# Abstracting Knowledge from Annotated Chinese-Chess Game Records

Bo-Nian Chen[1,⋆], Pangfang Liu[1], Shun-Chin Hsu[2], and Tsan-sheng Hsu[3,⋆⋆]

[1] Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
{r92025,pangfeng}@csie.ntu.edu.tw
[2] Department of Information Management,
Chang Jung Christian University, Tainan, Taiwan
schsu@mail.cju.edu.tw
[3] Institute of Information Science, Academia Sinica, Taipei, Taiwan
tshsu@iis.sinica.edu.tw

**Abstract.** Expert knowledge is crucial for improving the strength of computer Chinese-Chess programs. Although a great deal of expert knowledge is available in text format that using natural languages, manually transforming it into computer readable forms is time consuming and difficult. Written expert annotations of Chinese-Chess games show different styles. By analyzing and collecting commonly used phrases and patterns from experts' annotations, we introduce a novel pattern matching strategy. It automatically epitomises knowledge from a large number of annotated game records. The results of the experiments on the analysis of the middle phase of games indicate that our strategy achieves a low error rate. We hope to exploit this approach to collect automatically a great diversity of Chinese-Chess knowledge that is currently in text format.[1]

## 1 Introduction

Computer Chinese-Chess has developed well over the last 20 years. There are now several computer Chinese-Chess programs that show a human master level of playing expertise [19]. A popular strategy for program designers is to use variations of the $\alpha$-$\beta$ pruning search algorithm with rule-based evaluation functions. The algorithm is the core component that identifies the best move, while the evaluation function provides a standard measurement of the given position. In this game-playing model, knowledge of Chinese Chess is embedded in the evaluation functions.

⋆ Supported in part by National Science Council (Taiwan) Grants 94-2213-E-001-014.
⋆⋆ Corresponding author.
[1] Early concepts of this paper appeared in *"Automatic Expert Knowledge Retrieval Strategy"*, the 10th Conference on Artificial Intelligence and Applications, Taiwan, 2005, p. 11 (one page abstract only).

For the program to be strong enough to play against human masters, extensive expert knowledge must be embedded in the evaluation function [17]. A simple strategy is to add the expert knowledge as rules to the evaluation function manually; however, this strategy has a number of drawbacks. First, there is a bound on the number of rules that can be accommodated by the evaluation function. The more rules there are, the greater the likelihood of conflict between the rules. Second, for the sake of efficiency, we cannot afford to use high-level knowledge in the evaluation function. Third, adding rules manually is time consuming and subject to human error.

In this paper, we propose a strategy that transforms expert knowledge into information in a form that can be easily exploited by computer programs. In Section 2, we introduce our approach. In Section 3, we describe our strategy for retrieving expert knowledge automatically. In Section 4, we present the experimental results. Finally, in Section 5, we present our conclusions.

## 2 Preliminaries

In this section, we first define expert knowledge in computer Chinese Chess [19]. Then, we introduce traditional knowledge retrieval schemes and compare them with the automatic expert knowledge retrieval scheme proposed in this paper.

### 2.1 Expert Knowledge in Computer Chinese Chess

Chinese Chess, like Western Chess, is a two-player game in which each player alternately moves his[2] respective pieces toward a destination. Many theories about playing Chinese Chess have been developed and a large amount of knowledge has been accumulated. Some of this knowledge has been recorded in books or on web sites, but expert knowledge available to researchers is mostly in text format.

There are three phases in a Chinese Chess game. The first phase is the *opening phase*. It consists of the first 20 or so moves in the game. Opening strategies can be divided into two types: popular opening strategies [3] and rarely used opening strategies [2]. Many of the opening strategies developed by human masters have been published in opening game books [4]. There are also books that discuss the theories of opening games [8]. In the second phase of a game, called the *middle phase*, each player decides the tactics he will use to strengthen his position and thereby threaten the other player. Evaluating the advantage or disadvantage of a middle-phase position is difficult, even for a master [1]. There are also many books that teach the middle game knowledge by examples [7]. Third, we have the *end phase* when the players have very few pieces left. In this phase, heuristics and rules are used to play the game adequately as possible using the remaining pieces. There are many books about endgame heuristics and rules [16]. There are many hard endgame problems that cannot be solved by current computer programs.

[2] For brevity and readability we use ‘he’ and ‘his’ wherever we mean ‘he or she’ and ‘his or her’.

The positions of the opening games and the endgames have been stored in opening game databases [14] and endgame databases [6,10], respectively. However, expert knowledge about the middle game is not clearly defined. Instead, there are only annotated game records, which are just examples of human knowledge about the middle game. Thus, in this paper, we focus on expert knowledge about the middle game [13].

### 2.2 The Importance of Automatic Expert Knowledge Retrieval

Existing expert knowledge retrieval schemes have two potential drawbacks: (1) designers may need to spend a large amount of time implementing the rules one by one; and (2) the position sets of different expert knowledge schemes may be different. Consequently, if two schemes are applied simultaneously, their knowledge may overlap or conflict. The advantages of automatic expert knowledge retrieval are; (1) reducing the implementation time and (2) partitioning the position set into several classes so that each class can be marked as a rule.

## 3 Automatic Expert Knowledge Retrieval

In this section, we define our strategy for automatically retrieving expert knowledge from a large number of annotated game records in text format.

### 3.1 Expert Knowledge and Position Value

We focus on expert knowledge in the middle phase of games. We use a program that automatically transforms a text-formatted annotated move in a game record into a *position value*, which is the score of that position. Position value assignment has been used in many applications, such as [15,11]. The position value is an integer in the range of 0 to 9, where each number represents a class used to measure the advantage or disadvantage of a given position to the red side, as shown in Table 1.

Thus, a position that is advantageous to the red or black player is classified into the red-advantage or red-disadvantage class respectively.

### 3.2 Novel Pattern Matching Algorithm

Text data used to annotate game records is written in a special form of natural language. Instead of using classical natural language processing techniques, with emphasis on efficiency and accuracy, we use a novel pattern matching algorithm (see Figure 1) to parse the annotations of game records. The process uses a processing algorithm and a keyword set. The latter is separated into several classes, each of which is mapped to a position value. When the sentences of annotations are recognized by one of the grammar rules or several consistent rules, the score-related element of the current state is mapped to a class of position values.

The algorithm recognizes the following classes of keywords: *subject*, *score-related*, *passive*, *passive-exception*, *condition*, *condition-exception*, *negation*, and

**Table 1.** The position values with respect to the red side and their meanings

| Value | Class | Description |
|---|---|---|
| 0 | excellent | Red is sure to win. |
| 1 | very good | Red has very little chance of losing, and a better chance of winning than drawing. |
| 2 | good | Red has more chances of winning than losing or drawing. |
| 3 | advantage | The position of Red is a little better than that of the Black |
| 4 | even | Both sides have an equal chance of winning. |
| 5 | draw | Neither player can win. |
| 6 | disadvantage | The red position is not as good as that of Black. |
| 7 | bad | Red has more chances of losing than winning or drawing. |
| 8 | very bad | Red has very little chance of winning, and many more chances of losing than drawing. |
| 9 | worse | Red is sure to lose. |

*negation-exception*. In the following, we present a brief definition of each class of keywords. We note that the sentences are originally written in Chinese. In some of the examples, we used a word-by-word translation, which may not be grammatically correct in English.

**Subject element:** a phrase that indicates which player is the main subject of the discussion. For example:
"*For the current position, the black side has material advantage.*" (局面到此黑方多子)
"*The black side*" is a subject element.

**Score-related element:** a phrase that describes the property of a given position, which can be mapped to the position's value. For example:
"*For the current position, the black side has an advantage, and the red side has disadvantage.*" (局面到此，黑方優勢，紅方劣勢)
"*Advantage*" is a score-related element.

**Passive element:** a phrase that indicates a sentence is in the passive form, i.e., the subject and object are exchanged. If there is a passive element, the subject should be found after the score-related element. For example:
"*The rook of the red player is threatened.*" (紅方的六路車就受到威脅啦)
"*Threatened*" is a passive element. The difference between English and Chinese sentences is that the words indicating a passive meaning in English are always verbs, but in Chinese there are special words that indicate a passive meaning.

**Condition element:** a phrase that indicates the subsequent sentence is true, subject to some conditions, i.e., it is not a true statement now, but it will be true after the condition occurs. For example:
"*If you move the advisor, the rook on the right will not have a good place to move to and the situation would be more severe.*" (如果改走補仕則右車難以出頭，反而形勢更加不利)
"*If*" is a condition element; thus, we may not conclude that the red side will win. For simplicity, we ignore sentences with condition elements.

**Negation element:** a phrase that invalidates a score-related element. For example:
"*The red side has no advantage.*" (紅方並沒有佔到便宜)
"*No*" is a negation element, which means the score-related element "*advantage*" has no effect in this sentence.

**Exception element:** passive-exception elements, condition-exception elements, and negation-exception elements are all called exception elements. They are used to filter phrases that look similar to an element, but they have different meanings. These classes of phrases usually only occur in Chinese.
For example:
"*The position of the black side is not bad.*" (這樣黑方局面不錯)
Here "*not bad*", which contains a negative word even if it is translated into Chinese, does not really have a negative meaning; thus, it is a negative-exception element.

## 3.3 Chinese-Chess Annotation Abstraction Algorithm

This processing algorithm (Fig. 1) is a novel pattern matching strategy, designed especially for Chinese-Chess annotation abstraction. When the algorithm is applied to a sentence, it finds the elements and uses the grammar rules below to extract the meaning of the input sentence. We note that these rules are designed specifically for the annotations written in Chinese.

1. $Stmt \rightarrow Subject \quad Score \quad [Object]$
   This rule means that the player defined in the subject element receives the score defined in the score-related element. In Chinese Chess, there may be an *object* after the score-related element, which must be the opponent of the subject element. For example: "*In the current position, black has advantage, red has a disadvantage.*" (局面到此，黑方優勢，紅方劣勢。)
2. $Stmt \rightarrow Subject \quad Negation \quad Score \quad [Object]$
   This rule has a negation element, so the player defined in the subject element does not receive any points. For example: "*Black uses cannon to threaten red's knight, thus red loses the advantage.*" (但被黑方進包打傌，紅方先走之利的優勢沒有了。)
3. $Stmt \rightarrow Object \quad Passive \quad Score \quad [Subject]$
   This rule has a passive element, so the object element occurs first. Thus, the opponent of the player defined in the object element gets the score defined in the score-related element. For example: "*Black moves cannon, red's advantage is lost. Because black threatens red's knight, it also threatens red's cannon.*" (被黑方進過河包，紅方還是被反先了，因爲黑方有平包射傌的棋，或者出貼身車抓士角炮的棋。)
4. $Stmt \rightarrow Cond \quad Stmt$
   This rule has a condition element so that the subsequent sentence is ignored. For example: "*If the red side moves his rook to exchange the black's rook, the formation of the black side will be flexible, because the red pawn in the 1st column is threatened by the black cannon in the 9th column.* " (如果紅方進車兑車反而黑方陣容具有彈性，因爲紅方的一路兵要被黑方的九路包射。)

5. *Stmt* → *Score*
   This rule does not have a subject element. There are two cases for this rule, depending on the score-related element.
   (a) The score-related element indicates a draw position value. We therefore assign a draw score. For example: "*After exchanging rooks, a draw w̄ould be very likely.*" (兑車之後和棋的成份很大。)
   (b) In Chinese, a score-related element is a combination of a subject element and a score-related element. We therefore assign the score to the subject. For example: "*Red wins.*" (紅勝。)

The algorithm returns a position value when the sentences match one of the grammar rules, or several consistent rules. Conversely, it reports an error when a sentence does not match any grammar rule, or it matches several rules with contradictory results. Note, if an annotation consists of more than one sentence, we parse each sentence individually. A conflict between two sentences means that their scores are not consistent. In this situation, the algorithm chooses the last score of the sentences, since the conclusion is more likely to be in the last few sentences.

```
procedure CCAA(text, Score, Error)      // output score and error
     player = 0, subject = 0, score = 0, inconsistent_error = false
     active = 1, positive = 1, condition = not found
     keyword_found = not found
     for i < Len(text) do
          if(a new sentence is starting)
               subject = 0, active = 1, positive = 1
               condition = not found
          if(condition == not found)
               key_score = FindScoreKeyword (i, text)
               if(keyword is found)
                    if(keyword_found == found)
                         if( ! IsSlightError(key_score, score) )
                              inconsistent_error = true
                    if(keyword doesn't need subject)
                         score = active * positive * key_score
                    else
                         score = player * active * positive * key_score
               keyword_found = found
          if(FindPassiveKeyword(i, text) = found)
               active = -1
          if(subject not found or active == -1)
               subject = FindPlayerKeyword(i, text)
          if(FindConditionException(i, text) == not found)
               condition = FindConditionKeyword(i, text)
          if(FindNegativeException(i, text) == not found)
               positive = FindNegativeKeyword(i, text)
     if(keyword_found == not found)
          Score = 0, Error = not found
     else
          Score = score, Error = inconsistent_error
end procedure
```

**Fig. 1.** The Chinese-Chess Annotation Abstraction Algorithm

### 3.4 Automatic Procedure

The procedure for automatically retrieving knowledge from a large number of annotated game records uses an auto-feeder to read an annotated list of game records and then feeds them into the Chinese-Chess annotation abstraction algorithm one at a time. The keywords provided by the algorithm are stored in a file, called the *keyword pattern base*. Each output from the language processor is the position value for the corresponding record. The value is classified by the distributor, which automatically puts the position into the corresponding class. The collected data is then saved in a Position Evaluation Database. Note that the data in the database consists of scored positions. An advantage of the database is that it can be used to train a reliable evaluation function for search algorithms, or to measure the performance of an evaluation function.

## 4 Experimental Results

### 4.1 Experimental Design

The work in [17] describes many popular opening games and discusses games that even may be extended to the endgame. The author, G. L. Wu., is considered the best player in Taiwan. He uses several examples to formulate as precisely as possible the knowledge about the opening, the middle game, and the end game. An important issue is how to determine which position to select. To achieve this, Grandmaster Wu has listed explicitly the advantages and the disadvantages of a critical position in text format. With these lists, we first conducted an experiment on expert knowledge data taken from all of the annotated game records in all volumes of [17]. Since the leaf nodes of Wu's game tree are the most critical positions, we collected the annotations on the leaf nodes for our first experiment. There are 26,831 nodes and 2,043 leaf nodes on the game tree in [17]; each node represents a single position. We deleted 178 positions, which were not relevant to the measurement of the position.

The remaining 1,865 positions cover the middle phase, and in some cases the opening phase and the end phase. We first identify the phase of a game that is assigned to a position. There are many different definitions for the phases of a game. In [8], the opening game consists of about 10 plies; of course, there may be more or fewer plies, depending on the state of the position. In [9], the middle game starts before the actual battle proceeds. None of the phases is clearly defined. Hence, for ease of implementation, we adopted the following definitions.

**Opening game.** None of the rooks, cannons, and horses, (which are the *strong pieces*), are captured, but some pawns may be taken.

**Middle game.** For clarity, the middle game is divided into two cases: (1) at least one player has more than four strong pieces, and at least one strong piece of either player is captured; and (2) at least one player has exactly four strong pieces, one of which is a rook. Note that the rook is considered as two strong pieces, since its value is equal to two cannons or two horses, or one of each.

**Endgame.** There are two cases in the definition of the endgame: (1) both players have less than or exactly three strong pieces, and (2) neither player has a rook.

Based on the above definitions, the 1,865 positions consist of 424 opening game positions, 1,377 middle game positions, and 64 endgame positions. All the positions were annotated and the sentences were verified manually to ensure they were free from grammatical errors and scoring conflicts. This set of test data, called WU1865, is used to fine tune our algorithm and find various elements. In the current algorithm, there are 306 score-related elements, 6 subject elements, 5 passive elements, 13 condition elements, 7 condition-exception elements, 5 negation elements, and 21 negation-exception elements.

The position values of the 1,865 annotated positions were manually assigned in one week by the first author, who is a certified Chinese-Chess 2-Dan[3] player.

By comparing the answers annotated manually and the results from our algorithm, we were able to analyze the effectiveness of our approach statistically.

The statistical analysis of WU1865 is shown in Table 2. The numbers in the first row are position values of the manually assigned values, while the numbers in the first column are position values of the program results. Each grid in the *i-th* row and the *j-th* column represents the number of cases for which the answer is position value $j$, but our algorithm gives position value $i$, denoted by $val(i, j)$. The diagonal grids are correct cases, denoted by $val(i, i)$. If the numbers on the diagonal grids are much larger than on the vertical grids and horizontal grids, it means the algorithm is very reliable. The numbers in the last row and the last column are the summations of the rows or columns , denoted by $row\_sum(j)$ for summation of the *i-th* row and $col\_sum(i)$ for summation of the *j-th* column, respectively. We use $val(i, j)/row\_sum(i)$ and $val(i, j)/col\_sum(j)$ to measure the error rate; the lower the error rate, the better the performance of the algorithm.

As the experimental results show, the error rate is less than 1/10 for all rows and columns. Our algorithm can identify the following three errors.

1. No keyword error, named as *E1* error. There are two possible cases of this error: (1) the input sentences are not relevant to the measurement of a position, and (2) the keyword cannot be found in the keyword pattern base.
   For example: "*Moving the red rook ahead is considered a traditional and progressive move.*" (紅方進橫車是傳統的穩健走法)
   There are no keywords in this example, i.e., it does not contain any words relevant to the measurement of the position.
   For case 1, the input sentence should be discarded and is not discussed here.
   For case 2, we can increase our keyword pattern base to solve the problem.
   In Table 2, positions that cause no keyword error are classified as class E1.

[3] Dan is a measurement of the playing expertise of a Chinese-Chess player in Taiwan. The range of Dan is from 1-Dan to 9-Dan. 1-Dan to 3-Dan are roughly considered experts, 4-Dan to 6-Dan are roughly masters, and 7-Dan to 9-Dan are roughly grandmasters.

2. Inconsistent element error, named as *E2* error. This error occurs when two or more scores can apply to a position. As noted earlier, the algorithm chooses the last score as the score of the position. In Table 2, positions that cause inconsistent element errors are classified as class E2.
   For example: "*The black side was careless to check the red king and let a piece be captured. He should have moved k6+1, which would have given him an advantage.*" (黑方隨手下將造成白丟子，應走將6進1，黑方前景較爲看 好)
   In this example, there are two elements: let a piece be captured and advantage, which have different meanings for the position.
3. Scoring error: This occurs when the score generated by the algorithm is different from that of manual annotation and, of course, the scoring error is free from the first two errors. This type of error arises when the text uses complicated grammar that our algorithm cannot recognize. It is also difficult to formulate the underlying grammatical structure with our approach. This error is relevant to the correctness of the algorithm.
   For example: "*The black side moves his rook in front of the river, using the rightmost line containment to prevent the red side moving H4+6.*" (黑方進車巡河，利用邊線牽制，防止紅方傌四進六快攻)
   In this example, the keywords "containment" and "prevent", which refer to the black side are considered advantageous keywords in our algorithm, but the actual annotation of the position should be even.

Furthermore, it is noteworthy that $val(i, i+1)$, $val(i, i-1)$, $val(i+1, i)$, or $val(i-1, i)$ and the advantages or disadvantages are is not changed. Also in the *drawing state*, which comprises even and draw, changes between the two are called *slight errors*. This kind of error often occurs because of subjective judgments of the annotators. Even so, such errors are tolerable in computer Chinese-Chess applications.

We define the number of *significant errors* as the measurement of the total errors due to the limitations of the algorithm: $number\_of\_significant\_errors = scoring\_errors - slight\_errors$ According to the statistical results, there are 11 no-keyword errors, 110 keyword-inconsistent errors, and 129 scoring errors. There are also 75 slight errors in the training set. The number of significant errors is 54.

It took less than 1 minute to parse the annotation of 1,865 positions and generate a statistical analysis on an Intel Pentium IV 1.8GHz CPU with a 512MB RAM. The code size of the system is approximately 2,000 lines (not including the GUI code). The current size of the keyword pattern base is 3,158 bytes, which can be increased if necessary.

## 4.2 Some Detailed Results

Next, we used our program to analyze an untrained data set called NET_TEST. There are two sources of NET_TEST. The first is [18], which consists of 225 game records on a CD. On average, there are 5 annotated positions in each game record. The second source consists of web sites. There are 249 positions in the web annotated by some of the best grandmasters in China, such as Y. C. Xu.

**Table 2.** Comparison of manually annotated answers and algorithm-generated results for WU1865. The horizontal axis represents the number of positions manually annotated. The vertical axis represents the number of positions generated by the algorithm, where E1 represents no element error and E2 represents inconsistent element error.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Sum |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 85 | 10 | 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 105 |
| 1 | 4 | 98 | 10 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 113 |
| 2 | 9 | 10 | 548 | 5 | 0 | 1 | 2 | 1 | 1 | 0 | 577 |
| 3 | 0 | 0 | 10 | 82 | 1 | 1 | 4 | 0 | 0 | 0 | 98 |
| 4 | 0 | 0 | 0 | 2 | 97 | 0 | 2 | 0 | 0 | 0 | 101 |
| 5 | 1 | 0 | 0 | 2 | 1 | 29 | 3 | 2 | 0 | 1 | 39 |
| 6 | 0 | 0 | 0 | 0 | 5 | 1 | 186 | 9 | 0 | 0 | 201 |
| 7 | 0 | 0 | 0 | 1 | 1 | 0 | 9 | 393 | 4 | 2 | 410 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 61 | 0 | 63 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 36 | 37 |
| E1 | 0 | 0 | 2 | 2 | 1 | 2 | 1 | 2 | 1 | 0 | 11 |
| E2 | 1 | 0 | 19 | 16 | 23 | 7 | 28 | 13 | 3 | 0 | 110 |
| Sum | 100 | 118 | 598 | 110 | 129 | 41 | 236 | 423 | 71 | 39 | 1865 |

There are totally 1,418 annotated positions, 55 of which are not relevant to the measurement of the position. As a result, NET_TEST contains 1,363 annotated positions, comprised of 613 opening game positions, 627 middle game positions, and 123 endgame positions.

Before discussing the results of the experiment, we describe the properties of the test data. We discovered that there are three types of annotation in [17]

1. Grandmaster Wu made a conclusion to the specified position.
2. Grandmaster Wu discussed the advantages or disadvantages of the player's choice in a specific position.
3. In addition to 2, Grandmaster Wu discussed the advantages or disadvantages of several possible positions.

Our algorithm works well for points 1 and 2 because it is deterministic. However, in point 3, it is hard to determine which strategy is suitable for the critical position.

In our experiment, there are 351 no-keyword errors, 72 keyword-inconsistent errors, 353 scoring errors, and 100 slight errors.

In Table 3, most of the values are near the diagonal grids, and the number of significant errors is 253. Most errors occur in the column of class 4. The error rate in most rows and columns is less than or about 1/10, except for val(2, 4) and val(5, 4). The performance of our algorithm is good and stable for most cases. It is also convenient to classify two error cases, E1 and E2, separately because they are the first candidates to be modified manually.

**Table 3.** Comparison of manually annotated answers and algorithm-generated results for NET_TEST. The horizontal axis represents the number of positions manually annotated. The vertical axis represents the number of positions generated by the algorithm, where E1 represents no element error and E2 represents inconsistent element error.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Sum |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 7 | 0 | 0 | 2 | 0 | 1 | 1 | 0 | 0 | 0 | 11 |
| 1 | 0 | 28 | 3 | 2 | 3 | 1 | 1 | 0 | 1 | 0 | 39 |
| 2 | 0 | 0 | 190 | 13 | 74 | 4 | 13 | 1 | 1 | 1 | 297 |
| 3 | 0 | 1 | 1 | 64 | 7 | 1 | 3 | 2 | 0 | 1 | 80 |
| 4 | 0 | 0 | 0 | 1 | 55 | 3 | 3 | 2 | 0 | 0 | 64 |
| 5 | 1 | 1 | 2 | 11 | 70 | 69 | 12 | 3 | 2 | 1 | 172 |
| 6 | 0 | 1 | 0 | 2 | 16 | 3 | 52 | 0 | 0 | 0 | 74 |
| 7 | 0 | 1 | 7 | 14 | 37 | 2 | 10 | 99 | 0 | 1 | 171 |
| 8 | 0 | 0 | 0 | 0 | 4 | 2 | 0 | 0 | 20 | 0 | 26 |
| 9 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 3 | 6 |
| E1 | 1 | 10 | 31 | 42 | 184 | 27 | 23 | 25 | 7 | 1 | 351 |
| E2 | 0 | 2 | 9 | 15 | 29 | 4 | 10 | 2 | 1 | 0 | 72 |
| Sum | 9 | 44 | 244 | 160 | 480 | 118 | 128 | 134 | 32 | 8 | 1363 |

## 5 Concluding Remarks

We have proposed an automatic expert knowledge retrieval strategy that transforms human knowledge into information that can be easily implemented in computer programs. The experimental results show that the error rate of our strategy is low and errors in annotated game records are often detected. Using the program, a human expert can easily input Chinese Chess knowledge into the algorithm and confirm the computer's choices.

There are many books covering all phases of Chinese-Chess games. The current method of collecting data is to translate manually each game record into a computer file, which is very time consuming. Furthermore, the process is subject to human error. An automatic game-record-generating system can use either OCR (Optical Character Recognition) to obtain data from published works, or computer agents to find text data on the Web. It can then apply our automatic expert knowledge retrieval strategy to construct the information. In the future, we will incorporate machine learning techniques into the strategy to generate a new evaluation function based on the abstracted expert knowledge.

## References

1. Chang, S.S., Kuo, L.P.: Si Tsan Chong Ti Wu Chu (The Mistakes in Playing Chinese Chess). Su Ron Qi Yi Publishing House (2001)
2. Chi, R.S.: Hsiang Chi Pu Chu Chu Yau (The Points of Opening Strategies in Chinese Chess). San Hai Wun Hua Publishing House, p. 330 (1990)

3. Chiang, C.S.: Lio Sin Pu Chu Sin Pien Tan Suo (Discovery of The New Variations of Popular Opening Strategies in Chinese Chess). Chen Du Xi Tai Publishing House (1996)
4. Cho, T.Y., Tsu, C.G.: Xiao Lie So Pao Run Min Ti Yu Publishing House (1990)
5. Condon, J.H., Belle, K.T.: In: Frey, P.W. (ed.) Chess Skill in Man and Machine, 2nd edn., pp. 201–210. Springer, New York (1983)
6. Fang, H.R., Hsu, T.-s, Hsu, S.C.: Construction of Chinese Chess Endgame Databases by Retrograde Analysis. In: Marsland, T., Frank, I. (eds.) CG 2001. LNCS, vol. 2063, pp. 96–114. Springer, Heidelberg (2002)
7. Gin, C.T., Yan, D.: Hsiang Chi Chong Chu Tsan Su Ja Si (Analysis of Middle Game Techniques in Chinese Chess). Su Ron Qi Yi Publishing House (1986)
8. Huang, S.L.: Hsiang Chi Kai Chu Tsan Li (Theory of Chinese Chess Opening Games). Shi Che Wun Wu Publishing House (1986)
9. Huang, S.L.: Shi Tsan Chong Chu Tsan Li (Theory of Chinese Chess Middle Games). Shi Che Wun Wu Publishing House (1986)
10. Hsu, T.-s., Liu, P.Y.: Verification of Endgame Databases. ICGA Journal 25(3), 132–144 (2002)
11. Iida, H.: Heuristic Theories on Game-Tree Search. Ph.D thesis. Tokyo University of Agriculture and Technology, Tokyo (1994)
12. Jansen, P.: Problematic Positions and Speculative Play. In: Marsland, T.A., Schaeffer, J. (eds.) Computers, Chess, and Cognition, pp. 169–182. Springer, New York (1990)
13. Levinson, R., Snyder, R.: Distance. Toward the Unification of Chess Knowledge. ICCA Journal 16(3), 228–229 (1993)
14. Lincke, T.R.: Strategies for the Automatic Construction of Opening Books. In: Marsland, T., Frank, I. (eds.) CG 2001. LNCS, vol. 2063, pp. 74–86. Springer, Heidelberg (2002)
15. Lincke, T.R.: Position-Value: Representation in Opening Books. In: Schaeffer, J., Müller, M., Björnsson, Y. (eds.) CG 2002. LNCS, vol. 2883, pp. 249–263. Springer, Heidelberg (2003)
16. Tu, J.M.: Hsiang Chi Tsan Chu Li Dian (Bible of Endgame Examples in Chinese Chess). San Hai Wun Hua Publishing House (1990)
17. Wu, G.L.: Hsiang Chi Pin Fa (Strategies of Chinese Chess Opening Games). A Computer Software published by Sohare Information Co. Ltd. (1998)
18. Wu, G.L.: 2001 Chung Kuo Hsiang Chi Ker Run Sai (Chinese Chess National Personal Contest in 2001). A Computer Software published by Sohare Information Co. Ltd. (2001)
19. Yen, S.J., Chen, J.C., Yang, T.N., Hsu, S.C.: Computer Chinese Chess. ICGA Journal 27(1), 3–18 (2004)