Project Number 001907

DELIS
Dynamically Evolving, Large-scale Information Systems

Integrated Project
Member of the FET Proactive Initiative Complex Systems

Deliverable D6.4.4

Applying episode mining and data stream mining to the tasks of evolving graphs and distributed Web search
Start date of the project: January 2004

Duration: 48 months

Project Coordinator: Prof. Dr. math. Friedhelm Meyer auf der Heide
Heinz Nixdorf Institute, University of Paderborn, Germany

Due date of deliverable: December 2006
Actual submission date: January 2007
Dissemination level: PU – public

Work Package 6.4: Mining Episodes and Data Streams

Participants: Universita Pompeu Fabra (UPC), Barcelona, Spain

Authors of deliverable: José L. Balcázar (balqui@lsi.upc.edu)
Albert Bifet
Antoni Lozano
Gemma C. Garriga
1 Introduction

This workpackage has already contributed to the state of the art in episode mining, through both theoretical developments and implementations of the algorithms found. A previous deliverable has consisted of a prototype system called ISSA (Integrated System for Sequence Analysis) which can be applied to the analysis of sequential data; the major reference about this system, which includes large theoretical developments and several applications, is the PhD Dissertation of Gemma C Garriga, now a postdoc at Helsinki with Heikki Mannila.

The goal of this workpackage was to develop a system based on episode mining to be tested as a potentially useful subsystem of a distributed search engine. Our first implementation, whereas useful for the somewhat limited first set of problem, had naturally two drawbacks; one, that was already known, is that to be useful for a search engine one should be able to tackle datasets that are not sequentially structured but, rather, graph-like. This fact notwithstanding, the difficulties in the study had convinced us that our approach, to be successful, had to start with the somewhat simpler structures corresponding to sequential data. Moreover, as has been actually the case, we expected that an implementation of solutions to that problem had interest and utility in themselves, beyond being a stepping stone towards more ambitious mining algorithms.

A second limitation of our implementation, of which we were only aware after a certain degree of experimentation, was that our organization of the internal workings of the algorithms was too rigid; making changes to experiment with it was possible but not too easy. Therefore, we decided that, before embarking into generalizations of the algorithmics to deal with structured data, it was better to reengineer the whole system into an object-oriented approach, with very carefully designed classes. Therefore, our goals for this year were to reimplement the ISSA system and extend it with algorithms for association rules on partial orders and handling of tree-like unlabeled data.

(Due to an additional task that we decided to tackle on the way, namely, the incorporation of the CloSpan algorithm whose source code was made public by the Illimine group, the integration of the rules on partial orders is not fully complete, though working programs exist; and the handling of tree-like data is slightly behind schedule. We strongly believe we will be able to make up for the delay in time for checking integration with the prototype deliverables at the end of the project.)

2 Main contributions

We expected that the handling of link-based structures, at least in tree form, were translatable into appropriate processes on sequences representing them, through the already implemented tools. However, it turned out that our process was incomplete, in the sense that it could fail unless a specific combinatorial property did hold. We identified precisely that property, and mathematically proved that it was a sufficient condition for our process to work, but did not want to have this limitation in our implemented tool. A more precise description of this part of the work is in preparation.

A reworking of the process, through a mechanism of tree transformation that imposed the mentioned combinatorial property, was designed. The resulting method is valid, but, however, the computational effort that is consequence of the tree transformation mechanism made the process far too slow to be of any usefulness. We have considered two alternatives in order to continue our development; they are briefly described in the next sections.

2.1 Partial Orders

We had developed, for the clustering process made by ISSA, a way of identifying a partial order from the set of (maximal) total orders appearing in it; whereas this identification has remained a heuristic, particular cases of it (notably the case of injective partial orders) have been formally proved to provide a maximally specific generalization operator.
In parallel with the reengineering process (and, therefore, on the basis of the previous software deliverable “as is”), we studied manners of handling partially ordered data, instead of total orders. The results were rather successful, and were announced, together with other related results, in workshops (without proceedings). Let us explain them here just by way of an example: if the input partial orders are those of Figure 1, then each subset of them can be described by the corresponding partial order, according to the lattice of closed partial orders depicted in Figure 2.

(We wish to point out that, from this point on, we started referring to ISSA as standing for Integrated System for Structural Analysis, since this functionality transcends the original approach of Sequence Analysis.)

We did not deem the results obtained worth a higher publication attempt, at least not yet; they may join future work into stronger publication candidates.

2.2 Unlabeled Trees

The second alternative we pursued was based directly on assuming that the data is a link structure; not yet a graph but a tree. We chose a number of working hypotheses: labels on the nodes may be
Frequent_Mining\((D, min\_sup, T)\)

\[t \leftarrow \text{a node tree}\]

\[
\text{(Un)Ordered}_\text{Subtree}_\text{Mining}(t, D, min\_sup, T)
\]

\[\text{return}\]

Figure 3: The Subtree Frequent Mining algorithm

unreliable or irrelevant for a link-based analysis; likewise, other information local to the node such as the order in which hyperlinks appear is not to be given relevance; paths are not substitutes for direct links; and the start node was to be preserved. We consider these as the natural working hypothesis for the application we envision, but other applications may require algorithms based on different hypotheses. The formal version of these working hypotheses are: our basic combinatorial structures are unlabeled, unordered trees (this formalizes the first two hypotheses), and subtrees must preserve the parent relation (instead of the ancestor relation) and share the same root, that is, what is usually called “top-down induced subtrees” (this formalizes the last two working hypotheses).

Given the current state of the art, it was clear to us that the appropriate tool, extension of those present in ISSA, was closure-based analysis. This required a definition of intersection of unordered trees, in order to build a Galois connection and a closure operator matching the notion of “closed” based plainly on support considerations. Our work with sequences had warned us that mere support considerations give a weak notion of “closed” that does not necessarily correspond to a closure operator; moreover, formal closures may require to consider sets of the combinatorial structures under consideration, rather than single individuals as in support-based notions.

So it proved indeed: our trees exhibit a notion of intersection quite different from standard set-theoretic intersection, and quite close to the one that we used on sequences to develop ISSA. We proved that exponential size intersections may appear (in terms of the height of the given trees), analyzed a number of other parameters, and presented two algorithms computing the intersection of two trees: a naive one and another one, somewhat more sophisticated, based on dynamic programming, considerably faster. An example of intersections (on trees obtained from the now-standard Zaki generator, and not made up for the example at all) is shown at the end of this report in figure 8. This part appears in [DELIS-TR-0468].

Further, we have developed algorithms for the computation of support-based closures, mainly based on intuitions from the gSpan algorithm, and resorting to a novel representation for unlabeled trees consisting of sequences of natural numbers obeying a certain condition. We prove that a simple operation on these sequences allow us to traverse all candidate subtrees without repetition, in a controlled manner allowing for support and closedness testing in quite efficient ways. This provided an algorithm for mining frequent closed subtrees from a dataset of ordered trees.

Then, we demonstrated how to restrict the operation so that unordered trees are traversed efficiently; we use ordered trees to traverse them, and restrict the operation so that for each unordered tree, a single corresponding ordered tree is visited, instead of all the exponentially many possible ones. Mathematical validations of the correctness of this pruning were provided. These algorithms appear in the following figures; a publication presenting this development is being submitted to a conference simultaneously with this report being submitted.

Figure 6 illustrates the framework, which includes the choice of the minimum size subtree closed to call \text{Closed}_\text{Subtree}_\text{Mining} recursively. This subtree is the subtree of depth and size the number
Ordered Subtree Mining \((t, D, \text{min}_\text{sup}, T)\)

Input: A tree \(t\), a tree dataset \(D\), and \(\text{min}\_\text{sup}\).
Output: The frequent tree set \(T\).

1. for every \(t'\) that can be extended from \(t\) by a step operation
2. do if \(\text{support}(t') \geq \text{min}_\text{sup}\)
3. then Ordered Subtree Mining \((t', D, \text{min}_\text{sup}, T)\)
4. return

Figure 4: The Ordered Subtree Mining algorithm

Unordered Subtree Mining \((t, D, \text{min}_\text{sup}, T)\)

Input: A tree \(t\), a tree dataset \(D\), and \(\text{min}_\text{sup}\).
Output: The frequent tree set \(T\).

1. if \(t \neq \text{Canonical Representative}(t)\)
2. then return
3.
4. \(C \leftarrow \emptyset\)
5. for every \(t'\) that can be extended from \(t\) by a step operation
6. do if \(\text{support}(t') \geq \text{min}_\text{sup}\)
7. then insert \(t'\) into \(C\)
8. for each \(t'\) in \(C\)
9. do Unordered Subtree Mining \((t', D, \text{min}_\text{sup}, T)\)
10. return

Figure 5: The Unordered Subtree Mining algorithm
Closed_Mining($D, \text{min}_\text{sup}, T$)

Input: A tree dataset $D$, and $\text{min}_\text{sup}$.
Output: The closed tree set $T$.

1. $m \leftarrow$ minimum depth of all trees in $D$
2. $t \leftarrow$ a tree with $m$ nodes and $m$ depth
3. **Closed_Subtree_Mining**($t, D, \text{min}_\text{sup}, T$)
4. return

**Figure 6:** The Closed Subtree Mining algorithm

Closed_Subtree_Mining($t, D, \text{min}_\text{sup}, T$)

Input: A tree $t$, a tree dataset $D$, and $\text{min}_\text{sup}$.
Output: The closed frequent tree set $T$.

1. if $t \neq \text{Canonical\_Representative}(t)$
2. then return
3. $C \leftarrow \emptyset$
4. for every $t'$ that can be extended from $t$ by a step operation
5. do if $\text{support}(t') \geq \text{min}_\text{sup}$
6. then insert $t'$ into $C$
7. do if $\text{support}(t') = \text{support}(t)$
8. then $t$ is not closed
9. if $t$ is closed
10. for each $t'$ in $C$
11. do **Closed_Subtree_Mining**($t', D, \text{min}_\text{sup}, T$)
12. return

**Figure 7:** The Closed Subtree Mining algorithm

of minimum depth of all the trees of the data set. This fact allows to speed up the mining process beginning with a nontrivial tree.

Figure 7 shows the pseudo code of **Closed_Subtree_Mining**. It is similar to the corresponding algorithm **Unordered_Subtree_Mining**, adding a closure test in lines 7-10.

2.3 Web search

There are a number of opportunities that naturally arise to help distributed search engines by means of closure-based data mining on structured data. Several of them admit a classification according to whether they analyze static (that is, slowly changing) or dynamic data.

Many parts of the web graph are rather static; links get added between existing pages but a more frequent change is the apparition of new pages, linking to some existing ones. Other items in the web that offer much higher dynamics are click streams and the evolution of contents of blogs and other forms of forums. The modus operandi for applying closure-based mining to analyze them is not obvious, but the potential of novel methods of analysis based on these approaches is high.

On dynamically evolving content, even the simplest model construction methods confront the
problem of the huge amounts of data to process, and the necessity of working on the basis of a mere window of most recent information pieces: this raises the problem of choosing the window size. Demonstrable algorithmic efficiency and demonstrably improved statistics and predictive power are not easy to obtain simultaneously, but our recent advances manage to get both. We will apply within the last year of DELIS these tools to detect trends on blogs (see the next subsection).

Concentrating on web search, an interesting application may come from polysemy. The different senses of many polysemic words may correspond in some sense with the different expectations of different persons searching an engine upon the same query word. For instance, a web query “apple” may refer to a brand of computers, to a fruit, or to the Beatles’ Apple Records music brand; or a person issuing a web query on “prams” may refer very likely to a mathematical model of computation if that person is interested in parallel computational models, or to perambulators if a child just showed up in the life of that person. In most cases, one specific sense is more popular than all the others (eg, “rolling stones”): the closure-based analysis of the navigation pattern may reveal quite clearly a difference in behavior, with highly branching, short path exploration when the results are unsatisfactory, versus low-branching, deep navigation trees of the persons who found a good approximation to the sense they were interested in.

These characteristic, low-branching, deep trees, as well as the high-branching, shallow trees of the failed attempts, will be very clearly discernible from the analysis of navigation patterns via unordered closed tree mining, and will be instrumental in building the user model that a distributed search engine will need to overperform centralized search engines.

### 2.4 Data Streams

However, there will be also the case where the information to be handled evolves fast. We have been studying a technique taken from the Mining Data Streams area, essentially contributed (among others) by a DELIS researcher (Ricard Gavaldà, UPC) with a view to apply it to change detection problems. Essentially, the context is some information that “passes by” quite fast, and obeys some sort of probability distribution that is approximated by some learning algorithm. Now, let’s accept that the probability distribution is not static. As it evolves, we may need to recompute parameters of the learning algorithm. A natural way to approach this problem is to keep a “window” of data, evaluating parameters on it, and adjusting its size to balance some resilience to small probabilistic oscillations, with an ability to follow up when a distribution change is actually happening. Note that both goals are contradictory. They require to expand the window when no changes seem to be taking place, but to shrink it very much to track evolving probabilities. We have a specific proposal for self-tuning the sliding window width.

Among several approaches along this line, ours is singled out by the properties of being fully self-adaptable, with no need of tuning external parameters by the part of the user, and, simultaneously, mathematically proved to offer quite good guarantees of correctness of the estimated parameters. Moreover, initial testing (with quite simple predictors such as Naive Bayes, KMeans, and Kalman filters) shows promising behavior. However, our experiments have not been made yet with web-related data.

One reason is that the first versions of this contribution were quite inefficient in time and memory. However, we have made a recent advance in the form of a new adaptive windowing algorithm (AD-WIN2) that is competitive with state-of-the-art alternatives and offers additional advantages. Along the year 4 of DELIS we expect to be able, for instance, to use this technology to track fashion topics in blog communities, and thus obtain knowledge that would guide the user modeling of a distributed search engine.
Lattice of closed trees for the six input trees in the top row

Figure 8:
References