João Paulo da Costa Cordeiro

Extracção de Elementos Relevantes em Texto/Páginas da World Wide Web

Tese submetida à Faculdade de Ciências da Universidade do Porto
para a obtenção do grau de Mestre em Inteligência Artificial e Computação

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto

Junho de 2003

Agradecimentos

"Porque Deus amou o mundo de tal maneira que deu o seu Filho unigénito para que todo aquele que nele crê não pereça, mas tenha a vida eterna"

S. João 3:16

Quero expressar aqui o meu profundo agradecimento a Deus, por todo amor, esperança e sentido que trouxe à minha vida. Porque inúmeras vezes, nos momentos mais turbulentos da minha existência, trouxe paz e alegria ao meu coração. Obrigado Jesus porque, sem Ti, onde estaria eu?

Na dimensão humana, o meu primeiro agradecimento dirige-se ao meu orientador, Professor Doutor Pavel Brazdil. Sem ele, o prezado leitor não estaria a ler estas linhas, neste momento. Agradeço não só os inúmeros conselhos recebidos, mas também todo a paciência, apoio, disponibilidade e sempre boa disposição manifestada. Obrigado Professor Pavel!

Agradeço a todos os professores que tive no LIACC e da FEUP, que muito contribuíram para a minha formação.

Quero agradecer aos meus pais, pela educação recebida e pelo constante apoio e amor que sempre me deram.

Agradeço também à minha esposa – Adelina – todo o apoio, amor e paciência nos meus momentos mais "aéreos" e distantes.

Resumo

Nos últimos anos, o aumento da quantidade de informação digital disponível é um facto irrefutável, nomeadamente a que se encontra na *World Wide Web*, sob a forma de documentos de texto. Nunca na história da humanidade houve um tão elevado volume de informação acessível. Apesar dos aspectos positivos que isto representa e do potencial que permite, existe uma nova problemática que surge e consiste na necessidade de ferramentas eficazes na pesquisa e extracção de informação. O trabalho desenvolvido, no âmbito desta dissertação, enquadra-se neste contexto.

O principal objectivo deste trabalho consiste em aplicar um conjunto de técnicas da *Inteligência Artificial* (IA), nomeadamente da área da Extracção de Informação, para a criação de um sistema capaz de identificar e extrair certos elementos de texto, considerados relevantes, em documentos. Embora o alvo tenha sido o de implementar uma metodologia genérica, adaptável a qualquer domínio, fixamos a nossa atenção num domínio concreto, de modo demonstrar a essa mesma metodologia. Esse domínio consistiu nos anúncios (em Português) relativos à venda de habitações.

O sistema desenvolvido utiliza *Aprendizagem Supervisionada*, para que possa ser treinado, com uma colecção de documentos anotados e assim "aprenda" a reconhecer/extrair os elementos relevantes no texto. Uma das preocupações foi que o processo de treino produzisse conhecimento simbólico, de maneira que para além de poder ser aplicado, pudesse também ser analisado. Assim, no processo de treino são induzidas regras lógicas de extracção dos elementos relevantes, satisfazendo esta exigência.

A metodologia proposta foi devidamente avaliada, mostrados os resultados obtidos e feita alguma comparação com outros sistemas. O sistema obteve resultados muito satisfatórios, no domínio fixado, abrindo assim novas possibilidades para futuras aplicações interessantes.

Abstract

In the last years the amount of information available in digital form, including text documents available on World Wide Web, has increased rather dramatically. Never before have we witnessed such volumes of available information. Despite the positive aspects that this represents and the potential that this offers, this increase poses also new kinds of problems. It is not always easy to find always what is needed. We thus need tools that enable us to search for relevant documents and extract information from them. The work that we have developed, which is described in this dissertation, addresses this issue.

The main objective of the work discussed consists of selecting and identifying suitable techniques from the field of Artificial Intelligence and Information Retrieval and adapting them as necessary. Our goal was to construct a system capable of identifying and extracting certain text elements, considered relevant, from a given set of documents. Although our aim was to devise a methodology that is quite generic that could be adapted to any specific domain, we have focussed our attention to one domain in particular, to be able to exemplify the method. The domain chosen is that of announcements (in Portuguese) concerning sales or offers of houses or flats.

The methodology adopted includes Machine Learning methods that make the system easily adaptable to diverse domains. We have developed a system that can be trained to identify and extract the relevant elements in the text. One of our concerns was that the system should produce symbolic knowledge that, apart from being applied, could also be inspected. Our system induces, during the training process, logical rules that satisfy this requirement.

The methodology proposed have been evaluated and comparative studies carried out. We show that the system works quite satisfactorily in the domain chosen and so our work opens new possibilities for new interesting applications in future.

Índice

1 Introdução

1.1 Contextualização

Desde um passado muito remoto que o ser humano tem tentado conhecer e compreender melhor o universo que o rodeia, quer distante quer próximo. Tal como um rio perto da sua nascente, assim começou o saber humano. No princípio muito incipiente, muito incerto, restrito de um pequeno grupo de "sábios". Hoje o oceano do saber é imenso e a cada hora que passa já sabemos mais alguma coisa. A cada minuto, centenas de laboratórios e centros de saber operam, nos pontos mais diversos e remotos do planeta. Vivemos na já muito rotulada "era da informação". O ser humano do final do século XX e principio do século XXI está cada vez mais consciente e desperto para esta nova e surpreendente realidade que é a informação.

Por outro lado, o acesso a toda esta informação está, hoje mais que nunca, aberta a muitos. Na Idade Média eram os centros clericais que detinham e mantinham, muito bem, os manuscritos do conhecimento, mas no final do século XV, com Gutenberg, dá-se uma das revoluções mais marcantes da nossa história do conhecimento, com a invenção da imprensa tipográfica. A partir daí vários exemplares de manuscritos antigos, passaram a ser reproduzidos em série e portanto o acesso a estes meios aumentou consideravelmente. As implicações na história, resultantes desta invenção, foram tremendas.

Na segunda metade do século XX o advento da micro tecnologia vem dar um novo impulso ao avanço do saber humano e acessibilidade ao mesmo. Temos actualmente meios de informação ao nosso alcance que os nossos antepassados mal conseguiriam imaginar. Hoje é possível que um grupo de empresários realizem uma reunião, estando estes em continentes distintos, ou que um cientista posicionado no meio da floresta amazónica, realizando um qualquer estudo, esteja consultando informação vital para o seu trabalho que por sua vez está localizada numa qualquer base de dados em Nova York ou Paris. Nunca como hoje houve um

tão forte intercâmbio científico no nosso planeta. Dizem que vivemos numa "aldeia global", talvez a expressão "teia global", fosse mais apropriada, uma teia de troca de conhecimentos, de saber, uma teia que permite que em países subdesenvolvidos existam jovens que com poucos recursos possa aceder a este conhecimento e cultivarem-se.

Este trabalho não pretende, evidentemente, ser uma reflexão sobre a história do conhecimento humano ou outra coisa qualquer semelhante, mas o apresentado em cima é ainda e só a pré-introdução, o pano de fundo mais distante, de todo este trabalho. Uma coisa que, certamente, esta "sociedade da informação" tem produzido é informação, volumes incomensuráveis de conteúdos informativos, páginas e páginas de livros impressos ou virtuais. Este "monstro informativo" está ficando cada vez mais gigantesco, por cada vez que o planeta completa uma órbita em torno do seu próprio eixo. Assim surge frequentemente uma inevitável questão a todos nós:

"Onde se encontra a informação de que eu necessito, neste momento ?"

Este trabalho enquadra-se, num domínio de acção que tenta dar resposta a esta, cada vez mais presente e frequente, questão. A procura da informação relevante e necessária em tempo útil torna-se cada vez mais uma necessidade crítica.

Cada vez é mais difícil encontrar informação relevante na crescente "selva" de informação não estruturada, disponível. Quando, no final dos anos 60, surge o projecto ARPANET, o embrião daquilo que viria a ser a Internet, ninguém imaginava o tão rápido crescimento, disseminação e divulgação desta rede de computadores. Durante as duas décadas que se seguiram (70, 80), um verdadeiro exército de especialistas na área das Ciências da Computação, contribuíram para este crescimento. Por um lado os meios infraestruturais aumentaram e por outro os serviços disponibilizados, sobre estes meios, também foram aumentando. Vários protocolos foram desenhados e implementados, entre estes destacam-se os tão conhecidos: TELNET, UUCP, TCP/IP. Também o numero de países ligados a esta crescente rede de computadores subiu consideravelmente, neste período. Todavia, o que mais contribuiu para que a Internet se tornasse naquilo que hoje é, foram os vários serviços implementados, sobre esta rede, e disponibilizados para os utilizadores: e-mail, GOPHER, WWW.

Em 1991 o aparecimento da *"World Wide Web"* (WWW) veio dar um novo e potente impulso ao crescimento da Internet. Desenhado por Tim Berners-Lee no CERN, este sistema providência a possibilidade de ser publicado um conjunto de ficheiros de texto, denominados de páginas, com hiperligações mútuas. Este novo conceito de apresentar a informação teve um grande acolhimento por parte de um cada vez maior numero de utilizadores. A partir daí o numero de páginas publicadas na Internet, sob este sistema, cresceu exponencialmente. Utilizadores das mais diversas áreas começaram a publicar as suas páginas: cientistas, empresários, políticos, etc. Várias organizações humanas passaram também a publicar o seu "rosto electrónico", neste revolucionário meio de informação. Toda esta adesão humana, em massa, a estes novos meios, veio fazer com que actualmente tenhamos um pesado numero de conteúdos publicados no WWW. O gráfico que se apresenta em baixo mostra a evolução do numero de portais disponíveis no WWW.

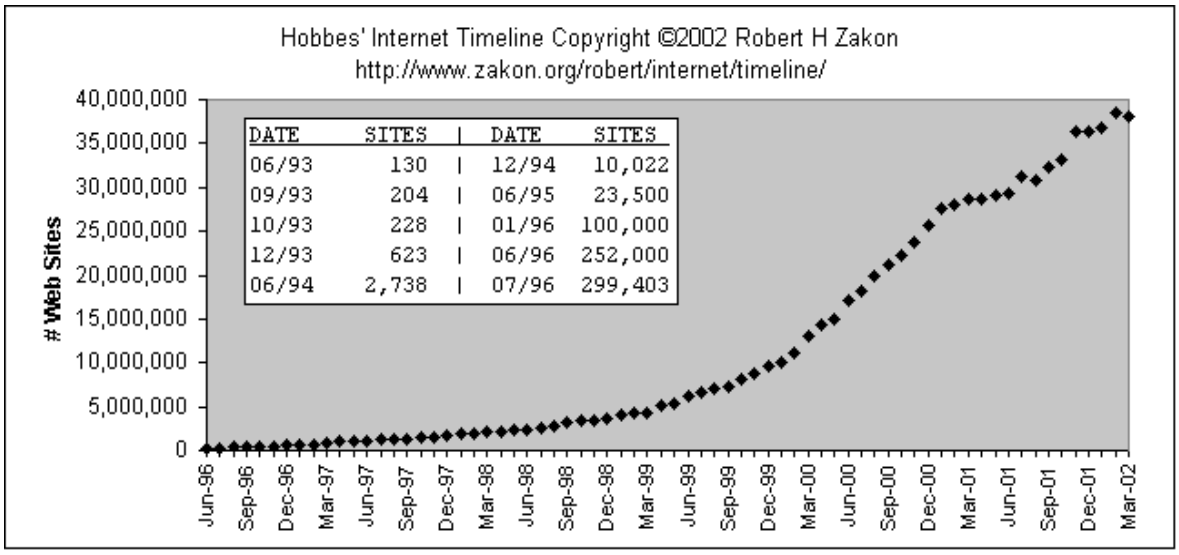


Figura 1.1 - Evolução de número de *Web Sites*.

Ninguém sabe ao certo qual o numero exacto de páginas disponíveis, neste sistema, todavia estima-se que andará por volta das 3,000,000,000, segundo dados de Maio de 2003.

Perante esta realidade torna-se bem evidente a necessidade de meios automáticos de procura de informação. A partir dos meados da década de 90 começaram a surgir os denominados "motores de pesquisa". Estes são pequenas aplicações informáticas, disponíveis em determinados sites/portais[1], que numa descrição simples, possibilitam a obtenção de um

1 Site – termo anglo-saxónica para designar um local no WWW (em português o termo "portal" costuma ser empregue com o mesmo significado).

conjunto de “links” (URL' s) de páginas contendo informação relacionada com aquilo que um utilizador procura. Por exemplo, páginas sobre: história, arte, ciência, política, etc. Embora muito úteis e suficientes em muitas situações, os motores de pesquisa têm se mostrado incapazes e muito limitados para satisfazer determinados tipos de pesquisas mais exigentes e especificas. Assim tem surgido toda uma comunidade científica, da área da Inteligência Artificial, que se debruça sobre esta problemática e tenta encontrar soluções inovadoras que ultrapassem as limitações dos motores de pesquisa.

1.2 Motivações

Atendendo ao estado actual da Internet, no que diz respeito aos conteúdos publicados no WWW, são de vital importância as ferramentas que pesquisam e extraem, de forma “Inteligente”, informação da Web. O problema pode colocar-se da seguinte forma: embora existam mais de 3,000,000,000 de páginas disponíveis no WWW e seja possível armazenar a quase totalidade destas numa máquina com capacidades adequadas, esta não “entenderá” nenhuma destas páginas. O termo “entender”, empregue anteriormente, tem uma conotação de representação semântica, isto é, embora uma máquina possa conter as páginas referidas, esta poderá não possuir qualquer representação semântica das mesmas. Poderá não ser capaz de reconhecer elementos considerados relevantes e contidos num subconjunto de páginas, por exemplo, para assim os extrair e armazenar nalguma forma ou estrutura representativa de alguma semântica.

A informação contida numa página do WWW é essencialmente texto, para além das marcas do HTML, se for uma página HTML, ou outras marcas SGML quaisquer que se encontram subjacentes na página e que são invisíveis para o utilizador. No essencial estamos a trabalhar com texto que na maior parte dos casos não contém qualquer estrutura implícita. É um documento de texto, escrito em linguagem natural, tal como o Português, sobre um determinado tema ou assunto. Temos portanto aquilo que é denominado de “*informação não estruturada*”, ao contrário daquilo que temos, por exemplo, numa Base de Dados, que é informação estruturada. São bem conhecidas as vantagens de ter informação estruturada quando surge a necessidade de pesquisar algo, no caso duma base de dados pode-se efectuar

uma query que responde à procura pretendida, mas se estamos perante a informação não estruturada, então o que fazer?

Uma das abordagens à questão anterior consistem em tentar gerar informação estruturada a partir da informação não estruturada existente, isto é, estruturar a informação. Nos últimos anos tem sido investido um enorme esforço, por parte de vários ramos das Ciências de Computação, para criar métodos, ferramentas, sistemas, capazes de estruturar automaticamente informação. Neste contexto o objectivo é dotar a informação não estruturada, como é o caso dum texto, de alguma estrutura, focando aquilo que é considerado semanticamente relevante, num dado domínio e par um objectivo especifico.

Esta é a principal motivação deste trabalho – é conceber um sistema capaz de conseguir extrair determinados elementos considerados relevantes, a partir duma determinada classe de documentos de texto. Embora já exista algum trabalho realizado nesta área, algum do qual é referido na secção 2.2 desta dissertação, não têm sido muito aquele que emprega técnicas de "aprendizagem máquina" ou aprendizagem automática", para a realização dessa tarefa.

O objectivo deste trabalho consistem em explorar um conjunto de técnicas de aprendizagem automática, no problema da extracção de informação a partir de texto, para um domínio concreto escolhido previamente (anúncios de venda de habitação) e na língua portuguesa, mostrando que o utilização dessas técnicas da IA é conveniente e consegue obter relativamente bons resultados, de maneira expedita.

1.3 Resumo dos capítulos

Apresentamos aqui uma breve descrição da estruturação do resto desta dissertação e do conteúdo de cada capítulo que se segue.

No primeira parte do capítulo dois (2.1) apresenta-se uma contextualização mais pormenorizada do domínio, a problemática em si, que motiva todo o trabalho. A segunda parte do capítulo dois (2.2) faz referência a algum trabalho já realizado nesta área, descrevendo alguns sistemas desenvolvidos.

O capítulo três inicia com uma apresentação geral da *Inteligência Artificial* e da subárea denominada de *Aprendizagem Automática*, descrevendo depois as principais técnicas experimentadas no trabalho implementado no âmbito da dissertação, tais como *Aprendizagem Baseada em Instâncias* (3.2), *Aprendizagem Bayesiana* (3.3) e *Indução de Árvores de Decisão* (3.4).

O capítulo quatro é o núcleo de toda a dissertação, apresentando o trabalho desenvolvido. Este capítulo encontra-se funcionalmente organizado segundo dois grandes grupos, correspondendo a duas questões abordadas – uma na área do "*Information Retrieval*"[1] (4.2) e outra na área do "*Information Extraction*" (4.3) – sendo esta última a que constitui o ponto principal do trabalho. A primeira secção do capítulo apresenta o domínio escolhido, para a implementação do trabalho e a última secção (4.5) descreve os pormenores dos sistemas implementados, quer a nível de estrutura e organização, quer de funcionamento.

No capítulo cinco são apresentados os resultados obtidos, para cada um dos sistemas implementados (na classificação de anúncios e na extracção de elementos) e resultantes do conjunto de experiências e testes realizados.

O capítulo seis conclui o trabalho e aponta possíveis melhoramentos futuros, que poderão ser continuados a partir deste.

1 Ver secção 4.1

2 Web / Text Mining

Com crescimento exponencial de informação disponível na World Wide Web, que se tem verificado nos últimos anos, torna-se imperioso que existam ferramentas eficazes na procura de informação considerada relevante. Existe já todo um trabalho que tem vindo a ser desenvolvido, essencialmente após a segunda metade da década de 90, nesta área. Também bastante esforço de investigação tem sido despendido, dando origem a várias técnicas que são já bem conhecidas entre os investigadores, nesta área. Como esta nova área é identificada como "Web/Text Mining", iremos apresentar um breve historial, esclarecendo assim as suas origens.

A expressão "Web Mining", é muito recente e tem a sua origem na expressão "Data Mining". A "Data Mining" designa uma área de trabalho e investigação, pertencente à Inteligência Artificial", que tem como principal objectivo a descoberta de conhecimento, de estruturas e relações no seio dos conteúdos das Bases de Dados. Esta expressão está associada a outra também muito utilizada recentemente – "Knowledge Discovery". De entre as várias definições de Data Mining, propostas por vários autores, existe uma que é aceite na generalidade:

> *A extracção implícita de conhecimento, previamente desconhecido, e potencialmente útil, a partir de dados.*

Nesta área, o alvo é dirigido à descoberta de conhecimento a partir dos dados, conhecimento esse que será posteriormente utilizado no apoio à tomada de decisões. Por exemplo, para profissionais da área do Marketing, é de uma enorme importância conhecer os principais hábitos dos consumidores. Para um político o possuir um determinado conhecimento acerca dos padrões comportamentais de uma população, pode ser vital para o lançamento de uma nova estratégia. Desde o aparecimento das Bases de Dados que muita gente procura obter o conhecimento subjacente e escondido nos dados que povoam os enormes bancos de informação. Existe um novo repositório de informação emergente, que avança com grandes desafios – "World Wide Web" (WWW). Este é sem dúvida um gigantesco banco de informação, com o acréscimo de a maioria dessa informação ser de carácter não estruturado,

no sentido de não existir nenhuma estrutura explícita associada à mesma, como acontece por exemplo num texto. A propósito disto, o WWW é metaforizado por alguns autores como sendo uma "selva de informação". Assim a expressão "Web Mining" consiste na pesquisa de conhecimento no World Wide Web – é o Data Mining orientado para o World Wide Web. O Web Mining é uma área de trabalho já com um tamanho considerável e contendo um conjunto de subáreas mais especificas.

Embora exista ainda alguma falta de consenso quanto ao reconhecimento e definição das várias subáreas do "Web Mining", a maioria dos autores é unânime em reconhecer a árvore que se mostra na figura 2.1, sobretudo os três primeiros nós da árvore.

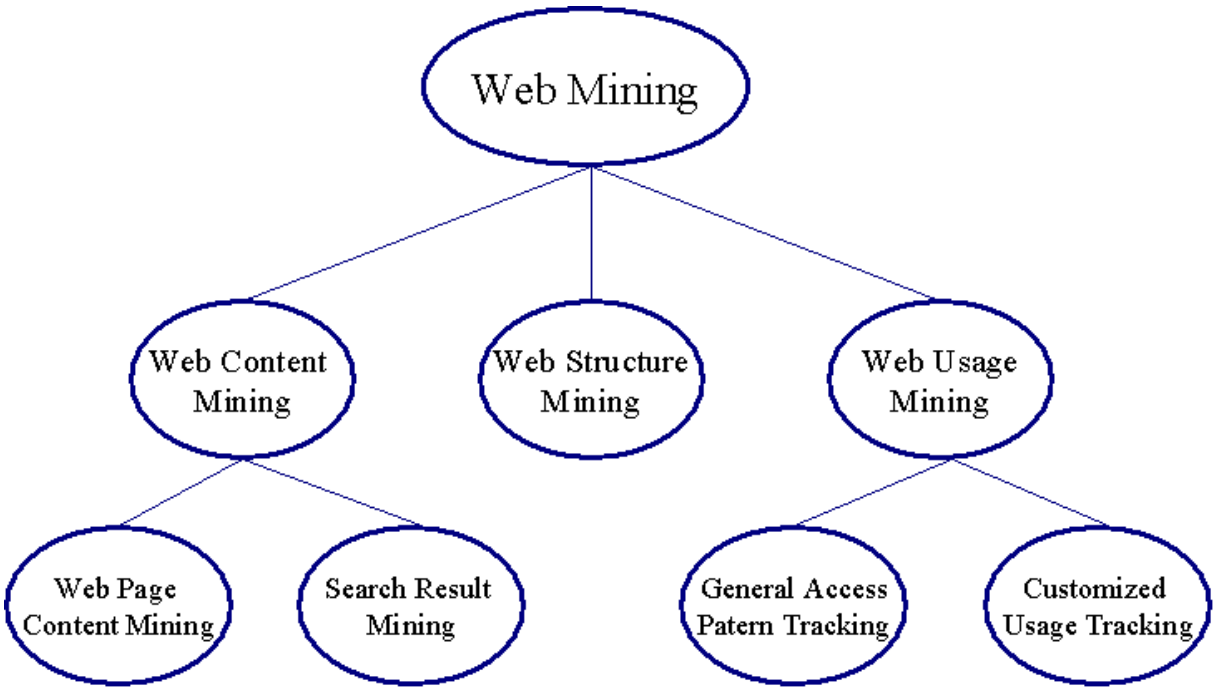


Figura 2.1 - Sub-domínios do *Web Mining*

Nas próximas secções passar-se-à a uma breve descrição das três principais áreas do Web Mining, representadas pelos três primeiros nós da árvore da figura anterior.

2.1 Web Mining

2.1.1 Web Content Mining

Neste subdomínio do Web Mining aquilo que está em causa é o conteúdo das páginas, em si. São estudadas metodologias de procura de paginas que obedeçam a um determinado critério, como é o caso dos motores de pesquisa, paginas relevantes e informativas para um determinado tópico ou assunto. Por outro lado são também estudadas metodologias de extracção de sub-partes de páginas. Existem, portanto, duas grandes preocupações ou orientações de investigação neste sub-dominio: "*Information Retrieval*" e "*Information Extraction*"

Extracção de Documentos (Information Retrieval) – O tópico diz respeito à obtenção de páginas que obedeçam a determinados critérios de alguns utilizadores. A título de exemplo, suponhamos que uma empresa imobiliária pretende obter o maior numero de páginas da Web que contenham informação relacionada com a venda ou compra de apartamentos. Neste caso é necessário fazer uma escolha daquilo que realmente interessa, no meio de tantas potenciais páginas disponíveis. É evidente que um ser humano pode fazer este trabalho, mas o problema está no tempo que uma pessoa necessita de despender para realizar esta tarefa. Assim existe um crescente interesse em todos os métodos automáticos que realizem este trabalho e cujo desempenho se aproxime o mais possível do conseguido pelos humanos.

Extracção da Informação (Information Extraction) – Este subdomínio tem como alvo a obtenção ou extracção de elementos pertencentes a determinados documentos [Riloff & Lehnert 1994]. Por outras palavras o input do sistema é uma sequência de texto e o output é uma representação dos elementos relevantes a extrair [Hobbs & Israel 1994]. Enquanto que o "Information Retrieval" pretende extrair documentos importantes, entre um universo de documentos possíveis, o "Information Extraction" pretende identificar elementos relevantes no interior de determinados documentos, os quais já sabemos que contém a informação que nos interessa. Os elementos relevantes extraídos serão depois armazenados nalguma estrutura previamente definida, por exemplos numa tabela de uma Base de Dados. As estruturas nas quais são armazenados os elementos extraídos, dos documentos, costumam ser designadas na

literatura por "*templates*" [Hobbs & Israel 1994]. Cada template é composto por um conjunto de elementos designados por "campos" [Riloff & Lehnert 1994], o equivalente aos campos duma tabela numa base de dados. Para melhor ilustrar isto, apresenta-se de seguida um exemplo do domínio do Information Extraction: Suponhamos que o alvo de um sistema deste género era analisar uma série de documentos, contendo cada um deles notícias relativas a operações financeiras e de investimento. Um exemplo de dois documentos, deste domínio, contendo informação relevante é mostrada na figura 2.3.

O objectivo seria extrair os elementos mais relevantes que compõe um investimento ou uma participação financeira de uma organização noutra. Assim esses elementos poderiam ser: a entidade investidora, o alvo do investimento, o valor do investimento e possivelmente o país no qual o investimento é alvo. Um template num problema deste género seria um como o apresentado de seguida:

```
TEMPLATE: Investimento
      Investidor
      Alvo
      Valor
      País
FIM TEMPLATE
```

Figura 2.2 - Exemplo dum template

Neste caso o template denominar-se-ia de Investimento e possuiria quatro slots: *Investidor*, *Alvo*, *Valor*, *País*. O objectivo seria então preencher este template automaticamente mediante os novos documentos analisados.

Figura 2.3 - Exemplos de anúncios.

Na figura anterior (2.3) os elementos que se encontram sublinhados são aqueles considerados relevantes para o preenchimento do template definido anteriormente. A próxima figura mostra o template preenchido com duas linhas, relativas aos elementos extraídos dos dois documentos apresentados anteriormente.

INVESTIDOR	*ALVO*	*VALOR (€)*	*PAÍS*
Jerónimo Martins	**«Sé»**	**492 00 000**	**Brasil**
Sonae Imobiliária	**?**	**110 000 000**	**Espanha**

Figura 2.4 - Template preenchido com dois exemplos.

Uma grande quantidade de esforço de investigação tem sido investida nos dois pontos descritos anteriormente, nos últimos anos. O forte desejo de aproximar o desempenho dos sistemas de “Information Retrieval” e “Information Extraction”, para níveis próximos dos do desempenho humano, tem motivado o aparecimento de campeonatos ao jeito de conferências, nos últimos anos. A partir de 1990 começaram a ser disputados os campeonatos TREC – “Text Retrieval Conference” para a “modalidade” de Information Retrieval, e MUC – Message Understanding Conference” na “modalidade” de Information Extraction. Nestes

campeonatos têm participado grupos de várias universidades e alguns de empresas. Várias são já as edições decorridas dos TREC's e dos MUC's, as ultimas versões destas conferências realizaram-se em Novembro de 2002, a *"TREC-10"*, e em Abril de 1998 a *"MUC-7"*.

2.1.2 Outros Subdomínios do Web Mining

Web Usage Mining

Este sub-domínio tem como alvo descobrir os padrões comportamentais de determinados utilizadores, no que diz respeito à consulta de conteúdos na Web. É sabido que a maioria dos servidores que se encontra na Internet faz um registo dos acessos realizados aos mesmos [Mena 1999]. Portanto, para muitas organizações é muito importante conhecer os hábitos de acesso dos utilizadores, por exemplo: que utilizadores acedem a determinados portais e porquê; se um utilizador acede a um recurso **A**, será que ele também estará interessado no recurso **B**, que provavelmente desconhecerá ? São questões semelhantes a esta que orientam e motivam todo o trabalho realizado neste subdomínio. Para muitas empresas, como as relacionadas com actividades comerciais, este género de informações é muito valiosa. Também o conhecimento dos padrões de navegação dos utilizadores, num determinado "site", ajudam os gestores desses "sites" a restruturar melhor os mesmos: saber por exemplo quais os conteúdos mais solicitados, qual a informação que deve estar mais "visível" e aquela que poderá estar mais escondida. Em muitas situações interessa ter "web sites" cuja estrutura se adapta dinamicamente, de acordo com os padrões de utilização e preferência dos utilizadores. Noutras interessa informar o utilizador de algum link que lhe poderá ser útil ou de algum produto que este poderá desconhecer, como é o caso do bem conhecido site: http://www.amazon.com, que por vezes sugere algum livro ou outro qualquer produto no qual este possa estar interessado.

Web Structure Mining

Para além dos conteúdos disponibilizados pelas páginas da web, existe um conjunto de informação subjacente às mesmas que, para determinados fins, se reveste de uma grande

importância. Este subdomínio do Web Mining foca, essencialmente, questões relacionadas com a estrutura topológica do World Wide Web [Mena 1999]. Tendo presente que as páginas publicadas na Web são hipertexto, cada página poderá conter várias ligações para outras páginas e uma determinada página poderá ser referenciada por outras. Podemos então modelar este universo por uma estrutura a que os matemáticos denominam de grafo, ou mais especificamente um grafo orientado.

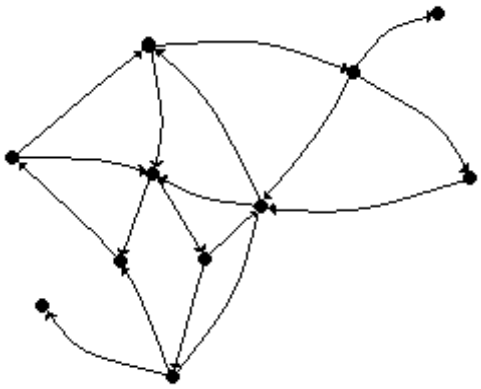


Figura 2.5 - Um grafo orientado

Sabendo que a teoria de grafos vem sendo desenvolvida desde Euler, contendo assim um legado histórico com mais de duzentos anos, é natural que esta seja utilizada neste subdomínio. Todavia têm-se feito trabalhos, recentemente, com o objectivo de expandir e adaptar a teoria de grafos a esta nova área. Novos tipos de grafos são investigados actualmente, como é o caso dos "*Grafos Aleatórios*", para melhor modelar a estrutura da Web. Neste subdomínio interessam questões como:

Quais as páginas que contêm mais ligações para outras?
Quais as páginas mais referenciadas ?

Uma página com muitas ligações para outras páginas, poderá ser indicativo da variedade de informação ou da riqueza de conteúdo dessa mesma página. Por outro lado e tal como nas citações bibliográficas, uma pagina que é muito referenciada, pode ser indicativo da importância ou popularidade da mesma.

Uma outra preocupação deste subdomínio é realizar aglomerações de páginas consoante determinadas propriedades ou segundo determinados assuntos [Berkhin 2002].

2.2 Text Mining

Em 1993, na conferência MUC-5, entre todos os sistemas propostos para realizar as tarefas de extracção exigidas, somente dois sistemas apresentavam a capacidade de gerar automaticamente regras de extracção: *"AutoSlog"* e o sistema *"PALKA"*. Desde essa data e essa conferência, muitos têm sido os sistemas apresentados com a capacidade de produzir alguma forma de regras de extracção. Entre os mais conhecidos temos os seguintes: CRYSTAL [Soderland 1996], LIEP [Huffman 1995], RAPIER [Califf 1999], WISHK [Soderland 1999], SRV [Freitag 1998], STALKER [Muslea et al. 1998], HASTEN [Krupka 1995].

Alguns destes métodos baseiam-se em dicionários, outros em métodos mais estatísticos, como por exemplo os baseados em modelos de Markov (HMM) [Peng 2000], e uma grande parte tem como fundamento a geração de regras para extracção. De seguida passarei a descrever alguns dos sistemas que foram referidos no parágrafo anterior.

2.2.1 AutoSlog

Este foi o primeiro sistema de extracção baseado na criação de um dicionário de extracção. Esse dicionário consiste numa colecção de padrões de extracção designados por "*concept nodes*". Este "*concept node*" consiste numa palavra de activação e nalguma restrição linguística a ser aplicada ao texto a extrair.

```
Name: Target-subject-passive-kidnapped
Trigger: Kidnapped
```

Figura 2.6 -Um concept node do sistema AutoSlog

Desenvolvido por Ellen Riloff em 1993 [Riloff 1993], este sistema opera com algumas heurísticas de padrões de linguagem. É treinado com um conjunto de exemplos anotados previamente e a palavra que desencadeia a extracção normalmente é um verbo, podendo também ser um substantivo se o objecto a ser extraído for um sintagma nominal (noun

phrase). Assim, os elementos a extrair poderão pertencer a uma das seguintes categorias sintácticas: *subject*, *direct object* ou *noun phrase*. As frases são primeiro analisadas com um analisador sintáctico e o resultado é depois confrontado com a matriz de heurísticas que determinará se será realizada uma extracção mediante a proximidade à melhor heurística.

Em 1995 Riloff apresentou uma versão melhorada do sistema onde foram incluidas três alterações mais significativas [Riloff 1996]. Por um lado deixou de ser necessário anotar todos os exemplos de treino, mas só aqueles que eram considerados relevantes para o domínio. Uma outra alteração importante foi o aumento do numero de heurísticas relativas a padrões de extracção. Por outro lado o numero de padrões de extracção possíveis de ser activados, passou a ser superior a um. São realizados alguns cálculos estatísticos adicionais, em relação à versão mais antiga, para criar um ranking com os padrões de extracção, guardando depois os melhores no dicionário.

2.2.2 HASTEN

O Hasten foi um sistema desenvolvido por George Krupka [Krupka 1995] e participante na conferência MUC-6 com distinção. Este sistema induz um conjunto de padrões de extracção a partir dos dados de treino (pequenas notícias num domínio). Cada padrão de extracção é designado por "*Egraph*" e é o equivalente ao "concept node" do sistema *AutoSlog*, descrito anteriormente, embora um pouco mais elaborado. A próxima figura mostra um exemplo dum "*Egraph*", para o evento dum ataque terrorista (ex: "*The Parliement was bombed by the guerrilas*"):

```
BOMBING:
  TARGET:        NP "semantic = physical-object"
  ANCHOR:        VG "root = bomb"
  PERPETRATOR:   NP "semantic = terrorist-group"
```

Figura 2.7 - Um *Egraph* do sistema *Hasten*

Cada *Egraph* expressa uma lista de pares (*Etiqueta Semântica*, *Elemento Estrutural*), definindo um conjunto de restrições à identificação dos elementos relevantes. O *Elemento Estrutural* define restrições semânticas ou sobre os elementos estruturais das frases, como por exemplo a sua sintaxe. Na figura anterior, um exemplo dum pare destes é: (TARGET, NP

"semantic = physical-object"), definindo uma restrição sobre o alvo do elemento a extrair: um sintagma nominal (NP), cujo sujeito é um objecto físico. Uma etiqueta comum a todos os Egraphs é a designada por "ANCHOR". Nesta etiqueta é definida a condição principal de identificação dum elemento relevante num texto, e é essencialmente constituída pela raíz do verbo que caracteriza a acção, por exemplo "root = bomb" na figura anterior. Para cada exemplo positivo de treino, são ajustados pesos relativos a cada *Egraph*, pesos esses que serão depois utilizados no processo de identificação dos novos elementos relevantes.

Para cada nova instância é feito um cálculo usando uma métrica de similaridade relativa a cada *Egrap*h. Este cálculo, tem também em consideração os pesos ajustados no processo de treino, estabelece um ranking para cada *Egraph* que traduz o quanto este é aplicável à extracção numa determinada instância de texto em análise. Se a similaridade duma instância a um *Egraph*, for superior a um certo limiar, fixado previamente, então é concretizada a extracção.

2.2.3 CRYSTAL

Este sistema surge em 1995 e é da autoria de Stephen Soderland [Soderland 1996]. Para além de ser um sistema baseado na técnica de criar um dicionário de extracção, com restrições morfológicas, sintácticas e de conceito, vai tentar induzir um conjunto de regras de extracção. Este sistema foi testado, inicialmente, em dois domínios: noticias relativas à sucessão de gestores de empresas, publicadas no "*Wall Street Journal*", sendo o objectivo a extracção dos intervenientes numa sucessão de administradores de organizações, e o segundo em pequenos resumos clínicos, com o alvo de identificar possíveis diagnósticos ou sintomas.

Inicialmente as frases dos vários exemplos são etiquetadas sintacticamente e são preenchidos uns templates que irão conter a frase segmentada e sintacticamente etiquetada. Na figura 2.8 é mostrado um exemplo de um destes templates, já preenchido. Tal como se pode visualizar no exemplo, para além de uma etiquetagem sintáctica é feita também um etiquetagem semântica, tendo em atenção o domínio em causa. É feita uma tentativa de identificar a ocorrência de certas classes de conceitos, nos elementos do texto, como por exemplos: <Generic Person>, <Person Name>, <Generic Organization>, <Event>, <Corporate Office>. Assim, da colecção

de exemplos de treino, apresentados ao sistema é feito, logo numa primeira fase, alguma generalização, com esta etiquetagem e preenchimento dos templates. É evidente que estes não são os templates finais a serem preenchidos com os elementos relevantes extraídos, mas irão ser as instâncias de treino do CRYSTAL. Observa-se também que estes templates consistem na definição de conceitos, como por exemplos o conceito de "*Instância*" , como é o caso do apresentado na figura 2.8, o conceito *Sucessão*, etc.

```
Input Sentence:
   He succeeds Jack Harper, a company founder who was
   named chairman.

CRYSTAL Instance:
   SUBJ:
      Terms:   HE
      Classes: <Generic Person>
      Mode:    affirmative
   VERB:
      Terms:   SUCCEEDS
      Root:    SUCCEED
      Mode:    active, affirmative
   OBJ:
      Terms:   JACK_HARPER %COMMA% A COMPANY FOUNDER
      Classes: <Person Name>, <Generic Organization>,
               <Generic Person>
      Mode:    affirmative
   REL-OBJ:
      Terms:   WHO WAS NAMED CHAIRMAN %PERIOD%
      Classes: <Past>, <Event>, <Corporate Office>
      Mode:    affirmative
```

Figura 2.8 - Template representando o conceito de *Sucessão*.

O sistema CRYSTAL utiliza uma versão do algoritmo de cobertura, para induzir regras com um certo grau de generalização. As restrições, impostas pelos templates iniciais, vão sendo relaxadas até que a cobertura dos exemplos de treino seja satisfeita com um certo erro máximo, fixado previamente. Um desafio enfrentado nesta área é saber qual o nível certo de generalização. Um nível muito específico faz com que existirão muitos novos exemplos que o sistema não conseguirá identificar e portanto extrair. Por outro lado um sistema muito genérico, fará com que nos novos exemplos apresentados, sejam extraídos elementos não relevantes, isto é, lixo. Um bom sistema será aquele que induza regras com um grau de generalização equilibrado e adequado. Assim, a estratégia utilizada pelo CRYSTAL é partir de regras ou conceitos muito específicos e particulares indo relaxando estes mesmos

conceitos até que o maior numero de exemplos de treino sejam cobertos e um certo nível de erro não seja ultrapassado. Para cada novo passo de generalização é feito o teste nos exemplos e medido o erro que determinará a paragem ou continuação do processo de indução.

Uma das preocupações deste sistema foi o de conseguir ser expressivo, no que diz respeito às regras de extracção.

2.2.4 LOLITA

O Sistema LOLITA é um projecto em desenvolvimento no laboratório de engenharia de linguagem natural, da Universidade de Durham, desde 1986. O nome deste sistema significa "*Large-scale, Object-based, Linguistic Interactor, Translator, and Analyser*" [Garigleano et al. 1998] e o sistema participou nas conferências MUC mais recentes (MUC-6 e MUC-7) com um sucesso considerável.

Este é um sistema de processamento de linguagem natural com fins genéricos, não tendo sido desenhado com o objectivo de satisfazer algum fim específico, nalgum domínio. O núcleo do sistema consiste numa complexa e vasta rede semântica ("SemNet"), com cerca de 100000 nós e 1500 regras gramaticais, inspirada na já célebre "WordNet" [Miller 1990]. Este sistema é considerado um dos maiores sistemas de processamento de linguagem natural [Garigliano et al. 1998] e serve de motor para um conjunto de aplicações implementadas que vão desde a tradução de sentidos entre línguas (ex: entre Inglês e Italiano) até à extracção de informação, com é o caso do trabalho descrito em [Constantino 1997]. A adaptação e utilização do sistema num domínio específico é feita com relativa facilidade [Constantino, 1997], através da definição dum conjunto de "Templates" de acção para o fim concreto. Um "Template" é uma estrutura já descrita e contendo um conjunto de campos (slots) a serem preenchidos segundo regras explicitamente definidas no próprio template.

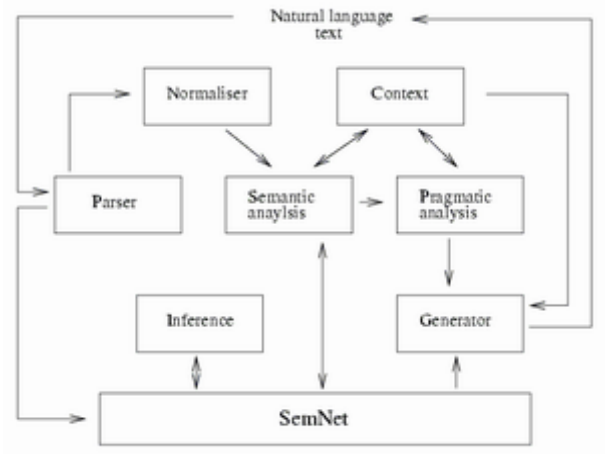


Figura 2.9 - Núcleo do sistema LOLITA

O texto introduzido é processado pelos módulos, representados na figura 2.9 e é gerada uma representação interna do texto, formalmente coerente com a "SemNet". O processamento realizado sobre o texto inclui análise morfológica (separação de palavras, etc), sintáctica e semântica, ultrapassando dificuldades como a resolução de anáforas.

O núcleo do sistema é constituído por um conjunto de módulos que interagem entre si e acima de tudo com o módulo principal e já referido - "SemNet" - tal como e mostrado na figura anterior, obtida de [Constantino 2001]. Um nó, da "SemNet", pode estar ligado a um subconjunto de outros nós da rede, à semelhança do que acontece num grafo orientado acíclico. Cada nó representa um conceito relativo a alguma entidade ou evento, por exemplo: "a entidade empresa", "o evento compra", etc. Cada ligação, entre dois nós, representa uma relação entre esses nós, como por exemplo relações de sujeito (ex: "João"), objecto (ex: "jornal"), acção (ex: "comprou"), especialização/generalização (ex: "FIAT" versus "Empresa"), instanciação/universalização, etc. Existem cerca de 60 tipos de ligações diferentes. A figura 2.10 que se segue mostra a representação da frase "*John will retire as chairman*", na SemNet.

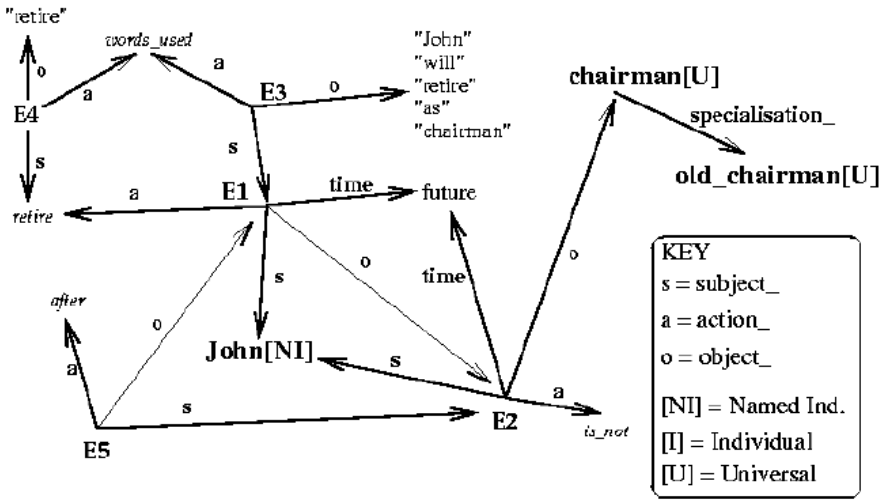


Figura 2.10 - Extracto da SemNet representando a frase:
"John will retire as chairman" [Garigliano et al. 1998]

O trabalho realizado pelo autor, consistiu em utilizar o sistema LOLITA para a extracção de informação no domínio dos mercados financeiros. Os textos processados consistiram em pequenos anúncios relativos a operações financeiras, como por exemplo o anúncio da compra de uma empresa, por parte de outra, ou quem passou a liderar o conselho administrativo duma empresa. Através da implementação de "templates", mencionado anteriormente, são definidos os conceitos e elementos relevantes a extrair. Uma das preocupações deste trabalho consistiu em tornar a definição dos templates facilmente realizável por qualquer utilizador.

```
Template-name:          T=TAKEOVER
Variables:              V=COMPANY1 is a company.
                        V=COMPANY2 is a company.
                        V=VALUE is money.
Template main-event:    V=COMPANY1 acquired V=COMPANY2.
                        V=COMPANY1 acquired V=COMPANY2 with V=VALUE.
                        The acquisition of V=COMPANY2 by V=COMPANY1.
                        The V=VALUE acquisition of V=COMPANY2 by V=COMPANY1.
                        V=COMPANY1 paid V=VALUE for V=COMPANY2.
                        V=COMPANY1 acquired a majority stake in V=COMPANY2.
                        V=COMPANY1 took full control of V=COMPANY2.
Definition of slots:
S=COMPANY-PREDATOR:     V=COMPANY1
S=COMPANY-TARGET:       V=COMPANY2
S=TYPE-OF-TAKEOVER:
        String-fill: HOSTILE T=TAKEOVER is hostile.
        String-fill: FRIENDLY otherwise
S=VALUE-OF-TAKEOVER:    The cost of T=TAKEOVER.
                        V=VALUE
```

Figura 2.11 - Parte duma template relativa ao evento da aquisição duma empresa, retirado de [Constantino, 2001]

Como já foi adiantado, uma template tem alguma semelhança com uma tabela duma base de dados, com o acréscimo de ser dotada de regras de preenchimento gerais da template e regras

de preenchimento dos campos, aqui designados por “slots”. A figura 2.11 mostra o extracto duma template (não estão todos os slots), relativo ao evento da aquisição duma empresa, por parte de outra. As strings com o prefixo “V=” designam variáveis e as começadas por “S=” definem slots. As linhas contidas na parte designada por “**`Template main-event:`**” definem regras para a identificação do evento e por sua vez as contidas em “**`Definition of slots:`**” definem os slots da template com as condições de preenchimento desses slots.

Este sistema é um sistema cujas regras de extracção de elementos relevantes são definidas pelos utilizadores, através das templates. O grande poder do sistema LOLITA reside na sua base de conhecimento interna, sob a forma de uma rede semântica, designada por “SemNet”, permitindo um bom processamento sobre a linguagem natural.

3 Métodos e Técnicas de Inteligência Artificial

Neste capítulo, após uma breve introdução e visão panorâmica da Inteligência Artificial (IA) e da sua subárea designada por *Aprendizagem Automática* (Machine Learning), serão apresentadas os principais conceitos e técnicas de IA envolvidos no desenvolvimento do presente trabalho.

3.1 Inteligência Artificial

> "*The branch of computer science that is concerned with the automation of intelligent behavior*"
>
> [Luger & Stubblefield, 1993]

De entre as várias definições de Inteligência Artificial a anterior traduz as orientações de uma das perspectivas mais seguidas na actualidade [Russell & Norvig 1995] e para a qual um sistema inteligente consiste num sistema que age racionalmente – para uma grande maioria de autores é o que é esperado dum sistema deste género.

Ao todo existem quatro prismas de olhar a IA, duas mais centradas nas semelhanças com os humanos (sistemas que pensam como humanos; sistemas que agem como humanos) e duas baseadas no conceito de racionalidade (sistemas que pensam racionalmente; sistemas que agem racionalmente), donde se enquadra esta ultima definição. Desde a origem formal da IA como ciência, em 1956, que a maneira de ver a IA tem passado por estas quatro prismas.

Embora para alguns autores as origens da IA remontem ao período clássico grego, com Platão e Aristóteles [Dreyfus 1979], é no contexto científico e tecnológico do final da II Guerra Mundial que se dá a fecundação e gestação desta nova ciência. De entre vários, destacam-se nomes, pelo seu contributo mais directo, como: Alan Turing, McCulloch, Pitts, Shannon, Von

Neuman, Minsky e McCarthy através do qual foi estabelecido o nome da IA, na célebre escola de verão de Dartmouth, em 1956 [McCarthy et al. 1956].

Após este período embrionário da IA dá-se um período de euforia generalizada (final da década de 1950 e década de 1960), pensando-se que esta ciência iria, em relativamente pouco tempo, acabar por criar uma entidade computacional capaz de ultrapassar os desempenhos mentais próprios do cérebro humano, em qualquer domínio. Uma marca de referência comum da IA, neste período, era a do desempenho humano. Vários foram os problemas e metas avançadas pelos críticos, para os quais a solução dos mesmos por sistemas desenvolvidos, atribuiriam aos mesmos o estatuto de sistema inteligente. O mais comum na época eram afirmações do género: *um sistema será considerado inteligente se for capaz de fazer isto, ou aquilo* .... Os vários "*isto*" e "*aquilo*" foram sendo ultrapassados e novas metas eram de novo desafiadas. É neste contexto que surge o célebre "*Teste de Turing*", que consistem basicamente em confrontar um sistema com uma pessoa, num processo de diálogo, sem que a pessoa saiba se está a comunicar com uma máquina ou com outra pessoa. Se o sistema for capaz de iludir essa pessoa, fazendo-a pensar que está a comunicar com um ser humano, então podemos considerar esse sistema inteligente.

No final da década de 1960 e inicio da década de 1970 a euforia veio a dar lugar a um visão mais realista da IA, contemplando um conjunto de dificuldades e limitações, inerentes ao mundo real, que não tinham sido ainda tidas em conta. De entre as várias dificuldades, destaca-se a tomada de consciência da existência de problemas pertencentes à classe dos NP-Completos, sendo necessárias heurísticas concretas para os atacar.

A década de 1970 viu florescer os denominados *sistemas periciais* que constituiriam um grande sucesso, quer comercial quer publicitário, em várias áreas de aplicação. De entre os muitos sistemas deste género, implementados nesta altura, destacam-se os seguintes: *DENDRAL*, na área da Química, para inferir estruturas moleculares, com base na análise do espectro de massa, *MYCIN* para diagnóstico de infecções hematológicas e o célebre, bem sucedido e muito publicitado *PROSPECTOR* que se destinava à descoberta de grandes jazigos de Molibdénio.

Na década de 1980 a IA entra nos meios empresariais. Muitas empresas, como a *Digital*, *Xerox*, *AT&T*, *IBM*, entre outras, possuíam o seu grupo próprio de IA. No final desta mesma década renasce o interesse pelas redes neuronais.

Na actualidade a IA compreende um conjunto de subáreas, que vão desde a robótica até ao processamento de linguagem natural, passando pela aprendizagem automática, entre outras. Embora para muitos críticos o *Teste de Turing* ainda não tenha sido satisfeito, os feitos alcançados por esta ciência são notáveis e têm implicações em muitas áreas da nossa vida quotidiana. O exemplo do carro que realiza uma viagem autónoma, entre duas cidades, do "*Deep Blue*" capaz de vencer o campeão do mundo, no Xadrez, ou das sociedades de agentes que cada vez mais povoam a web, concretizando tarefas de pesquisa ou comércio electrónico, são alguns dos feitos alcançados.

Certamente que todos concordarão que esta é uma área que ainda está no inicio duma longa carreira que ninguém sabe muito bem onde nos levará, embora ficcionada por muitos, da literatura ao cinema, mas que continua em marcha e cada vez com mais implicações para a humanidade.

3.2 Aprendizagem Automática

O *Teste de Turing* envolve pelo menos quatro grandes subáreas da IA [Russell & Norvig 1995]:

- Processamento de Linguagem Natural
- Representação do Conhecimento
- Raciocínio automático
- Aprendizagem Automática

portanto, a última é, sem dúvida, uma importante subárea da IA.

Pretende-se que os sistemas criados tenham a capacidade de adquirir conhecimento, relativo ao ambiente em que se integram, conhecimento esse que não foi inicialmente fornecido. Pretendem-se sistemas capazes de formular teorias a partir de dados relativos ao seu ambiente. Na literatura mais recente da IA, o termo "agente" é empregue para designar um sistema que utiliza metodologias da IA[1]. Assim um agente capaz de aprender é um agente capaz de se adaptar melhor ao ambiente e com maior capacidade de sucesso, relativamente ao objectivo para o qual foi desenhado.

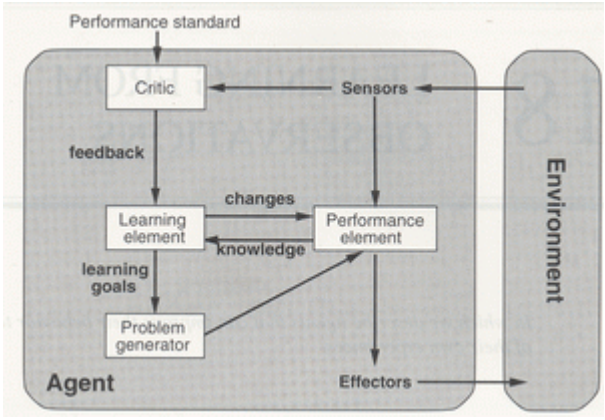


Figura 3.1 - um agente que aprende

De acordo com [Russell & Norvig 1995] o modelo geral de um sistema ou agente com capacidades de aprendizagem pode ser esquematizado como mostrado na figura anterior.

1 Este termo tem ganho uma importância reforçada, com o aparecimento dos *Sistemas Multi-Agente*.

Portanto, um agente é desenhado com um determinado objectivo que, na generalidade dos casos, depende de um conjunto de problemas existentes e que é necessário solucionar, tal como se pode perceber no esquema da figura 3.1. O agente obtém a percepção do ambiente através de sensores e desencadeia acções, nesse mesmo ambiente, através de efectuadores ou operadores que controla. Aquilo que distingue um agente cognitivo é a capacidade de gerar, estender e reformular a sua base de regras, como resultado da interacção entre este e o ambiente e guiado pelo objectivo de melhorar progressivamente o seu desempenho.

A aprendizagem é algo muito próprio e especial do ser humano e que todos nós reconhecemos como sendo muito importante. Desde muito cedo habituamos-nos a encetar processos de aprendizagem que nos permitem desenvolver e conhecer o universo que nos envolve e atingirmos os nosso objectivos, até sobreviver. Alargamos os nossos conhecimentos aprendendo em instituições afins que nos transmitem a matriz de saber adquirida no somatório das gerações passadas e que constituem um precioso património da humanidade, como é caso da ciência.

Existem duas formas de conhecimento e consequentemente duas formas cognitivas distintas, mas que se complementam – aquilo que na literatura se denomina de: “conhecimento simbólico” e “conhecimento sub-simbólico”. O primeiro traduz a aprendizagem de um conjunto de princípios lógicos, enquanto que o segundo traduza aprendizagem de um processo de acção – um “savoir faire”. A maior parte do conhecimento que adquirimos nas nossas escolas é simbólico, por exemplo quando estudamos História, Química ou Matemática, estamos adquirindo e assimilando ideias, conceitos e teorias, formalmente estabelecidas numa determinada linguagem inteligível. Por outro lado, quando aprendemos a caminhar, andar de bicicleta ou a jogar ténis, estamos aprendendo sub-simbolicamente, pois aprendemos simplesmente a saber fazer e por norma não nos é transmitida nenhuma teoria, mas nós vamos experimentando e melhorando o nosso desempenho.

Assim, também na IA estão presentes estas duas formas de aprendizagem: simbólico e sub-simbólico. Importantes progressos têm sido concretizados na aprendizagem, em qualquer destas formas. Tenta-se dotar os agentes de capacidades cognitivas simbólicas e/ou sub-simbólicas, consoante as necessidades, consoante o ambiente do agente e a natureza daquilo que é necessário aprender. No que diz respeito ao sub-simbólico temos, como exemplo de

técnicas empregues: as *Redes Neuronais*, a *Lógica Difusa*, a *Aprendizagem por Reforço*, a *Aprendizagem Bayesiana*, utilizada também neste trabalho, entre outras. No que diz respeito à aprendizagem simbólica, temos o exemplo da indução de classificadores, como é o caso das *árvores/regras de decisão*, também utilizadas neste trabalho, e a indução de regras lógicas (ILP). Em muitos domínios interessa que seja gerado conhecimento simbólico, capaz de ser facilmente percebido por quem interage com os sistemas, ou quem os usa.

A aprendizagem referida e utilizada no trabalho, é a denominada aprendizagem indutiva – aquela que é adquirida empiricamente (na linha de *Francis Bacon*, *David Hume* e *Bertrand Russel*, entre outros). Na IA, aprendizagem indutiva consiste em disponibilizar um conjunto de exemplos, denominados de treino, a um agente, de modo que este induza conhecimento relativo a alguma *função objectivo* ou *função a aprender*. Um exemplo ou instância de treino contém um conjunto de inputs e o resultado esperado: $\{(x_1, y_1),(x_2, y_2),...,(x_n, y_n)\}$. Pretende-se induzir algum f^*, tal que para todo, ou quase todo, o *i* se tem: $y_i = f^*(x_i)$.

Esta forma de aprendizagem também é classificada de aprendizagem supervisionada, pelo facto do agente ter acesso tanto aos inputs como aos outputs, isto é, o agente pode perceber o efeito duma decisão no ambiente ($f^*(x_i)$).

Nas três secções que se seguem serão apresentadas em mais detalhe, três formas de aprendizagem ("*Aprendizagem Baseada em Instâncias*", "*Aprendizagem Bayesiana*" e "*Indução de Árvores de Decisão*"), que foram utilizadas no nosso trabalho.

3.3 Aprendizagem Baseada em Instâncias

A aprendizagem baseada em instâncias é uma forma de aprendizagem que consiste basicamente em coleccionar e armazenar um conjunto de instâncias de treino para depois as utilizar no momento em que uma nova instância necessita de ser classificada. Este tipo de aprendizagem compreende um conjunto de métodos, dos quais se descreve a seguir o dos "*K-vizinhos*". Todo o trabalho computacional é adiado até ao momento em que é necessário classificar novas instâncias, daí estes métodos de aprendizagem serem designados de "*lazy*" (preguiçosos). Uma outra característica geral, destes métodos, é o facto de a classificação de uma nova instância ser realizada tendo em conta um subconjunto de instâncias treino, próximas da instância a classificar. Assim a função ou conceito objectivo a aprender, pode ser avaliada localmente, dependendo unicamente desse pequeno número de instâncias. Portanto para cada nova instância a classificar é avaliada uma função diferente, isto constitui uma diferença fundamental em relação a outros métodos de aprendizagem que tentam induzir uma função que caracterize todo o conjunto de treino observado.

A aprendizagem baseada em instâncias compreende essencialmente os três métodos seguintes: *Método dos k-vizinhos mais próximos, Regressão Local Ponderada ("Locally Weighted Regression")* e *Raciocínio Baseado em Casos*.

De seguida será apresentado o método dos k-vizinhos, dado que este é utilizado no nosso trabalho, na classificação de documentos, descrita na secção 4.2.1.

3.3.1 O Método dos K-Vizinhos Mais Próximos

O método dos k-vizinhos é um dos métodos mais simples dentre os métodos da aprendizagem baseada em instâncias [Mitchell 1997], este pressupõe que os domínios dos atributos são números reais e assim uma instância é interpretada como sendo um ponto no espaço euclidiano *n*-dimensional ($\mathbb{R}^n$), em que *n* corresponde ao número de atributos envolvidos num instância. Considerando que temos uma colecção de *m* instâncias $I = \{x_1, x_2, \ldots, x_m\}$ e

cada instância é expressa, atendendo ao valor de *m* atributos $A = \{A_1, A_2, \ldots, A_n\}$, então para uma qualquer instância $x \in I$, tem-se: $x \equiv \langle a_1(x), a_2(x), \ldots, a_n(x) \rangle$, em que, para qualquer *i*, $a_i(x)$ é o valor da instância x, relativa ao atributo A_i.

Sendo uma instância considerada como um ponto de $\mathbb{R}^n$, pode-se definir uma distância entre instâncias atendendo às métricas definidas nesse espaço, dentre as quais a mais utilizada é a *distância euclidiana*:

$$d(x, y) = \sqrt{\sum_i (a_i(x) - a_i(y))^2}$$

O objectivo deste método resume-se à aprendizagem duma função objectivo $f : \mathbb{R}^n \to C$, sendo *C* um conjunto finito que no caso de um problema de classificação contem as *r* classes possíveis: $C = \{c_1, c_2, \ldots, c_r\}$. Um conjunto de instâncias de treino consiste numa colecção de pares da forma $\langle x, f(x) \rangle$, sendo $x \in I$. Dada uma nova instância $x_?$ pretende-se estimar $f(x_?)$ e este estimador é normalmente notado por $f^{\wedge}(x_?)$.

Considerando que o conjunto *I*, referido anteriormente, consiste na colecção de instâncias de treino, então para uma nova instância que se pretende avaliar $x_?$, o estimador referido obtém-se da seguinte forma:

$$f^{\wedge}(x_?) \leftarrow \underset{c \in C}{argmax} \sum_{i=1}^{k} \delta(c, f(x_i))$$

em que δ é conhecida função delta de Kronecker, isto é $\delta(a, b) = 1$ se $a = b$ e $\delta(a, b) = 0$ se $a \neq b$. Portanto $f^{\wedge}(x_?)$ é estimado a partir do valor mais frequente de *f*, nas *k* instâncias mais próximas de $x_?$, daí este método ser designado dos "*k-vizinhos mais próximos*". Este *k* é um valor previamente fixado ou determinado dinamicamente por, validação cruzada, e geralmente um valor pequeno (*k = 3* ou *k = 5*).

Para o caso de $C = \mathbb{R}$ pode-se estimar $f(x_?)$, calculando média do valor dos k-vizinhos, isto é: $\hat{f}(x_?) \leftarrow \frac{1}{k}\sum_{i=1}^{k} f(x_i)$.

Uma versão mais refinada deste método consiste em atribuir pesos diferentes aos k-vizinhos [Mitchell 1997], dependendo por exemplo da distância – multiplicar cada parcela do somatório for um factor que depende do inverso do quadrado da distância, isto é: $w_i = d(x_?, x_i)^{-2}$

Como pontos positivos deste método destaca-se o facto do processo de aprendizagem ser muito simples e incremental, consiste unicamente em ir memorizando instâncias, e consequentemente o tempo de aprendizagem é baixo. É também um método aplicável, mesmo em problemas muito complexos e robusto no que diz respeito a instâncias com ruído. Relativamente aos pontos negativos, tem-se o preço pago pela simplicidade do treino e que obriga ao armazenamento de muita informação e em consequência a classificação de uma nova instância pode tornar-se lenta, pela necessidade de calcular a distância a todas as instâncias armazenadas. Outra dificuldade que este método encontra é o chamado *problema da dimensionalidade*, que reside no facto do número de instâncias (pontos) representativas necessárias, aumentar exponencialmente com o número de atributos das instâncias. Um outro problema, também relacionado com o número de atributos, consiste na possibilidade de existirem *atributos irrelevantes*, fazendo com que duas instâncias que até estariam próximas, encontrarem-se afastadas como consequência de valores muito diferentes nos atributos não relevantes.

3.4 Aprendizagem Bayesiana

"Probability is that degree of confidence dictated by the evidence through Bayes's theorem".

E.T. Jaynes

A abordagem bayesiana ao tema da aprendizagem, baseia-se essencialmente em toda a teoria estatística que envolve e deriva do "*Teorema de Bayes*". Toma-se como premissa que os itens e relações de interesse são a manifestação de leis de distribuição de probabilidade que estão por detrás. É portanto uma abordagem essencialmente quantitativa, à aprendizagem, olhando para o problema como a escolha da melhor hipótese de um espaço de hipóteses – aquela que é mais coerente com os dados do problema – aqui a função objectivo (conceito alvo a aprender) corresponde à escolha da melhor hipótese, escolha essa que é realizada tendo em conta o cálculo explícito da probabilidade de cada hipótese.

Esta abordagem é uma peça importante na área aprendizagem automática. Por um lado os resultados práticos conseguidos, em muitos domínios, mostram-se competitivos com os alcançados por outros métodos de aprendizagem, superando-os mesmo em muitas situações [Mitchell 1997], como é o caso das redes neuronais e das árvores de decisão. Por outro lado, este método ajuda a perceber outros métodos de aprendizagem que não manipulam explicitamente probabilidades.

3.4.1 Teorema de Bayes

O teorema de Bayes, sendo um dos resultados mais importantes da teoria das probabilidades é também o princípio fundamental da aprendizagem bayesiana.

Teorema – Se $\{A_1, A_2, \ldots, A_m\}$ é uma partição do espaço de resultados e B um qualquer acontecimento, com $P(B) > 0$ e para cada i $P(A_i) > 0$, então:

$$P(A_i \mid B) = \frac{P(A_i) P(B \mid A_i)}{\sum_{i=1}^{m} P(A_i) P(B \mid A_i)} \qquad i \in \{1, \ldots, m\}$$

designando *P(A)* a probabilidade do acontecimento *A* e $P(A|B)$ a probabilidade de A condicionada por B, definida por $\frac{P(A \cap B)}{P(B)}$.

Um consequência imediata, deste teorema é que para dois acontecimentos *A* e *B*, tais $P(A) > 0$ e $P(B) > 0$, se tem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Supondo que o nosso conjunto de dados (instâncias de treino) é designado por *D*, então pelo teorema anterior, temos uma forma de calcular a probabilidade duma hipótese *h*, tendo por base esses dados:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

em que a probabilidade $P(h|D)$ é a denominada "*probabilidade à posteriori*" e $P(h)$ a "*probabilidade à priori*" da hipótese *h*. Considerando que temos um espaço de hipóteses possíveis *H*, então pretende-se determinar qual a melhor hipótese , tendo em conta os dados observados: *D*. Se interpretarmos a melhor hipótese, como a mais provável, atendendo aos dados, isto é a hipótese com melhor valor de "*probabilidade à posteriori*", designada nalguma literatura por h_{MAP} [Mitchell 1997], então o que se quer é:

$$h_{MAP} = \underset{h \in H}{argmax} P(h|D)$$

e pela relação anterior: $h_{MAP} = \underset{h \in H}{argmax} \frac{P(D|h)P(h)}{P(D)}$. Como *P(D)* é constante em relação a h, a formula resume-se a: $h_{MAP} = \underset{h \in H}{argmax} P(D|h)P(h)$.

Nalguns contextos assume-se que todas as hipótese são igualmente prováveis e nesse caso esta formula ainda é mais simplificada, traduzindo-se naquilo que se designa por "*hipótese mais verosímil*" ("M*aximum likelihood*"): $h_{ML} = \underset{h \in H}{argmax} P(D|h)$. Muitos dos algoritmos de

aprendizagem têm implícito o conceito de procura de h_{ML} ou h_{MAP}, por exemplo a procura da minimização do quadrado dos erros, numa rede neuronal, não é mais que a procura de h_{ML}. Um outro exemplo é o que se passa na indução de árvores de decisão, com a tendência de escolher árvores pequenas. Se interpretarmos a árvore induzida como sendo a hipótese preferida, à luz da aprendizagem bayesiana concluímos que essa hipótese não é mais que h_{MAP}, pois uma forma equivalente de h_{MAP} é: $h_{MAP} = \underset{h \in H}{argmin} \; -\log_2 P(D|h) - \log_2 P(h)$, que expressa precisamente, pela teoria de informação, a hipótese mais pequena, assumindo uma certa codificação para as hipóteses.

Nos problemas de classificação, podemos aplicar o teorema de Bayes como classificador e o objectivo é encontrar qual a classe mais provável, tendo em conta as instâncias de treino observadas – é o caso do trabalho implementado e descrito na secção 4.2.2. Sendo *V* um conjunto de classificadores, queremos $v_{Bayes} = \underset{v_j \in V}{argmax} P(v_j|D)$ que pode ser reescrito da forma: $v_{Bayes} = \underset{v_j \in V}{argmax} \sum_{h \in H} P(v_j h|D)$ <=> $v_{Bayes} = \underset{v_j \in V}{argmax} \sum_{h \in H} P(v_j|hD)P(h|D)$ e sendo v_j independente de *D*, sendo *h* conhecido, tem-se o denominado classificador óptimo de Bayes: $v_{Bayes} = \underset{v_j \in V}{argmax} \sum_{h \in H} P(v_j|h)P(h|D)$.

Diz-se que este classificador é óptimo, significando que em média obtém melhores resultados que qualquer outro, a partir da mesma informação. Assim o erro cometido por este classificador é o patamar mínimo teórico, relativamente a qualquer outro classificador, sendo conhecido pelo *Erro do Bayes Óptimo*.

Percebe-se assim a importância que a abordagem bayesiana tem na aprendizagem indutiva, todavia a sua aplicação à prática é algo complicada, na medida em que é necessário conhecer as probabilidades exactas. Na prática só vamos ter as estimativas dessas probabilidades, que podem não estar correctas. O método "*Naive Bayes*" referido a seguir é uma abordagem simplificada da abordagem bayesiana, cuja implementação prática se encontra mais simplificada.

3.4.2 Naive Bayes

O método de *Naive Bayes* assume como premissa fundamental que os atributos das instâncias de dados (*D*) são independentes entre si. Supondo que uma instância $I \in D$ consiste num vector de valores relativos aos m atributos $\langle a_{1,} a_{2,} \ldots, a_m \rangle$ [1], pretende-se, de acordo com o apresentado na secção anterior, obter o v_{MAP}, dentre os classificadores possíveis de *V*, isto é:

$$v_{MAP} = \underset{v \in V}{argmax} P(v \mid a_{1,} a_{2,} \ldots, a_m).$$

mas, pelo teorema de Bayes, a igualdade apresentada é equivalente a:

$$v_{MAP} = \underset{v \in V}{argmax} \frac{P(a_{1,} a_{2,} \ldots, a_m \mid v) P(v)}{P(a_{1,} a_{2,} \ldots, a_m)}.$$

Como o termo que está denominador não interfere no resultado final, por ser independente do classificador em causa (*v*), pode ser suprimido ficando-se só com:

$$v_{MAP} = \underset{v \in V}{argmax} P(a_{1,} a_{2,} \ldots, a_m \mid v) P(v).$$

Admitindo agora a independência de $a_{1,} \ldots a_m$ chegamos à formula do classificador naive Bayes:

$$v_{MAP} = v_{NB} = \underset{v \in V}{argmax} P(v) \prod_{i=1}^{m} P(a_i \mid v)$$

Uma primeira observação é o facto desta fórmula ser mais facilmente aplicável na prática, dado que só é necessário calcular $k \cdot m$ probabilidades ($k = |V|$), enquanto que para o classificador v_{MAP} mais geral esse valor é da ordem m^k, simplificando muito em termos de complexidade computacional.

1 Ver secção anterior.

O classificador v_{NB} é um classificador amplamente utilizado, mostrando desempenhos que chegam a atingir e mesmo ultrapassar o demonstrado por outros classificadores como as redes neuronais e as árvores de decisão [Michie et al. 1994].

O método de *Naive Bayes* é utilizado no nosso trabalho, bem como uma sua versão melhorada, para induzir um classificador de documentos de texto, no nosso domínio (consultar subsecção 4.2.2 e 4.2.3).

3.5 Indução de Árvores de Decisão

As árvores de decisão são uma das metodologias de aprendizagem indutiva mais utilizadas na actualidade, quer a nível de aplicação, quer a nível de trabalho e investigação académica. Têm sido concretizadas aplicações desta metodologia da IA em domínios que vão desde a análise financeira e económica, até problemas de diagnóstico médico. Vários algoritmos de indução de árvores de decisão têm sido desenvolvidos e melhorados nos últimos anos, como é o caso do CART [Breiman et al. 1984], ASSISTANT [Cestnik et al. 1987], ID3 [QUINLAN, 1986], C4.5 [QUINLAN, 1993] e C5, este dois últimos são versões melhoradas do ID3.

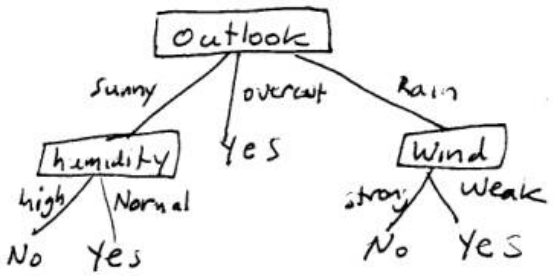


Figura 3.2 - Esboço de uma *Arvore de Decisão.*

Uma árvore de decisão consiste numa árvore em que cada nó define uma condição lógica sobre um atributo duma instância. Denominando um conjunto de instâncias por *S* e o conjunto de atributos considerado por $A = \{a_1, \ldots, a_m\}$, então para $x \in S$ tem-se $x \equiv \langle a_1(x), \ldots, a_2(x) \rangle$, sendo $a_i(x)$ o valor assumido pelo atributo a_i na instância *x*. Assim o nó duma árvore contém uma condição sobre algum elemento de *A*, por exemplo $a_k > 3.7$ ou $a_k = alto$. Cada ramo derivado dum nó consiste num possível valor do atributo considerado no nó. Cada folha da árvore representa um elemento duma classe.

Cada caminho, desde a raíz, a uma folha, corresponde a uma regra de decisão ou classificação. Portanto uma árvore de decisão é traduzível numa disjunção de conjunções lógicas de condições, condições essas sobre os valores de *A*, sendo cada ramo da árvore uma conjunção de condições e o conjunto dos ramos disjuntos (DNF – Disjuntive Normal Form). Isto significa que qualquer função da lógica do cálculo proposicional, é representável através duma árvore de decisão.

As árvores de decisão são propícias para problemas de classificação, cujo número de classes é finito. Uma das vantagens à partida, das árvores de decisão é o facto de ser gerado conhecimento simbólico ao contrário de muitas outras metodologias de aprendizagem, como por exemplo as redes neuronais. Aqui são geradas regras da lógica proposicional e portanto susceptíveis de compreensão, por parte dos seres humanos.

3.5.1 Ganho de Informação

O ganho de informação é a medida estatística que está na base da construção duma árvore de decisão, no algoritmo ID3, referido a seguir. Todavia esta medida, da teoria da informação de Shannon [Shannon & Weaver 1949], não é de utilização exclusiva do ID3, tendo aplicações variadas em outras áreas (ver secções 4.2.3 e 4.4.2).

Se tivermos um conjunto de várias instâncias *S*, e um conjunto de *n* classes $C = \{C_1, \ldots, C_n\}$, sendo p_i a probabilidade da classe C_i em *S*, então a entropia do conjunto *S*, é uma medida de homogeneidade deste, traduzida na seguinte igualdade:

$$Entropia(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

A entropia é uma medida aplicável à partição dum espaço de probabilidade, medindo o quanto esse espaço é homogéneo ou por outro lado o quanto ele é desordenado ou caótico. A entropia atinge o seu valor máximo, igual a $\log_2 n$, quando $p_1 = p_2 = \ldots = p_n = \frac{1}{n}$, expressando precisamente a existência dum máximo de homogeneidade.

Considerando um atributo *A* das instâncias de *S*, com *Dominio(A)* = $\{v_1, v_2, \ldots, v_r\} = V$ então a consideração de $A = v$, com $v \in V$ separa um subconjunto de elementos de *S*. Denomine-se esse subconjunto de elementos por S_v, então podemos voltar a calcular a entropia deste novo conjunto: $Entropia(S_v)$. Realizando esta operação para cada elemento de *V*, podemos determinar o quanto é esperado que seja reduzida a entropia, considerando que os valores de

A são conhecidos. De outro modo pretende-se saber qual o ganho de informação do atributo *A*, que é dado pela seguinte formula:

$$Ganho(S, A) = Entropia(S) - \sum_{v \in V} \frac{|S_v|}{|S|} Entropia(S_v)$$

em que $|S|$ e $|S_v|$ designam a cardinalidade dos conjuntos *S* e S_v, respectivamente.

3.5.2 Método Básico (Algoritmo ID3)

O algoritmo ID3 consiste num processo de indução de árvores de decisão. A construção da árvore é realizada de cima para baixo ("top-down"), com o objectivo de sempre escolher o melhor atributo para um nó da árvore. É um processo recursivo que após ter escolhido um atributo para um nó, começando pela raíz, aplica o mesmo algoritmo aos descendentes desse nó, até que certos critérios de paragem sejam verificados. Uma versão simplificada deste algoritmo pode ser descrito como se segue:

Dado um conjunto de exemplos S
Criar o nó raíz
Se todos os exemplos de S são da mesma classe C_j **Então**
  O nó actual é uma folha da classe C_j.
Senão
  Seleccionar o melhor atributo A, cujo domínio é: $V = \{v_1, v_2, \ldots, v_r\}$
  Dividir o conjunto de instâncias em $S_1, S_2, \ldots, S_m$, relativ. aos valores de *V*
  Criar recursivamente as sub-árvores $T_1, T_2, \ldots, T_m$, para $S_1, S_2, \ldots, S_m$.
  Gerar a árvore de decisão T, tendo por base o nó actual e $T_1, T_2, \ldots, T_m$.

Algoritmo 3.1 – *ID3 simplificado*

Para cada nó da árvore é escolhido o melhor atributo, entre os possíveis, essa escolha é concretizada tendo em conta a medida do *ganho de informação* referida anteriormente. Escolhe-se o atributo mais informativo para o nó actual. Esta escolha do atributo mais

informativo, faz com que em cada partição, das instâncias, gerada (S_v) os exemplos tendam a ser cada vez mais de uma só classe C_j (mais homogéneos).

A escolha do atributo mais informativo – que mais reduz a entropia – faz com que a tendência seja a de gerar árvores , que são, em geral, menos profundas. Em suma o algoritmo ID3 realiza uma procura no espaço das árvores de decisão, consistentes com os dados. Isto não significa que seja encontrada a melhor árvore, tenha-se presente que o problema da procura global da melhor árvore é um problema *NP* completo. O algoritmo ID3 realiza uma procura ávida ("*greedy*"), guiada pela heurística do ganho de informação e feita segundo a estratégia do "*subir a colina*" ("*hill-climbing*"). Como é sabido desta estratégia, corre-se o risco da solução convergir para um mínimo local.

Para os atributos cujos domínios sejam valores numéricos, reordenam-se as instâncias, de acordo com esse atributo e procuram-se pontos extremos (instâncias) nos quais existe uma mudança de valor da classe. Um ponto de mudança de classe marca uma partição binária do conjunto das instâncias, mediante uma condição lógica do tipo $A > x$, sendo *A* o atributo numérico em causa e *x* um valor calculado a partir dos dois valores consecutivos de *A* nesses pontos. Normalmente toma-se x igual à média dos valores de *A*, nos pontos consecutivos. Foi mostrado que, neste tipo de atributos, de todos os possíveis pontos de referência, aqueles que maximizam o ganho de informação separam dois exemplos com valores de classe diferentes [Fayyard & Irani 1993].

Um dos problemas a ter em conta na indução de árvores de decisão é o chamado problema de "*Overfitting*", isto é sobre-ajustamento da árvore aos dados de treino, obtendo um desempenho quase perfeito nesses, mas um desempenho pobre nos novos dados. Isto tem origem na possibilidade de existir ruído nos dados, devido a valores errados dos atributos ou das classes, ou devido à existência de um conjunto de atributos inadequado ou insuficiente. As extensões e desenvolvimentos ao algoritmo ID3, tem em consideração estes problemas, bem como outros de natureza prática e que serão referidos a seguir.

3.5.3 Método Melhorado (C4.5 / C5)

Uma das dificuldades do algoritmo ID3 consiste na possibilidade de gerar árvores demasiado ajustadas aos dados de treino – "*overfitting*" – com um desempenho quase perfeito, nestes dados, mas com um baixo desempenho nos novos dados, tal como foi referido no final da subsecção anterior.

Ultrapassando o Sobre-Ajustamento (*Overfitting*)

O algoritmo C4.5 [Quinlan, 1993] é um método melhorado do ID3 que, entre outros melhoramentos, combate o problema do *overfitting*, utilizando uma estratégia de "*poda da árvore*". Existem duas estratégias de combate ao problema do sobre-ajustamento, que pressupõe que a árvore tem uma complexidade inadequada e irreal. Assim o princípio orientador é o denominado princípio de *Occam* ou "*Ocam's razor*" que dá primazia à escolha de hipótese menos complexas, compatíveis com a realidade observada. As duas estratégias de simplificação de árvores de decisão, existentes são:

- Interromper prematuramente o crescimento da árvore (pré-poda)
- Deixar a árvore crescer livremente de depois podá-la (pós-poda)

O algoritmo C4.5 adopta a segunda estratégia (pós-poda). Podar uma árvore, neste contexto, significa reduzir algumas subárvores a folhas, ou de outra forma, um ramo da árvore, a partir de determinado nó é "cortado" (transformado em folha). O corte dum ramo da árvore é guiado por um teste estatístico que tem conta os erros num nó e a soma dos erros nos nós que descendem desse nó. Assim, para cada nó, a poda só se concretiza se o desempenho da árvore não diminuir. O sistema C4.5 faz uma estimativa pessimista do erro, para cada nó.

Outros Melhoramentos

Para além do problema do overfitting, o C4.5 ultrapassa problemas concretos e comuns do mundo real como: atributos com valores numéricos; valores omissos e dados contendo ruído. Uma outra possibilidade disponibilizada por este sistema é a capacidade de realizar validação

cruzada ("*N-fold cross validation*"), melhorando assim a estimativa do erro cometido pelo classificador.

Uma ultima característica que merece ser destacada é a possibilidade deste sistema em gerar regras de decisão a partir de árvores. Dado que uma árvore de decisão se encontra na forma normal disjuntiva (DNF), como referido, é relativamente fácil traduzir este classificador para um conjunto de regras de decisão, todavia o sistema C4.5 utiliza um método mais específico para gerar uma lista ordenada de regras, donde se destaca a realização da poda adicional nas regras induzidas inicialmente. Para um mesmo problema o número de regras induzidas é em geral inferior ao número de folhas da correspondente árvore já podada [Witten & Frank 2000]. Esta é uma característica importante, na medida em que um classificador na forma dum conjunto de regras é geralmente mais fácil de perceber que a correspondente forma da árvore de decisão.

O sistema C5 [Quinlan, 1997], é o sucessor do C4.5, melhorado, para lidar com as exigências do mundo real. O grande salto foi dado em termos de eficiência, quer a nível de tempo de processamento e de memória utilizada [Quinlan, 1997]. Por outro lado, os classificadores gerados são normalmente mais pequenos e precisos.

Para além do salto em eficiência o sistema C5 oferece mais alguns melhoramentos como: novos tipos de dados incorporados (ex: pode trabalhar com o tipo "não a aplicável" - N/A); atributos definidos a partir de combinações funcionais doutros atributos; utilização de custos diferenciados para os erros de classificação.

Uma outra característica que permite diminuir a taxa de erro dos classificadores, no C5, é a utilização de *Boosting* [Schapire 2002]. Esta técnica consiste em gerar vários classificadores, a partir dos mesmos dados de treino, e depois combiná-los num classificador final no qual cada classificador inicial participa votando com um certo peso. Este peso é ajustado durante o processo de treino [Quinlan 1996]. Nalguns casos a redução dos erros de classificação pode atingir 40% [Quinlan 1997].

O sistema C5 foi utilizado para induzir regras de extracção, no trabalho implementado no âmbito desta dissertação.

4 O Projecto de Extracção de Informação da Web

Neste capítulo é apresentado o trabalho realizado, no âmbito desta dissertação. Numa primeira parte será descrito o domínio de aplicação, bem como outras particularidades afins. Nas secções ou partes posteriores, serão apresentados os principais mecanismos aplicacionais, desenvolvidos e empregues.

4.1 O Domínio de Aplicação

Num tempo em que existe tanta informação disponível, nas mais variadas áreas, as possibilidades de escolha da aplicação de um trabalho deste género são elevadas. Como já foi referido anteriormente, um grande numero de problemas e necessidades têm sido colmatados com soluções das áreas do Processamento de Linguagem Natural (PLN) e Inteligência Artificial (IA), basta observar aquilo que tem sido apresentado nas conferências TREC e MUC[1]. É interessante constatar que as primeiras aplicações do trabalho realizado por estas duas áreas do conhecimento humano, foram a bolsa e toda a informação que órbita em torno desta, e informação relacionada com administração e gestão de empresas [Constantino 1997]. Já desde os anos 70 que uma das primeiras aplicações de ferramentas desenvolvidas pela IA são os mercados financeiros. Em muitos casos se não é o mercado de capitais que está em causa, é outro mercado de outro negócio qualquer e que tem um factor comum que é o de envolver valores financeiros - dinheiro. A titulo exemplificativo: actualmente os operadores dos mercados bolsistas têm ao seu dispor um elevado volume de informação, sendo esta vital para a tomada de decisões. Na maioria dos casos essas decisões têm de ser tomadas em tempo real, dispondo os operadores de muito pouco tempo para digerir toda a informação acedida. Uma grande parte desta informação é apresentada sob uma forma textual ou documental, como é o caso de pequenos anúncios ou notícias, o que complica e torna ainda mais morosa a tarefa dos operadores financeiros, dada a natureza não estruturada, dessa informação. Tendo isto em mente, todos os processos ou ferramentas capazes de sintetizar texto ou extrair elementos considerados relevantes deste, revestem-se de uma importância preponderante e de grande utilidade prática.

1 Conferências na área de “Information Retrieval” e “Information Extracion” (ver subsecção 2.1.1)

De entre os muitos domínios possíveis, resolveu-se optar por um que não diz respeito aos mercados financeiros mas é também um mercado – Mercado Imobiliário Português, mais especificamente: compra e venda de habitações. São bem conhecidos de todos nós os anúncios, publicados nos jornais diários ou semanários, relativos à venda e compra de alguma coisa ou oferta e procura de emprego. Com o advento da Internet e sua vulgarização em massa nasceu a possibilidade deste tipo de anúncios começar a ser realizado em espaços públicos "on-line". É isto que sucede actualmente em muitos dos *Grupos de Notícias* ("News Groups"), da Usenet, nacionais e internacionais. Relativamente a mercados nacionais apresentam-se em baixo alguns exemplos:

```
pt.mercado.emprego
pt.mercado.veiculos
pt.mercado.informatica
pt.mercado.imobiliario
```

Nestes espaços, qualquer utilizador da Internet, poderá consultar e publicar, de uma forma livre e fácil, um anúncio, utilizando um software cliente de correio electrónico. O numero de utilizadores destes serviços, tem vindo a aumentar, até porque são serviços gratuitos, se não contarmos com os custos de ligação à Internet. Portanto o numero de anúncios têm vindo a aumentar consideravelmente, nos últimos anos, sendo já desejáveis ferramentas de tratamento automático, quer para classificar e categorizar anúncios, quer para realizar extracções importantes de elementos dos mesmos. O trabalho desenvolvido foi dirigido neste sentido e orientado para os anúncios do tipo: `pt.mercado.imobiliario`. Apesar de ter sido escolhido este tipo de anúncios como alvo, tentou-se manter o trabalho o mais genérico possível, de modo a que uma migração deste para outro domínio possa ser realizada com facilidade. Como será verificado nas secções seguintes, o trabalho não está preso a particularidades do domínio de aplicação em causa, o que o torna facilmente transportável e adaptável para outros "universos".

De entre as várias maneiras que existem para aceder a documentos da USENET, uma delas consiste em utilizar o WWW, através, por exemplo, do motor de pesquisa *Google* (`http://www.google.com`). Na figura 4.1 é apresentado um anúncio do nosso domínio, obtido desta forma. Como se pode ver, temos uma página em HTML, onde o essencial é um

pequeno texto, em Português, informando acerca da disposição de alguém em vender um apartamento do tipo T2, em Rio de Mouro, pelo valor de 128 000 euros, para além de outras características descritas.

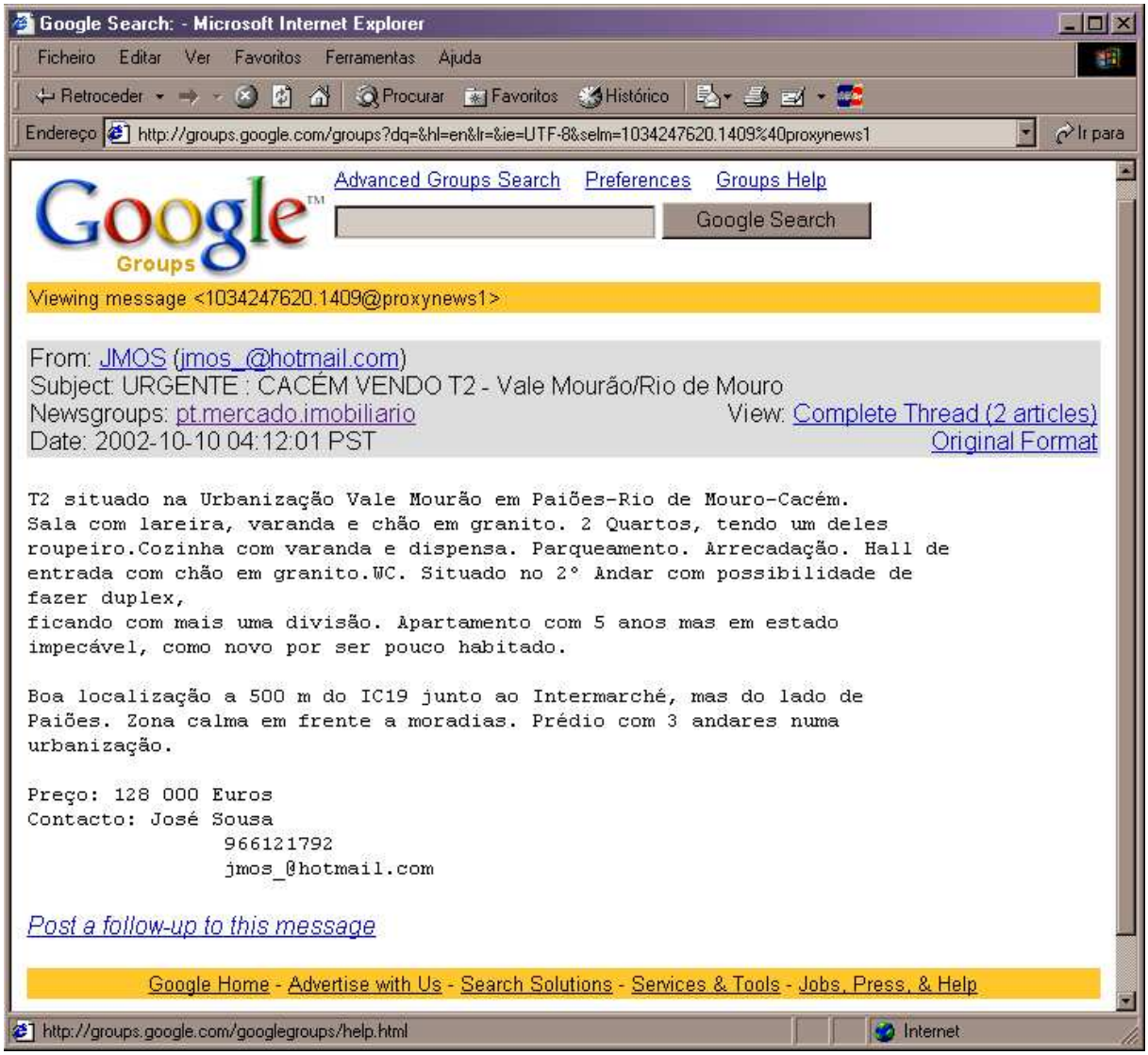


Figura 4.1 - Exemplo de um anúncio relativo à venda dum apartamento.

Teoricamente, os anúncios publicados neste grupo de notícias seriam exclusivamente de assuntos relacionados com habitações, todavia não são poucos os anúncios publicados, por engano ou outra razão qualquer, cujo tema é completamente alheio ao do mercado imobiliário, podendo surgir anúncios dos temas mais diversos que se possa imaginar. Mesmo em anúncios relacionados com o mercado imobiliário, existem assuntos que não dizem respeito à compra e venda de habitações, todavia faz sentido que estejam publicados ali.

Podem, por exemplo, surgir anúncios publicitários, aluguer e reparação de bens imobiliários, os anúncios poderão ser relativos a escritórios, armazéns, terrenos, etc. Existe, portanto, a necessidade de algo que classifique um anuncio, tendo em conta o seu conteúdo.

Torna-se importante saber quais os anúncios que dizem respeito à compra e venda de habitações e não a outro assunto qualquer, mesmo que relacionado com estes e muito próximo, como é o caso do aluguer de habitações. Preocupações deste género enquadram-se no âmbito do "Information Retrieval" e uma parte do trabalho desenvolvido e exposto nesta dissertação entra nesta área. Todavia o principal esforço deste trabalho e alvo preferencial, centra-se na área da extracção de elementos textuais, a partir de documentos - "Information Extraction". Nesta área a principal preocupação é saber extrair elementos textuais, que poderão ser palavras ou pequenas expressões, consideradas importantes, a partir desses documentos. Os elementos relevantes extraídos serão posteriormente armazenados sob uma forma estruturada, permitindo assim um rápido e eficiente acesso à informação.

Constata-se assim a existência de duas componentes bem delimitadas no trabalho realizado, complementares e dotadas de uma certa sequência cronológica e também funcional. Numa primeira fase põe-se o problema da escolha de documentos relevantes de entre uma vasta colecção disponível e numa segunda fase o problema centra-se em extrair os elementos relevantes do texto dos documentos previamente escolhidos. Posto isto torna-se evidente e natural apresentar estas duas fases em separado. Assim as duas secções que se seguem irão concretizar a exposição segundo esta ordem.

Numa primeira fase o sistema classifica uma colecção de anúncios do domínio em duas classes: "venda", "não venda". Isto é se o anuncio diz respeito à venda de uma habitação ou não. Este primeiro trabalho é importante para que o sistema possa seleccionar de uma forma automática os anúncios que lhe interessam, a partir da vasta colecção disponibilizada, para depois realizar a extracção dos elementos relevantes nestes anúncios ("Venda de habitações"). A título ilustrativo a figura 4.2 lista uma colecção de anúncios do grupo "`pt.mercado.imobiliario`" da Usenet.

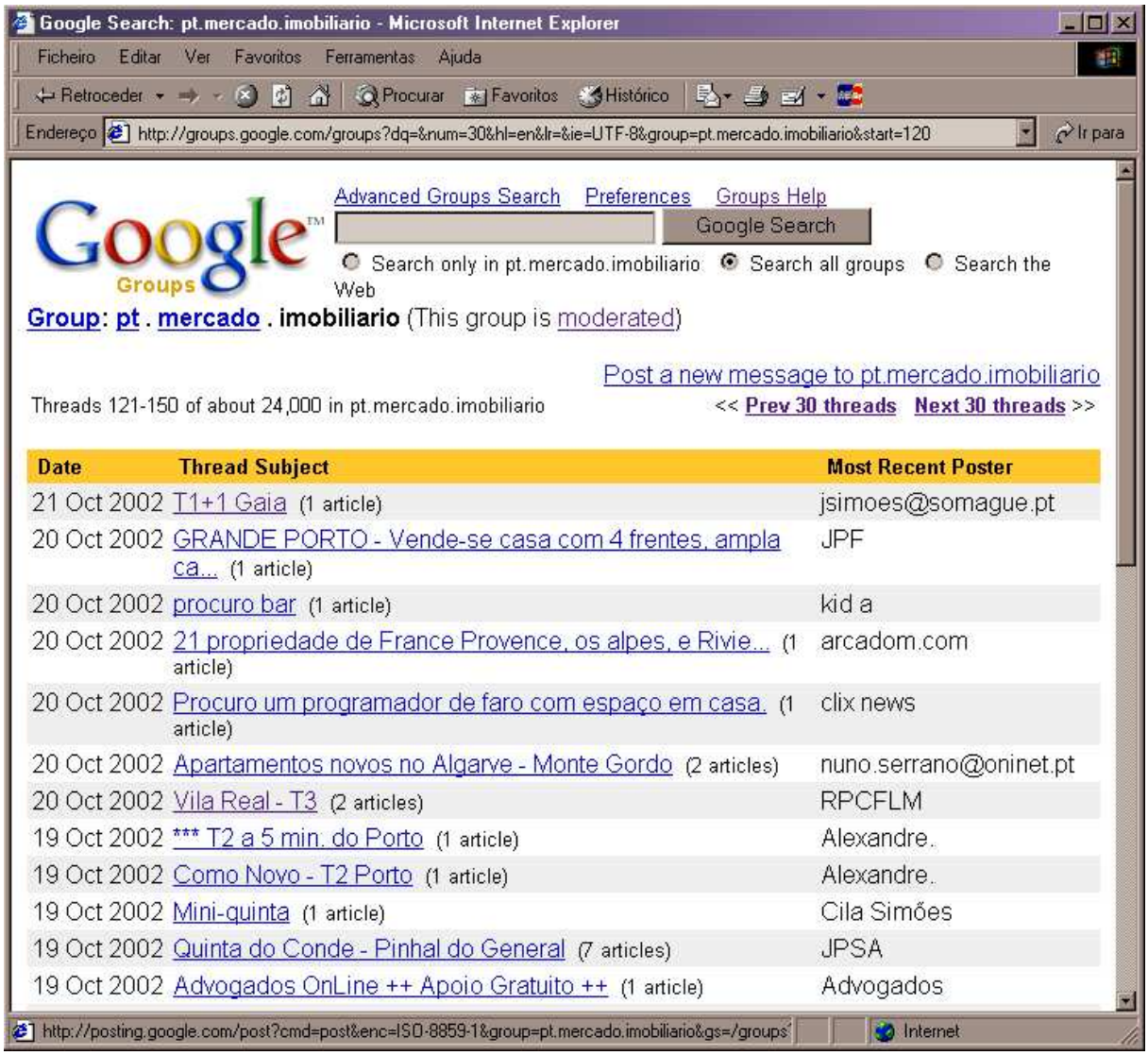


Figura 4.2 - Lista de anúncios de "`pt.mercado.imobiliario`"

Como se pode verificar na figura anterior, nem todos os anúncios dizem respeito à venda de habitações, é necessário realizar uma triagem sobre a colecção disponível. Após esta triagem temos uma colecção de anúncios pronta a ser processada.

4.2 Classificação de Anúncios

A primeira parte do trabalho realizado enquadra-se na problemática da obtenção de informação ("Information Retrieval"). Este processo de obtenção de informação traduz-se num trabalho de classificação. Mediante um elemento de informação é necessário proceder à sua correcta classificação – atribuir-lhe o "rótulo certo". 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", etc. O problema aqui abordado, consiste num problema de classificação binária, uma vez que o que se pretende é classificar um anúncio de forma a que ele seja seleccionável ou não. Aqui um anúncio é seleccionável se for relativo à venda de uma habitação, podendo esta ser um apartamento ou uma moradia. Consideramos então dois rótulos possíveis para as nossas classes e que consistem em "**venda**" e "**não venda**", significando que o anúncio diz respeito à venda duma habitação ou não, respectivamente.

Assim, o nosso problema inicial consiste na existência de documentos que nos interessam ("venda") entre outros que não nos interessam. O objectivo é implementar um sistema capaz escolher os documentos interessantes entre os existentes – "separar o trigo do joio". Este é um típico problema de "*Information Retrieval*" e já existe trabalho realizado, nesta área, como é referido e descrito no capitulo 2. Desde o início da década de 1990 que a discussão tem crescido e evoluído, em torno de problemas desta natureza. Existe actualmente, já um considerável trabalho realizado e técnicas desenvolvidas, nesta área. Pretendemos aqui explorar algumas dessas técnicas, experimentar alguns procedimentos recentes que melhoram o desempenho dos processos de classificação de documentos. Portanto, o que é apresentado nesta secção, consiste num trabalho de pré-processamento relativamente ao nosso objectivo principal, permitindo a implementação e exploração de algumas técnicas de "Information Retrieval", ao mesmo tempo que é realizado um trabalho útil para a próxima fase do nosso trabalho – *Extracção de Informação* – a fase mais importante deste.

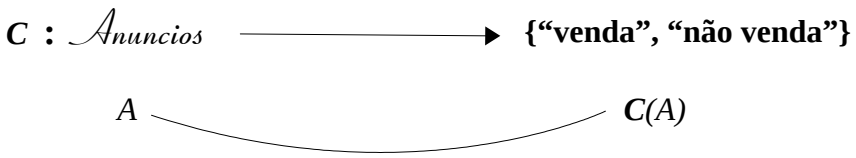


Figura 4.3 - Função de classificação de documentos.

A figura anterior esquematiza o que se pretende implementar, nesta fase: um sistema que seleccione anúncios da colecção dos apresentados no grupo de notícias `pt.mercado.imobiliario`, e que na nossa situação especifica consistem em anúncios relativos à venda de habitações. Tanto nesta primeira fase como na fase seguinte são utilizadas técnicas de "Aprendizagem Automática"[1] (Machine Learning), mais concretamente, *Aprendizagem Supervisionada*. Tendo isto presente, queremos que o sistema tenha a capacidade de ser treinado para realizar determinada tarefa, que nesta primeira fase consiste em distinguir os anúncios de "venda" e "não venda", de habitações. Cada anúncio constitui um exemplo susceptível de ser classificado como positivo ("venda") ou negativo ("não venda").

Na fase de treino é utilizada uma colecção de exemplos previamente classificados por um agente exterior ao sistema, normalmente um ser humano. Esta colecção é constituída por exemplos de ambas as classes: positivos e negativos, numa proporção equilibrada ou balanceada, isto é aproximadamente tantos positivos quanto negativos. A colecção de exemplos previamente classificados é depois submetida ao sistema que irá "aprender" a classificar este tipo de exemplos. De outra forma, um exemplo ou instância de treino consiste num par *(A, C(A))*, em que *A* é um anúncio e *C(A)* a correcta classificação desse anuncio. *O* nosso objectivo é induzir a função *C*.

Aqui são exploradas várias técnicas, nomeadamente: *Aprendizagem Baseada em Instâncias*[2] e *Naive Bayes*[3]. É ainda experimentada uma técnica de optimização do desempenho e denominada de "Features Subset Selection", na literatura [Mladenic 1998]. Os resultados de

1 Consultar secção 3.2
2 Secção 3.3
3 Secção 3.4

cada uma das técnicas exploradas são apresentados no capítulo 5, com as devidas comparações e comentários.

Em suma, iremos realizar um trabalho que entra nas nas áreas do "Information Retrieval", nesta secção, e na área do "Information Extraction", na próxima. Adiantamos aqui que algumas das ideias e técnicas empregues nesta secção, serão convenientemente adaptadas e utilizadas na próxima secção (4.3), como é o caso de "Features Subset Selection".

4.2.1 Aprendizagem Baseada em Instâncias (k-vizinhos)

Uma primeira abordagem à solução do problema referido anteriormente, seria a criação de um sistema dedicado à escolha dos anúncios pretendidos, com base num conjunto de critérios fixa e previamente estabelecidos. Esses critérios seriam definidos por alguém que tivesse um razoável conhecimento do domínio a que os anúncios dizem respeito. Por exemplo, para o nosso domínio, os critérios poderiam ter em consideração certas palavras ou pequenas expressões textuais, como as seguintes: "sala", "cozinha", "garagem", "preço", "vende-se apartamento", "óptimas vistas", etc. Esse sistema poderia então classificar um anúncio, com base na ocorrência destas expressões, por exemplo as frequências das ocorrências ou outra medida mais elaborada, digamos um certa função real *F*. Se $F(v|_{anuncio})$ fosse suficientemente próximo de um certo limiar pré-estabelecido então o anuncio seria classificado de "venda", senão seria "não venda". Aqui *v* é um vector de inteiros, contendo, as frequências de cada palavra (ou expressão) no *anuncio*. Repare-se que neste processo não seria utilizada qualquer forma de indução, qualquer espécie de aprendizagem. Seria um sistema estático e com uma definição rígida, de tal forma que um pequeno ajuste no domínio poderia levar à necessidade de o reprogramar.

Na aprendizagem baseada em instâncias, um dos algoritmos amplamente utilizados é o chamado algoritmo dos "K-Vizinhos Mais Próximos" (k-Nearest Neighbor) ou só "k-Vizinhos". A estratégia seguida por este algoritmo tem algumas semelhanças com a abordagem descrita no parágrafo anterior e este foi um algoritmo experimentado com a finalidade de satisfazer o objectivo aqui apresentado. Em traços muito gerais, o algoritmo dos "k-Vizinhos" considera cada instância como um vector **v** de **n** números reais, contendo os

valores dos **n** atributos, para essa instância. É também considerada a classe dessa instância, dada por uma função **f** de $\mathbf{R}^{\mathbf{n}}$ no conjunto de todas as classes, **f(v)** ∈ **{**"venda", "não venda"**}**, no nosso caso. Portanto uma instância é interpretada como um ponto no espaço euclidiano de dimensão **n**. Assim é considerada também a distância euclidiana entre dois pontos, como uma medida de distância entre instâncias. Sejam **v** e **w** dois vectores de duas instâncias, tem-se:

$$d(v,w) = \sqrt{\sum_{i=1}^{n} (v_i - w_i)^2}$$

A classificação de uma nova instância é feita tendo em conta as k instâncias mais próximas a essa instância, medindo-se essa proximidade com a distância euclidiana referida. Para classes discretas ou simbólicas, como é o nosso caso, considera-se normalmente a classe mais frequente entre os k-vizinhos.

A experiência realizada neste contexto, considerou vectores de frequências de certas palavras, nos documentos (anúncios) e a respectiva classe, como instâncias de treino. Essas palavras constituem os atributos que são determinados e fixados antes da fase de treino. Assim podemos considerar, como exemplo, a seguinte sequência de 6 atributos, para a formação de instâncias de treino no nosso domínio:

<"sala", "cozinha", "garagem", "preço", "vende-se apartamento", "óptimas vistas">.

Para estes atributos, mostram-se a seguir, exemplos de instâncias de treino:

```
(0,0,0,0,0,0)  "não venda"
(0,0,0,0,0,0)  "não venda"
(0,0,0,1,0,0)  "não venda"
(1,1,1,1,0,0)  "venda"
(0,1,0,3,0,0)  "venda"
(0,0,0,1,0,0)  "não venda"
(0,0,0,0,0,1)  "não venda"
(0,0,0,0,2,0)  "venda"
```

Ainda para este exemplo simplista (poucos atributos considerados e instâncias) e atendendo aos k-vizinhos, com k=3, uma nova instância da forma (1,1,1,1,0,0) seria classificada de "venda" e a instância (1,0,0,0,0,0) de "não venda".

Nesta abordagem continuamos com o problema dos atributos estarem pré-determinados, à semelhança da primeira abordagem descrita no inicio da presente secção. Todavia com este método é alcançado um melhoramento, em relação ao anterior, que consiste no facto do sistema não estar dependente de nenhum limiar (valor ou valores) a partir do qual se realiza uma determinada acção (classificação). Por exemplo, aqui não são definidas previamente regras, como a seguinte:

> *se ocorrerem as palavras: "vendo", "t3" e "preço", pelo menos uma vez cada uma, então o anúncio é classificado como "venda"*

Neste exemplo o limiar está implícito na expressão: "*pelo menos uma vez cada uma*". Na abordagem dos k-vizinhos, estes limiares estão, de certa forma, "escondidos" e dependentes dos exemplos de treino. Estes exemplos serão computados no momento da classificação do novo anúncio, não existindo portanto a necessidade de os pré-determinar.

A necessidade de definir previamente os atributos constitui uma dificuldade real, não só no que diz respeito à operabilidade do sistema e consequente problemática de migrar o mesmo para novos domínios, mas também quanto à sua eficácia ou desempenho. É sabido que uma má escolha de atributos ou até a escolha de demasiados atributos, não determinantes para a classificação, compromete seriamente o desempenho do sistema [Mitchell, 1997].

A próxima subsecção apresenta uma outra abordagem, na qual o problema da necessidade de definição prévia dos atributos é ultrapassada e que acabará por se mostrar mais adequada à satisfação da solução do nosso problema, aqui apresentado. No capitulo 5 é mostrada uma comparação, em termos de desempenho, entre estas duas abordagens, onde se concluí que o método apresentado a seguir produz uma solução preferível em relação ao actual.

4.2.2 Aprendizagem Bayesiana (Naive Bayes)

O algoritmo "Naive Bayes" enquadra-se na família de algoritmos de "Aprendizagem Bayesiana" (Bayesian Learning), tendo estes por base o teorema de Bayes que relaciona probabilidades condicionadas. Estes algoritmos são muito adequados para domínios cujos dados possuam um comportamento governado por probabilidades. No capitulo 3 é feita uma incursão na "Aprendizagem Bayesiana" e lá estão detalhados alguns princípios básicos, desta área da aprendizagem automática. A dificuldade prática pré-existente à aplicação de algoritmos de aprendizagem bayesiana consiste na necessidade de calcular um conjunto considerável de probabilidades ($O(m^k)$), relativas aos dados com os quais estamos a trabalhar, o que constitui uma barreira intransponível em muitas situações práticas, quer pela falta dessa informação, quer pela intratabilidade prática de todos esses cálculos. O algoritmo "Naive Bayes" é uma simplificação feita para contornar a dificuldade de calcular o enorme número de probabilidades, admitindo como hipótese a independência de certos eventos probabilísticos, mesmo que seja de forma forçada. Segundo vários autores ([Mitchell 1997], entre outros) este algoritmo encontra-se entre os mais capazes e melhores algoritmos de indução de classificadores para documentos de texto. Esta é precisamente a nossa necessidade concreta, apresentada nesta secção.

Um classificador "Naive Bayes" v_{NB}, tem a forma: $v_{NB} = \underset{v \in V}{argmax} P(v) \prod_i P(a_i | v)$, sendo V o conjunto das classes consideradas e a_i o atributo i da instância em causa. O valor dado por $P(v_j)$ designa a probabilidade da classe v_j e $P(a_i | v_j)$ a probabilidade do atributo a_i condicionado por v_j. Como se pode constatar pela fórmula anterior, a classificação é processada escolhendo a classe cujo o produto daquelas probabilidades atinge o valor máximo, temos um classificador "*máximo à posteriori*".

No nosso caso temos *V* = {"venda", "não venda"} e supondo *n* atributos: $a_1 \ldots a_n$, estes são constituídos por todas as palavras dos exemplos de treino (anúncios da Usenet). Assim ao conjunto $\{ a_1 \ldots a_n \}$ denomina-se de saco de palavras ("bag of words") [Mladenic 1998] e um documento é interpretado como um vector de frequências de palavras nesse documento – vector de "n-grams". Portanto, os nossas instâncias continuam sendo vectores ou pontos num

espaço euclidiano *n*-dimensional, á semelhança do abordado na secção anterior, com a diferença de aqui *n* ser um valor significativamente muito superior ao ali considerado. Enquanto que no algoritmo dos k-vizinhos o valor de *n* era igual ao número de atributos pré-estabelecidos, neste algoritmo *n* é o número de todas as palavras distintas que ocorrem nos exemplos de treino.

As probabilidades referidas anteriormente serão estimadas com base nos dados de treino e sendo assim $P(v_j)$ é o proporção de exemplos de treino da classe v_j e $P(a_i|v_j)$ poderá ser estimada calculando a proporção da ocorrência da palavra a_i nos exemplos da classe v_j. Todavia isto levanta um problema prático, que consiste na ocorrência de novas palavras nos exemplos de teste, que não ocorreram nos documentos de treino. Neste caso a probabilidade referida seria estimada em zero o que anularia o cálculo de v_{NB}, para qualquer classe considerada (v_j) o que conduziria à incapacidade de classificação do exemplo dado. Para contornar este problema, podemos utilizar o estimador da probabilidade de *Laplace* ou m-estimador [Mladenic 1998], dado pela seguinte expressão:

$$P(W|C) = \frac{Freq(W,C)+1}{\sum_{l} Freq(W_l,C) + |Vocabulario|}$$

Portanto, $P(W|C)$ significa a probabilidade condicional da ocorrência da palavra *W*, na classe *C*, *Freq(W,C)* é a frequência da palavra *W* na classe *C* e $|Vocabulario|$ designa o tamanho do "saco de palavras", referido anteriormente. Repare-se que neste estimador se *Freq(W,C)* = 0, o que significa que W é uma palavra nunca antes vista (exemplos de treino), então $P(W|C)$ atingirá o seu mínimo, mas que é diferente de zero e portanto não anulará o produto das probabilidades, consideradas para para a escolha de v_{NB}.

De seguida, no algoritmo 4.1, apresenta-se o presente algoritmo, descrito numa linguagem de pseudo-código. Este algoritmo foi implementado e no capítulo 5 ("Avaliação de Resultados") são apresentados os resultados obtidos e tecidas as devidas observações, comentários e comparações com outros algoritmos implementados, sobretudo a nível do desempenho.

IndutorNaiveBayesText(*Anúncios*, *V*: classes possíveis)
{
Vocabulário **<-** Conjunto de palavras distintas dos *Anúncios*
para cada $v_j \in V$ **fazer {**
$docs_j$ **<-** subconjunto de *Anúncios*, cujo valor é v_j
$P(v_j)$ **<-** $\frac{|docs_j|}{|Anúncios|}$
Texto **<-** $Concatenação_j \{docs_j\}$
n **<-** número de palavras distintas em *Texto*
para cada $w_k \in Vocabulário$ **fazer {**
n_k **<-** frequência de w_k em *Texto*
$P(w_k|v_j)$ **<-** $\frac{n_k + 1}{n + |Vocabulário|}$
}
}
}

Algoritmo 4.1 - Algoritmo de *Naive Bayes* para induzir um classificador de texto [Mitchell 1997]

A próxima subsecção consiste num esforço, no sentido de melhorar o desempenho do algoritmo *Naive Bayes* para a classificação de documentos de texto.

4.2.3 Aprendizagem Bayesiana (com escolha de atributos)

Na subsecção anterior constatamos que o algoritmo "Naive Bayes" faz uso de um elevado número de atributos – todas as palavras distintas dos exemplos de treino. É amplamente conhecido que os atributos considerados assumem uma grande importância em qualquer processo de aprendizagem automática. A utilização de atributos desnecessários pode comprometer a boa realização deste processo [Mladenic & Grobel. 1999]. Por outro lado um elevado número de atributos torna o processamento mais lento, como é evidente. Neste contexto, irá ser apresentada a aplicação de uma outra versão do algoritmo "Naive Bayes", que realiza um trabalho de pré-processamento importante, e que consiste na escolha

automática de um subconjunto de atributos "significativos, dentro do conjunto de todos os atributos possíveis.

Como foi referido anteriormente, os atributos são constituídos pelas diferentes palavras que ocorreram nos exemplos de treino. Após a reunião de todo este vocabulário, é gerada uma representação vectorial para cada documento (anúncio) existente na colecção de documentos utilizados no treino. Designe-se uma colecção desses documentos como sendo o seguinte conjunto: $\mathcal{D} = \{D_1, D_2, \ldots, D_m\}$, então para cada documento D_i é gerada uma sua representação vectorial v_i tendo em conta as palavras que ocorrem em $\mathcal{D}$. Assim supondo que o vocabulário reunido em $\mathcal{D}$ era constituído pelo conjunto $\mathcal{W} = \{w_1, w_2, \ldots, w_n\}$ então a componente j do vector v_i, este relativo ao documento D_i, seria igual à frequência de w_j em D_i, temos aqui o que é designado por representação dos documentos por vectores *TF* (*Term Frequency*) [Salton & Buckley 1987], expressando a frequência de w_j em D_i por $TF(w_j, D_i)$, isto é $v_i^{(j)} = TF(w_j, D_i)$. Uma outra representação muito similar a esta e largamente utilizada é a denominada *TFIDF* (*Term Frequency Inverse Document Frequency)* [Salton & Buckley 1987]. Neste caso $v_i^{(j)} = TF(w_j, D_i)\, IDF(w_j)$, sendo $IDF(w) = \log \frac{D}{DF(w)}$, $D = |\mathcal{D}|$ e $DF(w)$ é o número de documentos para os quais *w* ocorre pelo menos uma vez.

Pode suceder que *n* (tamanho do vocabulário) seja da ordem das dezenas ou mesmo centenas de milhar e regra geral muitos dos atributos considerados em $\mathcal{W}$ são completamente irrelevantes para a classificação dum documento. Assim interessa escolher um subconjunto de $\mathcal{W}$, para gerar a representação vectorial dos documentos, o que tem implicações na diminuição da dimensão dos vectores, aumentando assim a eficiência (menos atributos a considerar) do processo de classificação e aumentando também o desempenho geral do sistema (mais acertos), como alguns trabalhos o têm mostrado [Mladenic & Grobel. 1999], bem como nós também o pudemos constatar, para o nosso domínio, encontrando-se esse resultado expresso na secção 5.1.

A escolha de um subconjunto de atributos a partir de $\mathcal{W}$ é processada com a ajuda e orientação duma função de cálculo do valor dum atributo. Existem várias medidas conhecidas para este fim, em baixo são mostradas três que foram experimentadas neste trabalho e cuja comparação é referida na secção 5.1.

$$(1)\quad InfGain(w) = P(w)\sum_i P(C_i|w)\log\frac{P(C_i|w)}{P(C_i)} \;+\; P(\overline{w})\sum_i P(C_i|\overline{w})\log\frac{P(C_i|\overline{w})}{P(C_i)}$$

$$(2)\quad MutualInfoTxt(w) = \sum_i P(C_i)\log\frac{P(w|C_i)}{P(w)}$$

$$(3)\quad OddsRatio(w) = \log\frac{P(w|C_+)[1-P(w|C_-)]}{[1-P(w|C_+)]P(w|C_-)}$$

Mais algumas destas funções ou suas variantes, estão presentes em [Mladenic & Grobel. 1999]. Nas três formulas anteriores *P(w)* e $P(\overline{w})$ designam respectivamente as probabilidades de *w* ocorrer ou não, $P(C_i|w)$ a probabilidade da classe C_i sabendo que *w* ocorre, semelhantemente $P(w|C_i)$ é a probabilidade de ocorrer a palavra w na classe C_i e $P(C_i)$ é referente à probabilidade da classe C_i. Na terceira função C_+ e C_- designam duas classes, uma denominada positiva e outra negativa, respectivamente (quando a classificação é binária).

Fixando uma medida de avaliação dum atributo, das apresentadas anteriormente, é construída uma ordenação (*ranking)* de todos atributos de $\mathcal{W}$. Com base nesta ordenação é escolhido o subconjunto dos *r* melhores atributos ($0 < r \leqslant n$) que irão depois servir para gerar as instâncias representativas dos documentos.

O classificador naive bayes $v_{NB} = \underset{v\in V}{argmax}\, P(v)\prod_i P(a_i|v)$ utilizado na subsecção anterior foi aqui substituído por outro, mais adequado ao tratamento de texto, sugerido por [McCallum & Nigam 1998] e utilizado em [Mladenic & Grobel. 1999], dado por $v_{NB} = \underset{c\in C}{argmax}\, P(c|doc)$, sendo:

$$P(c|doc) = \frac{P(c) \prod_{w_j \in doc} P(w_j|c)^{TF(w_j, doc)}}{\sum_i P(c_i) \prod_{w_l \in doc} P(w_l|c_i)^{TF(w_l, doc)}}$$

em que *doc* consiste no documento a classificar, *P(c)* designa a probabilidade da classe $c \in C$, $P(c|doc)$ a probabilidade de termos a classe *c*, para o documento em causa e de forma geral $P(w|c)$ a probabilidade da palavra *w*, na classe *c*.

4.3 Extracção de Elementos - Preliminares

Esta secção contempla a segunda fase do trabalho realizado. Esta é designada de “segunda” por razões que se prendem com a ordem de trabalho, não significando que esta seja a menos importante ou naquela onde menos trabalho foi realizado, pelo contrário, esta é a área principal desta dissertação, onde mais esforço foi despendido. Aqui é feita uma incursão na área da extracção de informação - “Information Extraction”.

Como já foi referido anteriormente, o problema prende-se com a existência de consideráveis volumes de informação, apresentada sob a forma de documentos de texto, dos quais é necessário obter um conjunto de informações consideradas importantes para um determinado domínio e fim. O problema não está na quantidade de informação documental apresentada, mas na forma como esta informação se apresenta – sem qualquer tipo de estrutura explicitamente representada – o que normalmente se denomina de “informação não estruturada”.

Um documento de texto, não é mais que uma sequência finita de frases. Cada frase ou conjuntos de frases vão apresentando uma sequência de informações ou ideias. Esta sequência é determinada e dirigida pelo autor durante o processo de criação de um documento. Dois autores podem escrever dois textos, com exactamente os mesmos elementos informativos, mas apresentar esses elementos numa ordem diferente e revesti-los de um fundo literário que depende subjectivamente de cada um dos autores. Afinal é nisto que reside a génese da riqueza literária criada pelo ser humano.

É neste sentido também que a informação textual é referida aqui como sendo “não estruturada” e portanto a dificuldade está quando é necessário processar informação deste género, por exemplo encontrar frases que informem determinados eventos ou aspectos da realidade. Facilmente se reconhece a enorme dificuldade e esforço humano que é necessário despender, para a realização de tarefas deste género e consequentemente as dificuldades que se deparam se quisermos que esta informação seja pesquisada por processos automáticos.

Em contrapartida a informação “estruturada”, contém uma estrutura explicita que lhe está subjacente. Como exemplo de informação neste estado, temos a informação armazenada

numa base de dados, num quadro de um documento ou no formato XML. Tomando o exemplo das bases de dados, o que temos são tabelas que, regra geral, possuem relações entre si e traduzem um modelo semântico definido previamente – "Modelo Entidade-Relacionamento". Uma tabela é constituída por um conjunto de campos e define uma relação entre esses campos, modelando certos aspectos do "mundo real". Um campo é um elemento atómico numa base de dados, pertencente a uma tabela, e designa um elemento ou atributo do "mundo real". Na próxima figura apresenta-se um exemplo de uma tabela que modela a entidade *cliente*, num determinado negócio.

Cliente
Numero
Nome
Morada
Contacto

Figura 4.4 - Exemplo duma tabela.

A tabela anterior é constituída por quatro campos, sendo os três últimos atributos relativos a um determinado cliente. É neste sentido que se designa aqui "informação estruturada", informação organizada segundo certa estrutura, previamente determinada.

Temos portanto uma considerável colecção de documentos, dos quais pretendemos extrair elementos para proceder ao preenchimento de uma tabela do género da apresentada na figura anterior. O desafio que é posto aqui consiste na geração de informação estruturada a partir de fontes documentais não estruturadas. Como fazer isto? Que abordagens possíveis existem? Serão apresentadas três abordagens possíveis, começando pela mais simples e consequentemente mais óbvia, isto é, a primeira abordagem susceptível de surgir na mente de quem tenta atacar um problema desta natureza.

4.3.1 Extracção Utilizando Regras Pré-definidas

Temos portanto um problema que se apresenta e que pode ser sumariado como se segue:

> *"Existe uma numerosa colecção de documentos relativos a um determinado assunto fixo, dos quais se quer obter um conjunto de elementos relevantes, pré-estabelecidos"*

Para já poderá ser questionado qual o significado de "elemento relevante", num documento. Todavia a própria expressão referida elucida o conceito imanente a esta, pois num texto um elemento relevante é algo que depende da perspectiva do individuo ou grupo de indivíduos que o observam, isto é, algo estritamente subjectivo e relacionado com um determinado objectivo que se tenta alcançar. Para uns, elemento relevante, poderá ser informação relacionada com a compra de acções, por exemplo, enquanto que para outros o relevante poderá consistir em informação relacionada com a meteorologia.

Sendo assim, os elementos relevantes já estão definidos à priori e, naturalmente, dependem do domínio no qual se está a trabalhar e do ponto de vista de quem os define. A título exemplificativo, suponhamos que estávamos no domínio da compra e venda de um determinado bem: computador, viatura, habitação. Tínhamos portanto uma colecção considerável de pequenos anúncios desse bem, tal como o mostrado na figura seguinte:

From: INVESCACEM, LDA (invescacem@mail.telepac.pt)
Subject: vendo apartamento t3
Date: 2002-09-07 12:24:09 PST

```
na Cidade de Agualva Cacém, na freguesia do Cacém, concelho de
Sintra, vendo apartamento t3, junto da estação da CP, com
quartos 15,19 m2; 15,66 m2; 18,56 m2 e sala 31,10 m2, ainda
está em estuque nunca foi pintada, é espectacular; belissima
vista desafogada, muito soalheira.

OPORTUNIDADE
PREÇO 128.440,00?

mais informações
tel 219136000
tlm 917621828
```

Figura 4.5 - Anúncio de venda de apartamento, com elementos relevantes assinalados.

Então um elemento relevante poderá ser o *preço* desse mesmo bem. Isto significa que estaríamos interessados em extrair todas as informações relativas ao preço dos produtos, nos vários documentos. A figura anterior exemplifica um anúncio típico de venda de habitação. O

texto destacado refere três elementos que podíamos considerar relevantes, neste caso: preço, tipologia e localização.

Mas então como armazenar os elementos extraídos? Evidentemente que o objectivo final é ficar com essa informação armazenada segundo uma organização estruturada, tal como foi referido. Para tal será utilizado algo semelhante a uma tabela de uma base de dados e que é habitual designar por "Template" [Hobbs & Israel 1994]. Neste trabalho esta estrutura será designada de tabela. Assim e relativo a anúncios do género da figura 4.5, uma tabela que contemplasse os três elementos relevantes referidos, poderia ser a mostrada na figura seguinte:

Tipo
Preço
Local

Figura 4.6

Neste caso teríamos uma tabela constituída por três componentes. Estes são normalmente designados por "*slots*" [Hobbs & Israel 1994] ou *campos.* Aqui serão denominados de campos, à semelhança de uma tabela numa base de dados.

Em suma o nosso objectivo está esquematizado na figura 4.7, que se segue:

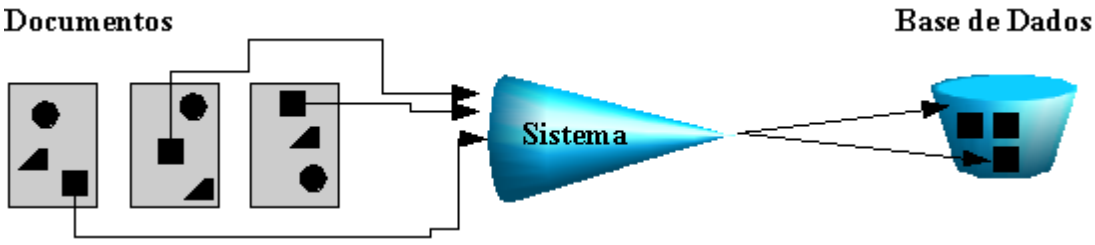


Figura 4.7 - Objectivo: extracção de elementos relevantes, para inserção em estruturas bem definindas.

Coloca-se então a seguinte questão: Como realizar esta tarefa? Tenhamos presente que este trabalho ainda é feito em larga escala sem auxílio de qualquer sistema automático, na maioria das organizações. Ali existem equipas cujo trabalho consiste em ler documentos de certo tipo, por exemplo como o da figura 4.5, e obter as informações relevantes inserindo

posteriormente esses elementos em estruturas de informação, bem definidas e protocolares, como é o caso de uma base de dados ou sob uma estrutura XML. Até ao momento o ser humano ainda detém o melhor nível de desempenho, na realização desta tarefa. Os sistemas automáticos experimentados, até agora, atingem um nível de desempenho que ronda os 60% de um ser humano, atingindo os 80% no melhor dos casos [Appelt & Israel 1999]. Então qual é o problema? Porquê não deixar os humanos realizar este tipo de tarefas, pois se o realizam tão bem?

Como todos reconhecemos, para realizar um trabalho do género do descrito é exigido um grande esforço, realizando uma tarefa bastante trabalhosa, repetitiva e um pouco aborrecida. Por outro lado, embora as pessoas consigam acertar mais, repare-se nas percentagens anteriores, os sistemas automáticos conseguem processar um maior numero de informação por unidade de tempo. Isto deve-se sobretudo à rapidez no processamento da informação, por parte dos computadores, que inclusive tem vindo sempre a aumentar nos últimos anos.

Portanto, apesar da percentagem de desempenho de uma máquina ser inferior à de um ser humano, o trabalho daquela é desejável em muitos domínios pois como o numero de documentos processados é muito superior, o volume de informação válida extraída é também superior, por unidade de tempo.

Recolocando de outra forma a questão anterior, tem-se:

"Que abordagem podemos conceber, para a solução do problema proposto?"

A primeira abordagem, que poderá ocorrer consiste em definir um conjunto de regras apropriado para a extracção de um determinado elemento relevante. Este conjunto será definido por um utilizador que conhece minimamente o domínio e terá de ser definido um conjunto de regras, por cada elemento relevante considerado. Portanto estamos a visualizar um sistema, no qual o utilizador introduz conjuntos de regras de extracção, obedecendo estas a certas convenções. Este sistema processará o conjunto de documentos e tentará aplicar as regras à sequência do texto, para assim encontrar o que pretende. A figura seguinte esquematiza esta descrição.

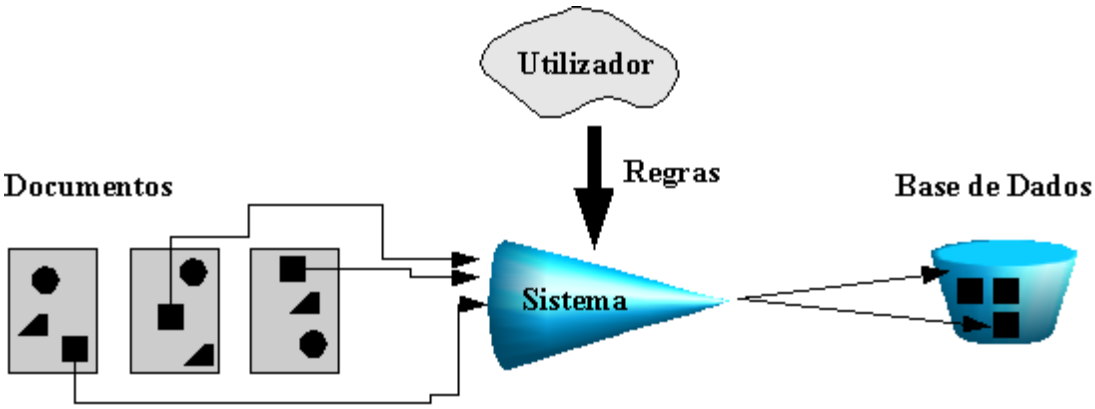


Figura 4.8 - O utilizador fornece um conjunto de regras de extracção ao sistema.

Uma questão importante aqui é: qual a forma das regras a fornecer ao sistema? Com base em quê é que estas são formuladas? Sendo um documento textual constituído por um conjunto de frases que por sua vez são constituídas por uma sequência de palavras, é de esperar que uma regra relativa à extracção de um elemento relevante, envolva palavras do texto, pelo menos aquelas que detém alguma relação com as ocorrências do elemento relevante no texto.

Passaremos a apresentar um exemplo para melhor ilustrar o que está sendo exposto. Suponhamos que o nosso domínio consiste em anúncios de compra e venda, mais concretamente, compra e venda de habitações. Vamos supor também, para simplificar, que o único elemento relevante que temos definido, diz respeito ao preço da habitação. Designe-se esse elemento relevante por *preço*, isto é, temos uma única tabela (template), com um único campo (slot), denominado de *preço*[1]. A título exemplificativo apresentam-se três regras relativas à extracção do elemento *preço*, que serão explicadas de seguida.

```
Regra 1: preço(texto, i) se texto[i-1]="preço".
Regra 2: preço(texto, i) se texto[i-2]= "quantia" e texto[i-1]="de"
Regra 3: preço(texto, i) se texto[i-1]= "valor" e texto[i+1]="euros"
```

Quadro 4.1 - Regras relativas à extracção do elemento *preço*

No quadro 4.1, "`texto`" é o vector da sequência de palavras que surgem num documento e "texto`[i]`", designa a i-ésima palavra, nessa sequência. Na notação empregue, "**`preço(`**`texto, i`**`)`**", no cabeçalho duma regra, indica que o objectivo é verificar se a i-ésima palavra do texto é uma instância do elemento *preço*. Para cada regra, esta verificação está

1 Os elementos relevantes considerados serão designados em itálico.

dependente da satisfação de condições, para a sua execução, mais concretamente poderemos ter conjunção e disjunção de condições simples. Nas regras apresentadas como exemplo, uma condição simples é verificada só se a i-ésima palavra do documento é igual a uma determinada constante pertencente ao vocabulário da linguagem, como é o caso da palavra "preço" ou "valor". Reforçando o que foi dito, estas constantes são especificadas pelo utilizador, bem como toda a regra. Repare-se que a sintaxe das regras do quadro 4.1 utiliza elementos da linguagem natural (como o "se", "e", "ou") e obedecem a um formalismo muito conhecido - "Clausulas de Horn" [Bratko 2001].

As regras apresentadas em 4.1, podem ser utilizadas no contexto de uma função que depende do texto dum documento e dum dado elemento relevante. À primeira satisfação de uma das regras, seria devolvido o índice da palavra ou do inicio e fim da expressão que constitui o elemento relevante. Em baixo é apresentado o esqueleto desta função, em pseudo código:

```
índices_de_conceito(texto, conceitox)
{
      para cada i ≤ j
            se conceito(conceitox, texto, i, j)
                  índice_de_conceito  <- (i:j)[1]

      índice_de_conceito  <- NA[2]
}
```

A função procura no conjunto das regras definidas, sucedendo à primeira regra com sucesso e falhando caso nenhuma sucedesse. Esta função podia ser utilizada para o preenchimento dos campos das tabelas definidas, como mostrado no seguinte exemplo (em pseudo-código):

```
(a:b) <- índices_de_conceito(texto, "preço")
se (a:b) ≠ NA então  TABELA.preço <- texto[a:b]
```

Tendo presente o quadro 4.1, embora tenhamos somente três regras, facilmente se reconhece ali já alguma capacidade para extrair preços dos anúncios referidos.

Todavia observamos que este pequeno conjunto se mostra ainda muito insuficiente. Aplicando aquele quadro de regras a um conjunto de 100 documentos, relativos a anúncios de

1 Designa o intervalo desde o índice i até ao índice j
2 NA – Não Aplicável.

compra e venda de habitação, obteve-se um desempenho bastante pobre e aquém do desejável, pois as três regras só conseguiram encontrar e extrair 5 preços correctamente. Além disso foram extraídos 2 elementos incorrectamente. Tendo em conta que nos 100 documentos[1] existiam 92 ocorrências de preços, podemos concluir que desempenho é muito baixo. É evidente que também só tínhamos 3 regras e portanto pode-se argumentar que se o utilizador definir mais regras então o desempenho será superior, todavia para que o mesmo se aproxime dos níveis referidos anteriormente, 60% do desempenho de um ser humano, serão certamente necessárias muito mais regras, semelhantes às do quadro 4.1, e isto só para extrair elemento *preço*. Se tivermos várias tabelas a preencher, tendo cada uma vários campos, imagine-se o esforço que seria necessário despender.

Um dos problemas com as regras de extracção do género das do quadro 4.1, é que estas são demasiado especificas. Basta ter em conta a enorme variedade sob a qual um preço pode ser expresso num anúncio, vejamos alguns exemplos:

... preço: 100 euros ...
... preço 75 contos ...
... pelo preço de 100 euros ...
... no valor de 75.00 € ...
... pelo valor de 27000 contos ...
... valor: 12.00 cts ...
...pela modica quantia de 81.00 euros ...
... aproximadamente por 77.00 dolares ...
... 1000.00 ..
...
...

Quadro 4.2 - Formas de preços

Por outro lado estas regras dependem fixamente das palavras que se encontram na vizinhança imediata do elemento a extrair, isto é da palavra ou sequência de palavras anteriores ou/e da palavra ou sequência posterior. Porém existem situações em que um elemento a extrair poderá não depender de qualquer vizinhança ou por outro lado poderá depender de uma palavra que se encontra em média 3 posições antes da palavra alvo, por exemplo. Tenhamos em mente

1 Para ver pormenores acerca dos dados, consultar secção 4.1 e 4.4.1

existem variadíssimas formas de escrever um anúncio do género do apresentado na figura 4.5, dependendo do estilo de escrita de cada autor.

Para que um utilizador fosse capaz de definir um quadro de regras minimamente aceitáveis ele teria de conhecer muito bem os documentos a processar, para assim definir regras adequadas. Este teria de analisar um grande número destes documentos e o seu esforço tenderia a se aproximar do de um operador humano que tem a função de realizar a tarefa de extracção. O que estamos a defender aqui é que o utilizador teria de "aprender" o padrão, as especificidades, o estilo dos anúncios a processar, para depois ser capaz de definir regras convenientes. Portanto existe a necessidade de um processo de adaptação ou aprendizagem. É precisamente esta ideia que será apresentada nas duas sub-secções que se seguem, para além de outras também importantes.

Antes de iniciar a próxima subsecção, apresentamos alguns pontos que mostram claramente, em nosso entender, as desvantagens e limitações desta primeira abordagem.

Desvantagens e Limitações

Em primeiro lugar esta abordagem exige que alguém defina as regras, tem de existir todo um esforço consciente na definição de regras de extracção, por parte de alguém. Como foi referido, o numero de regras, para que o sistema atinja um nível de desempenho minimamente aceitável, poderá ser elevado, só para um campo. Este número será muito ampliado se tivermos em consideração a possibilidade de ter várias tabelas e estas, por sua vez, vários campos. Portanto, o sistema exige que seja despendido um elevado esforço, embora seja verdade que após a criação das regras, este passaria a trabalhar autonomamente. Porém de tempos a tempos seria necessário proceder a algum ajuste das regras, para contemplar novas situações ou para dar resposta a uma eventual inovação na forma textual ou pequenas mudanças no estilo de escrita dos anúncios. Assim o tal "utilizador especialista", responsável pela definição de regras, não poderia "ausentar-se" definitivamente do sistema mas teria de continuar a existir um trabalho de manutenção, por parte deste utilizador e um olhar crítico sobre a actuação do sistema, de forma a realizar ajustes se necessário.

Uma segunda desvantagem desta abordagem, que é uma consequência directa da existência de um "utilizador especialista" a definir as regras, é o facto destas serem definidas segundo critérios subjectivos do próprio utilizador. Assim o sistema ficaria aberto e dependente da subjectividade dessa pessoa, não sendo garantido que a sua actuação seria a melhor, por exemplo se este utilizador fosse pouco experiente. Isto não significa que as regras definidas fossem completamente erradas, mas bastaria que estas estivessem mal ajustadas à realidade, fossem por exemplo demasiado gerais ou demasiado específicas, comprometendo desta forma o bom desempenho do sistema.

Uma outra limitação desta abordagem prende-se com a possibilidade de existirem regras muito complexas e que um utilizador, mesmo sendo um bom especialista no domínio em questão e conhecendo bem os anúncios, não teria capacidade de descobrir e portanto definir. Uma das grandes vantagens do "*Data Mining*" é a descoberta de conhecimento que não seria facilmente perceptível a um ser humano e assim permitir a possibilidade de serem omitidas regras importantes.

Podemos ter ainda o sistema a operar em ambientes muito dinâmicos, em que o tipo e a forma dos documentos processados vai mudando ligeiramente com o tempo, por exemplo. Uma mudança no estilo de documentos implica um desajuste das regras de extracção e consequentemente a necessidade de redefinição de regras.

Um ultimo ponto, muito relacionado com o anterior, diz respeito à possibilidade ou necessidade de migrar o sistema para outros domínios. Por exemplo, o sistema estar a processar anúncios relativos à comercialização de um produto e depois querermos colocá-lo a processar textos de História, com o objectivo de, por exemplo, extrair personagens históricas, ou datas de eventos importantes, etc. Isto exige uma completa redefinição do quadro de regras, novamente um grande esforço humano é exigido, para que o sistema fique adaptado ao novo domínio.
As razões aqui apresentadas denotam a debilidade desta abordagem e evidenciam a necessidade de uma outra abordagem, mais capaz de fazer face ao problema em causa.

4.3.2 Extracção Utilizando Sistemas de Aprendizagem

Pelo apresentado no final da secção anterior, fica clara a necessidade de uma abordagem tecnicamente mais adequada, para atacar o problema apresentado. Das limitações expostas, podemos perceber que uma das principais dificuldades e entraves, reside no facto de terem de existir utilizadores especialistas a "definir as regras do jogo", dito de uma forma menos formal. Portanto, quase todas as dificuldades apresentadas estão relacionadas com o facto do sistema depender do "input" de regras, que têm de ser fornecidas, regras essas que são determinantes no desempenho do sistema.

A abordagem apresentada nesta secção "liberta" o utilizador da necessidade de introduzir regras no sistema. O objectivo aqui é explorar a aprendizagem, isto é, pretende-se que o sistema, mediante a interacção com o mundo, tenha a capacidade de aprender autonomamente, para depois poder agir convenientemente. Coloca-se então a seguinte questão: "*aprender o quê?*". No nosso caso concreto, queremos que aprenda as regras relativas à extracção dos elementos considerados relevantes, em vez de estas serem definidas e fornecidas por um agente externo.

Este é um importante passo, que é dado na direcção da Inteligência Artificial (IA), pois uma importante subárea desta, consiste na *Aprendizagem Automática*[1]. Esta área da IA tem como alvo de dotar os sistemas, desta capacidade tão familiar e importante para o ser humano[2].

O problema, tal como foi referido, consiste na existência de grandes quantidades de informação não estruturada, sob a forma de texto. O objectivo consiste em produzir determinada informação estruturada, extraindo-a do grande volume de informação não estruturada disponível. O nosso sistema enquadra-se num ambiente, no qual recebe entradas ou percepções e no qual realiza acções[3]. No nosso caso esse ambiente consiste no vasto universo de documentos existentes e as acções consistem na identificação e extracção dos elementos relevantes.

1 Como referido na secção 3.2.
2 Ver secção 3.2.
3 De acordo com literatura recente da IA [Norving & Russel 1995]

Em todos os processos de aprendizagem, sejam automáticos ou não, existe um factor comum e que ainda não foi referido – "o treino" - uma fase fundamental da *aprendizagem supervisionada*. Se queremos um agente[1] que não dependa do fornecimento de regras, por parte de um humano, ou outro agente qualquer, mas que as induza, então teremos de o submeter a um processo de treino. Este processo é normalmente utilizado para induzir um classificador, como por exemplo uma rede neuronal ou uma árvore de decisão. Este é depois utilizado na solução dum problema. Então, no nosso caso concreto, em que é que consiste o processo de treino? Este traduzir-se-á em fornecer, ao sistema, um conjunto de exemplos previamente anotados por alguém e denominados de exemplos de treino. A partir destes exemplos será induzido um classificador para extrair os elementos relevantes considerados, do texto dos anúncios.

Geração de Exemplos Positivos

Um exemplo ou instância de treino, pode ser descrito como um vector de n elementos, dos quais um é denominado de "classe" e os restantes n-1 elementos são os "atributos", da instância. Aquilo que se procura saber é a relação que existe entre os atributos e classe, isto é, de que forma é que os valores dos atributos contribuem para o valor da classe.

No nosso problema concreto, a classe pode assumir dois valores: "sim" ou "não". Isto significa que se deve ou não, respectivamente, extrair uma determinada palavra ou expressão do documento, num determinado instante. Quanto aos atributos, temos, nesta abordagem, um conjunto fixo e pré-determinado, cujo domínio se encontra no universo de palavras e expressões existentes na língua portuguesa.

Portanto, em vez de serem fornecidas regras ao nosso agente, como na secção anterior, serão fornecidos exemplos de treino devidamente anotados. Coloca-se então a questão de saber em que é que consiste uma instância de treino, no nosso caso, uma vez que estamos a trabalhar com texto e não com bases de dados, por exemplo, onde é mais fácil definir uma instância. Aqui serão fornecidos, ao sistema, documentos previamente anotados, por agentes humanos que marcam, de uma forma convencionada, a ocorrência dos elementos relevantes no texto.

1 Na literatura mais recente da IA, o termo "agente" é empregue para designar um sistema que utiliza metodologias da IA (secção 3.2).

Por exemplo, nos anúncios referidos na secção anterior, relativos à compra e venda de habitação, considerando que o preço referido nesses anúncios constituía um elemento relevante, iria ser anotado, nos mesmos, as ocorrências desses elementos. Estas anotações podem ser realizadas de várias formas, mas aqui obriga-se a que seja respeitada uma sintaxe do género SGML, mais concretamente XML. De seguida apresenta-se o anúncio mostrado na figura 4.5, com os elementos que ali foram considerados relevantes, agora anotados com etiquetas XML.

From: INVESCACEM, LDA (invescacem@mail.telepac.pt)
Subject: vendo apartamento <tipo>t3</tipo>
Date: 2002-09-07 12:24:09 PST

```
na Cidade de <local>Agualva Cacém</local>, na freguesia do
<local>Cacém</local>, concelho de <local>Sintra</local>,
vendo apartamento <tipo>t3<tipo>, junto da estação da CP, com
quartos 15,19 m2; 15,66 m2; 18,56 m2 e sala 31,10 m2, ainda
está em estuque nunca foi pintada, é espectacular; belissima
vista desafogada, muito soalheira.

OPORTUNIDADE
PREÇO <preco>128.440,00</preco>?

mais informações
tel 219136000
tlm 917621828
```

Figura 4.9 - Exemplo de um texto anotado

Como queremos que o agente induza regras de extracção para cada elemento relevante, teremos de fornecer exemplos de treino para cada elemento. Como se pode constatar da figura anterior, num mesmo documento podem ocorrer instâncias de vários elementos relevantes, por exemplo: *preço*, *local*, *tipo*. Cada um destes elementos pode ocorrer mais do que uma vez, num documento, por exemplo pode haver num mesmo anúncio duas referências à localização, como acontece no exemplo do anúncio da figura anterior. Isto significa que cada anuncio pode contribuir com mais do que uma instância de treino, relativamente a um elemento relevante.

Aprendizagem Com Árvores de Decisão

Uma árvore de decisão é um classificador amplamente utilizado em muitos domínios que necessitam e envolvem aprendizagem automática. A par de uma árvore de decisão podem ser

criadas regras do género: " `if <condition> then <action>`"[1]. Este é precisamente o caminho delineado nesta secção.

Assim as regras obtidas no processo de treino serão utilizadas pelo sistema e aplicadas nos novos documentos, de modo a extrair os elementos relevantes e preencher os campos correspondentes nas tabelas, previamente determinadas.

Como é sabido, uma instância de treino a submeter a um indutor de árvores de decisão, não pode ser um fragmento de texto, mas como foi referido, terá de possuir uma estrutura semelhante a um vector de n elementos, dos quais n-1 são valores de atributos e o restante consiste na classe a que pertence a instância. No nosso caso temos dois valores possíveis para a classe: "sim", "não". Portanto, tem de haver um trabalho de pré-processamento dos dados (texto), de forma a que sejam geradas as instâncias de treino. O maior problema reside na escolha dos atributos: o que são os atributos das nossas instâncias de treino, relativamente a um dado elemento relevante? Esta foi uma questão que exigiu um raciocínio mais cuidadoso e uma reflexão mais demorada, atendendo a que existem muitas possibilidades.

À semelhança das regras dadas como exemplo na secção anterior no quadro 4.1, podemos considerar as palavras vizinhas do elemento de extracção : por exemplo, a palavra anterior, ou as duas palavras anteriores ao elemento relevante, ou além do esquema já definido a palavra posterior, que se encontra à direita do elemento. De facto, é legitimo pensar que existe alguma relação entre a ocorrência de um determinado elemento relevante e as palavras que lhe estão próximas, no documento. Por exemplo, é comum o preço de um produto ser precedido da palavra "preço" ou da expressão "valor de". A seguir apresentam-se exemplos de instâncias de treino para o elemento *preço*, considerando como atributos atributos as duas palavras que precedem o elemento:

Instâncias relativas ao elemento preço

```
(texto[i-2], texto[i-1], classe)
-----------------------------------
(NA[2], preço, sim)
(pelo, preço, sim)
(NA, valor, sim)
(valor, de, sim)
```

1 Ver secção 3.5
2 NA – Não Aplicável.

Repare-se que o domínio dos atributos é formado pelo vocabulário da linguagem, teoricamente, todas as palavras da língua portuguesa. Permanece portanto uma questão importante e que ainda não foi esclarecida:*"Quem define os atributos a considerar?"*

Nesta secção e como uma solução intermédia, assume-se que os atributos são definidos previamente por um especialista da área, que conhece bem a estrutura dos anúncios e sabe, por exemplo, que é suficiente considerar certas palavras (ex: a palavra esquerda e direita).

Geração de Exemplos Negativos

Estamos assim, quase em condições de gerar um conjunto de instâncias de treino, pois ainda permanece uma dificuldade não solucionada. Até agora só foi feita referência à geração de instâncias todas relativas a uma mesma classe: "sim". Portanto só temos instâncias relativas à acção de extrair um elemento, que como é sabido, neste contexto (classificação binária) são geralmente denominadas de instâncias positivas. Faltam-nos as instâncias negativas, ou seja aquelas que nos indicam que não devemos extrair. Este é um problema impeditivo e que tem de ser contornado.

Nesta abordagem optou-se por gerar as instâncias negativas, a partir do restante texto do anúncio, isto é o texto que não entra na constituição de instâncias positivas. Será apresentado um exemplo para que melhor se perceba. Suponhamos que estamos nos anúncios, que temos vindo a referir, os quais possuem como elemento relevante anotado o elemento *preço*. Suponhamos também que são considerados dois atributos: a palavra que precede a ocorrência do elemento e a que sucede a mesma. Seja **N** o numero de palavras envolvidas num instância positiva deste género. Relativamente a esta instância, podemos considerar o resto do texto do documento como potencial gerador de instâncias negativas, do elemento *preço*, isto partindo do princípio que nesse anúncio particular, o elemento *preço* ocorre uma única vez. Então geramos uma instância negativa relacionada com instância positiva existente, obtendo aleatoriamente nesse texto uma sequência de palavras com comprimento **N**. Por exemplo, no anúncio apresentado na figura 4.9, considerando como atributos, relativamente ao *preço*[1], a

1 O elemento (conceito) *preço* não se deve confundir com a palavra "preço", utilizada neste exemplo.

palavra esquerda e a palavra direita ao elemento, temos como instância positiva para esse elemento, a seguinte:

(preço, mais, sim)

Uma correspondente instância negativa, poderia ser a obtida de: "nunca foi pintada" e a instância negativa gerada seria:

(nunca, pintada, não)

Pelo processo descrito, é gerada uma colecção de instâncias de treino, com igual numero de instâncias negativas e igual numero de instâncias positivas.

Geração de Regras

Contornados estes obstáculos estamos em condições de submeter as instâncias de treino a um indutor de árvores de decisão, por exemplo o C4.5 [Quinlan 1993] ou C5 [Quinlan 1997]. Como foi referido e é amplamente conhecido, a partir de uma árvore de decisão podemos obter uma colecção de regras. No nosso caso essas regras indicam-nos a acção de extracção ou não, mediante condições sobre os atributos considerados. Assim teremos **n** regras da forma:

"if <condition> then <action>"

Nestas regras e em geral, a condição ("*condition*") poderá ser uma composição lógica de condições elementares – conjunção lógica. No exemplo que se tem vindo a apresentar, relativo ao elemento *preço*, e tendo em conta os atributos "palavra esquerda" (texto[i-1][1]) e "palavra direita" (texto[i+1][2]), uma condição poderia ser: texto[i-1] = "preço" ou então uma conjunção do género: texto[i-1] = "preço" `and` texto[i+1] = "contacto". Convém notar que as regras apresentadas nesta secção não seguem a sintaxe das geradas pelo C4.5 ou C5, mas uma representação própria. Para este mesmo exemplo, o conceito acção ("action"), resume-se a: "`extrair`" ou "`não extrair`". A acção "extrair", pode ser notada por:

1 Considerando que o elemento ocorre na posição *i*.
2 Duma forma simplista, estamos a pressupor aqui que o elemento relevante é composto duma única palavra.

"`tabela.preço <- texto[i]`"

isto significa que o campo denominado de "preço" da tabela, é preenchido com a i-ésima palavra do texto. Assim um exemplo de regra obtida seria o mostrado em baixo:

`if (` texto[i-1] = "preço" `and` texto[i+1] = "euros" `) then`
tabela.preço <-- texto[i]

Experiência

Segundo a abordagem descrita nesta secção, foi realizada uma experiência envolvendo os anúncios referentes à venda de habitação e já mencionados na secção anterior. Para simplificar, resolveu-se considerar como único elemento relevante o preço da habitação, designado por *preço*. Optou-se também por considerar como atributos as palavras esquerda e direita à ocorrência do elemento relevante. As instâncias de treino foram obtidas a partir de uma colecção de anúncios, contendo esta 55 ocorrências do elemento *preço*, alguns destes anúncios são apresentados no *Anexo A*. Apresenta-se a seguir um subconjunto das instâncias de treino obtidas a partir dos anúncios anotados.

Texto[i-1]	***Texto[i+1]***	***Acção ou Classe***
preço	euros	sim
estado	urgente	não
valor	euros	sim
dos	contactar	não
m2	euros	sim
apenas	euros	sim

Note-se que as instâncias mostradas anteriormente obedecem à formatação exigida pelo sistema C5 [Quinlan 1997], isto porque para proceder à indução das regras de extracção utilizou-se este sistema. O sistema C5 gera um ficheiro com as regras[1] induzidas e realizando uma transformação conveniente sobre o mesmo obtém-se a lista de regras com o aspecto e sintaxe descrita anteriormente. Mostra-se de seguida algumas dessas regras obtidas:

1 Exemplo de regra do C5: "`TextoEsq=preço, TextoDir=euros -> class sim`"

```
if ( texto[i-1]="preço" and texto[i+1]=”euros” ) then
      tabela.preço <-- texto[i]

if ( texto[i-1]="valor" and texto[i+1]=”euros” ) then
      tabela.preço <-- texto[i]

if ( texto[i-1]="preço" ) then
      tabela.preço <-- texto[i]
```

Figura 4.10 - Regras de extracção geradas, nesta primeira abordagem.

Resultados desta Experiência

Após a realização deste processo de indução de regras de extracção, ficamos com uma colecção que constitui uma “teoria” que permite processar anúncios. O passo seguinte é pôr esta teoria à prova, testando o desempenho das regras induzidas. Para tal foi utilizada uma outra colecção de anúncios diferente da colecção de treino, denominada de colecção de teste e que contém 86 ocorrências do elemento *preço*. Aplicando as regras induzidas, a esta colecção foram realizadas 327 extracções, tendo se conseguido obter só 5 extracções correctas. Isto significa que, em termos de desempenho, o sistema básico não funciona muito bem. No capitulo 5 apresentamos novamente estes resultados e é feita a avaliação, em termos de medidas habitualmente usadas nesta área (ex: precision, recall), que são definidas ali.

Comentário

Os valores mostrados, resultantes da experiência realizada, são claramente muito maus. Por um lado está a ser extraído muito “lixo”, pois o sistema concretizou 327 extracções, quando só existiam 86 ocorrências do elemento relevante e dessas 327 extracções só 5 eram realmente preços, o que dá uma precisão muito baixa da informação extraída. Por outro lado o sistema está a encontrar muito poucas ocorrências do elemento *preço*, pois das 86 existentes só conseguiu obter 5.

Limitações

Apesar do importante passo dado, no sentido de "libertar" o utilizador da responsabilidade da definição e fornecimento das regras ao sistema, tornando-o mais autónomo, continuam a existir algumas dificuldades nesta abordagem, que serão apresentadas de seguida.

Ainda relativamente ao utilizador, este continua a determinar, à priori, quais os atributos mais importantes a considerar, no processo de aprendizagem. Portanto persiste alguma pré-determinação em relação à escolha dos atributos e como é sabido, num processo de aprendizagem automática, uma má escolha do atributos é determinante para a ineficácia da teoria aprendida, tendo más consequências nos resultados. Portanto é desejável que seja o sistema a realizar a escolha dos atributos, de forma dinâmica e com base nalguma medida orientadora.

Uma terceira limitação está relacionada com o uso das próprias palavras nos domínios dos atributos. Isto é, para além dos atributos terem sido pré-determinados, estes são posicionais (ex: texto[i-1]) e assim o domínio destes fica restrito a um conjunto de palavras que ocorrem nas instâncias de treino. Com tratar as situações em que um atributo posicional pode assumir uma grande variabilidade de valores? Um exemplo disto é um atributo que possa assumir valores numéricos. Por exemplo, é natural pensar-se que o próprio texto[i] (o elemento a extrair) constitua um potencial atributo a considerar. Um exemplo de uma regra envolvendo texto[i], poderia ser:

```
if ( texto[i]="121.500" and texto[i+1]="euros" ) then
      tabela.preço <-- texto[i]
```

Repare-se o quão específica é esta regra e de pouca utilidade para extrair, nos novos anúncios. O problema aqui é que, por exemplo, o texto[i] pode assumir uma grande variação, podendo possuir um domínio que é mesmo infinito (todos os valores numéricos). Repare-se na seguinte lista :

```
m2 <preco>121.500</preco> euros
valor <preco>24.000</preco> contos
andar <preco>109.735,54</preco> euros
Preço: <preco>32 mil </preco> contos
preço: <preco>77.313</preco> euros
Seixal <preco>45.000</preco> contos
```

Nesta pequena lista de 6 exemplos de ocorrências do elemento *preço*, verifica-se que para cada ocorrência tem um texto[i] diferente (`121.500, 24.000, etc`), todavia constata-se que quase todos são valores numéricos. Portanto existe uma categoria abstracta que compreende a quase totalidade destes elementos. Isto sugere que talvez seja promissor considerar determinadas formas abstractas, representativas de expressões de texto, isto é, generalizações. Por exemplo, uma generalização o conceito *número real* é uma generalização de quase todas as strings de texto[i], na listagem anterior. Na próxima secção é mostrada a conveniência em trabalhar com generalizações, tentando induzi-as a partir dos dados de treino.

Outra dificuldade, também relacionada com os atributos posicionais, é o facto desta restrição ser muito rígida. Por exemplo, para o quadro de regras da figura 4.10, anterior, a identificação/extracção de texto[i] será realizada se a palavra que ocorre na posição anterior for "preço" ou "valor", Repare-se como isto é restritivo, pois mesmo que uma dessas palavras ocorra na posição i-2 ou i-3, a extracção não seria realizada, por exemplo: "*pelo valor de 24.000 euros*". Neste caso "*valor*" ocorre em texto[i-2]. Conclui-se aqui que é necessário uma outra forma de interpretar os atributos, de tal modo que no exemplo anterior a extracção fosse possível. A abordagem descrita na próxima secção, resolve este problema.

Uma outra fonte de dificuldades prende-se com o facto das instâncias negativas geradas, conterem um factor aleatório no processo da sua concepção. Embora seja uma solução possível, quando estamos na ausência deste tipo de instâncias, é sabido desde [Winston 1992] que há outras soluções mais eficazes. Assim é desejável que sejam encontradas instâncias negativas de uma forma mais adequada, para facilitar a indução dos conceitos em causa.

Pelas dificuldades e limitações enumeradas, evidencia-se a necessidade de uma nova abordagem, que será apresentada na próxima secção.

4.4 Extracção com Aprendizagem e Generalização

Da experiência realizada e descrita no final da secção anterior percebemos que a abordagem anterior não satisfaz a resolução do nosso problema inicial. Aquela abordagem contém um conjunto de limitações e dificuldades que são fonte de geração de problemas e baixo desempenho, estando as principais apresentadas no final da secção. Nesta abordagem é apresentada a realização do esforço de transposição daquelas dificuldades. O núcleo da maior parte do trabalho realizado, no âmbito desta dissertação, encontra-se exposto na presente secção.

Um dos passos importantes realizados na secção anterior consistiu na automação no processo de definição das regras. Deixou de ser um utilizador externo que define as regras e estas passaram a ser aprendidas. Em quase tudo no mundo, uma maior autonomia implica uma maior "responsabilidade", um melhor "saber fazer", e portanto o nosso sistema tem de ser dotado desta "capacidade". Além disso há outras tarefas que temos de realizar, como é o caso da escolha dos atributos, um trabalho de pré-processamento muito importante para o processo de aprendizagem. Na abordagem anterior o sistema estava liberto desta tarefa e esta continuava a ser realizada e fornecida pelo utilizador.

Um dos melhoramentos que esta abordagem pretende introduzir consiste em levar o sistema à realização da escolha dos atributos, "libertando" o utilizador desta tarefa. Assim, para além do sistema ficar mais autónomo, fica também mais fácil de utilizar, pois neste caso a única coisa que será fornecida é um conjunto de dados de treino. Tal como foi referido na secção anterior, estes dados de treino consistem em pequenos anúncios relativos a um determinado assunto, os quais têm assinalados com marcas SGML, os elementos considerados relevantes. Como ilustração apresentam-se, na figura 4.11, três anúncios reais anotados – anúncios de treino. Mais exemplos destes poderão ser consultados no *Anexo A*. Estes anúncios constituem a matéria prima do nosso sistema, a partir deles são geradas instâncias de treino com o objectivo de realizar a indução de regras de extracção dos elementos relevantes.

Anúncio nº1

```
<text>
   De:My Self0432035901@netcabo.pt
   Assunto:<tipo>T5</tipo> Duplex em <local>Gaia</local>
  Data:2002-05-10 15:01:24 PST

Excelente localização no centro da cidade. 2 WC, despensa,
terraço com marquise com 70 m2.
<preco>119700</preco> euros
966969663
</text>
```

Anúncio nº2

```
<text>
Apartamento pouco usada <tipo>T4</tipo>, 2 wc´s, 3º andar com
vista panoramica.
Excelente localização, a poucos metros da zona central de
<local>Loulé</local>.
Perto metros do tribunal, biblioteca, piscinas, e diversos
estabelecimentos comerciais.

Preço: <preco>132.180</preco>Euros
(negociavel)
936109097Envie uma continuação para esta mensagem
</text>
```

Anúncio nº3

```
<text>
    De: Macedo & Rapaz, Lda.macedo-rapaz@clix.pt
   Assunto:apartamento em <local>alvalade</local> urgente
  Data:2002-06-21 11:06:26 PST

  vende-se apartamento de <tipo>3 assoalhadas</tipo> com placa
em predio remodelado local calmo a precisar de algumas obras
refª 100640
valor <preco>121.500</preco> euros
tel 217951415 ou 963318686
cristina macedo
</text>
```

Figura 4.11 - Exemplos de alguns textos (anúncios) anotados

Recapitulando, o nosso objectivo consiste em extrair determinados elementos de documentos de texto. Documentos esses sem qualquer estrutura previamente definida. Estes consistem simplesmente em partes de texto (uma ou mais sequências de palavras) que serão identificadas pelo sistema, extraídas do texto e inseridas numa estrutura bem definida, denominada de tabela. No nosso domínio são considerados três elementos relevantes: o preço (*preço*), a tipologia da habitação (*tipo*) e a sua localização (*local*). Para qualquer elemento relevante (*elemento*), a marcação desse elemento, no texto dos documentos de treino, é efectuada utilizando as marcas <elemento> e </elemento>, identificando, respectivamente, o inicio e fim da ocorrência desse elemento relevante.

Em suma o nosso objectivo consiste na produção de informação estruturada a partir de informação não estruturada (ex: o texto de um anúncio), como referido no inicio da secção 4.3.1. O mundo do nosso sistema/agente consiste numa vasta colecção de documentos disponíveis e de um conjunto de tabelas que têm de ser preenchidas. O sistema tem a missão de identificar os elementos relevantes nos documentos extraindo-os e inserindo-os nas tabelas.

Antes de prosseguir com a apresentação dos pormenores do funcionamento do sistema, irá ser apresentada uma descrição de topo ou de "*Alto Nível*", do trabalho desenvolvido. Irá ser apresentado o algoritmo geral de funcionamento do sistema desenvolvido. Esta descrição será apresentada pressupondo que certas dificuldades, referidas em secções anteriores (ex: escolha dos atributos, exemplos negativos, etc) já estão resolvidas. Esta descrição algorítmica não descreve o modo de funcionamento do sistema, do ponto de vista do utilizador, mas sim o procedimento interno que norteia a acção do sistema. O funcionamento do sistema, do ponto de vista do utilizador, é apresentado na secção 4.5.2.

4.4.1 O Algoritmo de Extracção

Recapitulando brevemente o nosso contexto (Problema/Objectivo): temos um conjunto de anúncios, relativos à venda de habitações, semelhantes aos mostrados na figura 4.11 e nos quais se pretende identificar certos elementos considerados relevantes, nesta família de anúncios, como o preço e a tipologia, para depois os extrair e preencher os respectivos

campos nas tabelas de uma base de dados ou outra entidade de informação estruturada. Pretende-se um sistema que implemente técnicas de aprendizagem automática e daqui se depreende imediatamente que este será constituído por duas grandes fases: a fase de treino e a fase de aplicação/teste. Na fase de treino queremos que o sistema induza um conjunto de regras que depois irá aplicar, na fase de teste, a fim de realizar a identificação/extracção das ocorrências dos elementos relevantes considerados.

Das abordagens descritas anteriormente, permanecem três grandes problemas e cuja solução implementada, será descrita mais tarde, nesta secção. Estes três problemas são:

1. A escolha dos atributos.
2. A necessidade de utilizar categorias ou generalização de conceitos.
3. Os exemplos negativos.

Para já, vamos partir do princípio que estes problemas já foram ultrapassados e então queremos saber como actua o sistema. Começamos por admitir que os atributos foram escolhidos e fixados e que as instâncias negativas são de igual modo bem escolhidas. Para já, supomos a não necessidade de trabalhar com generalizações (ponto 2 da lista anterior).

Vamos então apresentar o algoritmo mais geral que traduz o trabalho implementado e que permite uma solução do nosso problema. Este algoritmo é apresentado numa sequência "*Top-Down*" (de cima para baixo), portanto do geral para o particular – primeiro o esquema genérico e depois o detalhar de cada um dos pontos apresentados, até que se atinja um nível em que já não existe necessidade de continuar a especificar os conceitos apresentados.

Designe-se então a nossa colecção de n anúncios que irão ser utilizados para o treino do sistema, por $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$, o algoritmo geral de acção do sistema é apresentado a seguir:

1. `Obter(` $\mathcal{At}$ `)`
2. **`Para cada`** `elemento relevante:` *elemento*, **`Fazer`**
   {
3. $V_{instâncias}(elemento)$ **<--** `GeraInstâncias(` $\mathcal{At}$, *elemento*, $Atrbs_{esq}$, $Atrbs_{dir}$ `)`
4. $V_{regras}(elemento)$ **<--** `Treino(` $V_{instâncias}(elemento)$ `)`
   }
5. `Obter(` $\mathcal{A}$ `)`
6. $V_{extracções}(elemento)$ **<--** `Extrair(` $\mathcal{A}$, $V_{regras}(elemento)$ `)`

Algoritmo 4.2 - Algoritmo de Extracção

No algoritmo anterior, as entidades referenciadas por *V*, como $V_{instâncias}(elemento)$, consistem em vectores de cadeias de caracteres - "strings". Os quatro primeiros pontos dizem respeito à fase de treino e os últimos dois à fase de aplicação/teste. No ponto 1 e 5, a função "`Obter`" caracteriza a obtenção dos anúncios, a partir da fonte (ex: "*News Groups*"), para o nosso sistema, é portanto o processo de entrada de dados no sistema ("*Input*"). Repare-se que $\mathcal{A}t \neq \mathcal{A}$, relativamente aos pontos 1 e 5, isto significa que em 1 temos uma colecção de anúncios de treino e portanto devidamente anotados, enquanto que no ponto 5 temos uma colecção de novos exemplos, não anotados, e dos quais se irá extrair as ocorrências dos elementos relevantes. No nosso algoritmo, "`GeraInstâncias`" é uma pseudo-função que caracteriza o trabalho de geração de instâncias de treino, a partir dos anúncios de treino ($\mathcal{A}t$) e para um determinado *elemento*. Essas instâncias são geradas para o vector designado por $V_{instâncias}(elemento)$ que por sua vez será utilizado no processo de treino e na indução do conjunto de regras de extracção do elemento relevante em causa (*elemento*), trabalho esse que será concretizado na pseudo-função designada por " `Treino`", no ponto 4. Por sua vez, esta função gera um outro vector constituído pelo conjunto de regras induzidas, designado por $V_{regras}(elemento)$. Isto completa o ciclo de treino do sistema que como se pode ler na linha 2 do algoritmo, é realizado para cada elemento relevante, do conjunto dos elementos definidos previamente.

A fase de aplicação/teste está expressa nos pontos 5 e 6 e é caracterizada pela pseudo-função de nome "`Extrair`", que, como se pode ver, depende dum conjunto de anúncios novos ($\mathcal{A}$) e do conjunto das regras geradas anteriormente, na fase de treino: $V_{regras}(elemento)$. Esta função produz como resultado uma colecção de instâncias extraídas ($V_{extracções}(elemento)$), relativas a um determinado *elemento*. A seguir são detalhadas as funções apresentadas no nosso algoritmo, nas linhas 3, 4 e 5, porém antes de avançar para isso será necessário introduzir alguma notação para que melhor se possa referir determinados aspectos e características do nosso problema.

Considere-se um qualquer anúncio *A* de uma colecção de anúncios anotados, portanto de treino. O texto contido no anúncio *A*, pode ser interpretado como uma sequência de palavras, à semelhança dum vector, isto é, podemos interpretar *A* como sendo um vector de palavras. Dum modo geral, designe-se esse vector por "**`Texto`**". Supondo que *A* contém N palavras, então a i-ésima palavra que ocorre em *A*, pode ser identificada por "**`Texto[i]`**" ($0 \leq i \leq N$). Ainda nesta notação, a sequência de palavras que vai desde a posição i à posição j, em *A*, é referida como "**`Texto[i:j]`**" ($0 \leq i \leq j \leq N$). Quando $i = j$ usamos a forma mais curta "**`Texto[i]`**".

Um determinado elemento relevante, designe-se *elemento*, pode ocorrer uma ou mais vezes em *A*. Defina-se então a ocorrência de *elemento* em *A*, por: *Ocorrencia(elemento, A*), consistindo isto em:

Ocorrencia(elemento, A)**:** `<elemento>`**`Texto[i:j]`**`</elemento>`

Por exemplo, no ultimo anúncio da figura 4.11, temos uma ocorrência do elemento *preço*, portanto - *Ocorrencia(preco, A*)**:** "**`<preco>`**`24.000`**`</preco>`**". Tendo em conta a possibilidade de múltiplas ocorrências dum elemento relevante num único anúncio, designamos a k-ésima ocorrência de *elemento* em *A* por : $Ocorrencia_k(elemento, A)$.

O número de ocorrências dum elemento relevante (*elemento*) num anúncio (*A*), será expresso pela função: *NOcorrencias(elemento, A)*. Por exemplo em qualquer anúncio da figura 4.11, temos *NOcorrencias(preco, A)* = 1.

Continuando com o que foi definido, para uma qualquer ocorrência dum elemento relevante num anúncio (*Ocorrencia(elemento, A*)), podemos considerar uma certa vizinhança a esta ocorrência que expressa um certo contexto no qual esta sucede. Esta vizinhança ou contexto será formado por uma contexto esquerdo e direito. O primeiro é constituído por um certo número de palavras que precede *Ocorrencia(elemento, A*) e o segundo por um certo número da sequência de palavras que sucede a mesma ocorrência. Assim, tendo presente a notação definida, considere-se a seguinte sequência de palavras dum anúncio *A*:

`Texto[i-n:i-1]` *Ocorrencia(elemento, A*) **`Texto[j+1:j+m]`**

então define-se "contexto esquerdo" de *Ocorrencia(elemento, A*), com **n** palavras por: $Contexto_{esq}(elemento, A, n) =$ **`Texto[i-n:i-1]`** e de igual modo o "contexto direito" de *Ocorrencia(elemento, A*), com **m** palavras por: $Contexto_{dir}(elemento, A, m)$ = **`Texto[j+1:j+m]`**.

Seguindo esta linha de definição, $Contexto(elemento, A, n, m)$ designa a junção ou concatenação das duas sequências relativas ao contexto esquerdo e direito, respectivamente **`Texto[i-n, i-1]`** e **`Texto[j+1, j+m]`**. No segundo anúncio da figura 4.11, temos:

$Contexto_{esq}(preço, A, 1) =$ "`Preço`"
$Contexto_{dir}(preço, A, 1) =$ "`Euros`"
$Contexto(preço, A, 1, 1) =$ "`Preço Euros`"

Exemplo 4.1 - Exemplos de Contextos

Ainda devido à possibilidade da existência de múltiplas ocorrências dum elemento num anúncio, tem-se: sempre que houver necessidade de referir o contexto de $Ocorrencia_k(elemento, A)$ (k-ésima ocorrência, de *elemento* em *A*) utiliza-se o índice k como se segue: $Contexto^k_{esq}(elemento, A, n)$, $Contexto^k_{dir}(elemento, A, m)$ e $Contexto_k(elemento, A, n, m)$, lendo-se, por exemplo para o contexto esquerdo: contexto esquerdo a *n* palavras da k-ésima ocorrência de *elemento* no anúncio *A*.

A partir desta notação é agora mais fácil descrever os algoritmos envolvidos nas pseudo-funções do algoritmo 4.2, apresentado anteriormente, que é o que será feito a seguir.

Geração de Instâncias (Função **"**`GeraInstâncias`**")**

Como foi referido, existe um quadro de pressupostos assumido aqui, um dos quais é que os atributos já estão escolhidos. Assuma-se que existem três conjuntos de atributos: $Atributos_{esq}$, , $Atributos_{dir}$ e A_{foco}. No primeiro e segundo conjunto estão incluídas uma colecção de palavras que podem ocorrer em $Contexto_{esq}(elemento, A, n)$ e $Contexto_{dir}(elemento, A, m)$, respectivamente. Portanto, estes atributos são palavras que ocorrem à esquerda ou à direita de *Ocorrencia(elemento, A*). Os domínios de cada um destes atributos encontra-se no conjunto binário {não, sim}, significando a não ocorrência ou ocorrência, respectivamente, da palavra em causa no contexto da ocorrência do elemento relevante, isto é $Contexto(elemento, A, n, m)$. Para o nosso problema concreto e para o elemento *preço,* um exemplo destes conjuntos poderá ser o mostrado de seguida:

$Atributos_{esq}$ = {preço, valor, venda, por, assunto}
$Atributos_{dir}$ = {euros}

Exemplo 4.2 - Atributos possíveis, para os contextos relativos ao elemento *preço*.

O terceiro conjunto de atributos, A_{foco}, contém um único atributo que diz respeito ao elemento relevante em si, isto é, considerando *Ocorrencia(elemento,A),* consiste na string em **`Texto[i:j]`**. O domínio deste atributo, compreende os valores possíveis da palavra (ou palavras), contidas em *Ocorrencia(elemento,A)*. Por exemplo para o nosso caso do elemento preço, poderíamos ter: Domínio(A_{foco}) = {132.180, 24000, 24.000, ..., etc}, repare-se nos exemplos da figura 4.11 anterior.

Com estes pressupostos assumidos, estamos agora em condições de apresentar o algoritmo envolvido na geração das instâncias de treino a partir do conjunto dos anúncios.

GeraInstâncias ($\mathcal{A}t$, *elemento*, $Atributos_{esq}$, $Atributos_{esq}$ **)**
devolve Vector de Strings
{

```
vector <-- ø
para cada A ∈ At fazer
    para cada k <-- 1 até NOcorrencias(elemento, A) fazer
    {
        sinstancia <-- ø

        para Ew_j ∈ Atributos_esq fazer
            se Ew_j ∈ Contexto^k_esq(elemento, A, n) então
                sinstancia <-- sinstancia + ",sim"
            senao
                sinstancia <-- sinstancia + ",não"

        para Dw_j ∈ Atributos_dir fazer
            se Dw_j ∈ Contexto^k_dir(elemento, A, m) então
                sinstancia <-- sinstancia + ",sim"
            senao
                sinstancia <-- sinstancia + ",não"

        sinstancia <-- sinstancia + A_foco + ",sim"
        sinst_nega <-- GeraInstNeg(sinstancia, elemento)

        vector <-- vector ∪ {sinstancia}
        vector <-- vector ∪ {sinst_nega}
    }

devolver vector
```

}

Algoritmo 4.3 - Geração de Instâncias

No algoritmo anterior é construído incrementalmente um vector de instâncias de treino. Em termos concretos, este pode ser interpretado como um vector de strings, sendo cada string uma instância positiva ou negativa. Inicialmente o vector está vazio e, como se pode verificar, para cada *Ocorrencia(elemento, A)* é gerada uma instância positiva e uma negativa, que depende da instância positiva gerada e é obtida através da função "`GeraInstNeg`". A forma como uma instância negativa é gerada será explicada mais adiante, nesta secção (subsecção 4.4.4). Nos dois ciclos mais internos do algoritmo, pode-se verificar que uma instância é uma string constituída por sequências de ",sim" ou ",não", consoante as palavras atributo de $Atributos_{esq}$ e $Atributos_{dir}$ estejam ou não nos respectivos contextos considerados. O símbolo "+" ali utilizado significa a operação de concatenação de strings. Os dois ultimos elementos a serem acrescentados à string de instância (`sinstancia`) são o valor do atributo

de foco (A_{foco}) e a classe da instância, que para uma instância positiva é, obviamente, sempre "sim". Portanto em cada anúncio A e, dentro deste, para cada ocorrência do elemento relevante, são geradas duas instâncias (positiva e negativa), e acrescentadas ao vector que será devolvido como resultado final da função.

Tendo presente os três exemplos mostrados na figura 4.11 e os atributos do exemplo 4.2, considerando ainda o elemento *preço*, apresenta-se de seguida as três instâncias positivas obtidas a partir das três ocorrências de *preço* naqueles três anúncios:

	$E_{preço}$	E_{valor}	E_{venda}	E_{por}	$E_{assunto}$	D_{euros}	A_{foco}	***Classe***
Instância 1	não	não	não	não	não	sim	119700	sim
Instância 2	sim	não	não	não	não	sim	132.180	sim
Instância 3	não	sim	não	não	não	sim	121.500	sim

Exemplo 4.3 - Exemplo de instâncias positivas.

Geração de Regras de Extracção (Função **"**`Treino`**")**

A função de treino apresentada no algoritmo 4.2 é a responsável por gerar um conjunto de regras de extracção ($V_{regras}(elemento)$), para um elemento relevante, que serão depois utilizadas no processo de extracção, representado pela função "`Extrair`", no mesmo algoritmo.

Neste trabalho foi utilizado um sistema gerador de árvores de decisão do tipo C4.5 [Quinlan 1993] na sua versão mais recente – C5 [Quinlan 1997][1]. A par deste gerador, foi também experimentado outro gerador de árvores de decisão, denominado de "*J48*" e pertencente à package de Data Mining com o nome WEKA [Witten & Frank 2000].

Conhecendo as especificações de entrada (input) do C5 e observando as instâncias do exemplo anterior (4.3), concluímos que estas se encontram no formato correcto e necessário para serem submetidas e consideradas como instâncias de treino pelo sistema C5. Portanto

1 Ver secção 3.5.3

essas instâncias ($V_{instâncias}(elemento)$), geradas pela função descrita anteriormente, são produzidas a pensar no sistema de treino utilizado.

Como é sabido, o sistema C5 pode produzir como resultado uma lista de regras, relativas à árvore de decisão induzida. Assim esta lista de regras é aquela que é devolvida pela função "`Treino`" do algoritmo 4.2, através dum vector de strings ($V_{regras}(elemento)$). Em suma o processo de treino consiste em gerar regras, que determinam as circunstâncias nas quais se realiza uma extracção do elemento relevante em causa.

Dois exemplos de regras contidas em $V_{regras}(preço)$, obtidas como resultado do treino são:

$V_{regras}[0]$ = "`ESQpreço = sim AND DIReuros = sim --> sim`"
$V_{regras}[1]$ = "`DIRcontos = sim --> sim`"

Exemplo 4.4 - Dois elementos do vector de regras gerado

Na secção 4.4.5 serão mostradas as regras induzidas pelo sistema C5, para os três elementos relevantes considerados (*preço, tipo, local*) e será também melhor explicada a forma como estas são utilizadas no processo de extracção.

Extracção de Informação (Função **"**`Extrair`**")**

Esta ultima função, presente no algoritmo 4.2, caracteriza a identificação e extracção das ocorrências dum elemento relevante nos novos anúncios (de teste: $\mathcal{A}$), com base nas regras induzidas anteriormente (vector $V_{regras}(elemento)$). O processo de extracção e consequente aplicação das regras induzidas será aprofundado na secção 4.4.5, pois existe um conjunto de problemas específicos, que é preciso abordar e que são importantes nesse processo. Recordamos que esta abordagem geral, partiu do princípio que estes e outros problemas já estavam resolvidos. Portanto, falta detalhar a resolução dos mesmos que será feito nas secções que se seguirão a esta.

Assim a função "`Extrair`" é apresentada também sob a forma dum pequeno algoritmo, mostrado a seguir:

```
Extrair (Anúncios, V_regras) devolve Vector de Strings
{
      vector <-- ∅
      para cada A ∈ Anúncios fazer
      {
            Texto <-- string do texto de A
            para cada regra ∈ V_regras fazer
            {
                  se Satisfaz(regra, " contexto_esq Texto[i:j] contexto_dir ") então
                        vector <-- vector ∪ {Texto[i:j]}
            }
      }
      devolver vector
}
```

Algoritmo 4.4 - Extracção de Informação

Neste algoritmo a função "*Satisfaz*" testa se uma regra de extracção se verifica num segmento do texto (da palavra `i` à palavra `j`, conforme notação introduzida no inicio desta subsecção) dum anúncio. Esta verificação tem em conta os contextos esquerdos e direitos, para além da porção de texto considerado (`Texto[i:j]`). Supondo que tínhamos a seguinte porção de texto, contida no dado anúncio: "(...) *pelo preço de 100 000 euros, negociável e* (...)" e que o tamanho dos contextos esquerdo e direito considerados era de três palavras, então a regra $V_{regras}[0]$ do exemplo 4.4 seria verificada positivamente, através da função "*Satisfaz*", quando fosse considerado `Texto[i:j]=` "*100 000*" (neste caso `j=i+1`), $contexto_{esq}$ = "*pelo preço de*" e $contexto_{dir}$ = "*euros negociável e*".

Todas as ocorrências de elementos relevantes nos anúncios, determinadas pelo vector de regras, vão sendo reunidas icrementalmente num vector. Este vector é depois devolvido como resultado.

4.4.2 Pré-processamento: Escolha dos Atributos

Como foi referido, sabe-se que os atributos considerados exercem uma força preponderante em qualquer processo de indução. É necessário escolher bons atributos, dentre as possibilidades. Assim, em qualquer domínio existe um espaço dentro do qual é promissor considerar a escolha de atributos.

Escolha do Contexto

Tendo presente a notação introduzida na subsecção anterior, consideramos natural para espaço de procura dos atributos o texto contido numa certa vizinhança de *Ocorrencia(elemento,A)*, para um determinado elemento relevante (*elemento*), em que *A* é um dos anúncios de treino ($\mathcal{A}_t$). Essa vizinhança foi definida anteriormente por *Contexto(elemento,m,n*), ou ainda de uma outra forma: $Contexto_{esq}(elemento, A, m)$ e $Contexto_{dir}(elemento, A, n)$. Tenhamos também presente que *m* e *n* indicam, respectivamente, o número de palavras consideradas para o contexto esquerdo e direito.

Os valores de *m* e *n* podem ser fixados com um valor elevado, todavia por razões óbvias de simplificação e por se saber que a partir duma certa distância à ocorrência do elemento relevante, o texto deixa de estar relacionado com a ocorrência daquele, é mais natural considerarem-se valores mais pequenos. De outra forma, a probabilidade de texto afastado de *Ocorrencia(elemento,A)* ter alguma relação com este, tende para zero, à medida que nos afastamos daquela ocorrência. A partir de uma certa distância de *Ocorrencia(elemento,A)* o texto ocorrido é independente de *Ocorrencia(elemento,A)*. Ilustrando no nosso domínio, significa que podemos ter *Ocorrencia(preço,A)* em Texto[i:j] e Texto[i-10:i-4] ter informação relativa, por exemplo, à *garagem da habitação*. Mas isto não significa que esta co-ocorrência de conceitos seja sempre verificada, ou na maioria dos casos. Não significa que sempre que se tenha uma ocorrência do elemento relevante *preço*, entre as posições i e j do texto, se esteja a falar de garagens entre as posições i-10 e i-4.

Assim o mais natural será considerar para *m* e *n* um determinado numero de palavras não muito grande, por exemplo 3 ou 4. Estes limites, que definem as dimensões dos contextos, poderão ser deixados para ajustar por um utilizador do sistema.

Representação de Contexto

Coloca-se então a questão de saber como escolher os atributos em *Contexto(elemento,m,n)*, antes de mais como considerar os atributos? Será conveniente utilizar o método seguido nas duas abordagens anteriores (preliminares – 4.3.1 e 4.3.2), considerando as posições das palavras vizinhas[1]? A resposta é negativa, tal como se verificou nos algoritmos apresentados anteriormente, pois achou-se mais conveniente considerar as palavras em si, como sendo os próprios atributos, em vez da posição de ocorrência destas em relação a *Ocorrencia(elemento,A*). Verificámos também que esta alternativa produz melhores resultados. No terceiro anúncio mostrado na figura 4.11, temos na décima linha uma *Ocorrencia(preço,A)*, considerando $Contexto_{esq}(preço, A, m)$ e *m=4*, têm-se para possíveis candidatos a atributos as palavras: "*valor*", "*100640*", "*ref*", e "*obras*". Neste método o domínio dos atributos é amplamente simplificado, ficando reduzido a um conjunto de dois valores {não, sim}, em que "não" significa a não ocorrência do atributo (palavra) e "sim" a sua ocorrência na vizinhança considerada.

A partir dos anúncios de $\mathcal{A}t$, podemos definir o vector esquerdo (**VSe**) e o vector direito (**VSd**), como sendo os vectores contendo as strings dos contextos equerdo e direito, respectivamente, no conjunto de anúncios anotados ($\mathcal{A}t$), de todas as ocorrências dum elemento relevante. No exemplo 4.5 são mostrados partes destes vectores, para o nosso domínio específico e relativamente ao elemento *preço*. Estes vectores foram construídos a partir de dados reais, repare-se que esta pequena lista contém as três ocorrências de preço, nos anúncios da figura 4.11 – são as três primeiras linhas.

1 palavra anterior, palavra seguinte, as duas palavras anteriores à ocorrência, etc

Algumas ocorrências de *preço*:

```
com 70 m2 <preco>119700</preco> euros 966969663
estabelecimento comerciais preço <preco>132.180</preco> euros negociavel
refª 100640 valor <preco>121.500</preco> euros tel 2179511415
no 2º andar <preco>109.735,54</preco> euros contacto
pelo preço de <preco>32 mil</preco> contos TM 938464173
muita luz preço <preco>77.313</preco> euros tm 914574849
do sapo seixal <preco>45.000</preco> contos mais informações
```

Vectores correspondentes:

VSe – Vector esquerdo	***VSx***	***VSd – Vector direito***
com 70 m2	119700	euros 966969663
estabelecimentos comerciais preço	132.180	euros negociavel
refª 100640 valor	121.500	euros tel 2179511415
no 2º andar	109.735,54	euros contacto
pelo preço de	32 mil	contos tm 938464173
muita luz preço	77.313	euros tm 914574849
do sapo seixal	45.000	contos mais informações

Exemplo 4.5 – Vectores de ocorrências e contextos

O vector central **VSx** designa o vector das ocorrências do elemento relevante (*preço*, neste caso). Os atributos serão seleccionados do conjunto de palavras contidas nos vectores esquerdo (**VSe**) e direito (**VSd**). Além disso, precisamos de mais alguns atributos, escolhidos de entre as palavras referentes aos exemplos negativos gerados. Isto será referido aqui, mais tarde.

A partir de um conjunto de palavras candidatadas a atributos, surge a questão de saber como escolher, desse conjunto, aquelas que são úteis no passo seguinte, isto é, na classificação. Existem algumas medidas que fazem sentido utilizar aqui, algumas amplamente empregues nos sistemas com aprendizagem automática, como é o caso do *ganho de informação*. Este tópico é abordado a seguir.

Ganho de Informação na Escolha dos Atributos

Tendo presente a Teoria da Informação de Shannon e as adaptações subsequentes na área da *Aprendizagem Automática* [Mitchell 1997], sabemos que o *ganho de informação* de um atributo **A**, numa colecção de **S** instâncias, é dado por:

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} * Entropy(S_v)$$

em que $|S|$ é o numero de instâncias da colecção e $|S_v|$ o numero de instâncias de *S* para as quais *A* têm o valor *v* (*A* = *v*). A função "Entropy" é calculada para uma colecção de instâncias *S*, com *c* classes pela bem conhecida formula:

$$Entropy(S) = \sum_{i=1}^{c} p_i \log_2(p_i)$$

em que para todo o i p_i é a probabilidade da classe *i*.

Uma primeira dificuldade aqui está relacionada com o facto de não estarmos a utilizar a função $Gain(S, A)$ para avaliar um conjunto de atributos de instâncias *S*, já definidas[1], mas a utilizar a referida função para escolher os atributos que irão constituir as futuras instâncias. Tendo presente que os nossos exemplos de treino são blocos de texto, é necessário identificar instâncias, nos mesmos exemplos, uma vez que a função $Gain(S, A)$ depende de números de instâncias: $|S|$, $|S_v|$. Como olhar para a totalidade do texto dos anúncios e identificar o que são as instâncias das várias classes?

Em termos de classes, sabemos que só temos duas, como tem sido referido – "extrair" e "não extrair". Estas são designadas de *sim* e *não*, respectivamente, de modo a tornar a notação mais expedita. Portanto temos *values(A) = {sim, não}*. A estimativa de $|S|$ e $|S_v|$ está relacionada com a estimativa de *p(sim)* e *p(não)*, uma vez que: $p(\text{sim}) = \frac{|S_{\text{sim}}|}{|S|}$ e *p(não)* é igual a *1.0 – p(sim)*, por um lado, e $p(não) = \frac{|S_{não}|}{|S|}$ por outro. O valor de $|S_{\text{sim}}|$ é

1 Que é feito, por exemplo, no processo de construção duma árvore de decisão, através do ID3.

conhecido, pois consiste no número de instâncias anotadas nos exemplos, para o elemento relevante considerado. Consequentemente têm-se $|S_{não}| = |S| - |S_{\text{sim}}|$ e assim o problema resume-se a estimar o valor de *p(sim)*, o que será realizado a seguir.

No nosso caso, queremos determinar atributos (palavras) informativos, relativamente aos contextos esquerdo e direito, das ocorrências dos elementos relevantes. Esta determinação será realizada em separado para cada um dos contextos, dado que uma palavra informativa para o contexto esquerdo, pode não o ser para o direito e vice-versa. Será explicado aqui como é feito o cálculo dos atributos mais informativos, para o contexto esquerdo, sendo semelhante o realizado para o contexto direito.

Assim e relativamente ao contexto esquerdo, convencionou-se que uma instância positiva (de extracção - *sim*), para um *elemento*, seria constituída por $Contexto_{esq}(elemento, A, n)$ mais o texto contido na ocorrência do elemento relevante – *Ocorrencia(elemento,A*). Designe-se este segmento de texto por $Instancia_{esq}(elemento)$. No exemplo 4.5 e para a primeira linha temos $Instancia_{esq}(elemento)$ = "com 70 m2 119700".

Podemos pensar no número de palavras contidas em $Instancia_{esq}(elemento)$, designe-se esse valor por **#palavras{** $Instancia_{esq}(elemento)$ **}**. Numa colecção de anúncios anotados, com **r** ocorrências dum determinado elemento relevante (*elemento*), o número designado por **#palavras{***elemento***}** representa o número de palavras envolvidas em todas as instâncias de *elemento*. Assim **#palavras{***elemento***}** = $\sum_{i=1}^{r} \#\ palavras\{ Instancia_{esq}^{i}(elemento)\}$.

Nas 7 ocorrências mostradas no exemplo 4.5 temos **#palavras{***elemento***} =** 29. Desta forma, estima-se $p(\text{sim}) \approx \frac{\#\ palavras\{\ elemento\ \}}{\#\ palavras\{\ Anúncios\ \}}$, em que **#palavras{***Anúncios***}** é o número total de palavras envolvidas nos anúncios de treino. Uma vez que $|S_{\text{sim}}|$ = 7 , no exemplo 4.5, e admitindo que os anúncios de treino que contém exclusivamente aquelas ocorrências relativas

a *preço*, têm **#palavras{***Anúncios***} =** 290, então: *p(sim)* = 0.1, *p(não)* = 0.9, $|S| \approx \frac{7}{0.1} = 70$ e $|S_{não}| \approx 70 - 7 = 63$.

Assim no nosso caso concreto temos:

$$Entropy(S) = -p(\text{sim}) * \log_2 p(\text{sim}) - p(não) * \log_2 p(não)$$

e em continuação ao exemplo iniciado, tem-se $Entropy(S) = 0.469$.

Neste contexto a função de ganho de informação pode ser designada com os parâmetros: S e w, em vez de S e A, isto é: *Gain(S,w)*, pois os nossos atributos são palavras. Assim para calcular *Gain(S,w)* temos de calcular $|S_w|$ e $|S_{\neg w}|$, significando $\neg w$ a não ocorrência da palavra w. Temos também de calcular $Entropy(S_w)$ e $Entropy(S_{\neg w})$. Estes dois últimos valores são calculados a partir das probabilidades condicionais: *p(sim|w)*, *p(não|w)*, *p(sim|¬w)*, *p(não|¬w)*, onde por exemplo *p(sim|w)* significa a probabilidade de extrair um elemento relevante, tendo em conta que a palavra *w* ocorre na vizinhança considerada deste.

$$Entropy(S_w) = -p(\text{sim}\,|\,w) * \log_2 p(\text{sim}\,|\,w) - p(não\,|\,w) * \log_2 p(não\,|\,w)$$

$$Entropy(S_{\neg w}) = -p(\text{sim}\,|\,\neg w) * \log_2 p(\text{sim}\,|\,\neg w) - p(não\,|\,\neg w) * \log_2 p(não\,|\,\neg w)$$

Tendo presente que *p(não|w) = 1.0 - p(sim|w)* e que *p(não|¬w) = 1.0 - p(sim|¬w)*, o problema resume-se a estimar as probabilidades e *p(sim|w)* e *p(sim|¬w)*. Para estimar *p(sim|w)* teremos de contar quantas instâncias de extracção (S_{sim}) contém a palavra *w* e dividir pelo número total de instâncias que contém a mesma palavra. Assim podemos considerar $TF(w)$ [1] e $TF(w|S_{\text{sim}})$, isto é, a frequência de *w* no texto dos anúncios e a mesma frequência para as instâncias de extracção consideradas (todas as $Instancia_{esq}(elemento)$), respectivamente. Com isto estima-se $p(\text{sim}\,|\,w) \approx \frac{min\{TF(w|S_{\text{sim}}),\ |S_{\text{sim}}|\}}{TF(w)}$ e de uma

1 *TF – Term Frequency*.

forma similar $p(\text{sim}\,|\,\neg w) \approx \frac{max\{\,|S_{\text{sim}}| - TF(w|S_{\text{sim}}),\ 0\}}{|S| - TF(w)}$. Seguindo o mesmo raciocínio estima-se $|S_w| \approx TF(w)$ e $|S_{\neg w}| \approx |S| - |S_w|$.

No exemplo que temos construído, vamos supor que se queria calcular *Gain*(*S*, preço[2]), em relação ao exemplo 4.5. Supõe-se também que *TF*(preço) = *4*, então têm-se o seguinte:

$$p(\text{sim}\,|\,\text{preço}) = \frac{min\{3,\ 7\}}{4} = 0.75 \quad \text{e} \quad p(\text{sim}\,|\,\neg\text{preço}) = \frac{max\{7-3,\ 0\}}{70-4} = \frac{4}{66} \approx 0.06061\,,$$

consequentemente têm-se *p*(não|preço) = *0.25* e *p(não|*¬preço) = *0.939*. Finalmente obtêm-se: $Entropy(S_{preço}) = 0.8113$, $Entropy(S_{\neg preço}) = 0.33$ e

$$Gain(S, \text{preço}) = 0.469 - \frac{|S_{preço}|}{|S|} * 0.8113 - \frac{|S_{\neg preço}|}{|S|} * 0.33$$

$$\Leftrightarrow$$

$$Gain(S, \text{preço}) = 0.469 - \frac{4}{70} * 0.8113 - \frac{66}{70} * 0.33 = 0.111$$

Este processo permite medir quais as palavras mais significativas, isto é, mais informativas que ocorrem na vizinhança dum elemento a extrair.

O ganho de informação é uma entre várias medidas possíveis a utilizar, para escolher alguns atributos do dado conjunto de possibilidades. Portanto pode fazer-se uma ordenação e escolher os primeiros atributos mais informativas. A tabela apresenta um exemplo de uma ordenação deste género:

Expressão	***Ganho Inf.***
preço	0.200
valor	0.170
venda	0.124
por	0.070
assunto	0.065
de	0.005
o	0.003
a	0.002

Tabela 4.1 - *Ganho de informação de palavras*

2 Calcular o ganho de informação para a palavra “preço”

Outras Medidas Possíveis na Escolha dos Atributos

Para além do ganho de informação existem outras medidas, algumas mais utilizadas na classificação de texto ("*Information Retrieval*") mas que poderão ser adaptadas para a extracção de elementos relevantes, como é o caso de "*Mutual Information Text*" e "*Odds Ratio*" [Mladenic & Grobel. 1999] e que de seguida se apresentam:

$$MutualInfoTxt(W) = \sum_i P(C_i) \log \frac{P(W \mid C_i)}{P(W)}$$

$$OddsRatio(W) = \log \frac{P(W \mid \text{sim})(1 - P(W \mid não))}{(1 - P(W \mid \text{sim}))P(W \mid não)}$$

Foram exploradas estas três medidas, para a selecção de atributos, no nosso problema concreto. Os resultados comparativos serão referidos no capitulo seguinte.

Para terminar esta secção, refira-se que o atributo mais importante, a ter em conta, e que ainda não foi referido é precisamente a ocorrência do elemento relevante, isto é a expressão textual que ocorre em *Ocorrencia(elemento,A)*, mais precisamente texto[i:j]. Este é um atributo que caracteriza a natureza do objecto a extrair e certamente se reveste de uma grande importância. É o único atributo posicional que consideraremos e consequentemente o seu domínio não será um conjunto equivalente a {0,1}, à semelhança dos restantes atributos, mas tem em conta as expressões que constituem as ocorrências do elemento relevante, isto é, o vector VSx.

4.4.3 Generalizando a conceitos

Uma das dificuldades referidas no final da abordagem anterior prende-se com o domínio dos atributos. O problema consiste na possibilidade de haver uma grande variabilidade nos valores dos atributos posicionais[1], o que torna difícil a definição de regras prontas a responder às novas instâncias. Um exemplo disto é o que acontece quando os valores desses atributos são números reais (por exemplo como sucede no exemplo 4.5, anterior, mais concretamente com os valores no vector VSx, relativo ao elemento *preço*). O nosso

1 Exemplo: palavra anterior à ocorrência do elemento relevante.

objectivo será tentar generalizar os valores, do domínio do atributo, a conceitos mais gerais que cubram a totalidade ou pelo menos a maior parte desse valores. No exemplo referido, pela simples observação de VSx, constatamos imediatamente que existe de facto uma grande variabilidade (não existem duas expressões iguais). Todavia existe uma forma geral que é modelo da maioria das expressões: quase todas são números reais, à excepção da quarta linha, VSx[4] = "32 mil" (um número seguido da palavra "mil"), neste caso. Assim utilizando um conceito geral que represente um número real é possível ultrapassar este problema, neste caso concreto.

Denominem-se estes conceitos gerais, simplesmente por *conceitos*, assim podemos pensar no conceito de número real e representar esse conceito com uma notação apropriada. Assim se definirmos o conceito número como uma string que contém exclusivamente caracteres do conjunto **{**"0", ... , "8", "9", ",", ".", "+", "–" , "e", "E"**}**, podemos dizer que as seguintes strings são instâncias do conceito número: "7", "6.78", "-5.32", "7,5E+2" "100000.00", etc. Se notarmos o conceito número por *NUMERO*, constatamos que os elementos de VSx, no exemplo 4.5, se resumem a concretizações de dois modelos de strings:

[*NUMERO*] e [*NUMERO* "mil"]

Nestes modelos de strings os caracteres '[' e ']', marcam respectivamente o início e fim das strings. Assim "7 mil" e "13,5 mil" são instâncias do modelo [*NUMERO* "mil"], enquanto que a expressão "muitos mil" não é uma instância deste modelo, nem "os 90 mil". Esta última expressão contém uma instância do modelo anterior ("90 mil"), mas não é uma instância deste.

À semelhança do conceito *NUMERO* existem muitos outros conceitos que referenciam entidades do mundo real, sendo útil considerá-los no processamento de informação textual. Pretende-se que o sistema possa conhecer à priori esses conceitos, ou teve de alguma forma acesso a essa informação (ex: Internet) ou alguém (ex: utilizador) facultou a mesma. Eis alguns exemplos de conceitos: *número*, *data*, *hora*, *moeda*, *país*, *localidade*, *viatura*, etc. Estes são só alguns dos conceitos universais, que fazem parte do nosso mundo, os quais o nosso senso comum está habituado a utilizar, quase de forma automática. Este tipo de conhecimento generalizado é susceptível de ser formalizado. Mas como poderão estar formalmente definidos estes

conceitos? Repare-se que no parágrafo anterior *NUMERO* foi definido informalmente. Existem vários formalismos capazes de concretizar estas definições e neste trabalho irá ser utilizado o formalismo das "*Gramáticas Independentes de Contexto*" (Context-Free Grammars) [Jurafsky & Martin 2000]. Assim e para o exemplo de *NUMERO* poder-se-á ter uma gramática completa que define este conceito:

```
NUMERO -> [NUMERO" "NUMERO]
NUMERO -> [NUMERO",”NUMERO]
NUMERO -> [NUMERO“.”NUMERO]
NUMERO -> [NUMERO“E+”NUMERO]
NUMERO -> [NUMERO“E-”NUMERO]
NUMERO -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Exemplo 4.6 - Definição formal do conceito *NUMERO*

Reconhecimento de Local

Um dos elementos relevantes considerados neste trabalho é o elemento *local*, como já foi referido. Para este elemento o vector VSx que se obtém, contém nomes de localidades, ruas, cidades, etc. Tal como no exemplo dos valores numéricos, também aqui é importante que o sistema seja capaz de reconhecer que o nome de uma rua, ou uma cidade, são instâncias de um conceito mais geral que podemos designar por *LOCAL*. De seguida mostram-se três exemplos de instâncias do elemento relevante *local* e que correspondem aos da figura 4.11:

```
assunto t5 duplex em <local>gaia</local> data
zona central de <local>loulé</local> perto
Assunto apartamento em <local>alvalade</local> urgente
```

Repare-se que nas três linhas estão contidas três diferentes instâncias do conceito *LOCAL* ("gaia", "loulé, "alvalade").

No sistema desenvolvido foi utilizado um ficheiro contendo todas as localidades, incluindo nomes de ruas, praças, etc, de Portugal, como definição do conceito de localidade: *LOCAL*.

Este ficheiro encontra-se disponível na Web, no portal dos "*CTT - Correios de Portugal*" (`http://codigopostal.ctt.pt/pdcp-files/todos_cp.zip`) . Isto constitui um exemplo de como a Internet pode facultar conhecimento do mundo real, ao nosso sistema.

Assim seguindo o formalismo das gramáticas independentes de contexto, podemos definir o conceito *LOCAL* com uma lista de regras gramaticais, à semelhança do que foi feito para o conceito *NUMERO*. Uma parte duma lista desse género é mostrada em baixo:

```
LOCAL -> "Rua do" LOCAL
LOCAL -> "Avenida de" LOCAL
...
LOCAL -> "Porto" | "Lisboa" | "Coimbra" | ...
LOCAL -> "Rua do Campo Alegre"
...
```

Sintetizando, o nosso sistema trabalha com um conjunto de conceitos pré estabelecidos, que em termos práticos lhe foram fornecidos, e assume isso como conhecimento à priori. Não significa que o sistema utilize todos os conceitos conhecidos, mas sim os que "achar convenientes". Utilizando os diferentes conceitos conhecidos, o sistema tenta criar uma generalização dos elementos contidos no vector VSx. Uma generalização de VSx é um outro vector, denomine-se de GVSx, que conterá as possíveis generalizações dos elementos de VSx.

GVSx <-- Generalizar(VSx)

Assim considerando ainda o exemplo 4.5, temos para aquele vector VSx[1] o vector de generalização GVSx resultante e mostrado em baixo:

```
GVSx = <[NUMERO], [NUMERO "mil"]>
```

Exemplo 4.7 - Vector de generalização.

Pode suceder que não seja possível generalizar os elementos de VSx, então nesse caso ter-se-à GVSx = VSx, isto é, a generalização será o próprio vector VSx. Como é descrito a seguir,

1 No exemplo 4.5 tem-se: VSx = <119700, 132180, ..., 45000>

para cada conceito é calculado um valor que indica se é promissor considerar esse conceito e proceder à generalização.

O vector GVSx, obtido a partir de VSx irá constituir o domínio do atributo A_{foco}, referido desde a subsecção 4.4.1, e utilizado na geração das instâncias de treino (reparar no algoritmo 4.3). Para concluir a presente secção falta explicar o modo como são feitas as generalizações de VSx, isto é, o que é feito na pseudo função "*Generalizar*", mostrada anteriormente?

Função de Generalização

Apresenta-se o algoritmo utilizado no processo de generalização dum vector VSx, descrito anteriormente, relativo às ocorrências dum elemento relevante nos anúncios de treino (ex: *preço* no exemplo 4.5) :

```
Generalizar(v: vector de strings) devolve vector de strings
{
      Lista <-- ∅
      para cada conceito ∈ Conceitos fazer
      {
            valor <-- CalcularValor(conceito, v, Anúncios)
            Lista <-- Lista  ∪  {(conceito,valor)}
      }

      para cada (conceito,valor) ∈ Lista fazer
            se  valor > δ  então
                  v <-- SubstConceito(v, conceito)

      devolver v
}
```

Algoritmo 4.5 - Processo de generalização.

Para cada conceito conhecido é calculado o valor desse conceito no vector em causa, relativo a VSx, e atendendo também ao conjunto dos anúncios utilizados no treino e que permitiram

gerar esse vector, é o que faz a função "`CalcularValor`". O valor calculado deve ser tal que perspective a importância relativa do conceito em causa, no que diz respeito a uma eventual generalização das instâncias desse conceito que ocorrem no vector, isto é se vale ou não a pena generalizar no vector com o conceito em causa. No exemplo 4.5 o valor do conceito *`NUMERO`* deve ser tal que seja superior a por exemplo o conceito *`(local)`*. Assim uma boa candidata para a função "`CalcularValor`" será a função de "*Ganho de Informação*", conforme referido na subsecção anterior (4.4.2), ou outra função do género (*Odds Ratio*).

Portanto o valor dum conceito, variável "*`valor`*" do algoritmo anterior, traduz o valor informativo de ser considerado um determinado conceito no processo de generalização dos elementos que compõe o vector em causa. Como o algoritmo 4.5 mostra, é criada uma lista de pares ordenados "`(`*`conceito`*`,`*`valor`*`)`", estabelecendo assim a associação entre cada conceito e o seu respectivo valor. Por fim a generalização ou generalizações são concretizadas no vector, para cada conceito cujo valor seja superior a um certo limiar mínimo (δ). As eventuais generalizações aos conceitos serão concretizadas através da função "`SubstConceito`". No exemplo 4.5 a generalização da linha 5 ("32 mil") de VSx ao conceito *`NUMERO`*, resultaria na string "*`numero`* mil" e o vector GVSx resultante seria o mostrado anteriormente no exemplo 4.7.

4.4.4 Geração de Instâncias Negativas

Num processo de indução de um classificador, como é o caso de uma árvore de decisão, são necessárias instâncias de treino relativas a cada uma das classes existentes. Existem domínios em que é muito difícil obter instâncias de treino para uma determinada classe ou classes. O ideal é que para cada classe o numero de instâncias de treino seja, em numero, um valor semelhante às outras classes. O nosso domínio concreto, contém duas classes ("sim" e "não"), queremos saber se extraiamos ou não um determinada expressão textual. Como temos um problema de classificação com duas classes, convém que tenhamos instâncias positivas ("sim") e instâncias negativas ("não") de treino. No nosso caso, a dificuldade centra-se em encontrar instâncias negativas.

Na abordagem com aprendizagem mostrada inicialmente (subsecção 4.3.2), essa dificuldade foi contornada gerando instâncias negativas de forma aleatória. A procura era feita no texto não contido nas ocorrências dos elementos relevantes. No final da secção anterior, este método de geração de instâncias negativas foi criticado e referido como um problema e impedimento para que o sistema alcance uma bom desempenho. A principal razão prende-se com a possibilidade das instâncias geradas desta forma serem pouco informativas, caracterizando assim mal as instâncias desta classe (não extrair: *não*).

Método Aleatório

A região de texto utilizada para a geração de instâncias negativas continuará a ser o texto do anúncio no qual não ocorre o elemento relevante para o qual se quer gerar essas instâncias. Considerando a notação utilizada na subsecção anterior, sendo $\mathcal{A}_t$ **= {A1, A2,..., Am}** uma colecção de **m** anúncios de treino (anotados), defina-se **T = A1 + A2 + ... + Am**[1], como a totalidade do texto que compõe os exemplos de treino. Então as instâncias negativas serão elaboradas tendo por base o texto contido em **T** subtraído do texto relativo a todas as ocorrências do elemento relevante considerado, isto é: **T \ (VSe + VSx + VSd)**. Temos de retirar este texto para não haver a possibilidade de ser escolhida aleatoriamente uma instância negativa que sabemos que é positiva.

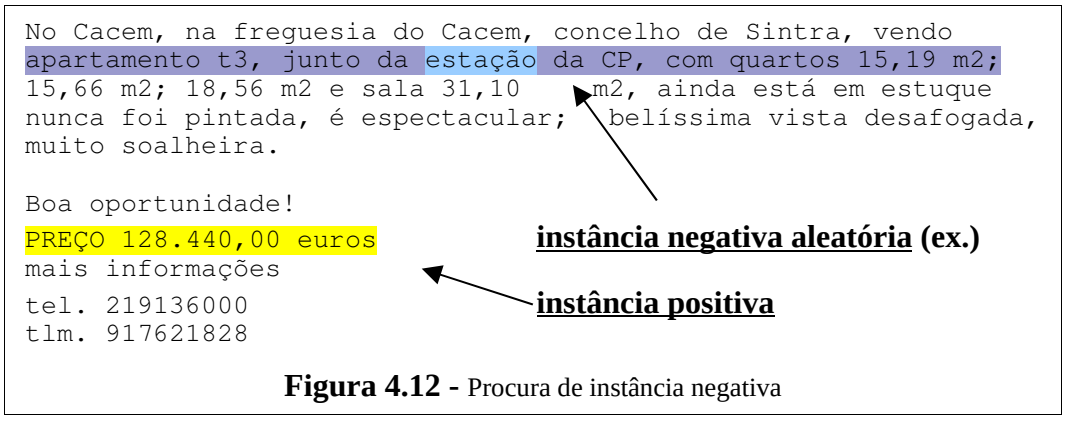


Figura 4.12 - Procura de instância negativa

1 Aqui o símbolo "**+**" representa a concatenação do texto dos anúncios.

A título ilustrativo, na figura anterior, o texto marcado e mais claro faz parte de **VSe + VSx + VSd**, para o elemento *preço*. Todo o resto do texto constitui uma parte[1] de **T \ (VSe + VSx + VSd)**. Assim, para gerar uma instância negativa de forma aleatória, têm-se em conta um exemplo positivo gerado anteriormente, nomeadamente quantas palavras constituem **Texto[i:j]** de *Ocorrencia(elemento,A)* e formam o valor do atributo de A_{foco}, para essa instância positiva. No exemplo de *Ocorrencia(preço,A)*, na figura 4.12, temos **Texto[i:j] =** "`128.440,00`", por isso a instância negativa a gerar toma também uma só palavra para o valor de A_{foco}, escolhendo aleatoriamente uma porção do texto em **T \ (VSe + VSx + VSd)**, tal como é ilustrado na figura 4.12, pela marcação mais escura. Os contextos esquerdo e direito são tomados de igual modo e tamanho, relativamente aos das instâncias positivas. Para a instância negativa aleatória, da figura 4.12, poderia ser escolhido A_{foco} = "`estação`" e os contextos esquerdo e direito a 3 palavras seriam "`t3, junto da`" e "`da CP, com`", respectivamente.

Uma experiência realizada com as primeiras versões do trabalho implementado no âmbito desta dissertação, mostraram que o desempenho do sistema era um pouco deficiente, concluindo-se que uma das razões que estavam na origem deste problema era o facto das instâncias negativas serem exclusivamente geradas pela forma descrita (aleatoriamente). Por exemplo par o elemento *preço*, uma das regras induzidas foi a seguinte:

se A_{foco} = *(numero)* **então** `sim`

Repare-se que esta regra é demasiado geral, pois considerando o anúncio da figura 4.12, para além do preço seriam extraídos também o número de telefone, o número de telemóvel, bem como os valores das várias áreas, ali especificados, como 15,19; etc. Portanto temos um desempenho muito baixo devido essencialmente a um valor de *precision* muito degradado - só neste exemplo, de sete elementos extraídos, só um é que nos interessa.

Estratégia "*near miss*"

Uma outra forma de geração de instâncias negativas foi pensada, para este nosso problema, seguindo a estratégia de "*near miss*" definida por [Winston 1992]. Esse processo consiste em

1 É só uma parte, pois só estamos a considerar este anúncio.

procurar instâncias negativas tão próximas quanto possível das instâncias positivas, entre o texto de **`T \ (VSe + VSx + VSd)`**. Tendo em conta a importância prática do atributo A_{foco}, a estratégia resumiu-se a: para cada instância positiva, considerar o valor de A_{foco} e procurar entre o texto uma ocorrência deste valor que não seja uma instância positiva, para esse elemento relevante. Se for encontrada essa ocorrência, então o texto constituinte da mesma será o valor de A_{foco}, para a instância negativa. Neste caso serão considerados os tamanhos dos contextos esquerdo e direito, tal com na correspondente instância positiva. Exemplificando com o anúncio contido em 4.12, uma instância negativa para o elemento *preço*, homologa da positiva e segundo este processo, poderia ser obtida da linha número 3, mostrada em baixo com um possível valor para A_{foco}, fixado em "`31,10`", tal como está assinalado no texto:

instância do conceito: `NUMERO`

"`15,66 m2; 18,56 m2 e sala 31,10 m2, ainda está em estuque`".

Supondo que o vector VSx foi generalizado[1] e um dos conceitos considerados foi `NUMERO`, então na linha anterior o valor assinalado ("`31,10`"), seria uma possibilidade a escolher para valor do atributo A_{foco}, por esse valor ser uma instância do conceito `NUMERO`, tal como está assinalado.

Conjugando o anterior com a instância positiva, no anúncio de 4.12 e admitindo que estamos a trabalhar com contextos esquerdo e direito de tamanho 3 (considerando 3 palavras para cada) então a instância positiva e correspondente negativa, gerada da forma descrita, são mostradas na tabela a seguir:

	$E_{preço}$	E_{valor}	E_{venda}	E_{por}	$E_{assunto}$	D_{euros}	A_{foco}	***Classe***
Instância +	sim	não	não	não	não	sim	`NUMERO`	sim
Instância –	não	não	não	não	não	não	`NUMERO`	não

Tabela 4.2 - Instância positiva e correspondente negativa, considerando generalizações e a estratégia "*near miss*" para a procura de instância negativa

1 Tal como descrito na subsecção anterior.

Repare-se que nesta tabela os valores do atributo A_{foco} são "`NUMERO`", para ambas as instâncias, como resultado da generalização realizada, para este caso.

O número de instâncias negativas geradas desta maneira é limitado, e por isso as instâncias negativas aleatórias continuam a ser utilizadas. Todavia o ideal é minimizar a utilização de instâncias negativas aleatórias, pelas razões apresentadas anteriormente. No computo final tenta-se gerar um número de instâncias de treino negativas, igual ao das positivas e se for possível com mais predominância do tipo "*near miss*".

4.4.5 Regras Induzidas e Sua Aplicação

Na subsecção anterior concluímos a descrição do processo de geração de instâncias. Estas são submetidas ao sistema C5 que irá gerar a lista de regras a aplicar na extracção dos elementos relevantes. Agora é chegado o momento de explicar como são aplicadas as regras induzidas, na extracção desses elementos.

Regras Induzidas

Tendo presente os três elementos relevantes considerados (*preço*, *tipo*, *local*), serão apresentadas, de imediato, as regras produzidas no treino, pelo sistema utilizado – C5, para cada um dos elementos relevantes. Foi feita a tradução da sintaxe das regras do C5 para uma sintaxe mais amigável e fácil de ler, do tipo "**se (** condição lógica **) então** classe"

Elemento *preço*

se ($E_{preço}$ = sim **&** $A_{foco} \in Dominio(preço)$ **) então** sim

se (D_{euros} = sim **&** $A_{foco} \in Dominio(preço)$ **) então** sim

se (D_{contos} = sim **&** $A_{foco} \in Dominio(preço)$ **) então** sim

se (D_{cts} = sim **&** $A_{foco} \in Dominio(preço)$ **) então** sim

se (D_{c} = sim **&** $A_{foco} \in Dominio(preço)$ **) então** sim

Elemento *tipo*

se (A_{foco} = "*(numero) assoalhadas*" **) então** sim

se (A_{foco} = "*(numero) quartos*" **) então** sim

se (A_{foco} = "*t2*" **) então** sim

se (A_{foco} = "*t1*" **) então** sim

se (A_{foco} = "*t3*" **) então** sim

se (A_{foco} = "*t5*" **) então** sim

se (A_{foco} = "*t0*" **) então** sim

se (A_{foco} = "*t2+1*" **) então** sim

se (A_{foco} = "*três quartos*" **) então** sim

se (A_{foco} = "*t 3*" **) então** sim

se (A_{foco} = "*(numero) ass*" **) então** sim

se (A_{foco} = "*dois quartos*" **) então** sim

se (A_{foco} = "*2ass*" **) então** sim

se (A_{foco} = "*(numero) assoalhada*" **) então** sim

se (A_{foco} = "*3ass*" **) então** sim

se (A_{foco} = "*t1+2*" **) então** sim

se (A_{foco} = "*t1+1*" **) então** sim

Elemento *local*

se (D_{data} = sim **&** $A_{foco} \in Dominio(local)$ **) então** sim

se ($E_{assunto}$ = sim **&** $A_{foco} \in Dominio(local)$ **) então** sim

se (E_{vendo} = sim **&** $A_{foco} \in Dominio(local)$ **) então** sim

se (E_{de} = sim **&** E_{e} = não **&** D_{em} = não **&** A_{foco} = `LOCAL` **) então** sim

se (E_{em} = sim **&** E_{com} = não **&** E_{e} = não **&** D_{e} = não **&** $A_{foco} \in Dominio(local)$ **) então** sim

se (E_{em} = não **&** E_{com} = não **&** E_{e} = não **&** D_{e} = sim **&** A_{foco} = `LOCAL` **) então** sim

se (E_{vendo} = *não* **&** E_{e} = *não* **&** D_{da} = *sim* **&** A_{foco} = `LOCAL` **)** ***então*** *sim*

se (A_{foco} = "*sta* `LOCAL`" **) então** sim

se (A_{foco} = "*quinta do* `LOCAL`" **) então** sim

se (A_{foco} = "`LOCAL LOCAL`" **) então** sim

se (A_{foco} = "`LOCAL` *do bosque*" **) então** sim

se (A_{foco} = "*st^a eulália*" **) então** sim

se (A_{foco} = "*quinta do s. joão*" **) então** sim

se (A_{foco} = "*praceta quinta de s. joão*" **) então** sim

se (A_{foco} = "*st^o ovideo*" **) então** sim

Estas foram as regras obtidas no treino, para os três elementos relevantes considerados. Verifica-se que alguns conjuntos de regras dependem mais de alguns atributos e outras de outros. Isto é, se repararmos bem, notamos a grande distinção entre o atributo A_{foco} e os restantes. Por exemplo, as regras induzidas para o elemento *preço*, não dependem de A_{foco} enquanto que as regras relativas ao elemento *tipo* dependem exclusivamente deste atributo. Pode-se também observar que para o elemento relevante *local*, existem regras que não dependem de A_{foco}, outras que dependem só deste e ainda outras que dependem deste e dos restantes atributos. Os atributos diferentes de A_{foco}, estão relacionados com os contextos considerados (esquerdo e direito). Concluímos assim que alguns elementos relevantes dependem mais do contexto envolvente que outros, no nosso caso o elemento *preço* tem uma forte dependência do contexto enquanto que o elemento *tip*o é praticamente independente deste. O elemento *local* é um meio termo entre a forte dependência de *preço* e a fraca dependência de *tipo*. Estas dependências são "sentidas" nos resultados obtidos, quando se faz variar o tamanho dos contextos. Esta análise é mostrada na secção 5.2.

Aplicação de Regras Induzidas

As regras apresentadas são aplicadas nos novos anúncios, com o propósito de realizar as extracções, de acordo com o algoritmo 4.4, apresentado no final da secção 4.4.1. Todavia, perante o descrito anteriormente, houve uma questão prática que surgiu, relativamente à aplicação das regras e consequente extracção. Essa dificuldade resume-se à questão de saber como aplicar uma regra que não depende do atributo A_{foco}, mas unicamente de atributos relativos aos contextos esquerdo e direito. Um exemplo disto, são as regras induzidas relativamente ao elemento *preço* e algumas relativas ao elemento *local*. Repare-se que nestas

regras existe uma condição relativa a A_{foco} (ex: $A_{foco} \in Dominio(local)$), mas que não foi produzida pelo C5. Esta condição foi acrescentada para melhor elucidar a aplicação de regras deste tipo, e acordo com a explicação que será feita a seguir.

Como vamos aplicar uma regra como a terceira da listagem anterior, relativa ao elemento *local*, à extracção de texto? Esta só determina que o contexto esquerdo deverá conter a palavra "assunto", dando total liberdade ao atributo A_{foco}. Mas como se pode aplicar isto? Extraí-se o texto que sucede a palavra "assunto", ou que se encontra duas palavras à frente? E que porção de texto se extrai? Uma palavra, duas ou três? Quererá isto significar que uma regra para a qual não esteja definido o atributo A_{foco} é inválida para o processo de extracção? Este foi um problema prático que nos obrigou a uma reflexão mais cuidada. A análise atenta do problema e do significado atributos veio a revelar o verdadeiro sentido das coisas e a solução para este aparente problema. O atributo A_{foco} detém uma importância crucial, mesmo não ocorrendo na regra, para a identificação dos elementos de texto a extrair. Ele detém uma importância superior aos restantes, na medida em que é aquele que mais caracteriza o elemento relevante.

Pelo facto do atributo A_{foco} não ocorrer na conjunção de condições duma regra, não significa que este atributo tenha a liberdade de assumir qualquer valor, pois a este está associado um domínio que foi gerado no processo de treino. Portanto só faz sentido deixar A_{foco} assumir valores dentro desse domínio. Designe-se esse domínio por *Domínio(* A_{foco}) ou *Domínio(elemento*), se soubermos qual é o elemento em causa (ex: *Domínio(local)*). Assim para as regras com o atributo A_{foco} livre, deverá fazer-se uma restrição dos valores deste atributo ao seu domínio – *Domínio(* A_{foco}). Em baixo mostram-se os domínios gerados para cada um dos elementos relevantes considerados. Note-se que pelo apresentado na subsecção 4.4.3, tem-se: *Domínio(* A_{foco}) = GVSx, o vector de generalizações de VSx.

Elemento *preço*

[euros *NUMERO*]
[*NUMERO* mil]
[*NUMERO*]

Elemento *tipo*

[três quartos]
[*NUMERO* assoalhadas]
[*NUMERO* assoalhada]
[*NUMERO* quartos]
[t5]
[t4]
[t-3]
[t3]
[t2]
[t1]
[t0]
[t2+1]
[dois quartos]
[2ass]
[*NUMERO* ass]
[3ass]
[t1+2]
[t1+1]

Elemento *local*

[quinta do s.joão]
[s.joão do *LOCAL*]
[*LOCAL LOCAL*]
[praia da *LOCAL*]
[quinta do *LOCAL*]
[sta *LOCAL*]
[*LOCAL*]
[*LOCAL* do bosque]
[stª eulália]
[*LOCAL* do *LOCAL*]
[*LOCAL* da *LOCAL*]
[s.b. *LOCAL*]
[praceta quinta de s. joão]
[st°ovideo]
[vn *LOCAL*]
[s. *LOCAL*]

Assim para a aplicação duma regra que não depende de A_{foco}, procura-se no texto a ocorrência de algum elemento do domínio de A_{foco}, se tal acontecer verifica-se se a regra em causa satisfaz os contextos dessa ocorrência. No exemplo dado anteriormente (2ª regra do elemento *local*), esta seria satisfeita na porção de texto, contida no esquema da seguinte figura:

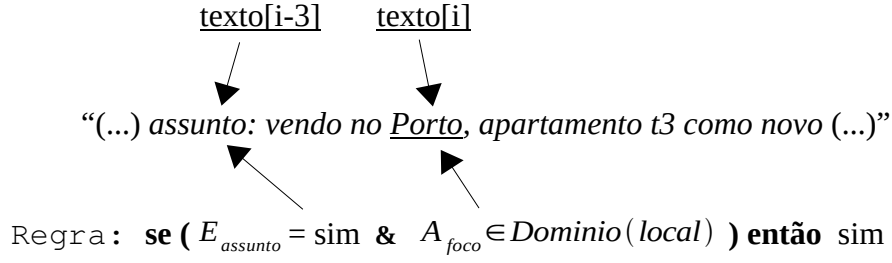


Figura 4.13 - *Extracção com restrição de A_{foco} a Dominio(local)*

pois a palavra "*Porto"* seria reconhecida como um $LOCAL \in Dominio(local)$ e por outro lado o contexto esquerdo, 3 palavras à esquerda da ocorrência do elemento (palavra "*Porto*"), contém a palavra "*assunto*", condição necessária para que a regra apontada seja satisfeita e permita extrair "*Porto*".

4.5 Funcionamento e Detalhes do Sistema

Nesta secção é apresentado o modo o *"modus operandi"* dos vários constituintes do trabalho realizado, do ponto de vista do utilizador. Pelo exposto nos capítulos anteriores, constata-se que na óptica do utilizador existem duas esferas de acção importantes, diferentes e bem demarcadas: *Classificação de Anúncios* e *Extracção de Elementos*. Tendo presente esta demarcação, será apresentado cada um dos itens em separado.

No trabalho realizado não houve a preocupação em criar uma interface bonita e amigável, para o utilizador. Certamente que uma boa interface é importante, todavia resolvemos concentrar o nosso esforço no trabalho concernente ao tema desta dissertação, não nos preocupando com estas questões acessórias. Assim os módulos do nosso sistema são operados num ambiente de linha de comandos. O trabalho foi completamente desenvolvido na linguagem *Java*, e portanto sob um organização *OOP* (*Object Oriented Programming*). No resto desta secção, o símbolo "`%`" será utilizado como indicador da *prompt* do sistema operativo.

4.5.1 Classificação de Anúncios

No que diz respeito à classificação, foram experimentadas duas abordagens: *O método dos k-vizinhos* e *Naive Bayes*, e na segunda foram ainda experimentadas algumas variações (sem e com escolha de atributos). Assim existem dois ficheiros principais que implementam duas aplicações em *Java*, para cada uma das abordagens:

	"k-vizinhos"	***"Naive Bayes"***
Ficheiro principal:	`IBkClassifAPT.java`	`NBayesClassifAPT.java`
Exemplo de execução:	`%java IbkClassifAPT ./site`	`%java NBayesClassifAPT ./site`

A segunda linha do quadro anterior, exemplifica uma execução do programa relativo à abordagem em causa. O parâmetro "`./site`", que é passado na linha de comandos, indica a uma directoria que contém uma colecção de anúncios, das duas classes consideradas. Cada anúncio dessa directoria deve estar em HTML e deve ainda existir um ficheiro denominado

"index.html" que contém precisamente o índice dos ficheiros da directoria que o sistema irá utilizar nos processos de treino e teste. Este ficheiro de índice contém a classificação de cada anúncio e deve ser semelhante ao extracto mostrado a seguir:

INSTÂNCIA - CLASE

apt501.htm - nvenda
apt502.htm - venda
apt503.htm - venda
apt504.htm - venda
apt505.htm - venda
apt506.htm - nvenda
apt507.htm - venda

Figura 4.14 - Anúncios Classificados

Cada linha é formada por um par (nome de ficheiro – classe), em que o nome do ficheiro contem um ligação para um anúncio, e a respectiva classificação. O termo "nvenda" designa não venda. Assim qualquer dos programas de classificação, irá à procura do ficheiro de indicie, na directoria indicada, guardando as referências para cada documento e a respectiva classe e criando uma lista de referências que depois pode ser consultada nas linhas de comandos de cada uma das aplicações, com o comando "`list`".

Passarei a descrever as principais funcionalidades, opções de linha de comandos, da aplicação "`NBayesClassifAPT.java`", sendo a outra aplicação semelhante a esta em termos de operabilidade e até um pouco mais simples (com menos comandos), uma vez que esta implementa a possibilidade do utilizador realizar escolha de atributos, de acordo com o descrito na secção 4.2.3. Após a entrada na aplicação o utilizador fica posicionado numa *prompt*, que neste caso é representada por: "`NaiveBayes>`". Pode ser obtida ajuda com o comando "`help`" ou simplesmente "?", donde será produzida a seguinte listagem:

```
classes – display the diferent classes
stats – display statistics
features <n> – sets n features (Feature Subset Selection)
positive <yes|no> – sets all positive features (Inf. Gain)
set measure – choose measure function, for feature selection.
learn <file path> <class>
classify <path>
freq(<word>) – frequency of "word" in the vocabulary
freq(<word>|<class>) – frequency of "word" in the "class"
fi(<word>) – feature value of "word"
crossv [n] – n-fold cross validation (default: n=10)
list – list examples referenced

help (?) – get help
exit – exit this prompt
```

Tabela 4.3 – Comandos de "NbayesClassifAPT"

Na listagem anterior, cada comando contém uma sua descrição sucinta, o que é suficiente na maioria dos comandos, para quem conhecer o contexto de trabalho desenvolvido.

Alguns comandos dependem de parâmetros recebidos, por exemplo o comando "**features <n>**" estabelece o número de atributos a considerar número esse que é indicado no parâmetro obrigatório "**<n>**" (ex: "**features 50**"). Os caracteres "<" e ">" indicam obrigatoriedade do parâmetro, enquanto que "[" e "]" indicam que o parâmetro é opcional, como é o caso do comando "**crossv [n]**", que realiza validação cruzada de *n* blocos, sendo *n = 10*, por omissão. O caracter "|" indica opcionalidade entre parâmetros, como acontece no comando "**positive <yes|no>**", aqui o utilizador indica que quer ou não que todos os atributos cujo ganho de informação seja maior que zero, sejam considerados. Existem ainda comandos que têm uma sintaxe funcional, como é o caso de "**freq(<word>|<class>)**", onde o caracter "|" tem um significado especial (condicionalidade). Este último comando calcula a frequência da palavra (word) na classe indicada (class), nos exemplos de treino considerados. Há ainda a salientar o comando "**set measure**" que apresenta um pequeno menu ao utilizador, permitindo-lhe seleccionar a medida para escolha de atributos, por exemplo: "*Ganho de Informação*" ou "*Odds Ratio*", conforme apresentado na secção 4.2.3.

4.5.2 Extracção de Elementos

À semelhança do que foi feito nas experiências de classificação realizadas, também aqui foi implementada uma pequena linha de comandos. Esta permite o controlo de todo o processo e é possível experimentar todas as descrições apresentadas neste trabalho (extracção), como o carregamento dos dados e manipulação destes, escolha dos dados de treino e teste, alguma

manipulação sobre o texto e algum cálculo estatístico sobre o mesmo. Antes de descrever as principais funcionalidades dessa linha de comandos, é apresentada uma imagem geral do sistema desenvolvido, com alguma descrição dos principais componentes.

O trabalho de extracção também foi implementado em linguagem Java. Foram implementadas um conjunto de classes, das quais quase todas se encontra representadas no modelo UML da figura 4.15. Nesta secção o termo "classe" é empregue com o significado da terminologia da OOP (*Object Oriented Programming*) e não de acordo com o significado habitual que tem sido dado neste trabalho e que é relativo à "*Data Mining*". Assim, a figura 4.15 mostra não só um conjunto de classes, com alguns atributos e métodos, mas também as relações de *herança* (OOP) que existe entre algumas delas. A principal classe, de todo o trabalho de extracção implementado, é a denominada de "ExemplosAPT". O nome quer significar um conjunto de exemplos de anúncios habitação (APT: abrevia AParTamento) que esta classe controla, através da disponibilização um conjunto de métodos que permitem satisfazer as exigências objectivos iniciais deste trabalho. A própria linha de comandos é implementada nesta classe. Como se pode perceber da figura 4.15, esta classe está relacionada com todas as restantes, quer por relações de herança, quer por utilizar um conjunto de objectos (ex: atributos desta classe – VARX: Variaves) que são instâncias das doutras classes.

A classe "ExemplosAPT" é derivada da classe "StatWords", sendo esta derivada da classe "HashStr" que por sua vez extensão da classe "Hashtable" da linguagem Java que define e e permite manipular uma tabela de hash dum qualquer objecto. A classe "HashStr" é uma adaptação da sua classe pai à manipulação de Strings (classe String do Java), permitindo estabelecer uma relação entre cada chave da tabela de hash, que neste caso é uma string, e um valor numérico inteiro. A classe "StatWords", cujo nome significa "Words Statistic" é uma adaptação à manipulação de texto, permitindo sobretudo uma visão estatística dos elementos do texto: frequências e probabilidades de caracteres e palavras, no texto, etc.

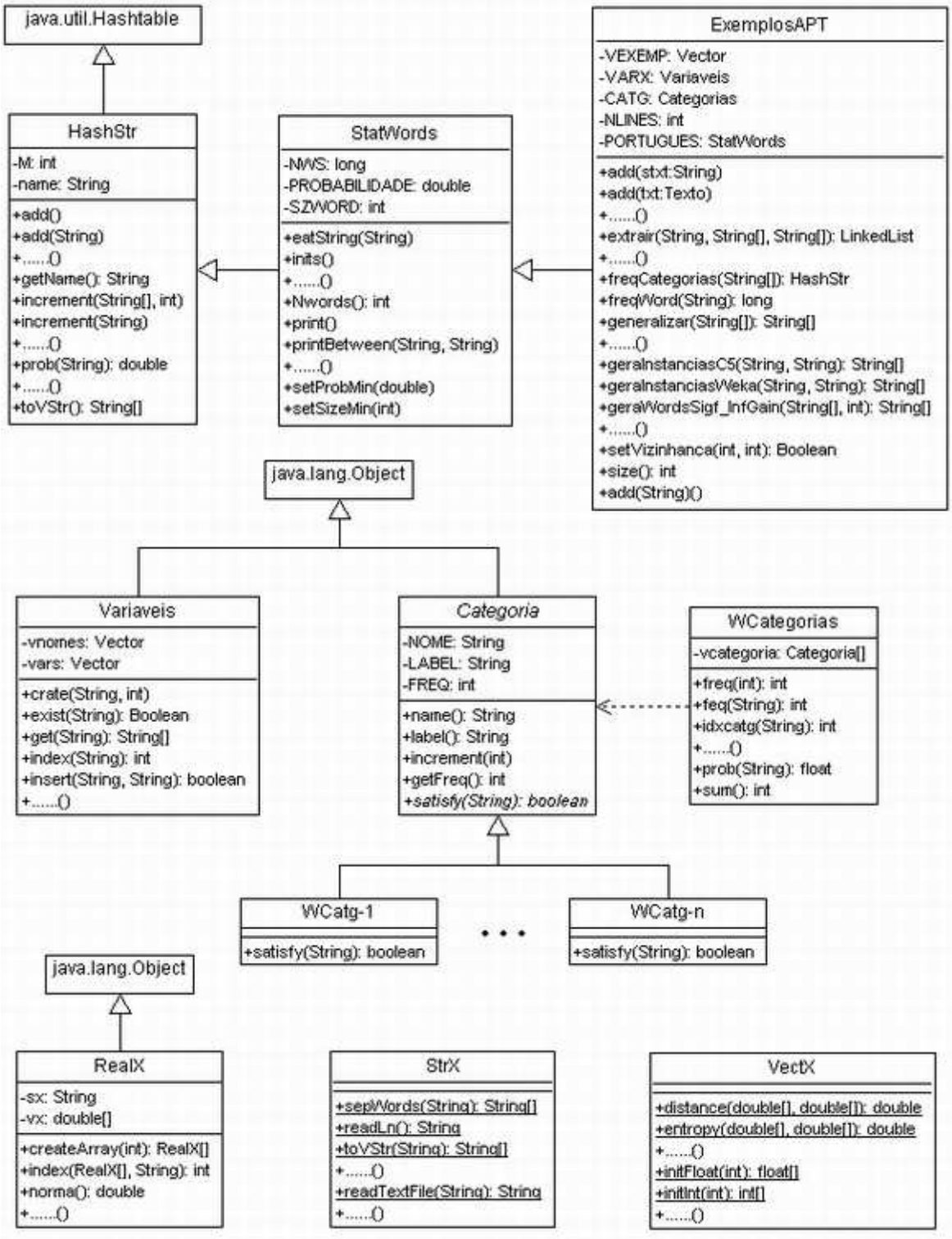


Figura 4.15 - Principais classes (**UML**)

Classe *Variáveis*: Esta classe serve para gerir um conjunto de variáveis da linha de comandos implementada, sendo aqui uma uma variável identificada por uma string (o seu nome) e o seu conteúdo correspondente a um vector de strings.

Classe *Categoria*: Esta é uma classe abstracta que define o que é uma categoria, no sentido do que é apresentado na secção 4.4.3. As *n* classes derivadas desta classe, representadas na figura anterior por "WCatg-1", ... "WCatg-n", definem uma colecção concreta de *n* categorias. Nestas *n* classes o método abstracto "*satisfy*", da classe abstracta "*Categoria*", está implementado e é aquele que para uma determinada categoria (ex: "Local") verifica se a string recebida é de facto um elemento dessa categoria (ex: *Porto*). A classe "WCategorias" depende directamente da classe "Categoria", pois não é mais que a definição duma estrutura que permite manipular uma colecção (vector) de categorias, repare-se esta classe tem, como único atributo, um vector de objectos do tipo "*Categoria*". Através desta classe é possível determinar, por exemplo, se uma dada string é uma instância de alguma categoria conhecida, segundo o que foi descrito na secção 4.4.3.

Classe *RealX*: É uma classe auxiliar que implementa uma relação entre uma string (atributo "sx") e uma colecção de valores reais (atributo "vx") que no mínimo terá um único valor real. Esta colecção de valores reais pode ser interpretada como um vector, um conjunto ou uma sucessão, dependendo daquilo que é necessário fazer. Um exemplo da aplicação desta classe é a criação de um "ranking" de palavras ou expressões, estando cada uma associada a um determinado valor real calculado, por exemplo o *Ganho de Informação*, neste caso teríamos um vector de elementos do tipo RealX, isto é RealX[].

Classes *StrX* e *VectX*: São classes estáticas implementadas, contendo cada um delas uma vasta colecção de funções estáticas para manipulação de strings ou "arrays" de strings, no caso de StrX, e arrays de números (inteiros ou reais), no caso de VectX. Funcionam como bibliotecas de funções que foram implementadas pelas necessidades surgidas durante o trabalho desenvolvido, por exemplo determinadas operações sobre texto, não existindo as mesmas implementadas na linguagem Java, apesar da rica colecção de classes e métodos disponíveis.

Cada uma das classes apresentadas na figura 4.15 está implementada num ficheiro separado cujo nome é constituído pelo nome da classe seguido da extensão ".java", por exemplo "ExemplosAPT.java". Mais alguns detalhes destas classes são fornecidos em anexo, como é o caso dos cabeçalhos de cada método, devidamente comentados.

De seguida será feita uma descrição da linha de comandos implementada, à semelhança do que foi feito na sub-secção anterior, apresentando assim o funcionamento do sistema do ponto de vista do utilizador.

Como foi referido, a classe "ExemplosAPT.java" é a classe mais importante do trabalho de extracção implementado e é ela que implementa a linha de comandos que permite operar o sistema. Após o início da aplicação (com o comando "`%java ExemplosAPT`") o utilizador fica posicionado na linha de comandos desta e cuja *prompt* é: "`comando>`". A tabela 4.5.2-1 que é mostrada a seguir consiste numa listagem de todos os comandos disponíveis, tendo sido conseguida através do comando "?" ou "help". Como se pode verificar, os comandos estão organizados por três grupos: "Variáveis", "Gerais" e "Atalhos Definidos".

O primeiro grupo de comandos, intitulado "Variáveis", apresenta uma lista de possibilidades sobre variáveis, sendo variáveis aqui interpretadas como um vector (array) de strings. Os comandos do tipo "`var <-- Função(.,...,.)`" são operações cujo o resultado é atribuído a uma variável (`var`), por exemplo o comando "`X <-- load(file)`" carrega todas as strings contidas no ficheiro com o nome "file" para a variável "`X`". Embora todos os comandos contenha um pequeno comentário, na tabela apresentada, serão descritos com maior detalhe alguns deles, por estarem directamente relacionados com o trabalho desenvolvido.

Comandos que Actualizam Variáveis

Os comandos "`var <-- gerNGramL(tag,N)`" e "`var <-- gerNGramR(tag,N)`" obtêm um vector de strings, a partir dos exemplos (anúncios) carregados, contendo as sequências de `N` palavras que ocorrem à esquerda e à direita, respectivamente, de todas as ocorrência da etiqueta "`tag`" (etiqueta XML, ex: "`<preco>`" ou "`</preço>`"). Ainda relacionados com estes

```
COMANDOS:
---------

        ------------------------------------------------
        [VARIAVEIS (uma var. contém um array de strings)]
        ------------------------------------------------
        freqchr <var> - especifico para uma variavel
        generalizar <str> - analisa e tenta generalizar a string
        loadv <var> <file> - ler uma variável dum ficheiro
        print <var> - imprime a variável se existir
        savev <var> <file> - grava a variável se existir
        setvvar(var,k,s) - altera o elemento k, na variável "var", para "s"
        sort <var> - ordena vname
        var <-- [s1,...,sn] - redefine "var" com o vector de strings s1...sn
        var <-- generalizar(var) - tenta generalizar o array de strings
        var <-- extract(file,vrules,DomAfoco) - extracção de file
        var <-- gerNEL(tag,N) - gera vector de N-GRAMS à esquerda
        var <-- gerNGramL(tag,N) - gera vector de N-GRAMS à esquerda da tag
        var <-- gerNGramR(tag,N) - gera vector de N-GRAMS à direita da tag
        var <-- gerVNegEx(tag,N,M) - gera vector de exemplos negativos, aleatoriamente
        var <-- loadVR(filename) - carrega um vector de regras
        var <-- gerWSig(var) - palavras mais significativas
        var <-- load(file) - carrega um vector, a partir de um ficheiro
        var <-- seekNeg(tag,vgen,vesq,vdir) - procura exemplos negativos
        var <-- splitWD(var) - separa por palavras
        var <-- tagB(tag) - linhas entre tags
        var <-- tagL(tag) - linhas esquerdas a tag
        var <-- tagR(tag) - linhas direitas a tag


        --------
        [GERAIS]
        --------
        exit - voltar ao sistema operativo
        freqchr - vector de frequências de caracteres
        freqw word - frequencia de uma palavra w
        GIC5 <elemento> <fich. outp.> - gera instâncias para o C4.5/C5
        GIWEKA <elemento> <fich. outp.> - gera instâncias para o WEKA
        load filename - ler um ficheiro de exemplos
        mxnContexto - mostra o tamanho dos contextos considerados
        nTextESQ=<n> - define o tamanho do contexto esquerdo
        nTextDIR=<n> - define o tamanho do contexto direito
        nContext=<n> - define o tamanho do contexto esquerdo e direito
        listFoco tag - mostra as instâncias de Afoco, de todos os exemplos
        probw word - P{word|treino}
        probCtg categoria - P{categoria|treino}


        ------------------
        [ATALHOS DEFINIDOS]
        ------------------
        GIL [filename] - Gera Instancias de Local (C4.5/C5)
        GIP [filename] - Gera Instancias de Preço (C4.5/C5)
        GIT [filename] - Gera Instancias de Tipo (C4.5/C5)

        gerar instancias c5 - gera as instâncias C5 dos três elementos
            relevantes considerados, para a directoria denominada C5 e
            com nomes de ficheiro: preco.*, tipo.*, local.*, sendo que
            "*" in {names, data}.

        testC5 <alvo> <tag> - realiza um teste de extracção, para um
            elemento relevante (tag), no ficheiro indicado por "alvo".
            As regras de extracção são lidas do ficheiro "./C5/<tag>.rules")
            gerado pelo C5. O output é gerado para o ficheiro:
            "EX<tag>.t", exemplo: "EXpreco.t", na directoria "./C5".

        testarC5 - executa o atalho "testC5", para cada um dos elementos
            relevantes considerados: {preco, tipo, local}. Como resultado
            são produzidos os ficheiros EXpreco.t, EXtipo.t e EXlocal.t
```

Tabela 4.4 – Comandos do módulo de extracção

dois comandos, o comando "`var <-- tagB(tag)`", gera o vector contendo todas as strings que ocorrem entre as etiquetas XML, indicadas por "`tag`".

O comando "`var <-- generalizar(var)`" realiza a generalização, se possível, das strings contidas no vector de strings da variável recebida como parâmetro, segundo o que foi apresentado na secção 4.4.3.

Os comandos "`var <--gerVNegEx(tag,N,M)`" e "`var<--seekNeg(tag,vgen,vesq,vdir)`" ambos geram instâncias negativas, o primeiro gera-as quase aleatoriamente enquanto que o segundo faz uma procura do tipo "near-miss" [Winston, 1992], tendo em conta o elemento relevante ("`tag`"), o vector de generalizações ("`vgen`"), e os vectores de contexto esquerdo ("`vesq`") e direito ("`vdir`").

O comando "`var <-- loadVR(filename)`" carrega um vector de regras de extracção, a partir dum ficheiro de regras gerado pelo sistema C5 (`*.rules` - ex: "*preco.rules*"). Este vector de regras é utilizado na acção de extracção, realizada com o comando apresentado a seguir.

O comando "`var <-- extract(file,vrules,DomAfoco)`" consuma a extracção de elementos relevantes, no ficheiro indicado ("`file`"), mediante um vector de regras ("`vrules`") e em conjunção com o domínio do atributo A_{foco} ("`DomAfoco`"), em conformidade com o descrito na secção 4.4.5.

Ainda neste conjunto de comandos, o comando "`print <var>`", permite visualizar o conteúdo de qualquer variável. A versão simplificada deste comando, isto é "`print`", lista todas as variáveis presentes e disponíveis à linha de comandos.

Comandos Gerais

No segundo grupo de comandos estão um conjunto de comandos gerais, como o próprio nome cabeçalho indica. Destes destacam-se os seguintes:

O comando "`load filename`" carrega uma colecção de anúncios de treino ("anotados"), a partir dum único ficheiro XML (ex: "`fx1.xml`" ou "`fx2.xml`"). Este ficheiro é resultante de pré processamento realizado sobre um conjunto de ficheiros de anúncios.

Os comandos "`mxnContexto`", "`nTextESQ=<n>`", "`nTextDIR=<n>`" e "`nContext=<n>`" fazem a gestão do tamanho dos contextos esquerdo e direito, ao elemento relevante, conforme o apresentado na secção 4.4.1.

Os comandos "`GIC5 <elemento> <fich out.>`" e "`GIWEKA <elemento> <fich outp>`" geram instâncias de treino para os sistemas C5 e Weka, respectivamente, a partir dos exemplos carregados e tendo em conta o elemento relevante indicado em "`<elemento>`" (ex: "local"). As instâncias são produzidas para o ficheiro cujo nome é indicado.

Atalhos

Um atalho consiste num comando que realiza vários comandos previamente definidos. Cada um dos atalhos está minimamente explicado, sendo fácil perceber quais os comandos anteriores utilizados por cada um.

5 Avaliação e Resultados

Neste capitulo é feita uma avaliação e apresentados os resultados obtidos com o trabalho desenvolvido no âmbito desta dissertação, descrito anteriormente, nomeadamente em todo o capítulo 4. Como se pode constatar no referido capítulo, foram apresentados dois módulos principais de trabalho realizado: "*Classificação de Anúncios*" (4.2) e "*Extracção de Elementos*" (4.3), com maior destaque para o segundo, uma vez que este é o foco principal deste trabalho. Consequentemente os resultados serão apresentados em separado e para cada um destes módulos.

O trabalho desenvolvido é avaliado segundo as medidas mais convencionais, utilizadas na área do "*Information Retrieval*" e "*Information Extraction*", isto é: *precision*, *recall* e $F_{measure}$. Passar-se-á à descrição sucinta de cada uma destas medidas e a sua importância para a avaliar o desempenho de sistemas nestas áreas. Em qualquer uma destas o objectivo final e concreto é que o sistema seja capaz, por um lado, de obter a informação pretendida da colecção de informação disponível e, por outro, obter só essa informação e não outra qualquer considerada não relevante. Um sistema será tanto melhor quanto maior o volume de informação relevante que consegue obter, mas também quanto mais "pura" for a informação extraída, mais expurgada de informação não relevante. Atendendo a este contexto, temos então as duas medidas apresentadas de seguida:

$$precision = \frac{informação\ relevante\ extraida}{informação\ extraída}$$

$$recall = \frac{informação\ relevante\ extraída}{informação\ relevante\ existente}$$

A primeira medida – *precision*, mede a pureza da informação obtida, o quanto ela é limpa de informação não relevante, a segunda – recall, mede o quanto da informação relevante, existente numa colecção alvo, foi de facto obtida. Como facilmente se percebe, são medidas do tipo probabilístico, situadas no intervalo real [0, 1].

A experiência realizada no final da subsecção 4.3.2, consistiu em utilizar uma abordagem preliminar de extracção, para o nosso problema concreto. As regras induzidas para o elemento

preço, através dessa abordagem, realizaram 327 extracções, numa colecção de anúncios que continha 86 ocorrências do mesmo elemento. Por outro lado, das 327 extracções só 5 eram correctas. Assim a aplicação das medidas referidas resulta em $precision = \frac{5}{327} \approx 0.0153$ e $recall = \frac{5}{86} \approx 0.0581$. Temos assim uma ilustração do cálculo destas medidas e também a confirmação quantitativa de que a abordagem de 4.3.2 era mesmo má.

Quando se quer uma única medida de avaliação dum sistema, utiliza-se a chamada $F_{measure}$ que resulta duma combinação das duas medidas anteriores:

$$F_{\beta} = \frac{(1+\beta^2) * precision * recall}{\beta^2 * precision + recall}$$

em que β é um parâmetro que pesa a importância entre *precision* e *recall*. Para $\beta = 1$ a medida *F* pondera com o mesmo peso os valores de *precision* e *recall*, portanto a qualidade ou não de ambos têm a mesma importância para o resultado:

$$F_1 = \frac{2 * precision * recall}{precision + recall}, \Rightarrow F_1(precision, recall) = F_1(recall, precision)$$

Se $\beta > 1$ então o valor de *recall* é pesado com maior importância, em relação ao valor de *precision*, tanto mais quanto maior for o parâmetro β, repare-se que $\lim_{\beta \to +\infty} F_{\beta} = recall$. Neste caso ($\beta > 1$) o avaliador do sistema dá mais importância à quantidade de informação relevante que o sistema consegue extrair. Se $0 \leqslant \beta < 1$, então é o valor de *precision* que é ponderado com maior importância, tanto mais quanto mais próximo de 0 estiver β, repare-se que $F_0 = precision$. Aqui o avaliador dá mais importância à pureza da informação obtida em relação ao número de elementos relevantes obtidos. Nos resultados mostrados nas secções que se seguem, considera-se $\beta = 1$ e portanto $F_{measure} = F_1$.

Um quadro habitualmente apresentado, em problemas de classificação, é a denominada "*matriz confusão*". Este quadro ou matriz relaciona a classificação realizada pelo sistema com

a classificação verdadeira, por exemplo se a classificação é binária, portanto com duas classes, denominem-se C_+ (positivo) e C_- (negativo), pode questionar-se quantos elementos de informação da classe C_+ foram classificados como tal e o mesmo para a classe C_-? Podemos também querer saber quantos elementos que pertenciam à classe C_+ foram mal classificados, como sendo elementos da classe C_-, pelo sistema, e vice-versa. Estes valores são expressos na matriz confusão, o quadro que se segue, quadro 5.1, é um exemplo de uma matriz confusão para um problema de classificação binária tal como o descrito e semelhante ao trabalho desenvolvido e apresentado na secção 4.2.

	$C_+^{\wedge}$	$C_-^{\wedge}$
C_+	c_{11}	c_{12}
C_-	c_{21}	c_{22}

Quadro 5.1 - Matriz confusão (duas classes)

No quadro anterior $C_+^{\wedge}$ e $C_-^{\wedge}$ designam a classe predita pelo sistema, positiva e negativa, respectivamente. Assim os valores c_{11} e c_{22} traduzem o número de elementos correctamente classificados e que num problema de duas classes também são designados de "*verdadeiros positivos*" (*TP – True Positive*) e "*verdadeiros negativos*" (*TN – True negative*). O valor c_{21} é o número de elementos da classe C_- que foram incorrectamente classificados como $C_+^{\wedge}$, também denominados de "*falsos positivos*" (*FP – False Positive*). Simétrica mente c_{12} expressa o número de elementos classificados como negativos ($C_-^{\wedge}$) mas cuja verdadeira classe era C_+, são os chamados "*falsos negativos*" (*FN – False Negative*). Como se percebe, o ideal é que se tenha $c_{21} + c_{12} = 0$.

A partir de uma matriz confusão, facilmente se calculam os valores de *precision* e *recall* e consequentemente $F_{measure}$, mencionados anteriormente, da seguinte forma:

$$precision = \frac{c_{11}}{c_{11} + c_{21}} \quad \text{e} \quad recall = \frac{c_{11}}{c_{11} + c_{12}}$$

isto é: $precision = \frac{TP}{TP + FP}$ e $recall = \frac{TP}{TP + FN}$.

5.1 Classificação de Anúncios

Na primeira secção deste capítulo são apresentados os resultados obtidos, referentes ao trabalho de classificação, descrito no capítulo 4 (secção 4.2). Foram experimentadas duas grandes abordagens ao problema, que consistia em classificar uma colecção de anúncios em duas classes: "*venda*" e "*não venda*". A primeira abordagem enquadra-se no tema *Aprendizagem baseada em Instâncias*, mais concretamente o método dos "*k vizinhos mais próximos*". A segunda abordagem utiliza *Aprendizagem Bayesiana*, em particular: *Naive Bayes*. Uma variante desta segunda abordagem, que também foi experimentada, consistiu em incorporar escolha de atributos no método ("*feature subset selection*") [Mladenic & Grobel. 1999], apresentado na subsecção 4.2.3. Assim e sistematizando, temos três abordagens experimentadas, no domínio da classificação:

- *k vizinhos mais próximos*
- *Naive Bayes*
- *Naive Bayes com escolha de atributos*

Os resultados obtidos em cada uma das abordagens, foram produzidas a partir duma colecção de 120 anúncios, dos quais 68 são relativos à venda de habitação (classe: *venda*) e 52 relativos a outros assuntos (classe: *não venda*). Foi feita validação cruzada em blocos de 10, conhecida por "*10-fold cross validation*", sobre estes dados, que consiste em dividir os exemplos em blocos de *n* elementos (neste caso *n* = 10), e depois treinar o sistema com todos os blocos, à excepção do bloco *i*, testando o sistema no bloco *i*, que foi deixado fora durante o treino. Este processo é repetido *n* vezes, variando *i* de *1* até *n*. Em cada teste (*i*) são contados e acumulados cada um dos valores de TP, FN, FP, TN e no final tem-se a matriz confusão, referida no início deste capítulo.

Começando assim pelo primeiro ponto (*k vizinhos mais próximos*), temos a matriz confusão, obtida por validação cruzada de 10 blocos, e as correspondentes medidas de desempenho: *precision*, *recall* e $F_{measure}$, obtidas nesta abordagem e sobre os dados referidos:

MATRIZ CONFUSÃO		
	venda	não venda
venda	53	15
não venda	13	39

precision:	0.803
recall:	0.779
$F_{measure}$:	0.791

Quadro 5.2 – "*k-vizinhos*"

Também para a segunda abordagem é apresentada um quadro homólogo ao anterior, traduzindo o desempenho obtido, no mesmos conjunto de dados:

MATRIZ CONFUSÃO		
	venda	não venda
venda	64	4
não venda	22	30

precision:	0.744
recall:	0.941
$F_{measure}$:	0.831

Quadro 5.3 – "*Naive Bayes*"

Este quadro mostra que em termos gerais o desempenho aumentou nesta abordagem, em relação à anterior, passando o valor de $F_{measure}$, de *0.791* para *0.831*, confirmando aqui também o que é geralmente defendido em relação a estas duas abordagens, nos problemas de classificação de documentos. No entanto podemos observar que o número de falsos positivos cresceu significativamente, fazendo com que o valor de *precision* se degradasse.

Uma variante da abordagem pelo método "*Naive Bayes*", que foi experimentada, consiste na utilização deste método, mas com escolha de atributos, tal como foi apresentado na subsecção 4.2.3. A seguir serão mostradas dois quadros semelhantes aos anteriores e que expressam o resultados obtidos, no primeiro com 50 atributos escolhidos e no segundo com 200:

MATRIZ CONFUSÃO

	venda	não venda
venda	58	10
não venda	11	41

precision:	0.841
recall:	0.853
$F_{measure}$:	0.847

Quadro 5.4 – *"Naive Bayes" (50 atributos)*

com 200 atributos (*features*) escolhidos obteve-se:

MATRIZ CONFUSÃO

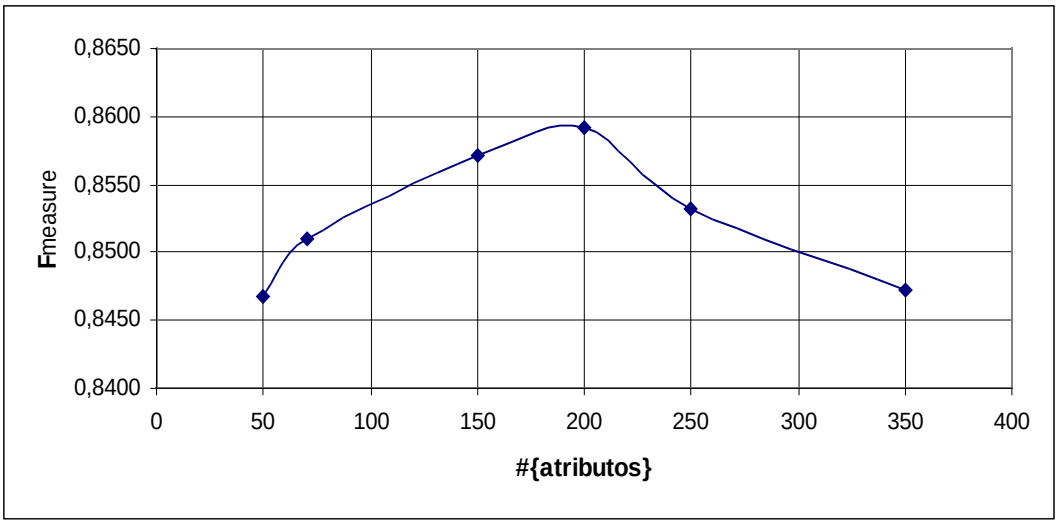
	venda	não venda
venda	61	7
não venda	13	39

precision:	0.824
recall:	0.897
$F_{measure}$:	0.859

Quadro 5.5 – *"Naive Bayes" (200 atributos)*

Dos dois quadros anteriores percebe-se que quando é utilizada escolha de atributos, o desempenho em geral aumenta, confirmando assim as conclusões de outros autores [Mladenic

& Grobel. 1999]. Repare-se que só com 50 atributos (Quadro 5.4) o valor de $F_{measure}$ aumentou para *0.847*, relativamente ao seu homólogo no quadro 5.3. Existe ainda o ganho de eficiência computacional, obtido por esta variante e que resulta da grande redução no número de atributos, uma vez que no método Naive Bayes, sem escolha de atributos, a classificação de um novo documento envolve todas as palavras contidas nos exemplos de treino. Comparando os quadros 5.4 e 5.5 concluímos que houve uma melhoria geral do desempenho, quando foram considerados mais atributos, no caso de 200, obteve-se um valor de $F_{measure}$ igual a *0.859*. Verificou-se também, para o nosso domínio, que para um número de atributos superior a *200*, o desempenho não melhora, tendendo mesmo a degradar-se, como se vê no gráfico mostrado a seguir:



Tal como é explicado na secção 4.2.3, a medida utilizada para o cálculo do valor dum atributo foi o ganho de informação. Constatou-se, no caso concreto, que o valor ***#{atributos}*** = 200 compreende sensivelmente o conjunto de atributos para os quais o ganho de informação é maior que zero, isto é existe algum ganho de informação. Assim a conclusão tirada é que considerar atributos não informativos tende a degradar o desempenho. No final da secção 4.2.3 foram referidas mais duas medidas para a escolha de atributos: *Mutual Information Text* e *Odds Ratio*. Os resultados obtidos com estas medidas foram sensivelmente próximas do obtido com o ganho de informação, tendo esta última mostrado melhor resultado, no nosso domínio concreto.

5.2 Extracção de Elementos

É chegado o momento de apresentar o desempenho do sistema, resultante do trabalho principal desenvolvido no âmbito desta dissertação e em resposta aos objectivos iniciais propostos. Para a avaliação do sistema foi utilizada uma colecção de cerca de 200 anúncios do domínio (venda de habitações), dividia em dois grandes grupos de 100 anúncios cada. Referencie-se estes dois grupos como: "Grupo A" e "Grupo B". Como foi referido anteriormente, foram considerados três elementos relevantes a extrair: *preço*, *tipo*, *local*. Assim o seguinte quadro mostra quantas instâncias de cada um destes elementos relevantes existem nos dois grupos de anúncios:

	Grupo A	***Grupo B***
preço	62	92
tipo	138	211
local	187	220

Os resultados foram obtidos realizando validação cruzada com estes dois grupos de anúncios, primeiro treinando o sistema com o Grupo A e testando-o no Grupo B e depois treinando-o com os anúncios do Grupo B e testando o sistema nos anúncios do Grupo A. O quadro seguinte mostra os resultados obtidos, para cada elemento relevante considerado:

	treino	*teste*	*extraídos*	*TP*	*FP*	*Precision*	*Recall*	$F_{measure}$
<preço>	*Grupo A*	*Grupo B*	104	82	22	*0,788*	*0,891*	*0,837*
<tipo>			185	185	0	*1,000*	*0,877*	*0,934*
<local>			272	177	95	*0,651*	*0,805*	*0,720*
<preço>	*Grupo B*	*Grupo A*	77	56	21	*0,727*	*0,903*	*0,806*
<tipo>			84	84	0	*1,000*	*0,609*	*0,757*
<local>			157	123	34	*0,783*	*0,658*	*0,715*

Na tabela anterior, a coluna "*extraídos*" indica o número total de extracções realizadas nos dados de teste, incluindo os verdadeiros positivos (true positive) e falsos positivos (false positivie). As colunas *TP* e *FP* indicam o numero de verdadeiros positivos e falsos positivos, respectivamente, entre o total de elementos extraídos, repare-se que $TP + FP = extraídos$. A próxima tabela sintetiza a tabela anterior, mostrando os valores de precision, recall e $F_{measure}$

calculados como a média dos dois valores obtidos na tabela anterior, para cada elemento relevante.

m = n = 3	precision	recall	$F_{measure}$
<preço>	0,758	0,897	0,821
<tipo>	1,000	0,743	0,846
<local>	0,717	0,731	0,717

Das tabela anterior, percebe-se que o elemento relevante mais difícil[1] de extrair, consiste no elemento *local*, como seria de esperar, enquanto que a tipologia, elemento *tipo*, é aquele no qual o desempenho foi maior. Um único valor que expresse o desempenho médio do sistema, para os três elementos relevantes é o valor médio de $F_{measure}$ e que é igual a *0,795*.

Análise das Dimensões do Contexto

Pelo exposto no capítulo 4.4, as regras de extracção induzidas, são geradas tendo em conta o contexto esquerdo e direito, à ocorrência dum elemento relevante, por exemplo para o elemento *preço*, estes dois contextos são designados por: $Contexto_{esq}(preço, A, m)$ e $Contexto_{dir}(preço, A, n)$ respectivamente. Os valores *m* e *n*, indicam o número de palavras consideradas para cada contexto. Os resultados expressos nas duas tabelas anteriores foram obtidos considerando $m = n = 3$, isto é, para cada contexto foram consideradas três palavras, a sequência de três palavras que precede a ocorrência do elemento relevante e a sequência de três que sucede essa mesma ocorrência. Pode então colocar-se a seguinte questão: será que considerando contextos mais latos, $m, n > 3$, se obtém melhores desempenhos e em que medida? Os dois quadros que se seguem são hómologos dos dois anteriores e mostram o desempenho, considerando $m = n = 7$:

	treino	teste	extraidos	TP	FP	Precision	Recall	$F_{measure}$
<preço>	Grupo A	Grupo B	139	86	54	0,612	0,924	0,736
<tipo>			185	185	0	1,000	0,877	0,934
<local>			291	191	100	0,656	0,868	0,748
<preço>	Grupo B	Grupo A	108	56	52	0,519	0,903	0,659
<tipo>			88	88	0	1,000	0,638	0,779
<local>			228	156	72	0,684	0,834	0,752

1 A dificuldade deve-se à natureza do elemento a extrair. Existem muitas referências a localidades que podem ser expressas e até abreviadas de maneira diferente. Por outro lado existem nomes de localidades que têm outros significados na língua portuguesa (ex: "*Tábua*", "*Caminha*").

o respectivo quadro dos valores médios:

m = n = 7	*precision*	*recall*	$F_{measure}$
<preço>	*0,565*	*0,914*	*0,697*
<tipo>	*1,000*	*0,757*	*0,857*
<local>	*0,670*	*0,851*	*0,750*

e o valor médio de $F_{measure}$ obtido é de *0,768*. Repare-se que em termos gerais o desempenho piorou sensivelmente, pois no primeiro caso o desempenho médio foi de 0,795 e agora de *0,768*. Se compararmos as colunas de *recall* verificamos que esta medida melhorou para todos os elementos, como se esperaria, pois considerando contextos mais amplos existem mais hipóteses de que as regras induzidas sejam satisfeitas e consequentemente mais elementos relevantes são encontrados. Todavia este ganho tem um preço que se traduz num decréscimo dos valores de *precision*, pelo facto de mais "lixo", estar a ser extraído, repare-se que este valor desceu de *0,758* para *0,565*, no elemento *preço* e de *0,717* para *0,670*, no elemento *local*.

O seguinte gráfico mostram a evolução do valor médio de $F_{measure}$ para contextos simétricos (m=n) e com tamanhos variando entre 3 e 7:

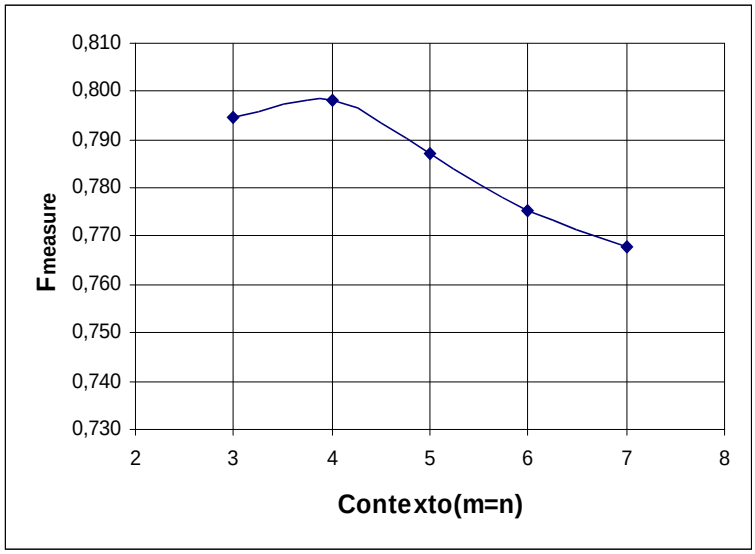


Figura 5.1 - Variação do tamanho do contexto

Uma primeira informação que é inferida da observação do gráfico anterior é o seu valor máximo que acontece para um contexto de tamanho 4. Portanto, para estes elementos e neste domínio parece que em termos de contexto o melhor é trabalhar com este valor. O gráfico também denota que o aumento do tamanho do contexto não implica necessariamente um aumento do desempenho, mas pelo contrário este tende a degradar-se, embora tenhamos ali diferenças muito pequenas há realmente um decréscimo geral do desempenho a partir do tamanho 4. Esta degradação é compreensível, uma vez que palavras muito afastadas à ocorrência dum elemento relevante sejam completamente independentes dessa ocorrência. A partir de certa distância ao elemento as palavras do texto deixam de estar relacionadas com o elemento ocorrido, o texto começa a referir-se a outros assuntos que nada têm a ver com aquela ocorrência. A medida que tende a degradar-se com o aumento do contexto é a *precision*, uma vez que o que acontece é que são extraídos mais elementos não relevantes e portanto a pureza da informação extraída decresce. O valor de *recall* tende a melhorar com o aumento do contexto pois aumentam as probabilidades de encontrar mais elementos relevantes. Estes fenómenos são mostrados no gráfico que se segue. As tabelas relativas aos contextos considerados {3 ... 7}, podem ser consultadas em anexo.

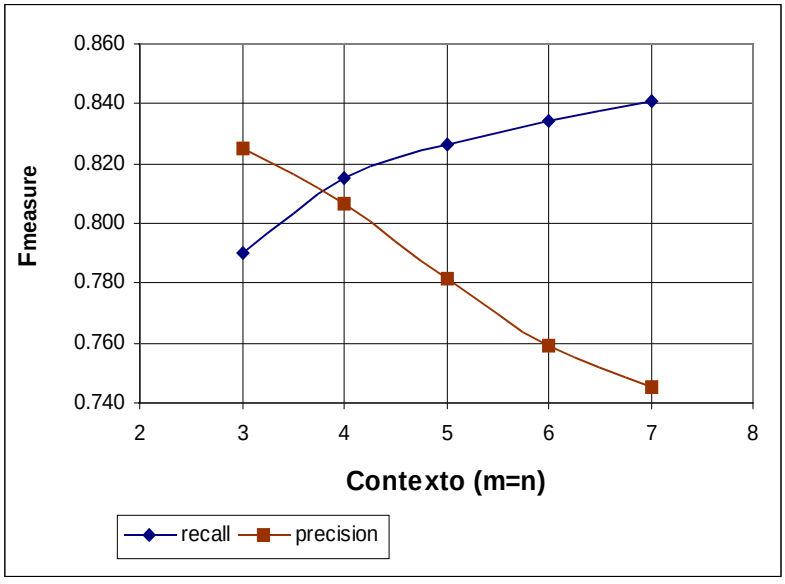


Figura 5.2 - Discriminação entre precision e recall

No gráfico anterior os valores de precision e recall são valores médios para os três elementos relevantes considerados. Nesta variação de contexto, se analisarmos o que acontece com cada um dos elementos relevantes, constata-se que existem algumas diferenças e que são inerentes

à natureza dos elementos a extrair. O próximo gráfico mostra o desempenho geral do sistema, expresso em $F_{measure}$, para cada elemento relevante:

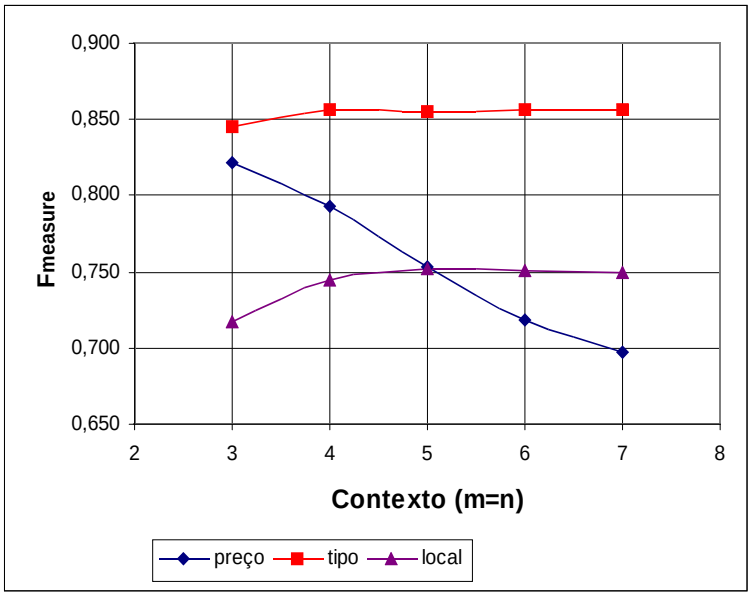


Figura 5.3 - Discriminando os três elementos

Daqui podemos perceber que existem elementos relevantes que possuem uma maior dependência do contexto, em relação a outros. Por exemplo o desempenho sobre o elemento *tipo* mantém-se quase constante com a variação das dimensões de contexto. Isto está em conformidade com as regras induzidas para este elemento e referidas no capitulo 4, que mostram que a identificação deste elemento depende exclusivamente do atributo A_{foco} . Vê-se também que existem diferenças entre o elemento *preço* e o *local*, pois no elemento *preço* o desempenho começa logo a degradar-se e muito a partir do valor 3, enquanto que para o *local* o desempenho só com dimensões de contexto superiores (> 5) começa a degradar-se perdendo muito pouco. Portanto um contexto mais amplo para o elemento *preço* não é benéfico, mostrando que este elemento depende dum pequeno contexto, enquanto que para o elemento local é vantajoso considerar-se uma contexto mais amplo.

6 Conclusões

Em conclusão deste trabalho, afirma-se que os objectivos delineados no início (secção 1.2), foram atingidos, na medida em que foram adaptadas e implementadas um conjunto de técnicas de aprendizagem indutiva, no problema da classificação de documentos e da extracção de informação a partir de texto e num domínio concreto da Língua Portuguesa. Foram também experimentados e comparados alguns métodos já conhecidos. A seguir apresentamos um resumo das conclusões que tiramos deste trabalho.

Classificação de Documentos

Relativamente à classificação de documentos (4.2), os resultados apresentados corroboram a primazia do método *Naive Bayes*, relativamente ao dos *K-Vizinhos*. Por outro lado a variante do método *Naive Bayes*, com escolha de atributos (secção 4.2.3) melhora os resultados, como é salientado na secção 5.1. Todavia, verificou-se que uma excessiva escolha de atributos não melhora o desempenho, devendo considerar-se um certo número destes. Observamos que este número é próximo do número de atributos que apresentam algum (> 0) ganho de informação, relativamente à classificação pretendida.

Extracção de Elementos Relevantes

No que diz respeito à extracção de elementos relevantes, tem-se como primeira conclusão que é possível treinar um sistema para que este induza um conjunto de regras de extracção de elementos, apresentando bons resultados, tal como é expresso nos dados da secção 5.2.

Foi realizada e descrita uma experiência inicial na qual se obtiveram maus resultados, tal como é aí indicado (extracção de *preço*, com 86 ocorrências existentes, foram feitas 327 extracções e só 5 eram verdadeiras). Esta experiência alerta-nos para a necessidade de ser realizada uma boa escolha de atributos (nessa experiência, era pré-determinada por um utilizador – atributos posicionais fixos, ex: palavra anterior à ocorrência do elemento

relevante). As melhorias no trabalho subsequente, deveram-se à utilização de contextos que envolvem uso de conjuntos de palavras ("*bag of words*" para os contextos[1]) e da utilização de generalizações a conceitos, relativamente ao atributo A_{foco} (ex: conceito `NUMERO` e `LOCAL`). Um outro melhoramento utilizado consistiu numa melhor forma de geração de instâncias negativas.

A utilização dum "conjunto de palavras" para os contextos levou a que fosse questionado qual o tamanho dos contextos a considerar. Foi estudada essa variação, para os três elementos relevante considerados, cujos resultados são mostrados na secção 5.2. Do apresentado, podemos concluir que a dependência entre a ocorrência dum elemento no texto e o seu contexto é variável. Em algumas situações a dependência é muito forte, quase exclusiva do contexto (ex: elemento *local*), enquanto que noutras essa depêndencia é quase nula (ex: elemento *tipo*), nestes casos a identificação dum novo elemento a extrair, centra-se mais no texto do elemento em si (atributo A_{foco}). As regras de extracção induzidas, traduzem o grau de depêndencia, entre o elemento relevante e o seu contexto.

Uma outra conclusão relativa à dimensão do contexto considerado, é o facto de, para elementos muito dependentes do contexto, ser necessário algum cuidado em não escolher uma dimensão de contexto demasiado elevado, para não não degradar o desempenho, tal como foi ilustrado no gráfico da figura 5.1. Para contextos com tamanhos demasiado elevados, aumenta a possibilidade de serem escolhidos atributos (palavras) irrelevantes, com consequências evidentes no desempenho.

Assim, conseguiu-se a criação de um sistema capaz de extrair elementos relevantes, a partir de documentos de texto da língua portuguesa. Este sistema enquadra-se na área da *Aprendizagem Automática* , na medida em que são utilizadas algumas técnicas desta área e explicitamente *Aprendizagem Supervisionada*, através dum sistema que gera regras (C5). Uma das vantagens do sistema é o facto do utilizador não ter de fornecer as regras ao sistema, nem escolher os atributos, mas unicamente um conjunto de anúncios anotados. Uma outra vantagem diz respeito à facilidade de adaptar o sistema a alterações do domínio e mesmo à sua migração para domínios completamente distintos (basta fornecer novos documentos de treino).

1 Ver final da subsecção 4.3.2 e secção 4.4.2

Para além dos aspectos qualitativos apresentados, destaca-se que foi alcançado um desempenho médio de 79.5% ($F_{measure}$ médio)[1], relativamente aos três elementos relevantes considerados. Repare-se que este é um valor que se aproxima do desempenho humano, muito próximo dos melhores valores referidos no início da subsecção 4.3.1, para o desempenho dos sistemas actuais (60% a 80% do desempenho humano), muitos deles muito mais complexos.

Assim afirmamos que os nossos objectivos iniciais e expectativas foram atingidos, se não mesmo ultrapassados.

Trabalho Futuro

Apesar dos resultados obtidos, este trabalho pode ser muito mais expandido e enriquecido. Uma primeira sugestão vai no sentido de ser utilizada generalização a conceitos, nos contextos, à semelhança do que é feito relativamente ao atributo A_{foco}. Por exemplo, poderia ser feita uma análise que concluí-se que as palavras "preço", "custo", "valor" eram instâncias de um conceito mais geral (ex: conceito `PREÇO`) e fosse considerado esse conceito como atributo do contexto, em vez das palavras em si.

Alguns dos sistemas de extracção desenvolvidos (referidos alguns na secção 2.2), utilizam muitos conhecimentos da área do *Processamento de Linguagem Natural* (PLN), quer a nível de sintaxe, quer a nível de semântica (ex: LOLITA – secção 2.2.4). O trabalho aqui desenvolvido, não utiliza qualquer informação gramatical ou sintáctica. Todavia essa informação pode ser importante para a identificação dos elementos relevantes, por exemplo: o elemento é precedido por um verbo, ou sucedido por um adjectivo. Então sugere-se a realização de etiquetagem sintáctica do texto (*Part of Speech Tagging*) e estender o conjunto de potências atributos em conformidade.

O trabalho de anotação dos anúncios poderá ser algo custoso e consumidor de tempo, para um utilizador. Assim sugere-se a exploração de uma técnica, denominada de "*Active Learning*" [Thompson et al. 1999], de modo a que o utilizador só tenha de anotar uma parte dos anúncios

1 Secção 5.2

e o sistema vai sendo treinado com estes e sugerindo novos anúncios, os mais promissores, para anotação. Assim este método facilitará o trabalho de anotação do utilizador.

Uma ultima sugestão prende-se com a utilização de léxico e ontologias já pré-definidas. Alguma informação deste tipo, foi utilizada aqui, nomeadamente na generalização do domínio do atributo A_{foco}. O ideal seria poder utilizar mais estruturas de carácter universal já existentes, como é o caso da *WordNet*[1].

Considerações Finais

Evidentemente que cada uma destas extensões possíveis tem os seus custos em termos de desenvolvimento e adaptação ao problema de extracção de elementos relevantes, para a língua portuguesa. Além disso, melhoramentos em termos de taxa de acerto etc, não é sempre garantido.

Em resumo o problema que colocámos nesta tese é importante, na medida em que existe uma necessidade crescente de se conseguir encontrar a informação pretendida, no vasto mundo da *World Wide Web*. O nosso trabalho assenta nas técnicas de *aprendizagem simbólica*, tornando esta meta atingível sem grandes esforços. Esperamos que o nosso trabalho sirva de base para aqueles que decidam segui-lo.

1 http://www.cogsci.princeton.edu/~wn/

7 Bibliografia

[Appelt & Israel 1999] Appelt D. E.; Israel D. J.
"*Introduction to Information Extraction Technology"*
IJCAI-99: Tutorial, 1999

[Berkhin 2002] Berkhin P.
"*Survey Of Clustering Data Mining Techniques*"
Accrue Software, Inc, 2002

[Bratko 2001] Bratko I.
"*PROLOG Programming for Artificial Intelligence*"
Addison-Wesley, 2001

[Breiman et al. 1984] Breiman L., Friedman J., Olshen R., Stone C.
"*Classification and Regression Trees*"
Wadsworth, Inc., California, 1984

[Cestnik et al. 1987] Cestnik B. , Kononenko I. , Bratko I. (1987)
"*ASSISTANT 86: A Knowledge-Elicitation Tool for Sophisticated User*"
In Progress in Machine Learning

[Califf & Mooney 1999] Califf M. E.; Mooney R. J.
"*Relational learning of pattern-match rules for information extraction*"
In Proceedings of the Sixteenth National Conf. on Artificial Intelligence, pages 328–334, 1999.

[Constantino 1997] Constantino M.
"*Financial Information Extraction using pre-defined and user-definable Templates in the LOLITA System*"
University of Durham, 1997

[Constantino 2001] Constantino M.
"*The LOLITA user-definable template interface*"
University of Durham, 2001

[Dreyfus 1979] Dreyfus H. L.
"*What Computers Can' t Do: The Limits of Artificial Intelligence*"
Harper and Row, New York, 1979

[Freitag 1998] Freitag D.
"*Toward GeneralPurpose Learning for Information Extraction*"
In Proceedings of the 36th Annual Meeting of the Association
for Computational Linguistics, 1998

[Garigleano et al. 1998] Garigliano R., Urbanowicz A., Nettleton D. J.
"*Description of the LOLITA System, as used in MUC-7*"
University of Durham, 1998.

[Hobbs & Israel 1994] Hobbs J., Israel D.
"*Principles of Template Design*"
SRI International, 1994.

[Huffman 1995] Huffman S.
"*Learning Information Extraction Patterns From Examples*"
Workshop on new approaches to learning for natural language processing
IJCAI-95: 127-142, 1995.

[Jurafsky & Martin 2000] Jurafsky D. , Martin J.
"*Speech and Language Processing*"
Prentice Hall, 2000

[Krupka 1995] Krupka G. R.
"*Description of the SRA System as Used for MUC-6*"
In Proceedings of the Sixth Message Understanding Conference (MUC-6),
221-235. San Mateo: Morgan Kaufmann, 1995.

[Kushmerick 2000] Kushmerick N.
"*Wrapper induction: Efficiency and expressiveness*"
Artificial Intelligance, 2000; 118:15-68

[McCarthy et al. 1956] MacCarthy J. , Minsky M. L. , Rochester N. , Shannon C. E.
"*A PROPOSAL FOR THE DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE*"
Dartmouth College, 1956

[McCallum & Nigam 1998] MacCallum A. , Nigam K.
"*A Comparison of Event Models for Naive Bayes Text Classification*"
In AAAI-98 Workshop on Learning for Text Categorization, pages
41-48, Madison, WI, 1998.

[Mena 1999] Mena J.
"*Data Mining Your Website*"
Digital Press, 1999

[Michie et al. 1994] Michie D. , Spiegelhalter D.J. , Taylor C.C.
"*Machine learning of rules and trees.*"
Ellis Horwood, 1994

[Miller 1990] Miller G.
"*Wordnet: An online lexical database*"
International Journal of Lexicography, 1990

[Mitchell 1997] Mitchell T.
"*Machine Learning*"
McGraw-Hill, 1997

[Mladenic 1998] Mladenic D.
"*Machine Learning on non-homogeneous, distributed text data*"
University of Ljubljana, 1998

[Mladenic & Grobel. 1999] Mladenic D. , Grobelnik M. (1999)
"*Feature selection for unbalanced class distribution and Naive Bayes*"
Machine Learning: Proceedings of the Sixtheenth International Conference

[Muslea 1998] Muslea I.
"*Extraction patterns: from information extraction to wrapper generation.*"
Technical report, ISI-USC, 1998.

[Peng 2000] Peng, F.
"*HMM for Information Extraction*"
Computer Science Department, University of Waterloo, 2000

[Quinlan 1986] Quinlan J. R.
"*Induction of decision trees*"
Machine Learning 1:81--106. Reprinted in Shavlik and Dietterich (eds.)
Readings in Machine Learning.

[Quinlan 1993] Quinlan J. R.
"*C4.5: Programs for machine learning*"
San Fransisco: Morgan Kaufmann, 1993

[Quinlan 1996] Quinlan J. R.
"*Bagging Boosting and C4.5*"
Proceedings, Fourteenth National Conference on Artificial Intelligence, 1996

[Quinlan 1997] Quinlan J. R.
"*C5.0 Data Mining Tool*"
www.rulequest.com, 1997

[Riloff 1993] Riloff E.
"*Automatically Constructing a Dictionary for Information Extraction Tasks*"
National Conference on Artificial Intelligence, 1993.

[Riloff & Lehnert 1994] Riloff E., Lehnert W.
"*Information Extraction as a Basis for High-Precision Text Classification*"
University of Massachusetts, 1994.

[Russell & Norvig 1995] Russell S. , Norvig P.
"*Artificial Intelligence: A Modern Approach*"
Prentice-Hall, Inc. 1995

[Salton & Buckley 1987] Salton, G. , Buckley, C.
"*Term weighting approaches in automated text retrieval.*"
Cornell University, Department of Computer Science, 1987

[Schapire 2002] Schapire E. R.
"*The boosting approach to machine learning: An overview.*"
MSRI Workshop on Nonlinear Estimation and Classification, 2002.

[Shannon & Weaver 1949] Shannon C. E. , Weaver W.
"*The mathematical theory of communication*"
Urbana IL: University of Illinois Press, 1949.

[Soderland 1996] Soderland S. G.
"*Learning Domain-specific Text Analysis Rules*"
University of Massachusetts at Amherst, 1996

[Soderland 1999] Soderland S. G.
"*Learning information extraction rules for semi-structured and free text*"
Machine Learning, 34 (1-3), 1999.

[Thompson et al. 1999] Thompson C. A. , Califf M. E. , Mooney R. J.
"Active *Learning for Natural Language Parsing and Information Extraction* "
In Proceedings of the Sixteenth International Machine Learning Conference, pp.406-414, Bled, Slovenia, 1999

[Winston 1992] Winston P. H.
"Artificial Inteligence"
Addison-Wesley, 1992.

[Witten & Frank 2000] Witten I., Frank E.
"*Data Mining*"
Morgan Kaufmann Publishers, 2000

◆ Anexo A – Alguns anúncios anotados

<anuncio>
<source>c:\jpc\mestrado\tese\java\site\apt160.htm**</source>**
<text>
   De: FS (nc204010@netcabo.pt)
Assunto: 'URGENTE!!! VENDO ANDAR+GARAGEM

  Data: 2002-08-16 15:09:11 PST

  VENDO **<tipo>**5 ASSOALHADAS**</tipo>**+GARAGEM NO **<local>**PRAGAL-ALMADA**</local>**

TRATA O PRÓPRIO.

FAVOR CONTACTAR PARA:

TELEMÓVEL - 962 821 090
</text>
</anuncio>

<anuncio>
<source>c:\jpc\mestrado\tese\java\site\apt161.htm**</source>**
<text>
   De: Nuno Correia (nunosc@netcabo.pt)
Assunto: Venda \ permuta moradia

  Data: 2002-08-16 05:40:34 PST

Venda \ permuta

Moradia **<local>**Cacém**</local>** Linha de **<local>**Sintra**</local>**
6 anos
<tipo>6 ass**</tipo>**.
<tipo>5 quartos**</tipo>** (uma suite)
3wc
cozinha totalmente equipada
sala com 40m2 com lareira e terraço
1 marquise, 1 adega, 1 arrecadação
garajem, churrasqueira, forno,
quintal com jardim e árvores de fruto.

93 510 10 10
</text>
</anuncio>

<anuncio>
<source>c:\jpc\mestrado\tese\java\site\apt162.htm**</source>**
<text>
   De: Tó Vitor (antonio.ildefonso@netvisao.pt)
   Assunto: vende-se apartamento em **<local>**Beja**</local>**

  Data: 2002-08-16 04:16:55 PST

  Vendo apartamento **<tipo>**T3**</tipo>** em **<local>**Beja**</local>**, com excelente localização, em óptimo estado de conservação.

Tem **<tipo>**3 quartos**</tipo>** (um deles com ar condicionado), duas casas de banho, cozinha
com móveis em bom estado, e sala ampla com lareira e recuperador de calor (e ar condicionado também).

Quem estiver interessado, é favor contactar para
antonio.ildefonso@netvisao.pt.
</text>
</anuncio>

<anuncio>
<source>c:\jpc\mestrado\tese\java\site\apt163.htm**</source>**
<text>
De: Falcon (falcon.lda@netvisao.pt)
Assunto: **<local>**Brejos de Azeitão**</local>**

Data: 2002-08-16 03:27:08 PST

Moradia isolada com **<tipo>**5 ass**</tipo>**. grandes, lote 480 m2, quintal excelente, garagem
com 20 m2, barbecue, salão com lareira, quartos com roupeiro e chão flutuante, alarme, aquecimento e aspiração central, cozinha com lavandaria, caxilharia em alumínio, vidros duplos, escadas em granitos, despensa, hall, local previlegiado, vista para **<local>**arrábida**</local>**.

Peça fotos que enviámos por e-mail:
falcon.joao@netvisao.pt

João Lourenço
Comercial

Falcon – Mediação Imobiliária, Lda
Rua de Cacheu, 7 – A
Amora

Tel./Fax: 21 222 1423
E-mail: falcon.lda@netvisao.pt
</text>
</anuncio>

<anuncio>
<source>c:\jpc\mestrado\tese\java\site\apt168.htm**</source>**
<text>
De: Rusan Rolis (ninguem@megamail.pt)
Assunto: Zona <local>Santarém</local> - Casa Tradicional Ribatejana

Data: 2002-08-14 23:41:52 PST

Casa apalaçada, totalmente restaurada. 8 divisões, tertúlia ribatejana,2 salas anexas, patio, garagem para 2 carros.Mantem a traça original de 1926.
Apenas **<preco>**236.930**</preco>** euros
Pormenores e fotos em:

http://www.geocities.com/xequemate
ou na revista: "CASAS DE PORTUGAL"
</text>
</anuncio>

Anexo B – Regras induzidas (C5)

Elemento *preço*

```
See5 [Release 1.16]     Tue Jun 24 13:43:57 2003
-------------------

    Options:      Rule-based classifiers

Class specified by attribute `extrair'

Read 187 cases (22 attributes) from preco.data

Rules:

Rule 1: (25, lift 2.9)
      ESQpreço = yes
      ->  class yes  [0.963]

Rule 2: (19, lift 2.9)
      DIReuros = yes
      ->  class yes  [0.952]

Rule 3: (13, lift 2.8)
      DIRcontos = yes
      ->  class yes  [0.933]

Rule 4: (18/1, lift 2.7)
      DIRcts = yes
      ->  class yes  [0.900]

Rule 5: (3, lift 2.4)
      DIRc = yes
      ->  class yes  [0.800]

Rule 6: (130/6, lift 1.4)
      ESQpreço = not
      DIReuros = not
      DIRcts = not
      DIRcontos = not
      DIRc = not
      ->  class not  [0.947]

Default class: not

Evaluation on training data (187 cases):

            Rules
        ----------------
          No      Errors

           6    7( 3.7%)   <<


         (a)   (b)    <-classified as
        ----  ----
          56     6    (a): class yes
           1   124    (b): class not

Time: 0.2 secs7
```

Elemento *tipo*

```
See5 [Release 1.16]     Fri May 02 11:06:25 2003
-------------------

    Options:
      Rule-based classifiers

Class specified by attribute `extrair'

Read 279 cases (22 attributes) from tipo.data

Rules:

Rule 1: (25, lift 1.9)
      Afoco = (NUMERO)-assoalhadas
      ->  class yes  [0.963]

Rule 2: (19, lift 1.9)
      Afoco = (NUMERO)-quartos
      ->  class yes  [0.952]

Rule 3: (32/1, lift 1.9)
      Afoco = t2
      ->  class yes  [0.941]

Rule 4: (11, lift 1.9)
      Afoco = t1
      ->  class yes  [0.923]

Rule 5: (23/1, lift 1.9)
      Afoco = t3
      ->  class yes  [0.920]

Rule 6: (5, lift 1.7)
      Afoco = t5
      ->  class yes  [0.857]

Rule 7: (4, lift 1.7)
      Afoco = t4
      ->  class yes  [0.833]

Rule 8: (4, lift 1.7)
      Afoco = t0
      ->  class yes  [0.833]

Rule 9: (3, lift 1.6)
      Afoco = t2+1
      ->  class yes  [0.800]

Rule 10: (2, lift 1.5)
      Afoco = três-quartos
      ->  class yes  [0.750]

Rule 11: (2, lift 1.5)
      Afoco = t-3
      ->  class yes  [0.750]

Rule 12: (5/1, lift 1.4)
      Afoco = (NUMERO)-ass
      ->  class yes  [0.714]
```

```
Rule 13: (1, lift 1.3)
      Afoco = dois-quartos
      ->  class yes  [0.667]

Rule 14: (1, lift 1.3)
      Afoco = 2ass
      ->  class yes  [0.667]

Rule 15: (1, lift 1.3)
      Afoco = (NUMERO)-assoalhada
      ->  class yes  [0.667]

Rule 16: (1, lift 1.3)
      Afoco = 3ass
      ->  class yes  [0.667]

Rule 17: (1, lift 1.3)
      Afoco = t1+2
      ->  class yes  [0.667]

Rule 18: (1, lift 1.3)
      Afoco = t1+1
      ->  class yes  [0.667]

Rule 19: (138, lift 2.0)
      Afoco = bizarre
      ->  class not  [0.993]

Default class: not


Evaluation on training data (279 cases):

            Rules
        ----------------
          No      Errors

          19    3( 1.1%)   <<


         (a)   (b)    <-classified as
        ----  ----
        138          (a): class yes
          3   138    (b): class not


Time: 0.1 secs
```

Elemento *local*

```
See5 [Release 1.16]     Fri May 02 11:06:52 2003
-------------------

    Options:
      Rule-based classifiers

Class specified by attribute `extrair'

      ** This demonstration version cannot process **
      ** more than 400 training or test cases.     **

Read 400 cases (22 attributes) from local.data

Rules:

Rule 1: (76, lift 2.1)
      DIRdata = yes
      ->  class yes  [0.987]

Rule 2: (59, lift 2.1)
      ESQassunto = yes
      ->  class yes  [0.984]

Rule 3: (16, lift 2.0)
      ESQpst = yes
      ->  class yes  [0.944]

Rule 4: (22/1, lift 2.0)
      ESQvendo = yes
      ->  class yes  [0.917]

Rule 5: (36/4, lift 1.9)
      ESQde = yes
      ESQe = not
      DIRem = not
      Afoco = (LOCAL)
      ->  class yes  [0.868]

Rule 6: (40/5, lift 1.8)
      ESQem = yes
      ESQcom = not
      ESQe = not
      DIRe = not
      ->  class yes  [0.857]

Rule 7: (10/1, lift 1.8)
      ESQem = not
      ESQcom = not
      ESQe = not
      DIRe = yes
      Afoco = (LOCAL)
      ->  class yes  [0.833]

Rule 8: (8/1, lift 1.7)
      ESQvendo = not
      ESQe = not
      DIRda = yes
      Afoco = (LOCAL)
      ->  class yes  [0.800]
```

```
Rule 9: (3, lift 1.7)
      Afoco = (LOCAL)-do-(LOCAL)
      ->  class yes  [0.800]

Rule 10: (2, lift 1.6)
      Afoco = sta-(LOCAL)
      ->  class yes  [0.750]

Rule 11: (1, lift 1.4)
      Afoco = quinta-do-(LOCAL)
      ->  class yes  [0.667]

Rule 12: (1, lift 1.4)
      Afoco = (LOCAL)-(LOCAL)
      ->  class yes  [0.667]

Rule 13: (1, lift 1.4)
      Afoco = (LOCAL)-do-bosque
      ->  class yes  [0.667]

Rule 14: (1, lift 1.4)
      Afoco = stª-eulália
      ->  class yes  [0.667]

Rule 15: (1, lift 1.4)
      Afoco = quinta-do-s.joão
      ->  class yes  [0.667]

Rule 16: (1, lift 1.4)
      Afoco = praceta-quinta-de-s.-joão
      ->  class yes  [0.667]

Rule 17: (1, lift 1.4)
      Afoco = st°ovideo
      ->  class yes  [0.667]

Rule 18: (324/111, lift 1.2)
      DIRdata = not
      ->  class not  [0.656]

Default class: not


Evaluation on training data (400 cases):

          Rules
        ----------------
          No      Errors

          18   34( 8.5%)   <<


         (a)   (b)    <-classified as
        ----  ----
         165    22    (a): class yes
          12   201    (b): class not


Time: 0.0 secs
```

Anexo C – Principais classes e métodos

Class ExemplosAPT

```
java.lang.Object
  |
  +--java.util.Dictionary
        |
        +--Hashtable
              |
              +--HashStr
                    |
                    +--StatWords
                          |
                          +--ExemplosAPT
```

All Implemented Interfaces:
java.lang.Cloneable, java.util.Map, java.io.Serializable

---

public class **ExemplosAPT**
extends StatWords

See Also:
Serialized Form

Inner classes inherited from class java.util.Map
`java.util.Map.Entry`

Constructor Summary

ExemplosAPT() Costrutor.	

Method Summary

void	**add**(String stxt) Adiciona uma String em forma de texto, recorrendo ao outro metodo **add**.
void	**add**(Texto txt) Adiciona o texto **txt** aos exemplos existentes.
String[]	**extraccao**(String file, String[] vregras, String[] DomAfoco) Metodo geral de extracção dos elementos relevantes.
LinkedList	**extrai**(String stxt, String sAfoco, String sregra) Particularização do método "extrair".
LinkedList	**extrair**(String stxt, String[] vregras, String[] DomAfoco) Realiza a extracção, num texto **stxt**, atendendo às regras recebidas em **vregras** e com o domínio do atributo *Afoco* passaso no parâmetro **DomAfoco**.

LinkedList	**extraix**(String stxt, String sAfoco, String sregra) Equivalente a: **extraix(String stxt, String sAfoco, String sregra, 0, stxt.length())**.
LinkedList	**extraix**(String stxt, String sAfoco, String sregra, int a, int b) Tenta aplicar a regra **sregra** à extracção, no texto **stxt**, tendo em conta o valor do atributo de foco, passado em **sAfoco**, entre **stxt[a]** e **stxt[b]**.
HashStr	**freqCategorias**(String[] vs) Para determinar qual a frequência de cada categoria, no vector de strings recebido em **vs**.
long	**freqWord**(String word) Calcula a frequência da palavra em **word**, na colecção dos exemplos de treino carregados.
String[]	**generalizar**(String[] vs) Realiza o trabalho de generalização, omitindo o valor mínimo, igual a 1.2.
String[]	**generalizar**(String[] vs, double minOR) Tenta Generalizar um vector de Strings, utilizando a medida de "*Odds Ratio*" [Mladenic, 2001].
String[]	**genVNegExemp**(String tag, int nwords, int nlines) Gera um array de exemplos negativos, para uma tag.
boolean	**geraInstanciasC5**(String tag, String filename) A partir dos vectores de exemplos existentes, são geradas instâncias para serem submetidas ao sistema C5.
boolean	**geraInstanciasWEKA**(String tag, String filename) A partir dos vectores de exemplos existentes, são geradas instâncias para serem submetidas ao sistema WEKA.
String[]	**geraWordsSigf_InfGain**(String[] Vt, int N) Palavras mais significativas, com base no Ganho de Informação.
String[]	**geraWordsSigf**(String[] frases, int N) Devolve um array com as N palavras mais significativas ou informativas, a partir de um array de exemplos.
String[]	**getNElemLeft**(String tag, int N) Devolve um array de n elementos, à esquerda de uma marca **tag**, procurando em todos os exemplos contidos.
String[]	**getNGramLeft**(String tag, int N) Devolve um array de n-grams, à esquerda de uma marca procurando em todos os exemplos contidos.
String[]	**getNGramRight**(String tag, int N) Devolve um array de n-grams, à direita de uma marca **tag**, procurando em todos os exemplos contidos.
long	**getNumWords**() Calcula o número de palavras contidas na colecção dos exemplos de treino, carregados.
String[]	**getTags**() Devolve um array com todas as tags existentes nos exemplos lidos.
Texto	**getText**(int index) Obtem o exemplo referênciado em **index**

String[]	**getVBetween**(String tag) Extrai todas as ocorrências entre as tags: **tag**
String[]	**getVLeft**(String tag) Extrai todas as ocorrências à esquerda da tag: **tag**, até ao inicio da linha.
String[]	**getVRight**(String tag) Extrai todas as ocorrências à direita da tag: **tag**, até ao fim da linha.
protected void	**inits**() Realiza as inicializações gerais, incluindo a superclasse.
boolean	**load**(String fileName) Preenche o objecto actual (this) com os exemplos contidos no ficheiro **fileName**.
static void	**main**(String[] args) MAIN
String	**marcaCategorias**(String frase) É equivalente a **marcaCategorias(String, false, null)**.
void	**marcaCategorias**(String[] vs, boolean flagNwords, String scateg) Semelhante a **marcaCategorias(String, boolean, String)**, mas para um array de strings.
void	**marcaCategorias**(String[] vs, String scateg) É equivalente a **marcaCategorias(String[], false, String)**.
String	**marcaCategorias**(String frase, boolean flagNwords) É equivalente a **marcaCategorias(String, boolean, null)**.
String	**marcaCategorias**(String frase, boolean flagNwords, String scateg) Devolve a string **frase** com possíveis ocorrências das categorias substituidas.
int	**nvizDir**() Devolve o tamanho da vizinhança direita, definida.
int	**nvizEsq**() Devolve o tamanho da vizinhança esquerda, definida.
boolean	**parseCommand**(String command) Responder aos vários comandos introduzidos.
void	**printAfoco**(String tag) Imprime as instâncias de *Afoco* dos exemplos carregados, para um determinada etiqueta (**tag**).
boolean	**setVizDir**(int n) Define o comprimento da vizinhança direita, a **n** palavras.
boolean	**setVizEsq**(int n) Define o comprimento da vizinhança esquerda, a **n** palavras.
boolean	**setVizinhanca**(int m, int n) Define o comprimento das vizinhanças esquerda e direita, a **m** e **n** palavras, respectivamente.
int	**size**() Numero de exemplos de treino carregados.
void	**test20030411_1409**()

void	**test20030411_1746**()
boolean	**verificaVizinhanca**(String regra, String vizEsq, String vizDir) Verifica se uma regra de extracção é coerente com o contexto esquerdo e direito, indicado nos parâmetros **vizEsq** e **vizDir**.

Methods inherited from class StatWords
eatString, Nwords, print, print, print, printBetween, procFile, setProbMin, setSizeMin

Methods inherited from class HashStr
add, add, caos, entropy, entropy, freq, getName, increment, increment, load, print, prob, rankNDouble, readFile, readFile, save, sum, toVStr, toVStr

Methods inherited from class Hashtable
clear, clone, contains, containsKey, containsValue, elements, entrySet, equals, get, hashCode, isEmpty, keys, keySet, put, putAll, rehash, remove, toString, values

Methods inherited from class java.lang.Object
finalize, getClass, notify, notifyAll, wait, wait, wait

Class NBayesClassifAPT

```
java.lang.Object
  |
  +--NBayesClassifAPT
```

public class **NBayesClassifAPT**
extends java.lang.Object

Field Summary

int[]	**confMatrix**
static int	**INFGAIN**
static int	**ODDSRATIO**
String[]	**vFtrain**

Constructor Summary

NBayesClassifAPT(String[] vclssname) To creat this object, it is necessary to send an array with the class names.	

Method Summary

void	**addExample**(String stxt, String classname) Adds a new example for the indicated class: **classname**
String[]	**classes**() Return the class name vector.
String	**classifyExample**(String stxt) CLASSIFY_NAIVE_BAYES_TEXT(Doc) : If possible, uses feature subset selection (number of features previously defined).
String	**classifyExampleSimple**(String stxt) CLASSIFY_NAIVE_BAYES_TEXT(Doc) : Like in Tom Mitchell's, ML book
void	**clearExamples**() Objects are reallocated.
void	**crossValidation**(int N) Teste de validação cruzada: "N-fold cross validation".
String[]	**execTest**(int ia, int ib) Executa o teste no sistema, para os documentos referênciados cujos indices estão compreendidos entre **ia** e **ib**.

void	**execTrain**() Equivalente a **execTrain(-1,-1)**, todos os exemplos.
void	**execTrain**(int ia, int ib) Executa o treino do sistema, para os documentos referênciados cujos indices não estão compreendidos entre **ia** e **ib**.
double	**featureValue**(String w) Returns the feature value for the word **w**.
int	**freq**(String word)
int	**freq**(String word, String classname) Frequency of a word in a class.
int	**freqClss**(String classname) Class frequency.
String[]	**getFeatures**() Returns the features vector (all features selected).
String[]	**getFeatures**(int n) Returns the features vector.
String	**getMeasure**()
String[]	**getVexamples**()
HashStr	**hsWords**(String s) Make word frequências handler, from a text passed in **s** (TF - Term Frequency).
double	**infGain**(String w) ---------- Claculates the feature value, based on "Information Gain" ---------- IG(w) = Sigma{i} P(Ci) * [P(w|Ci)*L(w,Ci) + P(~w|Ci)*L(~w,Ci)] sendo: L(x,c) = log(P(x|c) / P(x))
static void	**main**(String[] args)
int	**numFeatures**() Number of features selected.
long	**numVocab**() Number of different words, in training set.
long	**numWords**() Total number of words, in training set.
double	**oddsRatio**(String w) ---------- Claculates the feature value, based on "Odds Ratio" ----------
void	**printPerformance**() Output de medidas de performance.

double	**prob**(String word) Word probability
double	**prob**(String word, String classname) Word probability in some class.
double	**probClss**(String classname) Class probability
double	**probF**(String word) Feature probability
double	**probF**(String word, String classname) Word probability in some class.
boolean	**setFeatures**(int n) Feature subset selection, with «"Odds Ratio"», Like [Mladenic, Grobelnic 1998]
void	**setMeasure**(int measure)
boolean	**setPath**(java.io.File dir) Defines the training directory

Anexo D – Código dos principais métodos

Classe "*ExemplosAPT*"

```
/**
 * Palavras mais significativas, com base no Ganho de
 * Informação.
 */
public String[] geraWordsSigf_InfGain(String[] Vt, int N)
{
    if ( Vt == null )  return null;

     Texto twVt= new Texto(Vt);

     double Pext= (double) twVt.getNumWords() / this.getNumWords();
     double Pnext=(double) 1.0 - Pext;
     double ES= F(Pext) + F(Pnext);  //--------> Entropy(S)
    long cardS= (long) (Vt.length / Pext); //---> Card(S)

    Hashtable hasht= new Hashtable();
    so.println("a) hasht.size(): "+hasht.size());
     twVt.setPosWord(0);

     for (;;)  {
         String w= twVt.nextWord();
         if ( w == null )  break;
        if ( hasht.containsKey(w) )  continue;

        // frequências
        long Fw= this.freqWord(w); //------> frequencia total de: w
        long Fxw= twVt.freqWord(w); //------> frequencia nos casos de extracção

        // cardinalidades
        long cardSw= Math.min(Fxw, Vt.length) + Math.min (Fw-Fxw, cardS-Vt.length);
        long cardSnw= cardS - cardSw;

        // probabilidades
        double PExtW=(double) Math.min(Fxw, Vt.length) / cardS;
        double PnExtW= 1.0 - PExtW;
         double PExtnW=(double) Math.max(Vt.length-Fxw, 0) / cardS;
         double PnExtnW= 1.0 - PExtnW;

         // entropias
         double ESw= F(PExtW) + F(PnExtW);
         double ESnw= F(PExtnW) + F(PnExtnW);

         // ganho de informação
         double GainSw= ES - ((double) cardSw/cardS * ESw +(double) cardSnw/cardS * ESnw );

        hasht.put(new String(w), new Double(GainSw));

        /* Mostrar Ganhos
        so.println("Gain(S,W): "+GainSw+"\t\tw: "+w);
         so.println("|Sw|: "+cardSw+"\tP{ext|w}: "+PExtW+"\tP{ext|~w}: "+PExtnW);
        so.println("-----------------------------------------------------------------");
        */
     }
    so.println("ES:..................... "+ES);
    so.println("|S|:..................... "+cardS);
    so.println("Pext:................... "+Pext);
    so.println("Vt.length:........ "+Vt.length);

    if ( hasht.size() <= 0  )  return null;

    String[] vs= HashStr.rankNDouble(hasht,N);
    return vs;
}
```

```
/**
 * Metodo geral de extracção dos elementos relevantes. Para um ficheiro, indicado
 * em <b>file</b>, um vector de regras, em <b>vregras</b>, e um vector com o
 * domínio do atributo de "foco" em <b>DomAfoco</b>, é concretizada a extracção
 * dos elementos relevantes.<br>
 */
public String[] extraccao(String file, String[] vregras, String[] DomAfoco)
{
   if ( vregras == null  ||  vregras.length < 1 )  return null;

   LinkedList lista= new LinkedList();
   try  {
       BufferedReader br= new BufferedReader(new FileReader(file));

        int nanunc= 0, nextr= 0;
       boolean inText= false;
       String source=null, line;
       do  {
           // Ler próximo anúncio
           // -------------------
           String stxt= null;
           do  {
               line= br.readLine();
               if ( line == null )  break;

               if ( line.indexOf("<source>") != -1 )  {
                   source= StrX.betweenTags(line,"<source>");
                   so.println("source: "+source+"   "+nextr);
                   int p= source.lastIndexOf('\\');
                   if ( p >= 0 )  source= source.substring(p+1);
                   nanunc++;
               }

               if ( line.indexOf("<text>") != -1 )  {
                   inText= true;
                   stxt= "";
               }
               if ( line.indexOf("</text>") != -1 )  {
                   inText= false;
                   break;
               }

               if ( inText )  {
                   line= StrX.removeTags(line.toLowerCase());
                   stxt+= ' ' + line;
               }
           }
           while ( true );
           if ( line == null )  break;
           if ( stxt == null  ||  stxt.length() < 1 )  continue;
           // -------------------

           stxt= StrX.removeTags(stxt.toLowerCase());
           stxt= filtraTexto(stxt);
           //so.print("----------------\n");
           stxt= "[BEGIN] " + stxt + " [END]";
           //so.print(stxt+"\n");
           //so.print("----------------\n");

           LinkedList lx= extrair(stxt, vregras, DomAfoco);
           if ( lx != null )  {
               for (int i=0; i<lx.size(); i++)  {
                   String si= (String)lx.get(i);
                   si= "F("+source+") -> " + si;
                   if ( ! lista.contains(si) )  {
                       lista.add(new String(si));
                       nextr++;
                   }
               }
           }
       }
       while ( true );

        so.println("\n\n\n---------------------------------");
        so.println("N(anuncios):..... "+nanunc);
        so.println("N(extracts):..... "+nextr);

       br.close();
```

```
    }
    catch (IOException ioe)  {
        se.println("ERROR: reading file "+file);
        se.println(ioe);
        return null;
    }

    return StrX.toVStr(lista);
}



/**
 * A partir dos vectores de exemplos existentes, são geradas
 * instâncias para serem submetidas ao sistema C5.<br>
 */
public boolean geraInstanciasC5(String tag, String filename)
{
    PrintStream outD= null, outN= null;
    boolean closeD= false, closeN= false;

    // abertura de ficheiros
    // ---------------------
    if ( filename == null || filename.length() < 1 )  {
        outD= System.out;
        outN= System.out;
    }
    else  {
        int p= filename.indexOf('.');
        if ( p > 0 )  filename= filename.substring(0,p);
        String fnames= filename + ".names";
        String fdata= filename + ".data";

        try  {
            FileOutputStream fosN= new FileOutputStream(fnames);
            FileOutputStream fosD= new FileOutputStream(fdata);
            outN= new PrintStream(fosN);
            outD= new PrintStream(fosD);
            closeN= true;  closeD= true;
        }
        catch (IOException e)  {
            outN= System.out;
            outD= System.out;
        }
    }
    // ---------------------

    String[] Btag= getVBetween(tag);  //Between tag
    String[] Ltag= null;  //Left tag
    String[] Rtag= null;  //Right tag

    Ltag= getNGramLeft(tag , nvizEsq());
    Rtag= getNGramRight(tag, nvizDir());

    if ( Ltag == null  ||  Btag == null  ||  Rtag == null )  return false;

    VARX.insert("B"+tag,Btag);
    VARX.insert("L"+tag,Ltag);
    VARX.insert("R"+tag,Rtag);

    String[] GBtag= this.generalizar(Btag);
    String[] WSLtag= this.geraWordsSigf_InfGain(Ltag,10);
    String[] WSRtag= this.geraWordsSigf_InfGain(Rtag,10);

    VARX.insert("GB"+tag,GBtag);
    VARX.insert("WSL"+tag,WSLtag);
    VARX.insert("WSR"+tag,WSRtag);

    outN.print("extrair\n\n");

    for (int i=0; i<WSLtag.length; i++)
        outN.print("ESQ"+WSLtag[i]+": yes,not.\n");

    for (int i=0; i<WSRtag.length; i++)
        outN.print("DIR"+WSRtag[i]+": yes,not.\n");

    outN.print("between: ");
    //outN.print("Afoco: ");
```

```
for (int i=0; i<GBtag.length; i++)  {
    String sattrb= GBtag[i].replace(' ','-').replace(',',';');
    outN.print(sattrb+",");
}
outN.print("bizarre.\n");

outN.print("extrair: yes,not.\n");

if ( closeN )  outN.close();

//------------------------------
// Ciclo de geração de instâncias
//------------------------------
String etag= StrX.endtag(tag);
String sinstancia, sinstneg, sneg;
LinkedList listneg= new LinkedList();
String[] vsleft, vsrigh, vsleftneg, vsrighneg;
for (int i=0; i<Btag.length; i++)  {
    sinstancia= "";
    sinstneg= "";
    sneg= this.genNegExemp(tag, StrX.countWords(Btag[i]));
    vsleft= StrX.trim( StrX.toVStr(Ltag[i]) , worddata.pontuacao );
    vsrigh= StrX.trim( StrX.toVStr(Rtag[i]) , worddata.pontuacao );

    vsleftneg= StrX.toVStr(StrX.leftTag(sneg,tag));
    vsrighneg= StrX.toVStr(StrX.rightTag(sneg,etag));

    if ( vsleftneg != null )
       vsleftneg= StrX.subVect(vsleftneg, vsleftneg.length-nvizEsq(),
                               vsleftneg.length);

    if ( vsrighneg != null )
       vsrighneg= StrX.subVect(vsrighneg,0,nvizDir());

    vsleftneg= StrX.trim(vsleftneg, worddata.pontuacao);
    vsrighneg= StrX.trim(vsrighneg, worddata.pontuacao);

    // ocorrência do atributo na vizinhança esquerda
    for (int j=0; j<WSLtag.length; j++)  {
       sinstancia+=  StrX.indexOf(vsleft,WSLtag[j]) != -1 ? "yes," : "not,";
       sinstneg+= StrX.indexOf(vsleftneg,WSLtag[j]) != -1 ? "yes," : "not,";
    }

    // ocorrência do atributo na vizinhança direita
    for (int j=0; j<WSRtag.length; j++)  {
       sinstancia+=  StrX.indexOf(vsrigh,WSRtag[j]) != -1 ? "yes," : "not,";
       sinstneg+= StrX.indexOf(vsrighneg,WSRtag[j]) != -1 ? "yes," : "not,";
    }

    String betw= null;
    for (int j=0; j<GBtag.length; j++)
       if ( satisfazGen(GBtag[j], Btag[i]) )  {
           betw= GBtag[j].replace(' ','-').replace(',',';');
           break;
       }
    sinstancia+= betw != null ? betw+",yes" : "bizarre,yes";

    sneg= StrX.betweenTags(sneg,tag);
    betw= null;
    for (int j=0; j<GBtag.length; j++)
       if ( satisfazGen(GBtag[j], sneg) )  {
           betw= GBtag[j].replace(' ','-').replace(',',';');
           break;
       }
    sinstneg+= betw != null ? betw+",not" : "bizarre,not";

    outD.print(sinstancia+"\n");     //---> instância positiva.
    listneg.add(new String(sinstneg));      //---> instância negativa (aleatoria).
}
//------------------------------


//------------------------------
 //geração de exemplos negativos
//------------------------------
 String[] vexneg= this.seekNegExemp(tag, GBtag, WSLtag, WSRtag);
String[] vinstneg= StrX.toVStr(listneg);  //---> exemplos artificiais (aleatórios)
int j= 0;
```

```
        for (int i=0; i<vexneg.length; i++)  {
            outD.print(vexneg[i].replace(' ','-')+"\n");
            if ( j < vinstneg.length )  outD.print(vinstneg[j++]+"\n");
            if ( i == Btag.length )  break;
        }
        for (int i=j; i<vinstneg.length; i++)  {
            if ( i == Btag.length )  break;
            outD.print(vinstneg[i]+"\n");
        }

        if ( closeD )  outD.close();

        return true;
    }
```