CENTRO UNIVERSITÁRIO FEEVALE

FABRÍCIO LUÍS COELHO

CLASSIFICAÇÃO SEMI-AUTOMÁTICA DE MONOGRAFIAS

Novo Hamburgo, junho de 2008.

FABRÍCIO LUÍS COELHO

flcoelho@gmail.com

CLASSIFICAÇÃO SEMI-AUTOMÁTICA DE MONOGRAFIAS

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

Professor orientador: Rodrigo Rafael Villareal Goulart

Novo Hamburgo, junho de 2008.

RESUMO

Com a crescente expansão da Internet, localizar informações precisas fica 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 uma metodologia de classificação semi-automática para textos.

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 methodology to classify Science Computing monograph, using supervised learning.

Keywords: Text categorization. Artificial Intelligence. Supervised learning.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SVM	*Support Vector Machines*
CDU	Classificação Decimal Universal (Tabela)
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
GPL	*General Public License*
ACM	*Association for Computing Machinery*
TC1	Trabalho de Conclusão 1
TC2	Trabalho de Conclusão 2
XML	*eXtensible Markup Language*
ISO	*International Organization for Standardization*
RI	Conjunto composto de arquivos com resumo, palavras-chave e introdução de cada monografia
AS	Conjunto composto de arquivos com substantivos e adjetivos de cada monografia
S	Conjunto composto de arquivos com substantivos de cada monografia
CJ	Conjunto
W3C	*World Wide Web Consortium*

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. Para isso, no primeiro capítulo conceitua-se aprendizagem, os métodos utilizados na aprendizagem de máquina, com um exemplo de aplicação para ilustrar os conceitos, e algumas ferramentas que podem auxiliar nesse processo.

Algumas metodologias utilizadas na classificação de textos e processos que as integram são objeto do segundo capítulo, que reúne estudos de alguns trabalhos parcialmente relacionados a este.

O terceiro capítulo propõe uma taxonomia para as Ciências da Computação, objeto de discussão junto a parte do corpo doscente da Feevale, e cobre toda a parte prática de

preparação e organização dos textos para os experimentos realizados, esses detalhados e analisados no quarto capítulo.

Por fim, são apresentadas as conclusões do trabalho.

1. APRENDIZAGEM DE MÁQUINA

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.

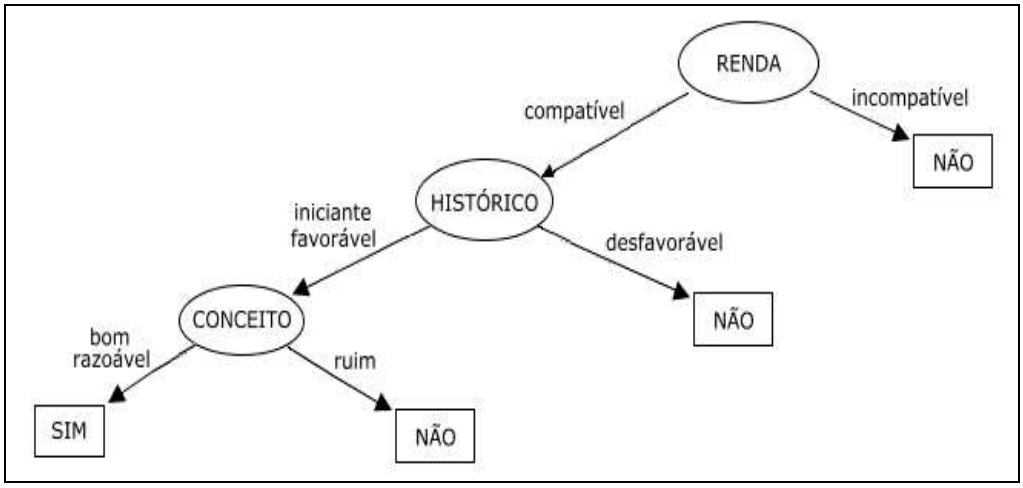

Figura 1: Árvore de decisão booleana

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.

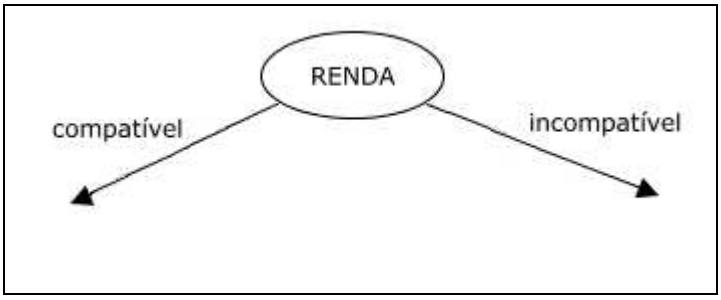

Figura 2: Ilustração parcial da árvore de decisão

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.

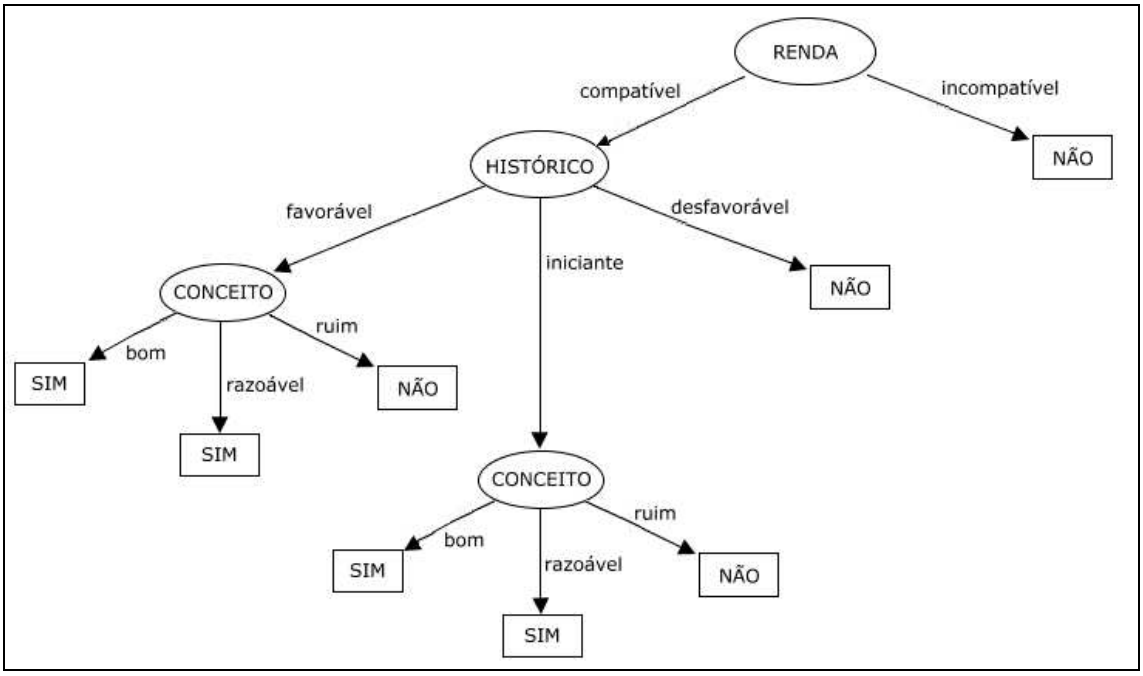

Figura 3: Ilustração da árvore de decisão resultante

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**	**IC**
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

CC = Corretamente classificado. IC = Incorretamente classificado.

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.

1.3. Ferramentas

Há diversas ferramentas para auxiliar no processo de classificação de textos. Três são brevemente descritas a seguir: *Weka*, *RapidMiner* e *Xtractor*.

1.3.1. 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.

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

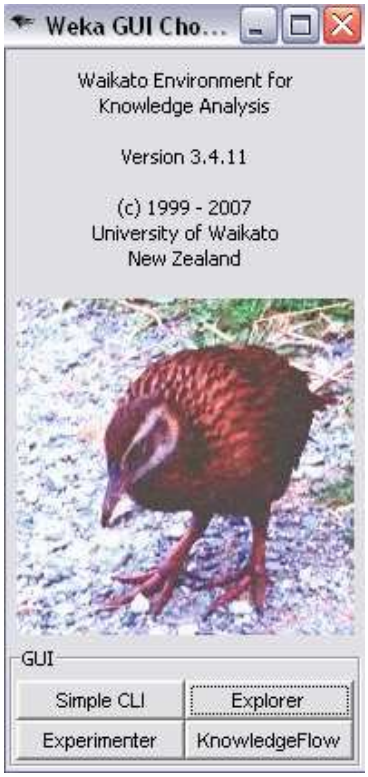

Figura 4: *Weka* - tela inicial

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].

[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.

Para iniciar a utilização do ambiente *Explorer*, 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.

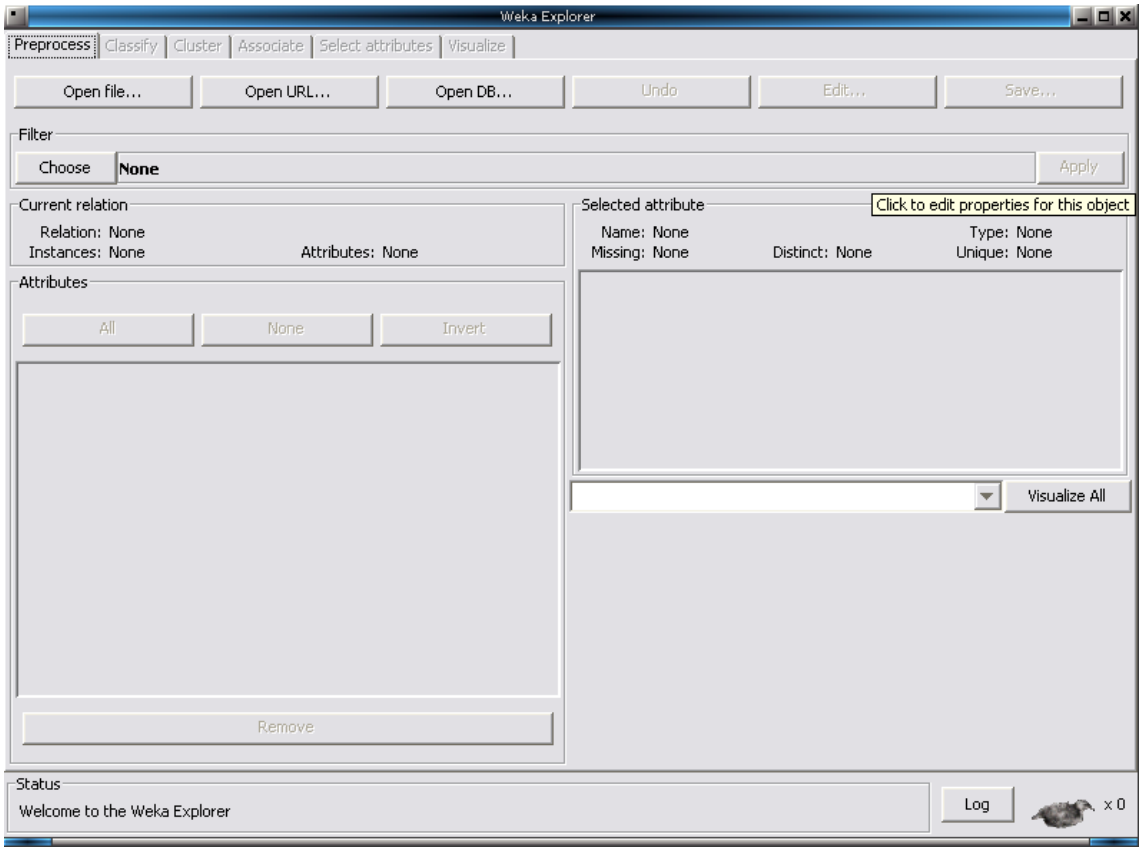

Figura 5: Interface do ambiente *Explorer* (*Weka*)

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.

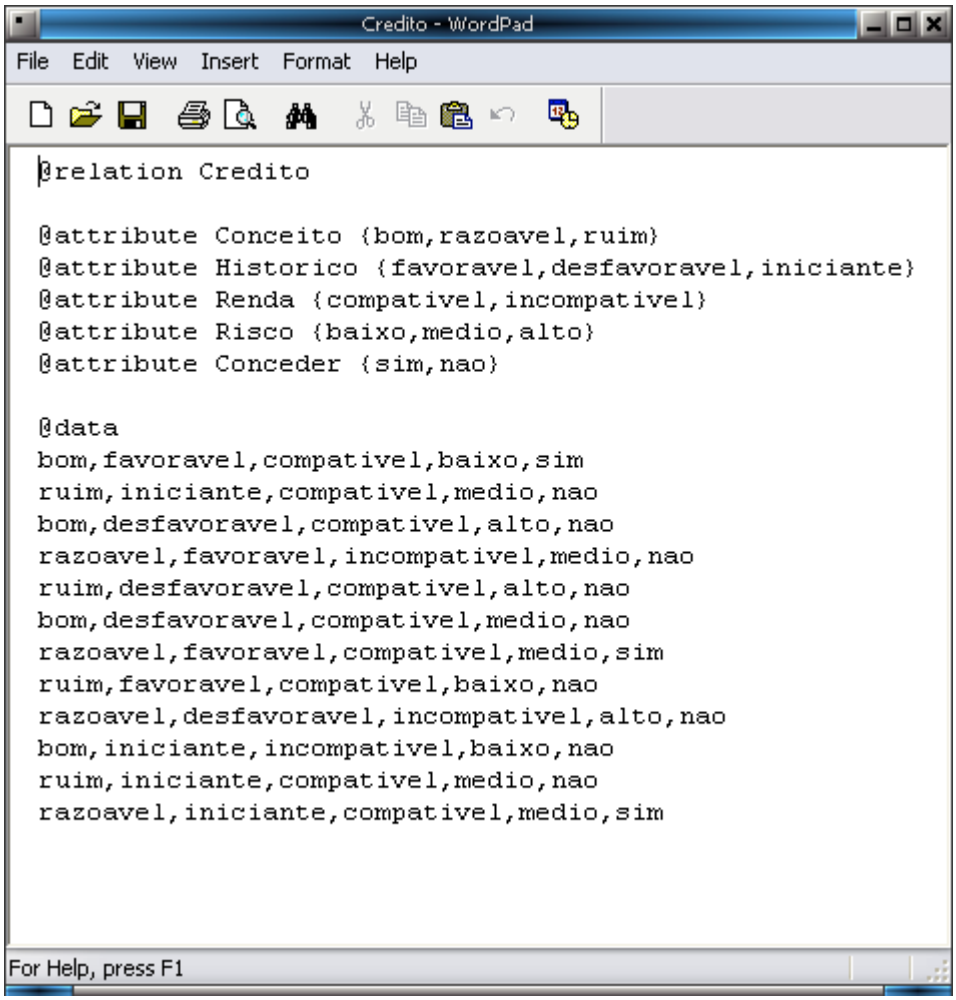

Figura 6: Conteúdo do arquivo Credito.arff

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.

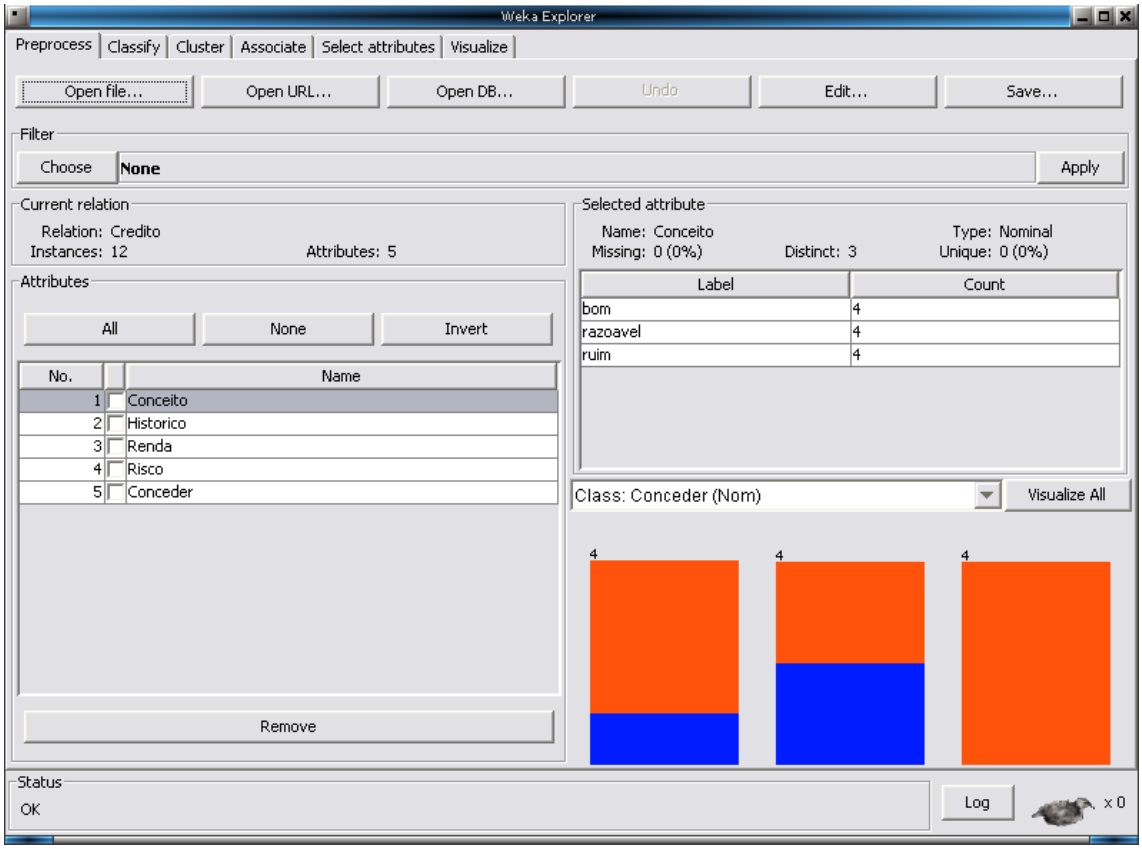

Figura 7: Interface do Explorer após abertura do arquivo Credito.arff

O quadro *Current relation*, exibe o nome da relação (Credito), e as quantidades de instâncias (12) e atributos (5). Abaixo, há o quadro *Attributes*, 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 *Attributes*.

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.

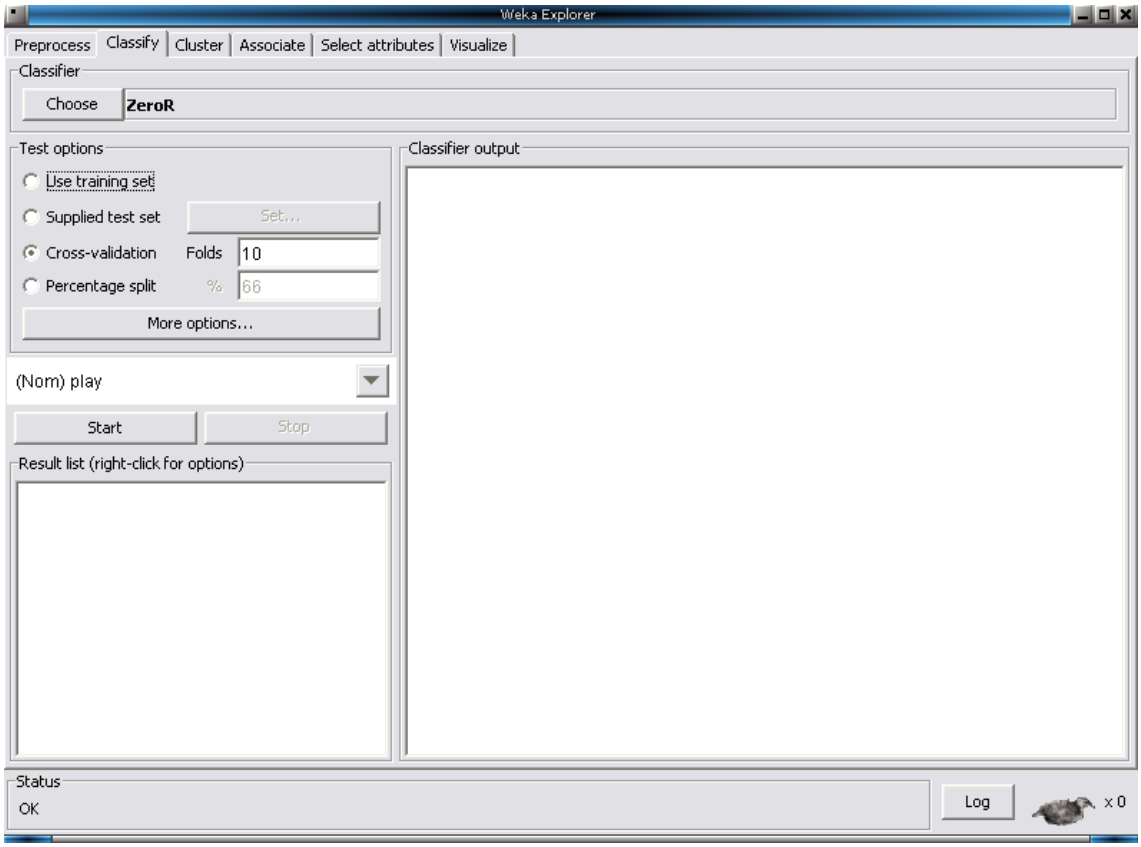

Figura 8: Interface do Explorer, na aba *Classify*

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.

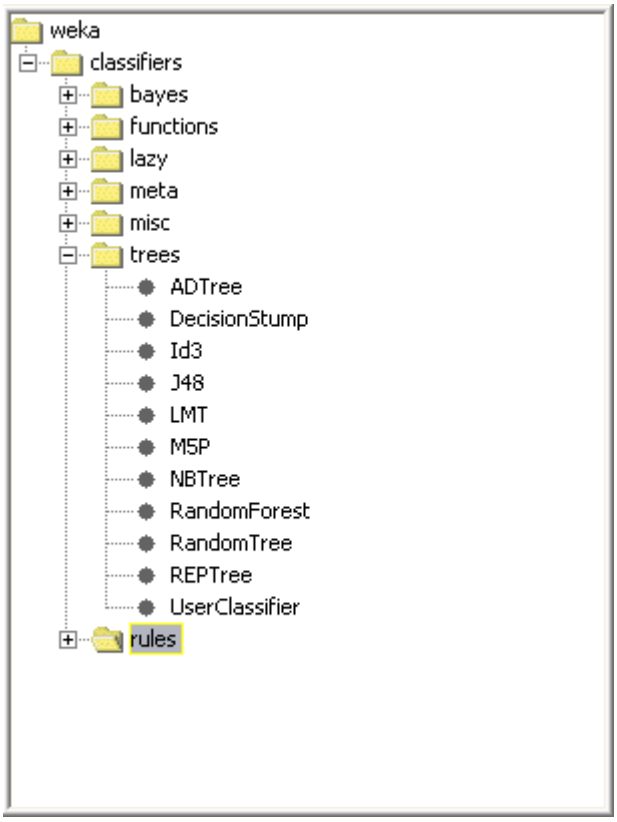

Figura 9: Caixa de seleção de classificadores

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.

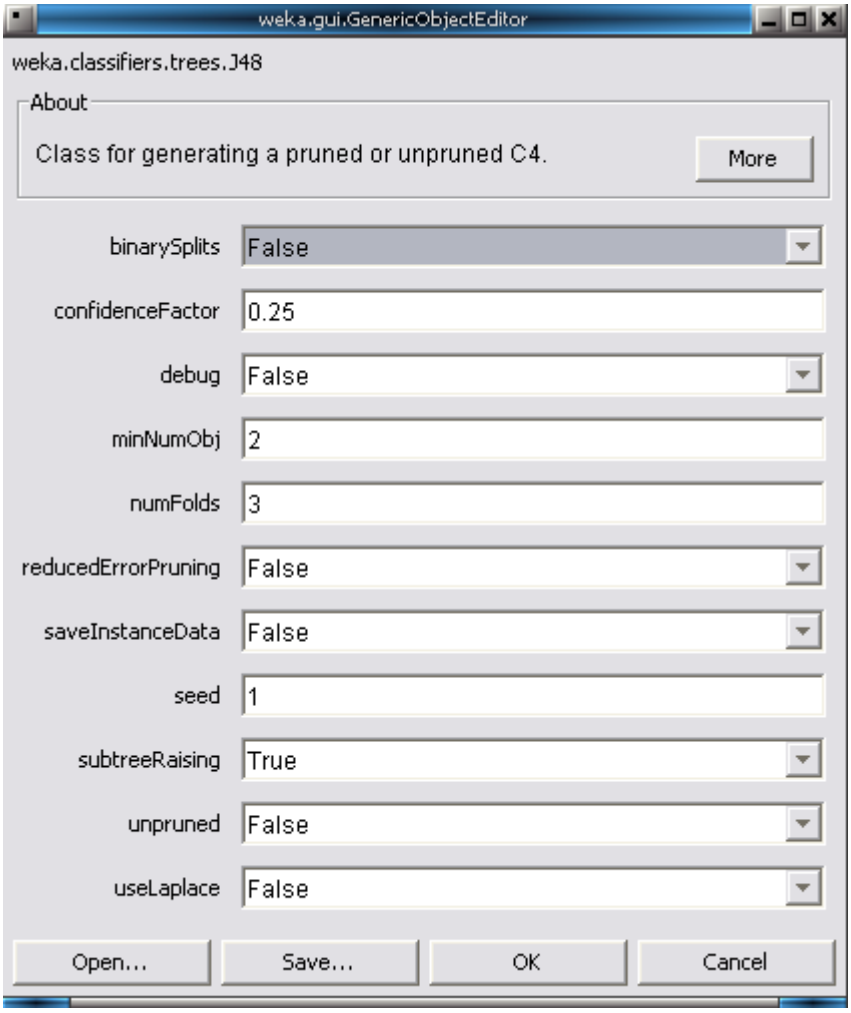

Figura 10: Interface de configuração e informações do classificador

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.

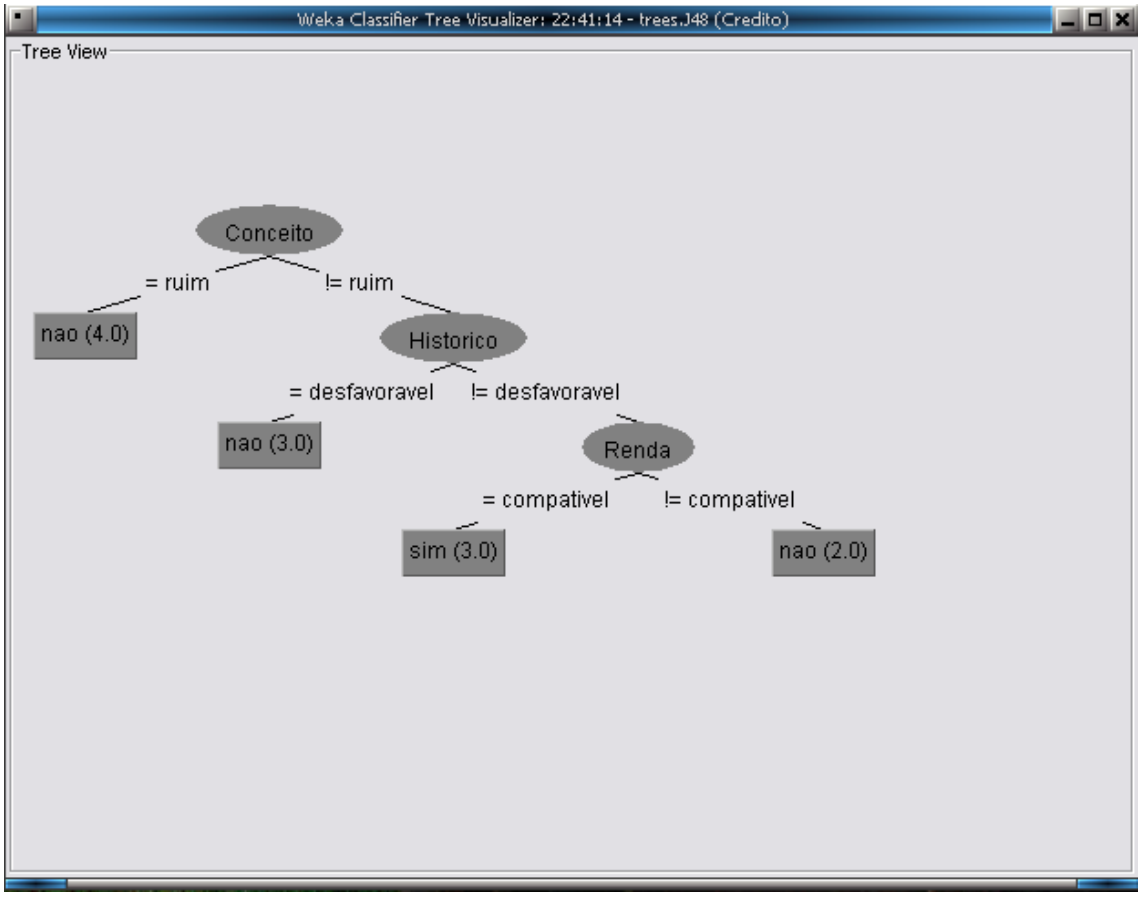

Figura 11: Árvore de decisão gerada pela ferramenta Weka, com algoritmo J48

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*.

1.3.2. RapidMiner

RapidMiner[3] é uma ferramenta com interface gráfica que conta com muitos recursos para diversas tarefas de mineração de dados. A versão utilizada nos experimentos foi a *Free* 4.1beta2, licenciada sob GPL versão 3, com *Text Plugin* 4.1beta2. O ambiente da ferramenta pode ser visualizado na Figura 12.

[3] RAPIDMINER FREE 4.1BETA2. Disponível em <http://rapid-i.com/>. Acesso em 02/06/2008.

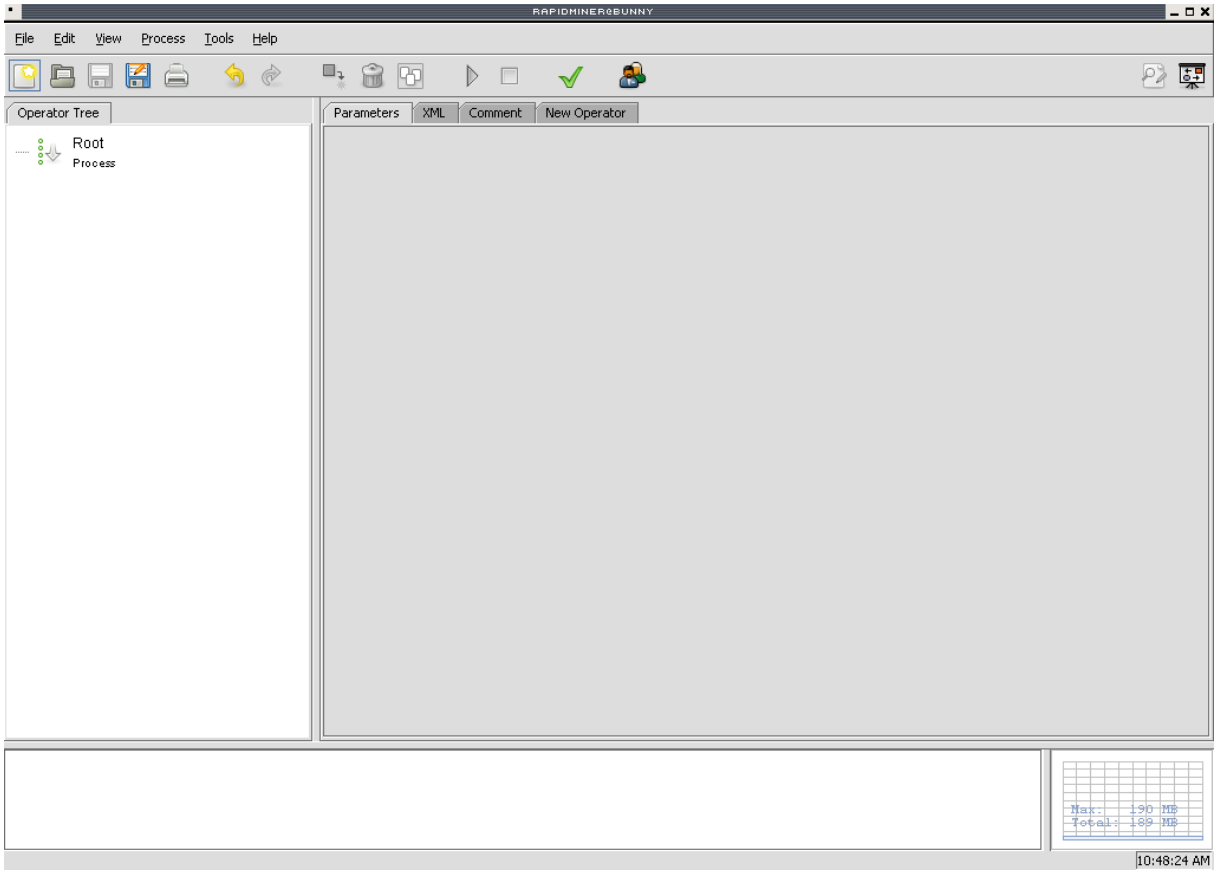

Figura 12: Ambiente de trabalho do *RapidMiner*

O *RapidMiner* opera com *processos*. Isto significa que, para realizar uma classificação de textos nesta ferramenta, é necessário incluir *operadores* para cada processo que antecede a tarefa final desejada. Alguns operadores possuem *parâmetros* configuráveis. A lista de operadores, sua seqüência de execução e parâmetros são armazenados em um arquivo XML de fácil edição.

A ferramenta conta com diversos operadores, para as mais diversas finalidades de mineração de dados. Para texto, por exemplo, há operadores para manipular um único texto de entrada (*SingleTextInput*) ou vários (*TextInput*), filtros com lista de *stopwords* para língua inglesa e alemã (*EnglishStopwordFilter* e *GermanStopwordFilter*) ou arquivo de *stopwords* (*StopwordFilterFile*), segmentadores de texto (*NGramTokenizer*, *StringTokenizer* e *NGramGenerator*), entre outros. Uma visão geral da variedade de operadores disponíveis pode ser visualizada na Figura 13.

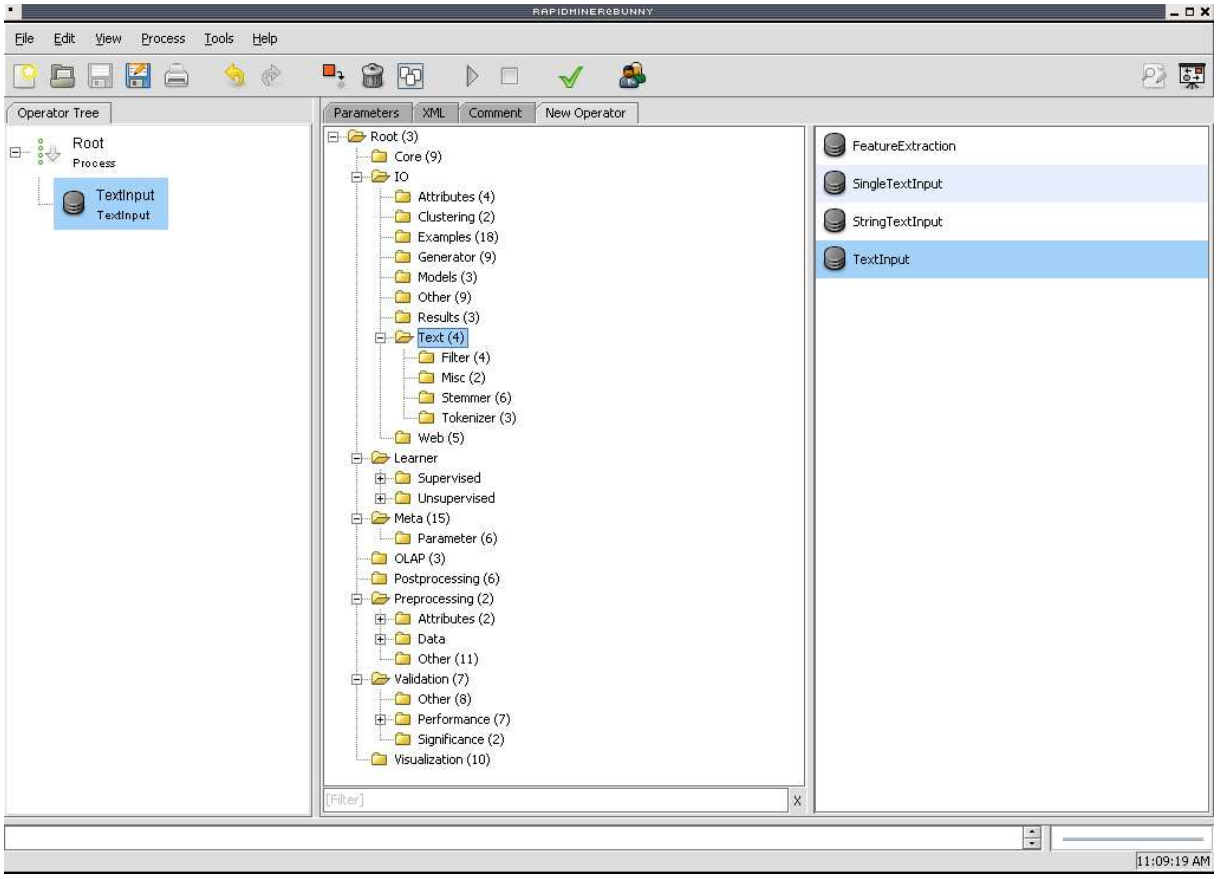

Figura 13: Amostra de operadores do RapidMiner

O operador *TextInput* é capaz de gerar vetores de palavras a partir de coleções de texto. Possui quatro métodos para geração dos vetores: *TermFrequency*, *TermOccurrences*, *BinaryOccurrences* e TFIDF.

1.3.3. Xtractor

A ferramenta *Xtractor*[4] (GASPERIN, 2003a) é o resultado de esforços conjuntos da Universidade do Vale do Rio dos Sinos (Unisinos) e a Universidade de Évora, de Portugal, tendo sido desenvolvida para padronizar a saída do parser PALAVRAS[5] (BICK, 2000, apud GASPERIN, 2003a), a partir da conversão dos arquivos gerados por este último para o formato XML. O parser, por sua vez, é parte de um conjunto de ferramentas denominado VISL (*Virtual Interactive Syntax Learning*), desenvolvido na *Southern University of Denmark*, por Ehhard Bick.

[4] XTRACTOR. Disponível em <http://abc.di.uevora.pt/xtractor/app/main>. Acesso em 02/06/2008.
[5] VISL Portuguese. Disponível em <http://visl.sdu.dk/visl/pt>. Acesso em 02/06/2008.

O PALAVRAS possibilita análise sintática e semântica de frases ou textos em diversos idiomas, inclusive o Português. Como resultado, apresenta marcações no texto de entrada, de acordo com o tipo de análise selecionada. Esse material é usado pelo *Xtractor* para gerar três arquivos distintos, no formato XML (*eXtensible Markup Language*): *.words*, *.POS* e *.chunks*.

O arquivo *.words* armazena uma lista com todas as palavras do texto, na ordem em que aparecem. O *.POS* tem as palavras em sua forma canônica e as respectivas informações morfossintáticas relacionadas. Já o *.chunks* carrega informações sintáticas da estrutura do texto. Esses três arquivos são utilizados em conjunto com folhas de estilo XSL, construídas conforme recomendação da W3C[6] (LOURENÇO, 2007), na construção de tesauros, explicada na seção 2.3, adiante.

[6] XSL Transformations (XSLT). Disponível em <http://www.w3.org/TR/xslt>. Acesso em 20/07/2008.

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.

Algumas 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*". 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 *stopwords*[7], 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 *stopwords*, cálculo da freqüência relativa, remoção de palavras pouco freqüentes,

[7] *Stopwords* são palavras frequentes que não caracterizam os textos. Ex.: artigos, proposições, pronomes etc.

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 *stoplist*, 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.

2.3. Combinações lingüísticas em classificação de textos

Lourenço (2007) elaborou algoritmos para construção semi-automática de tesauros[8]. Além de tesauros, esses algoritmos também produzem como resultado três listas de palavras mais freqüentes em cada documento: substantivos, adjetivos e verbos, todos em sua forma canônica. Nas listas, cada palavra é acompanhada de um número que corresponde à freqüência em que ocorre no documento. Deste modo, os algoritmos de Lourenço podem auxiliar na construção de dicionários com palavras do texto.

[8] Tesauros “pode ser entendido como uma ontologia, onde conceitos relacionados de forma hierárquica, por exemplo, estão representados por lexemas ou conjuntos de lexemas.”. (LOURENÇO, 2007, p.28).

Para definir qual das listas geradas pelos scripts de Lourenço é a mais adequada para classificação, foi verificado que Silva (2004) avalia os resultados de um pré-processamento baseado em combinações gramaticais na classificação de textos. As combinações utilizadas no trabalho, classificadas com a ferramenta *Weka*, são: substantivos, substantivos - adjetivos, substantivos - nomes próprios, nomes próprios - adjetivos e substantivos - nomes próprios - adjetivos. A autora também faz experimentos com a seleção de parcelas variáveis dos termos de cada categoria - 6, 12, 18, 24 e 30 termos mais freqüentes -, concluindo que as menores taxas de erro para árvores de decisão, na comparação com os métodos tradicionais, utilizando freqüência relativa[9] e TF-IDF[10], foram obtidas com as combinações substantivos - adjetivos e substantivos - nomes próprios - adjetivos.

Os resultados obtidos na dissertação de Silva (2004) trouxeram embasamento teórico para a proposta inicial, de selecionar apenas parte das palavras significativamente-freqüentes nos trabalhos para a classificação de textos. Sugere, inclusive, algumas opções de quantidade de termos freqüentes nos textos, para limitar o universo de palavras relevantes, ao realizar experimentos com combinações de 6, 12, 18, 24 e 30 termos mais freqüentes.

Para resolver o problema da criação de vetores numéricos, necessários ao J48, foi empreendida nova pesquisa na Internet. Uma última opção seria desenvolver uma ferramenta própria para isso, o que foi desnecessário graças à descoberta da ferramenta *RapidMiner* (apresentada na seção 1.3.2). A ferramenta oferece um ambiente rico em recursos para mineração de dados e aprendizagem de máquina e dispõe de um *plugin* específico para construção de vetores de palavras e mineração de textos. Para completar, todos os algoritmos da ferramenta *Weka* estão disponíveis na *RapidMiner* - ambas foram desenvolvidas em Java, o que proporciona a compatibilidade. A ferramenta não apenas resolveu o problema dos

[9] Freqüência relativa é calculada através da divisão da freqüência de um determinado termo em um documento pelo número total de temos existentes no documento (SILVA,2004, p.10).
[10] TF-IDF refere-se à freqüência do termo - freqüência inversa de documentos, na sigla em inglês. É calculada dividindo-se o número de vezes que determinado termo ocorre no documento pelo número de documentos em que o mesmo termo aparece.

vetores numéricos, mas também permitiu realizar a classificação dos textos com uma grande variedade de opções de configuração.

3. METODOLOGIA DE CLASSIFICAÇÃO PROPOSTA

O objetivo inicial deste trabalho era desenvolver um protótipo de aplicação para classificar monografias de conclusão, do curso de Ciências da Computação, de forma semi-automática, a partir de técnicas de aprendizagem supervisionada. A partir das pesquisas e dos experimentos realizados, entretanto, optou-se por realizar a descrição da metodologia mais adequada para realizar a tarefa. Esta metodologia foi utilizada neste trabalho e é sucintamente descrita abaixo:

1. Seleção de trabalhos;
2. Criação de arquivos, em formato TXT, com os resumos, palavras-chave e introduções dos trabalhos selecionados;
3. Submissão dos arquivos criados no passo 2 à ferramenta *Xtractor*, descrita na seção 1.3.3;
4. Submissão dos arquivos gerados no passo 3 aos *scripts* de Lourenço (2007), apresentados na seção 2.3;
5. Criação de novos arquivos, contendo a seleção dos dez substantivos mais freqüentes de cada monografia, a partir dos resultados obtidos no passo 4;
6. Realização da classificação com a ferramenta *RapidMiner*, introduzida na seção 1.3.2, utilizando o algoritmo treinado neste trabalho.

Nas seções a seguir serão detalhadas a taxonomia desenvolvida para as Ciências da Computação, a forma como as monografias foram selecionadas e classificadas e o modo como os dados foram preparados para os experimentos.

3.1. 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!*[11], 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*[12], 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.

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

[11] COMPUTER SCIENCE IN THE YAHOO! DIRECTORY. Disponível em: <http://dir.yahoo.com/Science/Computer_Science/>. Acesso em 15/11/2007.
[12] GOOGLE DIRECTORY - COMPUTERS > COMPUTER SCIENCE. Disponível em: <http://dir.google.com/Top/Computers/Computer_Science/>. Acesso em 15/11/2007.

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.

3.1.1. Validação da taxonomia

As pesquisas realizadas na Internet, através dos principais portais de conteúdo e de discussões na Wikipédia, e na tabela de organização de livros da Biblioteca da Feevale, relatadas no capítulo 3, não possibilitaram encontrar uma taxonomia apropriada para atender as expectativas deste trabalho. A escolha foi, então, direcionada para as classes de artigos e documentos científicos do currículo Lattes, do CNPq. Como a metodologia inicialmente proposta para este trabalho estipulava o desenvolvimento da taxonomia por meio de entrevistas com professores do curso de Ciências da Computação, parte do corpo docente deveria ser consultada para validar a escolha.

Um critério precisaria ser estabelecido para definir quais professores seriam convidados a opinar sobre a adequabilidade das classes do currículo Lattes para representar a vasta área das Ciências da Computação. O critério selecionado foi consultar o grupo de orientadores das monografias que deviam ser selecionadas para classificação. Essa opção permitiu aproveitar a contribuição dos profissionais também na etapa de classificação manual dos trabalhos, a ser detalhada no capítulo seguinte.

A consulta aos docentes foi realizada por e-mail. A mensagem, enviada para dezessete professores, apresenta sucintamente este trabalho, explica a necessidade de uma taxonomia para o curso e demonstra as classes e subclasses escolhidas. Por fim, é solicitada a opinião do professor sobre o tema. Os resultados obtidos são demonstrados no Gráfico 1.

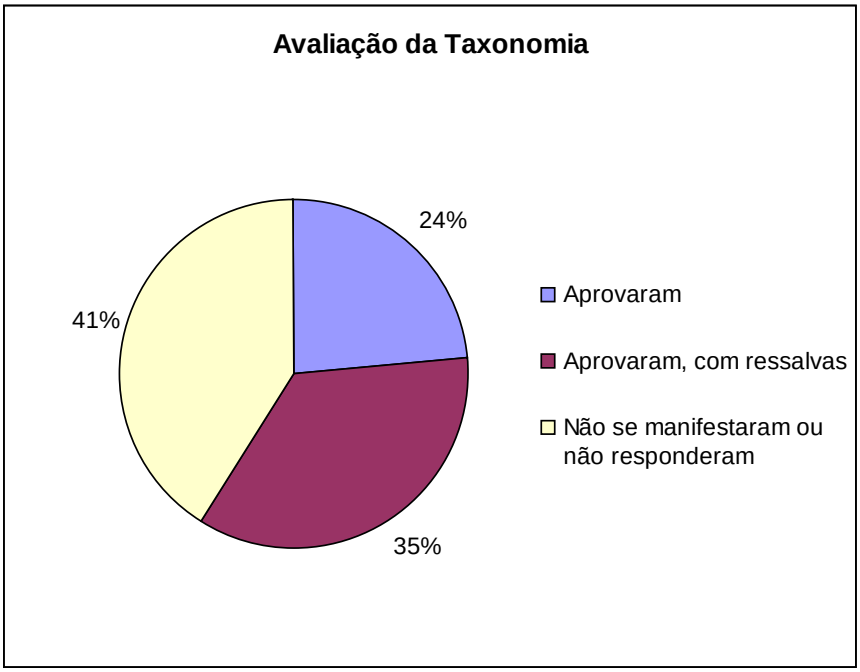

Gráfico 1: Avaliação da taxonomia

Nenhum orientador se mostrou contrário ao modelo escolhido, mas a maioria dos que se manifestaram, isto é, 60%, sugeriu correções ou melhorias. As sugestões dos professores que aprovaram a metodologia com ressalvas foram divididas em dois grupos: *taxonomia incompleta* e *taxonomia com inconsistências*. A relevância dessas opiniões pode ser visualizada no Gráfico 2.

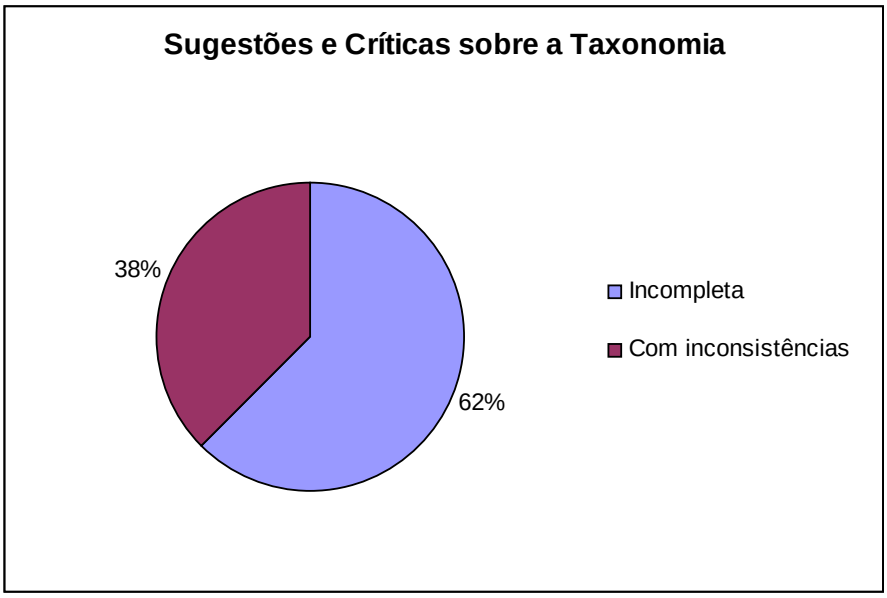

Gráfico 2: Grupos de opiniões

O grupo *taxonomia incompleta* compreende as opiniões relacionadas à falta de algumas subcategorias. Este grupo compõe a maior parte das manifestações e retrata a dificuldade de enquadrar alguns trabalhos nas subclasses existentes, já que foi solicitado aos avaliadores que determinassem como classificariam alguns dos trabalhos que orientaram. Nesse contexto, a Tabela 7 demonstra as subcategorias que, na avaliação desses profissionais, deveriam ser adicionadas à taxonomia, para ampliar sua abrangência.

Tabela 7: Subcategorias sugeridas pelos orientadores

Classe	Subclasse
Inteligência Artificial	Descoberta de Conhecimento
Sistemas de Computação	Redes de Computadores Segurança de Redes de Computadores
Metodologias e Técnicas da Computação	Compiladores

Também fazem parte deste grupo sugestões como utilizar conceitos de classificação de livros para escolha das categorias e fazer uma união da taxonomia proposta com a utilizada pela ACM.

No grupo *taxonomia com inconsistências* constam opiniões mais complexas. Numa delas, um professor questiona o enquadramento da subcategoria *Modelos Analíticos e de Simulação* na categoria *Matemática da Computação*. O docente argumenta que utiliza a simulação como técnica analítica sem trabalhar simulação computacional. Outro professor entende que a categoria *Inteligência Artificial* está mal estruturada, pois, segundo ele, os *Sistemas Especialistas* são parte dos *Sistemas Inteligentes*, assim como os *Sistemas Tutores*

Inteligentes. Ele sugere, ainda, incluir as subcategorias *Processamento de Linguagem Natural* e *Aprendizado de Máquina* - compreendendo *Redes Neurais* e *Algoritmos Genéticos*, entre outras -, e completa afirmando que a subcategoria *Teleinformática* deveria ser rebatizada como *Redes de Computadores*.

Considerando a contribuição dos professores, a taxonomia escolhida foi considerada adequada para este trabalho, uma vez que todos que se manifestaram a aprovaram, ainda que com algumas ressalvas.

3.2. Coleta de dados

A etapa de coleta de dados, neste trabalho, consiste na escolha das monografias a classificar. Esses dados foram selecionados, classificados manualmente e preparados para treinamento e teste de um algoritmo de aprendizagem de máquina.

Os dados selecionados deviam representar a taxonomia adequadamente, para proporcionar um bom conjunto de treino ao algoritmo de aprendizagem. Para isso, era necessário estabelecer um método de escolha. Por outro lado, a taxonomia proposta é recente e ainda não havia sido aplicada a trabalhos da Instituição, o que inviabiliza a composição, a priori, de um conjunto equilibradamente representativo.

A fonte dos dados foi a página[13] dos trabalhos de conclusão do curso de Ciências da Computação da Feevale, que oferece, entre outras informações e serviços, as monografias de *Trabalho de Conclusão I* (TC1), *Trabalho de Conclusão II* (TC2) e *Trabalhos Anteriores*, para consultar e copiar. Além desta forma simplificada de classificação, ainda é possível selecionar trabalhos por título, autor ou semestre de publicação, mas estas opções de seleção estão disponíveis apenas para a seção *Trabalhos Anteriores*.

Um aspecto priorizado na seleção foi que os trabalhos deveriam ser recentes, para permitir um melhor domínio dos orientadores sobre os trabalhos, visto que seriam convidados a contribuir na fase de classificação manual destes. Outra característica considerada importante foi que deveriam ser selecionados somente TC2, isto é, monografias completas.

[13] FEEVALE: TC-ONLINE. Disponível em <http://nead.feevale.br/tc/index.php>. Acesso em 02/06/2008.

Definidas as condições, optou-se por coletar todos os trabalhos desenvolvidos durante os dois semestres do ano 2007, num total de quarenta e cinco monografias, sendo quatorze do primeiro e trinta e uma do segundo semestres. A lista dos documentos selecionados pode ser conferida no ANEXO A.

3.3. Classificação manual das monografias

A tarefa de classificação manual geralmente é um trabalho demorado, que exige atenta leitura de cada monografia, além de conhecimento das áreas que compreendem cada uma das classes da taxonomia proposta para a Ciências da Computação.

Com o objetivo de tornar este processo mais ágil e digno de maior confiabilidade, os orientadores dos trabalhos selecionados foram questionados sobre qual a melhor classe para enquadrar cada uma das monografias. A contribuição dos professores, além de enriquecer processo e conferir maior credibilidade ao processo de classificação, uma vez que reflete a opinião dos profissionais sobre trabalhos desenvolvidos com sua participação, ainda contribuiu para validar a taxonomia escolhida para este trabalho, como relatado no capítulo 5.

Primeiro, foi necessário estender a planilha com a lista de documentos selecionados, acrescentando os nomes dos respectivos orientadores. Com os recursos da planilha eletrônica, os trabalhos foram ordenados por orientador. Cada professor, então, recebeu um *e-mail* com a lista de trabalhos em que figurou como orientador, contendo instruções sobre como poderia contribuir para este trabalho.

3.3.1. Resultado da classificação

Alguns orientadores - quatro, dos dezoito para os quais a mensagem fora enviada - não retornaram a mensagem enviada, mesmo após reiteração. As monografias correspondentes foram, então, classificadas pelo próprio autor deste trabalho. A Tabela 8 apresenta um resumo do resultado da classificação.

Tabela 8: Quantidade de monografias nas classes da taxonomia proposta

Classe	Quantidade de monografias
Inteligência Artificial	03
Matemática da Computação	01
Metodologias e Técnicas da Computação	25
Sistemas de Computação	15
Teoria da Computação	01

Em razão da reduzida quantidade de documentos selecionados e à baixa representatividade verificada em algumas classes, não foram incluídas as subclasses no resultado desta pesquisa.

O Gráfico 3 demonstra a representatividade de cada classe no conjunto das monografias selecionadas.

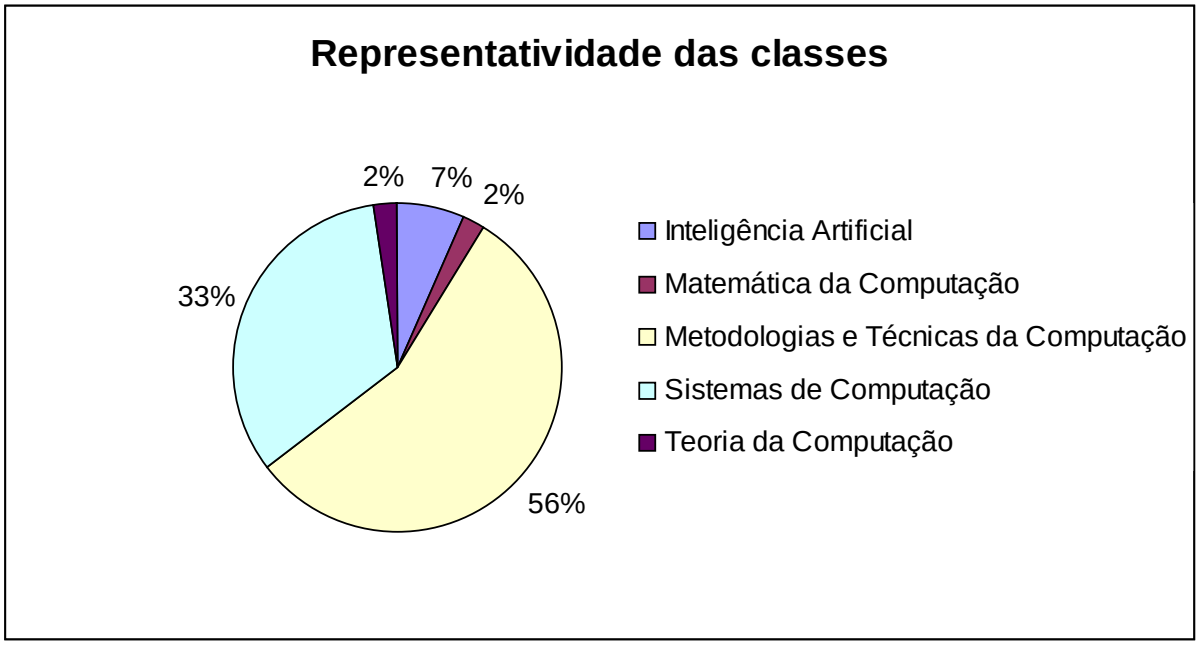

Gráfico 3: Trabalhos em cada classe

3.4. Preparação dos dados

Preparar os dados é a fase mais trabalhosa nos processos de mineração de textos. Em sua forma bruta, os documentos eletrônicos podem variar muito em características como codificação, formato, tamanho, extensão, etc. Assim, uma das primeiras etapas necessárias no processo é homogeneizar o material.

O objetivo desta primeira tarefa de preparação dos dados era converter todos os documentos para texto sem formatação, com extensão TXT.

3.4.1. Homogeneização dos documentos

As monografias selecionadas, num total de 45, foram obtidas no portal de trabalhos do Curso de Ciências da Computação da Feevale. Cada trabalho foi baixado do site individualmente, no formato em que estava disponível. O material, composto de 31 arquivos no formato PDF e 14 no formato DOC - alguns dos últimos, extraídos de arquivos compactados, no formato ZIP -, foi armazenado em diretórios distintos.

Num primeiro momento, destaca-se nos documentos a falta de padrão de formato e nomes. Alguns receberam o nome do autor, outros o título do trabalho. Certos trabalhos foram nomeados com um ou mais números, aparentemente aleatórios. Todos foram renomeados, num primeiro momento, recebendo o nome completo de seus respectivos autores.

Há várias maneiras de converter arquivos de um formato para outro. Como um dos objetivos deste trabalho é automatizar tarefas, foi iniciada uma procura na Internet por conversores para os formatos DOC e PDF.

No caso do formato DOC, o conversor utilizado foi o Catdoc[14]. O programa roda através de linha de comando Linux, o que facilitou a criação de script para realizar as conversões automaticamente. Nos experimentos realizados, o Catdoc reconheceu adequadamente a codificação do *Microsoft Word*, gerando resultado legível e fiel aos textos originais.

Para o formato PDF, há o conversor *PStoTEXT*[15], que também opera por linha de comando, em ambiente Linux. O programa foi testado com algumas das monografias, porém o resultado não se mostrou legível: muitos caracteres não são reconhecidos pelo programa, o que prejudica a formação de algumas palavras e até frases inteiras. Como o conversor não apresentou resultados com a qualidade necessária, seu uso foi descartado. Para realizar a conversão foi necessário, então, utilizar um método manual e rudimentar: cada arquivo foi aberto no programa *Adobe Acrobat Reader*, tendo seu conteúdo selecionado e copiado para a

[14] CATDOC. Disponível em <http://directory.fsf.org/project/catdoc/>. Acesso em 31/05/2008.
[15] PSTOTEXT. Disponível em <http://pages.cs.wisc.edu/~ghost/doc/pstotext.htm>. Acesso em 02/06/2008.

área de transferência; um editor de texto convencional[16] foi aberto, o conteúdo da área de transferência foi colado no aplicativo e o arquivo foi salvo com o nome do autor, mais a extensão TXT.

3.4.2. Definição das características dos arquivos a classificar

Antes do início do processo de classificação e da continuidade da preparação dos dados ainda era preciso definir alguns pontos. Primeiro, se a classificação deveria ser realizada a partir do conteúdo integral de cada monografia, ou se contemplaria apenas alguns segmentos específicos destas. Segundo, se seriam realizados experimentos com combinações gramaticais e, caso positivo, quais combinações seriam utilizadas.

Em relação ao primeiro ponto, decidiu-se utilizar apenas alguns segmentos das monografias para o processo de classificação: *resumo, palavras-chave* e *introdução*. A justificativa para esta decisão está na representatividade dos segmentos escolhidos: o resumo traz um breve relato sobre todo o trabalho; as palavras-chave sintetizam bem o assunto; e a introdução faz uma abordagem geral sobre tudo o que é abordado em cada capítulo da monografia. O conjunto dos arquivos que contém as monografias foi denominado, então, M (monografias) e o dos arquivos com os segmentos selecionados de M foi denominado RI (resumo-introdução). Assim, pode-se dizer que $RI \in M$.

Optou-se, também, por realizar classificação com combinações gramaticais. As combinações escolhidas foram: *substantivos*; e *substantivos + adjetivos*. A opção é fortemente influenciada pelos resultados disponibilizados pelos scripts de Lourenço (2007) que, além das relações de substantivos e adjetivos mais freqüentes, oferece também a relação de verbos. Os conjuntos destes arquivos foram denominados S (substantivos) e SA (substantivos + adjetivos).

Mas há algumas inovações na escolha das combinações gramaticais, que diferem da abordagem utilizada por Silva (2004). Enquanto Silva utiliza conjuntos de termos mais freqüentes selecionados a partir de todo o conteúdo de cada texto, este trabalho monta as combinações a partir dos segmentos formados pelos resumos, palavras-chave e introduções de cada monografia, isto é, os mesmos conjuntos descritos anteriormente. Deste modo, pode-se

[16] O editor utilizado para esta tarefa foi o KATE, disponível na suíte KDE, para plataforma Linux.

dizer que S ∈ SA; SA ∈ RI. A Figura 14 representa conceitualmente os conteúdos de cada conjunto.

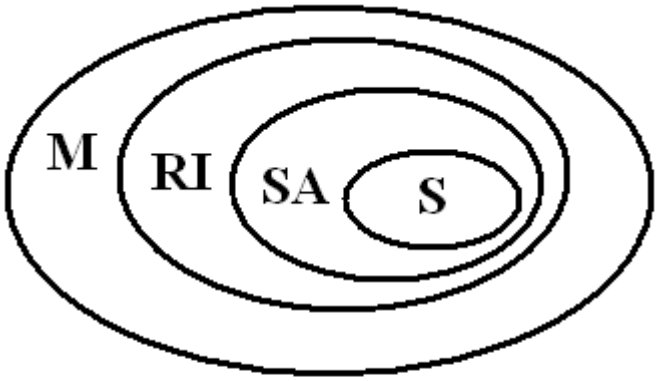

Figura 14: Representação conceitual dos conteúdos de cada conjunto

Há, também, outras características importantes nos conjuntos gramaticais que merecem destaque. O conjunto SA compreende todos os substantivos e adjetivos de RI. O conjunto S, por sua vez, é composto de arquivos que contém somente os dez substantivos mais freqüentes.

Definidas as características dos arquivos que comporiam cada conjunto, deu-se a continuidade ao processo de homogeneização dos documentos. Uma vez que todos os arquivos já estavam em formato TXT, foi necessário criar novos arquivos para compor o conjunto RI. Este processo foi realizado manualmente, isto é, cada monografia foi aberta na janela de um editor de texto, o resumo foi selecionado, copiado para a área de transferência e colado na janela de outro editor. Os mesmos passos foram realizados para as palavras-chave e para as introduções das monografias.

3.4.3. Pré-processamento para os conjuntos de informações lingüísticas

Para obtenção dos conjuntos SA e S, seria necessário utilizar os *scripts* elaborados por Lourenço (2007). Os *scripts* exigem a utilização da ferramenta *Xtractor*. Segundo o autor, esta ferramenta é usada para gerar as anotações lingüísticas de documentos textuais escritos em língua portuguesa.

Cada um dos 45 elementos do conjunto RI foi submetido ao *Xtractor*, por meio de *upload* no site da ferramenta. Em seguida, os documentos foram analisados sintaticamente em duas etapas exigidas pelo *Xtractor*, gerando como resultado três arquivos XML (*eXtensible Markup Language*) para cada original: um com a extensão "*words*", onde se encontra a seqüência integral das palavras que aparecem no texto; outro com a extensão "pos" (POS - Part-Of-Speech), em cujo conteúdo estão as palavras em sua forma canônica e informações morfossintáticas; e último com a extensão "xml" (*chunks*), que contém informações sintáticas da estrutura do texto.

Os 135 arquivos resultantes das transformações realizadas no *Xtractor* foram, então, submetidos aos scripts de Lourenço (2007), dos quais obtiveram-se as listas de adjetivos, substantivos e verbos, codificados em UTF-8, todos listados em ordem alfabética e acompanhados de um número que corresponde à quantidade de vezes em que aparecem no texto original.

Para gerar o conjunto S, foi desenvolvido um script que ordena os substantivos de acordo com a freqüência em que ocorrem nos arquivos e transfere para outro arquivo somente as dez últimas palavras. Outro script extrai os números (freqüência) dos novos arquivos, devolvendo os 45 elementos do conjunto S.

O conjunto SA, por sua vez, foi obtido através de um script que combina as listas de adjetivos e substantivos, retornando 45 arquivos híbridos. Outro script extrai os números que acompanham as palavras e ordena-as alfabeticamente, devolvendo os elementos do conjunto SA.

Considerando que os conjuntos SA e S tiveram sua codificação modificada de ISO-8859-1 para UTF-8, durante o processo, decidiu-se também converter o conjunto RI para este formato, a fim de preservar a similaridade dos conjuntos. Essa conversão foi realizada por outro script especialmente desenvolvido para este trabalho.

3.4.4. Composição dos conjuntos de treino e teste

Por tratar-se de técnica de aprendizagem supervisionada de máquina, a construção de árvores de decisão requer conjuntos distintos para treino e teste. Para suprir esta necessidade, decidiu-se dividir as monografias em três grupos - A, B e C -, utilizando as classes como

critério de distribuição dos documentos. A Tabela 9 demonstra a distribuição de monografias em cada grupo, considerando suas classes.

Tabela 9: Distribuição das monografias nos grupos

Classe	Grupo A	Grupo B	Grupo C
Inteligência Artificial	1	1	1
Matemática da Computação	1	0	0
Metodologias e Técnicas da Computação	8	8	9
Sistemas de Computação	5	5	5
Teoria da Computação	0	1	0
Total	15	15	15

Esta distribuição foi idealizada para permitir uma validação cruzada dos grupos no processo de classificação. Isto é, na etapa de classificação, que seria realizada posteriormente, os grupos seriam combinados para compor os conjuntos de treino e teste. Assim, seriam realizadas três etapas de treinamento, com os grupos AB, BC e CA - onde AB = A $\cup$ B; BC = B $\cup$ C; e CA = C $\cup$ A -, e três etapas de teste, respectivamente com os grupos C, A e B.

Como existiam três conjuntos distintos de arquivos para realizar experimentos - RI, SA e S -, estes foram organizados em três grandes conjuntos de treino e teste, a saber:

- Conjunto 1 (CJ1) - Pasta (diretório) com os 45 elementos do conjunto **RI**, distribuídos em três pastas correspondentes aos grupos A, B e C. Os elementos de cada grupo, por sua vez, foram divididos em cinco pastas, de acordo com suas respectivas classes, conforme a Tabela 9.
- Conjunto 2 (CJ2) - Pasta (diretório) com os 45 elementos do conjunto **S**, distribuídos em três pastas correspondentes aos grupos A, B e C. Os elementos de cada grupo, por sua vez, foram divididos em cinco pastas, de acordo com suas respectivas classes, conforme a Tabela 9.
- Conjunto 3 (CJ3) - Pasta (diretório) com os 45 elementos do conjunto **SA**, distribuídos em três pastas correspondentes aos grupos A, B e C. Os elementos de cada grupo, por sua vez, foram divididos em cinco pastas, de acordo com suas respectivas classes, conforme a Tabela 9.

A Figura 15 demonstra graficamente a composição de cada conjunto.

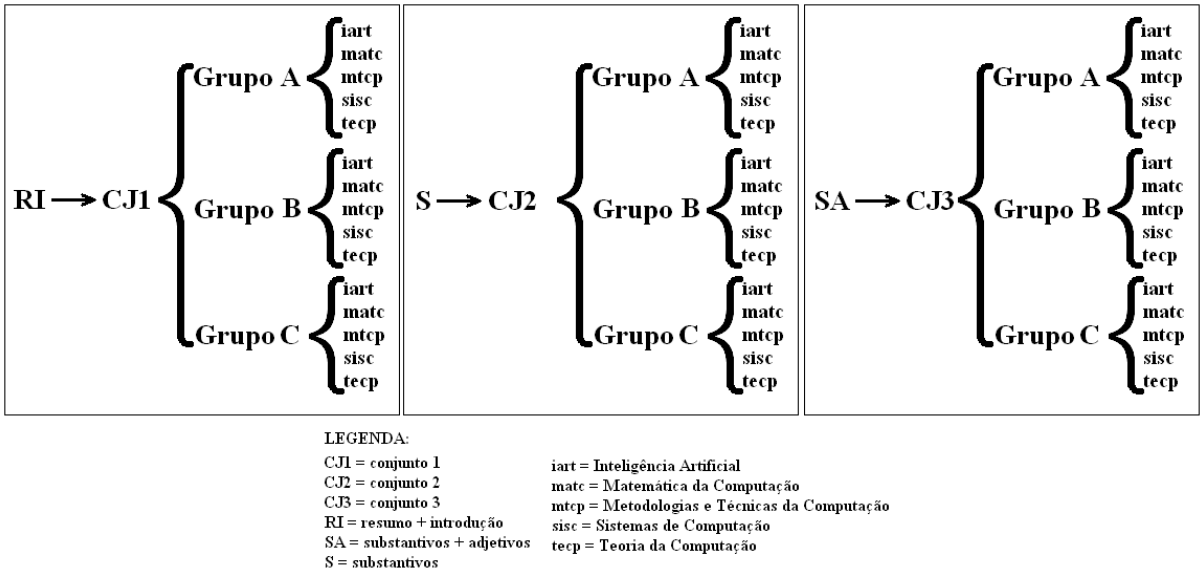

Figura 15: Organização dos conjuntos

3.4.5. Outros conjuntos utilizados em simulações

Após a realização dos primeiros experimentos com os grupos e conjuntos descritos na seção anterior, percebeu-se a necessidade de ampliar o universo de simulações, para proporcionar melhores conclusões sobre os métodos mais adequados para classificação das monografias.

Cinco novas monografias, de períodos, autores e orientadores aleatórios, foram selecionadas, com a intenção de melhorar a acurácia do algoritmo na caracterização de classes como Inteligência Artificial e Teoria da Computação. Os novos textos foram preparados como os antecessores - seleção de resumos, palavras-chaves e introduções; ferramenta *Xtractor*; *scripts* de Lourenço (2007); seleção dos dez substantivos mais freqüentes; e combinação de substantivos + adjetivos - e distribuídos nos grupos A, B e C em novos conjuntos. Para evitar que os grupos sejam confundidos com os originais, os novos grupos foram denominados X, Y e Z, respectivamente.

Tabela 10: Composição dos novos grupos

Classe	Grupo X	Grupo Y	Grupo Z
Inteligência Artificial	2	2	1
Matemática da Computação	1	0	0
Metodologias e Técnicas da Computação	8	8	9
Sistemas de Computação	5	5	5
Teoria da Computação	1	2	1
Total	17	17	16

A partir das novas composições, foram criados os conjuntos 4, 5 e 6 (CJ4, CJ5 e CJ6), conforme abaixo:

- Conjunto 4 (CJ4) - Pasta (diretório) com os 50 elementos do conjunto **RI**, distribuídos em três pastas correspondentes aos grupos X, Y e Z. Os elementos de cada grupo, por sua vez, foram divididos em cinco pastas, de acordo com suas respectivas classes, conforme a Tabela 10.
- Conjunto 5 (CJ5) - Pasta (diretório) com os 50 elementos do conjunto **S**, distribuídos em três pastas correspondentes aos grupos X, Y e Z. Os elementos de cada grupo, por sua vez, foram divididos em cinco pastas, de acordo com suas respectivas classes, conforme a Tabela 10.
- Conjunto 6 (CJ6) - Pasta (diretório) com os 50 elementos do conjunto **SA**, distribuídos em três pastas correspondentes aos grupos X, Y e Z. Os elementos de cada grupo, por sua vez, foram divididos em cinco pastas, de acordo com suas respectivas classes, conforme a Tabela 10.

Considerando que, apesar dos ajustes efetuados nos grupos, as classes de Inteligência Artificial, Teoria da Computação e, especialmente, Matemática da Computação ainda eram praticamente inexpressivas para um adequado treinamento do algoritmo, os conjuntos 7, 8, 9, 10, 11 e 12 foram desenvolvidos, tendo como diferencial de seus antecessores a composição das classes. Isto é, os trabalhos pertencentes à Inteligência Artificial, Teoria da Computação e Matemática da Computação foram agrupados na classe “Outros”.

A Tabela 11 mostra as novas composições dos grupos, a partir da fusão das classes:

Tabela 11: Distribuição das monografias com a fusão de classes

Classe	A	B	C	X	Y	Z
Metodologias e Técnicas da Computação	8	8	9	8	8	9
Sistemas de Computação	5	5	5	5	5	5
Outros	2	2	1	4	4	2
Total	15	15	15	17	17	16

Com as novas composições dos grupos, os novos grandes conjuntos (7, 8, 9, 10, 11 e 12), ficaram assim:

- Conjunto 7 (CJ7) - Pasta (diretório) com os 50 elementos do conjunto **RI**, distribuídos em três pastas correspondentes aos grupos X, Y e Z. Os elementos de cada grupo, por sua vez, foram divididos em três pastas, de acordo com suas respectivas classes, conforme a Tabela 11.
- Conjunto 8 (CJ8) - Pasta (diretório) com os 50 elementos do conjunto **S**, distribuídos em três pastas correspondentes aos grupos X, Y e Z. Os elementos de cada grupo, por sua vez, foram divididos em três pastas, de acordo com suas respectivas classes, conforme a Tabela 11.
- Conjunto 9 (CJ9) - Pasta (diretório) com os 50 elementos do conjunto **SA**, distribuídos em três pastas correspondentes aos grupos X, Y e Z. Os elementos de cada grupo, por sua vez, foram divididos em três pastas, de acordo com suas respectivas classes, conforme a Tabela 11.
- Conjunto 10 (CJ10) - Pasta (diretório) com os 45 elementos originais do conjunto **RI**, distribuídos em três pastas correspondentes aos grupos A, B e C. Os elementos de cada grupo, por sua vez, foram divididos em três pastas, de acordo com suas respectivas classes, conforme a Tabela 11.
- Conjunto 11 (CJ11) - Pasta (diretório) com os 45 elementos originais do conjunto **S**, distribuídos em três pastas correspondentes aos grupos A, B e C. Os elementos de cada grupo, por sua vez, foram divididos em três pastas, de acordo com suas respectivas classes, conforme a Tabela 11.
- Conjunto 12 (CJ12) - Pasta (diretório) com os 45 elementos originais do conjunto **SA**, distribuídos em três pastas correspondentes aos grupos A, B e C. Os elementos de cada grupo, por sua vez, foram divididos em três pastas, de acordo com suas respectivas classes, conforme a Tabela 11.

3.4.6. Processos e parâmetros de treino e teste na ferramenta *RapidMiner*

A preparação da ferramenta *RapidMiner* para a realização dos experimentos consiste na escolha dos operadores que devem realizar o pré-processamento do texto, sua seqüência e parâmetros, e na seleção do algoritmo de aprendizagem. Dois modelos devem ser criados: um para treinamento do algoritmo, outro para teste.

O modelo de treino foi construído a partir da seguinte ordem lógica: leitura dos arquivos textuais, segmentação (*tokenização*) das cadeias de caracteres em palavras; conversão dos caracteres para letras minúsculas; aplicação de filtro de *stopwords*; treinamento do algoritmo e gravação do modelo. Os seguintes operadores do *RapidMiner* foram utilizados para realizar estes processos: *TextInput*, *StringTokenizer*, *ToLowerCaseConverter*, *StopwordFilterFile*, W-J48, *ModelWriter*.

Os parâmetros utilizados para o operador *TextInput*, foram:

- texts = “AB”; “BC”; “CA” [*relação das pastas com os grupos de treino*];
- default_content_type = "txt" [*formato/tipo padrão dos arquivos de entrada*];
- default_content_encoding = "UTF-8" [*codificação dos textos*];
- default_content_language = "portuguese" [*informação do idioma do conteúdo*];
- prune_below = “-1” [*default*];
- prune_above = “-1” [*default*];
- vector_creation = “TermFrequency”/“TFIDF” [*utilizadas as duas combinações*];
- use_content_attributes = “true” [*indica para o algoritmo incluir informações sobre os atributos do conteúdo no resultado do pré-processamento*];
- input_word_list = “” [*nome do arquivo que contem a lista de palavras do texto; informado somente para a fase de teste do algoritmo*];
- output_word_list = “nome_do_experimento.lista” [*nome do arquivo que conterá a lista de palavras extraídas do texto*];
- id_attribute_type = “number” [*indica a forma de identificação das fontes*];
- namespaces = “” [*default*];
- create_text_visualizer = “false” [*default*];
- on_the_fly_pruning = “-1” [*default*];

Parâmetros utilizados para o operador *StopwordFilterFile*:

- file = “stoplist.txt” [*nome do arquivo com a stoplist*];
- case_sensitive = “false” [*default*].

Parâmetros do operador W-J48 - todos de acordo com a configuração *default* do operador:

- keep_example_set = “false”;
- U, R, B, S, L, A = “false”;
- C = “0.25”;
- M = “2.0”;
- N, Q = “”.

Operador *ModelWriter*:

- model_file = “nome_do_experimento.mod” [*arquivo onde será salvo o modelo*];
- overwrite_existing_file = “true” [*indica se o arquivo deve ser sobrescrito, se existente*];
- output_type = “XML” [*formato do arquivo de saída*].

A construção do modelo de teste partiu da seguinte ordem lógica: leitura dos arquivos textuais, segmentação (*tokenização*) das cadeias de caracteres em palavras; conversão dos caracteres para letras minúsculas; aplicação de filtro de *stopwords*; recuperação do modelo treinado; aplicação do modelo. Os seguintes operadores do *RapidMiner* foram utilizados para realizar estes processos: *TextInput*, *StringTokenizer*, *ToLowerCaseConverter*, *StopwordFilterFile*, *ModelLoader*, *ModelApplier*.

Os parâmetros selecionados para o operador *TextInput* foram idênticos aos relacionados no modelo de treino, exceto pelos parâmetros "*texts*", que recebe grupo distinto, que não tenha participado do treinamento, e "*input_word_list*", que recebe o arquivo gerado no treinamento (*output_word_list*) como entrada. Maiores detalhes sobre os parâmetros dos operadores da ferramenta podem ser encontrados na documentação que acompanha o *RapidMiner* e o *TextPlugin*.

Com o objetivo de padronizar os experimentos, todos os conjuntos (CJ1, CJ2, CJ3, CJ4, CJ5, CJ6, CJ7, CJ8, CJ9, CJ10, CJ11 e CJ12) foram submetidos aos modelos retro descritos.

4. RESULTADOS DOS EXPERIMENTOS

A etapa de experimentação foi realizada com o objetivo de encontrar o melhor conjunto de procedimentos para realizar a classificação semi-automática de monografias proposta neste trabalho.

No total foram realizadas 8 (oito) séries de experimentos, com 9 (nove) simulações cada. Os detalhes dos experimentos são descritos nos subcapítulos a seguir.

4.1. Série de experimentos n.° 1

Compreende simulações realizadas com os três conjuntos - CJ1, CJ2 e CJ3 - com validação cruzada dos grupos - AB x C, BC x A e CA x B. A principal característica desta série está no método de criação dos vetores: *TermFrequency*. Os resultados obtidos constam na Tabela 12.

Tabela 12: Resultados da série de experimentos n.° 1

Conjunto:	RI			S			SA		
Grupos treino:	AB	BC	CA	AB	BC	CA	AB	BC	CA
Grupo teste:	C	A	B	C	A	B	C	A	B
Exemplos CC:	8	8	4	11	12	12	8	4	5
Exemplos IC:	7	7	11	4	3	3	7	11	10
Total ex. treino:	30	30	30	30	30	30	30	30	30
Total ex. teste:	15	15	15	15	15	15	15	15	15
Total exemplos:	45	45	45	45	45	45	45	45	45
% CC:	53	53	27	73	80	80	53	27	33
% IC:	47	47	73	27	20	20	47	73	67
Média %CC:	44,44			77,78			37,78		

CC = Corretamente Classificado. IC = Incorretamente Classificado.

4.2. Série de experimentos n.° 2

Assim como ocorreu na série anterior, compreende simulações realizadas com os três conjuntos - CJ1, CJ2 e CJ3 - com validação cruzada dos grupos - AB x C, BC x A e CA x B. Diferentemente das simulações anteriores, nesta série o método de criação dos vetores utilizado foi o TFIDF. Os resultados obtidos constam na Tabela 13.

Tabela 13: Resultados da série de experimentos n.° 2

Conjunto:	RI			S			SA		
Grupos treino:	AB	BC	CA	AB	BC	CA	AB	BC	CA
Grupo teste:	C	A	B	C	A	B	C	A	B
Exemplos CC:	9	4	7	11	4	12	8	4	5
Exemplos IC:	6	11	8	4	11	3	7	11	10
Total ex. treino:	30	30	30	30	30	30	30	30	30
Total ex. teste:	15	15	15	15	15	15	15	15	15
Total exemplos:	45	45	45	45	45	45	45	45	45
% CC:	60	27	47	73	27	80	53	27	33
% IC:	40	73	53	27	73	20	47	73	67
Média %CC:	44,44			60,00			37,78		

CC = Corretamente Classificado. IC = Incorretamente Classificado.

4.3. Série de experimentos n.° 3

Esta série foi realizada com os conjuntos CJ4, CJ5 e CJ6, descritos na seção 8.4.1, com validação cruzada nos grupos XY x Z, YZ x X e ZX x Y. O método de criação dos vetores foi o *TermFrequency*. Os resultados obtidos constam na Tabela 14.

Tabela 14: Resultados da série de experimentos n.° 3

Conjunto:	RI			S			SA		
Grupos treino:	XY	YZ	ZX	XY	YZ	ZX	XY	YZ	ZX
Grupo teste:	Z	X	Y	Z	X	Y	Z	X	Y
Exemplos CC:	10	7	4	13	12	11	9	3	6
Exemplos IC:	6	10	13	3	5	6	7	14	11
Total ex. treino:	34	33	33	34	33	33	34	33	33
Total ex. teste:	16	17	17	16	17	17	16	17	17
Total exemplos:	50	50	50	50	50	50	50	50	50
% CC:	63	41	24	81	71	65	56	18	35
% IC:	37	59	76	19	29	35	44	82	65
Média %CC:	42,40			72,18			36,40		

CC = Corretamente Classificado. IC = Incorretamente Classificado.

4.4. Série de experimentos n.° 4

Esta série de simulações foi realizada com os mesmos conjuntos da série n.° 3, porém com método de criação dos vetores TFIDF. Os resultados obtidos constam na Tabela 15.

Tabela 15: Resultados da série de experimentos n.° 4

Conjunto:	RI			S			SA		
Grupos treino:	XY	YZ	ZX	XY	YZ	ZX	XY	YZ	ZX
Grupo teste:	Z	X	Y	Z	X	Y	Z	X	Y
Exemplos CC:	11	7	7	12	9	11	8	1	6
Exemplos IC:	5	10	10	4	8	6	8	16	11
Total ex. treino:	34	33	33	34	33	33	34	33	33
Total ex. teste:	16	17	17	16	17	17	16	17	17
Total exemplos:	50	50	50	50	50	50	50	50	50
% CC:	69	41	41	75	53	65	50	6	35
% IC:	31	59	59	25	47	35	50	94	65
Média %CC:	50,37			64,22			30,39		

CC = Corretamente Classificado. IC = Incorretamente Classificado.

4.5. Série de experimentos n.° 5

Série de simulações realizada com os conjuntos CJ7, CJ8 e CJ9, descritos na seção 8.4.1 com validação cruzada nos grupos XY x Z, YZ x X e ZX x Y. Método de criação dos vetores: *TermFrequency*. Os resultados obtidos constam na Tabela 16.

Tabela 16: Resultados da série de experimentos n.° 5

Conjunto:	RI			S			SA		
Grupos treino:	XY	YZ	ZX	XY	YZ	ZX	XY	YZ	ZX
Grupo teste:	Z	X	Y	Z	X	Y	Z	X	Y
Exemplos CC:	10	7	5	11	12	11	6	4	3
Exemplos IC:	6	10	12	5	5	6	10	13	14
Total ex. treino:	34	33	33	34	33	33	34	33	33
Total ex. teste:	16	17	17	16	17	17	16	17	17
Total exemplos:	50	50	50	50	50	50	50	50	50
% CC:	62	41	29	69	71	65	38	24	18
% IC:	38	59	71	31	29	35	62	76	82
Média %CC:	44,36			68,01			26,23		

CC = Corretamente Classificado. IC = Incorretamente Classificado.

4.6. Série de experimentos n.° 6

Esta série de simulações foi realizada com os mesmos conjuntos da série n.° 5, com método de criação dos vetores TFIDF. Os resultados obtidos constam na Tabela 17.

Tabela 17: Resultados da série de experimentos n.° 6

Conjunto:	RI			S			SA		
Grupos treino:	XY	YZ	ZX	XY	YZ	ZX	XY	YZ	ZX
Grupo teste:	Z	X	Y	Z	X	Y	Z	X	Y
Exemplos CC:	10	7	8	11	12	11	8	2	3
Exemplos IC:	6	10	9	5	5	6	8	15	14
Total ex. treino:	34	33	33	34	33	33	34	33	33
Total ex. teste:	16	17	17	16	17	17	16	17	17
Total exemplos:	50	50	50	50	50	50	50	50	50
% CC:	62	41	47	69	71	65	50	12	18
% IC:	38	59	53	31	29	35	50	88	82
Média %CC:	50,25			68,01			26,47		

CC = Corretamente Classificado. IC = Incorretamente Classificado.

4.7. Série de experimentos n.° 7

Série de simulações realizada com os conjuntos CJ10, CJ11 e CJ12, descritos na seção 8.4.1 com validação cruzada nos grupos AB x C, BC x A e CA x B. Método de criação dos vetores: *TermFrequency*. Os resultados obtidos constam na Tabela 18.

Tabela 18: Resultados da série de experimentos n.° 7

Conjunto:	RI			S			SA		
Grupos treino:	AB	BC	CA	AB	BC	CA	AB	BC	CA
Grupo teste:	C	A	B	C	A	B	C	A	B
Exemplos CC:	11	7	5	13	12	6	7	4	6
Exemplos IC:	4	8	10	2	3	9	8	11	9
Total ex. treino:	30	30	30	30	30	30	30	30	30
Total ex. teste:	15	15	15	15	15	15	15	15	15
Total exemplos:	45	45	45	45	45	45	45	45	45
% CC:	73	47	33	87	80	40	47	27	40
% IC:	27	53	67	13	20	60	53	73	60
Média %CC:	51,11			68,89			37,78		

CC = Corretamente Classificado. IC = Incorretamente Classificado.

4.8. Série de experimentos n.° 8

Esta série de simulações foi realizada com os mesmos conjuntos da série n.° 7, mas com método de criação dos vetores TFIDF. Os resultados obtidos constam na Tabela 19.

Tabela 19: Resultados da série de experimentos n.° 8

Conjunto:	RI			S			SA		
Grupos treino:	AB	BC	CA	AB	BC	CA	AB	BC	CA
Grupo teste:	C	A	B	C	A	B	C	A	B
Exemplos CC:	12	7	8	12	12	11	7	4	6
Exemplos IC:	3	8	7	3	3	4	8	11	9
Total ex. treino:	30	30	30	30	30	30	30	30	30
Total ex. teste:	15	15	15	15	15	15	15	15	15
Total exemplos:	45	45	45	45	45	45	45	45	45
% CC:	80	47	53	80	80	73	47	27	40
% IC:	20	53	47	20	20	27	53	73	60
Média %CC:	60,00			77,78			37,78		

CC = Corretamente Classificado. IC = Incorretamente Classificado.

4.9. Análise dos resultados

O Gráfico 4 compila todos os resultados das séries de experimentos.

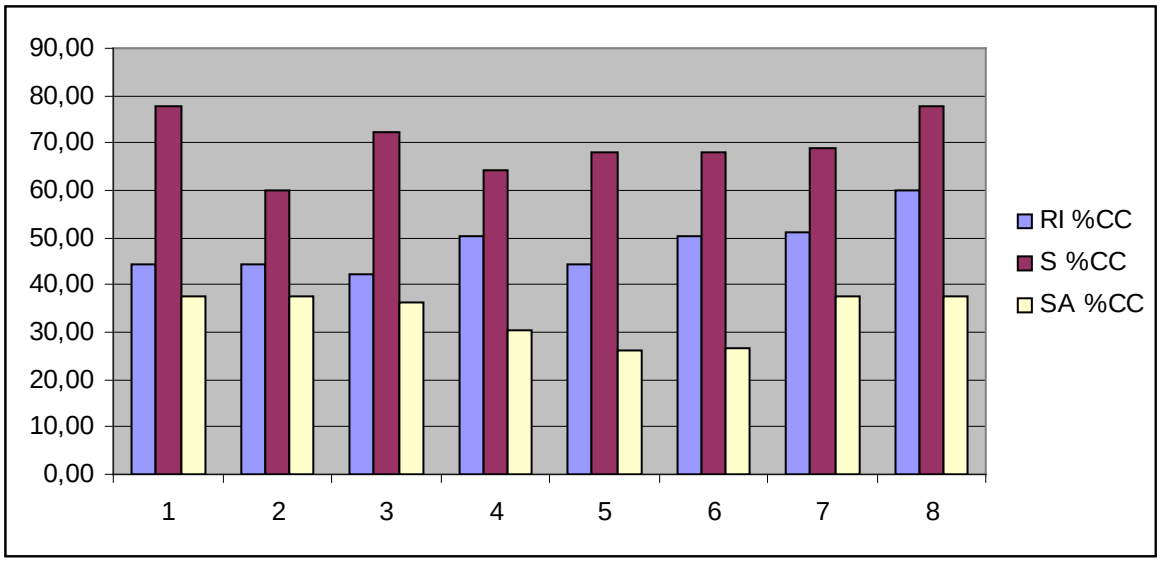

Gráfico 4: Resultados das séries de experimentos

A análise do gráfico permite observar que o melhor desempenho de classificação, em todos os experimentos, é o do conjunto S, composto dos dez substantivos mais freqüentes. Os melhores resultados com este conjunto ocorreram nos experimentos 1 e 8, quando o algoritmo atingiu 78% de confiabilidade. Este conjunto apresentou uma tendência inicial de melhores resultados com o método da Freqüência Relativa (*TermFrequency*) de criação de vetores, como é possível observar nos primeiros quatro experimentos. Esta tendência, no entanto, não se repetiu nos experimentos seguintes.

O conjunto RI apresentou resultados diversos, a maioria com confiabilidade igual ou inferior a 51%, salvo no experimento 8, quando atingiu 60%. Este conjunto demonstrou melhores resultados com o método TF-IDF de criação de vetores.

O conjunto SA, por sua vez, apresentou os resultados mais fracos. Em nenhum experimento foi possível obter uma confiabilidade média superior a 38%. Os piores resultados apareceram nos experimentos 5 e 6, quando ficaram abaixo de 30%. A piora pode ter relação com a presença de mais elementos para classificar, uma vez que os resultados dos experimentos 3, 4, 5 e 6, todos com 50 elementos, se mostraram piores que os demais.

CONCLUSÕES

Este trabalho apresentou o desenvolvimento de uma metodologia para classificação semi-automática de monografias do curso de Ciências da Computação, utilizando técnicas de aprendizagem supervisionada.

As experiências realizadas demonstraram que os scripts desenvolvidos por Lourenço (2007), combinados com o uso de informações lingüísticas sugerido por Silva (2004), podem ser utilizados para melhorar a confiabilidade da classificação com o algoritmo J48, proporcionando resultados até 30% melhores que o uso de segmentos dos textos (resumos, palavras-chaves e introduções) na ferramenta *RapidMiner*.

É importante ressaltar que o algoritmo treinado neste trabalho possui boa acurácia para as classes de Metodologias e Técnicas da Computação e Sistemas de Computação, mais representativas no conjunto de monografias utilizado. Isso pode indicar que esses resultados poderiam ser ainda melhores se as demais classes da taxonomia - Inteligência Artificial, Matemática da Computação e Teoria da Computação - fossem melhor representadas por monografias. Deste modo, sugere-se realizar novo treinamento, com conjuntos de trabalhos mais homogêneos, para obtenção de resultados que representem melhor as demais classes da taxonomia.

Para trabalhos futuros, sugere-se ampliar as discussões sobre a taxonomia proposta para as Ciências da Computação, realizar experimentos com conjuntos de monografias mais homogêneos e desenvolver ferramenta capaz de automatizar as fases da metodologia de classificação proposta.

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.

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.

GASPERIN, Caroline, et al. **Extracting XML syntactic chunks from Portuguese corpora**. TALN, 2003. França, 2003, 10p.

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.

LOURENÇO, Adão Luís. **Construção semi-automática de tesauros**. FEEVALE, 2007. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação). Ciências Exatas e Tecnológicas, Centro Universitário Feevale, 2007. 77 p.

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.

RAPIDMINER. **RapidMiner 4.0 User Guide; Operator Reference; Develper Tutorial; & The Word Vector Tool and the RapidMiner Text Plugin User Guide; Operator Reference; Develper Tutorial**. Rapid-I, 2007. Disponível em <http://www.rapid-i.com>. Acesso em 02/06/2008.

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.

SILVA, Cassiana Fagundes da Silva. **Uso de informações lingüísticas na etapa de pré-processamento em mineração de textos**. UNISINOS, 2004. Dissertação de Mestrado (pós-graduação em computação aplicada). Ciências Exatas e Tecnológicas, Universidade do Vale do Rio dos Sinos, 2004. 96 p.

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.

ANEXOS

ANEXO A

Período	Autor	Monografia
2007/01	Achylles M. da Silva Filho	Portal do Conhecimento: um ambiente tecnológico com a finalidade de facilitar a Gestão do Conhecimento
2007/02	Adriana Heck	Desenvolvimento Web para Ritmo Veículos
2007/02	Alessandro C. dos Reis	Sistemas de Conhecimento: Estudo de Caso para Modelagem de Extensão Universitária
2007/01	André Wagner	Criação de um Kit de Desenvolvimento de Software para Programação Modular de Emuladores
2007/01	Andréa B. Souza	Estudo de Técnicas para minimizar os Ataques de DoS (Denial of Service)
2007/02	Andrei P. Heckel	Identificação por Rádiofreqüência (RFID): Estudo Teórico e Experimentação via Simulação
2007/02	Ariégi de O. Vasques	Desenvolvimento de um Sistema Web de Controle de Documentos baseado nas Normas de Qualidade ISO 9001:2000
2007/02	Caetano Y. Vieira	UC-Measurer: Sistema para Cálculo de Prazos e Custos na Implementação de Softwares baseado na apuração de pontos por Caso de Uso
2007/01	Carlos A. Weissheimer Jr.	Honeynet - Estudo Teórico e Prático
2007/02	Charles Ricardo Staub	Ferramenta para Calibração da Intensidade Luminosa de LEDs em Painéis Eletrônicos de Mídia Externa
2007/02	Daniel Michels	Desenvolvimento Web Dinâmico de um Portal de Ensino de Matemática
2007/02	Daniel R. Dummer	Engenharia Reversa aplicada à Exploração e Proteção de Software
2007/02	Daniela S. Martins	Proposta de Desenvolvimento de um Data Mart na Área de Gestão de Pessoas e Aplicação de uma Ferramenta OLAP
2007/01	Diego C. Kern	Inferência sobre Ontologias
2007/02	Diogo A. Pereira	InfraGuiBuilder - Framework para Geração Dinâmica e Customização de Interfaces Gráficas de Usuário
2007/02	Fabiano C. da Fonseca	Cobertura de Terrenos baseada em Inteligência de Enxames
2007/01	Fabio Lagemann	Análise de Imagens Microscópicas de Revestimentos Compósitos

2007/02	Fabio L. Sarmento	Qualidade em Ferramentas de Teste
2007/02	Felipe Lucchese	Utilizando Wardriving para a Detecção de Vulnerabilidades em Redes Locais Sem Fio na Região de Farroupilha
2007/02	Fernando Mertins	Ontologia de Domínio na Análise de Blogs
2007/02	Guilherme G. Konzen	Ferramentas de Código Aberto para o Combate ao SPAM
2007/01	Gustavo Dagnese	Construindo um Portal de Tecnologia
2007/02	Igor H. Berlitz	Gerador Gráfico de Relatórios utilizando a classe FPDF
2007/01	Jeferson T. Utzig	Rede de Sensores sem Fio - Um Estudo Teórico com Experimentação via Simulação
2007/01	João B. Mossmann	Estudo de Técnicas de Processamento de Imagens aplicadas ao Apoio do Diagnóstico de Derrames Serosos de Etiologia Benigna e Maligna
2007/02	Jônatas Caberlon	Uma Solução, baseada em Ferramentas Oracle, para o Desenvolvimento de um Sistema de Business Intelligence
2007/01	Leandro Fussiger	NLPSearch - Framework para integração de Sistemas de PLN e API's de Mecanismos de Busca
2007/02	Leônidas K. Saldanha	Firewalls de Baixo Custo
2007/01	Lucas Graebin	Estudo comparativo de Sistemas de Arquivos Distribuídos
2007/02	Luiz F. S. Pauperio	Proposta de um Modelo para a Gestão de Risco
2007/02	Luiz S. Andrés	Metodologia para Geração Automática de Textos com Concordância Verbal e Nominal
2007/02	Maikon W. Monzon	Aplicação de Redes Neurais em Sistema de Detecção de Intrusão
2007/02	Maiquel Steinmetz	Modelagem de um Sistema de Inteligência Competitiva para a busca de Perfil de Clientes
2007/02	Marcio M. Gomes	Compilador Aplicado à Extração de Dados de Análises Ambientais
2007/01	Marcos Caletti	IPS (Intrusion Prevention System) - Um Estudo Teórico e Experimental
2007/01	Mariana Kreisig	Proposta de ambiente de cooperação para desenvolvimento de software integrado ao framework jFace
2007/02	Mauro R. Brugnera	Sistema de Gerenciamento Domótico
2007/02	Rafael M. Ruidias	A Viabilidade de Negócios de TI alinhada ao Planejamento Estratégico das Empresas
2007/02	Sergio Strieder	Implementação de Central Telefônica VoIP
2007/02	Sidinei P. Gonchoroski	Utilização de Técnicas de KDD em um Call Center Ativo
2007/02	Thiago C. Feldmann	Projeto de Banco de Dados em Grid visando Alta Disponibilidade
2007/02	Tiago B. Santos	Simulação Multiagentes Aplicada ao Mercado de Capitais
2007/02	Tiago M. Uriartt	Simulador para Empacotamento 3D
2007/01	Toni B. Marques	Problema de Roteamento de Veículos: Estudo de Caso aplicado à Distribuição de Pacotes de Jornais

		em Empresa Jornalística
2007/02	Vinícius V. Vetromilla	Sistema de Controle de Acesso utilizando RFID