CENTRO UNIVERSITÁRIO FEEVALE

FABRÍCIO LUÍS COELHO

CLASSIFICAÇÃO SEMI-AUTOMÁTICA DE MONOGRAFIAS

(Título Provisório)

Trabalho de Conclusão I

Novo Hamburgo, dezembro de 2007.

FABRÍCIO LUÍS COELHO

flcoelho@gmail.com

CLASSIFICAÇÃO SEMI-AUTOMÁTICA DE MONOGRAFIAS

(Título Provisório)

Centro Universitário Feevale
Instituto de Ciências Exatas e Tecnológicas
Curso de Ciência da Computação
Trabalho de Conclusão I

Professor orientador: Rodrigo Rafael Villareal Goulart

Novo Hamburgo, dezembro de 2007.

RESUMO

Com a crescente expansão da Internet, a localização de informações precisas torna-se dia-a-dia mais difícil. Para agilizar o processo de aprendizagem, torna-se necessário o desenvolvimento de ferramentas que facilitem a busca de informações, transferindo a tarefa de classificá-las e ordená-las a aplicações computacionais. Este trabalho foca-se na classificação de monografias, utilizando método de aprendizagem supervisionada para desenvolver um protótipo de classificador semi-automático.

Palavras-chave: Classificação. Inteligência Artificial. Aprendizagem Supervisionada.

ABSTRACT

Nowadays it’s becoming very hard for people to find the exactly information they are looking for, because of the great diary expansion of the Internet. So, it’s necessary to transfer the task of searching information to the computers, by developing search tools that make easier to find specific paper, and increase the people capability to learning. This work aims to develop a partially automatized prototype of a text classyfier, using supervised learning, to classify Science Computing monograph.

Keywords: Text categorization. Artificial Intelligence. Supervised learning.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

Graças à Internet, pessoas de quase todos os povos e culturas têm meios à disposição para disseminar suas idéias pelo mundo. Não à toa, portanto, a rede mundial de computadores experimenta uma expansão exponencial no volume de informações disponível.

Com tantas informações disponíveis, não basta disponibilizar artigos ou textos na rede e contar com uma classificação simples de ordem alfabética, seja ela por assunto ou por autor, para ser lido. Este tipo de classificação torna penoso localizar até mesmo mensagens de e-mail, numa conta com uso contínuo e prolongado, se não forem organizadas por pastas ou rótulos.

Assim, o processo de organização das informações na Internet precisa ser constantemente avaliado e repensado, para permitir o rápido acesso aos conhecimentos procurados.

Com o objetivo de contribuir para este processo, o presente trabalho tem por finalidade incluir a produção intelectual do curso de Ciência da Computação no contexto da classificação de documentos. Sendo a classificação um problema de aprendizagem de máquina, o tema é aqui dividido em três capítulos: o primeiro conceitua aprendizagem, aborda os métodos utilizados na aprendizagem de máquina e apresenta um exemplo de aplicação para ilustrar os conceitos; o segundo dispõe sobre as metodologias utilizadas na classificação de textos e nos processos que as compõem, a partir do estudo de dois trabalhos parcialmente relacionados; o terceiro começa a abordagem do tema sob o enfoque do objetivo deste trabalho, ao iniciar o estudo de uma taxonomia para as Ciências da Computação. Ao final, são apresentadas as conclusões.

1. MÉTODOS DE APRENDIZAGEM

Aprender significa “tornar-se apto ou capaz de alguma coisa, em conseqüência de estudo, observação, experiência, advertência, etc.” (FERREIRA, 2004).

Em computação, qualquer sistema dito inteligente, deve ser capaz de aprender. Segundo Herbert Simon (1983, apud LUGER, 2004), *aprendizado de máquina* pode ser uma alteração de comportamento qualquer, capaz de melhorar a performance de um sistema, no momento em que for necessário refazer uma atividade previamente já realizada, ou uma outra atividade similar.

Russel (2004), explica o processo de aprendizado de máquina a partir de observações, de modo que um sistema seja capaz de tomar decisões, atuar e aprender com o resultado de suas ações. Para que isso seja possível, é preciso definir: *o que* deve ser aprendido – os *componentes* da aprendizagem; uma forma de *representação* para estes componentes; o método de *realimentação*, que será utilizado para o aprendizado dos componentes.

Os *componentes* do sistema de aprendizagem contém informações sobre as características do próprio sistema e sobre o universo onde o sistema deve atuar, além de metas e formas para avaliar a atuação do sistema, base do processo de realimentação. Para garantir um bom funcionamento do sistema, é necessária uma boa forma de *representação* para os componentes, seja através de polinômios lineares, de sentenças lógicas ou de descrições probabilísticas.

O processo de *realimentação* é o que compõe a parte inteligente do algoritmo. Seu princípio matemático é a construção de uma hipótese a partir do valor correto de uma função desconhecida, dada como entrada. A dificuldade está em avaliar se a hipótese é

suficientemente próxima da função para que exemplos ainda não conhecidos sejam previstos com exatidão. O sucesso é sensível ao espaço de hipóteses determinado. Há três tipos de *realimentação*: aprendizagem supervisionada, aprendizagem não-supervisionada, e aprendizagem por reforço.

O desafio da *aprendizagem supervisionada* é capacitar o sistema a atuar de acordo com o padrão observado nos exemplos de entradas e saídas. Matematicamente, isso significa que o algoritmo deve ser capaz de construir uma fórmula capaz de produzir resultados semelhantes aos valores de entradas e saídas observados nos exemplos de treinamento. Esta tarefa é mais difícil em ambientes parcialmente observáveis, onde nem sempre os efeitos das ações do sistema são imediatamente visíveis.

Na aprendizagem *não-supervisionada*, o objetivo é um sistema capaz de atuar sem prévio conhecimento de exemplos de saída. Neste tipo de aprendizagem, o sistema deve ser capaz de estabelecer um padrão somente com exemplos de entradas.

A *aprendizagem por reforço*, por sua vez, consiste no aperfeiçoamento do sistema por meio da avaliação dos resultados de suas ações e de indicações quanto ao comportamento do ambiente.

1.1. Aprendizagem supervisionada

Na aprendizagem supervisionada aplicada a textos, objeto deste trabalho, um conjunto de exemplos de documentos pré-classificados, cuja classe já é conhecida, é selecionado para realizar o treinamento do algoritmo. A classe apresenta o conceito sobre o qual o programa deverá efetuar previsões (ABREU, 2005). Com base nas características observadas nos exemplos, o algoritmo é induzido a construir um classificador que realizará a categorização de documentos ainda não vistos (SEBASTIANI, 2002), aplicando os conhecimentos aprendidos.

Antes de ser colocado em prática o classificador deve ser testado em um novo conjunto de documentos pré-classificados, o conjunto de testes. Para isso, é importante que os documentos que compõem os conjuntos de treinamento e de testes sejam diferentes. Se o algoritmo de treinamento for exposto aos dados que serão utilizados para teste, a qualidade da avaliação ficará comprometida.

Este tipo de aprendizagem de máquina proporciona grandes benefícios. O algoritmo de aprendizagem e o classificador gerado são independentes, o que significa que, se houver

atualização ou adição de novas categorias, basta selecionar novos exemplos e realizar novamente o treinamento. Além disso, o mesmo algoritmo de aprendizagem pode ser utilizado para treinar classificadores de diversos domínios diferentes (SEBASTIANI, 2002).

Entre os métodos de aprendizagem supervisionada estão a indução de regras e as árvores de decisão. Os métodos ora relacionados são abordados a seguir, com maior ênfase às árvores de decisão, por constituírem se no método escolhido para este trabalho.

1.1.1. Árvores de decisão

Um método de aprendizagem supervisionada simplificado e eficiente é o de *árvores de decisão*. Este tipo de algoritmo gera uma decisão a partir do conjunto de atributos de uma determinada situação ou objeto. Graficamente, uma árvore de decisão pode ser representada como uma estrutura hierarquicamente organizada semelhante a uma árvore real invertida, que deve ser percorrida da raiz para as folhas (RODRIGUES, 2004). A árvore é percorrida, a partir de testes efetuados em cada nó, rumo às suas folhas, que armazenam os valores de retorno para cada combinação de atributos.

Árvores de decisão podem ser utilizadas para escrever praticamente qualquer tipo de função. Para simplificar, este texto explica e exemplifica as *árvores de decisão booleanas*, onde cada exemplo pode ser classificado como positivo ou negativo. A Figura 1 ilustra uma árvore de decisão booleana.

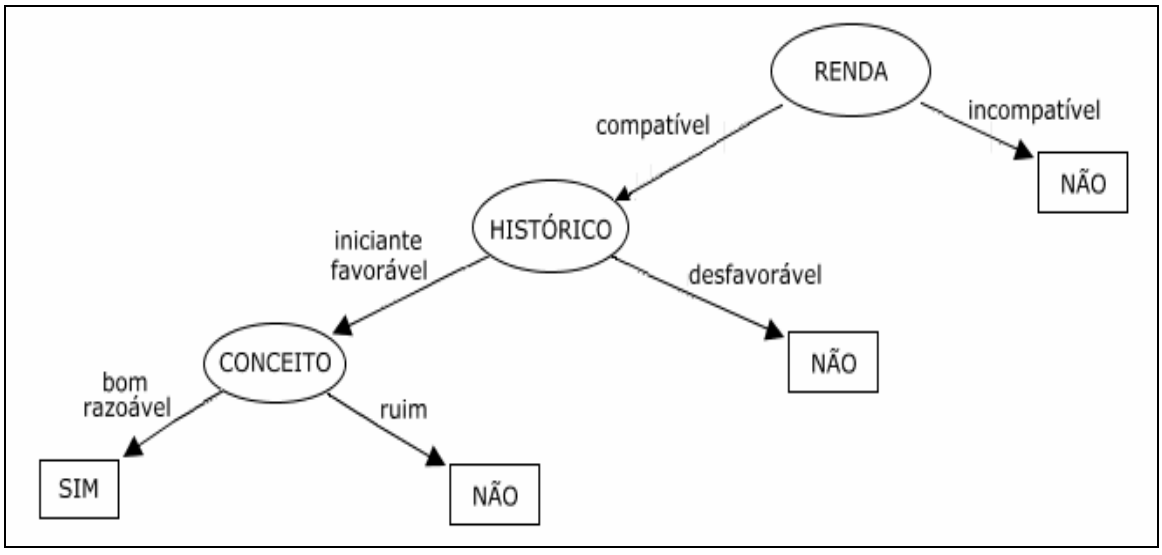

Figura 1. Árvore de decisão booleana

Nas árvores de decisão booleanas, a estrutura da árvore cresce exponencialmente, conforme a quantidade de linhas da tabela-verdade relacionada. Quando induzidas por um conjunto de exemplos de treinamento, objetivam definir a classificação correta para cada exemplo. Assim, se houver um caminho para uma folha referente a cada exemplo, a árvore será grande e capaz de classificar corretamente exemplos já vistos, mas não de extrair padrões.

Um método capaz de reduzir a estrutura da árvore de decisão é forçá-la a dar prioridade ao atributo mais relevante. Este método, no entanto, pode resultar em exemplos com descrição idêntica e classificações diferentes, o que representa *ruído nos dados*.

Outro procedimento que proporciona melhores resultados na árvore final é a classificação dos atributos com medidas formais para expressar valores máximos e mínimos possíveis. Este método favorece a escolha de um atributo perfeito, capaz de seccionar os exemplos em conjuntos onde, de um lado são representados apenas positivos e, de outro, apenas negativos.

Quando os conjuntos de treinamento utilizados com árvores de decisão são grandes, existe o risco da árvore apresentar uma regularidade incoerente nos dados. Esta anomalia é conhecida como *superadaptação*. Uma saída para minimizar a ação desses atributos irrelevantes é a técnica de *poda* de árvore de decisão, que utiliza estatística, ou validação cruzada, para melhorar a eficiência da estrutura.

O processo de indução de árvores de decisão pode apresentar problemas como: omissão de dados, atributos com valores múltiplos, atributos de entrada com valores contínuos e inteiros e atributos de saída com valores contínuos. Esses problemas devem ser considerados, ao projetar sistemas baseados em árvores de decisão.

1.1.1.1. Construindo uma árvore de decisão

A construção de uma árvore de decisão é um processo que compreende, basicamente, as seguintes etapas (REZENDE, 2003): seleção de exemplos para os conjuntos de treinamento e testes; seleção do atributo de secção; realização do treinamento; e poda.

A seleção dos exemplos deve considerar algumas particularidades. Por exemplo, quando os exemplos se referirem a uma mesma classe, o resultado será uma árvore identificada por um nó-folha. Quando não houver exemplos, a definição da classe a que a

folha será associada dependerá de informações externas ao conjunto. Por outro lado, se o conjunto for de variadas classes e exemplos, deve ser refinado em conjuntos menores, cada qual contendo uma classe.

Selecionar um atributo de secção adequado é uma etapa muito importante, na construção da árvore. Este atributo pode ser escolhido das seguintes formas: aleatoriamente; selecionando-se o de menor variação; pelo atributo com a maior variação de valores; optando-se pelo que resulta no menor tamanho de subárvores; pelo índice Gini (BREIMAN, FRIEDMAN, OLSHEN & STONE, 1984, apud REZENDE, 2003); ou pela razão de ganho (QUINLAN, 1993, apud REZENDE, 2003).

Na etapa de treinamento, em geral, cada avaliação é selecionada a partir de um atributo com resultados exclusivos entre si. Os testes são efetuados de modo recursivo para cada subconjunto. Deste modo, em cada nó, as opções direcionam para os segmentos de árvore estruturados a partir de um subconjunto específico de testes. Ao final, deve ser efetuada uma poda na árvore, para que se obtenha uma generalização mais eficiente (REZENDE, 2003).

Para exemplificar uma árvore de decisão, a partir de uma adaptação de Quinlan (1993; apud REZENDE, 2003), pode-se verificar a viabilidade de concessão, ou não, de crédito por uma instituição financeira a alguns hipotéticos pretendentes. O exemplo a seguir é livremente inspirado em algumas situações possíveis do cotidiano bancário, tendo sido desenvolvido a partir da experiência profissional deste autor na área onde atua. Cada exemplo contém os seguintes atributos:

- **conceito** - refere-se ao conceito do cliente na praça e pode assumir os seguintes valores: bom, razoável ou ruim;
- **histórico** - traduz o passado do cliente na instituição, podendo ter os seguintes valores: favorável, desfavorável ou iniciante;
- **renda** - atributo pré-classificado, no qual a instituição determina se o cliente possui renda compatível ou incompatível com o crédito pretendido;
- **risco** - outro atributo pré-classificado, em que a instituição determina se o cliente oferece um risco baixo, médio ou alto de inadimplência, a partir do estudo de suas características;

- **conceder?** – atributo-classe, indica a classe do exemplo, podendo ter os seguintes valores: sim ou não.

A Tabela 1 apresenta alguns exemplos que serão utilizados para treinamento de uma árvore de decisão.

Tabela 1 – Exemplos de concessão de crédito.

Exemplo	Conceito	Histórico	Renda	Risco	Conceder?
1	bom	favorável	compatível	baixo	sim
2	ruim	iniciante	compatível	médio	não
3	bom	desfavorável	compatível	alto	não
4	razoável	favorável	incompatível	médio	não
5	ruim	desfavorável	compatível	alto	não
6	bom	desfavorável	compatível	médio	não
7	razoável	favorável	compatível	médio	sim
8	ruim	favorável	compatível	baixo	não
9	razoável	desfavorável	incompatível	alto	não
10	bom	iniciante	incompatível	baixo	não
11	ruim	iniciante	compatível	médio	não
12	razoável	iniciante	compatível	médio	sim

Primeiramente, o atributo de secção deve ser selecionado. Para este exemplo, uma boa opção é selecionar o de menor variação: renda. Como resultado, pode ser observado na Tabela 2. Foram gerados dois subconjuntos: o primeiro com exemplos pertencentes às duas classes possíveis; o outro composto apenas por exemplos negativos.

Tabela 2 - Construindo uma árvore de decisão (passo 1).

Teste	Ex	Renda	Histórico	Conceito	Risco	Conceder?
	1	compatível	favorável	bom	baixo	sim
	2	compatível	iniciante	ruim	médio	não
	3	compatível	desfavorável	bom	alto	não
	5	compatível	desfavorável	ruim	alto	não
if renda = compatível	6	compatível	desfavorável	bom	médio	não
	7	compatível	favorável	razoável	médio	sim
	8	compatível	favorável	ruim	baixo	não
	11	compatível	iniciante	ruim	médio	não
	12	compatível	iniciante	razoável	médio	sim
	4	incompatível	favorável	razoável	médio	não
if renda = incompatível	9	incompatível	desfavorável	razoável	alto	não
	10	incompatível	iniciante	bom	baixo	não

A Figura 2 ilustra o primeiro segmento de árvore de decisão gerado a partir da seleção do atributo de secção.

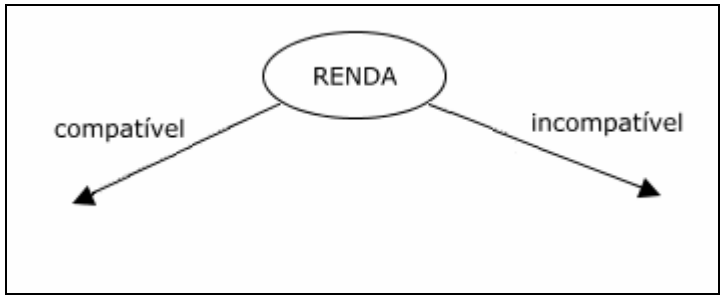

Figura 2: Ilustração parcial da árvore de decisão.

Em seguida, um novo teste requer a seleção de outro atributo. Utilizando o mesmo método, é possível observar (Tabela 3) que o atributo “histórico” retorna três subconjuntos, um deles apenas de exemplos negativos.

Tabela 3: Construindo uma árvore de decisão (passo 2).

Teste	Ex	Renda	Histórico	Conceito	Risco	Conceder?
if renda = compatível **e** histórico = desfavorável	3	compatível	desfavorável	bom	alto	não
	5	compatível	desfavorável	ruim	alto	não
	6	compatível	desfavorável	bom	médio	não
if renda = compatível **e** histórico = favorável	1	compatível	favorável	bom	baixo	sim
	7	compatível	favorável	razoável	médio	sim
	8	compatível	favorável	ruim	baixo	não
if renda = compatível **e** histórico = iniciante	2	compatível	iniciante	ruim	médio	não
	11	compatível	iniciante	ruim	médio	não
	12	compatível	iniciante	razoável	médio	sim
if renda = incompatível	4	incompatível	favorável	razoável	médio	não
	9	incompatível	desfavorável	razoável	alto	não
	10	incompatível	iniciante	bom	baixo	não

Restando dois conjuntos, compostos de classes distintas, convém notar que o atributo "conceito" possui valores idênticos nos exemplos 8, 2 e 11, classificados como negativos, restando os exemplos 1, 7 e 12, classificados como positivos. Assim, para o terceiro teste é conveniente selecionar o atributo "conceito" e, como resultado, os exemplos ficam adequadamente distribuídos, conforme demonstrado na Tabela 4.

Tabela 4: Construindo uma árvore de decisão (passo 3).

Teste	Ex	Renda	Histórico	Conceito	Risco	Conceder?
if renda = compatível **e** histórico = favorável/iniciante **e** conceito = bom/razoável	1	compatível	favorável	bom	baixo	sim
	7	compatível	favorável	razoável	médio	sim
	12	compatível	iniciante	razoável	médio	sim
if renda = compatível **e** histórico = desfavorável	3	compatível	desfavorável	bom	alto	não
	5	compatível	desfavorável	ruim	alto	não
	6	compatível	desfavorável	bom	médio	não
if renda = compatível **e** conceito = ruim	8	compatível	favorável	ruim	baixo	não
	2	compatível	iniciante	ruim	médio	não
	11	compatível	iniciante	ruim	médio	não
if renda = incompatível	4	incompatível	favorável	razoável	médio	não
	9	incompatível	desfavorável	razoável	alto	não
	10	incompatível	iniciante	bom	baixo	não

Uma ilustração da árvore de decisão gerada pode ser observada na Figura 3.

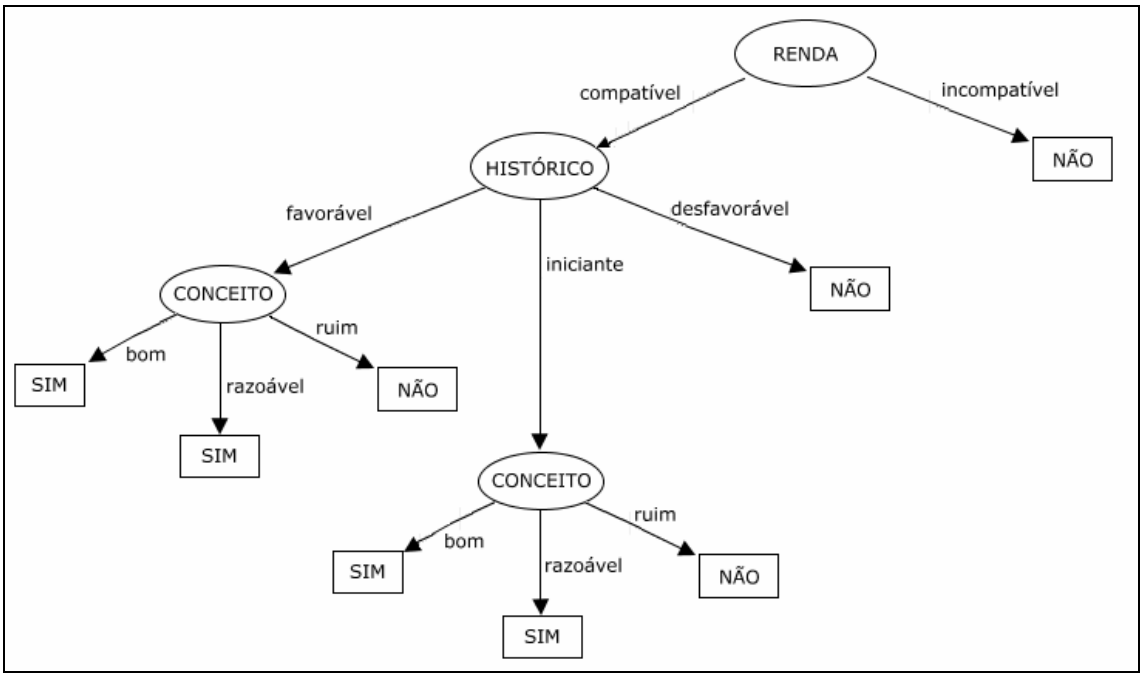

Figura 3: Ilustração da árvore de decisão resultante.

Como a árvore de decisão final não apresenta ruídos, dada a reduzida quantidade de exemplos, é desnecessário podar a árvore.

A Figura 1, constante no capítulo 1.1.1, retro, reproduz uma versão simplificada desta mesma árvore, onde os ramos “favorável” e “iniciante”, do atributo *histórico*, estão unidos, permitindo a representação de apenas um atributo *conceito*, representados em duplicidade na Figura 3.

1.1.2. Indução de regras

Segundo Rezende (2003) árvores de decisão recursivamente induzidas separam exemplos em subconjuntos menores. A *indução de regras* efetua este processo de forma direta e pode ser classificada em *ordenada* ou *não-ordenada*.

Na *indução ordenada de regras*, o algoritmo realiza repetições em busca de um termo abrangente a vários exemplos de uma mesma classe. Ao encontrá-lo, os exemplos são removidos do conjunto, a regra associada ao termo é inserida ao final da lista e o processo é reiniciado, até esgotar o universo de termos possíveis para o conjunto.

Ao testar um novo exemplo, o algoritmo verifica as condições de cada regra, ordenadamente, até que o exemplo satisfaça-as. O exemplo é, então, associado à classe correspondente. Para os casos omissos, há uma regra padrão.

O exemplo demonstrado no subcapítulo anterior, se aplicado à indução ordenada de regras resultaria numa lista de regras semelhante à da Tabela 5, abaixo.

Tabela 5: Regras ordenadas para os exemplos de concessão de crédito.

	Regra	**CC**	**CI**
R_1	**Se** conceito = ruim **então** conceder = não	4	0
R_2	**senão se** histórico = desfavorável **então** conceder = não	3	0
R_3	**senão se** renda = compatível **então** conceder = sim	3	0
R_4	**senão** conceder = não	2	0

Na *indução não-ordenada de regras*, o algoritmo efetua iterações para todas as classes. Ao encontrar uma regra, os exemplos ligados a esta são removidos e os que correspondem a classes incorretamente cobertas são mantidos, para serem comparados a todas as novas regras encontradas.

Na comparação com árvores de decisão, as regras apresentam-se mais compreensíveis e consomem menos espaço. Como desvantagens, as regras são mais lentamente induzidas e necessitam de muitos ajustes de parâmetros.

1.2. Aprendizagem não supervisionada

Segundo Rodrigues (2004), inicialmente a aprendizagem não-supervisionada não conta com dados rotulados ou classificados, tampouco as classes precisam estar determinadas. A aprendizagem ocorre por observação e descoberta. Dentre os métodos que possuem estas características, estão as redes neurais não supervisionadas, os modelos de mistura e algoritmos de análise de grupos (*clustering*).

A aprendizagem não supervisionada em redes neurais, segundo Ferneda (2006), trabalha com padrões de entrada, cujas regularidades são detectadas na medida em que a rede vai estabelecendo parâmetros internos para classificar o que é aprendido. O sucesso deste método está intrinsecamente ligado à redundância nos dados de entrada, fator imprescindível para uma adequada detecção de padrões pelas redes neurais.

Este tipo de aprendizagem pode apresentar bons resultados para grandes conjuntos de dados. Um exemplo de possível aplicação da aprendizagem não supervisionada aplicada à formação de agrupamentos é a análise de um espectro composto por inúmeras estrelas. Não há rótulos para identificar as estrelas analisadas, portanto é necessário realizar a formação de

agrupamentos não supervisionados para fazê-lo e, a partir daí, atribuir as categorias - anã branca, gigante vermelha... (RUSSEL, 2004). Da mesma forma, o conceito pode ser utilizado para identificar categorias diversas na Ciência, cuja taxonomia seja conhecida ou não.

2. METODOLOGIAS DE CLASSIFICAÇÃO DE TEXTOS

A classificação de textos não é uma preocupação recente. Trabalhos nesta área já são realizados desde o princípio dos anos sessenta (SEBASTIANI, 2002). Contudo, a partir dos anos noventa despertaram maior interesse na área de sistemas de informação, não apenas por aumentar o interesse em aplicações do tipo, mas também graças ao aumento da capacidade de processamento dos computadores.

No contexto atual, onde a quantidade de informações produzidas diariamente cresce em ritmo alucinante, a categorização de textos pode ser empregada em muitas áreas distintas, desde a organização de processos na área jurídica, até mesmo para organizar documentos coletados na Internet para uma pesquisa, ou, no escopo deste trabalho, para classificar monografias de conclusão do curso de Ciências da Computação.

Duas dissertações foram selecionadas, para proporcionar um melhor entendimento da tecnologia e dos processos necessários à classificação textual. Uma delas utiliza técnica de aprendizagem não supervisionada para agrupar documentos de acordo com o grau de semelhança, num processo também conhecido como "*clustering*". A outra emprega classificação para separar páginas da Internet que contenham anúncios de produtos à venda das demais. Embora diferentes, as técnicas empregadas nestes trabalhos oferecem bons subsídios para que sejam atingidos os objetivos deste. Os trabalhos são detalhados a seguir.

2.1. Agrupamento, ou *clustering*, de documentos

Como já abordado, a formação de agrupamentos é uma técnica de aprendizagem não-supervisionada. Leandro Wives (1999) aplica a metodologia para, automaticamente, organizar documentos de texto em grupos, de acordo com o grau de semelhança e, assim, localizar, manipular e analisar mais facilmente esses documentos.

O objetivo da criação de agrupamentos é encontrar objetos com características comuns e inseri-los em conjuntos contendo outros objetos afins. Dentre as diversas técnicas existentes para realizar agrupamentos, a escolhida para a dissertação se baseia na classe *graphic-theoretic*. Os algoritmos baseados nesta classe realizam três passos que compreendem identificar e selecionar características, calcular as similaridades e identificar os clusters.

No primeiro passo, as palavras são analisadas e são selecionadas as que melhor caracterizam o objeto. O resultado são listas de palavras relevantes associadas a cada documento. O segundo passo aproveita as listas geradas pelo primeiro para identificar o grau de semelhança entre os documentos, gerando uma matriz com os valores de semelhança entre eles. Finalmente, as correlações entre os componentes de cada matriz são analisadas e os grupos são compostos.

É importante salientar que nem todas as palavras podem ser analisadas como atributos dos documentos, já que são utilizadas apenas para fornecer um sentido às orações, caso de preposições, conjunções e artigos. Estas e outras palavras são denominadas *stop-words*, ou palavras negativas, devem ser removidas do processo em uma etapa de pré-processamento, em geral juntamente das palavras que ocorrem com menor freqüência no texto, que devem ser separadas das mais relevantes.

Há outras técnicas para redução do universo de palavras analisadas que, aliadas à remoção de palavras negativas e menos freqüentes, buscam reduzir o tempo total de processamento, agilizando o processo de agrupamento. Estas e outras técnicas relacionadas não serão abordadas neste trabalho por não serem relevantes à classificação de textos em aprendizagem supervisionada.

As pesquisas na área de agrupamentos resultaram no desenvolvimento do algoritmo best-star, idealizado a partir de melhorias implementadas no algoritmo star, e na ferramenta Eurekha que, associada ao algoritmo best-star, foi utilizada para realizar os experimentos de organização de documentos.

O método empregado é composto dos seguintes passos: identificação de palavras, remoção de stop-words, cálculo da freqüência relativa, remoção de palavras pouco freqüentes, utilização de palavras mais freqüentes, uso de uma fórmula de semelhança - definida pelo próprio autor, para a dissertação -, e aplicação de algoritmo de agrupamento.

Para avaliar os resultados, o autor utiliza como medidas de desempenho abrangência (ou recall) e precisão (ou precision). A primeira verifica a proporção de classes que

efetivamente foram atribuídas, relativa à quantidade esperada de classes. A segunda, por sua vez, verifica a proporção de classes atribuídas de forma correta em relação à quantidade de classes atribuídas. Estas técnicas são associadas às técnicas de microaveraging e macroaveraging (LEWIS, 1991, apud WIVES, 1999). Microaveraging é a mais utilizada e verifica a abrangência ou a precisão de todo o conjunto trabalhado. Macroaveraging é mais restrita, pois analisa a abrangência ou a precisão de cada grupo isoladamente. A média obtida a partir da aplicação dessas métricas indica a eficiência do método.

Os testes são realizados com uma coleção de documentos da agência Reuters. A coleção é largamente utilizada para testes comparativos em algoritmos de classificação. Na dissertação, os testes são realizados com a subdivisão HAYES SPLIT, tendo sido selecionada por possuir o menor subconjunto de dados, bem como para ser facilmente comparada com os testes apresentados por David Lewis (LEWIS, 1991, apud WIVES, 1999), referentes a um sistema de classificação denominado “Construe”. Os parâmetros selecionados para os testes são: 722 documentos, ou textos; algoritmo best-star; remoção de palavras negativas.

Tabela 6 – Resultados dos testes.

Experimento	**Microaverage recall**	**Microaverage precision**	**Grupos**	**Textos agrupados**	**Textos não indexados**
Wives	0,95	0,43	266	687	139
Construe	0,89	0,92	-	-	-

Fonte: WIVES, 1999.

Como pode ser observado, o experimento do autor apresenta um grau de microaverage recall mais elevado do que o obtido pelo sistema comparado, porém o grau de microaverage precision mostra-se muito abaixo do parâmetro. Esta diferença é atribuída ao método utilizado para cálculo (microaverage), já que é mais indicado para medir o desempenho em classificação de informações. Ainda segundo o autor, o número de textos não indexados, 139, também é fator decisivo na queda de precisão verificada. Como aspecto positivo, a ferramenta conseguiu agrupar documentos que não possuíam categorias, à priori.

2.2. Classificação de textos baseada em Support Vector Machines

Martins (2003) utiliza aprendizagem supervisionada para realizar classificação

booleana de páginas da Internet. O objetivo é dividir as páginas de acordo com o conteúdo. Os exemplos positivos compreendem páginas contendo textos associados a valores reconhecidos como monetários, enquanto os exemplos negativos compreendem qualquer página no qual este padrão não seja encontrado.

Para realizar os experimentos, o autor seleciona o algoritmo SVM^{light}, baseado no método Support Vector Machines (VAPNIK ,1995, apud MARTINS, 2003). A seguir é descrito o método utilizado.

2.2.1. Metodologia

O principal desafio das experimentações é adaptar o material a classificar – páginas de sites brasileiros, na Internet, contendo produtos à venda – à entrada padrão do algoritmo selecionado, a fim de realizar as etapas de aprendizagem e testes. Para isso, são realizados os seguintes passos: 1) obtenção de documentos e (2) seu pré-processamento; 3) extração de conhecimento e (4) sua avaliação.

Os documentos compreendem amostras de páginas coletadas na Internet. As páginas são armazenadas em repositórios locais, já pré-categorizadas em exemplos positivos – quando detectado algum texto associado a valores monetários, indicativo de precificação de produto à venda – e negativos – quando não possuem as características indicadas. Esta tarefa é realizada com auxílio de ferramenta desenvolvida para o projeto. Além disso, todas as amostras coletadas são, também, observadas pelo autor da dissertação, via browser, para posterior comparação com os resultados.

A etapa de pré-processamento objetiva extrair, sintática ou estatisticamente, determinadas características de cada documento, a fim de representá-lo de forma compacta; a representação é realizada a partir da identificação de atributos, atribuição de pesos e redução da representação (IMAMURA, 2001, apud MARTINS, 2003). Martins (2003), utiliza os seguintes recursos para a etapa de pré-processamento: parsing, para remoção de tags e interpretação de caracteres especiais; tokenising, para reconhecimento de palavras significativas nos textos, exceto números e stop list, e sentenças que indiquem preços; composição de dicionário, durante o treinamento, contendo um conjunto de palavras do texto e seus pesos; atribuição de pesos às palavras.

Como resultado dos passos 1 e 2 são gerados os arquivos de entrada para os processos de treinamento e teste do algoritmo SVM^{light}, no terceiro passo.

2.2.2. Resultados obtidos

O autor não avalia os resultados, limitando-se a demonstrá-los em tabelas. O que se observa é que os resultados obtidos indicam elevados graus exatidão (accuracy) e abrangência (recall), com precisão (precision) máxima na maior parte das amostras, o que indica que o algoritmo SVM^{light} apresenta alta performance, ao menos quando utilizado para classificação booleana.

Quando os resultados obtidos pelo SVM^{light} são comparados pelo autor com os resultados de testes efetuados em conjunto idêntico utilizando um agente baseado no algoritmo c4.5 (QUINLAN, 1993, apud REZENDE, 2003), o SVM^{light} demonstra melhores precisão e exatidão, mas menor abrangência.

Na avaliação da capacidade de classificação dos exemplos preditos em outros conjuntos, é possível observar elevados graus de precisão, exatidão e abrangência do algoritmo, atingindo muitas vezes os seus valores máximos.

3. TAXONOMIA PARA AS CIÊNCIAS DA COMPUTAÇÃO

Taxonomia é um termo largamente utilizado nas Ciências Biológicas para indicar as classes em que se subdividem espécies, por exemplo, de animais ou plantas. No escopo deste trabalho, o termo é utilizado para referenciar as classes e suas subdivisões, que servirão de apoio à categorização das monografias de conclusão do curso de Ciências da Computação.

Outras pesquisas foram realizadas, na Internet e na biblioteca da Feevale, à procura de uma proposta de taxonomia para fundamentar este trabalho. No diretório do Yahoo![1], por exemplo, existem 52 subcategorias para a categoria Computer Science. Dentre as categorias, há desde Careers for Women (Carreiras para Mulheres) e Organizations (Organizações), até Quantum Computing (Computação Quântica) e Neural Networks (Redes Neurais).

O diretório do Google[2], por sua vez, contém 17 subcategorias para a mesma finalidade, contendo uma variedade de temas como Academic Departments (Departamentos Acadêmicos) e People (Pessoas), até Parallel Computing (Computação Paralela) e Software Engineering (Engenharia de Software).

Há, ainda, uma discussão iniciada na Wikipédia, em português europeu, com o objetivo de estruturar o tema organizadamente, mas o trabalho está apenas começando, possui diversas categorias duplicadas, confunde a Ciência da Computação com Informática e carece de fundamento técnico.

Com a ajuda de uma bibliotecária da Instituição, foi possível obter a descrição da área de Ciências da Computação de acordo com a Tabela de Classificação Decimal Universal – CDU, na qual a biblioteca da Feevale se baseia para organizar seus livros.

---

[1] - COMPUTER SCIENCE IN THE YAHOO! DIRECTORY. Disponível em: <http://dir.yahoo.com/Science/Computer_Science/>. Acesso em 15/11/2007.

[2] - GOOGLE DIRECTORY - COMPUTERS > COMPUTER SCIENCE. Disponível em: <http://dir.google.com/Top/Computers/Computer_Science/>. Acesso em 15/11/2007.

Através da tabela, nota-se que os livros são categorizados por assunto relacionado à área de informática, quando o objetivo dos trabalhos do curso possui tendências bem mais restritas à área profissional.

As categorias constantes nos diretórios dos grandes portais, ou na discussão da enciclopédia livre, assim como a tabela de organização de livros na Biblioteca, são propostas pouco consistentes à solução do problema proposto por este trabalho, já que são carentes de fundamento acadêmico. A alternativa encontrada é a utilização das classes de artigos e documentos científicos do currículo Lattes, de autoria do CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico). O currículo Lattes é muito disseminado no meio acadêmico, servindo como base de dados de referência para professores, alunos e instituições de pesquisas, e, atualmente, divide a Ciência da Computação em cinco categorias e dezenove subcategorias, abaixo relacionadas:

1. **Inteligência Artificial**, subdividida em: Sistemas Especialistas, Sistemas Inteligentes, Sistemas Multiagentes, Sistemas Tutores Inteligentes;

2. **Matemática da Computação**, subdividida em: Matemática Simbólica, Modelos Analíticos e de Simulação;

3. **Metodologia e Técnicas da Computação**, subdividida em: Banco de Dados, Engenharia de Software, Linguagens de Programação, Processamento Gráfico (Graphics), Sistemas de Informação;

4. **Sistemas de Computação**, subdividida em: Arquitetura de Sistemas de Computação, Hardware, Software Básico, Teleinformática;

5. **Teoria da Computação**, subdividida em: Análise de Algoritmos e Complexidade de Computação, Computabilidade e Modelos de Computação, Linguagens Formais e Autômatos, Lógicas e Semântica de Programas.

4. WEKA

Weka é um conjunto de ferramentas que agregam diversos algoritmos para mineração de dados e aprendizagem de máquina. O nome da aplicação deriva das iniciais de *Waikato Environment Knowledge Analysis*. Este software, desenvolvido pela Universidade de Waikato, da Nova Zelândia, está disponível[1] na página da instituição, sob licença GPL.

Por ser desenvolvida em Java, Weka é portável para qualquer sistema operacional compatível com a linguagem de programação. A tela inicial da ferramenta, atualmente na versão 3.4.11, é demonstrada na Figura 4.

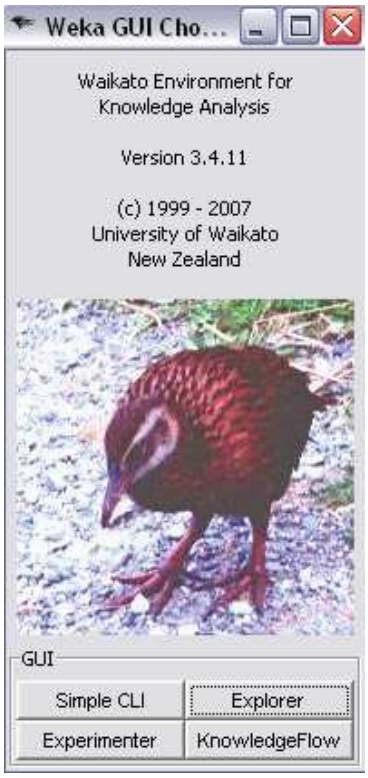

Figura 4. Weka – tela inicial

[1]Weka 3. Disponível em: <http://www.cs.waikato.ac.nz/~ml/weka/>. Acesso em 24/12/2007.

Em sua tela principal, o software apresenta uma interface gráfica com quatro botões – *Simple CLI*, *Explorer*, *Experimenter* e *KnowledgeFlow* – que representam os ambientes disponíveis. O primeiro ambiente, *Simple CLI*, apenas demonstra como os algoritmos da aplicação são executados via linha de comando. O botão *Explorer* leva ao ambiente do software onde são efetuadas as classificações, seleções de algoritmos e análises. *Experimenter* é um ambiente onde é possível realizar experimentos e testes comparativos de performance para classificadores ou problemas de regressão (regression problems). *KnowledgeFlow* é uma interface baseada em Java Beans que permite montar e realizar experimentos de aprendizagem de máquina.

Neste trabalho, a ferramenta será utilizada para gerar a árvore de decisão necessária à classificação das monografias da Ciência da Computação. As árvores são geradas a partir do ambiente *Explorer* da ferramenta. Apenas as etapas necessárias à construção da árvore serão adiante detalhadas[2].

4.1. O ambiente *Explorer*

Para iniciar a utilização do ambiente, é preciso selecionar um arquivo, um endereço da Internet (URL), ou uma base de dados. Esta tarefa é realizada na aba *Preprocess* (pré-processamento) a única disponível quando o *Explorer* é iniciado. As demais abas desta aplicação são disponibilizadas tão logo cumprida esta primeira etapa. A interface do ambiente é demonstrada na Figura 5.

[2]EN:WEKA 3.4.11 - WEKADOC. Disponível em: <http://weka.sourceforge.net/wekadoc/index.php/en:Weka_3.4.11>. Acesso em 24/12/2007

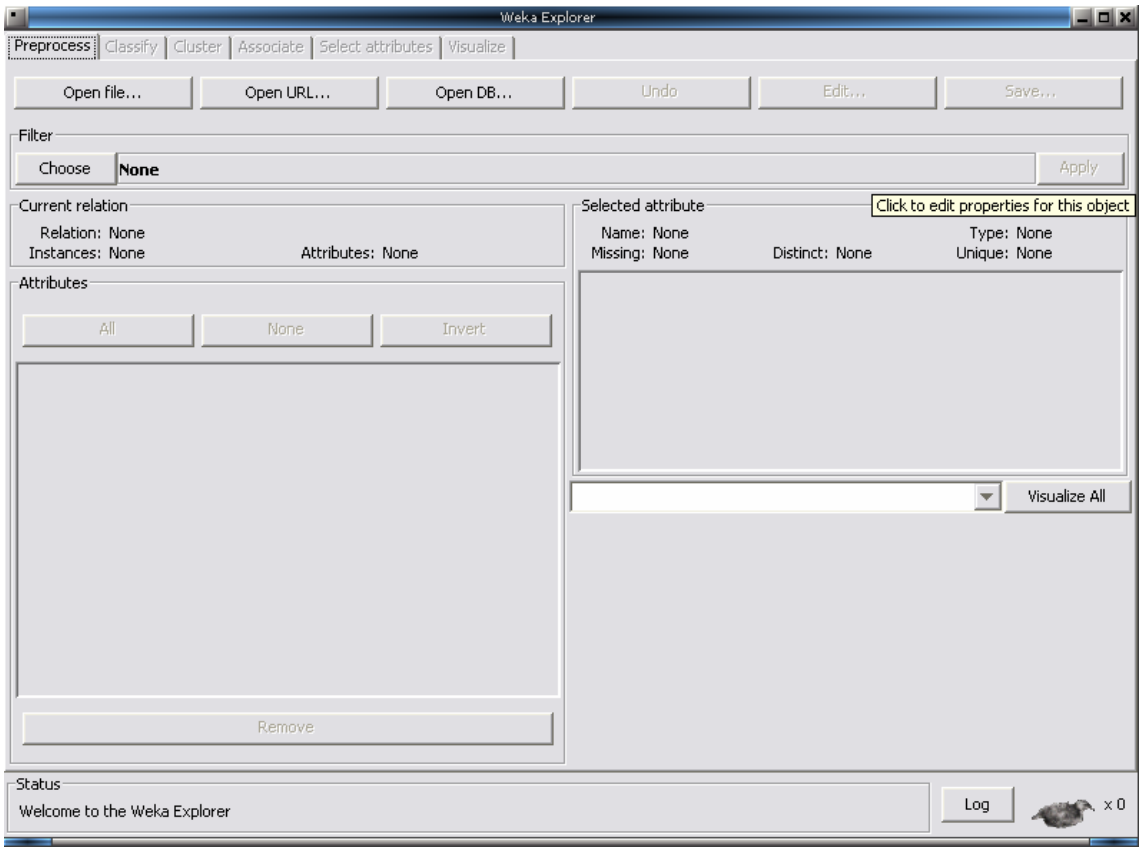

Figura 5. Interface do ambiente *Explorer (Weka)*.

As abas existentes no *Explorer* são: a) **preprocess**: permite seleção e alteração dos dados a serem trabalhados; b) **classification**: nesta aba é possível realizar treinamento e teste de esquemas de aprendizagem que classificam ou efetuam regressão; c) **cluster**: aba que oferece recursos para aprendizagem de agrupamentos de dados; d) **associate**: aqui há recursos para aprendizagem de regras de associação de dados; e) **select attributes**: nesta aba é possível selecionar os atributos mais relevantes aos dados; f) **visualize**: esta aba disponibiliza uma série de gráficos interativos em duas dimensões para análise dos dados. Para o escopo deste trabalho, apenas os recursos das duas primeiras abas serão utilizados.

Weka suporta alguns formatos de arquivos, como CSV e C4.5, além do formato próprio da ferramenta, o ARFF. Este é um formato de arquivo em texto, contendo atributos e seus respectivos dados, para serem manipulados pela aplicação. Um exemplo de arquivo no formato ARFF é apresentado na Figura 6, adiante.

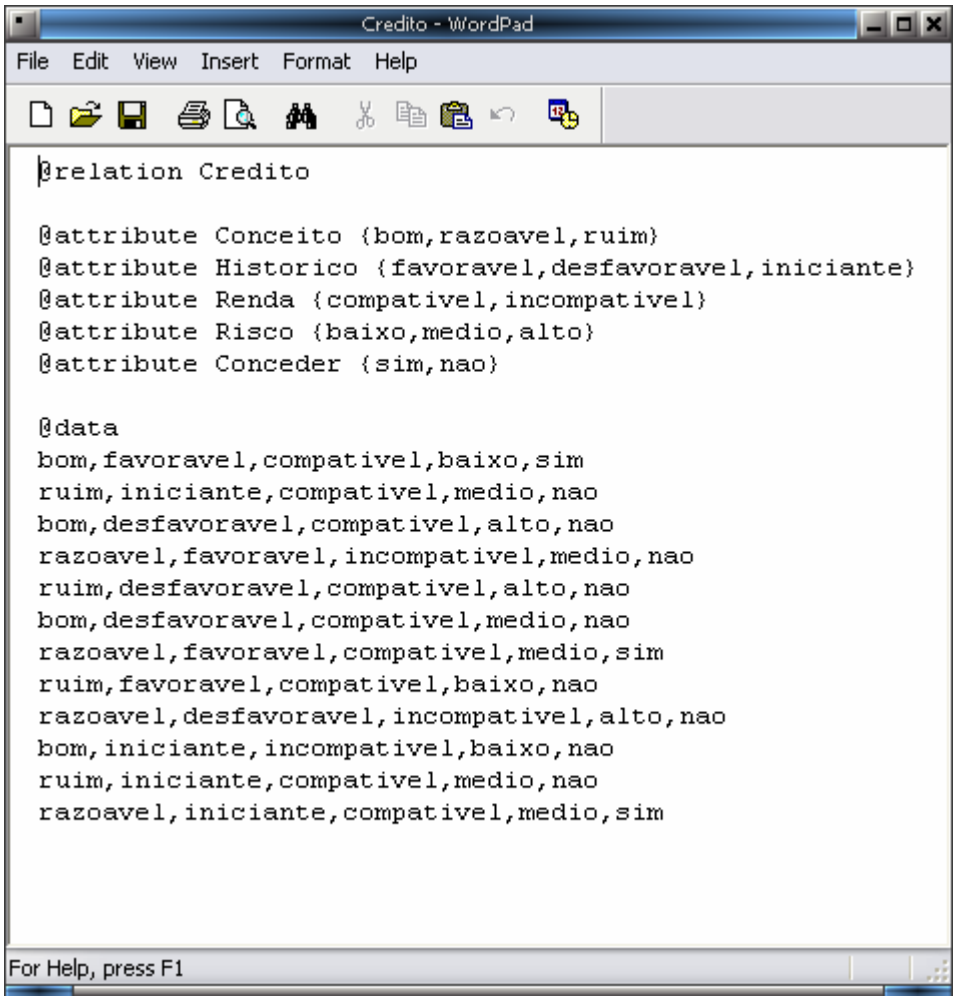

Figura 6. Conteúdo do arquivo *Credito.arff*.

O arquivo *Credito.arff* foi desenvolvido a partir do exemplo demonstrado no capítulo 1.1.1.1. Como demonstrado na Figura 6, arquivos neste formato têm a seguinte estrutura: nome da relação (*relation*); lista de atributos (*attribute*) e seus respectivos tipos de dados; e os dados (*data*). Os dados são dispostos de acordo com a ordem em que os atributos aparecem. Deste modo, a primeira linha de dados – *bom,favoravel,compativel,baixo,sim* – representa a seqüência de atributos – *conceito,historico,renda,risco,conceder* – anteriormente descrita.

Quando o arquivo é aberto no *Explorer* – aba *Preprocess* –, o programa disponibiliza diversas informações, como demonstrado na Figura 7.

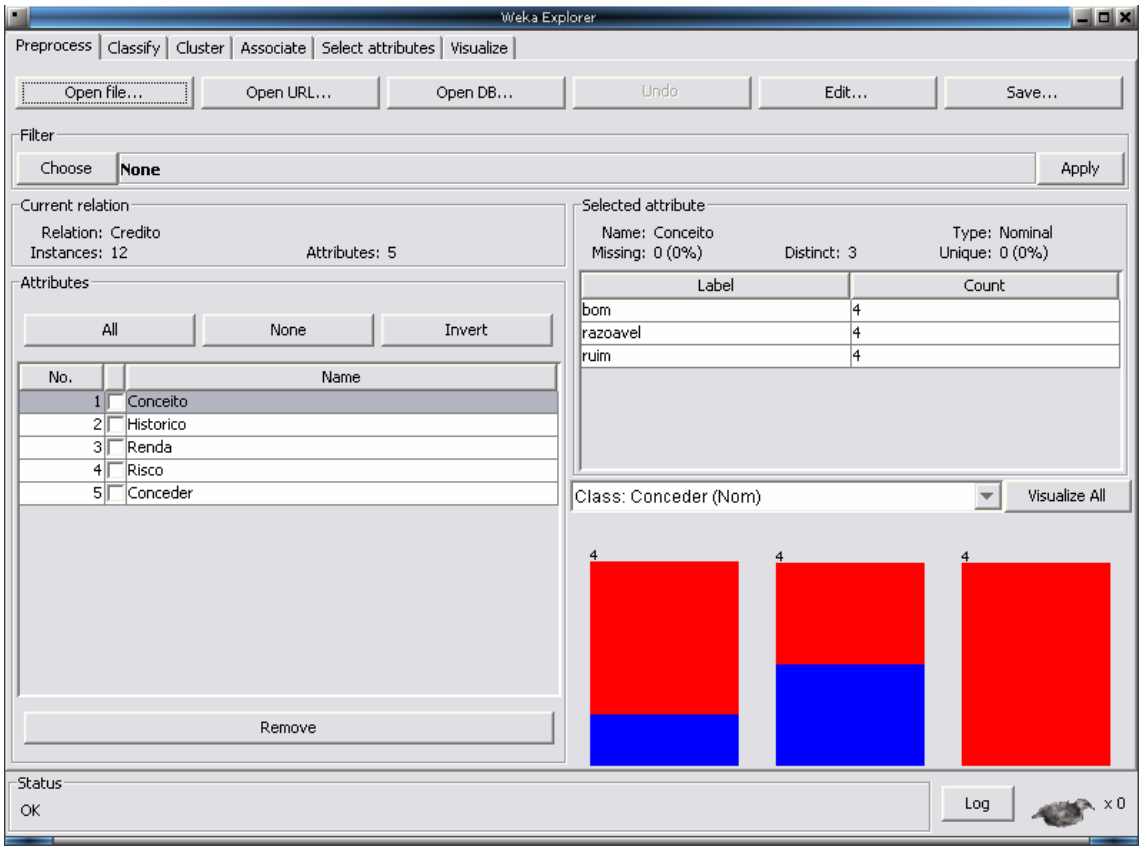

Figura 7. Interface do *Explorer* após abertura do arquivo *Credito.arff*.

O quadro *Current relation*, exibe o nome da relação (Credito), e as quantidades de instâncias (12) e atributos (5). Abaixo, há o quadro *Attibutes*, onde são exibidos os atributos presentes na relação, na ordem em que aparecem, juntamente com uma caixa de seleção, onde é possível selecionar cada um dos atributos individualmente. Os botões presentes na parte superior deste quadro – *All*, *None* e *Invert* – permitem selecionar todos ou nenhum dos atributos, bem como inverter a seleção realizada. Há, ainda, o botão *Remove*, cuja função é excluir um ou mais atributos selecionados.

Em *Selected attribute* são exibidas informações sobre o primeiro atributo da relação, como: nome (*name*); tipo (*type*), podendo ser nominal ou numérico; ausente (*missing*), que representa a quantidade de instâncias nas quais este atributo não consta; distinto (*distinct*), referente à quantidade de valores diferentes que este atributo assume nos dados; e único (*unique*), que indica quantas instâncias possuem um valor para este atributo que não se repete em nenhuma das outras instâncias. Abaixo são exibidos os rótulos (*Label*) existentes para o atributo, com a contagem (*Count*) de vezes em que cada um aparece nos dados. Ainda mais abaixo, há um histograma colorido que, com base na classe selecionada, indica de que maneira o rótulo se distribui pelas categorias existentes. Também é possível visualizar

gráficos para todos os atributos, ao clicar em *Visualize* All. Todas as informações presentes neste quadro são alteradas de acordo com o atributo selecionado no quadro *Attibutes*.

A etapa de pré-processamento oferece alguns filtros, no quadro *Filter*. Ao clicar em *Choose*, abre-se uma caixa com os filtros disponíveis. Há filtros supervisionados e não-supervisionados. Nesta etapa é possível, também, editar os dados da relação, através do botão *Edit*, e armazenar as alterações realizadas, clicando no botão *Save*. Qualquer alteração efetuada indevidamente nos dados, como a remoção de atributos ou a aplicação de filtros, por exemplo, pode ser desfeita pelo botão *Undo*.

No exemplo selecionado para demonstração, não foram aplicados filtros, uma vez que não foram encontrados maiores detalhes sobre suas funções, na documentação do programa.

A segunda etapa do processo de construção de árvore de decisão ocorre na aba *Classify*, demonstrada na Figura 8.

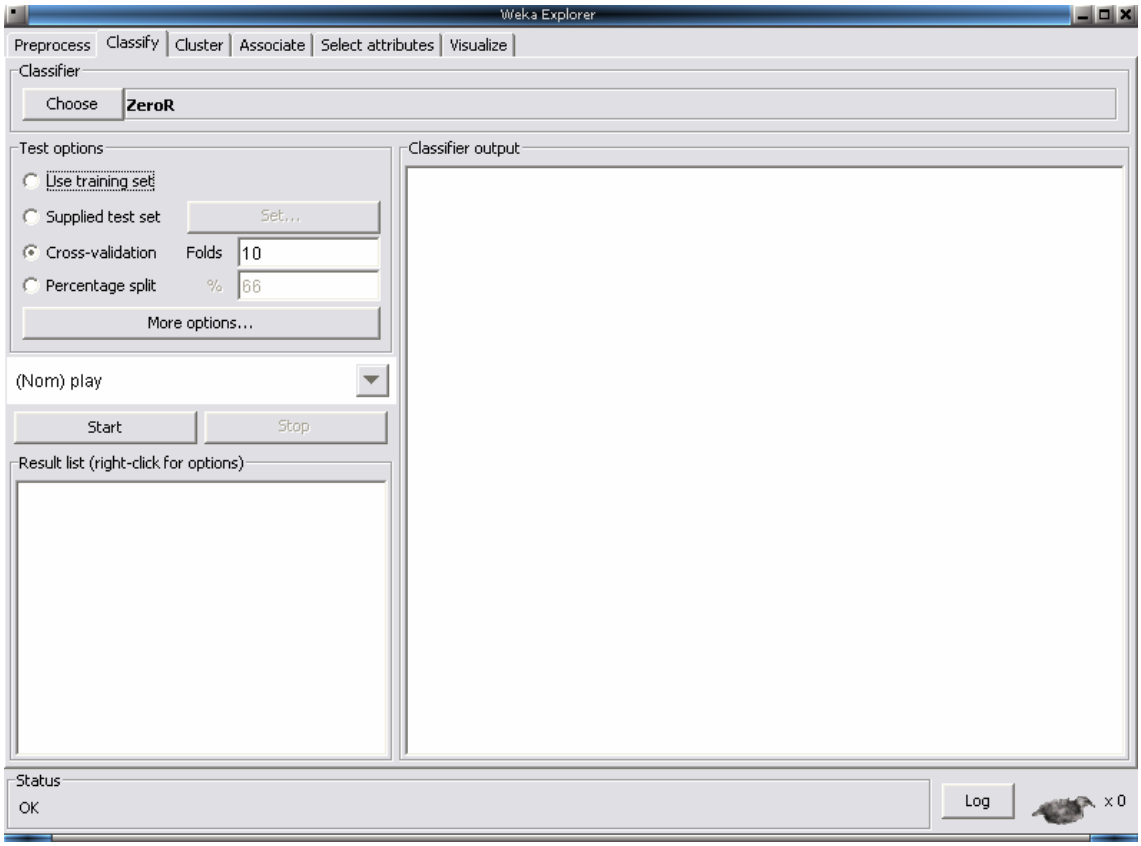

Figura 8. Interface do *Explorer*, na aba *Classify*.

Na aba *Classify*, a primeira opção disponível é a de seleção do classificador, que deve ser escolhido através do botão *Choose*. Como é possível verificar na Figura 9, *Weka* oferece uma grande variedade de classificadores.

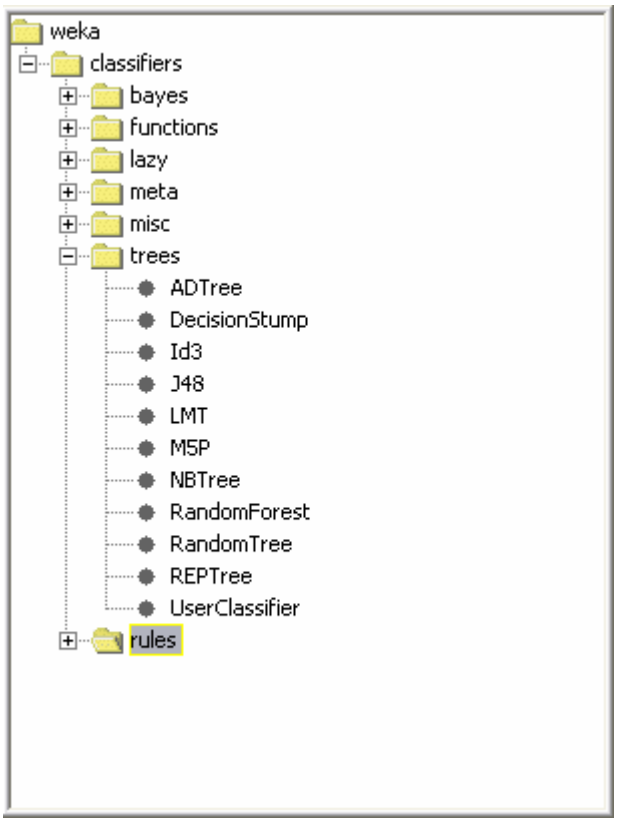

Figura 9. Caixa de seleção de classificadores.

Para o exemplo ora estudado, o classificador selecionado é o *J48*, uma implementação em Java do algoritmo C4.5, de Quinlan, para geração de árvores de decisão.

Após a seleção do classificador, é possível obter maiores informações sobre o algoritmo, clicando com o botão direito do mouse sobre o quadro *Classifier*, ou mesmo configurar algumas opções para seu comportamento, conforme mostra a Figura 10.

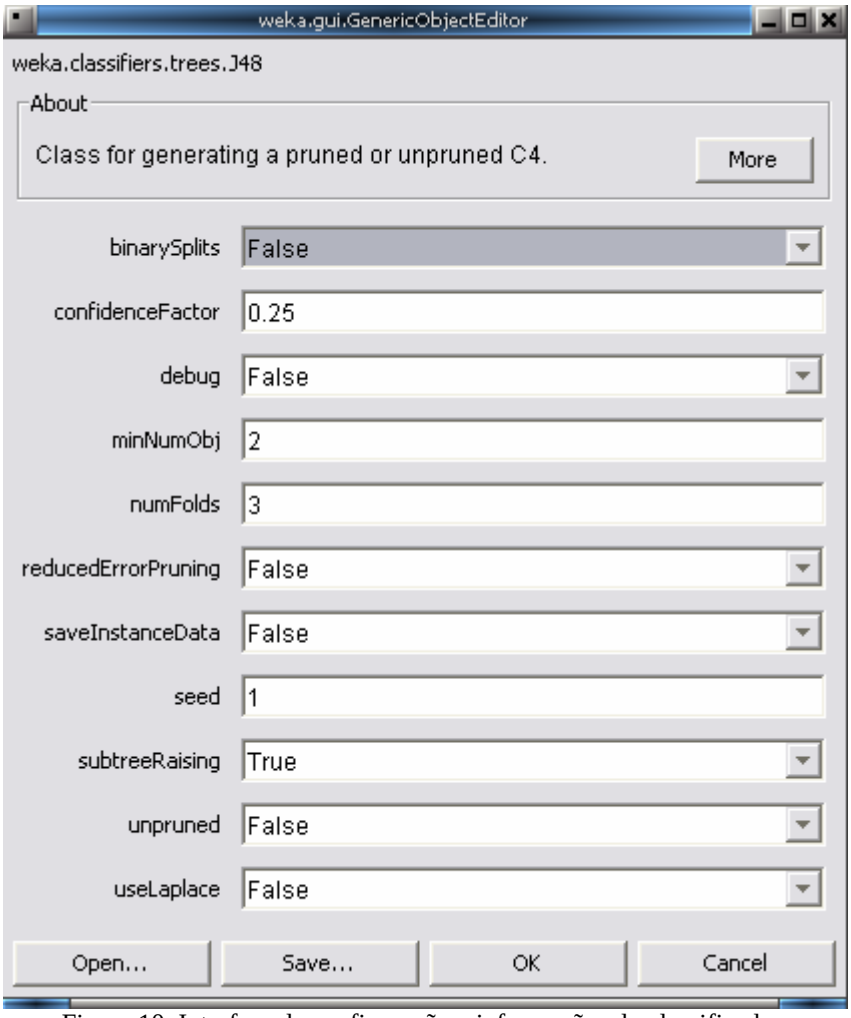

Figura 10. Interface de configuração e informações do classificador.

O botão *More* detalha informações sobre o algoritmo selecionado e sobre as opções de configuração disponíveis nesta tela.

Se forem aplicadas as configurações padrões do algoritmo para classificar o exemplo *Credito.arff* o resultado não será uma árvore de decisão. Para entender a razão disso, é necessário consultar a ajuda de configuração do algoritmo, que revela que a opção *minNumObj* (demonstrada na Figura 10) a se refere ao número mínimo de instâncias por folha da árvore, cujo valor padrão é *2*. Além disso, a opção *binarySplits*, cujo valor padrão é *False*, indica se deve ser utilizada divisão binária para atributos nominais, quando da construção da árvore. Assim, considerando que o exemplo a classificar – *Credito.arff* – dispõe de um conjunto pequeno de treinamento e, como já visto no capítulo 1 (Figura 3), resulta em uma árvore binária, para obter resultados mais relevantes, o valor de *minNumObj* deve ser alterado para *1*, e o valor de *binarySplits* deve ser mudado para *True*. Um teste com esses ajustes resulta em uma árvore de decisão semelhante à construída manualmente no capítulo 1.1.1.1

(Figura 3). A árvore gerada pelo classificador pode ser visualizada na Figura 11.

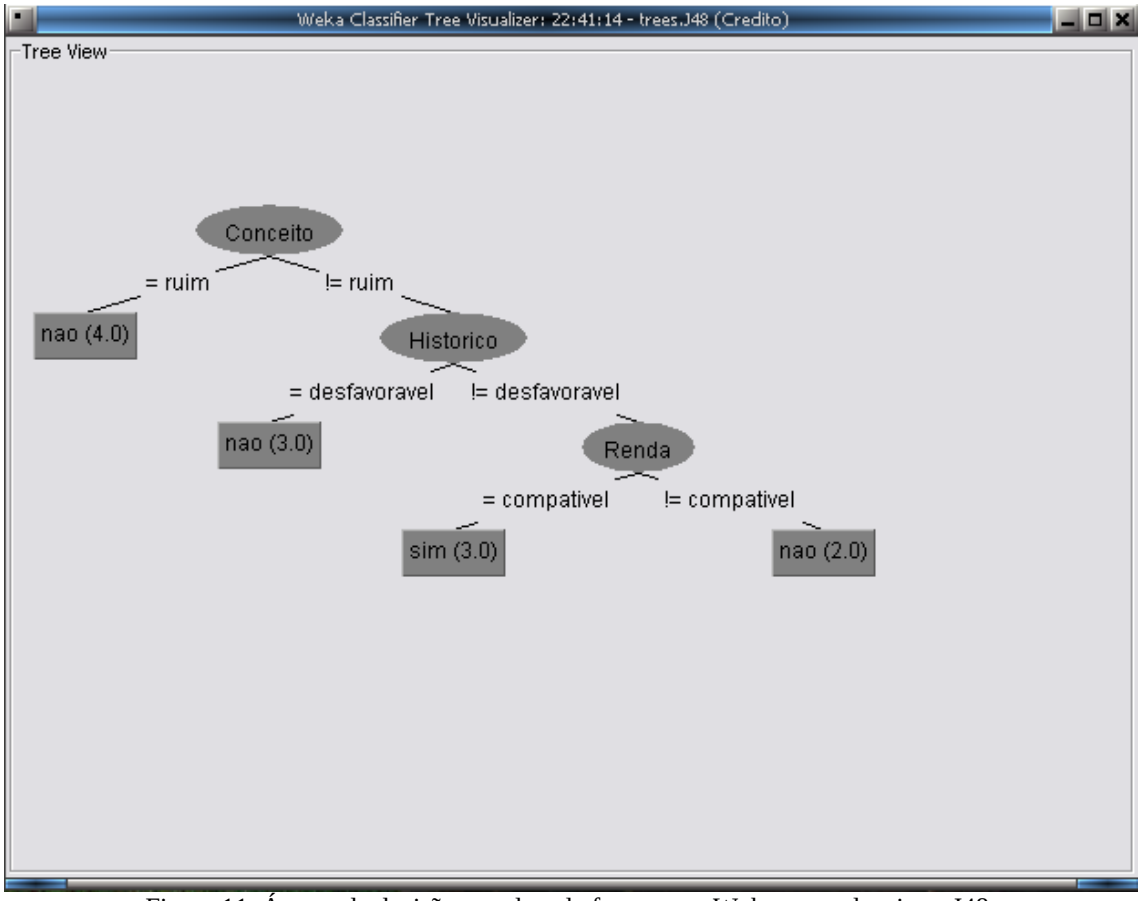

Figura 11. Árvore de decisão gerada pela ferramenta Weka, com algoritmo J48.

A árvore de decisão gerada é bem simplificada, pois possui apenas uma instância por folha, o que não pode ser detectado caso o algoritmo estiver em sua configuração padrão – *minNumObj = 2*.

CONCLUSÃO

Wives (1999) baseia-se em técnicas de agrupamento – aprendizado não-supervisionado – capazes de identificar similaridades nos textos e agrupá-los de acordo com o grau de semelhança. Martins (2003), por outro lado, utiliza a técnica de classificação Support Vector Machines – SVM (VAPNIK, 1995, apud MARTINS, 2003) para classificar páginas da Internet contendo preços de produtos, numa classificação binária para apenas uma classe. Dentre os trabalhos relacionados, nenhum realiza testes com textos da área de computação, ou propõe um modelo de categorização para esta área, o que não permite uma comparação direta, tanto entre si, como com o presente trabalho.

A metodologia empregada por Martins (2003) na classificação por SVM (VAPNIK, 1995, apud MARTINS, 2003), entretanto, pode ser adaptada para realizar a classificação proposta por este trabalho, que utilizará o algoritmo j4.8, uma adaptação do algoritmo c4.5 para a plataforma Java, disponibilizada pela ferramenta Weka.

Entre os objetivos traçados no anteprojeto deste trabalho, foram concluídas as seguintes etapas: estudo de técnicas de aprendizagem supervisionada, de árvores de decisão e de métodos de classificação de textos, além da taxonomia do curso de Ciência da Computação, aqui definida.

Embora a taxonomia definida neste trabalho não tenha sido realizada através de entrevistas com professores do curso, conforme inicialmente planejado, ela será devidamente submetida à apreciação de alguns integrantes do corpo discente deste curso para sugestões de alteração ou complementação na segunda parte do trabalho.

Além da consulta a ser realizada para homologação da taxonomia, a segunda etapa

deste trabalho compreenderá as seguintes tarefas: a) estudar o comportamento prático de algoritmos e técnicas de árvores de decisão; b) selecionar e classificar manualmente as monografias que comporão os conjuntos de aprendizagem e testes para o algoritmo; c) desenvolver o protótipo de aplicação para classificação semi-automática de textos; d) avaliação do protótipo; e) avaliar os resultados.

REFERÊNCIAS BIBLIOGRÁFICAS

AAS, Kjersti; EIKVIL, Line. **Text Categorisation: A Survey**. Norwegian Computing Center, 1999. Noruega, 1999. 37p.

ABREU, Sandra Collovini de. **Análise de expressões referenciais em corpus anotado da Língua Portuguesa**. UNISINOS, 2005. Dissertação de Mestrado (pós-graduação em computação aplicada). Ciências Exatas e Tecnológicas, Universidade do Vale do Rio dos Sinos, 2005. 105 p.

BREIMAN, FRIEDMAN, OLSHEN & STONE, 1984, apud REZENDE, 2003.

COMPUTER SCIENCE IN THE YAHOO! DIRECTORY. Disponível em: <http://dir.yahoo.com/Science/Computer_Science/>. Acesso em 15/11/2007.

FERNEDA, Edberto. **Redes neurais e sua aplicação em sistemas de recuperação de informação**. Ci. Inf., Brasília, v. 35, n. 1, p. 25-30, jan./abr. 2006.

FERREIRA. Aurélio Buarque Holanda. **Aurélio Século XXI, O Dicionário da Língua Portuguesa**. Editora Positivo, 3.ed.. 2004.

GOOGLE DIRECTORY - COMPUTERS > COMPUTER SCIENCE. Disponível em: <http://dir.google.com/Top/Computers/Computer_Science/>. Acesso em 15/11/2007

GOOGLE RESEARCH. Disponível em: <http://research.google.com/pubs/papers.html>. Acesso em 13/09/2007.

LUGER, George F. **Inteligência Artificial: estruturas e estratégias para a solução de problemas complexos**. 4.ed.. Porto Alegre: Bookman, 2004. 774 p.

MARTINS Jr., José. **Classificação de páginas na Internet**. USP, 2003. Dissertação de Mestrado (mestrado em Ciências de Computação e Matemática Computacional). Instituto de Ciências Matemáticas e de Computação, USP – São Carlos, 2003. 81 p.

REZENDE, Solange Oliveira. **Sistemas inteligentes: fundamentos e aplicações**. 2003. Editora Manole Ltda. Barueri, SP.

RODRIGUES, Marco António dos Santos. **Árvores de classificação**. Universidade dos Açores, 2004/2005. Monografia. Departamento de Matemática da Universidade dos Açores, Portugal. 32 p.

RUSSEL, Stuart et al. **Inteligência Artificial**. 2.ed. Traduzido por: Vandenberg D. de Souza. Rio de Janeiro: Elsevier, 2004. 1021 p. Tradução de Artificial Intelligence.

SEBASTIANI, Fabrizio. **Machine learning in automated text categorization**. Consiglio Nazionale delle Ricerce, 2002. Itália, 2002. 47p.

WEKA 3 - DATA MINING WITH OPEN SOURCE MACHINE LEARNING SOFTWARE IN JAVA. Disponível em <http://www.cs.waikato.ac.nz/~ml/weka/>. Acesso em 24/12/2007.

EN:WEKA 3.4.11 – WEKADOC. Disponível em: <http://weka.sourceforge.net/wekadoc/index.php/en:Weka_3.4.11>. Acesso em 24/12/2007.

WIKIPÉDIA. Desenvolvido pela Wikimedia Foundation. Apresenta conteúdo enciclopédico. Disponível em <http://pt.wikipedia.org/w/index.php?title=Aprendizagem&oldid=7759044>. Acesso em: 22/11/2007.

WIKIPÉDIA. Desenvolvido pela Wikimedia Foundation. Apresenta conteúdo enciclopédico. Disponível em <http://pt.wikipedia.org/w/index.php?title=Aprendizagem_de_m%C3%A1quina&oldid=7854942>. Acesso em: 22/11/2007.

WIVES, Leandro Krug. **Um estudo sobre agrupamento de documentos textuais em processamento de informações não estruturadas usando técnicas de "clustering"**. UFRGS, 1999. Dissertação de Mestrado (pós-graduação em Ciência da Computação). Instituto de Informática, Universidade Federal do Rio Grande do Sul, 1999. 102 p.