Um método para identificar regras de associação utilizando somente os logs de acesso de um servidor web

Fernando José Braz[1], **Ivo Marcos Riegel**[1], **Valter Luís Estevam Junior**[2]

[1]Instituto Federal Catarinense - Câmpus Araquari
Caixa Postal 21 – 89.245-000 – Araquari – SC – Brasil

[2]Instituto Federal Catarinense - Câmpus Videira
89.560-000 – Videira – SC – Brasil

{fernando.braz,ivo.riegel}@ifc-araquari.edu.br, valter@ifc.edu.br

Abstract. *Patterns of use of a web site can be used to personalize services, identify consumer profiles and customize navigation. To identify these patterns using chains of navigation contained in access log files of web servers is necessary to segment the records in user sessions. As the HTTP protocol does not store this information, some strategy must be traced to join records into sessions.*
This article describes a method for identifying association rules between pages of a web site that uses as a source data, only the contents of access log files of web server.

Resumo. *Padrões de uso de um site podem ser usados para personalizar os serviços, identificar o perfil de consumidores e personalizar a navegação. Para identificar esses padrões usando cadeias de navegação contidas nos arquivos de log de acessos de servidores web é necessário segmentar os registros em sessões de usuário. Como o protocolo HTTP não armazena essas informações, deve ser traçada alguma estratégia para agrupar registros em sessões.*
Este artigo descreve um método para identificar regras de associação entre páginas de um web site que utiliza como fonte de dados somente o conteúdo dos arquivos de logs de acesso de um servidor web.

1. Introdução e Motivação

A mineração de dados na *web* possui três vertentes: *web content mining*, *web structure mining* e *web usage mining*. O trabalho apresentado neste artigo está inserido no contexto da *web usage mining* a qual tem por objetivo extrair padrões de utilização de ambientes *web* a partir do processamento de cadeias de interação de usuários com *web sites*.

A extração de padrões de utilização de um *web site* pode ser utilizada para:

- **personalizar os serviços prestados:** como exemplo, se os usuários que acessam o FAQ também acessam o serviço de "Fale conosco" então os serviços poderiam aparecer numa mesma página.
- **identificar perfis de consumidores:** num sistema de comércio eletrônico, identificar que usuários que acessam páginas sobre computadores também acessam páginas sobre impressoras pode auxiliar a traçar estratégias de oferta combinada, ou seja, na compra do computador X leve também a impressora Y.

- **projetar web sites mais práticos:** num ambiente virtual de aprendizagem, saber que os usuários que acessam um curso A também acessam um curso B mostra que é interessante adicionar *links* para os demais cursos do aluno na página do curso que ele estiver acessando.
- **melhorar estratégias de cache:** muitos servidores de *proxy* trabalham também com mecanismo de *cache*, ou seja, armazenam neles as requisições mais frequentes de modo a reduzir a carga sobre o servidor *web* e diminuir o tempo de espera pelo conteúdo. Se um padrão para um *web site* considera que quando o usuário acessa as páginas A e B ele também acessa a página C então, quando o usuário acessar as páginas A e B a página C já poderia ser carregada em *cache* antes mesmo que uma requisição a ela seja feita. Isso reduziria a latência quando a página C for solicitada.

Para identificar padrões de uso é necessário definir e/ou implementar mecanismos para coletar as trilhas de navegação. Estratégias possíveis para isso seriam: configurar o servidor *web* para armazenar arquivos de *log*; projetar o sistema para armazenar dados em *cookies* ou incluir em cada página *scripts* para registrar os cliques do usuário. Tais técnicas podem ser utilizadas de modo isolado ou combinado e podem ser mais precisas em reconstruir as sequências de navegação.

O método de coleta de dados mais prático sob o ponto de vista de implementação é o mecanismo de *log*. Os servidores *web* utilizados atualmente já trazem essa ferramenta por padrão. Os arquivos de *log* de acessos gerados contém informações sobre cada uma das requisições que os usuários do *web site* solicitaram. Cada linha do arquivo corresponde a uma requisição. Um exemplo das informações armazenadas pelo *log* de acessos do servidor Apache é tal como segue:

cti.videira.ifc.edu.br:80 201.14.186.174 - - [07/Oct/2012:13:31:47 -0300] ”GET /moodlecti/course/view.php?id=58 HTTP/1.1”200 9639 ”http://cti.videira.ifc.edu.br/moodlecti/Mozilla/5.0 (Windows NT 6.1; WOW64; rv:15.0) Gecko/20100101 Firefox/15.0.1”

Os campos e a ordem em que aparecem pode variar conforme as configurações que o administrador definir. Para o exemplo tem-se os campos: servidor com a respectiva porta, endereço IP do usuário, data e hora da requisição, método de requisição com a página requisitada, protocolo de comunicação e versão, código de retorno, tamanho do arquivo transferido, página de origem da requisição e por último os dados de navegador e sistema operacional também chamado de user-agent.

Conforme mostra o exemplo de registro de *log*, não há informações sobre o início e o final das sequências de interação, apenas o registro de que ocorreu uma solicitação. Sendo assim, objetivo deste trabalho é apresentar uma abordagem para identificar regras de associação a partir da segmentação de sessões de usuário utilizando unicamente arquivos de *log* de acessos.

O artigo está organizado da seguinte maneira: na seção 2 faz-se um breve resumo sobre as estratégias utilizadas para identificar sessões e sobre algoritmos de regras de associação. Na seção 3 é discutido o modelo proposto. Na seção 4 são discutidos os procedimentos adotados juntamente com os resultados obtidos. Depois, na seção 5, são apresentadas as conclusões e as considerações finais.

2. Trabalhos relacionados

Para encontrar padrões de uso confiáveis utilizando arquivos de log de acessos deve-se identificar as sessões de cada usuário do sistema. Tal procedimento não é trivial visto que o protocolo utilizado para navegação na *web* (HTTP) não é orientado nem a conexão e nem a sessão. Após identificadas as sessões, um procedimento possível é aplicar algoritmos de mineração de dados que retornem regras de associação entre as páginas solicitadas. Nas subseções seguintes são abordadas algumas heurísticas utilizadas para segmentar sessões e os algoritmos de regras de associação no contexto da Web Usage Mining.

2.1. Heurísticas para Segmentação de Sessões

Existem duas heurísticas principais que são utilizadas para identificar sessões de usuário:

- **Heurísticas baseadas em tempo:** que podem ser de dois tipos
  - **Tempo total de sessão:** considera o tempo total desde a primeira requisição do usuário até sua última interação. Este tempo não deve passar um valor prévio estipulado.
  - **Tempo entre requisições:** utiliza o intervalo de tempo entre duas requisições do usuário ao web site para considerar o abandono de uma sessão. Caso o tempo entre uma requisição e outra seja superior a um valor estipulado considera-se que o usuário encerrou a sessão antiga e está iniciando uma nova sessão.
- **Heurística baseada na estrutura do site:** utiliza a topologia do site para saber se existe conexão entre as requisições de um mesmo usuário. Quando entre requisições consecutivas houver conexão (*link* de uma para outra) então pode-se considerar como pertencente à mesma sessão. Caso contrário, inicia-se uma nova sessão [Brusso et al. 2000].

De acordo com [Chitraa and Davamani 2010], se o tempo entre requisições exceder a 10 minutos então tem-se uma nova sessão, contudo, isso está em desacordo com o trabalho de [Catledge and Pitkow 1995] que aponta um tempo de 25,5 minutos. Adicionalmente, os trabalhos de [Beauvisage 2000], [Chaofeng 2004] e [Ypma and Heskes 2002] também utilizam o tempo de 30 minutos como timeout.

Para identificar sessões utilizanto *timeout* deve-se primeiro organizar os registros separando-os por usuário. Sempre que uma requisição é feita a um servidor *web* o endereço IP do usuário é armazenado juntamente com as demais informações já apresentadas anteriormente. Quando mais de um usuário está sob um mesmo ponto de acesso com a *Internet* as suas requisições são armazenadas com o mesmo endereço IP. Uma maneira de diferenciá-los é utilizar o campo *user-agent* do arquivo de *log* de acessos.

Após identificados os diferentes usuários é necessário ordenar os registros por usuário (*IP+User-Agent*) e depois por data e hora. Então, calcula-se o tempo entre as requisições de cada um dos usuários. Um procedimento formal para identificar sessões por tempo entre requisições (timeout) está mostrado no algoritmo 1.

2.2. Regras de Associação em Web Usage Mining

As regras de associação dentro da *web usage mining* são utilizadas, na maioria das vezes, para estabelecer relações entre páginas solicitadas a um servidor *web*. Com base em

***Algoritmo 1**: Identificar sessões por timeout*

```
L é uma lista de listas de requisições de usuários
LRU é uma lista de registros de usuário ordenados por tempo,
T é o timeout, LS é uma lista de sessões
R é um registro do arquivo de log de acessos e S é uma sessão de usuário
Entrada: L, T
Saída: LS
para cada LRU em L faça
    criar sessão S;
    para cada R em LRU faça
        se S está vazio então
            adicionar R a S;
        senão
            se R.tempo > T então
                adicionar S a LS;
                criar sessão S;
                adicionar R a S;
            senão
                adicionar R a S;
se S não está vazio então
    adicionar S a LS;
```

regras de associação pode-se afirmar, por exemplo, que, quando um usuário acessa uma página a e uma página b ele também acessa uma página c. Uma ação possível com esse conhecimento seria colocar um *link* em posição de destaque direcionando o usuário da página a para a página c ou da página b para a página c sempre que o usuário acessar as páginas a e b. Porém, antes de fazer isso é necessário saber qual a frequência da regra dentre todas as sessões e qual a garantia de que, quando o usuário acessa a e b ele também acessa c.

A frequência de um conjunto de itens A é chamada suporte, definido como $sup(A)$ que representa a porcentagem de transações da base de dados que contém os itens de A. Seguindo o mesmo raciocínio o suporte de uma regra de associação de $A \Rightarrow B$ é dado por $sup(A \cup B)$. A garantia ou confiança de uma regra, dada por $conf(A \Rightarrow B)$, representa o percentual de transações que contém B dentre as transações que contém A. Pode-se calcular a $conf(A \Rightarrow B)$ dividindo o $sup(A \Rightarrow B)$ pelo $sup(A)$ [Witten and Frank 2005].

Para encontrar regras de associação é necessário determinar todos os conjuntos de itens que possuem o suporte maior ou igual a um valor mínimo estipulado. Tais conjuntos são chamados de conjuntos de itens frequentes. Para cada conjunto de itens frequentes deve-se gerar combinações dos itens do conjunto distribuídos entre antecedente e consequente. Como exemplo, se temos um conjunto de itens frequente $I = \{A, B, C\}$, as combinações entre eles distribuídas entre antecedente e consequente seriam: $(A \Rightarrow B, C)$, $(B \Rightarrow A, C)$, $(C \Rightarrow A, B)$, $(A, B \Rightarrow C)$, $(A, C \Rightarrow B)$ e $(B, C \Rightarrow A)$. Resta calcular a confiança de cada regra e verificar se é superior a um valor mínimo estipulado. Para todas as regras o valor do suporte é igual a $sup(I)$,

3. Modelo Proposto

O objetivo da segmentação de sessões em arquivos de *log* é reconstruir, do modo mais fiel possível, as sequências de interação do usuário do sistema. Conforme apresentado na seção 2, uma das heurísticas baseadas em tempo utiliza o *timeout* para definir um ponto de término das interações. Contudo, ela nada permite afirmar sobre o ponto de início além de que é o primeiro registro com tempo entre requisições acima do *timeout*. Em sistemas onde isso não seja relevante os resultados se mostram satisfatórios, tal como nos trabalhos de [Beauvisage 2000], [Chaofeng 2004] e [Ypma and Heskes 2002]. Contudo, em sistemas com controle de acesso pode ser interessante levar em consideração o ponto de início da interação ou os pontos de início possíveis.

O modelo proposto será discutido utilizando o exemplo da figura 1 que mostra o recorte de um conjunto hipotético de requisições. A primeira linha mostra a sequência de páginas solicitada, na segunda linha aparece o tempo transcorrido desde a última requisição e a terceira linha mostra as sessões que o usuário realmente acessou. Aplicando o algoritmo 1 construiu-se a quarta linha.

Páginas Acessadas Pelo Usuário	k	a	d	c	b	i	a	d	a	d	j	d	k
Tempo entre requisições (min)	0	20	1	2	0,5	50	80	1	2	1	3	2	4
Situação real	s1	s2				s3	s4						
Timeout (30 minutos)	s1					s2	s3	s4					
Frequência + Timeout (30 minutos)	d	s1				d	s2						

Figura 1. Exemplo de segmentação de sessões utilizando os algoritmos 1 e 2

Pode-se observar no exemplo que o usuário iniciou 4 sessões. Destas, duas contém apenas uma requisição. As páginas *k* e *i* poderiam ter sido solicitadas digitando a url no navegador. O algoritmo que utiliza somente o *timeout* foi capaz de identificar 4 sessões de usuário, sendo que duas delas com apenas uma requisição. Das sessões identificas com mais de uma requisição, numa está faltando uma requisição e na outra há uma requisição a mais do que deveria.

O modelo proposto para encontrar regras de associação pressupõe que a identificação de sessões de usuário possa ser feita tomando as páginas mais frequentes como relacionadas com o *login*, que é o início natural de uma sessão de usuário num sistema com controle de acesso. Na sequência estão enumerados os passos do modelo:

1. **Filtrar registros não significativos**
   Sempre que se requisita uma página que contenha imagens são armazenadas: a requisição à página e mais uma requisição para cada uma das imagens. O mesmo vale para vídeos, áudio, animações, etc. Se forem utilizados *scripts* externos à página, estes também serão requisitados e registrados no *log*. Deste modo é necessário remover estas informações se for assumido que o interesse do usuário é pelo acesso à página e não necessariamente pelas imagens ou outros conteúdos multimídia que são carregados com ela.
2. **Organizar os registros**
   Uma vez aplicados os filtros, os registros devem ser ordenados primeiro pela concatenação do endereço IP com o campo *user-agent*. Depois por data e hora. Feito isso, deve-se separar os registros em listas de requisições. Uma para cada usuário agrupando-as numa lista de listas (L) usada como entrada no algoritmo 2.

Na sequência, deve-se percorrer todas as listas de L e em cada uma delas percorrer os registros calculando o tempo entre as requisições.

3. **Calcular a frequência de cada uma das páginas do site:**
   Considerando o exemplo deve-se identificar as requisições sem levar em conta as repetições $R_i = \{a, b, c, d, i, j, k\}$. Depois deve-se calcular a frequência de cada página de R_i usando a equação 1. Nela r_i é uma requisição qualquer tomada no conjunto R_i, $\sum r_i$ é o número de vezes que r_i ocorre em $R = \{k, a, d, c, b, i, a, d, a, d, j, d, k\}$ e $tam(R)$ é a quantidade de itens de R.

$$F(r_i) = \frac{\sum r_i}{tam(R)} \tag{1}$$

   Aplicando este procedimento aos dados do exemplo, tem-se as páginas a e d como as mais frequentes. Por fim, deve-se definir a quantidade de páginas mais frequentes a ser utilizada e então adicioná-las à uma lista de requisições frequentes (LF) que será usada como entrada no algoritmo 2.
4. **Estipular um valor de *timeout***
   Basta definir o tempo que será utilizado para considerar uma sessão como encerrada. O valor típico é 30 minutos conforme já discutido. Este tempo também é um valor de entrada para o algoritmo 2.
5. **Segmentar as sessões utilizando o algoritmo 2**.

***Algoritmo 2**: Sessões por frequência de requisições combinada ao timeout*

```
L é uma lista que contém listas de requisições de usuário
LF é uma lista de requisições mais frequentes, LS é uma lista de sessões
R é uma requisição do arquivo de log de acessos, S é uma sessão de usuário
T é o tempo de timeout
Entrada: L, LF, T
Saída: LS
para cada LRU em L faça
    criar sessão S;
    para cada R em LRU faça
        se S está vazio então
            se R está contido em LF então
                adicionar R a S;
        senão
            se R.tempo > T então
                adicionar S a LS;
                criar sessão S;
                se R está contido em LF então
                    adicionar R a S;
            senão
                adicionar R a S;
    se S não está vazio então
        adicionar S a LS;
```

6. **Aplicar o algoritmo de mineração**: Nesta etapa, com as sessões já idetificadas basta aplicar o algoritmo de mineração tal como explicado na sessão 2.

Ao aplicar o algoritmo 2 ao conjunto de dados do exemplo chega-se à última linha da figura 1. Nota-se que há duas requisições que são descartadas (assinaladas com um d). Elas coincidiram com as requisições das sessões s1 e s3 da situação real. O ponto interessante a observar é que as sessões destacadas com sombreamento contém os mesmos registros. O que dá um bom indício de que a estratégia permite reconstruir as sessões originais. Os experimentos para verificar esta hipótese são descritos na próxima seção.

4. Experimentos e Resultados

O modelo proposto e o algoritmo de *timeout* foram aplicados aos dados do *log* de acessos do servidor *web* onde está instalado uma versão do ambiente virtual de aprendizagem *Moodle*. O ambiente é utilizado pelos professores de informática e pelos alunos dos Cursos Técnicos de Nível Médio nas modalidades concomitante, subsequente e integrado, Curso Superior de Bacharelado em Ciência da Computação e Curso de Pós-Graduação Lato Sensu em Desenvolvimento *Web* do IF Catarinense - Câmpus Videira.

Na etapa 1, executada para ambos os algoritmos, foram extraídos 48 arquivos de *log* com um total de 770 MB e mais de 2 milhões de registros.

Como o moodle é um ambiente utilizado para dar suporte à atividade de aprendizagem a distância é normal que o mesmo contenha arquivos de inúmeras extensões (.doc, .odt, .ppt, .pdf, .PNG, .xls, etc), correspondentes aos materiais postados nas disciplinas. Não era interessante investigar regras que tratassem de um arquivo específico postado numa disciplina, mas sim, investigar se um usuário acessa um recurso numa disciplina e o que mais ele acessa quando o faz. O mesmo vale para os cursos. Não é o objetivo identificar associações com um curso específico, mas se a página de cursos era acessada. Então, os parâmetros do método de requisição GET foram removidos. Ao fazer isso as extensões dos arquivos também foram descartadas. Outros filtros realizados foram remover qualquer requisição com código de retorno 404 e excluir as requisições vindas dos endereços 200.135.55.4 e 200.135.55.8. Isso porque eram oriundas da rede interna do IF Catarinense e por conta disso não era possível identificar os usuários. Nem mesmo concatenando o IP com o user-agent.

Ao final da etapa 1 restaram exatamente 239710 registros datados de 18/01/2012 a 18/12/2012, totalizando 335 dias. Então, partiu-se à etapa dois, também para ambos os algoritmos. Nela os registros foram ordenados primeiramente pelo resultado da concatenação do endereço IP com as informações do campo *user-agent* e posteriormente pela data e hora da solicitação. Como resultado foram identificados 9892 usuários. Este número não tem relação direta com o número de usuários cadastrados no moodle, visto que um usuário com conta no sistema poderia ser reponsável por vários acessos com diferentes endereços IP e *user-agents*. Como exemplo, supondo que o usuário atualizasse seu navegador quatro vezes no período investigado, somente para os acessos que faria a partir de seu computador ele seria responsável por 4 usuários distintos.

A etapa 3 é exclusiva para o algoritmo de frequência de páginas + *timeout* e obteve como resultado, entre outros valores, os contidos na tabela 1. Na qual há três requisições que sozinhas concentram pouco menos de 2/5 do total de requisições. A primeira é a página inicial do ambiente onde há os campos para efetuar o *login*. A segunda é uma página utilizada também para efetuar o *login* que pode ser acessada a partir

de "/moodlecti/" ou quando expira a sessão do usuário controlada pelo *script* PHP e a terceira á a página de visualização de curso. Na penúltima linha está a página de *logout*, que possui uma frequência de 1,3% do total de requisições. Um valor baixo comparado com as páginas do *login*, o que evidencia que os usuários não costumam encerrar suas sessões utilizando a operação de *logout*.

Tabela 1. Páginas do moodle com os percentuais de requisições obtidos utilizando a equação 1. As páginas com frequência inferior a 1,5% foram suprimidas.

Páginas	Frequência	%
/moodlecti/	32.466	13,5
/moodlecti/login/index.php	32.024	13,4
/moodlecti/course/view.php	27.629	11,5
/administrator/index.php	17.465	7,3
/moodlecti/mod/resource/view.php	8.365	3,5
/moodlecti/mod/assignment/view/moodlecti/php	7.727	3,2
- - -	- - -	- - -
/moodlecti/login/logout.php	3.069	1,3
Total de Registros	**239.710**	**100**

Como apenas três itens correspondem a pouco menos de 2/5 das requisições, ficou definido que este seria o tamanho da lista de itens frequentes.

Na etapa 4 (definição de timeout), que é a mesma para ambos os algoritmos. Foram selecionados os valores de 2, 5, 10, 20 e 30 minutos. Os *timeouts* de 10 e 30 minutos foram escolhidos por serem recorrentes na literatura conforme discutido na seção 2. O *timeout* de 20 minutos foi escolhido por ser um valor intermediário entre 10 e 30 minutos e os *timeouts* de 2 e 5 min foram selecionados para verificar o comportamento dos algoritmos com tempos inferiores a 10 minutos.

O próximo passo foi identificar as sessões de usuário, etapa 5. Os algoritmos 1 e 2 foram executados para os tempos de *timeout* escolhidos e os resultados estão nas tabelas 2 e 3.

Tabela 2. Dados gerais sobre sessões de usuário utilizando o algoritmo 1

Timeout (min)	Sessões	Sessões/dia	Média req.
2	47.519	141,8	4,2
5	46.358	138,4	4,4
10	45.814	136,8	4,5
20	45.023	134,4	4,7
30	44.612	133,2	4,8

Conforme mostra a tabela 2 os valores são próximos. Observa-se que quanto menor o tempo para abandono de sessões maior é o número de sessões e menor é o tamanho das sessões. Contudo, a diferença entre o maior e o menor número de sessões indentificadas é de 2907 sessões, ou 6,1% da maior quantidade de sessões. Conclui-se que não há diferenças significativas quanto ao número de sessões. Sob o ponto de vista do tamanho médio, os resultados sugerem que o número médio de requisições por sessão

Tabela 3. Dados gerais sobre sessões de usuário utilizando o algoritmo 2

Timeout (min)	Sessões	Sessões/dia	Média req.
2	17.181	51,3	8,9
5	15.714	46,9	10,3
10	14.822	44,2	11,4
20	14.058	42,0	12,4
30	13.688	40,9	12,9

de usuário é de 5. Um número muito reduzido se considerar-se, como exemplo, uma situação típica dentro do sistema que é acessar um recurso dentro de um curso. Para isso seria necessário acessar exatamente 5 páginas, o que o resultado sugere é que, na média os usuários acessam o ambiente para consumir não mais que essas 5 páginas. Sob o ponto de vista da quantidade diária de sessões, o valor próximo a 137 sessões/dia sugere que cada usuário estabelece 1 sessão por dia fora da rede do IF Catarinense. E isto incluindo sábados, domingos, feriados e férias.

Por outro lado, ao testar os mesmos valores de *timeout* combinado com a estratégia de páginas frequentes observou-se que para o tempo de 2 min foram produzidos dados discrepantes comparados aos demais intervalos de tempo testados. Neste caso foram identificadas 1467, ou 8,5%, de regras a mais do que para o tempo de 5 min. Fora isso, as sessões se mostram mais homogêneas para os tempos de 10, 20 e 30 minutos, para os quais a diferença no número de sessões é de 1134, ou 7,6%, para uma variação de 20 minutos no tempo.

Analisando o tamanho médio, vê-se números que refletem que os usuários, na média, utilizam o sistema acessando uma quantidade maior de recursos se comparado com a estratégia de timeout, 11 contra 5. O número médio de sessões por usuário e por dia fica próximo a 0,5.

Ampliando a análise, dado que um usuário só pode acessar um conteúdo efetuando o *login* ou em "/moodlecti/" ou em "/moodlecti/login/index.php" e que, se uma das formas de chegar em "/moodlecti/login/index.php" é estando na página "/moodlecti/" clicar sobre um dos cursos do menu lateral ou sobre um *link* para *login* no canto superior direito tem-se que na maioria das vezes ambas as páginas ocorrem juntas. Então, o que ocorre com as demais requisições a "/moodlecti/"? Elas deveriam ser em número muito superior pois também é a página inicial do usuário.

Quando o usuário está navegando, é comum o uso do botão voltar. Ao clicar sobre ele o navegador reconstrói a página anterior em tela sem efetuar uma nova requisição ao servidor *web*. Assim, o acesso a esta página não fica registrado no *log* de acessos. Uma abordagem para resolver este problema foi apresentada por [Chitraa and Davamani 2010] e faz uso de complemento de caminho. O objetivo do complemento de caminho é preencher lacunas na trilha de navegação de um usuário tomando por base as requisições que são atendidas pelo servidor de *proxy* e dados sobre padrões de acesso que podem ser extraídos do lado do cliente usando, por exemplo, *cookies*.

Isso não foi possível fazer visto que não há um servidor de *proxy* no ambiente onde o *Moodle* está instalado e não foram implementados e nem adaptados *scripts* ao mesmo. O impacto que a falta desta informação tem sobre os padrões encontrados não

é significativo visto que não se está investigando as sequências mais prováveis mas, as associações entre as páginas sem importar a ordem em que elas são solicitadas.

A figura 2 apresenta as conexões entre as principais páginas do *Moodle*.

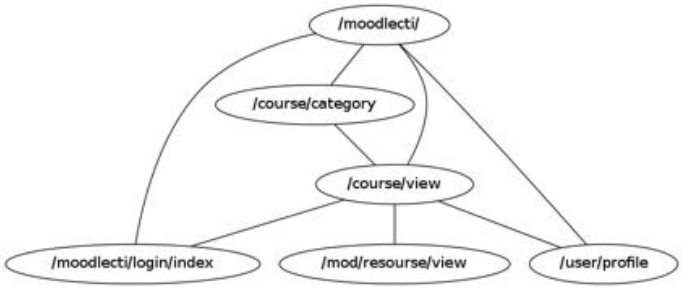

Figura 2. Fragmento da topologia do ambiente *Moodle* do IF Catarinense - Câmpus Videira.

Por fim, etapa 6, o algoritmo de mineração foi aplicado sobre cada um dos conjuntos de sessões das tabelas 2 e 3. Os valores utilizados para o suporte e confiança mínimos iniciaram elevados pelo objetivo inicial de filtrar regras com alto suporte e alta confiança. Como a estratégia de *timeout* não estava produzindo resultados, os valores foram sendo sistematicamente reduzidos até chegar em 10% para ambos os parâmetros. A quantidade de regras geradas está presente na tabela 4 e algumas regras nas figuras 5 e 6.

Tabela 4. Quantidade de regras de associação encontradas para vários valores de suporte e confiança mínimos

Parâmetros		**Timeout (min)**					**Req. freq. + Timeout (nº pts - min)**				
Sup. Mín.	Conf. Mín	2	5	10	20	30	3 - 2	3 - 5	3 - 10	3 - 20	3 - 30
80	80	0	0	0	0	0	0	0	0	0	0
60	60	0	0	0	0	0	2	6	10	10	10
40	40	0	0	0	0	0	10	10	10	10	10
20	20	4	4	4	4	4	10	20	24	96	134
10	10	6	6	6	8	8	146	160	170	170	170

Tabela 5. Regras produzidas utilizando o algoritmo 1 com suporte e confiança mínimos de 10% e sessões identificadas com *timeout* de 30 minutos.

Regras	**Sup. %**	**Conf. %**
/moodlecti/login/index.php ⇒ /moodlecti/	31,15	79,03
/moodlecti/ ⇒ /moodlecti/login/index.php	31,15	58,81
/moodlecti/course/view.php ⇒ /moodlecti/	24,79	65,07
/moodlecti/ ⇒ /moodlecti/course/view.php	24,79	47,61
/moodlecti/course/view.php ⇒ /moodlecti/login/index.php	13.15	34,50
/moodlecti/login/index.php ⇒ /moodlecti/course/view.php	13.15	33,36
/moodlecti/grade/report/index.php ⇒ /moodlecti/course/view.php	10.19	95,14
/moodlecti/course/view.php ⇒ /moodlecti/grade/report/index.php	10.19	26,73

Analisando os dados apresentados para os diferentes valores de *timeout* aplicados ao algoritmo 1, observa-se que não há diferenças. Para todos os valores de tempo a

Tabela 6. Regras produzidas utilizando o algoritmo 2 com suporte e confiança mínimos de 60% e sessões identificadas com 3 páginas mais frequentes e *time-out* de 30 minutos.

Regras	Sup. %	Conf. %
/moodlecti/course/view.php ⇒ /moodlecti/login/index.php	74,99	93,36
/moodlecti/login/index.php ⇒ /moodlecti/course/view.php	74.99	84,42
/moodlecti/ ⇒ /moodlecti/login/index.php	73,76	90,46
/moodlecti/login/index.php ⇒ /moodlecti/	73,76	83,03
/moodlecti/course/view.php ⇒ /moodlecti/	67,77	84,37
/moodlecti/ ⇒ /moodlecti/course/view.php	67,77	83,13
/moodlecti/course/view.php, /moodlecti/ ⇒ /moodlecti/login/index.php	65,83	97,13
/moodlecti/login/index.php, /moodlecti/course/view.php ⇒ /moodlecti/	65,83	87,78
/moodlecti/ ⇒ /moodlecti/login/index.php, /moodlecti/course/view.php	65,83	80,74
/moodlecti/login/index.php ⇒ /moodlecti/course/view.php, /moodlecti/	65,83	74,10

quantidade e as regras geradas foram as mesmas. Por outro lado, nas colunas referentes ao algoritmo 2 da tabela 4, há diferença na quantidade de regras para os valores de suporte e confiança de 60%, 20% e 10%, respectivamente.

Fazendo uma análise qualitativa das regras das tabelas 5 e 6. Observa-se que praticamente todas as regras da tabela 6 possuem confiança superior às da 5. Tomando como exemplo uma regra como quando o usuário acessa a página "/moodlecti/course/view.php" ele também acessa a página "/moodlecti/login/index.php", com o algoritmo 1 tem-se que o suporte foi de 13,1% e a confiança de 34,5% enquanto que usando o algoritmo 2 o suporte foi de 75,0% e a confiança de 93,4%.

Ao aplicar o algoritmo de mineração sobre os conjuntos de sessões era de se esperar que os padrões encontrados fossem os mesmos, independentemente da estratégia adotada para segmentar sessões. Apesar de serem produzidas regras semelhantes os valores de suporte e confiança são muito diferentes.

Considerando o algoritmo 1 e tomando por base a estrutura das páginas do *moodle* (figura 2) seria de esperar que associações entre páginas como as encontradas surgissem para valores superiores a 50% tanto para suporte quanto para confiança. Tal como mostrado na tabela 6.

Não há como o usuário acessar, por exemplo, os conteúdos das disciplinas sem efetuar o login e sem acessar a página do curso. Um motivo para os resultados da tabela 5 seria a identificação de muitas sessões de tamanho pequeno com o algoritmo 1. Muitas delas também devem conter páginas pouco utilizadas.

Conclui-se que não é interessante utilizar a estratégia que toma somente o *timeout* para identificar sessões de usuários no ambiente estudado porque as regras de associação mais compatíveis com a sequência de navegação natural do ambiente *Moodle* são os encontrados utilizando o algoritmo 2. Há muitas outras regras que não puderam ser mostradas neste trabalho por conta da limitação de espaço, mas se encontram disponíveis em `cti.videira.ifc.edu.br/~valter/webmining/regras/`.

5. Considerações Finais

Analisando as regras de associação geradas e os dados sobre as sessões, conclui-se que o algoritmo 1 não se mostrou eficiente no tratamento das sessões do ambiente *Moodle*. Neste ponto a combinação de páginas mais frequentes com *timeout* foi mais eficiente uma vez que permitiu gerar mais regras dentro do conjunto de dados. Além de que as regras geradas possuem maior frequência, maior garantia de que ocorram e são compatíveis com a topologia do *site*.

O algoritmo 2 se mostrou eficiente num ambiente com controle de acesso, porém, acredita-se que se for aplicado num ambiente de livre acesso não apresente resultados tão satisfatórios quanto os relatados neste trabalho.

Outro ponto interessante de reforçar é que se fosse utilizada uma heurística baseada na estrutura do *site* provavelmente os resultados seriam tão bons quanto os aqui apresentados. Todavia, a estratégia adotada necessita apenas que se indique a quantidade de páginas mais frequentes a considerar, sem qualquer informação adicional sobre a estrutura do ambiente que está sendo estudado.

Em trabalhos futuros seria interessante verificar se os resultados obtidos usando o algoritmo 2 se apresetariam tão bons para outros sistemas com controle de acesso.

Referências

Beauvisage, T. (2000). Topological indicators for the analysis of users paths through the web. Disponível em: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.62.8716&rep=rep1&type=pdf`. Acesso em: jan-2013.

Brusso, M. J., Navaux, P. O. A., and Geyer, C. F. R. (2000). Um modelo para a descoberta de regras de associação aplicado à mineração do uso da web. v.1 n.1. Disponível em: `http://seer.ufrgs.br/cadernosdeinformatica/article/view/v1n1p29-35/8803`. Acesso em: jan-2013.

Catledge, L. D. and Pitkow, J. E. (1995). Characterizing browsing strategies in the world-wide web. In *Computer Networks and ISDN Systems*, pages 1065–1073.

Chaofeng, L. (2004). Research and development of data preprocessing in web usage mining. Disponível em: `http://www.seiofbluemountain.com/upload/product/201001/1264129077i6mdnlwh.pdf`. Acesso em: jan-2013.

Chitraa, V. and Davamani, A. S. (2010). An efficient path completion technique for web log mining. In *IEEE International Conference on Computational Intelligence and Computing Research*.

Witten, I. H. and Frank, E. (2005). *Data Mining Practical Machine Learning Tools and Techniques*. Elsevier, 2nd edition.

Ypma, A. and Heskes, T. (2002). Automatic categorization of web pages and user clustering with mixtures of hidden markov models. In *WEBKDD 2002 - Mining Web Data for Discovering Usage Patterns and Profiles*. Disponível em: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.3626&rep=rep1&type=pdf`. Acesso em: jan-2013.