Faster and Broader Associative Search for Supporting Human's Idea Association in Divergent Thinking Support System

Kobkrit Viriyayudhakorn[1,a)] Susumu Kunifuji[1,b)]

Abstract: Idea Association is a human thought process that leads to a new idea from an original idea that associated by some principles. It was be supported by association search, which is an information retrieval that dynamically computes similarity between a query and all indexed documents, converge to highly related information, and present them to users, as a module of divergent thinking support system. Supporting divergent thinking does not require such highly specific information; Association search, which is very high computational intensive, can be optimized for faster computation, which increase system's responsiveness without dropping its effectiveness in supporting human's idea association. In this paper, we proposed a faster association search for divergent thinking support system. Evaluations for comparing its effectiveness with the original association search are performed. The evaluations, results, and discussions are described in detail.

Keywords: Associative Information Search Engine, Idea Processor, Creativity Support System, Divergent Thinking Support System

1. Introduction

2. GETAssoc

*G*eneric *E*ngine for *T*ranspose *Asso*ciation (GETAssoc) is words-to-documents and words-to-words associative search engine developed at National Institute of Informatics (NII)[1]. GETAssoc provides associative search capability on a textual database. It can accurately retrieve a set of relevant documents and a set of associative words of an input query sentence.

Due to its scalability, GETAssoc can handles a dynamic association search for about twenty million documents with in few seconds[2]. GETAssoc was used or applied in many practical Web search engine applications, such as Webcat Plus [*1], Imagine [2] [*2], Cultural Heritage Online [3] [*3], and Pictopic [*4].

2.1 Dependencies

To perform associative searches, GETAssoc requires two key dependencies as follows.

2.1.1 Word Article Matrix (WAM)

Word Article Matrix (WAM) is a huge sparse matrix storing each term frequency occurred in each document found in the textual database. Given a textual database has N unique terms (types) written in M documents, the size of WAM is $M \times N$. The rows of WAM are indexed by documents and the columns of WAM are indexed by unique terms. The $(i, j)^{\text{th}}$ entry of WAM is the number of occurrence of a term j found in a document i. GETAssoc indexes the textual database by constructing WAM. When WAM is completely constructed, they are horizontally and vertically compressed into two growable hash tables for saving disk storage and faster computation[1]. The hash table that yielded by horizontally compressed WAM maps a document to the list of terms that found in that document. The hash table that yielded by vertically compressed WAM maps a term to the list of documents that contain that term. For explanation in the future step of this paper, both hash tables are called as the "horizontal hash table" and the "vertical hash table" respectively. With these two hash tables, GETAssoc is ready for associative searches.

2.1.2 Similarity functions

Given D is a set of documents, and T is a set of terms found in the textual database. GETAssoc provides predefined similarity functions as follows.

(1) Smart measures [4]
(2) Okapi BM25 [5]
(3) Cosine [6]

[1] School of Knowledge Science, Japan Advanced Institute of Science and Technology
[a)] kobkrit@jaist.ac.jp
[b)] kuni@jaist.ac.jp

[*1] `http://webcatplus.nii.ac.jp/`
[*2] `http://imagine.bookmap.info/`
[*3] `http://bunka.nii.ac.jp`
[*4] `http://photobank.pictopic.info/`

(4) Dot Product [6]

They are defined as functions of type $D \times MS(T) \rightarrow \mathbf{R}_{\geq 0}$ in Fig. 1, where $MS(X)$ is a multiset of X which is a set that members of set X are allowed to appeared more than once, document $d \in D$, query $q \in MS(T)$, term $t \in T$, and

- $w_{q,t} = log(\frac{N}{f_t+1})$
- $w_{d,t} = log(f_{d,t} + 1)$
- f_t is the total number of documents that contain t.
- f_d is the total number of terms in document d.
- $f_{x,t}$ is the total number of the occurrence of term t in x.
- N is the total number of documents in D.

GETAssoc sets $\theta = 0.2, k = 0.2$, and $b = 0.75$ as default.

2.2 Associative Searches Algorithm

The followings are procedures in associative searches of GETAssoc.

Step-1: Counting term frequency in query Given query $q \in MP(T)$ with length e, it is converted to be the set of ordered pairs $\{(x, \text{count}(x)) : x \in T\}$ with length i where $\text{count}(x)$ is a function of type $T \rightarrow \mathbf{N}$ that returns number of occurrence of unique term x in q or $f_{q,x}$ used in computing similarity functions. The time complexity of this step is $O(e)$ for both average and worst cases.

Step-2: Ranking the most significant unique terms in query Given a constant o as the number of most significant unique terms that characterize (summarize) of query q. $\text{sim1}(q, x)$ is a similarity function that returns degree of significant of term x in query q (treats the query q as a document and x as a one-word query). The simplest form of $\text{sim1}(q, x)$ is $f_{q,x}$. In GETAssoc, $\text{sim1}(q, x)$ can be customized as a preference. Smart similarity function in Equation 1 is chosen by default. The top-o unique terms in query q ranked with respect to the $\mathrm{sim1(q, x)}$ similarity function are selected as the set G which is the summary of query q.

We treat running time of $\mathrm{sim1(q, x)}$ to constant c, the running time for computing $\mathrm{sim1(q, x)}$ for every unique term x in q is $c \cdot i$. Searching a max value of $\mathrm{sim1(q, x)}$ for x in q takes running time e and it searches for o times, the time complexity of this step is $O(c \cdot i + o \cdot e)$ for both average and worst cases.

Step-3: Extracting the list of documents that contain at least one term in the summary of query We obtained the set G is the summary of query q, $o =| G |$ from the previous step. For each $g \in G$, it is used as a key for obtaining documents that contain term g at least one time from the vertical hash table defined in Section 2.1.1. The set of unique documents H that contain at least one term in G are extracted by union all found documents of G.

Searching an element in a growable hash table takes about $O(1 + \frac{N}{k})$ in the average case and $O(N)$ in the worst case where k is a number of hash table's buckets. Since the hash table is growable, the value of k is depended on various parameter such as the amount of available memory, N, and so on. Thus k is left as a constant.

Since found documents of each term g are unionized, the average number of found unique documents in each g is unpredictable, we leave it as constant b in the average case and M in the worst case. We repeatedly searching an element for o times, therefore the time complexity of this step is $O(o(b + 1 + \frac{N}{k}))$ and $O(o(M + N))$ for the average case and the worst case respectively.

Step-4: Associative computation The set G, which is a summary of query q ($o =| G |$), and the set H, which is a set of documents that contain at least one term in G are given. For each $h \in H$, $\text{sim2}(h, G)$, which is a similarity function that returns degree of similarity between document h and the set G (treats G as a query) are computed. In GETAssoc, $\text{sim2}(h, G)$ can be customized as a preference. Smart similarity function in Equation 1 is chosen by default.

Given constant c time for computing $\text{sim2}(h, G)$, the time complexity of this step is $O(o^2 \cdot b \cdot c)$ and $O(o \cdot M \cdot c)$ for the average case and the worst case respectively.

Step-5: Extracting the most associative documents The top-p documents in H ranked with respect to the $\mathrm{sim2(h, G)}$ similarity function for $h \in H$ are selected as the most associative documents to the query q. These top-p most associative documents are the document result set R with length p. The result set R are search results of words-to-documents associative searches of GETAssoc. Note that, for some circumstance, the top-y most associative documents where $y < p, y \in \mathbf{N}$ are removed from the search results since it is too close to the query q.

For searching a max value of $\mathrm{sim2(h, G)}$ for $h \in H$ takes running time $o \cdot b$ and M for the average case and the worst case respectively and it searches for p times, the time complexity of this step is $O(o \cdot b \cdot p)$ for the average case and $O(M \cdot p)$ for the worst cases.

Step-6: Extracting the associative keywords For each document in the top-s most associative document in the document result set R where $s < p, s \in \mathbf{N}, p =| R |$, it is used as a key for retrieving the set of terms that written in that document from the horizontal hash table defined in Section 2.1.1. The set of found unique terms are unionized as the set of most associative keywords V.

Similar to Step-3, searching an element in a growable hash table takes about $O(1+\frac{M}{k})$ in the average case and $O(M)$ in the worst case where k is a number of hash table's buckets. Since all found unique terms of each document r in the top-s of the result set R are unionized as the set of associative keyword Y, the average number of found unique terms in each r is unpredictable, we leave it as constant a in the average case and N in the worst case. We repeatedly searching an element for s times, therefore the time complexity of this step is $O(s(a + 1 + \frac{M}{k}))$ and $O(s(N + M))$ for the average case and the worst case respectively.

Step-7: Ranking the most significant associative keywords Given the set of associative keyword V and a constant l as the size of most significant associative keywords to the query q. $\text{sim1}(V, x)$ for $x \in V$ is a similarity function that returns degree of significant of a keyword x in query

Smart [4] :

$$\text{smart}(d,q) = \frac{1}{avg(f_d) + \theta(f_d - avg(f_d))} \sum_{t \in q \cap d} log(\frac{N}{f_t}) \cdot \frac{1 + log(f_{d,t})}{1 + log(avg_{\Theta \in d}(f_{d,\Theta}))} \cdot \frac{1 + log(f_{q,t})}{1 + log(avg_{\Theta \in q}(f_{q,\Theta}))} \tag{1}$$

Okapi BM25 [5] :

$$\text{okapibm25}(d,q) = \sum_{t \in q \cap d} log(\frac{N - f_t + 0.5}{f_t + 0.5}) \cdot \frac{f_{d,t} \cdot (k+1)}{f_{d,t} + k \cdot (1 - b + b \cdot \frac{f_d}{avg(f_d)})} \tag{2}$$

Cosine similarity [6] :

$$\text{cosine}(d,q) = \frac{\sum_{t \in q \cap d}(w_{q,t} \cdot w_{d,t})}{\sqrt{\sum_{t \in q}(w_{q,t}^2) \cdot \sum_{t \in d}(w_{d,t}^2)}} \tag{3}$$

Dot product [6] :

$$\text{dotproduct}(d,q) = \sum_{t \in q \cap d} (w_{q,t} \cdot w_{d,t}) \tag{4}$$

Fig. 1 Four similarity measures and their equations

V (treats the query V as a document and x as a one-word query). Similar to the Step-2, sim1(V, x) can be customized as a preference. Smart similarity function in Equation 1 is chosen by default. The top-l unique keywords in the set of associative keyword V ranked with respect to the sim1(V, x) similarity function are selected as the set of most associative keywords, which is result sets of words-to-words associative searches of GETAssoc.

We treat running time of sim1(V, x) to constant c, the running time for computing sim1(V, x) for every associative keyword x in V is $c \cdot s \cdot a$ and is $c \cdot s \cdot N$ for the average case and the worst case respectively. Searching a max value of sim1(V, x) for x in V takes running time $s \cdot a$ and N for the average and the worst case respectively, and it searches for l times, the time complexity of this step is $O((c+l)(s \cdot a))$ and $O((c \cdot s + l)N))$ for the average case and the worst cases respectively.

2.3 Total Time Complexity

The total time complexities of words-to-words associative searches by GETAssoc is $O(e + c \cdot i + o \cdot e) + o(b + 1 + \frac{N}{k}) + o^2 \cdot b \cdot c) + o \cdot b \cdot p + s(a + 1 + \frac{M}{k}) + (c+l)(s \cdot a)) = O(N + M)$ for the average case, and $O(e + c \cdot i + o \cdot e) + o(M + N) + o \cdot M \cdot c + M \cdot p + s(N + M) + (c \cdot s + l)N) = O(N + M)$ for the worst case.

3. Faster and Boarder Associative Search for Supporting Divergent Thinking

GETAssoc can obtain associative information with very high accuracy and fast computation. But for some application, such as in the divergent thinking support system, it does not requires such deep accuracy, instead it requires broader scope of associative information for helping idea association process of human thinking. The proposed method is intensively based on the GETAssoc.

3.1 Dependencies

The hash table Z mapped from each unique term $t \in T$ to the list of associative keywords of term t and their similarity score is the main dependency of this method. The hash table Z is a words-to-words associative information mapping. To construct this hash table, the following requirements are needed.

(1) List of all unique terms $t \in T, N = | T |$ that found in the textual database.

(2) GETAssoc associative search engine that already indexed the textual database. Both horizontal and vertical hash tables are constructed, see Section 2.1.1.

For each unique term $t \in T$, it is sequentially queried into the GETAssoc search engine for obtaining a list of two-tuple of top-l most associative keywords of term t and their similarity scores. Each unique term $t \in T$ and the list of two-tuples are inserted as a key and value pair into the hash table Z.

3.2 Associative Searches Algorithm

The followings are procedures in associative searches of the proposed method.

Step-1: Extracting all unique terms in the input query Give query $q \in MP(T)$ with length e, it is converted to be the set of unique terms U with length i found in q. The time complexity of this step is $O(e)$ for both average and worst cases.

Step-2: Extracting the associative keywords For each unique terms $u \in U$ with length i, the list with length j of the most associative keywords and its similarity score pairs are retrieved from the hash table Z.

For each pair of each unique terms u, it is inserted into another hash table called V which maps from an associative keyword to its similarity score. If an associative keyword of a pair have not yet been indexed in V, the pair is inserted into hash table V. If an associative keyword of a pair have been indexed in V, its similarity score is summed with the total similarity score of that associative keyword stored in the hash table V.

The time complexity for searching an element from hash table is $O(1 + \frac{N}{k})$ and $O(N)$ for average and worst cases respectively. The time complexity for inserting an element into hash table is $O(1)$ and $O(N)$ for average and worst cases. Since searching an element repeats for i times and inserting an element repeats for $i \cdot j$ times, the time complexity in this step is $O(i(1 + \frac{N}{k} + j))$ and $O((i \cdot N)(1 + j))$ for the average case and the worst case where k is the bucket size of hash table Z.

Step-3: Ranking the associative keywords The top-l most associative keyword are selected from the hash table V with length o where $o \in \mathbf{N}$ constructed from previous step. The top-l most associative keyword are search results of a words-to-words associative search by using the proposed method.

Searching the maximum value in hash table V takes running time o and N for the average and the worst case respectively, and it searches for l times, the time complexity of this step is $O(l \cdot o)$ and $O(l \cdot N)$ for the average case and the worst cases respectively.

3.3 Total Time Complexity

The total time complexity of words-to-words associative searches by the proposed method is $O(e+i(1+\frac{N}{k}+j)+l\cdot o) = O(N)$, and $O(e+(i\cdot N)(1+j)+l\cdot N) = O(N)$ for the average case and the worst case respectively.

Since normally $M >> N$, the time complexity of the proposed algorithm for an associative search is believed to be much lesser than GETAssoc because the dynamical associative computation in GETAssoc is pre-computed and prepared before a search operation is performed. The speed of two algorithms will be experimented and compared in Section 4 for proven the hypothesis.

3.4 Differences from GETAssoc

The search results of words-to-words associative search from the proposed method and from GETAssoc are quite different. Associative keywords yielded from GETAssoc are retrieved from the intersection of the sets of documents that related with each term in query q. To be selected as the output of GETAssoc, the associative keywords must be highly related with all terms found in query q as much as possible.

It is different to associative keywords yields from the proposed method, which retrieved from the union of the set of documents that related with each term in query q. The associative keywords of the proposed method have just only highly related to a term found in query q as a requirement to be selected as a output.

Figure 2 visualizes the difference of associative keywords result in both algorithms when querying "Apple computer". Figure 2 splits into two space, the document space is located at the left hand side, while the terms space is located at the right hand side. Documents in red background color and keywords in red color is the associative documents and associative keywords yielded by GETAssoc. Documents in green background color and keywords in green color is the associative documents and associative keywords yielded by the proposed method.

The associative keywords yielded by the proposed method are broader than GETAssoc's keywords, which believed to be more suitable for supporting the human's idea association process for divergent thinking tasks. The output of proposed method should yield better or equal efficiency to GETAssoc's output on supporting human thinkings. The efficiency of two algorithms will be experimented and evaluated in Section 5 for proven the hypothesis.

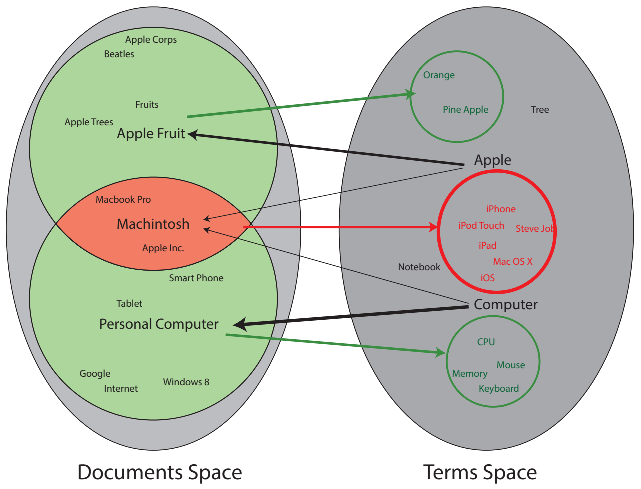


Fig. 2 Different associative keywords yielded from GETAssoc (red color) and the proposed method (green color) when the query is "Apple computer."

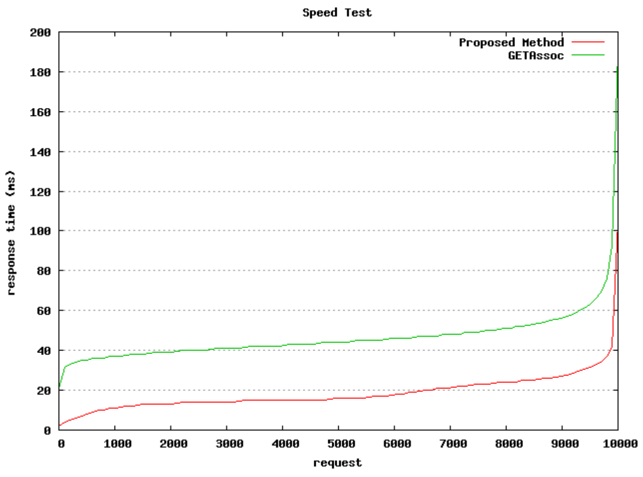


Fig. 3 Speed in response of GETAssoc and the proposed method

4. Speed Experiment

We setup a web server receiving request for associative searches and respond back to users. The web server's settings for both GETAssoc and the proposed method are identical.

4.1 Experimental Settings

We uses apache benchmarking tool (ab) generating 10,000 queries with 10 parallel connection to the web server. Both GETAssoc and the proposed method are tested.

4.2 Experimental Results

Figure 3 shows the speed in benchmark test of both algorithms. The proposed algorithm is approximate twice faster than GETAssoc.

5. Divergent Thinking Experiment

To test the efficiency of two associative searches algorithms in supporting human's idea association process, we setup a Web application for collecting ideas from participants on four creative questions, while showing associative keywords yielded from an algorithm. An algorithm that can generate associative keywords that can significantly increase number of ideas that can be thought up by participants because of seeing these keywords are helpful in supporting idea association process.

5.1 Experimental Settings

5.1.1 Web Interface

The single-user Web application's interface is shown in Figure 4, the header of the page shows a creative question asked to a participant and the countdown timer. At the center panel of the screen, yellow rectangle labels that contains answers inputted by users are shown. The two types of suggestions which are associative keywords are separately shown on the screen as follows.

(1) **Overall Suggestions** are orange associative keywords located at the bottom of the screen. Each time when a user input a new idea, a creative question text, its description text, and the content of all idea labels are concatenated as a query submitted to an associative search engine. Overall suggestions are updated according to the output from such search.

(2) **Individual Label Suggestions** are blue associative keywords located at the above of each idea label. It is generated by using the content of that idea label only as a query submitted to an associative search engine. Suggestion of each labels are queried and shown when the mouse pointer of a participant places on that label.

Participants are told to read associative keywords in both types of suggestions and in the meanwhile think about the creative question. If participants can think up a new idea because associative keywords help in triggering it, participants click on that these associative keywords one-by-one, it will be added as an item in the “inspire by” item-list located at the bottom of the screen. Then, participants type down their ideas into the “content” text box located above the “inspire by” item-list, and press enter. A new idea labels containing the input content are generated and add into the center panel of the screen.

If participants click on an associative keyword by mistake, it can be removed by clicking a “X” symbol next to it. If participants can think up a new idea by solely using his/her brain, the “inspire by” item-list is left blank. With the above rules, we can separate the number of ideas that participants can be think up supported by an associative search engine from all input ideas.

For a creative question, a participant have 15 minutes to think up new ideas and answer it as much as possible.

5.1.2 Associative Search Engines

The experiment conducts by using three associative search engine as follows.

(1) **GETAssoc** (See Section 2)

(2) **Proposed method** (See Section 3)

(3) **Random** construct a suggestion list by randomly picking a term from all N terms appeared in a textual database.

(4) **Empty** show no suggestion.

For GETAssoc and the proposed method (while in both constructing the hash table V and performing associative searches) use the following configurations.

(1) Dot product functions are used in $sim1$ and $sim2$ similarity functions (see Section 2.1.2) since its associative keywords are most readable compare to other three.

(2) $o = 70$.

(3) $p = 0$.

(4) $s = 10$.

(5) $l = 15$ for the individual label suggestions and $l = 30$ for the overall suggestions.

5.1.3 A textual database

Dump file of all English Wikipedia documents on January 2, 2013[*5] is used as a textual database that be used in the experiment. It was indexed by all three algorithms. It has following characteristic as follows.

(1) $M = 3,445,076$ documents.

(2) $N = 4,234,985$ unique words (types).

Note that the number of unique words is unnaturally higher than expect, since Wikipedia contain a lot of name entities, and all words were intentionally not be converted into lowercase for helping participants to easily distinguish between words and name entities. However, all nouns and verbs are lemmatized back to its root form by using NLTK python library[*6] before being indexed by three testing algorithms.

5.1.4 Creative Questions

The following creative questions based on Guildford's alternative use of task [7] which is a popularly divergent thinking test are asked during the experiments.

(1) List the unusual use of eraser.

(2) List the unusual use of spoon.

(3) List the unusual use of CD-ROM.

(4) List the unusual use of wheel.

(5) List the unusual use of cup.

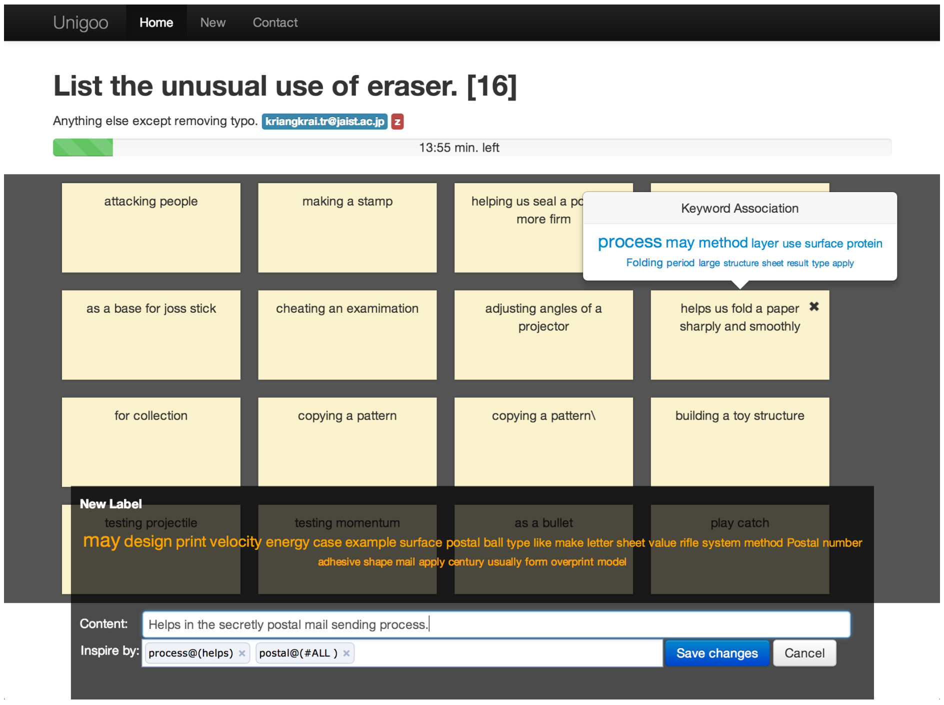
(6) List the unusual use of book.

Since the number of ideas are affected by the difficulty of creativity questions, We selected four out of six questions above that have the smallest gap in their difficulty. We have performed the preliminary experiment by asking 15 individuals who are not participants of the divergent thinking experiment to answer a question as much as possible in three minutes each (eighteen minutes in total). The number of ideas answered for these six questions are shown in Table 1. Since the four first questions have the smallest gap on the average number of ideas (1.267) compared with the other combinations. They are used as the creative questions in the divergent thinking experiment.

5.1.5 Participants

Participants are various nationality 32 students of the Japan Advanced Institute of Science and Technology. They are either Master degree or Doctoral degree students stud-

[*5] `http://dumps.wikimedia.org/enwiki/20130102/`
[*6] http://nltk.org/



List the unusual use of eraser. [16]

Anything else except removing typo. kriangkrai.tr@jaist.ac.jp z

13:55 min. left

attacking people

making a stamp

helping us seal a pc more firm

Keyword Association

process may method layer use surface protein Folding period large structure sheet result type apply

as a base for joss stick

cheating an examimation

adjusting angles of a projector

helps us fold a paper sharply and smoothly

for collection

copying a pattern

copying a pattern\

building a toy structure

New Label

testing projectile

testing momentum

as a bullet

play catch

may design print velocity energy case example surface postal ball type like make letter sheet value rifle system method Postal number adhesive shape mail apply century usually form overprint model

Content: Helps in the secretly postal mail sending process.

Inspire by: process@(helps) × postal@(#ALL) ×

Fig. 4 Web application used for collecting ideas and showing associative keyword in the experiment

Table 1 Preliminary Experiment on the Difficulty of Creative Questions

Question	Avg. Number of Ideas
Eraser	3.200 ± 1.781
Spoon	3.800 ± 2.484
CD-ROM	3.867 ± 1.727
Wheel	4.467 ± 1.959
Cup	5.134 ± 3.137
Book	5.334 ± 2.526

Table 3 Descriptive Statistic of the number of ideas that participants can be thought up by seeing suggestions

Algorithm	**Avg. number of ideas**
Proposed Method	4.4375 ± 2.9065
GETAssoc	4.2813 ± 3.6477
Random	1.5938 ± 2.6744
Empty	0 ± 0

ied in the School of Knowledge Science, the School of Information Science, and the School of Material Science. All participants have good English language proficiency, which is TOEIC score > 785 or TOEFL paper-based score > 590.

To avoid the effects of tool experiences, The 32 students are split into eight teams (four students each). Four questions and four associative search engine were assigned to four students in a team in different orders as shown in Table 2.

5.2 Experimental Results

Table 3 and Figure 5 show the average number of ideas that participants can be thought up by seeing suggestions using four different associative search engines. The average number of thought up idea supported by showing associative keywords are subjected to a two-way between-subject ANOVA on the four creative questions and the four associative search algorithms, since the average number is highly influenced by these two factors.

The influence of four creative question are not statistically significant on the number of ideas thought by seeing suggestions ($F(3, 121) = 1.806, p = 0.150, \eta^2 = 0.029$, n.s.$, p > 0.05$). The influence of four different associative search engines who generates the associative keywords are staticaslly significant on the number of ideas thought by seeing suggestions at 0.001 significant level ($F(3, 121) = 21.029, p = 0.000, \eta^2 = 0.333$).

A post hoc analysis using the Tukey test indicated on the four different associative search engines show the statistically significant on the difference among four different algorithm as follows.

- The proposed method and GETAssoc have no significant difference to each other. ($p = 0.995$, n.s., $p > 0.05$).
- The proposed method and GETAssoc both have significant difference over Random algorithm (both $p = 0.000$) at 0.001 significant level.
- The proposed method and GETAssoc both have significant difference over Empty algorithm (both $p = 0.000$) at 0.001 significant level.
- Random and Empty algorithms have no significant difference to each other. ($p = 0.083$, n.s., $p > 0.05$).

6. Conclusion

The proposed associative search, which twice faster than the original associative search algorithm (GETAssoc) used in supporting divergent thinking, seems better than GETAssoc but it has no statistically significant difference to GETAssoc in efficiency of supporting human's idea association process. The proposed method, which provides broader associative keywords than GETAssoc, seems more suitable

Table 2 Algorithms and questions assignment.

Participant	Question 1	Question 2	Question 3	Question 4
1	Proposed Method	GETA	Random	Empty
2	GETA	Random	Empty	Proposed Method
3	Random	Empty	Proposed Method	GETA
4	Empty	Proposed Method	GETA	Random

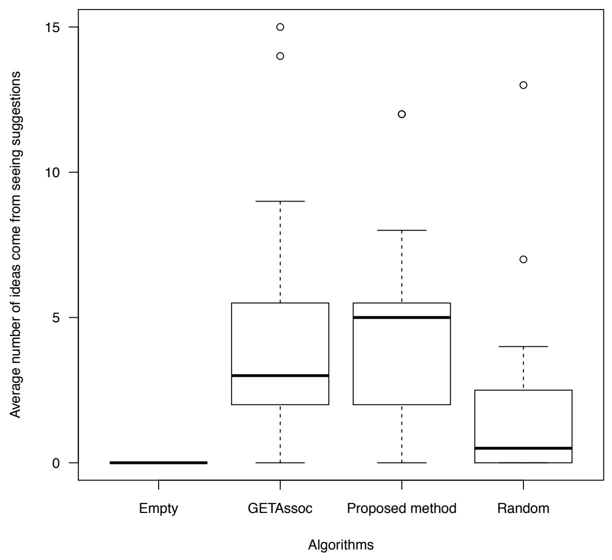


Fig. 5 Average number of ideas that participants can be thought up by seeing suggestions

for divergent thinking application. Based on the number of idea labels that can thought up by looking on suggestion keywords, associative information yielded must higher performance in support idea's association process than the random information.

Acknowledgments We thank to Pakorn Techaveerapong and Pakakorn Sitthisak for fruitful suggestions on analyzing GETAssoc and the proposed algorithm. We also thank to Siriwon Taewitjit for commenting on the experimental design. Last but not least, we thank the anonymous reviewers for careful reading.

References

[1] A. Takano. Association computation for information access. In *Discovery Science*, pages 33–44. Springer, 2003.
[2] Masayuki Nakao, Kensuke Tsuchiya, Yoshiaki Harita, Kenji Iino, Hiroshi Kinukawa, Satoshi Kawagoe, Yuji Koike, and Akihiko Takano. Extracting failure knowledge with associative search. *New Frontiers in Artificial Intelligence*, pages 269–276, 2008.
[3] Noriko Kando and Jun Adachi. Cultural Heritage Online: Information Access across Heterogeneous Cultural Heritage in Japan. In *Electronic Proceedings of International Symposium on Digital Libraries and Knowledge Communities in Networked Information Society (DLKC 2004)*, 2004.
[4] Amit Singhal, Chris Buckley, Mandar Mitra, and Ar Mitra. Pivoted document length normalization. pages 21–29. ACM Press, 1996.
[5] S.E. Robertson and S. Walker. Okapi/keenbow at trec-8. *NIST Special Publication SP*, pages 151–162, 2000.
[6] Ross Wilkinson, Justin Zobel, and Ron Sacks-David. Similarity measures for short queries. *NIST Special Publication SP*, pages 277–286, 1996.
[7] Joy Paul Guilford. The nature of human intelligence. 1967.

Kobkrit Viriyayudhakorn was born in 1986. He received B.Sc. and M.S. degrees from the Sirindhorn International Institute of Science and Technology, Thammasat University in 2008, and 2010 respectively and Ph.D. degree from the Japan Advanced Institute of Science and Technology in 2013. His research interest includes Creativity support system, Information retrieval, Natural language processing, Data mining, and Machine learning.

Susumu Kunifuji was born in 1947. He recieved B.E., M.E., and D.E. degrees from Tokyo Institue of Technology in 1871, 1974, and 1994, respectively. He worked as a researcher at the International Institute for Advanced Study of Social Information Science, FUJITSU Ltd.(1974-1982), Chief researcher at the Institute for New Generation Computer Technology(1982-1986), Manager of the International Institute for Advanced Study of Social Information Science, FUJITSU Ltd.(1986-1992), Professor of School of Information Science at JAIST(1992-1998). He is currently Professor of School of Knowledge Science(1998-) and Vice President(2011-) at JAIST. He is a member of IPSJ, IEICE, JSAI, and JCS among others.