# Machine-Learning of Shape Names for the Game of Go

Kokolo Ikeda, Takanari Shishido, Simon Viennot

Japan Advanced Institute of Science and Technology
kokolo@jaist.ac.jp
sviennot@jaist.ac.jp

**Abstract.** Computer Go programs with only a 4-stone handicap have recently defeated professional humans. Now that the strength of Go programs is sufficiently close to that of humans, a new target in artificial intelligence is to develop programs able to provide commentary on Go games. A fundamental difficulty in this development is to learn the terminology of Go, which is often not well defined. An example is the problem of naming shapes such as Atari, Attachment or Hane. In this research, our goal is to allow a program to label relevant moves with an associated shape name. We use machine learning to deduce these names based on local patterns of stones. First, strong amateur players recorded for each game move the associated shape name, using a pre-selected list of 71 terms. Next, these records were used to train a supervised machine learning algorithm. The result is a program able to output the shape name from the local patterns of stones. Including other Go features such as change in liberties improved the performance. Humans agreed on a shape name with rate about 82%. Our algorithm achieved a similar performance, picking the name most preferred by the humans with rate about 82%. This performance is a first step towards a program that is able to communicate with human players in a game review or match.

## 1 Introduction

Until recently, the research about the game of Go was focused on obtaining strong programs, but the best programs are now able to win against professionals with a 4-stone handicap, which is a level sufficient to play against most amateur players. Apart from strength, a new target for research is now to entertain human players or teach them how to improve at the game. Beginner players often learn the game from strong players, but strong players are not all good teachers, especially because other skills than strength are required for entertaining or teaching. There are not so many skillful teachers and they are expensive, so there is a high need for programs that would be able to teach the game to the players. Ikeda et al. proposed the following 6 requirements for such a program: 1) acquire an opponent model 2) control the advantage of the board position 3) avoid unnatural moves 4) use various strategies 5) use a reasonable amount of time for each move and resign at the correct timing 6) comment on the game after it is finished [1].

In this research, we are interested in the last requirement of making the program able to comment on the moves of the player. We consider only a sub-problem, which consists of using the correct shape names (like Attachment or Hane) to refer to game moves. Since the Go board is very large (usually 19x19), Go players usually refer to moves with such shape names instead of the coordinates that are used in many other games. Handling correctly these shape names is a pre-requisite step for generating comments with a program, but it is also useful in itself to help beginner players understand and memorize the names of the shapes. In this research, we use machine learning to learn the shape name associated to a given move and a given board state. The target game here is the game of Go, but it must be noted that a similar method could be applied to other games, for example to Chess, to output the name of tactical moves such as "fork" or "discovered attack" from the board state.

## 2 Related Works

With the improvement of algorithms and hardware performance, the level of computer players has now reached a sufficient level of strength for most games, and more research is now done in the field of entertainment and naturalness. For example, in the case of Mario Bros, which is a representative side-scrolling game, there are computer competitions not only for finishing the level as fast as possible, but also for playing as human-like as possible, or for generating levels as fun as possible for humans [2].

Ikeda et al. proposed 6 requirements for entertaining players at the game of Go, with a concrete approach for playing varied strategies or avoiding unnatural moves [1], but no concrete approach was proposed for the requirement of comments or more generally communication with the players.

A trial mode of discussion and comments can be found in some commercial programs. For example, in "Yasashii Igo", the computer players are able to speak and can use to some extent the shape names instead of the coordinates, with sentences like "The cut is a good move". In the recent program "Tencho no Igo 5", the shape of the moves can be read by the voice of a professional female player, which helps to improve the user experience [3]. It is not publicly disclosed how these programs transform a move into the corresponding shape name, but it is probable that they are using a rule-based system with a list of conditional statements. One goal of our research is to develop a reproducible and systematic method for such systems.

If it was possible to associate perfectly a shape name to a move in a given board position, rule-based systems could be sufficient. However, there are many shapes for which the difference is subtle and difficult to express simply, such as "Magari vs Osae", "Nobi vs Hiki", "Tsume vs Extension". A classical way to face such a kind of problem is to use supervised machine-learning [4]. In supervised machine learning, a large set of inputs with the associated correct outputs is needed, and then the parameters of some function model that relates the input to the output are optimized. For example, from a list of 100 examples of Nobi

and Hiki, a machine-learning method can possibly find the relation between the board positions and these shape names, and find the right shape names for a given unknown board position, even if the definition of Nobi and Hiki is not perfectly clear.

Machine-learning can be done through many possible algorithms, that can be applied to learn many candidate function models between the input and the output. Some important factors to consider are the number of different elements of the input, and whether the input and the output are discrete or continuous variables. In this research, machine-learning is done with decision trees, but other classical algorithms like neural networks or support vector machines could be considered too. Decision trees have the advantage of giving a result that can be analysed, making it possible to know for what set of conditions on the inputs a given output will be obtained.

## 3 Proposed Approach

In this research, our goal is to create a program able to tell the shape name of a move, when given a move in a board position. The research is done in the following order.

1. First, we evaluate the possibilities of existing programs to tell the name of moves. Tencho Igo 5 is one strong program that has the ability of telling the shape name of the moves, so we evaluate its performance with strong human players.
2. Next, we perform a supervised machine-learning algorithm for single-move shapes. Some Go terms refer to a set of consecutive moves instead of a single move, but such higher-level terms are not considered in this research and left as a future work. The learning data is a set of shape names each associated to a move and a board position. It is gathered with the help of some strong Go players. Then, we design features that seem promising to distinguish the shapes, like the absolute position or surrounding pattern of stones. We output the value of these features in a file, and the associated shape name given by the human players. Classical machine-learning algorithms can then be performed on this file. We use a simple decision-tree learning algorithm.
3. Then we ask professional players to evaluate the performance of our program at telling the shape names of moves. Shape names are not always unique for a given move, so the professionals were asked to evaluate the shape names in a graded scale of satisfaction, instead of a binary correct/incorrect way.

## 4 Gathering of learning data and performance of existing methods

### 4.1 Limited set of target shapes

In this research, we try to learn the names of the most basic and classical Go shapes. As shown in Table 1, we selected 71 basic shapes to be the tar-

get of the learning. For example, upper-Attachment, lower-Attachment, outside-Attachment, inside-Attachment are not distinguished and are all classified as an Attachment; high one-space Approach or two-space Approach are not distinguished and all considered as an Approach. Moreover, terms referring to the meaning of a move more than the shape are not included in the list, such as Attack, Defence, Ladder-breaker, Kusuguri, Ko-threat, Yosumi or Kikashi.

### 4.2 Gathering of Learning data

The learning data needed for this research consists of a list of board positions, with the move played in that position and the name of the shape associated to this move. To gather this learning data, we asked the cooperation of some strong Go players. We asked them to input the name of the shapes with the free software Multigo [5], which can be used to read and also add comments to the record of a Go game.

We use an input format such as "Tobi, Extension(90)", that allows to record not only one name for the shape, but multiple candidates. Multiple candidates are allowed because it is frequent that the same move can be referred by multiple shape names. The number between parenthesis after a candidate name for the shape is an evaluation between 70 and 100 by the human player of how much this name is adapted to refer to the targeted shape. The evaluation of the first candidate is always 100 points and so is not written. The players that recorded the shape names and their evaluation were told that 90 corresponds to a name almost as adequate as the first candidate, a value of 80 to a name that feels not so right, and a value of 70 to a name that feels a bit strange but could still be possible.

**Table 1.** Target shapes and number of appearances

| | | | | |
|---|---|---|---|---|
| Connection(1404) | Sagari(223) | Guzumi(88) | Kata(50) | Tsukidashi(18) |
| Osae(1062) | Extension(209) | Tobitsuke(87) | Soi(46) | Wariuchi16) |
| Hane(940) | Butsukari(203) | Pincer(84) | Narabi(46) | Tobikomi(10) |
| Atari(827) | Hai(193) | Watari(80) | Kado(44) | Keima-connection(9) |
| Nobi(639) | Hiki(192) | Hanedashi(67) | Tobisagari(40) | Counter-pincer(7) |
| Push-through(612) | Ko-tori(176) | Tori(67) | Hasamitsuke(39) | Hoshishita(7) |
| Tobi(575) | Approach(170) | Shimari(66) | Ogeima(37) | Ryokakari(7) |
| Cut(531) | Kaketsugi(151) | Kake(66) | Hirakizume(37) | Hekomi(6) |
| Attachment(441) | Nige(139) | Uchikomi(65) | Horikomi(36) | Geta(4) |
| Keima(386) | Kakae(135) | Suberi(64) | Oki(35) | Takamoku(4) |
| Kosumi(352) | Komoku(133) | Boshi(62) | Hazama(26) | Mokuhazushi(5) |
| Nuki(351) | Fukurami(123) | Warikomi(62) | Tachi(26) | |
| Oshi(302) | Hoshi(105) | Tsume(60) | Tsukekoshi(20) | |
| Nozoki(295) | Kosumitsuke(103) | Takefu(54) | Hanekomi(18) | |
| Magari(251) | Atekomi(101) | San-san(54) | Sashikomi(18) | |

### 4.3 Learning data

We describe in this section the details of the gathered learning data. The players that recorded the shape names are 6 strong players, ranked above 4 dan on the KGS server. The game records come from 60 games between professionals or top amateurs players, for a total of 11,526 moves. As shown in Table 1, the most frequent shape found in these moves is the Connection (Tsugi), found 1404 times, followed by the Osae, 1062 times.

There are big differences between the number of appearances of the shapes in the game records. Some famous shapes appear less than 10 times, mainly because despite the fact that they are famous, they appear rarely in a game. It implies that it will probably be hard to classify them during the machine-learning.

Different game records were dispatched to the game players, except for one game record (with 117 moves) that was given to all players. The frequency at which the players recorded the same shape name on this common game record was found to be 82.2% when considering only the first candidate, and only 87.0% even when considering up to the second candidate. Despite the fact that 5 of the 6 players come from the same community, a difference of almost 20 points of percentage in the name used for a move shows the difficulty of defining and assigning a shape name to a move. Figure 1 shows some examples where the human players did not give the same shape name.

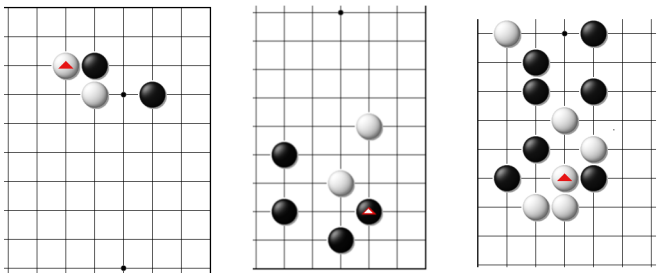

**Fig. 1.** Shapes labeled with multiple names by human players. Left: Hane, Osae. Middle: Kosumi, San-san. Right: Osae, Push-through, Cut, Magari, Guzumi

### 4.4 Performance of existing methods

In this section, we evaluate the performance of two existing programs for naming the shapes of Go moves. The first program is the popular commercially available "Tencho No Igo 5", which has a mode for reading the shape names of the moves.

We gave 4 of the game records to the Tencho program, and it returned the shape name for a total of 262 moves. Then, we asked to a strong Go player to evaluate the names given by Tencho and to classify them in the following

4 categories: 1) Correct shape name, 2) Unnatural shape name, 3) Absence of shape name, 4) Incorrect shape name.

The 262 moves were divided in these 4 categories as follows. 1) 65.6 %, 2) 2.3 %, 3) 30.2 %, 4) 1.9 %. It shows that the number of mistakes in the shape names is small, but that in many cases, no shape name was given. Especially, high-level shapes such as Atekomi, Guzumi or Hasami-tsuke are not output by the program. We guess that for a commercial software, the priority was given to avoiding mistakes rather than trying to give a shape name to all the moves.

Nomitan is a Go program developed in Japan Advanced Institute of Science and Technology (JAIST) in the Ikeda and Iida laboratories. The program contains a hand-coded set of 554 conditional rules that outputs the shape name corresponding to a move. When applied on the 11,526 moves of section 4.3, the output of Nomitan matched the first candidate of human players with a rate of 73.7%. The rate was 76.6% when considering up to the second move. Compared to the matching rate of 82.2% and 87% between human players, there is almost a drop of 10 percentage points. This feature of Nomitan was originally designed for a $9 \times 9$ board[6], and the absence of shapes such as Extension (Hiraki) and Boshi is part of the reason for this drop.

The goal of this paper is to propose an approach that gives better performance than the current existing programs.

## 5 Machine learning and preliminary experiments

### 5.1 Design of the features

It is preferable for the supervised machine-learning to use as inputs not the raw board state, but the value of some well-chosen abstract features. The chosen features have a direct influence on the performance of the machine-learning. Too rough features prevent a good representation ability, but too detailed features lead to overfitting and a poor generalization performance.

As a starting point, we have used the features already implemented in the rule-based shape naming functions of Nomitan. These features are a simplified version of features initially designed for a strong Go program. The only feature really specific to shape naming is the feature related to the Cut. After removing some useless features, this set of candidate features contains 25 features. The distance used in the features is not the euclidean distance, but the R-distance defined by $d(\delta x, \delta y) = \delta x + \delta y + \max(\delta x, \delta y)$ as in [7].

- F1 (2 features) (x, y) coordinates so that $y \leq x \leq 10$ after rotation and reflection. Needed to categorize shapes such as Hoshi and Komoku.
- F2 Line of the stone, i.e. the distance to the closest board edge.
- F3 R-distance to the closest stone of the same color. If there are no other surrounding stones, a distance of 2 means a Narabi, 3 a Kosumi, 4 a Tobi and 5 a Keima.
- F4 R-distance to the closest enemy stone (opposite color). If there are no other surrounding stones, a distance of 2 is an Attachment, 3 a Kado or Kata.

– F5 Line of the closest stone of the same color.
– F6 Line of the closest enemy stone. Often useful to distinguish Kado and Kata, Oshi and Hai.
– F7 Number of enemy stones with only one liberty left in horizontal or vertical contact. If this value is bigger than zero, the shape is often a Nuki.
– F8 Number of enemy stones with two liberties in horizontal or vertical contact. If this value is bigger than zero, the shape is often Atari.
– F9 Number of stones of the same color with only one liberty left in horizontal or vertical contact.
– F10 Number of stones of the same color with only two liberties left in horizontal or vertical contact.
– F11 Number of liberties of the group of stones containing the stone just played.
– F12 Whether there is a stone of the same color left-down diagonally, and enemy stones on the left and the down. This is directly related to a Cut (Kiri).
– F13 (12 features) State of the 12 board intersections at R-distance 2 to 4 (0: empty, 1: stone of the same color, 2: enemy stone, 3: out of the board).

### 5.2 Machine-learning algorithm

The supervised machine-learning is done with J48 (implementation in Java of C4.5) from the data-mining Weka software [8][9] as follows.

1. The raw gathered data consists of a collection of board positions with their associated moves and corresponding shape names, in the sgf file format. However, Weka cannot use directly such files, so we use first a script in combination with Nomitan to compute the value of the features described in Section 5.1, and output them in a csv file that can be read with Weka.
2. The csv file is read through Weka, and some information useless for the machine learning (such as the game record number or the number of moves) is removed. Then, a decision tree is constructed with J48, and the matching ratio is obtained. It takes less than 1s on a typical PC, and even the 10-fold cross-validation takes less than 10s.
3. We also obtain the matching ratio between the decision tree output and the second shape name candidate.

### 5.3 Preliminary experiment results

In order to determine how well our machine-learning method works, we run a preliminary experiment with the features described in Section 5.1. The matching ratio after the machine-learning is as follows:

– 75.3% matching on the first candidate
– 76.8% matching when considering up to the second candidate

It is already slightly better than the 73.7% and 76.6% of the rule-based system of Nomitan, but still far from the 82.2% and 87% of matching ratio between humans.

We compare in Table 2 the results of Nomitan and of the machine-learning on some shapes related to the surrounding pattern of stones. The machine-learning is clearly not very efficient on these shapes, which shows that some improvement of the features is needed. The shapes appear a few hundred times, so we can expect the machine-learning to work efficiently after a re-design of the features.

**Table 2.** Proportion of correct answers for shapes related to surrounding patterns

| Correctness ratio | Nomitan | Machine-learning |
|---|---|---|
| Magari(251) | 76.6 | 41.6 |
| Push-through(612) | 83.2 | 59.3 |
| Oshi(302) | 85.4 | 65.2 |

## 6 Improvement of the features and evaluation experiment

In this section, we describe how we improved the features to solve the problematic shapes presented in the previous section.

### 6.1 Improvement of the features

The features related to the surrounding pattern of stones use 12 board intersections. The problem of these features is that it does not take into account rotation and reflection equivalences. For example, the 8 patterns of Figure 2 all correspond to the same 3x3 pattern when rotations and reflections are considered. The corresponding shape name is usually called Push-through (De). If the 8 patterns are considered separately, it increases the number of conditions needed in the decision tree, and more importantly, the quantity of learning data for each pattern is greatly reduced. So, we included rotations and reflections in the design of the features. The 8 patterns are reduced to a canonical pattern with the following order: place under the played stone as many stones from the player as possible, then as many enemy stones as possible; then on the left, the right, the bottom-left, the bottom-right and the top-left.

On the example of Figure 2, (b) and (e) are given the priority with condition 1 of placing stones of the player below, and then the priority is given to (e) with condition 2 of placing stones of the players on the left. Then, all 8 patterns are represented by the single (e) canonical pattern.

The effect of this improvement was quite important, with an increase of 5 points of the learning matching ratio. The rate of correct answers for the 3 shapes of Table 2 raised a lot to reach Magari: 75.5%, Push-through: 83.6%, Oshi: 81.9%. We also tried to use the Tengen point (central point of the Goban)

as a reference in the order, to distinguish patterns oriented differently towards the Tengen, but the improvement of the learning matching ratio was only 1 point in that case. Keeping the information about the orientation towards the Tengen is useful for some shapes such as Oshi and Hai, but for most other shapes, it is more important to group the patterns under a single canonical pattern. It increases the number of learning examples for this canonical pattern.

In addition, we have removed some too-fine features and tuned some parameters. The improvement from each of the following modifications was around 0.3 point so we cannot be sure that all of them would be welcome on a different set of learning data.

- Feature related to the line of the stone is removed.
- The 8 intersections at an R-distance of 5 are added to the stone pattern.
- After rotation, the 3 intersections above and the 3 intersections below the pattern are added to the pattern.
- After rotation, the number of stones of the player on the bottom-left, left, bottom-right, bottom of the bottom-left, bottom of the bottom, bottom of the bottom-right are added.
- Confidence parameter of J48 is set to 0.1 instead of the default 0.25 value.
- SubTree parameter of J48 is set to false.

After all these improvements, the matching ratio on the first shape name reached 82.0%, and 85.4% when considering up to the second shape name. It is significantly better than the performance of the rule-based system of Nomitan, and very close to the 82.2% and 87.0% matching ratio between humans.

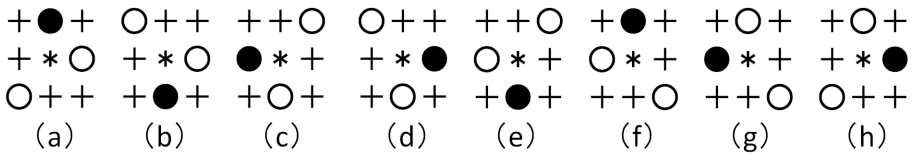


**Fig. 2.** 8 equivalent patterns, often corresponding to a Push-through (De)

### 6.2 Relative importance of the features

We have also tested the influence on the matching ratio of removing some features or changing the size of the local patterns. If only the local patterns (F13) are used, the matching ratio drops from 82.0% to 75.3%. If patterns of size 4 are used instead of 5, it drops to 70.3%, and then to 61.6% with patterns of size 3. If only F7 and F8 are removed, the maching ratio drops from 82.0% to 78.9%. The recall of Atari and Nuki shapes is significantly decreased by 17% and 37%. These shapes are clear examples where local patterns of stones are not sufficient.

### 6.3 Proportion of correct shape names and remaining problems

The global matching ratio of the shape names is around 82%, but there are big differences between the shapes, some of them being better categorized than others. On Figure 3, we show the proportion of correct naming for the different shapes, in function of the number of appearances of the shape in the game records (logscale). The general tendency is that the proportion of correct naming increases with the number of examples, but even for a similar number of appearances in the game records, there is a large spreading of the proportion of correct naming between the shapes. For example, for shapes such as Hoshi, Komoku, Takamoku, Mokuhazushi, the proportion of correct naming is almost 100% even though these shapes appear only a small number of times. This is because these shapes are easily identified by their position on the board.

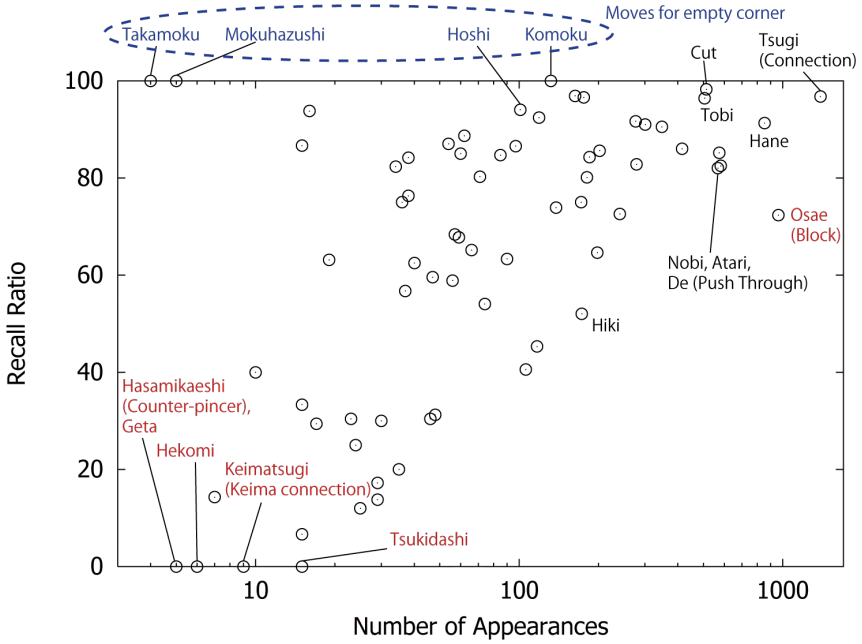


**Fig. 3.** Proportion of correct names in function of the number of appearances

By contrast, Hirakizume, another shape with a similar low appearance ratio has a very low proportion of correct naming, around 30%. Figure 4 shows an example named as a Hirakizume by the program, but this shape is clearly not a Hirakizume. The reason for this error is that the rule for Hirakizume in the decision tree is "third or fourth line, with a stone of the same color at R-distance 6 and an enemy stone at R-distance less than 6". These conditions are indeed observed in a Hirakizume shape, but more conditions are needed such as "the stone of the same color and the enemy stone must also be on the third or fourth

line". However, there are only 35 appearances of Hirakizume (0.3%) in the game records, which is probably not sufficient to deduce the full set of conditions characteristic of a Hirakizume shape. This current limitation of the machine-learning could be improved by creating more learning data.

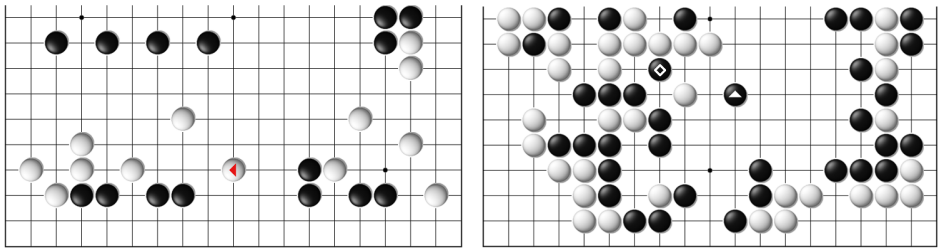

**Fig. 4.** Examples of naming mistakes. Hirakizume instead of Uchikomi (left), Boshi instead of Nozoki (right)

### 6.4 Evaluation by a professional player

In order to evaluate if the shape naming obtained with the machine-learning is satisfactory from the point of view of humans, we asked to a professional player (Nihon-kiin 6-dan player) to evaluate the shape names recorded in some game records. As explained in Section 3, we use a satisfaction scale instead of just a binary correct/incorrect scale.

First, we selected randomly 5 game records from Section 4.2 where the shape names were recorded by strong human players. For 3 of these game records, we kept only the first 100 moves, and for the 2 other game records, we kept only the moves from 101 to 200. Then, for this total of 500 moves, we named the shapes with the decision tree obtained from the machine-learning of Weka.

The professional player was not told whether the shape names were recorded by human players or by an algorithm, and was asked to rank the shape name of each move in the following scale.

1. A professional would use the same shape name.
2. A professional would not use this shape name, but it is acceptable.
3. This shape name seems a bit awkward.
4. This shape name is clearly incorrect.

Moreover, the shape name of each move was evaluated with a total score, with roughly the following scale: 90 points if this shape name could be used in a professional review of a game, 80 points if it could be used in a game review by 3-dan amateur players, and 70 points if it could be used by 6-kyu lower-level players. We show the result of the shape name evaluation in Table 3. The

**Table 3.** Evaluation by a professional of the shape names given by amateur players, and by the machine-learning method. Number of times of bad names and total score.

| Game records from | (2) | (3) | (4) | Total score |
|---|---|---|---|---|
| Human Players | 5.4 | 3.8 | 4.0 | 84.6 |
| Weka | 4.4 | 3.6 | 4.6 | 83.8 |

columns (2), (3) and (4) show the average number of times in 100 moves where the shape name was evaluated as (2), (3) or (4) by the professional player.

The machine learning total score has an average of only 0.8 points lower than the average total score of human players. The performance of the machine learning is close to strong amateur players.

## 7 Conclusion

In this paper, we presented a method to name automatically the shape of moves for the game of Go. We used machine-learning on learning data recorded by strong human players, who were asked to record the name of each move in a set of game records. Abstract features were used in the machine-learning, and after optimization, the shape naming quality is close to the level of strong amateur players, both in terms of matching ratio and from the point of view of a professional player. There is a limited number of small mistakes, but the main problem is that some big mistakes are found, mainly on shapes that appear rarely. Hopefully, it could be improved by using more learning data, or designing better abstract features. Naming correctly the shape of the moves will probably be important in the future, in order to create programs that are able to teach the game to human players.

## References


1. K. Ikeda, S. Viennot, Production of Various Strategies and Position Control for Monte-Carlo Go - Entertaining human players, IEEE-CIG145-1522013
2. IEEE-CIG (Computer Intelligence and Games) Competitions http://geneura.ugr.es/cig2012/competitions.html
3. http://batora1992.blog.fc2.com/blog-entry-17.html
4. C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2007.
5. http://www.ruijiang.com/multigo/
6. JAIST CUP 2012, Game Algorithm Competition, 9x9 Entertainment Go Contest http://www.jaist.ac.jp/jaistcup/2012/jc/9ro.html
7. R. Coulom, Computing Elo Ratings of Move Patterns in the Game of Go, ICGA Workshop, 2007
8. J. R. Quinlan, C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993
9. J48, open source java implementation of C4.5 algorithm http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html