k)$ is defined as equation (2).

$$f(p_j, i_k) = \begin{cases} weight & if\ id_{p_j} = id_{i_k} \\ 0 & otherwise \end{cases} \tag{2}$$

3.1.1.2 Behaviors and Evaluation Function

Behaviors of B cell word-agents include moving, interacting, mutating, selecting, and cloning. Moving is defined according to the number of neighbor regions. As an agent, it has eight neighbor regions as Figure 5.

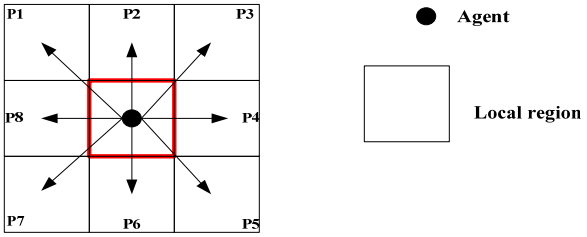

Figure 5. Local regions

where P1, P2, P3, P4, P5, P6, P7, and P8 are neighbor regions. First, the agent senses the number of agents in the current region in which the agent live. Then, the agent senses numbers of eight neighbor regions, and possible target regions where numbers of agents in these regions are less than that of the current region are selected. Finally, the agent randomly selects a neighbor region from the possible target regions as the target moving region. The algorithm is as Figure 6.

Moving

Input : P_{curr}
Output : P_{move}
1. $V_p = 0, P_{new} = 0, n = 0;$
2. $\text{int min} = getAgentCount(P_{curr});$
3. *for* $i : 1 \sim 8$
4. $P_{new} = getNewPosition(i, P_{curr});$
5. $n = getAgentCount(P_{new});$
6. *if* $(n < \text{min})$
7. $V_p.push(P_{new});$
8. *end*
9. $i++;$
10. *end*
11. $P_{move} = randomSelectPosition(V_p);$
12. $mapStateToBehavior(self);$

Figure 6. Moving

where P_{curr} donates the current position. P_{move} donates the target moving region, and V_p donates the possible target regions.

Interacting is realized by adaptive matching attributes between B cell word-agents and antigen word-agents, which means that if a B cell word-agent's neighbor is an antigen word-agent, the self paratope of the B cell word-agent are matched to the antigen's idiotope adaptively. The algorithm is showed in detail in Figure 7.

Interacting

Input : P_{curr}
Output : P_{match}, S_{curr}
1. $Agents = getNeighbours(P_{curr})$;
2. *for* $i: 0 \sim Agents.size$
3. *if* $(isAntigen(Agents_i))$
4. $(P_{match}, affinity) = computeAffinity(selfParatope, antigenIdiotope)$;
5. *if* $(affinity > T_{affinity})$
6. $S_{curr} = mutated$;
7. $S_{curr}^{Agent_i} = died$;
8. $mapStateToBehavior(self)$;
9. $mapStateToBehavior(Antigen\ wordAgent)$;
10. *break*;
11. *end*
12. *end*
13. $i++$;
14. *end*

Figure 7. Interacting

where P_{match} donates the positions of bits of attributes. S_{curr} donates the current state, and $T_{affinity}$ donates the threshold of affinities. Affinity is calculated by equation (1). The agent senses neighbor agents first. Then, if there is an antigen word-agent, and the affinity between two agents is larger than $T_{affinity}$, the state of the B cell word-agent is set mutated, and the state of the antigen word-agent is set died. Finally, these two agents select behaviors as next actions according to the current state.

Mutating is realized based on mutation as the genetic algorithm. Weights of bits of receptors are tuned by mutation. From the point of immunology, mutation is not regular, which means that the bit and increment of mutation cannot be predicted. The algorithm is as Figure 8. Firstly, a random probability is generated for each matched bit. If the probability is larger than a threshold, the mutated increment is added to the weight of the bit as equation (3).

$$f_{mutatedIncrement}(w_i) = \begin{cases} deta & if\ w_i = 0 \\ w_i \times (1+\alpha) & otherwise \end{cases} \quad (3)$$

where *deta* is an defined increment. w_i is the weight of bit *i*, and α donates the mutated factor. Then if the mutated affinity is larger than the primary affinity, the state is set as matured.

Mutating

Input : $P_{match}, S_{curr}, idiotope, affinity$
Output : S_{curr}
1. $mutatedAffinity = 0,\ p = 0;$
2. *for* $i : 0 \sim P_{match}.size$
3. $p = randomGenerateProbability();$
4. *if* $(pro < MUTATEPRO)$
5. *if* $(w_i == 0)$
6. $w_i = deta;$
7. *else*
8. $w_i += \alpha \times w_i;$
9. *end*
10. *end*
11. $i++;$
12. *end*
13. $mutateAffinity = computeAffinity(selfParatope, idiotope);$
14. *if* $(mutateAffinity > affinity)$
15. $S_{curr} = matured;$
16. *end*
17. $mapStateToBehavior(self);$

Figure 8. Mutating

where $MUTATEPRO$ donates the threshold of the probabilities of mutation, and α donates the increment of weights.

Selecting is realized based on a rule and a feedback. The rule is that if the affinity is larger than others', the agent might be selected. Moreover, if the agent's feedback is positive, it might be selected. The algorithm is as Figure 9. Firstly, the agent senses neighbors. Then if the B cell neighbor matching the same antigen word-agents have lower affinities than that of the agent, the neighbor's state is set as active. Finally, if the agent's state is still matured, which means that it is the winner in the procedure of local selecting, and the agent gain a negative feedback, then the agent's state is set as active.

Selecting

Input : $P_{curr}, S_{curr}, mutateAffinity$
Output : S_{curr}
1. $Agents = getNeighbours(P_{curr})$;
2. *for* $i: 0 \sim Agents.size$
3. *if* $(isBCELL(Agents_i))$
4. *if* $(matchSameAntigen(Agents_i, self))$
5. *if* $(mutateAffinity < mutateAffinity_{Agents_i})$
6. $S_{curr} = active$;
7. *break*;
8. *end*
9. *end*
10. *end*
11. $i++$;
12. *end*
13. *if* $(S_{curr} == matured)$
14. *if* $(getNegativeFeedback())$
15. $S_{curr} = active$;
16. *end*
17. *end*
18. $mapStateToBehavior(self)$;

Figure 9. Selecting

The target of the autonomy learning is to optimize adaptive relations between immune word-agents, where the relations are regulated by tuning weights. If the relations are optimized, it will gain a positive feedback defined by the evaluation function as equation (4).

$$f_{evaluation}(agent_i) = \begin{cases} 1 & if \quad \Delta > 0 \\ -1 & otherwise \end{cases} \tag{4}$$

where Δ donates the feedback. For example, if the relations are dependency relations, Δ can be defined as the increment of maximum spanning trees (MSTs) scores. The evaluation function can be defined as equation (5).

$$f_{evaluation}(s_i) = \begin{cases} 1 & if \quad \Delta > 0 \ \ or \ \ sc < 0 \\ -1 & if \quad sc(s_i) > 0 \\ 0 & otherwise \end{cases} \tag{5}$$

where s_i donates the sentence i. Δ is defined as equation (6), and $sc(s_i)$ donates the errors of dependency relations generated by mutation defined as equation (7).

$$\Delta = \arg\max'(\sum_{se_i \in T} se_i(w, w')) - \arg\max(\sum_{se_i \in T} se_i(w, w')) \tag{6}$$

$$sc(s_i) = \sum_{e_i' \in T'} e_i'(w, w') - \sum_{e_i'' \in T''} e_i''(w, w') \quad (7)$$

where se_i donates strength of dependency relations between word *w* and *w'*. T donates the right MST while T' and T'' are two wrong MSTs. e_i donates dependency relations between words as equation (8).

$$e_i(w, w') = \begin{cases} 1 & if\ the\ dependency\ relation\ is\ correct \\ 0 & otherwise \end{cases} \quad (8)$$

Cloning is realized based on mutated affinities and regulations by IIN, which can improve efficiency of immune defenses in second immune responses. The clone algorithm is shown as Figure 10.

Cloning

Input : P_{curr}, *mutateAffinity*
Output : S_{curr}
1. $stimulus = 0, suppression = 0, sti = 0, sup = 0;$
2. $Agents = getNeighbours(P_{curr});$
3. *for* $i : 0 \sim Agents.size$
4. *if* $(isBCELL(Agents_i))$
5. $sti = computeAffinitiy(selfParatope, idiotope_{Agent_i});$
6. $sup = computeAffinitiy(selfIdiotope, paratope_{Agent_i});$
7. $N_{stimulus} += sti;$
8. $N_{suppression} += sup;$
9. *end*
10. $i++;$
11. *end*
12. $cloneNumber = 1;$
13. $N_{difference} = N_{stimulus} - N_{suppression};$
14. *if* $(N_{difference} > 0)$
15. $cloneNumber += 1;$
16. *else if* $(N_{difference} < 0)$
17. *if* $(|N_{difference}| > mutateAffinity)$
18. $cloneNumber -= 2;$
19. *else*
20. $cloneNumber -= 1;$
21. *end*
22. *end*
23. *if* $(cloneNumber > 0)$
24. $copyAgents(cloneNumber, self);$
25. *else*
26. $removeAgents(cloneNumber, self);$
27. *end*
28. $S_{curr} = active;$
29. $mapStateToBehavior(self);$

Figure 10. Cloning

Firstly, the agent senses its neighbors. Then, according to neighbors, stimulus and suppression of the agent can be calculated as equation (9) and (10).

$$N_{stimulus} = \sum_{u \in U} f_{affinity}(selfParatope, idiotope_{Agent_i}) \quad (9)$$

$$N_{suppression} = \sum_{u \in U} f_{affinity}(selfIdiotope, paratope_{Agent_i}) \quad (10)$$

Finally, the clone number is determined as equation (11).

$$N_{clone} = \begin{cases} 1 & if\ N_{difference} > 0 \\ 0 & if\ N_{difference} \le 0\ and\ |N_{difference}| < mutateAffinity \\ -1 & if\ N_{difference} \le 0\ and\ |N_{difference}| > mutateAffinity \end{cases} \quad (11)$$

where $N_{difference}$ is defined as equation (12).

$$N_{difference} = N_{stimulus} - N_{suppression} \quad (12)$$

3.1.1.3 State and Policy

According to molecular immunology, B cells have specific states such as active, mutated, and matured. Therefore, B cell word-agents have the same states. Active state, that is the initial state, refers that agents can move and recognize other agents. When receptors of agents match others' adaptively, the agents are in the matched state. Mutated state refers that B cell word-agents matched other agents enter into the state, and weights of receptors can be tuning by mutating. If the agents are in the matured state and gain positive feedbacks, they can be replicated by cloning. Policies for transferring states can be shown as Figure 11.

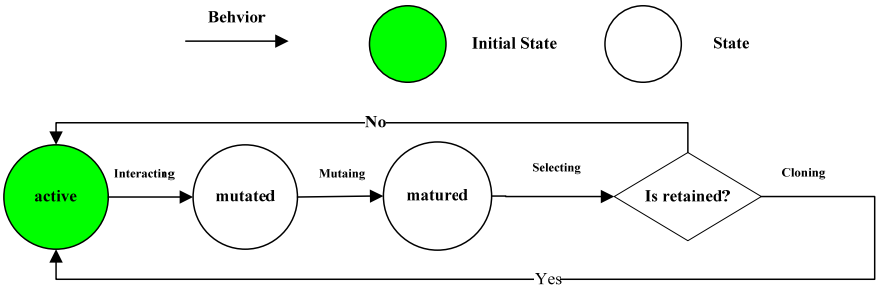

Figure 11. Policy of B cell word-agent

3.1.2. Antigen word-agent

Antigen word-agents are quadruples $< A, S, B, P >$. According to idiotypic immune network, idiotype of B cell is a mirror of receptors of antigens. Therefore, Attributes of antigens is designed as idiotypes of B cells. Behaviors include moving and interacting that are the same to those of B cell word-agents. States include active and died states. Policies are shown as Figure 12.

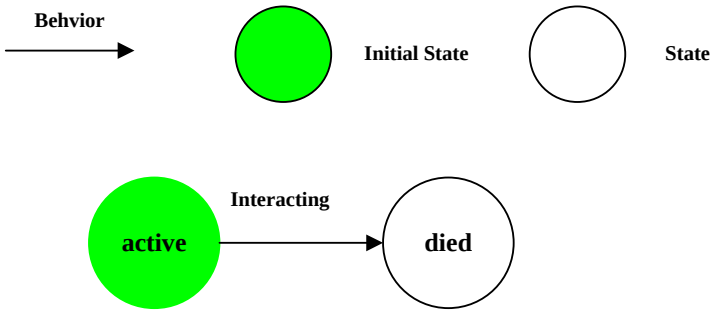

Figure 12. Policy of antigen word-agent

3.2. Environment and system object function

The environment, which is divided as local regions as Figure 13, is the place where agents interact and share information.

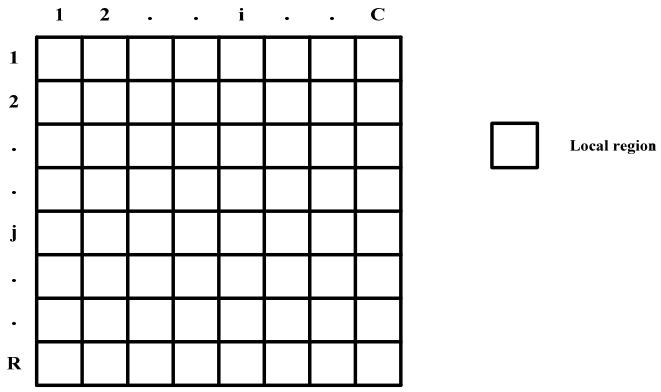

Figure 13. Environment

where C denotes the number of columns, and R denotes the number of rows. Agents interact in local regions, and results are stored in the local regions. From the point of machine learning, each agent contains a part of parameters tuned by mutation. The results of weights regulated can be stored in the environment. The system object function can be defined as equation (13).

$$f_{system} = \arg\max \sum_{i=1}^{N} \sum_{j=1}^{T} \sum_{k=1}^{K} f_{evaluation}(agent_k) \tag{13}$$

where N donates learning times. T donates the number of samples learned, and K donates the number of immune word-agents. The object of autonomy learning is to maximize the count of the positive feedback.

3.3. Immune word-agent autonomy learning

The target of immune word-agent autonomy learning is to optimize adaptive relations between immune word-agents. The algorithm of the learning system is shown as Figure 14.

RegulateByImmuneTheories

Input : *learning data* : $T = \{(x_t, y_t)\}_{t=1}^{T}$
Output : w
1. $w_0 = 0; v = 0; i = 0;$
2. *initializeImmuneSystem*(T);
3. *for* $n : 1 \cdots N$
4. *for* $t : 1 \cdots T$
5. $Ags = constructAntigens((x_t, y_t))$;
6. $w^{i+1} = regulateWeightsInImmuneSystem(Ags)$;
7. $i = i + 1$;
8. *end*
9. $v = v + w^{i+1}$;
10. $w^{i+1} = v / (N \times T)$;
11. *end*
12. w = w^{i+1};

Figure 14. Immune word-agent autonomy learning

where N donates the times of learning, and T donates learning samples. A single learning instance is considered in each learning time and divided as B cell word-agents and antigen word-agents, and parameters as paratope attributes of B cell word-agents are updated by immune word-agent interactions. The algorithm returns an average weight vector: an auxiliary weight vector v is maintained so that it accumulates the values of w after each learning time, and the returned weight vector is the average of all the weight vectors throughout learning because averaging has been shown to help reduce over-fitting [68].

Moreover, the initialization of the learning is shown as Figure 15.

initializeImmunSystem

Input : *learning data* : $T = \{(x_t, y_t)\}_{t=1}^{T}$
Output : *BAgents*
1. $P_{local} = 0;$
2. *for* $t : 0 \sim T$
3. $P_{local} = randomSelectPosition()$;
4. $BAgents = constructBAgents((x_t, y_t))$;
5. $distributeAgent(BAgents, P_{local})$;
6. $t++$;
7. *end*

Figure 15. Initialization of the immune system

where P_{local} donates local positions. B cells in the immune system are distributed randomly. And, considering a learning sample, B cells are built by breaking samples that are based on rules and specific questions solved.

3.4. Dependency parsing with the model

The task of dependency parsing is to construct accurately dependency relations between words in a sentence [69-70]. Dependency parsing models can be classified as data driven models built on sentences labeled dependency relations with statistical machine learning models and grammar driven models constructed with grammars built by experts or extracted from corpus. The procedure to construct data driven models includes two steps: (1) learning: given a training set S_{train}, we extract dependency features from S_{train}, and the model M is induced in infinite steps with these features; (2) evaluating: given a testing set S_{test} and an evaluation method E, if the performance is above an expected value E_p, the procedure of constructing M is finished, otherwise go to step (1). Graph based models in data driven models gained attentions because of their high prediction accuracies. Dependency parsing is realized with these models by constructing MSTs where nodes are words, and edges are dependency relations between words. For example, results of dependency parsing on the sentence "汉族/nR 医学/n 又/d 有/v 中医/n 之/uJDE 称/n" are shown as Figure 16.

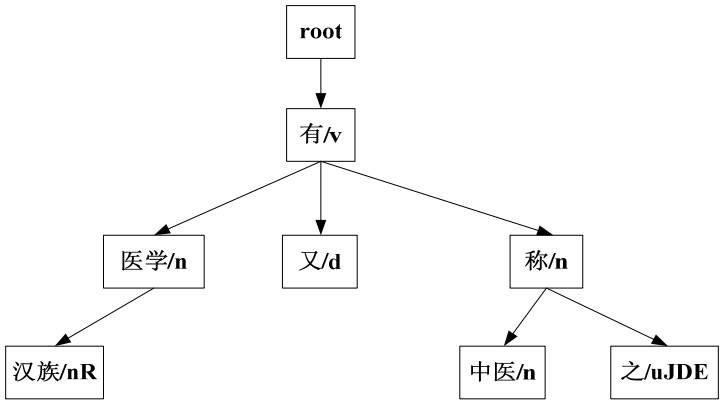

Figure 16. Results of dependency parsing

The basic flow of constructing graph based models is: (1) to build learning and testing corpus; (2) to define feature templates; (3) to extract and select features; (4) to set parameters; (5) to learn weights of the model from learning corpus by constructing MSTs; (6) to evaluate the model, if the performance value of the model is larger than the expected value, building the model is finished, otherwise go to step (4);

We construct AISParser for dependency parsing based on the multi-word-agent autonomy learning model. Attributes of word-agents are designed first. Through comparing B cells in idiotypic networks and words in dependency parsing networks, we found that as shown in Figure 17.

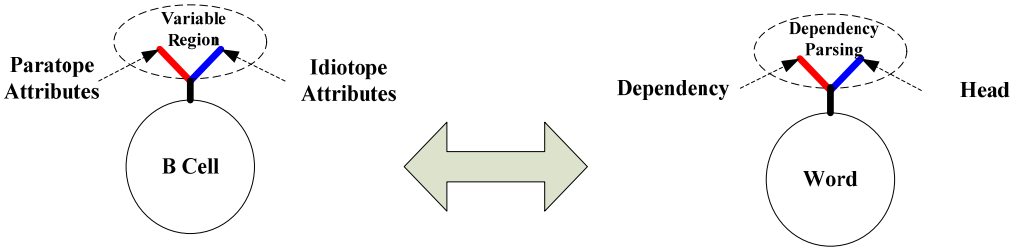

Figure 17. Comparing B cell receptors and words in dependency parsing network

From comparing B cell word-agents with words, dependency relations are similar to specific immune relations. Therefore, attributes of word-agents for dependency parsing is designed naturally as Figure 18.

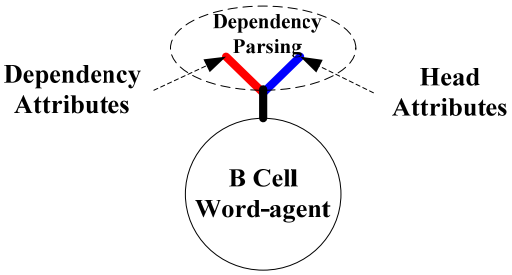

Figure 18. Attributes of B cell word-agent

Moreover, the bit in A is designed as Figure 19.

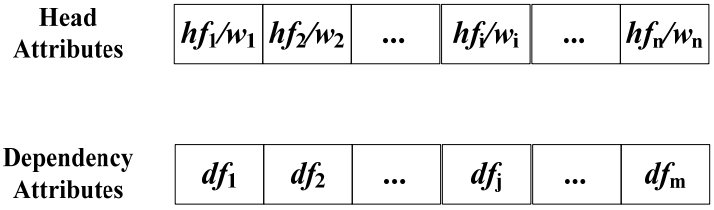

Figure 19. Bits in attributes

where df_i and hf_j donate head and dependency features, and n and m are numbers of features. w_i donates the weights of features. There are four steps to construct the receptor of a B cell word-agent: (1) to extract pairs of words that are heads in these pairs; (2) to extract features; (3) to remove redundancy of features to construct paratope; (4) to construct idiotope like step (1) to (3). Others for a word-agent are constructed based on simulating B cells and antigens, which is introduced from section 3.1 to 3.2. And the learning procedure is accomplished, which is shown in section 3.3.

4. Experiment and analysis

4.1. Data and experiment sets

Data sets, feature templates, the baseline system, and the evaluation method are introduced in this section. Chinese Penn Treebank 5.1 (CTB 5.1) [71] as data sets is divided into learning data and testing data. The form is shown as shared tasks as CoNLL 2006/2007. Penn2Malt tool and the head-finding rules [72-73] are used to accomplish phrase-to-dependency conversion. Feature templates are shown as Table 1.

Table 1. Feature templates of dependency

Word	Part-of-speech (POS)	Word and POS
p-word, c-word, p-word_c-word	p-pos, c-pos, p-pos, c-pos, p-pos_p-pos+1_c-pos-1_c-pos, p-pos-1_p-pos_c-pos-1_c-pos, p-pos_p-pos+1_c-pos_c-pos+1, p-pos-1_p-pos_c-pos_c-pos+1	p-word_c-word_c-pos, p-word_p-pos_c-word, p-word_p-pos_c- pos, p-pos_c-word_c-pos, p-word_p-pos_c-word_c-pos

In the Table 1, p-word donates head words, c-word donates dependency words, p-pos donates head POS, c-pos donates dependency POS, p-word+1 donates right words nearby head POS, p-word-1 donates left words nearby head POS, p-pos+1 donates right words nearby dependency POS, and c-pos-1 donates left words near to dependency POS. MSTParser [19] is the baseline system because it is an online learning based model that is similar to our model. We adopt the unlabeled evaluation method of dependency parsing to evaluate the performance, which is defined as equation (14).

$$Accuracy = \frac{correct\ labels}{total\ labels} \tag{14}$$

4.2. Results and analysis

4.2.1. Experiments on a small-scale data Set

We selected 100 learning samples from the learning set and 50 testing samples from the testing set. The number of features for dependency parsing is 20,564. System parameters are set according to human experiences as Table 2.

Table 2. System parameters for small-scale data experiments

Parameters	Values
MUTATEPRO	0.08
α	1e-5
deta	0.1
Number of rows	5
Number of columns	5
Number of learning times	50

Four experimental results with the same system parameters are shown as Figure 20.

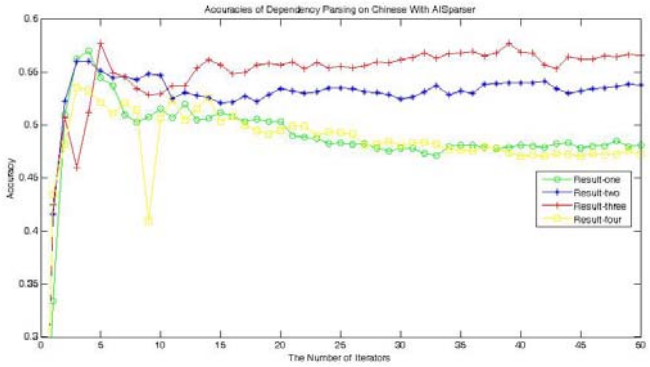

Figure 20. Results of learning in different learning rounds

From Figure 20, we can gain the following conclusions: (1) the model can tune parameters iteratively to improve the accuracies according to a sample. Therefore parameter values can be updated without considering the whole corpus, which is improve learning efficiencies; (2) the convergence values of four results are different because mutation bits are chosen randomly. The same mutation bits and increments cannot be insured in different running times; (3) the accuracy values are always improved. Sometimes they might be lower than previous values, but they can be improved by tuning parameters. For figuring out whether different system parameters affect accuracies, we compare different results of different system parameters as Figure 21.

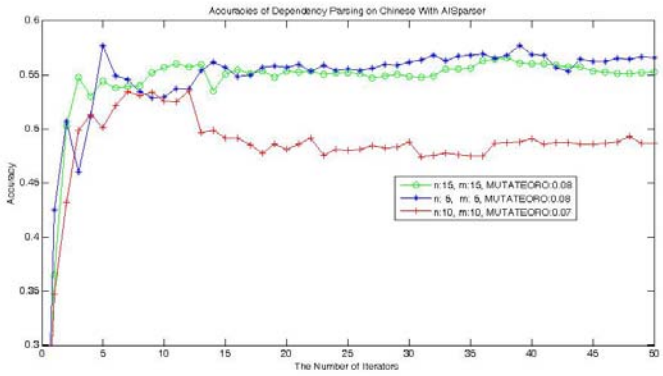

Figure 21. Results of different system parameters

Where n donates the number of columns of local regions, and m donates the number of rows. From Figure 21, different system parameters can lead to different accuracies. Therefore, we should set parameters elaborately to gain high accuracies. Moreover, we compare accuracies of our model with those of MSTParser as Fig.22.

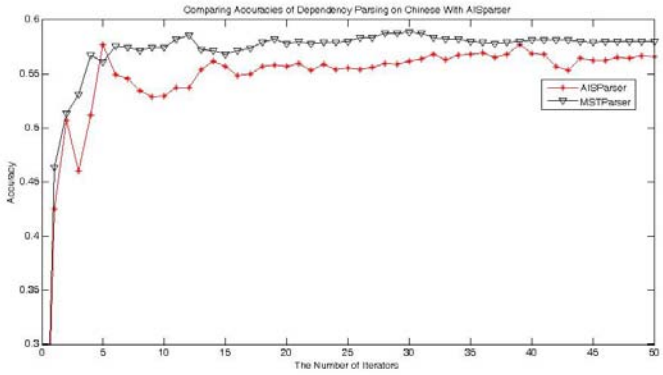

Figure 22. Comparisons of performances between AISParser and MSTParser

From Figure 22, we find that: (1) the highest accuracy of AISParser is 0.577,506 and that of MSTParser is 0.587,486. These two values are almost the same. (2) the accuracy of AISParser is converged to a high and stable value. (3) the accuracies of AISParser is not stable because of its random when learning iteratively.

4.2.2. Experiments on a large-scale data

We selected 1000 learning samples from the learning set and 300 testing samples from the testing set. The number of features for dependency parsing is 122,314. System parameters are set as table 3.

Table 3. System parameters for large-scale data experiments

Names of Parameters	Values
MUTATEPRO	0.06
α	1e-5
deta	0.1
Number of rows	15
Number of columns	15
Number of learning times	20

The results are shown as Figure 23.

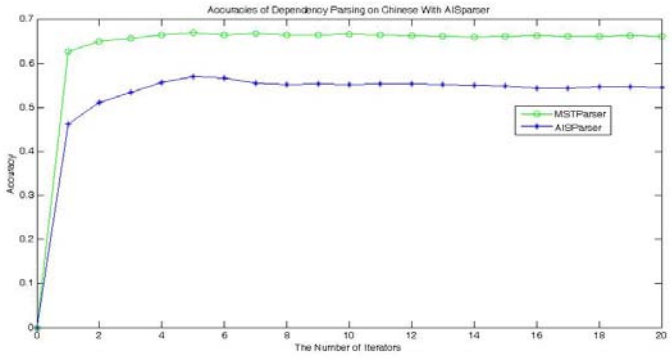

Figure 23. Results of comparison on 1000 Samples

From Figure 23, we can gain the following results: (1) the model can tune parameters iteratively, and the performance can be improved; (2) the accuracy of AISParser is lower than that of MSTParser significantly; (3) the accuracy of AISParser is converged to a little low and stable value because when the number of features are large, its ability of tuning is not strong enough to achieve a high accuracy. Maybe we should find a new way to realize mutations and tune the quantity in a small step to gain a high accuracy; (4) the accuracies of AISParser is not stable because of its random when learning iteratively;

4.2.3. Experiments on different scale data sets

We selected different scale samples from the learning set and 300 testing samples from the testing set. System parameters are set as table 4.

Table 4. System parameters for different scale data experiments

Number of Samples	MUTATEPRO	α	deta	Number of rows	Number of columns
10	0.07	1e-5	0.1	15	15
100	0.01	1e-5	0.1	5	5
200	0.08	1e-5	0.1	15	15
500	0.08	1e-5	0.1	15	15
1000	0.6	1e-5	0.1	15	15

The accuracy comparison of AISParser and MSTPaser is shown in Figure 23.

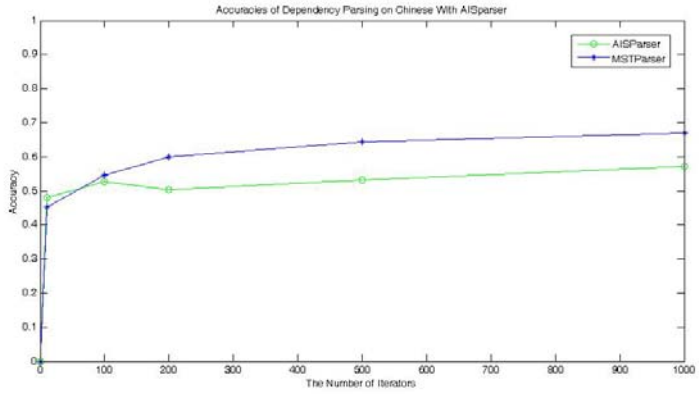

Figure 24. Results of comparison with different parameters

From Figure 24, we can gain the following results: (1) the accuracy of AISParser are higher than that of MSTParser when the number of learning sample is ten, which means that the optimization ability of

dependency tree structures of AISParser is stronger than that of MSTParser; (2) with increment of numbers of learning samples, the accuracy of AISParser are lower than that of MSTParser because local optimum results emerge, which means that current methods to tuning parameters based on random mutations cannot adapt to the task of large-scale parameter regulation.

5. Conclusion and future work

We present a multi-word-agent autonomy learning model based on clone selection theory and idiotypic immune network theory. Under the framework of AOC, we simulate words as B cells and antigens involved in adaptive immune responses, and word relations simulate specific relations between B cells and antigens. Word relations can be optimized iteratively by interactions between immune word-agents in continued adaptive immune responses. The main advantage is that performances can be improved by tuning parameters constantly with clonal selection theory and idiotypic immune network theory. Moreover, the model is well-understood because people do not have to master advanced mathematical knowledge. Experiment results of dependency parsing on Chinese show the effectiveness of constructing dependency trees of sentences by optimizing dependency relations between words with the model. Future work could include: (1) to improve accuracies of dependency parsing and reduce volatility of results by introduce more immune theories such as negative selection theory and danger theory; (2) to realize a parallel version of this model in order to improve model efficiencies; (3) to apply this model to other domains. For example, this model is applied to sentiment analysis on texts by constructing sentiment immune word-agents.

6. Acknowledgement

This work was supported by National Natural Science Foundation of China (Project No. 60975077) and National Natural Science Foundation of China (Project No. 90924015)

7. References

[1] Shai Shalev-Shwartz, “Online Learning and Online Convex Optimization”, Foundations and Trends in Machine Learning. vol. 4, no. 2, pp.107-194, 2011.
[2] Avrim Blum, “On-Line Algorithms in Machine Learning”, In Proceedings of the Workshop on On-Line Algorithms, pp. 306-325, 1996.
[3] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain”, Psychological Review, vol.65, no.6, pp.386-407, 1958.
[4] K. Crammer, Y. Singer, “Ultraconservative online algorithms for multiclass problems”, Journal of Machine Learning Research, vol.3, no. 4-5, pp. 951-991, 2003.
[5] J. Kivinen, M.Warmuth, “Relative loss bounds for multidimensional regression problems”, Journal of Machine Learning, vol.45, no.3, pp.301-329, 2001.
[6] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, Y. Singer, “Online passive aggressive algorithms”, Journal of Machine Learning Research, vol.7, pp. 551-585, 2006.
[7] Sandra Kübler, Ryan McDonald, Joakim Nivre, Dependency Parsing, Synthesis Lectures on Human Language Technologies, Morgan and Claypool Publishers, 2009.
[8] Niels K. Jerne, “The Generative Grammar of the Immune System”, Nobel lecture, 1984.
[9] Hershberg, U., Efroni, S., “The immune system and other cognitive systems”. Complexity, vol.6, no.5, pp.14-21, 2001.
[10] H. Souza-e-Silva, R. M. Z. dos Santos, “Is the immune network a complex network?”, arXiv.org, pp.1-10, 2012.
[11] Liu Haitao, “The Complexity of Chinese Syntactic Network”, COMPLEX SYSTEMS AND COMPLEX ITY SCIENCE, vol. 4, no.4, pp.38-44, 2007.
[12] LIU Zhiyuan, ZHENG Yabin, SUN Maosong, “Complex Network Properties of Chinese Syntactic Dependency Network”, COMPLEX SYSTEMS AND COMPLEXITY SCIENCE, vol.5, no.2, pp.37-45, 2008.
[13] Ferrer i Cancho, R. Solé, and R. Köhler, “Patterns in syntactic dependency networks,” Physical Review E, vol. 69, no. 5, pp.1-8, 2004.

[14] De Castro, L.N.; Von Zuben, F.J., "Learning and optimization using the clonal selection principle", IEEE Transactions on Evolutionary Computation, vol.6, no.3, pp.239-251, 2002.
[15] Jiming Liu, Xiaolong Jin, Kwok Ching Tsui, "Autonomy-Oriented Computing (AOC): Formulating Computational Systems With Autonomous Components", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, vol.35, no.6, pp. 879-902, 2005.
[16] Mark d'Inverno, M. Luck, Understanding Agent Systems, Springer, 2004.
[17] N. Littlestone, M. Warmuth, Relating data compression and learnability, Technical report, University of California Santa Cruz, Snata Cruz, CA, 1986.
[18] N.Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm", Machine Learning, vol.2, no.4, pp.285-318, 1988.
[19] Ryan McDonald, Koby Crammer, Fernando Pereira, "Online large-margin training of dependency parsers", In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05), pp.91-98, 2005.
[20] Massimiliano Ciaramita, Vanessa Murdock, Vassilis Plachouras, "Online Learning from Click Data for Sponsored Search", In Proceedings of the 17th international conference on World Wide Web (WWW '08), pp.227-236, 2008.
[21] Pannagadatta K. Shivaswamy, Thorsten Joachims, "Online Learning with Preference Feed-back", arXiv:1111.0712, pp.1-5, 2011.
[22] Mohamed Aly, "Online Learning for Parameter Selection in Large Scale Image Search", In Proceedings of 4th IEEE Online Learning for Computer Vision Workshop (OLCV), pp.35-42, 2010.
[23] Forrest, S., Perelson, A., Allen, L., R., Cherukuri, "Self-nonself discrimination in a computer", In Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, pp.202-212, 1994.
[24] Dasgupta, D., Forrest, S., "An anomaly detection algorithm inspired by the immune system", Artificial Immune System and Their Applications, Springer-Verlag, 1999.
[25] Dasgupta, D., Yu, S., Nino, F, "Recent Advances in Artificial Immune Systems: Models and Applications", Applied Soft Computing, vol.11, no.2, pp.1574-1587, 2011.
[26] W. Luo, J. Wang, X. Wang, "Evolutionary negative selection algorithms for anomaly detection", In Proceedings of 8th Joint Conference on Information Science (JCIS2005), pp.440-445, 2005.
[27] B. Xu, W. Luo, X. Pei, M. Zhang, X. Wang, "On average time complexity of evolutionary negative selection algorithms for anomaly detection", In Proceedings of the 2009 World Summit on Genetic and Evolutionary Computation (2009 GEC Summit), pp. 631-638, 2009.
[28] Y. Zhanga, W. Luo, Z. Zhang, B. Li, X. Wang, "A hardware/software partitioning algorithm based on artificial immune principles", Applied Soft Computing, vol.8, no. 1, pp.383-391, 2008.
[29] Dasgupta, D., KrishnaKumar, K., Wong, D., Berry, M., "Negative selection algorithm for aircraft fault detection", In G. Nicosia et al. (Eds), In Proceedings of the third International Conference on Artificial Immune Systems (ICARIS 2004), pp.1-13, 2004.
[30] Gan, Z., Zhao, M. B., & Chow, T. W. S.. "Induction machine fault detection using clone selection programming", Expert System with Applications, vol.34, no.4, pp.8000-8012, 2009.
[31] Dasgupta, D., Gonzalez, F., "An immunity-based technique to characterize intrusions in computer networks", IEEE Transactions on Evolutionary Computation, vol.6, no.3, pp.281-291, 2002.
[32] N. Jerne, "Towards a Network Theory of the Immune System". Ann. Immunol, vol.125C, no.1-2, pp. 373-389, 1974.
[33] Y. Ishida, "Fully distributed diagnosis by PDP learning algorithm: towards immune network PDP model", In Proceedings of IEEE International Joint Conference on Neural Networks, San Diego, pp.777-782, 1990.
[34] J.D. Farmer, N.H. Packard, A.S. Perelson, "The immune system, adaptation, and machine learning", Physica D, vol.2, no.1-3, pp.187-204, 1986.
[35] C.-M. Ou, C. R. Ou, "Immune Network and Immune Memory Mechanism: Perspectives on Dynamical Systems," In Proceedings of 2011 International Conference on Technologies and Applications of Artificial Intelligence, pp.313-318, 2011.
[36] Madi A, Kenett DY, Bransburg-Zabary S, Merbl Y, Quintana FJ, et al, "Network Theory Analysis of Antibody-Antigen Reactivity Data: The Immune Trees at Birth and Adulthood", PLoS ONE vol.6, no.3, pp.1-11, 2011.

[37] G. Coelho, F. Von Zuben, R. Takahashi, K. Deb, E. Wanner, S. Greco, "A Concentration-Based Artificial Immune Network for Multi-objective Optimization," Evolutionary Multi-Criterion Optimization, vol. 6576, pp. 343-357, 2011.
[38] Timmis, J., Neal, M., Hunt, J., "An artificial immune system for data analysis". BioSystems, vol.55, no.1, pp.143-150, 2000.
[39] De Castro LN, Von Zuben FJ, "Learning and optimization using the clonal selection principle", IEEE Trans Evolut Comput, vol.6, no. 3, pp. 239-251, 2002.
[40] Gong M, Jiao L, Yang J, Liu F, "Lamarckian learning in clonal selection algorithm for numerical optimization", International Journal on Artificial Intelligence Tools, vol.19, no. 1, pp.19-37, 2010.
[41] Gong M, Jiao L, Zhang L, "Baldwinian learning in clonal selection algorithm for optimization", Information Sciences, vol.180, no.8, pp.1218-1236, 2010.
[42] Maier, N. R. F, Psychology in Industry, Boston: Houghton-Mifflin, 1965.
[43] Tiwari MK, Prakash Kumar A, Mileham AR, "Determination of an optimal assembly sequence using the PsychoCA", Journal of Engineering Manufacture, vol.219, pp.137-149, 2005.
[44] Liu X, Shi L, Chen R, Chen H, "A novel clonal selection algorithm for global optimization problems", In Proceedings of International conference on information engineering and computer science, pp.1-4, 2009.
[45] Yang J, Sun L, Lee HP, Qian Y, Liang Y, "Clonal selection based memetic algorithm for job shop scheduling problems", Journal of Bionic Engineering, vol.5, no.2, pp.111-119, 2008.
[46] Garain U, Chakraborty MP, Dasgupta D, "Recognition of handwritten indict script using clonal selection algorithm", In Proceedings of the 5th international conference on Artificial Immune Systems, pp.256-266, 2006.
[47] Moghaddam MZ, Kardan A, "Clonal selection algorithm for partitioning and scheduling of codesign systems", In Proceedings of the 5th International colloquium on signal processing and its applications, pp. 267-272, 2009.
[48] Yong L, Sunjun L, "A hybrid model for solving TSP based on artificial immune and ant colony", In Proceedings of International conference on information engineering and computer science, pp.1-5, 2009.
[49] Rocha, Luis M, "Complex Systems Modeling: Using Metaphors From Nature in Simulation and Scientific Models", BITS: Computer and Communications News. Computing, Information, and Communications Division, 1999.
[50] Wolfram S., "Cellular Automata as Models of Complexity", Nature, vol.311, pp.419-424, 1984.
[51] Jiming Liu, "Autonomy-Oriented Computing (AOC): Formulating Computational Systems With Autonomous Components", In Proceedings of the fourth international conference on natural computation, pp.3-11, 2008.
[52] J. Liu, S. Zhang, J. Yang, "Characterizing Web usage regularities with information foraging agents", IEEE Transactions on Knowledge and Data Engineering, vol.16, no.5, pp.566-584, 2004.
[53] S. Zhang, J. Liu, "From local behaviors to the dynamics in an agent network", In Proceedings of WI'06, pp.572-580, 2006.
[54] S. Zhang, J. Liu, "A massively multi-agent system for discovering HIV-immune interaction dynamics", In Proceedings of MMAS'04, pp.161-173, 2004
[55] J. Liu, B. Hu, "On emergent complex behaviour, selforganised criticality and phase transitions in multi-agent systems: Autonomy Oriented Computing (AOC) perspectives", International Journal of Modelling, Identification and Control, vol.3, no. 1. pp.3-16, 2008.
[56] X. Jin, J. Liu, "From individual based modeling to Autonomy Oriented Computation", In Agents and Computational Autonomy, pp.151-169, 2003.
[57] J. Liu, H. Jing, Y. Y. Tang, "Multi-agent oriented constraint satisfaction", Artificial Intelligence, vol.136, no.1, pp.101-144, 2002.
[58] K. C. Tsui, J. Liu, "Multiagent diffusion and distributed optimization", In Proceedings of AAMAS'03, pp.169-176, 2003.
[59] X.-F. Xie, J. Liu, "How Autonomy Oriented Computing (AOC) tackles a computationally hard optimization problem", In Proceedings of AAMAS'06, pp.646-653, 2006.
[60] M. J. Kaiser, K. C. Tsui, J. Liu, "Self-organized autonomous Web proxies", In Proceedings of AAMAS'02, pp.1397-1404, 2002.
[61] L. Gan, J. Liu, X. Jin, "Agent-based, energy efficient routing in sensor networks", In Proceedings of AAMAS'04, pp. 472-479, 2004.

[62] B. Yang, J. Liu, “An Autonomy Oriented Computing (AOC) approach to distributed network community mining”, In Proceedings of SASO’07, pp.151-160, 2007.
[63] B. Yang, J. Liu, “Discovering global network communities based on local centralities”, ACM Transactions on the Web, vol.2, no. 1, pp.1-32, 2008.
[64] J. Liu, J. Wu, Multiagent Robotic Systems, CRC Press, Boca Raton, FL, USA, 2001.
[65] Jianxian Cai, Jingsong Yang, "Autonomous Learning Strategy Research for Mobile Robot", JCIT: Journal of Convergence Information Technology, Vol. 8, No. 1, pp. 290 - 297, 2013
[66] J. Liu, X. Jin, Y. Wang, “Agent-based load balancing on homogeneous minigrids: Macroscopic modeling and characterization”, IEEE Transactions on Parallel and Distributed Systems, vol.16, no.7, pp.586-598, 2005.
[67] Anwar AlYatama, Kassem Saleh, Nidal Nasser, "A Novel Hierarchical and Heterogeneous Mobile Agent-Based Wireless Sensor Networks (HHMA-WSN) Architecture", JNIT: Journal of Next Generation Information Technology, Vol. 3, No. 4, pp. 1 ~ 9, 2012
[68] M. Collins, “Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms”. In Proceedings of the EMNLP 2002, pp.1-8, 2002.
[69] J. Nivre,. “Dependency Grammar and Dependency Parsing”, MSI report 05133, 2005.
[70] Wenliang Chen, Min Zhang, Haizhou Li, “Utilizing Dependency Language Models for Graph-based Dependency Parsing Models”, In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pp. 213-222, 2012.
[71] Nianwen Xue, Fei Xia, Fu-Dong Chiou, Martha Palmer, “The Penn Chinese Treebank: Phrase Structure Annotation of a Large Corpus”, Natural Language Engineering, vol.11, no. 2. pp.207-238, 2005.
[72] Yue Zhang, Stephen Clark, “A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing”. In Proceedings of EMNLP 2008, pp.562-571, 2008.
[73] Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, Haizhou Li, “Joint models for Chinese POS tagging and dependency parsing”, In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp.1180-1191, 2011.