Utilização de XML numa plataforma de Data Mining distribuído

Ruy Ramos, Carlos Adriano Gonçalves and Rui Camacho

LIACC, Rua de Ceuta 118 - 6^o 4050-190 Porto, Portugal
FEUP, Rua Dr Roberto Frias, 4200-465 Porto, Portugal

{ruyramos,rcamacho}@fe.up.pt, cadriano.goncalves@gmail.com

Resumo O processo de Extracção de Conhecimento em Bases de Dados (*Knowledge Discovery in Databases* - KDD) envolve a análise de extensas bases de dados e recurso a complexos algoritmos de análise de dados (*Data Mining*). Este processo requer, geralmente, recursos computacionais dedicados e de elevado custo o que reduz significativamente o número de utilizadores capazes de efectuar tais análises. Neste artigo apresentamos uma arquitectura baseada em computadores pessoais distribuídos numa rede de computadores de uma organização e que permite a realização de tarefas de KDD sem recursos computacionais dedicados e sem perturbar o funcionamento da organização. A arquitectura é denominada *Harvard* - ***HARV****esting* ***A****rchitecture of idle machines fo****R*** ***D****ata mining*. O Harvard utiliza uma linguagem de especificação e controlo de tarefas baseada em XML. A linguagem XML no caso do Harvard é imprescindível para a interoperabilidade entre os diferentes componentes do ambiente descrevendo claramente todos os aspectos da tarefa de *KDD* a ser executada de forma distribuída. Os resultados alcançados por diferentes nós do sistema são transcritos em XML, de modo a facilitar a apresentação ao utilizador do ambiente *Harvard* e ainda permitir integração com outros sistemas de extracção de conhecimento.

1 Introdução

O crescente uso das tecnologias da informação associado ao crescimento e desenvolvimento económico de vários sectores (empresarial, governo, comunidade científica e académica, entre outros) têm permitido que as organizações e a sociedade em geral reúnam uma enorme quantidade de dados e informações sistematicamente em Bases de Dados (BD) [?].

Do ponto de vista humano, torna-se praticamente impossível interpretar, analisar e obter resultados a partir de grandes quantidades de dados sem o auxílio de computadores. Torna-se muito difícil diferenciar informações verdadeiramente importantes e úteis em tão grande quantidade de dados [?,?]. Neste caso, a aplicação de técnicas automáticas de análise capazes de "extraír" informação útil torna-se inevitável dada a complexidade e a extensão das BD.

A aplicação de técnicas de *Data Mining* para auxiliar a análise de grandes quantidades de dados torne-se uma alternativa viável e aplicável, usando para isso os algoritmos de *Machine Learning* como *Inductive Logic Programming* (ILP) [?] e *Association Rules Discovery* (ARD) [?]. Uma desvantagem destes algoritmos é que são computacionalmente muito exigentes e este aspecto tem limitado o seu uso em aplicações de extracção de conhecimentos no mundo real.

Além disso, os algoritmos de *Machine Learning* foram originalmente construídos para abordagens sequenciais monolíticas pressupondo sempre um ambiente centralizado. É necessário que os dados estejam todos num único local para que as técnicas de *Data Mining* possam ser aplicadas. A quantidade de informação actualmente disponível é tão grande que centralizá-la se torna oneroso e extremamente complexo. Há casos mesmo em que os dados estão dispersos e não podem ser centralizados devido a limitações de rede, falta de espaço de armazenamento, questões de segurança, confidencialidade ou privacidade. Assim, há necessidade de adoptar novas abordagens para análise de dados nos seus respectivos locais de armazenamento.

Desenvolver técnicas de *Data Mining* para arquitecturas distribuídas tem sido um grande desafio da comunidade científica de *Knowledge Discovery in Databases - KDD*. O desenvolvimento de um novo estágio do KDD denominado de DPKDD (*Distributed and Parallel Knowledge Discovery in Databases*) está ligado principalmente pelo avanço em áreas como *storage, access e analysis* [?] e ao uso de tecnologias como *Grid Computing* [?].

A nossa proposta contempla uma arquitectura computacional dedicada ao processo de *Data Mining Distribuído* denominada de *Harvard - **HARV**esting **A**rchitecture of idle machines fo**R** **D**ata mining,* com os seguintes objectivos: fornecer ao analista de dados (utilizador) uma linguagem simples mas poderosa para especificar as tarefas de análise de dados (*KDD*); viabilizar a utilização optimizada de computadores pessoais quando ociosos em prol de um processamento cooperativo e útil; construir uma plataforma que possa ser usada independentemente dos sistemas operativos instalados (*Linux* ou *Windows*) na organização; permitir análise de dados de forma distribuída (computação distribuída e acesso a dados fisicamente distribuídos); permitir o uso de diferentes algoritmos de *Machine Learning* e suas implementações (ferramentas) sem qualquer alteração da plataforma, sendo portanto independente da ferramenta usada pelo analista; e permitir recuperação de fallhas do próprio sistema (tolerância a falhas).

Além dos objectivos apresentados, o *Harvard* apresenta as seguintes vantagens: o utilizador pode facilmente definir o processo de análise de dados mais adequado para suas necessidades; com este ambiente pode-se expandir o uso da análise de dados a um vasto leque de organizações já que os custos envolvidos são bastante baixos, as máquinas usadas são as mesmas que servem para as tarefas rotineiras. E além disso, o processamento computacional *Harvard* não perturba o trabalho normal da organização uma vez que só usa recursos computacionais desocupados ou ociosos.

O *Harvard* permite ao utilizador descrever cada tarefa do processo de KDD através da linguagem de anotação XML (*Extensible Markup Language*)[?] e especificar o fluxo de sua execução (*workflow*) numa linguagem descritiva de sintaxe simples mas no entanto poderosa. O sistema corre num *Servidor* com uma colecção ilimitada de estações *Cliente*. Tanto o *Servidor* como os *Clientes* são programados em *Java*. Os *Clientes* podem aceder aos dados directamente de uma base de dados (usando JDBC) e todo *software* necessário de análise de dados, via protocolo HTTP.

Em termos de trabalhos correlatos destacamos o Condor [?] da Universidade de Wisconsin-Madison e o BOINC (*Berkeley Open Infrastructure for Networking Computing*) [?]. O ambiente Condor caracteriza-se por gerir um ambiente composto de vários servidores e/ou *cluster*, como o "Beowulf". É basicamente um ambiente *Grid Computing* para gerir a capacidade de processamento de recursos computacionais de um "campus" ou de uma organização. Tem como características principais a gestão de *jobs*, definição de políticas de uso do ambiente distribuído e gestão e monitoramento dos recursos computacionais. Há uma variante denominada Condor-G [?] que facilita a integração com ambientes *Grid Computing* baseados no *Globus Toolkit* [?]. Por outro lado, o BOINC é uma arquitectura computacional distribuída que utiliza recursos computacionais registados voluntariamente a partir da Internet. É uma plataforma que permite aos cientistas gerir recursos computacionais públicos disponíveis na Internet em benefício de projectos científicos de grande dimensão. Alguns projectos como o *SETI@home*, *Folding@home* utilizam a plataforma BOINC. A arquitectura é composta por servidores de projectos que operam suas próprias aplicações e bases de dados, e partilham processamento voluntário disponível a partir de estações ligadas à Internet que "doam"ciclos de processamento por intermédio de aplicações instaladas para esta finalidade.

O artigo está organizado da seguinte forma: na Secção 2 descrevemos como as tarefas do processo de KDD podem ser especificadas por meio de uma linguagem simples mas poderosa usando como base XML; na Secção 3 apresentamos a arquitectura do *Harvard* e seu funcionamento; e por fim apresentamos as conclusões e trabalhos futuros na última Secção.

2 O processo de Extracção de conhecimento

Segundo Fayyad [?], *Data Mining* é o núcleo principal de um processo mais amplo denominado de *Knowledge Discovey in DataBases* (KDD), ou Descoberta de Conhecimento em Bases de Dados, conforme apresentado na Figura ??.

Figura 1. Processo de Extracção do Conhecimento

Knowledge Discovey in DataBase (KDD) é um processo não trivial de identificar, validar e reconhecer padrões de dados que possam prover informação

válida gerando conhecimento sobre uma determinada Base de Dados. Os dados referem-se a representação de factos e os padrões são definidos por uma expressão que descreve um subconjunto desses mesmos dados [?].

De acordo com *Fayyad* [?] o processo de KDD é composto pelas fases:

- preparação dos dados;
- selecção de dados;
- pré-processamento de dados;
- transformação de dados;
- *data mining*;
- interpretação e avaliação do conhecimento.

2.1 A fluxo do processo de análise de dados

O *Harvard* inicia o processo de análise de dados lendo, de um ficheiro, a especificação do fluxo do processo (*workflow*) juntamente com a descrição de cada uma das tarefas do processo KDD. O fluxo do processo é representado por um grafo com dois tipos de nós: sequenciais e paralelos. Cada nó regista o conjunto de tarefas a serem executadas, representadas e designadas como UT (Unidades de Trabalho). Na Secção 3, que descreve a arquitectura do *Harvard*, as UT são descritas com maior detalhe.

No caso do nó sequencial, as tarefas deverão necessariamente ser executadas em ordem de precedência devido às suas interdependências no processo de análise de dados. Por outro lado, os nós paralelos especificam tarefas que podem ser executadas em simultâneo.

Na Figura **??**, apresentamos um exemplo de descrição de um processo KDD. A figura mostra ainda em detalhe um conjunto de tarefas definidas $T1$ ... $T15$ que abrangem todo o processo, desde o pré-processamento, seleccionando os atributos mais relevantes, até à consolidação dos resultados ($T15$). A descrição de cada tarefa Tn está codificada em XML em ficheiros distintos que serão processados pelo módulo Gestor de Tarefas (GT) do *Harvard*, conforme detalhado na Secção 3.

2.2 A especificação das tarefas

A linguagem de especificação das tarefas, conforme ilustrado na Figura **??**, parte inicialmente da sintaxe básica da XML. A partir deste contexto são definidas variáveis com seus respectivos parâmetros que posteriormente serão interpretadas pelo *Havard* no seu módulo que trata da especificação da tarefa - o Gestor de Tarefas (GT).

Na especificação de cada tarefa o utilizador (analista de dados) deverá fornecer em detalhe toda a informação relevante. Isso inclui descrever entre outros ítens:

- local e nome do ficheiro que contém os dados para análise;
- algoritmo(s) a ser(em) usado(s) com respectivos parâmetros de análise;

```
# This is the Tasks control description
# using the Task Control Language (.tcl)

seq
  T1  # make a 70%/30% train/test set

  par # execute the tasks in parallel
    seq
     T2  # get dataset without Att1
     par
       T3  # eval dataset without Att1 using m = 10
       T4  # eval dataset without Att1 using m = 50
     endpar
    endseq

    seq
     T5  # get dataset without Att5
     par
       T6  # eval dataset without Att2 using m = 10
       T7  # eval dataset without Att2 using m = 50
     endpar
    endseq

  barrier T[3-4], T[6-7] # wait for all tasks to finish

  T8  # choose the best set of attributes
  T9  # make the 5-fold CV blocks

  par
    T[10-14] # do each CV i

  barrier T[10-14]  # wait for all CV folds

 T15 # run with all data to produce the final theory

endseq
```

Figura 2. Descrição do processo KDD em tarefas sequencias e paralelas.

- nome da aplicação que implementa o algoritmo escolhido (*e.g.*: C4.5)
- ferramentas de pré-processamento a ser usada (*e.g.*: aplicação de *script perl* para eliminar ou consolidar atributos);
- local e formato de representação dos resultados.

2.3 O uso da linguagem XML no *Harvard*

A linguagem XML usada no *Harvard* permite que o utilizador possa especificar claramente o processo de KDD pois "provê um conceito para descrever, armazenar, permutar e manipular dados estruturados" [?,?].

Decorre que a linguagem XML se caracteriza por permitir estruturar a informação de forma hierárquica, facilitar a edição devido à sintaxe simples (qualquer editor de texto pode ser usado), e permitir a fácil compreensão dos dados estruturados, já que não requer nenhuma ferramenta sofisticada para visualização. No caso do *Harvard* os dados estruturados em XML permitem a fácil interpretação pelos diferentes módulos da plataforma, e também a geração dos resultados para posterior integração, seja com outras ferramentas de análise de dados, ou para o armazenamento em bases de dados.

Considerando que o *Harvard* é desenvolvido em *Java*, o processamento de ficheiros em XML é facilitado devido as bibliotecas (*classes*) disponíves. E considerando as características da linguagem XML, alterações em ficheiros XML não significam necessariamente alterações em código de programação Java. Pode-se,

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE tasks SYSTEM ''tasks.dtd'' >
#  Run C4.5 on a data set stored in a Data Base and return the
#  in a "results table" of the same DB
#####      Task Description Language (.tdl) file         #####
<workunit>
 <id> T1 </id>
                      ### data ###
   <fetch-data>
      <method> jdbc     </method>
      <server> dbserver </server>
      <user>   dmuser   </user>
      <db>     kdd99    </db>
      <password> _______</password>
      <db-access>
       <source-data>
          <query> select * FROM data LIMIT 5000 OFFSET 100 </query>
          <file>  kdd99.data </file>
      </source-data>
      ...
     </db-access>
   </fetch-data>
                    ### source code ###
  <fetch-code>
   <source-code>  # C4.5 code to construct the Decision Tree
     <getMethod> http </getMethod>
     <url> http://www.fe.up.pt/~rcamacho/c4.5 </url>
   </source-code>
   ...
  </fetch-code>
                  ### sub-tasks execution ###
 <execution>
    <results-storage>
      <method> jdbc </method>
      <server> dbserver </server>
      <user>   dmuser   </user>
      <db>     kdd99    </db>
      <password> _______ </password>
    </results-storage>
### sub-task 1 ###
 <subtask>
   <exec-mode>   noninteractive           </exec-mode>
   <exec-command> c4.5 -f kdd99 -m 100 -u </exec-command>
   <results-file> c45.output              </results-file>
   <exec-time>  30 </exec-time>
 </subtask>
### sub-task 2 ###
   ...
 </execution>
                  ### required resources ###
   <resources>
     <hd> 10 </hd>
     <ram> 0.5 </ram>
     <opsystem> linux </opsystem>
   </resources>
</workunit>
```

Figura 3. Descrição em detalhe de uma tarefa.