Normalização Automática de Descrições de Hotéis

Nuno Filipe Roque Miranda

Orientador: Paulo Miguel Duarte Torres Quaresma
Co-Orientador: Teresa Cristina de Freitas Gonçalves

Universidade de Évora
Mestrado de Engenharia Informática

Outubro 2010

Normalização Automática de Descrições de Hotéis

Nuno Filipe Roque Miranda

Orientador: Paulo Miguel Duarte Torres Quaresma
Co-Orientador: Teresa Cristina de Freitas Gonçalves

Universidade de Évora
Mestrado de Engenharia Informática

Outubro 2010

Resumo

As descrições de produtos turísticos na área da hotelaria, aviação, rent-a-car e pacotes de férias baseiam-se sobretudo em descrições textuais em língua natural muito heterogénea com estilos, apresentações e conteúdos muito diferentes entre si. Uma vez que o sector do turismo é bastante dinâmico e que os seus produtos e ofertas estão constantemente em alteração, o tratamento manual de normalização de toda essa informação não é possível.

Neste trabalho construiu-se um protótipo que permite a classificação e extracção automática de informação a partir de descrições de produtos de turismo. Inicialmente a informação é classificada quanto ao tipo. Seguidamente são extraídos os elementos relevantes de cada tipo e gerados objectos facilmente computáveis. Sobre os objectos extraídos, o protótipo com recurso a modelos de textos e imagens gera automaticamente descrições normalizadas e orientadas a um determinado mercado. Esta versatilidade permite um novo conjunto de serviços na promoção e venda dos produtos que seria impossível implementar com a informação original.

Este protótipo, embora possa ser aplicado a outros domínios, foi avaliado na normalização da descrição de hotéis. As frases descritivas do hotel são classificadas consoante o seu tipo (Local, Serviços e/ou Equipamento) através de um algoritmo de aprendizagem automática que obtém valores médios de cobertura de 96% e precisão de 72%. A cobertura foi considerada a medida mais importante uma vez que a sua maximização permite que não se percam frases para processamentos posteriores.

Este trabalho permitiu também a construção e população de uma base de dados de hotéis que possibilita a pesquisa de hotéis pelas suas características. Esta funcionalidade não seria possível utilizando os conteúdos originais.

Palavras-chave: Extracção de Informação, Representação Ontológica, Expressões Regulares , Aprendizagem Automática, Classificação de Textos, Domínio do Turismo.

Automatic Normalization of Hotel Descriptions

Abstract

The description of tourism products, like hotel, aviation, rent-a-car and holiday packages, is strongly supported on natural language expressions. Due to the extent of tourism offers and considering the high dynamics in the tourism sector, manual data management is not a reliable or scalable solution. Offer descriptions – in the order of thousands – are structured in different ways, possibly comprising different languages, complementing and/or overlap one another.

This work aims at creating a prototype for the automatic classification and extraction of relevant knowledge from tourism-related text expressions. Captured knowledge is represented in a normalized/standard format to enable new services based on this information in order to promote and sale tourism products that would be impossible to implement with the raw information.

Although it could be applied to other areas, this prototype was evaluated in the normalization of hotel descriptions. Hotels descriptive sentences are classified according their type (Location, Services and/or Equipment) using a machine learning algorithm. The built setting obtained an average recall of 96% and precision of 72%. Recall considered the most important measure of performance since its maximization allows that sentences were not lost in further processes.

As a side product a database of hotels was built and populated with search facilities on its characteristics. This ability would not be possible using the original contents.

Keywords: Information Extraction, Ontology Representation, Regular Expressions, Machine Learning, Text Classification, Tourism domain.

Crer e Querer para Vencer.

Agradecimentos

Gostaria de agradecer em primeira instância aos meus orientadores, Professor Paulo Quaresma e Professora Teresa Gonçalves, pois sem o seu apoio e disponibilidade total para guiar, orientar e esclarecer dúvidas, este trabalho não teria sido possível realizar.

Gostaria também de agradecer a outros Professores do Departamento de Informática que foram importantes no meu percurso e formação académica, nomeadamente ao Professor Luís Rato, ao Professor José Saias, à Professora Irene e ao Professor Francisco Coelho.

Fora do Departamento de Informática, gostaria de agradecer à Professora Birgit Arnholdt-Schmitt e ao Professor Paulo Correia.

Fora do âmbito da Universidade, mas não menos importante na minha formação, gostaria de agradecer à Professora Lurdes Granadeiro e ao Professor Jorge Vinhais pela sua incansável vontade e disponibilidade de ajudar até altas horas da noite.

Gostaria também de agradecer ao Ricardo Raminhos e à VIATECLA, peças chave na realização deste trabalho.

Numa vertente mais pessoal, gostaria de agradecer aos meus Pais, Tina e Francisco que possibilitaram e patrocinaram esta longa jornada, não esquecendo obviamente a minha irmã Sílvia e cunhado Paulo.

Gostaria ainda de agradecer aos meu colegas de "guerra" pelos bons momentos e espírito de ajuda e camaradagem durante o curso; João Saraiva, Rui Caridade, Carlos Madeira, Paulo Pires e um grande e especial destaque para o Caniço (Henrique Silva).

Para terminar e não menos importante, o agradecimento muito especial à minha namorada Patrícia pela sua paciência e compreensão do meu afastamento na parte final da tese, e claro, sem esquecer o seu amor e carinho constante.

Índice

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Actualmente a nossa sociedade auto designa-se por "A sociedade da informação". No entanto tal designação é frequentemente usada num sentido lato e de pura mediatização sem grande análise em profundidade sobre a qualidade/quantidade e a real utilidade da informação que dispomos.

É certo que nos últimos 50 anos acumulamos e coleccionamos mais informação do que na restante história da humanidade. No entanto surgem dúvidas, quanto ao real proveito de tais quantidades massivas de informação. Pois pior do que não ter informação é ser inundado por informação e não saber "navegar"nela.

- De que interessa ter uma enorme Biblioteca com milhares de obras se estas não se encontrarem devidamente identificadas, organizadas, catalogadas?
- De que interessa ter uma enorme Biblioteca se não soubermos ler o seu conteúdo?
- De que interessa ter uma enorme Biblioteca onde os livros se encontram dispersos numa entropia tão elevada que é impossível estabelecer qualquer relação ou ligação entre conceitos e conteúdos das informações neles contidas?

Tais questões não se aplicam apenas a uma Biblioteca desorganizada, aplicam-se também a todo o conhecimento humano, e mais profundamente ao próprio ser humano que não é mais que um conjunto de informações, conceitos, emoções, sentimentos e vivências interligadas entre si e indexadas a um fluxo temporal. Tal cruzamento de emoções, sentimentos e vivências formam em si a unicidade de um ser humano. Pois mais importante do que ter apenas dados em qualquer contexto, é ter acesso omnisciente dos dados pretendidos e sobretudo dos conceitos provenientes do cruzamento desses mesmos dados, obtendo-se assim informação e não apenas dados.

A origem da palavra "Informática", informação + automática, introduzida em 1957 pelo cientista Alemão Karl Steinbuch referindo-se ao conceito de processar informação automaticamente, conceito esse que encaixa perfeitamente no contexto deste trabalho que se propõe a recolher, processar e normalizar informação automaticamente.

1.1 Motivações

O mercado do turismo na vertente de comercio electrónico[1] em que os produtos são apresentados, promovidos e vendidos por via electrónica, tem registado nos últimos anos um grande aumento de procura em relação aos balcões tradicionais de venda de produtos e serviços de turismo. Este mercado além de permitir ao consumidor escolher os seus destinos de férias confortavelmente em casa, permite fazer em tempo real a comparação de bens e produtos entre vários operadores, algo que seria impossível nos balcões tradicionais de venda. A vertente de comércio electrónico do turismo possui a mais-valia que permite aos operadores turísticos efectuar menores taxas devido a terem menos despesas. Desta forma é possível ao consumidor final ter produtos e serviços mais acessíveis.

Tendo em conta o forte aumento da procura de bens de comercio electrónico de turístico e tendo como foco, aumentar a eficácia e qualidade das plataformas online que disponibilizam esses serviços, é imperativo superar algumas limitações técnicas ocorrentes da conjugação dos vários intervenientes do mundo do turismo e criar formas mais eficazes e agradáveis de apresentar os bens e produtos turísticos.

Por exemplo, e restringindo apenas ao domínio da hotelaria, quando um consumidor pesquisa nas ferramentas de comercio electrónico por hotéis num determinado local e numa determinada data, submete uma consulta e o sistema devolve uma lista de hotéis que cumprem esses requisitos. Cada hotel, é acompanhado por algumas fotografias e uma descrição textual das suas características, tais como serviços, equipamento disponibilizado, pontos de interesse nas proximidades do hotel, características arquitectónicas, etc.

No entanto essas descrições textuais em língua natural não seguem qualquer padrão ou estrutura homogeneizada, nem em forma nem em conteúdo, uma vez que dependem da "inspiração"ou do marketing do indivíduo que as escreveu. Em casos limites, é possível ter a mesma informação elementar descrita de modos completamente diferentes, como por exemplo:

a) Os espectaculares quartos têm uma deslumbrante casa de banho e uma boa climatização devido ao eficiente Ar Condicionado que possuem.

[1] Do Inglês, *e-commerce*.

b) Os quartos têm W.C. e A.C.

Ou ainda:

a) O Hotel dispõe de um Restaurante requintado e com ementa à la carte, um Bar com os mais diversos tipos de cocktails e snacks e dispõe ainda uma Sala de Reuniões completamente equipada e muito confortável.

b) O Hotel dispõe de Restaurante, Bar e uma Sala de Reuniões.

No primeiro exemplo apresentado, os elementos informativos chave são casa de banho e ar condicionado, na primeira variante esses atributos aparecem escritos por extenso e com algum floreado linguístico. Na segunda variante os mesmos elementos aparecem directamente sem qualquer floreado linguístico e de modo abreviado.

O segundo exemplo onde os atributos chave são restaurante, bar e sala de reuniões, os atributos já se encontram escritos da mesma maneira, no entanto, um dos casos é mais simples e directo enquanto o outro tenta dar uma melhor impressão a partir dos mesmos atributos. Por exemplo, "sala de reuniões muito confortável"é algo bastante subjectivo e que não acrescenta real valor aos atributos existentes no hotel, neste caso o que interessa é que o hotel tem sala de reuniões sendo fúteis descrições que lhe acrescentem valor subjectivo, como o confortável, aprazível, bonita, etc.

Muitas vezes, as várias descrições de hotéis são traduzidas automaticamente de país para país, de idioma para idioma. Isto acaba por levar a outro problema, já que as frases podem não estar correctamente escritas a nível de sintaxe e de semântica, e tendo constantes falhas a nível dos diacríticos. É então imperativo arranjar solução para esse problema durante a extracção e normalização da informação.

1.2 Objectivos

Os objectivos propostos e estipulados para este trabalho podem-se dividir em dois níveis. A nível mais teórico, pretende-se estudar:

- Os algoritmos de classificação existentes;
- As formas de representação de conhecimento recorrendo a ontologias;
- As formas de extracção de informação a partir de bases textuais.

Já a nível prático os objectivos passam por:

- Testar e seleccionar os algoritmos de aprendizagem mais adequados para a classificação;
- Classificar automaticamente as frases presentes nas descrições dos hotéis em relação ao tema (equipamento, serviço, localização);
- Criar uma estrutura ontológica para representar os conceitos e as relações inter-conceitos no âmbito da hotelaria;
- Criar um mecanismo que recorrendo à informação presente na ontologia consiga extrair as entidades relevantes das frases previamente classificadas;
- Elaborar um mecanismo que a partir das entidades extraídas e normalizadas, crie automaticamente descrições padronizadas e que possam tomar "orientações"especificas de acordo com o mercado alvo.

1.3 Principais Contribuições

Como principais contribuições, este estudo:

- Define uma arquitectura modular e genérica para a tarefa de extracção de informação de descrições textuais.
- Propõe e avalia uma solução baseada em:
  - Técnicas de classificação automática capazes de classificar frases de acordo com o tipo de informação que contêm.
  - Expressões regulares capazes de extrair os diversos itens que descrevem o tipo de informação previamente classificado.
  - Distancias de Levenshtein capazes de corrigir erros ortográficos presentes nas descrições textuais originais.
- Cria uma aplicação capaz de extrair informação de descrições textuais de hotéis, manipulá-la e construir novas descrições normalizadas e versáteis. Direccionadas por exemplo, para diferentes tipos de clientes.

Este estudo foi direccionado para a normalização de descrições de hotéis. No entanto, a sua aplicação a outros domínios apenas requer a substituição do modelo de classificação e das expressões regulares que, em conjunto, serão capazes de extrair outro tipo de informação turística.

Quanto à solução proposta, a utilização de técnicas de classificação automática mostrou ser eficaz permitindo fazer uma pré-selecção das frases a serem processadas pelas técnicas dedicadas à extracção de informação. A utilização de expressões regulares

e distâncias de Levenshtein também se revelaram capazes de extrair a informação existente nas descrições textuais

Relativamente à aplicação construída, esta mostrou-se fundamental na construção de uma base de dados que serve de apoio a uma aplicação comercial da VIATECLA e que torna possível a pesquisa de hotéis através das suas diferentes características.

1.4 Contextualização do Trabalho

O trabalho aqui exposto foi desenvolvido no âmbito de um projecto de colaboração, que resultou no Laboratório de Excelência .NET em Évora. Este laboratório é fruto de um protocolo tripartido entre a VIATECLA [53], a Universidade de Évora e a Microsoft com o objectivo de introduzir a experiência e o conhecimento académico em contexto empresarial. Neste sentido, o Laboratório surge como um espaço multi-disciplinar que conta com a presença de docentes, investigadores e alunos (projectos de fim de curso, mestrado e doutoramento).

O projecto "Normalização Automática de Descrições de Hotéis"apresenta-se como o primeiro projecto realizado no Laboratório de Évora, tendo em vista a extracção automática das características mais relevantes presentes nas descrições de hotéis existentes no KEYforTRAVEL [54] . A solução encontrada utiliza tecnologias nas áreas da aprendizagem, classificação automática, extracção de informação, processamento de língua natural e representações ontológicas.

O KEYforTRAVEL é uma solução desenvolvida pela VIATECLA, que providencia uma resposta rápida e eficaz, na interligação dos vários participantes da indústria do turismo, sendo assegurada a visualização e troca de informação de cada participante, assim como das várias áreas do processo de venda do produto.

Fundamental a todo este processo é a componente inicial de pesquisa que permite obter e normalizar informação de produtos de turismo de diversas fontes heterogéneas. É aqui que se enquadra o projecto "Normalização de Descrições de Hotéis"com o intuito de facilitar o tratamento dessas fontes heterogéneas.

1.5 Abordagem Académica vs Empresarial

Como já foi referido, este projecto nasceu de uma colaboração entre o meio académico (Universidade de Évora) e o meio empresarial (VIATECLA). Tal diferença de meios gerou notoriamente alguns conflitos e atritos na forma de trabalhar.

O ambiente académico incentiva a um desenvolvimento mais ponderado, pausado e olhando em todas as direcções em redor das problemáticas emergentes ao desenvolvimento do projecto, permitindo assim "olhar em redor"antes de se dar o próximo passo, de modo a permitir pesar e avaliar os vários caminhos alternativos de resolver os problemas que foram surgindo.

Por sua vez o ambiente empresarial incentiva a um desenvolvimento mais célere, tendo sempre em conta os prazos e objectivos estabelecidos, abdicando muitas vezes da observação em redor de um problema para descobrir caminhos alternativos. Prefere, também, manter-se focada no objectivo final e percorrer o percurso em linha recta, ignorando muitas vezes, atalhos e melhores caminhos paralelos ao percurso inicialmente traçado.

A junção destas duas abordagens durante este projecto nem sempre foi pacífica, o que levou à existência de alguns atritos. No entanto, após terem sido superados alguns problemas iniciais, o desenvolvimento do projecto acabou por juntar o melhor dos dois mundos, mantendo uma visão abrangente e periférica das varias abordagens a seguir, e ao mesmo tempo, o foco nos objectivos finais.

1.6 Organização da Dissertação

Esta dissertação está dividida em diversos capítulos que retratam partes distintas do trabalho realizado.

- **Capítulo 1:** Faz a introdução do trabalho realizado assim como a sua contextualização nos projectos do Laboratório .Net e da VIATECLA. Fala ainda das motivações e objectivos esperados.
- **Capítulo 2:** Faz a introdução a diversos conceitos técnicos e científicos essenciais utilizados ao longo deste trabalho, e sem os quais seria impossível compreender a problemática que deu origem a este trabalho assim como a construção da solução. O Capítulo também faz a introdução às ferramentas utilizadas durante a realização deste trabalho, tanto ferramentas utilizadas directamente e que fizeram parte constituinte do trabalho, como de ferramentas de uso indirecto em que apenas se empregou os seus resultados neste trabalho.
- **Capítulo 3:** São apresentados diversos trabalhos de diferentes autores que se inserem na temática deste trabalho e que serviram de inspiração e estado da arte para guiar a realização do trabalho. Este estudo está dividido em duas partes, uma dedicada à classificação de textos em língua natural e outra dedicada à extracção de informação de textos também em língua natural.

- **Capítulo 4:** São apresentados em detalhe os diversos passos tomados na realização e construção deste trabalho e do protótipo resultante.
- **Capítulo 5:** São apresentados os diversos resultados sobre os testes efectuados. Testes para basear decisões quanto ao caminho e escolhas a tomar, e testes sobre os resultados finais obtidos.
- **Capítulo 6:** Neste capítulo é feita a conclusão sobre a realização do trabalho assim como a indicação de melhoramentos futuros.

Capítulo 2

Conceitos, Metodologias e Ferramentas

Neste capítulo, são introduzidos diversos conceitos técnicos e científicos, essenciais e utilizados ao longo deste trabalho e sem os quais seria impossível compreender e desenvolver este trabalho.

2.1 Scrum

Scrum [49] é uma técnica de desenvolvimento ágil de software que se caracteriza por ser iterativa e incremental, tendo como objectivo normalizar e estruturar o desenvolvimento de aplicações complexas.

Originalmente o *Scrum* foi uma técnica desenvolvida num contexto de desenvolvimento de projectos industriais, que por sua vez foi buscar inspiração à táctica *Scrum* de *Rugby*, que se baseia em progredir no campo de jogo através de várias corridas[1] que vão sendo reajustados à medida da evolução do jogo.

O *Scrum* aplicado a projectos de desenvolvimento de software tem três intervenientes: o dono do produto, o Chefe de *Scrum*[2] e a equipa de desenvolvimento.

O dono do produto ou da aplicação de software é a pessoa ou pessoas para quem será desenvolvida a aplicação. O Chefe de *Scrum* é o gestor que gere o projecto e a equipa de desenvolvimento, assim como as metodologias, tecnologias e abordagens usadas durante o seu desenvolvimento prático. A equipa de desenvolvimento é responsável pelo desenvolvimento e implementação da aplicação.

[1] Do Inglês, *Sprints*.
[2] Do Inglês, *Scrum Master*.

No desenvolvimento de uma aplicação de software através da metodologia do *Scrum* são efectuados vários passos. No inicio do projecto, o Chefe de *Scrum* conjuntamente com o dono do produto reúnem-se e estabelecem uma Lista de Requisitos[3] de produto, que é uma lista de requisitos/objectivos de alto nível para a aplicação.

O desenvolvimento da aplicação é dividido em várias etapas, sendo denominadas por corridas[4]. Por norma cada corrida tem a duração compreendida entre duas a quatro semanas, sendo entregue no fim de cada, uma parte concluída do projecto. No final das várias corridas obter-se-á a totalidade dos objectivos do projecto.

No início de cada corrida é efectuada uma reunião com a equipa de desenvolvimento onde é definida uma lista de objectivos denominada de Lista de Requisitos da Corrida[5].

Essa lista é um conjunto de tópicos concretos de implementação, que cobrem uma parte dos objectivos de alto nível definidos na Lista de Requisitos do produto.

No final de cada corrida é também efectuada outra reunião onde são apresentados resultados ao dono do produto e onde se discutem as dificuldades e problemas que surgiram durante a corrida.

Durante as corridas, o Chefe de *Scrum* efectua diariamente uma breve reunião de 15 minutos com a equipa de desenvolvimento. Essas reuniões denominam-se Reunião Diária de *Scrum*[6]. Aí debatem-se os avanços que foram efectuados desde a ultima reunião, o trabalho que vai ser efectuado até à próxima reunião, e se surgiram dificuldades tanto no trabalho efectuado como no trabalho a efectuar.

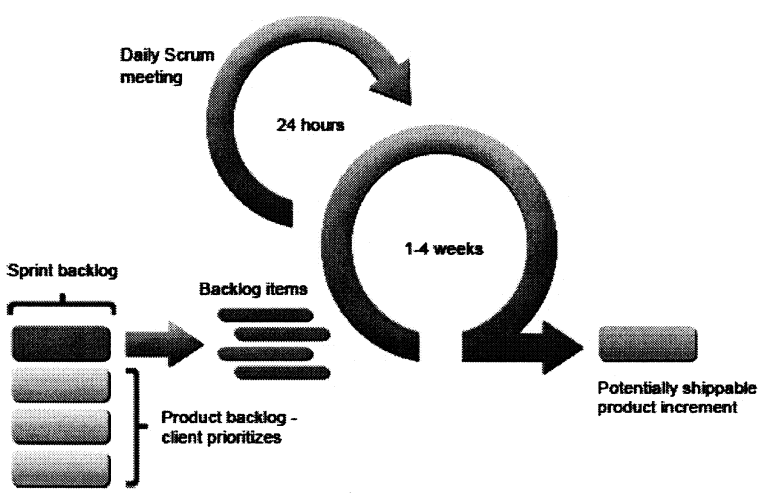

Figura 2.1: Arquitectura Scrum.

Esta metodologia foi utilizada durante todo o processo de desenvolvimento do projecto

[3]Do Inglês, *Backlog.*
[4]Do inglês, *sprints.*
[5]Do Inglês, *Sprint Backlog.*
[6]Do inglês, *Daily Scrum Meeting.*

de Normalização Automática de Descrições de Hotéis, em que por norma a duração das corridas foi de duas semanas. O papel de Dono do Produto foi partilhado pela Universidade de Évora e pela VIATECLA; já o papel de Chefe de *Scrum* foi efectuado pela VIATECLA.

2.2 Ontologias e Web Ontology Language

Uma ontologia é um modelo de dados que representa um conjunto de conceitos dentro de um determinado domínio bem como as relações entre esses conceitos, as suas propriedades e também pode representar as restrições dos conceitos, hierarquia, relações e propriedades nesse domínio.

Desta forma uma ontologia permite que sejam feitos levantamentos estruturados sobre os mais diversos campos do conhecimento humano de uma maneira hierárquica e organizada.

As ontologias podem ser representadas em diversas linguagens, sendo a OWL uma das mais difundidas e utilizadas. A sua sigla têm origem no inglês *Web Ontology Language* e é um *standard* do *World Wide Web Consortium* (W3C) [9], e como o seu nome indica, foi desenvolvida para ser utilizada na Web Semântica. No entanto, pode ser utilizada facilmente noutros domínios para representar qualquer ontologia sobre um determinado domínio.

Os mais atentos podem questionar o porquê do acrónimo ser OWL e não o mais correcto e lógico WOL. Tal deve-se ao termo OWL servir de homenagem a William A. Martin [38], pioneiro na década de 70 em trabalhos sobre criação de linguagens de representação de conhecimento, cujo o projecto de maior destaque se designou por *One World Language*, (OWL).

O OWL possui três dialectos; o OWL Full, o OWL DL e o OWL Lite. A diferença entres eles está no grau de expressividade que permitem associar aos conceitos e relações do domínio em estudo, sendo o OWL Lite o de expressividade mais simples, seguido pelo OWL DL, e pelo OWL Full que é o mais complexo e expressivo.

Os três dialectos estão contidos uns nos outros, podendo ser vistos como extensões de expressividade do dialecto anterior. Isto significa que uma ontologia definida em OLW Lite é valida em OWL DL, e por sua vez uma ontologia definida em OWL DL também é válida em OWL Full. O inverso destas relações já não se verifica.

É frequente questionar-se, porquê não utilizar simplesmente XML para representar uma ontologia. Poder-se-ia definir uma ontologia, porque o OWL é definido sobre XML. Mas uma estrutura XML é mais limitada e iria apenas conter os dados numa determinada

forma sintáctica. A utilização de uma estrutura ontológica (OWL) permite, ter além dos dados organizados sintacticamente, uma relação semântica entre os objectos e atributos da estrutura, o que seria impossível com o XML.

A semelhança do conteúdo entre ficheiros OWL e XML deve-se ao facto do OWL derivar de outro standard reconhecido pela W3C, o RDF que por sua vez é um dialecto de XML. Assim, podemos ver o OWL como XML especializado em ontologias e com capacidades aumentadas para exprimir a semântica dos objectos ontológicos.

As ontologias são geralmente constituídas por quatro elementos básicos:

- Classe - Grupos ou colecções abstractas que tanto podem conter ou agrupar outras classes ou instâncias de classes.

- Instância de classe - São elementos concretos de uma determinada classe em que os atributos tomam valores concretos. Não são entidades abstractas mas objectos concretos e objectivos.

- Atributo - São características que descrevem propriedades das classes, e que podem tomar diferentes valores nas várias instâncias de uma determinada classe.

- Relações - Como o nome indica, são relações entre classes, instâncias de classes e atributos. As relações podem ter ou não restrições.

Neste projecto foi criada uma ontologia sobre o domínio dos hotéis e dos seus atributos envolventes. Para representar de uma forma esquemática o universo das características dos hotéis e para permitir a criação automática de instâncias das várias classes representadas na ontologia.

Durante o processo de normalização das descrições de hotéis é necessário criar objectos representativos da informação e obter dados sobre as relações destes objectos informativos. A ontologia serve para estruturar a informação obtida e ajudar a criação dos vários objectos à medida que a informação é extraída das descrições dos hotéis.

A ontologia utilizou o dialecto OWL DL, por ser suficientemente expressivo para representar os objectos e as suas relações dentro do domínio dos hotéis. O OWL Lite seria inadequado devido às suas restrições de cardinalidade, e o OWL Full sendo um super-conjunto do OWL DL, seria excessivo nas suas características, pois os extras em relação à versão OWL DL são desnecessários no domínio de estudo, com a desvantagem de criar ficheiros mais pesados na representação da ontologia.

2.3 Inteligência Artificial

Este projecto recorreu em grande parte a diversas técnicas de aprendizagem automática. No entanto o aprendizagem automática não é mais que um sub-domínio da inteligência artificial.

E afinal o que é a tão mediática inteligência artificial?

Em termos genéricos, é um ramo da computação que se dedica a criar algoritmos e métodos que permitam a dispositivos computacionais ter um "intelecto"semelhante aos seres humanos, quer na capacidade de captação automática de novos conhecimentos, análise de problemas, raciocínios dedutivos e indutivos, quer na resolução de problemas para os quais não tenham sido objectivamente programados para resolver.

Historicamente a inteligência artificial começou a ser difundida mediaticamente e a ser do conhecimento do público a partir dos anos 50 do século passado, em grande parte devido à crescente cultura de ficção científica, que dava uma ideia optimista que num espaço de 30 ou 40 anos, o homem seria capaz de criar dispositivos intelectualmente semelhantes, ou até superiores ao homem.

No entanto, tal evolução na área da inteligência artificial mostrou-se muito mais lenta e difícil do que se tinha julgado inicialmente. Mostrando que as formas com que o ser humano usa para questionar, observar, deduzir, induzir e concluir são mecanismos muito complexos e que apesar da evolução vertiginosa da tecnologia nas últimas décadas, o homem apenas deu alguns passos no caminho do domínio da inteligência artificial.

Mesmo sem ter a total capacidade de reproduzir o intelecto humano artificialmente por inteiro e no seu modo mais complexo, os conhecimentos actuais, já permitem efectuar bastantes protótipos e experiências bastante complexas e eficazes em sub-ramos da inteligência artificial.

Dos vários sub-ramos da inteligência artificial, existem dois ramos, que são especialmente importantes para o desenvolvimento deste projecto, o ramo do processamento de linguagem natural, e a aprendizagem automática.

2.3.1 Processamento de Linguagem Natural

O processamento de língua natural é uma área que reúne conceitos de inteligência artificial e linguística. Tem como principais objectivos permitir que os computadores possam "perceber"e processar directamente a linguagem utilizada pelos humanos, não apenas para que a comunicação homem/máquina seja mais natural e intuitiva, como também para que os computadores possam ter a capacidade de executar aplicações

capazes de trabalhar directamente sobre língua natural, com total percepção sintáctica, semântica, lexical e morfológica dos conteúdos processados.

Numa óptica de interface homem/máquina, a comunicação entre homem e máquina tem sido efectuada principalmente através de linguagens de programação, comandos de consola, menus e botões gráficos. Embora possam ser mais ou menos intuitivos conforme os casos e a experiência do utilizador, são sempre métodos castrativos. No entanto, ambiciona-se num futuro próximo que a comunicação homem/máquina possa ser o mais natural possível.

Numa óptica diferente, sem ter a ver com formas de interface homem/máquina, o processamento de língua natural também desperta elevado interesse, nomeadamente nas áreas de extracção, processamento e manipulação de textos em língua natural. Como exemplos dessas áreas temos aplicações para pesquisa, extracção e recuperação de informação em documentos, classificação de documentos, traduções de documentos, aplicações que produzam automaticamente resumos de textos ou simplificações, sistemas de pergunta e resposta automática, etc. É nesta óptica da manipulação e extracção de informação a partir de textos em linguagem natural que este projecto se encontra inserido.

2.3.2 Aprendizagem Automática

A aprendizagem automática em semelhança com o Processamento de Linguagem Natural, também é um sub-ramo da inteligência artificial que se dedica a criar algoritmos que permitam aos computadores aprenderem automaticamente de uma forma semelhante à humana. Embora tal objectivo de ter uma máquina com uma aprendizagem automática e cognitiva como a dos humanos ainda esteja longe de ser totalmente atingido. Hoje em dia existem algoritmos capazes de efectuar aprendizagens eficazes em domínios concretos nas mais diversas áreas.

Convém frisar que aprendizagem é muitas vezes confundida com o acto de memorizar. No entanto, são acções completamente diferentes, pois memorizar apenas se limita a apreender a informação fornecida e posteriormente a identificar. Tomemos um exemplo pratico:

- É inserido numa base de dados de um computador, um conjunto enorme de automóveis, com informação descritiva e a respectiva marca. Os atributos sobre os automóveis vão desde a cilindrada, potência, local de fabrico, peso, tipo de pneus, garantia oferecida, país de origem, tipo de motor, etc, etc.
- No passo seguinte a base de dados é inquirida com os dados de vários automóveis a fim de identificar a marca dos mesmos. Conseguir-se-á identificar a marca

dos automóveis apresentados, desde que estes fizessem parte da base de dados. Qualquer automóvel novo que não fizesse parte da base de dados seria descartado e impossível de descobrir a sua marca, mesmo que as características deste automóvel fossem muito idênticas a outros automóveis presentes na base de dados.

É aqui, que reside a grande diferença da memorização para a aprendizagem, pois a aprendizagem pretende "generalizar"as características de uma determinada classe, permitindo criar regras ou padrões que identifiquem universalmente essa classe ao contrário de memorizar extensivamente todas as suas instâncias conhecidas.

Com técnicas de aprendizagem automática e partindo dessa base de dados com informação dos automóveis, poder-se-ia criar um modelo de aprendizagem com regras ou abstracções que permitisse classificar correctamente um novo automóvel que não fizesse parte da base de dados. A base de dados deixaria de ser o nosso "conhecimento"sobre automóveis, e passaria a ser a nossa base de aprendizagem que nos permitiria identificar e classificar novos carros que surgissem. A esta tarefa de etiquetar objectos com classes semânticas a partir do conjunto de atributos descritivos, designa-se por classificação.

Muitas técnicas de classificação automática fazem uma aproximação ao raciocínio indutivo humano, que a partir de uma quantidade finita de dados, cria uma relação para um domínio infinito. Raciocinar indutivamente é partir de premissas particulares, em busca de uma lei geral, universal.

Um exemplo muito básico de um raciocino indutivo é:

- O ferro conduz electricidade
- O ferro é metal
- O ouro conduz electricidade
- O ouro é metal
- O cobre conduz electricidade
- O cobre é metal
- **Logo os metais conduzem electricidade**

Estas técnicas são geralmente efectuadas em duas fases: a fase de treino ou de aprendizagem e a fase de classificação ou de previsão.

A fase de aprendizagem, descrita de uma forma genérica e de alto nível, é a fase em que, partindo de um conjunto finito de objectos de um determinado domínio, é feita a análise e extracção de atributos que os fazem diferenciar de outros objectos pertencentes a outras classes semânticas.

Cada objecto é descrito através das suas características, e classificado quanto à sua classe. Com esta informação, atributos e classes, o algoritmo de aprendizagem cria

um modelo que vai permitir abstrair e criar regras universais de classificação. É o designado modelo de aprendizagem. Esse modelo vai ter regras que traçam a relação entre as características dos objectos e a sua classe. Na Figura 2.2 pode-se observar uma abstracção esquemática da distribuição dos objectos pertencentes a duas classes.

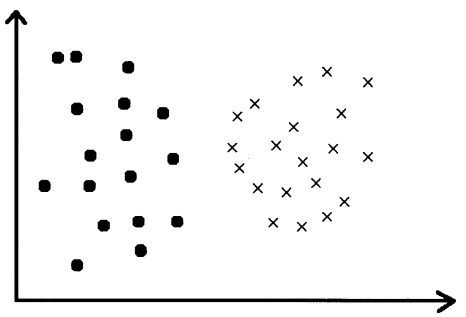

Figura 2.2: Amostra da ocorrências de duas classe dos objectos em estudo.

Levanta-se a questão de qual a melhor regra que permite abstrair o comportamento dessas classes para generalizar a separação das classes através do treino sobre um conjunto finito. Podem ser traçadas infinitas regras para caracterizar as classes. Na Figura 2.3 essas regras tomam a forma de rectas, e exemplificam a diversidade que podem tomar. A parte mais complicada da aprendizagem é escolher qual a melhor regra entre os objectos das classes que melhor as discrimina.

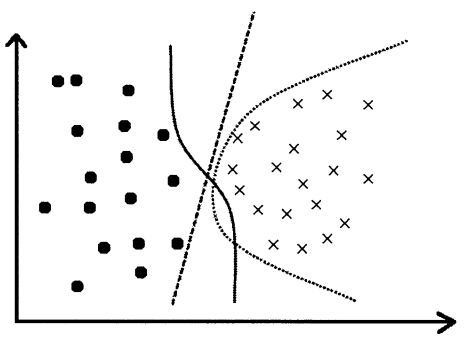

Figura 2.3: Possíveis regras de separação das duas classes.

Numa analogia ao raciocínio cognitivo humano, o modelo de aprendizagem seria a nossa capacidade de conseguir classificar ou agir a situações novas, a partir de experiências ou observações passadas. O esquema conceptual da fase de aprendizagem é descrito na Figura 2.4.

Após a etapa de classificação em que é criado o modelo de aprendizagem, ocorre então a fase de classificação ou previsão. Na fase de Classificação começa-se por recolher as características que descrevem o novo objecto que se pretende classificar. Depois esses dados são utilizados pelo modelo de aprendizagem previamente criado, e como

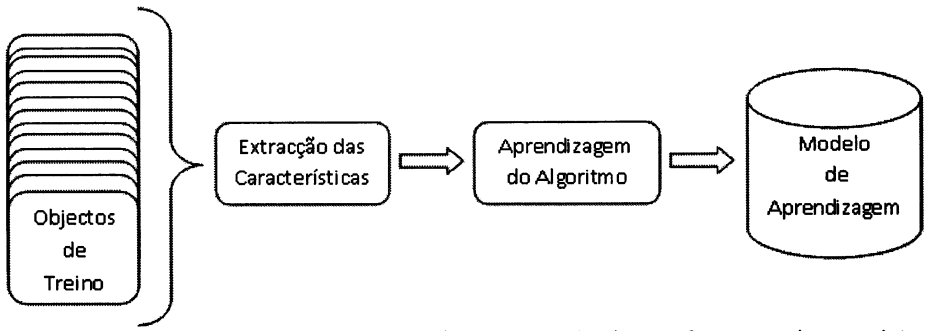

Figura 2.4: Aprendizagem de um Algoritmo de Aprendizagem Automática

resultado, obtém-se indutivamente a classe a que esse novo objecto pertence. Tal procedimento pode ser observado conceptualmente na Figura 2.5.

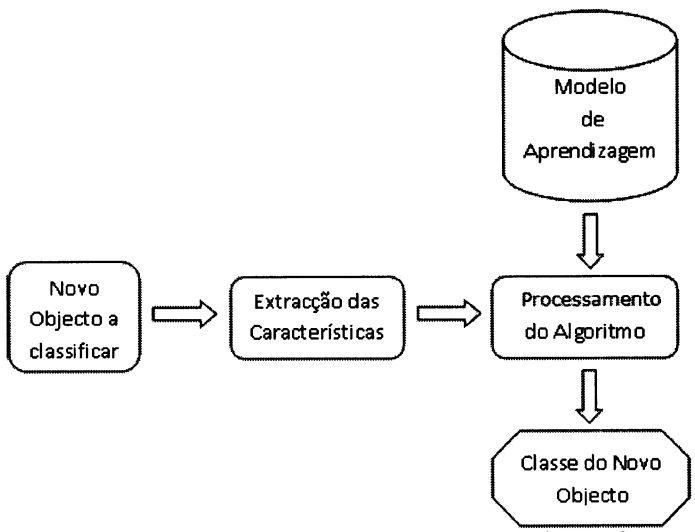

Figura 2.5: Classificação efectuada por um algoritmo de Aprendizagem Automática

Um requisito para o bom funcionamento de um algoritmo de aprendizagem automática é que este seja tolerável a dados de aprendizagem imperfeitos ou com ruído, pois nem sempre é possível que os dados que servem para aprendizagem estejam completamente bem classificados. E é o que acontece em maior parte dos casos práticos. Ou seja, nos objectos classificados para a aprendizagem pode haver uma taxa residual de erros nessa classificação. O erro dos dados tanto pode estar do lado da classificação da classe do objecto como pode estar na recolha de um determinado atributo dos objectos. Assim é fundamental que o algoritmo de aprendizagem automática tenha capacidade de tolerância ao ruído presente nos dados de aprendizagem.

Outro caso em que o algoritmo deve ser tolerante, é na ocorrência de valores aberrantes[7] nos dados, ou seja, à ocorrência de casos pontuais de dados aberrantes. Tomando um exemplo muito simples, tendo uma determinada família de objectos em estudo, podendo

[7]Do Inglês, *outliers*.

esses objectos tomar classificações referentes às classes do conjunto C. Por sua vez os objectos de estudo tem vários atributos. Assumimos que os objectos tomam a classe c_1 quando todos os atributos $\{a_1, a_2, ..., a_n\}$ toma valores entre $[x, y]$. E que tal ocorre em 99,8% dos casos em que o objecto em causa é classe c_1. Nos restantes 0,2% de casos, os objectos c_1 tomam à mesma valores de $[x, y]$ para os atributos $\{a_1, a_2, ..., a_{n-1}\}$ e que o atributo a_n toma o valores aberrante entre $]y; \infty[$. Nestes casos o algoritmo tem de perceber que o atributo a_n em certos casos foge à regra observada na maioria dos casos e assim desvalorizar ou anular o seu peso quando este toma valores fora do contexto habitual.

No entanto esta sensibilidade ao que é ou não aberrante, é algo muito minucioso, e que pode determinar o sucesso ou o fracasso do nosso algoritmo de aprendizagem automática, pois se for demasiado insensível a ruídos e valores aberrantes pode contaminar e degradar a qualidade do modelo de aprendizagem criado.

Por outro lado se for demasiado sensível aos casos aberrantes, pode descartar atributos importantes e relevantes para a classificação do objecto. E assim pensar que um atributo toma valores aberrantes e que na verdade não o são. Deste modo pode-se perder informação vital e consequentemente falhar a classificação dos objectos.

Há algoritmos de classificação automática que são mais propensos a excesso de sensibilidade a valores aberrantes, e outros mais propensos a serem insensíveis a esses casos. Ainda alguns algoritmos possuem parâmetros que permitem ajustar o grau de sensibilidade.

A classificação automática ou previsão de classes através de aprendizagem automática não é isenta de erros e falhas, em toda a Secção 2.6 são apresentadas algumas métricas que permitem fazer a avaliação das taxas de erro e de acerto que nos permitem avaliar qualitativamente o desempenho de um determinado algoritmo de aprendizagem automática.

A aprendizagem automática pode ser sub-dividida em dois tipos, supervisionada e não-supervisionada.

Aprendizagem Automática Não-Supervisionada. A aprendizagem não-supervisionada é uma aprendizagem que não tem qualquer tipo de entidade externa que ensine e faculte exemplos de aprendizagem ao algoritmo de aprendizagem automática. Isto pode parecer estranho à primeira vista, pois se o algoritmo de aprendizagem automática não tem qualquer conhecimento prévio ou exemplos de como agrupar os objectos correctamente como pode ele agrupar o que quer que seja correctamente?

É precisamente na resposta à pergunta anterior que reside a motivação de existir a aprendizagem automática não-supervisionada, pois a aprendizagem não-supervisionada

tem um objectivo bastante diferente da aprendizagem supervisionada. A aprendizagem supervisionada parte de um conjunto de objectos pré-classificados e induz para novos casos. A aprendizagem não-supervisionada, parte de início sem nenhuma "ideia"pré-adquirida e vai então agrupar os objectos em classes ditas abstractas, a partir das propriedades dos objectos.

O objectivo é permitir que quando se tem grandes quantidades de objectos complexos e aparentemente caóticos de serem classificados por humanos, tornando-se assim impossível de existir uma entidade com o papel de "professor", pois os humanos com o papel de "professor", não têm capacidade de analise e síntese das propriedades dos objectos devido à sua elevada complexidade.

Mas com a aprendizagem não-supervisionada é possível que os algoritmos de aprendizagem automática efectuem a criação de um conjunto de classes classificativas para uma família de objectos, que de outro modo seria impossível obter devido à complexidade dos objectos em estudo.

Aprendizagem Automática Supervisionada. Na aprendizagem supervisionada, os algoritmos de aprendizagem automática tem acesso a uma entidade externa que quase pode ser vista como um "Professor".

Essa entidade externa vai ser responsável por fornecer ao algoritmo bons exemplos para que ele sobre esses exemplos possa criar os seus modelos de aprendizagem e então criar regras indutivas para generalizar a partir do conjunto limitado fornecido pelo "professor".

O "professor"é que vai dispor do conhecimento da classificação dos objectos, e vai saber atribuir para determinado conjunto de atributos uma determinada classe classificadora do objecto. Assim o algoritmo de aprendizagem automática irá aprender baseado nos "conhecimentos"do "professor".

Mas esta aprendizagem "Aluno - Professor"tem sempre em vista o objectivo que o algoritmo não fique restrito a saber classificar apenas casos iguais aos apresentados pelo "professor", mas que tenha alguma inteligência indutiva para saber classificar eventuais novos casos que surjam diferentes dos apresentados pelo "professor".

Nos casos práticos, o papel de "professor"é efectuado por humanos que classificam previamente conjuntos de dados seguindo um conjunto de regras obtidas através da observação e do raciocino lógico humano, ficando assim esses algoritmos "viciados"pelos "professores".

Esta técnica é utilizada por exemplo em vários domínios de classificação automática, tais como classificação de imagens e de textos, em que são apresentados exemplos previamente classificados e rotulados para depois o algoritmo generalizar essa classificação

e automatizá-la a novos casos.

Este projecto utilizou esta técnica, pois os algoritmos utilizados tinham como base de partida para a sua aprendizagem um conjunto de frases classificadas manualmente por pessoas quanto ao tipo das frases. Pois um dos objectivos era que os algoritmos posteriormente conseguissem classificar automaticamente e com sucesso outras novas frases que fossem apresentadas.

2.3.3 Problemas da Aprendizagem Supervisionada

Os algoritmos de aprendizagem supervisionada nem sempre se mostram eficientes e com bons resultados. Durante a fase de aprendizagem sobre um determinado subconjunto de um domínio, podem não gerar boas regras indutivas para todo o domínio. Esses problemas podem ficar-se a dever a vários factores internos ao funcionamento do algoritmo de aprendizagem automática utilizado, sendo geralmente problemas de sobre ajustamento ou de sub ajustamento.

Sobre Ajustamento. O sobre ajustamento é uma falha que ocorre quando o algoritmo cria o seu modelo de aprendizagem a partir do seu conjunto de treino e gera um modelo restrito e muito adaptado exclusivamente a esse conjunto.

Quando testado um algoritmo com uma falha de sobre ajustamento com um conjunto de testes muito semelhante ou igual ao conjunto de aprendizagem, os resultados são excelentes, pois o algoritmo ficou demasiado focado nos exemplos de aprendizagem, tornando-se ignorante em relação a novos casos que surjam, mas muito bom para os que já conhece.

Quando o algoritmo procede à classificação de um conjunto de testes que é diferente do conjunto de aprendizagem , vão ocorrer maus resultados na classificação.

Quando ocorre este problema, o processo de aprendizagem falha, pois pode quase dizer-se que houve uma "memorização"e não uma aprendizagem, já que consegue-se classificar bem os mesmos elementos em que a aprendizagem foi baseada, mas quaisquer outros novos elementos diferentes que surjam, são mal classificados.

Este problema pode ser observado numa simples abstracção de funções em R^2, como se pode verificar na Figura 2.6:

Na imagem da esquerda, as cruzes representam o conjunto de treino, e a linha **a)**, uma aproximação perto do ideal da forma representada pelas cruzes. Novos objectos (cruzes) que surjam seguindo a distribuição dos seus semelhantes, irão ficar localizados perto dessa linha **a)**.

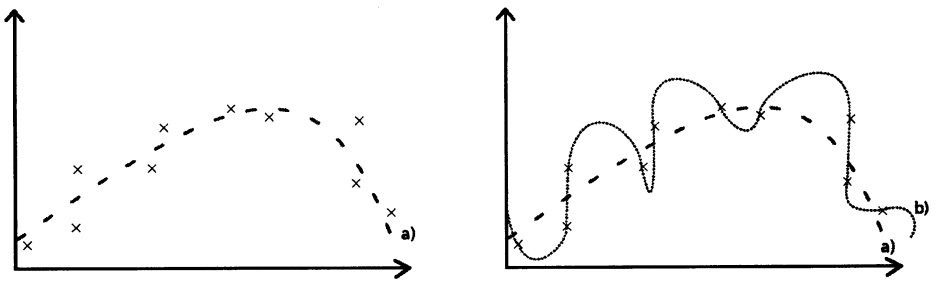

Figura 2.6: Exemplo de Sobre Ajustamento

Na Figura da direita observa-se a mesma distribuição de objectos e a mesma linha ideal **a)**, mas surge uma nova linha **b)** que sofre do problema de sobre ajustamento, pois essa linha é muito precisa em relação aos objectos utilizados durante a aprendizagem; no entanto tem pontos que se afastam bastante do comportamento do comportamento esperado. Caso surja um novo objecto que segue a distribuição dos seus semelhantes, podem ocorrer duas situações: ou o objecto é próximo de um dos objectos utilizados durante a aprendizagem e será bem classificado, ou então pode ficar longe dos objectos de aprendizagem e consequentemente muito longe da linha **b)** e assim ser mal classificado.

Através deste exemplo facilmente se conclui que um modelo de aprendizagem demasiado enquadrado com o seu conjunto de treino, fica com deficiências na sua capacidade indutiva, pois apenas classificará correctamente os casos idênticos ou muito semelhantes aos ocorridos durante a aprendizagem.

Sub Ajustamento. Este problema caracteriza-se pelo algoritmo de aprendizagem automática criar um modelo demasiado generalista ou demasiado simples. Isto implica que quando for testado, o modelo vá falhar bastante na classificação de novos casos. Mesmo que seja testado sobre o conjunto de aprendizagem, também terá uma taxa de erros bastante elevada. Neste ponto, este problema difere da falha de sobre ajustamento, pois esta apenas falha em novos casos, acertando quando testado sobre o conjunto de aprendizagem.

Para perceber graficamente este problema também podemos recorrer à abstracção de funções em R^2, como pode ser observado na Figura 2.7.

Na imagem da esquerda, as cruzes representam o conjunto de treino, e a linha **a)**, uma aproximação perto do ideal da forma representada pelas cruzes. Novos objectos (cruzes) que surjam seguindo a distribuição dos seus semelhantes, irão ficar localizados perto dessa linha **a)**.

Na Figura da direita, a linha **b)** representa um modelo que sofre do problema de sub-

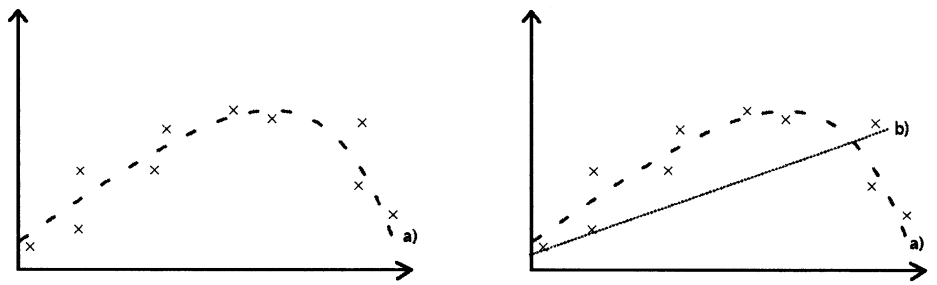

Figura 2.7: Exemplo de Sub Ajustamento

ajustamento, pois a linha é muito simplista e não consegue discriminar correctamente todos os objectos, e em certas situações afasta-se muito dos próprios objectos utilizados durante a aprendizagem.

Os novos objectos da classe que surjam para serem classificados, irão em grande parte ser mal classificados, quer sejam objectos próximos ou distantes dos objectos empregues durante a fase de aprendizagem.

Má amostragem. Na construção de um classificador pode surgir outro problema que é externo ao algoritmo de aprendizagem. Aliás, é um problema que pode contaminar uma experiência de aprendizagem automática independentemente do algoritmo escolhido, isto porque ocorre à priori à execução do algoritmo e chama-se má amostragem.

Este problema deve-se à qualidade da amostra de treino, pois se essa amostra não for representativa do conjunto que se quer avaliar, o algoritmo poderá falhar.

Tomemos um exemplo prático, temos um domínio de instâncias S, sendo essas instâncias agrupadas em várias classes, $C = \{C_1, ..., C_n\}$ sendo P_i a probabilidade de cada classe C_i, com P_i igual para todas as classes. Aleatoriamente é extraído de S um sub-conjunto S', para servir de conjunto de treino do algoritmo.

No entanto o nosso conjunto S', mesmo tendo sido extraído aleatoriamente de S, nada garante que seja igualmente representativo de todas as classes C de S. Este conjunto de amostra S', por exemplo, pode ter uma grande representatividade das classes $C' = \{C_1, ..., C_m\}$ com $1 < m < n$ e não ter qualquer representatividade das classes $C'' = \{C_{m+1}, ..., C_n\}$.

Nestes casos, o algoritmo teria facilidade em criar regras para todas as classes de C', mas teria dificuldades para criar regras para a classe do conjunto C''. Aliás se as classes de C'' tivessem sido completamente negligenciadas do conjunto de treino S', nem seriam criadas regras para esse conjunto de classes, sendo assim completamente excluídas do modelo de aprendizagem criado.

Quando o algoritmo fosse então classificar elementos do conjunto S tendo como base de partida o modelo de aprendizagem criado a partir de S' sempre que surgissem classes do conjunto C'', estas seriam mal classificadas como sendo classes de C'.

Para evitar este problema, é necessário que o conjunto S' escolhido para criar o modelo de aprendizagem seja equilibrado, estando representadas todas as classes $\{C_1, ..., C_n\}$, com o número de ocorrências de cada classe semelhante. Caso o número de ocorrências não for semelhante em cada classe, o menor valor de ocorrências de uma determinada classe deve ser suficiente para que o algoritmo de aprendizagem automática crie regras que consigam futuramente identificar elementos dessa classe no conjunto S.

2.4 Algoritmos de Classificação Automática

Os algoritmos de aprendizagem automática supervisionada agrupam-se em várias famílias, conforme os seus mecanismos base de funcionamento. Dentro de cada família, o princípio geral de funcionamento é semelhante, variando apenas alguns pontos ou afinações.

No desenvolvimento deste trabalho, foram usadas três famílias de algoritmos em maior profundidade: algoritmos de Árvores de Decisão, que como o nome indica baseados em árvores de decisão, algoritmos de Aprendizagem Bayesiana, baseados no teorema de Bayes e analise estatística, e os SVM (Máquina de Suporte de Vectores)[8]

De seguida iremos ver em maior profundidade o funcionamento e as principais diferenças de cada uma destas três famílias de aprendizagem automática.

2.4.1 Árvores de Decisão

Os algoritmos de aprendizagem automática baseados em árvores de decisão, são uma das família mais fáceis de perceber conceptualmente o seu funcionamento. Baseiam-se em simples árvores de decisão onde cada nó é uma condição e cada folha é um resultado final. A Figura 2.8 apresenta um exemplo para determinar se um dia é indicado ou não para jogar ténis.

O funcionamento da árvore é muito simples. Parte-se da raiz, que é o primeiro nó e onde se encontra a primeira condição, depois segue-se caminho conforme o nosso atributo cumpre essa condição. Cada ramo da árvore corresponde a um dos valores possíveis do atributo do nó de onde partem esses ramos. Segue-se sucessivamente para

[8]Do Inglês, *Support Vector Machine.*

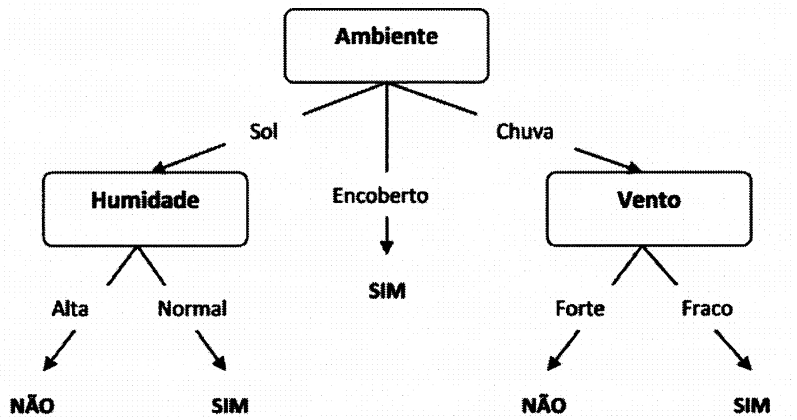

Figura 2.8: Árvore de decisão simples em que cada nó é uma condição e cada folha é um resultado final. Neste caso típico, para determinar se a classe "Ir Jogar Ténis"toma valores positivos ou negativos basta ir respondendo às condições e ir descendo a árvore até chegar a uma das folhas com o resultado final.

o nó seguinte até chegar às folhas da árvore. Cada folha tem a classificação final, podendo haver várias folhas com o mesmo resultado.

Desta descrição é possível concluir que uma árvore de decisão não passa de uma disjunção de conjunções lógicas sendo os ramos as conjunções e os nós as disjunções.

A árvore de decisão é também facilmente traduzida em regras *IF-THEN-ELSE*[9]. A árvore da Figura 2.8 é traduzida nas seguintes regras da Tabela 2.1.

- `IF Ambiente = Sol AND Humidade = Alta THEN Jogar = Não`
- `IF Ambiente = Sol AND Humidade = Normal THEN Jogar = Sim`
- `IF Ambiente = Chuva AND Vento = Forte THEN Jogar = Não`
- `IF Ambiente = Chuva AND Vento = Fraco THEN Jogar = Sim`
- `IF Ambiente = Encoberto THEN Jogar = Sim`

Tabela 2.1: Tradução da árvore em regras *IF-THEN-ELSE*

Como se pode verificar, o funcionamento dos algoritmos de aprendizagem baseados em árvores de decisão é bastante simples. No entanto, a construção da árvore em si é um processo mais complicado. E é aí que geralmente residem as diferenças entre os vários algoritmos concretos.

Entropia e Ganho de Informação. O segredo para construir uma árvore de decisão eficiente está na maneira como a árvore é feita.

[9]Em Portugês, SE-ENTÃO-SENÃO.

Uma das principais características utilizadas para construir a árvore é saber obter a decisão em cada nó que permita ter uma entropia miníma, o que é equivalente a dizer, obter a decisão em cada nó com o maior ganho de informação possível. Isto com o objectivo de tornar a árvore o mais pequena possível e consequentemente com menos testes condicionais para se chegar aos resultados finais.

Já vimos que uma árvore de decisão é um conjunto de nós que são disjunções das possibilidades que o atributo visado nesse nó pode tomar. Levanta-se então a questão de que atributos (e suas respectivas disjunções) devam estar mais perto da raíz da árvore, e que atributos devam estar mais perto das folhas.

A fórmula genérica para calcular a entropia de um conjunto de classes

$$Cs = \{C_1, C_2, ..., C_n\}$$

do nosso domínio S tendo cada classe a probabilidade P_i de ocorrer, e sendo $A = \{a_1, a_2, ..., a_j\}$ o conjunto de atributos que permite distinguir as classes de Cs é:

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

Para determinar qual o primeiro atributo a ser testado e a ser colocado na raiz da árvore, calcula-se qual é o atributo que vai fornecer um maior ganho de informação que se resume a ser a maior diferença entre a entropia existente no conjunto antes e depois do teste. Este ganho pode ser calculado pela fórmula:

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

Sendo V o domínio dos valores possíveis do atributo a_i, e $|S|$ a cardinalidade do nosso conjunto geral, e $|S_v|$ a cardinalidade do sub conjunto de $|S|$ com elementos que são diferenciados dos restantes através do atributo a_i.

Algoritmo ID3. O algoritmo de aprendizagem automática ID3 [41] foi inventado por Ross Quinlan e é considerado um marco e um ponto de partida nos algoritmos de árvores de decisão, pois é um dos mais simples e fáceis de compreender. No entanto o seu uso neste trabalho foi limitado a título de experiência, pois rapidamente o seu uso foi substituindo pelo seu irmão mais eficaz C4.5 [43].

O algoritmo ID3 tem um modo de criação da árvore de decisão muito simples, baseado no cálculo da entropia e do ganho de informação vistos no tópico anterior. Simples-

mente ele calcula o ganho de informação para todas as disjunções de atributos sobre o nosso conjunto. O atributo/disjunção que apresentar maior ganho de informação será imediatamente colocado na raiz da árvore. Depois disto todo o processo é repetido iterativamente para cada sub-ramo da árvore, até esgotar os atributos diferenciadores dos nossos elementos do conjunto em estudo.

De seguida podemos ver a sua simplicidade no pseudo-código do algoritmo iterativo:

- Cria o nó (raiz no primeiro caso)
- Se $|Cs| = 1$, então o nó é uma folha da única classe de Cs
- Senão, com $|Cs| > 1$, escolher o atributo A separador dos elementos de $|Cs|$ com menor entropia
- Criar recursivamente uma sub-árvore para cada valor possível do atributo A escolhido.

Tabela 2.2: Pseudo-código do funcionamento do algoritmo ID3

No entanto, o algoritmo ID3, devido a seguir a regra da escolha dos nós sempre em função da menor entropia possível, resulta num algoritmo com tendências para sobre-ajustamento. Assim obtém bons resultados a classificar o conjunto de treino usado para a sua aprendizagem, mas os resultados são fracos quando testado sobre um novo conjunto de dados da mesma família dos utilizados durante a aprendizagem.

Para superar este problema e permitir que o algoritmo consiga mais facilmente identificar e classificar correctamente novos casos foram implementadas varias melhorias, culminando no algoritmo de aprendizagem automática C4.5.

Algoritmo C4.5. Este algoritmo é uma versão melhorada do ID3, que conta com nova abordagem e regras na construção da árvore, para que ela não seja sobre-ajustada aos casos de treino.

Este algoritmo também foi desenvolvido pelo mesmo autor do ID3, Ross Quinlan. Tanto o algoritmo ID3 como o C4.5 [43] são algoritmos livres, no entanto existe uma versão comercial do C4.5 com alguns melhoramentos chamada de C5.0 [45].

Ao contrário do ID3 que só foi usado a título experimental durante a execução deste trabalho, o C4.5 foi um dos "algoritmos fortes"deste trabalho. Não foi utilizado o algoritmo C4.5 "original", mas sim uma reimplementação em JAVA distribuída na plataforma WEKA [17].

Todo o processo de construção da árvore de decisão do C4.5 é igual ao do algoritmo ID3. A principal diferença e melhoria é que o C4.5 após efectuar a construção da

árvore de decisão, efectua a chamada poda da árvore, com o objectivo de cortar da árvore os ramos demasiado longos e demasiado específicos e que são responsáveis por sobre-ajustar a árvore ao conjunto de aprendizagem.

Esta técnica é chamada de pós-poda, pois ocorre após a árvore estar toda criada. Existem também outros algoritmos da família das árvores de decisão que usam outra técnica apelidada de pré-poda, que consiste em restringir o crescimento da árvore logo durante a sua criação [42].

A pós-poda do C4.5 tem como objectivo reduzir a complexidade da árvore, que implica eliminar algumas das suas sub-árvores, reduzindo assim a altura da árvore e aproximar as folhas à raiz.

Para ser efectuada uma determinada poda é efectuada uma avaliação estatística. Para cada nó são avaliados os erros de classificação que resultam desse nó e dos seus nós descendentes; só é efectuada a poda do nó se esta não implicar uma redução no desempenho da árvore. Neste aspecto o C4.5 é um pouco conservador, pois esta avaliação é pessimista, de modo a que não se corra o risco de reduzir a eficácia da árvore. Existem outros algoritmos que "ariscam"mais e efectuam uma poda mais drástica da árvore.

Outra característica e melhoria do C4.5 em relação ao ID3, é que este permite trabalhar com atributos contínuos ou discretos, enquanto o ID3 apenas permite atributos discretos. Para trabalhar com esses valores contínuos o C4.5 estima um parâmetro de decisão, e consoante o valor da variável contínua for superior ou inferior a esse valor assim é convertida em valores discretos.

O C4.5, ao contrario do ID3, permite ainda usar atributos desconhecidos durante a criação da árvore, e que os atributos tenham diferentes pesos entre si.

A implementação em JAVA do C4.5 chamada de J48 [17] também permite ao contrário do ID3 e C4.5, efectuar validação cruzada de modo a poder obter métricas de avaliação sobre o desempenho.

2.4.2 Aprendizagem Probabilística ou Bayesiana

Aprendizagem Probabilística ou Bayesiana é outra grande família de algoritmos de aprendizagem automática. Como na família anteriormente visada, em que todos os algoritmos tinham em comum uma árvore de decisão na sua base, esta família dos algoritmos probabilísticos ou Bayesianos [23] também tem algo comum na sua base de funcionamento: cálculos probabilísticos que têm como base o teorema de Bayes.

O teorema de Bayes pode-se representar pela seguinte fórmula:

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

Sendo:

- $P(A)$ a probabilidade à priori ou também designada probabilidade marginal do acontecimento A. E designa a probabilidade do acontecimento A sem se saber qualquer informação sobre o acontecimento B.

- $P(A|B)$ a probabilidade condicional de A dado B, também designada de probabilidade posteriori de A. Esta probabilidade depende do valor de B e pode ser definida como
$$\frac{P(A \cap B)}{P(B)}$$

- $P(B|A)$ a probabilidade condicional de B dado A, ou probabilidade posteriori de B condicional a A. Pode ser definida como
$$\frac{P(A \cap B)}{P(A)}$$

- $P(B)$ a probabilidade à priori de B.

O teorema de Bayes também pode ser representado de uma forma alternativa, preferida até por alguns autores:

$$P(A|B)P(B) = P(A \cap B) = P(B|A)P(A)$$

Para derivar a fórmula original a partir desta equação, basta descartar a fórmula central e fazer a dedução inversa, partimos de:

$$P(A|B) \text{ e } P(B|A)$$

logo

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

então

$$P(A|B)P(B) = P(A \cap B) \text{ e } P(B|A)P(A) = P(A \cap B)$$

juntando os membros por $P(A \cap B)$ obtemos então novamente:

$$P(A|B)P(B) = P(A \cap B) = P(B|A)P(A)$$

No entanto surge a dúvida de como se pode utilizar o Teorema de Bayes das teorias probabilísticas para a criação de modelos indutivos. A resposta surge com a chamada Inferência Bayesiana. Assumimos que o nosso teste de aprendizagem automática tem como base para o seu modelo de aprendizagem um conjunto de objectos em estudo com C o conjunto de classes possíveis para esses objectos e A o conjunto de atributos que qualificam esses objectos. E temos a hipótese h, que relaciona determinado conjunto de atributos com a classe dos objectos. Por sua vez temos o domínio H de todas as hipóteses possíveis.

Assim facilmente adaptamos a fórmula do Teorema de Bayes aos nossos objectivos, obtendo:

$$P(h_i|C) = \frac{P(C|h_i)P(h_i)}{P(C)}$$

Tendo em conta as várias hipótese h_i de H, pretendemos obter a h_i com maior valor de probabilidade à posteriori $P(h_i|D)$ e assim obteremos a melhor hipótese para a relação entre determinados atributos do objecto com a sua eventual classificação.

Podemos definir essa hipótese h_i de máxima probabilidade à posteriori em H, por h_{MAP}[10]. E a sua obtenção resume-se a:

$$h_{MAP} = \arg\max_{h \in H} P(h|C)$$

Utilizando o teorema de Bayes:

$$h_{MAP} = \arg\max_{h \in H} \frac{P(C|h)P(h)}{P(C)})$$

Por sua vez podemos simplificar, pois na maioria dos casos as várias h_i de H têm todas a mesma probabilidade, tornando na fórmula anterior $P(h)$ numa constante que pode ser removida. Por sua vez, a probabilidade $P(C)$ é sempre uma constante e também pode ser removida. Obtém-se assim a fórmula simplificada:

$$h_{ML} = \arg\max_{h \in H} P(C|h)$$

[10]Sendo "MAP"a sigla do Inglês *"Maximum A Posteriori"*.

Esta hipótese é designada por máxima verosimilhança[11].

Assim é possível usar o Teorema de Bayes como classificador num algoritmo de aprendizagem automática em que se tenta obter sempre a melhor hipótese que relaciona os atributos dos objectos à classe com a máxima de verosimilhança.

Naïve Bayes. Naïve Bayes [23] é um dos algoritmos de aprendizagem automática mais conhecido e utilizado que tem como base para o seu funcionamento um classificador probabilístico baseado no teorema de Bayes. A designação "Naïve"provém do algoritmo pressupor que os vários atributos que descrevem os objectos são independentes, o que na realidade raramente acontece. Assim, entre os vários atributos que discriminam a classe do objecto, cada atributo contribui independentemente para a probabilidade do objecto fazer parte de uma classe ou outra, não havendo qualquer correlação entre os diversos atributos na hora de decidir a classe do objecto.

No entanto o facto do algoritmo fazer essa simplificação não implica que ele obtenha maus resultados. Pelo contrário, o algoritmo Naïve Bayes é um algoritmo que na maior parte dos domínios apresenta bons resultados.

O modelo probabilístico do Naïve Bayes é facilmente extraído do Teorema de Bayes. Assumindo que temos um conjunto de objectos em estudo sendo C o conjunto de classes possíveis para esses objectos e A o conjunto de atributos que caracterizam esses objectos, então a probabilidade condicionada de uma classe C_i ocorrer após todos os atributos sendo estes independentes uns dos outros, pode ser representada por:

$$P(Ci|A_1, ..., A_n)$$

Adaptando o Teorema de Bayes à fórmula obtemos:

$$P(C_i|A_1, ..., A_n) = \frac{P(A_1, ..., A_n|C_i)P(C_i)}{P(A_1, ..., A_n)}$$

Como já vimos anteriormente, o denominador não depende da probabilidade da classe C_i e os valores $\{A_1, ..., A_n\}$ são valores conhecidos, assim todo o denominador é uma constante e pode ser descartado, simplificando obtemos:

$$P(A_1, ..., A_n|C_i)P(C_i)$$

Pelo Teorema da Probabilidade Conjunta têm-se que:

[11] Do Inglês, *Maximum Likelihood.*

$$P(C_i, A_1, ..., A_n|C_i)$$

que sendo os atributos $\{A_1, ..., A_n\}$ independentes, é equivalente a ter

$$P(C_i)P(A_1|C_i)P(A_2|C_i)P(A_3|C_i)...P(A_n|C_i)$$

Terminando, obtemos que a probabilidade de um determinado objecto pertencer a uma determinada classe considerando os atributos independentes é:

$$P(C_i|A_1, ..., A_n) = P(C_i)\prod_{j=1}^{n} P(A_j|C_i)$$

2.4.3 Máquinas de Vectores de Suporte (SVM's)

As Máquinas de Vectores de Suporte, mais conhecidas por SVM's[12], são uma família de algoritmos de aprendizagem automática desenvolvida inicialmente pelos trabalhos de Vapnik e Chervonenkis [52].

De uma maneira muito simplista, temos os objectos da nossa colecção que pretendemos classificar. Esses objectos podem ser classificados de modo binário, tomando as classes C e C', e são caracterizados pelos atributos do conjunto $A = \{A_1, A_2, ..., A_n\}$.

Cada objecto pode ser representado na estrutura de SVM's como sendo um vector n-dimensional num espaço vectorial de dimensão n obtendo uma determinada disposição geográfica consoante os valores dos seus atributos.

O classificador dos SVM's surge como um algoritmo que vai obter e optimizar um hiperplano de dimensão $n - 1$ dentro do nosso espaço n-dimensional, que separa as duas classes. Esse hiperplano pode ser visto como uma fronteira, mas ao invés de ser uma fronteira bidimensional como a dos mapas, é uma fronteira de dimensão $n - 1$.

Quando qualquer novo objecto for adicionado à colecção e se pretender a sua classificação referente à classe C ou C', basta representar esse objecto no espaço vectorial n-dimensional, e ver se a sua representação ocorre de um lado ou de outro da "fronteira"que separa as duas classes.

No entanto, no espaço vectorial pode existir uma infinidade de hiperplanos capazes de dividir as duas classes de objectos, levantando a questão de qual hiperplano se adequa melhor. Sendo o algoritmo SVM's responsável por essa decisão.

[12]Do Inglês, *Support Vector Machines.*

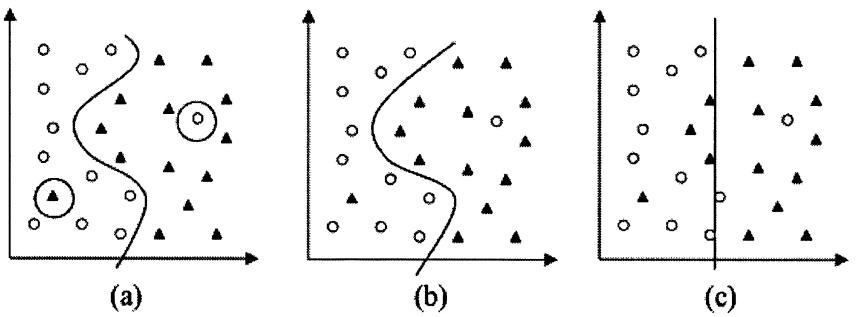

Figura 2.9: Exemplos de classificação Binária. Desde o modelo excessivamente complexo, permissível a *outliers* desviantes e a provável má classificação para novos casos (a), modelo equilibrado e já insensível a *outliers* desviantes (b), e modelo demasiado simplista e com elevado número de classificações erradas (c).

Essa decisão é baseada numa optimização matemática, que por norma tenta obter o hiperplano que consegue maximizar a separação entre as classes, de modo a que a distância média do hiperplano aos elementos mais próximos de C e C' seja a maior possível.

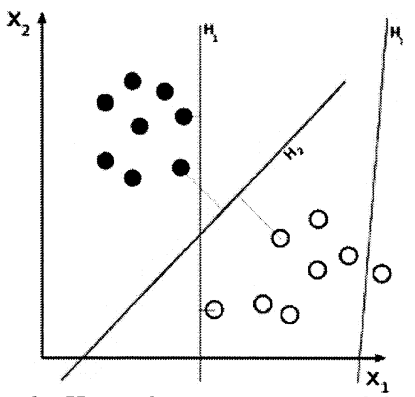

Figura 2.10: Exemplos de Hiperplanos numa simplificação em duas dimensões para fácil compreensão. Temos duas classes, sendo o H_3 um hiperplano falhado, pois divide mal as classes. Os hiperplanos H_1 e H_2 separam bem as duas classes, no entanto o H_2 é o melhor, pois maximiza a margem média aos elementos mais próximos das duas classes.

O hiperplano geralmente é representado por

$$w \cdot x - b = 0$$

Onde w é o vector normal perpendicular ao hiperplano e as margens equidistantes que maximizam a distância aos elementos das classes são representadas por $w \cdot x - b = 1$ e $w \cdot x - b = -1$ para a margem superior e inferior respectivamente. A distância entre as margens é dada por $\frac{2}{||w||}$ e a distância do hiperplano à origem é $\frac{b}{||w||}$

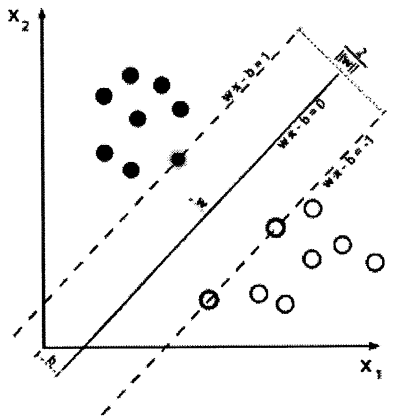

Figura 2.11: Hiperplano e suas margens

Estes são os SVM's lineares (por utilizarem um hiperplano). São os mais simples e fáceis de perceber, no entanto, poucos são os casos reais que possam ser correctamente modulados com um hiperplano linear de margens rígidas. Geralmente utilizam-se SVM's lineares com margens "suaves", ou seja, ao contrário dos de margem rígida que não permitem a existência de que qualquer objecto do conjunto entre o espaço delineado pelas margens, os de margens suaves permitam que possam haver excepções para adaptar o modelo a casos aberrantes e ruídos existentes nos dados para aumentar a flexibilidade do modelo a dados reais.

A tolerância à entrada de objectos dentro das margens é definida por parâmetros que permitem definir a quantidade de objectos e a profundidade a que se podem encontrar dentro das margens.

Para além de se definir a tolerância permitida para objectos dentro das margens, é usual definir também um valor de expulsão, ou seja, se um objecto desviante entrar dentro das margens até certo limite é tolerado, no entanto se passar esse valor limite de tolerância dentro das margens esse objecto simplesmente será ignorado e considerado um valor aberrante. Pode-se ver um exemplo na Figura 2.12:

No entanto nem sempre os SVM's lineares e os SVM's suaves são suficientes para se adaptarem a todos os casos de estudo com sucesso, pois em determinados casos os dados assumem tal complexidade que são impossíveis de discriminar correctamente com um hiperplano linear ou linear suave.

Para estes problemas existem os SVM's não lineares, que podem tomar diversas formas e complexidades. Esses diversos SVM's não lineares são criados por diferentes funções de núcleo[13]. Sendo esta função responsável pelo calculo e optimização da hiper-superfície separadora. Como exemplo, núcleos não lineares mais difundidos são os polinomiais, radiais e hiperbólicos.

[13]Do Inglês, *Kernel function.*

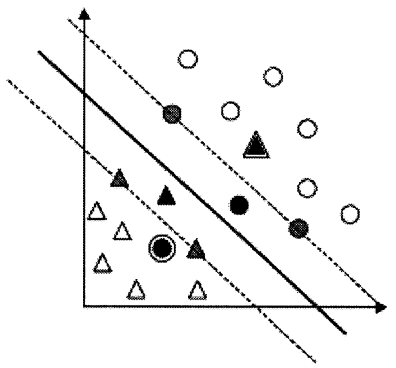

Figura 2.12: SVM's Suaves, onde os objectos cinzentos são tolerados devido a encontrarem-se nos valores de tolerância, e os objectos pretos são objectos descartados devido a encontrarem-se muito para lá dos valores de tolerância.

Embora possam ter objectivos diferentes, os princípios de funcionamento, dos diferentes núcleos são idênticos e simples de perceber conceptualmente.

Relembrando, temos os nossos objectos de estudo com n características ou atributos, sendo cada atributo representado numa dimensão diferente, logo para representarmos o nosso objecto de n atributos, necessitamos de um espaço vectorial de n dimensões chamado X, e a superfície separadora de classes terá $n-1$ dimensões. Como esse hiperplano toma uma forma linear, caso seja impossível separar as classes de modo linear, os *kerneis* dos SVM's não lineares promovem o nosso espaço vectorial original X de dimensão n para um novo espaço vectorial Y de dimensão n' com $n' > n$, e promovem também o hiperplano de dimensão $n-1$ para um hiperplano de dimensão $n'-1$, sendo obviamente $n'-1 > n-1$.

O objectivo de tal incremento de dimensão prende-se com o facto de algo que não é descritível linearmente numa dimensão, passa a poder ser discriminado linearmente numa dimensão superior.

No entanto os objectos ao serem promovidos a uma dimensão superior são mapeados por uma função Φ que os aloca no espaço de dimensão superior de um modo específico que permita que sejam classificados linearmente. Sendo Φ definida por:

$$\Phi = X \rightarrow Y : X \in \Re^n, Y \in \Re^{n'}, n < n'$$

Assim a função Φ tem como conjunto de partida o domínio do espaço vectorial de dimensão n, e como conjunto de chegada o domínio definido pelo espaço vectorial de dimensão n'.

A escolha da função mapeadora Φ é de extrema importância, pois nem todas as funções

incrementadoras de dimensão de n para n' vão distribuir os objectos no conjunto de chegada de um modo que sejam separáveis de modo linear. Por isso a escolha de Φ é fundamental para o sucesso da promoção de dimensão.

Tal procedimento é baseado no teorema de Cover [2] que diz que um conjunto de dados não linear no espaço X pode ser transformado num espaço Y com alta probabilidade desses dados passarem a linearmente separáveis. Para isso a transformação efectuada por Φ não pode ser linear e que Y tem de poder tomar qualquer dimensão, mesmo que extremamente elevada em relação a dimensão de X.

Para compreender tal conceito podemos ilustrar um simples exemplo com uma função Φ transformadora de $\Re^2 \rightarrow \Re^3$ em que se transforma algo não linear em $\Re^2$ mas que passa a ser linear em $\Re^3$.

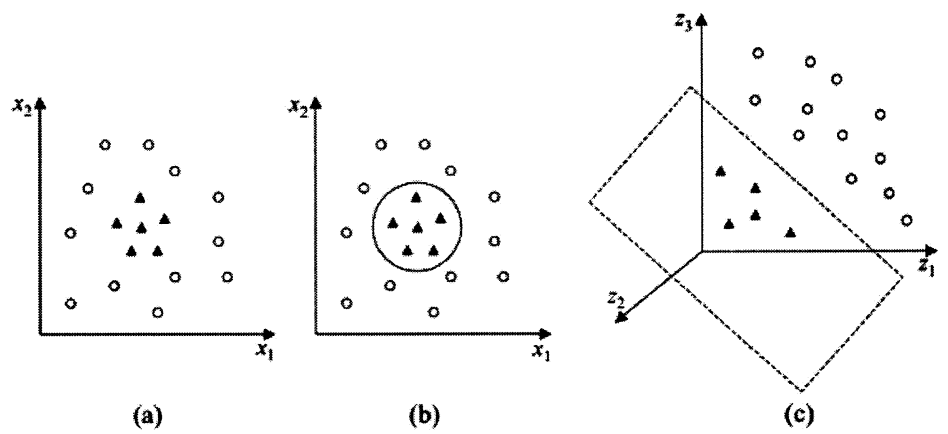

Figura 2.13: Linearização por aumento de dimensão de $\Re^2 \rightarrow \Re^3$, sendo (a) o cojunto de dados não linear em $\Re^2$, (b) a fronteira não linear que separa as classes dos dados, (c) os mesmos dados já lineares em $\Re^3$

Temos como base a Figura 2.13 em que (a) passa a (c) através da função de transformação:

$$\Phi(X) = \Phi(x_1, x_2) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$

e o plano de separação passa a ser descrito por:

$$f(X) = w \cdot \Phi(X) + b = w_1x_1^2 + w_2\sqrt{2}x_1x_2 + w_3x_2^2 + b = 0$$

Os dados são mapeados de uma dimensão inferior para uma superior, e nessa dimensão superior já se poderia utilizar um SVM linear ou SVM linear suave para separar as classes.

No entanto por vezes a dimensão do conjunto de chegada Y pode ser tão elevada que o cálculo e a determinação da função transformadora Φ ideal pode ser algo bastante complexo ou até inviável.

Assim procede-se a uma simplificação que parte de transformar dois pontos em conjunto $\Phi(X_i)\cdot\Phi(X_j)$ em vez de apenas um $\Phi(X)$.

Assim são obtidos os conhecidos Kernels dos SVM's.

Os Kernels são geralmente representados por K e não são mais que uma função que recebe dois pontos X_i e X_j do conjunto de partida e calcula o produto escalar desses pontos no conjunto de chegada de dimensão superior e que se representa por:

$$K(X_i, X_j) = \Phi(X_i)\cdot\Phi(X_j)$$

Voltando ao nosso exemplo de $\Re^2 \rightarrow \Re^3$ da Figura 2.13, e tendo dois pontos $X_i = (x_{1i}, x_{2i})$ e $X_j = (x_{1j}, x_{2j})$ do conjunto de partida $\Re^2$ obtemos como resultado:

$$K(X_i, X_j) = \left(x_{1i}^2, \sqrt{2}x_{1i}x_{2i}, x_{2i}^2\right)\cdot\left(x_{1j}^2, \sqrt{2}x_{1j}x_{2j}, x_{2j}^2\right) = (X_i\cdot X_j)^2$$

Este resultado mostra que as funções de núcleo são muito poderosas, pois simplificam o processo de transformação dos dados entre espaços vectoriais, já que permite fazer esta transformação sem ter de se conhecer concretamente a função Φ. Esta é gerada implicitamente durante a utilização de determinada função de núcleo.

O SVM's não lineares mais conhecidos geralmente são o tipo Polinomial homogéneo:

$$k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i}\cdot\mathbf{x_j})^d$$

Em que d com $d > 1$ é o grau do polinómio, e nos casos que $d = 1$ obtemos um polinómio de grau 1, que nada mais é que um Kernel de um SVM linear.

Também existe a variante de Kernel Polinomial não homogéneo com a seguinte forma:

$$k(\mathbf{x_i}, \mathbf{x_j}) = ((\mathbf{x_i}\cdot\mathbf{x_j}) + k)^d$$

com k a tomar valores $k \neq 0$, pois caso tomasse o valor $k = 0$ obteríamos um Kernel Polinomial homogéneo.

Existe ainda Kernels mais complexos, como é o caso do Kernel Radial:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

com γ a tomar valores $\gamma > 0$, e a variante do Kernel Radial Gaussiana com a seguinte fórmula:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma^2}\right)$$

$$\text{Com } \sigma \neq 0$$

Por fim apresenta-se o Kernel com a função de Tangente Hiperbólica com a seguinte formula:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\kappa \mathbf{x_i} \cdot \mathbf{x_j} + c)$$

$$\text{com } \kappa > 0 \text{ e } c < 0$$

No decorrer deste trabalho apenas se utilizou o Kernel linear. Pois o trabalho incidiu sobre dados textuais e tomando base no artigo [21] usar Kernels de complexidade superior à linear não é justificável em problemas de classificação de texto. O algoritmo de SVM's utilizado na elaboração do trabalho é a versão disponibilizada na ferramenta WEKA, que é uma versão de SVM's com a optimização sequencial mínima [24] geralmente referida por SMO[14]

2.5 Bases Representativas

A maior parte dos algoritmos de classificação de texto baseados em aprendizagem supervisionada trabalha sobre um saco de palavras que é uma listagem de palavras que ocorrem no domínio em estudo. Sobre essas palavras são efectuados estudos de contagem e frequências de modo a extrair daí regras de classificação. Assim uma frase inicialmente constituída por palavras passa a ser representada por um vector de atributos que discriminam essa frase. Como os algoritmos de classificação não lidam directamente com as frases de palavras, vão ter como entrada os vectores de atributos.

A forma de representar essas contagens e frequência pode seguir diversas abordagens distintas. A essas abordagens chama-se "Bases Representativas".

As bases representativas podem ser obtidas a partir dos seguintes parâmetros:

[14]Do Inglês, *Sequential Minimal Optimization.*

f_{ij} – número de ocorrências da palavra i no documento j

fd_j – o número total de palavras no documento j

ft_i – o número de documentos em que o termo i aparece pelo menos uma vez

v_{ij} – a "importância"de um determinado termo i no documento j.

Base Binária É a base representativa mais simples onde o valor do atributo toma o valore de 1 ou 0 consoante a palavra ocorra ou não, respectivamente. Uma palavra que ocorra apenas uma vez ou ocorra um número elevado de vezes, tem o mesmo peso, que neste caso é 1.

Em certos cenários tal limitação não é importante, mas noutras situações não é bom, pois não discrimina a cardinalidade da ocorrência do termo, apenas a assinala.

É um dos métodos mais simples pois não é normalizado nem tem em conta a ocorrência do termo no conjunto. Tem como vantagem não requerer qualquer calculo pós-contagem.

E toma a seguinte representação:

$$v_{ij} = \begin{cases} 1, f_{ij} > 0 \\ 0, else \end{cases}$$

Base de Ocorrências de Termos Esta base representativa sendo mais complexa que a binária também é relativamente simples, pois também não é normalizada. No entanto permite obter mais informação que a representação em base binária, pois permite registar a cardinalidade de ocorrências dos termos na frase. Quando um termo ocorre n vezes, é o próprio valor n que o representa, ou seja:

$$v_{ij} = f_{ij}$$

Base de Frequência de Termos Esta base representativa é em tudo idêntica a Base de Ocorrências de Termos, com a excepção de sofrer normalização Euclidiana. Assim um termo em vez de ser representado pelo seu número absoluto de ocorrências é representado no intervalo $[0, 1]$. Este valor é obtido a partir de:

$$v_{ij} = \frac{f_{ij}}{fd_j}$$

TFIDF Por fim temos a base representativa TFIDF[15], que tem em conta a ocorrência do termo no documento e no conjunto de todos os documentos ou corpus.

Um termo por um lado tem mais peso quanto mais se repete no documento, mas por sua vez esse termo perde peso quantas mais vezes se repetir no corpus dos documentos. Esta base representativa aplicada a este trabalho em vez de utilizar como unidade de classificação o documento utilizou a frase.

Obtém-se pela fórmula:

$$v_{ij} = \frac{f_{ij}}{fd_j} \log\left(\frac{|D|}{ft_i}\right)$$

2.6 Métricas de Avaliação

Como for referido anteriormente, os algoritmos de aprendizagem automática podem obter melhores ou piores resultados. Podem ter um bom modelo de aprendizagem que consegue abstrair uma regra universal a partir do conjunto de treino e classificar bem qualquer outro novo objecto, ou então, ter um modelo de aprendizagem deficiente que gera regras que falham ao generalizar para todos os objectos de um determinado domínio. Existem várias medidas de desempenho que podem ser utilizadas para avaliar o modelo.

Matriz confusão. A matriz confusão é uma matriz que permite avaliar o desempenho de um determinado algoritmo de aprendizagem automática, pois para um domínio de testes, relaciona a classificação realizada pelo algoritmo com a classificação verdadeira desse domínio de testes, dando assim não só a quantidade de erros e acertos, mas também os tipos de erros.

A matriz confusão tem a seguinte forma:

$$M = \begin{pmatrix} m_{11} & m_{12} & \dots & m_{1C} \\ m_{21} & m_{22} & \dots & m_{2C} \\ \vdots & \vdots & \ddots & \vdots \\ m_{C1} & m_{C2} & \dots & m_{CC} \end{pmatrix}$$

A matriz confusão é uma matriz $C \times C$ onde os items m_{ij} representam o número de casos da amostra que sendo j foram classificados como i. Para um determinado item m, os casos em que i é igual a j, é um acerto, caso contrário, é um erro. Assim o somatório dos valores de uma determinada coluna j representa o número total das

[15]Sigla proveniente do Inglês, *Term Frequency-Inverse Document Frequency.*

ocorrências reais dessa classe j no domínio de testes, enquanto o somatório dos valores de uma linha i representa o número total das ocorrências estimadas para a classe i.

Nos casos em que o domínio de testes é totalmente caracterizado por apenas duas classes, e a matriz confusão toma a forma 2×2. Nestes casos a matriz de confusão também pode ser denominada de tabela de contingência, e as duas classes são apenas uma, e a sua negação.

Nesses casos a matriz confusão toma a seguinte forma:

		Valores Reais	
		A	B
Valores Estimados	A'	VP	FP
	B'	FN	VN

Assumindo a classe A como a classe positiva, e a classe B como a classe negativa, temos:

- **VP** - O número de verdadeiros positivos, isto é, exemplos positivos que foram correctamente classificados como positivos.
- **VN** - O número de verdadeiros negativos, isto é, exemplos negativos que foram correctamente classificados como negativos.
- **FP** - O número de falsos positivos, isto é, exemplos negativos que foram erradamente classificados como positivos. Este tipo de erro também é conhecido por erro de de tipo 1, ou erro α.
- **FN** - O número de falsos negativos, isto é, exemplos positivos que foram erradamente classificados como negativos. Este tipo de erro também é conhecido por erro de tipo 2, ou erro β.

Um algoritmo de aprendizagem automática que consiga classificar correctamente todos os casos, a matriz confusão daí resultante será uma matriz apenas com valores na diagonal principal.

Exactidão. A exactidão é uma métrica generalista que nos permite perceber a taxa de classificações efectuadas correctamente pelo algoritmo. Embora seja frequentemente usada noutros domínios, é uma métrica "fraca"pois não permite qualificar o erro, apenas quantifica-lo.

Para calcular a exactidão a partir da matriz de confusão, basta utilizar a seguinte fórmula:

$$exactidao = \frac{VP + VN}{VP + VN + FP + FN}$$

Erro. O erro é também uma métrica "fraca", pois apenas quantifica a taxa de erros, não permitindo a discriminação do tipo de erro.

Para calcular o erro, basta utilizar a seguinte fórmula:

$$erro = \frac{FP + FN}{VP + VN + FP + FN} = 1 - exactidao$$

Precisão. A precisão é uma métrica que nos permite avaliar a qualidade da classificação de um determinado algoritmo de aprendizagem automática. Ao contrário da exactidão que apenas dá a taxa de elementos correctamente classificados, a precisão permite saber do conjunto dos elementos classificados como pertencentes a uma classe, os que de facto o são. Permitindo assim observar o tipo do erro ocorrido. Por outras palavras, é o grau de pureza da classificação efectuada.

Para calcular o seu valor a partir da matriz confusão, utiliza-se a seguinte fórmula:

$$precisao = \frac{VP}{VP + FP}$$

Cobertura. A cobertura é uma métrica semelhante à precisão que permite observar o tipo de erro ocorrido na classificação de determinado algoritmo de aprendizagem automática. No entanto ao invés de dar a "pureza"da classificação efectuada, esta métrica dá nos a noção da informação relevante que foi perdida durante a classificação.

Para calcular a cobertura a partir da matriz confusão utiliza-se a fórmula:

$$cobertura = \frac{VP}{VP + FN}$$

Medida-F A medida-F[16] é uma métrica que combina a precisão e a cobertura. Isto surge da necessidade de ter uma métrica que dê uma visão global do desempenho do algoritmo, visto que, usando apenas uma delas corre-se o risco de ter bons valores perto de 100%, tendo no entanto na outra métrica muito maus resultados, o que levanta duvidas quanto a globalidade do desempenho do algoritmo.

[16]Do Inglês, *F-Measure*.

Pode-se cair no erro de considerar que a métrica exactidão dá uma visão geral do erro do classificador, sem ter o problema do desiquilíbrio da precisão e da cobertura. Mas usar a exactidão para dar uma visão geral, pode dar uma noção errada, principalmente quando o número de casos negativos é muito superior ao de casos positivos.

Vamos ver um exemplo prático que ilustra este cenário, e elucida sobre a utilidade da medida-F.

Temos por exemplo um conjunto com um total de 1000 elementos dos quais 995 são casos negativos e apenas 5 casos positivos. O algoritmo faz a classificação sobre esse conjunto e erradamente classifica todos os casos como sendo negativos excepto um dos 5 positivos. Temos então:

$$VP = 1, VN = 995, FN = 4, FP = 0$$

Calculando a exactidão:

$$exactidao = \frac{VP + VN}{VP + VN + FP + FN} = \frac{1 + 995}{1 + 995 + 0 + 4} = 99,6\%$$

Neste caso, obtemos uma exactidão de 99,6%, o que é um resultado bastante bom, se não fosse a singularidade de apenas ter acertado num caso positivo. Assim é fácil de mostrar que tendo uma boa exactidão podemos ter resultados terrivelmente maus em que se falhou quase a totalidade dos casos positivos.

Com a precisão e cobertura obteríamos respectivamente:

$$precisao = \frac{VP}{VP + FP} = \frac{1}{1 + 0} = 100\%$$

$$cobertura = \frac{VP}{VP + FN} = \frac{1}{1 + 4} = 20\%$$

O que também não ajuda muito a perceber a qualidade global dos resultados obtidos. Temos uma boa precisão, pois o pouco que se acertou como casos positivos, são realmente casos positivos e temos uma cobertura má, pois perdeu-se a quase totalidade de casos positivos.

A medida-F surge como uma métrica que resolve estes casos, dando uma boa perspectiva geral do desempenho do algoritmo.

A medida-F essencialmente é uma média ponderada da cobertura e da precisão, e obtém-se pela seguinte fórmula:

$$F_\beta = \frac{(1+\beta^2) \times precisao \times cobertura}{\beta^2 \times precisao + cobertura}$$

Na fórmula surge um novo parâmetro β, que é um parâmetro que permite jogar com a importância atribuída entre a precisão ou a cobertura. Para um $\beta = 1$ atribui-se igual peso à precisão e à cobertura. Assim, o resultado de cada uma das métricas influencia de igual maneira o resultado final.

Com β a tomar valores entre $0 \leq \beta < 1$, é atribuído um peso maior à precisão do que à cobertura. Assim o resultado final será mais influenciado por qualquer alteração no valor da precisão, do que igual alteração no valor da cobertura. Esse desequilibro, é tanto maior, quanto mais β se aproximar de zero. Aliás, quando $\beta = 0$ a importância da cobertura é nula, o que significa que F_0 é igual à precisão.

Com β a tomar valores entre $0 \leq \beta < 1$ o cálculo de F atribui mais importância a pureza da informação obtida do que ao número de elementos relevantes obtidos.

Por outro lado com $\beta > 1$, a cobertura terá maior peso no resultado final de F, quanto maior for o parâmetro β. No limite, quando β tende para infinito:

$$\lim_{\beta \to +\infty} F_\beta = cobertura$$

F obtém o valor da cobertura, já que a precisão tem importância nula. Assim, quando β toma valores $\beta > 1$, a medida-F dá mais importância à quantidade de informação relevante obtida em detrimento do nível de pureza desta.

No decorrer deste trabalho utilizou-se sempre a métrica F com $\beta = 1$ a atribuir igual peso à cobertura e à precisão. No entanto também é muito frequente usar-se o F_2 em que a cobertura tem o dobro do peso da precisão, e o $F_{0,5}$ em que a precisão tem o dobro do peso da cobertura.

2.7 Ferramentas

Durante a realização deste projecto foram utilizadas várias ferramentas auxiliares, com o fim de facilitar a obtenção dos resultados finais.

Algumas ferramentas tiveram uma utilização mais periférica ao projecto e apenas ajudaram a construir estruturas e aplicações auxiliares, enquanto outras tiveram um papel mais central, chegando mesmo algumas a serem integradas no próprio projecto.

2.7.1 Protégé

O Protégé [18] é uma ferramenta de software livre e tem como objectivo ser um sistema de extracção de conhecimento que permita criar, editar, visualizar e manipular ontologias em diversos formatos.

É uma ferramenta desenvolvida em JAVA e utiliza a API Swing de JAVA para o desenvolvimento do seu interface gráfico. No entanto, devido a ter uma grande comunidade de desenvolvimento tem um sem-fim de plugins.

A aplicação começou a ser desenvolvida para ser utilizada em áreas de biomedicina pela Universidade de Stanford em colaboração com a Universidade de Manchester. No entanto, devido ao sucesso da aplicação depressa foi adoptada nas mais diversas áreas onde seja possível criar uma ontologia para representar relações de conhecimento num determinado domínio.

O Protégé tem duas formas de criar, editar e manipular as ontologias. Essas duas formas são:

- Editor Protégé Frames: É uma abordagem que permite ao utilizador criar e popular a ontologia através de frames e de acordo com o protocolo OKBC *(Open Knowledge Base Connectivity Protocol)* [4].

  Nesta forma de criação e edição de ontologias, a ontologia é representada por três conjuntos distintos em simultâneo, sendo eles:

  - Um conjunto de classes organizadas hierarquicamente que representam as características principais de um determinado domínio do conhecimento.
  - Um conjunto de meta-classes que estão associadas ao conjunto das classes e que guardam informação que descreve as propriedades e as relações entre as classes do conjunto de classes.
  - Um terceiro conjunto que colecciona as instâncias criadas a partir das classes e meta-classes, instâncias que não são mais do que classes com os atributos instanciados.

- Editor Protégé OWL: É outra abordagem e a mais utilizada para criar e editar ontologias.

  Esta abordagem utiliza o standard do consórcio *World Wide Web Consortium* (W3C) [9] para a Web Semântica, o OWL *(Web Ontology Language)* [20] para registar a estrutura do conhecimento de um determinado domínio.

  O OWL é uma linguagem desenhada especificamente para este fim, como tal, possui os mecanismos necessários para representar facilmente, classe, relações entre classes e propriedades das classes e também permite representar instâncias criadas a partir dessas classes.

  Uma das vantagens da abordagem OWL em relação à abordagem de Frames é que a abordagem OWL permite descrever a ontologia com uma maior expressividade semântica.

Neste projecto o Protégé serviu como ferramenta auxiliar para criar uma ontologia bastante extensa sobre o domínio envolvente da hotelaria. Esta ontologia além de servir para organizar os conceitos do domínio em questão, também serviu como "molde"para criar instâncias das classes extraídas das descrições de hotéis. Permitiu também guardar a informação referente às expressões regulares e distâncias de Levensthein [48] que ajudavam a identificar no texto as ocorrências de determinadas instâncias.

A criação dessa ontologia seguiu a abordagem Protégé OWL por ser a mais difundida e por usar OWL, que é uma derivação de RDF [5], que por sua vez é um dialecto de XML [58].

2.7.2 Jena Framework

Jena [3], é um Framework bastante completo e foi desenvolvido para construir e alterar aplicações de Web Semântica.

É inteiramente desenvolvido em JAVA e é uma ferramenta de código livre desenvolvida pelos *HP Labs Semantic Web Programme.*

Jena Framework pode ser vista como uma grande API que contém em si diversas API's mais pequenas e mais especializadas. Contém uma API de RDF e RDFS que permite ler e escrever estruturas em RDF, XML, N3 [60] e N-Triples [56]. No entanto estas sub API's não tiveram utilização directa no desenvolvimento deste trabalho.

Outra das sub API's mais importantes é a responsável por aceder, criar e extrair dados de estruturas OWL, a linguagem utilizada para construir a ontologia sobre o domínio da hotelaria bem como as instâncias de classes construídas. Desta forma, esta

API facilitou bastante o trabalho de acesso, edição e extracção de dados da estrutura ontológica desenvolvida.

Por fim, dispõe de uma API que é essencialmente um motor de pesquisa para lidar com SPARQL [40] e SPARUL [29].

2.7.3 WVTool

O WVTool (Word Vector Tool) [61] é uma API de código livre desenvolvida em Java por Michael Wurst. Tal API permite essencialmente transformar documentos de texto em vectores de palavras.

Esta ferramenta surgiu com o objectivo de suplantar a necessidade de uma API que fosse integrada facilmente nas mais variadas ferramentas de análise de documentos, classificação de textos e de extracção de informação de textos. Isto porque estas ferramentas não processam directamente os documentos de texto em língua natural, necessitando geralmente que esses textos estejam representados num formato de vector de palavras, vectores esses que podem variar na forma de cálculo e nas suas representações em diversos formatos e formatações que podem variar com a abordagem pretendida e com a aplicação que vai usar esses vectores de palavras.

O WVTool foi uma ferramenta desenhada com vista à simplicidade e a modularidade.

Os vários módulos, alguns deles opcionais, são configuráveis e parametrizáveis e o conjunto dos vários módulos forma a ferramenta completa, em que o resultado de saída de um módulo é o parâmetro de entrada do modulo seguinte.

Os principais módulos são:

- **Text Loader:** módulo simples responsável apenas pela abertura de um ficheiro com documentos de texto. A abertura desse ficheiro tanto pode ser local como através de uma URL.
- **Decoder:** Este módulo é responsável por efectuar a "descodificação"dos textos do documento a analisar. Caso o documento contenha apenas texto simples corrido não é efectuada qualquer descodificação. No entanto é possível efectuar a descodificação, caso o documento contenha os textos nalgum tipo de Markup Language (eg; HTML [59], XML [58], XHTML [57]). Também permite inclusivamente extrair o texto a partir de documentos em formato PDF.
- **Code Mapper:** Este módulo é opcional e é responsável, caso isso seja necessário, de transformar a codificação de caracteres[17] dos textos do documento a analisar,

[17]Do Inglês, *encoding*.

noutra codificação pretendida.

- **Tokenizer:** Módulo responsável por partir o texto em *tokens*, permitindo assim futuramente que cada palavra seja representada e analisada individualmente, de modo a usar uma representação do vector de palavras. Este *tokenizer* é uma versão simples, que apenas tem em conta os caracteres alfanuméricos e os não alfanuméricos, e que considera como caracteres separadores dos *tokens* todos os caracteres não alfanuméricos do padrão Unicode.

- **Word Filter:** Este módulo é opcional e é responsável por filtrar palavras nos textos a analisar. Tendo em conta uma lista de palavras de filtragem[18], sempre que encontra uma dessas palavras da lista, esta é excluída do texto para análise. Este módulo serve assim para filtrar palavras não discriminativas das classes, como por exemplo as palavras *a, o, da, de, do.* Estas são palavras muito comuns em todos os textos da língua portuguesa, no entanto são geralmente supérfluos e inúteis para a análise e classificação dos textos, sendo assim frequentemente eliminadas.

  A API traz uma lista de palavras de filtragem para o idioma inglês e alemão, mas é possível arranjar ou criar uma lista de palavras de filtragem para Português. Este módulo também permite usar mais que uma lista de palavras de filtragem, efectuando uma disjunção sobre elas e obtendo assim uma lista final de palavras de filtragem maior.

- **Stemmer/Reducer:** O *Stemmer* é outro módulo opcional, que tem como objectivo reduzir as palavras ao seu radical. No idioma português tal ferramenta é bastante importante, pois é uma linguagem rica em variantes e derivações da mesma palavra. No caso de nomes, temos o seguinte exemplo com várias palavras diferentes com o mesmo radical.

  Gato – gat
  Gatos – gat
  Gatão – gat
  Gatões – gat
  Gata – gat
  Gatas – gat
  Gatinha – gat
  Gatinhas – gat

  No caso dos verbos, das suas diferentes formas verbais, tempos e pessoa em que se encontram conjugados, também são ricos em radicais comuns, por exemplo:

[18]Do Inglês, *stopwords.*

Andava – and
Andei – and
Ando – and
Andarei – and
Andarás – and
Andaram – and
Andais – and

Este módulo traz vários algoritmos de *stemmer*: Porter Stemmer [19], Lovins Stemmer [25], e o Snowball Stemmer [28]. Os dois primeiros apenas funcionam para o idioma Inglês, e o Snowball tem suporte para muitas línguas, incluindo as línguas românicas e como tal o Português.

- **Vector Creation:** Este módulo é responsável por criar a representação final do vector de palavras. No entanto, esse vector de palavras pode ser criado conforme diferentes abordagens, pois tais abordagens podem ser mais ou menos adequadas a diferentes contextos e casos concretos de onde se esteja a extrair informação. Por sua vez, dentro de cada abordagem podem existir variações quanto à normalização dos resultados.

2.7.4 Weka

Weka (Waikato Environment for Knowledge Analysis) [17], é uma ferramenta de aprendizagem automática, que numa só aplicação reúne vários algoritmos e abordagens. Recorrendo a técnicas de mineração de dados e aprendizagem automática permite analisar a informação e efectuar modelação preditiva por indução. O Weka é uma ferramenta de software livre desenvolvida em Java pela Universidade de Wakato da Nova Zelândia.

A grande vantagem do WEKA, é que os diversos algoritmos de aprendizagem automática têm diferentes formatos de entrada e de saída, mas o WEKA consegue uniformizar tudo, e com os mesmos dados de entrada é possível efectuar testes com diversos algoritmos.

O Weka dispõe de um interface gráfico bastante completo e elaborado que permite analisar e fazer comparações entre os vários algoritmos, pois um algoritmo bom em determinado contexto pode ser bastante mau noutro contexto. Possui boas ferramentas de análise numérica e gráfica para fazer a correcta avaliação dos algoritmos mais adequados para um determinado conjunto de dados. Também é possível analisar e escolher os algoritmos conforme os seus desvios, ou seja, se tem boa cobertura, boa precisão ou ambas resultando num bom F1.

Nativamente trabalha com os dados no formato `.ARFF`, no entanto também permite importar dados em formato csv, libsvm, bsi e xrff. Dispõe ainda de um pré-processador que permite transformar e importar dados noutros formatos.

Embora o Weka disponha de um elaborado interface gráfico, também permite ser executado através de consola, o que é bastante útil quando é necessário automatizar e acelerar processos de criação de modelos e testes.

O Weka foi uma das principais ferramentas auxiliares para o desenvolvimento deste projecto, pois para além de ter sido integrado no próprio projecto, foi também, sobre esta ferramenta que se efectuaram bastante estudos de optimização sobre as bases representativas e os algoritmos de aprendizagem automática, que melhor se adequavam ao seu contexto.

O formato nativo do WEKA (`.ARFF`) é um ficheiro de texto dividido em três partes principais:

- RELATION onde se define o nome da relação em estudo
- ATTRIBUTE é a listagem dos atributos comuns aos objectos em estudo, atributos esses que vão permitir que os objectos sejam diferenciados entre si. Esses atributos podem ter diversos tipos: *Numeric*, *Nominal*, *String*, *Date*, *Relational* e *Class*.

  O atributo *class* é um atributo especial, pois não é como os demais atributos que qualificam ou quantificam determinada característica do objecto; o atributo *class* é onde se define a classe a que o objecto pertence.
- DATA é uma listagem dos dados referentes aos diversos atributos e a classe de cada um dos objectos em estudo; é a representação de algumas instâncias das várias classes possíveis.

  Cada objecto tem uma lista de atributos terminada com a classe a que o objecto pertence.

Na Tabela 2.3 surge um exemplo de um ficheiro `.ARFF` bastante simples.

2.7.5 .NET Framework

.NET Framework é uma plataforma unificada que permite o desenvolvimento e a criação de aplicações e serviços. Durante a realização deste trabalho, vários módulos foram desenvolvidos recorrendo a esta tecnologia.

```
@RELATION iris

   @ATTRIBUTE sepallength  NUMERIC
   @ATTRIBUTE sepalwidth   NUMERIC
   @ATTRIBUTE petallength  NUMERIC
   @ATTRIBUTE petalwidth   NUMERIC
   @ATTRIBUTE class        {Iris-setosa,
                            Iris-versicolor,
                            Iris-virginica}

 @DATA
   5.1,3.5,1.4,0.2,Iris-setosa
   4.9,3.0,1.4,0.2,Iris-setosa
   4.7,3.2,1.3,0.2,Iris-setosa
   4.6,3.1,1.5,0.2,Iris-setosa
```

Tabela 2.3: Exemplo de ficheiro `.ARFF`

Tal plataforma tem uma arquitectura semelhante à plataforma JAVA, pois as aplicações nela criada podem ser executadas em qualquer dispositivo, mesmo que tenha uma arquitectura diferente.

Ao criar uma aplicação através desta plataforma, esta não vai ser exclusiva para o sistema/arquitectura em que foi criada, mas vai poder ser executada em qualquer sistema que tenha a plataforma .NET instalada, porque ao compilar a aplicação em vez de ser criado código nativo é criado código binário[19] que será executado pela máquina virtual instalada em cada sistema.

Ao contrário da plataforma JAVA, em que as aplicações só podem ser unicamente em JAVA, na plataforma .NET as aplicações podem ser escritas em mais de 20 linguagens de programação distintas, tais como C#, COBOL, C++, Fortran, Haskell, JAVA, Java script, LUA, Pascal, Perl, Python, Ruby e SmallTalk, entre outras.

Também de um modo semelhante à plataforma JAVA, a plataforma .NET funciona em dois passos. As aplicações criadas na plataforma .NET são compiladas duas vezes, sendo esta característica que permite a portabilidade entre sistemas.

A primeira compilação ocorre sobre a linguagem em que a aplicação foi desenvolvida e gera uma linguagem comum intermédia à plataforma .NET. Essa linguagem intermédia em código binário é igual em qualquer sistema em que a aplicação seja compilada e independentemente da linguagem em que a aplicação tenha sido escrita.

Cada uma das linguagens suportadas pela plataforma .NET tem o seu próprio compilador específico para a linguagem intermédia.

[19]Do Inglês, *binary code*.

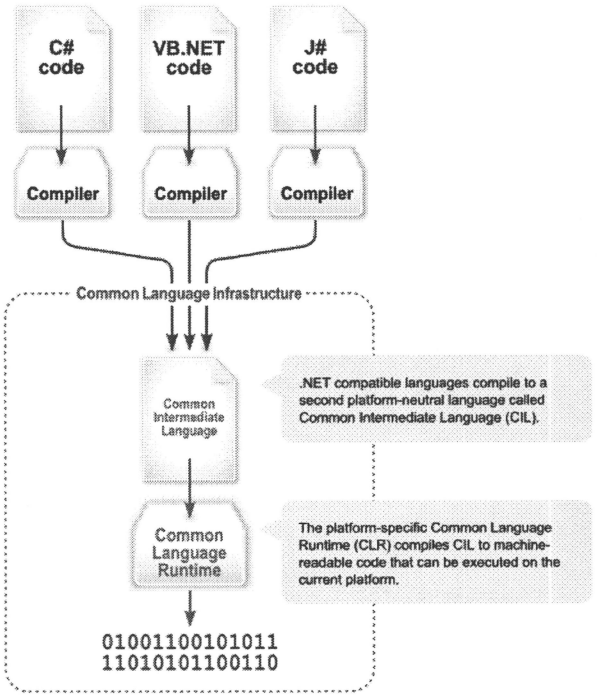

Figura 2.14: Esquema da Compilação multi-linguagem da plataforma .NET

A partir da linguagem intermédia, cada sistema tem um segundo compilador que transforma a linguagem intermédia em linguagem máquina nativa. Este segundo compilador é distinto para cada arquitectura do sistema em que corre a aplicação de modo a poder ser compatível e optimizado a cada arquitectura.

A plataforma .NET à semelhança da API de JAVA, também dispõe de uma enormíssima biblioteca de classes utilitárias, e que permitem ao programador usar um número bastante elevado de componentes, funções e objectos sem os ter de programar de raíz.

Essa biblioteca de classes é bastante completa e dispõe de muitos recursos na criação de interfaces gráficos, estruturas de dados, conectividade a bases de dados, criptografia, criação de interfaces Web, Web services e vários protocolos de comunicação.

Esta plataforma não é estática; desde o seu início em 2002 tem tido, ao longo do tempo, uma grande evolução, surgindo em cada nova versão mais funcionalidades e potencialidades.

Na Figura 2.15 são apresentados os diversos módulos que foram surgindo com a evolução da biblioteca ao longo do tempo.

Os módulos deste trabalho criados na plataforma .NET foram desenvolvidos na versão 3.5.

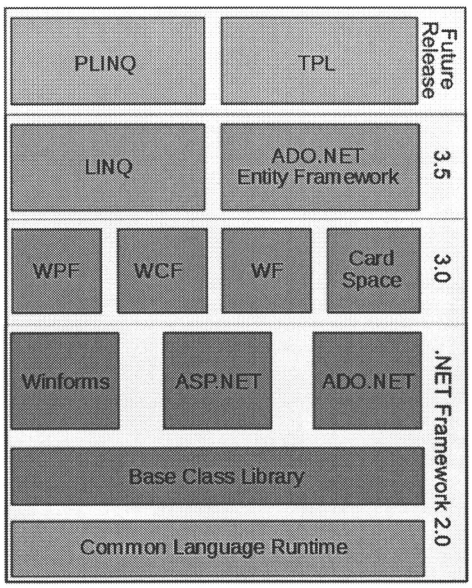

Figura 2.15: Pilha dos módulos da API da plataforma .NET

2.7.6 Visual Studio

O Visual Studio é um IDE, (Ambiente de Desenvolvimento Integrado[20]) bastante poderoso que permite desenvolver diversos tipos de aplicações, em diversas linguagens de programação e para diversas plataformas.

Permite desenvolver aplicações, desde as mais simples de linha de comandos, às mais completas com interface gráfico. Permite também desenvolver páginas Web e aplicações Web baseadas em Web Services.

Com este IDE é possível desenvolver projectos para diversas plataformas, como: .NET Framework [30], .Net compact Framework [30], Windows Mobile [31] e Windows CE[32].

A nível de linguagens o IDE Visual Studio permite trabalhar nativamente com C, C++, Visual Basic, C#, F#, M, Python. No entanto, permite também desenvolver em qualquer outra linguagem suportada na Framework .NET através da instalação de módulos de funcionalidades extra[21].

Este IDE suporta também nativamente várias tecnologias da área Web, tais como XML [58], XSLT [8], HTML [59], XHTML [57], Java Script [6] e CSS [55].

A nível de edição de código, o Visual Studio possui as tradicionais ferramentas dos IDE mais complexos, tais como realce de sintaxe/coloração de sintaxe[22], completação

[20]Em Inglês, *Integrated Development Environment.*
[21]Do Inglês, *plugins.*
[22]Do Inglês, *syntax highlighting.*

de código[23], permite inserir marcadores[24] no código e colapsar blocos de código.

A nível de *debug*, a característica mais notável é que permite efectuar debug a dois níveis: um durante a primeira compilação a nível do código intermédio, e outro durante a segunda compilação a nível da compilação para linguagem máquina.

Uma outra funcionalidade curiosa é a possibilidade de desenhar classes e diagramas de classes em UML. Que o próprio IDE utiliza para gerar automaticamente código C# ou VB .NET.

Permite ainda criar interfaces gráficas, desde as mais simples WinForms até as mais complexas interfaces Web baseadas em WPF[25] [33].

A adaptação a este IDE para a construção dos módulos deste projecto que utilizaram a plataforma .NET foi relativamente fácil pois é bastante semelhante a outros IDE's de JAVA com os quais estava familiarizado, nomeadamente o NetBeans e o Eclipse.

2.7.7 WCF

Resumidamente o WCF[26] [34], é uma API inserida no .NET Framework 3.0 em Dezembro de 2006. Esta API foi criada seguindo uma abordagem SOAP[27] [7] e tendo em vista o objectivo de permitir aos programadores da plataforma .NET a fácil criação de serviços, webservices e criação de aplicações distribuídas segundo a filosofia dos sistemas distribuídos.

A comunicação WCF entre processos clientes e processos servidores é feita sobre mensagens SOAP codificadas num formato binário optimizado de alta performance.

No entanto, um processo WCF, seja ele um cliente ou servidor, não está restrito a comunicar com outros serviços exclusivamente WCF. É possível comunicar com outros processos que não sejam WCF, sendo nestes casos a comunicação feita com a informação codificada em XML. Neste caso a performance é menor do que com o formato binário nativo.

Neste projecto, foi seguida uma abordagem cliente-servidor através de um Web service WCF, de modo a permitir que a aplicação final estivesse disponível online na Web ou disponível remotamente em canal próprio.

---

[23] Do Inglês, *code completion.*
[24] Do Inglês, *bookmarks.*
[25] Sigla do Inglês, *Windows Presentation Foundation.*
[26] Sigla do Inglês, *Windows Communication Foundation.*
[27] Sigla do Inglês, *Simple Object Access Protocol.*

2.7.8 SQL Server

O SQL Server é um sistema de gestão de bases de dados relacionais desenvolvido pela Microsoft.

À semelhança de outros, este sistema permite desenvolver consultas[28], vistas e manipular os dados através delas.

A plataforma permite ainda integrar o código SQL em aplicações da plataforma .NET, e receber instruções de consulta, inserção e edição provenientes das diversas linguagens de programação suportadas pela plataforma .NET.

As suas linguagens de consulta são o ANSI SQL e uma extensão de SQL, a T-SQL que foi desenvolvida pela Microsoft em conjunto com a Sybase.

A utilização desta ferramenta deveu-se ao facto de se ter usado dados reais na elaboração deste trabalho. Estes dados (vários campos associados às descrições de milhares de hotéis reais e actuais) provenientes da VIATECLA encontravam-se numa base de dados SQL-Server, daí a necessidade de recorrer a tal ferramenta para conseguir utilizar e trabalhar nativamente sobre os dados.

[28]Do Inglês, *queries*.

Capítulo 3

Trabalho Relacionado

O presente capítulo faz uma síntese do estado da arte relacionado com a área deste trabalho. Incidindo sobre duas áreas distintas, a área de classificação de textos em língua natural na Secção 3.1 e a área da extracção de informação a partir de textos em língua natural na Secção 3.2.

3.1 Classificação de Textos em Língua Natural

Existem diversos trabalhos na área de classificação de textos que se dividem em diferentes abordagens dependendo do nível de informação básica com que trabalham. Há trabalhos que utilizam um nível de sub-palavra em que a palavra é decomposta e analisada a sua morfologia. Outros trabalhos trabalham ao nível da palavra, em que a palavra é a entidade básica e é analisada a sua informação lexical. Surgem ainda abordagens que trabalham num nível acima da palavra, uns seguem a via semântica, em que o sentido dos textos é analisado, e a via pragmática em que o texto é analisado de acordo com o seu significado e contexto em que ocorre.

Este trabalho foi desenvolvido ao nível da palavra, pois toma a palavra como informação básica e utiliza a técnica chamada de saco de palavras[1]. A utilização da palavra como nível de informação básica é a aproximação mais comum para sistemas de classificação de textos devido à sua simplicidade de implementação e desenvolvimento, visto ser das que mais facilmente adapta a informação textual à maior parte de algoritmos de aprendizagem automática.

Por sua vez, a maior parte das abordagens que assentam neste nível, usam o saco de palavras que não é mais que uma colecção de palavras de um domínio. Os algoritmos

[1]Do Inglês, *bag-of-words*.

de aprendizagem utilizam essas palavras como atributos para criar os modelos de aprendizagem e conseguir posteriormente classificar textos de forma automática.

Existem no entanto, técnicas mais complexas que para além da manipulação directa sobre o saco de palavras, recorrem ainda à analise morfológica, sintáctica e semântica dos textos.

Thorsten Joachims [22] efectua testes de comparação entre diversos algoritmos para a mesma situação experimental: Naive Bayes, Rocchio, C4.5, K-n vizinhos os SVM com *kernel* RBF e polinomial. Joachims chega a conclusão que os SVM apresentam os melhores resultados quer a nível de eficácia temporal quer a nível de desempenho da precisão e cobertura.

Joachims explora os motivos que conduzem os algoritmos SVM a obterem bons resultados de classificação de textos em relação a outros algoritmos quando num contexto de aprendizagem supervisionada, e conclui:

- Os SVM permitem trabalhar facilmente e sem quebra de performance com uma elevada dimensão de atributos de entrada em problemas de classificação de texto.
- Por permitir um grande número de atributos sem quebras de performance, não é necessário descartar parte dos atributos menos importantes para se incrementar a performance.
- Os SVM's suportam o modo *sparse* nos atributos de entrada. O modo *sparse* apenas representas atributos com valores diferentes de zero, o que é muito importante, pois em problemas de classificação de texto a maioria dos atributos tomam valores zero. Assim os SVM's ao suportarem este modo, é lhes possível eliminar grande parte da redundância inútil dos dados de entrada, contribuindo também para a menor dimensão dos ficheiros de entrada.
- Por fim Joachims refere que a maior parte dos problemas de classificação de textos com diversas classes são geralmente representados por modelos lineares, o que encaixa perfeitamente no perfil dos SVM com *kernels* linear, sendo desnecessária a utilização de *kernels* com uma maior complexidade.

Para terminar, apenas há a referir que os trabalhos de Joachims foram inteiramente desenvolvidos no idioma inglês.

Akiko Aizawa [39] apresenta um método que incorpora técnicas de processamento de língua natural nos tradicionais processo de classificação de textos. Para isso utiliza modelos de linguagem probabilísticos baseados nos pesos dos termos, estimação de ocorrência de termos e recurso a analisadores morfológicos das palavras *(POS, part-of-speech)*.

Os resultados apresentados mostram que a utilização das técnicas de processamento de língua natural na classificação de textos utilizando SVM melhora visivelmente o desempenho do classificador (cobertura e precisão) quando comparado com a abordagem tradicional do simples saco de palavras.

Este trabalho também foi inteiramente desenvolvido no idioma Inglês.

Gonçalves e Quaresma [16] comparam o desempenho entre os algoritmos de SVM, C4.5 e Naive Bayes, na classificação de textos jurídicos na língua Portuguesa.

Além de apresentarem a tradicional abordagem pelo saco de palavras, foram usadas técnicas de transformação das palavras no seu lema e de selecção de palavras utilizando a sua classificação morfológica e recorrendo a listas de palavras de paragem para descartar palavras irrelevantes.

A nível de resultados, os melhores resultados de F1 foram apresentados pelo C4.5, mas seguido de muito próximo dos SVM com uma diferença quase irrelevante. No entanto a nível de performance temporal o SVM ganha com grande vantagem, pois para a experiência em causa demorou apenas 10 minutos ao invés das 8 horas necessárias para o processamento pelo C4.5. Já as técnicas de transformação utilizadas mostraram um significativo melhoramento dos resultados finais em comparação ao simples saco de palavras.

Silva e Vieira [51] apresentam um trabalho semelhante ao anteriormente descrito, mas utilizando um conjunto de dados distintos, mas chegam a conclusões semelhantes. O corpus do estudo teve como base um conjunto de artigos do Jornal Folha de São Paulo de 1994.

Neste trabalho foi realizada uma comparação entre os algoritmos SVM e árvores de decisão em tarefas de classificação de texto. Para além da utilização do saco de palavras tradicional, também recorrem a informação linguística na construção dos atributos que descrevem os documentos. Para a extracção de informações linguísticas foi utilizado o analisador sintáctico PALAVRAS [11], e a selecção dos atributos era feita na classe gramatical das palavras.

A nível de resultados, constataram que as classes gramaticais discriminantes que mais melhoravam os resultados dos SVM e das árvores de decisão foram os substantivos e nomes próprios, e os mais irrelevantes os verbos e sintagma nominais(que já funcionam ao nível da multi-palavra).

O desempenho dos algoritmos SVM e árvores de decisão foi semelhante, embora tenham constatado que para poucos atributos, as árvores de decisão levam uma ligeira vantagem sobre os SVMs, situação que se inverte quando o número de atributos aumenta.

Noutro trabalho de Silva [50], é a feita a mesma experiência, mas utilizando redes

neuronais artificiais para a etapa de classificação. Os resultados desta abordagem são bons, no entanto ficam ligeiramente atrás dos resultados obtidos pelos SVM em Silva [51].

Bloehdorn [1], constata melhoramentos em tarefas de classificação de textos com recurso ao uso de características extraídas de ontologias sobre o domínio dos textos a ser classificados. O estudo incide sobre o domínio da medicina e as ontologias auxiliares à classificação são geradas automaticamente através de aprendizagem não supervisionada.

A abordagem é baseada na distribuição de hipóteses, verificando durante o processo de classificação, se os termos analisados são semanticamente similares aos do contexto no qual estão inseridos na ontologia. Nesta fase pode ocorrer um processo de generalização, que tem como finalidade adicionar conceitos mais gerais aos conceitos existentes na ontologia. Desta forma, cria-se uma abrangência maior para com os conceitos na ontologia dos diversos documentos analisados e que possuem características em comum.

À semelhança de Bloehdorn [1], Wu *et al* [12] realiza a classificação de texto com recurso a informação de ontologias de domínio adquiridas dos textos, através de regras morfológicas e métodos estatísticos.

A ontologia de domínio pode ser utilizada para identificar a estrutura conceptual das frases de um documento, e assim classificá-la. Além disso, pode ser utilizada como base para outras aplicações além da classificação de textos, como por exemplo, sistemas de pergunta e resposta e sistemas de organização de conhecimento.

No mesmo trabalho foi efectuada a comparação de resultados com as diversas ontologias geradas em bruto e com versões das mesmas revistas por humanos, onde as ontologias revistas apresentavam resultados ligeiramente melhores.

Segundo os autores a utilização de ontologias de domínio apresenta vantagens relativamente a outros mecanismos de representação do conhecimento, já que podem ser lidas, interpretadas e editadas por seres humanos.

Em Cordeiro [10] é efectuada uma extracção de elementos relevantes em textos e páginas da Web. Este trabalho tem duas partes, sendo a primeira dedicada a classificação automática de textos no domínio relativo à venda de habitações e a segunda parte relativa à extracção de elementos relevantes referentes aos mesmos anúncios.

A primeira parte, resume-se a classificar os textos em duas classes, textos de venda de habitações e textos de não venda de habitações.

Para isso foram utilizadas e testadas três abordagens distintas de modo a poder ser efectuado um estudo da qual obteria melhores resultados. As abordagens foram k-n vizinhos mais próximos, Naive Bayes e Naive Bayes com escolha de atributos. A

diferença entre o Naive Bayes e Naive Bayes com escolha de atributos(50 e 200), é que no primeiro todas as palavras pertencentes ao saco de palavras de anúncios funcionam como atributos, enquanto com escolha de atributos só alguns dos mais relevantes são escolhidos para servirem de atributos.

A nível de resultados, Cordeiro concluiu que os melhores eram obtidos com o Naive Bayes de 200 atributos, seguindo-se o Naive Baies com 50 atributos, depois o Naive Bayes com todos os atributos e por fim o k-n vizinhos mais próximos.

3.2 Extracção de Informação em Textos

Existe uma diversidade enorme de trabalhos realizados que seguem diversas vias e abordagens no processo de extracção da informação de textos em língua natural. No entanto, na área de extracção de informação é obrigatório referir o ciclo de conferencias MUC do inglês *"Message Understanding Conferences"* decorridas entre 1987 (MUC 1) até 1997 (MUC 7) que promoviam a competição de sistemas de extracção de informação e que foram promissoras para a criação de vários sistemas inovadores e que ainda hoje são referência.

3.2.1 Sistemas MUC

Riloff [46] apresentou o primeiro sistema de extracção baseado na criação de um dicionário de extracção, denominado de AutoSlug. O dicionário consistia numa colecção de padrões de extracção designados por nós de extracção. Os nós de extracção eram apenas uma palavra de activação com algumas regras de restrição linguística a serem aplicadas ao texto de onde se pretendia extrair informação.

Este sistema era treinado com um conjunto de exemplos anotados manualmente e por norma as palavras de activação eram nomes, verbos ou substantivos. O sistema tinha um analisador sintáctico que classificava as palavras, após uma palavra de activação ser desencadeada eram evocadas as respectivas regras de restrições que iam determinar se essa palavra seria ou não considerada para a extracção de informação.

Dois anos mais tarde em Riloff [47] surgiu com uma nova versão que apresentava alguns melhoramentos significativos. As palavras de activação passaram a poder ter mais do que um conjunto de regras de extracção de modo a permitir ter diversos contextos e significados diferentes. Também deixou de ser necessário anotar todos os exemplos de treino, sendo apenas necessário anotar os mais relevantes para o domínio do problema em questão.

Nas ultimas conferencias realizadas, a MUC-6 e MUC-7 um dos sistemas que mais se destacou devido aos seus bons resultados foi o LOLITA [15] que tem origem no inglês *"Large-scale, Object-based, Linguistic Interactor, Translator, and Analyser"*. É um projecto em desenvolvimento desde 1986 no laboratório de engenharia de linguagem natural da Universidade de Durham.

Este sistema é completamente genérico para língua natural e pode ser facilmente adaptado a qualquer domínio. O sistema consiste numa complexa e vasta rede semântica, com mais de 100000 nós e 1500 regras gramaticais. Esta abordagem é inspirada na já célebre *WordNet* [13]. Este sistema é considerado um dos maiores sistemas de processamento de linguagem natural e serve de motor para um conjunto de aplicações implementadas que vão desde a tradução entre línguas até à extracção de informação como é o caso do trabalho descrito por Marco Constantino em [26].

A configuração e utilização do sistema num domínio específico é feita com relativa facilidade [27], através da criação de um conjunto de modelos específico para o domínio do problema em analise. Os modelos neste caso, não são mais que uma estrutura com um conjunto de campos para serem preenchidos segundo regras explicitamente definidas no próprio modelo.

Como ponto menos forte deste sistema, há a apontar o facto que as regras de extracção de elementos relevantes terem de ser definidas manualmente pelos utilizadores. Através dos modelos que dão alguma liberdade de ajuste e de afinação em domínios relativamente pequenos torna-se pouco eficiente de executar em domínios muito latos e abrangentes.

O grande poder do sistema LOLITA reside na sua base de conhecimento interna, que assenta numa rede semântica, que permite um processamento muito bom sobre a linguagem natural.

Krupa [44] participou na MUC-6 com o sistema Hasten, onde obteve bons resultados na competição desse ano.

O sistema também se baseia em nós de extracção com palavras activadoras e regras de semântica. As palavras activadoras têm de ser marcadas à mão para cada domínio, no entanto as regras de semânticas de cada palavra são criadas automaticamente com algoritmos de aprendizagem.

Após se criar a lista de nós activadores com as palavras e respectivas regras, o sistema está apto a extrair informação relevante de novos textos.

Sempre que se encontra uma palavra activadora no texto, são extraídas as suas regras de semântica e comparadas com às registadas no nó de activação dessa palavra. Se as regras estiverem até uma determinada distância, a palavra é considerada, e a

informação extraída.

Soderland [14] apresentou na MUC o sistema Crystal. O sistema era baseado num dicionário de extracção com uma série de palavras importantes no domínio em causa, conjuntamente com as respectivas restrições de validação para a extracção. Essas restrições incidiam nas estruturas morfológicas, sintácticas e de conceito.

Para criar o conjunto de treino as palavras tinham de ser etiquetadas ao nível da sintaxe e da semântica e eram ainda identificadas as ocorrências de conceitos. Os conceitos podiam variar consoante o domínio em estudo, mas por norma eram utilizados os conceitos "Pessoa Genérica", "Nome Próprio", "Organização Genérica", "Evento"e "Empresa".

Com esse conjunto de treino, o sistema infere regras de restrições à extracção da informação.

3.2.2 Sistemas fora do âmbito MUC

Em Cordeiro [10], já referenciado na Secção 3.1 de classificação de textos em língua natural, a segunda parte desse trabalho é relativa à extracção de elementos relevantes nos anúncios seleccionados na primeira parte do trabalho. Os elementos relevantes eram referentes ao tipo, preço e o local da habitação.

Para a fase de extracção foram usadas duas abordagens, sendo elas a extracção por via de regras pré-definidas manualmente e outra por via de regras definidas através de um sistema de aprendizagem com árvores de decisão com o algoritmo C4.5.

Tanto numa abordagem como na outra, as regras obtidas diziam respeito aos diversos elementos que se pretendiam extrair (tipo, local e preço), e as regras simplesmente definiam a vizinhança do elemento em análise para extracção.

No entanto Cordeiro chegou à conclusão que a abordagem com regras criadas pelo C4.5 era superior na extracção dos elementos relevantes para os três tipos de elementos (tipo, local e preço).

Cordeiro teve ainda de estudar a dimensão ideal do contexto a analisar, ou seja a quantidade de palavras antes e depois a serem analisadas de modo a decidir a extracção ou não da informação. Chegou à conclusão que o valor em que maximizava a cobertura e precisão era com janela de 4, ou seja analisando 4 palavras antes e 4 palavras depois.

Capítulo 4

Descrição do Trabalho

A abordagem escolhida para desenvolver e atingir os objectivos propostos neste trabalho seguiram uma linha de dividir e conquistar[1], em que se idealizaram várias pequenas etapas no projecto. Progressivamente cada etapa seria superada individualmente através de pequenas ferramentas e aplicações que se focavam em problemas concretos e mais simples. No final obter-se-ia um conjunto dessas várias ferramentas que em conjunto seriam capazes de resolver todo o problema.

A esse encadeamento de passos lógicos ou aplicações chamamos *pipeline*.

O *pipeline* é dividido em quatro partes principais, sendo elas:

- Identificação e classificação das frases
- Identificação das entidades
- Criação de Instâncias da Ontologia
- Criação de Descrições Textuais Normalizadas

O projecto ainda dispõem de outras três funcionalidades, que embora não sejam fundamentais, são parte integrante: um Web Service que permite disponibilizar a aplicação online, um Avaliador de instâncias OWL que permite retirar métricas dos resultados obtidos e um módulo de ajustes detalhados[2] que permite facilmente ao utilizador final refinar e melhorar o resultado do pipeline.

Estes três módulos externos foram desenvolvidos com o objectivo de adaptar melhor o projecto aos requisitos operacionais da VIATECLA.

[1] Do Inglês, *Divide and Conquer.*
[2] Do Inglês, *Fine Tuning.*

4.1 Arquitectura Geral

A arquitectura geral do trabalho, resumida de uma forma conceptual de alto nível, pode ser observada na Figura 4.1.

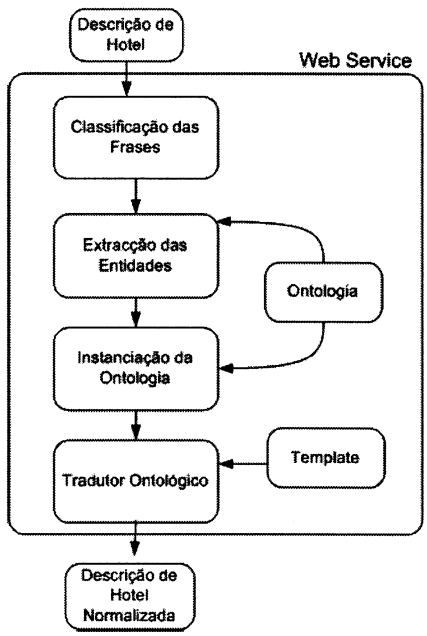

Figura 4.1: Arquitectura Geral

O primeiro módulo **Classificação das Frases** é responsável por dividir as frases consoante o seu tipo. Neste caso concreto as frases podem ser do tipo: Serviços, Equipamentos e Localizações.

O segundo modulo **Extracção das Entidades** é responsável por extrair as entidades de interesse de cada frase após esta ter sido classificada quanto ao tipo. As entidades são extraídas de acordo com os dados que se encontram na ontologia, (apenas as entidades que tenham sido previstas e inseridas na ontologia são extraídas).

O terceiro módulo **Instanciação da Ontologia** cria instâncias dos objectos declarados na ontologia, obtendo-se assim objectos pertencentes a uma determinada classe. Isto permite que os vários objectos extraídos de uma descrição de hotel sejam facilmente manipulados e processados automaticamente.

O quarto e último módulo, designado por **Tradutor Ontológico**, é o responsável por cruzar os objectos obtidos na fase anterior com os modelos de formatação[3] e assim gerar novas descrições de hotéis numa forma normalizada e em linguagem natural, perceptível aos humanos.

Estes quatro módulos serão apresentados em maior detalhe seguidamente.

[3]Do Inglês, *templates*.

4.2 Identificação e Classificação das Frases

Este primeiro módulo é o responsável por receber a descrição do hotel em bruto, efectuar um pré-processamento e submeter as frases a técnicas de classificação automática.

4.2.1 Dicionário de Palavras

Para esta primeira etapa de pré-processamento foi criado um dicionário de palavras ocorrentes nas descrições de hotéis. Para tal utilizou-se um conjunto de 130 descrições completas de hotéis e extraiu-se todas as palavras diferentes com mais de dois caracteres.

Obteve-se assim um conjunto de aproximadamente 600 palavras distintas.

Facilmente se observa que 600 palavras diferentes para 130 descrições é um número relativamente reduzido. A fraca diversidade de termos deve-se ao facto das descrições serem todas sobre o mesmo domínio da hotelaria.

Este dicionário de palavras, será mais tarde utilizado como lista de atributos para discriminar as frases tanto no treino como na fase de classificação automática.

4.2.2 Pré-Processamento

O pré-processamento prepara os dados provenientes da descrição de hotel de modo a serem utilizáveis pelas restantes fases do projecto.

A descrição do hotel, é pré-processada em duas fases. A primeira parte a descrição é dividida em frases, e a segunda separa essas frases em *tokens*.

Na parte final as frases e os seus respectivos *tokens* são processados de modo a extrair atributos úteis para a fase de classificação automática.

Para efectuar esta etapa criou-se um conjunto de aplicações em JAVA e também se recorreu à API auxiliar WVTools descrita no Capítulo 2.7. A parte de separar por frases e tokanizar é bastante simples de realizar. No entanto os dados precisavam de ser convertidos no formato `.ARFF` para posteriormente serem submetidos aos diversos algoritmos de aprendizagem automática por via da ferramenta WEKA, que é descrita no Capítulo 2.7. Para a tarefa de converter os dados num formato `.ARFF` foram criadas pequenas aplicações em JAVA e recorreu-se a algumas ferramentas disponíveis na API do WVTools.

Na construção destes ficheiros `.ARFF` utilizou-se as entradas do dicionário descrito na

secção anterior, como atributos de cada uma das frases.

Assim cada ficheiro `.ARFF` criado tem uma entrada para cada frase da descrição de hotel em análise, e cada entrada é descriminada pelos 600 atributos provenientes do dicionário de palavras que tomariam valor caso a palavra que o atributo representa ocorresse na frase em análise.

No entanto, a ocorrência ou não de um determinado termo da frase pode ser representado de diversas formas. São as diversas bases representativas já descritas, tendo sido efectuado um estudo para determinar qual a base representativa mais adequada em termos de qualidade/performance a este trabalho.

4.2.3 Bases Representativas

Como já foi referido, as diversas formas de registar a ocorrência dos atributos pode ser efectuada de diferentes maneiras. Então criou-se um método para obter as diversas bases de representação a partir dos mesmos dados com recurso à ferramenta auxiliar WVTools. Pois permitia que a partir dos mesmos dados, conseguia-se representá-los nas diversas bases representativas. No entanto teve de ser criado código JAVA para tratar alguns dos resultados, pois o WVTools não normalizava todas as bases representativas da maneira pretendida.

As diversas bases representativas usadas em experiência foram:

- Binária
- Ocorrências de Termos
- Frequência de Termos
- TFIDF

Para maior detalhe sobre os pormenores das diversas bases representativas consultar o Capítulo 2.

Por fim, optou-se pela base representativa binária, pois tendo em conta os resultados obtidos nas várias experiências, que podem ser observados no Capítulo 5 onde são claramente comparados os resultados obtidos para a mesma experiência utilizando as diferentes bases representativas.

As linhas de orientação para seleccionar a melhor base representativa, eram escolher a base com a maior cobertura possível e melhor desempenho temporal. A base representativa binária foi aquela que garantiu maiores coberturas, tendo também a vantagem secundária de ser a representação mais "leve" e rápida do ponto de vista computacional.

Os resultados, pretendiam-se orientados no sentido de obter a maior cobertura. Porque preferia-se ter o maior número possível de caso positivos mesmo aumentando o risco de

ter vários falsos positivos. Isto porque a segunda parte do trabalho ter uma forte tolerância aos falsos positivos da primeira fase, em que os falsos positivos seriam descartados, não afectando o correcto funcionamento do pipeline.

Caso contrário, se optasse por dar primazia a uma base representativa que favorecesse a precisão, iria haver menos falso positivos, mas o risco de perder alguns verdadeiros positivos iria ser maior. Uma vez perdidos, seriam completamente excluídos das fases posteriores do trabalho, perdendo-se assim informação relevante.

Concluindo, deu-se maior importância a uma base representativa que beneficiasse a cobertura de modo a maximizar o número de verdadeiros positivos classificados. Mesmo correndo o risco de ter mais falsos positivos, pois os falsos positivos seriam inofensivos nas fases seguintes do trabalho, o que não aconteceria com os falsos negativos.

No entanto a base representativa não era o único factor a influenciar a inclinação dos resultados no sentido de uma melhor cobertura ou precisão. A escolha do algoritmo de classificação também influenciou e muito a inclinação nos resultados, como pode ser observado na secção seguinte.

4.2.4 Classificação Automática

A fase de aprendizagem automática é aquela que permite classificar as frases das descrições de hotéis nos diferentes tipos: Equipamento, Localização e Serviço.

Esta fase de aprendizagem é dividida em duas partes distintas. A que gera um modelo para o algoritmo utilizado, e a fase que utiliza o modelo gerado para classificar as frases.

O corpus que serviu como aprendizagem para a classificação das frases consistiu num conjunto de 330 frases retiradas aleatoriamente de várias descrições reais de hotéis presentes na plataforma KEYforTravel. Esse conjunto de frases contem um conjunto de 4271 palavras e um média de 13 palavras por frase. Estas frases foram classificadas manualmente em relação às classes Equipamento, Localização e Serviço, obtendo-se três conjuntos distintos de classificação, cada um referente a uma classe, ocorrendo 56, 96 e 78 casos positivos em cada conjunto respectivamente. De onde se conclui que a mesma frase pode pertencer a zero classes, a uma, duas ou às três classes simultâneamente.

O facto de obtermos 3 classificações distintas para a mesma frase permitiu posteriormente criar três modelos de aprendizagem distintos, um específico para cada classe.

Esquematicamente a fase de aprendizagem pode ser representada pela seguinte Figura 2.4.

Embora cada algoritmo utilizado tenha mecanismos internos de processamento completamente diferentes na criação do modelo de aprendizagem, do ponto de vista conceptual

do WEKA, essas diferenças são completamente transparentes, pois o WEKA com o mesmo formato de entrada obtém o mesmo formato de saída, independentemente dos mecanismos internos de cada algoritmo utilizado.

Os principais algoritmos utilizados foram o C4.5 (reimplementação J48 do WEKA), SVM's (optimização SMO do WEKA) e o Naive Baiyes (versão Naive Bayes Multinomial do WEKA). Para maior detalhe e informação sobre a arquitectura e funcionamento de cada algoritmo consultar Capítulo 2.

A escolha do algoritmo mais adequado, também está relacionada com os mesmos factores que foram determinantes para a escolha da base representativa. Esses factores foram: boa cobertura e boa performance temporal.

Como já foi referido anteriormente deu-se mais primazia a uma base representativa que obtivesse uma boa cobertura mesmo que sacrificasse um pouco a precisão, agora mantém-se o mesmo em relação à escolha do algoritmo de classificação.

Assim, e após os resultados dos testes apresentados no Capítulo 5, chegou-se à conclusão que o algoritmo que mais favorecia a cobertura era o Naive Bayes, sendo também dos três o algoritmo com melhor performance temporal dentro da plataforma WEKA.

O funcionamento da parte da classificação automática propriamente dito, simplesmente cria automaticamente um ficheiro `.ARFF` para a descrição do hotel. Essa representação é utilizada com os modelos de aprendizagem criados previamente na fase de aprendizagem.

A fase de testes para apurar qual o melhor algoritmo, foi feita através da interface gráfica do WEKA. No entanto, após obter o melhor algoritmo, este foi integrado no pipeline recorrendo à linha de comandos do WEKA, descartando assim o "peso"de carregar em memoria a componente gráfica.

O pré-processamento e criação dos modelos de aprendizagem só é efectuada uma única vez. Esses modelos depois serão sempre os mesmos, tendo apenas de criar a representação `ARFF` de cada nova descrição a analisar.

Os resultados das várias experiências em que se testaram os vários algoritmos sobre as diferentes bases representativas pode ser consultado em detalhe no Capítulo 5 dos Resultados.

Para os testes e comparação entre os diversos algoritmos foi usada a validação cruzada com 10 pastas e testes de significância de 95%.

Esquematicamente a fase de classificação automática pode ser representa pela Figura 2.5.

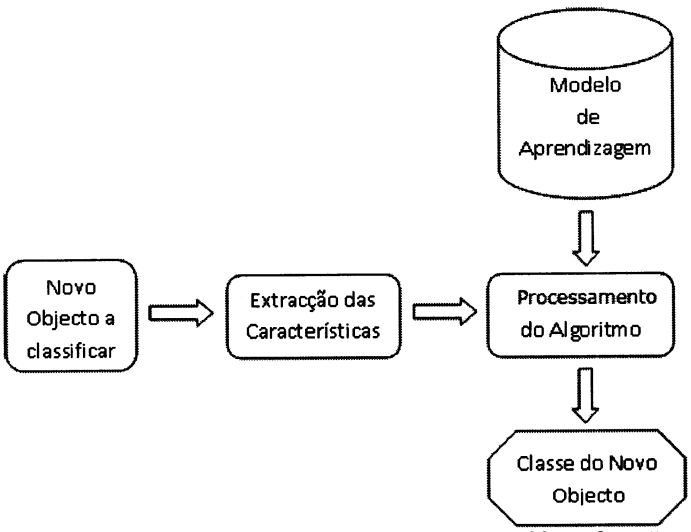

Figura 4.2: Classificação Automática - Classificação

4.3 Identificação das Entidades

Nesta etapa efectua-se a identificação e extracção de entidades relevantes existentes nas frases, que nesta altura já se encontram previamente classificadas quanto ao seu tipo.

Além de extrair as entidades relevantes, esta etapa tem como objectivo extra conseguir extrair entidades a partir de palavras com erros e falhas ortográficas, isto porque os textos das descrições de hotéis são "ricos"em erros e falhas ortográficas, devido a muitas vezes serem textos traduzidos de outras línguas.

Para efectuar a identificação das entidades recorreu-se a duas técnicas: expressões regulares e cálculos de distâncias de Levenshtein, podendo ser observado o seu funcionamento esquemático na Figura 4.3.

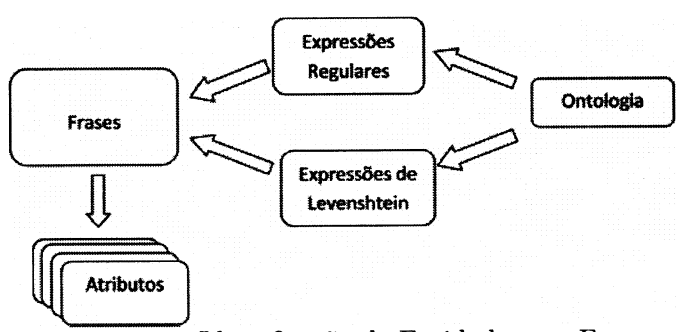

Figura 4.3: Identificação de Entidades nas Frases

Como pode ser observado na Figura 4.3, a extracção dos atributos das frases classificadas pode seguir duas vias, por expressões regulares ou por distâncias de Levenshtein. Cada via será explicada em pormenor nas secções seguintes do documento.

4.3.1 Correspondência de Padrões e Expressões Regulares

Nesta fase, em que já se tem as frases classificadas e agrupadas por tipo, são aplicados métodos de correspondência de padrões[4] com recurso a expressões regulares, devido ao problema do texto das descrições encontrar-se em linguagem natural sem qualquer padrão ou consistência.

Esta característica existe já que as entidades a identificar no texto podem ser descritas através de um conjunto de sinónimos e não apenas por um único termo; a entidade também pode ser descrita por extenso ou apenas por uma abreviatura e ainda pode ocorrer a situação em que está descrita com erros ou falhas ortográficas (sendo bastante comum ocorrerem bastantes falhas a nível de acentos e cedilhas).

Usam-se então as expressões regulares de modo a articular a correspondência de padrões, tornando-a flexível.

De seguida podemos observar alguns exemplos das possibilidades encontradas para os conceitos de Café, Mini-Bar, Televisão:

Café	Mini-Bar	TV
café	mini-bar	tv
Cafe	Mini Bar	T.V.
cafe	mini bar	Televisão
	MiniBar	televisão
	minibar	Televisao
		televisao

Tabela 4.1: Derivações de palavras

Nestes simples exemplos, podemos observar a diversidade de termos que uma determinada entidade pode ter.

Para cada conceito, foi então criado um conjunto de expressões regulares de modo a cobrir a maior parte das suas variantes. Essas expressões regulares estão reunidas na ontologia que foi criada, sendo as expressões regulares um atributo das classes representadas na ontologia. Por exemplo, na classe Café irá existir um atributo com as expressões regulares responsáveis por identificar as variantes de Café. Assim, sempre que se queira que o sistema identifique uma nova entidade, basta criar essa classe na ontologia e as respectivas expressões regulares associadas à identificação desse mesmo conceito.

A criação de uma nova entidade e as respectivas expressões regulares, ou a simples edição e alteração das expressões regulares associadas a uma entidade já existente,

[4]Do Inglês, *Pattern Matching*.

pode ser efectuada por três caminhos distintos:

- Pelo caminho de edição directa no código fonte, fazendo alterações directamente no ficheiro OWL que define a estrutura e elementos da ontologia. Este método é o menos aconselhado para o utilizador final, pois é mais complicado e susceptível a gerar falhas que invalidem e corrompam a estrutura do ficheiro OWL.

- Editando a estrutura ontológica através do Protegé. Esta abordagem é mais simples, segura e simpática já que permite ter uma ideia visual e intuitiva da ontologia. Tem ainda a vantagem, das alterações efectuadas serem validadas pelo editor de modo a garantir no final, obter um ficheiro OWL sempre válido.

- Por último, a edição utilizando o módulo de ajustes detalhados, que também permite criar de um modo visual e intuitivo novas classes de entidades, assim como a edição e correcção de expressões regulares já existentes. Esta abordagem também tem a vantagem de garantir que as alterações efectuadas são validadas mantendo o ficheiro OWL livre de corrupções.Tem também a vantagem de permitir de um modo prático a visualização causa-efeito sobre a edição das expressões regulares sobre um determinado exemplo de texto em língua natural, pois o utilizador sobre um texto de testes obtém as entidades extraídas e caso o resultado obtido não seja o esperado pode alterar as expressões regulares e imediatamente voltar a processar o mesmo texto até concluir o ajuste detalhado de modo a que todas as entidades sejam identificadas e extraídas.

4.3.2 Expressões e Distâncias de Levenshtein

Esta segunda etapa dentro da identificação de expressões surgiu para melhorar os resultados obtidos com as expressões regulares, já que estas conseguem recuperar termos com falhas ortográficas, mas são incapazes de recuperar de erros ortográficos mais graves.

Como já foi anteriormente referido, as descrições de hotéis são ricas em falhas e erros ortográficos devido a serem escritas por pessoas que não têm o Português como língua nativa ou por esses textos serem o resultado de traduções de outros idiomas. Surgiu assim a necessidade de obter um mecanismo robusto capaz de identificar os termos importantes e relevantes para a descrição de um hotel mesmo que estes termos se encontrem mal escritos.

Para tal recorreu-se à concepção e utilização de um algoritmo que utiliza a distância de Levenshtein para determinar se uma palavra mal escrita tem semelhanças com alguma das palavras do domínio das palavras relevantes nas descrições de hotéis.

Este algoritmo só é aplicado a termos que não tenham sido identificados pelas expressões regulares sendo uma espécie de segundo nível de filtragem.

As distâncias de Levenshtein são geralmente utilizadas em correctores ortográficos e o seu funcionamento é muito simples. O corrector ortográfico tem uma lista de palavras correctamente escritas, onde são efectuadas comparações sobre as palavras analisadas;

- Se a palavra que se pretende avaliar constar na lista é porque está bem escrita e não é considerada erro ou falha ortográfica;
- Caso não se encontre na lista é calculada uma "distância"à palavra da lista que mais se assemelha a palavra analisada. Se essa distância for suficientemente curta então a palavra em análise é considerada um erro ortográfico da palavra semelhante encontrada na lista de palavras correctas.

Em termos práticos o que ocorre nos correctores ortográficos é apresentar várias palavras da lista de palavras correctas que tenham uma distância próxima da palavra com erro ortográfico em análise. O seguinte exemplo, utiliza no máximo uma distância de Levenshtein com valor 3 para apresentar sugestões de palavras para corrigir o erro;

Palavra com erro	Sugestões de palavras correctas
proxima	proximal aproxima próxima proviam
expressoes	expressos expresses expressões expresseis expresso
Olivenza	Olivença Oliveira

Tabela 4.2: Variações de palavras com distancia 3

O "segredo"do bom funcionamento de um algoritmo de distâncias de Levenshtein prende-se com cálculo da distância de tolerância, de modo a adequar o mais possível a distância calculada as necessidades da aplicação em causa.

A Tabela 4.3 apresenta os erros anteriores, com a sugestão de palavras de correcção com uma distância de 1 da palavra original:

Palavra com erro	Sugestões de palavras correctas
proxima	proximal aproxima próxima
expressoes	expressos expresses expressões
Olivenza	Olivença

Tabela 4.3: Variações de palavras com distancia 1

Pode observar-se que para os mesmos exemplos o número de palavras sugeridas para correcção é menor, tendo no entanto a desvantagem de ignorar erros que impliquem mais do que uma letra errada na palavra.

A distancia 1 nas palavras grandes limita as hipóteses possíveis para palavras de substituição, mas em palavras pequenas usando a mesma distância 1 continua-se a ter um elevado número de palavras possíveis para substituição. A Tabela 4.4 dá um exemplo dessa situação.

Palavra com erro	Sugestões de palavras correctas
sato	gato pato rato fato mato sapo saco safo sado

Tabela 4.4: Elevado número de variações de palavras com distancia 1

A distância de Levenshtein usada pelo algoritmo deste projecto não é estática, mas sim dinâmica. Tal abordagem deve-se a poder dar uma maior flexibilidade ao algoritmo, para que se adapte caso a caso conforme o termo analisado. Termos maiores permitem distâncias de Levenshtein maiores e termos menores tem de ter distâncias de Levenshtein menores obrigatoriamente para evitar a ocorrência de erros e considerar uma palavra diferente como sendo um erro ortográfico de outra palavra.

4.4 Criação das Instâncias da Ontologia

Nesta etapa, desenvolveu-se inicialmente uma ontologia que permitisse servir de base para estruturar e organizar os conceitos no domínio da hotelaria. Para além de organizar e dar uma visão geral e abrangente dos conceitos envolvidos também foi desenvolvida com o objectivo de facilitar a futura instanciação de entidades de acordo com os elementos retirados das descrições dos hotéis.

4.4.1 Ontologia Criada

Como foi referido no Capítulo 2 dos conceitos envolvidos, uma ontologia é um modelo de dados que representa um conjunto de conceitos e seus relacionamentos dentro de um domínio.

Os objectos criados nesta ontologia serão então instanciados com os atributos retirados do texto, e o produto final deste trabalho, as descrições normalizadas serão construídas de acordo com as instâncias criadas a partir das descrições originais dos hotéis.

A ontologia desenvolvida tem uma forma hierárquica em árvore em que cada classe pode ter sub-classes que tomam o papel de especializações da classe mãe. A estrutura da ontologia criada pode ser visualizada na Figura 4.4 e na Figura 4.5.

Esta ontologia não serviu apenas para modelar teoricamente o domínio da hotelaria, mas também para armazenar o conjunto de expressões regulares e expressões de Levenshtein que permitem identificar os elementos para instanciar os objectos presentes na ontologia.

Tal decisão de colocar as expressões regulares e de Levenshtein na ontologia deveu-se ao facto de tornar independente a ontologia e as respectivas expressões regulares e de Levenshtein do resto do projecto. Desta forma é possível usar outra ontologia ou outro domínio completamente diferente mantendo o resto do projecto inalterado.

A ferramenta usada para criar e desenvolver a ontologia foi o Protegé, e dentro dos vários dialectos OWL usou-se o OWL DL, descartando-se o OWL Full por ser mais complexo e conter uma expressividade semântica muito superior à pretendida (e consequentemente criar ficheiros OWL mais "pesados"e complexos). E o OWL Lite devido à sua pouca expressividade não ser suficiente para todas as relações existentes na ontologia criada.

Figura 4.4: Primeira parte da Ontologia criada no domínio dos hotéis.

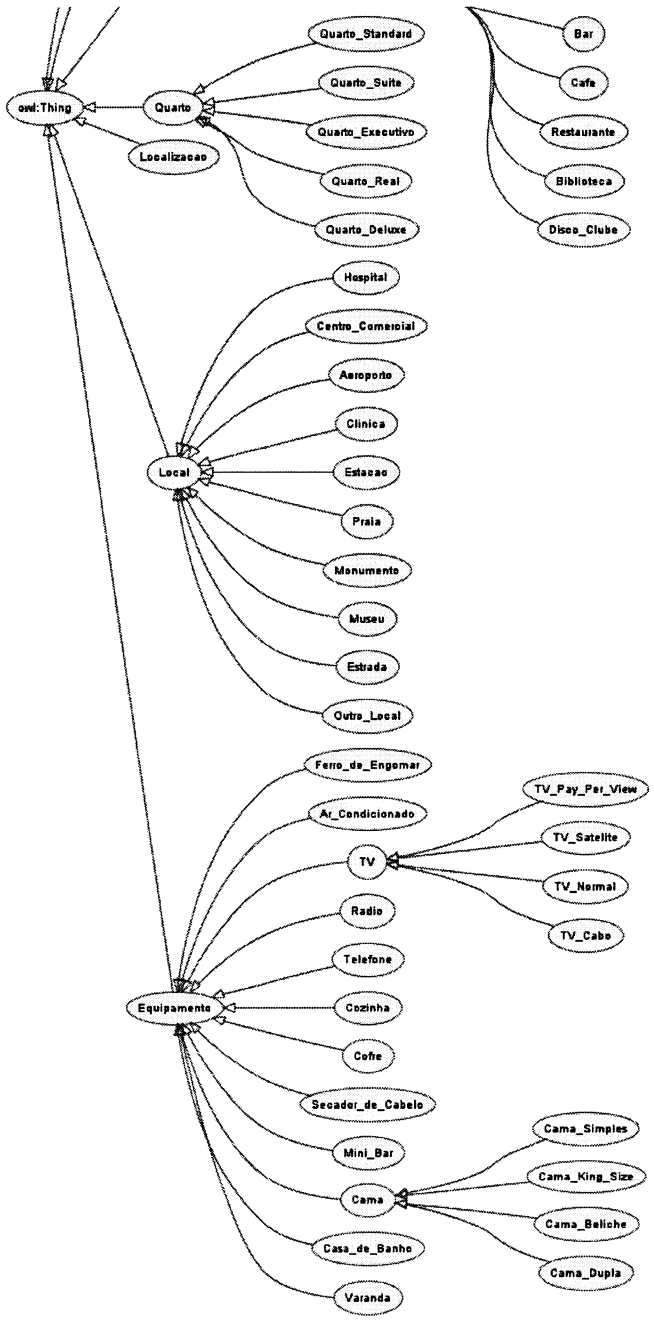

Figura 4.5: Segunda parte da Ontologia criada no domínio dos hotéis.

4.4.2 Instanciação dos Objectos da Ontologia

Para criar um conjunto de objectos computáveis e possíveis de tratar automaticamente para permitir a criação de descrições normalizadas, são utilizadas as frases classificadas por tipo e as expressões regulares e de Levenshtein presentes na ontologia.

Tanto as frases descritivas dos hotéis como as expressões regulares e de Levenshtein são cruzadas com o objectivo de identificar a ocorrência dos objectos ontológicos, para assim criar as instâncias dos objectos presentes na ontologia.

No final desta etapa, é criado outro ficheiro OWL diferente daquele que define a estrutura da ontologia. Este novo ficheiro contém as instâncias dos objectos ontológicos que ocorreram durante a análise das descrições de hotéis. Após esta fase do projecto é já possível observar os objectos/entidades extraídas através das anteriores etapas.

A Tabela 4.5 apresenta um exemplo do OWL gerado com as instâncias ontológicas extraídas da descrição de um hotel. As entidades com sufixo "ExpReg"foram obtidas através de expressões regulares que são aplicáveis aos termos correctamente escritos ou com falhas de diacríticos. As entidades com sufixo "Levenshtein"foram obtidas através das expressões Levenshtein que são aplicáveis aos termos com erros ortográficos.

```
...
<hotelonto:Mini_Bar rdf:ID="Mini_Bar_0_ExpReg"/>
<hotelonto:Telefone rdf:ID="Telefone_1_ExpReg"/>
<hotelonto:Casa_de_Banho rdf:ID="Casa_de_Banho_0_Levenshtein"/>
<hotelonto:Ar_Condicionado rdf:ID="Ar_Condicionado_1_Levenshtein"/>
<hotelonto:Cofre rdf:ID="Cofre_2_Levenshtein"/>
<hotelonto:Secador_de_Cabelo rdf:ID="Secador_de_Cabelo_3_Levenshtein"/>
<hotelonto:Varanda rdf:ID="Varanda_4_Levenshtein"/>
<hotelonto:ADSL rdf:ID="ADSL_2_ExpReg"/>
<hotelonto:Cafe rdf:ID="Cafe_1_ExpReg"/>
<hotelonto:Parque_Infantil rdf:ID="Parque_Infantil_3_ExpReg"/>
<hotelonto:Piscina_Olimpica rdf:ID="Piscina_Olimpica_1_Levenshtein"/>
<hotelonto:Restaurante rdf:ID="Restaurante_0_ExpReg"/>
<hotelonto:Sauna rdf:ID="Sauna_0_Levenshtein"/>
...
```

Tabela 4.5: Instâncias ontológicas obtidas de uma descrição de hotel

Sendo o OWL com instâncias ontológicas uma forma de representar os dados obtidos computacionalmente e facilmente processável por uma aplicação informática, não é propriamente "agradável"de ler ao ser humano.

Estes elementos normalizados são depois transformados em linguagem natural já normalizada.

4.5 Criação de Descrições Textuais Normalizadas

Nesta etapa, pretendia-se construir o resultado final de todo o projecto num formato atraente e de fácil leitura pelo ser humano e de uma forma completamente normalizada. Esta funcionalidade não acrescenta atributos das descrições originais recolhidos nas fases anteriores, apenas monta o aspecto final da descrição que será apresentado ao utilizador final.

Para tal tarefa foi criado um conjunto de modelos de texto[5] auxiliares em XML, que serão esqueletos da representação dos dados textuais uniformizados.

Os elementos do modelo, estão associados aos diversos objectos da ontologia, e só serão inseridos na descrição final caso tenha havido a recolha de instâncias referentes a esses objectos. Assim, as descrições finais serão sempre subconjuntos do conteúdo presente nos modelos.

O funcionamento do tradutor ontológico passa por percorrer o modelo e enviar partes dele para a descrição final, conforme as instâncias ontológicas disponíveis. Por exemplo, tendo a instância ontológica:

$$< hotelonto : Mini\_Barrdf : ID = "Mini\_Bar\_0" / >$$

O modelo ao achar tal instância fará a sua tradução para algo do tipo:

"O quarto está equipado com mini-bar."

Obtendo assim no final uma descrição padronizada dos atributos do hotel. O funcionamento conceptual deste passo pode ser visualizado na Figura 4.6.

Nem todos os objectos instanciados e identificados nas descrições têm de ser obrigatoriamente representados pelo modelo, pois em certas situações é útil ocultar as referências a determinados atributos do hotel.

Por exemplo, faz sentido o modelo para o contexto empresarial e de negócios omitir que o hotel tem serviço de *baby-sitter*; por outro lado o modelo de lazer pode ocultar que o hotel tem sala de reuniões. Assim os modelos não servem apenas para formatar os conteúdos mas também para decidir que conteúdos se adequam mais a cada objectivo.

Para testes, foram construídos três modelos distintos, mas é possível criar outros modelos de acordo com o fim pretendido ou com o modo que se queira articular a informação. Os três modelos criados seguem objectivos distintos, um destina-se ao

[5]Do Inglês, *templates*.

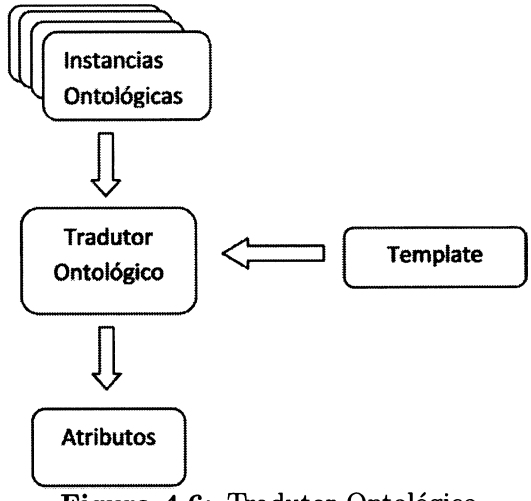

Figura 4.6: Tradutor Ontológico

mercado familiar e de lazer (Figura 4.6) , outro ao mercado empresarial (Figura 4.7), e o último gera descrições normais mas em inglês (Figura 4.8).

Na realização do tradutor ontológico recorreu-se ao Jena Semantic Web Framework [3] para facilita os acessos às estruturas OWL. Pois esta plataforma dispõem uma complexa API com métodos especializados para trabalhar com objectos e instâncias ontológicas.

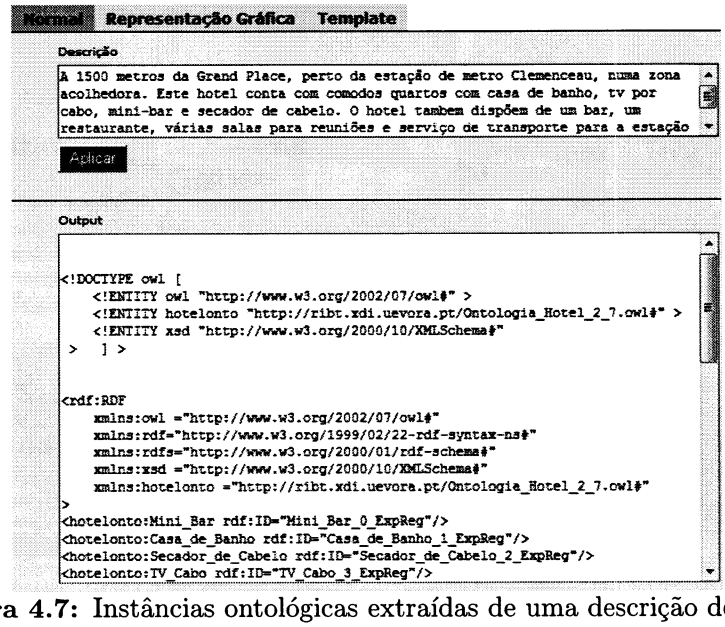

Figura 4.7: Instâncias ontológicas extraídas de uma descrição de Hotel

Para além de modelos que criam exclusivamente resultados textuais em língua natural também é possível criar um output gráfico directamente a partir das instâncias criadas. Esta versão gráfica apresenta logotipos conforme os equipamentos e serviços que o hotel disponibiliza [35]. Na Figura 4.7 é possível ver o OWL em bruto das instâncias ontológicas geradas. A Figura 4.8 mostra o resultado do processamento das instâncias

```
<Enumeration pre="Nas instalações do nosso Hotel também pode usufruir"
 post="" break="false">
    <Item type="Restaurante">
        <Prefix>do nosso</Prefix>
        <Value>restaurante</Value>
        <Suffix>com um conjunto gastronómico variado</Suffix>
    </Item>

    <Item type="Parque_Infantil">
        <Prefix>do nosso</Prefix>
        <Value>parque infantil</Value>
        <Suffix>onde as suas crianças poderão encontrar toda
 a diversão</Suffix>
    </Item>

    <Item type="Garagem">
        <Prefix>da nossa</Prefix>
        <Value>garagem</Value>
        <Suffix>coberta</Suffix>
    </Item>
```

Tabela 4.6: Extracto do modelo de lazer

```
<Enumeration pre="Nas instalações do nosso Hotel também pode usufruir"
 post="" break="false">

<Item type="Sala_de_Conferencias">
<Prefix>de uma</Prefix>
<Value>sala de reuniões</Value>
<Suffix>com toda a privacidade</Suffix>
</Item>

<Item type="Restaurante">
<Prefix>do nosso</Prefix>
<Value>restaurante</Value>
<Suffix>sempre disponivel de acordo com a sua disponibilidade</Suffix>
</Item>

<Item type="Lavandaria">
<Prefix>da</Prefix>
<Value>lavandaria</Value>
<Suffix> onde pode deixar a sua roupa para ser lavada e
 engomada</Suffix>
</Item>
</Enumeration>
```

Tabela 4.7: Extracto do modelo empresarial

```
<Enumeration pre="You may also enjoy" post="" break="false">

<Item type="Restaurante">
<Prefix>our</Prefix>
<Value>restaurant</Value>
<Suffix>with a diverse set of menus</Suffix>
</Item>

<Item type="Parque_Infantil">
<Prefix>leave your children on the</Prefix>
<Value>playground area</Value>
<Suffix>for their amusement</Suffix>
</Item>
</Enumeration>
```

Tabela 4.8: Extracto do modelo em inglês

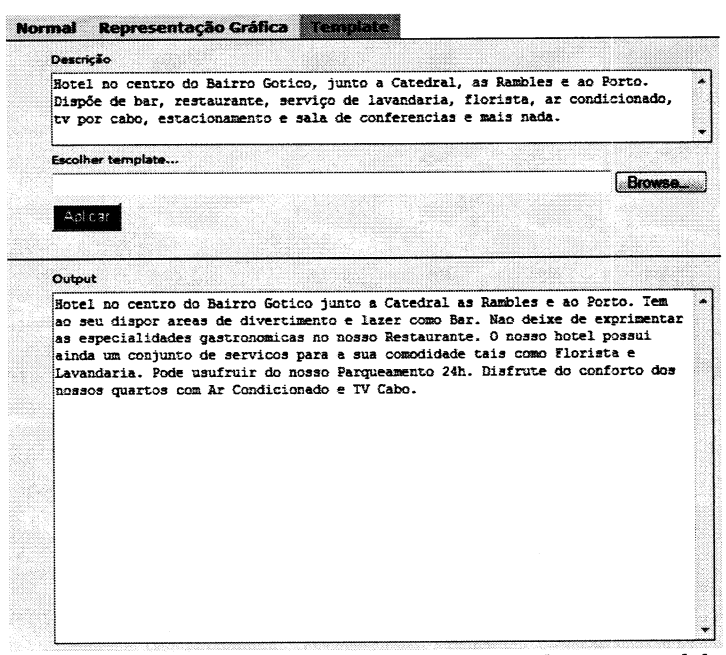

Figura 4.8: Output normalizado a partir de uma modelo

ontológicas com um modelo, originando uma descrição textual normalizada. Finalmente a Figura 4.9 apresenta o resultado do processamento das instâncias ontológicas para uma descrição gráfica com logotipos dos equipamentos e serviços disponibilizados pelo hotel.

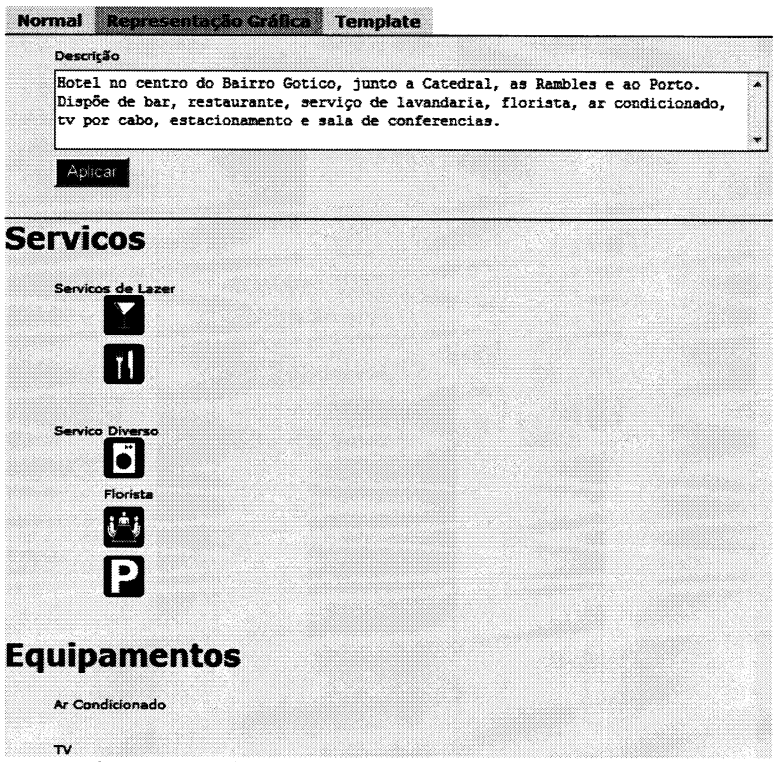

Figura 4.9: Output gráfico normalizado a partir de uma descrição de hotel

4.6 Criação de Base de Dados de Atributos de Hotéis

Foi criado um módulo que permite interligar o trabalho realizado a bases de dados com descrições de hotéis e processar sequencialmente todas as descrições de modo a extrair os seus atributos descritivos e agrupá-los por tipo. Tal processamento não utiliza qualquer modelo de texto pois tem apenas o objectivo de popular tabelas da base de dados com os atributos de cada descrição. Esses atributos são as instâncias ontológicas criadas a partir das descrições. Deste modo, a base de dados fica enriquecida, pois em vez de ter apenas uma descrição textual associada a um hotel, passa a ter também os atributos discriminados e agrupados por tipo (Equipamento, Localização, Serviços) associados ao hotel. Permite ainda efectuar pesquisas de hotéis tendo em conta os seus atributos (ex. pesquisar hotéis com lavandaria). Com estas funcionalidades é possível que futuras aplicações utilizem a base de dados com valor acrescentado.

Este módulo foi testado sobre uma base de dados da VIATECLA, onde se encontravam presentes 20 mil descrições de hotéis. Após o processamento, foram extraídos aproximadamente 155 mil atributos, o que faz uma média de 8 atributos por descrição.

Capítulo 5

Avaliação de Resultados

Este capítulo apresenta os resultados obtidos durante as fases experimentais e resultados finais do sistema. Nas fases experimentais foram obtidos resultados que serviram de sustentação às diversas escolhas efectuadas sobre bases representativas e algoritmos de classificação. Já nos resultados finais do sistema, são apresentados os valores quantitativos de diversas métricas que permitem avaliar a sua eficácia, e são apresentados resultados qualitativos sobre o produto final do sistema, as descrições normalizadas.

5.1 Bases Representativas e Algoritmos de Aprendizagem

Como já foi referido, para a fase de classificação automática foi necessário escolher a base representativa e o algoritmo que apresentassem melhores resultados. Cada uma das decisões pode ser vista como uma variável que se pretende maximizar de modo a atingir da melhor maneira possível os objectivos pré-determinados. Assim, para a base representativa (as diferentes formas de representar os pesos das palavras nas frases) foram testadas as seguintes alternativas: Representação Binária, Frequência de Palavras Normalizada e TFIDF Normalizado. Para a classificação automática aplicaram-se os seguintes algoritmos: J48 (Arvores de decisão C4.5), NBM (Naive Bayes) e SMO (SVM's). Estes dois problemas distintos, ou se preferirmos, as duas variáveis em jogo não são independentes entre si, e consequentemente não se poderia considerar e optimizar cada uma individualmente, já que a melhor base representativa depende do algoritmo, assim como o inverso também acontece.

No entanto, ainda surge uma terceira variável que também deve ser testada em conjunto com as anteriores, pois o resultado depende desta terceira variável. Esta variável não

tem como objectivo ser optimizada e representa as classes que as frases podem tomar (Equipamento, Localização e Serviços). Funciona como variável de contexto, mas que tem de ser tida em conta pois a melhor combinação de base representativa/algoritmo para uma classe de frases pode já não ser a melhor combinação para as outras classes.

Idealmente por questões de implementação seria mais simples desenvolver uma solução em que a melhor combinação base representativa/algoritmo fosse a mesma para as três classes (Equipamento, Localização, Serviço), no entanto à priori nada garantia que isso pudesse acontecer.

No total tem-se então 27 experiências distintas devido a serem:

$$3Bases \times 3Algoritmos \times 3Classes = 27Resultados$$

Para terminar a apresentação de variáveis entra ainda em jogo a complexidade temporal, pois a escolha das variáveis anteriores pode ter consequências na performance temporal. Esta, embora seja uma variável importante, não será de extrema importância nas decisões tomadas, apenas será tida em conta em situações de empate ou de diferenças mínimas entre as diversas configurações de bases representativas/algoritmos.

Outro factor importante para a avaliação dos resultados, prende-se com a maior prioridade dada à cobertura em relação à precisão. Escolha que foi tomada mesmo que para ter melhor cobertura se tivesse de abdicar de precisão.

Há ainda a acrescentar que os resultados obtidos durante a fase de testes foram efectuados com validação cruzada K=10 e com uma significância de 95%.

5.2 Matrizes de Confusão e Tempo Obtido

Na Tabela 5.1 podem ser observadas as matrizes confusão e os tempos para as 9 experiências de cada classe. Já na Tabela 5.2 é possível ver a micro-média obtida para as três classes e ter uma noção de como se comportam as diversas combinações base representativa/algoritmo em média nas três classes. Na Tabela 5.3 estão apresentados os valores médios dos tempos obtidos de modo a poder-se observar melhor os tempos obtidos para as 3 classes. Como é fácil observar, o tempo varia bastante conforme o algoritmo utilizado, no entanto fixando o algoritmo e mudando a base representativa os resultados temporais não têm uma variação assinalável. A negrito estão assinalados os melhores tempos, de onde facilmente se observa que os melhores tempos pertencem ao algoritmo NBM independentemente da base representativa utilizada.

Equipamento

Base	Alg.	VP	VN	FP	FN	Tempo(s)
Bin	J48	35	271	3	21	1,24
	NBM	54	244	30	2	.05
	SMO	46	271	3	10	2,09
Freq.	J48	35	272	2	21	1,09
	NBM	38	274	0	18	0,05
	SMO	47	271	3	9	2,22
TFIDF	J48	38	272	2	18	1,56
	NBM	42	273	1	14	0,04
	SMO	48	271	3	8	2,42

Localização

Base	Alg.	VP	VN	FP	FN	Tempo(s)
Bin	J48	58	224	10	38	1,59
	NBM	95	209	25	1	0,05
	SMO	78	225	9	18	2,05
Freq.	J48	61	225	9	35	1,69
	NBM	80	229	5	16	0,04
	SMO	79	223	11	17	2,36
TFIDF	J48	72	214	20	22	1,97
	NBM	84	231	3	12	0,04
	SMO	80	224	10	16	2,24

Serviços

Base	Alg.	VP	VN	FP	FN	Tempo(s)
Bin.	J48	45	249	3	33	1,62
	NBM	71	219	33	7	0,07
	SMO	57	242	10	21	2,35
Freq.	J48	51	250	2	27	1,9
	NBM	40	252	0	38	0,04
	SMO	53	244	8	25	2,64
TFIDF	J48	54	249	3	24	2,02
	NBM	36	250	2	42	0,06
	SMO	52	245	7	26	2,74

Tabela 5.1: Matrizes Confusão e Tempos das três classes

Micro-Média					
Base	**Alg.**	**VP**	**VN**	**FP**	**FN**
Bin.	J48	138	744	16	92
	NBM	220	672	88	10
	SMO	181	738	22	49
Freq.	J48	147	747	13	83
	NBM	158	755	5	72
	SMO	179	738	22	51
TFIDF	J48	164	735	25	64
	NBM	162	754	6	68
	SMO	180	740	20	50

Tabela 5.2: Micro-Média das matrizes confusão das três classes

Tempos					
Base	**Alg.**	**Equip.**	**Loc.**	**Serv.**	**Média**
Bin.	J48	1,24	1,59	1,62	1,48
	NBM	**0,05**	**0,05**	**0,07**	**0,06**
	SMO	2,09	2,05	2,35	2,16
Freq.	J48	1,09	1,69	1,9	1,56
	NBM	**0,05**	**0,04**	**0,04**	**0,04**
	SMO	2,22	2,36	2,64	2,41
TFIDF	J48	1,56	1,97	2,02	1,85
	NBM	**0,04**	**0,04**	**0,06**	**0,05**
	SMO	2,42	2,24	2,74	2,47

Tabela 5.3: Tempos de execução dos algoritmos nas diferentes bases representativas

5.3 Cobertura, Precisão e F1

Na Tabela 5.4 podem ser observadas para as diversas experiências as métricas obtidas a partir das matrizes de confusão da Tabela 5.1: Cobertura, Precisão, F1 e o respectivo desvio padrão para cada métrica. É de relembrar que os resultados obtidos foram testados com validação cruzada K=10 e com uma significância de 95%.

5.4 Comparação dos Resultados

A Tabela 5.5 apresenta os valores da Cobertura, Precisão e F1 com as experiências reorganizadas de modo a tornar fácil a comparação das experiências com a configuração da base binária e algoritmo NBM. Esta reorganização foi feita porque era notória na Tabela 5.4 que esta configuração apresentava melhores valores para a cobertura.

As cores indicam quais as experiências com melhores e piores resultados. A cinzento claro indica que essa experiência perde em relação ao NBM em base binária com significância de 0,05, a branco há um empate, e a cinzento escuro a experiência ganha em relação ao NBM em base binária, também com significância de 95%.

Ao interpretar a Tabela 5.5 podemos observar claramente que para as três classes Equipamento, Localização e Serviços, a cobertura da configuração base binária/NBM ganha sempre com mais de 5% de significância em relação a todas as outras. Por sua vez na precisão a configuração base binária/NBM perde sempre com mais de 5% de significância para as outras configurações nas classes de Equipamento e de Serviços. Na classe Localização perde também em todos os casos excepto para o algoritmo J48 em todas as bases. Nestes casos existe um empate. Na análise da medida F1 é possível observar que para as classes Localização e Serviços a configuração base binária/NBM empata a maior parte dos casos e ganha alguns. Ganha às várias bases com o J48 na classe Localização, e ganha na classe dos Serviços às bases de Frequência e a base TFIDF com o algoritmo NBM. Para a classe dos Equipamentos observam-se algumas derrotas e alguns empates. O algoritmo SMO independentemente da base representativa utilizada é melhor que a configuração base binária/NBM. Todos os restantes casos são empates.

Nesta análise final dos resultados obtidos, facilmente se chega a conclusão que a configuração base binária/NBM maximiza a cobertura comparativamente a todas as outras configurações, obtendo-se a garantia que a escolha é a correcta tendo em conta os objectivos iniciais de maximização [36]. Na variável tempo, tendo em conta o observado na Tabela 5.3 é também compatível com os resultados de optimização da cobertura, pois os melhores tempos registados ocorrem no algoritmo NBM.

Equipamento

Base	Alg.	Cob	Std.Dev	Prec	Std.Dev	F1	Std.Dev
Bin.	J48	.724	.221	.961	.082	.802	.165
	NBM	.969	.072	.657	.119	.776	.089
	SMO	.843	.155	.951	.084	.883	.102
Freq.	J48	.689	.224	.954	.090	.775	.165
	NBM	.693	.200	.990	.100	.800	.162
	SMO	.858	.139	.951	.084	.893	.093
TFIDF	J48	.716	.204	.932	.142	.789	.156
	NBM	.749	.193	.974	.114	.835	.149
	SMO	.865	.142	.942	.094	.893	.097

Localização

Base	Alg.	Cob	Std.Dev	Prec	Std.Dev	F1	Std.Dev
Bin.	J48	.601	.142	.863	.129	.696	.116
	NBM	.987	.034	.795	.106	.876	.068
	SMO	.816	.118	.912	.088	.854	.080
Freq.	J48	.608	.142	.858	.137	.699	.117
	NBM	.844	.114	.952	.074	.890	.077
	SMO	.830	.124	.905	.093	.858	.082
TFIDF	J48	.742	.139	.799	.109	.760	.095
	NBM	.868	.109	.969	.057	.911	.070
	SMO	.846	.117	.904	.094	.866	.079

Serviços

Base	Alg.	Cob	Std.Dev	Prec	Std.Dev	F1	Std.Dev
Bin.	J48	.589	.166	.955	.097	.713	.138
	NBM	.919	.092	.703	.112	.792	.089
	SMO	.724	.152	.893	.113	.789	.112
Freq.	J48	.623	.177	.956	.096	.736	.136
	NBM	.510	.171	.988	.102	.658	.159
	SMO	.711	.155	.892	.115	.779	.115
TFIDF	J48	.691	.169	.919	.127	.773	.129
	NBM	.451	.183	.960	.101	.592	.180
	SMO	.700	.153	.891	.108	.771	.106

Tabela 5.4: Métricas de cobertura, Precisão, F1 e respectivos desvios padrão

Equipamento

Base	Alg.	Cob	Prec	F1
Bin.	NBM	**.969**	**.657**	**.776**
	SMO	.843	.951	.883
	J48	.724	.961	.802
Freq.	NBM	.693	.990	.800
	SMO	.858	.951	.893
	J48	.689	.954	.775
TFIDF	NBM	.749	.974	.835
	SMO	.865	.942	.893
	J48	.716	.932	.789

Localização

Base	Alg.	Cob	Prec	F1
Bin.	NBM	**.987**	**.795**	**.876**
	SMO	.816	.912	.854
	J48	.601	.863	.696
Freq.	NBM	.844	.952	.890
	SMO	.830	.905	.858
	J48	.608	.858	.69920
TFIDF	NBM	.868	.969	.911
	SMO	.846	.904	.866
	J48	.742	.799	.760

Serviços

Base	Alg.	Cob	Prec	F1
Bin.	NBM	**.919**	**.703**	**.792**
	SMO	.724	.893	.789
	J48	.589	.955	.713
Freq.	NBM	.510	.988	.658
	SMO	.711	.892	.779
	J48	.623	.956	.736
TFIDF	NBM	.451	.960	.592
	SMO	.700	.891	.771
	J48	.691	.919	.773

Tabela 5.5: Comparação final de resultados em relação ao NBM base binária

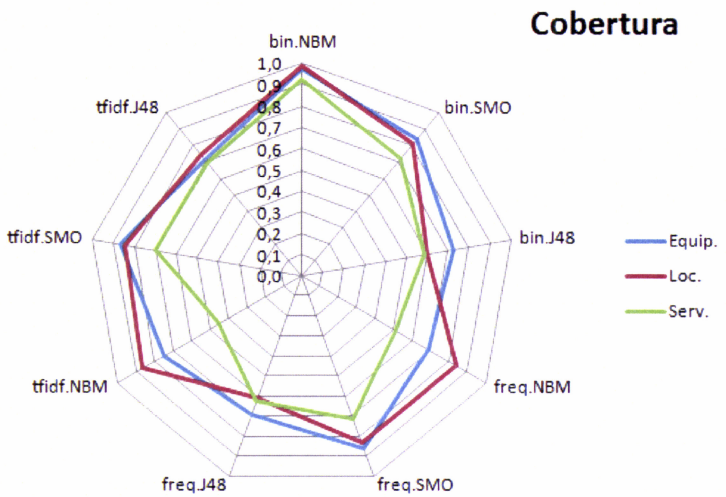

Figura 5.1: Comparação da cobertura nas diferentes classes e configurações

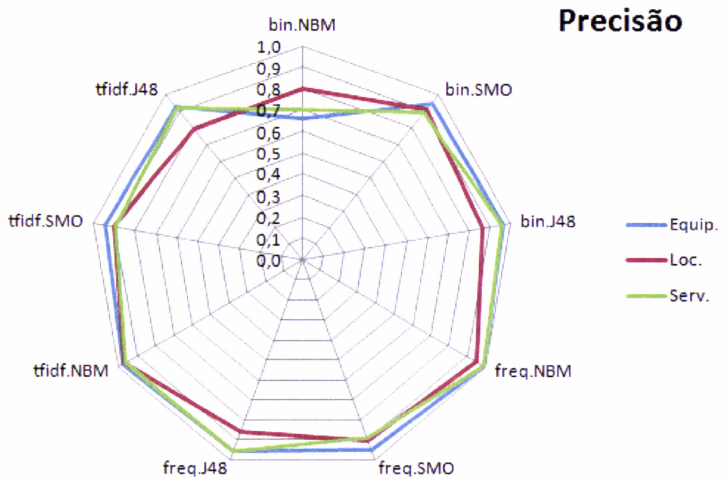

Figura 5.2: Comparação da precisão nas diferentes classes e configurações

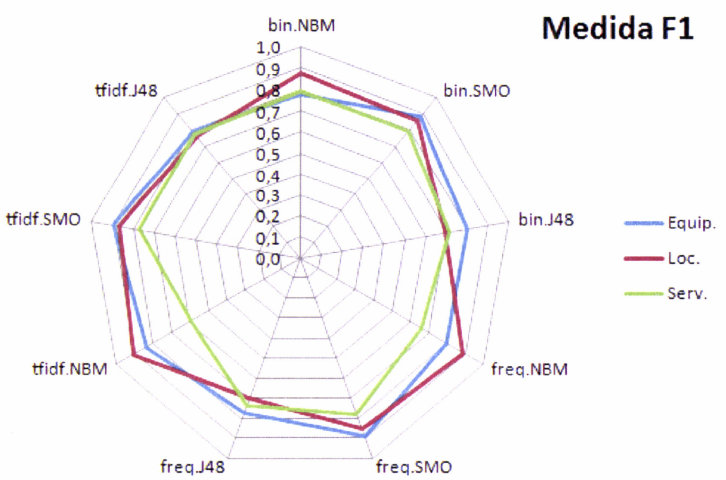

Figura 5.3: Comparação da medida F1 nas diferentes classes e configurações

Outra forma de constatar os mesmos resultados é observando os gráficos das Figuras 5.1, 5.2 e 5.3, onde as diversas configurações bases representativas/algoritmos são agrupadas de acordo com a cobertura, precisão e medida F1, respectivamente. No gráfico da Figura 5.1, facilmente se observa a vantagem que a configuração base binária/Naive Bayes toma em relação às demais, obtendo uma cobertura muito superior em todas as classes (Equipamento, Serviços, Localização). Já no gráfico da Figura 5.2 facilmente se observa que acontece o inverso, sendo a configuração base binária/Naive Bayes a que apresenta piores resultados de precisão em todas as classes. Por fim, o gráfico da Figura 5.3 apresenta os resultados da medida F1 das várias experiências, no entanto tal gráfico torna-se algo inconclusivo, pois obtém valores idênticos para todas as experiências, sendo assim necessário recorrer aos gráficos da precisão e cobertura para uma maior discriminação dos resultados.

A configuração vencedora em termos de precisão foi a que utilizou a base de frequência de termos e o algoritmo Naive Bayes obtendo precisões de 99,0%, 95,1% e 98,8% respectivamente para as classes Equipamento, Localização e Serviços. Esta configuração fica ao nível dos melhores registos temporais obtendo uma média de 0,04 segundos a classificar uma frase. Já na medida F1, a configuração vencedora foi a base de frequência com o algoritmo SMO que obteve os valores de 89,3%, 85,8% e 77,9% respectivamente para as classes Equipamento, Localização e Serviços. Neste caso o registo temporal situa-se entre os piores, pois regista uma média de 2,41 segundos para classificar cada frase. É de referir novamente, que estas configurações vencedoras foram descartadas a favor da configuração que obtém melhores coberturas, base binária/Naive Bayes.

5.5 Normalização Final

Após a fase anterior de optimização da base representativa e algoritmo a utilizar, o sistema ficou apto a funcionar.

Para testar e observar os resultados da normalização das descrições de hotéis com os diferentes modelos foram utilizadas descrições reais de hotéis presentes na plataforma KeyforTravel da VIATECLA [53].

As Tabelas 5.6, 5.7, 5.8 e 5.9 apresentam um exemplo de descrição de hotel (Tabela 5.6) normalizada segundo 3 modelos: Lazer (Tabela 5.7), Empresarial (Tabela 5.8) e Inglês (Tabela 5.9).

É de notar que nos exemplos anteriores, não só a forma da descrição é alterada mas também os atributos apresentados. Pode inclusive ocorrer que elementos chave da descrição inicial não apareçam em nenhuma descrição normalizada. Isto não se deve a

Ex Tivoli Almansor, O Tivoli Carvoeiro situa-se a 60 Km a Oeste do Aeroporto de Faro, na pitoresca aldeia de pescadores na Praia do Carvoeiro. O Hotel foi construído sobre a falésia da Praia Vale Covo, que abriga uma enseada com uma pequena praia, ladeada por rochas e grutas. Possui 293 quartos, equipados com casa de banho completa, Ar condicionado individual, TV satélite, telefone directo ao exterior, mini-bar, secador de cabelo, cofre e ADSL. Os quartos possuem varanda com vista para o mar. O hotel dispõe de um restaurante e um café, uma piscina olimpica, Health Club com Sauna e parque infantil. Para finalizar dispõem ainda de uma sala de reuniões tambem equipada com ADSL.

Tabela 5.6: Descrição original retirada do KeyforTravel

A nivel de localizações o Ex Tivoli Almansor, **O Tivoli Carvoeiro situa-se a 60 Km a Oeste do Aeroporto de Faro, na pitoresca aldeia de pescadores na Praia do Carvoeiro**. Todos os quartos do nosso Hotel possuem uma bela envolvente exterior observada por **varanda privativa**. **Casa de Banho** disponível em qualquer quarto. Todos os quartos encontram-se equipados com **TV Satélite** para seu entretenimento. Nas instalações do nosso Hotel também pode usufruir do nosso **restaurante** com um conjunto gastronómico variado e do nosso **parque infantil** onde as suas crianças poderão encontrar toda a diversão. O Hotel possui ainda uma **piscina Olímpica** para os seus dias de Verão e para os dias de Inverno a sala de **sauna** encontra-se disponível à espera de uma visita sua.

Tabela 5.7: Descrição normalizada pelo modelo de lazer destinada a famílias e turismo

Disponibiliza-se a cada hóspede **cofre** para a salvaguarda de pertences próprios. Cada quarto encontra-se equipado com **ligação ADSL** de alta velocidade e de uso gratuito. Nas instalações do nosso Hotel também pode usufruir do nosso **restaurante** sempre disponível de acordo com a sua disponibilidade. Nas instalações do nosso Hotel também pode usufruir de uma **sala de reuniões** com toda a privacidade. Todos os quartos do nosso Hotel possuem **varanda** privativa e **ar condicionado** para o seu conforto.A nível de localizações o Ex Tivoli Almansor, O Tivoli Carvoeiro **situa-se a 60 Km a Oeste do Aeroporto de Faro, na pitoresca aldeia de pescadores na Praia do Carvoeiro.**

Tabela 5.8: Descrição normalizada pelo modelo de empresarial

All hotel rooms have a private **balcony** over the surrounding landscape and **air conditioning**. **Bathroom** available in all rooms. You may also enjoy our **restaurant** with a diverse set of menus and leave your children on the **playground** area for their amusement. The Hotel also has an **Olympic pool** for sunny days and a **sauna room** for cold winter days.

Tabela 5.9: Descrição normalizada pelo modelo de lazer em Inglês

uma falha ou perca de informação, mas sim porque foi decidido que certos atributos não apareceriam nos modelos por não se enquadrar com a sua orientação. No entanto, nada impede que tais atributos não surjam num outro modelo distinto. Um exemplo desses é o Mini-bar, que é um elemento identificado na descrição original é criada a sua instância ontológica mas não é inserido em nenhum dos modelos.

A Tabela 5.11 apresenta as instâncias da ontologia criada a partir da descrição original (incluindo Mini-bar). Estas instâncias representam todos os objectos identificados antes de serem convertidos em língua natural pelos modelos. Aqui é possível ver todas as entidades sem a censura propositada dos modelos.

```
<hotelonto:Mini_Bar rdf:ID="Mini_Bar_0_ExpReg"/>
<hotelonto:Casa_de_Banho rdf:ID="Casa_de_Banho_1_ExpReg"/>
<hotelonto:Ar_Condicionado rdf:ID="Ar_Condicionado_2_ExpReg"/>
<hotelonto:Cofre rdf:ID="Cofre_3_ExpReg"/>
<hotelonto:Telefone rdf:ID="Telefone_4_ExpReg"/>
<hotelonto:Secador_de_Cabelo rdf:ID="Secador_de_Cabelo_5_ExpReg"/>
<hotelonto:Varanda rdf:ID="Varanda_6_ExpReg"/>
<hotelonto:ADSL rdf:ID="ADSL_3_ExpReg"/>
<hotelonto:Cafe rdf:ID="Cafe_0_ExpReg"/>
<hotelonto:Parque_Infantil rdf:ID="Parque_Infantil_5_ExpReg"/>
<hotelonto:Piscina_Olimpica rdf:ID="Piscina_Olimpica_4_ExpReg"/>
<hotelonto:Restaurante rdf:ID="Restaurante_1_ExpReg"/>
<hotelonto:Sauna rdf:ID="Sauna_2_ExpReg"/>
```

Tabela 5.10: Instâncias ontológicas obtidas da descrição da Tabela 5.6

É de referir ainda que todas as instâncias ontológicas têm o sufixo "ExpReg"que indica o método que extraiu essa instância ontológica. Todos os casos são de expressões regulares porque a descrição original da Tabela 5.6 está escrita sem erros ortográficos. Caso a descrição contivesse erros já entraria em funcionamento o mecanismo das distancias de Levenshtein e as instâncias ontológicas da descrição teriam o sufixo "Levenshtein"como se pode ver no exemplo da Tabela 5.11.

```
<hotelonto:Mini_Bar rdf:ID="Mini_Bar_0_ExpReg"/>
<hotelonto:Telefone rdf:ID="Telefone_1_ExpReg"/>
<hotelonto:Casa_de_Banho rdf:ID="Casa_de_Banho_0_Levenshtein"/>
<hotelonto:Ar_Condicionado rdf:ID="Ar_Condicionado_1_Levenshtein"/>
<hotelonto:Cofre rdf:ID="Cofre_2_Levenshtein"/>
<hotelonto:Secador_de_Cabelo rdf:ID="Secador_de_Cabelo_3_Levenshtein"/>
<hotelonto:Varanda rdf:ID="Varanda_4_Levenshtein"/>
<hotelonto:ADSL rdf:ID="ADSL_2_ExpReg"/>
<hotelonto:Cafe rdf:ID="Cafe_1_ExpReg"/>
<hotelonto:Parque_Infantil rdf:ID="Parque_Infantil_3_ExpReg"/>
<hotelonto:Piscina_Olimpica rdf:ID="Piscina_Olimpica_1_Levenshtein"/>
<hotelonto:Restaurante rdf:ID="Restaurante_0_ExpReg"/>
<hotelonto:Sauna rdf:ID="Sauna_0_Levenshtein"/>
```

Tabela 5.11: Instâncias ontológicas obtidas de uma descrição com erros ortográficos

Capítulo 6

Conclusões e Trabalho Futuro

Ao longo da dissertação apresentaram-se as motivações e os objectivos que guiaram o desenvolvimento deste trabalho, descreveram-se os conceitos, metodologias e ferramentas utilizadas para o seu desenvolvimento bem como trabalhos relacionados, propôs-se um sistema para normalização de descrições de hotéis e fez-se a sua avaliação (quer a nível de desempenho do sistema de classificação de frases, quer verificando o resultado com um exemplo prático). Neste trabalho:

- efectuou-se um estudo e levantamento de diversos de algoritmos de classificação automática. Foi feito um levantamento das principais características de cada um e uma análise de outros trabalhos que experimentaram e puseram à prova esses algoritmos. Inicialmente foi efectuado um leve levantamento de variados algoritmos, mas à medida da progressão do estudo, a análise focou-se em três deles: Árvore de Decisão, Naïve de Bayes e Máquinas de Vectores de Suporte;

- tendo em conta que os textos são apresentados aos algoritmos de aprendizagem utilizando uma representação saco de palavras, também foi efectuado um estudo das diversas medidas normalmente utilizadas (aqui designadas como bases representativas): binária, contagem de termos, frequência de termos e TFIDF juntamente com diversas formas de normalização dessas medidas;

- fez-se um levantamento das diversas formas de representar conhecimento recorrendo a ontologias. Analisaram-se as suas funções, aplicações e elementos constituintes, as principais linguagens e dialectos disponíveis para as representar e ainda as principais ferramentas que poderiam ser úteis nesta área durante a realização do trabalho. Escolheu-se a linguagem OWL DL e a ferramenta Protegé;

- estudaram-se formas de extracção de informação a partir de bases textuais. Tendo em conta o domínio deste trabalho (hotelaria) foi utilizado um sistema que utiliza

duas técnicas distintas: para casos normais a correspondência de padrões com expressões regulares, para casos com erros ortográficos, a utilização de distâncias de Levenshtein.

6.1 Conclusões

Após a fase mais teórica de levantamento e estudo dos algoritmos de classificação automática e bases representativas mais adequados, efectuaram-se testes para determinar as melhores combinações. Utilizando uma validação cruzada com 10 pastas compararam-se resultados, métricas e performances e escolheu-se o algoritmo Naïve de Bayes com base binária, uma vez que esta configuração apresentou as melhores coberturas (a medida considerada mais importante) e esteve também entre as mais rápidas.

O algoritmo de aprendizagem foi aplicado com a finalidade de classificar automaticamente as frases presentes nas descrições dos hotéis relativamente a Equipamento, Serviços, Localização. Os resultados obtidos, com coberturas de 96,8%, 98,6% e 91,9% (para as classes de Equipamento, Localização e Serviços, respectivamente) mostram que esta estratégia se mostrou adequada [36]. As outras combinações embora com melhores valores de precisão, tiveram valores de cobertura muito inferiores (ver Tabela 5.5). Para a mesma configuração eleita, a nível de precisão os valores obtidos foram de 65,7%, 79,5% e 70,3%, já os valores da medida F1 foram 77,6%, 87,6% e 79,2% respectivamente para as classes as classes de Equipamento, Localização e Serviços.

Por outro lado, o trabalho permitiu construir uma estrutura ontológica para representar os conceitos e as relações inter-conceitos no âmbito da hotelaria. A ontologia serviu também para alojar nos atributos das classes a informação necessária à extracção e consequente instanciação de objectos das classes.

Recorrendo à informação presente na ontologia, foi possível extrair os elementos relevantes (atributos descritivos do hotel) das frases previamente classificadas. A extracção é realizada com o auxílio de duas técnicas distintas: expressões regulares e distâncias de Levenshtein (para lidar com erros ortográficos). A partir do exemplo apresentado (ver Tabela 5.10) é possível concluir que este mecanismo consegue extrair todas as entidades existentes na descrição original. Com os elementos relevantes, é possível enriquecer bases de dados que possuam simples descrições de hotéis em forma textual, sendo possível criar novas tabelas discriminantes dos diversos atributos de cada descrição de hotel e assim permitir a realização de novos serviços de pesquisas de hotéis pelos seus atributos (ex. pesquisar hotéis com lavandaria) [37].

Finalmente, e com o objectivo de criar automaticamente descrições padronizadas e

que possam ter "orientações"específicas de acordo com o mercado alvo, criaram-se um conjunto de modelos que servem de esqueleto à construção da descrição textual normalizada [35]. Estes modelos, juntamente com as instâncias da ontologia criadas a partir de uma descrição de hotel, permitem a criação de novos textos descritivos normalizados e formatados de acordo com o mercado alvo a que se destinam (ver Tabelas 5.7, 5.8 e 5.9).

6.2 Trabalho Futuro

Existem diversas áreas do trabalho desenvolvido susceptíveis de trabalho futuro, quer a nível do aperfeiçoamento dos resultados, quer a nível de futuras extensões.

Em primeiro lugar pretende-se diminuir o tempo de normalização de uma descrição, que é relativamente elevado (entre os 6 e 10 segundos). Este problema deve-se ao protótipo ser constituído por diversos módulos independentes com diferentes linguagens de programação. Este facto implica um processo contínuo de escrita e leitura de dados em disco, para a informação transitar entre módulos. Se a aplicação for desenvolvida num único módulo haverá estruturas em memória ao longo de todo o processamento evitando assim o tempo necessário de escrita e leitura dos resultados intermédios.

Outra questão passível de optimização futura prende-se com a utilização da ferramenta WEKA para efectuar a classificação das frases. Como plataforma de testes e comparação entre algoritmos é uma ferramenta muito completa e eficaz; no entanto, para uma aplicação final não é a mais indicada devido aos seus longos tempos de processamento (carregamento das bibliotecas em memória sempre que é executado um algoritmo). Pretende-se assim substituir a WEKA por uma implementação dedicada do algoritmo de classificação seleccionado pela fases de testes: o Naïve de Bayes.

A mais longo prazo pretende-se criar um sistema multi-língua que receba as descrições noutras línguas que não a língua Portuguesa. Para tal será necessário criar, para cada uma das línguas suportadas, um modelo de classificação de frases e as expressões regulares para extrair a informação relevante (além do dicionário). Com esta funcionalidade a base de dados de hotéis deixará de estar dependente de traduções e poderá "alimentar-se"de descrições feitas noutras línguas.

Finalmente, pretende-se ainda alargar este tipo de normalização a outros produtos turísticos que sofrem das mesmas , tais como aluguer de viaturas e pacotes de férias, de modo a construir uma aplicação completa de turismo que utiliza descrições normalizadas e que permite fazer pesquisas sobre os seus produtos de forma estruturada.

Referências

[1] Stephan Bloehdorn and Andreas Hotho. 2006): Learning ontologies to improve text clustering and classification. In *Proceedings of the 29th Annual Conference of the German Classification Society (GfKl.* Springer, 2005.

[2] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[3] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Andy Seaborne, Kevin Wilkinson, Dave Reynolds, and Dave Reynolds. Jena: Implementing the semantic web recommendations. pages 74–83, 2003.

[4] Vinay K. Chaudhri and Richard Fikes. Okbc: A programmatic foundation for knowledge base interoperability. pages 600–607. AAAI Press, 1998.

[5] RDF: World Wide Web Consortium. Rdf, Consultado em Fevereiro 2010. http://www.w3.org/RDF/.

[6] World Wide Web Consortium. Js, Consultado em Fevereiro 2010. http://www.w3.org/standards/techs/jsw3call.

[7] World Wide Web Consortium. Soap, Consultado em Fevereiro 2010. http://www.w3.org/TR/soap/.

[8] World Wide Web Consortium. Xslt, Consultado em Fevereiro 2010. http://www.w3.org/TR/xslt.

[9] World Wide Web Consortium, Consultado em Março 2010. http://www.w3.org/.

[10] João Paulo da Costa Cordeiro. Extracção de elementos relevantes em texto/páginas da world wide web. Master's thesis, Departamento de Ciência de Computadores Faculdade de Ciências da Universidade do Porto, Julho 2003.

[11] Susana Afonso Eckhard, Eckhard Bick, Renato Haber, and Diana Santos. F!oresta sintfi(c)tica": A treebank for portuguese, 2002.

[12] Shih-Hung Wu et al. Text categorization using automatically acquired domain ontology. Sapporo, JP, 2003.

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

[14] Soderland S. G. Learnning domain-specific text analysis rules. University of Massachusetts at Amherst, 1996.

[15] Nettleton D. J. Garigliano R., Urbanowicz A. Description of the lolita system, as used in muc-7. University of Durham, 1998.

[16] T. Gonçalves and P. Quaresma. A preliminary approach to the multilabel classification problem of Portuguese juridical documents. In F. Moura-Pires and S. Abreu, editors, *EPIA-03, 11th Portuguese Conference on Artificial Intelligence*, LNAI 2902, pages 435–444, Évora, PT, December 2003. Springer-Verlag.

[17] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update, 2004.

[18] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Explanation of owl entailments in protégé 4.

[19] David A. Hull. Stemming algorithms - a case study for detailed evaluation. *Journal of the American Society for Information Science*, 47:70–84, 1996.

[20] Infixowl An Idiomatic. Interface for owl.

[21] T. Joachims. *Learning to Classify Text Using Support Vector Machines.* Kluwer Academic Publishers, 2002.

[22] Thorsten Joachims, Fachbereich Informatik, Fachbereich Informatik, Fachbereich Informatik, Fachbereich Informatik, and Lehrstuhl Viii. Text categorization with support vector machines: Learning with many relevant features, 1997.

[23] George John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann, 1995.

[24] S.S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to platt's smo algorithm for svm classifier design, 1999.

[25] Wessel Kraaij and Ren Ee Pohlmann. Viewing stemming as recall enhancement. In *In Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 40–48, 1996.

[26] Constantino M. Financial information extraction using pre-defined and user-definable templates in the lolita system. University of Durham, 1997.

[27] Constantino M. The lolita user-definable template interface. University of Durham, 2001.

[28] A Macfarlane. Pliers and snowball at clef 2002. In *In Working Notes for the CLEF 2002 Workshop*, 2002.

[29] Michael Martin, Jörg Unbehauen, and Sören Auer. Improving the performance of semantic web applications with sparql query result caching.

[30] Microsoft, Consultado em Janeiro 2010. http://msdn.microsoft.com/en-us/netframework/aa497273.aspx.

[31] Microsoft, Consultado em Janeiro 2010. http://www.microsoft.com/windowsphone/en-us/default.aspx.

[32] Microsoft, Consultado em Janeiro 2010. http://www.microsoft.com/windowsembedded/en-us/products/windowsce/default.mspx.

[33] Microsoft, Consultado em Janeiro 2010. http://msdn.microsoft.com/en-us/library/ms754130.aspx.

[34] Microsoft, Consultado em Janeiro 2010. http://msdn.microsoft.com/en-us/netframework/aa663324.aspx.

[35] N. Miranda, R. Raminhos, P. Seabra, J. Saias, T. Gonçalves, and P. Quaresma. Automatic knowledge extraction from tourism textual descriptions applied to e-marketing and product promotion. In *ICTD-09 - International Conference on Tourism Development and Management*, Kos Island, GR, September 2009.

[36] N. Miranda, R. Raminhos, P. Seabra, J. Saias, T. Gonçalves, and P. Quaresma. Standardisation of hotel descriptions. In *EPIA-09 - Encontro Português de Inteligência Artificial*, Aveiro, PT, October 2009. Springer-Verlag.

[37] N. Miranda, R. Raminhos, P. Seabra, J. Saias, T. Gonçalves, and P. Quaresma. Information extraction in tourism domain. In (submitted), editor, *ECIR'11: European Conference on Information Retrieval*, April 2011.

[38] Massachusetts Institute of Technology, Consultado em Março 2010. http://groups.csail.mit.edu/medg/people/wam.html.

[39] Symposium Nlprs Pages and Akiko Aizawa. Akiko aizawa linguistic techniques to improve the performance of automatic text categorization. In *In Proceedings 6th NLP Pac. Rim Symp. NLPRS-01*, pages 307–314, 2001.

[40] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of sparql. In *Proc. 5th International Semantic Web Conference (ISWC06*, pages 30–43. CommunityProjects/LinkingOpenData, 2006.

[41] J. R. Quinlan. Induction of decision trees. *Mach. Learn*, pages 81–106, 1986.

[42] J. R. Quinlan. Bagging, boosting, and c4.5. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. AAAI Press, 1996.

[43] R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[44] Krupka G. R. Description of the sra system as used for muc-6. 221-235. San Mateo: Morgan Kaufmann, 1995. In Proceedings of the Sixth Message Understanding Conference (MUC-6).

[45] RULEQUEST RESEARCH, Consultado em Março 2010. http://www.rulequest.com/see5-comparison.html.

[46] Ellen Riloff. Automatically constructing a dictionary for information extraction tasks. *National Conference on Artificial Intelligence*, 1993.

[47] Ellen Riloff. An empirical study of automated dictionary construction for information extraction in three domains. *Artificial Intelligence*, 85:101–134, 1996.

[48] Klaus Schulz and Stoyan Mihov. Fast string correction with levenshtein-automata. *INTERNATIONAL JOURNAL OF DOCUMENT ANALYSIS AND RECOGNITION*, 5:67–85, 2002.

[49] Ken Schwaber. Scrum development process. In *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA*, pages 117–134, 1995.

[50] C. Silva and R. Vieira. Uso de informações lingüísticas em categorização de textos utilizando redes neurais artificiais. In *SBNR-04, 8º Simpósio Brasileiro de Redes Neurais*, pages 1–16, Rio de Janeiro, BR, 2004.

[51] C. Silva and R. Vieira. Categorização de textos da língua Portuguesa com árvores de decisão, SVM e informações linguísticas. In *TIL-07, 5º workshop em Tecnologia da Informação e da Linguagem Humana*, pages 1650–1658, Rio de Janeiro, BR, July 2007.

[52] V. Vapnik and A. Chervonenkis. A note on one class of perceptrons. *Automation and Remote Control*, 25, 1964.

[53] Viatecla, Consultado em Janeiro 2010. www.viatecla.com.

[54] Viatecla, Consultado em Janeiro 2010. www.viatecla.com/keyfortravel.

[55] W3C. Css, Consultado em Fevereiro 2010. http://www.w3.org/Style/CSS/.

[56] W3C. Ntriples, Consultado em Fevereiro 2010. http://www.w3.org/2001/sw/RDFCore/ntriples/.

[57] W3C. Xhtml, Consultado em Fevereiro 2010. http://www.w3.org/TR/xhtml1/.

[58] W3C. Xml, Consultado em Fevereiro 2010. http://www.w3.org/XML/.

[59] W3C World Wide Web Consortium. Html, Consultado em Fevereiro 2010. http://www.w3.org/TR/html4/.

[60] W3C World Wide Web Consortium. N3, Consultado em Fevereiro 2010. http://www.w3.org/DesignIssues/Notation3.

[61] Michael Wurst. The word vector tool user guide operator reference developer tutorial, 2007.