Towards Constructing Story Databases Using Maximal Analogies Between Stories

Masaharu Yoshioka, Makoto Haraguchi, and Akihito Mizoe

Graduate School of Information Science and Technology
Hokkaido University
N-14 W-9, Sapporo 060-0814, Japan
{yoshioka,makoto,amring}@db-ei.eng.hokudai.ac.jp

Abstract. In order to construct story databases, it is crucial to have an effective index that represents the plot and event sequences in a document. For this purpose, we have already proposed a method using the concept of maximal analogy to represent a generalized event sequence of documents with a maximal set of events. However, it is expensive to calculate a maximal analogy from documents with a large number of sentences. Therefore, in this paper, we propose an efficient algorithm to generate a maximal analogy, based on graph theory, and we confirm its effectiveness experimentally. We also discuss how to use a maximal analogy as an index for a story database, and outline our future plans.

1 Introduction

Since many documents can now be accessed through the Internet, various methodologies for retrieving, organizing and accessing documents have been developed. Information retrieval and document classification are examples of such techniques. As the number of documents to be processed is generally very large, most of these systems use a "bag of words" approach, and use an index based on words in each document. A "bag of words" approach works efficiently, but cannot discriminate between two or more documents that have same word sets in different order. For instance, any "bag of words" indexing system does not distinguish between "a dog bit a man" and "a man bit a dog", despite their different meanings. Thus, there is a need for an extended indexing system that can handle differences such as this, and is not based on index terms alone.

In addition, most of standard document databases handle document by using indices that represent facts in the document (e.g. existence of terms and sentences). Because of this, these systems cannot discriminate between two or more documents that have same sentences in different order (plot). We call a document database that can retrieve documents by using a plot as a story database.

In order to construct a story database, it is crucial to have an effective index that represents the plot and event sequences in a document. Since a single document can have various aspects, we need to add multiple indices even for a single document. Text summarization is one of the techniques to extract a plot from

G. Grieser and Y. Tanaka (Eds.): Intuitive Human Interface 2004, LNAI 3359, pp. 243–255, 2004.

a story [1]. However, most of these techniques mainly focus on extracting main plots from a story and are not sufficient to create indices for a story database. Research on topic focused summarization [2] aims to construct different summaries from a single document, but it requires pre-defined topic sets that are also difficult to extract automatically.

To address this problem, we have already proposed the concept of maximal analogy (MA) between stories, which represents a generalized event sequence of documents with a maximal set of events [3]. We have also proposed a method for constructing this index from a set of provided documents. However, the computational cost of this method is too high to apply to pairs of large documents.

In this paper, we propose an efficient method for enumerating MAs from given document pairs based on graph theory. We also discuss how to use the MA as an index for story databases.

This paper is divided into four sections. In Section 2, we briefly review the concept of maximal analogy between stories (MA). In Section 3, we propose a new method to enumerate MAs from two given documents. In Section 4, we demonstrate experimental results and, based on these, we discuss how to use the MA as an index for a story database. Section 5 concludes this paper.

2 Maximal Analogies Between Stories

2.1 Requirement for the Indices of Story Databases

In this paper, we use the term "story" to mean a plot or event sequence in a document. An "event" is a basic unit that represents facts in a story. In this paper, we use a simplex sentence that has one verb and dependent words for this unit. In order to handle a complex sentence, we decompose it into a sequence of simplex sentences. As each single sentence in a document corresponds to an event, a given document is itself a story, and includes various sub-stories that are sub-sequences of the entire event sequence. Since it is difficult to define meaningful sub-stories a priori, we do not restrict to use any sub-sequences of the event as indices for a story.

To handle documents containing story information, indices of story databases should satisfy the following criteria.

(R1) Since a document can have various aspects and each of which can be represented as a story, an index should contain story information.
(R2) Such indices are automatically extracted from documents.
(R3) The determination of what constitutes a significant story in a document is subjective, so the indexing varies according to individuals.
(R4) Once such indices are discovered and constructed, documents should be quickly accessible by their indices.

The most significant requirement is to determine the most important event (sub-)sequence that characterizes the document. One possible standard approach is to evaluate the significance of events using their frequencies and the co-occurrences of words within them [4]. Although such a scheme is quite effective, some important words or sentences may be missed from a particular story

extracted from the document. The basic argument of this paper is stated as follows.

> The problem of what constitute important events in a document cannot be determined by examining only one document. If some event sequence is regarded as significant from a particular point of view, then we will find another similar document in which a similar event sequence also appears. Conversely, when we find a generalized event sequence common to all the documents, that a user or a group of users consider similar, it may be a candidate for becoming an important sequence, and may therefore be used as a possible index for documents.

More precisely, we say that an event sequence is common to a set of documents whenever the sequence is a generalization of some sequence in every document. As the act of generalizing event sequences depends on the subsumption of relationships between words, we use the EDR dictionary [5]. Furthermore, we consider the concept tree representation of events in Section 2.2. A concept tree is a special case of a concept graph [6], and can be used to represent the case structures of document sentences.

Given two documents judged to be similar (R3), the concept of maximal analogy between the two is introduced in Section 2.3 to formalize the common generalization of event sequences with a maximal set of events. As an MA is itself an event sequence, it can provide a solution for (R1).

Although we have not yet designed a query-answering system for documents indexed by MAs, their subsumption-checking never involves any combinatorial computations. Testing whether a document meets an MA may therefore be performed quickly (R4).

2.2 Minimal Common Subsumer of Concept Trees

After morphological analysis and parsing, each sentence in a document is represented as a rooted tree, with words as nodes and cases (or role symbols) as edges. A verb is chosen as the root, as in Fig. 2. As verbs are first-class entities of events, the tree of words will be simply called an event in Definition 1. Although such a word tree is normally formalized as a semantic network [7], we consider it as a type of concept graph [6], allowing us to define an ordering for trees by restricting a similar ordering for graphs.

To examine the semantic relationship between concept graphs, we use EDR [5], a machine-readable dictionary. As a word may have more than two possible meanings, the dictionary must provide the concepts involved in words together with the relationships between these concepts. The EDR dictionary supports both types of information for Japanese and English words and concepts. Each concept is designated by a unique identifier, called a concept ID. Let $Terms$ be the set of all words and concept IDs in the EDR dictionary. Then a partial ordering $\prec$ over $Terms$ can be given by

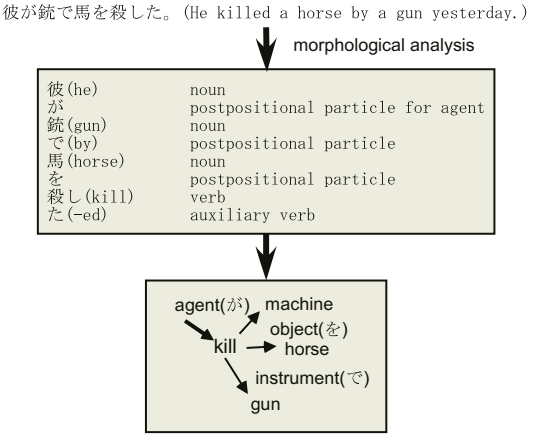

Fig. 1. Construction of a Concept Tree

$t_1 \prec t_2$ iff (1) t_1 and t_2 are both concept IDs and t_1 is more specialized one than t_2 in the concept dictionary, or
(2) t_1 is a word and t_2 is a concept ID that is more general one than any concept ID associated with t_1 in the word dictionary.

Based on this partial ordering for terms, we now have the following definition of concept trees and their ordering.

Definition 1.
`(Concept trees and their paths)` *Given a set $\mathcal{L}$ of role or case symbols, a path of length n is a sequence of roles $p = (\ell_1, ..., \ell_n)$, where $\ell_j \in \mathcal{L}$. The empty path, $\lambda = ()$, of length 0 is always regarded as a path denoting the root of a tree . A concept tree is then defined as a pair $g = (Path(g), term_g)$, also called an event, where $Path(g)$ is a finite and prefix-complete set of paths including the empty path, and $term_g$ is a term labeling function, where $term_g : Path(g) \rightarrow Terms$.*
`(Concept Tree Ordering)` *We say that a concept tree g_s subsumes another concept tree g_i iff, for every rooted path $p \in Path(g_s)$, both $p \in Path(g_i)$ and $term_{g_i}(p) \preceq term_{g_s}(p)$ hold. In this case, we also say that g_s is a generalization of g_i, or that g_i is a specialization of g_s.* ■

We use a morphological analyzer to construct a concept tree. We select noun and verb terms from a sentence and connect them according to the information of postpositional particle. Fig.1 shows an example of this process.

Intuitively, a concept tree g_s is more general than another concept tree g_i if every path in it is preserved in g_i and it uses more general terms than those used in g_i. For instance, both trees at the bottom of Fig. 2 are subsumed by the top tree. Now, the minimal common subsumer (MCS) of two concept trees is defined, similar to the case of the least common subsumers of concept graphs [6]. Formally, an MCS of g_1 and g_2 is defined as a tree consisting of the common

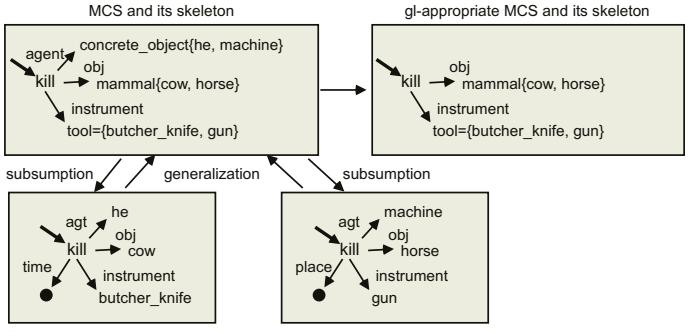

Fig. 2. Concept Trees, where the top example is an MCS of the bottom two

paths of g_j whose labels are some minimal upper bounds of the corresponding paired terms in g_j.

This may be formalized as follows:

$$MCS(< g_1, g_2 >) = (\ Path_{<g_1,g_2>}, \{\lambda_p | p \in Path_{<g_1,g_2>}, \\ \lambda_p = mst(\{term_{g_1}(p), term_{g_2}(p)\}\}), \\ \text{where } Path_{<g_1,g_2>} = Path(g_1) \cap Path(g_2)$$

where $mst(A)$ is a chosen minimal upper bound of a set of terms A. In this sense, mst is called a choice function.

2.3 Maximal Analogy and Its Bottom-Up Construction

The maximal analogy between stories is a sequence of generalized events obtained from different stories. Therefore, we can represent a MA by using a sequence of MCS obtained from the paired events. In order to preserve the order of events in different stories, we define *op-selection* to be a sequence of paired events.

Definition 2.
`(op-selection)` *Each document (D_i) is defined as an ordered sequence $g_1^{(i)}, ..., g_n$ of events $g_j^{(i)}$ in their order of appearance in the story. We denote $g_k^{(i)} < g_l^{(i)}$ whenever $g_k^{(i)}$ precedes $g_l^{(i)}$ in document D_i. For two given stories D_1, D_2, an op-selection θ of these two stories is an order preserving one-to-one correspondence of events in D_1 and D_2. That is, θ is a sequence $P_{i_1,j_1}, ..., P_{i_k,j_k}$, where $P_{i_n,j_n} =< g_{i_n}^{(1)}, g_{j_n}^{(2)} >\in D_1 \times D_2$, $g_{i_n}^{(1)} < g_{i_{n+1}}^{(1)}$ and $g_{j_n}^{(2)} < g_{j_{n+1}}^{(2)}$.*

In addition, in order to remove MCSs that are too abstract, we introduce the following cost function:

$$gcost(t, t') = \min\{length(p) \mid p \text{ is a path connecting } t \text{ and } t'\},$$
$$gcost(\{t_1, ..., t_n\}, t) = \max_j gcost(t_j, t), \text{ where we suppose } t_j \preceq t.$$

We also introduce a given upper-bound of the generalization, gl, and define a gl-appropriate MCS by using this gcost function.

gl-appropriate $MCS(< g_1, g_2 >) =$
$(VPath_{<g_1,g_2>}, \{\lambda_p | p \in VPath_{<g_1,g_2>}, \lambda_p = mst(\{term_{g_1}(p), term_{g_2}(p)\}\}),$
where $VPath_{<g_1,g_2>} = \{Path | Path \in Path(g_1) \cap Path(g_2),$
$gcost(\{term_{g_1}(p), term_{g_2}(p)\}, mst(term_{g_1}(p), term_{g_2}(p)) < gl)\}$
$VPath$ should contain more than two paths, including the root path.

Fig. 2 shows an example of a gl-appropriate MCS when $gcost(\{he, machine\}, concrete_object) > gl$.

We also define $gcost(< g_1, g_2 >)$ and $gcost(\theta)$ as follows.

$$gcost(< g_1, g_2 >) = \max\{gcost(\{term_{g_1}(p), term_{g_2}(p)\}, mst(term_{g_1}(p), term_{g_2}(p))) \mid p \in VPath_{<g_1,g_2>}\}$$
$$gcost(\theta) = \max\{gcost(P) \mid P \in \theta\}$$

Definition 3.
`(Maximal Analogy)` *Given two documents and an upper bound on the generalization level, gl, an op-selection θ is called gl-appropriate if all corresponding sentences have gl-appropriate MCSs. We then say that a gl-appropriate op-selection θ is maximal if there exists no gl-appropriate op-selection θ', such that it properly includes θ ($\theta' \supset \theta$).* ■

The construction of an MA is subject to the construction of maximal op-selections. In order to find maximal gl-appropriate op-selections efficiently, we use the following property corresponding only to the anti-monotonicity of support used in [8] .

`(Monotonicity of Cost)` $gcost(\theta) \leq gcost(\theta')$ if $\theta \subseteq \theta'$.

The construction is bottom-up to enumerate all possible op-selections without any duplication. For this purpose, we first introduce a partial ordering of the set of op-selections.

Let $D_j = g_1^{(j)}, ..., g_{n_1}^{(j)}$ be the entire sequence of events in this order. Then an op-selection θ is expressed as a sequence $P_{i_1,j_1}, ..., P_{i_k,j_k}$, where P_{i_ℓ,j_ℓ} is the ℓ-th pair of the i_ℓ-th event $g_{i_\ell}^{(1)}$ in D_1 and the j_ℓ-th event $g_{j_\ell}^{(2)}$ in D_2. P_{i_ℓ,j_ℓ} is called a singleton selection. The length k is called the level of θ. Then, the partial ordering $\prec$ among op-selections is defined by the transitive closure of the following direct successor relation:

$\theta_1 \prec \theta_2$ iff $\theta_1 = \theta P_{i,j}$ and $\theta_2 = \theta P_{i,j} P_{x,y}$ for some op-selection $\theta, P_{i,j}, x$ and y such that $i < x$ and $j < y$.

From this definition, it follows that any θ of level $k+1$ has only one direct predecessor θ_1 of level k, a prefix of θ. So, by induction on the level k, all the op-selections are enumerated without any duplication according to the ordering $\prec$. Furthermore, we list only op-selections that satisfy the cost condition during the entire enumeration process.

Base step for level 1 op-selections: We list only singleton op-selections that satisfy the cost condition:

$$OPS(1) = \{P_{ij} \mid gcost(P_{i,j}) \leq gl\}.$$

Inductive step for level $k+1$ op-selections: Suppose we have $OPS(k)$ holding all op-selections that satisfy the cost condition. We construct op-selections of the next level consistent with the condition as follows:

$$OPS(k+1) = \{\theta P_{i,j} P_{x,y} \mid \theta P_{i,j} \in OPS(k), P_{x,y} \in OPS(1), \\ i < x,\ j < y,\ gcost(\theta P_{i,j} P_{x,y}) \leq gl \ \ \}.$$

Note that, in the case of $k = 1$, θ is a null string.

Termination of construction: The generation of $OPS(k)$ terminates when we find a level ℓ such that $OPS(\ell) = \phi$, leaving ℓ to have a maximum $\min\{n_1, n_2\}$, where n_j is the number of events in the story D_j.

The number of selections generated and tested is minimized. To verify this, let $gcost(\theta)$ exceed the limit gl for an op-selection θ at level k. Then θ has its unique generation path $\theta_1 \prec .... \prec \theta_k = \theta$ of length $k-1$. As $gcost$ is monotonic, there exists a least j such that $gcost(\theta_j) > gl$. This θ_j is generated, tested and fails the condition, because $\theta_i \in OPS(i)$ for any $i < j$. However, as the predecessor θ_j is not listed in $OPS(j)$, none of its successors, including θ, are ever generated and tested.

The algorithm above shows a method for constructing the MA from two documents. For more than three documents, we iteratively apply the algorithm for two documents.

3 An Efficient Algorithm for Enumerating a Maximal Analogy

3.1 Enumerating MAs Using Directed Graphs

The algorithm discussed in the previous section works well for documents with few sentences, but it fails to enumerate MAs for larger documents. We thus require an efficient algorithm for this enumeration.

In the previous algorithm, the most significant problem is enumerating the many candidate op-selections that should be merged for generating MAs. Therefore, we need an algorithm that does not generate as many candidates. In this section, we propose a new algorithm based on graph theory.

Fig. 3 shows an example of a possible op-selection represented by a directed graph. Rounded nodes represent sentence pairs that have a gl-appropriate MCS. The two numbers in the node correspond to the number of sentences in the first and second document, respectively. Directed links show how connected nodes can be used as a sequence in op-selection.

By using this graph, the generation of op-selections can be formalized as the search for node sets that are connected by directed links. In this case, we have 11 (level 1), 17 (level 2), 7 (level 3), and 2 (level 4) op-selections, with

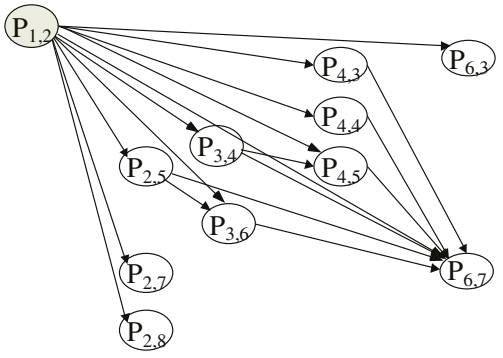

Fig. 3. Possible op-selections represented by a directed graph

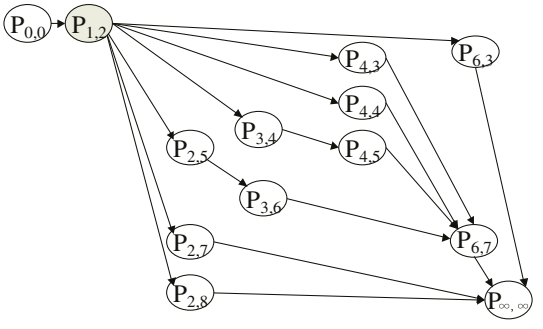

Fig. 4. Possible MAs represented by a directed graph

7 possible MAs (two from level 2, two from level 3 and two from level 4). By using the bottom-up approach, several subsets are generated for deriving one MA. For example, one level 4 op-selection MA $(P_{1,2}, P_{2,5}, P_{3,6}, P_{4,7})$ is generated from four level 3 op-selections, that is, as a subset of the MA $(P_{1,2}, P_{2,5}, P_{3,6})$, $(P_{1,2}, P_{2,5}, P_{4,7})$, $(P_{1,2}, P_{3,6}, P_{4,7})$, $(P_{2,5}, P_{3,6}, P_{4,7})$.

In order to reduce the number of these subset enumerations, we remove directed links that are ineffective in generating MAs. In the previous example, links between $P_{1,2}$ and $P_{3,6}$, $P_{2,5}$ and $P_{4,7}$ are ineffective because we can insert $P_{2,5}$ or $P_{3,6}$ between these nodes. In general, we can define ineffective links as follows:

Definition 4.
`(Ineffective Link)` *A link between two nodes $(P_{i,j}P_{k,l})$ that can combine other gl-appropriate nodes $(P_{x,y} : i < x < k, j < y < l)$ is an ineffective link.* ■

Since we can enumerate op-selections with ineffective links by using two or more links (e.g., links between $P_{i,j}$ and $P_{x,y}$, and $P_{x,y}$ and $P_{k,l}$), removal of these ineffective links does not affect the results for the enumeration of possible MAs.

In order to formalize this enumeration process, we introduce two virtual nodes $P_{0,0}$ and $P_{\infty,\infty}$ that we call start node and end node respectively. Fig. 4 shows

a graph that removes ineffective links from the graph in Fig. 3 and adds $P_{0,0}$ and $P_{\infty,\infty}$. By using this graph, it is possible to enumerate MAs by following all paths between $P_{0,0}$ and $P_{\infty,\infty}$.

We can summarize the algorithm as follows.

1. **Base step for level 1 op-selections:** (Same as previous algorithm) We list only singleton op-selections satisfying the cost condition:
$$OPS(1) = \{P_{i,j} \mid gcost(P_{i,j}) \leq gl\}.$$
2. **Directed Link generation:** We represent singleton op-selections as a node. We also add $P_{0,0}$ and $P_{\infty,\infty}$, and generate links between nodes (from $P_{i,j}$ to $P_{x,y}$) that satisfy the following criteria.
   - $i < x$ and $j < y$.
   - The link is not an ineffective link.
3. **Enumeration of MAs:** All possible MAs are enumerated by enumerating all directed paths between $P_{0,0}$ and $P_{\infty,\infty}$.

The high computational cost of step 1 is inevitable when generating MAs. The computational complexity of this step is $O(n * m)$ (where n, m are the numbers of sentences in the two documents. In step 2, we generate links between each node, and the node size is of order $O(n * m)$. For each node, we must check the connectivity of the link from $P_{i,j}$ as follows:

a Check the connectivity for $P_{i+1,x}(x = j + 1, \cdots, m)$.
b Check the connectivity for $P_{i+2,x}(x = j + 1, \cdots, k)$, where k is the smallest number of $P_{i+1,k}$ connected from $P_{i,j}$. It is not necessary to check the connectivity for $P_{i+2,x}(x = k+1, \cdots, m)$, because $P_{i+1,k}$ exists between $P_{i,j}$ and $P_{i+2,x}$
c Iterate steps a and b until $P_{n,x}$.

Therefore, the computational complexity of link generation for each node is $O(n+m)$ and the total computational complexity is $O(n*m*((n+m))$. However, since each link generation process (O(n+m)) is negligible compared to the graph generation in Step 1, it takes negligible time compared to Step 1.

A higher computational cost is incurred for Step 3. However, one important feature of this algorithm is that we can control the order of enumeration. For example, the longest MAs, subject to the longest path between $P_{0,0}$ and $P_{\infty,\infty}$ in this directed graph, can be calculated by using the following algorithm.

a Set the path length of each node ($pl_{i,j}$) to 0 and a set of candidate longest paths ($clp_{i,j}$) to ϕ.
b Start from node $P_{0,0}$ and follow the connected links to check whether each connected link is a candidate for a part of the longest path.
   (1) Let $P_{k,l}$ is the start node and $P_{m,n}$ is a node connected by a link. When $pl_{k,l} + 1 \geq pl_{m,n}$, the connected link is a candidate link.
   (2) When $pl_{k,l}+1 = pl_{m,n}$, new $clp_{k,l}$ is computed by the following operation $clp_{k,l} \cup \{clp, \text{links between } P_{i,j} \text{ and } P_{k,l} | clp \in clp_{i,j}\}$. When $pl_{k,l} + 1 > pl_{m,n}$, new $clp_{k,l}$ is computed by the following operation $\{clp, \text{links between } P_{i,j} \text{ and } P_{k,l} | clp \in clp_{i,j}\}$.
c $clp_{\infty,\infty}$ is a set of longest paths of this graph.

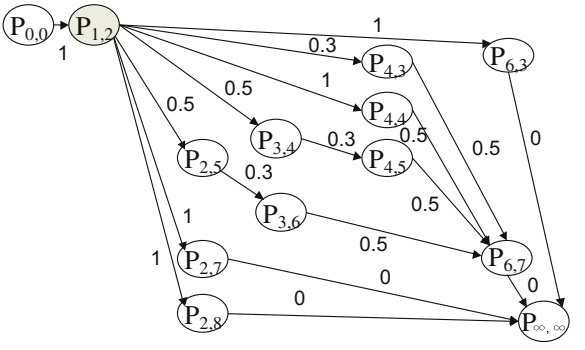

Fig. 5. Possible MAs represented by a weighted directed graph

3.2 Enumerating MAs by Using an Evaluation Function

In the previous algorithm, MAs were enumerated using the longest path. However, these longest MAs may not be appropriate. For example, each singleton op-selection has a gcost, and those with higher gcost may not be appropriate for inclusion if they are very abstract. Therefore, it is better to include a mechanism to order MAs by considering the gcost of each singleton op-selection.

In order to handle this, we formalize this problem with a weighted directed graph scheme. We define an evaluation function $eval(P_{i,j})$ to represent the suitability of including MAs based on their gcost function. This function increases for lower gcost values (i.e., 1 / (gcost + 1)). We also set the weights of links based on the nodes to which they are connected. For example, a link between $P_{i,j}$ and $P_{k,l}$ takes the weight $eval(P_{k,l})$. We also formalize the suitability of an MA by using the sum of the $eval(P_{i,j})$ values of all op-selections.

Since all MAs include $P_{\infty,\infty}$, the effect of $eval(P_{\infty,\infty})$ is canceled and the value of $eval(P_{\infty,\infty})$ does not have any influence. For convenience, we set $eval(P_{\infty,\infty}) = 0$.

Fig. 5 shows an example of the weighted directed graph of Fig. 4. In this case $eval(P_{2,5}) = eval(P_{3,4}) = eval(P_{6,7}) = 0.5$ and $eval(P_{3,6}) = eval(P_{4,3}) = eval(P_{4,5}) = 0.3$, and other nodes have $eval(P_{i,j}) = 1$. In this case MA ($P_{1,2}$, $P_{4,4}$, $P_{6,7}$) is the best MA based on this *eval* function.

The longest paths between $P_{0,0}$ and $P_{\infty,\infty}$ indicate the most appropriate MAs in this weighted directed graph. We can find out these paths by modifying algorithm discussed in Section . We modify step b. of the algorithm as follows.

b Start from node $P_{0,0}$ and follows the connected links to check whether each connected link is a candidate for a part of the longest path.
  (1) Let $P_{k,l}$ is the start node and $P_{m,n}$ is a node connected by a link. When $pl_{k,l} + eval(P_{m,n}) \geq pl_{m,n}$, the connected link is a candidate.
  (2) When $pl_{k,l} + eval(P_{m,n}) = pl_{m,n}$,
  $clp_{k,l} = clp_{k,l} \cup \{clp, \text{links between } P_{i,j} \text{ and } P_{k,l} | clp \in clp_{i,j}\}$.
  When $pl_{k,l} + eval(P_{m,n}) > pl_{m,n}$,
  $clp_{k,l} = \{clp, \text{links between } P_{i,j} \text{ and } P_{k,l} | clp \in clp_{i,j}\}$.

4 Experimental Results and Discussion for the Story Database

4.1 Experiments Generating Maximal Analogies

We have implemented this algorithm on a Linux-based PC (CPU: Dual Pentium III 1.0GHz, RAM: 4 GB). The basic procedure for obtaining MAs is:

1. Concept tree construction for each sentence
   We use the Cabocha parser [9] to obtain a case structure for each sentence, and convert this to a concept tree. The cases are therefore superficial.
2. Singleton *gl*-appropriate op-selection generation
   We compare pairs of concept trees created from two documents and generate a *gl*-appropriate op-selection. We use the EDR dictionary to calculate gcost.
3. Link generation
   We check the connectivity for each pair of *gl*-appropriate op-selections and generate links. We set the value of parameter *gl* to 4.
4. Generation of an appropriate MA
   We determine the highest appropriate MA using the evaluation function $eval(\theta) = \sum_{P \in \theta}(1/(1 + gcost(P)))$.
5. Enumeration of MAs
   We enumerate all possible MAs.

In order to analyze the computational cost of this algorithm and evaluate its efficiency, we use a children's story and a short folktale, as used in a previous paper [3] , as input for our algorithm.

Both of these two input stories contain 46 sentences, and have a common plot as follows.

(E1) There are two brothers.
(E2) The younger brother gains some property.
(E3) The elder brother kills the younger brother in order to steal the property.
(E4) The bone of the younger brother then sings a song that reveals the crime.
(E5) As a result, the older brother is caught and punished.

Table 1 shows the results obtained using these two input stories, and Table 2 shows the corresponding computational times for each step.

In some of the most appropriate MAs, (E1) and (E4) are correctly recognized and parts of (E3) and(E5) are also realized in the same MA. However, we find no generalized events corresponding to (E2). Since sentences in these documents have a finer granularity, our system can find more information than the results discussed in [3].

However, we still experience the same problem as the one discussed in [3]. Since the concept hierarchy of the EDR dictionary is constructed for general purposes, it does not classify terms into effective term groups that can perform the same role in different documents.

Table 1. Results obtained from two stories

gl-appropriate singleton op-selection	130
Number of links	715
Number of appropriate MAs	76
Maximum MA length	18
Number of possible MAs	905306

Table 2. Computational time for each step

Concept tree construction	17 s
Singleton *gl*-appropriate op-selection generation	3 s
Link generation	less than 1 s
Appropriate MA generation	1 s
Enumeration of MAs	672 s

4.2 Towards Constructing Story Databases Using MAs

From these results, we confirm that the number of possible MAs is very large, and that it is difficult to determine the most appropriate MA manually. Therefore, a good scoring function is required to select the appropriate indices.

We are planning two approaches to support this selection.

Definition of a better evaluation function. In order to improve the scoring function, we plan to apply the measure of importance or significance of terms [4] and use this for the evaluation function.

Usage of a base query. Since most MAs do not contain a common plot, which was assumed when extracting from a pair of stories, we introduce the concept of a base query to represent a simple skeleton of a common plot (that is, one described in few sentences) that should be included in MAs and used as a criterion for selecting meaningful MAs.

For future work, we also need a mechanism for retrieving a story based on the subsumed relationship between MAs and the query.

5 Conclusion

In this paper, we have proposed an efficient algorithm based on graph theory to generate maximal analogies (MAs), and confirmed the reduction in computation time. However, as the number of possible MAs is very large, we need a method to support selection of meaningful MAs for using MAs as effective indices of story databases. We also outline future work too.

References

1. Mani, I.: Automatic Summarization. John Benjamin Publishing Company (2001)
2. Firmin, T., Chrzanowski, M.J.: An evaluation of automatic text summarization systems. In Mani, I., Maybury, M.T., eds.: Advances in automatic text summarization, Cambridge, Massachusetts, The MIT Press (1999) 325–340
3. Haraguchi, M., Nakano, S., Yoshioka, M.: Discovery of maximal analogies between stories. In: Proc. of the 5th Int'l Conf. on Discovery Science - DS'02 LNCS 2534, Springer (2002) 324–331
4. Ohsawa, Y., Benson, N.E., Yachida, M.: Keygraph: Automatic indexing by cooccurrence graph based on building construction metaphor. In: Proceedings of the Advances in Digital Libraries Conference. (1998) 12–18
5. Japan Electronic Dictionary Research Institute, Ltd. (EDR): EDR ELECTRONIC DICTIONARY VERSION 2.0 TECHNICAL GUIDE TR2-007. (1998)
6. Cohen, W.W., Hirsh, H.: The learnability of description logics with equality constraints. Machine Learning **17** (1994) 169–199
7. Sowa, J.F., ed.: Principles of Semantic Networks. Morgan Kaufmann (1991)
8. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: Proc. 20th Int. Conf. Very Large Data Bases, VLDB, Morgan Kaufmann (1994) 487–499
9. Kudo, T., Matsumoto, Y.: Japanese dependency analysis using cascaded chunking. In: CoNLL 2002: Proceedings of the 6th Conference on Natural Language Learning 2002 (COLING 2002 Post-Conference Workshops). (2002) 63–69