A Novel Heuristic Error-Driven Learning for Recognizing Chinese Time Expression

He Ruifang, Qin Bing, Liu Ting, Pan Yuequn, Li Sheng
School of Computer Science and Technology
Harbin Institute of Technology, Harbin, 150001, China
{rfhe, bqin, tliu, yqpan}@ir.hit.edu.cn, lisheng@hit.edu.cn

Abstract

Recognizing time expression is useful in many natural language processing tasks, which can be used to temporal reasoning and anchoring events on the time line. In this paper, a heuristic error-driven learning framework is proposed for recognizing Chinese time expression, which integrates the heuristic search strategy A^* algorithm into error-driven learning. The heuristic function is designed and its monotonicity is theoretically proved, so that the correctness of A^* algorithm is guaranteed. Our method begins with time trigger word, uses Chinese dependency parsing to identify the extents of time expressions, availably resolves the problem of long distance dependency, and greatly improves the system performance; Subsequently, comparing incorrectly recognized time expressions with the standard ones and learning some rules, then we use the error-driven learning based on the A^* algorithm to heuristically filter the rules, which not only decreases the time complexity of learning rules, but also distinguishes the validity of each rule and the compatibility among rules. We evaluated this new method on the Chinese corpus of ACE2005 and got 6% increase of system performance. Finally, F = 77.96%, F = 77.92% was obtained on the closed and the open test set, respectively.

Keywords

Chinese time expression recognition, time trigger word, dependency parsing, heuristic error-driven learning, A algorithm*

1 Introduction

Understanding semantic of text is the ultimate goal of natural language processing, and temporal semantic is necessary for understanding text. Time expression recognition is one of the key technologies about temporal semantic labeling, which is crucial for temporal reasoning and anchoring events (Wu et al. 2005). It is valuable in question-answering (e.g., answering a "when" question by temporally anchoring events; answering a "how long" question by measuring the duration of the event), information extraction (e.g., normalizing information for database entry), summarization (e.g., chronologically ordering events on a timeline) and machine translation (e.g., recognizing tense to make translation more fluent) (Yang 2006). Recently, the relevant evaluation plans also reflect the importance of recognizing time expression. For example, TERN (Time Expression Recognition and Normalization), relation extraction and event extraction tasks in Automatic Content Extraction (ACE)[1] evaluation and TempEval in SemEval-2007[2]. TempEval involves the automatic identification of temporal referring expressions, events and temporal relations within a text. However, recognizing time expression is the first step.

Motivated by the growing interest in practical NLP applications, temporal information processing has absorbed more attention recently than ever. In ACL'2001, a workshop on temporal and spatial information processing was held, which detailedly discussed the temporal data annotation scheme and corpus construction. That was the first formal gathering in this area. In 2004, ACM Transaction on Asian Language Information Processing (TALIP) had published a special issue about temporal information processing. (Mani 2004) and (Wong et al. 2005) also gave good reviews about recent trend of temporal information processing. Mani indicated that there are three technologies urgently needed to develop, such as the ability to tag events and time expressions, to temporally anchor and order events, and to build models of the temporal structure of discourse. In this paper, we focus on Chinese time expression recognition.

TERN evaluation in ACE began in 2004, which is to identify the extents of temporal expressions in the surface text, and assign values of temporal attributes. The foundation work derives from the US government's DARPA TIDES program, which developed the TIMEX2 annotation scheme for marking the extent of English time expressions (with TIMEX2 tags) and normalizing their values. The TIMEX2 scheme has been adopted in part by the government in the ACE program's Relation Detection and Characterization (RDC)

[1] ACE2007 evaluation plan. http://projects.ldc.upenn.edu/ace/intro.html
[2] SemEval-2007. http://nlp.cs.swarthmore.edu/semeval/index.shtml

and Event Detection and Characterization (EDC) tasks. TimeML[3] scheme discussed in two Advanced Research and Development Activity (ARDA) summer workshops is also extended based on TIMEX2 scheme. At present, the TERN task will be supported for three of the ACE languages (Arabic, Chinese, and English). Some researchers also extended the TIMEX2 scheme to Korean, French, and Spanish (Jang et al. 2004; Vazov 2001; Estela et al. 2002), and did some initial work. Wenjie Li from Hong Kong Polytechnic University and Kam-fai Wong from the Chinese University of Hong Kong focus on Chinese temporal information processing (Wong et al. 2005). Mani from Georgetown University and so on focuses on English temporal information processing (Mani 2004). The participants in the 2004 TERN evaluation evinced a notable split in their approaches: systems performing only recognition were all machine-learning based, while systems performing the full recognition and normalization task were purely rule-based. Since time expression is relatively canonical, rule-based method is mostly adopted in normalization task, which could use the rules from recognition period. The system with best performance adopted more than 1000 handcrafted rules. However, the cost is great and it isn't easy to be extended to other domains. Generally speaking, rule-based method (Wu et al. 2005; Jang et al. 2004; Vazov 2001; Estela et al. 2002) is employed through analyzing structure rules inside time expressions and constraint rules outside time expressions. Nevertheless, it is difficult to identify the long distance dependent time expressions with their complex structures. Machine learning based method (Ahn et al. 2005; Hacioglu et al. 2005) considers time expression recognition as a sequence labeling task, which is an interesting attempt. Such a data-driven approach may be preferable due to its portability. Yet data sparseness exists in long distance dependent time expressions, which results in low recall of machine learning based method. Thus the improving space of performance is limited.

Considering the limitation of rule-based method and the data sparseness of sequence labeling based method, a new approach to recognizing Chinese time expression based on dependency parsing and heuristic error-driven learning is proposed in this paper. The merit of dependency parsing lies in its capability of conveniently expressing phrase structure, which is beneficial to resolve the problem of the long distance dependency; then error-driven learning can restrain the negative influence from the time expression recognition based on dependency parsing. The strategy of filtering rules in error-driven learning directly influences the performance and the performing efficiency of system. Brill (Brill 1995) had applied error-driven learning to a number of natural language problems, including POS tagging, prepositional phrase attachment disambiguation, and syntactic parsing. He adopted the greedy strategy to filter rules, yet this method is time-consuming.

[3] http://www.timeml.org/site/index.html

In order to overcome this limitation, Ruifang he (He 2007) employed batch learning strategy to filter rules, but which cannot distinguish the validity of each rule. This paper considers the filtering rule in error-driven learning as a search question in artificial intelligence. We applied the A^* search strategy to filter rules in order to decrease the learning cost and distinguish the validity of each rule and compatibility among rules. A new machine learning framework for combining A^* search strategy with error-driven learning is proposed in this paper.

The rest of the paper is organized as follows: Section 2 gives question definition and analysis; Section 3 describes the basic idea of heuristic error-driven learning in time expression recognition and the initial recognition method based on dependency parsing; Section 4 discusses the rule set automatically obtained by error-driven learning and the strategies of filtering rules; Section 5 presents the experiment setting and results analysis; We conclude and discuss the future work in Section 6.

2 Question definition and analysis

Time expression recognition in TERN is to identify the extents of temporal expressions in surface text, which are defined as chunks of text which convey direct or indirect temporal information (Wu 2005). Information about time first appeared in MUC (Message Understanding Conference) (Sundheim and Chinchor 1995; Chinchor 1998), which just included absolute and relative time information as part of general tasks of named entity recognition. In comparison with MUC, TIMEX2 annotation scheme gives a good description for different types of time expressions. They are shown in Table 1.

Type	Example
Absolute time	1998 年 6 月 17 日/ June 17,1998
Relative time	昨天、下周 / yesterday, next week
Duration time	两周/ two weeks
Set-denoting time	每小时/ every hour
Event-anchored time	第三周贴图活动揭晓/ third week posted activity announcement
Culturally-determined time	马年/ horse year
Non-specific time	对同志像*春天*般温暖，对敌人像*冬天*般冷酷/ to comrade like *spring* that kind of warmth, to enemies like *winter* that kind of cold

Table 1. Time expression types and examples in TERN

Through analyzing corpus, we found some characteristics: firstly, Chinese time expression contains time trigger word marking its existence, which is defined as a word or numeric expression whose meaning conveys a temporal unit or concept, such as “分钟\minute” or “天\day”. Furthermore, to be a time trigger word, the referent must be able to be oriented on a timeline, or at least oriented with relation to a time (past, present, future). Though some words are related to time, they cannot be oriented on timeline. They are defined as non time trigger word, such as “周刊\weekly” or “光年\light-year”. However, adjective non time trigger word are permitted within the extent of a time expression, such as “*早*些年*early* years” or “*下*个月*next* month”. Another kind of noun non time trigger word cannot be appeared within the extent of a time expression, such as “更年期\menopause”, which is defined as time stopping word. Secondly, the syntactic heads of the expressions are mostly time trigger words through dependency parsing. Dependency relation describes linguistic structure in terms of dependencies between words. Thus time expression can be identified by beginning with time trigger word through dependency curved arrows. Thirdly, statistics demonstrate that POS of 50% time expressions is “nt”(“nt” is taken from Chinese POS set, representing the time word) . These expressions can be regarded as time trigger words. If adopting dependency parsing to identify time expressions with one word or less words, whose extents will be extended. Consequently, we employed error-driven learning to automatically acquire rules and revise the wrongly recognized time expressions in order to reduce the noise brought by initial recognition based on dependency parsing. Therefore, we propose a new approach to combining dependency parsing with error-driven learning integrating A^* algorithm to identify the extents of time expressions. We will explore the validity of dependency parsing in long distance dependent time expression recognition, and furthermore how the heuristic error-driven learning restrains the negative influence from initial recognition.

3 Error-driven learning based Chinese time expression recognition

3.1 Basic idea

Error-driven learning was proposed by E.Brill in 1992 (Brill 1995), which was used to automatically acquire rules. It is apt to learn some latent linguistic rules and also has the advantage of combining linguistic characteristics with other models.

Figure 1 illustrates how heuristic error-driven learning based Chinese time expression recognition works, where rules are used to revise the incorrect time expressions recognized by dependency parsing. The algorithm is as follows:

1) Recognize time expressions in raw corpus based on Chinese dependency parsing;
2) Compare the initial recognition results with the standard results, and then acquire the candidate revised rules, which embody the triggering environments of rules.
3) Filter rules according to heuristic search strategy A^* until recognition performance isn't improved.

During the course of initial recognition, we use four types of dictionaries relevant to the analysis of Section 1. There are time trigger word dictionary, time stopping word dictionary, prefix dictionary and postfix dictionary, which are semi-automatically acquired.

1) Time trigger word dictionary construction

Standard time expressions in training corpus are handled by dependency parsing. The ones with POS "nt" are taken out. Then we extract the syntactic heads, and manually filter the words dissatisfying the definition of time trigger word.

2) Time stopping word dictionary construction

Let *A* time expressions annotated by word segmentation and POS labeling. Let *B* words with POS "nt". Then we can get the time stopping words: $B-(A\cap B)$, which are also filtered by hand according to the definition of time stopping word.

3) Prefix and postfix dictionary construction

Due to the property of dependency parsing, some modifiers of time expressions cannot be found by dependency relation, which are always prefixes or postfixes of time phrases. They haven't very strong characteristics, which are artificially collected, such as prefixes "大概\probably" or "以后\after", postfixes "左右\about" or "初\early".

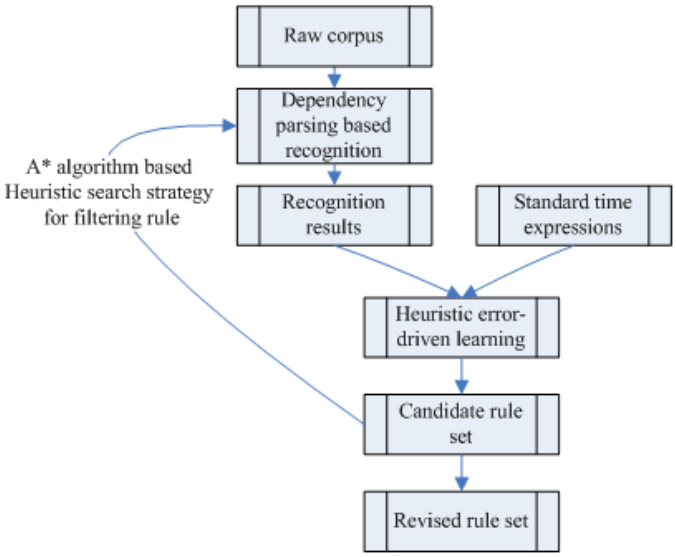
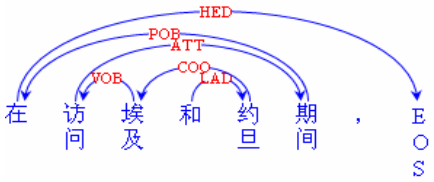


Figure 1. Heuristic error-driven learning based Chinese time expression recognition

3.2 Dependency parsing based time expression recognition

Dependency grammar was proposed by French linguist L.Tesniere in 1959, which has a deep effect on the development of linguistics. The syntactic structure of dependency grammar mainly includes dependency relations, say, binary relations between word pairs in a sentence, where one element is syntactic head, and another element is dependent word. Dependency relation expresses the semantic association between syntactic head and dependent word. Figure 2 shows the dependency parsing of an example.



HED：核心\ Head　POB：介宾关系\Prep Object　ATT：定中关系\Attribute　VOB：动宾关系\Verb Object　COO：并列关系\ Coordinate　LAD：前附加关系\Left Adjunct

Figure 2. Dependency parsing of “在访问埃及和约旦期间，\during the course of visiting Egypt and Jordan,”

A curved arrow in Figure2 between two words denotes the existence of dependency relation, which describes linguistic structure in terms of dependencies between words. Word at the beginning of a curved arrow is dependent on word at the end of ones. Tags on top of curved arrows, such as HED, POB and so on, denote types of dependency relations. For example, there is a dependency relation between “在\during” and “EOS”, and “在\during” is dependent on “EOS”; Special symbol <EOS> dominates the core elements of entire sentence. Dependency grammar is apt to express the phrase structure, say, which is easy to describe the long distance dependency relation between words. It is important for time expression recognition due to two reasons. One is that the syntactic heads of the time expressions are mostly time trigger words through dependency parsing, the other is that the time expressions with complex phrase structure, such as the event anchored time expression, generally have long distance dependency relations. Thus the extent of time expression can be identified at the beginning of time trigger word through dependency curved arrows. For example, as shown in figure 2, the event anchored time expression “访问埃及和约旦期间\during the course of visiting Egypt and Jordan” is identified at the beginning of time trigger word “期间\course” through dependency relations ATT, VOB, COO, LAD.

The core process of dependency parsing based time expression recognition is as the following: firstly, adopting Chinese dependency parsing from Information Retrieval Lab,

Harbin Institute of Technology, P.R.China, we preprocess the ACE corpus, including word segmentation, labeling POS, and dependency parsing. Then we scan sentence from right to left. If meeting with time trigger word or non time stopping word with POS “nt”, we begin with this word to recursively identify the extent of time expression through dependency relations. Dependent word can be found through syntactic head, thus dependent word of dependent word can also be recursively found until dependency curved arrow is broken. Due to the property of dependency parsing, some modifiers of time expressions cannot be found by dependency relation. Therefore, we need to modify the extent of time expression by using prefix and postfix dictionary. For instance, a sentence “在等了 15 分钟左右，\after waiting for about fifteen minutes.” , where “15 分钟左右\about fifteen minutes” is the standard time expression, we begin with time trigger word “分钟\minute” to recognize it, and just can get “15 分钟\fifteen minutes”. Consequently, we need to modify the extent with postfix “左右\about”.

Though the initial recognition can resolve the problem of long distance dependency to some extent, yet it still brings some noise. When time expressions with one word or less words are recognized, noise will be more serious. For example, time expression “11 号\on the 11th ”, if we begin with time trigger word “号\date” to identify it through dependency relation, we will get such wrong recognition result “印尼的雅奇省 11 号\ on the 11th at the Yaqi province of Indonesian”. In order to reduce noise, we will further adopt error-driven learning to automatically acquire rules, and revise the wrong results derived from dependency parsing based recognition.

4 Rule set learned by error-driven and strategy of filtering rules

4.1 Rule set

Let *A*, *B* denote initial recognition results and standard time expressions in Figure 1, respectively. We partition *A* and *B* into three subsets, $A=\{C_A,F_A,R_A\}$, $B=\{C_B,M_B,F_B\}$. C_A denotes the correct results in initial recognition, which corresponds to C_B, and C_A=C_B; F_A denotes the incorrectly recognized results, whose extents are wrongly extended, and corresponds to F_B, $F_A \neq F_B$. Let $a_i \in F_A, b_i \in F_B$, i=1…n, time expression a_i and b_i have common letters. M_B denotes the unrecognized standard time expressions, which haven’t been found counterpoint in A, $M_B \cap A=\varnothing$. R_A denotes the incorrectly recognized time expressions, which haven’t an intersection with B, $R_A \cap B=\varnothing$. Thus (F_A, F_B) and M_B form the corpus of error-driven learning. Our rule is made of time trigger word and pattern, which is formalized as $r(t,p)$. Time trigger word t is chosen from the last word of each

time expression in F_B and M_B. Pattern p includes POS information. For example, a sentence through word segmentation and POS labeling "印尼/ns 的/u 雅奇省/ns11/m 号/q 估计/v 有/v 大约/d40 万/m 的/u 民众/n 连续/a 第两/m 天/q 集会/v 示威/v，/wp"(about 400,000 people have demonstrations and rallies on the 11th at the Yaqi province of Indonesian), where standard time expression "11/m 号/q"(on the 11th) is wrongly recognized as "印尼/ns 的/u 雅奇省/ns11/m 号/q"(on the 11th at the Yaqi province of Indonesian) by dependency parsing based method. Through error-driven learning, the time trigger word of the learned rule is "号\date", and its pattern is "m+q". "m", "q" denotes the POS of standard time expression "11/m 号/q", respectively. Thus, the triggering environment of this rule can be described as the following: when the time triggering word "号\date" is scanned and the POS of the two words before it are "m" and "q", the incorrectly labeled time expression will be revised. All learned rules are written into rule file. If triggering environment is satisfied, the relevant rule will be used to revise the wrong result. Instead of revising the incorrect results, we can use the rules to have a new recognition. For example, "印尼的雅奇省 11 号\ on the 11th at the Yaqi province of Indonesian" can be revised as "11 号\on the 11th" by adopting rule ("号\date","m+q").

4.2 Strategies of filtering rules

Candidate rules learned by each iteration of error-driven process are excessive, so we need further to choose the valid rules. Filtering strategy of rules directly influences the performance and the running efficiency of system, which essentially belongs to search question in artificial intelligence. At present, since mature algorithm for learning rule doesn't exist, we just can use the background knowledge to go ahead step by step. During the process of resolving problem, there are some questions, such that how to find available knowledge and how to ascertain reasoning path in order to pay as little as possible. The process like that we need to continuously find available background knowledge according to practical problem so as to construct a reasoning path with cost as little as possible for satisfactorily resolving the problem is called search (Wang 1998).

During the process of search, we adopt state space representation to formalize the problem, which includes *state* and *operator*. *state* denotes recognition state of each iteration in error-driven learning, and *operator* is the learned rule. State space can be expressed as a triple: (S_0, F, S_g). S_0 is the initial state, F is the set of operators, and S_g is the goal state whose performance cannot be improved.

State space formed by search can be represented as tree structure. S_0 is the initial state

produced by dependency parsing based time expression recognition. Thus first generation rules $R_1^1, R_2^1 ..... R_n^1$ are produced by the first iteration of error-driven learning. When starting rules to have recognition again, set of new states $\{S_1^1, S_2^1 ... S_i^1 ... S_n^1\}$ is formed. If we choose a certain state to extend further, the second generation rules $R_1^2, R_2^2 ..... R_n^2$ are produced by the second iteration, and so on, a search tree will be formed.

The goal of filtering rules in error-driven learning is to find a rule path from S_0 to S_g with least cost, which will bring the most great improvement of performance. During the process of search, the key step is how to choose the next extended state. Different methods of choosing state correspond to different search strategies.

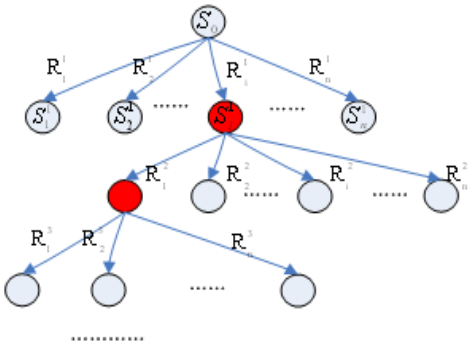


Figure 3. State space corresponding to greedy filtering strategy

4.2.1 Greedy filtering strategy

(Brill 1995) adopted greedy strategy to filter rules in resolving POS labeling with error-driven learning. State space produced by this strategy is shown in Figure 3. In each iteration of error-driven learning, he exhaustively started each rule to test the performance of system, and chose the next started rule according to its tested performance. However, the search cost is great. Let n the number of rules learned by each iteration, l the length of rule path from S_0 to S_g, thus time complexity is at least $O(n^l)$. We know the complexity is very high. This learning strategy is blind search, which has not used the characteristic of the problem itself. Therefore, the search space is huge. There are also many unusable states and running efficiency is not high.

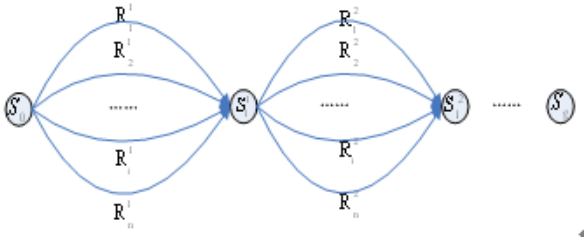


Figure 4. State space corresponding to batch filtering strategy

4.2.2 Batch filtering strategy

Basic idea of batch filtering strategy is to start all rules produced by each iteration of error-driven learning to recursively learn new rules until the performance is not improved. State space formed by this method is shown in Figure 4. We get the state S_0 through the initial time expression recognition based on dependency parsing. The initial recognition results are compared with the standard results, and then we acquire the first generation rules $R_1^1, R_2^1 \ldots\ldots R_n^1$. In the second iteration, we adopt the dependency parsing and $R_1^1, R_2^1 \ldots\ldots R_n^1$ to recognize the time expressions. Then we get the state S_1^1 and the second generation rules $R_1^2, R_2^2 \ldots\ldots R_n^2$. Similarly, we could reach the state S_g. Let l the length of rule path from S_0 to S_g, thus time complexity is at least O(l). Theoretically speaking, batch filtering strategy can effectively reduce time complexity in comparison with greedy filtering strategy, nevertheless, which cannot distinguish the validity of each rule.

4.2.3 Heuristic filtering strategy

Greedy filtering strategy can distinguish the validity of each rule, but search space is huge, and running efficiency is not high. Batch filtering strategy has high performing efficiency, however, which cannot distinguish the validity of each rule. Consequently, most of rules are useless. We adopt heuristic search A^* algorithm to overcome the above limitation, and the relevant search tree is similar to Figure 3. What is different from greedy filtering strategy is that heuristic information relevant to the problem is considered during the course of search so as to guide search go ahead along the optimal direction, but not starting each rule to test the performance of system. In each iteration, we adopt the dependency parsing and just a valid rule judged by heuristic information to recognize time expressions. This process chooses the state node with maximum possibility of performance improvement to extend

the search so as to accelerate the convergence speed and help to find the optimal solution.

The key step of A^* algorithm is how to define the evaluation function (Wang 1998; Luger 2004; Hart et al. 1968), which is used to estimate the importance of node x in search tree according to heuristic information. What is different from ordinarily heuristic search is that the evaluation function of A* algorithm must satisfy some constraint conditions, including admissibility, optimality and monotonicity of heuristic function $h(x_n)$.

Let evaluation function $f^*(x_n) = g^*(x_n) + h^*(x_n)$. $g^*(x_n)$ is the cost of the optimal rule path from start node S_0 to the current node x_n, $h^*(x_n)$ is the practical cost of the optimal rule path from the current node x_n to the final node S_g Thus, $f^*(x_n)$ is the practical cost of the optimal rule path from the start node to the final node. Though $f^*(x_n)$ is oracles, yet it doesn't exist in the realistic problem. We hope evaluation function $f(x_n)$ is an approximation of $f^*(x_n)$, say, $f(x_n) = g(x_n) + h(x_n)$. $g(x_n)$, the cost of path reaching the current state, is an approximation of $g^*(x_n)$, and $g(x_n) \geq g^*(x_n)$; $h(x_n)$ is the heuristic approximation about minimum cost of path from the current state to the goal state, which is the lower bound of $h^*(x_n)$, and $h(x_n) \leq h^*(x_n)$.

The relevant research (Wang 1998) has proved admissibility, optimality of A^* algorithm. In this paper, let $g(x_n)$ the depth of node x_n in search tree, here, $h(x_n)$ is defined according to the practical problem, which is needed to satisfy the monotonicity constraint condition, say, must have the following two properties (Wang 1998):

1. Heuristic information of the goal state is zero, $h(S_g) = 0$;
2. In search graph, let x_j is the arbitrary sub-node of x_i (i, j=1...n), then $h(x_i) - h(x_j) \leq c(x_i, x_j)$, where $c(x_i, x_j)$ is the cost of edge from x_i to x_j.

Next, we give the definition of heuristic function $h(x_n)$, and prove its monotonicity.

Under the background of Chinese time expression recognition, this paper considers the filtering rule in error-driven learning as the search question in artificial intelligence. Our goal is to find the optimal rule path with the most improvement of performance, in fact, the valid rules. Here, we define the heuristic function as the following:

$$h(x_n) = \begin{cases} \dfrac{1}{p(r_i(x_{n-1}))}, & \exists r_i(x_{n-1}); \\ 0, & \neg\exists r_i(x_{n-1}); \end{cases}$$

For every state node x_n, $h(x_n)$ is dependent on the heuristic information provided by x_{n-1}.

If $r_i(x_{n-1})$ exists, $r_i(x_{n-1})$ will be the *i*th rule produced by node x_{n-1}; $p(r_i(x_{n-1}))$ is the priority of $r_i(x_{n-1})$, where the priority means the number of incorrect time expressions $r_i(x_{n-1})$ learns from. Say, we acquire the rule $r_i(x_{n-1})$ by comparing $p(r_i(x_{n-1}))$ wrongly recognized time expressions with the relevant standard time expression. At the same time, if we start $r_i(x_{n-1})$, there will probably be $p(r_i(x_{n-1}))$ incorrectly recognized time expressions to be revised. For each state, it can have many subsequent state according to different rules, thus, $h(x_n)$ is a piecewise function. The higher the priority of $r_i(x_{n-1})$, the higher the probability that we start $r_i(x_{n-1})$ to revise the incorrectly time expressions. Thus in each iteration of error-driven learning, if we adopt the dependency parsing and the unstarted rule with highest priority to recognize time expressions, the most improvement of system performance will probably be brought. Simultaneously, the estimation cost of arriving new state is the least and we extend the old state along the optimal rule path. Therefore the monotonicity of heuristic function $h(x_n)$ is guaranteed. The higher the priority, the faster the improvement of system performance. It makes search go ahead along the direction with the fastest speed until the goal state is arrived. If there is retrospect, say, the above search strategy cannot improve the performance, we will continue to extend this node according to the rule next to the highest priority in order to have a high search efficiency.

If $r_i(x_{n-1})$ doesn't exist, say, we reach the goal state S_g, there is no new rule to be produced. It is unnecessary to have a heuristic error-driven learning, and $h(x_n)=0$.

About the monotonicity of heuristic function $h(x_n)$, we prove it in the following:

Proof:

1. Under the background of this paper, triple（S_0，F，S_g）is solvable state space. Search based on solvable state space can be terminated in limited steps due to admissibility and optimality of A^* algorithm, moreover, the optimal solution can be found. Because S_g is the goal state node in the optimal path, hence $h(S_g)=0$.

2. Let the cost of edge in search tree denote performance increment between relevant states: $c(x_i,x_j)=F_j-F_i$, where F_i, F_j is the system performance of the *i*th, *j*th state, respectively.

1) Let $p(r_k(x_{i-1}))=n(n>0)$, $p(r_s(x_{j-1}))=m(m>0)$, $i<j,(i,j,k,s=1...n)$;

Since rule with the highest priority is started in each loop, incorrect time expressions to

be revised in the last iteration basically don't exist in the next iteration, and the number of new errors is less than *n* or else the performance of system will not be improved. Thus, we have $n \geq m > 0$, $\frac{1}{n} \leq \frac{1}{m}$. Hence $h(x_i) = \frac{1}{n} \leq \frac{1}{m} = h(x_j)$, say,

$$h(x_i) - h(x_j) \leq 0 \tag{1}$$

If we substitute the right part of inequation $h(x_i) - h(x_j) \leq 0$ for $c(x_i, x_j)$, then different definitions of edge cost will bring different convergence speed.

2) Referring to the evaluation metric of Section 5.1, we first set the following parameters:

a_i : the number of correctly recognized time expressions in *i*th state;

b : the total number of standard time expressions;

c_i : the total number of recognized time expressions in *i*th state;

d_i : increment of the total recognized results in *i*th state due to adding the rule;

After adding a rule, the system performance of *i*th state can be represented as the following:

$$F_i = \frac{2*P*R}{P+R} = \frac{2*\frac{a_i+n}{b} * \frac{a_i+n}{c_i+d_i}}{\frac{a_i+n}{b} + \frac{a_i+n}{c_i+d_i}} = 2*\frac{a_i+n}{b+c_i+d_i} \tag{2}$$

Here,

$$c(x_i, x_j) = F_j - F_i = 2*\left(\frac{a_j+m}{b+c_j+d_j} - \frac{a_i+n}{b+c_i+d_i}\right) \tag{3}$$

In ideal state, $c_i = c_j$, $d_i = d_j$. Incorrectly recognized time expressions are revised owing to adding the rule. Here, priority n, m denotes the number of revised time expressions, respectively. Note that $a_j = a_i + n$, hence:

$$c(x_i, x_j) = 2*\frac{m}{b+c_i+d_i} > 0 \tag{4}$$

Comparing formula (1) with (4), it is clear that $h(x_i) - h(x_j) \leq c(x_i, x_j)$.

Know from condition 1, 2, we prove that heuristic function $h(x_n)$ satisfies the monotonicity of A^* algorithm.

Similarly, let l denote the length of the optimal rule path from S_0 to S_g, thus time complexity is at least O(l). Here, we can design different heuristic functions according to the different understanding on the problem containing heuristic information. Known from admissibility, optimality and monotonicity of A^* algorithm, the optimal solution can be found. Considering the search cost, we restrict the scope of retrospect nodes, thus what we find is the approximation of the optimal solution. Heuristic error-driven learning framework not only can be used in Chinese time expression recognition, but also can be used in other natural language processing application. It has the following merits:

1) Distinguish the validity of each rule;
2) Find the optimal rule path with the least search cost, which brings the most improvement of performance.
3) Guarantee the compatibility among rules, if there is conflict among rules, the rule leading to the decrease of performance will be excluded.

5 Experiments

5.1 Evaluation corpus and metrics

Chinese corpus of ACE2005 is used for evaluation in the experiments. Because the official does not distribute the testing corpus, we classify the distributed training corpus into training and testing part. Though our result cannot compare with the official evaluation results of ACE, it still has some reference. Table 2 gives the basic settings of corpus.

The bracketing performance as defined in the TERN is used to evaluate the systems, which means that there is an exact match between the system output and the reference expressions. Scores will be reported in terms of precision and recall numbers along with the $F_{\beta=1}$ metric.

	#article	#time expression	size
Training set	570	4465	104（MB）
Testing set	63	521	11.4（MB）

Table 2. Settings of datasets

5.2 Experiments setting and results analysis

Our experiment systems were run on PCs with AMD Athlon(tm) 64 X2, Dual Core Processor 3600+ at 2.00GHz, and memory 1GB. Four groups of comparison experiments were designed for evaluating the performance and efficiency of Chinese time expression recognition. We validated that whether the dependency parsing works in Chinese time expression recognition, and how the A^* search strategy influences the filtering of rules in heuristic error-driven learning. We also compared with the SVM based recognition method.

Experiment 1: dependency parsing based Chinese time expression recognition

Some words with POS "nt" can also be considered as a kind of time expression. However, the types of time expressions defined in ACE are more comprehensive. Therefore, we first extract the non time stopping word with POS "nt" as the baseline experiment, and further compare the performance of dependency parsing based recognition methods under different time trigger word construction schemes. Table 3 presents the bracketing performance, where T_1 stands for non time stopping words with POS "nt", T_2 represents the time trigger word dictionary constructed by semi-automatic method, and T_1T_2 is the combination of T_1 and T_2.

The results in Table 3 show that dependency parsing based recognition method is effective. Though the Chinese dependency parsing is not completely correct, yet it still brings great improvement for system performance. Apparently, the performance of close test is weakly better than that of open test. Fifty percent of time expressions were recognized by using T_1. If T_1 are considered as time trigger words, the performance will increase 14% by using dependency parsing based recognition method. If substituting T_1 for T_2, and adopting the same recognition method, the performance hadn't been improved much. It can be seen that time trigger dictionary is not perfect. If we combine T_1 with T_2, and adopt the dependency parsing based recognition method, the performance will be greatly improved. This shows that T_1 has a complementary function on the time trigger word dictionary T_2. The merit of dependency parsing based recognition method is that it can resolve the problem of long distance dependency to some extent, but it also can bring some noise. If identifying time expressions with one word or less words, noise will be more serious. In order to overcome the limitation of this method, we will further adopt error-driven learning to automatically acquire rules, and revise the wrong results derived from dependency parsing based recognition.

Train model	Train (Close test)			Open test		
	P	R	F	P	R	F
T_1(baseline)	0.5158	0.4905	0.5028	0.5721	0.5029	0.5352
T_1+dependency parsing	0.7592	0.5825	0.6592	0.7538	0.5643	0.6454
T_2+dependency parsing	0.6680	0.5174	0.5831	0.6650	0.5029	0.5727
T_1T_2+dependency parsing	0.6957	0.7675	0.7298	0.6977	0.7486	0.7222

Table 3. Comparison of bracketing performance of dependency parsing based recognition methods under different time trigger word construction schemes

Experiment 2, 3, 4: filtering rule strategy in error-driven learning

Based on experiment 1, the goal of experiment 2, 3, 4 is to evaluate the performance of heuristic error-driven learning that we proposed. Furthermore, we validated that how A^* search strategy of filtering rule influences the capability of recognition, in comparison with batch filtering rule proposed by (He 2007) and greedy filtering rule put forwarded by Bill (Brill 1995). The experiment results are shown in Table 4, where A^* DE stands for A^* search algorithm based strategy of filtering rule in heuristic error-driven learning. PDE denotes error-driven learning based on batch filtering rule strategy, and GDE represents error-driven learning based on strategy of greedy filtering rule. In addition, the time consume of open test is very little, which is ignored in table 4.

Train model	Train(Closed test)			approximate time of training	Open test		
	P	R	F		P	R	F
A^* DE	0.7433	0.8197	0.7796	27 minutes	0.7540	0.8061	0.7792
PDE	0.7311	0.8110	0.7690	54 seconds	0.7168	0.7678	0.7414
GDE	0.7641	0.8417	0.8010	15 hours	0.7545	0.8081	0.7804

Table 4. Comparison of running efficiency and performance of A*DE , PDE and GDE

Known from the results in table 4, PDE algorithm has little time consume, and effectively constrains the noise brought by dependency parsing based recognition method. Its performance is higher 4%, 2% than the best performance of experiment 1 on close test and open test, respectively. However, the validity of each rule cannot be distinguished. GDE algorithm can distinguish the validity of each rule, and achieve the highest performance. Nevertheless, the time complexity for GDE is at least $O(n^l)$, and the training

took almost 15 hours. In addition, n is the number of rules learned by each iteration, and *l* is the length of rule path from S_0 to S_g. In comparison with GDE, the time complexity for A^* DE is O(*l*), thus the training took little time. Its performance in closed test is little weaker, but that in open test is almost as much as that of GDE. Therefore, as far as performance and running efficiency, A^* DE algorithm is the best. It not only can distinguish the validity of each rule, but also can find the optimal rule path with the least search cost, which will bring the most improvement of performance. It also can guarantee the compatibility among rules. If there is conflict among rules, the rule leading to the decrease of performance will be excluded.

Related works in Chinese time expression recognition are very few, for reference we compare our recognition result of A^* DE with (Hacioglu et al. 2005). He adopted the Chinese corpus of TERN2004, and considered the time expression recognition as the sequence labeling question. SVM classifier was used in recognition task based on the features including word, POS, the tag of time expression recognized by handcraft rule and so on. Because man-made rule was used in the process of recognition, we cannot reconstruct his experiment. Here, we just have a rough comparison. Its score of precision, recall and $F_{\beta=1}$ is 83.8%, 74.0%, 78.6%, respectively. Its $F_{\beta=1}$ score is approximately equal to that of A^* DE algorithm. It has high precision, but has low recall. Our method is just contrary, which has high recall, but has low precision. This shows that performance of A^* DE still has much space for improvement.

We also can see from table 4, the performance of closed test is higher than that of open test. It is because that time trigger word dictionary is not perfect, and that 102 rules were totally learned by A^* DE, whose generalization capability is not strong. It can be noted that cascaded error of bottom module and inconsistency of corpus annotation can bring the incorrectly recognized results. Therefore, improvement of dependency parsing will enhance the recognition performance.

6 Conclusion and future work

Filtering rule in error-driven learning essentially belongs to the search question in artificial intelligence. This paper integrated the heuristic search strategy A^* algorithm into error-driven learning, and proposed a new approach to Chinese time expression recognition based on heuristic error-driven learning. We designed the heuristic function and theoretically proved its monotonicity, so that guaranteed the correctness of A^* algorithm.

The concept of time stopping word is proposed in this paper and we also gave the dictionary construction schemes of time trigger word, time stopping word, prefix and postfix. In the phase of initial recognition, dependency parsing based method is adopted to resolve the problem of long distance dependency; during the process of error-driven learning, this kind of machine learning effectively revises the noise brought by initial recognition. At the same time, A^* algorithm based heuristic search strategy availably avoids the disadvantages of greedy filtering strategy, such as huge search space and low running efficiency. It also overcomes the limitation of not distinguishing the validity of each rule in batch filtering strategy, which makes search go ahead along the fastest speed. The experiment showed that this method is feasible. The heuristic error-driven learning proposed in this paper is a new machine learning framework, which can be extended to other applications of natural language processing.

In the future, we will try to improve the performance of Chinese time expression recognition from the following aspects:

1) Strengthen the construction of time trigger word dictionary and improve the generalization capability of rules so as to adapt the application need of open domain.

2) During the course of heuristic error-driven learning, we haven't learned what kind of phrase is not time expression, say, we just learned the positive features of time expressions, but not the negative features. In the future, we will extend the learning scope of error-driven learning by extracting the negative features so as to filter the noise.

3) Identify the time expressions with ambiguities. Some strings of characters are time expressions in given contexts, but in other contexts they are not. There are many kinds of ambiguities. Time expressions with disambiguation also lead to the difficulty of recognition to some extent, for example:

Example1: 随后湖南队 14 号利用快攻连得 4 分，

Example2: 我看了看表，时间是 15 ： 20。

Some ambiguities are caused by the combination of numbers and time units, such as example 1. In this case, the expression "14 号" just refers to a football team member. However, in many news messages it is a date. Other ambiguities are caused by numbers, such as time expressions "15 ： 20" in example 2. However, it may be a score of a game in sports news messages.

7 Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant

No. 60435020, 60675034, and the National 863 High-tech Project No.2008AA01Z144. We also would like to thank the reviewers for their valuable comments and Lang Jun for his suggestions during the research.

8 Reference

Ahn, D., Adafre, S. F., and Rijke, M. de., 2005, Towards Task-Based Temporal Extraction and Recognition. *Proceedings Dagstuhl Workshop on Annotating, Extracting, and Reasoning about Time and Events.*

Brill, E., 1995, Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543--565.

Chinchor, N. ,1998, MUC-7 Information Extraction Task Definition. *In Proceedings of the 7th Message Understanding Conference.*

Estela, S., Martinez-Barco, Patricio, and Munoz, R., 2002, Recognizing and Tagging Temporal Expressions in Spanish. *Workshop on Annotation Standards for Temporal Information in Natural Language in the International Conference on Language Resources and Evaluation*.

Hacioglu, K., Chen, Y. and Douglas, B., 2005, Automatic Time Expression Labeling for English and Chinese Text, *to appear in Proceedings of Conference on Intelligent Text Processing and Computational Linguistics*, pp. 13-19, Mexico City-Mexico.

Hart, P.E., Nilsson, N. J., & Raphael, B. , 1968, A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(2), 100-107.

He, R.F., Qin, B., Liu, T., Pan, Y.Q., Li, S., 2007, Recognizing the Extent of Chinese time expressions based on the Dependency parsing and Error-Driven learning. *Journal of Chinese Information Processing*, 21(5): 36-40.

Jang, S.B., Baldwin, J. and Mani, I., 2004, Automatic TIMEX2 Tagging of Korean News. *ACM Transactions on Asian Language Information processing*, Vol. 3, No. 1, 51-65.

Luger, G. F..(Shi ZZ et al. Translate), 2004, Artificial Intelligence-Structures and Strategies for Complex Problem Solving. Fourth edition, China Machine Press, CITIC Publishing House (in Chinese).

Mani, I. , 2004, Recent Developments in Temporal Information Extraction[C]. *In Proceedings of the Conference on Recent Advances In Natural Language Processing.*

Sundheim, B., & Chinchor, N. , 1995, Named entity task definition, version 2.0. *In Proceedings of the 6th message understanding conference (MUC-6).*

Vazov, N., 2001, A System for Extraction of Temporal Expressions from French Texts based on Syntactic and Semantic Constraints. *Proceedings of the ACL Workshop on Temporal and Spatial Information Processing*, pp. 96-103.

Wang, Y., 1998, Principle and Method of Artificial Intelligence, Xi'an Jiaotong University Press.

Wong, K.-F, X, Y., Li, W., Y, C., 2005, An Overview of Temporal Information Extraction. *Int. J. Comput. Proc. Oriental Lang.* 18(2): 137-152

Wu, M., Li, W., Lu, Q., Li, B., 2005, CTEMP: A Chinese Temporal Parser for Extracting and Normalizing Temporal Information. *International Joint Conference on Natural Language Processing*, pp. 694-706, Jeju Island, Korea

Yang, Y., Fossum, V., and Abney, S., 2006, Latent features in automatic tense translation between chinese and english. *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pp 48-55, Sydney, Australia.