INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA

Classificação Multi-Etiqueta Hierárquica de Textos Segundo a Taxonomia ACM

António Paulo Gomes dos Santos
Novembro de 2008

POLITÉCNICO DO PORTO

Classificação multi-etiqueta hierárquica de textos segundo a taxonomia ACM

2007 / 2008

1000313 António Paulo Gomes dos Santos

Classificação multi-etiqueta hierárquica de textos segundo a taxonomia ACM

2007 / 2008

1000313 António Paulo Gomes dos Santos

Mestrado em Engenharia Informática

Ramo: Tecnologias do Conhecimento e Decisão

Novembro 2008

Orientador ISEP: **Maria de Fátima Coutinho Rodrigues**

Este projecto é dedicado à minha mãe...

Agradecimentos

Gostaria de agradecer a todas as pessoas que de alguma forma contribuíram para a realização deste projecto. Agradeço especialmente:

À minha orientadora de projecto, professora Doutora Maria de Fátima Coutinho Rodrigues, por toda a ajuda prestada, pela disponibilidade que sempre teve e sobretudo pelo enorme trabalho de revisão deste documento;

Ao professor Doutor Carlos Ramos, que na qualidade de director do grupo de investigação GECAD (Grupo de Engenharia de Conhecimento e Apoio à Decisão), por todas as condições para a realização deste trabalho;

Agradeço à FCT (Fundação para a ciência e Tecnologia) pelo apoio expresso através do financiamento da bolsa de investigação no âmbito do projecto COPSRO (*Computational Approach to Ontology Profiling of Scientific Research Organisations*) com a referência PTDC/EIA/69988/2006.

A todos os colegas do Departamento de Informática do ISEP/IPP e a todos os investigadores envolvidos no GECAD agradeço a disponibilidade e contribuições com críticas e sugestões.

A todos, muito obrigado!

Resumo

Muitos dos trabalhos de classificação existentes na literatura, envolvem a atribuição a cada instância (exemplo) de uma única classe, de entre um conjunto pré-definido de classes normalmente pequeno e organizado de forma plana. Porém, existem problemas de classificação mais complexos, em que a cada instância é possível atribuir mais do que uma classe, podendo as classes, estar organizadas numa estrutura hierárquica. Para estes problemas, existe um conjunto de abordagens para lidar com o facto de uma instância poder pertencer a mais do que uma classe (classificação multi-etiqueta). Existem também abordagens para lidar com a organização hierárquica das classes (classificação hierárquica).

Esta dissertação, apresenta um estudo das abordagens e conceitos de classificação multi-etiqueta e hierárquica, aplicados à classificação de documentos de texto. Trata-se, portanto, de um problema de classificação, em que as instâncias são documentos de texto, que podem pertencer a mais do que uma classe e estas encontram-se organizadas hierarquicamente.

Nos problemas de classificação de texto, uma fase importante, é o pré-processamento dos documentos. Um processo transformativo, aplicado normalmente para reduzir o número de termos de um documento, de forma a obter uma representação dos documentos, mais adequada para as fases seguintes. Nesta dissertação, são também estudadas as várias tarefas de pré-processamento que podem ser realizadas, como por exemplo, remoção de *stopwords*, *stemming*, esquemas de atribuição de pesos aos termos.

No estudo experimental realizado, foi utilizado o esquema de classificação ACM (*Computing Classification System*), que define um conjunto de classes, organizadas hierarquicamente, nas áreas científicas no campo da computação.

O estudo experimental realizado, consistiu no desenvolvimento de uma solução para automatizar a navegação e recolha de documentos classificados da biblioteca digital ACM, pré-processamento dos documentos, construção e aplicação de diferentes classificadores a documentos ainda não classificados e por fim a avaliação do seu desempenho de previsão. Foi proposta uma metodologia para classificação multi-etiqueta hierárquica que combina as abordagens usadas na classificação multi-etiqueta e na classificação hierárquica que se mostrou adequada para a resolução destes problemas.

Palavras Chave (Tema): Classificação de texto Multi-Etiqueta Hierárquico; Medidas de Avaliação; Esquema de classificação ACM;

Abstract

On many works in text classification literature, each instance (example) is assigned to one class, from a predefined class dataset usually little and flat. However, there are more complex classification problems in which an instance can belong to any number of classes and they can be organized hierarchical. For this kind of problems, there are a set of approaches.

This MSc thesis is focused on a study about approaches and concepts of hierarchical multi-label text classification.

Preprocessing is a very important stage on text classification. This transformation process is usually applied to reduce the number of terms present in documents, to generate document models that are appropriate for the following stages. In this report, several preprocessing tasks, like stopwords, stemming and term weighting are studied.

The experimental study is made using ACM (Association for Computing Machinery) Computing Classification System, where classes have a hierarchical structure on computer science field.

In this experimental study, a solution was developed to automate the navigation and collection of documents from ACM digital library. This thesis also covers: preprocessing of documents, building a document's classifier and its performance evaluation. A methodology for classification hierarchic multi-label that combines the used boardings in the classification multi-label and in the hierarchic classification was proposal that if it showed adequate for the resolution of these problems.

Keywords (by subject): Hierarchical multi-label text classification; Evaluation Measures; ACM classification scheme;

Índice

Índice de Figuras

Índice de Tabelas

Índice de Algoritmos

Notação

De seguida, são apresentados alguns dos símbolos mais utilizados nesta dissertação. Os símbolos utilizados mas não expostos, nesta lista, são apresentados junto ao seu contexto de utilização.

Símbolo	**Significado**
C	Conjunto de categorias $\{c_1, c_2, \dots, c_{\lvert C\rvert}\}$
$\lvert\mathrm{C}\rvert$	Cardinalidade (número de membros) do conjunto C
c_i	Uma classe ou categoria
D	Conjunto de documentos $\{d_1, d_2, \dots, d_{\lvert D\rvert}\}$
$\lvert\mathrm{D}\rvert$	Cardinalidade (número de membros) do conjunto D
d_i	Um documento (sempre que possível é usado o sub-índice i)
$DF(t_k)$	Frequência de documento (número de documentos em que o termo t_k ocorre)
$GI(D, t_k)$	Ganho de informação do conjunto Ddepois de escolhido o termo t_k
ε	Factor de redução
$H(C)$	Entropia do conjunto de categorias C
$H(C\lvert t_k)$	Entropia condicional. Entropia do conjunto C, sabendo que o termo $t_k$ocorreu
$MI(t_k; c_i)$	Informação mútua do termo t_k e categoria c_i
$\mathrm{P}(.)$	Probabilidade de um acontecimento (aqui representado por ".")
t_k	Um termo
T	Conjunto de termos $\{t_1, t_2, \dots, t_{\lvert T\rvert}\}$
T'	Subconjunto do conjunto T
$\lvert T\rvert$	Cardinalidade (número de membros) do conjunto T
$\lvert T'\rvert$	Cardinalidade (número de membros) do conjunto T'
$TF(t_k)$	Frequência de termo (número de vezes que o termo t_k ocorre)
$x^2(t_k, c_i)$	Medida *chi-square* do termo t_k e categoria c_i

Formatos

Código fonte, outros exemplos	Courier New, 10 ou 11 pt, envolvido numa moldura, linhas enumeradas.
Corpo do texto	Calibri, 12 pt.
Estrangeirismo	Times New Roman, 12 pt, Itálico.
Legendas (das figuras, código, tabelas)	Calibri, 11 pt, Itálico.
Palavras ou frases a destacar	São escritas com ênfase como por exemplo em Maiúsculas, Negrito, Itálico.
Tabelas cabeçalho	Cambria, 11 pt, cor clara sobre fundo escuro ou cor escura sobre fundo claro.
Tabelas corpo	Calibri, 11 pt

Capítulo 1. Introdução

Nesta dissertação é abordada a **classificação multi-etiqueta hierárquica de textos**, que consiste na **classificação de textos**, onde um documento pode pertencer a qualquer número de classes e estas estão organizadas hierarquicamente.

Na secção *1.1*, começa-se por discutir a **classificação de textos** em geral, para na secção *1.2* se introduzir o problema subjacente a esta dissertação, seguido de uma proposta de resolução na secção *1.3*. Na secção *1.4*, são apresentadas as principais ferramentas utilizadas.

Por fim, na secção *1.5*, são resumidas as contribuições da investigação realizada. Na secção *1.6* é apresentada a organização geral desta dissertação.

1.1 Classificação de textos

A **classificação de textos** (*text classification*) consiste na atribuição de uma ou mais categorias previamente existentes a documentos de texto (*documentos*, *parágrafos*, *frases*) com base no seu conteúdo. As categorias são também chamadas classes (*classes*), etiquetas (*labels*) ou tópicos (*topics*). Nesta dissertação será utilizada, preferencialmente categorias ou classes.

Para além do termo **classificação de textos** (*text classification*) é comum na literatura encontrar-se termos como **categorização de textos** (*text categorization*), **classificação de documentos** (*document classification*), **categorização de documentos** (*document categorization*), **descobrimento de tópicos** (*topic spotting*) [F. Sebastiani 2002, 1].

Mais formalmente, dado um conjunto de categorias $C = \{C_1, C_2, \ldots, C_{|C|}\}$ e um conjunto de documentos classificados $D = \{d_1, d_2, \ldots, d_{|D|}\}$, utilizando um método ou algoritmo de aprendizagem, pretende-se aprender um **classificador** (*classifier*) ou **função de classificação** (*classification function*) F, que mapeia documentos em categorias. O classificador ou função de classificação é depois utilizada para classificar novos documentos, ainda não classificados.

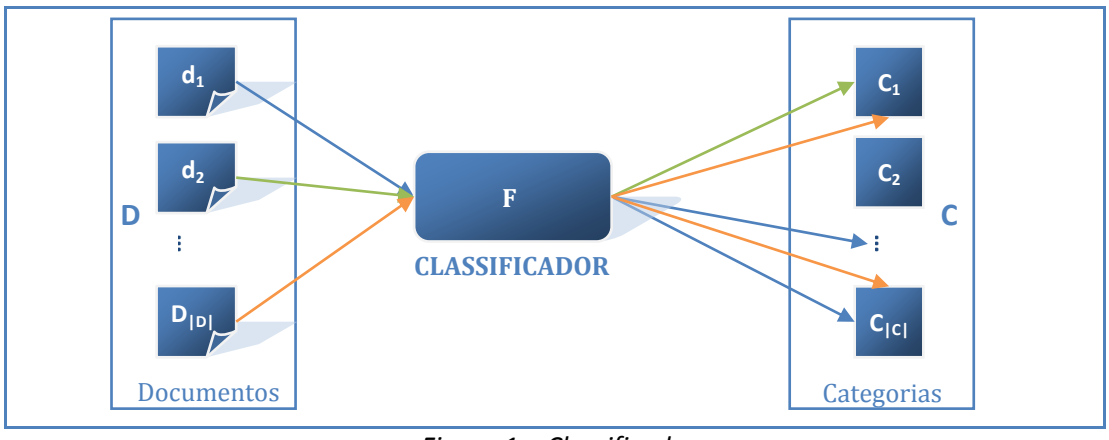


Figura 1 – Classificador

Na actualidade, em que grande parte da informação é produzida e armazenada digitalmente, a classificação de textos é claramente necessária. O correio electrónico, os artigos científicos, os procedimentos de conferências, bibliotecas digitais e repositórios *Web*, são apenas alguns exemplos de colecções electrónicas de textos, que podem ou já beneficiam da classificação de textos.

1.2 O problema

A **classificação multi-etiqueta hierárquica de textos** (*hierarchical multi-label text classification*), consiste na tarefa de atribuir qualquer número de classes a documentos de texto, estando as classes organizadas numa estrutura hierárquica. A classificação de textos é um caso de **aprendizagem supervisionada** (*supervised learning*), onde o conjunto de classes e exemplos de documentos pertencentes a essas classes, são dadas. Esta dissertação não irá abordar problemas de **aprendizagem não supervisionada** (*unsupervised learning*), chamados agrupamento de texto (*text clustering*), onde as classes não são conhecidas antecipadamente.

Como conjunto de classes pré-definidas, foram usadas as definidas no esquema ou **Sistema de Classificação ACM versão 1998**[1] (*ACM Computing Classification System version 1998*). As classes definidas neste esquema, dizem respeito a áreas científicas no campo da computação e encontram-se organizadas hierarquicamente.

Como conjunto de documentos já classificados, foram utilizados os recolhidos de forma automatizada, pela solução desenvolvida nesta dissertação e apresentada no Capítulo *7* (página *79*).

Tipicamente, um documento é relevante para vários tópicos, consequentemente pode pertencer a várias classes. Tipicamente, a organização hierárquica das classes facilita a sua organização. Estes factos, evidenciam a importância da classificação multi-etiqueta hierárquica.

1.3 A abordagem

Os primeiros sistemas de classificação de textos seguiam principalmente uma abordagem **engenharia do conhecimento** (*knowledge engineering*). Dado um conjunto de categorias pré-definidas, eram definidas manualmente por especialistas humanos regras para cada categoria. O sistema CONSTRUE [Hayes e Weinstein 1990] é um desses exemplos.

Na actualidade, a abordagem típica, dos sistemas de classificação envolvem o uso de técnicas de **Aprendizagem Automática** (*Machine Learning*) para criar classificadores em vez da especificação manual dos critérios para a atribuição de determinada categoria [F. Sebastiani 2002, 2]. O processo de Aprendizagem Automática tipicamente analisa um conjunto de documentos previamente classificados, e tenta induzir conhecimento que permita categorizar novos documentos.

[1] Válida (actualizada) em 2007

Na abordagem seguida, o projecto foi dividido em três grandes fases:

1. Recolha e extracção de informação de artigos classificados da biblioteca digital ACM;
2. Classificação multi-etiqueta hierárquica de textos;
3. Avaliação do desempenho da abordagem de classificação seguida.

1.3.1 Fase 1 – Recolha e extracção de informação da biblioteca digital ACM

A classificação de textos baseada no paradigma de aprendizagem automática (*machine learning*) (que é a utilizada nesta dissertação), assume a existência de uma colecção de documentos previamente classificados. Na literatura é comum encontrarem-se, disponíveis para *download*, colecções compiladas de documentos classificados.

Acontece que nenhuma das colecções actualmente disponíveis, é muito adequada para a classificação multi-etiqueta hierárquica de textos. Como tal, o objectivo desta primeira fase consistiu em criar uma colecção de documentos já classificados segundo o sistema ACM. Na secção *2.4* (página *9*), são analisadas as colecções mais utilizadas na classificação de textos.

Para a criação da colecção de documentos, foi desenhada e implementada uma solução, para de forma automática, recolher documentos e extrair os dados necessários. Na figura seguinte, encontra-se uma representação do processo de recolha e extracção dos dados.

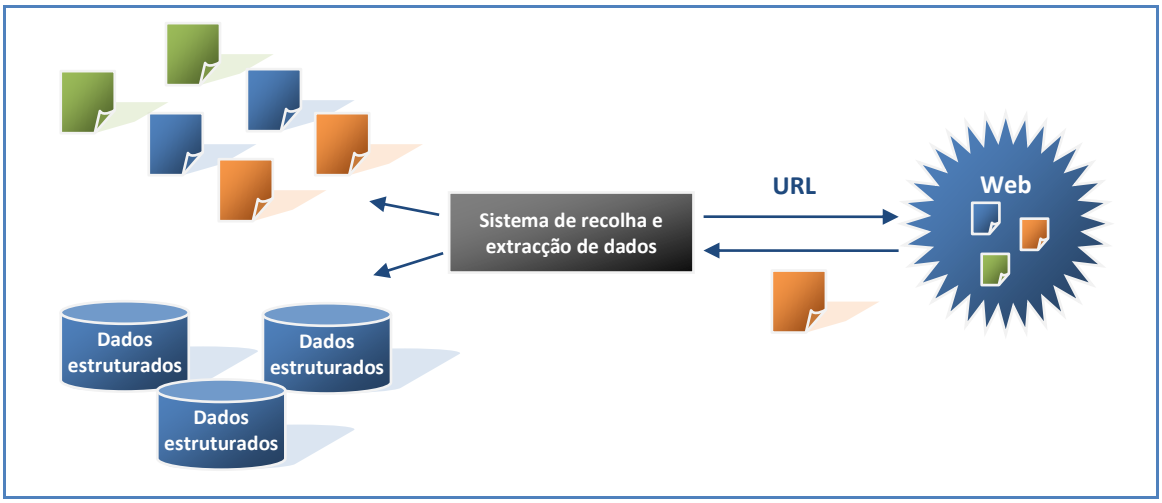


Figura 2 – Fase 1: Recolha de documento classificados

Partindo de um primeiro URL fornecido pelo utilizador, o sistema recolhe o recurso correspondente (página *Web*). Extrai dessa página, dados específicos considerados relevantes: como ligações específicas (*links*) usadas para alcançar outras páginas, dados importantes para a criação da colecção de documentos classificados (titulo documento, resumo, palavras-chave, classificação ACM, etc).

Como resultado desta fase, foi obtido um conjunto de documentos classificados e um conjunto de informações estruturadas que serviram de entrada (*input*) para a segunda fase.

1.3.2 Fase 2 - Classificação de novos textos

O objectivo desta fase, consistiu em partir de um conjunto de documentos classificados, apresentar e implementar uma abordagem para classificar novos documentos.

Com base na colecção de documentos criada na fase anterior, realizaram-se várias tarefas de pré-processamento e foram construídos vários classificadores, que partindo dos documentos pré-processados, induziram o conhecimento necessário para classificar novos documentos. Na figura seguinte, encontra-se uma representação genérica do processo de classificação e avaliação de textos.

Figura 3 – Etapas genéricas do processo de classificação de textos e avaliação

Das etapas apresentadas na figura acima, as realizadas na fase 2, foram: o pré-processamento, aprendizagem do classificador e classificação.

Como resultado desta fase, foram construídos vários classificadores.

1.3.3 Fase 3 - Avaliação dos resultados

A terceira fase surgiu da necessidade de avaliar os resultados da abordagem seguida neste projecto.

Como resultado desta fase, foram obtidas várias medidas de desempenho dos classificadores implementados na fase anterior.

1.4 Tecnologias utilizadas

Durante o projecto foram utilizadas diversas tecnologias para o desenvolvimento do mesmo. Foram utilizadas linguagens de programação, ferramentas e APIs especificas para realizar o trabalho pretendido. Destacam-se as seguintes:

1.4.1 Linguagens de programação ou representação

- JAVA Platform, Standard Edition 6 (JDK Version 1.6.0)
- XML – Extensible Markup Language

1.4.2 APIs

- SAX2[2] – Simple API for XML Version 2 (Desde o JDK 1.4 que vem incluído no mesmo)
- Jerico 2.5[3] – Parser HTML
- Weka 3.5[4] – Waikato Environment for Knowledge Analysis
- Wvtool 1.1[5] – Word Vector Tool
- Mulan 0.2[6] – Multi-label classification

1.4.3 Ferramentas de desenvolvimento

- NetBeans IDE 5.5

[2] http://www.saxproject.org
[3] http://sourceforge.net/projects/jerichohtml/
[4] http://www.cs.waikato.ac.nz/ml/weka
[5] http://wvtool.sf.net/
[6] http://mlkd.csd.auth.gr/multilabel.html

- NetBeans IDE 6.0 (NetBeans Profiler)
- NotePad++ 4.6

1.4.4 Outras ferramentas

- MySQL 5.0.45
- MySQL Query Browser 1.2.12
- MySQL Connector/J 5.1

1.5 Contribuições desta dissertação

- Desenho e implementação de uma solução para recolha de documentos e extracção de dados, que pode ser facilmente adaptada para efectuar a recolha e extracção direccionada noutros domínios (Capítulo *7*, página *79*);
- Criação de uma colecção de documentos de texto para problemas multi-etiqueta hierárquico e disponibilização à comunidade de investigação[7] (Capítulo *7*, página *79*);
- Estudo compreensivo de tarefas de pré-processamento de documentos de textos para classificação de textos (Capítulo *4*, página *21*);
- Estudo compreensivo sobre conceitos, abordagens e medidas de desempenho na classificação multi-etiqueta e hierárquica (Capítulo *5*, página *35*), (Capítulo *6*, página *65*);
- Definição de uma metodologia, que combina abordagens usadas na classificação multi-etiqueta e na classificação hierárquica, para lidar com problemas de classificação multi-etiqueta hierárquica (Capítulo *8*, página *99*);
- Estudo comparativo do desempenho de previsão de três algoritmos multi-etiqueta, aplicados de forma hierárquica a duas colecções de documentos (Capítulo *8*, página *99*), (Anexo *2*, página *133*), (Anexo *3*, página *137*).

De notar que, apesar da abordagem seguida estar focada principalmente na classificação de textos, muitas das técnicas podem ser aplicadas à classificação de outros objectos, como imagens [Denoyer e Gallinari 2004], [Gonçalves, et al. 2007], áudio [Trohidis, et al. 2008] e aplicadas a problemas de bioinformática [Costa 2008], etc.

1.6 Organização do relatório

Com o objectivo de facilitar a leitura e melhorar a compreensão por parte do leitor, o presente documento encontra-se organizado da seguinte forma:

Os capítulos deste relatório, estão implicitamente divididos em duas grandes partes, seguido do capítulo final de conclusão. A primeira que se inicia no Capítulo *1* e termina com o Capítulo *6* e a segunda que se inicia no Capítulo *7* e termina com o Capítulo *8*. Na primeira parte é apresentado o trabalho teórico realizado, enquanto na segunda é apresentado o trabalho prático.

No Capítulo *1*, é feita uma introdução à dissertação, seguida da apresentação de vários tópicos, tais como, uma introdução à classificação de textos em geral, à classificação multi-

[7] http://www.dei.isep.ipp.pt/~i000313/ e http://www.psantos.com.pt

etiqueta hierárquica de textos e é apresentada a abordagem seguida nesta dissertação. Sendo ainda apresentadas as tecnologias, ferramentas e APIs utilizadas. No Capítulo *2*, é descrito o esquema ou sistema de classificação ACM. No Capítulo *3*, são discutidos aspectos da recolha e extracção de informação da *Web*. No Capítulo *4*, são apresentadas e feitas reflexões das principais tarefas de pré-processamento de documentos. No Capítulo *5*, são introduzidos os principais conceitos de classificação, os diferentes tipos de classificação existentes, as abordagens existentes para lidar com a classificação multi-etiqueta e é feito um levantamento bibliográfico dos trabalhos de classificação multi-etiqueta hierárquica. No Capítulo *6*, são apresentadas as medidas de desempenho utilizadas nos diferentes tipos de classificação.

No Capítulo *7*, é apresentada a solução que permitiu criar uma colecção de documentos classificados segundo o esquema ACM. No Capítulo *8*, é relatada a metodologia utilizada para classificar novos documentos e avaliação de desempenho.

No Capítulo *9*, são realizadas as conclusões finais e apresentados propostas para trabalhos futuros.

Capítulo 2.
Sistema de classificação ACM

Na secção *2.1*, começa-se por apresentar o sistema de classificação ACM versão 1998, fazendo referência às versões anteriores e à organização que o definiu. É exibido um extracto da árvore de categorias ACM, um dos principais conceitos do sistema de classificação ACM.

Na secção *2.2*, é explicado como classificar os trabalhos usando o referido sistema. Estas instruções são particularmente úteis, para a preparação de um documento, para publicação pela imprensa ACM. Podendo ser utilizadas para classificar qualquer outro documento da área das ciências da computação.

Na secção *2.3*, são apresentadas estatísticas da árvore de categorias ACM, que permitem ter uma percepção de vários aspectos, como, a sua dimensão (quantidade de categorias), a distribuição destas pelos níveis, etc.

Na secção *2.4*, são apresentados as colecções de documentos que são utilizadas nos trabalhos de investigação de classificação.

2.1 Sistema de classificação ACM versão 1998

A **Associação para Maquinaria da Computação** (*Association for Computing Machinery - ACM*), foi fundada em 1947 como a primeira sociedade científica e educacional dedicada à computação.

O primeiro sistema de classificação ACM na área das ciências da computação foi publicado em 1964. Posteriormente, em 1982, a ACM publicou um sistema totalmente novo. Seguiram-se novas versões baseadas no sistema de 1982, em 1983, 1987, 1991 e a mais recente em 1998 válida em 2007 [CCS 2008].

O sistema de classificação ACM versão 1998[8] (*ACM Computing Classification System version 1998*) ou CCS98, envolve três conceitos: a **árvore** com quatro níveis, **termos gerais** (*general terms*) e **descritores de assunto implícitos** (*implicit subject descriptors*).

2.1.1 Árvore de classificação (Taxonomia ACM)

A árvore de classificação ACM ou taxonomia ACM, consiste num conjunto de categorias na área das ciências da computação, organizadas hierarquicamente (figura seguinte).

Definição 2.1.1 – Taxonomia: foi em tempos, a ciência de classificar organismos vivos, sendo mais tarde usada com um sentido mais abrangente na classificação de "coisas". Uma

[8] http://www.acm.org/class/1998/ - todas as categorias da versão 1998
http://www.acm.org/class/ - todas as categorias de todas as versões

taxonomia pode ser vista como um tipo de estrutura, onde podemos classificar "coisas" de uma forma hierárquica, que facilita a sua identificação, estudo ou localização.

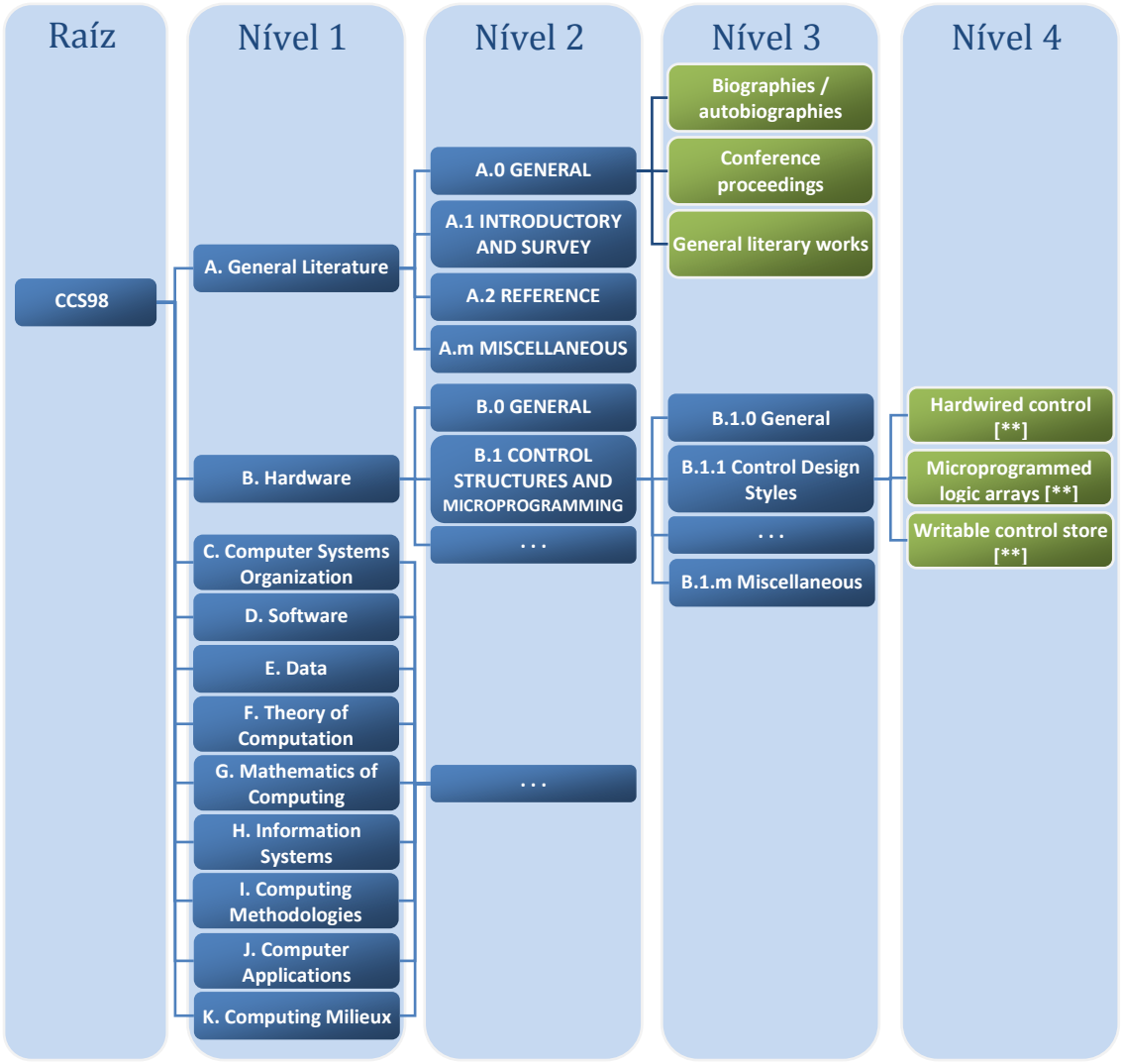


Figura 4 – Extracto da árvore de categorias ACM versão 1998

A árvore de categorias ACM, é o coração do sistema de classificação ACM [ACM CCS98 Intro 2008]. A árvore tem quatro níveis e é composta por:

- **Nós codificados ou numerados**
  Nestes nós a descrição é antecedida de um código. Encontram-se no máximo no terceiro nível, isto significa que não existem nós codificados no quarto nível.

  Exemplos (Figura 4, página 5):

  "A.1 INTRODUCTORY AND SURVEY", "B.1.1 Control Design Styles"

  Existem dos tipos de nós codificados que poderiam ser vistos como "especiais" porque aparecem ao longo da árvore ACM e são usados para cobrir necessidades "especiais". Eles são os nós *General* e *Miscellaneous* cujos respectivos códigos, são composto pelo código do nó pai seguido respectivamente em 0 e m, por exemplo, A.0, A.m, B.1.0, B.1.m (ver Figura 4, página 5). Os nós *General* são usados quando um documento cobre a maioria dos conceitos num determinado campo, os nós *Miscellaneous* se o documento não poder ser classificado sob nenhum outro nó.

- **Nós não codificados ou não numerados, chamados descritores de assunto**
  Não possuem qualquer código. Encontram-se normalmente (mas não obrigatoriamente) no quarto nível e encontram-se sempre sob um nó codificado.

  Exemplos (Figura 4, página 5):

  "Biographies/autobiographies", "Hardwired control [* *]"

Segundo [ACM CCS98 Intro 2008], a árvore de classificação está restringida a três níveis codificados, que pretendem reflectir a estrutura essencial das ciências da computação durante um longo período de tempo. Por sua vez os descritores de assunto, fornecem um nível não codificado extra, capaz de fornecer um detalhe suficiente para lidar com novos desenvolvimentos nos diferentes campos. Originalmente, pretendia-se que estes últimos fossem alterados com frequência, na prática, torna-se difícil apagar os obsoletos sem apagar as referências a trabalhos originalmente classificados sob eles. Desta forma, os descritores de assunto fazem parte permanente da árvore.
Os nós marcados com um asterisco deixaram de ser usados para classificação desde Janeiro de 1991, e os nós com dois asteriscos desde Janeiro de 1998, no entanto continuam a ser utilizados na procura.

2.1.2 Termos gerais

Existe um conjunto de dezasseis conceitos chamados termos gerais (*general terms*). Estes termos aplicam-se a todos os documentos, independentemente da categoria da árvore a que o documento possa pertencer.

Algorithms	Experimentation	Management	Security
Design	Human Factors	Measurement	Standardization
Documentation	Languages	Performance	Theory
Economics	Legal Aspects	Reliability	Verification

Tabela 1 – Termos gerais

2.1.3 Descritores de assunto implícitos

Para além dos descritores de assunto existentes explicitamente na árvore de classificação ACM, existe um conjunto denominado descritores de assunto implícitos. A lista de descritores implícitos pode ser consultada no endereço http://portal.acm.org/lookup/ccsnoun.cfm.

Os descritores de assunto implícitos são nomes de produtos, sistemas, linguagens e pessoas proeminentes no campo das ciências da computação, junto com o código de categoria sob os quais são classificados.
Por exemplo: "*C++*" seria colocado sob "*D.3.2 Language Classifications*" [CCS 2008].
Estes descritores não aparecem como parte formal do esquema porque são demasiados para serem incluídos sem tornar o esquema incómodo. Esta lista é dinâmica e é frequentemente actualizada com novos descritores. A lista está aberta a sugestões e por isso podem ser sugeridos novos descritores, acompanhados de uma pequena explicação do significado e se possível da categoria do CCS sob o qual o descritor deve ser classificado [CCS 2008].

2.2 Como classificar os trabalhos usando o CCS98

Um aspecto importante da preparação de um documento para publicação pela imprensa ACM (*ACM Press*) consiste em fornecer a informação de indexação de acordo com o sistema de classificação ACM. Este procedimento é benéfico para o autor pois a categorização exacta, fornece ao leitor uma referência rápida do seu conteúdo, facilita a procura por literatura relacionada assim como o aparecimento nos resultados das procuras feitas na biblioteca digital ACM (*ACM Digital Library*) e noutros recursos online.

A ACM necessita que seja o autor a indicar as categorias e termos gerais do seu documento por este ser o especialista no conteúdo do documento. A ACM [Classify 2008] fornece as instruções e orientações, apresentadas no 0 (página *129*).

2.3 Estatísticas da árvore ACM (categorias ACM)

As estatísticas seguintes dizem respeito à árvore de classificação ACM e permitem ter uma percepção da dimensão desta. Permitem uma percepção da distribuição das categorias ACM segundo algumas características.

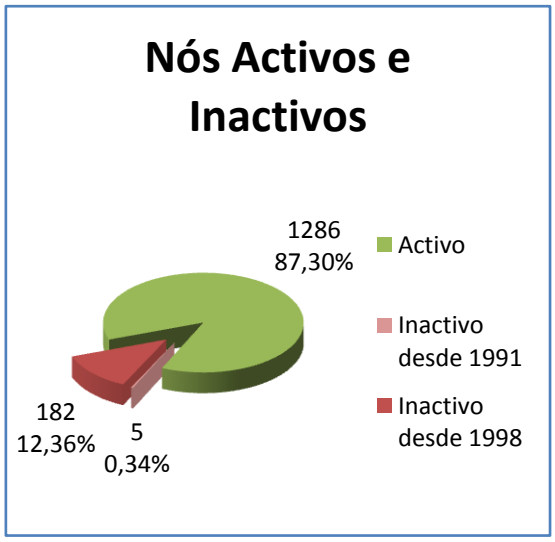


Figura 5 – Nós activos e Inactivos

Figura 6 – Nós codificados e descritores de assunto

Na figura anterior à (esquerda) é possível observar que a árvore ACM possui 1473 nós (categorias) codificados e descritores de assunto (explícitos), dos quais 1286 estão activos e os restantes inactivos, isto é, já não são usados para classificar novos documentos.

Na figura anterior à (direita) é possível observar que a árvore ACM possui 1473 nós (categorias) activos e inactivos, dos quais 368 são nós codificados e 1105 são nós descritores de assunto.

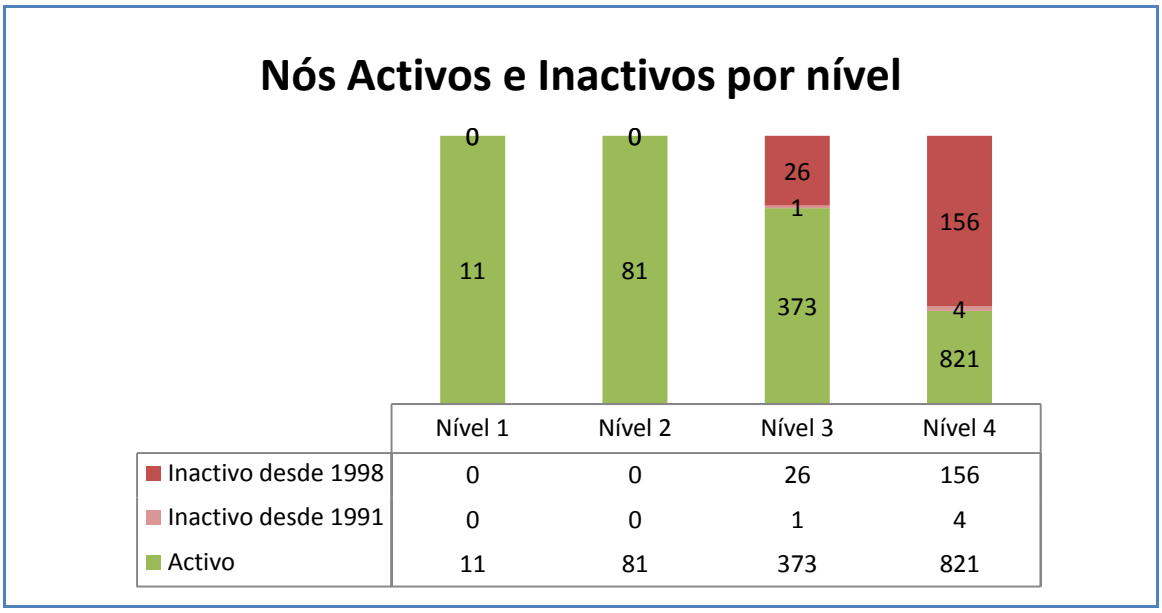


Figura 7 – Nós activos e inactivos por nível

Na figura anterior é possível ver a distribuição dos nós activos e inactivos por cada um dos quatro níveis da árvore ACM. Tanto o nível 1 como o nível 2 não possuem nós inactivos, enquanto o nível 3 e nível 4 possuem respectivamente 27 (26+1) e 160 (156+4) nós inactivos.

Na tabela seguinte pode-se observar quantos nós codificados estão activos, inactivos desde 1991 ou inactivos desde 1998 para cada um dos quatro níveis da árvore ACM, assim como a mesma informação para os nós descritores de assunto.

Nível	Nós codificados			Nós descritores assunto			TOTAL Linha
	Activos	Inactivos 91	Inactivos 98	Activos	Inactivos 91	Inactivos 98	
1	11	0	0	N.A.	N.A.	N.A.	**11**
2	81	0	0	0	0	0	**81**
3	263	0	13	110	1	13	**400**
4	N.A.	0	N.A.	821	4	156	**981**
TOTAL	**355**	**0**	**13**	**931**	**5**	**169**	**1473**

Tabela 2 – Nós codificados e descritores de assunto activos e inactivos por nível

Na tabela anterior N.A. significa não aplicável. Todas as três células da primeira linha dos nós descritores de assunto contêm o valor N.A., pois como já referido, cada nó descritor encontram-se sempre sob um nó codificado e estes últimos, só surgem no nível 1, consequentemente e teoricamente os nós descritores só poderiam aparecer a partir do nível 2, mas na prática só aparecem a partir do nível 3.

2.4 Conjunto de dados

Tal como em outras áreas existem conjuntos de dados (*datasets*), de acesso público, na classificação de textos também existe. Entre outras vantagens, o uso destes conjuntos de dados facilita a comparação dos resultados de diferentes algoritmos, aplicados por diferentes investigadores.

2.4.1 Reuters-21578

A colecção *Reuters-21578*[9] é provavelmente a colecção mais usada na classificação de textos. Têm sido vários os métodos de classificação de documentos testados com esta colecção [Dumais, et al. 1998] [Joachims 1998] [Yang 1999]. Esta colecção é composta por 21578 artigos de notícias distribuídos por 118 categorias, os artigos foram recolhidos e classificados originalmente por *Carnegie Group, Inc.* e *Reuters, Ltd.* no decorrer do desenvolvimento do sistema de classificação CONSTRUE [Hayes e Weinstein 1990].

Um artigo pode ter qualquer número de categorias, sendo a situação mais comum a atribuição de uma única categoria. As categorias não estão organizadas hierarquicamente, contudo existem autores que criam uma hierarquia artificial, como acontece por exemplo, em [Koller e Sahami 1997], [Toutanova, et al. 2001], [Wu, Zhang e Honavar 2005], [Esuli, Fagni e Sebastiani. 2006].

É comum os investigadores utilizarem um método para dividir a colecção num conjunto de treino e conjunto de teste. O método de divisão dos dados mais popular é o *ModAptesplit* [Dumais, et al. 1998], de onde resultam 9603 documentos de treino e 3299 documentos de teste. A distribuição dos documentos por categoria é muito desigual, e alguns trabalhos avaliam os sistemas com base em documentos que se encontrem nas 10 maiores categorias. As categorias encontram-se listadas na tabela seguinte.

Categoria	Número de documentos de treino	Número de documentos de teste
earn	2725	1051
acq	1490	644
money-fx	464	141
grain	399	135
crude	353	164
trade	339	133
interest	291	100
ship	197	87
wheat	199	66
corn	161	48

Tabela 3 - As dez maiores categorias da colecção Reuters-21578

2.4.2 20 Newsgroups

O *20 Newsgroups*[10] é outra colecção de dados usada na classificação de documentos, recolhida por *Ken Lang* [Lang 1995]. Consiste em 1000 artigos de cada um dos 20 grupos de notícias (*newsgroups*) ou grupos de discussão da *Usenet*. O nome do grupo de notícias é usado como categoria. Após a eliminação de artigos duplicados, contém 18941 artigos.

Apesar de não ser tão popular que a colecção *Reuters-21578*, é utilizado por muitos investigadores, por exemplo nos trabalhos de [Baker e McCallum 1998], [McCallum e Nigam

[9] Disponível em http://www.daviddlewis.com/resources/
[10] Disponível em http://kdd.ics.uci.edu/

1998], [Schapire e Singer 2000] e [Slonim e Tishby 2000]. Uma grande diferença em relação à colecção *Reuters-21578* prende-se com o facto de os textos não poderem pertencer a múltiplas categorias e estas encontra-se organizado hierarquicamente, por exemplo, *sci.crypt*, *sci.electronics*, *sci.med* são subcategorias de *sci* (*science*).

Categoria	Número de textos
alt.atheism	799
comp.graphics	974
comp.os.ms-windows.misc	985
comp.sys.ibm.pc.hardware	982
comp.sys.mac.hardware	961
comp.windows.x	980
misc.forsale	972
rec.autos	990
rec.motorcycles	994
rec.sport.baseball	994
rec.sport.hockey	999
sci.crypt	991
sci.electronics	981
sci.med	990
sci.space	987
soc.religion.christian	999
talk.politics.guns	910
talk.politics.mideast	940
talk.politics.misc	775
talk.religion.misc	628

Tabela 4 – As categorias da colecção 20 Newsgroup

A tabela mostra que a distribuição do número de textos é relativamente uniforme quando comparado com a colecção *Reuters-21578*.

2.4.3 ENZYME

A colecção ENZYME, compilada por *Rousu* e colegas [Rousu, Saunders, et al. 2006], representa sequências de proteínas e as categorias as reacções que elas catalisam. As categorias consistem num subconjunto do sistema **Classificação de Enzima** (*Enzyme Classification system*), organizadas numa hierarquia composta por 236 nós. As 172 categorias que compõem os nós folha da hierarquia, são as usadas na classificação. A colecção é composta por 7700 sequências de proteínas para treino e 1755 sequências para teste.

E.C.1 Oxidoreductases
E.C.1.1. Acting on the CH-OH group of donors.
E.C.1.1.1 With NAD(+) or NADP(+) as acceptor.
E.C.1.1.1.1 Alcohol dehydrogenase.

E.C.6 Ligases
E.C.6.1 Forming carbon-oxygen bonds.
E.C.6.1.1 Ligases forming aminoacyl-tRNA and related compounds.
E.C.6.1.1.1 Tyrosine–tRNA ligase.

Figura 8 – Exemplo da hierarquia

De notar que a colecção ENZYME, não é uma colecção de textos mas sim uma colecção de dados biológicos. Portanto, não é possível efectuar e consequentemente estudar o impacto de diferentes operações de pré-processamento de texto (*stemming*, n-gramas, etc.) nos resultados de um classificador. Porém, permite a aplicação e estudo de técnicas de classificação hierárquica.

2.4.4 Outros conjuntos

- **Reuters-RCV1**[11] (*Reuters Corpus Volume 1*), é composta por 806791 artigos de notícias (bem maior do que a colecção *Reuters-21578*), com aproximadamente 1 GB de texto. Esta colecção cobre um vasto conjunto de assuntos, desde política, negócios, desporto e ciência. Uma caracterização detalhada desta colecção pode ser encontrada em [Lewis, Yang, et al. 2004];
- **Enron email**[12]**,** é composta por 800000 emails de 150 utilizadores organizadas em 4700 directórios;
- **OHSHUMED**[13], criada por *Hersh W.* e colegas [W., et al. 1994]**.** Os documentos são compostos por títulos ou título mais resumo (*abstract*) de jornais médicos. Na verdade a colecção OHSHUMED é um subconjunto da base de documentos *Medline* [Haynes, et al. 1990].

2.5 Considerações finais

Num problema de classificação, parte-se de um conjunto de categorias pré-definidas, que neste trabalho correspondem às categorias definidas no sistema de classificação ACM, categorias e sistema esse que foram apresentados neste capítulo.

Na resolução de problemas de classificação, aplicando uma abordagem de aprendizagem automática (*machine learning*), como acontece nesta dissertação, é assumido que existe um conjunto de documentos classificados. Neste capítulo foram apresentados os principais conjuntos de textos usados na classificação de textos, assim como o conjunto de dados biológicos ENZYME, usado e referido na literatura para a classificação multi-etiqueta hierárquica.

Dos conjuntos apresentados, concluí-se que não são adequados para o estudo da classificação multi-etiqueta hierárquica de texto. Por exemplo, os conjuntos apresentados ou não possuem originalmente uma estrutura hierárquica (Reuters-21578, Reuters-RCV1), ou não são multi-etiqueta (20 Newsgroups), ou não são uma colecção de textos (ENZYME), ou não são fáceis de obter.

No estudo de pesquisa realizado, foram encontrados trabalhos, como os de *K. Wang* e colegas [Wang, Zhou e Liew 1999], *Tao Wang* e colega [Wang e Desai 2007], que fazem referência ao esquema de classificação ACM, contudo nenhum disponibilizava ou indicava um local para se obter uma colecção de textos classificados de acordo com o sistema de classificação ACM.

[11] Disponível em http://www.daviddlewis.com/resources/
[12] Disponível em http://www.cs.cmu.edu/enron
[13] Disponível em ftp://medir.ohsu.edu/pub/ohsumed9

Como tal, surgiu a necessidade de criar uma colecção própria para esta dissertação, o que explica o capítulo seguinte onde é discutido a recolha e extracção de informação da Web e a solução desenhada e implementada no Capítulo *7* (página *79*).

Capítulo 3.
Recolha e extracção de informação da Web

De forma a extrair dados ou informação das páginas *Web* (*Web data extraction*), primeiro é necessário recolher as páginas que contêm a informação pretendida. Quando estamos perante sítios *Web* muito grandes, como é o caso da biblioteca digital ACM a recolha de documentos de forma manual e o copiar e colar (*copy and past*) é inviável, devendo a recolha de documentos (páginas HTML ou outro tipo de formato) ser realizada de forma automática.

Na secção *3.1.1* (página *15*) é apresentado o processo de navegação e recolha automática de documentos (*Web crawling*), a aplicação que o faz (*Web crawler*) e as suas características. Nesta secção é ainda apresentada uma lista de *Web crawlers* de acesso público, cuja utilização evita as dificuldades e o consumo de tempo dispensados na implementação de uma solução.

Os sistemas mais conhecidos que efectuam a navegação autónoma e recolha de documentos da *Web* são os motores de busca também chamados motores de pesquisa (*search engines*).
Na secção *3.1.2* (página *17*) é apresentada a estrutura de um dos mais conhecidos motores de pesquisa: o *Google*. Esta exposição ajuda a entender o funcionamento de um motor de busca.

Uma vez recolhidas as páginas *Web* é possível proceder à sua análise e extracção de informação. Na secção *3.1.3* (página *19*) são abordados os *Wappers*. Um *wrapper* pode ser visto como um procedimento que é criado para extrair o conteúdo de uma fonte de informação particular.

Com base no trabalho de investigação realizado e reportado neste capítulo, foi desenhado e implementado uma solução para automatizar a navegação, recolha e extracção de informações (titulo, *abstract*, palavras-chave, classificação, etc.) das publicações científicas classificadas segundo o esquema de classificação ACM. Solução que é apresentada no Capítulo *7* (página *79*), juntamente com todo o trabalho experimental realizado.

3.1 Recolha das páginas Web e extracção dos dados

3.1.1 *Web* crawling

A *World Wide Web* ou simplesmente *Web* é usada por muitos sistemas como fonte de informação. Razões como o número de páginas *Web* disponíveis, o crescimento exponencial da *Web*, a sua heterogeneidade, tornam conveniente ou mesmo obrigatório a utilização de um processo automático para recolha da informação.
Web crawling é o processo pelo qual se navega e efectua a recolha (*download*) de páginas de uma maneira automática. O processo de *Web crawling* é desempenhado por um *Web crawler* que pode ser entendido como um programa ou *script*.

Definição 3.1.2 – Web Crawler (1): É um programa automatizado que acede a um sítio *Web* e movimenta-se por ele seguindo as ligações (*links*) presentes nas páginas. Conhecido como robô (*bot* ou *robot*), aranha (*spider*) ou réptil (*crawler*) [Matra 2008].

Definição 3.1.3 – Web Crawler (2): Um programa de indexação de páginas *Web* que constrói um índice, seguindo hiperligações (*hyperlinks*) continuamente de página *Web* para página *Web* [William 2000].

O processo de *Web crawling* é usado por vários sistemas e em particular nos motores de pesquisa, mas também para recolha de informação específica. Associado ao *Web crawling* existe uma gama de características genéricas a ter em conta independentemente da dimensão do projecto, elas podem ser divididas em características que um *Web crawler* **deve** ter e características que **deveria** ter. O primeiro conjunto de características pode ser visto como o de maior e o segundo como menor prioridade mas igualmente importantes.

Características que um *Web* crawler DEVE ter [Manning, Raghavan e Schütze 2007]**:**

- **Robustez**: A *Web* contém servidores que criam armadilhas para os *Web crawlers*, que maliciosamente geram páginas que os enganam, levando-os a ficarem presos em busca de um número infinito de páginas num determinado domínio. Um *Web crawler* precisa de ser concebido para lidar com estas armadilhas. Nem todas as armadilhas são maliciosas, algumas são um efeito colateral de mau desenvolvimento do *Web*site;
- **Cortesia**: Os servidores *Web* têm políticas implícitas e explícitas que regulam comportamentos, como: a taxa à qual um *crawler* os deve visitar, que zonas do sítio da Internet podem ser recolhidas.
  O *webmaster* pode recorrer a ficheiros robots.txt[14] para especificar politicas explícitas. Quando às politicas implícitas, estas devem ser respeitadas mesmo não estando especificadas, por exemplo: deve-se evitar fazer o *crawling* do mesmo sítio (*site*) com demasiada frequência.

Características que um *Web crawler* DEVERIA ter [Manning, Raghavan e Schütze 2007]**:**

- **Distribuição**: Deveria ter a capacidade de funcionar de uma maneira distribuída por várias máquinas;
- **Escalável**: A arquitectura deveria permitir aumentar a taxa de navegação e recolha pela adição de máquinas extras;
- **Desempenho e eficiência**: Deveria aproveitar ao máximo os vários recursos do sistema como processador, armazenamento e largura de banda;
- **Qualidade**: Dado que uma parte significativa da *Web* é de fraca utilidade em termos de servir as necessidades das consultas dos utilizadores, o *Web crawler* deve ter a tendência de recolher as páginas "úteis" primeiro;
- **Actualização**: Em muitas aplicações, deve funcionar de modo contínuo, deve obter cópias actualizadas das páginas recolhidas anteriormente. Os *Web crawlers* que funcionam continuamente, devem ser capazes de recolher uma página com uma frequência próxima da frequência com que a página é actualizada;

[14] Protocolo para limitar o acesso de spiders (robots) www.robotstxt.org/wc/norobots.html

- **Extensível**: Deveria ser concebido para ser extensível de várias maneiras: suportar novos formatos de dados, novos protocolos de busca e assim sucessivamente. Isto exige que a arquitectura seja modular.

O uso de um *Web crawler* para recolha da informação pode desempenhar um papel importante, no entanto a sua implementação, sobretudo com as características acima, com principal destaque para a robustez, é uma tarefa não trivial e demorada. O desenvolvimento pode ser evitado usando uma solução já existente. Algumas das soluções código aberto são:

Agent Kernel	http://www.agentkernel.com/content.agent?page_name=Home
DataparkSearch	http://www.dataparksearch.org/
GNU Wget	http://www.gnu.org/software/wget/
Heretrix	http://crawler.archive.org/
ht://Dig	http://www.htdig.org/
HTTrack	http://www.httrack.com/
JSpider	http://j-spider.sourceforge.net/
Larbin	http://larbin.sourceforge.net/index-eng.html
LWP::RobotUA	http://search.cpan.org/~marclang/ParallelUserAgent-2.57/lib/LWP/Parallel/RobotUA.pm
Methabot	http://bithack.se/methabot/
Nutch	http://www.nutch.org/
Ruya	http://sourceforge.net/projects/ruya/
Sherlock Holmes	http://www.ucw.cz/holmes/
Terrier	http://ir.dcs.gla.ac.uk/terrier/
Universal Information Crawler	http://uicrawler.sourceforge.net/
Web Crawler	http://www.noviway.com/Code/Web-Crawler.aspx
WebSPHINX	www.cs.cmu.edu/~rcm/websphinx/
WIRE	http://www.cwr.cl/projects/WIRE/
YaCy	http://yacy.net/

Tabela 5 – Web crawlers código aberto

3.1.2 Arquitectura do Google

Nesta secção apresenta-se a estrutura de um dos mais conhecidos motores de busca: o *Google*. Esta exposição ajuda a entender o funcionamento de um motor de busca.

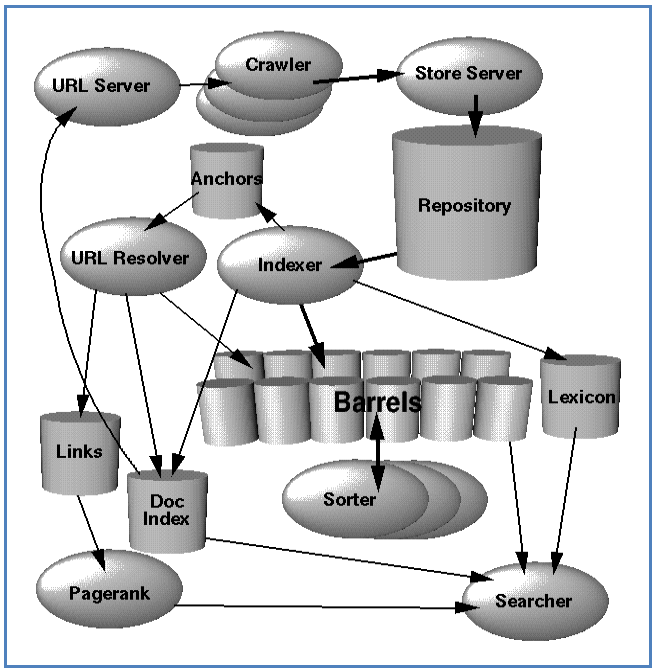


Figura 9 – Arquitectura do Google [Brin e Page 1998]

A navegação e recolha das páginas Web são realizadas por vários *crawlers* que recebem do servidor de URLs (*URL Server*) a lista de URLs (*Universal Resource Locator*).

As páginas *Web* recolhidas são depois enviadas ao servidor de armazenamento (*Store Server*) que as comprime e armazena num repositório. A cada página *Web* está associado um identificador denominado *docID* que lhe é atribuído sempre que uma página é processada. A função de indexação é executada pelo indexador (*Indexer*) e pelo ordenador (*Sorter*). O indexador lê o repositório, descomprime os documentos e analisa-os. Cada documento é convertido num conjunto de ocorrências de palavras chamado *hits*. Cada *hit* armazena o termo, a sua posição no documento e características a respeito da fonte (por exemplo: tamanho das fontes). O indexador distribui estes *hits* por barris (*Barrels*), criando índices parciais ordenados. Além disso, processa todas as ligações (*links*) das páginas, armazena-os num ficheiro de âncoras[15] com informação suficiente para determinar a origem, o destino e o texto de cada ligação (*link*).

O módulo resolução de URLs (*URL Resolver*) lê o ficheiro de âncoras, converte os URL's relativos em absolutos e transforma-os em *docIDs*. Coloca o texto da âncora num índice, associado ao *docID* que a âncora aponta. Também gera uma base de dados de ligações que são pares de *docIDs*. A base de dados de ligações é usada para o cálculo do *PageRank* para todos os documentos.

O ordenador vai aos barris que estão ordenados pelo *docID* e ordena-os por uma *palavraID* para criar o índice invertido. Esta operação é feita no local, recorrendo a uma área temporária. Também gera uma lista de palavras que desloca para um índice invertido. Um programa chamado *DumpLexicon* pega nessa lista conjuntamente com o léxico produzido pelo indexador e cria um novo léxico para ser usado pelo pesquisador (*Searcher*). Este percorre o servidor *Web* e utiliza o léxico construído pelo *DumpLexicon* conjuntamente com o índice invertido e o *PageRank* para responder aos pedidos [Brin e Page 1998].

[15] Âncora é o texto ou imagem que pode ser clicado e aparece numa hiperligação num documento HTML. No exemplo: * Site ACM *, "*Site ACM*" é a âncora.

3.1.3 Wrappers

3.1.3.1 O que é um wrapper?

Definição 3.1.4 – wrapper: Um *wrapper* pode ser visto como um procedimento que é criado para extrair o conteúdo de uma fonte de informação particular e entregar o conteúdo de interesse num formato que se auto-descreve [Eikvil 1999]. Na comunidade de base dados, um *wrapper* é um componente de software que converte dados e consultas de um modelo para outro. No ambiente Web é uma aplicação que extrai das páginas Web informação e converte-a num formato estruturado para posterior processamento.

Um *wrapper* consiste num conjunto de regras de extracção mais código necessário para aplicar essas regras, é normalmente específico para uma fonte. Para se extrair informação de várias fontes independentes é necessário um conjunto de *wrappers*. Idealmente, deve ser capaz de lidar com a alteração e natureza instável da *Web*, como falhas de rede, documentos mal formatados, alteração da disposição (*layout*), etc.
Os *wrappers* surgiram da necessidade de ferramentas capazes de extrair e integrar dados de múltiplas fontes *Web* e acabaram por evoluir independentemente da Extracção da Informação tradicional (*Information Extraction - IE*) [I. Muslea 1999]. Os sistemas de IE tradicionais, usam extracção de padrões baseados numa combinação de restrições sintácticas e semânticas. Porém, para documentos semi-estruturados como os que se aplicam os *wrappers*, habitualmente não são aplicados padrões linguísticos de extracção. De modo a lidar com novos domínios de aplicação, os investigadores introduziram um novo conjunto de padrões de extracção, padrões baseados em delimitadores, que não fazem uso das restrições linguísticas. Habitualmente as páginas que são geradas no momento (páginas dinâmicas), são-no preenchendo sempre o mesmo modelo. Por exemplo, o sítio *Web amazon* coloca o título, autor, comentários, etc., da mesma maneira em todas as páginas de livros. Depois de analisados alguns documentos de exemplo, habitualmente é possível identificar os fragmentos do modelo que representam os delimitadores de cada campo individual [I. Muslea 1998].

3.1.3.2 Criação de um wrapper

A criação ou geração de um *wrapper* (*wrapper generation*) pode ser feita **manualmente** ou recorrendo a uma abordagem **semi-automática** ou **automática** [Eikvil 1999].

Na **criação manual** o criador precisa de passar algum tempo a entender a estrutura da página *Web* e traduzi-la para código. Na prática consiste na codificação manualmente da extracção da informação das páginas *Web*, esta tarefa pode ser realizada recorrendo a uma linguagem de programação existente utilizando expressões regulares. Para pequenas aplicações, esta é uma abordagem acertada. A tarefa não é trivial, podendo tornar-se aborrecida e propensa a erros. Sistemas criados desta forma têm um custo elevado de manutenção, tornando-se num problema quando o número de páginas *Web* é grande e a estrutura varia com alguma frequência. Consequentemente estes problemas conduzem ao uso de uma outra abordagem.

A **criação semi-automática** de um *wrapper* tira proveito de ferramentas que ajudam a projectar o *wrapper*. Algumas ferramentas oferecem um interface gráfico onde o utilizador mostra ao sistema que informação extrair. Utilizando um interface gráfico, qualquer pessoa

pode efectuar uma programação por demonstração, mostrando à aplicação que campos extrair. Esta abordagem significa que o conhecimento especializado na codificação do *wrapper* não é necessário nesta fase, para além de ser menos propícia a erros. Contudo, é preciso para cada nova página e para cada página alterada demonstrar como devem ser extraídos os dados uma vez que estes sistemas não podem por si próprios induzir a estrutura de uma página *Web*.

A **criação automática** de um *wrapper* utiliza técnicas de **Aprendizagem Automática** (*Machine Learning*). No entanto, mesmo a criação automática de sistemas exige uma intervenção mínima de especialistas humanos. Habitualmente o sistema precisa de passar por uma fase de treino, onde é alimentado com exemplos de treino e em muitos casos esta aprendizagem precisa de ser supervisionada.

A indução de um *wrapper* (*wrapper induction*) é uma técnica para construir automaticamente *wrappers*, onde métodos de aprendizagem indutiva são usados para gerar as regras de extracção. O utilizador marca os dados relevantes num conjunto de páginas e o sistema aprende regras de extracção baseado nesses exemplos. A precisão destas regras depende do número e qualidade dos exemplos. Os exemplos são de boa qualidade se são representativos das páginas a serem processadas.

Neste projecto a criação do *wrapper* seguiu uma abordagem manual como descrito na secção *7.3* (página *94*).

3.2 Considerações finais

Neste capítulo, foram abordados aspectos da recolha em geral e extracção de dados da *Web*. A recolha, apenas de informação ou de páginas *Web* relevantes para um determinado tópico ou conjunto de tópicos, é conhecida na literatura como **recolha focada** (*focus crawler* [Chakrabarti, Van Den Berg e Dom 1999] ou *topic crawler* [Menczer 1997], [Menczer e Belew 1998]). Com o aumento da informação na *Web*, a recolha focada tem ganho popularidade, não é portando de estranhar, a quantidade de sistemas e projectos de diferentes áreas, que incluem um componente com essas finalidades.

A presente dissertação, pode ser vista como um exemplo que faz uso da recolha focada (e extracção de informação), como um meio para criar uma colecção de documentos classificados, segundo a taxonomia ACM. Outros projectos ou sistemas que fazem uso da recolha focada, são por exemplo, o site *SiteSeer* e *Medline*, que permitem respectivamente a procura de artigos científicos na área das ciências da computação e medicina [Bergholz e Chidlovskii 2003]. A metodologia *WebTopic* [Escudeiro 2004], que apoia o utilizador a manter continuamente actualizado uma colecção de documentos de tópicos específicos, definidos pelo utilizador. O trabalho de *Pirkola* [Pirkola 2007], que discute o uso da recolha focada, como meio para adquirir dados biológicos da *Web*.

No Capítulo *7* (página *79*) é apresentada a solução desenhada e implementada, inspirada no estudo levado a cabo no corrente capítulo, que permitiu a automatização da recolha e extracção de informação específica.

Capítulo 4.
Processo de Indexação dos Documentos

No desenvolvimento de um sistema de classificação de textos podem distinguir-se três grandes fases [Giorgetti e Sebastiani 2003]: **indexação dos documentos** (*document indexing*), **aprendizagem de um classificador** (*classifier learning*) e **avaliação do classificador** (*classifier evaluation*).

Os métodos empregues no processo de indexação, provém habitualmente do campo da **Recuperação de Informação** (*Information Retrieval*). Este processo, envolve um conjunto de tarefas de pré-processamento dos documentos, que podem ser realizadas de várias maneiras. A figura seguinte, ilustra um possível conjunto e possível sequenciação de tarefas de pré-processamento, do processo de indexação dos documentos.

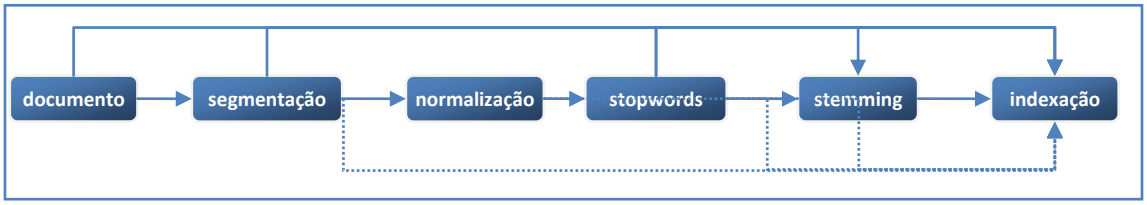


Figura 10 – Processo de indexação dos documentos

Definição 3.2.5 – pré-processamento: O pré-processamento, pode ser definido como um processo transformativo, aplicado normalmente para reduzir o número de termos de um documento, de forma a obter uma representação dos documentos, mais adequada para as fases seguintes.

As tarefas de pré-processamento, desempenham um papel importante na classificação de textos, visto poderem afectar as fases posteriores.

Na secção *4.1* (página *21*) é abordado a representação dos documentos, uma das primeiras escolhas, que é necessário fazer no processo de classificação de textos. Nas secções seguintes são abordadas as tarefas de pré-processamento, mais comuns, como por exemplo, segmentação do texto (secção *4.2*, página *23*), normalização dos termos (secção *4.3*, página *24*), remoção de *stopwords* (secção *4.4*, página *25*), *stemming* (secção *4.5*, página *26*), esquemas de atribuição de pesos aos termos (secção *4.6*, página *27*).

4.1 Representação dos documentos

Um dos primeiros passos no processo de classificação de textos é escolher uma representação mais adequada dos documentos de texto de modo a facilitar a sua manipulação.

Uma das formas mais comuns de representar um texto é através de um vector de palavras (ou termos), em que cada componente do vector representa uma palavra (ou termo) e possui um valor numérico associado (ver tabela seguinte).

Termos	doc_1	doc_2	doc_3	doc_4	doc_5	doc_6	…
projecto	4.1	0.0	3.7	5.9	3.1	0.0	
automática	4.5	4.5	0	0	11.6	0	
multi-etiqueta	0	3.5	2.9	0	2.1	3.9	
classificação	0	3.1	5.1	12.8	0	0	
aplicações	2.9	0	0	2.2	0	0	
projecto	7.1	0	0	0	4.4	3.8	
…							

Tabela 6 – Exemplo de vectores de termos de seis documentos

Por exemplo, na representação acima é possível observar cada um dos documentos $doc_1, doc_2, \ldots, doc_n$ representados por um vector de palavras ou termos, em que cada palavra possui um valor numérico que representa a importância desta no respectivo documento e na colecção de documentos. O conjunto de documentos representados desta forma, originam uma matriz de termos por documentos. Diz-se que os documentos representados desta forma, partilham ou estão representados num espaço vectorial comum, esta representação é conhecida como **modelo de espaço vectorial** (*vector space model*). Neste tipo de representação a ordem dos termos dentro do documento não é representada, nesta perspectiva, diz-se que os documentos são tratados como **saco de palavras** (*bag of words – BOW* ou *bag of terms*).

No **modelo de espaço vectorial** (*vector space model*), cada documento é encarado como um vector de n[16] componentes, um componente para cada termo, obtendo-se desta forma um espaço vectorial **n-dimensional** onde cada termo é um eixo e onde os documentos residem.
Como a representação gráfica dos documentos anteriores obrigariam a desenhar um espaço de 6 dimensões (uma por cada termo), difícil de imaginar, exibe-se na figura seguinte a representação de outros documentos num espaço tridimensional.

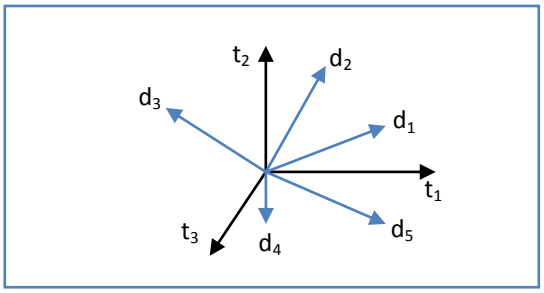


Figura 11 – Representação de documentos num espaço vectorial tridimensional

O modelo de espaço vectorial desempenha um papel muito importante, que para além de usado na classificação de textos é o ponto de partida para muitas outras tarefas de processamento automático de texto, como por exemplo, no **agrupamento** (*clustering*), na **sumarização** (*summarization*) e recuperação de informação [Baeza-Yates e Ribeiro-Neto 1999].

[16] n = número de termos distintos da colecção de documentos

Como já referido, a uma palavra ou termo do vector é associado um peso. O peso pode ser binário, para indicar a presença ou ausência do termo no texto, ou numérico, como por exemplo, o número de vezes que um termo ocorre no documento, por vezes normalizado pelo número total de termos do documento. Um esquema usado frequentemente na atribuição dos pesos é o TFxIDF *Term Frequency - Inverse Document Frequency*:

$$TF \times IDF = TF(t_k, d_i) \times IDF(t_k) = TF(t_k, d_i) \times \log \frac{|D|}{DF(t_k)} \quad (4.1.1)$$

Onde:

- $TF(t_k, d_i)$ é o número de vezes que o termo t_k ocorre no documento d_i;
- $|D|$ é o número total de documentos;
- $DF(t_k)$ é o número de documentos em que o termo t_k ocorre.

A fórmula anterior, atribui maior peso aos termos que ocorrem com mais frequência num documento, mas raramente na colecção de documentos.

Um texto para além de poder ser representado por palavras únicas (**unigramas**), pode ser representado por **bigramas** [Bekkerman e Allan 2004] ou genericamente **n-gramas**[17] (*n-grams*) [Caropreso, Matwin e Sebastiani 2001] pode ser também representado por conceitos, frases ou parágrafos. *David Lewis* [D. D. Lewis 1992a] foi o primeiro a estudar o efeito das frases sintácticas na classificação de textos. No seu estudo, um classificador *Naive Bayes* teve um desempenho um pouco abaixo de um classificador padrão utilizando um **saco de palavras únicas** (*bag of single words*). Outros trabalhos dão conta que não existe uma melhoria significativa de desempenho com a inclusão de frases sintácticas em classificadores baseados em regras [Scott e Matwin 1999], em classificadores *bayesianos* e em classificadores de Máquinas de Vectores de Suporte (*Support Vector Machines*) [Dumais, et al. 1998].

4.2 Segmentação do texto

Com o objectivo de criar um vector de termos ou representar os documentos de outra forma é necessário começar-se por segmentar o texto em unidades (processo conhecido como *tokenization*). Neste processo é também frequente eliminar-se certos caracteres como por exemplo sinais de pontuação e, passar-se todos os caracteres para maiúsculas ou minúsculas, esta passagem pode também ser realizada numa fase posterior de normalização dos termos (secção *4.3*, página *24*).

Segundo [Liao, Alpha e Dixon 2003] a segmentação do texto é um problema trivial para linguagens como o Inglês que possuem delimitadores espaços (espaços, tabs, nova linha, etc.). Segundo *Ian Witten* e *Frank Eibe* [Witten e Frank 2005, 310] a conversão do texto em palavras não é uma operação tão simples como parece. A tarefa de segmentação pode parecer directa mas quando analisada de forma mais pormenorizada e cuidadosa, conclui-se que existem um número de situações enganadoras. Por exemplo, o que fazer com palavras com hífen, como *multi-etiqueta*? ou na língua inglesa, palavras com apóstrofe, como *doesn't* e *aren't*? Devem ser tratadas como um único símbolo ou múltiplos símbolos? Por exemplo, *are not*, ou *doesn´t*?

[17] Subsequência de *n* itens de uma sequência. Os itens podem ser letras ou palavras. No contexto desta exposição refere-se a palavras.

Uma abordagem simples consiste em fazer a divisão em todos os caracteres não alfa numéricos, mas se em alguns casos a divisão parece bem, noutros não, por exemplo, *doens't* após divisão origina *doesn* e *t*. Habitualmente é preciso ter algum conhecimento do texto e domínio de aplicação do problema para se tomar uma decisão. Repare-se que existem termos que se pretende que sejam reconhecidos como um único: Dr., C++, C#, meu@email.pt, http://www.acm.org, 168.42.30.220.

4.3 Normalização dos termos

Em muitos casos é desejável que termos escritos de forma diferente sejam encarados como sendo o mesmo ou equivalentes. Por exemplo, é desejável que o termo USA e U.S.A sejam vistos como sendo o mesmo.

A normalização de termos tem em vista permitir que termos apesar de escritos de forma diferente possam ser considerados equivalentes. Uma das formas de efectuar a normalização consiste na criação implícita de classes de equivalência através de regras de mapeamento. Por exemplo, regras que permitam interpretar *contra-curva* e *contracurva* como equivalentes.
A definição de regras de mapeamento tem como vantagem a equivalência ser feita de forma implícita em vez de ser realizada antecipadamente. Estas regras são fáceis de criar, mas apenas para remoção de caracteres. Seria difícil, por exemplo, saber transformar *contracurva* em *contra-curva*.

Uma alternativa à criação de classes equivalentes por regras de mapeamento, consiste na manutenção de relações entre termos não normalizados. Este método pode ser alargado para suportar sinónimos, de modo a que, por exemplo, *carro* e *automóvel* sejam vistos como equivalentes.
As relações entre termos podem ser conseguidas de duas formas:

- Indexar os termos não normalizados e proceder à **expansão da consulta** (*query expansion*). A expansão de consultas consiste na adição de termos relacionados no momento em que se efectua a consulta. Permite aumentar a qualidade do processo de obtenção dos documentos;
- Proceder à expansão no momento da criação do índice. Por exemplo, se um documento contém o termo *automóvel*, ele é indexado também sob *carro* e vice-versa.

O uso de qualquer uma destas duas formas é menos eficiente que o uso de **classes de equivalência** por regras de mapeamento (*equivalence classes*). No caso da expansão da consulta é exigido mais processamento no momento da consulta, no caso da expansão no momento da criação do índice é exigido maior espaço de armazenamento.
Qualquer uma destas duas formas de expansão é mais flexível do que o uso de classes de equivalência porque permitem, por exemplo, a expansão assimétrica.
No exemplo abaixo é possível ver como a expansão assimétrica pode ser explorada. Se for efectuada uma consulta por *windows* (janelas), pode-se permitir que sejam devolvidos resultados como *Windows* (W em maiúscula, correspondente ao sistema operativo), mas não é aceitável se a consulta for *window* (janela), mesmo sendo aceitáveis resultados com *windows* (w em minúscula).

Termo da consulta	Termos nos documentos que devem ser devolvidos
Windows	Windows
windows	Windows, windows, window
window	windows, window

Tabela 7 – Exemplo de expansão assimétrica [Manning, Raghavan e Schütze 2007]

Existem outras formas de normalização. Em vários casos elas parecem ajudar, mas também podem provocar danos, como tal deve-se ter cuidado na sua aplicação:

- **Acentos**, em línguas como o Português, Espanhol, Francês entre outras existem palavras acentuadas. Pode ser desejável, por exemplo, que *porque* e *porquê* sejam mapeados. A normalização pode ser feita removendo os acentos. Ocasionalmente as palavras apenas são distinguidas pelo acento. Por exemplo, *avô* significa pai da mãe ou do pai, enquanto *avó* significa mãe do pai ou da mãe.
- **Capitalização**, uma estratégia habitual é a passagem de todos os caracteres para minúscula. Frequentemente é uma boa ideia, que permite que as palavras que surjam no inicio de uma frase concordem com outras, por exemplo, *Portuguesa* (maiúscula se iniciar uma frase) irá concordar com *portuguesa*. Por outro lado algumas palavras devem ficar fora deste processo. Tome-se como exemplo muitos nomes próprios que derivam de nomes comuns e que apenas se distinguem pela maiúscula, por exemplo, Ribeiro (nome próprio) e ribeiro (nome comum).
  Uma alternativa à passagem de todos os caracteres a minúsculas é a passagem apenas de alguns. A heurística mais simples consiste em converter para minúsculas palavras que iniciem uma frase e todas as palavras que acorram num título todo em maiúsculas ou em que a maior parte das palavras estão ou iniciem em maiúsculas. A mesma tarefa pode ser realizada de forma mais precisa recorrendo a outros métodos. Contudo, passar todos os caracteres para minúsculas é a solução mais prática.
- **Outras situações**, é possível realizar outras normalizações particulares de certas línguas. Por exemplo, pode ser desejável mapear a palavra *colour* assim escrita no Reino Unido com a palavra *color* escrita nos E.U.A, ou a palavra *Brasil* assim escrita em Portugal e *Brazil* escrita no Brasil.
  Datas, horas e outros dados semelhantes podem surgir em vários formatos. Pode ser desejável, por exemplo, reapresentar no mesmo formato as datas 01/02/98 e 01 de Fevereiro de 1998. No entanto, o processamento correcto pode ser complicado, tome-se como exemplo o facto de nos E.U.A 01/02/98 significar Janeiro/02/1998 e em geral na Europa 01/Fevereiro/1998.

4.4 Remoção de *stop-words*

É frequente remover-se as palavras que são extremamente frequentes e que parecem de pouco valor. Essas palavras são chamadas *stop-words* ou **palavras negativas**. As *stop-words* podem ser artigos, pronomes, preposições, advérbios entre outras palavras próprias do domínio de aplicação.

A remoção das *stop-words* é realizada com base numa lista denominada frequentemente por *stop-list*. De notar que as listas de *stop-words* dependem da língua. Uma estratégia para criar a lista consiste em ordenar os termos pela frequência que aparecem na colecção de textos a

analisar e obter os mais frequentes, os menos frequentes podem também ser adicionados à lista visto serem frequentemente descartados.

A remoção de *stop-words* reduz significativamente o tamanho do índice. Em vários casos não indexar *stop-words* não traz consequências, palavras como: *de*, *o*, *a*, *para* não são úteis. No entanto, não é verdade se em vez de palavras forem armazenadas frases, por exemplo, se em vez de se armazenar a frase *Voar para a Madeira*, que contém duas *stop-words*, se armazenar *Voar Madeira*, perde-se algum significado

4.5 Stemming

O *stemming* faz normalmente parte do processo de normalização dos termos. Por razões gramaticais, utiliza-se várias formas de uma palavra. Por exemplo, **desenvolve**, **desenvolvida**, **desenvolvidas** e **desenvolvimento**. Em várias situações, seria útil que estas palavras fossem vistas como equivalentes (tivessem uma mesma representação).

O processo de *stemming* reduz as palavras ao mesmo *stem*[18], por meio da retirada dos afixos (sufixos e prefixos) da palavra, permanecendo apenas a raiz [Spark-Jones e Willett 1997]. A raiz obtida não é necessariamente igual à raiz linguística [Chaves 2003].
O processo de *stemming* refere-se normalmente a um conjunto de heurísticas que são aplicadas para obter a raiz da palavra.
Por exemplo:

desenvolve
desenvolvida
desenvolvidas
desenvolvido
desenvolvidos
desenvolvimento

 desenvolv

O processo de *stemming* pode originar dois tipos de erro [Chaves 2003, 2]:

- *Overstemming*, que ocorre quando é removido não apenas o sufixo, mas uma parte do *stem*;
  Exemplo:
  gramática  grama (em vez do *stem* correcto gramát)
- *Understemming*, que ocorre quando um sufixo não é removido completamente;
  Exemplo:
  referência  referênc (em vez do *stem* correcto refer)

De notar que a aplicação de um *stemmer* faz aumentar a **abrangência** (*recall*[19]) e diminuir a **precisão** (*precision*[20]). Por exemplo, as palavras seguintes podem ser reduzidas ao *stem* **oper**:

```
operar operando opera operação operativo operativas operacional
```

ao realizar-se uma procura, por exemplo, por "sistema AND operativo" é devolvido um número de documentos superior (a abrangência aumenta), por outro lado, nos documentos

[18] *Stem* é o conjunto de caracteres resultante de um procedimento de *stemming*
[19] Fracção de documentos relevantes de uma colecção que são devolvidos
[20] Fracção de documentos devolvidos que são relevantes

devolvidos a proporção de documento de interesse (aqueles que contém "sistema operativo") diminui (a precisão diminui).

O algoritmo de *stemming* para a língua inglesa mais usado é o *Porter Stemmer* [Porter 1980]. A descrição e implementação para várias linguagens de programação pode ser encontrado no sitio de Internet oficial http://www.tartarus.org/~martin/PorterStemmer/.

O *stemming* baseia-se normalmente num conjunto de heurísticas para atingir o seu objectivo. Alternativamente ao *stemming* existe a *lemmatization*, que emprega uma análise mais precisa que se baseia numa análise morfológica das palavras, com o objectivo de devolver a palavra base conhecida como lexema[21] (*lemma*).

4.6 Selecção de características ou redução de dimensão

O número elevado de termos, é uma característica da classificação de documentos, com fortes implicações negativas na maior parte dos métodos de aprendizagem de um classificador. *Dunja Mladenic* [D. Mladenic 1998a, 1] faz as seguintes observações:

- Um número elevado de termos pode diminuir significativamente o processo de aprendizagem do classificador, acabando por fornecer um resultado similar ao obtido com um conjunto de termos mais pequeno;
- Os termos usados para descrever os documentos não são todos necessariamente relevantes e benéficos para a aprendizagem do classificador podendo mesmo reduzir a sua qualidade.

O *stemming* ou outra redução morfológica como a *lemmatization*, a remoção de *stopwords* são formas de diminuir o número elevado de termos. Outra forma de reduzir a dimensionalidade é efectuar uma **selecção de características** (*feature selection*) também conhecida como **redução de dimensão** (*dimension reduction*). Estas duas denominações, resultam do facto dos termos poderem ser vistos como características ou dimensões. A selecção de características tem como objectivo, a partir do conjunto T de termos que representam os documentos obter um subconjunto T' dos termos mais informativos ($|T'| < |T|$). *Fabrizio Sebastiani* e *Franca Debole* [Debole e Sebastiani 2003] medem o **factor de redução** (*reduction factor*) através de:

$$\varepsilon = \frac{|\mathrm{T}| - |\mathrm{T}'|}{|\mathrm{T}|} \qquad (4.6.2)$$

Normalmente a selecção de características é feita atribuindo uma pontuação a cada termo do conjunto T, aplicando uma função, método ou medida e escolhendo um subconjunto de termos T' que maximizam a função. Algumas das medidas habitualmente usadas são: o **ganho de informação** (*information gain*), **informação mútua** (*mutual information*), $\boldsymbol{x^2}$ (*chi-square*) e **frequência**. Grande parte destas medidas tentam capturar a relação entre um termo t_k e uma categoria c_i, *Zeng*, *Wu* e *Srihari* [Zheng, Wu e Srihari 2004] classificam as medidas de acordo com quatro relações possíveis:

[21] É a unidade mínima com sentido pleno, no plano lexical. Equivale ao tema ou radical da palavra [Pinto, Parreira e Lopes 1988]

	t_k	$\neg t_k$
c_i	(t_k, c_i)	$(\neg t_k, c_i)$
$\neg c_i$	$(t_k, \neg c_i)$	$(\neg t_k, \neg c_i)$

Tabela 8 – Quatro relações entre um termo t_k e uma categoria c_i

1. (t_k, c_i): t_k e c_i co-ocorrem;
2. $(t_k, \neg c_i)$: t_k ocorre sem c_i;
3. $(\neg t_k, c_i)$: c_i ocorre sem t_k;
4. $(\neg t_k, \neg c_i)$: nem t_k nem c_i ocorrem.

As duas primeiras relações dizem respeito à presença de um termo, enquanto as duas últimas à sua ausência.

John, *Kohavi* e *Pfleger* [John, Kohavi e Pfleger 1994] apontam as duas principais abordagens usadas em **aprendizagem automática** (*machine learning*) na selecção do subconjunto de características: a abordagem *wrappers* (*wrapper approach*) ou abordagem *filtro* (*filter approach*). Na abordagem *wrapper* é escolhido o subconjunto de características que conduz ao melhor desempenho de classificação de um dado algoritmo, enquanto na abordagem *filtro* são escolhidas as características mais relevantes sem considerar o algoritmo de aprendizagem. Esta última abordagem é computacionalmente mais fácil e por isso é a normalmente escolhida na classificação de textos [Kiritchenko 2005].

É também possível distinguir-se dois tipos de abordagem na selecção de características: a **abordagem global** (*global feature selection*) e a **abordagem local** (*local feature selection*). Na **abordagem global** existe um único conjunto de termos que é usado para classificar os textos em todas as categorias, na **abordagem local** existe mais do que um conjunto de termos associados a determinadas categorias. A figura abaixo ilustra as duas abordagens.

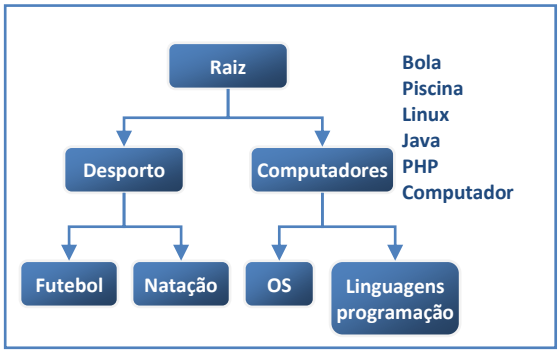


Figura 12 – Selecção de características global

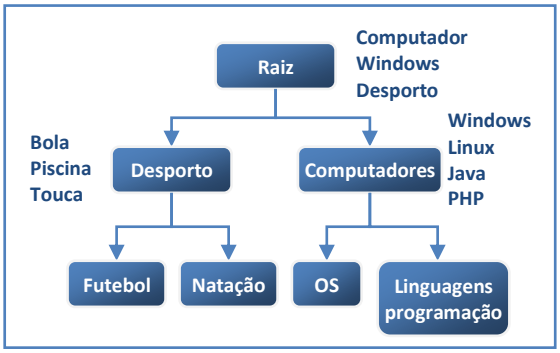


Figura 13 – Selecção de características local

Na hierarquia de categorias exibidas nas duas figuras anteriores, observa-se (à esquerda) a existência de um único conjunto de termos que são usados para todas as categorias e na outra figura observa-se que existe um conjunto de termos usado para se distinguir entre as categorias **Desporto** e **Computadores**, cada uma destas categorias tem o seu próprio conjunto de termos para distinguir as respectivas subcategorias.

4.6.1 Ganho de Informação

O **ganho de informação** (*information gain*) é uma medida usada na construção de árvores de decisão [J. R. Quinlan 1993], mas também noutras áreas. O cálculo do ganho de informação

tem por base o cálculo da **entropia** (*entropy*) que teve origem na teoria da informação de Shannon [Shannon e Weaver 1949].

Definição 4.6.6 – Entropia: A entropia é uma medida de incerteza de um acontecimento ou conjunto de dados podendo ser vista também como uma medida da impureza ou desordem de um acontecimento ou conjunto de dados. A entropia é medida em bits.

Exemplo de entropia associado a dois acontecimentos. Imagine-se dois acontecimentos:

1. O lançamento de uma moeda não viciada (50% - 50%);
2. Lançamento de uma moeda viciada em que é sabido que cara tem uma probabilidade de 99% de sair.

Ambos os acontecimentos estão envolvidos em incerteza, embora o segundo apresente uma incerteza (entropia) menor pois é sabido que existe 99% de probabilidades de sair cara.

Imagine-se agora um conjunto de documentos $D = \{d_1, \dots, d_{|D|}\}$, classificados numa categoria do conjunto $C = \{c_1, \dots, c_{|C|}\}$. A entropia (incerteza) do conjunto de documentos D, é a entropia associada ao conjunto de categorias C e é obtida como explicado em seguida.

A **entropia** do conjunto de categorias C, cuja probabilidade de cada categoria $c_1, \dots, c_{|C|}$ é respectivamente $P(c_1), \dots, P(c_{|C|})$, é obtida pela fórmula:

$$H(C) = H\big(P(c_1), P(c_2), \dots, P(c_{|C|})\big) = -\sum_{i=1}^{|C|} P(c_i)\, log_2\, P(c_i) \qquad (4.6.3)$$

Onde:

- $H(C)$ é a entropia da variável C (entropia do conjunto de dados D);
- $P(c_i)$ é a probabilidade de ocorrer a categoria c_i;
- $|C|$ é o número de categorias.

O logaritmo é base dois, para que a entropia seja expressa em bits [Cover e Thomas 2006].
Por definição, se $P(c_i) = 0$ então $P(c_i)\, log_2\, P(c_i) = 0$. Da fórmula observa-se que $H(C) = 0$ se e apenas se uma das probabilidades $P(c_i)$ é 1 e todas as restantes são 0. Esta situação apenas acontece quando não existe incerteza no resultado. Em todos os outros casos a entropia é positiva. A entropia é máxima quando a incerteza é máxima e o seu valor pode ser obtido por $\log_2 |c|$. A entropia é máxima quando a probabilidade de cada um dos acontecimentos é igual, isto é: $P(c_i) = \frac{1}{|C|}$ (i=1,2, …, |c|).

Exemplo de entropia de um conjunto de documentos.
Imagine-se um conjunto de documentos $D = \{d_1, d_2, d_3, d_4\}$, compostos pelos termos $T = \{t_1, t_2, t_3, t_4\}$ e classificados numa categoria do conjunto $C = \{a, b, c\}$.

		Termos				
		t_1	t_2	t_3	t_4	Categoria
Documentos	d_1	X		X		a
	d_2		X			b
	d_3	X	X	X		a
	d_4				X	c

Tabela 9 – Conjunto de documentos, classificados nas respectivas categorias

A entropia do conjunto de documento D, é a entropia do conjunto de categorias C.
A entropia do conjunto de categorias C, representada por $H(C)$, é:

$$H(C) = H\big(P(a), P(b), P(c)\big) = -\left(\frac{2}{4}log_2\frac{2}{4} + \frac{1}{4}log_2\frac{1}{4} + \frac{1}{4}log_2\frac{1}{4}\right) bits$$

O conjunto de documentos D anterior é composto pelos termos t_1, t_2, …, $t_{|T|}$ que se espera estarem relacionados com as categorias c_1, c_2, …, $c_{|C|}$. Se for escolhido um termo t_k que pode tomar n valores ($n = 2$, ocorre ou não ocorre) o conjunto D fica dividido em dois subconjuntos. O subconjunto dos documentos que possuem o termo t_k e o subconjunto dos documentos que não possuem t_k. A entropia esperada se o atributo t_k for escolhido é chamada **entropia condicional** (*conditional entropy*). É medida em bits e calculada pela fórmula:

$$H(C|t_k) = -\sum_{j=1}^{n} P(t_{k_j}) \times \sum_{i=1}^{|C|} P(c_i|t_{k_j}) \log_2 P(c_i|t_{k_j}) \qquad (4.6.4)$$

Onde:

- $H(C|t_k)$ é a entropia condicional da categoria C sabendo que o termo t_k ocorre;
- n e $|C|$ são respectivamente o número de valores que o termo t_k e a categoria pode tomar;
- $P(t_{k_j})$ é a probabilidade do termo t_k com o valor j ocorrer no conjunto D;
- $P(c_i|t_k)$ a probabilidade condicional da categoria c_i ocorrer sabendo que o termo t_k ocorre.

A entropia condicional permite obter a entropia esperada se um termo t_k for escolhido.
De notar que $-\sum_{i=1}^{|C|} P(c_i|t_{k_j}) \log_2 P(c_i|t_{k_j})$ corresponde à entropia (equação (4.6.3)) do subconjunto de documentos resultantes do facto de se ter escolhido o termo t_{k_j} quando este possui o valor j. Como tal, a equação anterior pode ser apresentada de forma equivalente, como:

$$H(C|t_k) = \sum_{j=1}^{n} P(t_{k_j}) \times \mathrm{H}\,(t_{k_j}) \qquad (4.6.5)$$

A diferença entre a entropia de um conjunto de documentos D antes ($H(D)$) e após a escolha de um termo t_k ($H(D|t_k)$), permite obter o **ganho de informação** em bits:

$$GI(D, t_k) = H(D) - H(D|t_k) \qquad (4.6.6)$$

Quando se pretende obter o termo que oferece maior ganho de informação, efectua-se o cálculo para cada um deles e escolhe-se aquele que possui o valor mais elevado. O ganho de informação pode ser visto genericamente entre duas variáveis aleatórias X e Y $GI(X,Y)$, como sendo a informação obtida pela diferença entre a incerteza de X antes e depois de se conhecer Y, dito por outras palavras, pode ser visto como a redução da incerteza de uma variável aleatória X trazida pelo conhecimento de uma outra Y. O ganho de informação $GI(X,Y) = 0$ se X e Y são independentes, e cresce monotonicamente com a sua dependência [Cover e Thomas 1991].

4.6.2 Informação Mútua

Em relação à **informação mútua** (*mutual information*), existe na literatura, uma diversidade de situações que podem causar confusão e dificultar a sua compreensão. As diferentes situações, prendem-se com o uso da **informação mútua** e do **ganho de informação**.

Segundo *Manning* e *Schutze* [Manning e Schutze 1999], o **ganho de informação** é também conhecido como **informação mútua**.

Yang e *Pedersen* em [Yang e Pedersen 1997], utilizaram o **ganho de informação** e a **informação mútua** (ou seja, tratam-nas como medidas diferentes). Definindo a **informação mútua** entre um termo t_k e uma categoria c_i, como:

$$I(t_k, c_i) = \log \frac{P(t_k, c_i)}{P(t_k) \times P(c_i)} \approx \log \frac{A \times N}{(A + C) \times (A + B)} \quad (4.6.7)$$

(esta definição é diferente da definida por *Shannon* e abordada nos próximos parágrafos)
Onde:

- A é o número de vezes que o termo t_k e a categoria C_i co-ocorrem;
- B é o número de vezes que o termo t_k ocorre sem a categoria C_i;
- C é o número de vezes que o termo t_k ocorre sem a categoria C_i;
- N é o número total de documentos;
- Restantes elementos significam o mesmo que na secção anterior.

Yang e *Pedersen* [Yang e Pedersen 1997] reportam um desempenho pobre obtido com a **informação mútua** o que não aconteceu com o **ganho de informação**. Sabendo-se que a fórmula utilizada (4.6.7) difere da usada em outros trabalhos, a conclusão retirada tem que ser analisada neste contexto.

Roberto Basili e *Alessandro Moschitti* [Basili e Moschitti 2001] utilizam também a fórmula (4.6.7) mas com logaritmo na base dois o que permite obter o resultado em *bits*:

$$I(t_k, c_i) = \log_2 \frac{P(t_k, c_i)}{P(t_k) \times P(c_i)} \quad (4.6.8)$$

Curiosamente esta fórmula é apresentada em [Silva 2003] com o nome de *association ratio*, proposta por *Church* e *Hanks* [Church e Hanks 1990] e baseada no conceito da informação mútua.

Nos trabalhos[Butte e Kohane 2000], [Dumais e Chen. 2000], [Silva 2003], [Jiang, et al. 2005], [Su e Zhang 2006], [Bekkerman, Eguchi e Allan 2006] é empregue o nome **informação mútua** [Shannon 1948] para fazer referência à fórmula seguinte que é equivalente ao **ganho de informação** (fórmula (4.6.6), apresentada na secção anterior):

$$MI(t_k; c_i) = \sum_{t_{k_j}=1}^{n} \sum_{c_i=1}^{|C|} P(t_{k_j}, c_i) \, log_2 \frac{P(t_{k_j}, c_i)}{P(t_{k_j})P(c_i)} = H(D) - H(D|t_k) \qquad (4.6.9)$$

A formula anterior, é definida em [Cover e Thomas 1991] mas restringindo os valores de n e $|C|$ ($n = |C| = 2$), sendo aplicada nos trabalhos [R. Bekkerman, R. El-Yaniv, et al. 2001] [R. Bekkerman, R. El-Yaniv, et al. 2003] [R. Bekkerman 2003, 10].

Ron Bekkerman [R. Bekkerman 2003, 10], chama à atenção para a confusão cometida em [Yang e Pedersen 1997].
Debole e *Sebastiani* [Debole e Sebastiani 2003], chamam à atenção, que muitos autores empregam um ou outro nome, aplicado à mesma fórmula.

Nesta dissertação, a definição adoptada para a **informação mútua,** é aquela definida em [Shannon 1948] e apresentada na fórmula (4.6.9). Consequentemente, a **informação mútua** é outro nome para o **ganho de informação** (fórmula (4.6.6), na secção anterior).

4.6.3 Estatística x^2 (*chi-square*)

Em estatística, a medida x^2 (*chi-square*) mede a independência entre duas variáveis aleatórias. Dois acontecimentos A e B são definidos independentes se $P(AB) = P(A)P(B)$ ou equivalentemente $P(A|B) = P(A)$ e $P(B|A) = P(B)$.

Na selecção de características, as duas variáveis são t_k que representa um termo e c_i que representa uma categoria, sendo o objectivo determinar a independência entre esta e o termo t_k:

$$x^2(t_k, c_i) = \sum_{t_k \in \{0,1\}} \sum_{c_i \in \{0,1\}} \frac{(N_{t_k c_i} - E_{t_k c_i})^2}{E_{t_k c_i}} \qquad (4.6.10)$$

Onde:

- N é a frequência observada;
- E é a frequência esperada.
  Por exemplo, E_{11} é a frequência esperada de t_k e c_i ocorrerem juntos num documento, assumindo que o termo e classe são independentes:

$$E_{11} = N \times P(t_k) \times P(c_i) = N \times \frac{N_{11} + N_{10}}{N} \times \frac{N_{11} + N_{01}}{N}$$

Exemplo: Dada a tabela de contingência seguinte:

	$t_k = rato = 1$	$t_k = \neg rato = 0$
$c_i = hardware = 1$	$N_{11} = 49$	$N_{01} = 141$
$c_i = \neg hardware = 0$	$N_{10} = 27652$	$N_{00} = 774106$

A frequência esperada do termo *rato* e a categoria *hardware*, é calculada da seguinte forma:

$$E_{11} = 801948 \times \frac{49 + 141}{801948} \times \frac{49 + 27652}{801948} \approx 6{,}6$$

Calculando as restantes frequências esperadas $E_{t_k c_i}$ da mesma maneira:

	$t_k = rato = 1$	$t_k = \neg rato = 0$
$c_i = hardware = 1$	$E_{11} \approx 6{,}6$	$E_{01} \approx 183{,}4$
$c_i = \neg hardware = 0$	$E_{10} \approx 27694{,}4$	$E_{00} \approx 774063{,}6$

Aplicando os valores na equação (4.6.10), obtém-se $x^2(t_k, c_i) \approx 284$.
Os termos t_k com os valores mais baixos de $x^2(t_k, c_i)$ são os mais independentes da categoria c_i, uma vez que na classificação de textos interessam os termos que **não são** independentes, seleccionam-se os termos com os valores mais elevados.

Uma fórmula equivalente à (4.6.10), mas aritmeticamente mais simples para calcular o x^2 é:

$$x^2(t_k, c_i) = \frac{(N_{11} + N_{10} + N_{01} + N_{00}) \times (N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11} + N_{01}) \times (N_{11} + N_{10}) \times (N_{10} + N_{00}) \times (N_{01} + N_{00})} \quad (4.6.11)$$

4.6.4 Frequência

Outra medida empregue na selecção de características, baseia-se na **frequência**. Esta medida pode ser calculada em termos de **frequência de documento** (*document frequency*) $DF(t_k)$, que corresponde ao número de documentos no qual o termo t_k ocorre [Yang e Pedersen 1997] ou como **frequência de termo** (*term frequency*) $TF(t_k)$, que corresponde ao número de vezes que o termo t_k ocorre. Esta medida requer que as *stopwords* sejam removidas antes de efectuado o cálculo [Mladenic e Grobelnik 1999]. Após efectuado o cálculo da frequência os termos com um valor inferior a um determinado limiar (*threshold*) predeterminado, são excluídos. A remoção é feita com base na assunção de que termos raros não são informativos para prever a categoria ou não influenciam o desempenho global, podendo mesmo melhorar a **taxa de acerto** (*accuracy*) se esses termos raros introduzem ruído [Yang e Pedersen 1997].

Nas experiências realizadas por *Yang* e *Pedersen* [Yang e Pedersen 1997], foi encontrada uma forte correlação entre as medidas: frequência de documento $DF(t_k)$, ganho de informação e x^2. Os autores concluem que esta correlação sugere que a frequência de documento que é o método mais simples e com o custo computacional mais reduzido, pode ser usado com confiança em vez do ganho da informação e x^2, quando o cálculo destas é demasiado dispendioso. *Dunja Mladenic* [D. Mladenic 1998b] com base nas experiências realizadas, observou e concordou com *Yang* e *Pedersen* [Yang e Pedersen 1997] de que o cálculo da frequência (usada após a remoção das *stopwords* e calculando tanto a frequência de documento como a frequência de termo) alcança bons resultados.

4.6.5 Outras medidas

Outras medidas, tem sido usadas na literatura para a atribuição de pesos aos termos, entre as quais a *DIA association factor* [Fuhr, et al. 1991], *NGL coefficient* [Ng, Goh e Low 1997], *relevancy score* [Wiener, Pedersen e Weigend 1995], *Odds Ratio* [D. Mladenic 1998a] e *GSS*

coefficient [Galavotti, Sebastiani e Simi 2000]. Todas estas medidas, são abordadas e resumidas em [F. Sebastiani 2002].

4.7 Considerações finais

De forma a treinar um classificador, é necessário partir de um conjunto de documentos já classificados, denominado de conjunto de treino e realizar um conjunto de tarefas de pré-processamento. As tarefas, segmentação do texto, normalização dos termos, remoção de *stop-words*, *stemming*, selecção de características ou redução de dimensão, foram discutidas no presente capítulo.

Capítulo 5.
Classificação de textos

Como já referido anteriormente, esta dissertação debruça-se sobre a **classificação multi-etiqueta hierárquica de texto**, onde cada documento pode pertencer a mais do que uma classe, ao contrário da classificação habitualmente encontrada na literatura, onde um documento apenas pode pertencer a uma única classe. Adicionalmente, as classes estão organizadas hierarquicamente, o que introduz um conjunto de especificidades, não encontradas quando a organização das classes é plana (não hierárquica).

Na secção *5.1* começa-se por apresentar definições essenciais para a compreensão das secções seguintes.

Na secção *5.2* (página *37*) são identificadas as características chave, que permitem distinguir os vários tipos de classificação (**classificação binária**, **multi-classe**, etc.). Sendo ainda apresentado, uma representação gráfica, que permite ver que tipo de classificação se obtêm combinando as características chave identificadas ou quais as características que um determinado tipo de classificação possui (Figura *14*, página *38*).

Nas secções *5.3* (página *39*), *5.4* (página *39*) e *5.5* (página *40*) são abordadas respectivamente a **classificação binária**, **classificação multi-classe** e a **classificação multi-etiqueta** em maior detalhe.

A secção *5.5* (página *40*), aborda diversos aspectos da **classificação multi-etiqueta**. É realizada uma comparação quantitativa, da dificuldade de prever correctamente a classe ou conjunto de classes na **classificação binária**, **multi-classe** e **multi-etiqueta** (Tabela *10*, página *40*). São também apresentadas as abordagens encontradas na literatura, para lidar com a classificação multi-etiqueta. São apresentadas as medidas **cardinalidade de etiqueta** e **densidade de etiqueta**, que tentam capturar o "quanto um problema é multi-etiqueta".

Nas secções *5.6* (página *45*) e *5.7* (página *45*) são abordadas respectivamente a **classificação plana** e **classificação hierárquica**, que dizem respeito à forma como o conjunto pré-definido de classes, está organizado.

A secção *5.7* (página *45*), aborda diversos aspectos da **classificação multi-hierárquica**. São identificadas as diferentes estruturas, utilizadas para organizar hierarquicamente as classes (estrutura árvore e grafo orientado acíclico), são identificados os tipos de relações mais comuns entre classes (**é-um**, **parte-de**), são apresentadas as diferentes abordagens para lidar com a classificação hierárquica.

Na secção *5.8* (página *50*), são apresentados algoritmos de aprendizagem, bem conhecidos na literatura, cujo entendimento se torna importante, porque juntamente com determinadas

abordagens podem ser usados para resolver problemas diferentes daqueles para os quais foram originalmente concebidos. A sua compreensão é também importante, caso se pretenda efectuar alterações no seu funcionamento para se resolver problemas diferentes daqueles para os quais foram concebidos.

5.1 Definições

Na classificação de textos, é dado:

- Um conjunto de classes $C = \{c_1, \dots, c_{|C|}\}$;
- Uma colecção de documentos $D = \{d_1, \dots, d_{|D|}\}$, cada documento contém um conjunto de termos e pertence a uma ou mais classes do conjunto C.

Definição 5.1.7 – Classificação de textos: A classificação de textos consiste na tarefa de atribuir um valor booleano a cada par $\langle d_i, c_j \rangle \in D \times C$ [F. Sebastiani 2002].

Um problema de classificação de textos pode ser distinguido pela cardinalidade do conjunto de classes pré-definidas (cardinalidade que é dada por $|C|$).

Definição 5.1.8 – Classificação binária de textos (*binary text classification*): A classificação de textos é chamada binária se $|C| = 2$.

Definição 5.1.9 – Classificação multi-classe de textos (*multi-class text classification*): A classificação de textos é chamada multi-classe se $|C| > 2$.

Um problema de classificação de textos pode ser distinguido pelo número de classes a que um documento pode pertencer.

Definição 5.1.10 – Classificação uni-etiqueta de textos (*single-label text classification*): A classificação de textos é chamada uni-etiqueta (*single-label*) se cada documento apenas pode pertencer a uma única classe [F. Sebastiani 2002].

Definição 5.1.11 – Classificação multi-etiqueta de textos (*multi-label text classification*): A classificação de textos é chamada multi-etiqueta se cada documento pode pertencer a mais do que uma classe, das $|C|$ existentes [Tikk e Biró 2003].

Um problema de classificação de textos pode ser distinguido pela forma como o conjunto de classes pré-definidas estão organizadas.

Definição 5.1.12 – Classificação plana de textos (*flat text classification*): A classificação de textos é chamada plana (não hierárquica) se as classes do conjunto C não estão organizadas de forma hierárquica.

Definição 5.1.13 – Classificação hierárquica de textos (*hierarchical text classification*): A classificação de textos é chamada hierárquica se as classes do conjunto C estão organizadas de forma hierárquica.

No trabalho de *Wang* e colegas [Wang, Zhou e He 2001], o termo **multi-classe** e no trabalho de *Hendrik Blockeel* e colegas [Blockeel, et al. 2002] o termo **multi-classificação**, são entendidos como a característica que permite que um documento possa pertencer a qualquer número de classes relevantes. Nesta dissertação, como em vários trabalhos encontrados na literatura, o termo **multi-classe** é usado para indicar que o conjunto de classes pré-definidas é superior a 2. E o facto de um documento poder pertencer a qualquer número de classes é chamado **multi-etiqueta**. Alguns trabalhos que vão ao encontro da posição adoptada nesta dissertação são[Schapire e Singer 2000], [Godbole e Sarawagi 2004], [Tsoumakas e Katakis 2007a], [Spyromitros, Tsoumakas e Vlahavas 2008].

5.2 Tipos de problemas de classificação

Após o estudo de vários trabalhos, sobre classificação, encontrados na literatura, constatou-se que existem vários tipos de classificação (classificação binária, classificação muli-classe, classificação multi-etiqueta, etc.) e que a diferentes tipos de classificação, correspondem diferentes características. Concluiu-se então, que as três características, que todos os tipos de classificação analisados, possuem, são:

a) Número de classes a que uma instância pode pertencer;
b) Número de classes existentes para classificação;
c) Organização das classes existentes para classificação.

Quanto à característica *a)*, têm-se que:

- Uma instância apenas pode pertencer a uma única classe;
- Uma instância pode pertencer a zero ou mais classes.

Relativamente à característica *b)*, pode-se ter:

- 2 (duas classes);
- > 2 (mais do que duas classes).

Finalmente a característica *c)*, poderá estar organizada de forma:

- Plana;
- Hierárquica.

De acordo com as características anteriores, o problema de **classificação multi-etiqueta hierárquico,** é caracterizado por:

- Uma instância poder pertencer a qualquer número de classes (característica *a*);
- O número de classes existentes para classificação ser superior a dois (característica *b)*);
- As classes existentes para classificação estão organizadas hierarquicamente (característica *c)*).

Considerando apenas o número de valores que as características *a)*, *b)* e *c)* podem tomar, teoricamente, podia-se ter oito (2^3) tipos de problemas de classificação, na prática existem apenas cinco problemas distintos, pois a combinação de determinados valores não fazem sentido!

Na figura seguinte, encontram-se as combinações possíveis referidas anteriormente, assim como a designação mais comum habitualmente encontrada na literatura.

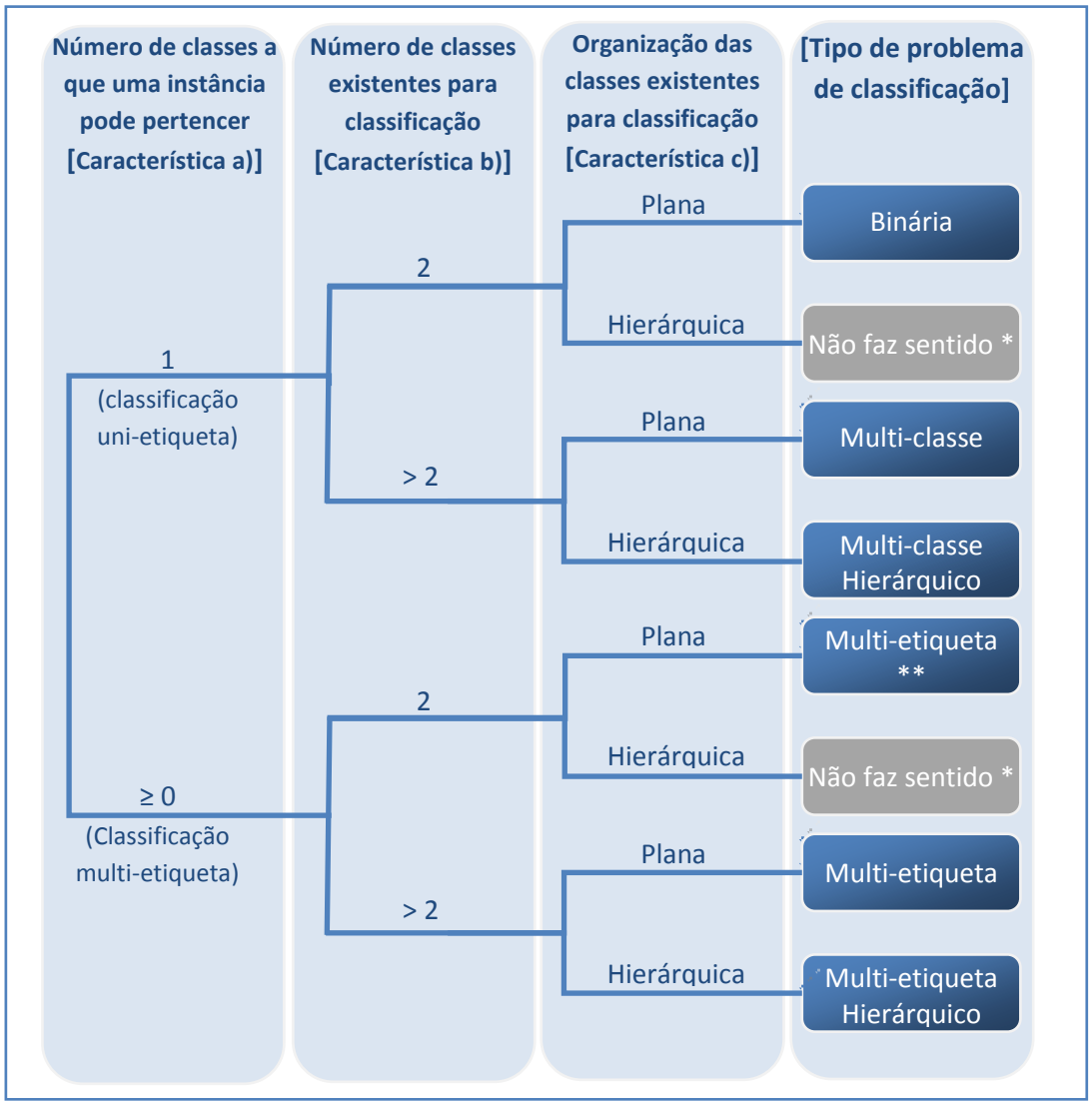


Figura 14 – Caracterização dos diferentes tipos de problemas de classificação

* Não faz sentido pelo facto do conjunto das classes existentes para classificação serem duas e estarem organizadas hierarquicamente.

** Teoricamente é possível que num problema, uma instância possa pertencer a qualquer número de classes, o número de classes existentes para classificação sejam duas e estas estejam organizadas de forma plana, contudo, na literatura, não foram encontrados quaisquer trabalhos com estas características.

Na figura anterior, pode-se observar, por exemplo, que num problema de **classificação binária**, cada instância apenas pertence a uma classe, o número de classes existentes para classificação são duas e estão organizadas de forma plana.

Em alguns trabalhos, apresentados na literatura, é possível encontrar-se a denominação **classificação multi-classe multi-etiqueta** ou **classificação multi-classe multi-etiqueta hierárquico**, em qualquer um dos casos a palavra multi-classe poderia ser suprimida, pois o

termo multi-etiqueta tem implícito que o problema é multi-classe, isto é, existem n classes (com $n > 2$) pré-definidas.

5.3 Classificação binária

Nos problemas de classificação do tipo binário ou simplesmente na **classificação binária**, uma instância apenas pode pertencer a uma classe das n existentes ($n = 2$).

Por exemplo, perante um conjunto de emails, classificar cada um deles como *spam* ou *não spam*.

A forma mais simples de classificação que podemos ter é a classificação binária, mas que para muitos domínios e problemas é o suficiente, por exemplo: distinguir o que é "sim" e o que é "não", ou entre "certo" e "errado", "verdadeiro" e "falso", "interessante" e "não interessante" [Cordeiro 2003].

As soluções empregues para solucionar problemas de **classificação binária** são importantes porque com as devidas alterações podem ser usados para solucionar outros problemas, como por exemplo, problemas **multi-classe** e **multi-etiqueta**. Um algoritmo de classificação binária poderia ser usado, por exemplo, num problema de classificação multi-classe, bastando transformar o problema multi-classe em problemas independentes de classificação binária. Imagine-se um problema multi-classe com as categorias $C = \{c_1, c_2, \dots, c_{|C|}\}$, seria necessário transformar em $|C|$ problemas independentes de classificação binária sob as categorias $\{c_i, \bar{c}_i\}$ com $i = 1, 2, \dots, |C|$ e posteriormente combinar os resultados dos diferentes problemas binários para produzir uma resposta ao problema multi-classe original.

De um ponto de vista probabilístico, uma vez que um documento pode pertencer a uma das duas classes, a probabilidade de classificar correctamente um documento, é de 50% $(1/2)$.

5.4 Classificação multi-classe

Na **classificação multi-classe**, uma instância apenas pode pertencer a uma classe das n existentes ($n > 2$) [Tikk e Biró 2003]. Outra designação é **um-de** (*one-of*) [Manning, Raghavan e Schütze 2007].

Por exemplo, perante um conjunto de noticias, classificar cada uma delas em apenas uma das categorias: desporto, politica, tecnologia.

Tanto a **classificação binária** como a **classificação multi-classe** são frequentemente denominados de **classificação uni-etiqueta** (*single-label clasification*) pois cada instancia apenas pode pertencer a uma única classe.

O que distingue a classificação binária da classificação multi-classe, é número de classes existentes para classificação. Na classificação binária são exactamente duas, na classificação multi-classe são n ($n > 2$).

De um ponto de vista probabilístico, uma vez que um documento pode pertencer a uma das n classes, a probabilidade de se classificar correctamente um documento, é de $1/n$ (com $n > 2$).

5.5 Classificação multi-etiqueta

Na **classificação multi-etiqueta**, uma instância pode pertencer a mais do que uma classe. Outras designações são: **qualquer um** (*any-of*), **classificação multi-valor** (*multivalue classification*) [Manning, Raghavan e Schütze 2007].

Por exemplo, perante um conjunto de notícias, classificar cada uma delas em uma ou mais das seguintes categorias: desporto, politica, tecnologia, Portugal, China. Um texto que aborde a realização dos jogos olímpicos na China, poderia ser classificado nas categorias: China e desporto.

O que distingue tanto a **classificação binária** como a **classificação multi-classe** da **classificação multi-etiqueta**, é o número de classes a que um documento pode pertencer. Nos dois primeiros tipos de classificação, um documento apenas pode pertencer a uma classe, na classificação multi-etiqueta pode pertencer a mais do que uma.

De um ponto de vista probabilístico, uma vez que um documento pode pertencer a um dos $2^n - 1$[22] subconjuntos de classes, que se podem formar com n classes, a probabilidade de se classificar correctamente um documento, é de $\frac{1}{2^n-1}$.
Na tabela e gráfico seguinte, é possível observar, a probabilidade de se classificar correctamente um documento, em função do número de classes pré-definidas e do tipo de problema de classificação (binária, multi-classe e multi-etiqueta).

Número Classes	Classificação Binária	Classificação Multi-classe	Classificação Multi-etiqueta
2	50,00%	----	----
3	----	33,33%	14,29%
4	----	25,00%	6,67%
5	----	20,00%	3,23%
6	----	16,67%	1,59%
7	----	14,29%	0,79%
8	----	12,50%	0,39%
9	----	11,11%	0,20%
10	----	10,00%	0,10%
20	----	5,00%	0,000095367522590%
30	----	3,33%	0,000000093132258%
40	----	2,50%	0,000000000090949%
50	----	2,00%	0,000000000000089%
100	----	1,00%	0,000000000000000000000000000079%

Tabela 10 – Probabilidade de se classificar correctamente um documento em função do número de classes, na classificação binária, multi-classe e multi-etiqueta

[22] $2^n - 1 = C_1^n + C_2^n + \ldots + C_p^n$. Em que $C_p^n = \frac{n!}{p!(n-p)!}, n, p \in N_0 e\ n \geq p$

Na figura seguinte, é possível observar os dados da tabela anterior de forma gráfica.

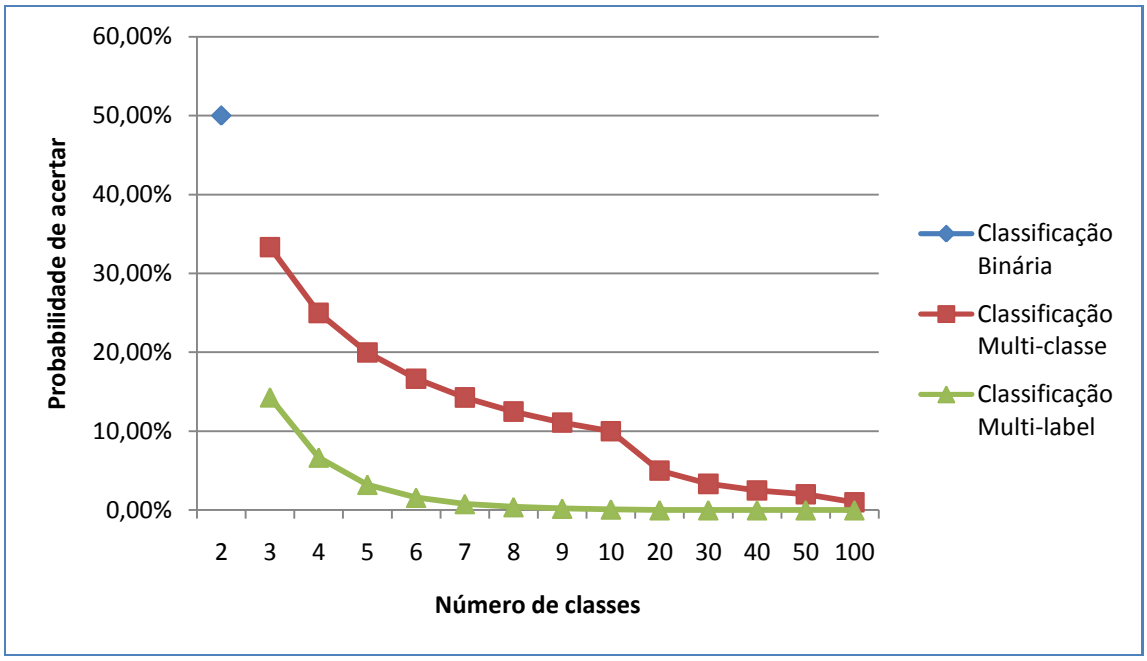


Figura 15 – Probabilidade de se classificar correctamente um documento em função do número de classes, na classificação binária, multi-classe e multi-etiqueta

5.5.1 Métodos para lidar com Problemas Multi-Etiqueta

Na literatura encontra-se de forma dispersa vários métodos para lidar com os problemas de classificação multi-etiqueta, estes métodos podem ser classificados em dois grupos [Tsoumakas e Katakis 2007a]:

- Métodos de **transformação do problema** (*problem transformation*);
- Métodos de **adaptação do algoritmo** (*algorithm adaptation*).

Os métodos do primeiro grupo caracterizam-se por transformarem ou decomporem o problema multi-etiqueta em um ou vários problemas de classificação uni-etiqueta e por serem independentes dos algoritmos de classificação.

Os métodos do segundo grupo caracterizam-se por estenderem os algoritmos de classificação existentes de modo a lidarem com a classificação multi-etiqueta. Existem extensões de árvores de decisão [Clare e King 2001], de Máquinas de Vectores de Suporte [Elisseeff e Weston 2002] [Godbole e Sarawagi 2004], de redes neuronais [Crammer e Singer 2003] [Zhang e Zhou 2006] e do método k-vizinhos mais próximos [Zhang e Zhou 2005].

5.5.1.1 Métodos de transformação do problema

Para exemplificar o funcionamento destes métodos considere-se a tabela seguinte que ilustra um problema multi-etiqueta. Cada instância (documento) possui uma ou mais categorias.

Doc.	Cat. 1	Cat. 2	Cat. 3	Cat. 4
1	X			X
2			X	X
3	X			
4		X	X	

Tabela 11 – Conjunto de documentos multi-etiqueta

Um método para transformar o problema multi-etiqueta num problema uni-etiqueta, consiste em aleatoriamente ou, seguindo um outro critério, escolher uma das etiquetas que a instancia possui e descartar as restantes. A tabela seguinte ilustra esta transformação, que será denominada TP1.

Doc.	Cat. 1	Cat. 2	Cat. 3	Cat. 4
1	X			
2			X	
3	X			
4		X		

Tabela 12 – Transformação do conjunto de documentos usando TP1

Este método tem como desvantagem, a perda de informação nas instâncias multi-etiqueta.
De notar que esta transformação poderá ter interesse em alguns cenários. Imagine-se um problema em que cada documento é classificado em uma categoria principal e n $(n \geq 0)$ secundárias. Se o objectivo fosse classificar novos documentos em relação à sua categoria principal, a remoção das categorias secundárias não representaria qualquer perda de informação.

Um segundo método para transformar o problema multi-etiqueta num problema uni-etiqueta, consiste em descartar as instâncias que possuam mais do que uma etiqueta. A tabela seguinte ilustra esta transformação, que será denominada TP2.

Doc.	Cat. 1	Cat. 2	Cat. 3	Cat. 4
3	X			

Tabela 13 – Transformação do conjunto de documentos usando TP2

Este método tem como desvantagem, a perda de informação resultante da não consideração das instâncias multi-etiqueta.

Um terceiro método para transformar o problema multi-etiqueta num problema uni-etiqueta, consiste em para cada conjunto diferente de categorias existentes no conjunto de documentos tratá-lo como uma etiqueta única. A tabela seguinte ilustra esta transformação, que será denominada TP3.

Doc.	Cat. 1 & Cat. 4	Cat. 3 & Cat. 4	Cat. 1	Cat. 2 & Cat. 3
1	X			
2		X		
3			X	
4				X

Tabela 14 – Transformação do conjunto de documentos usando TP3

Este método tem como desvantagem, poder conduzir a um largo número de classes (resultante da combinação das classes) com um número reduzido de exemplos por classe. Este método foi usado por exemplo em [Boutell, et al. 2004], [Diplaris, et al. 2005].

Um quarto método para transformar o problema consiste em para cada etiqueta existente no conjunto de etiquetas, aprender um classificador binário, que é responsável por indicar se uma dada instância possui ou não essa etiqueta.
A classificação de uma dada instância é obtida pela união das etiquetas indicadas pelos classificadores binários anteriormente aprendidos. A tabela seguinte ilustra esta transformação, que será denominada TP4.

Doc.	Cat. 1	¬Cat. 1
1	X	
2		X
3	X	
4		X

Doc.	Cat. 2	¬Cat. 2
1		X
2		X
3		X
4	X	

Doc.	Cat. 3	¬Cat. 3
1		X
2	X	
3		X
4	X	

Doc.	Cat. 4	¬Cat. 4
1	X	
2	X	
3		X
4		X

Tabela 15 – Transformação do conjunto de documentos usando TP4

Este método foi usado por exemplo em [Boutell, et al. 2004], [Goncalves e Quaresma 2003], [Diplaris, et al. 2005], [Lauser e Hotho 2003] e [Li e Ogihara 2003].

5.5.1.2 Métodos de adaptação do algoritmo

McCallum [A. McCallum 1999] define um modelo probabilístico no qual cada etiqueta possui associado um conjunto de palavras.

Scaphire e *Singer* apresentam o algoritmo *AdaBoost.MH* e *AdaBoost.MR* [Schapire e Singer 2000], duas extensões do algoritmo *AdaBoost*[23] [Freund e Schapire 1996], [Freund e Schapire 1997], para suportar classificação multi-etiqueta. Por sua vez o algoritmo *AdaBoost.MHKR* [Sebastiani, Sperduti e Valdambrini 2000] é uma extensão do algoritmo *AdaBoost.MH* e suporta também problemas multi-etiqueta.

Clare e *King* [Clare e King 2001] adaptaram o algoritmo *C4.5* [J. R. Quinlan 1993] usado na criação de árvores de decisão de forma a lidar com problemas multi-etiqueta.

Elisseeff e *Weston* [Elisseeff e Weston 2002] apresentam um algoritmo multi-etiqueta capaz de atribuir uma classificação às etiquetas. O algoritmo segue a filosofia das *Support Vector Machines* (SVMs), é um modelo linear que tenta minimizar uma função custo enquanto tenta manter uma grande margem.

[23] Algoritmo de *boosting*, isto é, um algoritmo para combinar vários algoritmos de classificação

Godbole e *Sarawagi* (2004) apresentam duas melhorias para as SVMs em conjunto com a transformação TP4 para problemas multi-etiqueta.
A primeira melhoria poderia ser resumida facilmente como sendo uma extensão da transformação TP4 para ser usada com qualquer algoritmo de classificação. A ideia principal é estender o conjunto de dados original com |L| características extras, uma para cada etiqueta com a previsão do classificador binário. Numa segunda volta são aplicados os classificadores binários sobre o conjunto de dados estendido. Através desta extensão a abordagem leva em consideração a potencial dependência entre as diferentes etiquetas. Esta melhoria é de facto um caso especializado da aplicação de *Stacking*[24] [Wolpert 1992] em cima da transformação TP4.
A segunda melhoria prende-se com a afinação da margem das SVMs para obter uma melhor classificação multi-etiqueta. A afinação da margem é feita pela remoção das instâncias de treino negativas muito similares que se encontram dentro de uma distância limiar do hiper-plano aprendido, e pela remoção das instâncias de treino negativas de uma classe completa se esta é muito semelhante à classe positiva. A afinação da margem é de facto independente da SVM, como tal pode ser usada como uma extensão da transformação TP4.

Thabtah, *Cowling* e *Peng* [Thabtah, Cowling e Peng 2004] apresentam uma abordagem chamada MMAC (*Multi-class*, *Multi-label Associative Classification).* Nesta abordagem são aprendidas um conjunto de regras de classificação recorrendo a regras de associação. As instâncias de treino cobertas pelas regras de associação são removidas e recursivamente são aprendidas novas regras sem as instâncias que restam, até que não existam mais instâncias. Os vários conjuntos de regras podem ter regras com pré-condições semelhantes mas diferentes etiquetas, estas regras são fundidas numa única regra multi-etiqueta. As etiquetas são classificadas e ordenadas de acordo com o seu suporte.

Zhang e *Zhou* [Zhang e Zhou 2005] apresentam uma adaptação do algoritmo kNN denominado ML-kNN de forma a suportar problemas multi-etiqueta. Na realidade este método segue o paradigma do método de transformação do problema T4. A indicação se uma dada instância possui ou não uma determinada etiqueta é feita aplicando o algoritmo kNN. A principal diferença entre a aplicação deste algoritmo e a aplicação do kNN original com a transformação TP4, é o uso de probabilidades a priori. O algoritmo ML-kNN tem também a capacidade de produzir uma lista ordenada das etiquetas mais prováveis.

5.5.2 Cardinalidade e densidade de etiqueta

Diferentes conjuntos de dados multi-etiqueta podem apresentar diferentes características no que respeita ao número de etiquetas que cada instância possui. Este facto pode influenciar o desempenho dos diferentes algoritmos muti-etiqueta. *Grigorios Tsoumakas* e *Ioannis Katakis* [Tsoumakas e Katakis 2007a] apresentam os conceitos **cardinalidade de etiqueta** (*label cardinality - LC*) e **densidade de etiqueta** (*label density - LD*) de um conjunto de dados.

[24] Método para combinar vários classificadores

Seja D um conjunto de dados multi-etiqueta, com $|D|$ instâncias (x_i, Y_i), $i = 1, 2, \ldots, |D|$. Seja $|L|$ o número de etiquetas pré-definidas (existentes para classificação).

Definição 5.5.14 – Cardinalidade de etiqueta: A cardinalidade de etiqueta do conjunto D, é o número médio de etiquetas de cada instância.

$$LC(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} |Y_i| \tag{5.5.12}$$

Definição 5.5.15 – Densidade de etiqueta: A densidade de etiqueta do conjunto D, é o número médio de etiquetas de cada instancia, dividido por $|L|$.

$$LD(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i|}{|L|} \tag{5.5.13}$$

Dois conjuntos de dados com a mesma cardinalidade da etiqueta mas com uma grande diferença de densidade da etiqueta, podem não exibir as mesmas propriedades e causarem diferentes comportamentos dos métodos de classificação multi-etiqueta [Tsoumakas e Katakis 2007a]. As duas medidas relacionam-se $LC(D) = |L|\, LD(D)$.

5.6 Classificação plana

Na **classificação plana** (*flat classification*), as classes pré-definidas para classificação estão todas ao mesmo nível, não existindo qualquer relação hierárquica entre elas. Na figura seguinte, onde os nós, representam classes, é possível observar este género de organização.

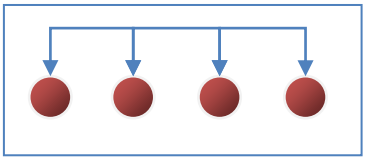


Figura 16 – Organização plana das classes

Na classificação plana é suposto as classes serem independentes. Na prática, esta independência nem sempre é conseguida. Por exemplo, num problema de classificação de textos, em que entre outras classes, existam a classe **Musica** e classe **Entretenimento**, é muito provável que um documento que pertença à classe **Musica** possa pertencer à classe **Entretenimento**. O não respeito da independência das classes, leva à deterioração das previsões, nessas classes.

5.7 Classificação hierárquica

Na **classificação hierárquica** (*hierarchical classification*), as categorias para classificação estão organizadas hierarquicamente. Aspectos importantes, que permitem distinguir o tipo de problema de classificação hierárquico são:

- A estrutura hierárquica utilizada para representar as relações entre as categorias (árvores, grafos acíclicos) e em que nível ou tipo de categorias (categorias folha ou categorias intermédias) são permitidas atribuir aos documentos. Na secção *5.7.1*, são apresentadas as estruturas habitualmente utilizadas e restrições que podem ser impostas;

- O tipo de relação existente entre as classes. Na secção *5.7.2* são apresentados os tipos de relações mais comuns.

São várias as colecções de texto que estão organizadas em hierárquicas: pastas de correio electrónico, repositórios *Web*, bibliotecas digitais (a biblioteca digital ACM é um exemplo), entre muitas outras. Até meados dos anos 95, na maioria das vezes a estrutura hierárquica das categorias era ignorada ao transformarem a hierarquia num conjunto plano de categorias [Kiritchenko, Matwin, et al. 2006]. O trabalho [Koller e Sahami 1997] é apontado em [Kiritchenko, Matwin, et al. 2006] como o primeiro estudo adequado do problema de classificação hierárquica de textos. E foi investigado posteriormente, por exemplo, nos trabalhos [Koller e Sahami 1997], [McCallum, Rosenfeld, et al. 1998], [D. Mladenic 1998c], [Wang, Zhou e Liew 1999], [Dumais e Chen. 2000], [Sun e Lim 2001], [Ruiz e Srinivasan 2002], [Granitzer 2003], [Hofmann, Cai e Ciaramita 2003], [Dekel, Keshet e Singer 2004].

5.7.1 Estruturas para organização hierárquica das categorias

Na classificação hierárquica, a organização é feita recorrendo a uma das seguintes estruturas: **árvores** (*Trees*) ou **grafos orientados acíclicos** - *GOAs* (*Directed Acyclic Graph - DAG*).

Definição 5.7.16 – Grafo: Um grafo G, é um par de (V, E), onde V é um conjunto de vértices, também chamados nós e, E é um conjunto de ramos entre os vértices.

Definição 5.7.17 – Grafo orientado acíclico (GOA): Um Grafo orientado acíclico, é um grafo onde os ramos têm uma direcção e não existem ciclos (partindo de um determinado nó não se consegue regressar a ele).

Definição 5.7.18 – Árvore: Uma árvore, é um caso especial de grafo orientado acíclico, onde os nós apenas podem ter um nó pai (à excepção do nó raiz que não possui nó pai).

Em cada uma das estruturas (GOA e árvore), pode ser imposto que os documentos apenas possam pertencer às categorias folha ou pode ser permitido que os documentos possam pertencer às categorias internas e folhas. Considerando que cada uma das duas estruturas, pode possuir uma destas duas variações, num problema de classificação hierárquica podemos encontrar uma de quatro estruturas. *Sun* e *Lim* [Sun e Lim 2001] identificam essas estruturas hierárquicas e dão-lhes a seguinte denominação:

- **Árvore de categoria virtual** (*virtual category tree*): estrutura em que as categorias estão organizadas numa árvore e os documentos apenas podem ser atribuídos às categorias folha [Dumais e Chen. 2000].
- **Árvore de categoria** (*category tree*): é uma extensão da árvore de categoria virtual que permite que os documentos sejam atribuídos às categorias internas e folhas [Wang, Zhou e He 2001].
- **Grafo orientado acíclico de categoria virtual** (*virtual directed acyclic category graph*): estrutura em que as categorias estão organizadas num grafo orientado acíclico e os documentos apenas podem ser atribuídos às categorias folha.
- **Grafo orientado acíclico de categoria** (*virtual directed acyclic category graph*): é uma extensão do grafo orientado acíclico de categoria virtual que permite que os documentos

sejam atribuídos às categorias internas e folhas. Esta é talvez a estrutura mais usada nos populares directórios *Web* como o *Yahoo* [Directório-Yahoo 2008], *Open Directory Project* [Directório-Dmoz 2008] e *Google* [Directório-Google 2008].

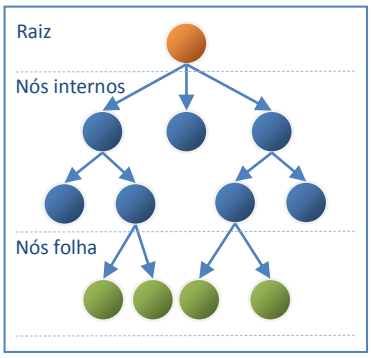


Figura 17 – Organização hierárquica em árvore

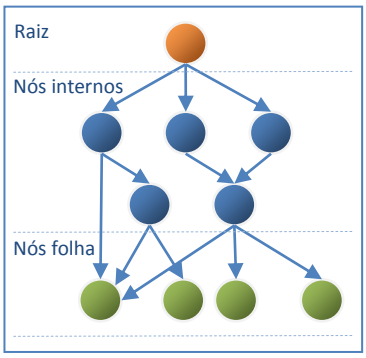


Figura 18 – Organização hierárquica em grafo orientado acíclico

Neste projecto, as categorias definidas na árvore ACM, estão organizadas numa estrutura hierárquica em árvore, podendo atribuir-se aos documentos, apenas categorias folha (nós folha). Segundo a classificação proposta em [Sun e Lim 2001] e apresentada em cima, a estrutura de categorias seria chamada **árvore de categoria virtual**.

5.7.2 Relações hierárquicas entre as categorias

Existem vários tipos relações entre entidades, por exemplo, **é-um** (*is-a*), **parte-de** (*part-of*), **feito-de** (*made-of*), **causa-para** (*cause-to*), entre outros. No entanto, nas hierarquias de categorias, habitualmente apenas duas são encontradas: **é-um** e **parte-de**.

É-um é uma relação de especificação. Por exemplo, um automóvel **é-um** veiculo. A relação do tipo **é-um**, é assimétrica (um automóvel é um tipo de veiculo, no entanto um veiculo não é apenas um automóvel) e transitiva (um automóvel é um tipo de veiculo, um veiculo é um meio de transporte, portanto, automóvel também é um meio de transporte).

Parte-de é uma relação partitiva. Por exemplo, um motor é **parte-de** um automóvel, deste modo motor é uma subcategoria de automóvel.

Numa hierarquia, em que a relação é do tipo **é-um** ou do tipo **parte-de**, se um objecto pertence a uma categoria também pertence a todas as suas categorias ascendentes. Esta restrição, introduz a noção de **consistência hierárquica** (*hierarchical consistency*). Qualquer sistema ou algoritmo de classificação hierárquica, precisa de efectuar previsões consistentes com a hierarquia.

5.7.3 Métodos para lidar com Problemas Hierárquicos

Na literatura encontram-se diferentes métodos para lidar com os problemas de classificação hierárquico, estes métodos podem ser agrupados em dois grupos:

- Métodos de **transformação do problema**;
- Métodos **hierárquicos**.

Os métodos do primeiro grupo caracterizam-se por transformarem ou decomporem o problema hierárquico em um problema plano (não hierárquico) (secção *5.7.3.1*, página *48*).

Os métodos do segundo grupo caracterizam-se por lidarem com a hierarquia e portanto preservarem a relação existente entre as classes. Existem basicamente dois métodos ou abordagens hierárquicas, a abordagem **hierárquica local** (ou *top-down level-based*), que será apresentada na secção *5.7.3.2* (página *49*) e a abordagem **hierárquica global** (ou *big-bang*), que será apresentada na secção *5.7.3.3* (página *49*)[Sun e Lim 2001].

Na figura abaixo apresenta-se uma hierarquia fictícia de categorias, que será usada para expor as estratégias que são seguidas na classificação hierárquica.

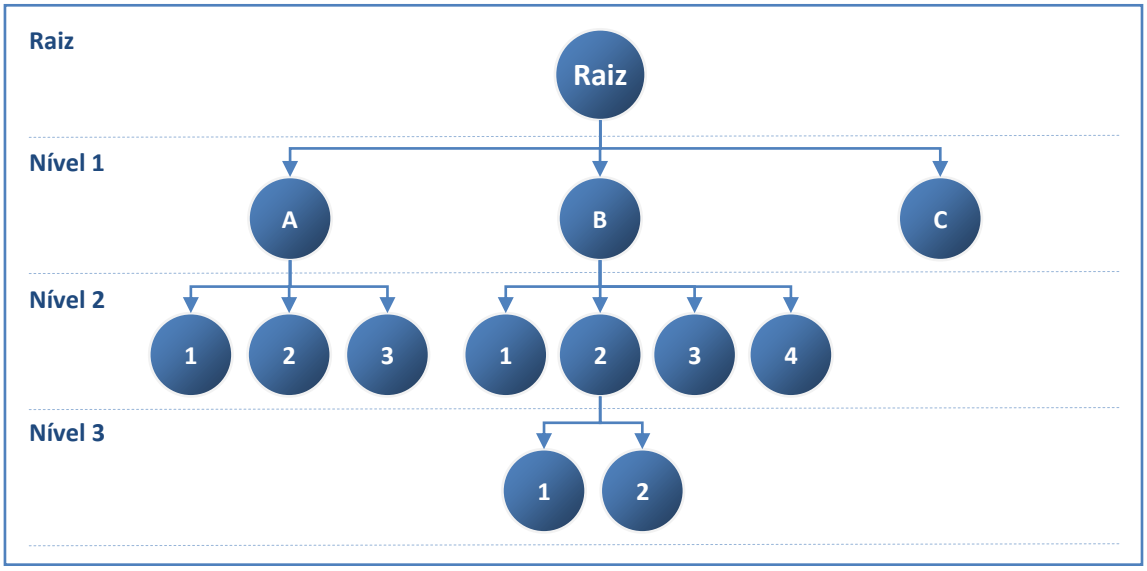


Figura 19 – Organização hierárquica em árvore das categorias

5.7.3.1 Métodos de transformação do problema

Um método para transformar o problema de classificação hierárquica, consiste em transformar a estrutura hierárquica das categorias numa estrutura plana e desta forma ser tratado como um problema de classificação plana, para o qual existem soluções há muito tempo conhecidas e com bons resultados demonstrados experimentalmente, por exemplo, nos trabalhos de [Yang 1999] e [Joachims 1998].

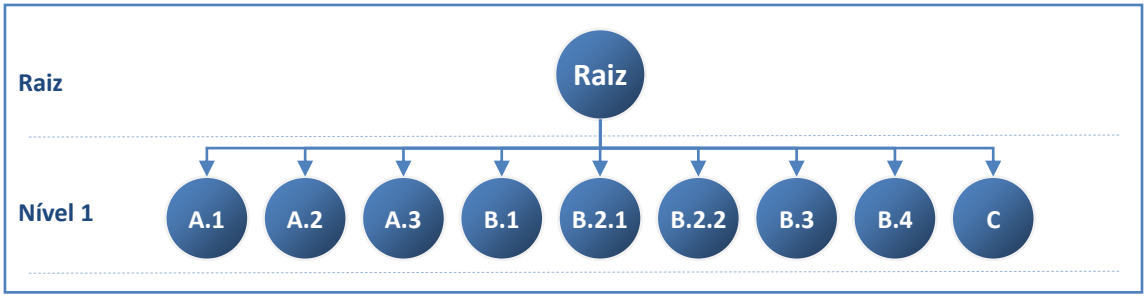


Figura 20 – Problema hierárquico transformado num problema plano

Esta abordagem não faz uso da hierarquia das categorias, originando a perda desse conhecimento. Apesar dos resultados em [Koller e Sahami 1997] terem sido bastante similares quando lidado com o problema de classificação de forma hierárquica e de forma plana, é opinião do autor desta dissertação que isto só tenha sido possível devido ao número reduzido de categorias envolvidas no problema. Os resultados relatados em [Kiritchenko, Matwin, et al.

2006] cujo número de categorias é superior, são favoráveis à utilização da hierarquia entre as categorias.

5.7.3.2 Método hierárquico - Abordagem hierárquica local

Na abordagem **hierárquica local** (ou *top-down level-based*)[Sun e Lim 2001], o algoritmo aplicado começa por construir um ou vários classificadores para as categorias do primeiro nível da hierarquia. O objectivo consiste em resolver este primeiro problema plano. O problema é resolvido construindo um ou mais classificadores, cuja missão é indicar a categoria relevante (no caso de um problema uni-etiqueta) ou categorias relevantes (no caso de um problema multi-etiqueta). Se nenhuma das categorias for relevante o processo de classificação termina, caso contrário, o próximo passo é repetir para todas as categorias indicadas como relevantes os procedimentos anteriores nos níveis seguintes da hierarquia. O processo de classificação termina quando se atingir uma categoria folha ou uma categoria interna, de acordo com as restrições do problema. Esta abordagem parece ser uma escolha natural uma vez que reflecte o modo como os humanos habitualmente desempenham a tarefa de classificação.

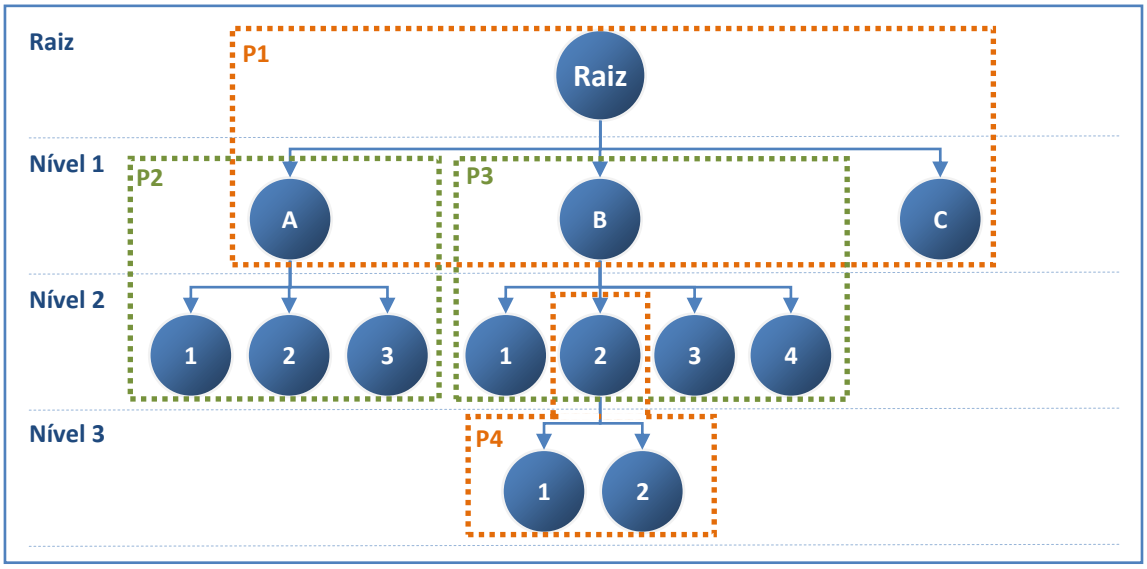


Figura 21 – Abordagem hierárquica local

A classificação **hierárquica local**, tem sido alcançada com diversos classificadores *Baysianos* em [Koller e Sahami 1997], com classificadores de Máquinas de Vectores de Suporte em [Dumais e Chen. 2000] e [Sun e Lim 2001], árvores de decisão em [Blockeel, et al. 2002] e redes neuronais em [Ruiz e Srinivasan 2002].

Esta abordagem é apontada como computacionalmente mais eficiente do que a abordagem global (próxima secção), tendo como fraqueza a **propagação do erro** (*error propagation*), isto é a escolha errada de uma categoria, faz com que o erro seja propagado por todos os seus nós descendentes.

5.7.3.3 Método hierárquico - Abordagem hierárquica global

Na abordagem **hierárquica global** (ou *big-bang*)[Sun e Lim 2001], a hierarquia de classes é tratada como um todo, sendo construído um único classificador responsável por descriminar todas as classes [Kiritchenko, Matwin, et al. 2006]. É semelhante à abordagem de transformação do problema classificação hierárquico num problema plano, só que de alguma

forma tem em consideração o relacionamento entre as classes da hierarquia. Por esta razão, a utilização de métodos de classificação plana, na sua forma original, não é possível, devendo-se realizar modificações para que as relações entre as classes sejam consideradas, caso se pretenda utiliza-los.

A construção de um classificador, usando a abordagem global é apontada como mais complexa do que seguindo a abordagem hierárquica local, é apontada também como computacionalmente pesada e não flexível, por exemplo, cada vez que há uma alteração da hierarquia de classes, o classificador precisa de ser treinado novamente.

Esta abordagem foi seguida, com um classificador baseado em regras [Sasaki e Kita 1998], [Sasaki e Kita 1998], classificador probabilístico [McCallum, Rosenfeld, et al. 1998], com um classificador estilo *Rocchio* em [Labrou e Finin 1999], com regras de associação em [Wang, Zhou e Liew 1999], [Wang, Zhou e He 2001], com versões estendidas do classificador *Naive Bayes* padrão [Toutanova, et al. 2001], com Máquinas de Vectores de Suporte [Tsochantaridis, et al. 2004] entre outros tipos de classificadores em [Gaussier, et al. 2002] e [Vinokourov e Girolami 2002].

5.8 Algoritmos de aprendizagem

5.8.1 Naive Bayes

São vários os trabalhos de investigação que examinam a teoria e desempenho da aplicação do classificador *Naive Bayes* na classificação de textos, nomeadamente [Yang e Liu 1999] e [F. Sebastiani 2002]. O classificador *Naive Bayes* é um classificador probabilístico que se baseia na aplicação do teorema de *Bayes* da probabilidade condicional, exibida na equação seguinte:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \tag{5.8.14}$$

sejam x e y dois acontecimentos, $P(x|y)$ é a **probabilidade condicional** de x dado y, também chamada **probabilidade à posterior** porque deriva ou depende do valor especificado y. $P(y|x)$ é a probabilidade condicional de y dado x. $P(x)$ é a probabilidade à priori de x, é à priori no sentido em que não leva em conta qualquer informação de y. $P(y)$ é a probabilidade à priori de y.

Se x for uma classe (categoria) c_j e y um documento d_i, em vez de $P(x|y)$ tem-se $P(c_j|d_i)$ que representa a probabilidade de um dado documento d_i pertencer à classe c_i. Na classificação de texto o objectivo é encontrar a "melhor" classe para o documento. Na classificação *Naive Bayes* a melhor classe é a mais provável, conhecida por **máxima probabilidade à posteriori** (*MAP*) c_{MAP} dada por $ArgMax_{c_j \in C} P(c_j|d_i)$. Esta probabilidade habitualmente é impossível de calcular directamente porque o documento d_i provavelmente nunca foi visto antes. Então, pode ser aplicado o teorema de *Bayes* para alterar a expressão para uma cujos constituintes possam ser estimados a partir dos dados de treino, nesse sentido e usando a equação (5.8.14) obtém-se:

$$c_{MAP} = ArgMax_{c_j \in C} P(c_j|d_i) = ArgMax_{c_j \in C} \frac{P(d_i|c_j)P(c_j)}{P(d_i)} \tag{5.8.15}$$

Como $P(d_i)$ é constante (é a mesma para todas as classes) pode ser eliminada, ficando:

$$= ArgMax_{c_j \in C} P(d_i|c_j)P(c_j) \quad (5.8.16)$$

onde $P(c_j)$ é a probabilidade à priori da classe c_j e pode ser estimada a partir do conjunto de treino. $P(d_i|c_j)$ é a probabilidade condicional do documento d_i dada a classe c_j e pode ser estimada considerando que o documento d_i é equivalente aos termos $t_1, t_2, \dots, t_{|T|}$ que o compõem. Assim $ArgMax_{c_j \in C} P(c_j|d_i)$ pode ser estimado como:

$$ArgMax_{c_j \in C} P(c_j|d_i) = ArgMax_{c_j \in C} P(t_1, t_2, \dots, t_{|T|}|c_j)P(c_j) \quad (5.8.17)$$

$$\approx ArgMax_{c_j \in C} P(t_1|c_j) \cdot P(t_2|c_j) \cdot \dots \cdot P(t_{|T|}|c_j)P(c_j) \quad (5.8.18)$$

onde $P(t_k|c_j)$ é a probabilidade condicional do termo t_k dada a categoria c_j, $P(c_j)$ tem o mesmo significado das equações anteriores. A passagem de $P(t_1, t_2, \dots, t_{|T|}|c_j)$ para $P(t_1|c_j) \cdot P(t_2|c_j) \cdot \dots \cdot P(t_{|T|}|c_j)$, é o que deu origem ao nome "naive" (ingénuo) por causa das assunções que faz: primeiro, é feita a assunção de **independência condicional** ao assumir-se que os termos $t_1, t_2, \dots, t_{|T|}$ são independentes uns dos outros, dada a classe c_j, segundo, é feita a assunção de **independência posicional** ao assumir que a posição dos termos no documento d_i não tem efeitos na probabilidade. De uma forma geral, estas assunções quase nunca são verdadeiras. Apesar disto, o classificador *Naive Bayes* tende a produzir resultados bastante bons, uma análise deste fenómeno pode ser encontrada em [Domingos e Pazzani 1997]. Mais recente, *Harry Zhang* fez uma análise cuidadosa e mostrou que há algumas razões teóricas para a eficácia aparentemente injusta de classificadores *Naive Bayes* [H. Zhang 2004].

A equação final que permite calcular a classe mais provável de um documento é a (5.8.18), que por comodidade, por ser reescrita da seguinte forma:

$$c_{MAP} = ArgMax_{c_j \in C} \hat{P}(c_j|d_i) \approx ArgMax_{c_j \in C} \hat{P}(c_j) \prod_{k=1}^{|T|} \hat{P}(t_k|c_j) \quad (5.8.19)$$

em vez de P é escrito $\hat{P}$ porque os verdadeiros valores dos parâmetros $P(c_j)$ e $P(t_k|c_j)$ não são conhecidos, mas sim estimados a partir do conjunto de treino.

$\hat{P}(c_j)$ é a estimativa da probabilidade à priori de um documento pertencer à classe c_j, encontrada por:

$$\hat{P}(c_j) = \frac{N_{c_j}}{N} \quad (5.8.20)$$

onde N_{c_j} é o número de documentos da classe c_j e N o número total de documentos, em ambos os casos do conjunto de treino.

$\hat{P}(t_k|c_j)$ é a estimativa da probabilidade condicional de um termo t_k ser encontrado nos documentos da classe c_j, encontrada por:

$$\hat{P}(t_k|c_j) = \frac{T_{t_k c_j}}{T_{t' c_j}} \quad (5.8.21)$$

onde $T_{t_k c_j}$ é o número de vezes que o termo t_k ocorre nos documentos de treino pertencentes à classe c_j. $T_{t' c_j}$ é o número total de termos que ocorrem em todos os documentos de treino pertencentes à classe c_j.

Na equação (5.8.21), se o termo t_k não existe nos documentos de treino, então $T_{t_k c_j}$ será zero assim como o resultado da equação. Visto a equação (5.8.19) utilizar a (5.8.21) a multiplicar, o resultado será zero. Este problema é resolvido aplicando-se a **correcção de *Laplace***, que simplesmente adiciona um ao numerador e B ao denominador à equação (5.8.21):

$$\hat{P}(t_k|c_j) = \frac{T_{t_k c_j}}{T_{t' c_j}} = \frac{T_{t_k c_j} + 1}{T_{t' c_j} + B} \tag{5.8.22}$$

onde B é o número total de termos diferentes em todo o conjunto de treino (independente da classe).

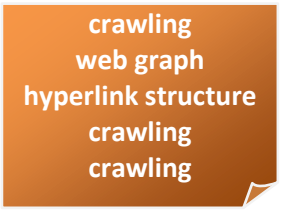
Na equação (5.8.19), são multiplicadas várias probabilidades condicionais, uma por cada termo distinto existente no documento d_i, o que pode originar erro de vírgula flutuante (o resultado da multiplicação de sucessivos números entre 0 e 1 tende para zero, podendo atingir um valor demasiado pequeno). Por esta razão é melhor efectuar o cálculo adicionando logaritmos de probabilidades em vez de multiplicar probabilidades. A classe com o valor do logaritmo de probabilidade mais elevado continua a ser a mais provável uma vez que $\log(xy) = \log(x) + \log(y)$ e a função logaritma é monotónica. Por conseguinte, a maximização que é utilizada na maioria das implementações do *Naive Bayes* é:

$$c_{MAP} = ArgMax_{c_j \in C} \hat{P}(c_j|d_i) \approx ArgMax_{c_j \in C} [\log \hat{P}(c_j) \log \sum_{k=1}^{|T|} \log \hat{P}(t_k|c_j)] \tag{5.8.23}$$

Por exemplo, dado um conjunto de treino com três documentos pertencentes à classe H.3.3 e um não pertencente, deseja-se classificar um novo documento.

Conjunto de treino

Classe = H.3.3 | Classe = H.3.3 | Classe = H.3.3 | Classe = ¬H.3.3



Classe =?

Resolução:

Para determinar a classe a atribuir ao novo documento é necessário determinar a classe mais provável, que é dada pela equação (5.8.19) ou (5.8.23). Será usada a equação (5.8.19), reescrita abaixo por comunidade:

$$c_{MAP} = ArgMax_{c_j \in C} \hat{P}(c_j|d_i) \approx ArgMax_{c_j \in C} \hat{P}(c_j) \prod_{k=1}^{|T|} \hat{P}(t_k|c_j)$$

na prática consiste em calcular $\hat{P}(c_j|d_i) = \hat{P}(c_j) \prod_{k=1}^{|T|} \hat{P}(t_k|c_j)$, isto é, para o documento a classificar e para cada classe c_j (neste exemplo $H.3.3$ e $\neg H.3.3$) calcular $\hat{P}(c_j|d_i)$ (neste exemplo calcular $\hat{P}(\text{H.3.3}|\text{doc})$ e $\hat{P}(\neg\text{H.3.3}|\text{doc})$). A classe que obtiver o resultado mais elevado é a classe a atribuir ao documento. Para determinar $\hat{P}(\text{H.3.3}|\text{doc})$ e $\hat{P}(\neg\text{H.3.3}|\text{doc})$ calcula-se:

- $\hat{P}(c_j)$, isto é, a probabilidade à priori de cada classe c_j (equação (5.8.20)):

  $\hat{P}(H.3.3) = \frac{3}{4}$

  $\hat{P}(\neg H.3.3) = \frac{1}{4}$

- $\hat{P}(t_k|c_j)$, isto é, a probabilidade de uma dada classe c_j conter o termo t_k existentes no documento a classificar (equação (5.8.22)):

  $\hat{P}(\text{crawling}|\text{H.3.3}) = \frac{5+1}{12+10} = \frac{6}{22} = \frac{3}{11}$

  $\hat{P}(\text{web graph}|\text{H.3.3}) = \hat{P}(\text{hyperlink structure}|\text{H.3.3}) = \frac{0+1}{12+10} = \frac{1}{22}$

  $\hat{P}(\text{crawling}|\neg\text{H.3.3}) = \frac{1+1}{3+10} = \frac{2}{13}$

  $\hat{P}(\text{web graph}|\neg\text{H.3.3}) = \hat{P}(\text{hyperlink structure}|\neg\text{H.3.3}) = \frac{1+1}{3+10} = \frac{2}{13}$

- Obtidas as probabilidades anteriores, pode-se finalmente obter $\hat{P}(c_j|d_i)$, isto é, $\hat{P}(\mathrm{H.3.3}|\mathrm{doc})$ e $\hat{P}(\neg\mathrm{H.3.3}|\mathrm{doc})$ (equação (5.8.19)):

$$\hat{P}(\mathrm{H.3.3}|\mathrm{doc}) = \frac{3}{4} \times \left(\frac{3}{11}\right)^3 \times \frac{1}{22} \times \frac{1}{22} \approx 0{,}00003$$

$$\hat{P}(\neg\mathrm{H.3.3}|\mathrm{doc}) = \frac{1}{4} \times \left(\frac{2}{13}\right)^3 \times \frac{2}{13} \times \frac{2}{13} \approx 0{,}00002$$

Como $\hat{P}(\mathrm{H.3.3}|\mathrm{d}) > \hat{P}(\neg\mathrm{H.3.3}|\mathrm{d})$ o classificador atribui ao documento a classe H.3.3. $\hat{P}(\mathrm{H.3.3}|\mathrm{d}) > \hat{P}(\neg\mathrm{H.3.3}|\mathrm{d})$ porque as três ocorrências do indicador positivo *crawling* (termos que aparecem nos documentos da classe $\mathrm{H.3.3}$) pesam mais do que a ocorrência dos dois indicadores negativos *web graph* e *hyperlink structure* (termos que aparecem nos documentos da classe $\neg\mathrm{H.3.3}$).

```
1  AprendizagemNaiveBayes(Categorias, DocumentosTreino)
2  {
3    Vocabulário ← ObterTermosDistintos(DocumentosTreino)
4    B ← ContarTermosDistintos(Vocabulário)
5    N ← ContarDocs(DocumentosTreino)
6    para cada cj ∈ Categorias fazer {
7       Ncj ← CountarDocsDaClasse(cj, DocumentosTreino)
8       probPriori[cj] ← Ncj / N
9       para cada tk ∈ Vocabulário fazer {
10           T_tk cj ← ContarTermoNaClasse(tk, cj, DocumentosTreino)
11           T_t' cj ← ContarTermosDaClasse(cj, DocumentosTreino)
12           probCondicional[tk][cj] ← (T_tk cj + 1) / (T_t' cj + B)
13      }
14   }
15   devolver probPriori, probCondicional
16 }
```

Algoritmo 1 – Algoritmo Naive Bayes para aprendizagem de um classificador

```
1  ClassificadorNaiveBayes(Categorias, DocumentosTreino,
2                          probPriori, probCondicional,
3                          documento)
4  {
5    D ← ObterTermosDoDocumento(documento)
6    para cada cj ∈ Categorias fazer {
7       pontuacao[cj] ← log probPriori[cj]
8       para cada tk ∈ D fazer {
9            pontuacao[cj] ← log probCondicional[tk][cj]
10      }
11   }
12   devolver maximaPont(pontuacao), classeMaxima(pontuacao)
13 }
```

Algoritmo 2 – Algoritmo Naive Bayes para classificar um novo documento

5.8.2 K-Vizinhos Mais Próximos

Tal como acontece com outras abordagens, cada um dos documentos do conjunto de treino deve ser representado como um vector de termos com pesos (por exemplo, tf.idf). Para classificar um novo documento d_i, são determinados os k documentos mais similares segundo uma medida m de similaridade e atribuído ao documento d_i a classe c_i que a maioria dos k documentos possui, k é um parâmetro inteiro ajustável pelo utilizador. Esta técnica é descrita em [Yang e Pedersen 1997], [Yang e Liu 1999] e [F. Sebastiani 2002] entre outros.

A escolha do parâmetro k, da função de similaridade e o modo como a medida de similaridade m dá origem à classificação a atribuir a novos documentos, permite ter-se várias variações do algoritmo **k-Vizinhos Mais Próximos** (*k-Nearest Neighbor - KNN*).

O parâmetro k é escolhido frequentemente com base em experiências ou conhecimento acerca do problema de classificação em causa. É desejável que seja impar para tornar menos provável o empate em caso de votação para escolher a classe a atribuir. Escolhas habituais são $k = 3$ ou $k = 5$, mas por vezes também são usados valores muito superiores. A importância de um vizinho pode ser levada em conta em função da sua distância, um vizinho mais afastado tem menos influência do que os vizinhos mais próximos.

Uma medida de similaridade usada frequentemente é a **similaridade do co-seno** (*cosine similarity*), mas outras são usadas, por exemplo, em [Larkey e Croft 1996] é utilizado como medida de similaridade as **pontuações de crença** (*belief scores*) do sistema de recuperação de informação *InQuery*.

Na **similaridade do co-seno**, cada documento é representado como um vector de termos, cada termo com um peso. A similaridade entre dois documentos d_1 e d_2 representados respectivamente pelos vectores $\vec{v}(d_1)$ e $\vec{v}(d_2)$ é calculada da seguinte forma:

$$\mathrm{sim}(d_1, d_2) = \cos\left(\vec{v}(d_1), \vec{v}(d_2)\right) = \frac{\vec{v}(d_1) \cdot \vec{v}(d_2)}{\|\vec{v}(d_1)\| \, \|\vec{v}(d_2)\|} \tag{5.8.24}$$

onde $\vec{v}(d_1) \cdot \vec{v}(d_2)$ representa o produto escalar ou produto interno dos vectores $\vec{v}(d_1)$ e $\vec{v}(d_2)$, enquanto $\|\vec{v}(d_1)\| \cdot \|\vec{v}(d_2)\|$ representa o produto da norma (ou comprimento) do vector $\vec{v}(d_1)$ pela norma do vector $\vec{v}(d_2)$.
Sendo o vector $\vec{v}(d_1) = \{v_1(d_1), v_2(d_1), \dots, v_n(d_1)\}$ e o vector $\vec{v}(d_2) = \{v_1(d_2), v_2(d_2), \dots, v_n(d_2)\}$ em que cada $v_k(d_i)$ com $k = 1,2, \dots, n$ (n é o numero total de termos do vector i) e $i = 1,2$ é um componente do vector i, o produto escalar $\vec{v}(d_1) \cdot \vec{v}(d_2)$ é obtido por $\sum_{k=1}^{n} v_k(d_1) v_k(d_2)$, isto é, pelo somatório do produto do componente $v_k(d_1)$ pelo $v_k(d_2)$. O produto escalar de dois vectores é um número, por isso se diz produto escalar.
Para determinar $\|\vec{v}(d_1)\| \cdot \|\vec{v}(d_2)\|$ calcula-se a norma $\|\vec{v}(d_i)\|$ com $i = 1,2$ de cada um dos vectores i através de $\sqrt{\sum_{k=1}^{n} {v_k}^2(d_i)}$, isto é, através da raiz quadrada do somatório do quadrado de cada componente v_k do vector $\vec{v}(d_i)$.
A equação (5.8.24) final é:

$$\text{sim}(d_1, d_2) = \cos\big(\vec{v}(d_1), \vec{v}(d_2)\big) = \frac{\vec{v}(d_1) \cdot \vec{v}(d_2)}{\|\vec{v}(d_1)\| \, \|\vec{v}(d_2)\|} = \frac{\sum_{k=1}^{n} v_k(d_1) v_k(d_2)}{\sqrt{\sum_{k=1}^{n} {v_k}^2(d_1)} \sqrt{\sum_{k=1}^{n} {v_k}^2(d_2)}} \quad (5.8.25)$$

o denominador da equação efectua a normalização dos vectores $\vec{v}(d_1)$ e $\vec{v}(d_2)$, dando origem aos vectores unitários $\overrightarrow{v_u}(d_1) = \frac{\vec{v}(d_1)}{\|\vec{v}(d_1)\|}$ e $\overrightarrow{v_u}(d_2) = \frac{\vec{v}(d_2)}{\|\vec{v}(d_2)\|}$ (vectores unitários têm norma igual a um), assim a equação pode ser reescrita como:

$$\text{sim}(d_1, d_2) = \cos\big(\overrightarrow{v_u}(d_1), \overrightarrow{v_u}(d_2)\big) = \overrightarrow{v_u}(d_1) \cdot \overrightarrow{v_u}(d_2) \quad (5.8.26)$$

onde $\overrightarrow{v_u}(d_1) = \left\{\frac{v_1(d_1)}{\|\vec{v}(d_1)\|}, \frac{v_2(d_1)}{\|\vec{v}(d_1)\|}, \dots, \frac{v_n(d_1)}{\|\vec{v}(d_1)\|}\right\}$ e $\overrightarrow{v_u}(d_2) = \left\{\frac{v_1(d_2)}{\|\vec{v}(d_2)\|}, \frac{v_2(d_2)}{\|\vec{v}(d_2)\|}, \dots, \frac{v_n(d_2)}{\|\vec{v}(d_2)\|}\right\}$. A similaridade entre dois documentos, pode ser vista como o produto escalar ou produto interno dos vectores normalizados dos dois documentos.

A equação (5.8.25) e (5.8.26) são equivalentes, isto é, originam o mesmo resultado.

Exemplo: Dada a frequência dos termos que ocorre em cinco documentos e, sabendo que doc_1, doc_2 e doc_3 pertencem à classe $H.3.3$, doc_4 à classe $\neg H.3.3$ (qualquer outra classe), qual a classe atribuir ao doc_5?

	H.3.3			¬H.3.3	???
	doc_1	doc_2	doc_3	doc_4	doc_5
browsing	0	0	10	0	0
crawling	20	20	10	10	30
hyperlink structure	0	0		10	10
navigation	0	0	10	0	0
search engine	10	0	0	0	0
URL normalization	10	0	0	0	0
visual analysis	0	10	0	0	0
web graph	0	0	0	10	10
web mining	0	10	0	0	0
web search	0	0	10	0	0

Tabela 16 – Frequência dos termos, de cinco documentos

Resolução:

Considerando-se, que $k = 1$, será atribuído ao doc_5 a mesma classe que o vizinho mais próximo, calcular a similaridade entre o doc_5 e os restantes (num conjunto de documentos elevado, o calculo de tantas similaridades pode ser dispendioso, existem heurísticas para melhorar essas situações).

Como indicado na equação (5.8.26), a similaridade entre dois documentos pode ser encontrada pelo produto escalar ou interno dos vectores normalizados dos dois documentos.

A normalização de um vector $\vec{v}(d_i)$ é obtida por $\overrightarrow{v_u}(d_i) = \left\{\frac{v_1(d_i)}{\|\vec{v}(d_i)\|}, \frac{v_2(d_i)}{\|\vec{v}(d_i)\|}, \dots, \frac{v_n(d_i)}{\|\vec{v}(d_i)\|}\right\}$, isto é, dividindo cada componente pela norma do vector.

Começando por determinar a norma do vector $\vec{v}(d_i)$ de cada documento d_i:

	$\|\vec{v}(doc_1)\|$	$\|\vec{v}(doc_2)\|$	$\|\vec{v}(doc_3)\|$	$\|\vec{v}(doc_4)\|$	$\|\vec{v}(doc_5)\|$
	24,4949	24,4949	20	17,32051	33,16625

Tabela 17 – Norma do vector de cada documento

Pode-se agora, normalizar cada um dos vectores $\vec{v}(d_i)$ dividindo cada componente pela norma do vector:

	$\vec{v}(doc_1)$	$\vec{v}(doc_2)$	$\vec{v}(doc_3)$	$\vec{v}(doc_4)$	$\vec{v}(doc_5)$
browsing	0	0	0,5	0	0
crawling	0,816497	0,816497	0,5	0,57735	0,904534
hyperlink structure	0	0	0	0,57735	0,301511
navigation	0	0	0,5	0	0
search engine	0,408248	0	0	0	0
URL normalization	0,408248	0	0	0	0
visual analysis	0	0,408248	0	0	0
web graph	0	0	0	0,57735	0,301511
web mining	0	0,408248	0	0	0
web search	0	0	0,5	0	0

Tabela 18 – Vectores normalizados

Por fim, a similaridade entre dois documentos com base no co-seno pode ser obtida pelo produto escalar ou interno dos vectores normalizados dos dois documentos (equação (5.8.26)):

$$\begin{aligned}\text{sim}(doc_5, doc_1) &= \cos\left(\overrightarrow{v_u}(doc_1), \overrightarrow{v_u}(doc_5)\right) = \overrightarrow{v_u}(d_5) \cdot \overrightarrow{v_u}(d_5) \\ &= 0 \times 0 + 0{,}816497 \times 0{,}904534 + 0 \times 0{,}301511 + 0 \times 0 + 0{,}408248 \times 0 \\ &+ 0{,}408248 \times 0 + 0 \times 0 + 0 \times 0{,}301511 + 0 \times 0 + 0 \times 0 = 0{,}738549\end{aligned}$$

$$\text{sim}(doc_5, doc_2) = 0{,}738549$$
$$\text{sim}(doc_5, doc_3) = 0{,}452267$$
$$\text{sim}(doc_5, doc_4) = 0{,}870388$$

O documento mais similar ao doc_5 é o doc_4 com um valor de $0{,}870388$ e visto que $k = 1$ então é atribuída a classe ¬H.3.3, pois é a classe do doc_4 o seu vizinho mais próximo.

```
1  AprendizagemKNN(DocumentosTreino)
2  {
3    freqTermosPorDoc ← contarTermosPorDoc(DocumentosTreino)
4
5    para cada termos ∈ termosDoc(d_i, freqTermosPorDoc) fazer {
6       para cada v_k ∈ termos fazer {
7            norma[d_i] ← norma[d_i] + v_k^2
8       }
9       norma[d_i] ← √norma[d_i]
10      para cada v_k ∈ termos fazer {
11           tfNormalizado[v_k][d_i] ← v_k / norma [d_i]
12      }
13   }
14   devolver tfNormalizado
15 }
```

Algoritmo 3 – Algoritmo K-Vizinhos Mais Próximos para aprendizagem de um classificador

```
1  ClassificadorKNN(Categorias, tfNormalizado, k, d)
2  {
3    v_k ← calcularNormaENormalizar(d)
4
5    SimVector ← calcularSimilaridade(tfNormalizado, v_k)
6
7    devolver calcularKNN(SimVector, k)
8  }
```

Algoritmo 4 – Algoritmo K-Vizinhos Mais Próximos para classificar um novo documento

5.8.3 Máquinas de Suporte Vectorial

Definição 5.8.19 – Hiperplano: Num espaço unidimensional, um hiperplano é um ponto que divide uma recta. Num espaço bidimensional, um hiperplano é uma linha divide um plano. No espaço tridimensional, um hiperplano é plano habitual que divide o espaço. Este conceito, pode ser aplicado ao espaço n dimensional ($n > 3$), onde o objecto separador é referido simplesmente como hiperplano.

As **Máquinas de Suporte Vectorial**[25] - MSV (*Support Vector Machines - SVM*) são um método introduzido por *Vladimir Vapnik* [Vapnik 1995], [Cortes e Vapnik 1995]. Os métodos de MSV utilizam o **espaço vectorial** (*vector space*) para representar os documentos. Na sua forma mais simples, uma MSV linear tem como objectivo encontrar um separador (hiperplano) entre duas classes com uma margem máxima. A margem é a distância do separador ao elemento mais próximo de cada uma das classes. A figura seguinte ilustra um exemplo de um problema simples de duas classes separáveis linearmente.

[25] Muitas vezes chamadas também Máquinas de Vectores de Suporte

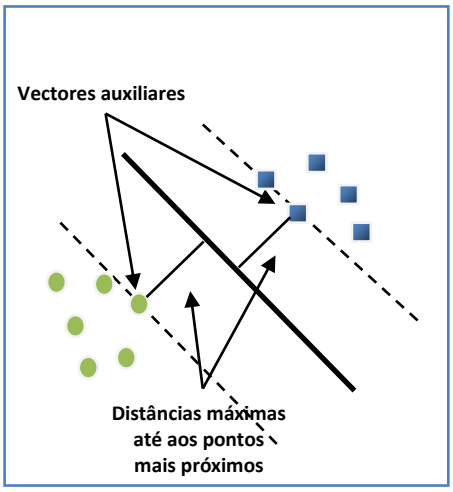


Figura 22 – Representação gráfica de uma MSV linear

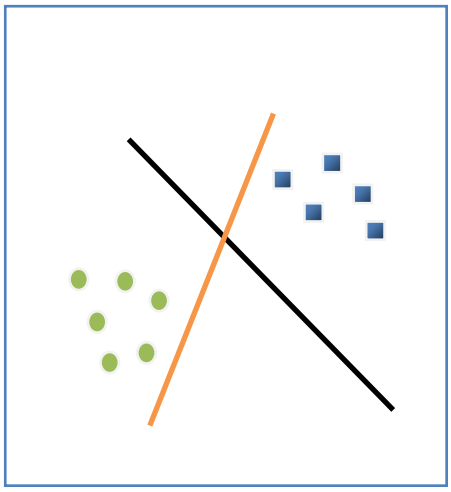


Figura 23 – Exemplo de separadores possíveis

Para duas classes, separáveis linearmente, existem vários separadores. Alguns métodos de aprendizagem como o algoritmo *perceptron* encontram um separador linear, mas não o óptimo, outros como o *Naive Bayes*, procuram o melhor separador linear segundo algum critério. Para uma MSV o critério é o separador que está mais afastado dos pontos mais próximos, não levando em conta os restantes. A distância do separador aos pontos mais próximos é chamada **margem**. Os pontos mais próximos são chamados **vectores auxiliares** (*support vectors*). Maximizar a margem parece ser um bom critério porque os pontos mais próximos são aqueles em que existe mais incerteza acerca da sua classificação.

Quando as classes não são linearmente separáveis, torna-se possível utilizar diferentes funções matemáticas, designadas *kernels*, que têm como objectivo mapeá-las num espaço em que já o sejam, sendo assim possível definir hiperplanos de decisão para o novo espaço.

As MSVs têm um desempenho ao nível dos melhores algoritmos, são robustas perante um conjunto elevado de termos, o que normalmente não acontece com outros algoritmos. Como principais fraquezas, apontam-se a sua sensibilidade a escolhas de valores de parâmetros e a dificuldade de interpretação do modelo gerado. Os trabalhos de *Joachims* [Joachims 1998] e *Lorena* [Lorena 2006], são dois exemplos onde se podem encontrar uma discussão mais detalhada sobre as MSVs. Várias das experiências realizados na literatura com MSVs, são feitos recorrendo à biblioteca libSVM, implementada por *Chih-Chung Chang* e *Chih-Jen Lin* e disponibilizada em http://www.csie.ntu.edu.tw/~cjlin/libsvm ou à biblioteca SVMLight, implementada por *Joachims* disponível em http://svmlight.joachims.org/. Neste trabalho, foi utilizada a biblioteca libSVM.

5.8.4 Rocchio

O algoritmo *Rocchio* [Rocchio 1971] foi criado originalmente para incluir uma **reacção de relevância** (*relevance feedback*) nas consultas do sistema de recuperação de informação SMART. Posteriormente foi adaptado para a classificação de textos [Ittner, Lewis e Ahn 1995].

A **reacção de relevância** era usada tipicamente na recuperação de informação para melhorar as consultas. Era submetida ao sistema de recuperação de informação uma consulta inicial e

dos documentos devolvidos, o utilizador indicava um conjunto de documentos relevantes. A consulta inicial e os termos dos documentos relevantes eram combinados para produzir uma nova consulta que era melhor a ordenar os documentos relevantes.

Uma vez representado cada documento no **espaço vectorial** (*vector space*) [Salton 1991] por um vector de termos com pesos, é calculado um vector protótipo ou perfil para cada categoria dos documentos. A classificação de um novo documento é feita comparando a proximidade do vector que representa o documento com todos os vectores protótipos existentes, sendo atribuída a categoria que possua o vector mais próximo, entendendo-se por mais próximo o mais similar segundo uma medida de similaridade.

A criação do vector protótipo ou perfil de cada categoria foi apresentado em [Ittner, Lewis e Ahn 1995], que utilizou a fórmula de *Rocchio*. O peso de cada termo no vector protótipo é uma combinação dos pesos dos termos dos documentos de treino relevantes D_r (isto é, pertencentes a uma determinada classe c_i) e não relevantes $\overline{D}_{nr}$ (isto é, não pertencentes à categoria c_i) e é calculado pela seguinte fórmula:

$$\omega_{tc_i} = \max\left\{0, \alpha \frac{1}{|D_r|} \sum_{d_j \in D_r} \omega_{td_j} - \beta \frac{1}{|\overline{D}_{nr}|} \sum_{d_j \in \overline{D}_{nr}} \omega_{td_j}\right\} \quad (5.8.27)$$

Onde:

- ω_{tc_i} é o peso do termo t no vector protótipo da categoria c_i;
- ω_{td_j} é o peso do termo t no vector que representa o documento d_j;
- $|D_r|$ é o número de documentos relevantes (isto é, pertencentes a uma determinada classe c_i);
- $|\overline{D}_{nr}|$ é o número de documentos não relevantes (isto é, não pertencentes a uma determinada classe c_i);
- D_r e $\overline{D}_{nr}$ tem o significado já referido;
- α e β parâmetros que ajustam a contribuição dos documentos de treino relevantes e não relevantes.

Em [Ittner, Lewis e Ahn 1995] os valores usados foram $\alpha = 16$ e $\beta = 4$.

Em [Joachims 1998], o uso de um pequeno conjunto de valores para $\alpha = \{0; 0{,}1; 0{,}25; 0{,}5; 1{,}0\}$ revelou um aumento de desempenho em comparação com o atingido anteriormente por outros autores. Contudo, o valor correspondente de β não é mencionado.
Em [Yang 1999] o desempenho obtido foi baixo, a atribuição de valores errados aos parâmetros α e β são uma possível explicação, possivelmente foram atribuídos os valores originais em [Ittner, Lewis e Ahn 1995].

Cohen e *Singer* [Cohen e Singer 1996] experimentaram 31 combinações de α e β e constataram a sensibilidade do algoritmo a estes parâmetros, concluindo que a escolha dos valores apropriados podem aumentar substancialmente o desempenho. Os trabalhos dos diferentes autores, permitem concluir que os parâmetros α e β estão fortemente

dependentes do conjunto de treino e diferentes valores produzem uma variação significante no desempenho.

Os componentes principais do algoritmo são: o método de atribuição do peso aos termos, o método de normalização do tamanho do documento e a medida de similaridade. Diferentes escolhas em cada um destes componentes, originam variantes do algoritmo.
Uma combinação popular, consiste em utilizar a medida TF-IDF na atribuição do peso aos termos, realizar a normalização Euclidiana para normalizar o tamanho do documento e utilizar o co-seno como medida de similaridade.

Exemplo: Partindo dos vectores normalizados de quatro documentos, doc_1, doc_2 e doc_3 pertencentes à classe $H.3.3$, doc_4 pertencente à classe $\neg H.3.3$ (qualquer outra classe), qual a classe a atribuir ao doc_5? Considerar $\alpha = \beta = 1$.

	H.3.3			¬H.3.3	???
	$\vec{v}(doc_1)$	$\vec{v}(doc_2)$	$\vec{v}(doc_3)$	$\vec{v}(doc_4)$	$\vec{v}(doc_5)$
browsing	0	0	0,5	0	0
crawling	0,816497	0,816497	0,5	0,57735	0,904534
hyperlink structure	0	0	0	0,57735	0,301511
navigation	0	0	0,5	0	0
search engine	0,408248	0	0	0	0
URL normalization	0,408248	0	0	0	0
visual analysis	0	0,408248	0	0	0
web graph	0	0	0	0,57735	0,301511
web mining	0	0,408248	0	0	0
web search	0	0	0,5	0	0

Tabela 19 – Tabela de vectores normalizados (Transcrita da secção 5.8.2)

Começa-se por obter o vector protótipo de cada uma das classes, para tal aplica-se a equação (5.8.27) que permite determinar o peso de cada termo do vector protótipo.

- Peso de cada termo do vector protótipo da classe $H.3.3$

  $\omega_{browsing} = 1 \times \frac{1}{|3|} \times (0 + 0 + 0{,}5) - 1 \times \frac{1}{|1|} \times 0 = 0{,}166667$

  $\omega_{crawling} = 1 \times \frac{1}{|3|} \times (0{,}816497 + 0{,}816497 + 0{,}5) - 1 \times \frac{1}{|1|} \times 0{,}57735 = 0{,}133647$

  $\omega_{hyperlink\ \ structure} = 0$ (visto que $\max\{0;\ -0{,}57735\}$)

  $\omega_{navigation} = \omega_{web\ \ search} = 0{,}166667$

  $\omega_{search\ engine} = \omega_{URL\ normalization} = \omega_{visual\ \ analysis} = \omega_{web\ mining} = 0{,}136083$

  $\omega_{web\ \ graph} = 0$

- Peso de cada termo do vector protótipo da classe $\neg \mathbf{H.3.3}$

  $\omega_{browsing} = \omega_{crawling} = 0$

  $\omega_{hyperlink\ \ structure} = 0{,}57735$

  $\omega_{navigation} = \omega_{search\ engine} = \omega_{URL\ normalization} = \omega_{visual\ \ analysis} = 0$

  $\omega_{web\ \ graph} = 0{,}57735$

  $\omega_{web\ mining} = \omega_{web\ \ search} = 0$

$$\cos\left(\overrightarrow{doc_5}, \overrightarrow{prototipo}_{H.3.3}\right) = 0{,}120889$$
$$\cos\left(\overrightarrow{doc_5}, \overrightarrow{prototipo}_{\neg H.3.3}\right) = 0{,}348155$$

Visto o co-seno desempenhar o papel de medida de similaridade, verifica-se que o vector mais similar ao $\overrightarrow{doc_5}$ é o $\overrightarrow{prototipo}_{\neg H.3.3}$ com o valor de $0{,}348155$, como tal é atribuída a classe ¬H.3.3 ao doc_5.

5.9 Trabalhos relacionados

Qualquer trabalho sobre classificação de texto, pode ser considerado, de alguma forma relacionado com a presente dissertação, no entanto, nesta secção procurou-se efectuar um levantamento dos trabalhos directamente relacionados, sendo portanto apresentados trabalhos de classificação multi-etiqueta hierárquica de textos e classificação multi-etiqueta hierárquica de outros objectos que não texto.

Andrea Esuli e colegas em [Esuli, Fagni e Sebastiani. 2006] e [Esuli, Fagni e Sebastiani 2008], propõem o *TreeBoost.MH*, um algoritmo de classificação multi-etiqueta hierárquico, que consiste numa variante hierárquica do *AdaBoost.MH* [Schapire e Singer 2000]. Por sua vez, o algoritmo *AdaBoost.MH* é uma variante multi-etiqueta do *AdaBoost* [Freund e Schapire 1996], [Freund e Schapire 1997], ambos da família dos algoritmos de *boosting* (algoritmos para combinar classificadores). O *TreeBoost.MH*, é um algoritmo recursivo que utiliza o *AdaBoost.MH* como passo base e, percorre a estrutura em árvore. Inclui ideias intuitivas, como por exemplo, a de que a selecção de características e de exemplos negativos de treino deve ser feita localmente, tendo em conta a topologia do esquema de classificação. Inclui também a intuição, de que a ponderação (peso) que os algoritmos de *boosting* actualizam em cada iteração, deve ser feita localmente. As experiências foram realizadas sobre as colecções de textos Reuters-21578 e Reuters-RCV1-v2[26].

Uma abordagem popular em problemas multi-etiqueta, é a sua decomposição em problemas uni-etiqueta independentes (TP4 abordada da secção *5.5.1.1*). Esta abordagem permite utilizar qualquer algoritmo binário já com provas dadas, com o custo de ignorar a relação entre classes. Porém na presença de uma hierarquia, estas relações transformam-se em restrições, que provavelmente serão violadas por este tipo de abordagem. Por exemplo, uma instância pode ser considerada positiva para uma determinada classe e negativa para a sua classe pai (violação da consistência hierárquica). *Barutcuoglu* e colegas em [Barutcuoglu, Schapire e Troya, et al. 2006] e [Barutcuoglu, Schapire e Troyanskaya 2006], propõem a construção de uma rede *Bayesiana* a partir da hierarquia, para inferir o conjunto mais provável de classes hierarquicamente consistentes, de entre (possivelmente inconsistentes) as previsões realizadas por classificadores independentes. A abordagem proposta, consegue lidar com hierárquicas estruturadas em árvore ou estruturadas como Grafo Orientado Acíclico (GOA). As experiências foram realizadas sobre o conjunto de dados biológicos *Gene Ontology* [Ashburner 2000], um conjunto multi-etiqueta cujas categorias estão organizadas hierarquicamente numa estrutura GOA. E realizadas sobre o conjunto de dados *Princeton Shape* [Shilane, et al. 2004],

[26] Disponível em http://trec.nist.gov/data/reuters/reuters.html

uma colecção de computação gráfica 1814 objectos gráficos organizados numa hierarquia em árvore de 170 classes.

Vens e colegas [Vens, et al. 2008], apresentam três abordagens, para a indução de árvores de decisão para classificação multi-etiqueta hierárquica (CMH). Eles comparam a criação de uma única árvore para a CMH, que efectua todas as previsões de uma só vez, com duas abordagens que se baseiam na construção de um conjunto de árvores de decisão (uma para cada classe). Na primeira abordagem, cada classe é vista como um problema independente de classificação binária (esta abordagem ignora a dependência hierárquica). Na segunda abordagem, os classificadores binários da primeira abordagem são aplicados de forma hierárquica. As três abordagens foram experimentadas em vinte e quatro conjuntos de dados *yeast*, utilizando como esquema de classificação o *MIPS*[27] *FunCat* [Mewes, et al. 1999] (estrutura em árvore) e no conjunto de dados biológicos *Gene Ontology* [Ashburner 2000] (estrutura GOA).

Cesa-Bianchi e colegas[Cesa-Bianchi, Gentile e Zaniboni 2006a], apresentam o *B-SVM*, um algoritmo que se propõe a melhorar o desempenho de um algoritmo SVM[28] hierárquico (chamado *H-SVM***)**, ao substituírem a sua aplicação *top-down* por uma aplicação recursiva *bottom-up* que resultou da análise do classificador *Bayesiano* aplicado à função *H-loss* em [Cesa-Bianchi, Gentile e Tironi, et al. 2005]. As experiências foram realizadas sobre a colecção de textos Reuters-RCV1 e OHSUMED (ambas abordadas na secção *2.4.4*).

Struyf e colegas [Struyf, et al. 2005], investigam como as árvores de agrupamento preditivo (*predictive clustering trees*), podem ser usadas para prever as funções de gene da *Saccharomyces cerevisiae*. Este problema é multi-etiqueta e hierárquico. As experiências foram realizadas em doze conjuntos de dados biológicos usados em [A. Clare 2003].

Rousu e colegas, [Rousu, Saunders, et al. 2005a], [Rousu, Saunders, et al. 2005b], [Rousu, Saunders, et al. 2006], propõem um algoritmo *kernel-based* para classificação multi-etiqueta hierárquica. O modelo de classificação é uma variante do *Maximum Margin Markov Network*, onde a hierarquia é representada como uma árvore de *Markov*.
As experiências foram realizadas sobre a colecção de textos Reuters-RCV1 e WIPO-alpha[29] e, sobre a colecção biológica ENZYME.

Outros trabalhos que investigam os problemas de classificação multi-etiqueta hierárquica são: [Wu, Zhang e Honavar 2005], [Cesa-Bianchi, Gentile e Zaniboni 2006b], [Alves, Delgado e Freitas 2008], [Aiolli, et al. 2008].

5.10 Considerações finais

Como conclusão deste capítulo, pode-se dizer que os objectivos propostos foram alcançados, destacando-se o conhecimento adquirido.

[27] Munich Information Center for Protein Sequences
[28] *Support Vector Machines* ou Máquinas de Suporte Vectorial (secção 5.8.3)
[29] http://www.wipo.int/classifications/en

5.10.1 Conhecimentos adquiridos

Nesta fase obteve-se o primeiro contacto com os conceitos básicos da classificação de textos e aprofundaram-se os mais relevantes para este trabalho, dos quais se destacam:

- Abordagens para lidar com problemas de classificação multi-etiqueta;
- Abordagens para lidar com problemas hierárquicos;
- Algoritmos de aprendizagem bastante populares na classificação de textos, nomeadamente o *Naive Bayes*, *K-Vizinhos Mais próximos* e *Máquinas de Suporte Vectorial*.

5.10.2 Objectivos atingidos

O objectivo do corrente capítulo, consistiu em investigar soluções para lidar com o problema de classificação multi-etiqueta hierárquica de textos. O objectivo foi cumprido, para isso foi necessário:

1. Dividir o problema inicial da classificação multi-etiqueta hierárquica em duas partes, uma, em que o objectivo consistiu em investigar abordagens para resolver a característica multi-etiqueta do problema, outra, em que o objectivo consistiu em investigar abordagens para resolver a característica hierárquica;
2. Investigar abordagens para resolver a característica multi-etiqueta;
3. Investigar abordagens para resolver a característica hierárquica.

O capítulo corrente, teve ainda como objectivo o estudo do estado da arte, da classificação multi-etiqueta hierárquica de textos, que foi apresentada na secção *5.9* (página *62*).

5.10.3 Principais dificuldades

Nesta fase, sentiu-se dificuldade em encontrar trabalhos de classificação multi-etiqueta hierárquica de textos. Contudo, a estratégia de dividir o problema de classificação multi-etiqueta hierárquico em duas partes, permitiu ultrapassar esta dificuldade e apresentar uma abordagem para lidar com o problema, abordagem essa que foi aplicada no estudo experimental, apresentado no Capítulo *8* (página *99*).

Capítulo 6. Avaliação dos métodos de classificação

A avaliação dos algoritmos de classificação é uma fase importante, em que para além da escolha das medidas de avaliação, é necessário escolher o método a utilizar. Os métodos mais conhecidos são o *holdout*, o *k-fold cross-validation*, o *leave-one-out* e o *bootstrap*.

Na secção *6.1*, apresentam-se os métodos mais conhecidos utilizados na avaliação dos algoritmos de classificação e na secção *6.2*, são apresentadas as medidas de avaliação usadas nos problemas de classificação binária, classificação multi-classe, classificação hierárquica e classificação multi-etiqueta.

6.1 Métodos de avaliação

A avaliação dos algoritmos de classificação é importante, pois permite obter uma medida de avaliação, que pode servir, por exemplo, de referência para se tentar progredir futuramente. Existem vários métodos para realizar a avaliação, contudo, determinar qual ou quais utilizar num problema em particular, não é tão simples como pode parecer à primeira vista.

Para efectuar a avalização de um método é comum partir-se de um conjunto de dados e dividi-lo em dois ou três conjuntos. Quando se divide em dois, é para obter um **conjunto de treino** e um **conjunto de teste**. Quando se divide em três, é para obter um **conjunto de treino**, um **conjunto de validação** e um **conjunto de teste**. O **conjunto de treino** é usado por um ou mais métodos de aprendizagem para obter um ou mais classificadores. O **conjunto de validação** é usado para optimizar parâmetros do classificador, ou para seleccionar um em particular. O **conjunto de teste** é usado para calcular a taxa de erro ou outra medida do classificador final optimizado. Cada um dos conjuntos, deve ser independente, isto é, cada um dos conjuntos deve conter instancias que não são usadas nos restantes, isto para se obter uma estimativa fiável da medida de avaliação.
Uma vez determinada a medida de avaliação, os dados do conjunto de teste são integrados no conjunto de treino, para aprender um novo classificador, para uso efectivo. Este procedimento, não tem nada de errado, é apenas uma forma de maximizar a quantidade de dados usados para gerar o classificador que será usado na prática.

Quando existe uma vasta quantidade de dados, não existem problemas no que respeita à criação do conjunto de treino, conjunto de teste e eventualmente conjunto de validação. Por outro lado se a quantidade de dados é limitada, levanta-se o problema de como fazer o melhor, com um conjunto de dados limitado. Existem vários métodos, dos quais a repetição da validação cruzada (*cross validation*), é o método de avaliação tendencialmente escolhido para resolver esta situação. De seguida apresentam-se esse e outros métodos conhecidos.

No método *holdout*, reserva-se uma certa quantidade de documentos para teste e a restante para treino (se necessário, define-se uma parte para validação) [Witten e Frank 2005]. Em termos práticos, é comum reservar-se um terço dos documentos para teste e os restantes dois terços para treino. É evidente que o conjunto de treino ou teste pode não ser representativo das classes existentes. Por azar, pode acontecer que certas classes não possuam exemplos de treino, nestas condições é difícil esperar que o classificador faça uma aprendizagem capaz de obter bons desempenhos nessas classes. É aconselhável efectuar um simples teste para verificar se todas as classes que figuram no conjunto de dados estão representadas na proporção adequada no conjunto de treino e teste. Alternativamente, pode-se tentar assegurar que a escolha é feita de forma aleatória de forma a garantir que cada classe é representada apropriadamente tanto no conjunto de treino e teste. Este procedimento é chamado estratificação (*stratification*), neste caso pode-se falar em *holdout* estratificado (*stratified holdout*).

No método *k-folds cross-validation*, o conjunto de dados disponível é particionado em k subconjuntos disjuntos (*k-folds*) de tamanho igual (ou aproximadamente, uma vez que em determinadas situações é matematicamente impossível). Supor que k é definido com o valor três. O conjunto de dados é dividido em três partições aproximadamente iguais e é utilizada à vez uma partição para teste e as restantes para treino. Isto é, utiliza-se dois terços para treino e um terço para teste, repete-se o procedimento três vezes, de modo a que no fim, todas as partições tenham sido utilizadas exactamente uma vez para teste (desta forma todos os exemplos são testados). A escolha do valor 10 para k é muito comum, outros valores como 5 ou 20 são apontados em [Witten e Frank 2005] como sendo escolhas quase tão boas, o valor 3 é apontado em [A. Clare 2003] como uma escolha popular. Como referido anteriormente é um método escolhido quando o conjunto de dados não é muito grande. Se é adoptada a estratificação, o que acontece com frequência, o método é chamado validação cruzada estratificada (*stratified cross-validation*).

O método *leave-one-out* também conhecido como *leave-one-out cross-validation*, é um caso especial da validação cruzada em que cada partição é de tamanho um. Tem-se assim, tantas partições quanto o número de exemplos do conjunto de dados. Este método é atractivo pela possível quantidade de instâncias utilizadas para treino, o que aumenta a oportunidade do classificador efectuar previsões correctas. Outro aspecto positivo, é o facto de ser um procedimento determinístico, já que não envolve nenhuma selecção aleatória. Um aspecto desfavorável, é o custo computacional elevado, porque todo o procedimento de aprendizagem precisa de ser executado n vezes o que pode ser impraticável para grandes conjuntos de dados. Por estas razões, este método é indicado para conjuntos de dados muito pequenos.

O método *bootstrap*, baseia-se no procedimento estatístico de amostragem com reposição. A ideia deste método, consiste em obter uma amostra com reposição do conjunto de dados para formar o conjunto de treino e teste. Geralmente, este procedimento é repetido várias vezes de modo a criar vários conjuntos de treino e teste. *Kohavi* [Kohavi 1995], efectua um estudo sobre o método de avaliação *bootstrap* e sobre o método de validação cruzada.

6.2 Medidas de avaliação

Habitualmente o desempenho da classificação de textos é avaliado em termos da capacidade de produzir classificações correctas (capacidade ou qualidade de previsão). Algumas vezes o desempenho é também medido em termos de eficiência, por exemplo, tempo de execução. Neste trabalho, a investigação restringe-se sobre a avaliação da capacidade de revisão.

As medidas de avaliação do desempenho da previsão a utilizar, dependem basicamente do tipo de problema de classificação e da forma como se pretende relatar os resultados. Os resultados podem ser relatados:

- **Por classe**;
- **Por modelo**, isto significa que em vez de se obter uma medida de desempenho para cada uma das classes, obtém-se uma medida de avaliação do desempenho como um todo;
- **Por nível** (questão que se coloca apenas quando o problema é hierárquico).

Várias medidas de avaliação, efectuam os cálculos a partir de uma **matriz de confusão**[30] (*confusion matrix*) também chamada **tabela de contingência**[31] (*contingency table*) [Bush, Edwards e Dudek 2008], que para cada classe, armazena o número de instâncias correcta e incorrectamente classificadas.

Definição 6.2.20 – Matriz de confusão: Matriz que contém informação acerca da classificação actual (correcta) e prevista por um sistema de classificação. O desempenho desse sistema é habitualmente avaliado utilizando os dados contidos na matriz [Provost e Kohavi 1998].

6.2.1 Problemas de Classificação binária

Na classificação binária, uma instância (documento), é classificada numa de duas categorias possíveis. Digamos que as categorias possíveis são c e $\bar{c}$ e, os resultados de um classificador binário, são apresentados na matriz de confusão que se segue.

Categoria c_i		Classificação correcta	
		c	$\bar{c}$
Classificação prevista	c	PV (TP)	PF (FP)
	$\bar{c}$	NF (FN)	NV (TN)

Tabela 20 – Matriz de confusão (2×2)

Nos problemas de classificação binária, uma das classes é vista como a classe positiva e a outra como negativa, na matriz acima c e $\bar{c}$ respectivamente. Tendo cada uma das células, o seguinte significado:

- **PV – Positivos Verdadeiros** (*True Positives - TP*), é o número de instâncias, correctamente previstos na classe positiva (c);

[30] Usada habitualmente no campo da inteligência artificial
[31] Usada habitualmente na estatística para guardar e analisar a relação entre duas ou mais variáveis

- **NV – Negativos Verdadeiros** (*True Negatives - TN*), é o número de instâncias, correctamente previstos na classe negativa ($\bar{c}$);
- **PF – Positivos Falsos** (*False Positives - FP*), é o número de instâncias, incorrectamente previstos na classe positiva (c);
- **NF – Negativos Falsos** (*False Negatives - FN*), é o número de instâncias, incorrectamente previstos na classe negativa ($\bar{c}$).

Bons resultados, correspondem a um largo número de instâncias na diagonal principal (diagonal que possui as previsões verdadeiras) e pequeno, idealmente zero, nas células fora dessa diagonal.

A partir da matriz de confusão, é possível calcular medidas de desempenho da previsão. Na classificação binária as medidas convencionalmente utilizadas são a previsão (Pr), a abrangência (Ab), a taxa de acerto (TA) e o erro (TE).

As medidas seguintes, são usadas para avaliar uma categoria c:

Definição 6.2.21 - precisão de uma classe c: A precisão de uma classe c, é a proporção de previsões feitas na classe c, que estão correctas.

$$Pr = \frac{PV}{PV + PF} \tag{6.2.28}$$

Definição 6.2.22 - abrangência de uma classe c: A abrangência de uma classe c, é a proporção de documentos da classe c, que foi correctamente prevista.

$$Ab = \frac{PV}{PV + NF} \tag{6.2.29}$$

Nem a precisão nem a abrangência são boas medidas isoladas [Yang e Liu 1999], [Kiritchenko 2005]. Em geral, elevados valores de abrangência podem ser obtidos à custa de uma precisão baixa e vice-versa. Como consequência, na classificação de textos é comum usarem-se medidas que combinam a abrangência e a precisão. Sendo uma das mais utilizadas, a **medida-F** (*F-measure*), proposta por *Rijsbergen* [Rijsbergen 1979].

Definição 6.2.23 – medida-F de uma classe c: A medida-F de uma classe c, corresponde ao significado harmónico[32] da precisão e abrangência.

$$F_\beta = \frac{(\beta^2 + 1) \cdot Pr \cdot Ab}{(\beta^2 \cdot Pr + Ab)}, \beta \in [0, \infty[\tag{6.2.30}$$

Onde Pr e Ab, possuem o mesmo significado, já anteriormente referido e β é um parâmetro usado para atribuir um peso à precisão e abrangência. O valor padrão de β é 1, que atribui pesos iguais à precisão e abrangência. É muito comum na literatura fazerem referência à medida F_1, trata-se da medida-F quando β é 1 e nesse caso a fórmula é:

$$F_1 = \frac{2 \cdot Pr \cdot Ab}{Pr + Ab} \tag{6.2.31}$$

[32] O significado harmónico de dois números tende a estar perto do mais pequeno dos dois

As medidas *Break-Even Point* (BEP) proposta por [D. D. Lewis 1992b] e a *Average 11-Point Precision*, são exemplos de outras medidas usadas para combinar a precisão e abrangência. A sua descrição pode ser encontrada por exemplo em [Yang e Liu 1999].

As duas medidas seguintes, avaliam a efectividade das previsões como um todo:

Definição 6.2.24 – taxa de acerto: A taxa de acerto, fornece a proporção de documentos correctamente classificados.

$$TA = \frac{PV + NV}{PV + NV + PF + NF} \tag{6.2.32}$$

Definição 6.2.25 – taxa de erro: A taxa de erro, fornece a percentagem de documentos incorrectamente classificados.

$$TE = 1 - TA = \frac{PF + NF}{PV + NV + PF + NF} \tag{6.2.33}$$

Outro modo de avaliação, é a utilização de curvas ROC (*receiver operating characteristic*). Uma curva ROC, é um gráfico que ilustra a relação entre a taxa de positivos verdadeiros ou sensibilidade e a taxa de positivos falsos ou $(1 - sensibilidade)$. Neste contexto, sensibilidade é outro nome para a abrangência. A taxa de positivos falsos, é dada por $PF/(PF + NV)$ e é representada no eixo horizontal. A taxa de positivos verdadeiros é representada no eixo vertical.

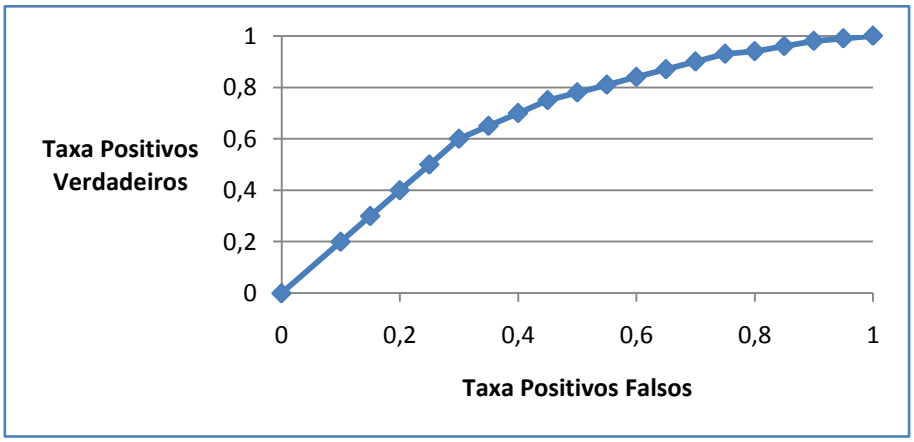


Figura 24 – Exemplo de curva ROC

6.2.2 Problemas de classificação multi-classe

Nos problemas de classificação multi-classe, tem-se $|C|$ categorias ($|C| > 2$), $c_i, \dots, c_{|C|}$ ($i = 1, 2, \dots, |C|$). Neste tipo de problema, para avaliar o seu desempenho previsivo, pode-se construir $|C|$ matrizes de confusão binárias, uma para cada classe $c_1, c_2, \dots, c_{|C|}$ e aplicar as medidas de avaliação apresentadas anteriormente. Alternativamente, pode-se construir uma matriz de confusão $|C| \times |C|$ e depois decompô-la, de forma a obter as $|C|$ matrizes.

		Classificação correcta			
		c_1	c_2	...	c_n
Classificação prevista	c_1	n_{11}	n_{12}	...	n_{1n}
	c_2	n_{21}	n_{22}	...	n_{2n}
	...	...	...	...	...
	c_n	n_{n1}	n_{n2}	...	n_{nn}

Tabela 21 – Matriz de confusão ($|C| \times |C|$)

A decomposição da matriz $|C| \times |C|$, é feita construindo uma matriz 2×2 para a classe c_1 contendo as classes c_1 e $\bar{c}_1$ ($\bar{c}_1$ designa todas as categorias que não c_1), outra matriz para a classe c_2 com as classes c_2 e $\bar{c}_2$ e assim sucessivamente para as $|C|$ classes. O preenchimento de uma matriz M (2×2), de uma classe $c_i = c_1, c_2, \dots, c_{|C|}$, é feito da seguinte forma:

Mi_{11} = Elementos pertencentes à classe c_i e o classificador prevê c_i;
Mi_{21} = Elementos pertencentes à classe c_i e o classificador não prevê c_i;
Mi_{12} = Elementos não pertencentes à classe c_i e o classificador prevê c_i;
Mi_{22} = Elementos não pertencentes à classe c_i e o classificador não prevê c_i.

(Mi_{jk}, significa, célula da linha i e coluna j, da matriz Mi)

Categoria c_i		Classificação correcta	
		c_i	$\bar{c}_i$
Classificação prevista	c_i	PV_i (TP)	PF_i (FP)
	$\bar{c}_i$	NF_i (FN)	NV_i (TN)

...

Categoria $c_{\|C\|}$		Classificação correcta	
		$c_{\|C\|}$	$\bar{c}_{\|C\|}$
Classificação prevista	$c_{\|C\|}$	$PV_{\|C\|}$ (TP)	$PF_{\|C\|}$ (FP)
	$\bar{c}_{\|C\|}$	$NF_{\|C\|}$ (FN)	$NV_{\|C\|}$ (TN)

Tabela 22 – Matrizes de confusão resultantes da decomposição da matriz $|C| \times |C|$

Com base nas matrizes de confusão binárias, resultantes da decomposição da matriz de confusão $|C| \times |C|$, pode-se calcular para cada classe c_i, medidas de desempenho como: precisão (Pr_i), abrangência (Ab_i) e medida-F (F_{β_i}), que possuem o significado apresentado na secção anterior.

$$Pr_i = \frac{PV_i}{PV_i + PF_i} \qquad \mathrm{Ab_i} = \frac{\mathrm{PV_i}}{\mathrm{PV_i} + \mathrm{NF_i}} \qquad F_{\beta_i} = \frac{(\beta^2 + 1) \cdot Pr_i \cdot Ab_i}{(\beta^2 \cdot Pr_i + Ab_i)}, \beta \in [0, \infty[\qquad (6.2.34)$$

De forma a avaliar o desempenho a nível global (e não apenas de cada classe individualmente), pode-se combinar as medidas de avaliação. A combinação, é feita calculando-se a média, podendo esta ser calculada através de dois métodos, um designado **macro-média** (*macroaverage*) e outro **micro-média** (*microaverage*).

Definição 6.2.26 – Medidas de Avaliação Macro-média: Para calcular uma medida de avaliação de forma macro-média, começa-se por calcular para cada categoria a medida de avaliação e por fim efectua-se a sua média. Para todas as classes, a medida em causa é somada e dividida pelo número total de classes ($|C|$).

A precisão, abrangência e medida-F, são calculadas de forma macro-média, da seguinte maneira:

$$MPr = \frac{\sum_{i=1}^{|C|} Pr_i}{|C|} \qquad MAb = \frac{\sum_{i=1}^{|C|} Ab_i}{|C|} \qquad MF_\beta = \frac{\sum_{i=1}^{|C|} F_{\beta_i}}{|C|} \tag{6.2.35}$$

Definição 6.2.27 – Medidas de Avaliação Micro-média: Para calcular uma medida de avaliação de forma micro-média, começa-se por criar uma matriz de confusão global, cujos valores das células, resultam da soma das células correspondentes das matrizes de confusão de cada categoria. Por fim calculam-se as medidas de avaliação em questão.

A precisão, abrangência e medida-F, são calculadas de forma micro-média, da seguinte maneira:

$$\mu Pr = \frac{\sum_{i=1}^{|C|} PV_i}{\sum_{i=1}^{|C|} (PV_i + PF_i)} \qquad \mu Ab = \frac{\sum_{i=1}^{|C|} PV_i}{\sum_{i=1}^{|C|} (PV_i + NF_i)} \qquad \mu F_\beta = \frac{\sum_{i=1}^{|C|} ((\beta^2 + 1) \cdot Pr_i \cdot Ab_i)}{\sum_{i=1}^{|C|} (\beta^2 \cdot Pr_i + Ab_i)} \tag{6.2.36}$$

Para avaliar o desempenho global do classificador, pode-se calcular também a taxa de acerto (TA) e taxa de erro (TE), que possuem o significado apresentado respectivamente na **Definição 6.2.24** e **Definição 6.2.25**. O cálculo de cada uma das medidas é feito da seguinte forma:

$$TA = \frac{\sum_{i=1}^{|C|} PV_i}{\sum_{i=1}^{|C|} (PV_i + PF_i)} \qquad TE = 1 - TA = \frac{\sum_{i=1}^{|C|} PF_i}{\sum_{i=1}^{|C|} (PV_i + PF_i)} \tag{6.2.37}$$

Exemplo: Imagine-se uma matriz 3×3, que resume as previsões efectuadas por um determinado classificador:

		Classificação correcta		
		c_1	c_2	c_3
Classificação prevista	c_1	65	15	45
	c_2	30	25	15
	c_3	5	60	40

Resolução:

Decomposição da matriz 3×3, em 3 Matrizes binárias, uma para cada classe c_i:

Classe c_1		Classificação correcta	
		c_1	$\bar{c}_1$
(*)	c_1	65	60
	$\bar{c}_1$	35	140

Classe c_2		Classificação correcta	
		c_2	$\bar{c}_2$
(*)	c_2	25	45
	$\bar{c}_2$	75	155

Classe c_3		Classificação correcta	
		c_3	$\bar{c}_3$
(*)	c_3	40	65
	$\bar{c}_3$	60	135

(*) Classificação atribuída pelo classificador

Cálculo da precisão (Pr_i), abrangência (Ab_i) e medida-F (F_{β_i}), para cada classe c_i:

$$Pr_1 = \frac{65}{65+60} \qquad Pr_2 = \frac{25}{25+45} \qquad Pr_3 = \frac{40}{40+65}$$

$$Ab_1 = \frac{65}{65+35} \qquad Ab_2 = \frac{25}{25+75} \qquad Ab_3 = \frac{40}{40+60}$$

$$F_{1_1} = \frac{2 \cdot Pr_1 \cdot Ab_1}{Pr_1 + Ab_1} \qquad F_{1_2} = \frac{2 \cdot Pr_2 \cdot Ab_2}{Pr_2 + Ab_2} \qquad F_{1_3} = \frac{2 \cdot Pr_2 \cdot Ab_2}{Pr_2 + Ab_2}$$

Cálculo da macro-media da previsão (MPr) e da abrangência (MAb):

$$MPr = \frac{1}{3}(Pr_1 + Pr_2 + Pr_3)$$

$$MAb = \frac{1}{3}(Ab_1 + Ab_2 + Ab_3)$$

Cálculo da micro-media da previsão (μPr) e da abrangência (μAb):

$$\mu Pr = \frac{65+25+40}{(65+60)+(25+45)+(40+65)}$$

$$\mu Ab = \frac{65+25+40}{(65+35)+(25+75)+(40+60)}$$

Cálculo da Taxa de Acerto (TA) e Taxa de Erro (TE):

$$TA = \frac{65+25+40}{(65+60)+(25+45)+(40+65)}$$

$$TE = \frac{60+45+65}{(65+60)+(25+45)+(40+65)}$$

6.2.3 Problemas de classificação hierárquicos

Para avaliar o desempenho de previsão de um método de classificação hierárquica, é possível aplicar directamente as medidas utilizadas na classificação plana (classificação binária e classificação multi-classe), como por exemplo, precisão, abrangência, entre outras, os trabalhos [Koller e Sahami 1997] e [Dumais e Chen. 2000], fazem-no. Contudo os problemas de classificação hierárquica, possuem especificidades que fazem com que estas medidas não sejam as mais adequadas.

Imagine-se os três cenários ilustrados abaixo, em que os círculos preenchidos representam a classe correcta, os círculos com contorno a negrito a classe prevista:

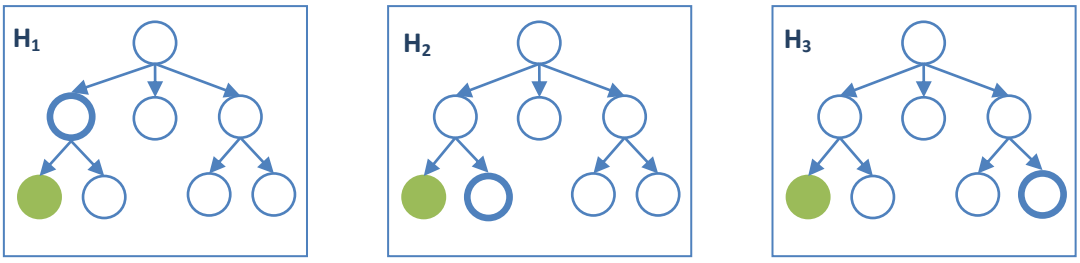


Figura 25 – Distância entre duas classes, baseada no número de ramos

Na hipótese 1 (H_1) foi prevista a classe "pai", da classe correcta. Na H_2, foi prevista a classe "irmã", da classe correcta. Na H_3 a classe prevista, está muito distante da correcta. Tem-se por tanto três cenários em que as previsões estão incorrectas, mas umas previsões mais próximas da correcta do que outras, por exemplo, tanto a previsão em H_1 como em H_2, estão mais próximas da previsão correcta do que a H_3. Podemos dizer que, se um documento não é correctamente classificado, devemos considerar o grau de incorrecção, por outras palavras,

devemos considerar o quanto errada é a classificação e, não apenas se a classificação está correcta ou errada.

Wang e colegas [Wang, Zhou e Liew 1999] chamam à atenção para a necessidade de se considerar o quanto uma classificação incorrecta está próxima da correcta e propõem a noção de **proximidade da classe** (*class proximity*), para representar o erro cometido na classificação. Para medir o erro da previsão ou quantificar uma classificação incorrecta, propõem o uso da distância entre dois nós na hierarquia de classes, baseada no número de ramos. Considerando a distância entre uma classe c_i e uma classe c_j, como o caminho mais curto entre ambas. Na figura seguinte é possível observar a distância em três situações distintas. Os círculos preenchidos representam a classe correcta, os círculos com contorno a negrito a classe predita, os ramos a negrito a distância mais curta entre as classes.

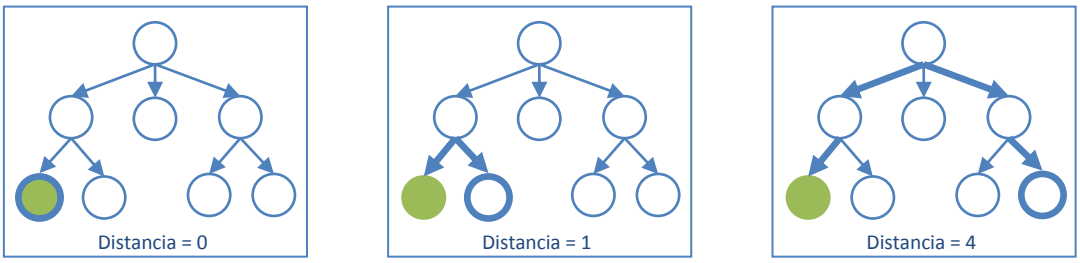


Figura 26 – Distância entre duas classes, baseada no número de ramos

Num trabalho posterior [Wang, Zhou e He 2001], os mesmos autores, optam por utilizar uma medida baseada na similaridade entre classes em vez da distância. Medindo a similaridade entre duas classes, através da similaridade dos documentos que pertencem a essas classes. Os autores assumem também que a classificação é multi-etiqueta, e o erro cometido entre o conjunto de classes atribuídas pelo classificador ($C_i^{'}$) e o conjunto das categorias correctas (C_i), é calculado com base na cobertura dos dois conjuntos:

$$E(C_i, C_i^{'}) = \frac{|Cob(C_i^{'}) - Cob(C_i)| + |Cover(C_i) - Cover(C_i^{'})|}{|Cob(C_i) \cup Cob(C_i^{'})|} \quad (6.2.38)$$

Onde:

- $Cob(C_i)$ é a cobertura do conjunto de classes C_i (conjunto de classes correctas);
- $Cob(C_i^{'})$ é a cobertura do conjunto de classes $C_i^{'}$ (conjunto de classes previstas).

A cobertura de um conjunto de classes, inclui todos os documentos etiquetados com as categorias que fazem parte do conjunto de classes em questão ou das categorias cujos ascendentes estão no conjunto.

Sun e *Lim* [Sun e Lim 2001], confirmam que a distância entre classes (apresentada nas figuras acima), como medida da incorrecção da classificação, foi proposta pela primeira fez por *Wang* e colegas [Wang, Zhou e Liew 1999], mas que estes, não definem medidas de desempenho. Por sua vez, *Sun* e *Lim* [Sun e Lim 2001], propõem modificações que permitem utilizar as medidas convencionais como a precisão, a abrangência, entre outras.

Quando um documento d_j é previsto correctamente na classe c_i, é contabilizado como 1 nos positivos verdadeiros da classe c_i. Contudo, se um documento d_j é previsto incorrectamente na classe c_i (se é um PF_i para a classe c_i), ou se é incorrectamente rejeitado da classe c_i (se é

um NF_i para a classe c_i), é necessário determinar o grau de incorrecção. O grau de incorrecção, é conseguido calculando-se a contribuição dos positivos falsos para a classe c_i ($PfCon_i$) e a contribuição dos negativos falsos para a classe c_i ($NfCon_i$). Uma vez calculadas as referidas contribuições, elas são incorporadas nas medidas tradicionais, originando assim as medidas "convencionais" modificadas:

$$Pr_i = \frac{\max(0,\ PV_i + PfCon_i + NfCon_i)}{PV_i + PF_i + NfCon_i} \tag{6.2.39}$$

$$Ab_i = \frac{\max(0,\ PV_i + PfCon_i + NfCon_i)}{PV_i + NF_i + PfCon_i} \tag{6.2.40}$$

Micro-média:

$$\mu Pr = \frac{\sum_{i=1}^{|C|}(\max(0,\ PV_i + PfCon_i + NfCon_i))}{\sum_{i=1}^{|C|}(PV_i + PF_i + NfCon_i)} \tag{6.2.41}$$

$$\mu Ab = \frac{\sum_{i=1}^{|C|}(\max(0,\ PV_i + PfCon_i + NfCon_i))}{\sum_{i=1}^{|C|}(PV_i + NF_i + PfCon_i)} \tag{6.2.42}$$

Macro-média:

$$MPr = \frac{\sum_{i=1}^{|C|} Pr_i}{|C|} \qquad MAb = \frac{\sum_{i=1}^{|C|} Ab_i}{|C|} \tag{6.2.43}$$

A taxa de acerto (TA_i) e taxa de erro (TE_i):

$$TA_i = \frac{PV_i + NV_i + PfCon_i + NfCon_i}{PV_i + NV_i + PF_i + NF_i} \quad TE_i = \frac{PF_i + NF_i - PfCon_i - NfCon_i}{PV_i + NV_i + PF_i + NF_i} \tag{6.2.44}$$

Para efectuar o cálculo das contribuições ($PfCon_i$ e $NfCon_i$), *Sun* e *Lim*, propõem duas formas, uma **baseada na distância** entre as categorias e outra baseada **na similaridade de categorias** (*Category-Similarity*). Caso se calcule as contribuições pela primeira forma, as medidas resultantes são designadas **medidas baseadas na distância**, se for utilizada a segunda forma, as medidas são designadas **medidas baseadas na similaridade de categorias** (*Category-Similarity*).

Medidas baseadas na distância de categoria

Para se obter a contribuição dos positivos falsos para uma classe c_i ($PfCon_i$) e contribuição dos negativos falsos para uma classe c_i ($NfCon_i$), começa-se por calcular a **contribuição** (*contribution*) de um documento d_j para a classe c_i, referida como $Con(d_j, c_i)$.

Para calcular a contribuição $Con(d_j, c_i)$, utilizando a distância entre duas categorias, o utilizador, precisa primeiro definir uma **distância aceitável** (*acceptable distance*), denotada por $Dist_\theta$, cujo valor precisa ser superior a zero.

A $Con(d_j, c_i)$, é então definida por:

$$Con(d_j, c_i) = \sum_{c' \in d_j.lbl} 1 - \frac{Dist(c', c_i)}{Dist_\theta} \quad (6.2.45)$$

Onde:

- $d_j.lbl$ representa o conjunto de etiquetas previstas para o documento d_j.

$$Con(d_j, c_i) = \sum_{c' \in d_j.agd} 1 - \frac{Dist(c', c_i)}{Dist_\theta} \quad (6.2.46)$$

Onde:

- $d_j.agd$ representa o conjunto de etiquetas reais do documento d_j.

A contribuição de d_j para a classe c_i, $Con(d_j, c_i)$, está num dos três intervalos:

1. $0 < Con(d_j, c_i) \leq 1$ quando $0 < Dist(c', c_i) < Dist_\theta$
2. $Con(d_j, c_i) = 0$ quando $Dist(c', c_i) = Dist_\theta$
3. $-1 \leq Con(d_j, c_i) < 0$ quando $Dist(c', c_i) > Dist_\theta$

No caso 1, a contribuição é positiva, quando o documento d_j é previsto na classe c', que não está muito longe da categoria correcta (está a uma distância inferior à aceitável). No caso 2, quando $Dist(c', c_i) = Dist_\theta$, a previsão continua a ser aceitável e não deve ser punida, como tal a contribuição é 0 (zero). No caso 3, como o documento d_j é previsto na classe c', que está a uma distância superior à aceitável, a contribuição é negativa.

De notar que um documento pode pertencer ou ser previsto em mais do que uma categoria. Para prevenir que um documento seja demasiado penalizado ou demasiado recompensado, a **contribuição** de cada documento d_j para a classe c_i deve ser restringida ao intervalo $[-1,1]$. Surge assim, a **contribuição refinada** (*Refined - Contribution*), denotada por $RCon(d_i, c_j)$:

$$RCon(d_j, c_i) = \min\left(1, \max\left(-1, Con(d_j, c_i)\right)\right) \quad (6.2.47)$$

Para todos os documentos pertencentes aos PF_i, o total das contribuições dos positivos falsos ($PfCon_i$), é:

$$PfCon_i = \sum_{d_j \in PF_i} RCon(d_j, c_i) \quad (6.2.48)$$

Para todos os documentos pertencentes aos NF_i, o total das contribuições dos negativos falsos ($NfCon_j$), é:

$$NfCon_i = \sum_{d_j \in NF_i} RCon(d_j, c_i) \quad (6.2.49)$$

Medidas baseadas na similaridade de categorias

Na proposta de *Sun* e *Lim* [Sun e Lim 2001], é também possível usarem-se medidas de desempenho de previsão baseadas na similaridade de categorias. Para tal é suficiente, alterar a forma de calcular a contribuição de um documento d_j para a classe c_i, $Con(d_j, c_i)$ equação (6.2.45) e (6.2.46).

Para se calcular a contribuição $Con(d_j, c_i)$, é necessário definir uma medida de similaridade entre duas categorias c_i e c_k, referida como $SC(c_i, c_k)$. É preciso também calcular a média da similaridade das categorias, referida como $MSC(c_i, c_k)$. A contribuição $Con(d_j, c_i)$ é então definida por:

$$Con(d_j, c_i) = \frac{\sum_{c' \in d_j.lbd} (SC(c', c_i) - MSC)}{1 - MSC} \quad (6.2.50)$$

Onde:

- $d_j.lbl$ representa o conjunto de etiquetas previstas para o documento d_j.

À semelhança da contribuição baseada na distância, a contribuição baseada na similaridade de categorias deve ser restringida ao intervalo $[-1,1]$. Essa restrição é realizada na equação da contribuição refinada (6.2.47).

Blockeel e colegas [Blockeel, et al. 2002], propõem uma medida também baseada na distância entre classes, mas considerando pesos diferentes para ramos de diferentes níveis. Os pesos decrescem com a profundidade dos níveis (ver figura abaixo). No mesmo trabalho, é proposto que os pesos sejam associados aos nós (classes) e a distância entre duas classes seja o peso do ascendente mais profundo comum a ambas.

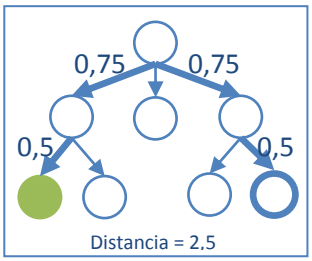


Figura 27 – Distância entre duas classes, baseada em pesos decrescentes dos ramos

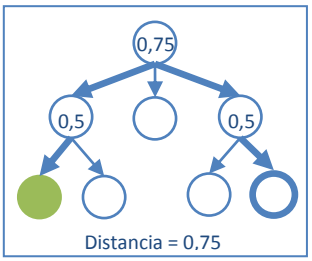


Figura 28 – Distância entre duas classes, baseada no peso do ascendente mais profundo, comum a ambas

6.2.4 Problemas de classificação multi-etiqueta

Na classificação uni-etiqueta, a previsão da classe de um documento apenas pode estar certa ou errada, pois o documento pertence ou não à classe prevista. Na classificação multi-etiqueta, a previsão pode estar certa, errada ou parcialmente certa (ou caso se prefira, parcialmente errada), pois em documentos com duas ou mais etiquetas a previsão pode acertar em todas elas, em nenhuma ou em parte. Portanto, a classificação multi-etiqueta requer diferentes medidas de avaliação das usadas na classificação uni-etiqueta.

Várias medidas foram propostas para avaliar os classificadores multi-etiqueta. *Tsoumakas* e *Vlahavas* [Tsoumakas e Vlahavas 2007b], agrupam essas medidas em dois grupos. As medidas

baseadas em exemplos (*example-based*) e as medidas **baseadas em etiquetas** (*label-based*). As primeiras, caracterizam-se por serem baseadas na diferença média do conjunto de classes reais e conjunto de classes previstas de cada documento do conjunto de teste. As segundas, caracterizam-se por serem calculadas com base na avaliação individual de cada etiqueta e determinação posterior da média.

As medidas seguintes, são baseadas em exemplos.

Schapire e *Singer* [Schapire e Singer 2000], consideram a medida *Hamming Loss*:

$$Hamming\ Loss = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \Delta\ Z_i|}{|L|} \tag{6.2.51}$$

Onde:

- $|D|$ é o número de documentos do conjunto de teste;
- $|L|$ é o número de existentes para classificação (conjunto pré-definido de classes);
- Y_i representa o conjunto de classes verdadeiras do documento i;
- Z_i representa o conjunto de classes previstas para o documento i;
- $|Y_i \Delta\ Z_i|$ é a diferença simétrica do conjunto Y_i e Z_i, que é equivalente à disjunção exclusiva (operação XOR) em lógica Booleana.

A medida *Classification Accuracy* [Zhu, et al. 2005] ou *Subset Accuracy* [Ghamrawi e McCallum 2005]:

$$Classification\ Accuracy\ = \frac{1}{|D|}\sum_{i=1}^{|D|} I(\ Z_i = \ Y_i) \tag{6.2.52}$$

Onde:

- $I(\text{verdade}) = 1$ e $I(\text{falso}) = 0$.

Todos os restantes símbolos, têm o mesmo significado exposto na fórmula anterior. Esta medida é muito restritiva pois exige que todo o conjunto de etiquetas previstas esteja correcto.

As medidas seguintes foram usadas em [Godbole e Sarawagi 2004]:

$$Precision = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap\ Z_i|}{|Z_i|} \qquad Recall = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap\ Z_i|}{|Y_i|}$$

$$F_1 = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{2|Y_i \cap\ Z_i|}{|Y_i| + |Z_i|} \qquad Accuracy = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap\ Z_i|}{|Y_i \cup\ Z_i|} \tag{6.2.53}$$

O segundo grupo de medidas, que *Tsoumakas* e *Vlahavas* [Tsoumakas e Vlahavas 2007b] classificaram como medidas baseadas nas etiquetas, podem fazer uso de qualquer medida conhecida de avaliação da classificação binária, como por exemplo, taxa de acerto (*accuracy*), precisão (*precision*), abrangência (*recall*). Podendo estas ser calculadas de forma macro-média e micro-média.

Para se poder fazer uso destas medidas, constrói-se uma matriz binária para cada classe, à semelhança das matrizes exibidas na Tabela *22* (página *70*), podendo-se então, utilizar as medidas reportadas na secção *6.2.2* (página *69*) para problemas de classificação multi-classe, que na prática são as medidas usadas na classificação binária, mas generalizadas para o caso multi-classe.

6.3 Considerações finais

6.3.1 Conhecimentos adquiridos

Neste capítulo, obteve-se o contacto com os conceitos envolvidos nos principais métodos de avaliação dos problemas de classificação, assim como, com as medidas de avaliação usadas nos diferentes tipos de classificação.

Do conhecimento adquirido, destaca-se a sensibilidade ganha no que toca às especificidades da classificação hierárquica e da classificação multi-etiqueta. Esta sensibilidade é importante e útil no momento em que se escolhe as medidas de avaliação, para avaliar o desempenho preditivo de um classificador.

6.3.2 Objectivos atingidos

O objectivo do corrente capítulo, foi investigar os diferentes métodos e medidas de avaliação da classificação hierárquica e da classificação multi-etiqueta.

6.3.3 Principais dificuldades

A principal dificuldade sentida, deveu-se ao facto de não existir ainda um consenso nem uma tendência do processo a seguir para avaliar tanto os problemas de classificação hierárquica como os problemas de classificação multi-etiqueta.

Capítulo 7.
Fase 1 - Recolha e extracção de informação da biblioteca digital ACM

A classificação de textos utilizando técnicas de Aprendizagem Automática assume a existência de uma colecção de documentos já classificados. Nesta secção é apresentado o trabalho de implementação realizado, que permitiu obter uma solução capaz de navegar autonomamente pelas páginas *Web* da biblioteca digital ACM e recolher os dados necessários para construir uma colecção de textos classificados com as categorias definidas no sistema ACM.

Com o sistema implementado, foram recolhidas 112000 páginas *Web*, das quais 106000 correspondem a páginas com informações de documentos científicos, como por exemplo, título, resumo (*abstract*), palavras-chave, termos gerais, classificação ACM, autores, ligação (*link*) para o documento integral em formato PDF (*Adobe's Portable Document Format*), etc.
As aquisições das informações de cada documento foram obtidas: primeiro com a identificação das páginas *Web* relevantes, seguido da sua aquisição (*download*) e extracção das informações.

As informações foram extraídas das páginas *Web* com base em regras escritas manualmente e que reconhecem padrões de informação. As páginas são razoavelmente estruturadas o que facilitam a escrita das regras de extracção de informação. Os valores extraídos foram armazenados numa base dados.

Começa-se por apresentar na secção *7.1* as páginas *Web* e os dados considerados relevantes. Na secção *7.2* (página *92*) apresenta-se a arquitectura da solução encontrada e na secção *7.3* (página *94*) a descrição da implementação recorrendo à linguagem de programação Java.

7.1 Páginas e dados relevantes da biblioteca digital ACM

Para se classificar um documento, é necessário:

- Que as categorias existam previamente;
- Saber quais os elementos que permitem determinar a categoria ou categorias a atribuir, ou no caso da classificação automática, a existência de uma colecção de documentos classificados que é usado na aprendizagem, ou seja na construção de um modelo de classificação.

Relativamente às categorias estas existem e estão bem identificadas, eram elas as definidas pela taxonomia ACM. Quanto à colecção de documentos classificados esta foi construída a partir da biblioteca digital ACM.

Numa análise à biblioteca digital ACM (Figura *31*, página *82*). Os dados inicialmente ai encontrados foram:

- Documentos científicos já classificados segundo a taxonomia ACM, como por exemplo, jornais (*journals*), revistam (*magazines*), *proceedings* entre outros;
- Um índice de palavras-chave (*keywords*) e um índice de nomes próprios (*proper nouns*) que poderiam vir a ser usados para extrair conhecimento.

Na figura seguinte é exibido um mapa de páginas *Web* relevantes para este projecto e as ligações (*links*) entre elas.

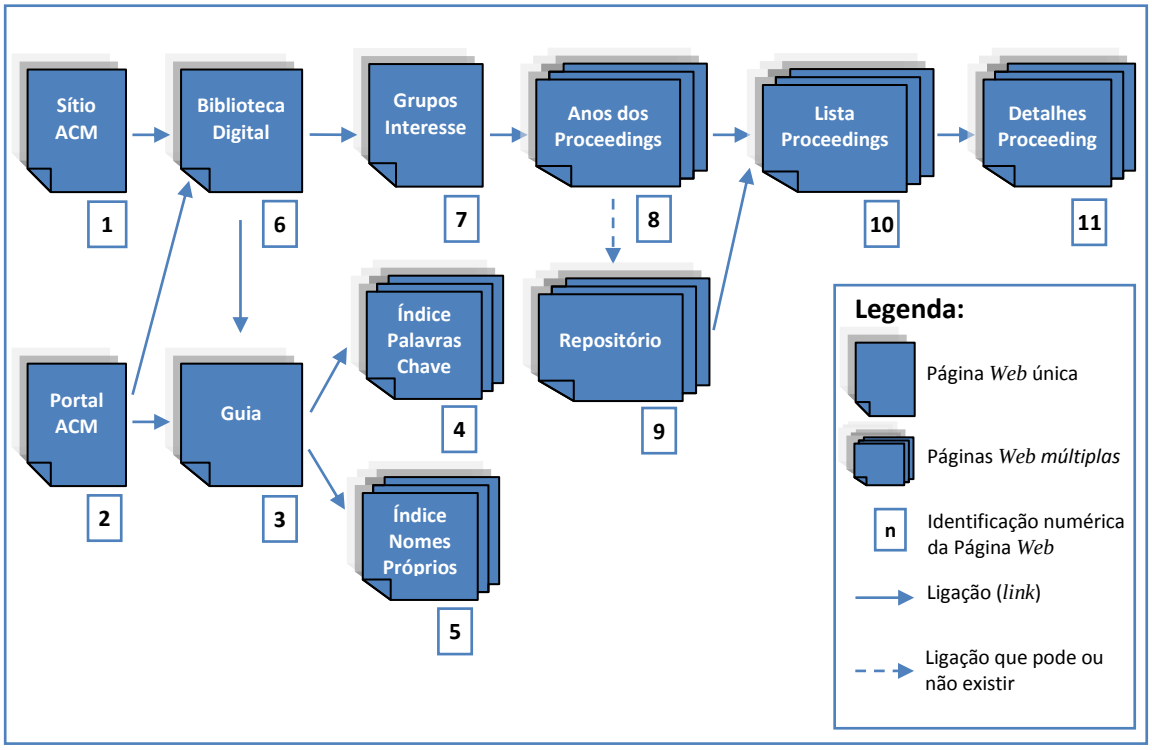


Figura 29 – Mapa das páginas Web relevante para o projecto

Na tabela seguinte é possível saber qual a figura correspondente à página *Web* representada no mapa acima e desse modo visualizar a página *Web* em causa.

Número	Figura
1	Figura 30 – Sítio ACM. Endereço http://www.acm.org, página *81*
2	Figura 31 – Portal ACM. Acesso directo pelo endereço http://www.portal.acm.org/, pág. *82*
3	Figura 32 – Guia para a literatura da computação, página *82*
4	Figura 33 – Índice palavras-chave, página *83*
5	Figura 34 – Índice nomes próprios, página *83*
6	Figura 35 – Biblioteca Digital ACM, página *85*
7	Figura 38 – Grupos de Interesse, página *87*
8	Figura 39 – Anos dos *proceedings* do grupo de interesse A-MOST, página *88*
9	Figura 40 – Repositório de todos os *proceedings* organizados por ano do grupo A-MOST, página *88*
10	Figura 41 – Lista de *proceedings* do ano 2007 do grupo A-MOST, página *89*
11	Figura 42 – Dados de um documento - Conjunto 1, página *91* até Figura 44 – Dados de um documento - Conjunto 3, página *91*

Tabela 23 – Relação entre o número das páginas Web e Figuras

Todas as páginas ilustradas no mapa da Figura *29* (acima), para além das ligações (*links*) representadas, contém ligações para páginas não relevantes para o projecto e por isso não foram representadas. Nas páginas *Web* representadas, podem distinguir-se dois tipos de página quanto à sua estrutura:

- **Página *Web* única**, corresponde a uma página que em toda a biblioteca digital ACM aparece uma única vez, por exemplo, as páginas *Web* 1, 2, 3, 6, etc;
- **Página *Web* múltipla**, significa que existem n páginas ($n > 1$) com estrutura semelhante ou igual, mas com conteúdo diferente.
  Por exemplo, para a página *Web* 4 (Figura *33*, página *83*) que corresponde ao índice de palavras-chave, existe uma página *Web* com todos os termos começados pela letra A, outra com os termos começados pela letra B e assim sucessivamente até à última letra do alfabeto.

Podem distinguir-se dois tipos de página quanto aos fins do seu conteúdo:

- **Páginas *Web* com conteúdo efectivo**, são páginas com informação de interesse para as fases posteriores do projecto. São as páginas *Web* 4, 5 e 11;
- **Páginas *Web* com conteúdo auxiliar**, são páginas com ligações (*links*) que permitem chegar a outras páginas *Web*. São todas as páginas à excepção das 4, 5, e 11.
  Por exemplo, a página *Web* 6 contém uma ligação (*link*) que conduz à página *Web* 7, por sua vez a página 7 tem n ($n > 1$) ligações, cada ligação conduz a uma página do tipo 8.

Passa-se agora à apresentação das páginas representadas no mapa da Figura *29* (página *80*):

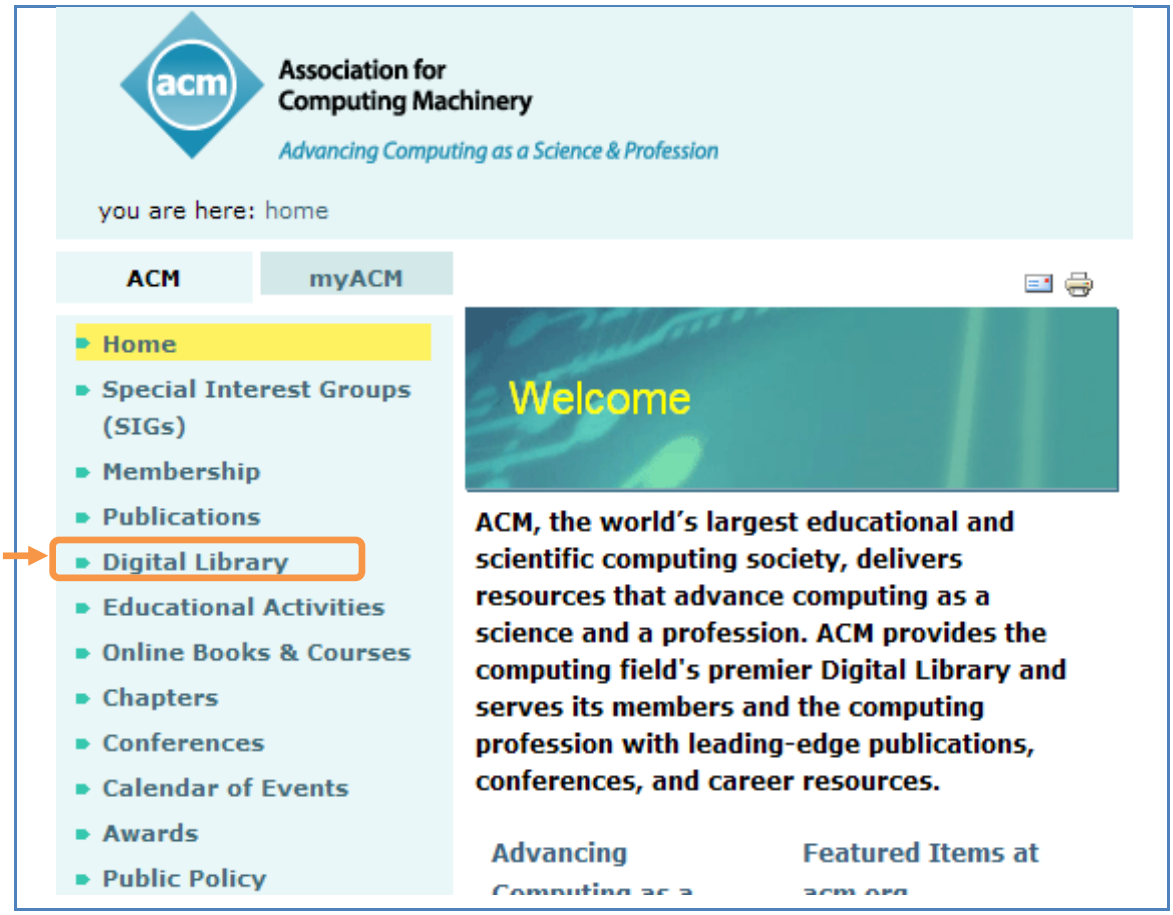


Figura 30 – Sítio ACM. Endereço http://www.acm.org

Figura 31 – Portal ACM. Acesso directo pelo endereço http://www.portal.acm.org/

A página *Web* exibida na figura anterior é uma **página *Web* única**, com **conteúdo auxiliar**.
A informação extraída é:

- Ligação (*link*) que permite aceder à biblioteca digital ACM.
  Tem como etiqueta o texto *Go to The ACM Digital Library*. Permite ir para a página *Web* da Figura *35* (página *85*);
- Ligação (*link*) que permite aceder ao Guia para a literatura da computação.
  Tem como etiqueta o texto *Go to The Guide*. Permite ir para a página *Web* da Figura *32*, (página *82*).

Enquanto na biblioteca digital ACM se tem acesso à totalidade do documento, no Guia para a literatura da computação na grande maioria das situações apenas se acede a um conjunto de informações bibliográficas.

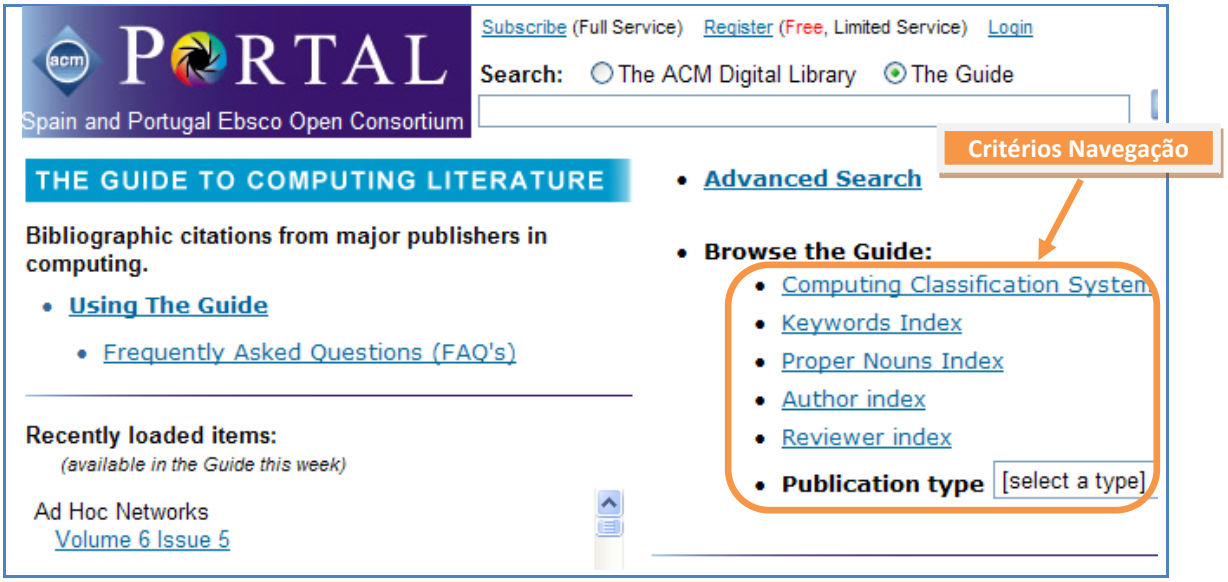


Figura 32 – Guia para a literatura da computação

A página *Web* exibida na figura anterior é uma **página *Web* única**, com **conteúdo auxiliar**.
A informação extraída é:

- Ligação (*link*) que permite aceder ao índice de palavras-chave;

Tem como etiqueta o texto *Keywords Index*. Permite ir para a página *Web* da Figura *33* (página *83*).

- Ligação (*link*) que permite aceder ao índice de nomes próprios.
  Tem como etiqueta o texto *Proper Nouns Index*. Permite ir para a página *Web* da Figura *34* (página *83*).

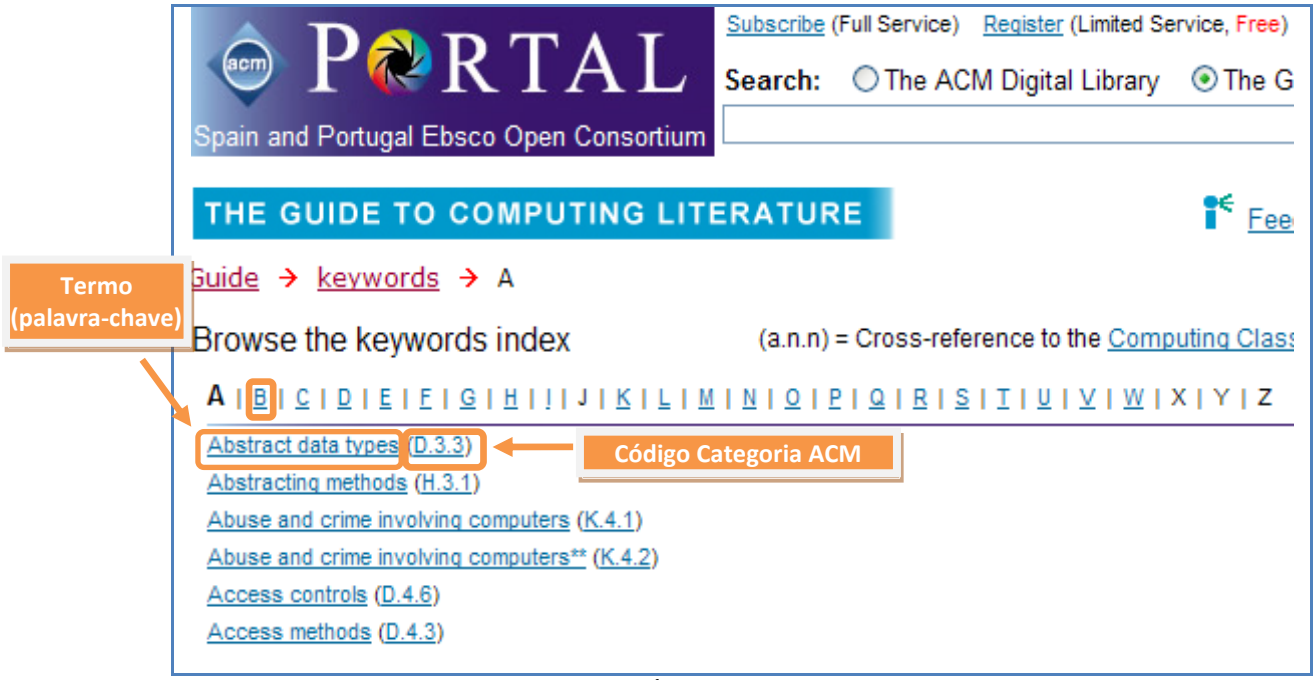


Figura 33 – Índice palavras-chave

A página *Web* exibida na figura anterior é uma **página *Web* múltipla**, com **conteúdo efectivo**.
A informação extraída é:

- Cada uma das palavras-chave e respectivo código da categoria ACM;
  Por exemplo, palavra-chave *Abstract data types* e código *D.3.3*.
- Todas as ligações que permitam aceder às restantes páginas do índice de palavras-chave.
  Por exemplo, a ligação que tem como texto a letra B.

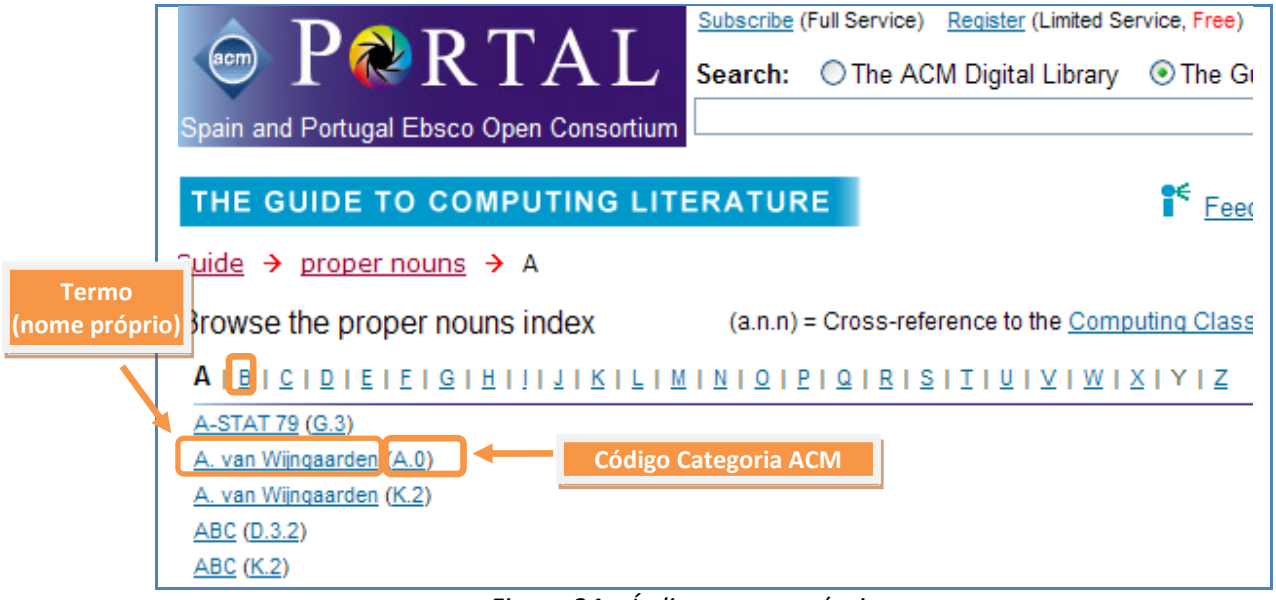


Figura 34 – Índice nomes próprios

A página *Web* exibida na figura anterior é uma **página *Web* múltipla**, com **conteúdo efectivo**.

A informação extraída é:

- Cada um dos nomes próprios e respectivo código da categoria ACM;
  Por exemplo, *A. Vanijngaaden* e código *A.0*.
- Todas as ligações que permitam aceder às restantes páginas do índice de nomes próprios.
  Por exemplo, a ligação que tem como texto a letra B.

Os dados encontrados no índice de palavras-chave e no índice de pronomes próprios, devem ser encarados com precaução, porque:

- Apesar do número total de termos (palavras-chave e nomes próprios) parecer elevado, tendo em conta o número de categorias existentes no terceiro nível da hierarquia do sistema de classificação ACM, intuitivamente se vê que o número médio de termos por categoria (obtido dividindo o número total de termos pelo número total de categorias existentes no terceiro nível) é reduzido;
- Navegando pelos índices, constatou-se que termos frequentemente usados em documentos relacionados com as ciências da computação, não apareciam.
  Para citar alguns: *text data mining*, *text mining*, *knowledge discovery*, *knowledge management*, *summarization*, *ontology*, *clustering* e *Information Retrival* (estes dois últimos ainda aparecem na árvore de classificação os restantes não).
  Esta constatação indiciou que os termos encontrados no índice de palavras-chave e nomes próprios poderiam não ser aqueles usados mais frequentemente e por isso poderiam não ser os melhores representantes dos diferentes documentos.
  Qual poderá ser a explicação para o não aparecimento nos índices de termos usados frequentemente?
  Algumas explicações possíveis mas especulativas, poderiam ser:
    - A não actualização dos índices;
      Esta justificação pode ser difícil de aceitar pois alguns termos são usados à muito tempo (anos), como por exemplo: *Information Retrival*
    - O não querer aumentar demasiado o tamanho dos índices.
      Esta justificação pode ser difícil de aceitar pois os índices em termos absolutos já possuem um tamanho considerável.

Em [Sullivan 2007] é apresentado o estudo do interface do portal ACM e são apontados alguns destes problemas.

Na biblioteca digital ACM (Figura *35*, página *85*), é possível aceder livremente a um conjunto de documentos já classificados.

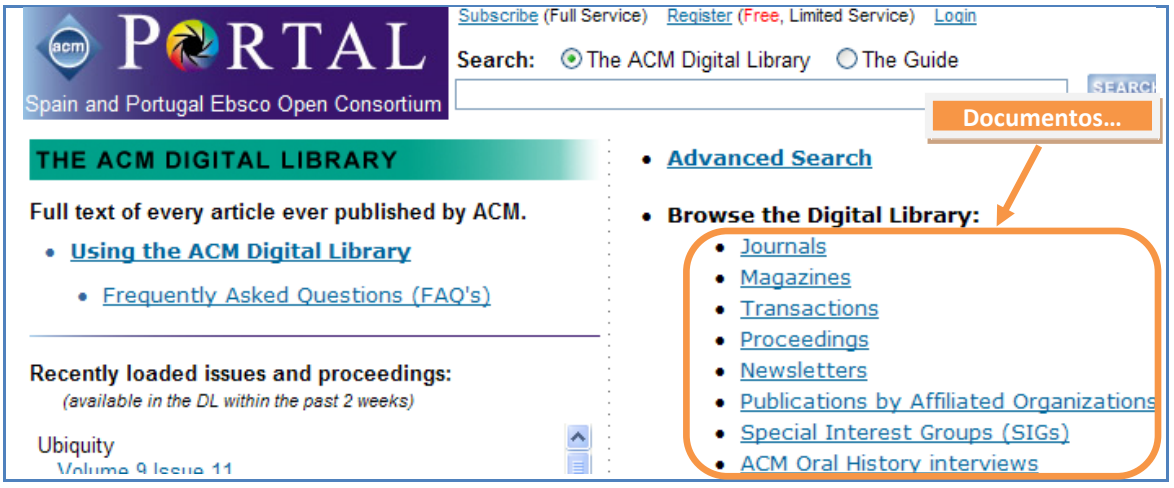


Figura 35 – Biblioteca Digital ACM

A página *Web* exibida na figura anterior é uma **página *Web* única**, com **conteúdo auxiliar**.
A informação extraída é:

- Ligação (*link*) que permite aceder aos *proceedings*, ou dito de forma mais correcta, ligação que permite aceder à página *Web* dos grupos de Interesse. Na Figura *36* (página *85*) é possível ver as páginas a percorrer até atingir a página com as informações de um determinado *proceeding*.

Dos vários tipos de documentos científicos disponibilizados (jornais, revistas, *proceedings*, entre outros), foi decidido utilizar-se os *proceedings*.
A escolha foi feita pelo facto dos *proceedings* serem o tipo de documento que aparecem em maior número, logo, à partida, é possível reunir um número de documentos elevado e representativo das categorias ACM, o que dispensa a recolha de outro tipo de documento.

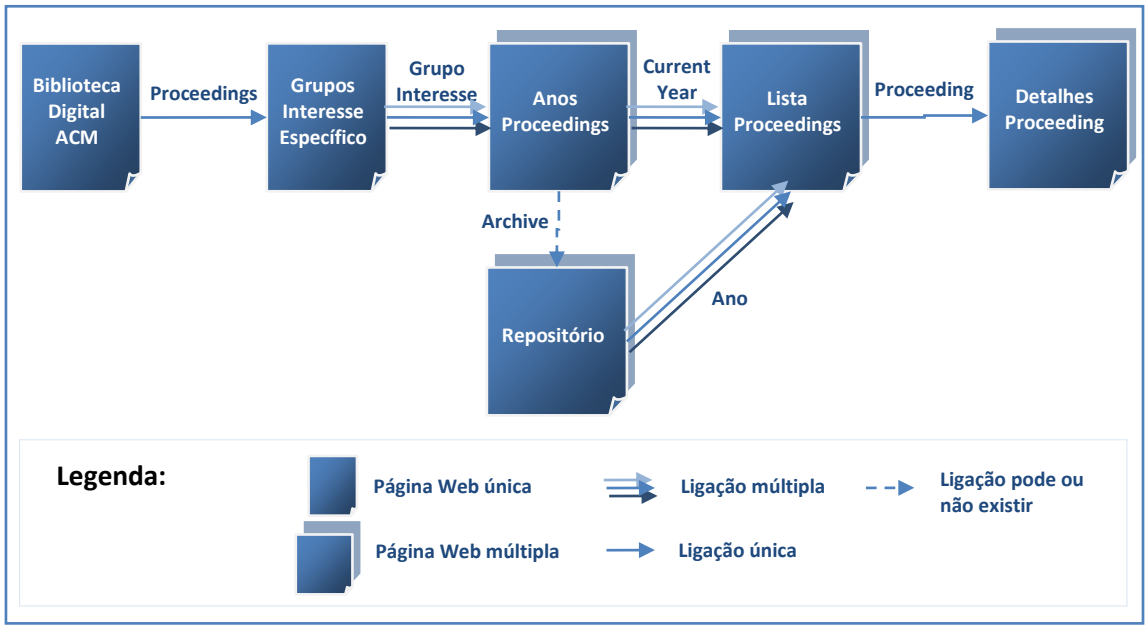


Figura 36 – Páginas a percorrer até se atingir as informações de um proceeding

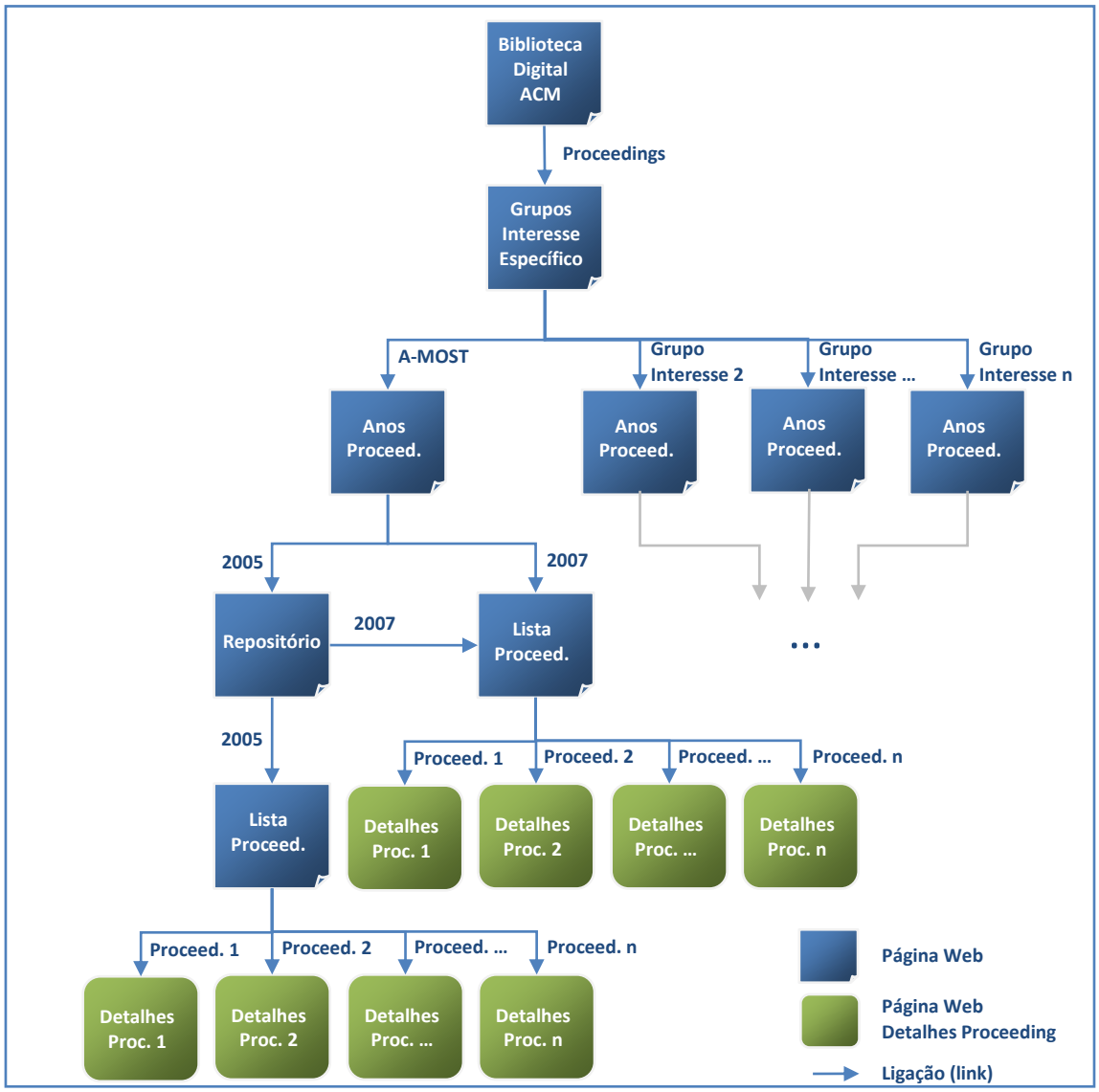


Figura 37 – Páginas a percorrer até se atingir as informações de um proceeding

Para se aceder às informações de um determinado *proceeding é* necessário percorrer um conjunto de páginas, para se aceder a vários *proceeding* o caminho tem que ser percorrido várias vezes.

Na figura anterior é ilustrado o caminho a percorrer para se aceder às informações de um *proceeding*:

1º Clicar na ligação *proceedings* (Figura *35*, página *85*) para ter acesso à página *Web* grupos de interesse específico (Figura *38*, página *87*);

2º Clicar num grupo de interesse específico para ter acesso à página *Web* anos *proceedings* (Figura *39*, página *88*);

3º Clicar no ano desejado (Figura *39*, página *88*). Caso se clique no ano indicado como "*Current Year*" é exibida a lista dos *proceedings* (Figura *41*, página *89*). Caso se clique no ano ou anos indicados abaixo de "*Archive*" é exibida uma espécie de repositório que permite escolher o ano dos *proceedings* que se deseja ver (Figura *40*, página *88*);

4º Estando-se na página repositório, clicar no nome e ano desejado (Figura *40*, página 88), que faz aparecer a página *Web* lista *proceedings* (Figura *41*, página *89*);

5º Estando-se na página lista *proceedings*, clicar na ligação do *proceeding* desejado (Figura *41*, página *89*), que faz aparecer a página *Web* de detalhes do *proceeding* exibida da (Figura *42*, página *90*) à (Figura *44*, página *91*).

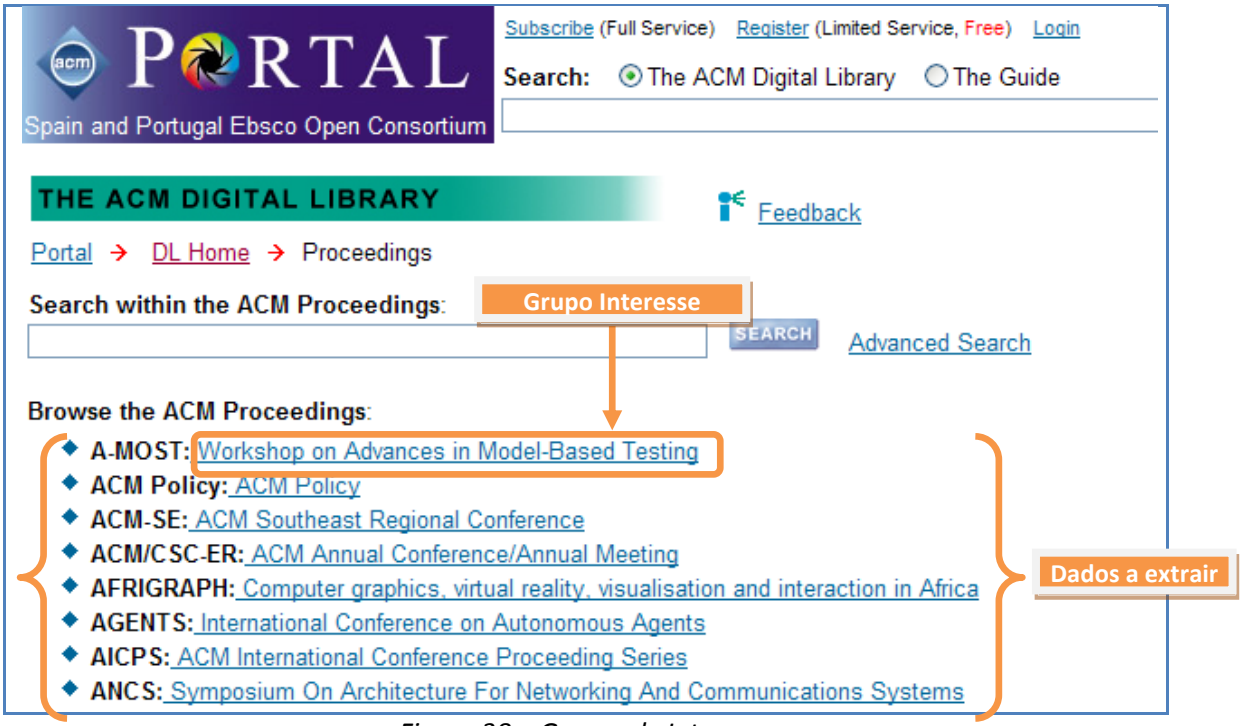


Figura 38 – Grupos de Interesse

A página *Web* exibida na figura anterior é uma **página *Web* única**, com **conteúdo auxiliar**.
A informação extraída é:

- Cada uma das ligações (*link*) que correspondem a um grupo de interesse específico.
  Cada ligação tem como etiqueta (texto) o nome do grupo de interesse específico e permite ir para a página idêntica à da Figura *39* (página *88*).

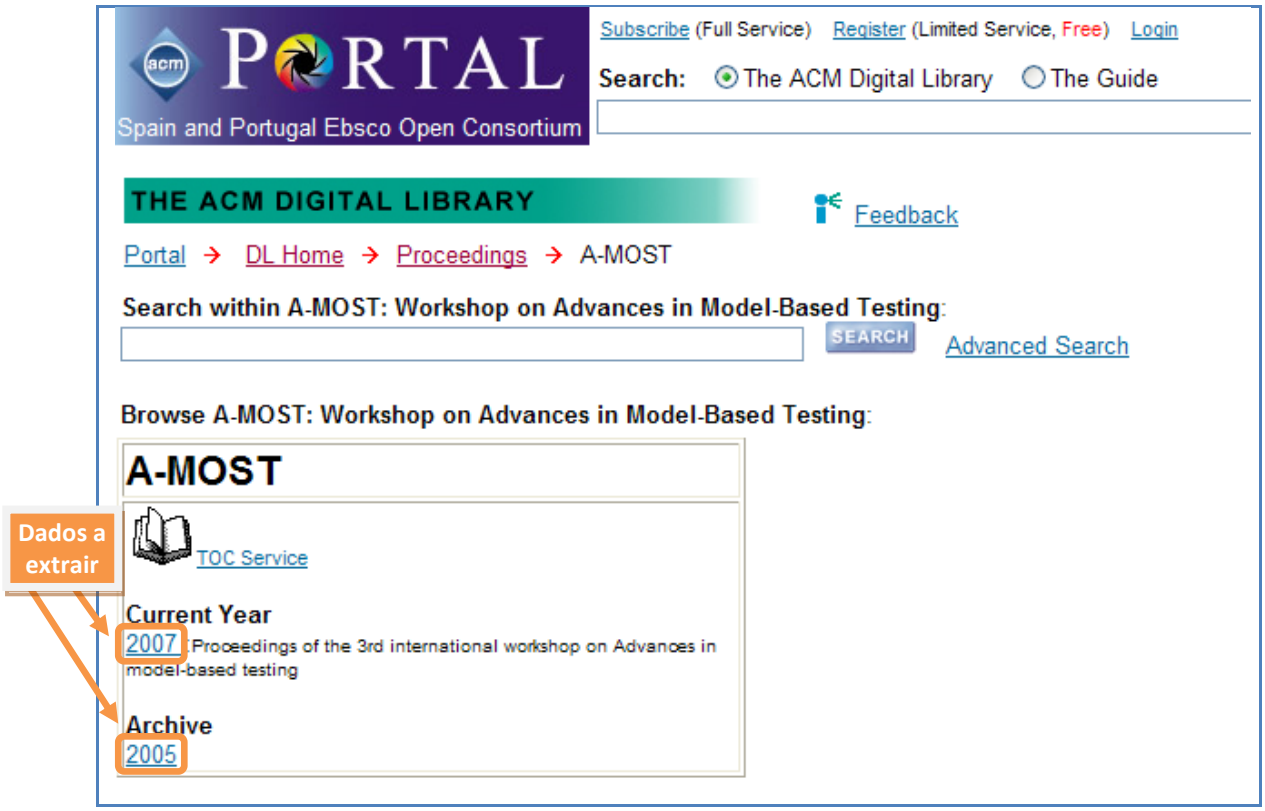


Figura 39 – Anos dos proceedings do grupo de interesse A-MOST

A página *Web* exibida na figura anterior é uma **página *Web* múltipla**, com **conteúdo auxiliar**.
A informação extraída é:

- A ligação (*link*) do ano corrente (caso exista).
  A ligação tem a etiqueta (texto) com o ano actual e permite ir para a lista de *proceedings* desse mesmo ano (Figura *41*, página *89*);
- A ligação (*link*) dos restantes anos (caso exista).
  A ligação tem a etiqueta (texto) um ano ou o primeiro ano seguido do último para os quais existem publicações. Independentemente do texto é exibida a página idêntica à da Figura *41* (página *88*).

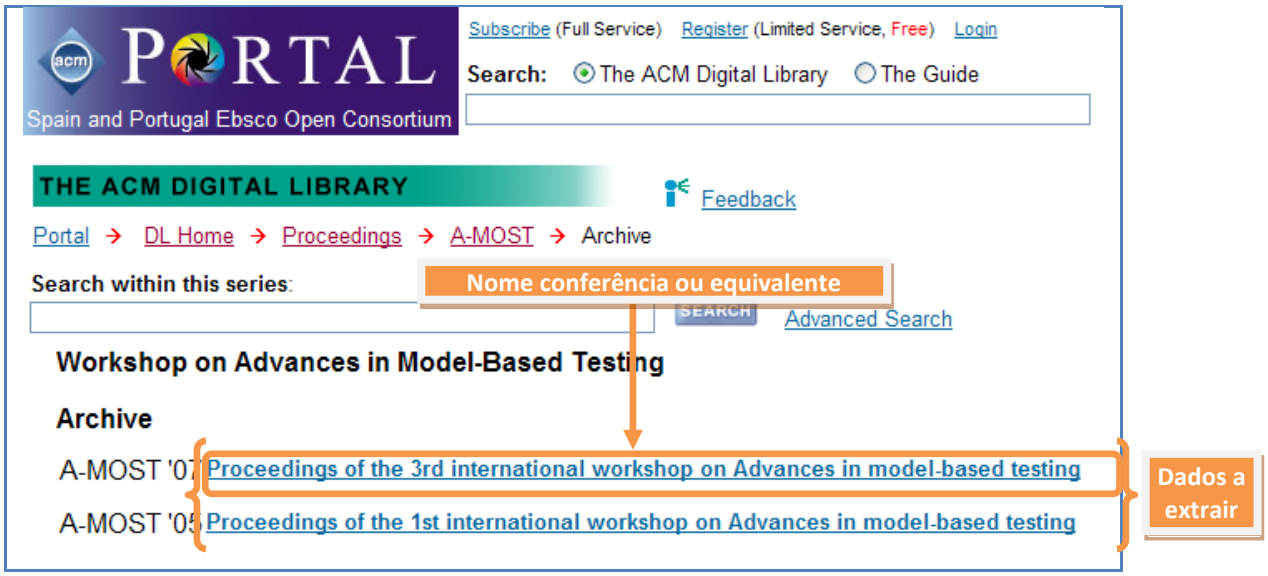


Figura 40 – Repositório de todos os proceedings organizados por ano do grupo A-MOST

A página *Web* exibida na figura anterior é uma **página *Web* múltipla**, com **conteúdo auxiliar**.
A informação extraída é:

- A ligação (*link*) de cada um dos anos existentes.
  A ligação tem a etiqueta (texto) o nome da conferência e permite ir para a lista de *proceedings* dessa conferência (Figura *41*, página *89*).

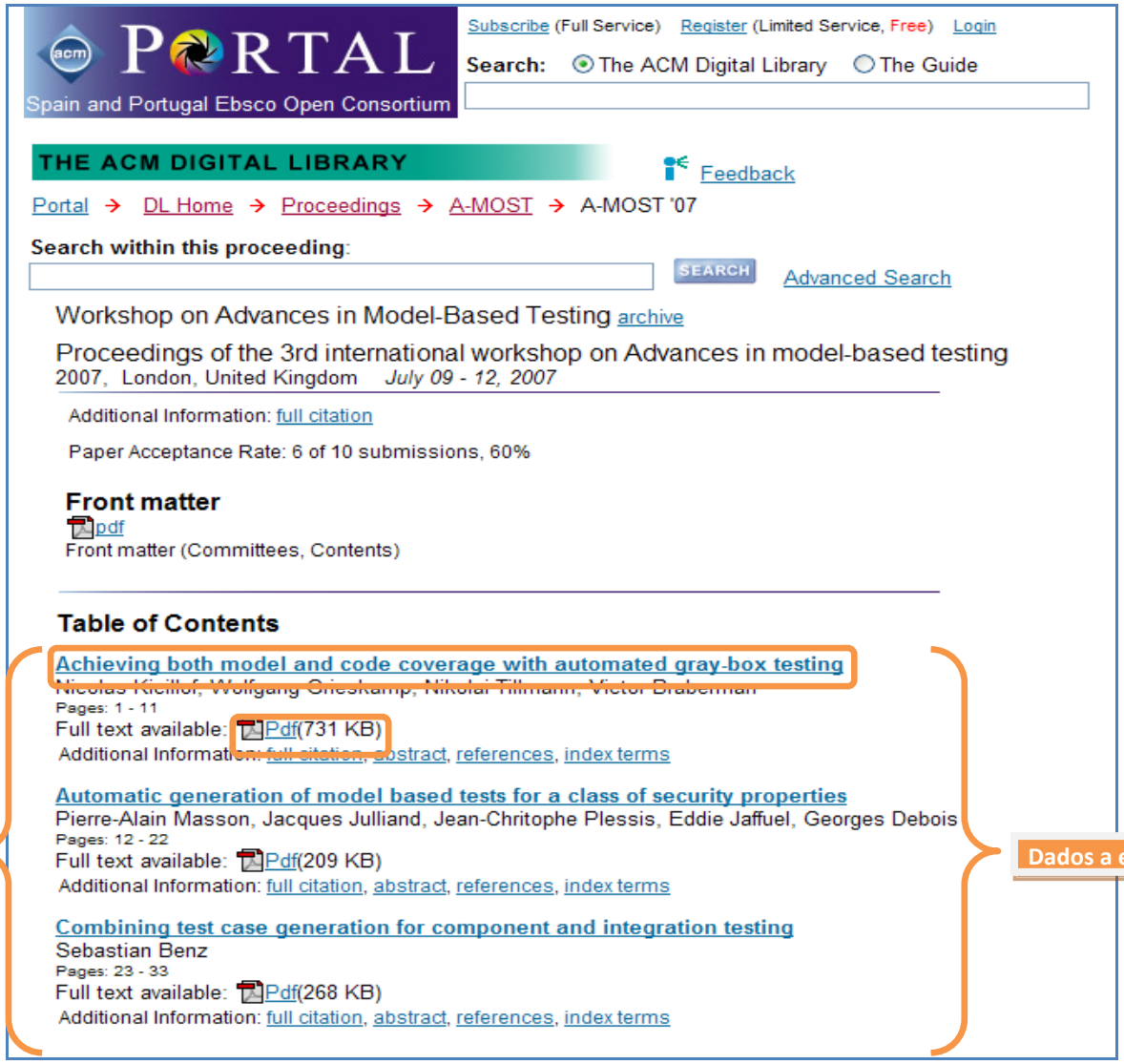


Figura 41 – Lista de proceedings do ano 2007 do grupo A-MOST

A página *Web* exibida na figura anterior é uma **página *Web* múltipla**, com **conteúdo auxiliar**.
A informação extraída é:

- A ligação (*link*) e respectiva etiqueta de cada um dos *proceedings*, assim como o seu tamanho e ligação para o ficheiro PDF que contém a totalidade do documento.
  A ligação de cada *proceeding* tem como etiqueta o nome do *proceeding* e permite ir para a página de detalhes do *proceeding* (da Figura *42*, página *90* à Figura *44*, página *91*).

Spain and Portugal Ebsco Open Consortium

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: The ACM Digital Library The Guide SEARCH

THE ACM DIGITAL LIBRARY Feedback

Portal → DL Home → Proceedings → A-MOST → A-MOST '07 → Citation

Achieving both model and code coverage with automated gray-box testing

Full text Pdf (731 KB)

Source **Workshop on Advances in Model-Based Testing** archive
Proceedings of the 3rd international workshop on Advances in model-based testing table of contents
London, United Kingdom
Pages: 1 - 11
Year of Publication: 2007
ISBN:978-1-59593-850-3

Authors
Nicolas Kicillof FCEyN -UBA
Wolfgang Grieskamp Microsoft Research
Nikolai Tillmann Microsoft Research
Victor Braberman FCEyN- UBA/CONICET

Publisher ACM New York, NY, USA

Bibliometrics Downloads (6 Weeks): 32, Downloads (12 Months): 217, Citation Count: 0

Additional Information: abstract references index terms collaborative colleagues

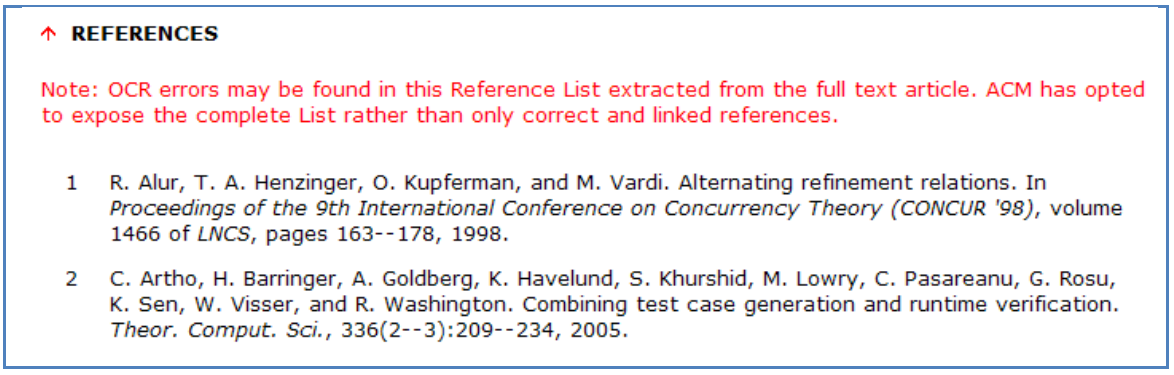
Tools and Actions: Review this Article
Save this Article to a Binder Display Formats: BibTex EndNote ACM Ref

DOI Bookmark: Use this link to bookmark this Article: http://doi.acm.org/10.1145/1291535.1291536
What is a DOI?

ABSTRACT

We have devised a novel technique to automatically generate test cases for a software system, combining black-box model-based testing with white-box parameterized unit testing. The former provides general guidance for the structure of the tests in the form of test sequences, as well as the oracle to check for conformance of an application under test with respect to a behavioral model. The latter finds a set of concrete parameter values that maximize code coverage using symbolic analysis. By applying these techniques together, we can produce test definitions (expressed as code to be run in a test management framework) that exercise all selected paths in the model, while also covering code branches specific to the implementation. These results cannot be obtained from any of the individual approaches alone, as the model cannot predict what values are significant to a particular implementation, while parameterized unit testing requires manually written test sequences and correctness validations. We provide tool support, integrated into our model-based testing tool.

Figura 42 – Dados de um documento - Conjunto 1



REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 R. Alur, T. A. Henzinger, O. Kupferman, and M. Vardi. Alternating refinement relations. In *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR '98)*, volume 1466 of *LNCS*, pages 163--178, 1998.

2 C. Artho, H. Barringer, A. Goldberg, K. Havelund, S. Khurshid, M. Lowry, C. Pasareanu, G. Rosu, K. Sen, W. Visser, and R. Washington. Combining test case generation and runtime verification. *Theor. Comput. Sci.*, 336(2--3):209--234, 2005.

Figura 43 – Dados de um documento - Conjunto 2

Figura 44 – Dados de um documento - Conjunto 3

A página *Web* exibida nas figuras anteriores, corresponde a uma **página *Web* múltipla**, com **conteúdo efectivo**.

A informação extraída é:

- Informação de cada um dos autores;
  (É possível encontrar documentos com um ou mais autores, para cada um deles é extraído o nome, ligação para o seu perfil e nome da instituição)
- Informação do *Publisher*;
- O *abstract*;
- A classificação primária e classificações adicionais (caso existam);
- Os termos gerais (*general terms*);
- As palavras-chave (*keywords*).

Relembro que as informações título, tamanho e ligação para o ficheiro PDF que contém a totalidade do documento foram extraídas na página que contém a lista de *proceedings* (Figura *41*, página *89*).

Das informações extraídas, o título, o *abstract* e as palavras-chave são os mais importantes porque de certa forma resumem o conteúdo do documento, ou dito de outra forma, dão indícios muito fortes do seu conteúdo. A classificação primária e adicional, são importantes porque permitem saber as categorias atribuídas ao documento. Os termos gerais (*general terms*) têm uma importância reduzida porque podem aparecer em qualquer documento e por

isso não são discriminatórios, ou seja, não existem à partida uma relação entre os termos e a categoria ou categorias atribuídas.

Identificadas as páginas e os dados relevantes para o projecto era necessário obter as páginas e extrair o conteúdo relevante. Na secção seguinte é apresentada a arquitectura do sistema que permite dar resposta a estas necessidades.

7.2 Arquitectura do sistema de recolha e extracção de informação

Devido à quantidade de páginas *Web* a obter e ao volume de dados a extrair a solução encontrada foi desenhar e implementar um sistema capaz de navegar pelas páginas *Web* consideradas relevantes e extrair o conteúdo considerado de interesse.

A solução tem a capacidade de navegar pelas páginas *Web* consideradas relevantes (Figura *29*, página *80*) e ignorar as restantes. A solução tem também a capacidade de extrair a informação considerada relevante e ignorar as restantes (assunto abordado na secção *7.3*, página *94*).
A arquitectura desenhada é apresentada na figura seguinte.

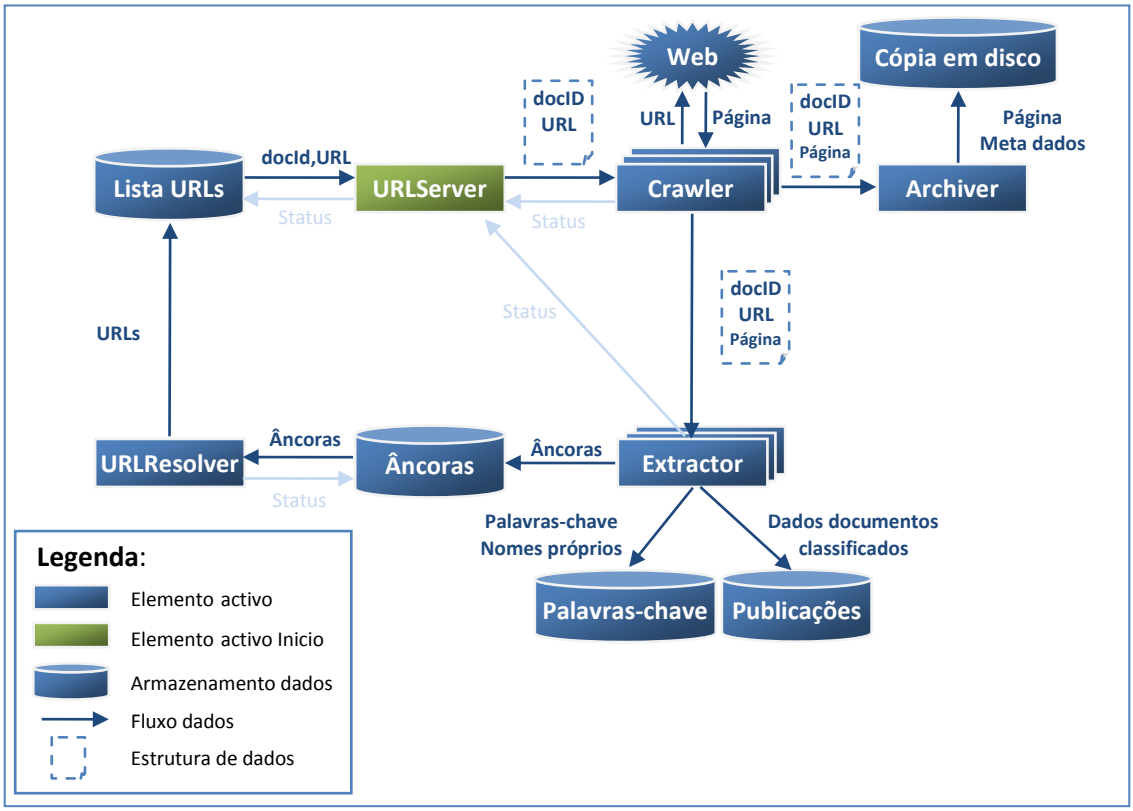


Figura 45 – Arquitectura do sistema de recolha e extracção de informação

A arquitectura desenhada foi inspirada na apresentada na arquitectura do *Google* [Brin e Page 1998] (secção *3.1.2*, página *17*).
Quando comparadas as duas arquitecturas, pode-se constatar que os elementos *URLServer*, *Crawler*, *StorageServer* (agora chamado *Archiver*) e *URLResolver* foram mantidos, o Indexador (*Indexer*) foi substituído pelo Extractor cuja principal diferença é a extracção apenas

dos dados considerados de interesse e não da totalidade. Os restantes elementos desapareceram.

A função de cada um dos elementos activos da arquitectura é a seguinte:

- **URLServer**, seleccionar um conjunto de URLs das páginas *Web* a obter e distribui-los pelos *crawlers*. Cada URL tem um identificador chamado *docId* que identifica unicamente o URL e respectivo documento (página HTML) em todo o sistema.
  Na primeira iteração do sistema a lista de URLs encontra-se vazia e por isso é necessário e suficiente fornecer como semente (*seed*) por exemplo o endereço do portal ACM (Figura *31*, página *82*). O *URLServer* está também encarregue de actualizar o estado (*status*) de um documento sempre que outro elemento o solicite;
- **Crawler**, recolher as páginas *Web* identificadas pelos URLs fornecidos e fornece-las ao *Archiver* e *Extractor*. Indicar o estado da operação de *crawling* ao *URLServer*;
- **Archiver**, compactar e armazenar em disco as páginas *Web* passadas pelo *crawler* e gerar um ficheiro de meta dados para cada página *Web*. Se necessário é possível reconstruir o sistema a partir do repositório sem se recorrer novamente à *Web*;
- **Extractor**, extrair os dados de interesse em função do tipo de página *Web* passada pelo *crawler*. Indicar o estado da operação de *extracção* ao *URLServer*;
- **URLResolver**, fazer o tratamento das ligações (*links*) armazenadas no repositório de âncoras e colocá-las na lista de páginas que devem ser obtidas (feito o *download*), apenas se estas ainda não se encontrarem na lista. Quando as ligações são extraídas de um documento, podem estar na forma de URLs relativos, sendo necessário transformar em URLs absolutos.

Os principais objectivos que levaram à separação das funções por diferentes elementos em vez de ser um único elemento a desempenhar todas as funções, foram:

- Facilitar a distribuição por diferentes máquinas;
- Facilitar o aumento da escalabilidade (aumentar a taxa de operação);
  Por exemplo, é possível aumentar o número de *crawlers* para aumentar o número de páginas recolhidas por unidade de tempo. É possível aumentar o número de extractores para aumentar o número de páginas analisadas por unidade de tempo.
- Permitir corrigir, evoluir, alterar elementos sem implicações directas para os restantes.
  Por exemplo, é possível ter elementos implementados em diferentes linguagens de programação. É possível substituir o *crawler* actual por outro.

Como referido anteriormente, o elemento *Archiver* armazena localmente as páginas *Web* recolhidas, criando um repositório que pode ser usado por exemplo para reconstruir o sistema sem recorrer novamente à *Web*. Nesta situação o modo de funcionamento do sistema é como ilustrado na figura seguinte.

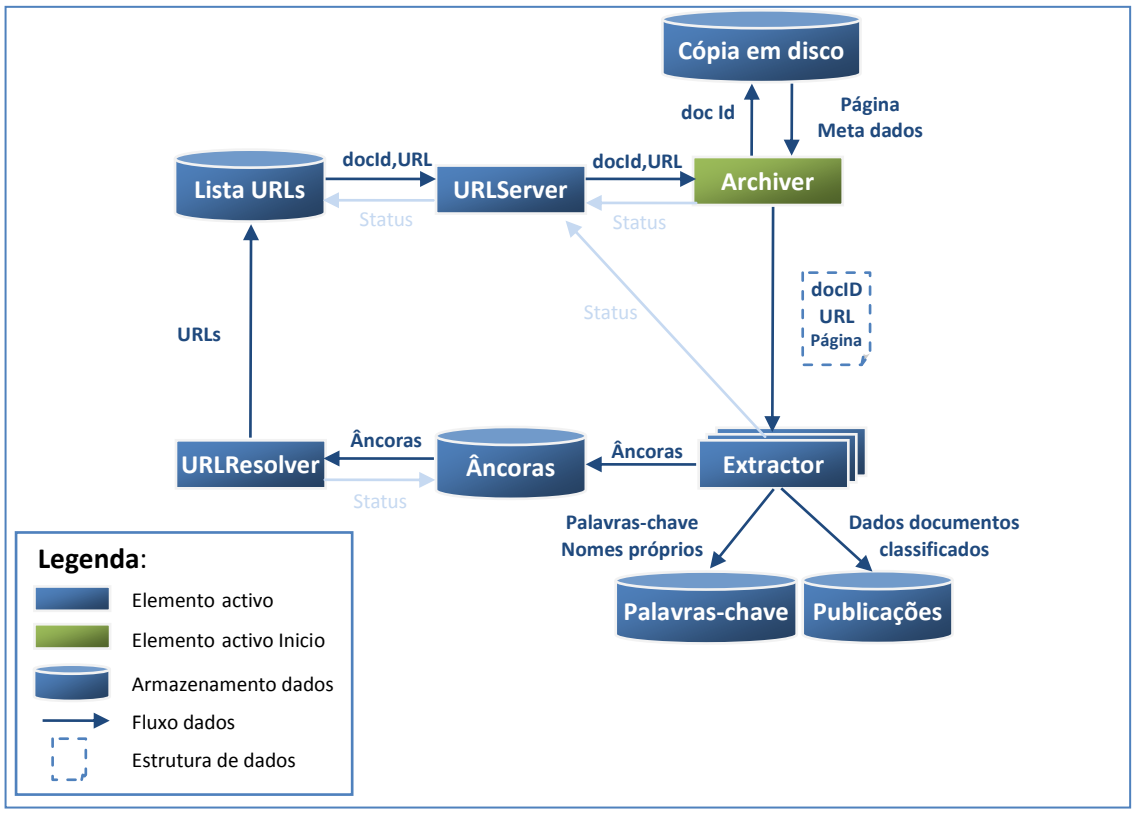


Figura 46 – Arquitectura do sistema a operar a partir da cópia em disco

Neste modo de funcionamento, que poderia ser denominado de **modo *off-line***, as diferenças relativamente ao **modo *online*** (Figura *45*, página *92*) são mínimas e são as seguintes:

- O *Crawler* desaparece, isto é, não é utilizado;
- O arranque do sistema é feito a partir do *Archiver* e não do *URLServer*.

Todos os elementos do sistema têm um papel importante no entanto o *Extractor* merece ser realçado e por essa razão é abordado separadamente na secção seguinte.

7.3 Extracção do conteúdo relevante

O sistema desenhado (Figura *45*, página *92*) e implementado (secção *7.4*, página *96*) efectua basicamente duas operações: navegação pelas páginas *Web* e extracção de informação. Apesar do sistema ter a capacidade de navegar apenas por páginas consideradas relevantes (Figura *29*, página *80*) e ignorar as restantes, a operação pode ser realizada facilmente, recorrendo a uma ferramenta disponível gratuitamente na *Web*, e porque razão é possível recorrer a uma ferramenta genérica para obter este comportamento específico?
Porque na verdade, este comportamento selectivo, acontece devido ao modo de funcionamento do *Extractor*.

Para além de outros dados, o *Extractor* é o responsável por extrair das páginas *Web* as ligações (*links*) e armazená-las num repositório de âncoras (Figura *45*, página *92*). Apesar das páginas conterem várias ligações, apenas são extraídas e armazenadas as relevantes, consequentemente apenas essas chegam ao(s) *crawler(s)*.

A distinção entre ligações relevantes e não relevantes depende da página em questão, mas em todas as páginas é conseguida através da escrita de **regras de extracção de padrões** de informação. As regras são escritas na fase de implementação do sistema, recorrendo às expressões regulares disponíveis na linguagem de programação Java. A abordagem seguida para extracção da informação revelou-se a acertada ao permitir obter os dados pretendidos com um custo de implementação reduzido.
Por exemplo, quando se está perante a página de entrada do portal ACM (Figura *31*, página *82*) de entre as várias ligações, as de interesse são: a que leva à página da biblioteca digital e contém o texto Go to *ACM Digital Library* e a ligação que leva à página do guia e contém o texto *Go to The Guide*. São escritas duas regras, uma que diz "**devem ser extraídas as ligações que contenham o texto** *Go to ACM Digital Library*" e outra que diz "**devem ser extraídas as ligações que contenham o texto *Go to The Guide***".

O exemplo anterior pode levar a pensar-se que em páginas com um número de ligações relevantes elevado, é preciso escrever tantas regras quanto o numero de ligações. Na realidade isto não acontece pois as regras usadas no exemplo anterior, são muito específicas, só foram usadas porque o número de ligações é reduzido e a página é única, em situações não tão simples são usadas regras mais genéricas.
Por exemplo, na página **Grupos de Interesse** (Figura *38*, página *87*), existe um número de ligações relevantes bem maior do que o ilustrado. Nesta página as ligações são extraídas escrevendo uma regra que diz "**devem ser extraídas as ligações que são antecedidas por um texto seguido por : (dois pontos)** ", esta descrição é suficiente para identificar correctamente as ligações de interesse.

As **regras de extracção de padrões** não se esgotam com os exemplos anteriores, no entanto ilustram o princípio básico das restantes.

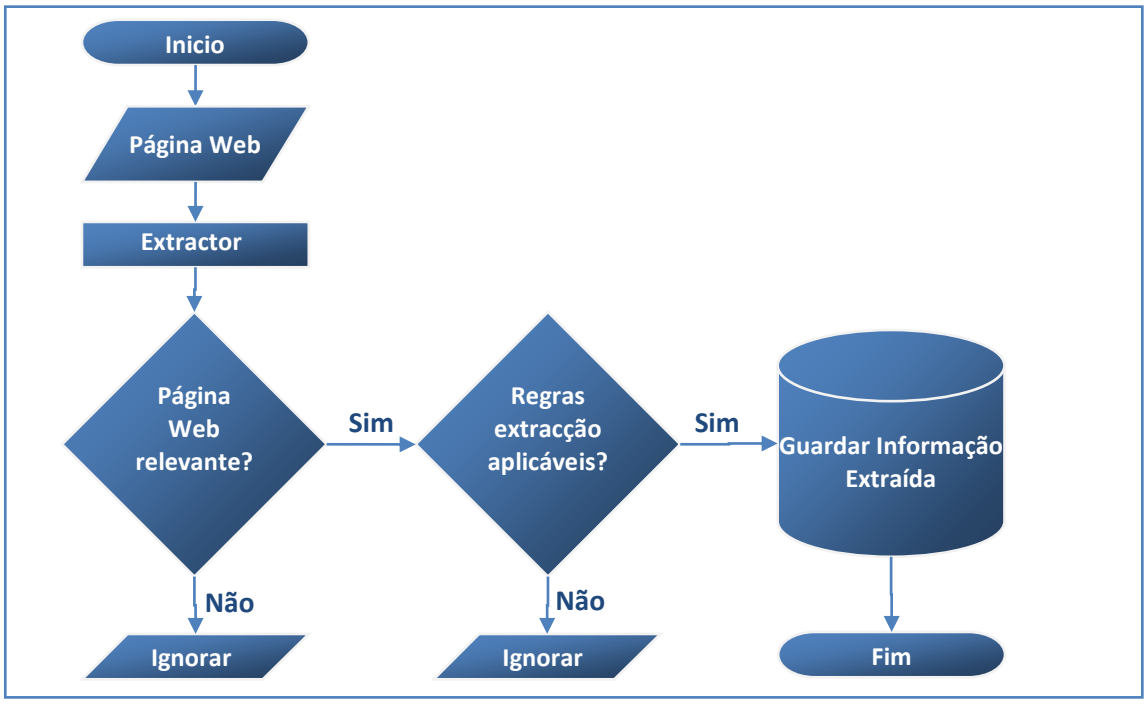


Figura 47 – Processo da identificação das páginas e informação relevante

A figura anterior exibe o processo usado pelo extractor quando recebe uma página *Web*: primeiro verifica se a página é considerada relevante, caso o seja, verifica nas regras de extracção criadas especificamente para a página em questão as que são aplicáveis e armazena a Informação extraída. A informação extraída passa por ligações que permitem chegar a outras páginas, dados de um determinado documento como titulo, resumo (*abstract*), palavras-chave.

7.4 Implementação do sistema de recolha e extracção de informação

O sistema de recolha e extracção de documentos, cuja arquitectura foi exibida na Figura *45* (página *92*) foi desenvolvido na linguagem de programação Java.

Para cada elemento activo: *URLServer*, *Crawler*, *Archiver*, *Extractor* e *URLRosolver* foi criada uma classe Java. Cada classe foi colocada num pacote (*package*) Java. Esta organização modular, permite separar de forma clara, cada um dos elementos do sistema.

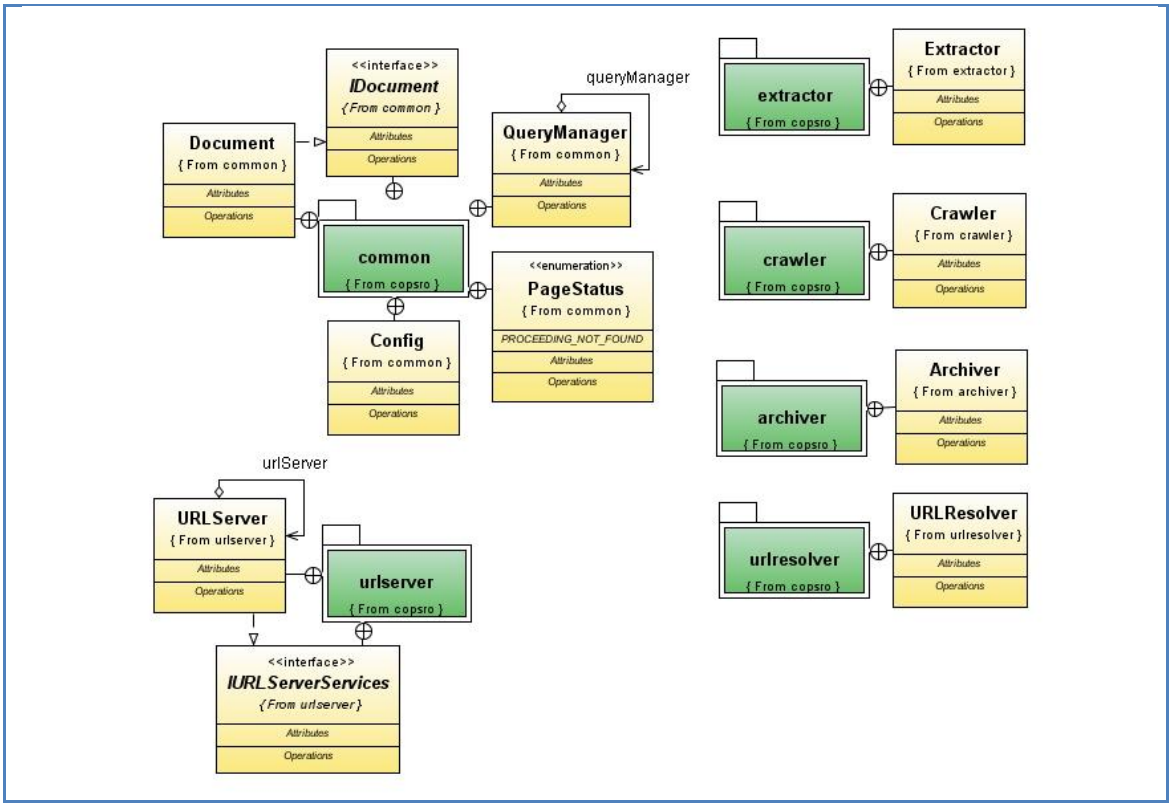


Figura 48 – Distribuição modular dos elementos do sistema

Na implementação realizada, apesar da organização modelar, apenas existe um processo isto é uma unidade de execução. Perante a necessidade ou interesse em aumentar o desempenho por exemplo pela distribuição do sistema por várias máquinas, a organização modelar actual facilita esse objectivo, podendo-se dizer de forma reduzida que apenas é necessário escolher e implementar os mecanismos de comunicação, como por exemplo RMI (*Remote Method Invocation*) no caso concreto da linguagem de programação Java.

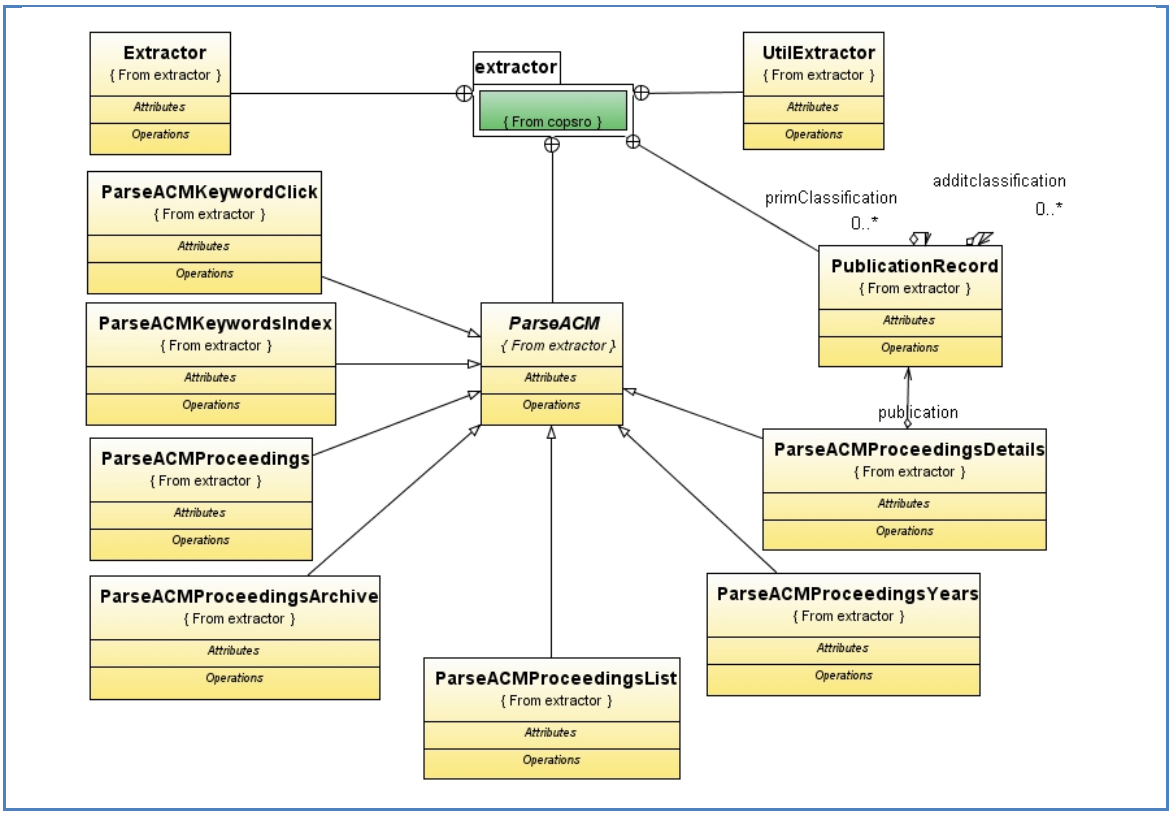


Figura 49 – Classes do Pacote Extractor

7.5 Armazenamento da árvore de classificação ACM noutro formato

A árvore de classificação ACM, é disponibilizada no site da ACM nos seguintes formatos:

- Documento de hipertexto, neste formato todos os nós de um mesmo pai são exibidos como ligações (*links*) na mesma página *Web*, permitindo a navegação pela árvore ACM;
- Ficheiro HTML único, a árvore é exibida integralmente na mesma página *Web*;
- Ficheiro ascii;
- Ficheiro XML (*Extensible Markup Language*).

```
<node id="acmccs98" label="ACMCCS98">
 <isComposedBy>
   <node id="A." label="General Literature">
    <isComposedBy>
     <node id="A.0" label="GENERAL">
      <isComposedBy>
       <node label="Biographies/autobiographies"/>
       <node label="Conference proceedings"/>
       <node label="General literary works"/>
      </isComposedBy>
     </node>
       <node id="A.1" label="INTRODUCTORY AND SURVEY"/>
       <node id="A.2" label="REFERENCE"/>
       <node id="A.m" label="MISCELLANEOUS"/>
  </isComposedBy>
…
```

Figura 50 – Extracto ficheiro XML da árvore de classificação ACM

Dos formatos de ficheiro disponibilizados, o XML é o que permite consultar mais facilmente a árvore ACM recorrendo a uma linguagem de consulta como o *XML-GL*, *XML-QL* e *XQuery*. Contudo a consulta da árvore ACM poderia ser ainda mais fácil se esta estivesse armazenada numa base dados. Como tal, recorrendo à API SAX2 (*Simple API for XML version 2*) procedeu-se ao armazenamento da árvore de classificação ACM em Base Dados.

nodeId	node_type	label	level	parent	note_type	new	related_to
A.	NULL	General Literature	1	acmccs98	NULL	0	NULL
A.0	NULL	GENERAL	2	A.	NULL	0	NULL
A.0.s0	s	Biographies/autobiographies	3	A.0	NULL	0	NULL
A.0.s1	s	Conference proceedings	3	A.0	NULL	0	NULL
A.0.s2	s	General literary works (e.g., fiction, pl...	3	A.0	NULL	0	NULL
A.1	NULL	INTRODUCTORY AND SURVEY	2	A.	NULL	0	NULL
A.2	NULL	REFERENCE (e.g., dictionaries, enc...	2	A.	NULL	0	NULL
A.m	NULL	MISCELLANEOUS	2	A.	NULL	0	NULL

Figura 51 – Extracto da árvore de classificação ACM em Base Dados

7.6 Considerações finais

Como conclusão da primeira fase deste projecto, pode-se dizer que todos os objectivos propostos foram alcançados, destacando o conhecimento adquirido e o resultado final (sistema de navegação e extracção de informação).

7.6.1 Conhecimentos adquiridos

Nesta fase obteve-se o primeiro contacto com novos conceitos e aprofundaram-se outros, de entre os quais se destacam:

- Navegação e obtenção de documentos de forma automatizada;
- Extracção de informação de páginas *Web*;
- Utilização de um *Profiler* (ferramenta que permite avaliar o desempenho e localizar fugas de memória).

7.6.2 Objectivos atingidos

O objectivo desta fase do projecto, foi a aquisição de um conjunto de documentos já classificados segundo a taxonomia ACM. O objectivo foi cumprido, para isso foi necessário:

1. Desenho do sistema de navegação e extracção de informação da *Web*;
2. Implementação do sistema desenhado.

7.6.3 Principais dificuldades

Nesta fase do projecto, sentiu-se dificuldades na procura de documentos classificados segundo a taxonomia ACM. Na fase de implementação sentiram-se algumas dificuldades na extracção de algumas informações dos *proceedings*, na resolução das fugas de memória (*memory leak*) que impediam a aplicação de executar durante longos períodos de tempo. Com muito esforço e empenho as dificuldades foram ultrapassadas, cumprindo assim os objectivos desta fase.

Capítulo 8.
Fase 2 e 3 - Classificação e Avaliação

Com a colecção de documentos construída e reportada no capítulo anterior, foram construídos e avaliados vários classificadores.

Na secção *8.1*, são abordados aspectos relacionados com a classificação ACM que ajudam a compreender determinadas decisões que foram tomadas e relatadas nas secções seguintes.

Na secção *8.2* (página *101*), são apresentadas as colecções de textos utilizadas no estudo experimental realizado.

Na secção *8.3* (página *102*), são apresentados os algoritmos escolhidos e aplicados e indicados os parâmetros utilizados nos algoritmos de classificação aplicados.

Na secção *8.4* (página *103*), é exposta a metodologia aplicada nesta dissertação para resolver o problema de classificação multi-etiqueta hierárquica.

Na secção *8.5* (página *105*), são apresentados e discutidos os aspectos mais relevantes dos resultados obtidos no estudo experimental e comparativo levado a cabo neste projecto.

Por fim, o presente capítulo é encerrado na secção *8.6* (página *110*), onde são destacados os aspectos mais importantes.

8.1 Classificação ACM

A taxonomia ou árvore de categorias ACM tem quatro níveis (Figura *4*, página *5*), cada nó representa uma categoria, podendo cada categoria ser codificada ou não codificada, estas últimas, são chamadas descritores de assunto (para detalhes consultar secção *2.1.1*, página *5*).

A figura seguinte exibe a classificação atribuída a um *proceeding* intitulado *"Intra-Class Utility-Fair Bandwidth Adaptation for Multi-Class Traffic in Wireless Networks"*.

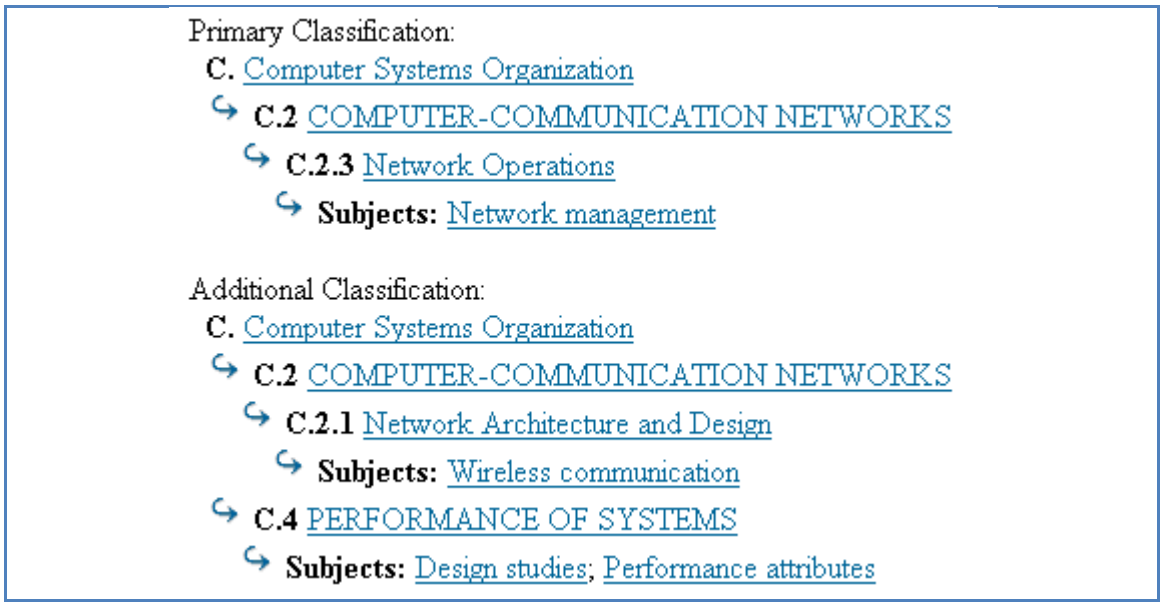


Figura 52 – Exemplo de classificação atribuída a um proceeding

Na figura anterior é exibida uma classificação primária e três adicionais:

- **Classificação primária**

  C. Computer System Organization
  C.2 COMPUTER-COMMUNICATION NETWORKS
  C.2.3 Network Operations
  Network management

A representação da classificação primária é facilmente visível, o mesmo pode não acontecer com as três classificações adicionais, elas são:

- **Classificações adicionais**

  C. Computer System Organization
  C.2 COMPUTER-COMMUNICATION NETWORKS
  C.2.1 Network Architecture and Design
  Wireless communication

  C. Computer System Organization
  C.4 PERFORMANCE OF SYSTEMS
  Design studies

  C. Computer System Organization
  C.4 PERFORMANCE OF SYSTEMS
  Performance attributes

Observa-se assim que uma classificação é um caminho que tem início numa categoria hierarquicamente superior (categoria do primeiro nível) e vai descendo até atingir uma categoria folha.

Dos documentos recolhidos no portal ACM observou-se que um documento pode ter entre zero e sete classificações primárias e entre zero e trinta e seis classificações adicionais. Um número tão elevado de classificações, apenas ocorre em casos extremos, como acontece por

exemplo com o *proceeding "Redesigning a large and complex website: how to begin, and a method for success"* que possui uma classificação primária e trinta e seis classificações adicionais.

Olhando-se para o número de classificações sem se fazer distinção entre classificação primária e adicionais, os documentos recolhidos estão distribuídos da seguinte forma quanto ao número de classificações:

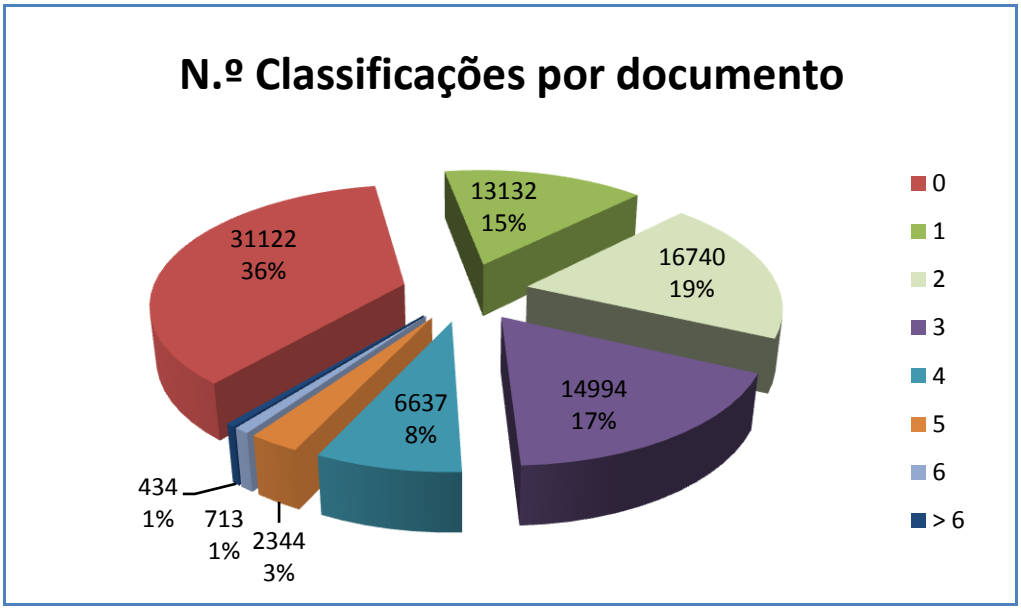


Figura 53 – Número de classificações dos documentos recolhidos na 1ª fase

Dos números acima, destaca-se o facto de 31122 documentos não possui qualquer classificação, 13132 documentos possuírem uma classificação e os restantes possuírem duas ou mais.

8.2 Colecção de documentos

Neste estudo foram usadas duas colecções multi-etiqueta de documentos, uma com 5000 e outra com 10000 documentos. As colecções diferem, no que diz respeito ao número de documentos, número de termos entre outras características que são exibidas na tabela seguinte. Por sua vez, cada documento corresponde a um artigo científico e é composto pelos seguintes dados: titulo, resumo (*abstract*), palavras-chave e respectiva classificação.

	Colecção de documentos	
	Multi-Etiqueta 5000	**Multi-Etiqueta 10000**
Número de documentos	5000	10000
Número total de etiquetas	11306	23786
Número médio de etiquetas por documento	2,2612	2,3786
Número máximo de etiquetas num documento	14	19
Número de termos distintos após remoção de *Stopwords*	11743	16475
Número de termos distintos após remoção de *Stopwords* e poda de termos com frequência < 3	4467	5697
Número médio de documentos por categoria do 1º Nível (11 categorias)	454,5	909
Número médio de documentos por categoria do 2º Nível (81 categorias)	61,7	123,4

Tabela 24 – Colecção de documentos

Em cada uma das tabelas seguintes, encontram-se a distribuição dos documentos por cada uma das categorias (etiquetas) do primeiro nível da árvore ACM. A tabela seguinte, diz respeito à colecção "multi-etiqueta 5000".

A	B	C	D	E	F	G	H	I	J	K
106	531	1651	3137	54	498	226	2393	1586	94	1030

Tabela 25 – Número de documentos por classe do 1º nível colecção, "multi-etiqueta 5000"

A	B	C	D	E	F	G	H	I	J	K
176	2736	3812	6491	112	943	522	4154	2948	131	1761

Tabela 26 – Num. de documentos por classe do 1º nível, colecção "multi-etiqueta 10000"

De notar que um documento possui uma ou mais etiquetas, por outras palavras, um documento pertence a uma ou mais categorias, por esta razão, cada documento contribui como uma unidade para cada etiqueta que possui.

8.3 Algoritmos de classificação utilizados

Os algoritmos de classificação multi-etiqueta utilizados foram o *Binary Relevance* (BR), o *Label Powerset* (LP) e o M*ultilabel k-Nearest Neighbor* (MLkNN) [Zhang e Zhou 2007]. Os dois primeiros são métodos de transformação do problema, enquanto o último é um método de adaptação do algoritmo. Os dois primeiros métodos foram escolhidos porque correspondem às duas abordagens mais básicas para problemas de classificação multi-etiqueta. O MLkNN foi escolhido, como representante mais recente dos métodos de adaptação do algoritmo.

O BR considera a previsão de cada etiqueta como uma tarefa binária independente, corresponde à implementação do método de transformação do problema 4 (TP4), abordado na secção *5.5.1.1* (página *41*). O método LP considera cada subconjunto do conjunto de etiquetas pré-definidas, como uma única etiqueta e tenta aprender um classificador binário para um desses subconjuntos, corresponde à implementação do método de transformação do problema 3 (TP3), abordado na secção *5.5.1.1* (página *41*). Finalmente, o MLkNN foi escolhido, como representante recente de alto desempenho dos métodos de adaptação do problema

[Zhang e Zhou 2007]. Os métodos aplicados encontram-se na API Mulan (*Multi-label classification*), disponível em http://mlkd.csd.auth.gr/multilabel.html.

Os algoritmos BR e LP foram aplicados utilizando como algoritmos de classificação base:

- SMO (*Sequential Minimal Optimization*), para treinar um classificador baseado em vectores de suporte utilizando *kernels* polinomiais [Platt 1998];
- IBK, corresponde à implementação do software Weka do *k Nearest Neighbor Algorithm* [Aha, Kibler e Albert 1991];
- *Naive Bayes Multinomial* [John e Langley 1995];
- SVM (*Support Vector Machine*), utilizando a biblioteca *libSVM* [Chang e Lin 2001];
- J48, um algoritmo de aprendizagem de árvores de decisão, que corresponde à implementação do Weka do conhecido algoritmo C4.5 [J. R. Quinlan 1993].

À excepção do algoritmo SVM, todos os restantes encontram-se no software Weka[33] [Witten e Frank 2005]. Todas as experiências envolvendo o algoritmo IBK, foram realizadas com o valor de K igual a 1 e a 5. Todas as experiências envolvendo o MLkNN, foram realizadas com k igual a 1, 5, 10, 30. Os diferentes valores atribuídos a k, tanto no algoritmo IBK, como no MLkNN, definem o número de documentos vizinhos, para os quais o respectivo algoritmo se baseia para tomar uma decisão. Em todos os restantes algoritmos, os parâmetros utilizados, foram os definidos por defeito nas respectivas APIs.

O software Weka é uma colecção de algoritmos de aprendizagem automática para tarefas de mineração de dados (*data mining*). O software é código aberto e contém ferramentas de pré-processamento de dados, classificação, regressão, agrupamento (*clustering*), regras de associação e de visualização. Os algoritmos e ferramentas podem ser aplicados através de um interface gráfico ou de uma API. No corrente trabalho, foi utilizada a API.

Os três algoritmos multi-etiqueta, aplicados, são algoritmos de classificação plana, como tal, não é possível resolver problemas de classificação hierárquicos, através da sua aplicação directa. Na secção seguinte é apresentada a abordagem para tornar a sua aplicação possível.

8.4 Metodologia para classificação multi-etiqueta hierárquica

A metodologia que foi aplicada nesta dissertação foi:

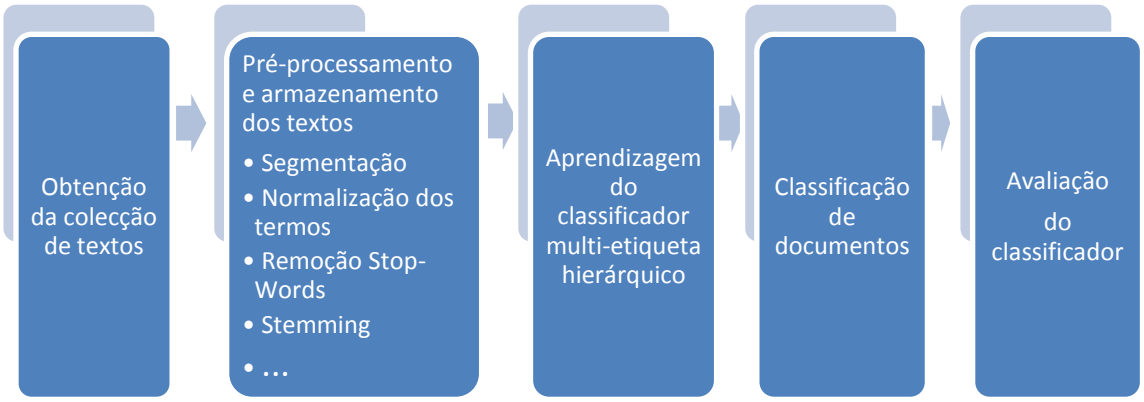


Figura 54 – Etapas da metodologia aplicada

[33] Disponível em http://www.cs.waikato.ac.nz/ml/weka

Etapa de obtenção dos documentos:
Nesta etapa, foi construída uma colecção de documentos com base na solução desenvolvida para automatizar a recolha de documentos da Web e extracção de dados.

Etapa de Pré-processamento e armazenamento dos textos:
Nesta etapa, foi utilizada a API Wvtool 1.1 (*Word Vector Tool*) [Wurst 2007], disponível em http://wvtool.sf.net/. As tarefas de pré-processamento realizadas, foram:

- Divisão dos textos em unidades únicas (procedimento conhecido como *tokenization*), em que todos os caracteres não letras foram assumidos como sendo separadores, resultando unidades (*tokens*) com apenas letras;
- Remoção de *stopwords*, baseadas numa lista de palavras inglesas padrão, incorporadas na API;
- Redução das palavras ao seu radical, pelo processo de *Stemming*, utilizando o algoritmo *Porter Stemmer*;
- Poda (eliminação) das palavras com frequência na colecção inferior a 3. Não se efectuou a eliminação de palavras com frequência superior a um dado valor, porque se assumiu, que parte dessas palavras foram eliminadas na remoção de *stopwords* ou seriam excluídas na selecção dos termos mais importantes (tarefa reportada mais abaixo);
- Representação de cada documento, como um vector de termos, em que cada termo tem um peso associado, segundo a medida tf.idf (*term frequency inverse document frequency*). O vector resultante para cada documento foi normalizado para o comprimento Euclidiano .

Posteriormente a este processo de pré-processamento, para cada conjunto de dados referidos na secção anterior, foram criados subconjuntos com os 1000, 800, 600, 400 e 200 termos mais importantes segundo a medida Ganho de Informação.

Etapa de aprendizagem do classificador:
Nesta etapa, foram aplicados os algoritmos multi-etiqueta planos, seguindo uma abordagem hierárquica local conhecida também como abordagem *top-down level based*, de forma a resolver o problema multi-etiqueta hierárquico, em mãos.

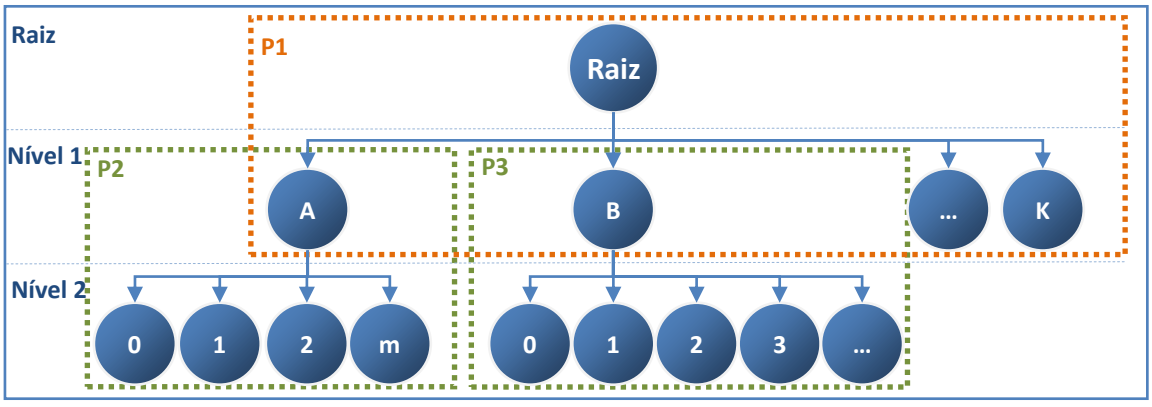


Figura 55 – Abordagem hierárquica local aplicada nesta dissertação

Para explicar a abordagem seguida, supor que se pretende classificar um novo documento. Começa-se por construir um ou mais classificadores no nó raiz (resolver o problema P1), cuja

missão consiste em indicar quais as categorias filho são relevantes para o documento (categorias do A, B, C, …, K do primeiro nível). Supor que foram indicadas como relevantes as categorias A e B. É agora necessário, descer para o nível 2 e construir um ou mais classificadores, em cada uma dessas categorias (revolver separadamente os problemas P2 e P3). Supor que o classificador ou classificadores responsáveis por escolher as categorias filhas de A, indicaram como relevantes a categoria 1 e o classificador ou classificadores responsáveis por escolher as categorias filhas de B, indicaram como relevantes as categorias 3 e 4. Temos então as seguintes classificações atribuídas ao documento: A → 1, B → 3 e B → 4 (de notar que a indicação de cada categoria a que o documento pertence é feita indicando o caminho desde a raiz, visto que se um documento pertence a uma subcategoria, pertence também a todos os seus ascendentes).

Etapa de classificação de documentos:
Nesta etapa, foram utilizados os classificadores criados na etapa anterior para classificar documentos que não foram utilizados na etapa de aprendizagem do classificador.

Etapa de avaliação dos classificadores:
Na etapa de avaliação dos classificadores, foram aplicadas diferentes medidas, com o objectivo de avaliar o desempenho preditivo de cada um deles.

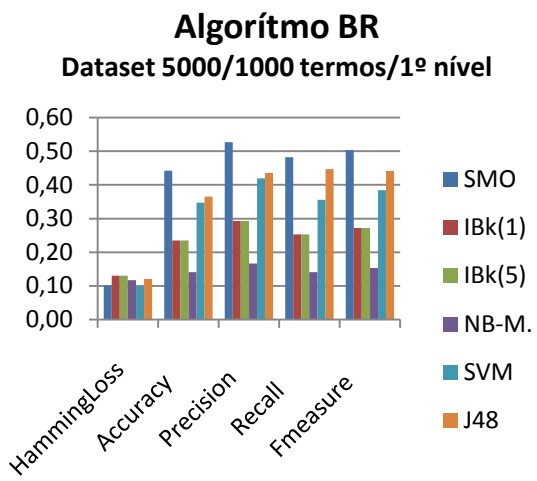
De notar que para cada classificador, as etapas: aprendizagem do classificador, classificação de documentos e avaliação do desempenho de previsão foram realizadas em conjunto e não separadamente. As experiências foram realizadas seguindo o método *k-fold cross validation*, com $k = 3$. A escolha do valor 3 para o k deveu-se ao facto de ser um número popular [A. Clare 2003] (talvez não tão popular como o valor 10 e 5) e o mais adequado, tendo em conta o elevado número de experiências a realizar e tamanho do conjunto de textos. Após a definição do valor de k igual a 3, o conjunto de textos em questão foi particionado em 3, sendo duas das partições utilizadas como conjunto de treino, e a restante como conjunto de teste. O conjunto de treino foi utilizado para aprender um classificador e o conjunto de teste para o avaliar.

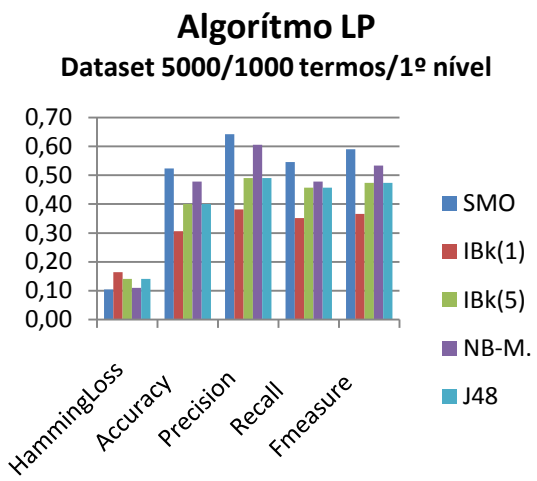
8.5 Resultados e discussão

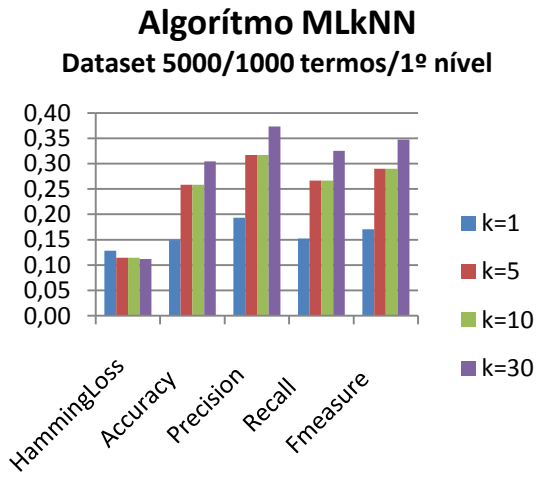
Os resultados obtidos na colecção de documentos "multi-etiqueta 5000" encontram-se resumidos Anexo *2* (página *133*) e os resultados obtidos na colecção de documentos "multi-etiqueta 10000" encontram-se no Anexo *3* (página *137*). Em ambas as colecções, utilizaram-se uma variedade de medidas, algoritmos e número de termos mais importantes, segundo o cálculo do ganho de informação (1000, 800, 600, 400 e 200 foram os números de termos mais importantes escolhidos). Relativamente às categorias da árvore ACM, foram utilizadas as do primeiro e segundo nível, pois a descida ao terceiro nível daria origem a categorias com um número reduzido de documentos.

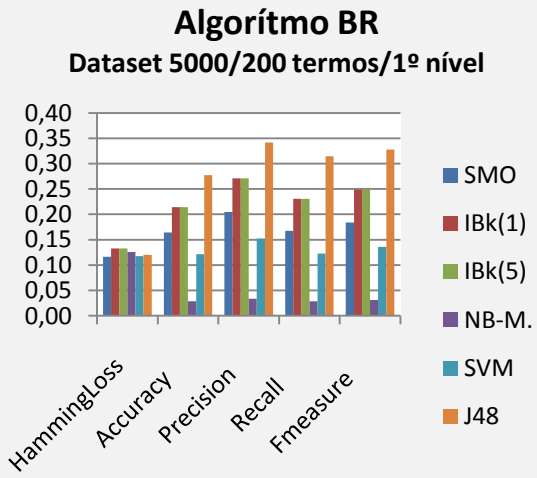
No conjunto de documentos "multi-etiqueta 5000", constatou-se que com a selecção de qualquer número de termos (1000, 800, 600, 400 e 200), o melhor desempenho no primeiro nível foi do algoritmo multi-etiqueta *Label Powerset* (LP) em combinação com o algoritmo SMO, enquanto no segundo nível foi sempre do algoritmo multi-etiqueta *Binary Relevance* (BR) em combinação com o algoritmo *Naive Bayes Multinomial*.

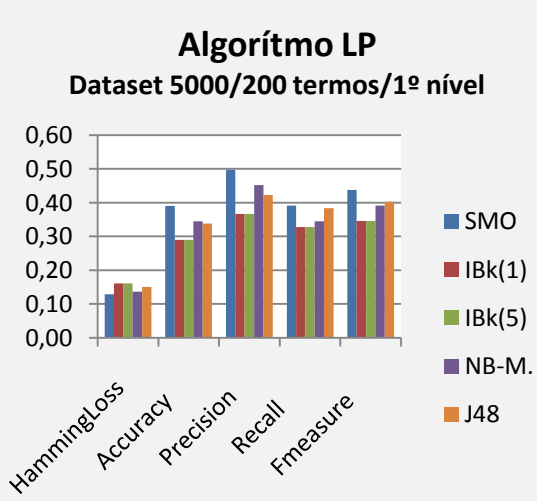
Resultados com a colecção "multi-etiqueta 5000", com 1000 e 200 termos, no 1º nível:











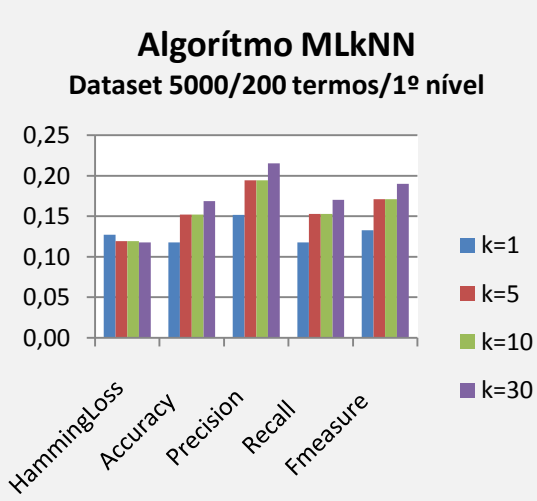
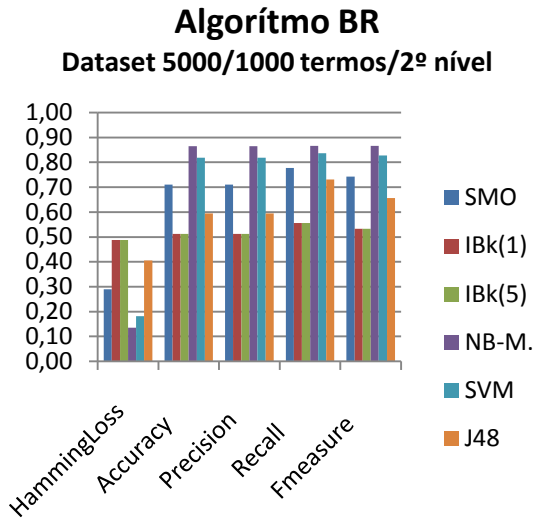
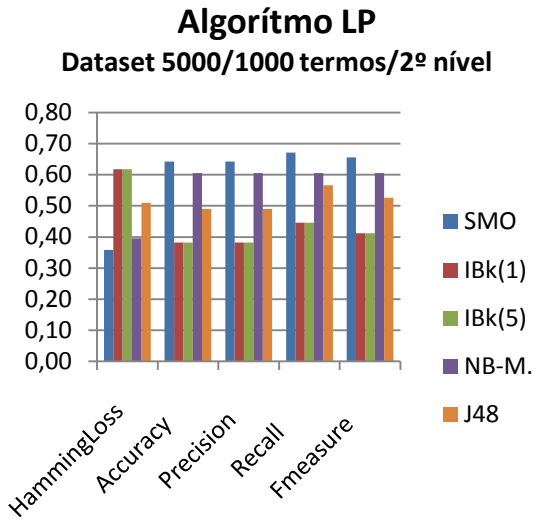
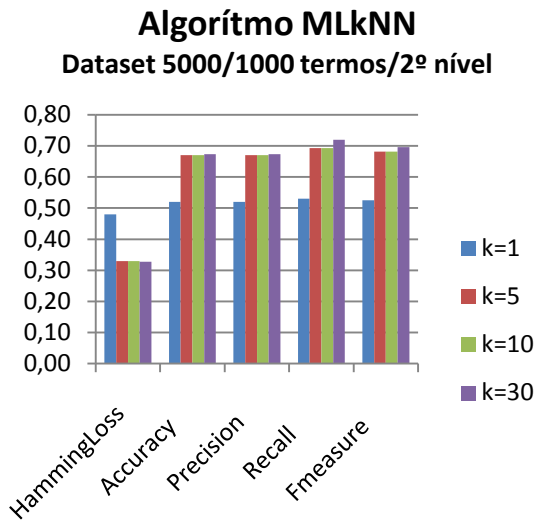


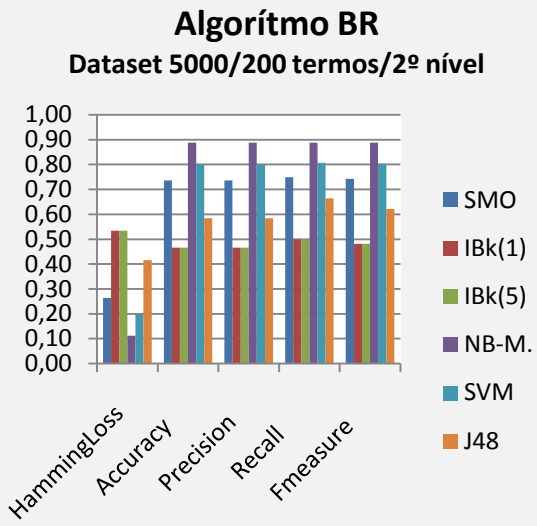
Tabela 27 – Resultados colecção "multi-etiqueta 5000", com 1000 e 200 termos, 1º nível

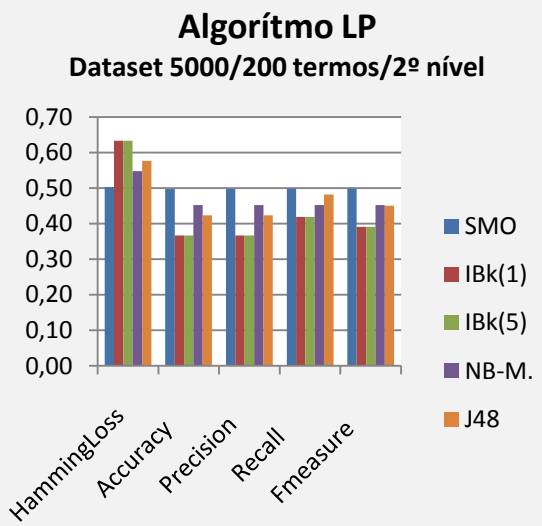
Resultados com a colecção "multi-etiqueta 5000", com 1000 e 200 termos, no 2º nível:











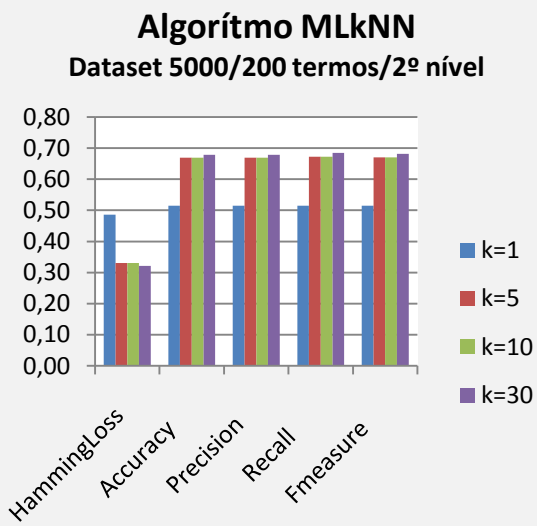
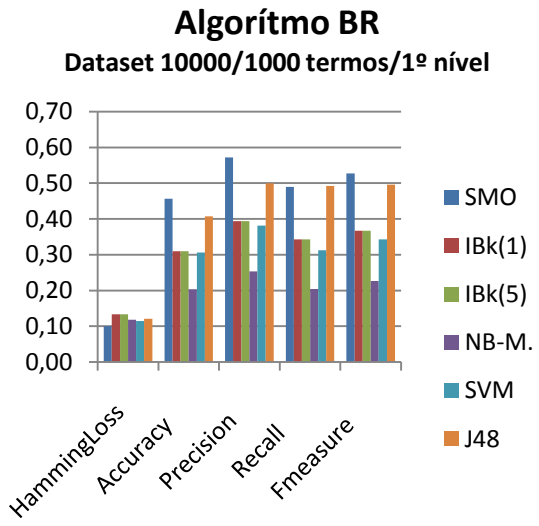
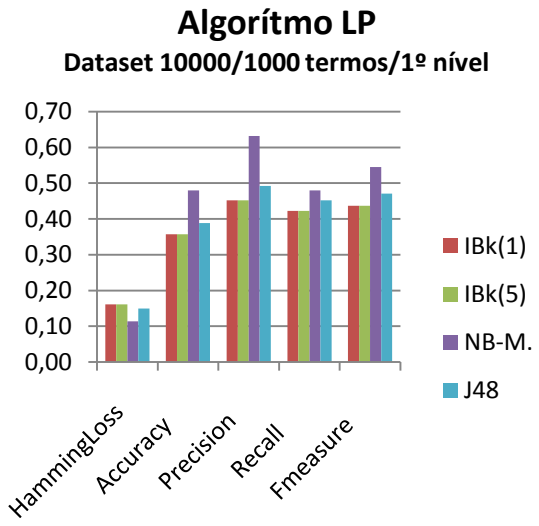


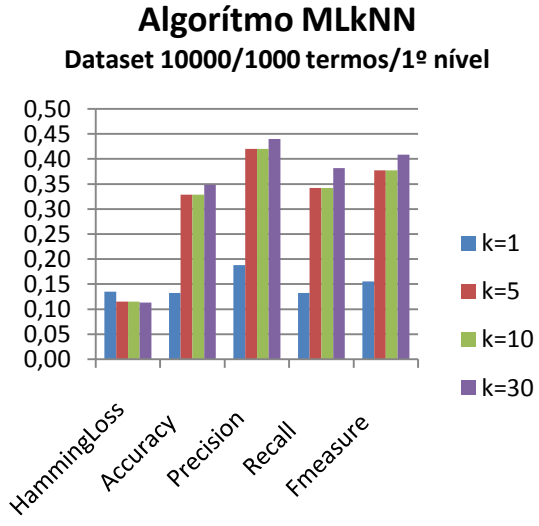
Tabela 28 – Resultados colecção "multi-etiqueta 5000", com 1000 e 200 termos, 2º nível

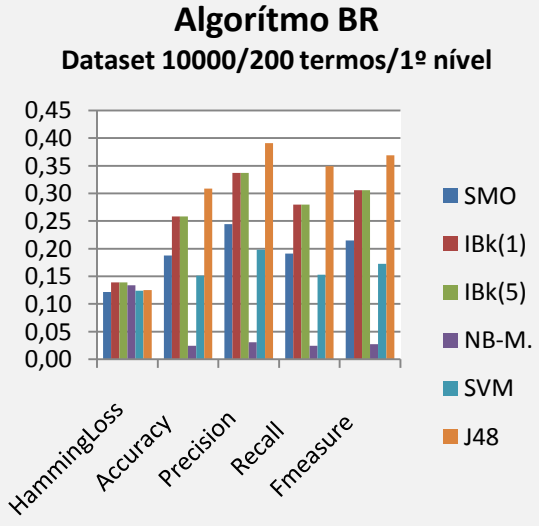
Os gráficos apresentados abaixo, mostram que no conjunto de documentos "multi-etiqueta 10000", constatou-se que com a selecção de qualquer número de termos (1000, 800, 600, 400 e 200), para o caso do primeiro nível, o melhor desempenho é partilhado pelo algoritmo multi-etiqueta *Label Powerset* (LP) em combinação com o *Naive Bayes Multinomial*,pelo algoritmo multi-etiqueta *Label Powerset* (LP) em combinação com o SMO e pelo algoritmo multi-etiqueta *Label Powerset* (LP) em combinação com o J48. Para o segundo nível, constatou-se que com a selecção de qualquer número de termos (1000, 800, 600, 400 e 200), o melhor desempenho é sempre do algoritmo multi-etiqueta *Label Powerset* (LP) com o *Naive Bayes Multinomial*.

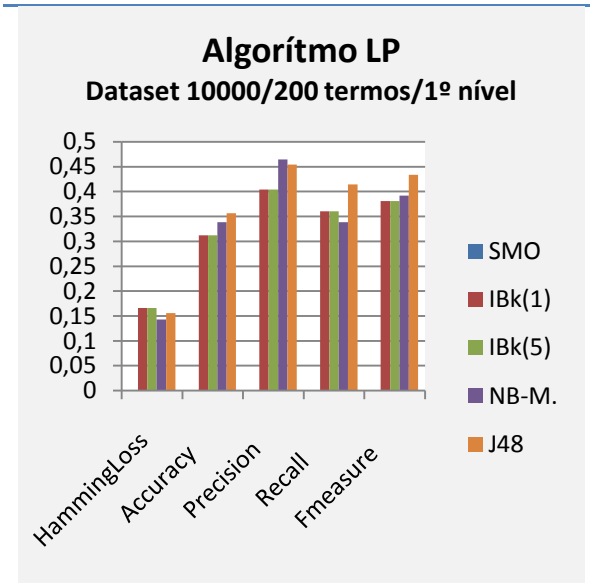
Resultados com a colecção "multi-etiqueta 10000", com 1000 e 200 termos, no 1º nível:











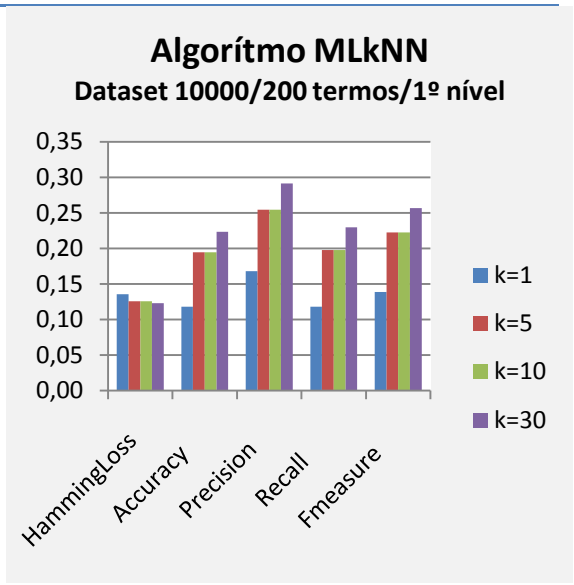
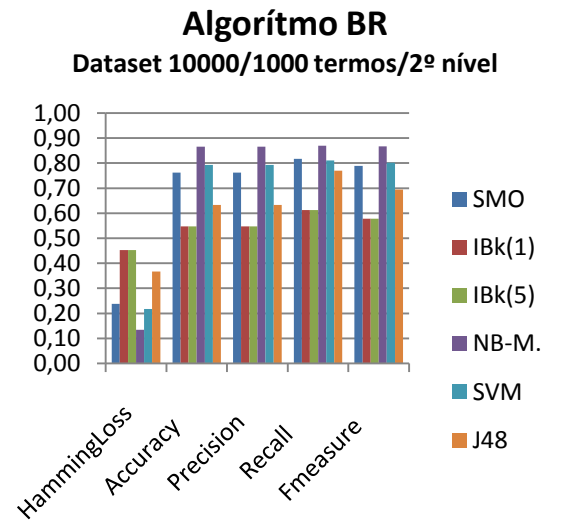
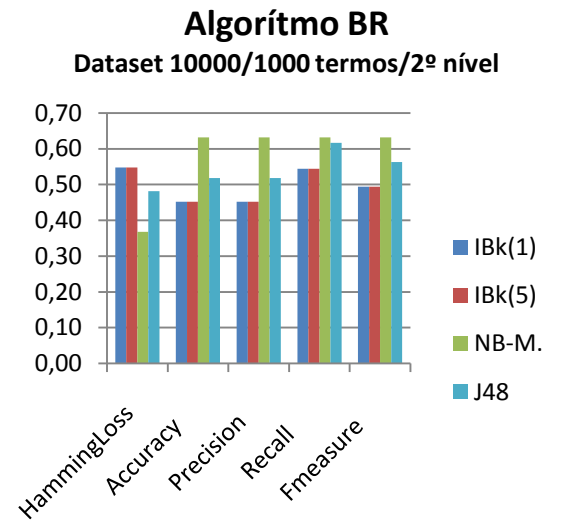
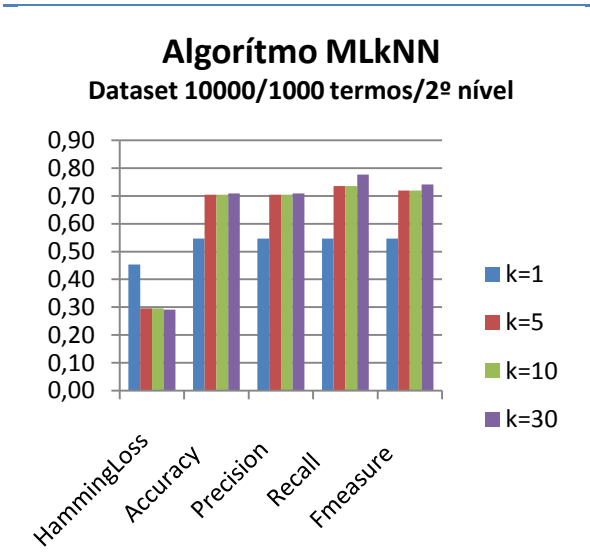


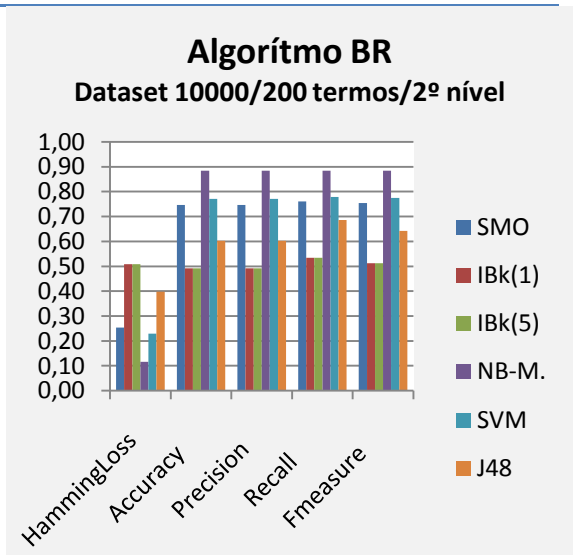
Tabela 29 – Resultados colecção "multi-etiqueta 10000", com 1000 e 200 termos, 1º nível

Resultados com a colecção "multi-etiqueta 10000", com 1000 e 200 termos, no 2º nível:









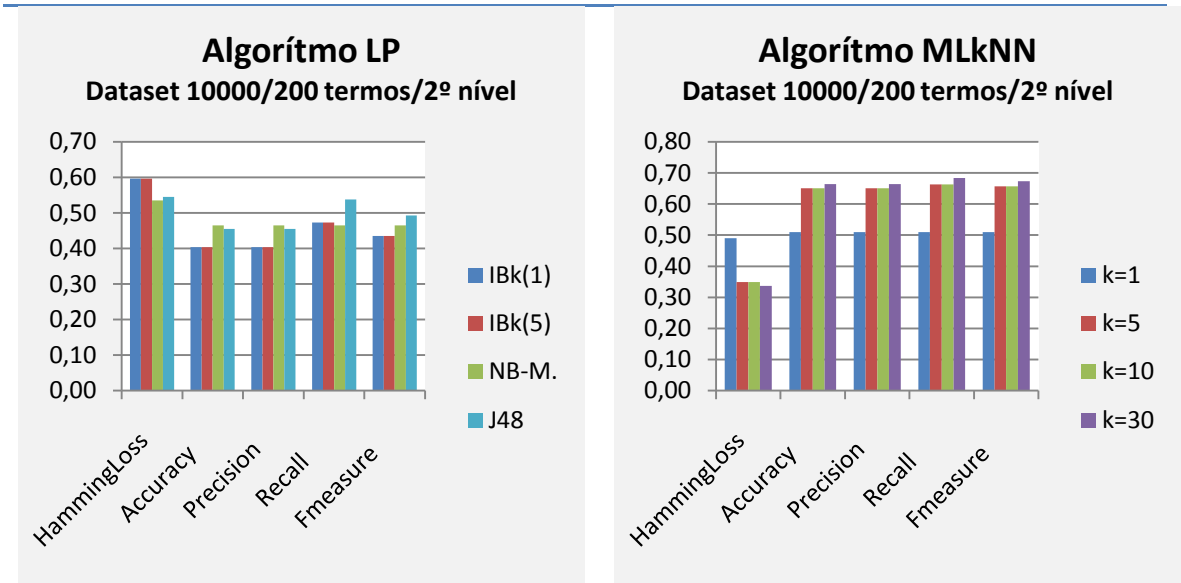


Tabela 30 – Resultados colecção "multi-etiqueta 10000", com 1000 e 200 termos, 2º nível

No Anexo *2* (página *133*) e Anexo *3* (página *137*), encontram-se os resultados exibidos acima em maior detalhe.

Como conclusão final e geral do estudo empírico realizado, observou-se que existe uma margem elevada de progressão com o conjunto de dados em causa. Se por um lado é verdade que os problemas multi-etiqueta hierárquicos têm uma dificuldade superior aos restantes tipos de problemas de classificação, também é verdade que a qualidade do conjunto de dados de treino é importante para a obtenção de bons resultados, neste estudo os resultados não foram mais elevados, possivelmente devido a estes dois factores.

8.6 Considerações finais

Em jeito de conclusão para este capítulo, pode-se dizer que foram cumpridos os objectivos propostos no inicio, destacando o conhecimento adquirido e o resultado final (classificador e sua avaliação).

8.6.1 Conhecimentos adquiridos

Nesta fase colocou-se em prática muitos dos conceitos aprendidos em capítulos anteriores. Foi também, adquirido conhecimento prático em diferentes ferramentas. Dos conhecimentos adquiridos, destacam-se os seguintes:

- Experiência prática, de tarefas de pré-processamento de textos;
- Conhecimento sobre abordagens para lidar tanto com problemas hierárquicos como multi-etiqueta;
- Conhecimento para avaliar classificadores multi-etiqueta;
- Experiência na utilização das APIs Wvtool, Weka e Mulan.

8.6.2 Objectivos atingidos

A fase 2 e Fase 3 do projecto, relatadas no presente capítulo, tiveram como objectivo, em primeiro lugar o estudo experimental de uma abordagem para classificar documentos de texto e em segundo lugar a avaliação da abordagem.

No estudo experimental realizado, a abordagem escolhida para solucionar o problema de classificação multi-etiqueta hierárquica, foi a abordagem local conhecida também como abordagem *top-down level based*, esta foi aplicada, variando-se determinados parâmetros. Das variações efectuadas, destacam-se a aplicação a duas colecções de textos multi-etiqueta (uma com 5000 textos, outra com 10000) e aplicados três algoritmos multi-etiqueta (secção *8.3*, página *102*). Dois dos algoritmos multi-etiqueta, por serem métodos de transformação do problema (secção *5.5.1*, página *41*), foram combinados com vários algoritmos de classificação base (algoritmos conhecidos e utilizados em problemas de classificação binária e multi-classe).

A abordagem aplicada, foi avaliada segundo várias medidas baseadas em exemplos (secção *6.2.4*, página *76*). Os resultados da avaliação podem ser consultados Anexo *2* (página *133*) e Anexo *3* (página *137*) ou de forma resumida na secção *8.5* (página *105*).

8.6.3 Principais dificuldades

No estudo experimental relatado no corrente capítulo, pode-se identificar como principais etapas: o pré-processamento dos documentos, a construção do classificador e a avaliação do mesmo.

O *RapidMiner Community Edition* [Ingo, et al. 2006] e o *Weka*, são softwares, código aberto muito populares, que possuem vários algoritmos e ferramentas de aprendizagem automática (*machine learning*) e *data mining*. Em qualquer dos softwares é possível efectuar as etapas nomeadas no parágrafo anterior, podendo estas ser realizadas através de um Interface gráfico ou programaticamente através das respectivas APIs.

Inicialmente, tentou-se efectuar a etapa de pré-processamento, recorrendo ao Interface Gráfico de cada um dos dois softwares, mas em ambos ocorreu o erro “falta de memória”, o que criou a primeira grande dificuldade. O erro ocorreu, presumidamente, porque ambos os softwares, carregam um texto, fazem a sua segmentação e aplicam as restantes tarefas de pré-processamento, repetindo o carregamento e pré-processamento de toda a colecção de textos, mas mantendo vários dados em memória. Esta dificuldade foi ultrapassada recorrendo à API Wvtool 1.1.

Na etapa de construção do classificador, a dificuldade, prendeu-se com a falta de software para levar a cabo experiências de classificação multi-etiqueta e classificação hierárquica. Por exemplo, o software Weka, que começou a ser desenvolvido em 1993 e possui uma vasta colecção de algoritmos, não possui, na actualidade (2008), algoritmos de classificação multi-etiqueta. Assim como o *RapidMiner Community 4.1*, que também não possui. Esta dificuldade foi ultrapassada utilizando a API Mulan (*Multi-label classification*) em associação com a API Weka.

Na etapa de avaliação do classificador, a dificuldade inicial prendeu-se com a escolha do método de avaliação, pois os métodos mais conhecidos, como acontece com o *holdout*, o *k-fold cross-validation*, o *leave-one-out* e o *bootstrap*, estão pensados para avaliar problemas de classificação plana. Esta dificuldade foi ultrapassada, adaptando-se o método *k-fold cross-validation*. Outra dificuldade sentida nesta etapa, foi a escolha das medidas de avaliação, pois

nos trabalhos encontrados, não existe consenso em relação a que medidas de avaliação aplicar. Esta dificuldade foi ultrapassada, tomando-se a decisão de se aplicar várias medidas baseadas em exemplos (secção *6.2.4*, página *76*). Pode-se dizer que as dificuldades da etapa de avaliação (tanto a escolha do método de avaliação como das medidas), provieram do facto de não existir consenso nem uma tendência do processo de avaliação que deve ser seguido no caso da classificação multi-etiqueta e hierárquica.

Capítulo 9. Conclusões

Neste capítulo faz-se o resumo da experiência adquirida na elaboração do projecto, relatando os objectivos realizados (secção *9.1*), contributos da dissertação (secção *9.2*) e trabalhos futuros (secção *9.3*).

Um tipo de problema de classificação que possua a característica multi-etiqueta é sem dúvida mais complexo de resolver do que um problema uni-etiqueta. Adicionalmente se as categorias se encontram organizadas de forma hierárquica, embora esta organização possa ser favorável à classificação dos documentos, no sentido em que pode ajudar a direccionar a procura, a avaliação é mais complexa como foi demonstrado.

O número reduzido de trabalhos científicos, o aumento da quantidade de informação textual digital, as múltiplas aplicações reais e os desafios colocados pela classificação multi-etiqueta hierárquica de textos, fazem desta área um campo de investigação ainda em aberto.

A motivação da presente tese, surgiu da constatação dos desafios colocados pela classificação multi-etiqueta hierárquica de textos não estarem ainda explorados. Os desafios colocados por este tipo de problemas são ainda muitos, nomeadamente, a falta de algoritmos específicos para o tratamento de problemas de classificação multi-etiqueta e hierárquica, técnicas adequadas para a sua avaliação e a consensualização de uma metodologia adequada a este tipo de problemas.

9.1 Objectivos realizados

O principal objectivo estipulado para o projecto, consistiu em realizar um estudo teórico e empírico da classificação multi-etiqueta hierárquica de textos utilizando as categorias definidas no sistema de classificação ACM.
O projecto foi dividido em três fases:

1. Recolha e extracção de informação de artigos classificados da biblioteca digital ACM;
2. Classificação multi-etiqueta hierárquica de textos;
3. Avaliação do desempenho da abordagem de classificação seguida.

9.1.1 Recolha e extracção de informação de artigos classificados da biblioteca digital ACM

Esta fase, cujo objectivo era construir uma colecção de documentos classificados com as categorias definidas no sistema ACM, foi cumprido na sua totalidade. Para atingir este objectivo foi necessário:

- Aquisição de conhecimento sobre navegação e recolha de dados na *Web* de forma automática;
- Aquisição de conhecimento sobre extracção de informação de documentos;
- Estudo sobre SAX2 (API para manipulação de XML), Jerico 2.5 (*Parser* HTML);

- Uso de ferramentas de monitorização e *profiling* de aplicações Java;
- Conhecimentos da linguagem Java.

O resultado desta fase foi uma solução implementada em Java, que permitiu automatizar a navegação, recolha de documentos e extracção de informação da biblioteca digital ACM, de forma a criar uma colecção de documentos classificados. Outro resultado foi uma solução capaz de ler a árvore de categorias ACM disponibilizada em ficheiro XML e armazena-la em base dados, para uma consulta mais flexível.

9.1.2 Classificação de novos textos

O objectivo desta fase foi investigar a classificação de textos, nomeadamente, as principais tarefas de pré-processamento de documentos e a classificação multi-etiqueta hierárquica de textos. O objectivo foi cumprido, tendo sido necessário:

- A aquisição de conhecimento sobre diversas tarefas de pré-processamento de documentos para a classificação de textos;
- A aquisição de conhecimentos das abordagens seguidas na literatura para lidar com a classificação multi-etiqueta e com a classificação hierárquica;
- Aprender a trabalhar com as APIs Weka, Wvtool e Mulan.

O resultado desta fase foi uma metodologia para a classificação multi-etiqueta hierárquica de textos, que se revelou adequada para este tipo de problemas.

9.1.3 Avaliação do desempenho da abordagem de classificação seguida

O objectivo desta fase foi:

- Investigar os métodos e medidas usadas para avaliar o desempenho preditivo, na classificação hierárquica e classificação multi-etiqueta de textos;
- Avaliar o desempenho preditivo das experiências realizadas nesta dissertação.

9.2 Contributos da dissertação

Todos os objectivos inicialmente propostos foram alcançados. Como principais contribuições pode-se destacar:

- Desenho e implementação de uma solução para recolha de documentos e extracção de dados, que pode ser facilmente adaptada para efectuar a recolha e extracção direccionada noutros domínios (Capítulo *7*, página *79*);
- Criação de uma colecção de documentos de texto para problemas multi-etiqueta hierárquico e disponibilização à comunidade de investigação[34] (Capítulo *7*, página *79*);
- Estudo compreensivo de tarefas de pré-processamento de documentos de textos para classificação de textos (Capítulo *4*, página *21*);
- Estudo compreensivo sobre conceitos, abordagens e medidas de desempenho na classificação multi-etiqueta e hierárquica (Capítulo *5*, página *35*), (Capítulo *6*, página *65*);
- Definição de uma metodologia, que combina abordagens usadas na classificação multi-etiqueta e na classificação hierárquica, para lidar com problemas de classificação multi-etiqueta hierárquica (Capítulo *8*, página *99*);

[34] http://www.dei.isep.ipp.pt/~i000313/ e http://www.psantos.com.pt

- Estudo comparativo do desempenho de previsão de três algoritmos multi-etiqueta, aplicados de forma hierárquica a duas colecções de documentos (Capítulo *8*, página *99*), (Anexo *2*, página *133*), (Anexo *3*, página *137*).

9.3 Trabalho futuro

A classificação de textos tem recebido a atenção dos investigadores, desde meados de 1990, contudo, a classificação multi-etiqueta hierárquica de textos, continua actual e a oferecer desafios aliciantes, que dão espaço para novos trabalhos de investigação, ou optimização de trabalhos já desenvolvidos. Como complemento do trabalho realizado nesta dissertação, a seguir são apresentados algumas propostas para trabalho futuro:

- Disponibilizar publicamente o classificador;
- Investigar formas de recuperar dos erros cometidos num determinado nível, isto é, evitar a propagação dos erros, principal inconveniente da abordagem local *top-down level based*;
- Investigar medidas que permitam obter a similaridade entre classes. Este conhecimento poderia ser usado para aferir a dificuldade de previsão de um problema (classes menos similares, teoricamente são mais fáceis de prever). Seria interessante, investigar como fazer uso do conhecimento dessas medidas, para melhorar as previsões;
- Realizar estudos comparativos e empíricos das medidas de avaliação da classificação multi-etiqueta hierárquica propostas nos últimos anos;
- Realizar experimentos seguindo uma abordagem local *top-down level based*, combinando em cada nível diferentes classificadores utilizando o *Boosting*, *Bagging* ou *Stacking*;
- Desenvolvimento de algoritmos incrementais de classificação multi-etiqueta hierárquica.

  O objectivo seria evitar, ter que se efectuar novamente todo o treino, perante a chegada de novos documentos de treino. O desenvolvimento não só dos algoritmos para classificação, como de outras tarefas, de forma incremental, permitiriam também lidar com volumes de dados (texto) teoricamente infinitos, pois ao contrário dos métodos que necessitam de carregar todos os dados para memória, os métodos incrementais, apenas têm necessidade de ter em memória, um conjunto de dados reduzido;
- Desenvolver um classificador, cujo modelo gerado seja fácil de interpretar (por exemplo, classificador baseado em árvores de decisão e regras de associação).

Bibliografia

ACM CCS98 Intro. “Introduction to the ACM Computing Classification System [1998 Version].” *ACM.* 25 de Abril de 2008. http://www.acm.org/class/1998/ccs98-intro.html [acedido em 25 de Abril de 2008].

Aha, D., D. Kibler, e M. Albert. “Instance-based learning algorithms.” *Machine Learning 6.* 1991. 37–66.

Aiolli, Fabio, Riccardo Cardiny, Fabrizio Sebastiani, e Alessandro Sperduti. “Preferential Text Classification: Learning Algorithms and Evaluation Measures.” *Journal Information Retrieval, forthcoming* [Springer Netherlands], 2008.

Alves, Roberto T., Myriam R. Delgado, e Alex A. Freitas. “Multi-label Hierarchical Classification of Protein Functions with Artificial Immune Systems.” *BSB 2008, LNBI 5167.* Springer-Verlag Berlin Heidelberg 2008, 2008. 1–12.

Ashburner, M. et al. “Gene ontology: Tool for the unification of biology.” *The Gene Ontology Consortium. Nature Genet., 25(1).* 2000. 25-29.

Baeza-Yates, Ricardo, e Berthier Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, 1999.

Baker, L. Douglas, e Andrew Kachites McCallum. “Distributional clustering of words for text classification.” *Proceedings of the Twenty-first ACM International Conference on Research and Development in Information Retrieval (SIGIR98).* 1998. 96–103.

Barutcuoglu, Zafer, Robert E. Schapire, e Olga G. Troyanskaya. “Hierarchical multi-label prediction of gene function.” *BIOINFORMATICS Vol. 22 no. 7 2006.* 2006. 830-836.

Barutcuoglu, Zafer, Robert E. Schapire, Olga G. Troya, e Christopher R. Decoro. “Bayesian Aggregation for Hierarchical Classification.” 2006.

Basili, Roberto, e Alessandro Moschitti. “A robust model for intelligent text classification.” *In proceedings of the thirteenth IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2001).* Dallas, Texas, 2001.

Bekkerman, R. “Word Distributional Clustering for Text Categorization.” MSc Thesis , Technion – Israel Institute of Technology, 2003.

Bekkerman, R., K. Eguchi, e J. Allan. “Unsupervised Non-topical Classification of Documents.” *CIIR Technical Report IR-472.* 2006.

Bekkerman, R., R. El-Yaniv, N. Tishby, e Winter Y. “Distributional Word Clusters vs. Words for Text Categorization.” *In Special Issue on Variable and Feature Selection of Journal of Machine Learning Research (JMLR).* 2003.

Bekkerman, Ron, e James Allan. “Using Bigrams in Text Categorization.” CIIR Technical Report IR-408, 2004.

Bekkerman, Ron, R. El-Yaniv, N. Tishby, e Winter Y. “On Feature Distributional Clustering for Text Categorization.” *In Proceedings of SIGIR 2001.* 2001.

Bergholz, André, e Boris Chidlovskii. “Crawling for Domain-Specific Hidden Web Resources.” *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE’03).* 2003.

Blockeel, H., M. Bruynooghe, S. Dzeroski, J. Ramon, e J. Struyf. “Hierarchical Multi-Classification.” *Proceedings of the SIGKDD Workshop on Multi-Relational Data Mining (MRDM).* 2002. 21-35.

Boutell, M.R., J. Luo, X Shen, e C.M. Brown. “Learning multi-label scene classification.” *Pattern Recognition, vol. 37, no. 9.* 2004. 1757-1771.

Brin, Sergey, e Lawrence Page. “The Anatomy of a Large-Scale Hypertextual Web Search Engine.” *In Proceedings of the seventh international conference on World Wide Web 7 WWW7 / Computer Networks.* 1998. 107-117.

Bush, William S., Todd L. Edwards, e Scott M. Dudek. “Alternative contingency table measures improve the power and detection of multifactor dimensionality reduction.” *BMC Bioinformatics 2008.* 2008.

Butte, A. J., e I. S. Kohane. “Mutual Information Relevance Networks: Functional Genomic Clustering Using Pairwise Entropy Measurements.” *Pacific Symposium on Biocomputing 5.* 2000. 415-426.

Cai, Lijuan. “Multilabel Classification over Category Taxonomies.” Ph.D., Brown University, 2008.

Caropreso, Maria Fernanda, Stan Matwin, e Fabrizio Sebastiani. “A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization.” *In Amita G. Chin (ed.), Text Databases and Document Management: Theory and Practice, Idea Group Publishing, Hershey.* US , 2001. 78-102.

CCS, ACM. “ACM Computing Classification System.” *ACM.* 24 de Abril de 2008. http://oldwww.acm.org/class/ [acedido em 24 de Abril de 2008].

Cesa-Bianchi, Nicolò, Claudio Gentile, A. Tironi, e Lucas Zaniboni. “Incremental algorithms for hierarchical classification.” *Advances in Neural Information Processing Systems 17.* MIT Press, 2005. 233-240.

Cesa-Bianchi, Nicolò, Claudio Gentile, e Luca Zaniboni. “Hierarchical Classification: Combining Bayes with SVM.” *Proceedings of the 23 rd International Conference on Machine Learning.* Pittsburgh, PA, 2006a.

Cesa-Bianchi, Nicolò, Claudio Gentile, e Luca Zaniboni. “Incremental Algorithms for Hierarchical Classification.” *Journal of Machine Learning Research 7*, 2006b: 31–54.

Chakrabarti, S., M. Van Den Berg, e B. Dom. “Focused crawling: A new approach to topic-specific web resource discovery.” *Computer Networks, 31(11–16).* 1999. 1623-1640.

Chang, Chih-Chung, e Chih-Jen Lin. “LIBSVM: a library for support vector machines.” *http://www.csie.ntu.edu.tw/~cjlin/libsvm (Software available at this address).* 2001. [acedido em 1 de Outubro de 2008].

Chaves, Marcirio Silveira. “Um estudo e apreciação sobre algoritmos de stemming para a Língua Portuguesa.” *IX Jornadas Iberoamericanas de Informática.* Cartagena de Indias, Colômbia, 2003.

Church, K., e P. Hanks. “Word association norms, mutual information and lexicography.” *Computational Linguistics, (16).* 1990. 22-29.

Clare, A., e R. King. “Knowledge discovery in multi-label phenotype data.” *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2001).* Freiburg: Germany , 2001. 42-53.

Clare, Amanda. *Machine Learning and Data Mining for Yeast Functional Genomics.* PhD thesis, University of Wales, Aberystwyth, 2003.

Classify, ACM How to. “HOW TO CLASSIFY WORKS USING ACM’S COMPUTING CLASSIFICATION SYSTEM.” *Association for Computing Machinery.* 21 de Abril de 2008. http://www.acm.org/class/how_to_use.html [acedido em 21 de Abril de 2008].

Cohen, William W., e Yoram Singer. “Context-sensitive learning methods for text categorization.” *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1996. 12-20.

Cordeiro, João Paulo da Costa. “Extracção de Elementos Relevantes em Texto/Páginas da World Wide Web.” Tese para obtenção do grau de Mestre, Faculdade de Ciências da Universidade do Porto, Porto, 2003, 174.

Cortes, C., e Vladimir Vapnik. “Support-vector networks.” *Machine Learning, 20.* 1995. 273-297.

Costa, Eduardo de Paula. “Investigação de técnicas de classificação hierárquica para problemas de bioinformática.” Disserção para obtenção do titulo de Mestre em Ciências - Ciências da Computação e Matemática Computacional, Instituto de Ciências Matemáticas e de Computação - ICMC-USP, USP-São Carlos, 2008, 184.

Cover, T. M., e J. A. Thomas. *Elements of information theory.* New York, US: John Wiley & Sons, 1991.

Cover, Thomas M., e Joy A. Thomas. *Elements of Information Theory (Second Edition).* Hoboken, New Jersey: Wiley, 2006.

Crammer, K., e Y. Singer. “A family of additive online algorithms for category ranking.” *Journal of Machine Learning Research 3 .* 2003. 1025-1058.

Debole, Franca, e Fabrizio Sebastiani. “Supervised term weighting for automated text categorization.” *Proceedings of SAC-03, 18th ACM Symposium on Applied Computing.* Melbourne, US, 2003. 784-788.

Dekel, O., J. Keshet, e Y. Singer. “Large margin hierarchical classification.” *In Proceedings of the 21st International Conference on Machine Learning.* Omnipress, 2004.

Denoyer, Ludovic, e Patrick Gallinari. “Bayesian network model for semi-structured document classification.” *Information Processing and Management 40 (2004).* Elsevier, 2004. 807-827.

Diplaris, S., G. Tsoumakas, P. Mitkas, e I. Vlahavas. “Protein Classification with Multiple Algorithms.” *Proceedings of the 10th Panhellenic Conference on Informatics (PCI 2005).* Volos, Greece, 2005.

Directório-Dmoz. 2008. http://www.dmoz.org/ [acedido em Junho de 2008].

Directório-Google. 2008. http://dir.google.com/ [acedido em Junho de 2008].

Directório-Yahoo. 2008. http://dir.yahoo.com/ [acedido em Junho de 2008].

Domingos, Pedro, e Michael Pazzani. “On the optimality of the simple bayesian classifier under zero-one loss.” *Machine Learning.* 1997. 103–130.

Dumais, S., e H. Chen. “Hierarchical classification of Web content.” *Proc. of the 23rd ACM Int. Conf. on Research and Development in Information Retrieval.* Athens, GR, 2000. 256–263.

Dumais, Susan T., John Platt, David Heckerman, e Mehran Sahami. “Inductive learning algorithms and representations for text categorization.” *Proceedings of the Seventh International Conference on Information and Knowledge Management (ACM-CIKM98).* 1998. 148–155.

Eikvil, Line. “Information Extraction From World Wide Web - A Survey -.” 1999.

Elisseeff, A., e J. Weston. “A kernel method for multi-labelled classification.” *Advances in Neural Information Processing Systems 14.* 2002.

Escudeiro, Nuno Filipe Fonseca Vasconcelos. “AUTOMATIC WEB RESOURCE COMPILATION USING DATA MINING.” Tese de Mestrado em Análise de Dados e Sistemas de Apoio à Decisão, Faculdade de Economia da Universidade do Porto, Porto, 2004, 167.

Esuli, Andrea, Tiziano Fagni, e Fabrizio Sebastiani. “Boosting multi-label hierarchical text categorization.” *Inf Retrieval (2008) 11.* Springer Science+Business Media, 2008. 287-313.

Esuli, Andrea, Tiziano Fagni, e Fabrizio Sebastiani. “TreeBoost.MH: A Boosting Algorithm for Multilabel Hierarchical Text Categorization.” *SPIRE 2006 - Thirteenth edition of the Symposium on String Processing and Information Retrieval.* 2006.

Fano, R. *Transmission of information.* Cambridge: MIT Press, 1961.

Freund, Y., e R. E. Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting.” *Journal of Computer and System Sciences, vol. 55, no. 1.* 1997. 119-39.

Freund, Y., e R. Schapire. “Experiments with a New Boosting Algorithm.” *In Proceedings of the 13th International Conference on Machine Learning.* 1996 . 148-156.

Fuhr, N., S. Hartmann, G. Knorz, G. Lustig, M. Schwantner, e K. Tzeras. “AIR/X – a rule-based multistage indexing system for large subject fields.” *In Proceedings of RIAO-91, 3rd International Conference “Recherche d’Information Assistee par Ordinateur”.* (Barcelona, ES, 1991), 1991. 606-623.

Galavotti, L., F. Sebastiani, e M. Simi. “Experiments on the use of feature selection and negative evidence in automated text categorization.” *In Proceedings of ECDL-00, 4th*

European Conference on Research and Advanced Technology for Digital Libraries. (Lisbon, PT, 2000), 2000. 59-68.

Gaussier, E., C. Goutte, K Popat, e F. Chen. "A hierarchical model for clustering and categorising documents." *Proc. of the 24th BCS-IRSG European Colloquium on Information Retrieval Research.* Glasgow, UK, 2002. 229–247.

Ghamrawi, N., e A. McCallum. "Collective multi-label classification." *In: Proceedings of the 3005 ACM Conference on Information and Knowledge Management (CIKM'05).* Bremen, Germany, 2005. 195-200.

Giorgetti, Daniela, e Fabrizio Sebastiani. "Automating Survey Coding by Multiclass Text Categorization Techniques." *Journal of the American Society for Information Science and Technology. Forthcoming.* 2003.

Godbole, S., e S. Sarawagi. "Discriminative methods for multi-labeled classification." *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2004).* 2004. 22-30.

Gonçalves, Rodrigo Mikosz, João Batista Ramos Cortes, Marcio Augusto Reolon Schmidt, e Marcos Benedito Schimalski. "Classificação hierárquica e fuzzy de imagens de alta resolução." *Anais XIII Simpósio Brasileiro de Sensoriamento Remoto.* Florianópolis, Brasil, 2007. 547-554.

Goncalves, T, e P. Quaresma. "A Preliminary Approach to the Multilabel Classification Problem of Portuguese Juridical Documents." *Proceedings of the 11th Portuguese Conference on Artificial Intelligence (EPIA '03).* 2003.

Granitzer, M. "Hierarchical Text Classification using Methods from Machine Learning." PhD thesis, Graz University of Technology, 2003.

Grobelnik, Marko. "Tutorial on Text Mining and Link Analysis for Web and Semantic Web." *ECOLEAD Summer School II 2007, Prague.* Ljubljana, Setembro de 2007.

Hayes, Philip J., e Steven P. Weinstein. "CONSTRUE/TIS: A system for content-based indexing of a database of news stories." *In Proc. Conference on Innovative Applications of Artificial Intelligence.* 1990. 49-66.

Haynes, R. B., K. A. Mckiibbon, C. A. Walker, e J. C. Sinclair. "Online Access to MEDLINE in Clinical Setting. A Study of Use and Usefulness." *Annals of Internal Medicine, 112.* 1990. 78-84.

Hofmann, T., L. Cai, e M. Ciaramita. "Learning with taxonomies: classifying documents and words." *In NIPS 2003: Workshop on syntax, semantics, and statistics.* 2003.

Ingo, Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, e Timm Euler. "Yale: rapid prototyping for complex data mining tasks." *In KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining.* New York: ACM Press, 2006. 935-940.

Ittner, David J., David D. Lewis, e David D. Ahn. "Text categorization of low quality images." *In Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval.* Las Vegas, US, 1995. 301–315.

Jiang, L., H. Zhang, Z. Cai, e J. Su. “Learning tree augmented naive Bayes for ranking.” *Proceedings of the 10th International Conference on Database Systems for Advanced Applications (DASFAA 2005), Springer(2005).* 2005.

Joachims, Thorsten. “Text categorization with support vector machines: learning with many relevant features.” *Proceedings of ECML-98, 10th European Conference on Machine Learning.* 1998. 137–142.

John, George H., e Pat Langley. “Estimating continuous distributions in bayesian classifiers.” *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence.* 1995. 338–345.

John, H. G., R. Kohavi, e K. Pfleger. “Irrelevant Features and the Subset Selection Problem.” *Proc. of the 11th International Conference on Machine Learning ICML94.* 1994. 121-129.

Kauffman, Chris, e George Karypis. “An Analysis of Information Content Present in Protein-DNA Interactions.” *Pacific Symposium on Biocomputing 13.* 2008. 477-488.

Kiritchenko, Svetlana. “Hierarchical Text Categorization and Its Application to Bioinformatics.” Ph.D., Faculty of Engineering University of Ottawa, Ottawa, Canada, 2005, 2000.

Kiritchenko, Svetlana, Stan Matwin, Richard Nock, e A. Fazel Famili. “Learning and Evaluation in the Presence of Class Hierarchies: Application to Text Categorization.” *L. Lamontagne and M. Marchand (Eds.): Canadian AI 2006, LNAI 4013.* 2006. 397–408.

Kohavi, R. “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection.” *In Proceedings of the 14th International Joint Conference on Artificial Intelligence.* 1995. 1137-1145.

Koller, Daphne, e Mehran Sahami. “Hierarchically classifying documents using very few words.” *Proceedings of 14th International Conference on Machine Learning ICML-97.* Nashville, Tennessee, 1997. 170-178.

Labrou, Y., e T. W. Finin. “Yahoo! as an ontology: Using Yahoo! categories to describe documents.” *In Proc. of the 8th Int. Conf. on Information Knowledge Management.* Kansas City, MO, 1999. 180-187.

Lang, Ken. “Learning to filter netnews.” *Proceedings of the 12th International Conference on Machine Learning.* 1995. 331–339.

Larkey, Leah S., e W. Bruce Croft. “Combining classifiers in Text Categorization.” *Submitted to 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR96).* Zurich, Switzerland, 1996.

Lauser, B., e A. Hotho. “Automatic multi-label subject indexing in a multilingual environment.” *Proceedings of the 7th European Conference in Research and Advanced Technology for Digital Libraries (ECDL 2003).* 2003.

Lewis, David D. “An evaluation of phrasal and clustered representations on a text categorization task.” *Proc. of the 15th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval.* 1992b. 37–50.

Lewis, David D., Yiming Yang, Tony G. Rose, e Fan Li. “RCV1: A New Benchmark Collection for Text Categorization Research.” *The Journal of Machine Learning Research, 5*, 2004: 361-397.

Lewis, David D.. “Feature Selection and Feature Extraction for Text Categorization.” *Proceedings of Speech and Natural Language Workshop.* Harriman, CA:Morgan Kaufmann, 1992a.

Li, T., e M. Ogihara. “Detecting emotion in music.” *Proceedings of the International Symposium on Music Information Retrieval .* Washington D.C., USA, 2003.

Liao, Ciya, Shamim Alpha, e Paul Dixon. *Feature Preparation in Text Categorization.* White Papers, Oracle, 2003.

Lorena, Ana Carolina. “Investigação de estratégias para a geração de máquinas de vetores de suporte multiclasses.” Tese de Douturamento em Ciências de Computação e Matemática Computacional, Instituto de Ciências Matemáticas e de Computadores - ICMC-USP, USP - São Carlos, 2006.

Manning, C., e H. Schutze. *Foundations of Statistical Natural Language Processing.* Cambridge, US: The MIT Press, 1999.

Manning, Christopher D., Prabhakar Raghavan, e Hinrich Schütze. *An Introduction to Information Retrieval.* Cambridge : Cambridge University Press, 2007.

Maron, M. E. “Automatic Indexing: An Experimental Inquiry.” *Journal of the Association for Computing Machinery 8(3).* 1961. 404–417.

Matra. *Glossary of Terms.* 2008. http://matra.sourceforge.net/misc/glossary.php [acedido em 30 de Abril de 2008].

McCallum, A. “Multi-label text classification with a mixture model trained by EM.” *Proceedings of the AAAI' 99 Workshop on Text Learning.* 1999.

McCallum, A., R. Rosenfeld, T. Mitchell, e A Ng. “Improving Text Classification by Shrinkage in a Hierarchy of Classes.” *Proceedings of the International Conference on Machine Learning (ICML).* 1998. 359–367.

McCallum, Andrew, e Kamal Nigam. “A comparison of event models for naive bayes text classification.” *Proceedings of AAAI-98 Workshop on Learning for Text Categorization.* 1998. 41–48.

Menczer, F. “ ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery.” *In D. Fisher, ed., Proceedings of the 14th International Conference on Machine Learning (ICML97).* Morgan Kaufmann, 1997.

Menczer, F., e R.K. Belew. “Adaptive Information Agents in Distributed Textual Environments.” *In K. Sycara and M. Wooldridge (eds.) Proceedings of the 2nd International Conference on Autonomous Agents (Agents '98).* ACM Press, 1998.

Mewes, H.W., et al. “MIPS: A database for protein sequences and complete genomes.” *Nucl. Acids Res., 27.* 1999. 44-48.

Mladenic, Dunja. “Feature subset selection in text-learning.” *10th European Conference on Machine Learning ECML98.* 1998a. 6.

Mladenic, Dunja. “Machine Learning on non-homogeneous, distributed text data.” PhD thesis, Faculty of Computer and Information Science, University of Ljubljana, Slovenia, 1998b, 120.

Mladenic, Dunja, e Marko Grobelnik. “Feature selection for unbalanced class distribution and Naive Bayes.” *Proceedings of the 16th International Conference on Machine Learning ICML-99.* San Francisco, CA: Morgan Kaufmann Publishers, 1999. 258-267.

Mladenic, Dunja. “Turning Yahoo into an Automatic Web-Page Classifier.” *Proceedings of the 13th European Conference on Aritficial Intelligence ECAI'98.* 1998c. 473-474.

Muslea, Ion. “Extraction Patterns for Information Extraction Tasks: A Survey.” *Proceedings of the American Association for Artificial Intelligence (www.aaai.org),1999.* 1999.

Muslea, Ion. “Extraction Patterns: From Information Extraction to Wrapper Induction.” *Information Sciences Institute, University of Southern California.* 1998.

Ng, H. T., W. B. Goh, e K. L. Low. “Feature selection, perceptron learning, and a usability case study for text categorization.” *In Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval .* (Philadelphia, US, 1997), 1997. 67-73.

Pinto, J. Manuel Castro, Manuela Parreira, e Maria do Céu Vieira Lopes. *Gramática do Português Moderno (4º edição).* Plátano Editora, 1988.

Pirkola, Ari. “Focused Crawling: A Means to Acquire Biological Data from the Web.” *Workshop on Data Mining in Bioinformatics, VLDB07.* 2007.

Platt, J. “Fast training of support vector machines using sequential minimal optimization.” *Scholkopf, B., Burges, C., Smola, A, eds: Advances in Kernel Methods - Support Vector Learning.* MIT Press, 1998.

Porter, Martin F. “An algorithm for suffix stripping.” *Program, 14(3).* 1980. 130-137.

Provost, F., e R. Kohavi. “On applied research in machine learning.” *Machine Learning, 30 .* 1998. 127-132.

Quinlan, J. R. “Constructing Decision Tree in C4.5: Programs for Machine Learning.” *Morgan Kaufman Publishers.* 1993. 17-26.

Quinlan, J. Ross. “Induction of decision trees.” *Mach. Learn., 1(1).* 1986. 81–106.

Rijsbergen, C. J. Van. *Information Retrieval, 2nd edition.* Butterworths, 2 edition, 1979.

Rocchio, J. “Relevance Feedback in Information Retrieval.” In *The SMART Retrieval System: Experiments in Automatic Document Processing, Chapter 14*, 313-323. Prentice-Hall, 1971.

Rousu, Juho, Craig Saunders, Sandor Szedmak, e John Shawe-Taylor. “On Maximum Margin Hierarchical Multilabel Classification.” 2005a.

Rousu, Juho, Craig Saunders, Sandor Szedmak, e Johnr Shawe-Taylo. “Kernel-based learning of hierarchical multilabel classification models.” *Journal of Machine Learning Research, 7.* 2006. 1601-1626.

Rousu, Juho, Craig Saunders, Sandor Szedmak, e Johnr. Shawe-Taylo. “Learning hierarchical multi-category text classification models.” *In L. De Raedt and S. Wrobel, editors, Proceedings of the 22nd International Conference on Machine Learning.* ACM Press, 2005b. 744-751.

Ruiz, Miguel E., e Padmini Srinivasan. “Hierarchical Text Categorization Using Neural Networks.” *Information Retrieval 5.* 2002. 87–118.

Salton, G. “Developments in Automatic Text Retrieval.” *Science, Vol. 253.* 1991. 974-979.

Sasaki, M., e K. Kita. “Rule-based text categorization using hierarchical categories.” *In Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics.* La Jolla, US, 1998. 2827–2830.

Schapire, Robert E., e Yoram Singer. “BoosTexter: A boosting-based system for text categorization.” *Machine Learning 39.* 2000. 135-168.

Scott, S., e S. Matwin. “Feature engineering for text classification.” *Proceedings of ICML-99, 16th International Conference on Machine Learning.* 1999.

Sebastiani, F., A. Sperduti, e N. Valdambrini. “An improved boosting algorithm and its application to automated text categorization.” *A. Agah, J. Callan and E. Rundensteiner, editors, Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management.* McLean: US, 2000. 78–85.

Sebastiani, Fabrizio. “Machine learning in automated text categorization.” *ACM Computing Surveys (CSUR).* 2002. 1-47.

Shannon, C. E. “A mathematical theory of communication.” *Bell System Technical Journal* 27 [1948]: 379–423 and 623–656.

Shannon, C. E., e W. Weaver. *The mathematical theory of communication.* University of Illinois Press, Urbana, 1949.

Shilane, P., M. Kazhdan, P. Min, e T. Funkhouser. “The Princeton Shape Benchmark.” *Proc. Shape Modeling International.* Genoa, Italy, 2004 .

Silva, Joaquim Francisco Ferreira. “Extracção de Unidades Textuais, Agrupamento, Caracterização e Classificação de Documentos.” Dissertação apresentada para obtenção do Grau de Douturamento em Informática, Universidade Nova de Lisboa - Faculdade de ciências e Tecnologia, Lisboa, 2003, 155.

Slonim, N., e N. Tishby. “Document Clustering Using word Clusters via the Information Bottleneck Method.” *ACM SIGIR 2000.* 2000. 208–215.

Spark-Jones, K., e P. Willett. “Readings in Information Retrieval.” San Francisco: Morgan Kaufmann, 1997.

Spyromitros, E., G. Tsoumakas, e I. Vlahavas. “An Empirical Study of Lazy Multilabel Classification Algorithms.” *Proc. 5th Hellenic Conference on Artificial Intelligence (SETN 2008).* Syros, Greece: Springer, 2008.

Struyf, Jan, Saso Dzeroski, Hendrik Blockeel, e Amanda Clare. “Hierarchical Multi-classification with Predictive Clustering Trees in Functional Genomics.” *The Workshop on Computational Methods in Bioinformatics at the 12th Portuguese Conference on Artificial Intelligence.* Springer, 2005. 272-283.

Su, J., e H. Zhang. “Full Bayesian Network Classifiers.” *Proceedings of the 23rd International Conference on Machine Learning (ICML2006).* 2006.

Sullivan, Tessa. *Tessa Sullivan: Technical Challenges in Classification Analy.* University of North Carolina. 13 de Dezembro de 2007. http://www.youtube.com/watch?v=rFMK3Ypr2rE [acedido em 29 de Abril de 2008].

Sun, Aixin, e Ee-Peng Lim. “Hierarchical Text Classification and Evaluation.” *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM 2001).* California, USA, 2001. 521-528.

Thabtah, F. A., P. Cowling, e Y. Peng. “MMAC: A New Multi-class, Multi-label Associative Classification Approach.” *Proceedings of the 4th IEEE International Conference on Data Mining, ICDM '04.* 2004.

Tikk, Domonkos, e Gyorgy Biró. “Experiments with multi-label text classifier on the Reuters collection.” *Proc. of the International Conference on Computational Cybernetics (ICCC 03).* Siófok, Hungary, 2003. 33-38.

Toutanova, K, F. Chen, K. Popat, e T. Hofmann. “Text classification in a hierarchical mixture model for small training sets.” *Proc. of the 10th Int. Conf. on Information and Knowledge Management’.* Atlanta, USA,, 2001. 105–112.

Trohidis, K., G. Tsoumakas, G. Kalliris, e I. Vlahava. “Multi-label Classification of Music into Emotions.” *Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008),.* Philadelphia, PA, USA, 2008.

Tsochantaridis, Ioannis, Thomas Hofmann, Thorsten Joachims, e Yasemin Altun. “Support Vector Machine Learning for Interdependent and Structured Output Spaces.” *Proceedings of the 21 st International Conference on Machine Learning.* Canada, 2004.

Tsoumakas, Grigorios, e Ioannis Katakis. “Multi-Label Classification: An Overview.” *International Journal of Data Warehousing and Mining 3.* 2007a. 1-13.

Tsoumakas, Grigorios, e Ioannis Vlahavas. “Random k-Labelsets: An Ensemble Method for Multilabel Classi¯cation.” *Proceedings of the 18th European Conference on Machine Learning (ECML 2007).* Warsaw, Poland: Springer Verlag, LNAI 4701, 2007b. 406-417.

Vapnik, Vladimir. *The Nature of Statistical Learning Theory.* New York: Springer, 1995.

Vens, Celine, Jan Struyf, Leander Schietgat, Saso Dzeroski, e Hendrik Blockeel. “Decision Trees for Hierarchical Multi-Label Classification.” *Journal of Machine Learning* Volume 73, Number 2 [2008]: 185-214.

Vinokourov, A., e M Girolami. “A probabilistic framework for the hierarchic organization and classification of document collections.” *Journal of Intelligent Information Systems 18(2/3).* Special Issue on Automated Text Categorisation, 2002. 153-172.

W., Hersh, Buckley C., Leone T., e Hickman D. “Ohsumed: an interactive retrieval evaluation and new large text collection for research.” *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval (Dublin, IE, 1994).* 1994. 192–201.

Wang, K., S. Zhou, e S. Liew. “Building Hierarchical Classifiers Using Class Proximity.” *Proceedings of the International Conference on Very Large Data Bases (VLDB).* 1999. 363-374.

Wang, K., S. Zhou, e Y. He. “Hierarchical classification of real life documents.” *Proc. of the 1st SIAM Int. Conf. on Data Mining.* Chicago, 2001.

Wang, Tao, e Bipin C. Desai. “Document Classification with ACM Subject Hierarchy.” *20th Canadian Conference on Electrical and Computer Engineering (CCECE 2007).* 2007. 792-795.

Wiener, E. D., J. O. Pedersen, e A. S Weigend. “A neural network approach to topic spotting.” *In Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval .* (Las Vegas, US, 1995), 1995. 317-332.

William, Arms. *Digital Libraries: Glossary (1999).* 2000. http://www.cs.cornell.edu/wya/DigLib/MS1999/glossary.html [acedido em 30 de Abril de 2008].

Witten, Ian H., e Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, 2nd edition.* Morgan Kaufmann, 2005.

Wolpert, D.H. “Stacked Generalization.” *Neural Networks, vol. 5.* 1992. 241-59.

Wu, Feihong, Jun Zhang, e Vasant Honavar. “Learning Classifers Using Hierarchically Structured Class Taxonomies.” 2005.

Wurst, Michael. “The Word Vector Tool.” *User Guide/Operator Reference/Developer Tutorial.* 2007.

Yang, Yiming. “An evaluation of statistical approaches to text categorization.” *Journal of Information Retrieval* Vol.1, No.1/2 [1999].

Yang, Yiming, e Jan O. Pedersen. “A Comparative Study on Feature Selection in Text Categorization.” *Proceedings of the 14th International Conference on Machine Learning ICML-97.* 1997. 412-420.

Yang, Yiming, e X Liu. “A re-examination of text categorization methods.” *In 22nd Annual International SIGIR.* Berkley, 1999. 42–49.

Zhang, Harry. “The optimality of naive Bayes.” *Proceedings of the 17th International FLAIRS conference (FLAIRS2004), Best paper award winner (second place).* AAAI Press, 2004.

Zhang, M. L., e Z. H. Zhou. “A k-nearest neighbor based algorithm for multi-label classifcation.” *Proceedings of the 1st IEEE International Conference on Granular Computing.* 2005. 718-721.

Zhang, Min-Ling, e Zhi-Hua Zhou. “Multi-label neural networks with applications to functional genomics and text categorization.” *IEEE Transactions on Knowledge and Data Engineering 18.* 2006. 1338-1351.

Zhang, M-L, e Z-H Zhou. “Ml-knn: A lazy learning approach to multi-label learning.” *Pattern Recognition, 40(7).* 2007. 2038-2048.

Zheng, Z., X. Wu, e R. Srihari. “Feature selection for text categorization on imbalanced data.” *SIGKDD Explorations Newsletter, 6(1).* 2004. 80-89.

Zhu, S., X. Ji, W. Xu, e Y. Gong. “Multi-labelled classifcation using maximum entropy method.” *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in Information Retrieval.* 2005. 274-281.

Anexo 1 - Como classificar os trabalhos utilizando o sistema de classificação ACM

1. Observar o primeiro e segundo nível da árvore de classificação ACM e tomar nota de cada uma das categorias do segundo nível que pareçam relevantes;

2. Utilizando os nós do primeiro e segundo nível já identificados, observar os nós do terceiro nível se aplicável (os nós A, E, G e J possuem zonas que não vão além do segundo nível) e os nós descritores de assunto abaixo deles. Identificar qualquer descritor que pareça aplicar-se ao documento. Por exemplo, para o documento "*An efficient method for checking object-oriented database schema correctness*" de *A. Formica et al.,* apresentado em Setembro de 1998 em *ACM Transactions on Database Systems*, as categorias devem incluir (categorias mais relevantes devem ser apresentadas primeiro):
   H.2.4 Systems---Object-oriented databases, *D.3.3 Language Constructs and Features---Data types and structures*, *e H.2.1 Logical Design---Schema and subschema*.
   Deve ser usado o nó codificado mais baixo e usar descritores quando aplicável (como habitualmente será). Deve ser usado o maior número de descritores sob um nó codificado desde que apropriado ao documento. Deve-se garantir que são usadas todas as categorias e nós que se apliquem ao documento. Um nó codificado deve ser usado sem qualquer descritor se: (1) nenhum dos descritores se aplica (2) se todos os descritores se aplicam. Por exemplo, usa-se *H.4.1 Office Automation* se o documento tratar conceitos gerais da automação de escritório (*office automation*);

3. Adicionalmente aos descritores existentes, o autor pode adicionar descritores de assunto implícitos (secção *2.1.3*, página *7*), que são nomes próprios (os nomes de linguagens, sistemas, e assim sucessivamente). Estes descritores são associados a um nó codificado e devem ser usados sempre que sejam significantes. Estes tipos de descritores não devem ser usados apenas porque o nome próprio é mencionado no documento. Por exemplo, um documento totalmente acerca de *FORTRAN* deve ser exibido sob a classificação *D.3.2 Language Classifications* com um descritor de assunto implícito de *FORTRAN*. Deve ser usado este descritor implícito apenas se *FORTRAN* é crucial para o conteúdo do documento;

4. Usa-se um nó *General* de qualquer nível apenas se o documento cobre a maioria dos conceitos num determinado campo. Normalmente isto só se aplica a documentos de pesquisa alargados. Por exemplo, "*Performance and dependability evaluation of scalable massively parallel computer systems with conjoint simulation*" de *A. Hein and M. Dal Cin*, apresentado em Outubro de 1998 em *ACM Transactions on Modeling and Computer Simulation: B.8.0 General*, *C.4 Performance of Systems*, *I.6.0 General*, *e I.6.5 Model Development*;

5. Se um documento não pode ser classificado sob nenhum nó, então usa-se o nó "*Miscellaneous*" no campo em causa;

6. Cuidado com uma armadilha comum, o uso de uma secção inapropriada da árvore. Por exemplo, não usar o descritor "*Microprocessors and microcomputers*" sob *B.7.1 HARDWARE---INTEGRATED CIRCUITS---Types and Design Styles*, a não ser que se esteja realmente a lidar com circuitos. Se o documento discute o microprocessador ou microcomputador de uma forma genérica, então o nó apropriado deverá ser *C.5.3 COMPUTER SYSTEMS ORGANIZATION---COMPUTER SYSTEM IMPLEMENTATION---Microcomputers* (com o descritor "*Microprocessors*" se o documento é especificamente acerca de microprocessadores). De forma semelhante, não se deve usar o descritor "*Law*" sob *I.2.1 ARTIFICIAL INTELLIGENCE---Applications and Expert Systems* se o documento envolve patentes ou aspectos legais de computação sem qualquer conotação AI (*Artificial Intelligence*). Neste caso, deve ser usado um nó sob *K.5 LEGAL ASPECTS OF COMPUTING*;

7. Ao utilizar-se o CCS, ter em atenção que existem nós inactivos juntamente com activos. Os inactivos estão marcados com um (*) ou dois asteriscos (**) e já não são utilizados para classificar novos documentos, apesar de trabalhos antigos anteriormente classificados sob estes termos continuam pesquisáveis. Portanto, evitar classificar os documentos sob qualquer termo marcado com um ou dois asteriscos. Alguns dos nós inactivos incluem apontadores entre parênteses para os termos de substituição*: B.3.4 Reliability, Testing, and Fault Tolerance** (B.8)*;

8. Depois de decidido quais os nós aplicáveis, preparar a classificação no seguinte formato:
   Categories and subject descriptors: Third-level node number [Second-level node title]: Third-level node title---first subject descriptor, second subject descriptor, etc.;
   Third-level node number…

   Como exemplo, seguem dois casos, em que o primeiro envolve um descritor de assunto implícito e o segundo envolve vários nós.

   Caso 1:
   Categories and subject descriptors: H.2.3 [Database Management]: Languages---SQL

   Caso 2:
   Categories and subject descriptors: D.2.7 [Software Engineering]: Distribution, Maintenance, and Enhancement---portability; H.2.3 [Database Management]: Languages---query languages; H.4.2 [Information Systems Applications]: Types of Systems---decision support; K.6.3 [Management of Computing and Information Systems]: Software Management---software selection

9. Observar a lista dos dezasseis termos gerais e decidir, se aplicáveis, quais. Estas escolhas são independentes da classificação atribuída ao documento. Por exemplo, para o documento no ponto 4 destas orientações, deveriam ser escolhidos como termos gerais: *Design*, *Performance*, *Reliability*;

10. Finalmente, preparar uma lista adicional de palavras-chave e frases. Estas são primeiramente palavras ou frases não cobertas pelo esquema existente. Esta situação

poderá ocorrer com maior probabilidade em níveis de maior detalhe do que aquele que possa ser expresso por uma árvore com quatro níveis. Garantir que as palavras-chave e frases fazem sentido se aparecerem separadamente. Por exemplo, a palavra "*breadth*" pode fazer sentido num determinado documento mas é improvável ser significante numa lista separada de palavras. Da mesma forma que um adjectivo não deve ser usado sem um substantivo. A pergunta a fazer é, "Alguém procuraria esta palavra-chave ou frase num índice?". Não devem ser repetidas palavras e frases de nenhum nível do sistema de classificação usados para classificar o documento. Por exemplo, para o documento no ponto 2, os autores escolheram como palavras-chave e frases adicionais: *databases*, *graph theory*, *inheritance conflicts*, *inheritance process*, *object-oriented database schemas*, *and recursive types*;

11. É fortemente recomendado que o autor observe outras publicações já publicadas, do mesmo tópico ou semelhante, para ver como foram classificadas.

Anexo 2 - Resultados da colecção “multi-etiqueta 5000”

	Binary Relevance						Label Powerset						MLkNN			
Nível 1	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**k=1**	**k=5**	**k=10**	**k=30**
HammingLoss	0,10	0,13	0,13	0,12	0,10	0,12	0,10	0,16	0,14	0,11	---	0,14	0,13	0,11	0,11	0,11
Accuracy	0,44	0,23	0,23	0,14	0,35	0,37	0,52	0,31	0,40	0,48	---	0,40	0,15	0,26	0,26	0,30
Precision	0,53	0,29	0,29	0,17	0,42	0,44	0,64	0,38	0,49	0,61	---	0,49	0,19	0,32	0,32	0,37
Recall	0,48	0,25	0,25	0,14	0,36	0,45	0,55	0,35	0,46	0,48	---	0,46	0,15	0,27	0,27	0,33
Fmeasure	0,50	0,27	0,27	0,15	0,38	0,44	0,59	0,37	0,47	0,53	---	0,47	0,17	0,29	0,29	0,35
Nível 2																
HammingLoss	0,29	0,49	0,49	0,13	0,18	0,41	0,36	0,62	0,62	0,39	---	0,51	0,48	0,33	0,33	0,33
Accuracy	0,71	0,51	0,51	0,87	0,82	0,59	0,64	0,38	0,38	0,61	---	0,49	0,52	0,67	0,67	0,67
Precision	0,71	0,51	0,51	0,87	0,82	0,59	0,64	0,38	0,38	0,61	---	0,49	0,52	0,67	0,67	0,67
Recall	0,78	0,56	0,56	0,87	0,84	0,73	0,67	0,45	0,45	0,61	---	0,57	0,53	0,69	0,69	0,72
Fmeasure	0,74	0,53	0,53	0,87	0,83	0,66	0,66	0,41	0,41	0,61	---	0,53	0,52	0,68	0,68	0,70

Tabela 31 – Avaliação da colecção “multi-etiqueta 5000”, seleccionando 1000 termos

	Binary Relevance						Label Powerset						MLkNN			
Nível 1	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**k=1**	**k=5**	**k=10**	**k=30**
HammingLoss	0,10	0,13	0,13	0,12	0,10	0,12	0,11	0,16	0,16	0,11	---	0,14	0,13	0,11	0,11	0,11
Accuracy	0,41	0,23	0,23	0,11	0,32	0,35	0,51	0,31	0,31	0,47	---	0,39	0,12	0,25	0,25	0,30
Precision	0,50	0,29	0,29	0,13	0,39	0,43	0,63	0,39	0,39	0,60	---	0,48	0,16	0,31	0,31	0,37
Recall	0,44	0,24	0,24	0,11	0,32	0,43	0,53	0,35	0,35	0,47	---	0,45	0,12	0,26	0,26	0,32
Fmeasure	0,47	0,26	0,26	0,12	0,35	0,43	0,58	0,37	0,37	0,53	---	0,47	0,14	0,28	0,28	0,34
Nível 2																
HammingLoss	0,28	0,47	0,47	0,12	0,19	0,42	0,37	0,61	0,61	0,40	---	0,52	0,46	0,30	0,30	0,32
Accuracy	0,72	0,53	0,53	0,88	0,81	0,58	0,63	0,39	0,39	0,60	---	0,48	0,54	0,70	0,70	0,68
Precision	0,72	0,53	0,53	0,88	0,81	0,58	0,63	0,39	0,39	0,60	---	0,48	0,54	0,70	0,70	0,68
Recall	0,77	0,56	0,56	0,88	0,83	0,71	0,65	0,45	0,45	0,60	---	0,56	0,54	0,72	0,72	0,73
Fmeasure	0,75	0,55	0,55	0,88	0,82	0,64	0,64	0,42	0,42	0,60	---	0,52	0,54	0,71	0,71	0,71

Tabela 32 – Avaliação da colecção “multi-etiqueta 5000”, seleccionando 800 termos

	Binary Relevance						Label Powerset						MLkNN			
Nível 1	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**k=1**	**k=5**	**k=10**	**k=30**
HammingLoss	0,10	0,13	0,13	0,12	0,11	0,12	0,11	0,16	0,16	0,12	---	0,14	0,13	0,12	0,12	0,11
Accuracy	0,37	0,24	0,24	0,08	0,27	0,35	0,50	0,32	0,32	0,44	---	0,39	0,08	0,24	0,24	0,28
Precision	0,45	0,30	0,30	0,10	0,33	0,41	0,62	0,40	0,40	0,56	---	0,48	0,10	0,29	0,29	0,34
Recall	0,40	0,25	0,25	0,08	0,28	0,43	0,52	0,36	0,36	0,44	---	0,45	0,08	0,24	0,24	0,29
Fmeasure	0,42	0,27	0,27	0,09	0,30	0,42	0,56	0,38	0,38	0,49	---	0,46	0,09	0,26	0,26	0,32
Nível 2																
HammingLoss	0,27	0,48	0,48	0,12	0,19	0,44	0,38	0,60	0,60	0,44	---	0,52	0,48	0,32	0,32	0,31
Accuracy	0,73	0,52	0,52	0,88	0,81	0,56	0,62	0,40	0,40	0,56	---	0,48	0,52	0,68	0,68	0,69
Precision	0,73	0,52	0,52	0,88	0,81	0,56	0,62	0,40	0,40	0,56	---	0,48	0,52	0,68	0,68	0,69
Recall	0,77	0,55	0,55	0,88	0,82	0,70	0,64	0,45	0,45	0,56	---	0,56	0,52	0,69	0,69	0,73
Fmeasure	0,75	0,54	0,54	0,88	0,82	0,63	0,63	0,42	0,42	0,56	---	0,52	0,52	0,68	0,68	0,71

Tabela 33 – Avaliação da colecção “multi-etiqueta 5000”, seleccionando 600 termos

	Binary Relevance						Label Powerset						MLkNN			
Nível 1	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**k=1**	**k=5**	**k=10**	**k=30**
HammingLoss	0,11	0,13	0,13	0,12	0,11	0,12	0,11	0,16	0,16	0,12	---	0,15	0,13	0,12	0,12	0,12
Accuracy	0,28	0,24	0,24	0,05	0,18	0,31	0,47	0,30	0,30	0,41	---	0,37	0,11	0,20	0,20	0,20
Precision	0,35	0,30	0,30	0,06	0,22	0,38	0,58	0,38	0,38	0,53	---	0,45	0,14	0,25	0,25	0,26
Recall	0,29	0,26	0,26	0,05	0,18	0,37	0,47	0,35	0,35	0,41	---	0,41	0,11	0,20	0,20	0,21
Fmeasure	0,32	0,28	0,28	0,06	0,20	0,37	0,52	0,36	0,36	0,46	---	0,43	0,13	0,22	0,22	0,23
Nível 2																
HammingLoss	0,27	0,51	0,51	0,11	0,20	0,43	0,42	0,62	0,62	0,47	---	0,55	0,44	0,33	0,33	0,31
Accuracy	0,73	0,49	0,49	0,89	0,80	0,57	0,58	0,38	0,38	0,53	---	0,45	0,56	0,67	0,67	0,69
Precision	0,73	0,49	0,49	0,89	0,80	0,57	0,58	0,38	0,38	0,53	---	0,45	0,56	0,67	0,67	0,69
Recall	0,76	0,53	0,53	0,89	0,81	0,68	0,59	0,44	0,44	0,53	---	0,51	0,56	0,68	0,68	0,70
Fmeasure	0,74	0,51	0,51	0,89	0,81	0,62	0,58	0,41	0,41	0,53	---	0,48	0,56	0,68	0,68	0,70

Tabela 34 – Avaliação da colecção “multi-etiqueta 5000”, seleccionando 400 termos

	Binary Relevance						Label Powerset						MLkNN			
Nível 1	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**k=1**	**k=5**	**k=10**	**k=30**
HammingLoss	0,12	0,13	0,13	0,13	0,12	0,12	0,13	0,16	0,16	0,14	---	0,15	0,13	0,12	0,12	0,12
Accuracy	0,16	0,21	0,21	0,03	0,12	0,28	0,39	0,29	0,29	0,34	---	0,34	0,12	0,15	0,15	0,17
Precision	0,20	0,27	0,27	0,03	0,15	0,34	0,50	0,37	0,37	0,45	---	0,42	0,15	0,19	0,19	0,22
Recall	0,17	0,23	0,23	0,03	0,12	0,31	0,39	0,33	0,33	0,34	---	0,38	0,12	0,15	0,15	0,17
Fmeasure	0,18	0,25	0,25	0,03	0,14	0,33	0,44	0,35	0,35	0,39	---	0,40	0,13	0,17	0,17	0,19
Nível 2																
HammingLoss	0,26	0,53	0,53	0,11	0,20	0,42	0,50	0,63	0,63	0,55	---	0,58	0,49	0,33	0,33	0,32
Accuracy	0,74	0,47	0,47	0,89	0,80	0,58	0,50	0,37	0,37	0,45	---	0,42	0,51	0,67	0,67	0,68
Precision	0,74	0,47	0,47	0,89	0,80	0,58	0,50	0,37	0,37	0,45	---	0,42	0,51	0,67	0,67	0,68
Recall	0,75	0,50	0,50	0,89	0,81	0,66	0,50	0,42	0,42	0,45	---	0,48	0,51	0,67	0,67	0,68
Fmeasure	0,74	0,48	0,48	0,89	0,80	0,62	0,50	0,39	0,39	0,45	---	0,45	0,51	0,67	0,67	0,68

Tabela 35 – Avaliação da colecção “multi-etiqueta 5000”, seleccionando 200 termos

(--- Experiências que não finalizaram devido à ocorrência de erro desconhecido)

Anexo 3 - Resultados da colecção “multi-etiqueta 10000”

	Binary Relevance						Label Powerset						MLkNN			
Nível 1	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**k=1**	**k=5**	**k=10**	**k=30**
HammingLoss	0,10	0,13	0,13	0,12	0,11	0,12	---	0,16	0,16	0,11	---	0,15	0,13	0,12	0,12	0,11
Accuracy	0,46	0,31	0,31	0,20	0,31	0,41	---	0,36	0,36	0,48	---	0,39	0,13	0,33	0,33	0,35
Precision	0,57	0,39	0,39	0,25	0,38	0,50	---	0,45	0,45	0,63	---	0,49	0,19	0,42	0,42	0,44
Recall	0,49	0,34	0,34	0,20	0,31	0,49	---	0,42	0,42	0,48	---	0,45	0,13	0,34	0,34	0,38
Fmeasure	0,53	0,37	0,37	0,23	0,34	0,50	---	0,44	0,44	0,55	---	0,47	0,16	0,38	0,38	0,41
Nível 2																
HammingLoss	0,24	0,45	0,45	0,13	0,22	0,37	---	0,55	0,55	0,37	---	0,48	0,45	0,30	0,30	0,29
Accuracy	0,76	0,55	0,55	0,87	0,79	0,63	---	0,45	0,45	0,63	---	0,52	0,55	0,70	0,70	0,71
Precision	0,76	0,55	0,55	0,87	0,79	0,63	---	0,45	0,45	0,63	---	0,52	0,55	0,70	0,70	0,71
Recall	0,82	0,61	0,61	0,87	0,81	0,77	---	0,54	0,54	0,63	---	0,62	0,55	0,74	0,74	0,78
Fmeasure	0,79	0,58	0,58	0,87	0,80	0,69	---	0,49	0,49	0,63	---	0,56	0,55	0,72	0,72	0,74

Tabela 36 – Avaliação da colecção “multi-etiqueta 10000”, seleccionando 1000 termos

	Binary Relevance						Label Powerset						MLkNN			
Nível 1	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**k=1**	**k=5**	**k=10**	**k=30**
HammingLoss	0,10	0,13	0,13	0,12	0,11	0,12	---	0,16	0,16	0,12	---	0,15	0,14	0,12	0,12	0,11
Accuracy	0,42	0,32	0,32	0,16	0,29	0,39	---	0,36	0,36	0,46	---	0,41	0,14	0,31	0,31	0,35
Precision	0,53	0,40	0,40	0,20	0,36	0,49	---	0,46	0,46	0,61	---	0,51	0,19	0,40	0,40	0,44
Recall	0,45	0,34	0,34	0,16	0,29	0,48	---	0,42	0,42	0,46	---	0,47	0,14	0,32	0,32	0,38
Fmeasure	0,49	0,37	0,37	0,18	0,32	0,48	---	0,44	0,44	0,53	---	0,49	0,16	0,36	0,36	0,41
Nível 2																
HammingLoss	0,24	0,44	0,44	0,13	0,22	0,38	---	0,54	0,54	0,39	---	0,49	0,47	0,30	0,30	0,31
Accuracy	0,76	0,56	0,56	0,87	0,78	0,62	---	0,46	0,46	0,61	---	0,51	0,53	0,70	0,70	0,69
Precision	0,76	0,56	0,56	0,87	0,78	0,62	---	0,46	0,46	0,61	---	0,51	0,53	0,70	0,70	0,69
Recall	0,80	0,62	0,62	0,88	0,80	0,76	---	0,55	0,55	0,61	---	0,61	0,54	0,73	0,73	0,75
Fmeasure	0,78	0,59	0,59	0,87	0,79	0,68	---	0,50	0,50	0,61	---	0,56	0,54	0,71	0,71	0,72

Tabela 37 – Avaliação da colecção “multi-etiqueta 10000”, seleccionando 800 termos

	Binary Relevance						Label Powerset						MLkNN			
Nível 1	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**k=1**	**k=5**	**k=10**	**k=30**
HammingLoss	0,11	0,13	0,13	0,13	0,11	0,12	---	0,16	0,16	0,12	---	0,15	0,14	0,12	0,12	0,11
Accuracy	0,40	0,32	0,32	0,12	0,34	0,39	---	0,36	0,36	0,45	---	0,41	0,14	0,32	0,32	0,35
Precision	0,50	0,41	0,41	0,16	0,44	0,48	---	0,46	0,46	0,60	---	0,52	0,19	0,42	0,42	0,44
Recall	0,42	0,35	0,35	0,13	0,35	0,47	---	0,42	0,42	0,45	---	0,48	0,14	0,34	0,34	0,37
Fmeasure	0,46	0,38	0,38	0,14	0,39	0,47	---	0,44	0,44	0,51	---	0,49	0,16	0,37	0,37	0,40
Nível 2																
HammingLoss	0,23	0,43	0,43	0,13	0,20	0,38	---	0,54	0,54	0,40	---	0,48	0,46	0,32	0,32	0,31
Accuracy	0,77	0,57	0,57	0,87	0,80	0,62	---	0,46	0,46	0,60	---	0,52	0,54	0,68	0,68	0,69
Precision	0,77	0,57	0,57	0,87	0,80	0,62	---	0,46	0,46	0,60	---	0,52	0,54	0,68	0,68	0,69
Recall	0,80	0,62	0,62	0,87	0,82	0,75	---	0,54	0,54	0,60	---	0,61	0,54	0,70	0,70	0,74
Fmeasure	0,78	0,60	0,60	0,87	0,81	0,68	---	0,50	0,50	0,60	---	0,56	0,54	0,69	0,69	0,72

Tabela 38 – Avaliação da colecção “multi-etiqueta 10000”, seleccionando 600 termos

	Binary Relevance						Label Powerset						MLkNN			
Nível 1	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**k=1**	**k=5**	**k=10**	**k=30**
HammingLoss	0,11	0,13	0,13	0,13	0,11	0,13	---	0,16	0,16	0,13	---	0,15	0,14	0,12	0,12	0,12
Accuracy	0,32	0,29	0,29	0,08	0,27	0,36	---	0,34	0,34	0,40	---	0,39	0,13	0,29	0,29	0,30
Precision	0,41	0,37	0,37	0,10	0,34	0,45	---	0,44	0,44	0,55	---	0,49	0,18	0,38	0,38	0,38
Recall	0,33	0,32	0,32	0,08	0,27	0,43	---	0,39	0,39	0,40	---	0,45	0,13	0,30	0,30	0,31
Fmeasure	0,37	0,34	0,34	0,09	0,30	0,44	---	0,41	0,41	0,46	---	0,47	0,15	0,33	0,33	0,34
Nível 2																
HammingLoss	0,25	0,47	0,47	0,12	0,22	0,40	---	0,56	0,56	0,46	---	0,51	0,47	0,35	0,35	0,33
Accuracy	0,75	0,53	0,53	0,88	0,78	0,60	---	0,44	0,44	0,54	---	0,49	0,53	0,65	0,65	0,67
Precision	0,75	0,53	0,53	0,88	0,78	0,60	---	0,44	0,44	0,54	---	0,49	0,53	0,65	0,65	0,67
Recall	0,78	0,59	0,59	0,89	0,80	0,72	---	0,51	0,51	0,54	---	0,58	0,53	0,67	0,67	0,71
Fmeasure	0,77	0,56	0,56	0,89	0,79	0,66	---	0,47	0,47	0,54	---	0,53	0,53	0,66	0,66	0,69

Tabela 39 – Avaliação da colecção “multi-etiqueta 10000”, seleccionando 400 termos

	Binary Relevance						Label Powerset						MLkNN			
Nível 1	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**SMO**	**IBk(1)**	**IBk(5)**	**NB-M.**	**SVM**	**J48**	**k=1**	**k=5**	**k=10**	**k=30**
HammingLoss	0,12	0,14	0,14	0,13	0,12	0,13	---	0,17	0,17	0,14	---	0,16	0,14	0,13	0,13	0,12
Accuracy	0,19	0,26	0,26	0,02	0,15	0,31	---	0,31	0,31	0,34	---	0,36	0,12	0,19	0,19	0,22
Precision	0,24	0,34	0,34	0,03	0,20	0,39	---	0,40	0,40	0,46	---	0,45	0,17	0,25	0,25	0,29
Recall	0,19	0,28	0,28	0,02	0,15	0,35	---	0,36	0,36	0,34	---	0,41	0,12	0,20	0,20	0,23
Fmeasure	0,21	0,31	0,31	0,03	0,17	0,37	---	0,38	0,38	0,39	---	0,43	0,14	0,22	0,22	0,26
Nível 2																
HammingLoss	0,25	0,51	0,51	0,12	0,23	0,40	---	0,60	0,60	0,54	---	0,55	0,49	0,35	0,35	0,34
Accuracy	0,75	0,49	0,49	0,88	0,77	0,60	---	0,40	0,40	0,46	---	0,45	0,51	0,65	0,65	0,66
Precision	0,75	0,49	0,49	0,88	0,77	0,60	---	0,40	0,40	0,46	---	0,45	0,51	0,65	0,65	0,66
Recall	0,76	0,53	0,53	0,88	0,78	0,69	---	0,47	0,47	0,46	---	0,54	0,51	0,66	0,66	0,68
Fmeasure	0,75	0,51	0,51	0,88	0,78	0,64	---	0,44	0,44	0,46	---	0,49	0,51	0,66	0,66	0,67

Tabela 40 – Avaliação da colecção “multi-etiqueta 10000”, seleccionando 200 termos

(--- Experiências que não finalizaram devido à ocorrência de erro desconhecido)