Graph-Based Induction for General Graph Structured Data

Takashi Matsuda[1], Tadashi Horiuchi[1], Hiroshi Motoda[1], Takashi Washio[1],
Kohei Kumazawa[2] and Naohide Arai[2]

[1] I.S.I.R., Osaka University, 8-1 Mihogaoka, Ibaraki, Osaka 567-0047, JAPAN
[2] Recruit Co., Ltd., 8-4-17 Ginza Chuo-ku, Tokyo 104-8001, JAPAN

1 Introduction

A machine learning technique called Graph-Based Induction (*GBI*) efficiently extracts typical patterns from a directed graph data by stepwise pair expansion (pairwise chunking). We expand the capability of the *GBI* so that it can handle not only a tree structured data but also a graph data with multi-inputs/outputs nodes and loop structure (including a self-loop) which cannot be treated in the conventional way. We show the effectiveness of our approach by applying to the real scale World Wide Web browsing history data.

2 Graph-Based Induction for General Graphs

The original *GBI* was so formulated to minimize the graph size by replacing each found pattern with one node that it repeatedly contracted the graph. The graph size definition reflected the sizes of extracted patterns as well as the size of contracted graph. This prevented the algorithm from continually contracting, which meant the graph never became a single node. Because finding a subgraph is known to be NP-hard, the ordering of links is constrained to be identical if the two subgraphs are to match, and an opportunistic beam search similar to genetic algorithm was used to arrive at suboptimal solutions. In this algorithm, the primitive operation at each step in the search was to find a good set of linked pair nodes to chunk (pairwise chunking) [Motoda97].

In this paper, we expand the capability of the *GBI* to handle general graph structured data such as directed graphs including multi-inputs/outputs nodes and loop structure (including a self-loop). We propose an idea to perform pairwise chunking without loosing the information of link connections. In order to apply *GBI* to general graph structured data, we adopt a method to represent the graph structured data using table forms by paying attention to link information between nodes. We introduce the “self-loop distinction flag” to identify self-loop when the parent node and child node are of the same kind ($Eg.a \rightarrow a$). Moreover, each time we perform the pairwise chunking, we keep link information between nodes in order to be able to restore the chunked pairs to the original patterns. The basic algorithm of the proposed method which extends *GBI* to handle a general graph structured data is shown in **Fig.1**. In this implemented program, we use the simple “frequency” of pairs as the evaluation function for the stepwise pair expansion. The method is verified to work as expected using artificially generated data and we evaluated experimentally the computation time of the implemented program. The computation time for 30,000 times repetition is shown in **Fig.2** for three kinds of graph structured data (Data4: loop type, Data5: lattice type, Data6: tree type) for which there are three kinds of node labels. From this figure, it is found that the computation time increases almost linearly with the number of chunking. **Table 1** shows the preliminary result for the classification problem of promoter DNA sequence data (total 106 cases).

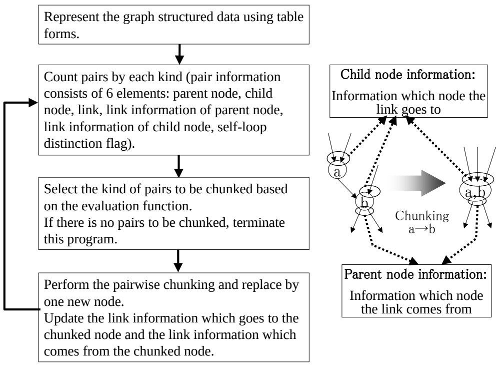
Represent the graph structured data using table forms.
Count pairs by each kind (pair information consists of 6 elements: parent node, child node, link, link information of parent node, link information of child node, self-loop distinction flag).
Select the kind of pairs to be chunked based on the evaluation function.
If there is no pairs to be chunked, terminate this program.
Perform the pairwise chunking and replace by one new node.
Update the link information which goes to the chunked node and the link information which comes from the chunked node.
Child node information:
Information which node the link goes to
a
b
Chunking
a→b
a b
Parent node information:
Information which node the link comes from

Fig.1. Proposed algorithm

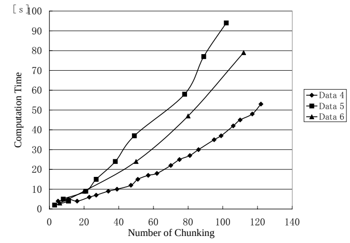

Fig.2. Computation time

Table 1 Experimental result

Learning Method	ID3	C4.5	*GBI*
Number of Errors	19	18	16

3 Application to WWW Browsing Histories

The performance of the proposed method has been examined through a real scale application. The data analyzed is the log file of the commercial WWW server of Recruit Co., Ltd. in Japan. The URLs on WWW form a huge graph, where URLs represent nodes connected by many links. When a client visits the commercial WWW site, he/she browses only a small part of the huge graph in one access session, and the browsing history of the session becomes a small graph structured data. The total number of the URLs involved in this commercial WWW site is more than $100,000$, and it is one of the largest site in Japan. Its total number of hit by the nation wide internet users always remains within the third place from the top in every month in Japanese internet record, and the typical size of the log file of the WWW server for a day is over $400MB$.

As the log file consists of the sequence of the access records, they are initially sorted by the IP addresses, and then we transform the subsequence of each client into the graph structured data (total 150,000 nodes). After this preprocessing, we executed the proposed method using the frequency of pairs as the evaluation function. When we use the frequency threshold $0.1\%, 0.05\%, 0.025\%$ of the total nodes, the number of derived chunk patterns results in respectively $33, 106, 278$.

We could extract some interesting browsing patterns of many clients such as a) clients follow some URLs in the same directory, b) clients go deep into the directories one after another, c) clients jump to the URLs in a different directory after following some URLs in the same directory.

References

[Motoda97] H. Motoda and K. Yoshida: Machine Learning Techniques to Make Computers Easier to Use, *Proc. of IJCAI'97*, Vol.1, pp.1622–1631, 1997.