An Incremental Concept Formation Approach to Acquisition of Anaphoric Regularity in Mandarin Chinese

Benjamin L. Chen
Application Software Department
Computer and Communication Research Laboratories
Industrial Technology Research Institute
W300/CCL, #1 R&D Rd. I.
Hsinchu, 30045, Taiwan, R. O. C.
e-mail: blchen@jaguar.ccl.itri.org.tw

Von-Wun Soo
Department of Computer Science
National Tsing-Hua University
Hsinchu, 30043, Taiwan, R. O. C.
e-mail: soo@cs.nthu.edu.tw

Abstract
A modified version of incremental learning model of Lebowitz's UNIMEM is proposed in this paper. This new model is called G-UNIMEM which is motivated from the natural language acquisition study. It can extract causal relations from a set of training instances annotated with specified cause and goal features. The experiment has been carried out on the acquisition of anaphoric regularity in Mandarin Chinese.

Keywords: incremental learning, case-based learning, natural language acquisition, anaphora

1 Introduction
In natural language acquisition domain, the restrictions of positive-only examples have prohibited many machine learning methods as a feasible natural language acquisition model. However, a case-based learning approach such as Lebowitz's UNIMEM [6] seems to be a candidate due to its capability to form concepts incrementally from a rich input domain. Nevertheless, to use UNIMEM directly to acquire natural language such as the acquisition of anaphoric regularity in Mandarin Chinese is still not sufficient. We have therefore modified UNIMEM into G-UNIMEM.

2 Review of UNIMEM
Lebowitz's UNIMEM is an incremental learning system that uses the GBM (Generalized-based Memory) paradigm to generalize from a large set of training instances. GBM maintains a hierarchy of GEN-NODEs to retain the training instances. Each GEN-NODE can be constructed from instances and more specific GEN-NODEs. The training instances reside in a given GEN-NODE tend to share similar features. The detail is not mentioned here. Though UNIMEM is an incremental learning model, it suffers the deficits of no formal semantics for predictiveness and predictability and no principled method for parameters to decide learning operators [3].

3 G-UNIMEM: a goal-oriented case-based learning model
G-UNIMEM is also an incremental learning system that uses GBM (Generalized-based Memory) to generalize concepts from a large set of training instances. The program has been implemented in Quintus PROLOG and on SUN workstation.

Similar to UNIMEM, G-UNIMEM organizes input training instances into a memory hierarchy according to the frequencies of features. However, its goal is to explicitly express the generalised causal relationships between two specified types of features: cause features and goal features. Since there may be inconsistency due to lack of cause features, further refinement is needed to obtain more consistent causal relations. Thus, there are four different modules in G-UNIMEM to complete different functions to achieve this purpose.

The classifier is the main module and three other modules: the rule generator, the rule filter and the feature selector serve as the roles to refine the causal relations in the memory hierarchy.

G-UNIMEM differs from UNIMEM in two respects. FIRSTLY, if a drinker got drunk many times after taking either whiskey and water or brandy and water, he would induce that water made him drunk with UNIMEM. This is intuitively incorrect. Whereas, with G-UNIMEM, he would induce that whiskey and water, brandy and water or water would cause him drunk. In this case, G-UNIMEM retains the possible causal accounts. SECONDLY, G-UNIMEM can extract explicit causal rules from memory hierarchy.

3.1 The classifier
The classifier is the first module that processes all training instances for G-UNIMEM. Its function is close to UNIMEM that organizes a hierarchy structure to incrementally accommodate a training instance and
at the same time generalize the features based on similarities among training instances. The forming hierarchy can be organized as either a g-c-hierarchy or a c-g-hierarchy depending on the setup of system, which is defined in Definition 1. In Appendix A we show the basic classifier algorithm. For example, the drinker with g-c-hierarchy would induce that whiskey and water, brandy and water or water would cause him drunk; whereas, he would induce whiskey and water, brandy and water with c-g-hierarchy. C-g-hierarchy is more efficient since no rules are needed to be generated. Fig. 3 and Fig. 4 show how a training instance causes the change of a GBM.

![Fig. 1 A g-c-hierarchy of GBM](image1)

![Fig. 2 A c-g-hierarchy of GBM](image2)

**Definition 1** A g-c-hierarchy is the hierarchy that every generalized goal feature resides in a GEN-NODE and there is no generalized cause feature that resides between the root node and this GEN-NODE. A c-g-hierarchy doesn't allow any generalized goal feature to reside in the GEN-NODE between the root node and any GEN-NODE where generalized cause features reside.

![Fig. 3 A GBM with two training instances](image3)

![Fig. 4 A new GBM after inserting a new instance \(((g,\text{type(pronoun)}),(g,\text{ante(theme)}))\)](image4)

However, because there may be conflicts in extracted causal relations, it needs other modules to detect the conflicting relations and resolve the conflicts.

### 3.2 The rule generator

Once the g-c-hierarchy has been constructed by the classifier, the causal relations (rules) will be extracted. The rule generator module serves as the role to generate causal rules from the g-c-hierarchy. It generates all causal rules from hierarchy as the regularity is retrieved for predictions.

In Fig. 6, if a testing instance is given for choosing anaphora with feature list \(((g,\text{type(??)}),(g,\text{ante(theme)}),(c,\text{f1(theme)}),(c,\text{anaphor(theme)}),(c,\text{p2(obj)}),(c,\text{p(v)})\)), the regularity is searched with a pre-order traversal. That is in the order sequence 1, 2, 3, 4 and 5. There may be more than one candidate for the prediction. The system can be setup to select either the first or the most specific one. If the first one is preferred, type(nil) is yielded as the prediction. If the most specific answer is preferred, all possible rules will be tried and the one with most contingent features matched will be the answer (i.e. type(pronoun)).

1: \[[(g,\text{type(nil)}),(g,\text{ante(theme)})]:- [c,\text{f1(theme)}],[c,\text{anaphor(theme)}]]
2: \[[(g,\text{type(nil)}),(g,\text{ante(agent)})]:- [c,\text{f1(agent)}],[c,\text{f2(theme)}],[c,\text{anaphor(agent)}]]
3: \[[(g,\text{type(nil)})]:- []\]
4: \[[(g,\text{type(pronoun)}),(g,\text{ante(theme)})]:- [c,\text{f1(theme)}],[c,\text{anaphor(theme)}]]

**Fig. 5 The sample rules generated from Fig. 1.**

The sample rules generated from Fig. 1 are shown in Fig. 5. Before generating rules, the GBM is adjusted.
so that all children of a GEN-NODE are ordered according to their confidence scores of features. Then all rules are generated in a post-order traversal. After rules are generated, it proceeds to rule filter module.

3.3 The rule filter

The rule filter removes those rules that are ill-formed and useless. For example, the causal rule 3 in Fig. 5. has no causes. It also detects conflicting rules. Conflicting rules are those that have different goal feature descriptions, which are accounted by the same cause feature descriptions. For example, the rule 1 and rule 4 in Fig. 5. are conflicting. These rules will be detected in this module and then to be resolved by the feature selector.

3.4 The feature selector

Any two conflicting rules are resolved by the feature selector through augmenting the two rules with mutual exclusive contingent cause features, which are prepared in advance. Dominant features were used in initial regularity acquisition stage; whereas contingent features were used in feature selection stage. The dominant features such as goal features are assumed to be those that must be present in every anaphoric rule. Contingent features are optional. Fig. 6. shows the GBM with g-c hierarchy after feature selection process.

After augmenting the two conflicting rules with two disjoint cause features, the drinker might conclude new rules: 1) water and wine made him drunk and 2) water and orange juice didn’t make him drunk.

3.5 Difference between UNIMEM and G-UNIMEM

Since the properties of G-UNIMEM, there are 7 differences between G-UNIMEM and UNIMEM. They are summarized as follows:

- Two category features: either goal feature or cause feature are used in G-UNIMEM.
- The feature description is uniform in UNIMEM whereas it is more flexible in G-UNIMEM. G-UNIMEM allows training instances with varied cause features and with uniform goal features.
- When there is an exact match between the features of a training instance and the features of a GEN-NODE, general operations are to split the node and thus there is no decrement of confidence score in G-UNIMEM.
- Since there is no decrement of confidence score, there is no deletion of feature in G-UNIMEM due to the confidence score is below threshold.
- There is no need for G-UNIMEM to set system parameters (i.e. thresholds).
- The confidence score of any immediate GEN-NODE after subtracting the number of resident instances is equal to the summation of all confidence scores of its immediate sub-GEN-NODEs.
- Those generalized GEN-NODEs with high confidence scores are retrieved with high priority in classification and prediction. This is equivalent to that some typical categories are retrieved more rapidly and named more frequently [3].

4 Automatic acquisition of anaphoric regularity

In Chinese natural language processing, there are both the problems of choosing and resolving anaphora. In Mandarin Chinese, several linguists have attempted to propose criteria for choosing anaphora but with controversial results. On the other hand, search-based computational techniques such as the history list approach [1] or Hobbs’s algorithm [4] for resolving anaphora are neither the best way to resolve Chinese anaphora nor to facilitate choosing anaphora.

Thus, to facilitate both choosing and resolving anaphora with accuracy and efficiency, we propose the case-based learning model G-UNIMEM. These are used to automatically acquire anaphoric regularity from a sample set of training sentences.

The results show that experiments with g-c hierarchy have a little higher accuracy rates (95.8%)
for resolving and 90.8% for choosing anaphora with 120 training instances) than those with g-c-hierarchy. Both accuracy rates are higher than those with Tai's criteria [9]. Thus, G-UNIOMEM with semantic roles as dominant features promises much higher accuracy rate.

5 Conclusion

We have illustrated an incremental learning model G-UNIOMEM for forming concepts from a group of training instances, which serves as finding relevant causes and causal relations between causes and effects. It inherits the advantages of UNIOMEM, allows more flexible concept descriptions and gets away from some deficits in UNIOMEM. G-UNIOMEM has been applied to the language acquisition problem. Although the 120 training and testing instances are small, the result has a considerable high accuracy rate. It concludes that G-UNIOMEM is a useful learning model.

References


Appendix A. The basic classifier algorithm

Input: The current node N of the concept hierarchy.

The name I of an unclassified instance.

The set of I's unaccounted features F.

Results: The concept hierarchy that classifies the instance.

Top-level call: Classifier( Top-node, I, F)

Variables: N, N', C and NC are nodes in the hierarchy.

G, H, and K are sets of features.

J is an instance stored on a node.

R is a variable of set.

Classifier(N, I, F).

Let G be the set of features stores in N.

Let H be the features in F that match features in G.

Let K1 be the features in F that do not match features in G.

Let K2 be the features in G that do not match features in H.

Adjust( H,K1,K2,H',K1',K2') and let H', K1' and K2' be the sets of features
/* adjust goal and cause features for g-c-hierarchy or c-g-hierarchy */

if N is not the root node,

then if H is empty set /* no features match */

then return False
else if both H' and K1' are not empty sets

then increase each confidence score in H';

NC contains the remaining features

and instances;

return Split.

else if H' and H are equal

for each child C of node N

/* continue match remaining features */

call Classifier(C, I, K1')

if any Classifier(C, I, K1') return True

or Split then break.

if R is [False ]

/* All trials fail, try to do generalization */

then for each instance J of node N

call Generalize(N, J, I, K1')

if any Generalize(N, J, I, K1') is True then break.

if R is [False ]

/* All trials fail, insert instance I into N */

then store instance I with features K1' into N.

return True.