INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

DEPARTAMENTO DE ENGENHARIA ELECTRÓNICA E TELECOMUNICAÇÕES E DE COMPUTADORES (DEETC)

ENGENHARIA INFORMÁTICA E DE COMPUTADORES

PROCURA DE PADRÕES EM DOCUMENTOS PARA EXTRACÇÃO E CLASSIFICAÇÃO DE INFORMAÇÃO

João Carlos Godinho Ferreira

DISSERTAÇÃO PARA OBTENÇÃO DO GRAU DE MESTRE
EM ENGENHARIA INFORMÁTICA E DE COMPUTADORES

Orientador:
Professor Dr. Paulo Trigo
Co-Orientador:
Professor Dr. João Ferreira

Novembro de 2008

"There are two types of knowledge. One is knowing a thing. The other is knowing where to find it."

Samuel Johnson

Resumo

A limitada capacidade dos computadores em processar documentos de texto e consequente dificuldade de extracção de informação desses documentos deve-se à dificuldade de processamento de informação não-estruturada. De modo a reduzir essa limitação é necessário aumentar a estrutura dos documentos com que os computadores trabalham.

Este trabalho propõe um modelo de classificação de documentos através de um processo de refinamento sucessivo da informação. A cada iteração a informação presente no documento é melhor caracterizada através da aplicação de um classificador apropriado. O processo de classificação recorre a informação estatística, usando o modelo de classificação de Bayes, sobre documentos ou fragmentos de documentos. O processo de classificação também recorre a técnicas para especificação de padrões de texto, usando expressões regulares para extrair informação que exibe um padrão conhecido.

A informação obtida é armazenada em XML, que permite a interrogação de colecções de documentos de modo automático (recorrendo a bases de dados de suporte nativo XML). O XML também é usado para transformar a informação original noutros formatos, como por exemplo o HTML. Este formato pode ser usado para sintetizar a informação de modo melhorar a sua apresentação.

Abstract

The limited capacity of computers in processing text documents and the consequent difficulty of extracting information from them derives from the difficulty of interpreting non-structured information. In order to reduce that limitation it is necessary to augment the structure of the documents being analysed by computers.

This work proposes a model of classification of documents through a process that necessarily refines information. At each iteration the information in the document is better characterized by using the appropriate classifier. The classification process emploies statistical information using the Bayes classification model on documents or documents fragments. The classification process also applies techniques for specification of text patterns using regular expressions, to extract information that exhibits a known pattern.

The retrieved information is stored in XML, which allows the interrogation of documents collections in an automatic way (using native XML support databases). XML is also used to transform the original information to other formats, such as HTML. This format can be used to synthesize the information to improve it's presentation.

Agradecimentos

Acabar esta dissertação envolveu bastante esforço, e que não teria sido possível sem a ajuda das pessoas aqui mencionadas. Começo por agradecer ao Professor Paulo Trigo, meu orientador e ao Professor João Ferreira, meu co-orientador. Sem os seus conhecimentos e disponibilidade este trabalho não seria possível.

Agradeço aos meus colegas e amigos Bruno Pereira, Tiago Garcia e Paulo Marques que sempre ajudaram no que foi preciso. Mesmo com os seus trabalhos nunca deixaram de estar presentes.

Agradeço aos meus pais Ana e Fernando e ao meu irmão Miguel pela força que me transmitiram e por me incentivarem a prosseguir. Agradeço também à minha namorada Raquel que tanto apoio me deu.

Finalmente agradeço a todas as outras pessoas que se disponibilizaram para me ajudar na escrita deste documento, com as suas criticas e sugestões.

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas

EI	**E**xtracção de **I**nformação
SVC	**V**ector **S**pace **C**lassification
fc	**F**eature **C**ount
cc	**C**ategory **C**ount
ms	**M**ili **S**econds
pdf	**P**ortable **D**ocument **F**ormat

Capítulo 1

Introdução

Este trabalho é motivado pela limitada capacidade dos computadores em processar documentos de texto e consequente dificuldade em extrair informação desses documentos. Os computadores lidam bem com informação estruturada (e.g. numa base de dados) mas têm maior dificuldade em trabalhar com informação não estruturada, como aquela presente na maioria dos documentos de texto. Esta capacidade de percepção e análise semântica é uma das características que diferenciam os humanos das máquinas e que permite aos humanos lidar facilmente com informação não estruturada. Como exemplo dessa diferença considere-se a leitura de um simples jornal. Para um computador um jornal é apenas um conjunto de caracteres e algumas fotografias. Um ser humano consegue, ao observar o mesmo jornal, perceber se é um desportivo ou económico, se é actual ou ainda quais são os assuntos retratados. O humano vê mais do que caracteres, vê o seu significado.

Interpretar e atribuir significado a um símbolo é um processo que transforma informação em conhecimento e que está mais "perto" do homem do que da máquina. Actualmente o computador está mais vocacionado para lidar com informação que apresente um formato estruturado; pelo contrário o Homem analisa e compreende conteúdos não estruturados.

Este trabalho explora a organização natural dos documentos para extrair, a partir deles, padrões de informação. Neste trabalho propõe-se um método de classificação para dotar os computadores de maior capacidade de estruturação de documentos contribuindo assim para que os consigam interpretar e consequentemente extrair informação.

1.1 Objectivos do trabalho desenvolvido

Neste trabalho é proposto um modelo de classificação baseado na caracterização incremental do conteúdo de documentos de texto. O modelo aplica repetidamente um, ou mais, classificadores diferentes para detectar padrões em documentos. A análise iterativa do documento permite refinar incrementalmente a informação obtida em cada iteração anterior. O objectivo final é ter a informação original organizada num formato semi-estruturado.

O modelo de classificação contempla a aplicação de classificadores em três fases de classificação distintas:

- **Fase I.** Aplicação do classificador Naïve Bayes com o objectivo de classificar cada um dos documentos numa colecção numa de várias possíveis categorias definidas numa taxonomia. O Naïve Bayes é um classificador estatístico que quantifica a probabilidade de um documento pertencer a cada categoria. O método de classificação determina em que categoria se classifica o documento (cf. Capítulo 3.1.2).
- **Fase II.** Após a classificação do documento inicia-se a tarefa de classificação dos seus fragmentos. As categorias passíveis de classificar cada fragmento dependem da categoria do documento determinada na fase I. O objectivo da classificação de fragmentos é encontrar padrões contidos no documento, para que a informação contida nesses padrões possa ser extraída na fase seguinte de classificação (cf. Capítulo 3.1.3).
- **Fase III.** A terceira fase de classificação consiste em aplicar o classificador de expressões regulares (RegEx) aos fragmentos do documento anteriormente classificados. Este classificador usa expressões regulares para identificar padrões de caracteres, para seguidamente extrair informação presente no texto identificado (cf. Capítulo 3.1.4).

O modelo descrito apresenta um conjunto de funcionalidades que aumentam a eficiência do processo de classificação, tornando-o mais rápido e eficiente:

- *Persistência do classificador.* Permite recuperar uma instância do classificador Naïve Bayes armazenada em disco.
- *Treino continuo do classificador.* Permite continuar o treino mesmo após esta fase ter terminado.
- *Aprovação de utilizador.* Possibilita que um utilizador aprove uma classificação, sendo possível corrigi-la no caso da classificação estar errada.

- *Construção de taxonomias.* São apresentadas duas ferramentas para auxiliar na definição de taxonomias. Numa das ferramentas a definição é feita de modo manual e na outra ferramenta a definição é feita de modo semi-automático com recurso a técnicas de agrupamento.

A fase final do processamento de um documento consiste no armazenamento de forma persistente da informação que se obteve durante o processamento do documento. O armazenamento deve ser feito numa forma estruturada para que os computadores a possam processar. O formato Extensible Markup Language (XML) é uma tecnologia actual e largamente utilizada em especial na gestão, apresentação e organização de informação. Tendo em conta as características deste formato e de outras alternativas analisadas (e.g. RDF), foi o escolhido o XML pois é simples de entender tanto por humanos como por computadores e que pode ser transformado noutro qualquer formato.

Este trabalho demonstra como a transformação da informação em XML é importante no uso e apresentação da mesma. Ao armazenar a informação em documentos XML é possível apresenta-la de outras formas, como por exemplo em HTML (isto é conseguido através de criação de regras de transformação eXtensible Stylesheet Language Transformations). A representação da informação em HTML é mais compreensível (para o Homem), mas também permite sintetizar a informação importante.

A estruturação da informação permite o uso de bases de dados de suporte nativo ao XML (e.g. eXist (Meier, 2003)) para interrogar colecções de documentos deste tipo. Isto permite interrogar grandes colecções de documentos e sintetizar mais facilmente a informação, tal como efectuar cálculos que englobem agregações de dados (e.g. calcular montantes gastos ou sumariar documentos que possuam palavras especificas).

O essencial do trabalho desenvolvido foi apresentado em (Ferreira, 2008).

1.2 Organização do documento

Este documento está organizado em seis capítulos cujo conteúdo é o seguinte:

- *Capítulo 1: Introdução.* Onde é apresentada a motivação, os objectivos do trabalho realizado e a organização deste documento.
- *Capítulo 2: Estado da Arte.* É feito o enquadramento deste trabalho e são apresentados sistemas que aplicam técnicas de classificação e reconhecimento de padrões semelhantes aquelas usadas neste trabalho. São caracterizadas as noções de classificação, classificador e de aprendizagem supervisionada e não supervisionada.

Descrevem-se também alguns algoritmos de classificação utilizados ao longo deste trabalho.

- *Capítulo 3: Modelo de classificação de documentos.* É descrito o modelo de classificação proposto. Em primeiro lugar é descrita a arquitectura do modelo de classificação e são explicadas as suas várias componentes. Os métodos explorados neste trabalho (apresentadas no capítulo anterior) são aqui caracterizados e adaptados ao objectivo deste trabalho. Neste capítulo são também apresentadas duas ferramentas auxiliares para a definição de categorias de documentos (taxonomia). No final do capitulo são discutidos alguns aspectos da implementação do protótipo.

- *Capítulo 4: Concretização.* Neste capítulo é descrita a concretização de um protótipo do modelo de classificação proposto (melhorar por refinamento sucessivo). É também apresentada a taxonomia de categorias usada pelo protótipo.

- *Capítulo 5: Experimentação e validação.* São apresentados dois exemplos de classificação distintos (uma factura e uma receita culinária) como forma de demonstrar o processo de classificação. Para finalizar este capitulo é apresentada a avaliação do modelo de classificação e são discutidos os resultados.

- *Capítulo 6: Conclusões e trabalho futuro.* Neste capítulo são alinhadas as ideias enunciadas a retirar deste trabalho. São também discutidos aspectos de desenvolvimento futuro.

Capítulo 2

Estado da Arte

A área da extracção de informação (EI) está em grande expansão e cruza muitos temas como o reconhecimento de padrões, agrupamento, classificação de documentos e extracção de informação. Este capítulo apresenta um conjunto de sistemas que aplicam tecnologias baseadas nestes temas, descreve também trabalhos que serviram de suporte à criação dos classificadores Naïve Bayes e RegEx, criados para o modelo de classificação proposto. Finalmente este capítulo discute a noções teóricas de agrupamento e classificação fundamentais à compreensão do trabalho realizado.

2.1 Trabalhos na Área

IEPAD: Information Extraction Based on Pattern Discovery. O sistema *IEPAD* (Chang and Lui, 2001) tem como objectivo a descoberta automática de regras de extracção de informação de páginas "web". O sistema recebe como entrada uma página HTML e transforma-a num conjunto de símbolos ("tokens") em que cada símbolo é representado por uma sequência de "bits". A descoberta de padrões repetitivos é realizada com recurso a uma estrutura de dados designada por Pat Tree (Morrison, 1968). A Pat Tree é um árvore construida a partir dos símbolos (a trás referidos) e é usada pelo explorador de padrões para identificar os padrões mais frequentes. Este sistema é composto pelos seguintes módulos: um gerador de regras, um visualizador de padrões e um módulo extractor que obtêm a informação desejada com base em paginas "web" semelhantes (de acordo com um critério do utilizador). Este sistema não incorpora intervenção humana e após resultados experimentais concluiu-se que possui uma taxa de 97% de eficiência de extracção em vários motores de busca.

GATE: General Architecture for Text Engineering. O GATE é uma infra-estrutura para desenvolvimento de componentes de software para processamento de linguagem humana (Cunningham, 2000, Cunningham et al., 2008). O GATE define uma arquitectura para processamento da linguagem, fornece também uma plataforma de desenvolvimento de aplicações que um utilizador pode adicionar novas ferramentas. Fornece ainda um ambiente gráfico de desenvolvimento construido sobre a plataforma a cima mencionada.

O Gate pode ser visto como uma arquitectura de software para engenharia da linguagem e é composto por componentes de três tipos:

- Recursos de linguagem (Rl): léxicos, gramáticas, conjuntos de documentos anotados ou ontologias.
- Recursos de processamento (Rp): funções de processamento como "parsers" entre outros.
- Recursos visuais (Rv): ferramentas para visualização e edição de informação.

Os recursos de processamento podem ser combinados de forma sequencial ("pipeline") de modo a formarem aplicações, sendo possível criar combinações em que o uso de alguns recursos é opcional. De entre as inúmeras ferramentas que a plataforma Gate fornece realçam-se as seguintes:

- CREOLE: uma colecção de objectos reutilizáveis para engenharia da linguagem.
- JAPE: um motor de anotação de padrões, determina (com base em expressões regulares) padrões de regras em anotações. O JAPE permite a identificação de expressões regulares em anotações sobre documentos.
- ANNIE: um sistema de extracção de informação que engloba um conjunto de Rp's. Estes recursos usam um conjunto de técnicas para implementar as mais variadas tarefas desde anotação semântica, particionamento de frases ("tokenisation") entre outras.

Semantic Annotation, Indexing and Retrieval. Este trabalho descreve o modo como deve ser construido um sistema que permita anotação, indexação, e recuperação de documentos. Para concretizar os conceitos descritos foi implementado um sistema para extracção de informação e anotação semântica denominado KIM (Kiryakov et al., 2004). A componente principal de extracção de informação desta plataforma é o reconhecimento de *Termos Relevantes* ("named entities") de acordo com uma ontologia

(designada por KIMO). O processamento passa por várias fases que envolvem componentes da plataforma Gate (Cunningham et al., 2002) bem como outras ferramentas de extracção de informação. A meta-data obtida da extracção de informação é guardada num repositório para ficheiros RDF (Sesame (Broekstra et al., 2002)) de onde a informação pode ser interrogada.

Learning Information Extraction Rules. Este sistema aprende regras de extracção de informação aplicando técnicas de programação com lógica indutiva (ILP) a linguagem natural. Para isso é usado um sistema denominado por FOIL (Aitken, 2002) para aprendizagem de relações valor-atributo, o que permite que instâncias destas relações sejam identificadas no texto. Este trabalho explora o problema da aprendizagem de regras de extracção de informação que derivem correctamente de factos que caracterizam o conteúdo de linguagens naturais. As relações aprendidas são aquelas que estão definidas numa ontologia pré-definida para um domínio. A ontologia fornece relações entre classes e sub-classes e define relações entre atributos. A indução de regras é vista como parte de um processo semi-automático que implica acções por parte de humanos. Essas acções passam também por criar um pequeno conjunto de textos marcados ontologicamente que são usados como entrada para o algoritmo. O uso de *ILP* é útil pois fornece uma representação natural das relações a serem aprendidas, além disso este método permite formas alternativas de representar as frases.

Carrot2. Este sistema é um motor de agrupamento automático de resultados de pesquisas na Internet. Esta ferramenta organiza em grupos temáticos ("clusters") e de forma automática resultados de pesquisas feitas nas "web" (Osinski and Weiss, 2005a, Weiss and Stefanowski, 2003). O Carrot2 permite adquirir e agrupar resultados de pesquisas de várias fontes nomeadamente motores de pesquisa como o Google, YahooAPI, Alexa Web Search e o PubMed, entre outros. Apesar de ter sido criado para organizar resultados de pesquisas, esta ferramenta também consegue organizar, em grupos temáticos, documentos de texto fornecido directamente ou através de ficheiros em formato XML.

Para a implementação da ferramenta de definição automática de taxonomias foram utilizados dois algoritmos deste motor de agrupamento (Lingo e STC). Para realizar a tarefa de agrupar tematicamente os documentos optou-se por emular o conteúdo de cada documento num ficheiro XML como forma de gerar os grupos. A concretização desta ferramenta é descrita em pormenor no Capítulo 4.5.

Actualmente estão disponíveis no Carrot2 cinco algoritmos de clustering que são adequados para diferentes tipos de agrupamento. A Tabela 2.1 ilustra algumas características dos algoritmos do Carrot2:

Algoritmo	*Rapidez*	*Hierárquico*	*Detalhe*
FuzzyAnts	**	sim	(Schockaert, 2004)
HAOG-STC	*****	sim	(Gotembniak, 2005)
Lingo	****	não	(Osinski and Weiss, 2005b, Osinski, 2003)
Rough k-means	***	não	(Lang, 2004)
STC	*****	não	(Stefanowski and Weiss, 2003)

TABELA 2.1: Lista de algoritmos suportados pelo Carrot2

Filtragem de correio não solicitado. O classificador Naïve Bayes é bastante utilizado para filtragem de correio electrónico, mas precisamente para filtragem de publicidade ("spam"). No trabalho realizado por (Segaran, 2007) é apresentado um classificador Naïve Bayes que tem como objectivo calcular com base nas palavras de uma mensagem de correio electrónico, a probabilidade dessa mensagem ser, ou não, publicidade. Esse cálculo permite determinar a categoria de um documento porque existem palavras que ocorrem com maior frequência em certos documentos. Por exemplo, mensagens de correio que contenham as palavras *dinheiro* e *jogo* tipicamente aparecem em publicidade a casinos e jogos de azar pelo que um filtro de publicidade deve marcar essas mensagens como publicidade.

Devido à capacidade de classificar um documentos com base no seu conteúdo leva a que o classificador Naïve Bayes tenha sido escolhido para fazer parte do trabalho aqui apresentado. No caso deste modelo de classificação o classificador Naïve Bayes tem a tarefa de classificar documentos e fragmentos de documentos numa de várias categorias conhecidas.

Expressões regulares. No trabalho apresentado por (Turchin et al., 2006) foi examinada a utilidade de expressões regulares para identificar dados clínicos pertinentes para a epidemiologia de tratamento de hipertensão. Para tal projectou-se sistema que emprega expressões regulares para identificar e extrair exemplos documentados de valores de pressão sanguínea e de intensificação de tratamento anti-hipertensivo a partir (do texto) de notas de médicos.

Á semelhança do trabalho a cima descrito, o classificador RegEx também usa expressões regulares para identificar e extrair informação. As expressões regulares são projectadas de modo a identificaram padrões de caracteres, posteriormente um algoritmo criado para este efeito processa o texto de modo a procurar pelos padrões descritos nas expressões regulares e desse modo extrair informação.

2.2 Agrupamento

O agrupamento ("clustering") é o modo mais conhecido de aprendizagem não supervisionada. Este modo de aprendizagem dispensa qualquer tipo de intervenção humana pois não necessita de conhecer *à priori* as categorias dos objectos a classificar. Em vez disso procura estrutura nos elementos de um conjunto de modo a identificar semelhanças entre eles para posteriormente criar sub-conjuntos de elementos "semelhantes".

O agrupamento pode ser descrito como a separação de um grupo de objectos em sub-grupos de objectos semelhantes. Cada grupo ("cluster") é composto por objectos semelhantes entre si e dessemelhantes dos objectos nos outros grupos (Figura 2.1). A representação de dados num reduzido número de grupos torna o processo de agrupamento mais simples mas causa perda de informação na medida em que objectos com características diferentes são agrupados juntos. Desse modo perde-se a noção da diferença entre os objectos e consequentemente perde-se a essa informação, por essa razão a escolha do número de grupos deve ser feita criteriosamente (Berkhin, 2002, Manning et al., 2008).

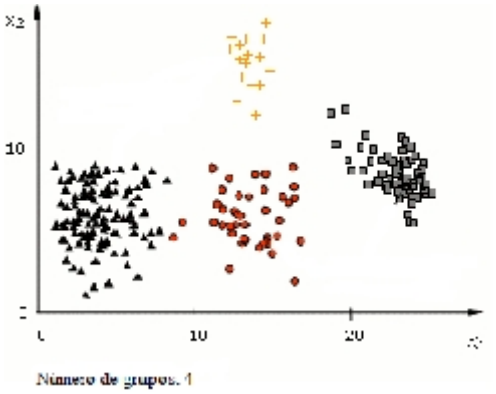

FIGURA 2.1: Exemplo de agrupamento.

A divisão de objecto em grupos pode ser feita essencialmente de dois modos: o modo hierárquico e o modo não hierárquico. Estes dois modos diferem essencialmente na relação entre os grupos e o modo como estes são representados. A Figura 2.2 ilustra os dois tipos de agrupamento e os principais algoritmos.

Agrupamento hierárquico. Neste tipo de agrupamento os grupos têm sempre uma relação de hierarquia entre si que pode ser representada por um diagrama em forma de árvore designado por dendograma. A construção hierárquica de grupos consegue-se através de um processo aglomerativo do tipo base-para-topo ("bottom-up") em que os grupos são unidos aos pares formando grupos maiores até que no topo da hierarquia exista um grande grupo composto por todos os objectos. A hierarquia pode também

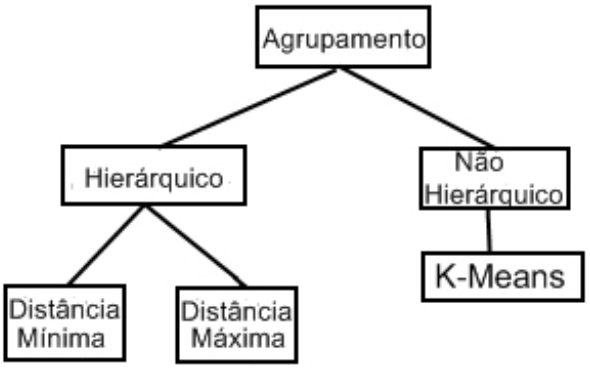

FIGURA 2.2: Taxonomia de técnicas de clustering.

ser construida pelo processo divisivo do tipo topo-para-baixo ("top-down") que é basicamente o processo inverso do processo aglomerativo. Neste método o processo de agrupamento inicia com um grande grupo composto por todos os objectos e divide-os em grupos mais pequenos até cumprir um determinado critério (tipicamente um número máximo de grupos).

Independentemente da abordagem seguida para a criação da hierarquia de grupos é necessário estabelecer uma métrica que indique como os grupos são unidos ou divididos. A maioria dos algoritmos de agrupamento hierárquico funcionam usando variantes dos métodos distância mínima ("single-link") e distância máxima ("complete-link"). No método "single-link" os grupos são criados unindo os grupos cujos elementos estão mais próximos. Devido a este método ser local tem apenas em conta as partes do grupo que estão mais próximas ignorando os elementos do grupo mais distantes. No método "complete-link" o par de cada grupo é escolhido pelo menor diâmetro (ou pela menor distância máxima entre dois grupos) e os grupos produzidos são grupos de elementos mais compactos do que o método "single-link" (Manning et al., 2008).

Agrupamento não hierárquico. Ao contrário do que acontece com o agrupamento hierárquico em que os grupos são relacionados entre si numa organização do tipo hierárquico, no agrupamento não hierárquico os grupos apenas possuem uma relação de distância entre si. Por não ter uma organização hierárquica este tipo de agrupamento é mais simples de realizar o que se reflecte no tempo de criação dos grupos. Existem essencialmente duas vertentes no agrupamento não hierárquico: o "soft clustering" onde um documento não pertence totalmente a um grupo mas tem um grau de pertença a vários grupos, exemplos de algoritmos que apliquem esta técnica são o "Latent semantic indexing" ou o "Fuzzy C means". Na vertente "hard clustering" cada objecto pertence a um e um só grupo, um dos algoritmos que aplicam esta técnica é o k-means.

Para o mesmo número de objectos, n o tempo de criação dos grupos através da técnica de agrupamento simples é de $O(n)$. Na técnica de agrupamento hierárquico o tempo de criação de grupos é, no pior caso, ("complete-link") de $O\left(n^2 \log n\right)$.

k-means. O algoritmo k-means é um algoritmo de agrupamento bastante popular devido à sua simplicidade de funcionamento e resultados de classificação. Apesar de se enquadrar na categoria de aprendizagem não supervisionada este algoritmo necessita de intervenção humana pelo que por vezes é caracterizado como sendo um algoritmo semi-automático. A intervenção humana resulta da incapacidade do algoritmo em definir autonomamente o número de grupos em que os objectos são divididos, assim o humano intervem apenas para definir o número de grupos a construir.

O funcionamento deste algoritmo pode ser descrito nos seguintes passos:

1. Define-se o número de grupos K.
2. Colocar K pontos no espaço de objectos a serem agrupados. Esses pontos representam os centróides dos grupos e podem ser definidos:
   - De forma aleatória;
   - em posições coincidentes com as amostras dos dados.
3. Atribuir cada objecto ao grupo cujo centróide está mais próximo.
4. Quando todos objectos tiverem sido atribuidos aos grupos, recalcula-se as posições dos K centróides.
5. Repetir os passos 3 e 4 até nenhum objecto trocar de centróide ou até se atingido um número máximo de iterações.

O passo 3 apresenta um aspecto importante deste algoritmo. Nesse passo cada documento é atribuido ao centróide mais próximo. A medida de proximidade que pode variar bastante pelo que existem várias métricas sendo as mais comuns: a distância de Manhattan, a distância baseada no valor do co-seno do ângulo entre vectores ou a distância euclidiana entre outras.

Das métricas referidas, as mais populares são: a métrica baseada no valor do co-seno do ângulo entre vectores $sim(\overrightarrow{d_j}, \overrightarrow{d_k}) = \frac{\overrightarrow{d_j} \cdot \overrightarrow{d_k}}{|\overrightarrow{d_j}| \cdot |\overrightarrow{d_k}|}$, e a distância euclidiana $|d_j, d_k| = \sqrt{\sum_{i=1}^{n} (d_{i,j} - d_{i,k})^2}$. Apesar da distância euclidiana ter algumas limitações como o facto de sofrer grande influência da dimensão do documento (os documentos pequenos tendem a ser próximos não pelo conteúdo mas pela sua dimensão), a simplicidade de cálculo da distância euclidiana faz com que seja bastante utilizada.

Lingo. Para além do algoritmo k-means este trabalho usa dois outros algoritmos de agrupamento que fazem parte da ferramenta Carrot2. Um desses algoritmo é o Lingo.

A maioria dos algoritmos de agrupamento segue uma metodologia em que a descoberta dos grupos é feita em primeiro lugar e posteriormente, com base no conteúdo desses grupos, os rótulos de cada grupo são determinados. Tipicamente os grupos são gerados agrupando os elementos que têm características comuns (maior semelhança entre si). Mas por vezes o uso de algumas medidas de semelhança entre documentos não permite que se perceba quais as características comuns entre os documentos que estiveram na origem do seu agrupamento. Para evitar este problema o algoritmo Lingo inverte a metodologia típica, criando primeiro os rótulos dos grupos (perceptíveis para o ser humano) para depois atribuir os documentos aos grupos.

Inicialmente o algoritmo obtem frases (ou termos) frequentes, isto é, são escolhidas as frases que ocorram um numero mínimo de vezes (noção de limiar). As frases escolhidas na fase anterior são usadas na indução dos rótulos dos grupos e posteriormente os documentos são associados ao grupo cujo rótulo melhor caracteriza cada documento. Este processo é implementado em três fases:

1. Construção de uma matriz termo-documento $\mathcal{A}$. Esta matriz é construida com os documentos de entrada para o algoritmo e com todos os termos que neles existam e que cumpram um limite mínimo de ocorrências.

2. Descoberta de conceitos abstractos. Na descoberta dos conceitos abstractos é aplicado o método SVD ("Singular Value Decomposition") sobre a matriz criada anteriormente. O objectivo é encontrar os conceitos abstractos dos documentos de entrada (apenas são usados na fase posterior os k primeiros vectores).

3. Emparelhamento de frases. O emparelhamento de frases e poda de rótulos constitui o último passo onde são descobertas as descrições dos grupos. Nesta fase é usada a distância de co-senos para calcular o quão próximo uma frase frequente está de um conceito abstracto. Da matriz gerada com os valores de similaridade entre estes dois elementos é produzido um conjunto de rótulos para os vários grupos. Uma vez definidos os rótulos dos grupos é usado o modelo vectorial VSM ("Vector Space Model") para atribuir os documentos de entrada aos rótulos dos grupos induzidos na fase anterior (Osinski, 2003).

STC. O algoritmo *Suffix Tree Clustering* (STC) considera a interdependência entre termos e funciona com base no pressuposto que temas comuns são expressos usando frases idênticas. Por exemplo, dois documentos que contenham a frase "o Mercedes é um

grande carro" em principio tratam do mesmo assunto e portanto podem ser agrupados juntos.

A maior vantagem deste algoritmo face a outros que usam apenas a frequência de termos é que normalmente as frases maiores têm maior poder informativo e podem ser usadas directamente para rotular os grupos. O STC está organizado em duas fazes:

1. Descoberta de conjuntos de documentos que partilhem pelo menos uma frase (grupos base).
2. União de grupos em entidades de maior dimensão (grupos finais).

Na primeira fase os grupos base são descobertos através da criação de uma árvore designada por Sufix Tree para todos os termos de entrada. Após a criação da árvore existe um conjunto de frases partilhadas por pelo menos dois documentos. Cada grupo $\mathcal{A}$ é descrito por uma frase m_a e pelo conjunto de documentos que partilham m_a.

Seguidamente é atribuido a cada grupo uma pontuação resultado de uma métrica que envolve o número de termos em m_a (fases pequenas são penalizadas) e o inverso da frequência de cada termo. Apenas os grupos com uma pontuação mínima ("merge threshold") passam à próxima fase, onde o processo de união é uma variação do algoritmo de agrupamento hierárquico aglomerativo (AHC). Seguidamente a pontuação dos grupos é recalculados e os restantes grupos não novamente unidos (Stefanowski and Weiss, 2003).

2.3 Classificação

A classificação documentos de texto é a tarefa de associar uma categoria (de entre várias possíveis) a um documento. A categoria atribuida é aquela que melhor caracteriza o conteúdo do documento e pode ser vista como uma representação descritiva. Mais formalmente a classificação pode ser descrita do seguinte modo: dado as variáveis $\mathcal{D}$ que compõe o domínio dos documentos e $\mathcal{C}$ que consiste no conjunto de categorias pré-definidas, classificar é aproximar uma função desconhecida $\mathcal{G}' : \mathcal{D} \times \mathcal{C} \rightarrow \{\mathcal{T}, \mathcal{F}\}$ (que descreve como os documentos devem ser classificados) através da função $\mathcal{G} : \mathcal{D} \times \mathcal{C} \rightarrow \{\mathcal{T}, \mathcal{F}\}$ chamada de classificador onde $\mathcal{T}$ indica que um documento é classificado com uma dada categoria e $\mathcal{F}$ indica que um documento não é classificado com essa categoria (Sebastiani and Ricerche, 2002).

A função $\mathcal{G}$ atribui a um documento $\mathcal{D}$ uma categoria $\rfloor_i$ e representa o núcleo de qualquer classificador. O modo como um classificador determina a categoria do texto depende muito da implementação mas normalmente passa por aplicar um conjunto de funções

discriminantes $G = \{g_1(x), \ldots, g_c(x)\}$ a uma variável x composta por um vector com as características (propriedades) que descrevem o documento a classificar onde g_i representa a função discriminante que divide o espaço de características $\mathcal{R}_n$ em $\mathcal{C}$ ($g_i : \mathcal{R}^n \rightarrow \mathcal{R}\ i = 1, \ldots, c$).

A Figura 2.3 descreve de modo geral como são atribuidas as categorias a documentos num classificador.

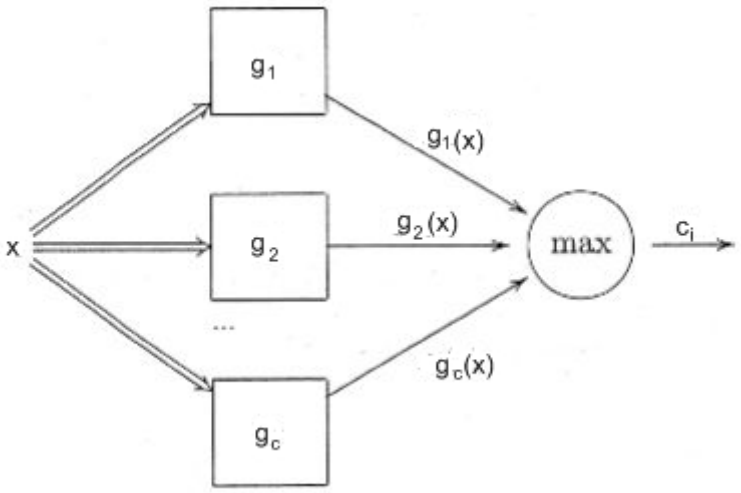

FIGURA 2.3: Modelo canónico de um classificador.

As setas duplas indicam o vector de características multidimensional que é a entrada do classificador, a saída das caixas são os valores das funções discriminantes $g_i(x)$, e a saida do classificador é a categoria a que corresponde a função com o maior valor. Como se pode observar na Figura 2.3 o vector de características de um documento é avaliado por várias funções discriminantes e é classificado por todas. Aquela que possuir maior valor é a categoria que classifica o documento $g_{i*}(x) = \max_{i=1,\ldots,c}\{g_i(x)\}$.

A região de decisão para uma categoria é o conjunto de pontos para os quais a i-nésima função discriminante tem o maior valor. Todos os pontos na região de decisão $\mathcal{R}_i$ são atribuidos a classe c_i. As regiões de decisão são especificadas pelo classificador ou mais precisamente pelas funções discriminantes $\mathcal{G}$. Os limites de uma região de decisão são chamados de limites de classificação e contêm os pontos onde os valores das funções discriminantes são iguais (Kuncheva, 2006).

A classificação de documentos difere do agrupamento essencialmente devido ao tipo de aprendizagem que usa. Enquanto que no agrupamento são usadas técnicas de aprendizagem não supervisionada, na classificação tipicamente usa-se métodos de aprendizagem supervisionada. Considera-se aprendizagem supervisionada quando, existe um conhecimento prévio das categorias usadas na classificação e quando o processo de classificação

é antecedido por uma fase de treino onde o classificador recebe um conjunto de exemplos devidamente classificados com as categorias possíveis. Devido à necessidade de treino a aprendizagem supervisionada necessita de um especialista humano para adquirir a colecção de treino mas também para classifica-la correctamente.

Esta formalização não contempla a definição das categorias pois essa definição é muitas vezes específica de cada problema de classificação. Neste trabalho a definição das categorias é feita numa taxonomia própria (especificada em formato XML, c.f. Capítulo 4.1).

Seguidamente apresenta-se o modelo probabilístico Naïve Bayes no qual se baseia o classificador com o mesmo nome. Este tipo de classificador usa aprendizagem supervisionada e é dos mais conhecidos nas áreas da classificação e reconhecimento de padrões.

Modelo probabilístico Naïve Bayes. O modelo probabilístico aplicado no classificador Naïve Bayes é um modelo baseado na teoria de decisão de Bayes (cf. Apêndice A), que assume a independência das características que definem o objecto a classificar. Uma definição mais completa para este modelo probabilístico seria "modelo de características independentes".

De um modo simplista um classificador Naïve Bayes assume que a presença ou ausência de uma característica de um objecto é independente da presença ou ausência de qualquer outra característica. Por exemplo, uma fruta pode ser classificada como um limão se for amarelo, redondo e tiver mais que dois centímetros de diâmetro. Para um classificador Naïve Bayes estas três características são consideradas independentes e contribuem de modo igual para o cálculo da probabilidade de o fruto ser um limão. Isto pode ser um pressuposto errado porque os três atributos podem depender uns dos outros (por exemplo a cor depender do tamanho do limão).

Apesar do seu desenho simplificado, os classificadores Naïve Bayes funcionam bastante bem em aplicações práticas. Uma das razões da sua popularidade, reside no facto de a colecção de treino necessária para estimar todos os parâmetros necessários à classificação ser relativamente pequena.

De forma abstracta, o modelo probabilístico de um classificador Naïve Bayes é um modelo condicional $\Pr(C|F_1, F_2, \ldots, F_n)$ onde C é o conjunto de categorias, $F_1, \ldots, F_n$ é o conjunto de características. O grande problema deste modelo surge quando o número de características n é elevado ou quando uma característica pode tomar um grande número de valores. Estes acontecimentos podem tornar o cálculo da probabilidade condicional anteriormente referida inviável. Portanto é necessário reformular este modelo de modo

a que seja possível utiliza-lo (Pop, 2006). O teorema de Bayes relaciona probabilidades condicionais e marginais dos eventos estocásticos C e F:

$$\Pr(C|F) = \frac{\Pr(F|C)\Pr(C)}{\Pr(F)} \tag{2.1}$$

onde: $\Pr(C)$ é a probabilidade *à priori* de C; $\Pr(F)$ é a probabilidade *à priori* dos dados de treino F e; $\Pr(F|C)$ é a probabilidade de F dado C. Usando o teorema de Bayes para as várias características variáveis F_n podemos rescrever a equação anterior do seguinte modo:

$$\Pr(C|F_1,\ldots,F_n) = \frac{\Pr(C)\Pr(F_1,\ldots,F_n|C)}{\Pr(F_1,\ldots,F_n)}$$

Na prática apenas estamos interessados no numerador desta fracção, porque o denominador não depende de C e os valores dos atributos F_i são conhecidos, portanto o denominador é constante. O numerador é equivalente à probabilidade conjunta do modelo 2.1 que pode ser reescrito usando repetidas aplicações da definição de probabilidade condicional:

$$\Pr(C, F_1, \ldots, F_n) =$$
$$\Pr(C)\Pr(F_1|C)\Pr(F_2|C,F_1), \Pr(F_3|C,F_1,F_2)\ldots\Pr(F_n|C,F_1,F_2,\ldots,F_{n-1})$$

É neste ponto onde a noção de independência estatística (cf. Apêndice A) deste modelo é aplicada, assumindo que cada característica F_i é independente da característica F_j se $i \neq j$ e $\Pr(F_i|C,F_j) = \Pr(F_i|C)$ o modelo 2.1 pode ser expresso da seguinte forma:

$$\Pr(C|F_1,\ldots,F_n) = \Pr(C)\Pr(F_1|C)\Pr(F_2|C),\ldots = \Pr(C)\prod_{i=1}^{n} Pr(F_i|C)$$

A escolha da categoria correcta depende da função definida do seguinte modo:

$$c = \arg\max_c \Pr(C=c) \times \prod_{i=1}^{n} Pr(F_i = f_i|C = c_i) \tag{2.2}$$

Para realizar classificações, um classificador Naïve Bayes aplica a Equação 2.2 ao vector de características do objecto a classificar para todas as categorias C possíveis. A categoria que tiver maior probabilidade (cf. Figura 2.3) é aquela que caracteriza o objecto.

Capítulo 3

Modelo de Classificação de Documentos

Este capítulo descreve o modelo de classificação de documentos proposto neste trabalho. São apresentadas todas as componentes do modelo assim como as suas extensões, no fim do capítulo é ilustrado, através de um pequeno exemplo, o resultado da classificação de um documento. O modelo de classificação opera sobre os seguintes conceitos base:

- *Documento.* O documento é o objecto a classificar, sendo composto apenas por texto está organizado numa sequência de fragmentos. Assume-se que um documento está sempre dividido em fragmentos e que estes elementos estão separados entre si por, pelo menos, uma linha em branco. Esta divisão é necessária para que exista uma separação entre blocos de informação distintos, não sendo necessária para fragmentos contidos noutros fragmentos.
- *Fragmento.* O fragmento consiste num conjunto de linhas de texto que referem um assunto específico. O conteúdo de um fragmento de texto pode ser dividido noutros fragmentos como forma de detalhar a informação.
- *Termo.* O Termo é o elemento de menor dimensão com que este modelo de classificação trabalha. Cada fragmento é composto por palavras que, após serem sujeitas a uma fase de pré-processamento, ficam com seu conteúdo transformado e dão origem a *Termos*.

Modelo que integra os 3 elementos. A Figura 3.1 ilustra o modo como os três elementos anteriormente descritos se relacionam. O Documento é o elemento que está no topo da hierarquia sendo composto por vários elementos do tipo Fragmento. O elemento

Fragmento pode, ou não, ser composto por outros elementos do mesmo tipo. Deste modo o Fragmento pode ter uma hierarquia interna, sendo o Fragmento no topo dessa hierarquia é designado por Fragmento Raiz e os restantes são designados por Fragmentos descendentes. O Fragmento raiz é aquele que engloba toda a informação sobre um assunto especifico, os restantes fragmentos são aqueles que detalham a informação do Fragmento raiz. Por sua vez cada Fragmento é composto por um ou vários elementos do tipo Termo.

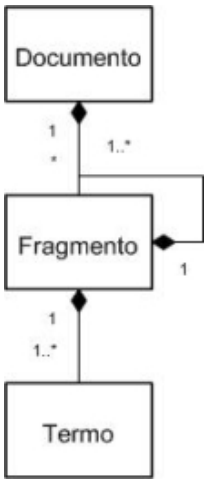

FIGURA 3.1: Relação entre *Documento*, *Fragmento* e *Termo*.

Detalhe sobre o elemento Fragmento. O Fragmentos é o elemento mais importante com que o modelo de classificação trabalha, por essa razão é necessário descrever as suas propriedades. O modelo de classificação atribui a cada fragmento uma categoria como forma de identificar a informação presente no fragmento, permitindo posteriormente a extracção dessa informação. Contudo o conteúdo de um fragmento de texto pode ser dividido noutros fragmentos como forma de detalhar a informação.

Por exemplo, para uma receita de culinária podemos encontrar três zonas distintas:

- A zona onde é feita a listagem de ingredientes.
- A zona onde estão descritos os passos de preparação.
- A zona com concelhos para a confecção.

Cada uma destas zonas é um fragmento diferente, outro exemplo são facturas que normalmente possuem uma zona que identifica o cliente e outra zona que identifica o prestador de serviço.

Para ilustrar como a informação presente num fragmento é refinada através da identificação dos seus fragmentos descendentes, são apresentados dois exemplos. No primeiro

exemplo considere-se que existe uma receita culinária composta por um fragmento denominado por *Lista de ingredientes* e cujo texto se apresenta de seguida.

```
20g sal
1.5Kg carne de porco
Litro Água
```

Este texto apresenta três linhas do fragmento *Lista de ingredientes*, que só por si não ajudam a compreender como esta informação pode ser refinada. Para isso o texto é organizado no formato tabular:

Fragmento	*Quantidade*	*Unidade*	*Componente*
Ingrediente	20	g	sal
Ingrediente	1.5	Kg	carne de porco
Ingrediente	1	Litro	Água

TABELA 3.1: Organização do fragmento *Lista de ingredientes* em formato tabular.

A Tabela 3.1 ilustra como as linhas e colunas podem ser vistas como fragmentos (descendentes) de *Lista de ingredientes*. A *Lista de ingredientes* é composta por um fragmento designado por *ingrediente* (a que correspondem as linhas da tabela). Por sua vez o fragmento *ingrediente* é composto por outros três fragmentos (*Quantidade*, *Unidade* e *Componente*) que, deste modo, detalham ainda mais a informação. Assim o que inicialmente era o conteúdo de apenas um fragmento pode ser estruturado num conjunto de outros fragmentos.

O exemplo seguinte ilustrada um fragmento de uma factura designado por *EletricaTab*. Este fragmento é composto por uma tabela representa a discriminação dos serviços prestados por uma empresa.

Descricao	*Horas*	*Taxa*	*Montante*
Art1	23	12,5	223,98
Art2	2	12,5	45,79

Esta tabela contem duas linhas que representam a discrição dos serviços adquiridos. Tal como no exemplo anterior esta informação pode ser organizada de outro modo.

Fragmento	*Descricao*	*Horas*	*Taxa*	*Montante*
Tuplo	Art1	23	12,5	223,98
Tuplo	Art2	2	12,5	45,79

TABELA 3.2: Exemplo de um fragmento: *EletricaTab*.

A Tabela 3.2 apresenta basicamente a mesma informação o fragmento *EletricaTab* com a diferença que tanto a linhas como as colunas podem agora ser vistas como fragmentos distintos. Cada linha da tabela é vista (do ponto de vista do modelo de classificação) como um fragmento designado por *Tuplo* que pode conter por sua vez outros fragmentos, neste caso são fragmentos que descrevem os serviços prestados (*Descricao*,*Horas*,*Taxa*,*Montante*).

3.1 O modelo de classificação

O modelo de classificação assenta num processo de aplicação sucessiva de classificadores que realizam classificações com base numa taxonomia. O processo de classificação está dividido em três fases distintas que Figura 3.2 demonstra.

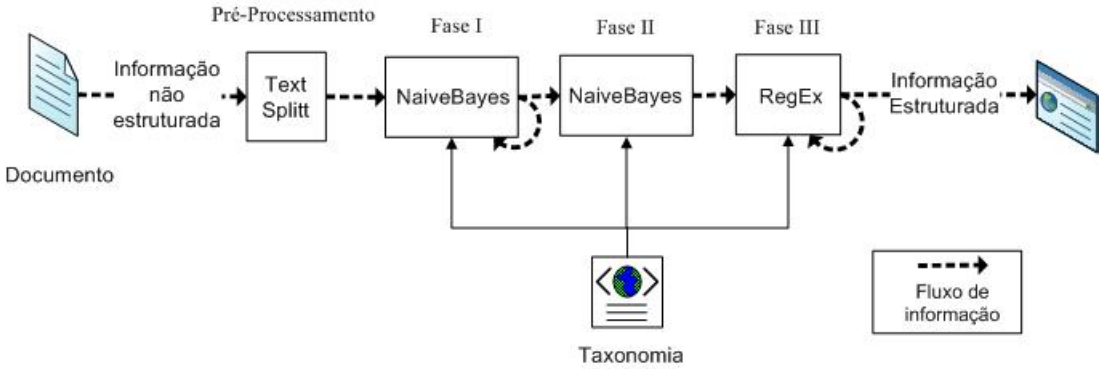

FIGURA 3.2: Processo de classificação.

Como se pode observar o processo de classificação é antecedido por uma fase de pré-processamento do texto (cujo funcionamento será explicado mais a frente). Seguidamente o classificador Naïve Bayes é utilizado na primeira fase de classificação com o objectivo de classificar o documento e na segunda fase de classificação com o objectivo de classificar os fragmentos raiz. A classificação realizada na terceira fase é da responsabilidade do classificador RegEx que classifica os fragmentos descendentes. A Figura 3.2 demonstra que, tanto na fase I como na fase III existe um fluxo de informação que realimenta os respectivos classificadores, este fluxo representa a aplicação sucessiva dos respectivos classificadores sobre os elementos Documento e Fragmento e tem como objectivo refinar a informação. Isto acontece porque o Documento (fase I) pode ser classificado mais que uma vez se a categoria do documento possuir sub-categorias. Nos fragmentos (fase III) a aplicação sucessiva do classificador RegEx advêm da possibilidade de um fragmento possuir outros fragmentos descendentes, deste modo a classificação deve ser realizada para cada um desses fragmentos. A classificação realizada na segunda fase têm um carácter

simples pois a taxonomia de categorias não permite que os fragmentos raiz possuam sub-categorias, assim um fragmento raiz não é classificado mais que uma vez. Esta decisão justifica-se com o facto de a informação resultante da classificação de um fragmento raiz já ser suficientemente refinada. A segunda razão que leva a que os fragmentos não possuam sub-categorias têm a ver o facto de a classificação deste elemento implicar o uso de poucos termos (dado a sua a reduzida dimensão), desse modo a capacidade do classificação Naïve Bayes em distinguir uma de entre várias categorias de fragmentos seria baixa e portanto sujeita a erros. A Tabela 3.3 resume o tipo de classificação realizada por cada classificador sobre o Documento e sobre os Fragmentos.

Elemento-Classificação	*Naive Bayes*	*RegEx*
Documento	Recursiva	———
Fragmento	Simples	Recursiva

TABELA 3.3: Tipo de classificação realizada sobre os elementos Documento e Fragmento.

O processo de classificação ilustrado pela Figura 3.2 é implementado pelo algoritmo que se apresenta (de forma simplificada) de seguida.

1. Obter a lista $\mathcal{L}$ de categorias possíveis para classificação do documento.
2. Classificar o texto do documento numa das categorias da lista $\mathcal{L}$.
3. Preencher, se possível, a lista $\mathcal{L}$ com as sub-categorias da categoria identificada no passo 2. Caso isto não seja possível, continua no passo 5.
4. Repetir o passo 2.
5. Obter a lista $\mathcal{L}'$ de categorias de fragmentos (raiz) que pertençam à categoria do documento identificada no passo 2.
6. Classificar cada fragmento de texto do documento numa das categorias da lista $\mathcal{L}'$.
7. Obter a lista $\mathcal{L}''$ de categorias de fragmentos descendentes daquele identificado no passo 6.
8. Usar as expressões regulares de cada categoria presente na lista $\mathcal{L}''$ para identificar o padrão de cada fragmento de texto.

Para caracterizar a informação presente num documento este algoritmo começa por classificar o documento numa de várias categorias conhecidas à partida. Visto que uma

categorias pode ter sub-categorias o processo de classificação repete-se para essas sub-categorias (passo 3). Após a classificação do documento, o passo seguinte é identificar os blocos de informação presente nesse documento. Esta tarefa é conseguida classificando os fragmentos raiz do documento, sendo que as categorias passíveis de serem usadas dependem da categoria do documento identificada anteriormente. A informação identificada aqui é extraída na fase seguinte (passo 8). A fase final é a extracção da informação de cada fragmento classificado no passo 6 e posterior armazenamento da informação em XML.

O facto do processo de classificação estar dividido em três fases permite que possa ser utilizado com dois propósitos diferentes. As duas primeiras fases correspondem à classificação de documentos pelo que só por si pode ser visto como uma parte de um sistema de gestão documental. Por outro lado as três fases de classificação usadas em conjunto servem então para identificar e extrair a informação transformando-a num formato estruturado.

A taxonomia de categorias. No terceiro passo do algoritmo é feita a procura por sub-categorias. Esta noção está relacionada com o modo como a taxonomia de categorias está definida. Tal como muitas taxonomias criadas para representar as mais variadas situações o dia-a-dia, esta taxonomia possui uma estrutura hierárquica. Neste tipo de organização as categorias do topo da hierarquia são sempre mais genéricas do que as categorias do fundo da hierarquia, assim uma sub-categoria pode ser vista como uma particularização para outra categoria.

Por exemplo, considere-se a factura de compra de um artigo. O facto de o classificador Naïve Bayes classificar um documento com a categoria *Factura*, só por si não é muito interessante. Mais importante do que saber que o documento é uma factura é saber que a factura foi emitida por uma determinada empresa e que corresponde à conta da televisão ou da Internet. Com esse conhecimento é possível obter mais informação (e.g. saber onde se gasta dinheiro).

As entradas na taxonomia de categorias possuem, para além da designação das categorias, o nome do classificador capaz de realizar classificações com essa categoria. Isto é necessário porque existe mais que um classificador a realizar classificações (Naïve Bayes e RegEx), sendo que o classificador RegEx apenas pode ser utilizado em fragmentos.

No caso das entradas da taxonomia de fragmentos existe ainda mais informação. Para além do nome de um classificador, existe, no caso dos fragmentos cujo classificador é o RegEx, a informação de qual a expressão regular que permite extrair a informação especifica da categoria do fragmento. Outra informação presente em cada entrada da

taxonomia de fragmentos é a indicação de qual a categoria de documento a que o fragmento pertence. Isto é necessário porque as categorias de documentos e de fragmentos estão localizadas em ficheiros diferentes e portanto é necessária a indicação de qual a categoria de documento a que o fragmento pertence.

Herança de Fragmentos Tirando partido da organização hierárquica das categorias, este modelo de classificação implementa uma propriedade designada por herança de categorias de fragmentos. Esta propriedade define que sub-categorias de documentos podem herdar categorias de fragmentos.

Isto implica que quando se realiza a classificação de um documento que para o qual a taxonomia de categorias de fragmentos não tenha entradas, seja necessário percorrer a árvore de categorias de documentos até encontrar uma categoria ascendente que possua fragmentos definidos. Se por exemplo, for classificado um documento na categoria Sanduiche (descendente de Receita) e se esta não possuir categorias de fragmentos definidos, o algoritmo procura na categoria ascendente (Receita) pela definição de fragmentos de modo a aplica-los no documento. As Figuras 3.3 e 3.4 ilustram excertos da taxonomia de categorias onde se pode observar as propriedades a cima descritas.

```
<documento tipo="Factura">
    <classificador tipo="BayesClassifier"/>
    <documento tipo="F.TvFio">
        <classificador tipo="BayesClassifier"/>
    </documento>
    <documento tipo="F.TvCabo">
        <classificador tipo="BayesClassifier"/>
    </documento>
    <documento tipo="F.Eletrica">
        <classificador tipo="BayesClassifier"/>
    </documento>
</documento>
<documento tipo="DiarioRepublica">
    <classificador tipo="BayesClassifier"/>
</documento>
```

FIGURA 3.3: Excerto da taxonomia de categorias de documentos.

3.1.1 Pré-Processamento

A fase de pré-processamento do texto é um procedimento opcional dado ser possível a classificação de um documento sem realizar qualquer transformação no texto.

Esta fase consiste em ler o documento a classificar, dividir o seu conteúdo em palavras e finalmente filtrar todas aquelas que não cumpram um conjunto de requisitos. O principal

```
<fragmento nome="TvFioTab" tipo="bloco" objecto="F.TvFio">
<classificador tipo="ClassificadorNaiveBayes" objecto="fragmento"/>
<fragmento nome="tuplo" tipo="tabela">
    <classificador tipo="RegEx">
        <expressao>\d+\s*\w+\s+\d+[,.]\d+\s*\d+[,.]\d+\s*\d+[,.]\d+\s
    </classificador>
        <fragmento nome="Codigo" tipo="atributo">
            <classificador tipo="RegEx">
                <expressao>\d+</expressao>
            </classificador>
        </fragmento>
        <fragmento nome="Descricao" tipo="atributo">
            <classificador tipo="RegEx">
                <expressao>\S+</expressao>
            </classificador>
        </fragmento>
        <fragmento nome="IVA" tipo="atributo">
            <classificador tipo="RegEx">
                <expressao>\d+[,.]\d\d</expressao>
            </classificador>
        </fragmento>
```

FIGURA 3.4: Excerto da taxonomia de categorias de fragmentos.

objectivo desta fase é preparar o texto para ser usado nas fases I e II de classificação, mas tem ainda o objectivo aumentar a rapidez do processo de classificação bem como reduzir a quantidade de memória gasta durante o mesmo. A filtragem de palavras é uma tarefa realizada pelos seguintes filtros:

- *Remoção de sinais de pontuação e transformação para minúsculas*: Aquando da divisão do texto em palavras são removidos todos os sinais de pontuação (?,.:!... etc). Estes sinais de pontuação são removidos porque não contribuem para a classificação do texto.

  As palavras são neste ponto transformadas em minúsculas porque na maioria das vezes, palavras escritas em maiúsculas ou minúsculas tem o mesmo significado. Com esta operação pretende-se reduzir o número de palavras redundantes na classificação.

- *Poda (trimming)*: Apenas as palavras cuja dimensão está entre um número mínimo e máximo de caracteres são utilizadas. O objectivo deste corte é eliminar todas as palavras que não contribuam para a identificação de um tipo de documento. São removidas todas as palavras que por serem muito pequenas são muito comuns e que por esse motivo possuem pouco poder de caracterização de documentos. Por outro lado se as palavras forem exageradamente grandes podem corresponder a erros topográficos. As palavras para passarem por este filtro devem possuir uma dimensão dentro do intervalo $[4, 19]$ caracteres.

- *Remoção de "Stop words"*: As "Stop words" são um conjunto de palavras que são muitas vezes utilizadas na escrita mas não contribuem para a caracterização um documento. Podem ser artigos (e.g. um, uma), pronomes (e.g. tu) entre outros elementos da nossa gramática. Com esta filtragem é reduzido o número de palavras que serão usadas na caracterização dos documentos tornando assim o processo de classificação mais rápido.

- *Remoção de palavras repetidas*: Filtro que remove todas as palavras repetidas. Este filtro é necessário porque o classificador Naïve Bayes usa a noção de conjunto para representar o documento. Como tal na fase de treino cada documento contribui, no máximo, um termo igual para o treino de uma categoria.

Determinar quais as palavras a usar é uma tarefa importante e delicada. A opção de usar apenas termos como elementos que caracterizam o documento a classificar não é simples pois existe a questão de qual a correcta divisão dos termos, quais os sinais de pontuação a incluir, etc. Resumindo, não existe uma solução ideal, existe sim uma solução de compromisso que está sujeita a mudanças.

A fase de pré-processamento tem como objectivo essencial a preparação do texto exclusivamente para o classificador Naïve Bayes porque o texto processado de um modo que deixa de fazer sentido para os humanos. Por essa razão o texto original é preservado para que seja possível extrair a informação de forma correcta na fase III. A preservação do texto original é importante porque a aplicação das expressões regulares só pode ter sucesso se o texto onde estas forem aplicadas cumprir com o padrão para o qual foram criadas. De outro modo a informação extraida estaria errada e não seria perceptível para um humano.

3.1.2 Fase I: Classificação de documentos

Depois de se realizaram as tarefas de processamento do texto, tem inicio a classificação do documento. O processo de classificação inicia-se com a leitura, a partir da taxonomia criada para o efeito, das categorias reconhecidas pelo sistema e uma vez que a cada categoria está associada um classificador eventualmente diferente, apenas são obtidas as categorias cujo classificador é o classificador por omissão. A noção de classificador por omissão é necessária porque o modelo de classificação não está comprometido com um classificador em particular, por essa razão pode existir mais que um classificador que realize a tarefa de classificar documentos. Assim é necessário especificar qual dos classificadores disponíveis é usado.

A tarefa do classificador Naïve Bayes é, com base na formula expressa pela Equação 2.2, calcular a probabilidade do texto pertencer a cada uma das categorias e retornar a categoria que possuir a maior probabilidade. Essa categoria é, do ponto de vista do classificador, a categoria do documento. O que este resultado indica é que com base nos exemplos de treino que o classificador já teve, o conjunto dos termos que constituem o documento a classificar ocorrem com maior frequência na categoria retornada do que nas restantes categorias.

Tendo em conta que cada categoria pode ter sub-categorias, o processo de leitura das categorias (passíveis de classificar o documento) e de classificação de um documento pode repetir-se para todas as sub-categorias. Deste modo um documento pode ter várias classificações (categorias) sendo que a última é aquela que identifica o documento. Cada nova classificação pode ser vista como uma forma de detalhar a informação resultante da classificação anterior.

Actualmente a fase I apenas realiza classificações através do classificador Naïve Bayes, mas este modelo de classificação foi desenhado para permitir a alteração do classificador usado, desde que tenha a capacidade para classificar um documento numa categoria de entre várias. Isso é conseguido através da definição de uma interface comum para todos os classificadores. O classificador que for usado num determinado momento recebe como argumento o documento a classificar e eventualmente as categorias nas quais este deve ser classificado, como resultado o classificador deve devolver sempre uma categoria. No Capítulo 6 discute-se uma alternativa ao Naïve Bayes.

Modelo de um documento. Com a conclusão desta fase é possível saber quais as categorias de fragmentos que fazem parte do documento a classificar. O passo seguinte é obter da taxonomia essas categorias e toda a informação que estas possuem de modo a construir em memória um modelo do documento. Esse modelo que representa o documento a classificar e deve conter, no fim do processo de classificação, a informação existente no documento originam mas de forma estruturada. O modelo do documento é constituido por uma classe *Documento* com um atributo que indica qual a categoria do documento e por um conjunto de instâncias da classe *Fragmento* que representam os fragmentos de um documento de uma determinada categoria.

3.1.3 Fase II: Classificação de fragmentos raiz

Nesta fase são classificados os fragmentos do documento numa de várias categorias, sendo que as categorias possíveis são aquelas identificados aquando da construção do modelo em memória e cujo nome do classificador associado é o mesmo classificador do documento

(Naïve Bayes). Esta tarefa é realizada com o objectivo de identificar informação especifica do documento e de preparar essa informação para a extracção na fase seguinte.

O processo de classificação dos fragmentos tem algumas diferenças do processo de classificação de documentos. A primeira diferença tem a ver com o facto de nos fragmentos a noção de sub-categoria não ter o mesmo significado que nos documentos. No caso dos fragmentos as sub-categorias correspondem a fragmentos descendentes de outros fragmentos e não dizem respeito à especializações das categorias ascendentes. São estas sub-categorias (em particular as expressões regulares que as compõem) que são usados pelo RegEx e permitem a extracção de informação. A última diferença está no processo de avaliação das probabilidades de cada categoria. Ao contrário dos documentos em que o classificador calcula a probabilidade de o texto pertencer a cada categoria, para eleger a mais provável, no caso dos fragmentos existe a possibilidade de mais que um fragmento ser classificado com a mesma categoria pelo que classificador tem uma abordagem um pouco diferente.

Em vez de classificar um fragmento de cada vez, todos os fragmentos do mesmo documento são classificados em simultâneo. O classificador calcula a probabilidade de cada fragmento pertencer a cada categoria e guarda, numa estrutura interna, a informação de qual a categoria com maior probabilidade para cada um dos fragmentos. Em vez de retornar essa informação imediatamente verifica entre todos os fragmentos que estão classificados segundo a mesma categoria quais os que têm maior probabilidade retornando apenas aqueles que possuem maior probabilidade para cada categoria. Por exemplo: os fragmentos *1, 2 e 3* foram classificados segundo as seguintes categorias A, B *e* C com as seguintes probabilidades: $\Pr(1 \in A) = 0.5$, $\Pr(2 \in A) = 0.6$ e $\Pr(3 \in B) = 0.9$. No caso da categoria A, com mais de um fragmento associado (fragmentos 1 e 2), o classificador retorna apenas o fragmento que possui maior probabilidade, neste caso o fragmento 2.

O algoritmo de classificação admite que um documento possua fragmentos que não tenham categorias definidas na taxonomia. O que isto que dizer é que um documento não tem que ter definidas categorias para todos os seus fragmentos de texto. Deste modo não é necessário treinar o classificador Naïve Bayes para reconhecer os fragmentos que, por não possuirem informação relevante, não interessa identificar. Apesar de ser possível não definir as categorias de todos os fragmentos de um documento essa situação deve ser evitada. Isto porque todos os fragmentos de texto são classificados em simultâneo, mesmo aqueles que não possuem categorias, ao entrar no processo de classificação existe a hipótese de serem classificados com uma categoria que não é a sua aumentando a probabilidade de existirem más classificações.

3.1.4 Fase III: Classificação de fragmentos descendentes

As expressões regulares usadas pelo classificador RegEx são basicamente padrões de procura definidos para cadeias de caracteres. Para a criação das mesmas foi analisado um trabalho de Andy Heninger (Heninger, 2004) onde é abordado o modo como as expressões regulares podem ser usadas na análise de texto para procura de palavras chave, extracção de campos, ou ainda na edição ou transformação de texto. Neste documento é discutida a aplicação de expressões regulares em texto "Unicode" (Consortium and Allen, 2006) bem como as melhores abordagens para trabalhar com um grande repositório de caracteres "Unicode". Pelo facto de nos documentos de texto, principalmente em português, existirem inúmeras palavras acentuadas a aplicação de expressões regulares que identifiquem padrões de caracteres "Unicode" é essencial. Assim as expressões regulares criadas para extrair informação dos documentos neste modelo de classificação têm obrigatoriamente de identificar este tipo de caracteres.

O objectivo desta fase é preencher com informação os fragmentos do documento do modelo em memória criado na fase I. Isto é conseguido através da identificação de padrões no texto é da responsabilidade do classificador RegEx cujo modo de funcionamento se descreve de seguida.

Cada categoria atribuida na fase anterior a um fragmento de texto, tem a capacidade de identificar uma zona de texto no documento que trata de um assunto especifico, mas tem ainda outro propósito. Pode ser vista como um invólucro pois agrupa sempre outras categorias de fragmentos cujo classificador é o RegEx. Essas sub-categorias possuem expressões regulares que o classificador usa para identificar e extrair informação. O algoritmo que compõe este classificador usa a expressão regular de cada fragmento para identificar o texto que cumpre um padrão, ao identificar esse padrão extrai o texto e armazena-o no fragmento correspondente (no modelo em memória).

Tal como na fase de classificação de documentos existe aqui um processo de refinamento da informação. Isso é conseguido repetindo o processo de identificação e extracção de informação mas sobre o texto extraido anteriormente. Sobre esse texto aplica-se as expressões regulares de cada fragmento que compõe o fragmento original (ascendente) de modo a obter uma informação cada vez mais detalhada. Este processo repete-se para todos os fragmentos que sejam compostos por outros fragmentos.

Recordando o exemplo demonstrado na Tabela 3.1 onde existe um fragmento chamado *Lista de Ingredientes* identificado com recurso ao classificador Naïve Bayes. Observa-se que dentro desse fragmento existe outro designado por *ingrediente*, que possui uma expressão regular que permite identificar uma linha com a descrição de um ingrediente (quantidade, unidade, componente). Este fragmento *ingrediente* é composto por outros

três fragmentos cujas expressões regulares permitem detalhar a informação do fragmento *ingrediente* (diminuindo o nível de granularidade da informação).

Na maioria das situações a expressão regular de um fragmento que contem outros fragmentos pode ser a agregação das expressões regulares dos fragmentos descendentes. Isto torna a definição dessas expressões regulares um processo mais simples, no entanto em algumas situações é útil trabalhar mais a expressão regular, pois esta pode ajudar na identificação de padrões no texto.

3.1.5 Persistência do resultado da classificação

Após a última fase de classificação existe uma representação do documento original mas em que a informação está estruturada e que necessariamente deve ser transformada num formato persistente que qualquer computador possa processar. Foram analisados dois formatos para a definição das taxonomias de categorias e para persistência da informação obtida no processamento dos documentos, esses dois formatos são o XML e o RDF.

XML. O XML é um formato "standard" que permite organizar a informação de modo estruturado através da definição hierárquica de marcas (Team et al., 2001). Graças a estas características é possível apresentar a informação contida nestes documentos de várias formas recorrendo ao XSLT, no entanto as regras de apresentação estão comprometidas com o XML devido às expressões XPath que apenas fazem sentido para o XML para o qual foram criadas. Para além desta limitação existe ainda outra, o reduzido número bases de dados de suporte nativo ao XML (e.g. eXist).

RDF. O RDF foi criado tendo como principal objectivo criar um modelo simples de dados, com uma semântica formal (Antoniou and van Harmelen, 2004). Este formato apresenta uma capacidade de modelação do conhecimento abstracta não ficando comprometido com a forma como este é representado. No RDF existem repositórios de meta-dados que permitem armazenar este tipo de ficheiros bem como realizar interrogações realizadas sobre uma modelação pouco comprometida com o formatado onde a informação está armazenada. Por outro lado a apresentação da informação em formato HTML, é possível graças à representação XML, no entanto sofre do mesmo problema que o XML na medida em que as regras de apresentação estão comprometidas com o XML gerado. A Tabela 3.4 apresenta a avaliação feira das características dos dois formatos.

	XML	*RDF*
Capacidade de modelação abstracta	+	++
Representação da modelação	+	++
Capacidade de interrogação da info.	+	+
Apresentação da Informação	++	-

TABELA 3.4: Características dos formatos XML e RDF

Tendo em conta as características de ambos os formatos optou-se pelo XML devido à melhor capacidade de apresentação da informação. Mas este não é um assunto encerrado e o uso do RDF continua a ser uma hipótese a considerar.

O processo de transformação da informação para XML é iterativo, o elemento *Documento* (da representação em memória) dá origem a um novo documento XML em que raiz do documento é a categoria com que foi classificado o documento. Cada fragmento em memória vai dar origem a uma marca XML, cujo nome é a categoria com que o fragmento foi classificado na fase II e o conteúdo é a informação extraida na fase III. De seguida é apresentado os excertos de uma receita culinária da categoria *PratosMundo* e do ficheiro XML que resulta da classificação:

Transformação da informação. Com a produção de um documento XML o processo de transformação da informação termina, mas graças à natureza estruturada deste formato é possível transformar o seu conteúdo de várias formas de modo a melhorar a apresentação da informação obtida. Este trabalho integra um pequeno modulo cuja função é transformar em páginas HTML a informação sintetizada no XML. Deste modo é possível apresentar num "Browser" a informação importante numa forma agradável para o utilizador.

A transformação da informação para HTML é conseguida através da aplicação de regras de transformação expressas na linguagem XSLT. O processo de transformação consiste em usar um processador de XSLT que recebe o ficheiro com regras de transformação e o ficheiro XML a transformar para produzir HTML.

Para realizar a transformação é necessário que exista um ficheiro de regras para cada categoria de documento e que esta possua categorias de fragmentos. Apenas faz sentido a existência dos ficheiros de regras para as categorias de documentos que possuam fragmentos, pois as regras de transformação insidiem sobre a informação presente nestes elementos. É importante realçar que estas regras de transformação estão profundamente comprometidas com o formato com que a informação foi armazenada (XML), qualquer alteração a esse formato implica igualmente alterar as regras de transformação.

```
Ingredientes:
1 litro de água
pimenta doce
manteiga

Modo de Fazer:
Tempere a carne com o sal e a pimenta doce.
Leve ao fogo para cozinhar.
Misture, despejando a coalhada, até ferver.
Sirva com arroz.
```

(a) Receita Culinária

```
<PratosMundo>
<Lista_de_ingredientes conteudo="">
  <ingrediente conteudo="1 litro de água">
      <quantidade conteudo="1"/>
      <componente conteudo="litro de água"/>
  </ingrediente>
  <ingrediente conteudo="pimenta doce">
      <componente conteudo="pimenta doce"/>
  </ingrediente>
  <ingrediente conteudo="manteiga">
      <componente conteudo="manteiga"/>
  </ingrediente>
</Lista_de_ingredientes>
<Passos_de_preparacao conteudo="">
  <passo conteudo="Tempere a carne com o sal e a pimenta doce"/>
  <passo conteudo="Leve ao fogo para cozinhar"/>
  <passo conteudo="Misture, despejando a coalhada, até ferver"/>
  <passo conteudo="Sirva com arroz"/>
</Passos_de_preparacao>
</PratosMundo>
```

(b) Resultado Classificação

FIGURA 3.5: Transformação de uma receita

3.2 Funcionalidades do modelo de classificação

Este modelo de classificação possui um conjunto de funcionalidades que permitem melhorar a eficiência de classificação e aumentar a velocidade de processamento dos documentos, essas funcionalidades são descritas nesta secção.

Aprovação do Utilizador. Esta funcionalidade (disponível apenas na fase I de classificação) consiste em dar ao utilizador a capacidade aprovar ou rejeitar uma classificação de um documento. Sempre que é realizada a classificação deste elemento, o utilizador é interrogado para indicar se concorda ou não com a classificação. Caso não concorde este deve escolher a categoria correcta de entre as categorias possíveis para o documento em questão. Essas categorias são as mesmas que o classificador utilizou inicialmente para determinar a categoria errada, não sendo possível ao utilizador indicar uma categoria que não esteja definida na taxonomia. Deste modo o utilizador possui maior controlo sobre a classificação dos elementos a classificar aumentando assim a eficiência da classificação.

Apesar de também ser possível implementar esta funcionalidade na terceira fase de classificação foi decidido não o fazer. Esta decisão justifica-se porque dificilmente um utilizador a usaria uma funcionalidade em teria que seleccionar manualmente, para cada fragmento no documento, o texto correcto. Esta tarefa é pouco prática e extremamente aborrecida levando ao desinteresse. Esta funcionalidade apesar de útil pode ser entediante pelo que o seu uso é opcional deixando ao critério do utilizador a activação desta funcionalidade.

Treino do classificador. Posto que o tipo de aprendizagem do classificador Naïve Bayes é supervisionada, é necessária a existência de uma fase de treino para garantir o correcto ao funcionamento do classificador. Desta necessidade surgiu uma nova funcionalidade, o *Treino do classificador*. Esta funcionalidade consistem em usar a classificação de novos documentos para realimentar o classificador. O uso das classificações correctas para continuar o treino possibilita que, ao possuir mais exemplos de treino, o classificador realize melhores classificações.

O treino do classificador consiste em fornecer ao classificador o texto a classificar, assim como a categoria correcta. Esta funcionalidade pode ou não ser usada em conjunto com a aprovação da classificação, ficando ao critério do utilizador a utilização de ambas. Ao ser usada em conjunto com a aprovação o treino do classificador abre a porta a uma outra hipótese. Para além de aumentar o treino com a aprovação de uma classificação é possível também realizar o processo inverso ao treino. O processo inverso ao do treino consiste em destreinar o classificador indicando o texto e a categoria que devem ser removidos do treino. Ao passar para o utilizador a capacidade de treinar o classificador, o utilizador ganha também uma grande responsabilidade, porque um mau treino pode resultar em más classificações.

Persistência. O uso de um classificador como o classificador Naïve Bayes pode apresentar alguns problemas à aplicação prática de um sistema de classificação. O facto deste classificador necessitar de treino prévio à realização de qualquer classificação, pode ser problemático na medida em que o treino normalmente consiste na leitura de ficheiros do disco rígido de um computador. E como se sabe o acesso ao disco é um dos processos mais demorados num computador. Assim os classificadores deste tipo podem eventualmente ser lentos no inicio de funcionamento se a colecção de ficheiros de treino for grande. O que não sendo eventualmente um problema muito grave pode comprometer o desempenho global de uma aplicação.

A solução que este trabalho propõe consiste em armazenar em disco a instância do classificador quando este já tiver completado a fase de treino. Esta tarefa (também chamada de "serialization") consiste em transformar a instância do classificador em memória numa

sequência de "bytes" de modo a que esta possa ser armazenada em disco. A instância do classificador contêm basicamente um conjunto de estruturas de dados com os termos "treinados" e as respectivas categorias. Este processo dá a possibilidade ao algoritmo de classificação de ser mais veloz na iniciação do classificador da fase I porque, ao escolher a instância do objecto classificador armazenada em disco, não necessário realizar acessos a disco para o treino. O uso de persistência do classificador torna também possível o uso da funcionalidade de *treino do classificador*. Pois sem a persistência, o treino do classificador com os novos documentos seria uma tarefa inglória na medida em que assim que a aplicação terminasse o treino perdia-se.

Esta solução tem vantagens mas também podem ser discutidas desvantagens de entre as quais o espaço ocupado pelo objecto armazenado, é preciso pensar se o ganho em desempenho compensa o espaço em disco ocupado pela colecção de documentos de treino original mais o objecto armazenado. Como este modelo de classificação não foi testado com um elevado número de documentos, não é possível realizar uma comparação baseada em dados recolhidos de experiências. Mas ainda assim é possível conjecturar que o espaço ocupado não será insustentável tendo em conta a capacidade dos discos modernos. Se for tido em conta o facto de o objecto em disco ter basicamente a mesma informação que a colecção de treino, manter essa colecção de documentos pode não ser necessário. Outro aspecto a ter em conta é o facto de o objecto em disco ter sempre menor dimensão do que a colecção de documentos original porque os documentos antes de serem utilizados no treino são sujeitos a um pré processamento que lhe diminui a dimensão.

3.3 Extensões ao modelo: Definição de Taxonomias

No Capítulo 4.1 são descritas as categorias de documentos e fragmentos definidas para testar a concretização deste modelo de classificação. Tanto a definição das categorias e como das expressões regulares dos fragmentos foi realizada pela mesma pessoa que implementou o protótipo deste modelo. Mas normalmente essas definições seriam feitas por alguém especialista no meio onde este modelo de classificação fosse aplicado. Por outras palavras, um técnico que conheça a fundo os documentos a classificar. Para auxiliar a definição das categorias da taxonomia projectaram-se duas ferramentas para complementar este modelo de classificação:

- *Definição manual*: Esta ferramenta permite ao utilizador definir manualmente as categorias e a sua organização hierárquica, este utilizador deve portanto conhecer bem os documentos a caracterizar assim como deve ter conhecimentos que permitam a construção de expressões regulares para a identificação de informação presente nos fragmentos.

- *Definição automática*: Nesta extensão a definição das categorias é feita de modo semi-automático com recurso a uma adaptação do sistema Carrot2 para aplicação de três algoritmos de agrupamento (k-means, Lingo e STC). Após a definição automática das categorias um utilizador pode corrigi-las se necessário.

O modo com as duas ferramentas de criação de taxonomias se relaciona com o processo de classificação é demonstrado pela Figura 3.6.

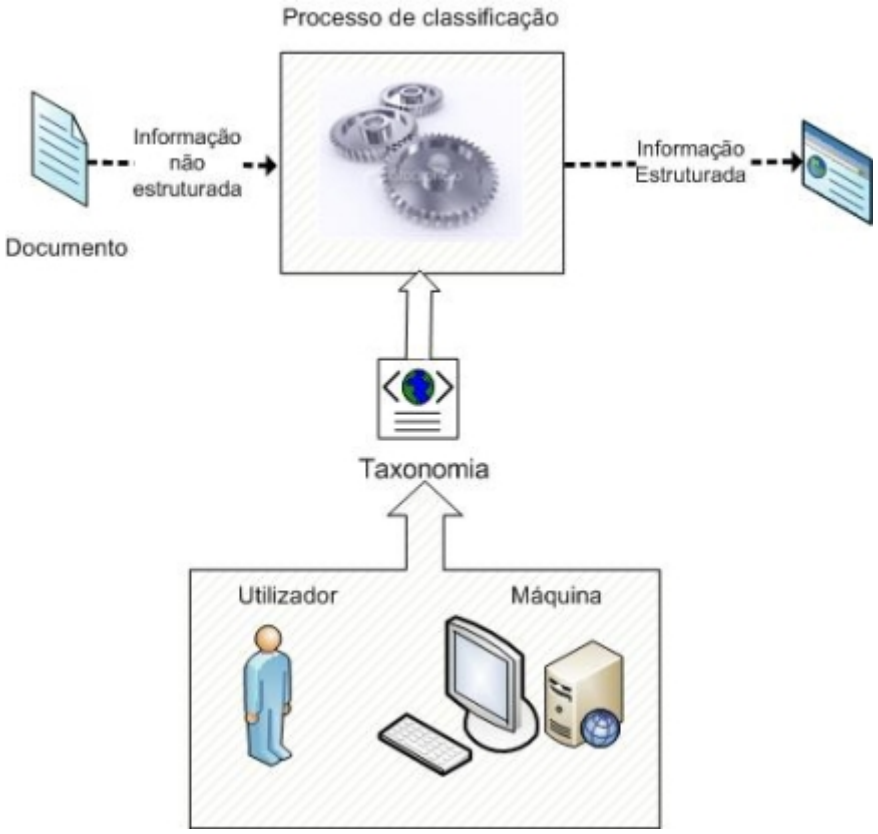

FIGURA 3.6: Arquitectura do modelo de classificação.

Capítulo 4

Concretização

Este capítulo apresenta a taxonomia de documentos e fragmentos utilizada na concretização do modelo de classificação. É também apresentada à implementação do protótipo criado para demonstrar a aplicação do modelo de classificação proposto no capítulo anterior.

4.1 Taxonomia de categorias

A par do protótipo do modelo de classificação foi também desenvolvida uma taxonomia de categorias com base na qual o protótipo realiza classificações. A taxonomia está dividida em dois documentos XML distintos, um para categorias de documentos e outro para categorias de fragmentos, esta divisão foi feita apenas com o objectivo de tornar mais fácil a leitura da taxonomia. A escolha deste formato de documento recaiu no XML pelas mesmas razões invocadas na escolha do formato para persistência da informação, descritas no Capítulo 3.2.

A taxonomia de categorias define as designações das categorias de documentos e de fragmentos, define também qual o nome do classificador que consegue processar cada documento/fragmento. Esta indicação é necessária para que seja possível o uso, por parte do modelo de classificação, de mais que um classificador em simultâneo, isto é, para possibilitar que numa mesma taxonomia estejam definidas várias categorias de documentos que são utilizadas por classificadores diferentes.

Nas categorias de fragmentos a noção de classificador é ainda mais importante porque os fragmentos podem ser processados tanto pelo classificador Naïve Bayes como pelo classificador RegEx consoante a fase de classificação e o tipo de fragmento. A Figura

4.1 ilustra quais as categorias de documentos que a taxonomia define bem como o modo como estão organizadas.

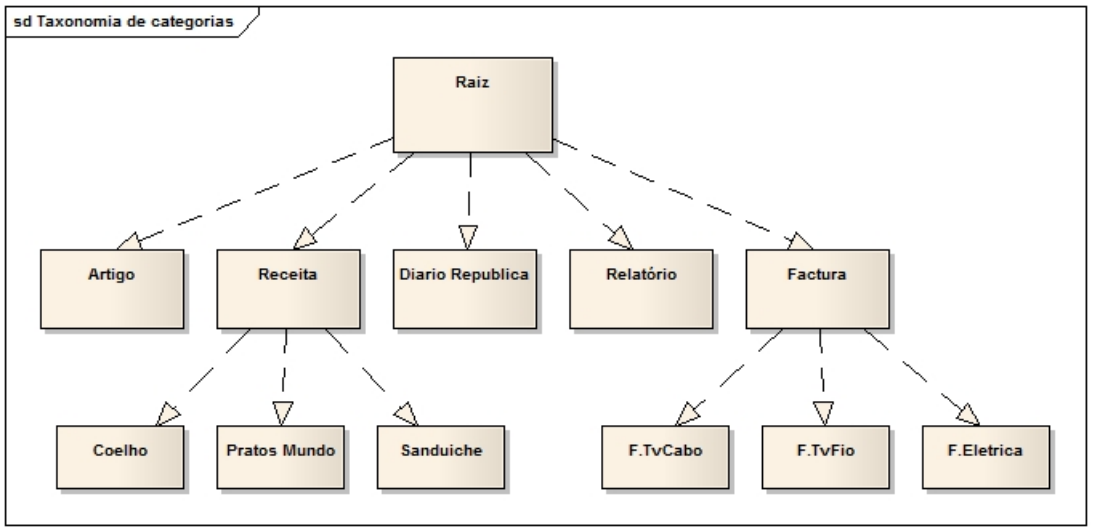

FIGURA 4.1: Taxonomia de Documentos.

Como se pode observar na Figura 4.1 existem dois conjuntos de sub-categorias. Foram escolhidas apenas dois conjuntos porque não se considera necessário criar uma grande árvore de categorias para demonstrar a aplicação do modelo. Estes conjuntos de sub-categorias tentam ser representativas da noção de sub-categoria.

As categorias de fragmentos são as únicas (até este momento) que podem ser usadas por dois classificadores diferentes, isso mesmo é demonstrado na Figura 4.2 onde se observa todas as categorias de fragmentos.

Nesta figura apenas estão representadas as categorias de documentos que possuem fragmentos (*Factura* e *Receita*), as restantes categorias não possuem categorias de fragmentos e por isso apenas podem ser usadas na fase I de classificação. Outro aspecto importante desta imagem é a notação das categorias de fragmentos para os fragmentos de *Factura*. Devido a todas as especializações possuirem duas grandes categorias de fragmentos (informação "inline" e informação tabular) apresenta-se, de modo a simplificar a leitura, os nomes com das categorias *Info* e *Tab* antecedidas pelo o caracter *. Esse caracter representa o nome da categoria ascendente (e.g. F.EletricaTab, F.TvFioTab, etc). Na Figura 4.2 observa-se que, ao contrário da categoria *Factura* em que são as sub-categorias possuem fragmentos, a categoria *Receita* possui fragmentos associados directamente a si a não as categorias descendentes (cf. Secção 3.1.3).

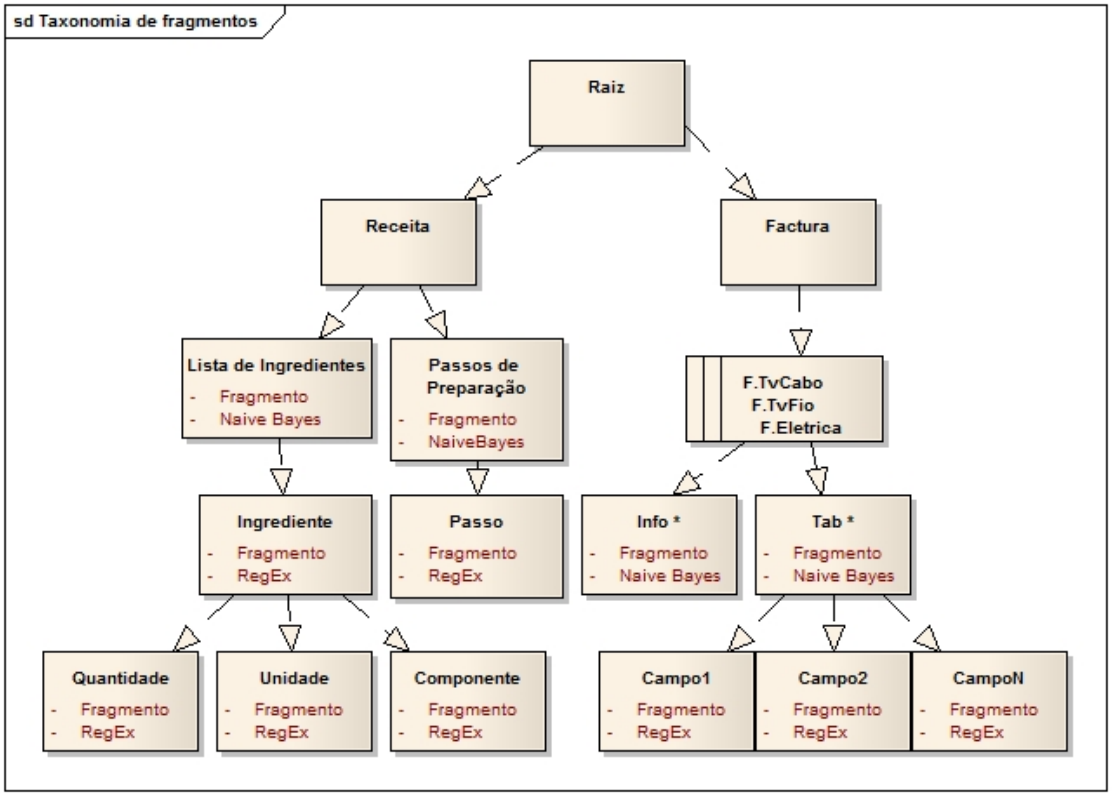

FIGURA 4.2: Taxonomia de Fragmentos.

4.2 Algoritmo de classificação

A classificação de documentos é realizada pelos classificadores Naïve Bayes e RegEx, mas tão importante como os classificadores é o algoritmo que os aplica de forma a caracterizar os documentos incrementalmente. O algoritmo 1 é a concretização daquele apresentado no Capítulo 3, nesta concretização está demonstrada a característica de aplicação sucessiva de classificadores como forma de refinar a informação. Nas linhas 3 e 4 o documento é sucessivamente classificado até se obter a designação que melhor identifica o documento. Seguidamente são classificados os fragmentos (linha 8) de modo a identificar blocos de informação dentro do documento. Finalmente na linha 15 é extraída a informação identificada na linha 8.

Ambos os classificadores usados neste algoritmo implementam uma interface comum de modo a uniformizar o código e a compromete-lo o menos possível com um classificador em particular. As secções seguintes tratam de apresentar a implementação realizada destes dois classificadores.

Algoritmo 1 Core algorithm

```
1:  procedure CLASSIFICA(text, doc)
                                                          ▷ Fase I
2:      cats ← LoadCategories()
3:      while doc_cat ← Classify(text, cats) do
4:          cats ← LoadCategories(doc_cat)
5:      end while
                                                          ▷ Fase II
6:      frag_cat ← LoadFragmentCategories(doc_cat)
7:      frag_text ← GetFrags(doc)
8:      Classify(frag_text, frag_cat)
                                                          ▷ Fase III
9:      if hasFrags(doc_cat) then
10:         frag_def ← LoadFragment(doc_cat)
11:     else
12:         frag_def ← LoadAncestorFragment(doc_cat)
13:     end if

14:     for frag ← frag_text do
15:         Classify(frag, frag_def)
16:     end for
17: end procedure
```

4.3 Concretização do classificador Naïve Bayes

A escolha deste classificador advêm da necessidade de organizar documentos segundo categorias e portanto o classificador Naïve Bayes foi escolhido porque permite a classificação de documentos com base no seu conteúdo, ou seja, com base nos termos de um documento é possível determinar a probabilidade de um documento pertencer a uma determinada categoria. Este classificador usa a noção de termo como elemento de caracteriza o documento e quando tal acontece assume-se que existem termos com maior probabilidade de ocorrerem em determinados tipos de documentos do que em outros tipos de documentos.

Este classificador tenta resolver o problema de determinar qual a categoria que melhor caracteriza um documento, isto é, eleger uma categoria sabendo que um documento tem um dado texto. Tendo em conta que este é um classificador estatístico, retorna apenas uma estimativa (probabilidade) de qual a categoria que melhor caracteriza o documento. Este problema pode ser traduzido pela probabilidade condicionada $\Pr(Categoria|Documento)$, mas dado que não é possível realizar este cálculo de forma directa pelo que é necessário recorrer ao teorema criado pelo matemático Thomas Bayes(cf. Apêndice A). Para realizar o cálculo da probabilidade pretendida, é necessário realizar o cálculo de três outras probabilidades expressas pela Equação 4.1.

$$\Pr(Categoria|Documento) = \frac{\Pr(Documento|Categoria) \times \Pr(Categoria)}{P(Documento)} \quad (4.1)$$

A probabilidade $\Pr(Documento|Categoria)$ obtem-se calculando "a probabilidade de um documento ocorrer dado que sabemos que ocorre uma categoria em particular". Esta probabilidade é calculada com base em informação estatística resultante do treino do classificador. A probabilidade $\Pr(Categoria)$ é a probabilidade de um documento escolhido aleatoriamente pertencer a uma categoria, portanto o seu cálculo é realizado dividindo o número de documentos numa categoria pelo número total de documentos. Por sua vez a probabilidade $\Pr(Documento)$ é desprezável pois o seu valor é igual independentemente da categoria.

Núcleo do classificador. O Núcleo do classificador Naïve Bayes é composto por duas variáveis responsáveis por armazenar informação estatística necessária ao cálculo de probabilidades. A variável *fc* ("feature count") é uma lista de todos os atributos treinados pelo classificador. Cada atributo por sua vez possui, outra lista interna com todas as categorias a que este já foi associado bem como o número de vezes que essa associação foi feita. Para o preenchimento da variável *fc* é necessário implementar um método que apesar de simples tem grande importância pois é responsável por dividir um documento em termos, as operações realizadas por este método são descritas no Capítulo 3.1.1. A Tabela 4.1 ilustra um exemplo de informação presente nesta variável.

	Termo		
Categoria	*bola*	*pé*	*mão*
bom	6	3	0
mau	0	3	6

TABELA 4.1: Representação da lista *feature count*.

A variável *cc* ("category count") funciona como um dicionário para o número de vezes que cada classificação já foi utilizada, isto é, corresponde ao número de documentos classificados com cada classe.

Probabilidade de um atributo. Posto que é possível determinar a frequência com que cada termo ocorre em cada categoria é necessário converter esses números em probabilidades. Uma probabilidade é um número (que varia entre 0 e 1) indicativo da possibilidade de um evento ocorrer sendo que 1 indica que o evento ocorre sempre e 0 indica que o evento nunca ocorre.

Neste classificador a probabilidades de um termo pertencer a uma determinada categoria é calculada dividindo o número de vezes que esse termo ocorre em documentos dessa categoria pelo número total de documentos classificados com essa categoria. Para isso criou-se um método para realizar esta mesma operação denominado *featureProbability*. Ao valor retornado por este método é chamado de probabilidade condicionada, relembrando a Tabela 4.1 a probabilidade de ocorrência do termo *pé* sabendo que a categoria é *Bom* é $\Pr(pé|bom) = 0.5$.

O método *featureProbability* devolve um resultado preciso para os termos e classificações que já viu até agora, mas existe um problema derivado da utilização apenas da informação que já viu até este ponto. Isto torna o classificador extremamente sensível a termos que raramente tenha processado. Esta situação pode levar a erros de classificação porque um termo que tenha sido classificado segundo uma categoria num número reduzido documentos, não implica que esse termo pertença sem sombra de dúvidas a essa categoria e não a outra, o classificador simplesmente não consegue fazer melhor. De modo a minorar este problema criou-se uma solução que passa por atribuir uma probabilidade padrão a cada categoria e à medida que mais e mais termos são classificados segundo uma categoria específica a probabilidade de um termo pertencer a uma determinada categoria aproxima-se mais de 1 ou 0. A probabilidade padrão para cada categoria é um valor tipicamente consensual, 0.5. É também necessário decidir qual o peso que a probabilidade padrão deve ter no cálculo de uma probabilidade. Um peso inferior a 1 faz variar as probabilidades mais bruscamente pois a probabilidade assumida fica com menos importância.

A facilidade de configuração estes parâmetros é uma vantagem deste classificador pois torna-o flexível e se os parâmetros forem bem "afinados", pode-se conseguir melhores classificações. Criou-se então um método que devolve uma probabilidade calculada segundo os requisitos anteriores (*weightedProbability*). Para demonstrar com as probabilidades evoluem com o treino vamos supor o seguinte exercício: A colecção de treino é composta por um documento A que contem o termo *dinheiro*; o documento é classificado com a categoria *bom*. Se realizarmos o treino do classificador com base nesta colecção e invocarmos por três vezes o método de cálculo *weightedProbability* com os parâmetros padrão (1 para o peso e 0.5 para probabilidade padrão), observamos os seguintes resultados:

```
Primeira iteração:
A probabilidade da palavra dinheiro ocorrer dado que a categoria é bom é: 0.25%
Segunda iteração:
A probabilidade da palavra dinheiro ocorrer dado que a categoria é bom é: 0.16%
Terceira iteração:
A probabilidade da palavra dinheiro ocorrer dado que a categoria é bom é: 0.12%
```

Com o recurso a esta técnica consegue-se que a probabilidade de um termo pertencer a uma dada categoria evolua mais gradualmente, o que reforça ainda mais a importância do treino.

Probabilidade de um documento. Uma vez calculadas as probabilidades dos termos pertencerem a uma determinada categoria deve-se combinar essas probabilidades de modo a obter a probabilidade de o documento inteiro pertencer a essa categoria. Para realizar essa tarefa implementou-se uma classe denominada por *classificadorBayes*.

Tendo em conta que este classificador assume que as probabilidades dos termos são independentes podemos calcular a probabilidade de um documento inteiro multiplicando as probabilidades de todos os seus termos. Como forma de demonstrar o cálculo da probabilidade de um documento vamos supor o seguinte exercício: Existe um documento A composto por apenas dois termos {*Sporting* , *Benfica*}. Sabendo que a probabilidade de ocorrer o termo *Sporting* dado que a categoria é *bom* é de 0.8% e que a probabilidade de ocorrer o termo *Benfica* dado que a categoria é *bom* é de 0.2% podemos chegar à conclusão que a probabilidade ocorrer o documento A dado que a categoria é bom é de $\Pr(A|Bom) = 0.8 \times 0.2 = 0.16$.

Para realizar esta operação implementou-se um método que devolve a probabilidade de um documento ocorrer dado uma categoria. Este método multiplica as probabilidades de cada um de modo a obter uma probabilidade global. Neste ponto é possível calcular a probabilidade $\Pr(Documento|Categoria)$. Relembrando a Equação 4.1 conclui-se que as duas primeiras parcelas já são calculáveis e tendo em conta que podemos desprezar a última o cálculo da probabilidade $\Pr(Categoria|Documento)$ torna-se possível.

Assim termina a tarefa do classificador de Bayes, cabe então ao algoritmo de classificação obter a probabilidade de o documento pertencer às restantes categorias possíveis e com base nessas probabilidades classificar o documento.

Classificação. A etapa final da classificação de um documento é a escolha de uma categoria. Esta tarefa é realizada pelo algoritmo de classificação que deve eleger a categoria que melhor caracteriza o documento deverá ser a categoria com maior probabilidade, mas a simples atribuição da categoria ao documento com base na maior probabilidade pode não ser a melhor opção. Podem existir situações em que decidir erradamente em favor de uma categoria tem consequências mais graves do que decidir erradamente a a favor de outra categoria. Assim existem categorias mais importantes e nesses casos a classificação deve ser feita com um grau elevado de certeza. Do mesmo modo decidir em favor de uma categoria que possui uma probabilidade marginalmente superior a outra é

arriscado. Para lidar com estas situações o classificador deve especificar o quão maior deve ser uma probabilidade em relação a outra. Esta regra designada por *regra do limiar* é descrita do seguinte modo: Sejam duas categorias ω_i e ω_j, com os respectivos valores de probabilidade, seja também α um valor real qualquer. Assumindo que um categoria tem uma probabilidade de ocorrência superior que à outra. A categoria escolhida para classificar o documento é:

$$\begin{aligned} &\text{se } \Pr(\omega_i) \geq \alpha \Pr(\omega_j), \text{ decidir } \omega_i \\ &\text{se } \Pr(\omega_j) \geq \alpha \Pr(\omega_i), \text{ decidir } \omega_j \\ &\text{caso contrário não decidir nada.} \end{aligned} \tag{4.2}$$

Método de cálculo. Devido ao cálculo da probabilidade global do documento resultar na realização de muitas multiplicações de valores menores ou iguais a um, existe o risco de o resultado ser tão pequeno que os computadores deixam de ter capacidade para representar um valor tão pequeno, nesses casos o valor é zero. Esta situação também conhecida como "floating point overflow" é um problema na medida em que um documento que pertença a uma categoria pode ter probabilidade 0% nessa categoria. Este problema pode acontecer se o documento for composto por um elevado número de termos e pode levar a que a eficácia de classificação seja comprometida, no entanto esta situação não ocorre com tanta frequência nas classificações em que a dimensão dos documentos é pequena como por exemplo na filtragem de "spam".

Para lidar com esta situação este protótipo implementa uma segunda uma versão do classificador Naïve Bayes. Nessa implementação a multiplicação de probabilidades foi substituida pela soma de logaritmos de probabilidades (Manning et al., 2008). Este método de cálculo é válido pois baseia-se numa propriedade matemática dos logaritmos. Essa propriedade determina que o logaritmo da multiplicação de dois números é igual à soma dos logaritmos desses dois números: $\log(x \times y) = \log(x) + \log(y)$ (Definição no Apêndice A)

A nova implementação do classificador Naïve Bayes, substitui a formula original de cálculo da probabilidade $\Pr(Categoria|Documento)$ pela formula descrita na Equação 4.3. O resultado desta operação não é uma probabilidade porque enquanto que uma probabilidade toma valores entre $[0, 1]$, um logaritmo pode tomar valores entre $]-\infty, +\infty[$. Uma vez que o resultado do cálculo de um logaritmo não é uma probabilidade essa designação não pode ser utilizada sendo, neste contexto, usada a designação *pontuação*.

$$\begin{aligned} &\log(\Pr(Documento|Categoria) \times \Pr(Categoria)) = \\ &\log \Pr(Documento|Categoria) + \log \Pr(Categoria) \end{aligned} \tag{4.3}$$

Uma vez que a formula de cálculo desta probabilidade é diferente, o modo como é determinada a categoria do documento é obrigatoriamente diferente. A categoria escolhida pelo classificador deixa de ser escolhida através da Equação 2.2, passa a ser escolhida com base na Equação 4.4.

$$c = \arg\max_{c} \Pr(C = c) + \sum_{i=1}^{n} Pr(F_i = f_i | C = c_i) \tag{4.4}$$

Onde c é a categoria escolhida, C é o espaço de categorias possíveis e F é o conjunto de termos que caracterizam o documento.

Esta segunda implementação do classificador Naïve Bayes corrige o problema "floating-point underflow" mas não é uma solução perfeita. Devido a este método retornar pontuações muito próximas entre categorias (quando comparados com aqueles obtidos na primeira implementação do classificador Naïve Bayes) a aplicação da regra do limiar (cf. Equação 4.2) torna-se mais difícil na medida em que as categorias melhor classificadas vão possuir pontuações mais próximas.

Na primeira implementação do classificador Naïve Bayes a regra expressa pela Equação 4.2 tem um valor de $\alpha = 2$ que indica que a categoria escolhida tem que ter o dobro da probabilidade da segunda categoria melhor classificada. Na segunda implementação deste classificador o valor de α é 0.1, o que indica que categoria escolhida tem que ter uma pontuação 10% superior à segunda categoria melhor pontuada. Com pontuações tão próximas é compreensivel que por vezes a regra do limiar não permita a classificação de um documento.

Treino. Dado que o cálculo das probabilidades depende de informação estatística é necessário treinar o classificador. Esta tarefa consiste em fornecer ao classificador exemplos de documentos classificados com uma categoria de modo a que este reconheça as categorias definidas na taxonomia (cf. Capítulo 4.1).

O treino supervisionado assemelha-se ao tipo de aprendizagem a que as crianças são submetidas durante os primeiros anos de vida. Esta aprendizagem conta com a presença constante de tutores que incentivam a comportamentos considerados correctos e corrigem comportamentos errados. Na prática o treino consiste em preencher com informação as duas variáveis anteriormente descritas (*fc* e *cc*). O método treino recebe um documento e uma categoria, para cada termo no documento o método *incFeature* incrementa o número de ocorrências desse termo naquela categoria, como nesta implementação cada documento só pode contribuir no máximo com um termo igual o incremento realizado

para é, no máximo, de um. Seguidamente o método *incCat* que incrementa o número total de documentos classificados naquela categoria.

Para testar o funcionamento deste protótipo foi criada uma colecção de documentos de treino. A Tabela 4.2 ilustra o número de documentos de treino para cada categoria. Os documentos foram obtidos através de processos semi-automáticos de extracção de texto das mais variadas fontes.

Categoria	*Num.Documentos*
Artigo	50
DiarioRepublica	50
Factura	50
Receita	50
Relatorio	50
Total	250

TABELA 4.2: Colecção de treino.

A categoria *Artigo* contêm pequenas reportagens retiradas de páginas "web" de jornais nacionais. Os temas retratados economia, politica, meteorologia ou desporto. A categoria *DiarioRepublica* é composta por decretos lei retirados da página de diários da republica electrónicos, em formato "Portable Document Format" (*PDF*). As facturas que compõem colecção foram obtidas a partir de exemplos reais e de aplicações de facturação existentes no mercado. Os documentos da categoria *Receita* foram obtidos aleatoriamente a partir de páginas "web" e livros em formato "pdf". A categoria *Relatorio* corresponde a um conjunto de relatórios de trabalhos académicos. Foram escolhidas estes cinco conjuntos de documentos porque permitem testar o funcionamento do protótipo implementado com documentos de naturezas distintas e de temas bastante díspares. Com esta colecção de documentos pretende-se perceber quais as qualidades e fragilidades da classificação deste modelo de classificação.

Discussão. Nesta secção foi apresentada a concretização do classificador Naïve Bayes. A ideia por trás desta teoria é muito simples, minimizar o risco de classificação mais precisamente o risco de classificar mal um documento/fragmento. Concluimos que a formula de Bayes (Equação A.1) permite-nos inverter a probabilidade condicionada Pr($Documento|Categoria$) de modo a calcular a probabilidade de uma categoria dado um documento.

Os classificadores com base no teorema de Bayes sofrem de alguns problemas como o problema "floatingpoint underflow" causado pela sucessiva multiplicação de valores menores que um. Esta operação realizada muitas vezes origina que a probabilidade seja um valor tão pequeno que o computador deixa de ter capacidade para o calcular. Como

alternativa a este método implementou-se uma versão do classificador de Bayes que faz uso de uma regra matemática que consiste na soma de logaritmos, esta regra por não realizar operações de multiplicação, não sofre do problema anterior.

Apesar de muito simples os classificadores do tipo Naïve Bayes conseguem obter taxas de classificação correctas bastante aceitáveis (Zhang, 2004). Frequentemente tem sido demonstrado que o Naïve Bayes consegue igualar ou melhorar a performance de alguns classificadores mais sofisticados sobre muitos conjuntos de dados (Domingos and Pazzani, 1997, McCallum and Nigam, 1998, Caruana and Niculescu-Mizil, 2006). Uma das maiores vantagens deste classificador sobre os restantes métodos de classificação é a rapidez do processo de treino e de interrogação do treino mesmo com grandes quantidades de documentos. Outro aspecto que pode ser considerado uma vantagem é que a classificação de um documento implica apenas a manipulação matemática das probabilidades dos seus termos. Para além disso o treino é um processo que pode ser incremental ao contrário de outros métodos que necessitam do treino todo de uma só vez (*decision trees* e *suport-vector machines*). A maior desvantagem do Naïve Bayes é a incapacidade de tirar partido da ordem natural das palavras no texto para classificação.

4.4 Concretização do classificador RegEx

O classificador RegEx apesar de eficaz tem uma implementação relativamente simples (cf. Algoritmo 2). Este classificador foi implementado com recurso a uma biblioteca Java com o mesmo nome deste classificador. Esta biblioteca é constituida por um conjunto de classes que permitem a identificação de sequências de caracteres contra o padrão expresso por uma expressão regular.

O processo de classificação inicia-se com a identificação de texto que cumpra o padrão expresso por uma expressão regular, mais precisamente a expressão regular do fragmento recebido como argumento. Isto permite a identificação sucessiva de texto que cumpra o mesmo padrão várias vezes (linha 2). Uma vez identificada a informação presente numa linha de texto tem inicio a etapa de refinamento da informação. Isto é conseguido através dos fragmentos descendentes daquele recebido como argumento (linha 3). Usando a expressão regular presente em cada um deles é novamente identificado o texto que cumpra cada padrão. Deste modo cada fragmento descendentes extrai mais informação detalhando aquela que foi obtida inicialmente, isto é, ao particionar texto adquirido numa primeira fase, é possível refinar a informação extraida (linha 4). Este processo de refinamento da informação do fragmento ascendente repete-se enquanto existir um fragmento composto por outros fragmentos. O passo final é armazenar a informação em cada fragmento do modelo em memória (linha 8).

Algoritmo 2 RegEx algorithm

```
1:  procedure CLASSIFICA(text, fragment)
2:      while str ← find(text, fragment.RegEx) do
3:          for frag̀ ← fragment.getChild() do
4:              while str̀ ← find(str, frag̀.RegEx) do
5:                  frag̀.setContent(str̀)
6:                  fragment.setChild(frag̀)
7:              end while
8:          end for
9:          ret_frag.setChild(fragment)
10:     end while
11:     return ret_frag
12: end procedure
```

Discussão. A aparente simplicidade deste algoritmo esconde uma boa capacidade de adaptação aos vários formatos de informação (*inline* e tabular). O que isto quer dizer é que apesar de o modo como os fragmentos destes dois formatos de informação estão organizados na taxonomia XML ser diferente, o algoritmo consegue extrair a informação de ambos sem que haja código especifico para cada um deles.

Como se sabe a taxonomia com base na qual este algoritmo realiza a extracção de informação está em formato XML. Por essa razão o Algoritmo 2 está comprometido com a forma como a informação está representada no XML e se por alguma razão a organização e conteúdo da taxonomia fosse radicalmente alterada, também este algoritmo teria de ser alterado.

4.5 Concretização das extensões

Definição manual da taxonomia. Esta ferramenta foi criada para auxiliar na definição da taxonomia de categorias de documentos e fragmentos. Esta extensão é composta por uma interface visual que permite abstrair quem define as categorias dos detalhes da linguagem XML, permitindo que o trabalho se centre na organização das categorias e na definição de expressões regulares.

Posto que a organização das categorias é hierárquica pode ser facilmente representada numa árvore. Na definição das categorias e dos respectivos classificadores foi aplicada uma regra de modo a garantir que não existem inconsistências. Essa regra diz que apenas os elementos que possuam sub-categorias podem ter um classificador associado, caso contrário o classificador que o classifica a categoria em questão é o classificador da categoria ascendente. Esta regra faz sentido na medida em que, se uma categoria não tem sub-categorias, definir um classificador para si diferente da categoria ascendente torna

a taxonomia confusa e com uma lógica pouco consistente. No caso de uma categoria possuir sub-categorias é possível definir um classificador diferente do seu ascendente o que permite que a árvore de categorias possua sub-árvores especificas para alguns classificadores.

Nos fragmentos, para além dos elementos mencionados, também é possível definir as expressões regulares nas categorias de fragmentos cujo classificador é o RegEx. Esta ferramenta não valida a correcção destas expressões pelo que a sua criação deve ser feita criteriosamente.

Definição automática da taxonomia. Suponhamos que somos alguém que tem a tarefa de classificar uma colecção de documentos que nunca foi classificada antes. Se a dimensão da colecção não for muito grande apesar de entediante é um trabalho fazível mas se a colecção for de maior dimensão, a nossa tarefa torna-se quase impossível. A solução têm que passar obrigatoriamente por algum processo automático ou então pela inconveniente contratação de muito pessoal.

A segunda ferramenta foi criada é exactamente com esse objectivo, fornecer ao utilizador um processo automático de classificação de documentos. Para cumprir este objecto a aplicação criada organiza documentos em categorias de modo fácil e automático. Para isso integrou-se um motor de agrupamento automático de resultados de pesquisas na Internet. O motor utilizado é de distribuição chama-se Carrot2 e apesar de ser vocacionado para agrupar resultados de pesquisas na "web" pode ser adaptado a documentos de texto. Dos vários algoritmos que o Carrot2 suporta foram apenas utilizados dois deles o Lingo e o STC, além destes algoritmos foi também implementado o k-means cujo funcionamento foi descrito no Capítulo 2.2. Esta ferramenta para definição automática de taxonomias foi implementada apenas ao nível da primeira fase de classificação e apenas para as categorias de documentos, tem no entanto implementada uma interface gráfica para facilitar o trabalho de definir as categorias. O funcionamento desta ferramenta consiste em emular os ficheiros a organizar em formato XML necessário ao funcionamento do Carrot2. Esses ficheiros processados pelo motor de agrupamento em conjunto com os parâmetros necessários ao funcionamento do algoritmo escolhido, resultam numa lista com nome dos documentos organizados por temas. Neste ponto o utilizador da aplicação tem os documentos agrupados sem que tivesse sequer de escolher o número de grupos, o que não acontece em alguns algoritmos de agrupamento.

Mas tendo em conta que os os algoritmos usados nesta ferramenta não são infalíveis existem duas situações a que é necessário a intervenção humana. A primeira é a nomeação que é dada aos grupos. Tipicamente os algoritmos usam métricas que se baseiam no número de vezes que os termos ocorrem para calcular o nome dos grupos, que muitas vezes

não produzem rótulos compreensíveis para o Homem. Por essa razão é necessário que por vezes o utilizador renomeie os grupos manualmente. O segundo aspecto a ter em conta é que a precisão do agrupamento varia de algoritmo para algoritmo. Mesmo escolhendo um bom algoritmo é admissível que alguns documentos sejam sido mal agrupados, por isso a intervenção humana é mais uma vez necessária para corrigir eventuais erros de agrupamento.

A interface gráfica desta aplicação representa, à semelhança da definição manual, a organização das categorias numa árvore, e para permitir a correcção dos dois problemas a cima referidos os nós da árvore são mutáveis tanto a nível de nome como a nível de posição (sistema "drag and drop"). A imagem seguinte ilustra o agrupamento feito com o algoritmo Lingo para um pequeno grupo de dezasseis facturas de três empresas diferentes onde se pode verificar alguns dos problemas descritos.

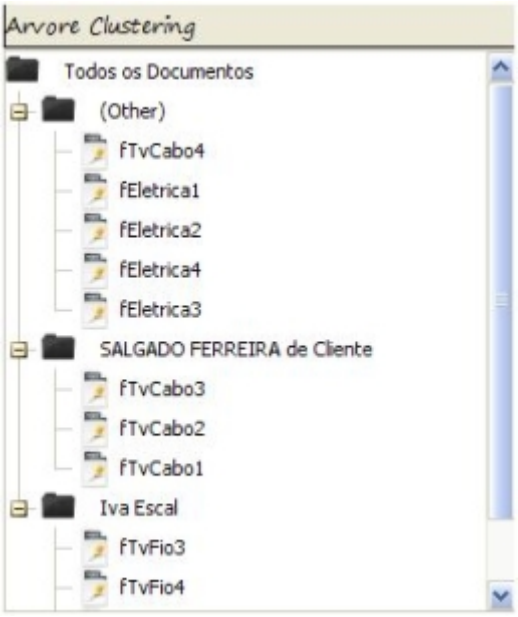

FIGURA 4.3: Exemplo de agrupamento.

Como se pode observar na Figura 4.3 os grupos gerados não estão totalmente correctos pois no primeiro grupo um documento que não é do mesmo tipo que os restantes, para além disso os nomes dos grupos pouco dizem sobre os documentos que os compõem pelo que é necessário a interversão humana para realizar correcções neste agrupamento.

Capítulo 5

Experimentação e Validação

Como base na taxonomia apresentada no Capítulo 4.1 será demonstrado o processo de classificação de dois documentos, após o qual é feita a validação do modelo de classificação proposto. É avaliado do desempenho do classificador Naïve Bayes e são discutidas situações em que o classificador RegEx realiza, ou não, boas classificações.

5.1 Exemplos de classificação

No primeiro exemplo de classificação as opções de treino e aprovação de utilizador estão desactivadas para facilitar a compreensão do processo de classificação. No segundo exemplo todas as opções estão activas incluindo a opção de transformação de resultado de classificação em formato HTML para apresentação ao utilizador.

5.1.1 Receita Culinária

O primeiro exemplo é uma classificação é realizada sobre uma receita de culinária. A receita trata de um prato Árabe que consiste num assado de carne bovina cozida. O processo de classificação desse documento é o seguinte.

A primeira etapa de qualquer classificação é a leitura do texto que compõe o documento para posterior processamento. É importante que o texto seja lido de modo a ficar na mesma disposição que o texto do documento original. Isto é necessário porque as expressões regulares que permitem a extracção da informação, foram criadas com base na disposição original do texto. Seguidamente é criada uma instância da classe *Documento* que representa (em memória) o documento a classificar. A essa instância é associada o classificador por omissão (classificador Raiz), neste caso a uma instância do classificador

Naïve Bayes. Se a instância já existir em disco é recuperada, caso contrário é criada uma nova o que implica ler a colecção de treino e inicializar todas as estruturas internas do classificador.

O passo seguinte é a leitura (da taxonomia XML) de todas as categorias de documentos cujo nome do classificador é idêntico ao do classificador por omissão (*ClassificadorNaiveBayes*). Com o texto do documento e com o conjunto de categorias (*Artigo, DiarioRepublica, Factura, Receita e Relatorio*), o classificador inicia o processo que terminará com a classificação do texto numa das anteriores categorias. O classificador inicia o seu trabalho calculando a probabilidade do texto pertencer a cada categoria, o resultado desta operação é ilustrado na Tabela 5.1. Uma vez que este classificador usa em a soma de logaritmos em vez da multiplicação de probabilidades, não se pode dizer que o texto tem probabilidade de pertencer a uma categoria mas diz-se antes que um documento tem uma pontuação X na categoria Y.

Categoria	*Pontuacao*
Artigo	-149.6
DiarioRepublica	-156.3
Factura	-176.6
Receita	-64.8
Relatorio	-173.7

TABELA 5.1: Pontuação das categorias para o exemplo de uma receita culinária.

Como se pode observar a pontuação mais elevada é a da categoria *Receita* e uma vez que a pontuação cumpre a regra descrita na Equação 4.2 esta classificação é válida. De seguida o algoritmo de classificação procura na taxonomia se a *Receita* categoria possui sub-categorias e uma vez que esta categoria possui três sub-categorias (*Coelho*,*PratosMundo*,*Sanduiche*), o processo de classificação repete-se tendo como categorias possíveis estas sub-categorias. O resultado dessa classificação é o seguinte:

Categoria	*Pontuacao*
Coelho	-83.8
PratosMundo	-70.9
Sanduiche	-100.4

TABELA 5.2: Pontuação das sub-categorias para o exemplo de uma receita culinária.

Sendo a pontuação da categoria *PratosMundo* a mais elevada e visto que esta categoria não tem sub-categorias o documento é classificado com a categoria *PratosMundo*.

Assim que termina a fase I de classificação são classificados os fragmentos do documento. Para tal é necessário identificar na taxonomia de categorias de fragmentos quais os fragmentos que pertencem à categoria *PratosMundo*. Uma vez que a categoria *PratosMundo*

não possui categorias de fragmentos definidos, o algoritmo procura na categoria ascendente (*Receita*) os fragmentos definidos (*Lista_ de_ Ingredientes*, *Passos_ Preparação*).

Frag. – Cat.	*Lista_de_Ing.*	*Passos_Prep.*	*Resultado*
F.Fragmento 1	-6.5	-6.4	Ind.
F.Fragmento 2	-65.5	-70.2	Lista_de_Ing.
F.Fragmento 3	-10.2	-73.1	Passos_Prep.

TABELA 5.3: Pontuação das categorias para os fragmentos de *Receita*.

A Figura 5.3 apresenta os resultados das pontuações dos três fragmentos existentes nesta receita bem como a classificação atribuida a cada um deles. Nesta tabela os fragmentos são identificados pela palavra Fragmento seguida pela ordem de ocorrência no documento. Esta designação não tem a ver com o seu conteúdo, serve apenas para identificar os fragmentos. Como se pode observar o primeiro fragmento (que é composto pelo titulo da receita) têm a classificação de *Indeterminado* pois não cumpre com a regra do limiar. Apesar de ter uma pontuação mais baixa nas duas categorias o primeiro fragmento não foi classificado com nenhuma categoria devido à regra do limiar que foi criada para lidar exactamente com este tipo de situação. Os restantes fragmentos são correctamente classificados com as categorias *Lista_ de_ Ingrediente* e *Passos_ Preparacao*.

Com o documento e os fragmentos devidamente classificação inicia-se a fase III de classificação. Esta etapa começa com a leitura das categorias de fragmentos descendentes aqueles que foram identificados na fase II. Neste caso o fragmento *Lista_ de_ Ingredientes* identificado na fase II é composto por um fragmento *ingrediente* cujo classificador é o RegEx. Esse fragmento é usado na fase III para a identificação da informação presente na *Lista_ de_ Ingredientes*. O classificador RegEx começa então por identificar no texto a informação expressa pelo padrão descrito na expressão regular do fragmento *ingrediente*, essa expressão é composta pelo conjunto de expressões regulares dos fragmentos que o compõem. O padrão identifica uma linha composta uma quantidade, uma unidade e um componente de uma receita. Esta operação é repetida enquanto existirem linhas de ingrediente no texto.

Uma vez identificado o texto de uma linha de ingredientes (e.g. 1 litro de água) o processo de reconhecimento de padrões no texto é repetido para os fragmentos descendentes de *ingrediente*. Este fragmento possui três outros fragmentos cujos designados por *quantidade*, *unidade* e *componente*. A tabela seguinte apresenta as suas expressões regulares e qual a informação que estas permitem obter.

A Tabela 5.4 ilustra qual o texto que cada expressão regular consegue identificar. Se nos fragmentos *quantidade* e *unidade* o texto extraido é aquele esperado, no caso do fragmento *componente* o texto identificado não é exactamente aquele que se pretendia. Observa-se

Fragmento	RegEx	Texto Identificado
quantidade	\d{1,4}	1
unidade	(g |litro | kg |ml)	litro
componente	(.*)	de água

TABELA 5.4: Excerto da informação obtida por cada fragmento de uma *Receita*

que o texto do fragmento *componente* contem a palavra "de" quando idealmente só teria a palavra "água", esta situação ocorre porque a expressões regulares não filtra este tipo de palavras tal como acontece no filtro de "Stop Words".

O segundo fragmento identificado na fase II é um fragmento denominado por *Passos_de_Preparação* e contem apenas um fragmento descendente, esse fragmento chama-se *passo* e o seu objectivo é identificar passos de preparação de uma receita. *Passo* contêm uma expressão regular que permite a identificação frases separadas por ponto final independentemente se as estas englobam uma ou mais linhas. Este fragmento não é composto por outros fragmentos pelo que o texto extraido não é refinado e resulta apenas da aplicação da seguinte expressão regular: $[\backslash p\{L\}\backslash s\backslash d:,()]$.

Com a extracção da informação o processo de classificação fica assim completo, a informação que originalmente se encontrava não estruturada for transformada numa representação semi-estruturada que o computador já pode processar. A última etapa consiste em transformar a representação do documento em memória em XML para que a informação possa ser consultada mais tarde.

5.1.2 Factura

O segundo exemplo de classificação é a classificação de uma factura. O documento usado neste exemplo de classificação é uma réplica exacta de uma factura emitida por uma empresa mas com os dados alterados. À semelhança do primeiro exemplo de classificação, os primeiros passos do processo de classificação são a leitura o texto, obtenção as categorias passiveis de classificar o documento e criar as instâncias da classe *Documento* e do classificador por omissão.

A principal diferença reside no facto de desta vez o classificador Naïve Bayes não ser criado de raiz mas é usada uma instância de um classificador anteriormente armazenado em disco. Esta opção permite reduzir o tempo de criação da instância do classificador Naïve Bayes de $7021ms$ para $3565ms$ o que corresponde a uma redução de cerca de $50,7\%$. Não sendo uma redução real muito visivel dado a reduzida dimensão da colecção de treino, é possivel ter uma boa ideia dos ganhos possiveis.

Independentemente do modo como o classificador é iniciado a classificação na fase I é feita do mesmo modo que o exemplo anterior. Observando a tabela 5.5 conclui-se que a pontuação do texto nas várias categorias é bastante diferente do exemplo anterior como seria de esperar.

Categoria	*Pontuacao*
Artigo	-200.8
DiarioRepublica	-169.9
Factura	-78.7
Receita	-239.5
Relatorio	-237.4

TABELA 5.5: Pontuação das categorias para o exemplo de uma factura.

Como as opções *Aprovação de utilizador* e *Treino* estão activas o utilizador é inquirido no sentido de aprovar a classificação. Se o utilizador não aprovar a classificação *Factura* é-lhe dado a escolher a categoria correta (de entre as cinco categorias possíveis). Após a aprovação da classificação o documento é classificado com a categoria escolhida e o classificador é treinado com este texto e a respectiva categoria. Assim que o treino é finalizado o classificador pode ser novamente armazenado em disco para se assegurar a persistência do novo treino.

Dado que a categoria *Factura* possui sub-categorias, a taxonomia é mais uma vez consultada para obtenção dessa informação. Com o novo conjunto de sub-categorias *F.TvCabo, F.TvFio* e *F.Eletrica* o processo de classificação.

Categoria	*Pontuacao*
F.TvFio	-210.9
F.TvCabo	-26.1
F.Eletrica	-211.0

TABELA 5.6: Pontuação das sub-categorias para o exemplo de uma factura

A categoria *F.TvCabo* possui uma pontuação bastante superior às restantes e por isso é escolhida para classificar o documento. Uma vez determinada a categoria deste documento os seus fragmentos raiz são igualmente classificados com o classificador Naïve Bayes. Este documento difere do documento do exemplo anterior porque contem um número de fragmentos bastante superior. O modo como o processo de classificação está implementado permite que existam fragmentos no documento que não possuam categorias definidas na taxonomia. Esta prática não é muito recomendada pois originar erros de classificação mas possibilita que não seja obrigatório definir as categorias para todos os fragmentos de cada documento. No entanto a classificação conjunta de fragmentos e a regra do limiar fornecem alguma protecção contra estes erros.

Frag. – Cat.	*TvCaboTab*	*TvCaboInfo*	*Resultado*
F.Fragmento 1	-52.9	-57.4	Ind.
F.Fragmento 2	-47	-46.7	Ind.
F.Fragmento 3	-10.7	-35	TvCaboTab
F.Fragmento 4	-5	-54.3	TvCaboTab
F.Fragmento 5	-41.3	-52.5	TvCaboTab
F.Fragmento 6	-65.2	-10	TvCaboInfo

TABELA 5.7: Pontuação das categorias para os fragmentos de *Factura*.

Como é possível observar através na Tabela 5.7 existem seis fragmentos no documento mas apenas estão definidas duas categorias. Deste modo é natural que vários fragmentos sejam classificados com a mesma categoria (*TvCaboTab*). Para eleger a categoria correcta é necessário analisar, de entre os fragmentos classificados com a mesma categoria, qual deles tem a maior pontuação.

O processo de extracção de informação do documento classificado como *F.TvCabo* é antecedido da leitura dos fragmentos que compõem aqueles identificados na fase II (*TvCaboInfo* e *TvCaboTab*). Ao contrário do exemplo anterior em que a categoria que classifica o documento não tinha fragmentos associados, neste caso é a categoria *F.TvCabo* que tem a definição dos seus fragmentos.

O fragmento classificado com a categoria *TvCaboTab* é composto por uma tabela com a descrição dos serviços prestados pela empresa. O processo de extracção da informação deste fragmento é realizado, numa primeira fase, através da identificação o texto presente em cada linha da tabela. Para isso existe o fragmento *Tuplo* (descendente de *TvCaboTab*) que contem a expressão regular capaz de identificar todos os valores que devem existir numa linha dessa tabela. A fim de identificar a informação de toda a tabela apenas é necessário aplicar a expressão regular de *Tuplo* a todas as linhas da tabela. Para perceber como este fragmento extrai a informação e quais os fragmentos que a detalham apresenta-se Tabela 5.8 que demonstra a informação obtida quando se analisa a primeira linha do fragmento *TvCaboTab*.

Fragmento	RegEx	Texto Identificado
Tuplo	\p{L}[\p{L}+\s]+€\s\d+[,.]\d+	Televisão €22.99
Nome_Item	\p{L}[\p{L}+\s]+	Televisão
Valor	\d+[,.]\d+	22.99

TABELA 5.8: Excerto da informação obtida por cada fragmento de uma *Factura*

O padrão descrito pela expressão regular de *Tuplo* é bastante flexível pois não está comprometido com a informação a identifica. Neste exemplo o artigo "Televisão" poderia ser identificado com uma expressão regular do tipo: $Televisao\backslash s€\backslash d + [,.]\backslash d+$ onde o

nome "Televisão" está explicito. Em vez disso a expressão regular de *Tuplo* foi criada para admitir um artigo genérico (qualquer outra coisa que não "Televisão"). Uma vez mais esta informação pode ser refinada recorrendo às sub-categorias de *Tuplo*. Neste caso são as categorias *Nome_Item* e *Valor* que extraem (do conteúdo do fragmento ascendente) o nome do produto adquirido.

Finalmente o resultado da classificação deste documento é armazenado num documento XML. Com a transformação da informação em XML, é possível usar o módulo de apresentação desenvolvido neste protótipo. Este módulo usa como entrada o referido ficheiro XML e um ficheiro de regras de transformação para produzir um documento HTML. A Figura 5.1 ilustra uma simples transformação da informação obtida no processamento desta factura, mais precisamente a transformação da informação presente nos dois fragmentos ($TvCaboTab, TvCaboInfo$) definidos na taxonomia para esta categoria de documentos ($F.TvCabo$).

FIGURA 5.1: Representação HTML da informação presente numa factura.

5.2 Validação

A validação deste modelo, consiste em avaliar o desempenho do classificador Naïve Bayes. Em classificação desempenho é avaliado em termos da taxa de erro, que indica a quantidade de erros cometidos pelo classificador. No caso da validação deste neste modelo, é avaliada a taxa de eficácia de classificação que indica a percentagem de classificações

bem efectuadas pelo classificador da Naïve Bayes para as categorias de documentos e respectivas sub-categorias.

Tendo em conta que existe uma colecção de documentos de treino esta poderia ser usada na validação, mas usar a mesma colecção no treino e na validação pode não ser a melhor prática porque sabe-se "à priori" a classificação correcta de cada documento da colecção de treino e portanto os resultados obtidos dessa avaliação diriam pouco acerca da eficácia de classificação. O que se pretende realmente saber é a eficácia de classificação de documentos que o classificador nunca viu (previsão de classificação). Assim é necessário obter, para além da colecção de documentos para treino, uma colecção de documentos diferente para teste.

Quando a quantidade de documentos é grande isto não é um problema pois facilmente se divide a colecção global de documentos num conjunto de treino e noutro conjunto de teste, em que ambos são representativos e podem fornecer uma boa indicação da performance futura. O problema surge quando a colecção de documentos não é extensa, que é o caso da colecção de documentos criada para este trabalho. Para lidar com este problema optou-se por realizar validação cruzada que é o método recomendado quando a colecção de documentos é pequena.

A validação cruzada consiste em dividir a colecção de documentos em três partes de dimensão semelhante (o número de partes pode ser um K qualquer) usando uma parte para teste e as restantes para treino. O processo é repetido três vezes de modo a que cada uma partes seja usada pelo menos uma vez no treino e no teste diminuindo a possibilidade de a colecção de treino não ser representativa. O resultado final da validação é a média das três validações.

Os resultados da validação apresentam-se sob o forma de matrizes de confusão que fornecem informação para compreender o desempenho do classificador e a origem dos erros mais frequentes (Marques, 2005). Seguidamente são apresentados os resultados dos testes de classificação realizados sobre o classificador Naïve Bayes. Um aspecto que se deve salientar é que na avaliação do classificador Naïve Bayes foi usada a implementação que realiza a classificação com base na soma de logaritmos e não a multiplicação de probabilidades.

Iteração I

(a) Categorias

		Saida do Classificador						
		Artigo	Diario R.	Factura	Receita	Relatorio	Impossivel C.	Eficácia %
Categoria Correcta	Artigo	0	0	0	0	2	15	0,0%
	Diario R.	0	16	0	0	0	0	100,0%
	Factura	0	0	15	0	0	1	93,7%
	Receita	0	0	0	16	0	0	100,0%
	Relatorio	0	0	0	0	16	0	100,0%

(b) Especializações - Receita

		Saida do Classificador - Especializações (Receitas)				
		Coelho	PratosMundo	Sanduiche	Impossivel C.	Eficácia %
Decisão Correcta	Coelho	5	0	0	0	100,0%
	PratosMundo	0	6	0	1	85,7%
	Sanduiche	0	0	4	0	100,0%

(c) Especializações - Factura

		Saida do Classificador - Especializações (Facturas)				
		F.Eletrica	F.TvCabo	F.TvFio	Impossivel C.	Eficácia %
Decisão Correcta	F.Eletrica	4	0	0	1	80,0%
	F.TvCabo	0	6	0	0	100,0%
	F.TvFio	0	0	5	0	100,0%

TABELA 5.9: Matrizes de confusão: Iteração I

Iteração II

(a) Categorias

		Saida do Classificador						
		Artigo	Diario R.	Factura	Receita	Relatorio	Impossivel C.	Eficácia %
Categoria Correcta	Artigo	0	0	0	0	2	15	0,0%
	Diario R.	0	17	0	0	0	0	100,0%
	Factura	0	0	15	0	0	0	100,0%
	Receita	0	0	0	17	0	0	100,0%
	Relatorio	0	0	0	0	17	0	100,0%

(b) Especializações - Receita

		Saida do Classificador - Especializações (Receitas)				
		Coelho	PratosMundo	Sanduiche	Impossivel C.	Eficácia %
Decisão Correcta	Coelho	3	0	0	0	100,0%
	PratosMundo	0	2	0	5	28,5%
	Sanduiche	0	0	6	0	100,0%

(c) Especializações - Factura

		Saida do Classificador - Especializações (Facturas)				
		F.Eletrica	F.TvCabo	F.TvFio	Impossivel C.	Eficácia %
Decisão Correcta	F.Eletrica	4	0	0	0	100,0%
	F.TvCabo	0	4	0	0	100,0%
	F.TvFio	0	0	4	0	100,0%

TABELA 5.10: Matrizes de confusão: Iteração II

Iteração III

(a) Categorias

		Saida do Classificador						
		Artigo	Diario R.	Factura	Receita	Relatorio	Impossivel C.	Eficácia %
Categoria Correcta	Artigo	0	1	0	0	3	12	0,0%
	Diario R.	0	17	0	0	0	0	100,0%
	Factura	0	0	16	0	0	0	100,0%
	Receita	0	0	0	17	0	0	100,0%
	Relatorio	0	0	0	0	16	0	100,0%

(b) Especializações - Receita

		Saida do Classificador - Especializações (Receitas)				
		Coelho	PratosMundo	Sanduiche	Impossivel C.	Eficácia %
Decisão Correcta	Coelho	4	0	0	0	100,0%
	PratosMundo	0	5	0	8	38,4%
	Sanduiche	0	0	6	1	85,7%

(c) Especializações - Factura

		Saida do Classificador - Especializações (Facturas)				
		F.Eletrica	F.TvCabo	F.TvFio	Impossivel C.	Eficácia %
Decisão Correcta	F.Eletrica	5	0	0	0	100,0%
	F.TvCabo	0	4	0	0	100,0%
	F.TvFio	0	0	5	0	100,0%

TABELA 5.11: Matrizes de confusão: Iteração III

Resumo

(a) Eficiência média - Categorias

Resumo - Eficácia Média	
Categoria	Eficácia %
Artigo	0,0%
Diario R.	100,0%
Factura	97,9%
Receita	100,0%
Relatorio	100,0%

(b) Eficiência média das Especializações - Receita

Resumo - Eficácia Média	
Categoria	Eficácia %
Coelho	100,0%
PratosMundo	50,8%
Sanduiche	95,2%

(c) Eficiência média das Especializações - Factura

Resumo - Eficácia Média	
Categoria	Eficácia %
F.Eletrica	95,3%
F.TvCabo	100,0%
F.TvFio	100,0%

TABELA 5.12: Resumo das matrizes de confusão

Discussão dos resultados de classificação. Após a observação dos resultados das três iterações é possível chegar a algumas conclusões. O que se pode concluir imediatamente é que que existem quatro categorias em que o classificador Naïve Bayes apresenta uma eficácia muito boa (perto dos 100%). Essas categorias são: *Receita*, *DiarioRepublica*, *Relatorio* e *Factura*.

Este resultado justifica-se com a duas razões: A primeira delas é a estrutura dos documentos, que por ser estática possui algumas palavras que se encontram em todos os documentos mesmo que o resto do conteúdo varie (e.g. cabeçalhos das tabelas nas facturas). Independentemente de a factura referir um ou outro individuo as tabelas estão sempre presentes e portanto os seus cabeçalhos são iguais. Também nas receitas esta situação se verifica uma vez que na maioria das receitas existe sempre uma ou mais palavras que identificam certas zonas do documentos como por exemplo, ingredientes ou passos de preparação.

A segunda razão para a elevada eficácia de classificação nestas categorias está relacionada com o padrão de ocorrência de palavras nos documentos dessas categorias. Observando o exemplo das receitas de culinária, verifica-se que existem palavras relacionadas com a confecção dos ingredientes que se encontram em muitas receitas diferentes mas não ocorrem com tanta frequências nas restantes categorias de documentos.

A questão que se deve colocar é «em que é que a existência de padrões de ocorrência de palavras contribui para a boa eficácia da classificação?» Na realidade contribui muito, pois a existência de conjuntos de palavras que ocorrem com maior frequência em documentos de determinadas categorias faz com que seja possível distinguir os documentos dessas categorias das restantes, ou seja, os padrões de palavras têm um poder discriminatório. Relembrando que no classificador Naïve Bayes, o cálculo da probabilidade de pertença de um documento a uma categoria depende do número de termos classificados com essa categoria (treino), os dois factores referidos anteriormente tomam ainda mais importância.

Os documentos das categorias *DiarioRepublica* e *Relatorio* são correctamente classificados pelo classificador Naïve Bayes, mais uma vez em consequência de um padrão de ocorrência de palavras mais comuns nestas categorias de documentos do que nas restantes. Por exemplo, no caso dos artigos e decretos lei publicados em diário da Republica existem palavras como "ministério", "decreto-lei" ou "republica" que ocorrem com maior frequência neste tipo de documento do que em facturas ou receitas culinárias. Isto faz com que seja possível diferenciar os documentos desta categoria de documentos de outras categorias e está na origem da taxa de eficiência.

No caso da categoria *Relatório*, composta por relatórios de trabalhos académicos sobre bases de dados, microprocessadores ou programação em várias linguagens. Uma vez que nestes documentos existem muitas palavras técnicas que não ocorrem com frequência nos documentos das restantes categorias, essas palavras também possuem um bom poder discriminatório.

A categoria onde o classificador teve pior resultado foi a categoria *Artigo* onde a eficiência foi de 0%. Inicialmente este resuldado foi encarado com ceticismo porque este classificador apresenta taxas de eficiência rasoáveis para as restantes categorias, e por isso não deveria falhar totalmente na classificação de documentos desta categoria. A razão pela qual o classificador falha na classificação dos artigos jornalisticos é a falta de padrões de ocorrência de palavras, em contraste com o que acontece nas categorias melhor classificadas. Relacionada com esta situação está o facto de os artigos tratarem de assuntos distintos e não de um tema especifico, deste modo não existe um conjunto suficientemente grande de palavras que contribua para a diferênciação dos documentos desta categoria das restantes.

Estes documentos, por não possuirem um conjunto de palavras que os permita distinguir dos documentos outros das restantes categorias são sempre mal classificados. A má classificação pode ocorrer em duas situações: os documentos são classificados na categoria errada; ou não cumprem com a *regra do limiar* (c.f. Equação 4.2) o que leva a que nenhuma categoria seja escolhida. O não cumprimento da regra do limiar é a situação

mais recorrente pois a vulgaridade das palavras dos documentos desta categoria não permite que nenhuma categoria se destaque em termos de pontuação.

No caso das sub-categorias a taxa de eficiência é bastante elevada na maioria dos casos, com excepção da sub-categoria *PratosMundo*. As tabelas das três iterações de validação demonstram que, na maioria das situações em que uma sub-categoria foi mal classificada, o erro deveu-se à regra do limiar. O que implica que o documento não teve a pontuação suficiente ser classificado sem sobra de dúvida numa categoria. As sub-categorias de *Factura* foram aquelas com maior taxa de eficiência, este facto deve-se principalmente devido a estrutura rígida destes documentos.

Considerações sobre o método de cálculo. No método de cálculo de probabilidades apresentado no Capítulo 4.3, é referido que a eficiência do classificador Naïve Bayes é maior quando não ocorre o problema "floating point overflow". As tabelas 5.2 e 5.3 apresentam o resultado das classificações realizadas por este classificador com a versão original (que tem o problema "floating point overflow") do classificador Naïve Bayes e com a versão modificada do mesmo.

		Saida do Classificador					
		Artigo	Diario R.	Factura	Receita	Relatorio	Impossivel C.
Categoria Correcta	Artigo	0	1	0	1	6	32
	Diario R.	0	24	0	0	0	26
	Factura	0	0	49	0	0	1
	Receita	0	0	0	50	0	**0**
	Relatorio	0	0	0	0	9	41

FIGURA 5.2: Tabela de classificação de documentos (versão Naïve Bayes original)

		Saida do Classificador					
		Artigo	Diario R.	Factura	Receita	Relatorio	Impossivel C.
Categoria Correcta	Artigo	0	1	0	0	7	32
	Diario R.	0	50	0	0	0	0
	Factura	0	0	49	0	0	1
	Receita	0	0	0	50	0	**0**
	Relatorio	0	0	0	0	50	0

FIGURA 5.3: Tabela de classificação de documentos (versão Naïve Bayes modificada)

Estas tabelas provam que a segunda versão do classificador Naïve Bayes tem de facto a melhor taxa de eficiência. Outra coisa que se pode concluir é que a maioria dos documentos que anteriormente eram impossíveis de classificar, pertenciam de facto a uma das categorias e a sua má classificação se devia ao conteúdo dos documentos ser extenso e dar origem a "floating point overflow".

Classificador RegEx. O classificador RegEx não pode avaliado do mesmo modo que o classificador Naïve Bayes pois não existe um processo automático para confirmar a exactidão das suas classificações. Para avaliar este classificador seria necessário que um humano confirma-se manualmente se a informação extraida do documento estava correcta. No entanto é possível tecer algumas considerações acerca das classificações feitas por este classificador. Uma vez que este classificador não está dependente de um processo de treino a sua eficácia também não é afectada por ele. Em vez disso usa padrões expressos por expressões regulares para identificar informação em documentos. Por essa razão este classificador apenas falha se ocorrer uma de duas situações:

- O classificador Naïve Bayes classificou mal o documento ou um dos seus fragmentos. Ao ser mal classificado um destes elementos as expressões regulares aplicadas sobre os fragmentos não vão certamente identificar a informação pretendida. Recordando o exemplo de classificação de uma factura estudado no Capítulo 5.1 podemos verificar como este problema pode ocorrer. Supondo que o documento é mal classificado com a categoria *Receita*, quando o classificador RegEx proceder à extracção da informação do fragmento *TvCaboTab* nunca irá encontrar no texto a informação que corresponde ao padrão de uma tabela.
- O texto para o qual foi criada uma expressão regular não cumpre sempre um padrão de ocorrência. Um exemplo desta situação é a extracção de informação de um fragmento classificado como *lista de ingredientes* em que existam linhas que não possuem os três elementos *quantidade-unidade-componente*.

A última situação a acima referida é provavelmente o maior problema relacionado com o uso de expressões regulares, pois se as regras criadas não forem suficientemente genéricas a extracção de informação torna-se complicada e muito susceptível a erros. O uso de expressões regulares para extracção de informação em documentos em que o texto é livre é desaconselhado porque dificilmente existe informação estruturada onde se possa encontrar padrões.

Capítulo 6

Conclusões e trabalho Futuro

Esta dissertação propõe um modelo de classificação de documentos para transformar informação não estruturada num formato estruturado adequado à manipulação automática por computadores. O modelo de classificação consiste num algoritmo que aplica repetidamente classificadores para identificar padrões de informação em documentos. A identificação de padrões permite extrair a informação por eles contida. Este processo é composto por três fases de classificação onde o documento é inicialmente classificado pelo classificador Naïve Bayes numa categoria conhecida, sendo possível repetir esta classificação para sub-categorias daquela que classificou inicialmente o documento. A fase seguinte consiste em classificar (com recurso ao classificador Naïve Bayes) os fragmentos raiz do documento como forma de identificar blocos de informação. Na terceira e última fase de classificação o classificador RegEx identifica padrões de informação recorrendo aos fragmentos descendentes daqueles classificados na fase anterior. A sucessiva aplicação desses classificadores permite refinar incrementalmente a informação obtida.

Como forma de melhorar o desempenho do classificador Naïve Bayes foi desenvolvida uma implementação deste classificador para corrigir o problema designado por "floatingpoint underflow". Demonstrou-se que com esta segunda implementação que substitui o produto de probabilidades pela soma de logaritmos de probabilidades a taxa de eficiência da classificação de algumas categorias sobe até perto de 100%.

De entre as funcionalidades implementadas neste modelo de classificação realça-se a persistência do classificador Naïve Bayes que permite reduzir o tempo de inicialização deste classificador em cerca de 50%. A funcionalidade do treino continuado que permite eventualmente aumentar a eficiência de classificação na medida em o classificador Naïve Bayes continua a receber exemplos de treino.

A avaliação do classificador Naïve Bayes apresentada no Capítulo 5.2 demonstra que documentos com estrutura e com um conjunto palavras comuns numa determinada categoria, são geralmente bem classificados pois o classificador apresenta uma eficácia de aproximadamente 100% (na maioria das categorias de documentos). Com base nos resultado obtidos na avalidação conclui-se também que quanto mais geral é o conteúdo de um documento, menor é a eficiência de classificação do classificador. Na avaliação da eficiência deste classificador para as sub-categorias de documentos conclui-se que a taxa de classificações correctas varia entre os 50,7% e os 100%. Tal como nos caso das categorias de documentos a eficiência de classificação depende do conteúdo do documento, em alguns documentos pertencentes às sub-categorias apresentadas o seu conteúdo não suficientemente distinto por forma a que os documentos sejam classificados sem sobra de dúvida das sub-categorias.

Nesta dissertação foi apresentado o classificador RegEx que tem como finalidade a procura de padrões em texto para extracção de informação, esses padrões são descritos por expressões regulares criadas para extrair alguma informação em particular. Este classificador (usado após as fases I e II de classificação) realiza a operação de extracção com grande eficiência na medida em que o texto de onde a informação é extraida já foi previamente classificado. Deste modo as regras de extracção são quase sempre aplicadas sobre a informação para a qual foram criadas e portanto permitem a correcta extracção de informação.

Apesar de o algoritmo de classificação deste classificador estar condicionado com nenhuma forma de representação de conhecimento, a implementação depende do modo como como a taxonomia de categorias está organizada. Se esta fosse radicalmente alterada a implementação do classificador também teria de ser modificada.

6.1 Trabalho futuro

O modelo de classificação apresentado neste trabalho não está fechado. Existem várias áreas onde é possível continuar o seu desenvolvimento, como por exemplo o desenvolvimento de novos classificadores para complementar os que foram implementados. Uma alternativa ao classificador Naïve Bayes clássico é a variante multinomial ("multinomial Naïve Bayes") que alguns autores defendem como sendo melhor que o modelo original, principalmente com grandes colecções de documentos de treino (Witten and Frank, 2005, Schneider, 2005).

Como alternativa aos classificadores baseados no modelo de Bayes existem outras soluções que aplicam técnicas de agrupamento para fazer a tarefa de classificação de documentos. Uma vez que o modelo aqui apresentado já faz uso destas técnicas para definição e organização de categorias a sua adaptação para agrupar documentos seria uma tarefa simples. Do mesmo modo que a ferramenta de criação de taxonomias agrupa tematicamente o conjunto de documentos de treino também pode agrupar um novo documento em conjunto com os documentos de treino. O grupo a que o documento fosse associado seria considerado como a sua categoria.

No entanto, o uso de métodos de agrupamento para classificação de documentos nunca seria um processo totalmente automático pois também estes métodos possuem problemas (c.f. Capítulo 4.5) pelo que, a supervisão humana seria sempre necessária. Outro problema que teria de ser ultrapassado é a dificuldade em manter persistentes os resultados de agrupamentos anteriores. O método de persistência sugerido neste método não é possível aplicar no agrupamento pelo que outras soluções teriam de ser equacionadas como por exemplo, o recurso a ferramentas de recuperação de informação (e.g. Lucene (Jakarta, 2004)).

A nível da taxonomia de categorias também existe trabalho a fazer. Neste momento o modelo de classificação apenas permite que as sub-categorias de documentos suportem herança simples de fragmentos. O que isto quer dizer é que uma categoria de documentos apenas pode herdar definição de fragmentos de outra categoria ascendente se não possuir fragmentos próprios, não sendo possível combinar fragmentos próprios com outros herdados. Uma alteração futura a este modelo passa por permitir que as sub-categorias de documentos possam usar tanto os seus fragmentos como os fragmentos da categoria ascendente.

Outra possível alteração à taxonomia de categorias seria permitir a criação de grupos de categorias. Esta alteração pode ser interessante porque a classificação de um documento implica calcular a probabilidade de esse documento pertencer a cada uma das categorias possíveis. Se o número de categorias for baixo esse cálculo é rápido, mas se o número de categorias for elevado o tempo de cálculo pode-se tornar demasiado longo. Tendo em conta esta situação pode ser interessante permitir a definição de grupos de categorias de modo a restringir as categorias que são tidas em conta para a classificação do documento.

Inicialmente foi estudada uma alternativa ao XML para representar a taxonomia de categorias e para armazenar a informação obtida. O XML foi escolhido mas a capacidade de modelação do conhecimento de forma abstracta do RDF, pode levar a que tanto a taxonomia de categorias como o forma de armazenar a informação obtida não fique tão comprometida com um formato. Assim o RDF poderá eventualmente ser usado

para substituir o XML tanto a nível da taxonomia de categorias como do resultado da classificação.

Apêndice A

Definições

Teoria de decisão de Bayes. A teoria de decisão bayesiana é muito utilizada em problemas de classificação de padrões. A ideia da teoria de decisão de Bayes é a de fazer as escolhas que minimizem o risco de classificar erradamente algo. O problema da decisão deve ser colocado em termos probabilísticos e os valores de todas as probabilidades relevantes são conhecidos ou calculáveis.

Como exemplo, seja Ω o conjunto de rótulos de categorias $\Omega = \{\omega_1, \omega_2\}$ nas quais o objecto x pode ser catalogado e seja $\Pr(\omega_i), i = \{1, 2\}$ a função de probabilidade *à priori* de ω onde $0 \geq \Pr(\omega_i) \leq 1$ e $\sum_{i=1}^{2} \Pr(\omega_i) = 1$.

Para que um classificador minimize o erro seria necessário classificar sempre os documentos segundo a classe com maior probabilidade *à priori*. Mas este seria um classificador com um desempenho bastante modesto pois o erro de classificação estaria sempre relacionado com a probabilidade de ocorrência de cada uma das categorias $\Pr(\omega_i)$.

Para criar um classificador com melhor desempenho é necessário optar por uma solução que envolva um vector de características $x \in R^n$ que descreva o documento (objecto). A classificação de um objecto passa a ser dependente das características desse objecto, e é descrita por uma função densidade de probabilidade condicional $\Pr(x|\omega_i) > 0$ também por vezes chamada de função de verosimilhança ("likehood") da categoria ω_i em relação ao documento descrito por x. Neste momento já possuimos a probabilidade *à priori* $\Pr(\omega_i)$ e a probabilidade condicional $\Pr(x|\omega_i)$, mas o que pretendemos saber é $\Pr(\omega_i|x)$ ou seja, a probabilidade de a classe ser ω_i dado x. Para realizar este cálculo recorre-se à formula de Bayes:

$$\Pr(\omega_i|x) = \frac{\Pr(x|\omega_i)\Pr(\omega_i)}{\Pr(x)} \tag{A.1}$$

onde

$$\Pr(x) = \sum_{i=1}^{2} \Pr(x|\omega_i) \Pr(\omega_i). \tag{A.2}$$

A formula de Bayes pode ser descrita informalmente do seguinte modo:

$$\Pr \textit{ à posteriori} = \frac{\Pr \textit{ verosimilhança} \times \Pr \textit{ à priori}}{\Pr \textit{evidencia}} \tag{A.3}$$

Dado que $\Pr(x)$ é um factor comum a todas as classes ω_i este pode ser simplesmente ignorado para efeitos de cálculo da probabilidade *à posteriori*. Pode-se assumir que se conhece *à priori* o valor da probabilidade de ocorrência de uma classe $\Pr(\omega_i)$ mas no caso de esta probabilidade ser desconhecida é facilmente estimada com recurso à colecção de treino $\Pr(\omega_i) \approx N_i/N$ em que N é o número de elementos da colecção de treino e N_i é o número de elementos classificados com a classe i.

Para determinar a classe com que um objecto deve ser classificado podemos usar a regra de classificação de Bayes,

$$\text{Se } \Pr(\omega_1|x) > \Pr(\omega_2|x), \text{ x classificado com } \omega_1$$

$$\text{Se } \Pr(\omega_1|x) < \Pr(\omega_2|x), \text{ x classificado com } \omega_2$$

Para justificar esta decisão vamos calcular a probabilidade de erro quando tomamos uma decisão,

$$\Pr(erro|x) = \begin{cases} \Pr(\omega_1|x) & \text{se classificarmos com } \omega_2 \\ \Pr(\omega_2|x) & \text{se classificarmos com } \omega_1. \end{cases}$$

Para um dado x minimiza-se a probabilidade de erro classificando com ω_1 se $\Pr(\omega_1|x) > \Pr(\omega_2|x)$ ou classificando com ω_2 caso contrário. Por exemplo se $\Pr(\omega_1|x) = 0,2$ e $\Pr(\omega_2|x) = 0,8$, então classificar x como ω_1 tem uma probabilidade de erro de $\Pr(erro|x) = 0,8$, e classificar x como ω_2 tem um probabilidade de erro de $\Pr(erro|x) = 0,2$. (Duda et al., 2001, Theodoridis and Koutroumbas, 2006, Kuncheva, 2006).

Independência estatística. As variáveis x e y dizem-se estatisticamente independentes se e só se

$$\Pr(x, y) = \Pr(x) \Pr(y)$$

Pode-se entender a independência estatística do seguinte modo. Supondo que $p_i = \Pr x = v_i$ é fracção de tempo que $x = v_i$ e que $q_j = \Pr y = w_j$ é a fracção de tempo que $y = w_j$. Considere-se as situações em que $x = v_i$. Se continuar a ser verdade que a fracção de tempo em que $y = w_j$ tem o mesmo valor que q_j, então conclui-se que o valor de x não trouxe informação adicional sobre os possíveis valores de y; nesse sentido y é independente de x. Finalmente se x e y são estatisticamente independentes, é claro que a fracção de tempo em que um especifico par de valores (v_i, w_j) ocorre vem igual ao produto das duas fracções $p_i q_j = \Pr(v_i) \Pr(w_j)$ (Milho, 2003).

Propriedade multiplicativa dos logaritmos.

Definition A.1. O logaritmo de um produto é igual a soma dos logaritmos dos seus factores (BETTINGER and ENGLUND, 1963). A forma logarítmica é:

$$\log_b(m \times n) = \log_b m + \log_b(n)$$

. Para provar esta equação seja, $x = log_b m$ e $y = \log_b n$ então

$$m = b^x$$

e

$$n = b^y$$

. Multiplicando temos

$$m \times n = b^{y+x}$$

. Portanto,

$$\log_b(m \times n) = x + y = \log_b m + \log_b(n)$$

Apêndice B

Interfaces gráficas

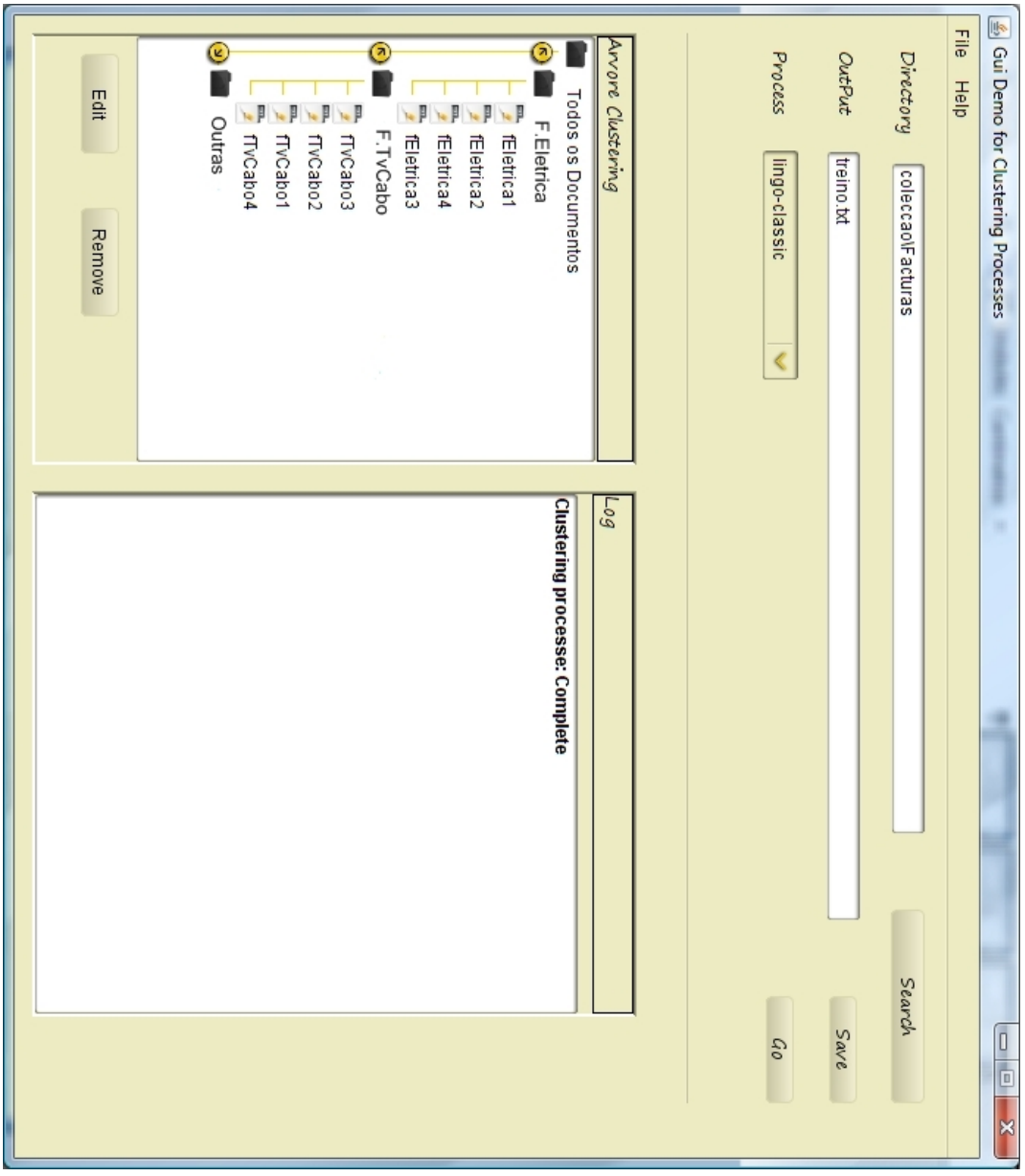

FIGURA B.1: Interface Visual: Protótipo

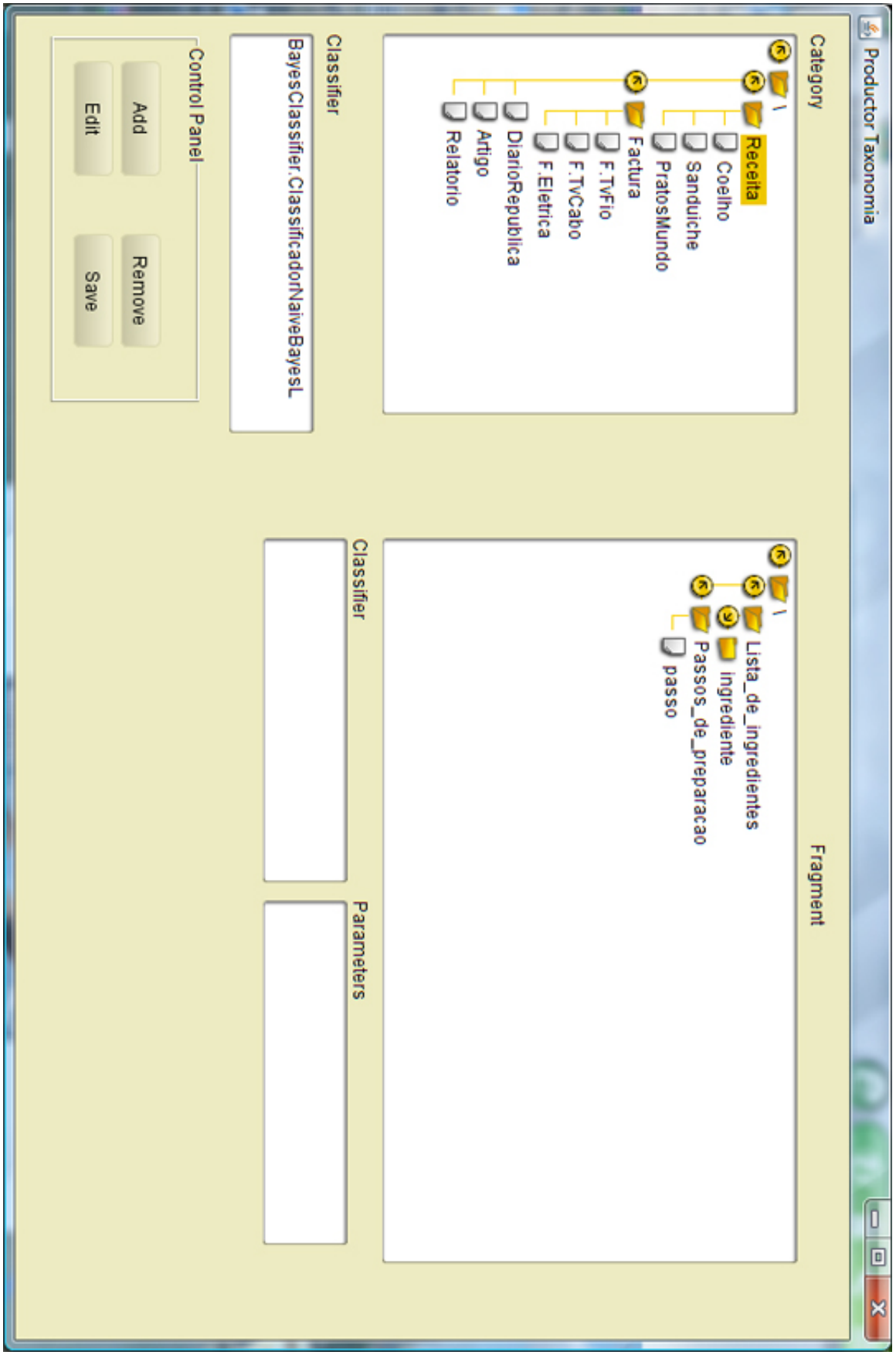

FIGURA B.2: Produtor Taxonomia Manual

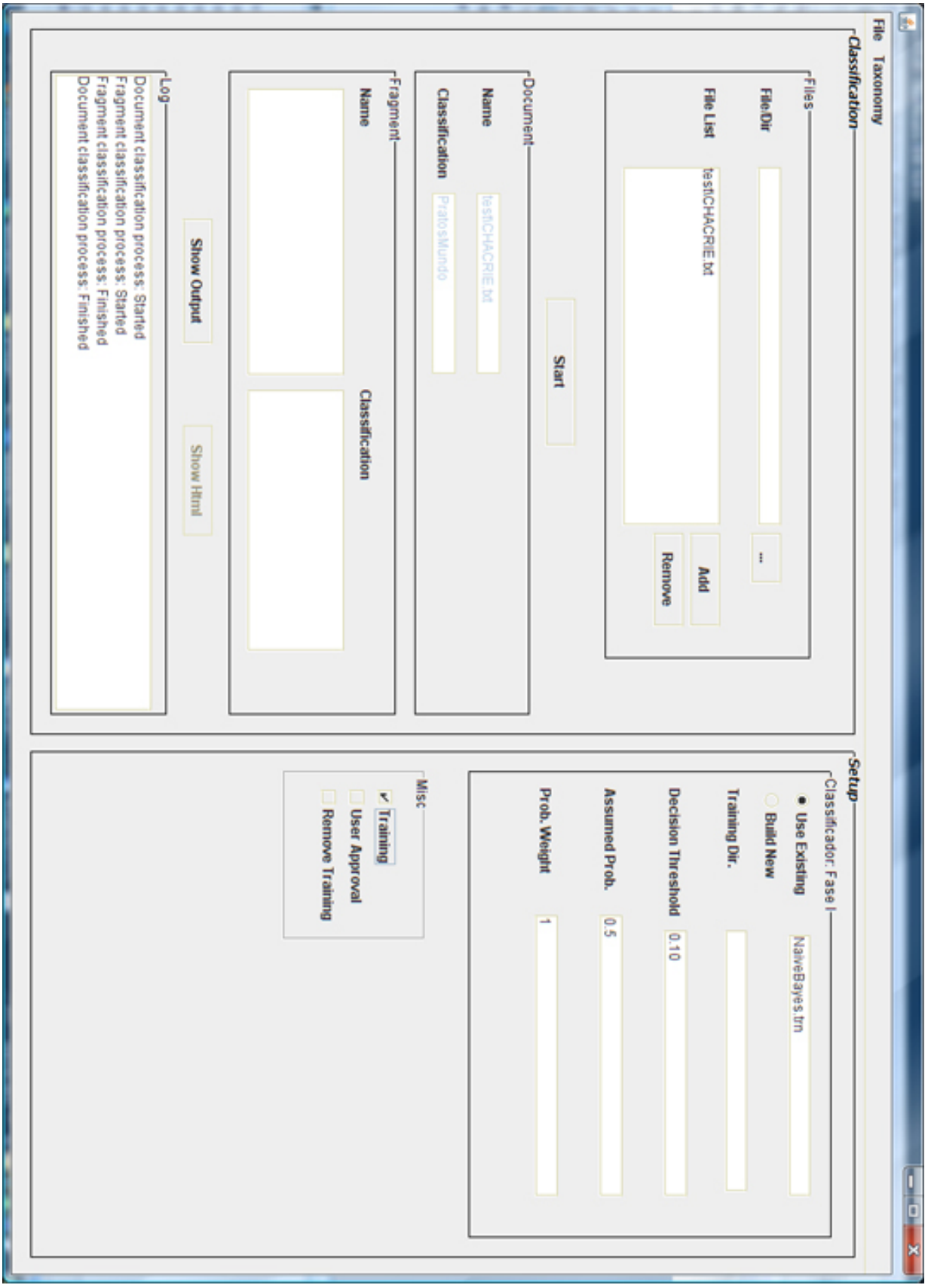

FIGURA B.3: Produtor Taxonomia Automático

Bibliografia

Aitken, J. (2002). Learning Information Extraction Rules: An Inductive Logic Programming approach. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 355–359. http://citeseer.ist.psu.edu/586553.html.

Antoniou, G. and van Harmelen, F. (2004). *A Semantic Web Primer*. The MIT Press, London: England.

Berkhin, P. (2002). Survey of clustering data mining techniques.

BETTINGER, A. K. and ENGLUND, J. A. (1963). *Algebra and Trigonometry*. INTERNATIONAL TEXTBOOK COMPANY, Scranton, Pennsylvania.

Broekstra, J., Kampman, A., and van Harmelen, F. (2002). Sesame: A generic architecture for storing and querying rdf and rdf schema. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 54–68, London, UK. Springer-Verlag.

Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. *Proceedings of the 23rd international conference on Machine learning*, pages 161–168.

Chang, C. and Lui, S. (2001). IEPAD: information extraction based on pattern discovery. *Proceedings of the 10th international conference on World Wide Web*, pages 681–688.

Consortium, T. U. and Allen, J. (2006). *The Unicode Standard, Version 5.0*. Addison-Wesley Professional, fifth edition.

Cunningham, H. (2000). *Software Architecture for Language Engineering*. PhD thesis, University of Sheffield. `http://gate.ac.uk/sale/thesis/`.

Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. (2002). GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*.

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Ursu, C., Dimitrov, M., Dowman, M., Aswani, N., Roberts, I., Li, Y., et al. (2008). Developing Language Processing Components with GATE Version 4 (a User Guide). *Change*, 16(2.1):16.

Domingos, P. and Pazzani, M. (1997). On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29(2):103–130.

Duda, R., Hart, P., and Stork, D. (2001). *Pattern classification*. Wiley New York.

Ferreira, J. (2008). Procura de padrões em documentos para extracção e classificação de informação. In *Quartas Jornadas de Engenharia de Electrónica e Telecomunicações e de Computadores*.

Gotembniak, K. (2005). HAOG-STC Algorithm. In *Carrot2*. http://project.carrot2.org.

Heninger, A. (2004). Analyzing unicode text with regular expressions. In *Proceedings of the 26th Internationalization and Unicode Conference*.

Jakarta, A. (2004). Apache Lucene-a high-performance, full-featured text search engine library.

Kiryakov, A., Popov, B., Terziev, I., Manov, D., and Ognyanoff, D. (2004). Semantic annotation, indexing, and retrieval. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(1):49–79.

Kuncheva, L. I. (2006). *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience.

Lang, N. (2004). A tolerance rough set approach to clustering web search results. *Faculty of Mathematics, Informatics and Mechanics, Warsaw University*.

Manning, C., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Marques, J. S. (2005). *Reconhecimento de Padrões. Mètodos Estatísticos e Neuronais*. ISTPress, Lisboa, 2ª edition.

McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification. *AAAI-98 Workshop on Learning for Text Categorization*, 752.

Meier, W. (2003). eXist: An Open Source Native XML Database. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 169–183.

Milho, I. (2003). Teoria de Probabilidades.

Morrison, D. (1968). Patricia - Practical Algorithm To Retrieve Information Coded in Alphanumeric. *Journal of the ACM (JACM)*, 15(4):514–534.

Osinski, S. (2003). *An Algorithm for Clustering of Web Search Results*. PhD thesis, Master thesis, Poznan University of Technology.

Osinski, S. and Weiss, D. (2005a). Carrot^2: Design of a flexible and efficient web information retrieval framework. In *AWIC*, pages 439–444.

Osinski, S. and Weiss, D. (2005b). A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems*, 20(3):48–54.

Pop, I. (2006). An approach of the naive bayes classifier for the document classification. In *General Mathematics Vol 14 No 4*, pages 135–138, Sibiu, Romania. UNIVERSITY OF SIBIU - FACULTY OF SCIENCES - DEPARTMENT OF MATHEMATICS Electronic Edition.

Schneider, K.-M. (2005). Techniques for improving the performance of naive bayes for text classification. In *In Proceedings of CICLing 2005*, pages 682–693.

Schockaert, S. (2004). *Het clusteren van zoekresultaten met behulp van vaagmieren (clustering of search results using fuzzy ants)*. PhD thesis, Master thesis, University of Ghent.

Sebastiani, F. and Ricerche, C. N. D. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34:1–47.

Segaran, T. (2007). *Programming Collective Intelligence Building Smart Web 2.0 Applications*, pages 117–140. O'Reilly.

Stefanowski, J. and Weiss, D. (2003). Carrot and language properties in web search results clustering. In *AWIC*, pages 240–249.

Team, W. A., Dix, C., Kovack, R., Rafter, J., Hunter, D., and Pinnock, J. (2001). *Beginning XML*. Wrox Press Ltd., Birmingham, UK, UK.

Theodoridis, S. and Koutroumbas, K. (2006). *Pattern Recognition*. Academic Press.

Turchin, A., Kolatkar, N., Grant, R., Makhni, E., Pendergrass, M., and Einbinder, J. (2006). Using Regular Expressions to Abstract Blood Pressure and Treatment Intensification Information from the Text of Physician Notes. *Journal of the American Medical Informatics Association*, 13(6):691–695.

Weiss, D. and Stefanowski, J. (2003). Web search results clustering in polish: Experimental evaluation of carrot. In *In IIS03*, pages 209–220.

Witten, I. and Frank, E. (2005). Data mining: Practical machine learning tools and techniques.

Zhang, H. (2004). The optimality of naive Bayes. *Proceedings of the Seventeenth Florida Artificial Intelligence Research Society Conference*, pages 562–567.