IPSJ Transactions on Programming Vol. 4 No. 3 42–54 (June 2011)

*Regular Paper*

# Associative Search on Shogi Game Records

Kobkrit Viriyayudhakorn[†1] and Mizuhito Ogawa[†2]

Associative search is information retrieval based on the similarity between two different items of text information. This paper reports experiments on associative search on a large number of short documents containing a small set of words. We also show the extension of the set of words with a semantic relation, and investigate its effect. As an instance, experiments were performed on 49,767 professional (non-handicapped) Shogi game records with 1,923 next move problems for evaluation. The extension of the set of words by pairing under semantic relations, called *semantic coupling*, is examined to see the effect of enlarging the word space from unigrams to bigrams. Although the search results are not as precise as next move search, we observe improvement by filtering the unigram search result with the bigram search, especially in the early phase of Shogi games. This also fits our general feeling that the bigram search detects castle patterns well.

## 1. Introduction

Associative search is information retrieval based on the similarity between two items of text information. Similarity is computed based on associative information generated from statistics regarding a large text database.

Associative search engines, which perform associative search, have been implemented in recent decades. For example, the keyword associator [1] was proposed by Watanabe from Fujitsu laboratory, and another, the Generic Engine for Transposable Association computation (GETA) [2] was proposed by Takano from Hitachi laboratory. GETA is continuously being developed in National Institute of Informatics (NII).

The former pre-computes keyword association and is applied in the Group Idea Processing System (GrIPS) [3], which computes the similarity between two ideas.

†1 School of Knowledge Science, Japan Advanced Institute of Science and Technology, Japan
†2 School of Information Science, Japan Advanced Institute of Science and Technology, Japan

The latter, GETA, quickly handles a dynamic associative search on more than ten million documents. It was first applied as DualNavi [4] over Heibonsha Encyclopedia. Now, there are several web sites based on GETA, such as, Webcat Plus[⋆1] over bibliographical data of libraries of all Japanese universities, Imagine[⋆2], Cultural Heritage Online [5][⋆3], and Pictopic[⋆4]. GETA are applied for database of documents in natural languages, in which each document is mostly quite lengthy, say 500-10,000 words. The whole set of words appearing in these databases is huge due to the nature of natural languages. To our limited knowledge, we found no applications of GETA on a document database in an artificial language with a small set of words.

Shogi (Japanese chess) is a two player game, and is popular in Japan. Its game state consists of 40 piece positions (including captured pieces) with non-duplicating locations, and totalling over 2,296 ($= (9\times 9+1)\times(8+6)\times 2$) possible piece positions. We regard a Shogi game state as a 40 word document.

Computer Shogi, which is much more difficult than chess as an example of artificial intelligence due to the larger search space, has a long history starting from the late 70's [6]. In the early days, implementations were a combination of Shogi game record search (for early phases) and min-max like search. They stayed at the novice to medium amateur player level. In 2005, Bonanza[⋆5] introduced machine learning techniques, which obtained great success. Nowadays, top-ranked computer Shogis are approaching to the professional level.

This paper reports experiments of association search, which completely apart from existing computer Shogi techniques, on a large number of short documents over a small set of words. We also show the extension of the set of words with semantic relations, and investigate its effect. Experiments are performed on 49,767 professional (non-handicapped) Shogi games records, and evaluated by prediction accuracy on 1,923 next move problems.

As preprocessing, we first generate and normalize all Shogi game states

⋆1 http://webcatplus.nii.ac.jp
⋆2 http://imagine.bookmap.info
⋆3 http://bunka.nii.ac.jp
⋆4 http://photobank.pictopic.info
⋆5 http://www.geocities.co.jp/bonanza_shogi

(5,613,402 states) appearing in the Shogi game records. Shogi game states can be regarded as a corpus of an artificial language with

- Small word space (when a word is a piece position),
- Short documents (each game state contains exactly 40 words), and
- A huge number of documents (game states).

Here, normalization means that the mover side is always set to the black side. Then, the core associative information, WAM (Word Article Matrix), of GETA, is generated. During the experiments, all illegal moves (violating Shogi rules) are deleted from the search results. In the implementation, we use GETAssoc, which is a variant of GETA intended for personal use and easy installation on Linux.

We further extend the set of words by pairing under a semantic relation, to see the effect of enlarging the word space. We call this pairing *semantic coupling*. In Shogi game states, a word is a piece position. We focus on the relation between the position of two pieces on the board, if one piece can move to another (i.e., if they are on different sides, one can capture another; if they are on the same side, one ties to another piece). Then, the number of semantic couplings becomes nearly 20,000. We refer to a single piece position as a unigram, and a semantic coupling as a bigram.

The experiments evaluate the accuracy of associative search regarding the solutions of next move problems. We also compare prediction accuracy under phase categorization of next move problems (which was manually done by a medium-level amateur Shogi player).

First, experiments are performed only with unigrams to set appropriate similarity measures and the number of article summaries (characteristic keywords). Second, experiments with bigrams are compared. We had a general feeling that associative search with bigrams detects castle patterns well. Last, we tried the combination of unigrams and bigrams; their union, and filtering by the latter.

As our conclusion, although the search results are not so precise as the next move search of Shogi, we observe the improvement by filtering the unigram search result with the bigram search (intersection of the sets of the top-$n$ results), especially in early phases of Shogi games.

Our experiments are preliminary. In the context of computer Shogi, an associative search is a naive methodology; for instance, machine learning-based methods use several thousands parameters, whereas an association search uses only one similarity measure, which fits a likely search rather than an exact search. From the computer Shogi viewpoint, the associative search will have better fitness for categorizing Shogi game records into, say, their game phases and/or their game opening patterns. They are left for future work.

This paper is organized as follows. Section 2 explains associative search and GETA. Section 3 explains Shogi fundamentals. Section 4 shows how to generate Shogi game states from Shogi records. Section 5 describes associative search on Shogi database. Section 6 gives the testbed system description. Sections 7 and 8 show the experimental results for unigrams and bigrams, respectively. Section 9 concludes the paper.

## 2. Associative Search and GETAssoc

### 2.1 Associative Search

In an associative search, an article is regarded as a multiset [7] of tokens (typically, an article is a document and a token is a word). A query is a (multi)set of tokens, and the search result is a ranking among tokens with respect to a given similarity measure.

Let $ID_1$ be a set of articles, and let $ID_2$ be a set of tokens.

**Definition 1.** An *associative system* is a quadruplet $\mathcal{A} = (ID_1, ID_2, a, \mathrm{SIM})$ where $a$ is an associative function and SIM is a similarity function such that

$$\begin{cases} a: & ID_1 \times ID_2 \to \mathbf{N} \\ \mathrm{SIM}: & ID_2 \times MP(ID_1) \to \mathbf{R}_{\geq 0} \end{cases}$$

where $\mathbf{N}$ is the set of natural numbers, $\mathbf{R}_{\geq 0}$ is the set of non-negative real numbers, and $MP(ID)$ is the set of non-empty multisets consisting of elements in $ID$ (which is equivalently the set of functions from $ID$ to $\mathbf{N}$). We say that $\mathcal{A}^t = (ID_2, ID_1, a^t, \mathrm{SIM}^t)$ is the transpose of $\mathcal{A}$, where $a^t(y, x) = a(x, y)$ and a given $\mathrm{SIM}^t : ID_1 \times MP(ID_2) \to \mathbf{R}_{\geq 0}$.

For $X \subseteq ID_1$ (resp. $ID_2$) and $n \in \mathbf{N}$, let $A(X, n)$ (resp. $A^t(X, n)$) be the function collecting the top $n$-elements in $ID_2$ (resp. $ID_1$) with respect to the similarity $\mathrm{SIM}(y, X)$ (resp. $\mathrm{SIM}^t(y, X)$) for $y \in ID_2$. An *associative search* is

$A^t(\{y \mid (y,v) \in A(X,m)\}, n)$

for $m \in \mathbf{N}$.

In the definition of an associative search, $m$ is not specified. From empirical study, in GETAssoc (see Section 2.3) $m$ was set to 200 by default, to balance efficiency and precision. Note that during an associative search, we first compute $A(X,m)$. This result is regarded as a *summary* that characterizes $X$.

Typical examples of associative searches are:

- $ID_1$ is the set of documents, $ID_2$ is the set of words, and $a(d,w)$ is the number of occurrences of a word $w$ in a document $d$. In this case, an associative search is documents-to-documents.
- $ID_1$ is the set of words, $ID_2$ is the set of documents, and $a^t(w,d) = a(d,w)$. In this case, an associative search is words-to-words.

Note that an associative search does not require structured documents, and the associative search ignores the ordering of words; it does not distinguish between, for example "Weather is not always fine" and "Weather is always not fine".

From now on, we fix $ID_1$ as the set of documents and $ID_2$ as the set of words. To search a document $d$ with a query $q$, $d \in ID_1$ and $q \in MP(ID_2)$. Usually $q \notin ID_2$ (i.e., there is no precise matching). In this situation, an associative search is performed as

**Step 1.** A summary (characteristic keywords) of $q$ is produced. This is performed by taking top-$m$ words among $w \in q$ by $\mathrm{SIM}^t(w, ID_1)$. $ID_1 \in MP(ID_1)$ is regarded as the multiset consisting all documents appearing precisely once.

**Step 2.** For the set $q' \subseteq MP(ID_2)$ of top-$m$ words in $q$, top-$n$ documents in $ID_1$ are selected by evaluating $\mathrm{SIM}(d,q')$ for each $d \in ID_1$.

GETAssoc permits a similarity function SIM of the form

$$\mathrm{SIM}(d,q) = \sum_{t \in q} \frac{\mathrm{wq}(t,q) \cdot \mathrm{wd}(t,d)}{\mathrm{norm}(d,q)} \tag{1}$$

with the assumptions that $\mathrm{wd}(t,d) = 0$ if $t \notin d$ and $\mathrm{wq}(t,q) = 0$ if $t \notin q$. Typically,

- The value of $\mathrm{norm}(d,q)$ is dependent only on $d$. (In such cases, $\mathrm{SIM}^t$ is obtained by simply swapping wq and wd.)
- Both wq and wd are defined dependent on the association function $a$.

For an efficient associative search implementation (e.g., GETA), we assume $\mathrm{SIM}(y,X) = 0$ if $a(x,y) = 0$ for each $x \in X \subseteq ID_1$ and $y \in ID_2$.

### 2.2 Similarity Functions

GETAssoc accepts user-defined similarity functions of form in Eq. 1 as well as default similarity functions,

- Smart measure [8],
- Okapi BM25 [9],
- Cosine [10],
- Dot Product [10].

These similarity functions are defined as functions of type $ID_2 \times MP(ID_1) \to \mathbf{R}_{\geq 0}$ as in Fig. 1, where $d \in ID_1$, $q \in MP(ID_1)$, $t, w \in ID_2$, and

- $w_{q,t} = log(\frac{N}{f_t+1})$
- $w_{d,t} = log(f_{d,t}+1)$
- $f_t$ is the number of documents that contain $t$.
- $f_d$ is the number of words in document $d$.
- $f_{x,t}$ is the number of occurrences of the word $t$ in $x$.
- $N$ is the number of documents in $ID_1$.

GETA sets $\theta = 0.2, k = 0.2$, and $b = 0.75$ for the default measure in Fig. 1.

These default similarity functions are directly applied to **Step 2** of an associative search. For **Step 1**, the dual similarity functions of type $ID_1 \times MP(ID_2) \to \mathbf{R}_{\geq 0}$ are needed, which are obtained by transposing the relation as $d \in w$ (a document $d$ contains a word $w$) when $w \in d$ (a word $w$ appears in a document $d$). In practice, there are small differences, e.g., setting of stop words, but in principle, GETAssoc uses the dual functions on both document-to-document and word-to-word similarity.

### 2.3 GETAssoc and an Example

The Generic Engine for Transposable Association Computation[⋆1] (GETA) is an associative search engine developed at National Institute of Informatics (NII) [2]. Its implementation was first released in the late 90's. In 2009, GETAssoc was released as a variation of GETA, intended for personal use with easy installation.

⋆1 http://getassoc.cs.nii.ac.jp

Smart measure [8] :

$$\frac{1}{avg(f_d) + \theta(f_d - avg(f_d))} \sum_{t \in q \wedge t \in d} log(\frac{N}{f_t}) \cdot \frac{1 + log(f_{d,t})}{1 + log(avg_{\omega \in d}(f_{d,\omega}))} \cdot \frac{1 + log(f_{q,t})}{1 + log(avg_{\omega \in q}(f_{q,\omega}))} \tag{2}$$

Okapi BM25 [9] :

$$\sum_{t \in q \wedge t \in d} log(\frac{N - f_t + 0.5}{f_t + 0.5}) \cdot \frac{f_{d,t} \cdot (k+1)}{f_{d,t} + k \cdot (1 - b + b \cdot \frac{f_d}{avg(f_d)})} \tag{3}$$

Cosine similarity [10] :

$$\frac{\sum_{t \in q \wedge t \in d}(w_{q,t} \cdot w_{d,t})}{\sqrt{\sum_{t \in q}(w_{q,t}^2) \cdot \sum_{d \in q}(w_{d,t}^2)}} \tag{4}$$

Dot product [10] :

$$\sum_{t \in q \wedge t \in d} (w_{q,t} \cdot w_{d,t}) \tag{5}$$

**Fig. 1** Four similarity measures and their formulas.

We use GETAssoc version 1.1.

The key data structure of GETA is a WAM (Word Article Matrix), which represents an association function in Definition 1. WAM is usually a huge sparse matrix of which rows are indexed by names of documents and columns are indexed by words. When $ID_1$ is a set of words and $ID_2$ is a set of documents, the cross point of the row of a word $w$ and the column of a document $d$ is $a(w, d)$, which is the number of occurrences of a word $w$ in a document $d$. Then, the transpose $a^t(w, d)$ is obtained as a transposed WAM. In GETA implementation, a huge and sparse WAM is compressed either vertically or horizontally. These two compressed matrices enable us to compute association functions $a$ and $a^t$, respectively.

**Example 1.** Consider a query "twitter dollar", which is regarded as a two-word document. The WAM is shown in Table 1. The $\mathrm{SIM}^t(w, ID_1)$ has the dual form of Eq. 1, for a document $d \in ID_2$, and we set $wq(d, ID_1) = 1$, $wd(d, w) = a(w, d)$, and $norm(w, ID_1)$ as the number of documents that contain $w$. i.e., $norm(twitter, ID_1) = 1$ and $norm(dollar, ID_1) = 3$. Then, the scores of "twitter" and "dollar" are 2 and 1.333.... If we take the top-ranked word only as a summary, the most similar document to the query "twitter dollar" is "IT news".

**Table 1** A sample WAM.

| | twitter | tennis | dollar | facebook |
|---|---|---|---|---|
| IT news | 2 | 0 | 1 | 4 |
| Sports news | 0 | 2 | 1 | 0 |
| Economic news | 0 | 0 | 2 | 0 |

## 3. Shogi

### 3.1 Shogi Fundamentals

Shogi (Japanese chess) is a two-player game similar to Chess played on a $9 \times 9$-square board. The differences are that there are no distinctions like black and white on pieces, and each piece is reusable when captured. The player who has

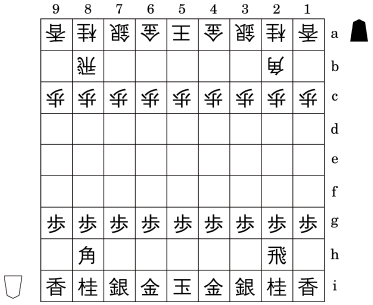



**Fig. 2** The initial state of Shogi game.

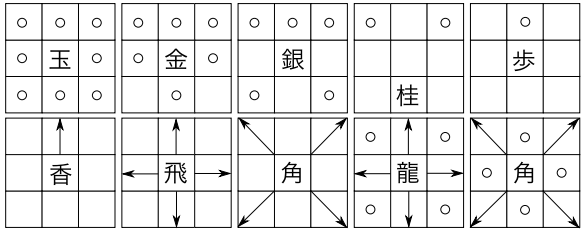



**Fig. 3** Legal moves of each piece in Shogi.

the turn of the initial move is referred to as black, and the other is referred to as white. At each move, the turn of the next move changes to the opponent. The player who has the turn is the mover, and the other is the watcher (of a game state).

Initially, each player has 20 pieces, consisting of one King (Oh or Gyoku), one Rook (Hisha), one Bishop (Kaku), two Gold generals (Kin), two Silver generals (Gin), two Knights (Keima), two Lances (Kyousha), and nine Pawns (Fu). Each piece is designed as a pentagon, and (the opposite of) its direction shows which side (black or white) it belongs to. Shogi starts with the fixed piece positions (Fig. 2).

Their legal moves are illustrated in Fig. 3. Except for the King and the Gold general, pieces can be promoted when they enter the opponent's base (the top three rows for black, and the bottom three rows for white). Note that promotion is an option of a player. Except for the Rook and Bishop, the possible moves after promotion are the same as for a Gold general. The promoted Rook has possible moves of both Rook and Silver general, the promoted Bishop has those of both Bishop and Gold general. When a player has no legal moves that allow his king to avoid capture (or, a player resigns to admit the loss), it is the end game.

In Shogi, there are handicapped games, in which an expert removes some pieces from his own initial positions when playing against a novice. We focus on non-handicapped games only in this article.

### 3.2 Game State Description

A location on the board is represented by a pair of a number from 1 to 9, which represents the horizontal position, and a lowercase alphabetical letter from a to i, which represents the vertical position. For example, **1a** is the top right corner.

We use **K** for King, **R** for Rook, **B** for Bishop, **G** for Gold general, **S** for Silver general, **N** for Knight, **L** for Lance, and **P** for Pawn. A promoted piece is represented by adding **+** in front of the letter, e.g., **+P** for a promoted pawn.

A piece position consists of either four letters, five letters, or two letters. The first two letters in the former two cases represent the location, followed by the owner of the piece and the kind of piece (with **+** if promoted). For instance, **1aWL** is the white lance at the top right corner as shown in Fig. 4, where **B** indicates the black player and **W** the white player.

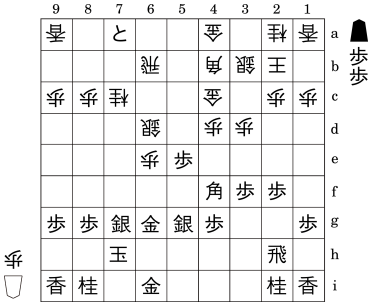
The latter case represents the captured piece. The first letter represents the owner of the piece (the capturing player). The second letter represents the kind of piece. For instance, **BP** is a captured Pawn, which is under control of the black player.

A game state is a collection of 40 piece positions without duplications of locations, but with possible duplications of captured pieces. Fig. 4 shows an example of a Shogi game state and its description.

### 3.3 Legal Moves

A move is a transition from one game state to another caused by a legal move of a piece, as shown in Fig. 3.

A move notation is, for instance, **P-7f**. The first letter represents the kind of piece, and the second letter represents the kind of action of the piece, either **-** a




BP BP WP 1aWL 1cWP 1gBP 1iBL2aWN 2bWK
2cWP ... 8cWP 8gBP 8iBN 9aWL 9cWP 9gBP 9iBL

**Fig. 4** A sample Shogi game state.

move, ***** a drop (a captured piece is reused on the board), or **x** a capture. The last two letters represent the destination position.

If a move allows the player to promote a piece, a **+** is added to the end if the promotion is taken, or an **=** if the promotion is not taken. For instance, **Nx5c+** represents a Knight capturing on **5c** with promotion.

In case where the piece is ambiguous, the starting position is inserted after the letter for the piece. For example, in Fig. 4, black player has two Gold generals, which can move to square **6h**, which can be distinguish as **G6g-6h** (from above) and **G6i-6h** (from below).

The legal move set of a game state is the collection of move notations of all possible legal moves of mover side pieces. For example, the legal move set of the game state in Fig. 4 is

P-1f L-1h P-2e R-2g R-1h R-3h R-4h R-5h R-6h ...
G-7f G-6f G-5f G6g-6h G6i-6h G-5h G-7i G-5i ...
P-9f L-9h P*6c P*6f P*6h P*7b P*7d P*7e P*7f P*7i

## 4. Shogi Game State Generation

### 4.1 Shogi Database

The 49,767 professional Shogi game records, which consist of 5,613,402 moves, are downloaded from Shogi archive[⋆1].

All game records are used as the source for generating game states as in Section 5.1, which are indexed by the record number and the number of moves (from the initial state), and used to generate the WAM.

The 1,923 next move problems are compiled from the web site[⋆2]. Each page consists of an image that represents a question game state, e.g., Fig. 4, and an answer text of the correct next move. Every piece drawn in the image is recognized by bitmap pattern matching to extract piece positions. A next move answer is extracted by string pattern matching in the text.

A Shogi game record is a sequence of move notations in chronological order from the initial Shogi game state (Fig. 2). Thus, Shogi game states are generated by tracing move notations one by one from the initial Shogi game state.

### 4.2 Game State Normalization

A piece position is conventionally recorded from the black player's viewpoint. However, a next move problem is viewed from the mover side. To avoid confusion, we normalize each generated game state to show piece positions from the mover side as below.

- If the mover is black,
  - (1) Leave it unchanged.
- If the mover is white,
  - (1) Rotate the Shogi board.
  - (2) Swap the owner of every piece on the board (including the captured pieces).

This normalization changes presentation of two players in a game state from black-white to mover-watcher. After normalization, the black player is always referred to as the mover, and the white player is always referred to as the watcher.

⋆1 http://wiki.optus.nu/shogi
⋆2 http://navy.ap.teacup.com/nobuo90

## 5. Associative Search on Shogi Database

### 5.1 Associative Search for Next Move

Associative search of similar game states to a given game state is adopted by regarding:

(1) A piece position as a word.
(2) A game state as a document.

Note that each document contains exactly 40 words.

A next move problem is a given game state, and asks the most beneficial move, e.g., G6i-6h in Fig. 4. The associative search for the next action is,

(1) For a database of game records, all generated game states are used to make a WAM.
(2) A given game state (of the next move problem) is translated into a document.
(3) Perform associative search and find the most similar game state.
(4) Find its next move from its corresponding Shogi game record.

If the next move matches the original answer of the query next move problem, we say the associative search result is correct. Since different game records may contain the same game state, there may be multiple answers. In such cases, if they contain the original answer, we also say correct.

### 5.2 Extension of the Word Space

To see the effect of enlarging a word space, we propose semantic coupling, which pairs two words under a semantic relation. More precisely, let $W$ be the set of words and let $\rightsquigarrow \subseteq W \times W$ be the binary relation over $W$. Then,

$$S(W, \rightsquigarrow) = \{(w, w') \in W \times W \mid w \rightsquigarrow w'\}.$$

In Shogi game states, a word is a piece position. We focus on the relation between the position of two pieces on the board, if a piece can move to another (i.e., if they are on different sides, one can capture another; if they are on the same side, one ties to another), e.g., (**4fBB**, **1cWP**), (**4bWB**, **6dWS**) in Fig. 4. Then, the number of semantic couplings becomes nearly 20,000. We refer to a single piece position as a unigram, and a semantic coupling as a bigram.

## 6. Experimental System Description

The system for associative search of a next move consists of two workflows. They are implemented on a 64-bit PC with Intel i7 1.6 Ghz and 6 GB memory, running Linux Ubuntu 10.10. Note that we use a single core only.

### 6.1 WAM Construction Workflow

The WAM construction workflow is shown in Fig. 5 as the horizontal solid arrows from "Shogi Game Records" to "WAM Construction". Shogi game states are generated and normalized by the game state generator. During the genera-

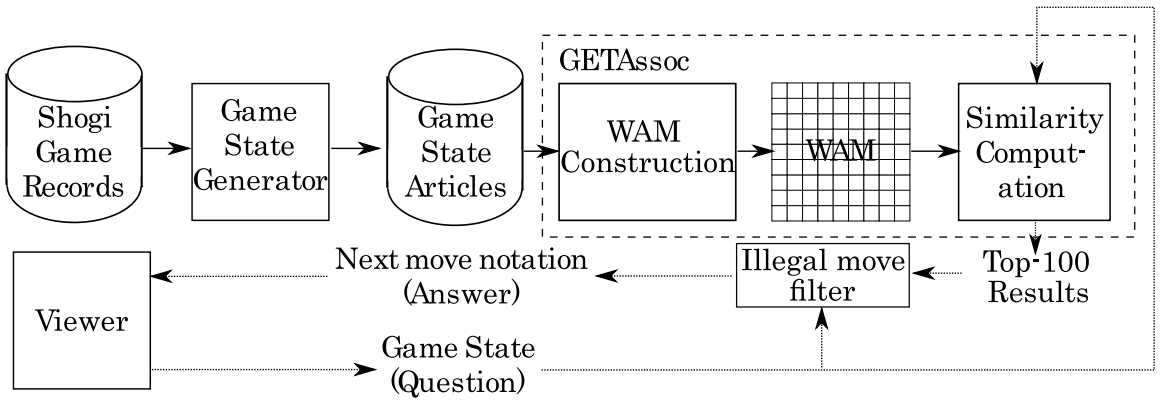



**Fig. 5** Experimental system for next move search.

tion, indexing of each game state is also generated, such that the original record can be referred.

The 49,767 Shogi game records generate 5,613,402 moves, which are 3.1 GB in total. The WAM construction workflow takes about three hours.

### 6.2 Next Move Search Workflow

The next move search workflow is shown in Fig. 5 as the dotted arrows. The game state of the next move problem is queried to GETAssoc. This query is configured by four different similarity measures, Smart measures [8], Okapi BM25 [9], Cosine [10], and Dot Product [10].

Note that resulting next moves may be illegal. Instead of taking just the top-ranked next move, we take the top 100 next moves and filter by the set of legal moves (Filter, in Fig. 5) to find the highest ranked legal next move.

### 6.3 System Configuration

Recall that documents-to-documents associative search consists of two steps; similarity from a query document to characteristic keywords, and similarity from characteristic keywords to associated documents (Section 2.1). By default, GETAssoc implementation computes the top 200 characteristic keywords. Note that, by nature of similarity measures, each characteristic keyword is included in a query document. Since each Shogi game state contains exactly 40 words, we modify the configuration to perform experiments under choices of the number $n$ of characteristic keywords (Section 7.2).

After the most similar game state is found, we refer to the original game record and the number of moves, and the system identifies where that game state appears in a game record and returns its next move, as the most likely one.

## 7. Experiments with Unigrams

We first apply experiments with unigrams to find suitable parameters

- The choice of a similarity measure, and
- The number of characteristic keywords.

and observe their behaviours. For each parameter setting, an experiment on the 1,923 Shogi next move problems takes about 3 hours. We will use only default similarity functions shown in Fig. 1 (Section 2.2).

### 7.1 Similarity Measure Comparison: Prescreening

We first examine the accuracy of next move answer by the following four similarity measures under the default setting, i.e., the number of characteristic keywords = 40. As observed below, Dot product is significantly worse than the others, and we avoid it. Fig. 6 shows examples of searched results.

| Measure | All |
|---|---|
| Smart | 9.41% |
| Okapi BM25 | 9.56% |
| Cosine | **9.93%** |
| Dot product | 6.55% |

### 7.2 Keyword Size Comparison

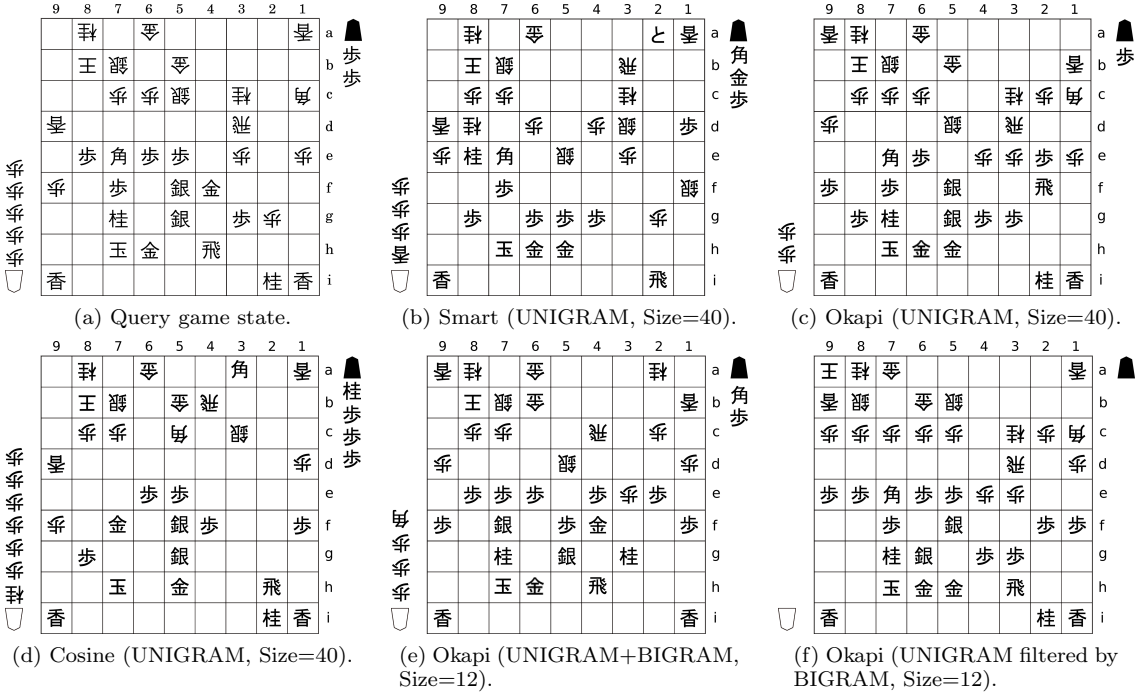
Since each article (game state) contains exactly 40 words, the set of detected characteristic keywords are the same as an article. Thus, to give focus among keywords may improve accuracy of results. We compare the behaviours of three similarity measures, Smart measure, Okapi BM25, and Cosine. Note that, as shown in Fig. 8, when the characteristic keyword size is too small, game states with the same highest scores increase. (For readability, we omit the graph when the number of duplications exceeds three. For instance, when the keyword size is one, the duplications become 23.17 for Okapi and 16.04 for Cosine). Due to too many duplications, we omit the prediction accuracy when the number of characteristic keywords is less than 10.

Fig. 7 shows the changes in accuracy. For more than 10 characteristic keywords, there are peaks between 10-15. Okapi and Smart have another peak around 25-30, and Cosine has another peak at 40. Considering the duplications (Fig. 8), these peaks are quite essential.

Fig. 9 shows the average number of intersections of top ranked game states of similarity measures. It shows that Smart and Cosine show a similar tendency, and Okapi is unique.

### 7.3 Phase Categorization and Its Effect

In Shogi, there are five phases, which are early(E), early-intermediate(EI), intermediate(I), intermediate-late(IL), and late(L). Early(E) usually corresponds



(a) Query game state. (b) Smart (UNIGRAM, Size=40). (c) Okapi (UNIGRAM, Size=40).

(d) Cosine (UNIGRAM, Size=40). (e) Okapi (UNIGRAM+BIGRAM, Size=12). (f) Okapi (UNIGRAM filtered by BIGRAM, Size=12).

**Fig. 6** Searched most similar game states for the query game state.

to Shogi opening patterns. As the game proceeds, the number of possible game states increases, and similar game states to a given game state will diverge. We expect that results would be better in earlier stages. The game phases of 1,923 Shogi next move problems are manually categorized by a medium-level amateur Shogi player (about 1 dan).

| **Phases** | IE | I | IL | L | Total |
|---|---|---|---|---|---|
| **Amount of next move problems** | 105 | 1167 | 331 | 320 | 1923 |

We compare the three similarity measures, Smart, Okapi BM25, and Cosine, for each phase with the sizes 10, 20, 30, and 40 of characteristic keywords (**Size** in Table 2). Table 2 shows the experimental results, in which we can observe the visible improvement in EI and I. Note that although Cosine with keyword size = 10 looks more precise, we need to remember that, with small keyword size, duplication of the results increases (especially for Cosine), which raises the accuracy.

From now, we show experimental results with the phase categorization.

## 8. Experiments with Bigrams

### 8.1 Bigrams

As described in Section 5.2, we perform experiments with bigrams to see their effect. The first experiments are with bigrams only (without unigrams). Note that the number of characteristic keywords (bigrams) is no longer bounded 40; the average size of documents is 43.74. The experimental results are shown in

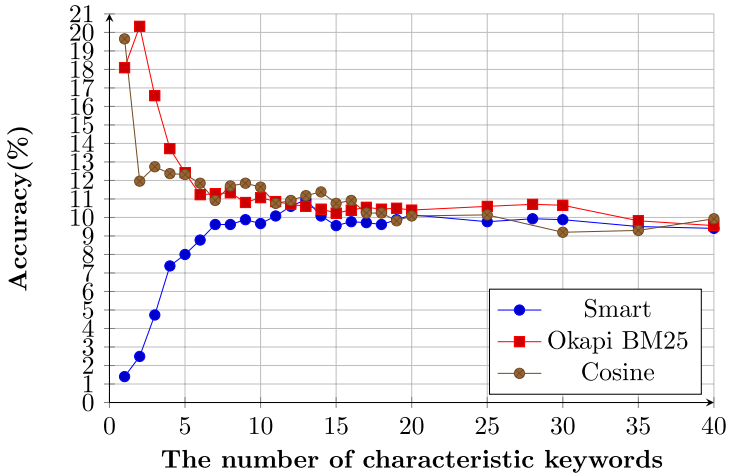



**Fig. 7** The accuracy with keywords size from 1 to 40 (UNIGRAM).

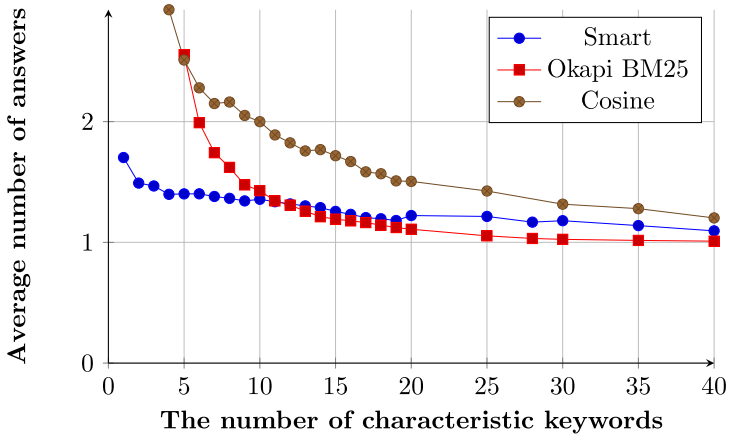



**Fig. 8** The number of game states with the same highest scores (UNIGRAM).

Table 3, which shows visible decreasing with every measure.

As shown in Fig. 10, there is slightly less duplication in top-ranked game states, compared with unigrams; mostly below 1.1 as the average.

For bigrams of semantic couplings, the formation of the group of pieces becomes more influential. We have a general feeling that search with bigrams detects castle patterns well. The key pieces (such as Bishop, Rook, Gold general, Silver general, and King) have more chances to be a member of a group (due to their higher possible moves), compared with many other pieces, such as Pawn. A Pawn does not often appear in bigrams, even though it contains valuable information of a Shogi game state.

We also observe a similar tendency (but less visible) in unigrams at the intersections of top-ranked game states of similarity measures, as shown in Fig. 11. It

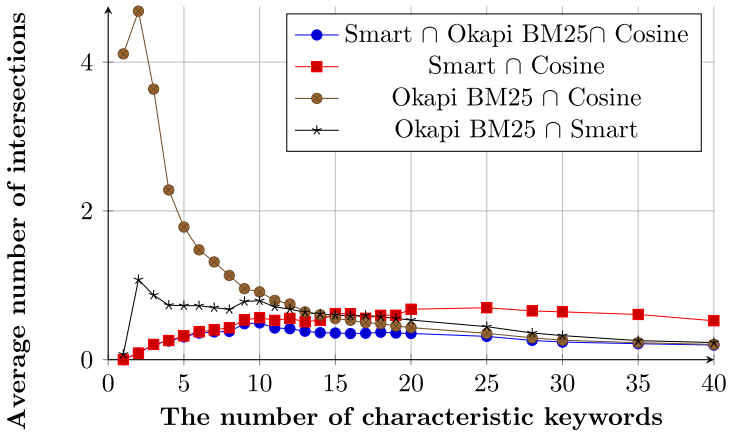



**Fig. 9** The average number of the intersections among similarities (UNIGRAM).

**Table 2** The accuracy in each game phase (UNIGRAM).

| Size | Measure | EI | I | IL | L | All |
|---|---|---|---|---|---|---|
| 10 | Smart | 29.52% | 10.36% | **6.04%** | 7.50% | 9.67% |
| | Okapi | 34.28% | 11.48% | 5.74% | 7.50% | 11.07% |
| | Cosine | **38.09%** | **12.16%** | 5.13% | **7.81%** | **11.64%** |
| 20 | Smart | 29.52% | 10.62% | 5.74% | **6.50%** | 10.14% |
| | Okapi | **32.38%** | 10.62% | **6.34%** | **6.50%** | **10.40%** |
| | Cosine | 29.52% | **10.96%** | 5.43% | 5.31% | 10.08% |
| 30 | Smart | **31.42%** | 10.11% | 5.74% | **6.25%** | 9.88% |
| | Okapi | 27.62% | **11.39%** | **6.94%** | **6.25%** | **10.66%** |
| | Cosine | 29.52% | 9.68% | 5.13% | 5.00% | 9.20% |
| 40 | Smart | 30.47% | 9.68% | 5.14% | **5.94%** | 9.41% |
| | Okapi | 27.62% | 10.03% | **5.74%** | **5.94%** | 9.57% |
| | Cosine | **34.28%** | **10.45%** | 5.13% | 5.00% | **9.93%** |

**Table 3** The accuracy at each game phase (BIGRAM).

| Size | Measure | EI | I | IL | L | All |
|---|---|---|---|---|---|---|
| 10 | Smart | **21.90%** | 10.10% | **6.64%** | 5.31% | 9.35% |
| | Okapi | 20.00% | 10.78% | 4.83% | **5.94%** | 9.45% |
| | Cosine | **21.90%** | **11.30%** | 4.83% | 5.63% | **9.82%** |
| 20 | Smart | **25.71%** | 10.18% | 5.43% | 5.63% | 9.45% |
| | Okapi | 23.80% | **10.53%** | **6.04%** | **5.94%** | **9.72%** |
| | Cosine | **25.71%** | 10.10% | 5.74% | 5.63% | 9.46% |
| 30 | Smart | **26.66%** | 9.41% | **7.25%** | 6.56% | **9.51%** |
| | Okapi | 24.76% | **9.85%** | 5.74% | **7.19%** | **9.51%** |
| | Cosine | 25.71% | 8.90% | 6.05% | 5.93% | 8.84% |
| 40 | Smart | **30.47%** | 9.50% | 6.04% | 5.93% | 9.51% |
| | Okapi | 27.61% | **10.10%** | **6.34%** | **6.56%** | **9.82%** |
| | Cosine | 27.61% | 8.73% | 5.44% | 4.69% | 8.52% |

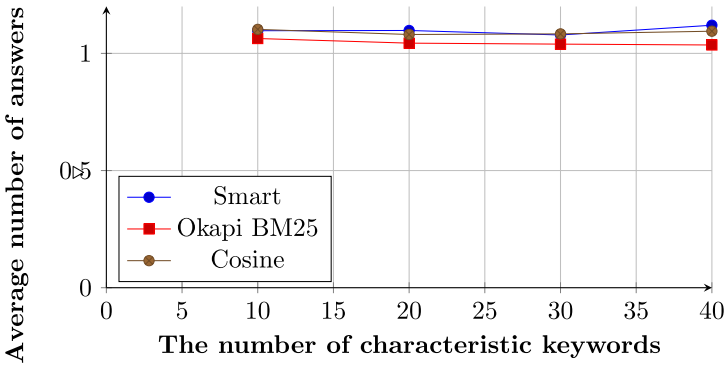



**Fig. 10** The number of game states with the same highest scores (BIGRAM).

again shows that Smart measure and Cosine show a similar tendency, and Okapi is unique, though the differences are smaller.

### 8.2 Combination of Unigrams and Bigrams

There are two possible choices for combinations of unigrams and bigrams; taking the union (UNIGRAM+BIGRAM) of the word spaces, and filtering (intersection of the set of top-$n$ results) one with another. In either case, we set the sizes of characteristic keywords (unigrams) to be either 12 or 27 (described as $n$ in Table 4), as examples of two peaks of the best keyword sizes (Section 7.2). For bigrams, we simply use GETA default setting (characteristic keyword size = 200), since Table 3 shows a general tendency to improve when the keyword

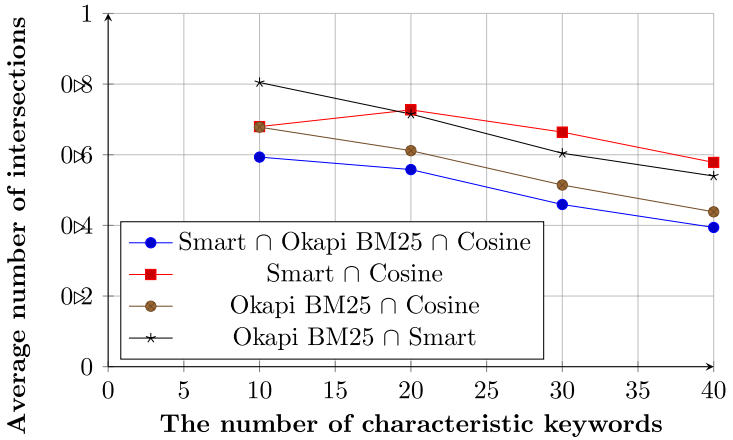



**Fig. 11** The average number of the intersection among similarities (BIGRAM).

**Table 4** The accuracy at each game phase (UNIGRAM+BIGRAM).

| $n$ | Measure | EI | I | IL | L | All |
|---|---|---|---|---|---|---|
| 12 | Smart | **28.57%** | 9.84% | **6.64%** | 5.63% | 9.62% |
| | Okapi | **28.57%** | **10.70%** | **6.64%** | **6.88%** | **10.34%** |
| | Cosine | 27.62% | 10.01% | 5.74% | 5.94% | 9.56% |
| 27 | Smart | **28.57%** | 9.76% | **6.64%** | 6.25% | 9.67% |
| | Okapi | **28.57%** | **10.10%** | **6.64%** | **6.56%** | **9.93%** |
| | Cosine | 27.62% | 9.76% | 6.04% | 5.62% | 9.41% |

size increases.

The search result of the former is shown in Table 4. An example search result is also found in Fig. 6. It does not show visible improvement compared with bigrams only.

For the latter choice, we choose filtering the search results of unigrams by bigrams (UNIGRAM filtered by BIGRAM), based on our observation that bigrams detect castle patterns well. This is implemented by taking the intersection of top-$n$ ranked game states by unigrams and bigrams. The experiments are performed for $n = 100$ with both Smart and Okapi BM25, and $n = 50, 20$ with Okapi BM25, since Okapi BM25 shows visible improvement in the early game phase with $n = 100$.

The experimental results in Table 5 show that filtering unigrams by bigrams

**Table 5** The accuracy in each game phase (UNIGRAM filtered by BIGRAM).

| $n$ | Size | Measure | EI | I | IL | L | All |
|---|---|---|---|---|---|---|---|
| 100 | 12 | Smart | 26.67% | 9.24% | 5.13% | 5.00% | 8.78% |
| | | Okapi | **36.19%** | **10.87%** | **5.14%** | **6.57%** | **10.56%** |
| | 27 | Smart | **29.52%** | 10.18% | 5.74% | 4.69% | 9.56% |
| | | Okapi | 27.61% | **10.53%** | **6.94%** | **6.25%** | **10.14%** |
| 50 | 12 | Okapi | 33.34% | 10.53% | 4.83% | 5.625% | 9.98% |
| 20 | 12 | Okapi | 32.38% | 9.33% | 4.53% | 5.00% | 9.04% |

shows visible improvement with Okapi BM25, and larger $n$ is slightly better. Considering that there are few duplications of game states with the same highest score, this improvement is greater than it seems.

## 9. Conclusion

This paper reported experiments of association search on a large number of short documents over a small set of words. We also showed the extension of the set of words with semantic relations, called semantic coupling. As an instance, experiments were performed on 49,767 professional (non-handicapped) Shogi game records with 1,923 next move problems as an evaluation. To our limited knowledge, this is the first attempt to use GETA over a corpus of an artificial language.

There are several observations.

- Under classification of phases of Shogi game records, an earlier phase has better accuracy. This is quite expected, since the number of possible game states in earlier phase is fewer.
- With unigrams (piece positions) only, the size of the set of words is 2,296 and the size of each document is exactly 40. In this case, there are two peaks of accuracy; the first peak at 10-15 characteristic keywords, and the second peak are at 25-30. There are around 1.2 (or more) duplications on average around these peaks, except for Okapi which has higher.
- With bigrams (semantic coupling by possible moves of pieces) only, the size of the set of words is nearly 20,000, and the average size of documents is 43.74. There is visible reduced accuracy with every measure, but bigrams detect castle patterns well. There are fewer than 1.1 duplications on average.
- As combinations of unigrams and bigrams, filtering search results with unigrams by bigrams shows the best improvement in the early game phase with Okapi BM25.
- Smart and Cosine show a similar tendency of search results, whereas Okapi BM25 is unique.

The experimental results are quite disappointing for next move search of Shogi. However, in the context of Computer Shogi, an associative search is a naive methodology; for instance, machine-learning-based methods use several thousands of parameters, whereas an association search uses only one similarity measure, which fits a likely search rather than an exact search.

From the Computer Shogi viewpoint, we expect associative search will have better fitness for categorizing Shogi game records into, say, their game phases and/or their game opening patterns. These applications are left for future work.

**Acknowledgments** We thank Akihiko Takano (NII) and Susumu Kunifuji (JAIST) for fruitful suggestions. We also thank Masahiro Yasugi (Kyoto University) and Atsushi Maeda (Tsukuba University) for commenting on ideas regarding bigrams during our oral presentation at the IPSJ SIGPRO meeting. Last but not least, we thank the anonymous reviewer for careful reading.

## References

1) Watanabe, I.: Divergent thinking support system: keyword associator, *Proc. SICE Joint Symposium*, pp.411–418 (1991) (in Japanese).
2) Takano, A.: Association Computation for Information Access, *Discovery Science*, pp.33–44, Springer (2003).
3) Kohda, Y., Watanabe, I., Misue, K., Hiraiwa, S. and Masui, M.: Group Idea Processing System: GrIPS, *Transactions of the Japanese Society for Artificial Intelligence*, Vol.8, pp.601–601 (1993).
4) Takano, A., Niwa, Y., Nishioka, S., Iwayama, M., Hisamitsu, T., Imaichi, O. and Sakurai, H.: Associative information access using dualnavi, *International Conference on Digital Libraries: Research and Practice, 2000 Kyoto*, pp.192–196, IEEE (2002).
5) Kando N. and Adachi, Jun.: Cultural Heritage Online: Information Access across Heterogeneous Cultural Heritage in Japan, *Electronic Proceedings of International Symposium on Digital Libraries and Knowledge Communities in Networked Information Society (DLKC 2004)* (2004).
6) Iida, H., Sakuta, M. and Rollason, J.: Computer Shogi, *Artificial Intelligence*,

Vol.134, No.1-2, pp.121–144 (2002).
7) Dershowitz, N. and Manna, Z.: Proving termination with multiset orderings, *Comm. ACM*, Vol.22, No.8, pp.465–476 (1979).
8) Singhal, A., Buckley, C., Mitra, M. and ArMitra.: Pivoted document length normalization, pp.21–29, ACM Press (1996).
9) Robertson, S.E. and Walker, S.: Okapi/keenbow at trec-8, *NIST Special Publication SP*, pp.151–162 (2000).
10) Wilkinson, R., Zobel, J. and Sacks-David, R.: Similarity measures for short queries, *NIST Special Publication SP*, pp.277–286 (1996).

(Received December 21, 2010)
(Accepted March 29, 2011)

**Kobkrit Viriyayudhakorn** is a doctoral candidate at Japan Advanced Institute of Science and Technology. His research interest is computational creativity such as creativity support systems and creative autonomy.

**Mizuhito Ogawa** received his M.S. in 1985 and Ph.D. degree in 2002, both from the University of Tokyo. He worked in NTT Laboratories from 1985 until 2001, and in JST from 2002 to 2003. From 2003, he has been working at Japan Advanced Institute of Science and Technology. His research interest includes formal methods, from its theoretical basis such as theory in rewriting, formal language, and combinatorics, to its implementation, such as scalable program analyses and theorem provers.