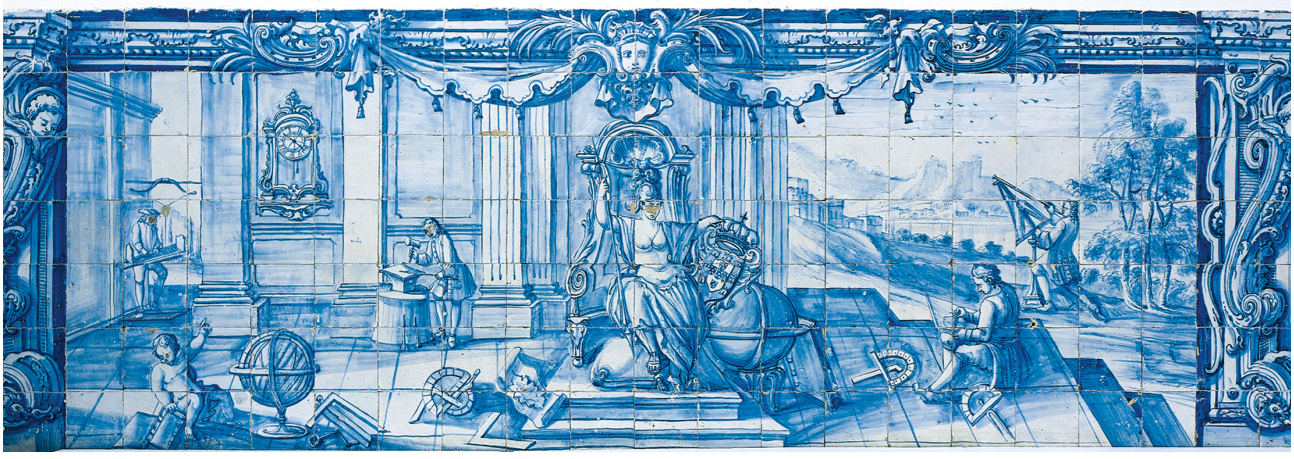
Marcação das Partes do Discurso Usando Computação Evolucionária

Ana Paula Neves Ferreira da Silva

Tese apresentada à Universidade de Évora
para obtenção do Grau de Doutor em Informática

ORIENTADORA: *Irene Pimenta Rodrigues*

ÉVORA, Julho de 2013



INSTITUTO DE INVESTIGAÇÃO E FORMAÇÃO AVANÇADA

Marcação das Partes do Discurso Usando Computação Evolucionária

Tese apresentada à
Universidade de Évora
para obtenção do Grau de Doutor
em Informática

por

Ana Paula Neves Ferreira da Silva

Universidade de Évora
Escola de Ciências e Tecnologia
Departamento de Informática

Julho de 2013

Marcação das Partes do Discurso Usando Computação Evolucionária

Esta dissertação foi elaborada
sob a supervisão de

Irene Pimenta Rodrigues
Professora Associada
do Departamento de Informática
da Escola de Ciências e Tecnologia
da Universidade de Évora

Para as minhas filhas Beatriz e Catarina

Agradecimentos

Gostaria de agradecer às pessoas que tornaram possível este trabalho:

Em primeiro lugar gostaria de agradecer à Professora Doutora Irene Rodrigues, pela sua orientação, incentivo e sobretudo por compreender as condições nem sempre favoráveis em que decorreu o trabalho.

Agradeço aos docentes do Departamento de Informática da Universidade de Évora a forma sempre afável com que me receberam, e as sugestões de melhoria oferecidas nas apresentações públicas das fases intermédias do trabalho.

Queria também agradecer a todos os colegas da Unidade Técnico-Científica de Informática, da Escola Superior de Tecnologia de Castelo Branco, em particular aos colegas da área de Programação, Algoritmos e Desenvolvimento de Software, o espirito de companheirismo e entreajuda sempre demonstrado.

A um conjunto de pessoas que de forma indirecta, mas não menos importante, contribuíram para a concretização deste trabalho: à minha família, em especial à minha Mãe e ao meu Pai, a todos os meus amigos, em particular à família Cotrim e à família Azeiteiro, os meus sinceros agradecimentos.

Por fim, queria agradecer àqueles que me acompanharam e apoiaram diariamente ao longo desta jornada, ao Arlindo, à Beatriz e à Catarina, um especial e profundo agradecimento.

Ana Paula Silva

Castelo Branco, Março 2013

Apoio Financeiro

Quero agradecer ao Instituto Politécnico de Castelo Branco o apoio concedido a este trabalho, que se traduziu não só pelo financiamento das propinas, como pelo ajustamento das condições de trabalho.

Gostaria também de agradecer ao centro de investigação CENTRIA, pelo apoio concedido ao nível da participação nas conferências onde foram apresentadas algumas das publicações que resultaram deste trabalho.

Ana Paula Silva

Castelo Branco, Março 2013

Marcação das Partes do Discurso Usando Computação Evolucionária

Resumo

A marcação das partes do discurso constitui uma tarefa de considerável importância na área de processamento de língua natural. O seu objectivo consiste em marcar automaticamente as palavras de um texto com etiquetas que designam as partes do discurso adequadas.

A abordagem proposta nesta tese divide o problema em duas tarefas: uma de aprendizagem e outra de optimização. Foram adoptados algoritmos da área da computação evolucionária em cada uma das fases. Destacamos a utilização de inteligência de enxame, não só pelos bons resultados alcançados, mas também por se revelar uma das primeiras aplicações deste tipo de algoritmos a este problema.

A abordagem foi pensada com o objectivo de poder ser alargada a outras tarefas de processamento de língua natural, com características comuns à da marcação das partes do discurso. Os resultados obtidos em *corpora* em língua Inglesa e Portuguesa encontram-se entre os melhores publicados.

Part-of-Speech Tagging Using Evolutionary Computation

Abstract

Part-of-speech tagging is a task of considerable importance in the field of natural language processing. Its purpose is to automatically tag the words of a text with labels that designate the appropriate parts-of-speech.

The approach proposed in this thesis divides the problem into two tasks: a learning task and an optimization task. Algorithms from the field of evolutionary computing were adopted to tackle each of those tasks. We emphasize the use of swarm intelligence, not only for the good results achieved, but also because it is one of the first applications of such algorithms to this problem.

This approach was designed with the aim of being easily extended to other natural language processing tasks that share characteristics with the part-of-speech tagging problem. The results obtained in English and Portuguese language *corpora* are among the best published.

Índice

Lista de Tabelas

Lista de Figuras

Lista de Acrónimos

AG Algoritmo Genético

AEF Autómato de Estados Finito

EE-AMC Estratégia Evolutiva com Adaptação da Matriz de Co-variância

GIC Gramáticas Independentes do Contexto

OEP Optimização por Enxames de Partículas

OCG Optimização por Colónias de Formigas

PG Programação Genética

PLN Processamento de Língua Natural

PdD Partes do Discurso

TNM Teoria *Naive* da Morfologia

WSJ *Wall Street Journal*

Capítulo 1

Introdução

Nesta tese apresentamos uma nova abordagem evolucionária ao problema da marcação das partes do discurso das palavras de um texto. O problema é dividido em duas tarefas distintas, uma de aprendizagem e outra de optimização. Estas são resolvidas recorrendo, não só a algoritmos evolucionários, mas também a optimizadores baseados em enxames de partículas, no que constitui uma das primeiras abordagens a este problema utilizando inteligência de enxame. Os marcadores resultantes foram testados em diferentes *corpora* em língua Inglesa e ainda num *corpus* em língua Portuguesa, tendo conduzido a resultados bastantes competitivos com os publicados na literatura da área.

Embora o trabalho desenvolvido se tenha centrado sobretudo na tarefa de marcação das partes do discurso, acreditamos que o resultado final pode servir de base a um novo paradigma aplicável à resolução de outras tarefas da área de processamento de língua natural. De facto, o estudo da arte realizado permitiu-nos identificar um conjunto de problemas que partilham, juntamente com a marcação das partes do discurso, algumas características que os qualificam como candidatos à aplicação da abordagem que aqui propomos.

Neste capítulo fazemos uma breve introdução ao trabalho de investigação que suporta esta tese. Esta introdução inclui uma descrição do problema da marcação das partes do discurso, a apresentação dos motivos que nos levaram a seguir esta linha de investigação, e a identificação dos contributos que acreditamos poderem ser retirados do trabalho desenvolvido. Concluímos o capítulo com a apresentação da estrutura do documento.

1.1 A Marcação das Partes do Discurso

A marcação das partes do discurso das palavras de um texto constitui uma tarefa de considerável importância na área de processamento de língua natural. A sua relevância

manifesta-se sobretudo pelo facto de ser um passo fundamental para a concretização de um vasto conjunto de outras tarefas, como é o caso da análise sintáctica de frases, tradução automática, recuperação de informação, reconhecimento de voz, etc. O objectivo da tarefa consiste em marcar automaticamente as palavras de um texto, com etiquetas que designam as partes do discurso adequadas. Tipicamente, as palavras de uma língua são agrupadas em classes gramaticais (ou partes do discurso) que representam as funções com que estas podem ser usadas numa frase. Contudo, numa grande maioria das línguas, uma mesma palavra pode ser usada de diferentes formas, podendo assim pertencer a mais do que uma classe gramatical.

De forma a conseguir marcar as palavras de um texto, um marcador automático deverá ter em consideração, para cada palavra, as partes do discurso das palavras vizinhas. No entanto, estas podem também possuir várias hipóteses de marcação. Para resolver este problema, a marcação deve, necessariamente, incluir um método que permita desambiguar o conjunto de etiquetas possíveis de uma palavra.

Os métodos usados para resolver esta tarefa podem ser agrupados em dois grupos distintos, tendo em consideração a forma como representam a informação que utilizam na tarefa de desambiguação. Num dos grupos, podemos reunir as abordagens que utilizam informação estatística sobre os contextos possíveis das diferentes hipóteses de marcação de uma palavra. Nestas propostas, o objectivo é atribuir, a cada palavra de uma frase, a parte do discurso mais provável, de acordo com as classes gramaticais das palavras vizinhas. Para tal, são usados modelos probabilísticos, como o modelo de Markov. No outro grupo, podemos incluir os marcadores que propõem a marcação à custa da utilização de regras, cujo objectivo é corrigir os erros resultantes de uma marcação básica inicial. Estas regras, inicialmente obtidas de forma manual por especialistas, começaram a ser, mais recentemente, descobertas de forma automática. As regras permitem a consideração de outros aspectos para além das categorias gramaticais das palavras vizinhas.

Recentemente, têm sido apresentados trabalhos que incluem a utilização de algoritmos evolucionários para resolver algum dos aspectos desta tarefa. Também neste caso estes trabalhos se situam num dos grupos referidos atrás. Numa linha da abordagem é usada informação estatística para efectuar a marcação à custa de um algoritmo evolucionário. Aqui a informação é representada de forma semelhante às abordagens estatísticas tradicionais. Noutra linha, a marcação é realizada por aplicação de um conjunto de regras que corrigem uma marcação inicial. Estas regras, orientadas para a correcção de erros, são obtidas à custa da utilização de um algoritmo genético. Em qualquer um dos casos, a única informação utilizada para efectuar a desambiguação é a informação de contexto, definida com base nas etiquetas das palavras vizinhas.

1.2 Motivação

Embora a marcação das partes do discurso seja uma tarefa que tem tido especial atenção na área do processamento da língua natural, atenção esta que se traduz num conjunto significativo de publicações (como podemos ver nos próximos capítulos), a abordagem evolucionária ao problema merece, no nosso entender, um estudo mais aprofundado. Consideramos que este estudo deve incluir a aplicação de outros algoritmos da área da computação evolucionária ao problema. Além disso, as abordagens já propostas sugerem a exploração destes algoritmos em dois aspectos fundamentais da tarefa: representação da informação utilizada para realizar a desambiguação e processo automático para concretizar a marcação de acordo com a informação recolhida. Por último, acreditamos que é possível definir uma abordagem que integre conjuntamente os aspectos positivos dos dois grupos de abordagens mencionados: estatísticos e baseados em regras.

À semelhança da marcação das partes do discurso, outras tarefas de processamento de língua natural envolvem, na sua resolução, um qualquer tipo de processo de desambiguação: análise sintáctica, marcação de frases nominais, reconhecimento de entidades mencionadas, análise de sentimentos, etc. Uma das abordagens mais frequentes para resolver este tipo de tarefas, como já vimos, assenta em informação estatística. De uma forma geral, podemos identificar um padrão comum na aplicação desta abordagem reflectindo uma divisão típica dos problemas em duas partes distintas:

- Construção de um modelo estatístico, a partir de recursos linguísticos, sobre determinados aspectos da informação, os quais se consideram relevantes para o processo de desambiguação.
- Resolução da tarefa, com base num algoritmo específico, o qual deverá, de alguma forma, usar a informação estatística obtida na primeira fase.

Existem, por isso, características que são transversais à aplicação desta abordagem a várias tarefas:

- A informação usada toma geralmente a forma de valores numéricos que representam dados estatísticos sobre a frequência com que determinados aspectos ocorrem nos recursos linguísticos observados.
- O modelo probabilístico adoptado limita, em alguns casos, a informação usada para resolver o problema.
- O volume da informação usada é geralmente muito elevado, o que se reflecte na eficiência do método que a recebe para propor soluções para o problema.

- As soluções propostas pelos métodos usados na segunda fase caracterizam-se por uma certa dependência em relação ao domínio dos recursos linguísticos usados para recolher a informação estatística.

Neste trabalho propomos uma nova abordagem evolucionária para resolver o problema da marcação das partes do discurso. No entanto, acreditamos que esta pode ser extensível a outras tarefas de processamento de língua natural que exibam as características atrás descritas, traduzindo-se assim num novo paradigma para resolver este tipo de problemas. Este paradigma assume, contudo, duas hipóteses fundamentais:

- É possível, a partir de recursos linguísticos específicos para a tarefa que se pretende resolver, generalizar a informação tipicamente utilizada, aprendendo regras de desambiguação, por aplicação de um algoritmo de classificação. Estas regras não terão como objectivo desempenhar o papel de um classificador, mas antes servirem como heurística para resolver a tarefa em causa.
- É possível formalizar o problema como um problema de procura e usar as regras descobertas na primeira fase, como heurística, para guiar a procura de uma solução no espaço de estados do problema.

A área da computação evolucionária inclui um conjunto de algoritmos de optimização global que têm vindo a ser aplicados, com reconhecido sucesso, a um conjunto vasto e variado de problemas em áreas como a optimização, a procura e a aprendizagem. Estes algoritmos caracterizam-se por serem facilmente modelados a diferentes tarefas e representações. São também algoritmos de optimização global, superiores, por isso, a muitas abordagens gananciosas. Apresentam-se, por isso, como ferramenta adequada a integrar a abordagem que aqui propomos, já que podem ser usados nas duas fases do problema e sobretudo porque a sua versatilidade contribuí para reforçar a possibilidade de aplicação da abordagem a outras tarefas de processamento de língua natural.

1.3 Contributos

O trabalho de investigação desenvolvido teve como objectivo testar a validade das duas hipóteses apresentadas na secção anterior, na resolução do problema da marcação das partes do discurso. O problema foi dividido em duas partes distintas: recolha e representação da informação e processo automático para concretizar a marcação, de acordo com a informação recolhida.

Na primeira parte investigámos a possibilidade de generalizar a informação tipicamente utilizada pelas abordagens assentes em modelos probabilísticos, à custa da aplicação

de um algoritmo de classificação, para a descoberta de regras, onde foram explorados diferentes algoritmos da área da computação evolucionária. Aqui foi de particular relevância a experiência adquirida nesta área, pela realização de trabalho anterior que se centrou no estudo da aplicação de algoritmos de inspiração biológica em tarefas de mineração de dados. Salientam-se como contributos específicos do trabalho desenvolvido nesta parte, os seguintes aspectos:

- Generalização da informação tipicamente utilizada nas abordagens probabilísticas, que pode ser confirmada nas experiências realizadas em diferentes *corpora*.
- Redução do volume de informação utilizado para efectuar a marcação. Esta redução assume valores na ordem dos 90%, relativamente à informação de partida.
- Utilização de uma representação que permite facilmente a inclusão de outros aspectos, para além da informação de contexto, partilhando assim os benefícios das abordagens existentes baseadas em regras de transformação. Nas experiências realizadas foram consideradas algumas características das palavras, abrindo assim caminho à utilização de outras fontes de conhecimento.
- A generalização da informação de contexto permitiu a exploração de diferentes formas de contexto, não obrigando à definição, *a priori*, do número de palavras que obrigatoriamente têm que ser consideradas na vizinhança de uma palavra.
- Permitir o desenvolvimento de marcadores menos dependentes do domínio do *corpus* usado para obter a informação estatística.

A segunda parte do trabalho preocupou-se em testar a validade da segunda hipótese colocada. Aqui centrámos-nos na definição de um marcador das partes do discurso capaz de propor uma solução para o problema, à custa da informação obtida na primeira parte. A solução proposta formaliza o problema como um problema de optimização combinatória e propõe explorar o respectivo espaço de estados usando como heurística as regras inicialmente descobertas. Os contributos específicos da solução proposta para esta parte são:

- Formalização do problema como um problema de optimização combinatória, a qual permitiu a utilização de algoritmos de procura global, como é o caso dos algoritmos da área da computação evolucionária utilizados. Destaca-se a aplicação de optimização baseada em enxames de partículas, não só porque se trata, tanto quanto é do nosso conhecimento, da primeira aplicação destes algoritmos a este problema, mas sobretudo porque conduziu a resultados bastante competitivos com os publicados para os *corpora* usados no trabalho experimental.

- Todo o conhecimento relacionado com a língua é passado como parâmetro de entrada, o que resulta num marcador cujo funcionamento é independente da língua para a qual possa ser executado.

Como contributos gerais gostaríamos de salientar o facto desta ser a primeira abordagem ao problema da marcação das partes do discurso que utiliza algoritmos da área da computação evolucionária para resolver todos os aspectos da tarefa. De salientar, também, que a proposta apresentada estabelece uma metodologia genérica, que pode ser entendida como um padrão passível de ser aplicado a outros problemas, estabelecendo assim um novo paradigma para resolver um conjunto mais alargado de tarefas da área de processamento de língua natural.

A aplicação de algoritmos representativos da área da computação evolucionária a este problema contribui, de uma forma geral, para o desenvolvimento do conhecimento destas duas áreas, e, mais especificamente, para melhor compreender as potencialidades dos algoritmos evolucionários e inteligência de enxame quando aplicados à área do processamento de língua natural. Do trabalho realizado resultou um conjunto de artigos publicados em actas de conferências internacionais e nacionais, revistas e capítulos de livros:

- Alguns aspectos relacionados com a aprendizagem evolucionária de regras de classificação são discutidos em [Sousa et al., 2003, 2004, 2005]. Estes artigos resultam de trabalho anterior desenvolvido entre 2003 e 2005, o qual é brevemente resumido no Capítulo 5.
- Em [Silva et al., 2012*b*, 2013*a*; Silva and Rodrigues, 2013] descrevemos as várias componentes da nossa abordagem evolucionária ao problema da marcação das partes do discurso.
- A aplicação de inteligência de enxame ao problema da marcação das partes do discurso é apresentada em [Silva et al., 2012*a*, 2013*b*].

1.4 Estrutura

No próximo capítulo descrevemos com maior detalhe a tarefa da marcação das partes do discurso, apresentando as abordagens tradicionais ao problema. O Capítulo 3 tem como objectivo introduzir os princípios fundamentais da área da computação evolucionária, apresentando os seus algoritmos mais representativos. No Capítulo 4 fazemos um apanhado de alguns dos contributos da área da computação evolucionária em tarefas de processamento de língua natural. Os capítulos enunciados respondem à necessidade

de contextualização desta tese em duas áreas distintas das ciências da computação: processamento de língua natural e computação evolucionária.

Os restantes capítulos apresentam as diferentes partes do trabalho de investigação desenvolvido. No Capítulo 5, abordamos o problema da representação e recolha da informação utilizada para resolver o problema da marcação das partes do discurso. Aqui descrevemos a descoberta de regras de desambiguação à custa da aplicação de dois algoritmos representativos da área da computação evolucionária. O sistema de marcação automática proposto é delineado no Capítulo 6. Os resultados experimentais obtidos, em cada uma das partes do trabalho desenvolvido, são apresentados em conjunto no Capítulo 7. Por fim, no Capítulo 8, reunimos as conclusões que retirámos do trabalho de investigação atrás apresentado, e as perspectivas de trabalho futuro resultantes do mesmo.

Capítulo 2

Marcação das PdD

Neste capítulo descrevemos a tarefa de marcação das partes do discurso das palavras de um texto e apresentamos as abordagens clássicas à resolução do problema: abordagem estatística e abordagem baseada em regras de transformação.

2.1 O Problema da Marcação das PdD

O processo de etiquetar (ou marcar), automaticamente, cada uma das palavras de um texto, com a parte do discurso correspondente, é denominado em inglês por *part-of-speech tagging*, ou simplesmente *POS tagging*. Em português iremos usar o termo marcação das PdD. Os sistemas automáticos que efectuam esta marcação são denominados, em inglês, por *POS tagger*, ou simplesmente *tagger*. Em português adoptaremos o termo etiquetador (ou marcador) das PdD, ou simplesmente marcador.

As palavras de uma determinada língua são, geralmente, agrupadas em categorias linguísticas, normalmente designadas por partes do discurso (PdD), ou classe gramatical. Quase todas as línguas possuem as classes Nome e Verbo, variando significativamente nas restantes categorias. Na gramática portuguesa são consideradas dez classes gramaticais: Substantivo, Adjectivo, Numeral, Pronome, Artigo, Verbo, Advérbio, Preposição, Conjunção e Interjeição. Uma mesma palavra pode, no entanto, pertencer a mais do que uma classe, dependendo da sua função na frase. De facto, segundo [De Rose, 1988], cerca de 40% das palavras no *corpus* de *Brown* são ambíguas, ou seja, pertencem a mais do que uma classe gramatical.

Um marcador das PdD é, por isso, um sistema automático que recebe como entrada um texto não anotado e devolve o mesmo texto, mas com cada uma das palavras marcada com uma determinada etiqueta.

A avaliação do desempenho dos marcadores das PdD é realizada tendo em conta as classes gramaticais que seriam atribuídas por um especialista. Nesse sentido, são usados textos marcados manualmente por especialistas, normalmente designados por *corpus*. Esta marcação é assumida como correcta, e usada como referência para avaliação das marcações sugeridas pelos marcadores automáticos. Ou seja, a marcação de uma determinada palavra é considerada correcta quando é igual à marcação definida manualmente. Como é natural, os especialistas humanos que realizaram a marcação manual, não são infalíveis. Por este motivo, podem ser detectados alguns erros de marcação. No entanto, a marcação apresentada no *corpus* é, por definição, considerada correcta.

As etiquetas usadas na marcação pertencem ao conjunto de etiquetas definido previamente para o sistema. Estas consistem em acrónimos que designam classes gramaticais. Os *corpora* anotados disponíveis usam diferentes convenções para marcar as palavras, e por isso os conjuntos de etiquetas variam, regra geral, de *corpus* para *corpus*.

A marcação das PdD é uma das tarefas básicas do processamento de língua natural. Básica, não por que seja simples, mas no sentido em que se revela um passo necessário para um conjunto de outras tarefas, como sejam a análise sintáctica das frases, sistemas de reconhecimento de voz, sistemas de recuperação de informação, etc. A tarefa consiste em escolher, para cada uma das palavras do texto, de entre as diferentes possibilidades, a classe gramatical correcta, atendendo ao contexto em que a palavra está inserida. Por exemplo, a palavra *dado* pode ser usada como substantivo, adjectivo, ou ainda como verbo:

- Como substantivo, *A Maria joga o dado.*
- Como adjectivo, *O Manuel é um rapaz muito dado.*
- Como verbo, *O Manuel depois de ter dado a aula foi para casa.*

O contexto em que cada palavra surge na frase ajuda a decidir qual a parte de discurso correcta. É com base nesta observação que a maioria dos marcadores define a sua abordagem. Uma boa forma de percebermos a importância do contexto é concretizando o problema para uma frase em particular. O exemplo que se apresenta na Tabela 2.1 é ilustrado em [Araujo, 2002*b*]. A frase escolhida foi retirada do *corpus* de *Brown*: *"This the therapist may pursue in later questioning"*.

Como podemos ver, para a maioria das palavras da frase, existe mais que uma possibilidade de marcação. Assim, numa tentativa de resolver o problema, poderíamos optar por desambiguar a palavra "*questioning*", definindo-a como sendo um nome comum. Isto, no caso de termos optado por desambiguar a palavra imediatamente antes, "*later*", como sendo um adjectivo. No entanto, pode acontecer que esta palavra seja também ela ambígua, o que levanta um conjunto de dependências que têm que ser resolvidas simultaneamente.

A importância, e dificuldade, desta tarefa despertou grande interesse na comunidade científica, o que levou ao desenvolvimento de um conjunto significativo de marcadores das PdD automáticos. A grande maioria destes marcadores são probabilísticos e baseiam-se em dados estatísticos recolhidos de textos sendo estes marcados manualmente. Estes sistemas não necessitam de conhecimento sobre as regras da língua, nem tentam de alguma forma deduzi-las, e por esse motivo podem ser aplicados a textos em qualquer língua, desde que nela sejam treinados previamente. Isto poderá ser um problema, já que nem sempre existem recursos disponíveis para efectuar o treino, nomeadamente *corpus* anotados.

Existem algumas tarefas de processamento de língua natural para as quais a marcação das PdD surge como uma passo prévio óbvio, como por exemplo, a análise sintáctica de frases. No entanto, existem outras tarefas que beneficiam com a marcação das PdD, como é o caso dos sistemas de conversão texto-fala [Steven Bird and Loper, 2009]. Consideremos a frase *"They refuse to permit us to obtain the refuse permit"*. Como podemos ver, a palavra *refuse* é usada por duas vezes na frase. No entanto, está a ser usada com significados distintos. Na primeira ocorrência, a palavra é usada com a função de Verbo com o significado de "recusar". Na segunda ocorrência, *refuse* é usada como um Substantivo com o significado de "refugo", ou "lixo". Em inglês estas duas palavras não são homófonas. O verbo *refuse* pronuncia-se "refUSE", enquanto o substantivo se pronuncia "REFuse". É por isso necessário saber que palavra está a ser usada, de forma a que esta possa ser pronunciada correctamente.

Tabela 2.1: As classes apresentadas na tabela fazem parte do conjunto de etiquetas definido no *corpus* de *Brown*: DT - Determinante/Pronome, AT - Artigo, NN - Nome Comum, IN - Preposição, JJR - Adjectivos Comparativos, etc. As etiquetas sublinhadas correspondem às correctas, de acordo com a marcação do *corpus* de *Brown*

"This	*the*	*therapist*	*may*	*pursue*	*in*	*later*	*questioning."*
DT	AT	NN	NNP	VB	RP	RP	VB
QL			MD	VBP	NNP	RB	NN
					RB	JJ	JJ
					NN	JJR	
					FW		
					IN		

2.2 Abordagem Estatística

Como já referimos, a maioria dos sistemas de marcação actuais são baseados em modelos estatísticos, definidos sobre um conjunto de parâmetros cujos valores são extraídos

a partir de textos marcados manualmente. O objectivo deste tipo de modelos é atribuir, a cada palavra de uma frase, a parte do discurso mais provável, de acordo com o seu contexto, i.e., de acordo com as classes gramaticais das palavras que a rodeiam. Para tal, são recolhidos, para cada possibilidade de marcação de uma palavra, dados estatísticos sobre o número de ocorrências dos diferentes contextos. Obtidos estes dados, deverá ser escolhida, para cada palavra, a parte do discurso mais provável. No entanto, como também já vimos, as palavras circundantes podem, também elas, ter várias possibilidades de classificação, sendo necessário usar um modelo estatístico que permita seleccionar a melhor marcação para toda a sequência, de acordo com o modelo.

2.2.1 Marcador baseado em uni-gramas

Os marcadores baseados em uni-gramas baseiam-se num algoritmo estatístico simples o qual atribui a cada palavra a classe gramatical mais provável. O marcador passa por uma primeira fase de treino, que consiste em analisar uma parte substancial do *corpus* (denominado conjunto de treino) e guardar para cada palavra a classe gramatical mais comum (uni-grama). Ou seja, a classe com que essa palavra aparece mais vezes marcada, no *corpus* analisado. Por exemplo, um marcador uni-grama quando treinado no *corpus* de *Brown*, iria marcar a palavra "*frequent*" com a etiqueta JJ, já que esta palavra é usada mais frequentemente como um adjectivo (por exemplo, "*a frequent word*"), do que como um verbo (por exemplo, "*I frequent this cafe*").

2.2.2 Marcador baseado em n-gramas

Quando efectuamos a marcação das PdD com base em uni-gramas o contexto usado para efectuar a desambiguação é de apenas uma palavra. Considera-se apenas a palavra que se pretende marcar, excluindo outros contextos mais alargados. Um modelo deste tipo apenas consegue marcar cada palavra com a classe mais provável aprendida previamente. Isto significa que uma palavra como *"wind"* será sempre marcada com a mesma classe, independentemente se é usada no contexto *"the wind"* ou *"to wind"*. Um marcador baseado em n-gramas é uma generalização do marcador baseado em uni-gramas, em que o contexto utilizado é a palavra que está a ser alvo de marcação, em conjunto com as partes do discurso das $n-1$ palavras que a precedem. Um marcador baseado em n-gramas marca uma palavra com a classe gramatical mais provável, atendendo ao contexto em causa.

Formalmente, o problema da marcação das PdD das palavras de uma frase pode ser definido como em [Alba et al., 2006],

$$t_{1,n} = arg \max_{t_{1,n}} P(t_{1,n} \mid w_{1,n}) \tag{2.1}$$

onde $arg\max_x f(x)$ representa o valor de x que maximiza $f(x)$, e $t_{1,n}$ é a sequência de classes gramaticais das palavras $w_{1,n}$ que compõem a frase.

Se assumirmos que a parte do discurso de uma palavra apenas depende da classe da palavra que imediatamente a precede, e que esta dependência não varia ao longo do tempo, então é possível a adopção de um modelo de Markov para efectuar a marcação. Seja w_i a palavra na posição i no texto, t_i a classe gramatical de w_i, $w_{i,j}$ a sequência de palavras da posição i até à posição j, e finalmente $t_{i,j}$ a sequência de etiquetas correspondentes às palavras $w_{i,j}$. Então o modelo definido estabelece que

$$P(t_{i+1} \mid t_{1,i}) = P(t_{i+1} \mid t_i) \tag{2.2}$$

Se, além disso, assumirmos que a probabilidade de uma palavra surgir numa determinada posição apenas depende da parte do discurso atribuída a essa posição, a sequência óptima de etiquetas para as palavras da frase pode ser estimada como sendo:

$$t_{1,n} = arg\max_{t_{1,n}} P(t_{1,n} \mid w_{1,n}) = \prod_{i=1}^{n} P(w_i \mid t_i)P(t_i \mid t_{i-1}) \tag{2.3}$$

Assim, os parâmetros do modelo de Markov podem ser obtidos a partir de um *corpus* marcado manualmente. Geralmente, essa informação é armazenada numa tabela, denominada tabela de treino, ou tabela de n-gramas. Estas tabelas podem ser construídas percorrendo o *corpus* de treino e guardando todos os diferentes contextos que nele surgem, assim como o número de vezes em que neles surgem cada uma das etiquetas consideradas na marcação do *corpus*.

O modelo de Markov descrito é conhecido por marcador bi-grama, dado que faz predições atendendo somente à classe gramatical da própria palavra e da que a imediatamente precede. Ou seja, as decisões baseiam-se apenas em dois aspectos da informação: a etiqueta da palavra corrente e da palavra anterior. No entanto, o modelo pode ser estendido de forma a contemplar contextos mais alargados, considerando gramas de diferentes tamanhos. Por exemplo, o marcador tri-grama efectua predições tendo como base etiquetas das duas palavras imediatamente à esquerda de uma determinada palavra. Uma vez definido o modelo estatístico, a maioria dos marcadores utiliza o algoritmo de *Viterbi* [Forney, 1973], baseado em programação dinâmica, para encontrar a sequência de classes gramaticais que maximiza a probabilidade, de acordo com o modelo de Markov escolhido.

Um dos problemas deste tipo de marcadores reside precisamente na dimensão destas tabelas, que geralmente são compostas por um número elevadíssimo de entradas [Steven Bird and Loper, 2009]. Um segundo problema está relacionado com a informação de contexto usada. A única informação que um marcador n-grama utiliza diz respeito

às etiquetas das palavras à esquerda de uma determinada palavra. No entanto, as próprias palavras podem ser recipientes de informação relevante, assim como a informação de contexto à direita da palavra alvo. Na secção seguinte descrevemos o marcador de Brill, que contempla informação lexical para realizar a desambiguação.

2.3 Abordagem Usando Regras de Transformação

A abordagem clássica ao problema de marcação das PdD baseada em regras de transformação foi proposta por Brill em [Brill, 1995]. O marcador proposto é geralmente denominado por marcador de Brill. Trata-se de um sistema de marcação baseado em regras, denominadas regras de transformação, e apresenta-se como alternativa aos sistemas baseados em modelos estatísticos. O processo de marcação pode ser dividido em duas fases: marcação inicial das palavras do texto de entrada, e a correcção dos erros de marcação presentes na marcação inicial, por aplicação de regras de transformação. A marcação inicial é realizada por um marcador por defeito cuja complexidade pode variar desde a marcação aleatória das palavras, até à utilização de uma marcador baseado em n-gramas. Depois de efectuada a marcação inicial, segue-se a correcção de erros por aplicação de regras pertencentes a uma lista ordenada de regras de transformação. O processo termina quando já não é possível aplicar nenhuma regra.

As regras de transformação são obtidas à custa de um algoritmo de aprendizagem que recebe um conjunto de padrões de transformação, um texto anotado com marcações consideradas correctas, e o mesmo texto anotado por um marcador por defeito. O algoritmo devolve como saída uma lista ordenada de regras de transformação que resultam da instanciação dos padrões de transformação. Esta instanciação é conseguida pela comparação do texto com as anotações padrão, i.e., com o texto anotado automaticamente. Os padrões de transformação são da forma:

"Substituir a etiqueta $etiqueta_1$ **pela etiqueta** $etiqueta_2$ **quando** $etiqueta_3$ **é encontrada numa determinada posição."**

O algoritmo de aprendizagem é responsável pela instanciação das regras. A instanciação de um padrão resulta numa regra de transformação ou reescrita. Um exemplo de instanciação para o padrão anterior, considerando as categorias definidas para o *corpus* do *Wall Street Journal* do *Penn Treebank*, é a regra "Substituir a etiqueta **NN** pela etiqueta **VB** caso a etiqueta anterior seja **TO**". O marcador de Brill apresenta 97.20% de exactidão no *corpus* do *Wall Street Journal*.

Para que uma regra de reescrita seja aplicada, é necessário que se verifique o ambiente de disparo correspondente, ou seja, que as condições da regra sejam satisfeitas. As

transformações são aplicadas da esquerda para a direita, e têm um atraso no efeito. Isto significa que aplicações de uma mesma transformação não podem influenciar-se mutuamente. Por exemplo, a transformação, B substitui A se a classe precedente é A, deverá alterar AAAA em ABBB, em vez de ABAB.

O sistema apresentado em [Brill, 1995] tem como entradas o *corpus* do *Wall Street Journal* do *Penn Treebank*, um marcador por defeito baseado em uni-gramas, um conjunto de padrões de transformação e uma função objectivo. O sistema começa por usar o marcador por defeito para marcar cada palavra do *corpus* de treino com a classe gramatical mais frequente. De seguida, o algoritmo de aprendizagem constrói, por instanciação dos padrões de transformação, uma lista ordenada de regras de transformação que irão alterar a marcação inicial para uma mais próxima da correcta. A lista ordenada de transformações é obtida através de uma procura ao estilo trepa-colinas. Resumidamente, o algoritmo determina, em cada iteração, a transformação cuja aplicação resulta no melhor valor da função objectivo. A melhor transformação é adicionada à lista ordenada, e os documentos de treino são actualizados por aplicação da nova regra. Assim, em cada iteração, são testadas todas as instanciações possíveis dos padrões de regras utilizados, escolhendo-se a que resulta num melhor valor da função objectivo (maior incremento na exactidão da marcação dos textos de treino). Os padrões usados por Brill foram os seguintes:

Alterar a etiqueta de e_1 para e_2 quando:

1. a palavra anterior (seguinte) está marcada com a etiqueta e_3;
2. a segunda palavra à esquerda (direita) está marcada com a etiqueta e_3;
3. uma das duas palavras à esquerda (direita) está marcada com a etiqueta e_3;
4. uma das três palavras à esquerda (à direita) está marcada com a etiqueta e_3;
5. a palavra imediatamente à esquerda está marcada com a etiqueta e_3 e a palavra imediatamente à direita com a etiqueta e_4;
6. a palavra à esquerda (direita) está marcada com a etiqueta e_3, e a segunda palavra à esquerda (à direita) está marcada com a etiqueta e_4;

onde e_1, e_2, e_3 e e_4 tomam valores no conjunto de todas as etiquetas consideradas no *corpus* utilizado.

A aprendizagem desenrola-se até ao ponto em que nenhuma instanciação permite melhorar a marcação dos textos de treino. A solução encontrada pelo algoritmo corresponde à lista ordenada de regras obtida.

As regras de transformação aprendidas quando usadas para realizar efectivamente correcções em textos anotados previamente, alteram a marcação de uma palavra de e_x para e_y se e só se acontecer uma de duas coisas:

1. A palavra não fazia parte das palavras dos textos de treino;
2. A palavra surgiu pelo menos uma vez com a marcação e_y no *corpus* de treino.

De uma forma geral, os marcadores estocásticos baseados em *n-gramas* não consideram relações entre palavras. Nos modelos de Markov, como vimos anteriormente, as probabilidades de transição de estados expressam a probabilidade de uma determinada etiqueta imediatamente suceder n outras, emitindo probabilidades que expressam as hipóteses de uma palavra estar a ser usada com uma determinada função na frase.

Brill considerou, no entanto, que outras relações podem mostrar-se úteis na tarefa de marcação, como é o caso das relações que podem ser estabelecidas entre uma palavra e a palavra anterior, ou entre uma parte do discurso e a palavra que a sucede. Este tipo de relações não são, directamente, capturadas nos marcadores baseados em modelos de Markov. Na versão do marcador de Brill que acabámos de descrever, também não são contempladas outras relações, senão as estabelecidas entre classes gramaticais. No entanto, em [Brill, 1995] o autor apresenta uma extensão ao marcador, de forma a incluir transformações contextuais que façam referência a palavras, assim como a partes do discurso. Foram considerados os seguintes padrões de transformação:

Alterar a etiqueta de e_1 para e_2 quando:

1. a palavra anterior (seguinte) é a palavra p_1;
2. a segunda palavra à esquerda (direita) é p_1;
3. uma das duas palavras à esquerda (direita) é p_1;
4. a palavra corrente é p_1 e a palavra anterior (seguinte) é p_2;
5. a palavra corrente é p_1;
6. a palavra anterior (seguinte) é p_1 e a etiqueta anterior (seguinte) é e_3;
7. a palavra corrente é p_1, a palavra anterior (seguinte) é p_2, e a etiqueta à esquerda (à direita) é e_3;

onde p_1 e p_2 são variáveis que tomam valores no conjunto de todas as palavras pertencentes ao *corpus* de treino, e e_1, e_2 e e_3 tomam valores no conjunto de todas as partes do discurso consideradas. Duas das transformações aprendidas pelo marcador, quando treinado no *corpus* do *Wall Street Journal* foram:

Alterar a etiqueta:

- de **IN** para **RB** se a segunda palavra à direita é *as*;
- de **VBP** para **VB** se uma das duas palavras anteriores é *n't* .

As abordagens que acabámos de descrever são representativas das duas correntes que podem ser encontradas na literatura para atacar o problema. Estas duas visões caracterizam-se pela forma como reúnem e representam a informação necessária para resolver o problema: valores numéricos que traduzem informação estatística relevante para a tarefa, e regras de transformação orientadas para a correcção de erros que determinam um processo de marcação específico que se caracteriza pela correcção sistemática de erros de marcação.

Capítulo 3

Computação Evolucionária

Neste capítulo pretendemos fazer uma introdução à área onde se integram os algoritmos principais que utilizamos no trabalho de investigação subjacente a esta tese. Começamos por debater os aspectos fundamentais do problema de optimização global, e da área da computação evolucionária, na qual identificamos dois grandes grupos de algoritmos: algoritmos evolucionários e inteligência de enxames. Para cada um deles apresentamos as variantes mais conhecidas. Dentro dos algoritmos evolucionários destacamos os algoritmos genéticos, estratégias evolutivas, programação evolucionária e programação genética. A optimização por enxames de partículas e colónias de formigas são abordadas dentro do grupo dedicado à inteligência de enxames.

3.1 Optimização Global

Em qualquer ramo da actividade humana está sempre presente a necessidade de encontrar soluções óptimas para os problemas que se colocam, sejam elas as mais económicas, as mais rápidas, as mais eficientes ou as mais robustas. Também na natureza esta procura do estado óptimo está sempre presente, desde os átomos, que procuram configurações que minimizem a energia dos seus electrões, até aos seres vivos mais complexos que, através da selecção natural, são levados a adaptar-se o melhor possível ao ambiente que os rodeia. A área científica que se ocupa deste tipo de problemas é normalmente denominada optimização global.

O objectivo da optimização global de um determinado problema consiste em encontrar os elementos x^* de um conjunto $\mathbb{X}$ que possam ser considerados os melhores segundo um conjunto de critérios $\mathbf{f} = f_1, f_2, ..., f_n$. Na versão mais simples do problema $\mathbf{f}$ tem um único elemento f, o qual é denominado função objectivo. Quando $|\mathbf{f}| > 1$, i.e. existe mais do que um objectivo que deve ser optimizado concorrentemente, estamos

na presença de um problema de optimização multi-critério, o qual implica a utilização de técnicas especiais dentro da área da optimização global.

Uma função objectivo é assim uma função $f : \mathbb{X} \mapsto \mathbb{Y}$, na qual os elementos de $\mathbb{X}$ podem ser qualquer tipo de solução para o problema em causa, e.g. vectores de números reais, permutações de inteiros ou mesmo até algoritmos. A função f mapeia cada solução x para um valor real $y \in \mathbb{Y} \subset \mathbb{R}$. Assumindo um problema de maximização, no qual pretendemos optimizar $y = f(x)$, podemos ver f como uma medida da qualidade das soluções presentes em $\mathbb{X}$.

Uma técnica ou algoritmo de optimização global procura óptimos globais para a função objectivo $f(x)$. Um óptimo global x^* pode ser um máximo global $\hat{x}$, quando $f(\hat{x}) \geqslant f(x)\ \forall x \in \mathbb{X}$, ou um mínimo global $\check{x}$, quando $f(\check{x}) \leqslant f(x)\ \forall x \in \mathbb{X}$. Uma das maiores dificuldades da optimização global surge quando, além de um ou vários óptimos globais, existem também outros pontos, que não sendo melhores que todos os pontos do domínio, são pelo menos melhores que todos os pontos numa determinada vizinhança ϵ. Estes pontos são denominados máximos locais, quando $f(\hat{x}_l) \geqslant f(x), \forall x \in \mathbb{X}, |x - \hat{x}_l| < \epsilon$ e mínimos locais, quando $f(\check{x}_l) \leqslant f(x), \forall x \in \mathbb{X}, |x - \check{x}_l| < \epsilon$. A definição de óptimo nos problemas multi-objectivo é mais complexa, havendo diversos critérios possíveis para os identificar.

As abordagens ao problema de optimização global podem ser divididas em abordagens deterministas e abordagens probabilísticas. Quando a relação entre as características das possíveis soluções e sua qualidade é bem conhecida, e o espaço de procura não é muito grande, é possível definir boas heurísticas que permitem a técnicas bem conhecidas, como, por exemplo, a procura no espaço de estados, encontrar soluções óptimas de forma determinista e com complexidade temporal e espacial dentro de limites práticos. No entanto, para espaços de procura muito vastos, ou quando a relação entre as características de uma solução e a sua utilidade não é bem conhecida ou é muito complexa, estes métodos começam a falhar, podendo resultar na exploração exaustiva do espaço de procura. Nestas situações é frequentemente mais efectivo utilizar algoritmos de optimização probabilísticos.

Muitos algoritmos probabilísticos, geralmente agrupados sob a denominação de métodos de Monte Carlo, trocam a garantia de encontrar o óptimo global pela possibilidade de encontrarem uma solução ligeiramente sub-óptima em tempo útil. Entre os muitos exemplos deste tipo de abordagem, podemos listar o trepa-colinas estocástico, a recristalização simulada (do Inglês: *Simulated Annealing*) ou a procura tabu. Estes métodos fazem também parte do grupo alargado das meta-heurísticias. Tipicamente usam uma heurística genérica, frequentemente inspirada num mecanismo biológico ou num processo físico, para prioritizar, seleccionar ou gerar quais as soluções candidatas a investigar.

Para resolver um problema específico, as meta-heuríticas combinam e/ou adaptam a heurística central com a função objectivo do problema, obtendo-se um algoritmo específico. A estrutura interna daquele raramente é tida em conta neste tipo de abordagem, ou por não ser totalmente conhecida ou por ser de difícil exploração, sendo o problema assim tratado como um entidade opaca. Deste facto resulta outra denominação adicional frequentemente utilizada para este tipo de abordagem: métodos do tipo caixa-negra.

Dentro desta classe geral de métodos, um grupo de algoritmos que nas últimas décadas tem obtido êxito assinalável num vasto número de problemas, encontra-se agrupado sobre a denominação genérica de computação evolucionária (do Inglês *Evolutionary Computation.*). Esta constitui um ramo da inteligência computacional [Engelbrecht, 2007] que originalmente se centrava no estudo e desenvolvimento de algoritmos de optimização baseados na exploração dos princípios da evolução natural. Mais recentemente engloba também meta-heurísticas baseadas no comportamento de grupos de insectos ou outros animais sociais, a denominada inteligência de enxame.

Além das meta-heurísticas de inspiração biológica, a computação evolucionária distingue-se de outras áreas da optimização por se basear na modificação iterativa de uma população de soluções candidatas, em vez de um candidato único, como ocorre em muitos outros algoritmos de optimização. Esta característica traz-lhe várias vantagens, como sejam a exploração paralela do espaço de procura e a possibilidade inerente de uma implementação distribuída. Nesta tese iremos utilizar vários algoritmos que se integram nesta área, pelo que nas próximas secções nos iremos debruçar sobre algumas das famílias de algoritmos evolucionários mais utilizados, as suas propriedades comuns e características distintivas. Uma discussão detalhada da optimização global, e do posicionamento da computação evolucionária dentro desta, pode ser encontrada, por exemplo, em [Weise, 2008].

3.2 Computação Evolucionária

Historicamente, muitos dos métodos e algoritmos hoje integrados na área da computação evolucionária, têm a sua génese no desejo de criar mecanismos computacionais capazes de resolver problemas utilizando princípios semelhantes aos apresentados por [Darwin, 1872] para descrever a evolução por selecção natural. Darwin assentou a sua teoria sobre quatro pilares essenciais:

- Variação: dentro de uma população existem características e comportamentos que variam entre indivíduos, como sejam o tamanho, a cor e forma de padrões no revestimento, a rapidez, etc... O grau de variação varia de característica para

característica: enquanto a cor dos olhos pode variar substancialmente dentro de uma população, o número de olhos, por exemplo, é geralmente o mesmo.

- Herança: Alguns dos traços presentes nos progenitores passam consistentemente para os descendentes - são traços transmissíveis. Já outros, desenvolvidos frequentemente por interacção com o ambiente, não são passíveis de herança pela geração seguinte.

- Crescimento populacional: As populações reproduzem-se a um nível superior ao necessário para manter o seu tamanho constante. Como, na natureza, os recursos existentes são finitos, esta situação leva inevitavelmente a situações de competição pelos recursos existentes. Este facto, associado a outros desafios presentes pelo ambiente, como a predação, resultam na morte duma parte substancial de cada geração.

- Sobrevivência e reprodução diferenciais: Dentro da população, os indivíduos com traços mais favoráveis, são os que, estatisticamente, têm mais sucesso, sendo esse sucesso medido essencialmente pelo número de descendentes a que conseguem transmitir esses traços. Indivíduos mais favorecidos vivem mais tempo, atraem mais e/ou melhores parceiros, e tendem portanto a produzir mais descendentes, os quais deverão herdar alguns dos seus traços.

Note-se que existe um quinto pilar implícito na teoria, já que os mecanismos descritos assentam no facto do processo evolutivo decorrer sempre no contexto de uma população de indivíduos. A "qualidade"de um indivíduo só faz sentido no contexto da população, não sendo um valor absoluto, mas sim relativo. A conjugação destes mecanismos ao longo do tempo produz indivíduos progressivamente melhor adaptados ao ambiente em que vivem, sendo este o "problema"que é resolvido pela evolução por selecção natural.

A computação evolucionária imita a evolução por selecção natural, criando paralelismos entre os mecanismos anteriores e elementos algorítmicos presentes nas abordagens evolucionárias, e substituindo o problema de sobrevivência colocado pelo ambiente por um problema de optimização. Assim, um método desta área, procura o indivíduo de um espaço de indivíduos possíveis, que maximiza (ou minimiza) uma determinada função de desempenho, a qual mede a sua qualidade como solução para o problema a ser resolvido. Para que essa procura seja possível, torna-se necessário definir, para cada novo problema, um conjunto de elementos que incorpora os pilares da evolução por selecção natural atrás descritos:

- Representação - A primeira etapa no desenvolvimento de uma abordagem evolucionária a um novo problema envolve sempre a definição de uma representação computacional para as possíveis soluções do problema. Esta pode ser tão simples

como um vector de variáveis binárias ou tão complexa que necessite da combinação de várias estruturas de dados para ser construída. É essencial porque, por um lado, está associada à definição do espaço de procura $\mathbb{G}$, e a estrutura deste espaço condiciona fortemente o sucesso do processo de optimização. Por outro lado, é a definição de uma codificação padrão para as soluções que permite e suporta a herança de características que permite o processo evolutivo.

- Variação - Para podermos explorar $\mathbb{G}$ necessitamos de operadores que, a partir de um ou mais elementos do conjunto sejam capazes de gerar outros pontos promissores do espaço de procura. Para tal, estes operadores usam heurísticas (ou informação sobre o problema) na criação de novas soluções. Heurísticas típicas podem ser procurar na vizinhança de soluções promissoras já encontradas ou recombinar características de soluções já presentes na população. São estes operadores que garantem que características interessantes encontradas numa população serão transmitidas à população seguinte. São também estes operadores que deslocam a população no espaço de procura $\mathbb{G}$, desejavelmente na direcção de áreas onde as soluções apresentem melhor qualidade.

- Mapeamento do espaço de procura para o espaço do problema - À imagem do que acontece na natureza, também na computação evolucionária está frequentemente presente uma dicotomia entre a representação da solução no espaço de procura $\mathbb{G}$ e a expressão dessa representação no espaço do problema $\mathbb{X}$. Na natureza $\mathbb{G}$ corresponde a todas as combinações possíveis do material genéticos dos indivíduos e $\mathbb{X}$ corresponde a todos os indivíduos expressos por essas combinações de material genético. Na computação evolucionária uma solução $g \in \mathbb{G}$, codificada numa estrutura de dados, pode também ser convertida num ponto x do espaço do problema $\mathbb{X}$. Também por paralelismo com a evolução por selecção natural, g é geralmente denominado o genótipo do indivíduo e x o seu fenótipo.

- Avaliação - Não é possível haver optimização sem avaliação da qualidade ou desempenho das soluções candidatas. Na computação evolucionária, para cada problema, tem assim de ser definida uma função $f(x)$ que para cada elemento de $\mathbb{X}$ devolva um valor $v \in \mathbb{V} \subseteq \mathbb{R}^+$ que permita comparar a sua qualidade com a dos restantes elementos da população. Note-se que estamos aqui a assumir o caso simples em que o problema de optimização tem apenas uma função objectivo. No caso da optimização multi-critério o resultado das n funções objectivo seria um vector de valores $\mathbf{y} \in \mathbb{R}^n$, mas também aqui teria de haver um mapeamento para um valor único v que permitisse a avaliação do indivíduo.

- Selecção - De forma a, mais uma vez, imitar a evolução por selecção natural, também na computação evolucionária é necessário garantir que, dentro de uma população, indivíduos com melhor avaliação produzam mais descendentes (ou tenham maior probabilidade de produzir descendentes) do que indivíduos com pior

avaliação. Embora haja outras estratégias possíveis, a mais frequente consiste em agir na selecção dos indivíduos aos quais vão ser aplicados os operadores de variação, garantindo que quanto melhor for um indivíduo maior probabilidade tem de ser escolhido. Obtém-se assim a reprodução diferencial desejada, com os melhores indivíduos a produzirem mais descendentes.

- Substituição - Como o tamanho da população é geralmente fixo ao longo do processo evolutivo nestes algoritmos, cada vez que novos indivíduos são criados, utilizando os mecanismos de variação, verifica-se o mesmo mecanismo de crescimento populacional que é um pilar da evolução por selecção natural. Havendo mais indivíduos do que recursos (neste caso "espaço"dentro da população) é necessária uma estratégia que permita ajustar o número de indivíduos aos limites pré-estabelecidos. A abordagem mais comum consiste em substituir toda a população de progenitores pelo mesmo número de descendentes (substituição geracional), embora existam várias outras estratégias possíveis.

Uma abordagem evolucionária combina os elementos anteriores num algoritmo de optimização específico para o problema a ser abordado, no sentido em que vários daqueles elementos dependem substancialmente das características do problema. No entanto, todos os algoritmos dentro da computação evolucionária assentam também em princípios comuns, por um lado por, na generalidade, partilharem os elementos atrás descritos e, por outro, por manipularem espaços e conjuntos que não só são partilhados por estes algoritmos como estão subjacentes ao próprio processo geral de optimização [Weise, 2008]:

- Espaço de procura - Todos os elementos g que podem ser construídos utilizando a representação definida para o problema, podendo assim ser objecto dos operadores de variação/procura, constituem o espaço de procura $\mathbb{G}$. Cada g apresenta uma codificação do indivíduo, o seu **genótipo**, o qual tipicamente se divide em partes distintas, denominadas **genes**, cada uma delas codificando uma característica específica do indivíduo. Agrupamentos de genes são denominados **cromossomas**, podendo o genótipo de um indivíduo conter um (o mais habitual) ou vários cromossomas. Os valores possíveis para determinado gene são os seus **alelos**. A posição onde podemos encontrar um determinado gene no cromossoma é o seu **locus**. Esta nomenclatura (tomada por empréstimo à biologia) era originalmente específica de um grupo particular de algoritmos evolucionários, os algoritmos genéticos, tendo-se o seu uso entretanto generalizado.

- Espaço do problema - O espaço do problema $\mathbb{X}$ inclui todas as soluções candidatas x para o problema. Normalmente os diversos x são referidos como sendo soluções candidatas, soluções possíveis ou simplesmente soluções, devendo-se distinguir entre estas denominações e o termo solução óptima. Assim uma solução é

qualquer elemento x pertencente a $\mathbb{X}$ enquanto que uma solução óptima x^* é uma solução para o problema de optimização, i.e. corresponderá a um mínimo ou máximo da função objectivo. O conjunto $\mathbb{S} \subseteq \mathbb{X}$ contém todas as soluções óptimas. Um mecanismo de mapeamento permite descodificar cada $g \in \mathbb{G}$ numa solução possível $x \in \mathbb{X}$. Em alguns tipos de problemas, por exemplo de optimização em $\mathbb{R}^n$, $\mathbb{G}$ e $\mathbb{X}$ podem coincidir.

- Espaço dos objectivos e espaço do desempenho - No caso geral do problema de optimização temos n funções objectivo $f_i(x)$ que mapeiam cada solução x num vector de reais $\mathbf{y}$. O conjunto destes vectores constitui o espaço de objectivos $\mathbb{Y} \subseteq \mathbb{R}^n$. Ainda no caso geral do problema multi-objectivo, é necessário haver um processo que permita comparar e/ou ordenar as soluções em função dos valores do vector $\mathbb{Y}$. Este processo pode consistir na atribuição de um valor de desempenho v que faça corresponder a cada vector um valor escalar (por exemplo utilizando uma média pesada dos y_i, ou um critério de ordenação como a relação de Pareto), implicando a existência de um último espaço, o espaço do desempenho $\mathbb{V} \subseteq \mathbb{R}^+$, onde os vectores objectivo são mapeados. É esta última medida v que é considerada o desempenho duma solução, desempenho este que frequentemente é relativo (numa ordenação, por exemplo), só fazendo sentido no contexto da população actual.

- População - Ao longo do processo evolutivo todos os algoritmos incluídos na área da computação evolucionária mantêm um vector $\mathbf{p}$ de indivíduos denominado população (ou, em alguns casos, enxame). No caso mais simples, um indivíduo, p_i, pode armazenar apenas o genótipo, g_i, da solução correspondente. Frequentemente, no entanto, pode ser útil armazenar o fenótipo, x_i, do indivíduo, ou outra informação, como por exemplo a sua avaliação, v_i. Nestes casos, cada indivíduo corresponde, respectivamente, a um tuplo $\langle g, x\rangle$ ou $\langle g, x, v\rangle$. Nestas representações do indivíduo usamos a notação $p_i.g, p_i.x$ e $p_i.v$ para aceder, respectivamente, ao seu fenótipo, genótipo e desempenho.

Para uma abordagem evolucionária poder ser aplicada a um problema de optimização, torna-se necessário definir os diversos operadores que permitem o mapeamento entre os diversos espaços e conjuntos atrás referidos, bem como a manipulação dos seus elementos. Embora a implementação, denominação e mesmo organização destes operadores possa variar, as funcionalidades a eles associadas podem ser encontradas na generalidade destes algoritmos:

- inicia() - Este operador devolve a primeira população $\mathbf{p}^0$, a chamada população inicial. A estratégia mais comum consiste em criar esta população por amostragem uniforme do espaço de procura $\mathbb{G}$, normalmente criando soluções de forma

aleatória. Convém no entanto referir que também é comum utilizar conhecimento específico sobre o problema em causa para escolher soluções de área mais promissoras dentro do espaço de procura. Noutros casos, boas soluções já conhecidas (e.g. encontradas por outros algoritmos de optimização) podem ser incluídas nesta primeira população, também assim melhorando a sua qualidade inicial.

- mapeia(p_i) - O operador mapeia recebe um indivíduo p_i e implementa o mecanismo de mapeamento do espaço de procura $\mathbb{G}$ para o espaço do problema $\mathbb{X}$. Assim, este operador constrói, a partir do genótipo $p_i.g$ de um indíviduo (i.e. a representação do indivíduo no espaço de procura), o seu fenótipo $p_i.x$, o qual deverá corresponder a uma solução possível para o problema, ou seja um ponto do espaço $\mathbb{X}$.

- Funções objectivo $f_i(x)$ - Funções que medem os diversos aspectos do problema que devem ser optimizados. Estas funções permitem mapear cada fenótipo x do espaço do problema $\mathbb{X}$ num vector de reais $\mathbf{y} \in \mathbb{Y} \subseteq \mathbb{R}^n$, o chamado espaço dos objectivos. Refira-se de novo que, no caso mais simples (e mais comum), apenas existe uma função objectivo $f(x)$ a optimizar.

- avalia($\mathbf{f}, \mathbf{p}$) - Este operador calcula uma medida de desempenho $p_i.v$ para cada indivíduo p_i com base nos valores do correspondente vector $\mathbf{y}$. Este valor avalia a qualidade do indivíduo quando comparado com o restante da população, permitindo assim decidir quão prioritária é a criação e a exploração dos seus descendentes. O operador avalia(p) tipicamente utiliza o operador mapeia(p_i), quando necessário, para produzir o fenótipo $p_i.x$ do indivíduo a partir do seu genótipo $p_i.g$. Posteriormente, o vector $\mathbf{y}$ é calculado utilizando as funções $f_j(x)$, i.e. $y_j = f_j(p_i.x)$. Como já foi referido, as estratégias para o cálculo de $p_i.v$, variam substancialmente, podendo ir de $p_i.v = f(p_i.x)$, quando há apenas uma única função objectivo f, até à utilização de critérios de ordenação complexos, na optimização multi-critério, passando também pela simples soma pesada dos n factores em avaliação, i.e. $p_i.v = \sum_{j=1}^{n} w_j f_j(p_i.x)$.

- selecciona($\mathbf{p}$) - A função deste operador é determinar, em função da avaliação dos indivíduos na população actual $\mathbf{p}$, quais os indivíduos que produzirão descendentes que poderão passar para a próxima geração, i.e. quais os indivíduos a que serão aplicados os operadores de variação, produzindo novos pontos a explorar no espaço de procura. Tipicamente, o operador devolve uma população temporária $\mathbf{q}$ contendo os indivíduos a reproduzir. Uma estratégia comum consiste em fazer $|\mathbf{q}| = |\mathbf{p}|$, mas garantindo que o número de cópias de um indivíduo p_i ocupa uma fração de $\mathbf{q}$ proporcional ao quociente da divisão do seu desempenho pelo desempenho total da população, $p_i.v / \sum_{j=1}^{m} p_j.v$. Isto significa, por exemplo, que um indivíduo responsável por 50% do desempenho da população numa dada geração, ocupará (aproximadamente) metade da população intermédia, podendo

assim gerar metade dos descendentes da geração seguinte. Mais uma vez, é de referir a existência de muitas outras estratégias de selecção, havendo ainda algoritmos onde esta está implícita, em vez de ser utilizado um operador de selecção explícito.

- varia($\mathbf{q}$) - Este operador implementa o mecanismo de variação essencial tanto na evolução por selecção natural como na computação evolucionária. Geralmente consiste na aplicação de um ou mais operadores de variação aos elementos da população temporária, $\mathbf{q}$, já selecionada a partir da população principal, $\mathbf{p}$, criando uma nova população de descendentes $\mathbf{r}$. Os operadores de variação, são tipicamente operadores unários ou binários (ou, em alguns casos, ternários), agindo sobre o genoma de um ou dois (ou três) indivíduos, denominados progenitores, e produzindo, tipicamente, um ou dois novos indivíduos, os descendentes. São estes operadores os únicos responsáveis pela criação dos pontos a explorar no espaço de procura, convindo, portanto, que a sua acção guie a procura na direcção das zonas mais promissoras do espaço, de forma ao óptimo ser eventualmente encontrado. Todos os operadores de variação se baseiam numa qualquer heurística de exploração, por exemplo "procurar na vizinhança de uma boa solução", de onde resulta uma estratégia de exploração de $\mathbb{G}$. Embora exista um número elevado de operadores na bibliografia, estes podem geralmente ser separadas em dois grandes grupos, conforme a abordagem que sigam para a criação de novos genomas envolva a recombinação, ou a mutação dos genomas originais. Tipicamente, o mecanismo de variação de uma abordagem evolucionária, inclui a aplicação de um ou mais operadores de cada grupo. Nos itens seguintes descrevemos um operador genérico de recombinação e um operador genérico de mutação.

- recombina($\mathbf{x_1}, \mathbf{x_2}$) - Os operadores de recombinação agem sobre dois ou mais genomas dos indivíduos progenitores dando origem a um ou mais genomas descendentes que resultam, como o nome indica, da recombinação dos genes dos progenitores. Estes operadores geram assim novos pontos de procura baseados numa heurística. Segundo esta, ao combinarmos em novas configurações, características de indivíduos sujeitos a um processo de selecção, e, portanto características elas próprias seleccionadas pela sua utilidade em ajudarem o indivíduo a sobreviver, obteremos novos indivíduos, também eles promissores, i.e. com valores de avaliação elevados. Os operadores de recombinação, tipicamente, não criam novos indivíduos muito distantes do sub-espaço de $\mathbb{G}$ que inclui todos os genomas da população $\mathbf{p}$. O seu ponto forte é, sobretudo, a procura local em torno da população actual e não a introdução de diversidade nessa mesma população.

- muta($\mathbf{x}$) - Uma abordagem evolucionária, baseada apenas em operadores de recombinação, pode estar sujeita a severas limitações em termos de procura, caso um determinado gene, necessário para a solução original, se extinga no material

genético da população. Para evitar estes (e outros) problemas, é normal utilizar numa abordagem evolucionária, além dos operadores de recombinação, operadores de mutação. Estes caracterizam-se por agirem sobre apenas um genoma base, criando também um descendente, ao alterar de forma aleatória um ou mais genes do genoma original. Os operadores de mutação complementam os operadores de recombinação, ao permitirem a introdução de diversidade no material genético da população. Garantem assim que todo o espaço $\mathbb{G}$, ou pelo menos um sub-espaço alargado deste, continua acessível ao algoritmo de optimização, mesmo quando a população está concentrada numa pequena região, por exemplo em torno de um óptimo local.

- substitui($\mathbf{p}, \mathbf{r}$) - Após os mecanismos de variação terem sido aplicados, dando origem a uma população de descendentes $\mathbf{r}$, é necessário criar uma nova população principal, a partir da população de progenitores, $\mathbf{p}$, e dos seus descendentes $\mathbf{r}$. O operador de substituição tem a missão de implementar uma das estratégias de substituição, das quais a substituição geracional, na qual toda a população de progenitores é substituída pelas seus descendentes, é talvez a mais comum. Outra estratégia popular consiste em construir a nova população de tamanho n com os n melhores indivíduos retirados do conjunto de n indivíduos originais mais os seus m descendentes. Frequentemente este operador inclui mecanismos de elitismo, os quais garantem que, independentemente da estratégia de substituição seguida, uma percentagem dos melhores indivíduos da população anterior é sempre copiada para a nova população.

- termina($t, \mathbf{p}$) - Este é o último operador necessário à implementação de um algoritmo evolucionário, e basicamente determina quando o algoritmo de optimização deve parar. Como já foi dito, uma abordagem evolucionária não garante que encontra o óptimo global, logo são necessários critérios que determinem quando o algoritmo deverá terminar. Geralmente estes estão relacionados com os recursos computacionais consumidos (número limite de gerações t, número de indivíduos avaliados...), com o desempenho do melhor indivíduo encontrado (teste de optimalidade, obtenção de um desempenho objectivo), ou com o estado de convergência do algoritmo.

Com todos estes elementos definidos é então possível descrever um algoritmo evolucionário genérico (Algoritmo 3.1). O algoritmo começa por inicializar o contador de gerações t a 0, criando e avaliando de seguida a população inicial $\mathbf{p}^0$, com recurso, respectivamente, aos operadores inicia() e avalia($\mathbf{p}^0$). O algoritmo evolucionário é um algoritmo iterativo que vai processar uma população de indivíduos $\mathbf{p}^t$ a cada geração t, até que a condição de paragem termina($t, \mathbf{p}^t$) seja verdadeira.

Algoritmo 3.1 Algoritmo evolucionário genérico

$t \leftarrow 0$
$\mathbf{p}^t \leftarrow$ inicia()
avalia($\mathbf{p}^t$)
enquanto não termina($t, \mathbf{p}^t$) **faz**
  $\mathbf{q}^t \leftarrow$ selecciona($\mathbf{p}^t$)
  $\mathbf{r}^t \leftarrow$ varia($\mathbf{q}^t$)
  $\mathbf{p}^{t+1} \leftarrow$ substitui($\mathbf{p}^t, \mathbf{r}^t$)
  avalia($\mathbf{p}^{t+1}$)
  $t \leftarrow t + 1$
fim enquanto

O ciclo principal do algoritmo inicia-se com a criação de uma população de reprodução temporária, r, devolvida pelo operador selecciona($\mathbf{p}$), onde os indivíduos deverão ter um número de cópias que reflicta o seu desempenho, i.e., estatisticamente, quanto melhor for um indivíduo, mais cópias deverá ter nesta população de reprodução. As cópias dos indivíduos em $\mathbf{r}$ serão então utilizadas pelo operador de variação, varia($\mathbf{q}$), como argumentos para os operadores de recombinação e mutação, produzindo assim uma nova população $\mathbf{r}$ de possíveis soluções descendentes. Esta população pode então ser utilizada para substituir $\mathbf{p}$ de forma completa ou parcial, conforme definido pela estratégia implementada por substitui($\mathbf{p}, \mathbf{r}$). Após a avaliação dos indivíduos da nova população, $\mathbf{p}$, resultante da etapa anterior do algoritmo, o contador de geração t é incrementado e dá-se início a nova iteração do algoritmo. É este processo iterativo que, à imagem da evolução por selecção natural, irá progressivamente produzir melhores indivíduos, através da selecção preferencial dos melhores e sua posterior variação. O algoritmo executa assim uma procura de cariz global no espaço de genomas $\mathbb{G}$, guiada pelo desempenho dos indivíduos, no sentido de optimizar as funções objectivo f_i.

O algoritmo evolucionário genérico apresentado aqui ilustra adequadamente os princípios base deste tipo de algoritmo de optimização. Na prática, no entanto, são necessárias implementações concretas que permitam resolver problemas reais. Nas últimas décadas têm surgido muitos algoritmos que, embora assentes em princípios comuns, divergem em vários aspectos, desde a representação base, até à própria metáfora inspiradora, passando pelos sempre críticos operadores de variação. Nas secções seguintes descreveremos, brevemente, algumas das abordagens mais populares e/ou com mais sucesso. Os algoritmos apresentados serão divididos em dois grupos: algoritmos evolucionários e algoritmos de inteligência de enxame.

3.3 Algoritmos Evolucionários

Os algoritmos evolucionários representam o maior grupo de algoritmos dentro da computação evolucionária. Distinguem-se de outros algoritmos da mesma área por assentarem firmemente na metáfora biológica da evolução, i.e., por procurarem implementar uma versão computacional da evolução por selecção natural. Por este facto, são estes algoritmos os mais fáceis de mapear no algoritmo evolucionário genérico apresentado anteriormente.

3.3.1 Algoritmos genéticos

Os algoritmos genéticos (do inglês *genetic alghorithms*) constituem, provavelmente, o grupo de algoritmos evolucionários de utilização mais generalizada. Propostos inicialmente por [Holland, 1975, 1992] e desenvolvidos, entre outros, por [Goldberg, 1989], a versão clássica do algoritmo genético (AG) assentava numa representação binária das soluções. Os operadores de variação são definidos para lidar com vectores binários, sendo normalmente utilizado um operador de recombinação, e um operador de mutação.

Nos algoritmos genéticos, a recombinação consiste no cruzamento de dois vectores progenitores a partir de um *locus* escolhido aleatoriamente em cada operação de recombinação, denominando-se por isso recombinação de um ponto de corte. A mutação, geralmente aplicada aos indivíduos resultantes da recombinação, implica a possibilidade de alteração de cada *bit* com uma probabilidade muito baixa. A selecção, originalmente, era feita de forma proporcional ao desempenho dos indivíduos e a substituição é, tipicamente, geracional. [Mitchell, 1996] e [Haupt and Haupt, 2004] apresentam boas introduções aos algoritmos genéticos, com o segundo texto a preocupar-se mais com a descrição de aplicações práticas.

É possível que a representação extremamente genérica em que os algoritmos genéticos se baseiam seja uma das razões da sua popularidade. Estando este de raiz preparado para evoluir sequências de *bits*, pode ser aplicado de forma directa a qualquer problema de optimização, combinatória ou numérica, desde que as suas soluções sejam passíveis de codificação binária, o que sabemos ser sempre verdade quando a optimização é feita com recurso a computadores digitais.

Esta generalidade do algoritmo genético, com uma distinção clara entre o genótipo de um indivíduo (codificação binária) e o seu fenótipo (solução descodificada) tem, no entanto, os seus custos, já que pode implicar um mapeamento complexo entre genótipo e fenótipo, dispendioso do ponto de vista computacional, além de uma difícil compreensão da forma como as operações de procura em $\mathbb{G}$ se reflectem em $\mathbb{X}$. Este

último problema pode, por exemplo, dificultar o desenvolvimento de operadores de procura (variação) mais eficientes.

A constatação das dificuldades anteriores, aliada ao corpo muito substancial de investigação feita nesta área nas últimas décadas, tem conduzido a uma progressiva especialização em termos de aplicação do algoritmo genético. Foram desenvolvidas variantes, especialmente em termos de operadores de recombinação e mutação, para representações típicas (por exemplo, baseadas em números inteiros, reais, permutações, regras se-então e outras estruturas de dados). Actualmente, a aplicação de um algoritmo genético a um novo problema, começa pela escolha da representação que melhor se adequa ao problema em causa, permitindo uma fácil codificação do fenótipo, seguida pela identificação de operadores padrão para essa representação. Esta disponibilidade de representações, e operadores, para diferentes domínios, torna os algoritmos genéticos uma das abordagens evolucionárias contemporâneas mais flexíveis em termos de aplicação, podendo, por exemplo, serem utilizados com a mesma facilidade em problemas de optimização numérica e problemas de optimização combinatória.

3.3.2 Estratégias evolutivas

O segundo grupo de algoritmos que iremos descrever engloba as denominadas estratégias evolutivas (do alemão *evolutionsstrategie*). Historicamente, estes algoritmos têm tido um desenvolvimento paralelo ao dos algoritmos genéticos, com muita da investigação de base a ter sido feita por grupos alemães nas décadas de 70 e 80 do século passado [Rechenberg, 1973; Schwefel, 1974, 1981]. Hoje em dia, as estratégias evolutivas, são algoritmos de optimização global muito populares, especializados em optimização numérica em $\mathbb{R}^n$ [Beyer and Schwefel, 2002].

Ao contrário dos algoritmos genéticos, as estratégias evolutivas não se baseiam numa representação genérica como os vectores binários. Tendo sido, originalmente, desenvolvidas para optimizar vectores de parâmetros reais, desde a sua concepção que a representação de um indivíduo está orientada para a optimização numérica. Um aspecto essencial, em que desde logo se distinguem dos algoritmos genéticos, consiste em não haver distinção entre o fenótipo e o genótipo do indivíduo, logo $\mathbb{G}$ e $\mathbb{X}$ coincidem. Um indivíduo é simplesmente representado por um tuplo $\langle \mathbf{x}, \sigma \rangle$, no qual $\mathbf{x}$ e σ são vectores de números reais, o primeiro correspondendo a uma possível solução para o problema de optimização, e o segundo armazenando um valor de desvio padrão σ_i para cada parâmetro x_i.

A variação é realizada normalmente por um operador de mutação, o qual funciona através da adição a cada x_i de um valor aleatório, com distribuição normal de média 0 e desvio padrão σ_i. Note-se a importância da inclusão dos desvios padrão na repre-

sentação dos indivíduos. Estes são perturbados aleatoriamente antes da mutação do indivíduo, o que quer dizer que variam, tanto de parâmetro para parâmetro, como ao longo da execução do algoritmo evolucionário. Como desvios padrão que produzam indivíduos com elevado desempenho são transmitidos com maior probabilidade à geração seguinte, estes algoritmos evoluem não só a solução, mas também um dos operadores de variação, através da optimização dos parâmetros que o controlam.

A variação por recombinação é menos utilizada do que o operador de mutação. À imagem do que acontece com os operadores de recombinação utilizados nos algoritmos genéticos para representações reais, a recombinação podes ser discreta ou linear. No caso discreto, os descendentes são criados recombinando directamente pares parâmetro/desvio padrão de *locus* correspondentes, retirados dos progenitores. Já no caso linear, como o nome indica, os genes resultantes são uma combinação linear dos genes dos progenitores.

A estratégia evolutiva original trabalhava com uma população de tamanho 1, gerando um descendente que substituía o progenitor caso lhe fosse superior. Esta estratégia era denominada $(1 + 1)$-ES, sendo que a expressão $(1 + 1)$ indicava uma população temporária constituída por um progenitor mais um descendente, de entre os quais o melhor passava para a próxima geração. A generalização seguinte era denominada $(1 + \lambda)$-ES, com um progenitor a dar origem a λ descendentes e o melhor de todos os indivíduos a passar para a próxima geração. Note-se que estas primeiras estratégias eram mais algoritmos trepa-colinas estocásticos do que algoritmos evolucionários, já que não se baseavam em populações com múltiplos indivíduos.

Nas versões actuais do algoritmo, os mecanismos de selecção/substituição mais utilizados seguem estratégias (μ, λ)-ES ou $(\mu + \lambda)$-ES. Na primeira, de uma população de μ indivíduos são gerados, por variação, λ descendentes, dos quais os μ melhores irão substituir os progenitores. Neste caso temos de ter obrigatoriamente $\lambda > \mu$ para haver pressão de selecção sobre a população. A segunda estratégia é uma generalização mais próxima das estratégias originais, com μ indivíduos a darem também origem a λ descendentes (embora λ possa ser menor do que μ), mas com os μ indivíduos da próxima geração a serem os melhores seleccionados de entre a reunião de progenitores e descendentes.

Actualmente, é utilizada a notação $(\mu/\rho+, \lambda)$-ES para descrever uma determinada estratégia evolutiva, onde μ é o tamanho da população, λ é o número de descendentes criados em cada nova geração e ρ indica o número de progenitores utilizados em cada operação de recombinação. $+$ e , são utilizados de forma exclusiva, implicando, respectivamente, que a nova população é seleccionada de entre os progenitores e descendentes, ou apenas de dentre os λ descendentes.

Apesar de haver claras diferenças entre as estratégias evolutivas e os algoritmos genéti-

cos em termos dos operadores de variação e selecção (e da própria representação base), o elemento diferenciador de maior relevo é sem dúvida a capacidade de auto-adaptação do algoritmo a um problema específico, através da já mencionada co-evolução dos desvios padrão que controlam o operador de mutação. Versões mais recentes, como a estratégia evolutiva com adaptação da matriz de co-variância (do inglês *covariance matrix adaptation evolution strategy (CMA-ES)*) (EE-AMC) [Hansen and Kern, 2004], juntam ainda à representação uma matriz de co-variância entre os parâmetros, a qual é ela própria evoluída, de maneira a permitir detectar e explorar mais efectivamente as direcções do espaço de procura que resultam em maiores incrementos de desempenho.

3.3.3 Programação evolucionária

Actualmente, dentro da área da computação evolucionária, os algoritmos genéticos e as estratégias evolutivas - e a multitude das suas variantes - constituem claramente as abordagens dominantes, tanto em termos de utilização prática como de investigação produzida. Do ponto de vista histórico, convém, no entanto, referir ainda uma terceira abordagem, a programação evolucionária (do inglês *evolutionary programming*) [Fogel et al., 1966; Fogel, 1999], já que, simultaneamente com as outras duas já referidas, foi das primeiras a ser introduzida.

No seu formato original, a programação evolucionária, utilizava representações baseadas em estruturas de dados, e.g. autómatos finitos, para evoluir mecanismos de decisão inteligentes. Em termos de variação, esta abordagem centrava-se sobretudo no mecanismo de mutação, sendo que, inclusivamente, na sua aplicação a domínios reais, é bastante semelhante às variantes iniciais das estratégias evolutivas. Esta semelhança, possivelmente aliada à fácil utilização de algoritmos genéticos com a representação original, conduziu ao progressivo desvanecimento desta terceira linha de abordagem na computação evolucionária.

3.3.4 Programação genética

Além das três abordagens historicamente mais relevantes, referidas nas secções anteriores, existe na bibliografia da área um número significativo de variantes e abordagens alternativas impossível de tratar nesta introdução. Limitaremos-nos a apresentar aqui uma das variações mais significativas, quer pela importância do seu domínio de aplicação, quer pela disseminação da sua utilização - a programação genética (do inglês *genetic programming*) [Banzhaf et al., 1997; Koza, 2003; Langdon and Poli, 2002].

A programação genética (PG) foi introduzida por [Koza, 1992, 1994] como uma variante do algoritmo genético para a evolução de algoritmos e programas de computador.

Consiste numa variação do algoritmo genético especificamente desenvolvida para trabalhar com uma representação assente em árvores sintáticas, utilizadas para descrever os referidos algoritmos/programas, bem como outras representações relacionadas, e.g. funções matemáticas ou árvores de decisão.

Na programação genética os indivíduos são avaliados pelo desempenho que demonstram na resolução de determinado problema, realizada pela interpretação da correspondente árvore sintáctica. Cada árvore é construída a partir de um conjunto de nós pré-definidos e dependentes do problema. Estes estão divididos num conjunto de terminais, que apenas podem ser utilizados como folhas da árvore (por exemplo constantes e variáveis) e num conjunto de funções, as quais constituem os nós interiores da árvore e podem receber como argumentos os terminais, ou os valores de retorno de outras funções.

Sendo esta abordagem uma variação do algoritmo genético, é normal que vários elementos, como a estrutura geral e os mecanismos de selecção utilizados, sejam semelhantes. A maior diferença surge em termos de operadores de recombinação, já que, na programação genética, estes estão especialmente adaptados a lidar com as árvores sintácticas que constituem a base da representação. Assim, a recombinação consiste na troca entre os dois progenitores de sub-árvores seleccionadas aleatoriamente. Como pode haver troca de sub-árvores de qualquer posição dos indivíduos, torna-se improvável a indisponibilidade de determinado gene num qualquer *locus* do indíviduo, e a mutação não é assim tão essencial como em outros algoritmos genéticos. Quando utilizada, consiste simplesmente na substituição de uma sub-árvore do indivíduo por outra criada aleatoriamente.

A programação genética tem obtido substancial sucesso em muitas tarefas, frequentemente atingindo resultados semelhantes, ou superiores aos de peritos humanos em tarefas complicadas [Koza, 2003; Poli et al., 2008]. Como qualquer outro algoritmo evolucionário, a sua aplicação apresenta também alguns desafios específicos, nomeadamente, neste caso, ligados à representação utilizada. Com efeito, o facto desta ser de tamanho variável, permite o crescimento desmesurado dos indivíduos (denominado *bloating*) ao longo da execução do algoritmo [Poli, 2003]. Este facto é não só pernicioso do ponto de vista dos custos computacionais relacionados com a avaliação de indivíduos muito grandes, como introduz complexidade adicional na aplicação dos operadores, e manutenção da diversidade da população.

3.4 Inteligência de Enxame

O segundo grande grupo de abordagens, frequentemente englobado na computação evolucionária, é constituído pelos chamados algoritmos de enxame, os quais são o objecto de uma área de estudo denominada inteligência de enxame [Bonabeau et al., 1999;

Eberhart et al., 2001]. Estes algoritmos vão geralmente buscar inspiração ao comportamento cooperativo de grupos organizados de animais, como bandos de aves [Kennedy and Eberhart, 1995], colónias de formigas [Dorigo, 1992] ou enxames de abelhas [Karaboga and Akay, 2009]. Assim, enquanto nos algoritmos evolucionários o ênfase é colocado na competição entre os indivíduos, nos algoritmos de enxame é mais frequente encontrar instâncias em que é a colaboração descentralizada entre os elementos do bando, ou enxame, que é central ao mecanismo de procura ou optimização.

Apesar da diferente abordagem presente nos algoritmos de enxame, muitos destes, como por exemplo a optimização por enxames de partículas (do inglês *particle swarm optimization*) (OEP) descrita na próxima secção, podem ainda assim ser mapeados no algoritmo evolucionário genérico que descrevemos, sendo possível identificar mecanismos de variação, substituição ou de selecção. Estes paralelismos, bem como a utilização de populações de soluções, justificam a inclusão dos algoritmos de inteligência de enxame na área da computação evolucionária, embora existam algoritmos de enxame baseados em metáforas completamente diferentes, como a optimização por colónias de formigas (do inglês *ant colony optimization*), que também descreveremos numa das próximas secções.

3.4.1 Optimização por enxames de partículas

A optimização por enxames de partículas engloba provavelmente o conjunto de algoritmos mais conhecidos da área da inteligência de enxame. O algoritmo original foi proposto por [Kennedy and Eberhart, 1995] e assentava numa população de agentes (chamados partículas) que imitavam o comportamento de bandos de aves em voo. Os agentes deslocavam-se num espaço de procura, mantendo uma memória da melhor posição encontrada. As regras de actualização da posição das partículas têm em conta, por um lado essa memória, e por outro lado a melhor posição encontrada pela melhor partícula vizinha. Esta vizinhança tanto pode ser global (i.e. cada partícula é atraída para a melhor partícula do enxame (o líder do "bando"), como obedecer a qualquer regra mais restritiva. Desta forma, o algoritmo tem uma componente social, no sentido em que cada agente oscila entre a sua convicção de onde a solução se encontra, e a pressão para seguir o comportamento do bando.

Em termos práticos, o algoritmo padrão de OEP é um algoritmo de optimização global em $\mathbb{R}^n$, utilizando uma representação assente em três vectores reais de tamanho n. Uma partícula i ocupa uma posição no espaço de procura representada por $\mathbf{x_i}$, a qual representa uma potencial solução para o problema de optimização. Adicionalmente, é também mantido um vector $\mathbf{p_i}$, o qual armazena a melhor solução que a partícula i encontrou durante a execução do algoritmo até à iteração corrente. Finalmente, um vector $\mathbf{v_i}$ guarda a velocidade com que a partícula se desloca no espaço de procura.

A imitação da deslocação de um ave quando integrada num bando, aliada à representação anterior, permite criar uma heurística de exploração do espaço de procura específica dos algoritmos de optimização por enxame de partículas, que basicamente consiste numa estratégia de exploração em que as partículas exploram regiões promissoras do espaço de procura ao oscilar entre as melhores soluções que encontraram, e a melhor solução encontrada pelo enxame na sua totalidade (ou um subconjunto deste). Para implementar este comportamento, o algoritmo padrão mais usado actualiza a posição e velocidade de cada partícula i utilizando as seguintes equações [Clerc, 2010]:

$$\mathbf{v}_i^{t+1} = \chi(\mathbf{v}_i^t + \mathbf{u}(0,\phi_1) \otimes (\mathbf{p}_i^t - \mathbf{x}_i^t) + \mathbf{u}(0,\phi_2) \times (\mathbf{p}_g^t - \mathbf{x}_i^t)) \tag{3.1}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^t \tag{3.2}$$

$$\text{com } \phi = \phi_1 + \phi_2 > 4, \tag{3.3}$$

$$\text{e } \chi = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}} \tag{3.4}$$

Nas equações de actualização 3.1, a velocidade depende tanto da distância $(\mathbf{p}_i^t - \mathbf{x}_i^t)$ entre a partícula e a sua melhor posição em memória, como da distância $(\mathbf{p}_g^t - \mathbf{x}_i^t)$ entre a partícula e a melhor posição $\mathbf{p}_g^t$ do enxame, ou vizinhança. A cada iteração dos algoritmos são gerados novos vectores de números aleatórios $\mathbf{u}(0,\phi_1)$ e $\mathbf{u}(0,\phi_2)$ com distribuição uniforme entre os dois parâmetros, garantido assim o movimento oscilante das partículas. $\otimes$ representa o produto componente a componente dos vectores que recebe como argumentos. A escolha dos ϕ_1 e ϕ_2 e χ obedecendo às condições 3.3 e 3.4 garante a convergência do algoritmo, de acordo com [Clerc, 2010]. Valores típicos são $\phi_1 = \phi_2 = 2.05$ e $\chi \approx 0.7298$.

Como já foi referido, é possível encontrar paralelismo entre este algoritmo e os algoritmos evolucionários. Mais especificamente, o cálculo da velocidade pode ser visto como uma forma de recombinação entre três indivíduos (o próprio, a sua memória e a memória do melhor vizinho). A manutenção de uma população de memória, onde apenas as melhores posições encontradas por cada indivíduo são armazenadas, constitui uma forma de selecção. Já em relação à mutação, não é possível identificar nenhum mecanismo com o mesmo efeito, o que, inclusivamente, causa algumas dificuldades ao algoritmo, sobretudo em termos de manutenção de diversidade no enxame. Com efeito, o algoritmo base, após convergir para um óptimo local, tem dificuldade em continuar a exploração e encontrar outro óptimo, que se desejaria que fosse o global. Esta e outras limitações são tratadas nas muitas variantes e extensões existentes do algoritmo básico de OEP [Banks et al., 2007, 2008; Poli et al., 2007].

O algoritmo de optimização por enxames de partículas foi inicialmente proposto como um algoritmo de optimização em $\mathbb{R}^n$, e é neste domínio que é ainda extremamente

popular, devido a características que incluem, entre outras, a facilidade de implementação, a rapidez de convergência e as poucas exigências em termos computacionais. Existem também, no entanto, versões binárias do algoritmo, com a primeira a ser apresentada por [Kennedy and Eberhart, 1997]. Esta revelar-se-á importante para o trabalho apresentado nesta tese, sendo descrita num capítulo posterior.

3.4.2 Optimização por colónias de formigas

O último algoritmo descrito nesta introdução à computação evolucionária ilustra a utilização de outros paradigmas, fora da evolução por selecção natural, na criação de algoritmos que ainda assim são geralmente colocados dentro desta área. É também exemplo de um algoritmo de optimização global, que, ao contrário dos apresentados anteriormente (com a possível excepção dos algoritmos genéticos), tem aplicação mais natural em problemas de optimização combinatória do que em problemas de optimização numérica. Trata-se de um algoritmo de inteligência de enxame, denominado algoritmo de optimização por colónias de formigas (OCF) [Dorigo, 1992; Dorigo and Stützle, 2004; Dorigo and Blum, 2005], baseado num mecanismo de stigmergia [Dorigo et al., 2000; Garnier et al., 2007].

A stigmergia refere-se à coordenação indirecta entre agentes ou acções, através de marcas deixadas no ambiente. Estas marcas, ou alterações realizadas no ambiente, vão influenciar a próxima acção do agente que a realizou ou de outro que com ela interaja, levando assim a um comportamento coordenado, de forma distribuída, entre os diversos agentes. Note-se que a stigmergia pressupõe, tal como a selecção natural, a existência subjacente de uma população de indivíduos, ou, neste caso, agentes.

Um exemplo clássico de stigmergia na natureza envolve a procura de um caminho óptimo entre a saída de uma colónia de formigas e qualquer fonte de alimento. À medida que se deslocam, as formigas deixam no solo uma marca química, denominada feromona. Se imaginarmos que existem dois caminhos, de comprimento diferente, entre a saída da colónia e a fonte de alimento, e se estes caminhos forem inicialmente escolhidos de forma aleatória pelas formigas, aquelas que acidentalmente escolherem o caminho mais curto, terminam mais rapidamente o seu trajecto, depositando aí feromona mais rapidamente, o que causa um ligeiro aumento da concentração desta em relação ao caminho mais longo.

É sabido que as formigas, quando presentes com a escolha entre dois caminhos, têm maior probabilidade de escolher o caminho com maior concentração de feromona. Consequentemente, o caminho mais curto, com uma pequena vantagem inicial em termos de concentração de feromona, irá ter maior probabilidade de atrair mais formigas, que, por sua vez, acabarão o trajecto mais rapidamente, aumentando ainda mais a diferença

de feromona entre os dois caminhos. O que se observa na realidade é que, enquanto de início 50% das formigas escolhe cada um dos caminhos, rapidamente começa a haver uma maior proporção a seguir pelo caminho mais curto, e, passado algum tempo, já raramente uma formiga segue pelo caminho mais longo. A experiência tem iguais resultados com caminhos de complexidade arbitrária.

O algoritmo de optimização por colónias de formigas imita o processo descrito acima para resolver problemas de optimização combinatória, mais especificamente problemas de procura de um caminho óptimo num grafo, como o problema do caixeiro viajante. Para tal, uma população de agentes ("formigas"computacionais) vai modificando iterativamente um valor numérico, o qual simula a concentração de feromona, associado a cada arco do grafo. Também no algoritmo os agentes têm maior probabilidade de seguir por arcos com maior concentração de feromona, aumentando, por sua vez, essa mesma concentração, numa tradução mais ou menos directa do mecanismo stigmergético observado na natureza.

Refira-se ainda que, embora as abordagens mais populares continuem a ser as que, tal como a proposta original, são vocacionadas para problemas de optimização combinatória, também no caso da optimização por colónias de formigas já existem extensões que permitem utilizar estes algoritmos em problemas de optimização numérica [Dréo and Siarry, 2002; Socha and Dorigo, 2008].

Capítulo 4

Computação Evolucionária em Tarefas de PLN

Neste capítulo fazemos um apanhado das contribuições da área da computação evolucionária em tarefas de processamento de língua natural (PLN). As referências estão agrupadas por temas representativos de tarefas de processamento de língua natural, sendo que, em cada grupo, são apresentados os vários trabalhos encontradas na pesquisa bibliográfica realizada, não havendo aqui separação pelo tipo de algoritmo de inspiração biológica utilizado.

4.1 Segmentação de Palavras

A segmentação de palavras é uma sub-tarefa importante do processamento de língua natural, com um vasto leque de aplicações que vão desde a hifenização de palavras à conversão texto-fala. Em [Kazakov et al., 1998] é apresentada uma abordagem híbrida ao problema da segmentação de palavras, a qual combina técnicas de aprendizagem supervisionada e não-supervisionada, para geração de regras de segmentação, a partir de uma lista de palavras. Os autores utilizam um algoritmo genético para determinar a melhor segmentação de um conjunto de palavras, e usam-na como entrada de um algoritmo de indução de programação em lógica, para determinar um conjunto de regras de segmentação que possa ser usado em palavras desconhecidas.

O modelo de segmentação adoptado pelos autores limita o número de segmentos a dois: qualquer sub-cadeia à esquerda/direita da palavra é interpretada como prefixo/sufixo. O modelo baseia-se na "Teoria *Naive* da Morfologia"(do inglês *Naive Theory of Morphology*), TNM, definida pelos autores num trabalho anterior.

Assuma-se que, para um determinado conjunto de palavras, as posições dos limites

dos segmentos correspondem a um vector de inteiros, os quais variam entre zero e o tamanho da respectiva palavra. O conjunto de palavras, juntamente com um vector com estas características, introduz uma TNM. Uma TNM permite assim definir dois léxicos, definidos pelos conjuntos Prefixos e Sufixos, constituídos pelas correspondentes segmentações das palavras. Nestes conjuntos os prefixos (respectivamente os sufixos) são enumerados sem repetição.

A qualidade de uma TNM é estimada pela soma dos caracteres, $n = p+s$, no léxico dos prefixos (p) e sufixos (s). Quanto menor for o total, melhor é a TNM. O limite máximo, n_{max}, é dado pelo número w de caracteres de todas as palavras da lista de palavras. Este caso corresponde à situação em que nenhum prefixo, nem sufixo, é gerado mais do que uma vez pela teoria.

Com base nesta teoria, os autores definem um critério para efectuar a segmentação das palavras: entre o conjunto de TNMs, seleccione-se a que corresponde a um menor número de caracteres no correspondente par de léxicos. Este enviezamento é baseado na hipótese de que sub-cadeias compostas por morfemas reais, ocorrem nas palavras com uma frequência superior a qualquer outra sub-cadeia à esquerda, ou à direita, da palavra. Desta forma, uma TNM com um n pequeno, produzirá léxicos onde os "prefixos"e "sufixos"correspondem, frequentemente, a morfemas, ou combinações destes.

Os autores utilizam um algoritmo genético para evoluir a melhor teoria. Assim, cada indivíduo corresponde a um cromossoma, o qual é representado à custa de um vector de inteiros que define uma TNM. A função de desempenho usada mede a diferença $n_{max} - n$. Desta forma, a função nunca assumirá valores negativos e o algoritmo evoluirá no sentido da maximização deste valor.

O operador de mutação é usado de duas forma distintas: fazendo um deslocamento, em uma posição, do morfema, à esquerda, ou à direita; ou escolhendo, aleatoriamente, uma nova posição dentro do intervalo de posições possíveis. O operador de recombinação escolhido, foi o operador de recombinação com um ponto de corte. O melhor indivíduo, em cada geração, é mantido ao longo das gerações. Ao fim de um número pré-estabelecido de gerações, o algoritmo devolve o melhor indivíduo encontrado que corresponderá à melhor teoria observada, durante o processo de evolução.

Numa segunda fase, a lista de palavras segmentadas obtida pelo algoritmo genético, é usada como entrada de dois algoritmos de aprendizagem indutiva, nomeadamente os sistemas FOIDL e CLOG [Suresh Manandhar and Erjavec, 1998]. O resultado final é um programa em lógica, representado sob a forma de uma lista ordenada de regras (cláusulas), precedida de uma lista de excepções, representadas sob a forma de factos. Os autores afirmam que o resultado deverá ser capaz de efectuar a segmentação de palavras desconhecidas.

4.2 Marcação das PdD

Nesta secção apresentamos um conjunto de trabalhos que envolvem o uso de algoritmos evolucionários para resolver o problema da marcação das PdD das palavras de um texto.

Em [Lankhorst, 1995], os autores propõem um algoritmo genético para realizar a classificação automática de palavras, baseado apenas no número de observações de pares de palavras num texto. Não se trata de resolver, directamente, o problema de marcação das partes de discurso. O sistema apresentado, apenas agrupa um conjunto de palavras em n categorias, não fazendo a identificação destes grupos com partes do discurso concretas.

Os autores utilizam, como função de avaliação, uma medida de informação teórica sobre a frequência de dados, centrada no conceito de entropia. Assumindo que se pretende categorizar o conjunto de palavras $w = \{w_1, ..., w_n\}$ de n palavras em m categorias $V = \{v_1, ..., v_m\}(m < n)$, e considerando o alfabeto $A = \{1, ..., m\}$, os indivíduos da população são representados à custa de um cromossoma $C = \langle c_1, ..., c_n \rangle$ com $c_i \in A$. O valor de um determinado gene (alelo) define o número da categoria da respectiva palavra,

$$c_j = i \Leftrightarrow w_j \in v_i (1 \leq i \leq m, 1 \leq j \leq n) \tag{4.1}$$

ou de forma equivalente,

$$v_i = \{w_j \in w \mid c_j = i, 1 \leq j \leq n\} (1 \leq i \leq m) \tag{4.2}$$

Dada *a priori* a matriz N de frequência das palavras, a matriz M de frequência das categorias é calculada por,

$$M_{pq} = \sum_{w_i \in v_p} \sum_{w_j \in v_q} N_{ij} \tag{4.3}$$

Os autores realizaram experiências com dois tipos de representações: inteira (como apresentado em cima) e binária (cada categoria é representada por uma sequência de *bits*). As experiências permitiram concluir como mais adequada a representação inteira, já que foi a que conseguiu obter melhores resultados. Fizeram ainda uma comparação com os resultados obtidos em [Lankhorst and Moddemeijer, 1993], onde foi usado um método ganancioso para calcular a combinação óptima local das palavras, através da construção de uma árvore de agrupamentos, onde as palavras foram agrupadas usando como critério a perda mínima de informação mútua, até terem sido determinadas todas as categorias desejadas. Os resultados obtidos permitiram concluir que o algoritmo genético obtém melhor desempenho do que o método ganancioso, embora as diferenças não sejam muito significativas.

Os autores apresentam, como vantagens do sistema proposto, o facto da abordagem não depender de conhecimento linguístico, e não existirem restrições ao número de categorias lexicais usadas, podendo assim ser inferida uma categorização mais detalhada do que a tradicionalmente usada. Além disso, os agrupamentos não têm que ser puramente sintácticos, podendo incluir aspectos semânticos, já que são apenas determinados pela utilização das palavras classificadas. Palavras com a mesma classe gramatical, mas com significados distintos, são geralmente utilizadas em contextos diferentes, os quais são usados pelo algoritmo para determinar a categoria de cada palavra.

Como problemas são referidos o elevado tempo de processamento, e o facto do sistema não lidar bem com palavras ambíguas. As palavras são colocadas numa única categoria, contudo sabemos que existem palavras que podem ser usadas com diferentes funções, podendo, por isso, pertencer a diferentes classes gramaticais. Para resolver o problema, os autores sugerem usar uma representação que permita que cada palavra tenha um determinado grau de associação com cada uma das categorias possíveis, em vez de apenas um número, a designar a associação a uma única categoria. O número de observações dos pares de palavras poderia, assim, ser distribuído pelas categorias de forma proporcional ao (relativo) grau de associação.

O sistema proposto em [Araujo, 2002*b*] usa um algoritmo evolucionário para resolver o problema da marcação das PdD. Para classificar as palavras de um texto, o marcador deverá ser executado, isoladamente, para cada uma das frases do texto correspondente. No algoritmo proposto, cada indivíduo da população representa uma solução candidata para o problema da marcação das PdD, para a frase que recebe como entrada. Cada indivíduo é constituído por um cromossoma, composto por tantos genes quantas as palavras da frase que se pretende marcar. Cada gene representa, por sua vez, uma classe gramatical candidata para a palavra na posição homologa na frase: o primeiro gene propõe uma etiqueta para a primeira palavra, o segundo para a segunda, e assim sucessivamente. A estrutura escolhida para representar um gene permite guardar a etiqueta candidata, assim como informação relacionada com o contexto.

A informação relacionada com o contexto é obtida numa fase inicial de treino do sistema, sendo armazenada sob a forma de uma tabela, designada tabela de treino. Esta tabela guarda, para todas as partes de discurso presentes no *corpus* de treino, o número de vezes em que esta surge num determinado contexto. Os autores utilizam contextos que contemplam as partes do discurso que surgem antes, e depois de uma determinada palavra. O número de etiquetas a considerar antes e depois, define o tamanho do contexto. Os autores conduziram experiências para diferentes tamanhos de contexto.

A medida de desempenho de um determinado cromossoma é calculada através de uma função de avaliação que corresponde à soma dos resultados da avaliação de cada um dos genes que o compõem. Assim, a avaliação de um cromossoma, implica a avaliação prévia de cada um dos seus genes. Um gene é avaliado com base na informação estatística

recolhida, e armazenada, na tabela de treino. O valor obtido traduz-se numa estimativa, da exactidão da parte do discurso proposta pelo gene, para uma determinada palavra da frase.

O processo de evolução tem, assim, a responsabilidade de escolher, para uma determinada frase, a sequência de partes do discurso que maximiza a probabilidade total, tendo como base os dados estatísticos previamente obtidos.

Os autores concluem que o *corpus* utilizado para o treino do sistema tem uma grande influência no seu desempenho, sendo que o ideal será a utilização de um *corpus* que se revele uma boa amostra da língua. No entanto, na maioria das vezes, os *corpus* disponíveis são específicos de um determinado domínio. Os autores usaram no trabalho experimental o *corpus* de *Brown*. Foram realizadas diferentes experiências de forma a estudar os principais aspectos que influenciam o sistema: o tamanho e forma dos contextos utilizados para a tabela de treino, o tamanho do *corpus* de treino e os parâmetros do algoritmo evolucionário.

Relativamente ao tamanho do contexto, os autores concluíram que contextos de tamanho superior a três são irrelevantes, sendo que na maioria dos casos, um tamanho igual a um revela-se suficiente. De salientar que o algoritmo evolucionário recebe como entrada uma tabela de treino específica definida para um determinado contexto. Ou seja para contextos que contemplam um número pré-estabelecido de palavras à direita e/ou à esquerda da palavra alvo.

No que diz respeito ao tamanho do conjunto de treino, os autores concluem que apenas um aumento significativo da exactidão do sistema poderá justificar a utilização de um *corpus* de treino maior. Isto porque o tamanho da tabela de contexto influencia bastante a velocidade do processo evolutivo, i.e. quanto maior é a tabela, mais lento se torna o processo.

Em [Araujo, 2003] encontramos a aplicação de um algoritmo evolucionário desalinhado (do inglês *messy evolutionary algorithm*) ao problema da marcação das PdD. Neste trabalho, a autora pretende investigar a validade da assunção de que a escolha da classe gramatical para uma palavra (de entre as classes possíveis para a mesma) apenas depende das classes das palavras que a rodeiam. Como já vimos, esta é a ideia por trás dos marcadores baseados em métodos estatísticos. A autora comparou os resultados obtidos pelo algoritmo evolucionário desalinhado com os obtidos com um algoritmo evolucionário clássico, seguindo a abordagem apresentada em [Araujo, 2002*b*]. Os resultados obtidos mostram que o algoritmo evolucionário desalinhado consegue, sistematicamente, resultados ligeiramente melhores que os alcançados com o algoritmo evolucionário clássico. A autora conclui que este facto, juntamente com o tipo de blocos construtores obtidos na fase primordial, indicam a existência de outras relações entre as etiquetas, para além das que existem entre as palavras vizinhas.

Em [Araujo et al., 2004; Alba et al., 2006] os autores comparam diferentes metaheurísticas para a resolução deste problema. Assim, além do algoritmo genético já proposto em [Araujo, 2002*b*], os autores experimentam, ainda, a utilização de um algoritmo CHC [Eshelman, 1991] e do algoritmo de procura por recristalização simulada. O objectivo do trabalho foi determinar qual o algoritmo que permite obter resultados mais exactos, estudar qual a melhor codificação, e, por último, observar o impacto da utilização de paralelismo em cada um dos métodos usados. Os aspectos fundamentais da aplicação destas metodologias ao problema são semelhantes aos definidos no primeiro trabalho apresentado em [Araujo, 2002*b*], não havendo diferenças substanciais, para além dos aspectos específicos de implementação de cada algoritmo.

Os resultados experimentais relatados mostraram que o algoritmo genético com codificação binária consegue obter os melhores resultados. Relativamente ao algoritmo CHC, os melhores resultados foram obtidos com uma codificação binária e implementação paralela. O algoritmo de procura por recristalização simulada foi o que apresentou piores resultados. Este trabalho comparativo permitiu comprovar as vantagens de utilização de algoritmos evolucionários ao problema de marcação das PdD, em oposição à aplicação de um algoritmo de procura clássico, como o de recristalização simulada.

O sistema proposto em [Wilson and Heywood, 2005] inspira-se no sistema de marcação de Brill [Brill, 1995], baseado em regras de transformação orientadas para a correcção de erros. Os autores propõem um algoritmo genético para fazer a aprendizagem das regras de transformação, em alternativa ao algoritmo trepa-colinas usado por Brill. Assim, um indivíduo da população é representado à custa de um cromossoma que deverá codificar uma lista ordenada de regras de transformação. Foi escolhida uma representação binária de tamanho fixo e definido um número total de regras igual a 378 regras. A ordem inicial das regras é determinada de forma aleatória. Durante o processo de evolução, o ordenamento é alterado através do operador de recombinação.

Uma regra é codificada à custa de uma sequência de 48 *bits*. Consequentemente, um indivíduo é constituído por 48×378 *bits*, que codificam uma lista ordenada de regras que potencialmente resolvem o problema da marcação das PdD. Os 48 *bits* de uma regra codificam sete componentes de informação diferentes: as três etiquetas à esquerda da posição alvo, a etiqueta que deve substituir a etiqueta da posição alvo no caso da regra ser aplicável, e finalmente as três etiquetas à direita da posição alvo. Para codificar cada uma das etiquetas contempladas à esquerda e à direita são usados sete *bits*. O primeiro *bit* indica se essa etiqueta deve, ou não, ser tida em consideração aquando da aplicação da regra, e os seis *bits* seguintes codificam a etiqueta a procurar nessa posição. O total de *bits* necessários para codificar uma regra é por isso igual a $(3 \times 7 + 6 + 3 \times 7 = 48)$. O *corpus* utilizado no trabalho experimental foi o *corpus* do *Wall Street Journal* do *Penn Tree Bank*, tendo sido adoptado o respectivo conjunto de etiquetas, com um total de 45 etiquetas diferentes. Estas foram armazenadas numa tabela de 45 entradas.

Os valores binários, correspondentes aos seis *bits* usados para representar as etiquetas contempladas na regras, são convertidos no correspondente valor decimal, e este valor é por sua vez usado para indexar a tabela de etiquetas referida. Se o valor decimal resultante exceder o limite da tabela, os autores usam o resto da divisão inteira por 45.

Na Figura 4.1, apresenta-se uma parte de um indivíduo de forma a ilustrar de que forma é feita a interpretação da representação binária.

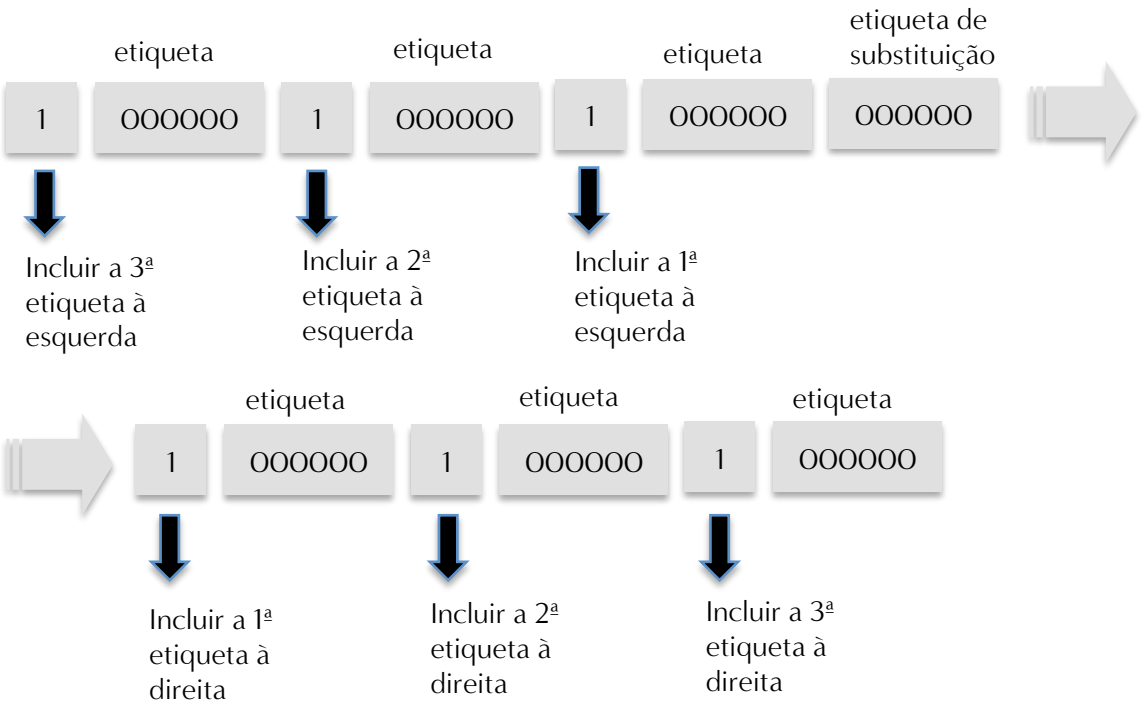


Figura 4.1: Representação binária das regras usadas em [Wilson and Heywood, 2005].

O algoritmo genético foi implementado de duas formas distintas. Numa primeira abordagem, foram feitas experiências para populações de 20 e 100 indivíduos. O número de gerações não foi pré-estabelecido, considerando-se que a evolução decorreria até que um indivíduo tivesse sido avaliado em $6E5$ palavras. Para que tal fosse conseguido, os autores guardaram, para cada indivíduo, o número de palavras total em que cada um foi avaliado. Cada indivíduo foi inicialmente identificado com um índice, o qual se manteve inalterado durante todo o processo de evolução. Em cada iteração foram escolhidos da população, aleatoriamente, quatro indivíduos. Cada um deles foi avaliado num ficheiro (índices 0 a 25) do *corpus* do *Wall Street Journal* (cada ficheiro contém em média 49591 palavras) escolhido aleatoriamente. Em cada iteração, os dois piores indivíduos foram substituídos pelos dois melhores, tendo-se seguido a aplicação dos operadores de variação, neste caso foram usados os operadores de mutação (com probabilidade igual a 0.5) e recombinação (com probabilidade igual a 0.9).

Tabela 4.1: Resultados obtidos pelos melhores indivíduos de cada uma das três versões do marcador evolucionário de Wilson, quando aplicados ao *corpus* do *Wall Street Journal*.

Tam da Pop	C. Paragem	Nº Ficheiros	Exactidão
20	600000 palavras	escolhidos aleatoriamente	0.898
100	600000 palavras	escolhidos aleatoriamente	0.69
20	25 gerações	Documentos 0-12	0.873

Na segunda abordagem, todos os indivíduos foram avaliados nos primeiros 13 ficheiros do *corpus* do *Wall Street Journal*, o que permitiu que todos os indivíduos fossem avaliados para as mesmas 628841 palavras. Nesta versão, os autores consideraram um número fixo de gerações igual a 25. Em cada uma delas foram escolhidos, aleatoriamente, quatro indivíduos, sendo cada um avaliado pela marcação dos 13 primeiros ficheiros do *corpus*. À semelhança do algoritmo anterior, os dois piores indivíduos foram substituídos pelos melhores, sendo de seguida aplicados os operadores de mutação e recombinação, com probabilidades de 0.5 e 0.9, respectivamente.

A função de desempenho escolhida para medir o desempenho dos indivíduos, em ambas as implementações do algoritmo, correspondeu à exactidão conseguida na marcação dos textos (no primeiro caso, um texto escolhido aleatoriamente, no segundo o total dos 13 primeiros ficheiros) quando aplicadas as respectivas regras.

Antes do processo de evolução, o algoritmo começa por atribuir às palavras do *corpus* as partes de discurso mais comuns. Ou seja, cada palavra é marcada com a classe com que aparece mais vezes marcada no *corpus* de treino, tal como acontece no marcador de Brill. Segundo os autores, esta marcação inicial consegue, em média uma exactidão de 0.803 (considerando os resultados em todos os ficheiros).

Com a primeira abordagem, o melhor desempenho conseguido por um indivíduo foi de 0.931. A solução codificada, quando aplicada a todos os ficheiros do *corpus*, obteve uma exactidão de 0.898. Este resultado foi obtido com uma população de 20 indivíduos. Os autores concluíram que um aumento da população, de 20 para 100 indivíduos, não se traduziu em melhores resultados, antes pelo contrário, o melhor resultado alcançado obteve uma exactidão de 0.869.

O melhor indivíduo obtido nas experiências conduzidas com a segunda versão do algoritmo, obteve uma exactidão total de 0.873, quando aplicado aos 13 ficheiros do *corpus* onde foi efectuado o treino. As experiências permitiram observar uma rápida convergência do algoritmo, razão pela qual os autores decidiram não ultrapassar as 25 gerações. A melhor implementação correspondeu assim à primeira abordagem, quando utilizada com uma população de 20 indivíduos. A Tabela 4.1 faz um apanhado dos resultados publicados.

Atipicamente, os resultados apresentados foram obtidos no mesmo *corpus* onde foram aprendidas as regras o que não permite avaliar o comportamento das mesmas em situações que não fizessem parte das da amostra do *corpus* de treino.

Em [Joosse, 2006] encontramos uma abordagem semelhante à de [Wilson and Heywood, 2005] aplicada à língua Holandesa. A proposta de marcador apresentada baseia-se no marcador de Brill. À semelhança do trabalho apresentado em [Wilson and Heywood, 2005], o algoritmo genético é usado para evoluir o conjunto de regras de transformação, substituindo o algoritmo trepa-colinas usado por Brill. No entanto, em [Joosse, 2006], cada indivíduo representa uma selecção das regras de transformação obtidas num primeiro passo.

O algoritmo usado começa por marcar cada uma das palavras do *corpus* de treino com a classe gramatical mais comum. De seguida compara a marcação obtida pelo marcador inicial com a marcação definida no *corpus* de treino. Cada erro identificado origina uma regra de transformação. O algoritmo genético é de seguida usado para evoluir o melhor conjunto de regras de transformação. Cada indivíduo é representado por uma sequência binária, cujo tamanho é igual ao número de regras obtidas na primeira fase. Cada *bit* da sequência referencia uma determinada regra. Se o *bit* for 0 significa que o conjunto proposto pelo indivíduo não contempla essa regra, se for 1 significa que a regra faz parte do conjunto de regras de transformação proposto pelo indivíduo.

Os autores comparam os resultados obtidos pelo marcador com os conseguidos por três outros desenvolvidos para a língua Holandesa nomeadamente, um marcador baseado em redes neuronais (do inglês *neural networks*), um marcador baseado em máquinas de vectores de suporte (do inglês *support vector machines*), e uma implementação do marcador de Brill para a língua Holandesa.

A análise dos resultados permitiu concluir que o algoritmo genético conduziu a valores de desempenho aquém do que era esperado, tendo sido superado pelos outro três marcadores.

Outra abordagem ao problema da marcação das PdD que faz uso de algoritmos genéticos é a apresentada em [Piasecki and Gaweł, 2005]. O autor apresenta um marcador baseado em regras para a língua Polaca. São utilizados quatro tipos de regras: linguísticas (definidas manualmente), baseadas em memória e regras descobertas por um algoritmo genético a partir do *corpus* IPI PAN.

Os autores pretendem descobrir regras que, ao contrário das regras de Brill, não usem um contexto de tamanho fixo que contemple apenas algumas posições na vizinhança da posição alvo, mas sim um contexto flexível que em casos especiais poderá apenas ser limitado pelas fronteiras da frase.

A descoberta de regras pelo algoritmo genético foi realizada a partir de um subconjunto

da parte anotada do *corpus* IPI PAN, constituído por 582179 palavras. O conjunto de etiquetas usado foi o proposto no *corpus.* A forma como foi realizada a anotação do *corpus* condicionou consideravelmente a abordagem seguida pelos autores. No *corpus* IPI PAN foram introduzidas, para além das classes gramaticais, 12 categorias gramaticais tais como: número, forma, género, pessoa, grau, etc. Cada categoria tem associada um conjunto de valores possíveis. Assim, uma etiqueta não consiste apenas numa parte do discurso, mas sim numa classe gramatical seguida de uma sequência de valores das categorias correspondentes. Por exemplo, a palavra *"przyczyna"* recebe a categoria $subst : sg : inst : f$ com o significado: *classe: nome, número: singular, forma: instrumental e género: feminino.*

Dado que o número de sequências de etiquetas é muito elevado, os autores decidiram representar uma etiqueta como sendo uma estrutura de atributos. Optaram ainda em dividir o problema de escolher a melhor etiqueta em sub-problemas relacionados com a escolha de valores para aqueles atributos.

As regras são da forma **Se** *condição* **Então** *conclusão.* Os autores adoptaram o paradigma de Michigan, ou seja cada indivíduo representa uma regra e a população representa um conjunto de regras. A função de avaliação, assim como os operadores genéticos são aplicados aos indivíduos. Foram contemplados dois tipos diferentes de regras. Esta diferença foi determinada pelo consequente, o qual poder ser de dois tipos: *positivo* ou *negativo.* As regras positivas eliminam uma ou mais das possíveis etiquetas. As negativas foram introduzidas para codificar certo tipo de restrições linguísticas que devem ser preservadas em sequências de etiquetas.

Dado que as etiquetas são estruturas de atributos, as regras foram definidas de forma a poder operar nas suas partes. Este facto verifica-se em ambos os lados de uma regra: a conclusão pode afectar qualquer parte da etiqueta resultante, e o antecedente pode testar qualquer parte das etiquetas no contexto.

Os autores definiram três tipos de conclusões:

1. Uma classe gramatical, exemplo Se *condição* Então **subst**;
2. Uma classe gramatical, juntamente com valores para determinadas categorias, por exemplo: Se *condição* Então **subst:sg:_:m3**;
3. Apenas o valor de uma categoria gramatical. Por exemplo, Se *condição* Então **_:sg**

O símbolo _ é usado para indicar que o valor de um atributo não tem relevância para a regra correspondente.

O antecedente das regras foi dividido em duas camadas: uma camada de testes e uma de operadores. O número de elementos em cada camada é o mesmo e é igual ao tamanho

do contexto. Os testes verificam a presença de uma determinada classe e/ou certos valores de determinada categoria. Os operadores da segunda camada implementam restrições linguísticas que testam a concordância da etiqueta em causa, com as etiquetas localizadas dentro e fora do contexto.

O genótipo de cada indivíduo é constituído por dois cromossomas. O primeiro descreve um conjunto de testes e o segundo codifica os operadores, associados a posições subsequentes no contexto. Ou seja, existe um operador para cada etiqueta do contexto, o qual testará a concordância da etiqueta num contexto mais alargado. O trabalho experimental foi realizado com populações de 300 indivíduos, durante 60 gerações. As experiência mostraram que os melhores resultados são conseguidos com um contexto igual a $2 - 1$. Ou seja, as duas etiquetas à esquerda, e a etiqueta imediatamente à direita de uma determinada posição.

O marcador das PdD desenvolvido aplica sucessivamente os quatro tipo de regras: linguísticas, baseadas em memória, e regras negativas e positivas descobertas pelo algoritmo genético. Os módulos destas últimas são aplicados segundo uma ordem pré-estabelecida. As que dizem respeito às classes gramaticais são aplicadas em primeiro lugar, sendo de seguida aplicados os módulos relacionados com as categorias, pela ordem: maiúscula ou minúscula, número, género e pessoa.

Os autores testaram, separadamente, o desempenho das regras, obtendo uma exactidão de 89.2% para as regras baseadas em memória, 83.4% para as regras positivas descobertas pelo algoritmo genético, e 78.7% para as regras negativas. Foram, também, realizadas experiências para testar o desempenho de diferentes combinações de regras. Os resultados mostram que a melhor combinação de regras envolve a utilização dos quatro tipos: linguísticas, baseadas em memória e descobertas pelo algoritmo genético (regras negativas e regras positivas) :

1. Utilização das regras positivas e negativas descobertas pelo algoritmo genético leva a uma exactidão de 84.2%;
2. Utilização das regras baseadas em memória, juntamente com as regras positivas e negativas, permite alcançar uma exactidão de 89.8%;
3. Utilização das regras linguísticas, juntamente, com os restantes três tipos, conduz aos melhores resultados, com uma exactidão igual a 90.0%.

Relativamente à utilização do algoritmo genético, os autores concluem que é praticamente impossível forçar o algoritmo a adquirir regras que cubram toda a gama de exemplos, pelo menos seguindo o paradigma de Michigan. Os autores concluem que 300 regras para um módulo é insuficiente para se conseguir uma boa cobertura. No entanto, admitem que um aumento no número de regras seria extremamente dispendioso, em termos de tempo de processamento.

Tabela 4.2: Pequeno excerto de uma gramática para a língua Inglesa.

(1)	S	$\longrightarrow$	NP VP
(2)	S	$\longrightarrow$	S PP
(3)	NP	$\longrightarrow$	n
(4)	NP	$\longrightarrow$	det n
(5)	NP	$\longrightarrow$	NP PP
(6)	PP	$\longrightarrow$	p NP
(7)	VP	$\longrightarrow$	v NP
(8)	n	$\longrightarrow$	'I'
(9)	n	$\longrightarrow$	man
(10)	n	$\longrightarrow$	park
(11)	v	$\longrightarrow$	saw
(12)	det	$\longrightarrow$	a
(13)	det	$\longrightarrow$	the
(14)	p	$\longrightarrow$	in

4.3 Análise Sintáctica de Frases

O trabalho desenvolvido na área da análise sintáctica de línguas naturais tem início com a teoria linguística desenvolvida por Chomsky que classificou as línguas em quatro classes principais: línguas sem restrições, línguas sensíveis ao contexto, línguas independentes do contexto, e línguas regulares. Estas línguas são produzidas pela aplicação de regras de re-escrita, também designadas regras de produção, da forma $\alpha \longrightarrow \beta$, que estabelecem que uma *string* α é re-escrita sob a forma de uma outra *string* β.

Chomsky defendeu o facto de as gramáticas independentes do contexto (GIC) não serem suficientes para especificar línguas naturais, tendo insistido na necessidade da utilização do contexto. No entanto, existem estudos [G. Gazdar and Sag, 1985]) que defendem o poder das GIC na cobertura de uma vasta gama de línguas naturais. De facto, do ponto de vista prático da análise sintáctica, o desenvolvimento de um algoritmo de análise para gramáticas sensíveis ao contexto tem-se mostrado difícil. Isto traduz-se no facto da generalidade dos sistemas desenvolvidos fazerem uso de GIC. Por este motivo, nesta secção, os exemplos apresentados dizem respeito a este tipo de gramáticas.

Na Tabela 4.2 apresenta-se um pequeno excerto de uma gramática independente do contexto para a língua inglesa retirado de [Tanaka, 1993].

Como podemos ver, cada regra da gramática é da forma **Ant** $\longrightarrow$ **Con**, e em cada an-

tecedente **Ant** apenas surge um símbolo. Aos símbolos que surgem nos **Ants** das várias regras chamamos símbolos não-terminais. No caso do exemplo anterior, os símbolos do conjunto $\{S, NP, PP, VP, n, v, det, p\}$ são símbolos não terminais. Os símbolos da gramática que não são utilizados em nenhum antecedente **Ant**, denominam-se símbolos terminais. No exemplo anterior temos o conjunto $\{I, man, park, saw, a, the, in\}$.

Formalmente, podemos definir uma gramática G à custa de um 4-tuplo,

$$G =< T,\ NT,\ S,\ P >,$$

onde T representa o conjunto de símbolos terminais; NT, o conjunto de símbolos não terminais, disjunto de T; $S \in NT$, representa o símbolo inicial, e finalmente, P, representa o conjunto de regras de re-escrita, também denominadas regras de produção. Podemos agora definir, formalmente, uma gramática independente do contexto da seguinte forma:

Uma gramática, G, diz-se independente do contexto se todas as produções em P forem do tipo:

$$A \longrightarrow x$$

com $A \in NT$ e $x \in (NT \cup T)^*$.

Para sabermos se uma frase é gerada por uma gramática, socorremo-nos do conceito de derivação, geralmente denotado por $\Longrightarrow$. Trata-se de uma relação binária entre formas sentenciais. Assim, diz-se que:

$$f_{s1} \longrightarrow f_{s2}$$

se e só se $f_{s1} = xAy$ e $f_{s2} = xvy$ e existir uma produção na gramática da forma: $A \longrightarrow v$, com $x, y, v \in (NT \cup T)^*$ e $A \in NT$.

Fazendo uso do conceito de derivação, dizemos que uma frase de entrada F é reconhecida por uma gramática GIC, $G =< T,\ NT,\ S,\ P >$, caso possa ser derivada a partir de S, por aplicação das regras de produção P.

Usando o excerto da gramática apresentada em cima podemos derivar a frase *"I saw a man"* da seguinte forma:

`S` $\Longrightarrow$ `NP VP` $\Longrightarrow$ `n VP` $\Longrightarrow$ `'I' VP` $\Longrightarrow$ `'I' v NP` $\Longrightarrow$ `'I' saw NP` $\Longrightarrow$ `'I' saw det n` $\Longrightarrow$ `'I' saw a n` $\Longrightarrow$ `'I' saw a man`.

Esta derivação é, normalmente, designada derivação à esquerda, já que em cada passo se re-escreve o símbolo não terminal mais à esquerda. De forma semelhante podemos

obter a derivação à direita, re-escrevendo sempre, em primeiro lugar, o símbolo não terminal mais à direita. Assim, informalmente, podemos definir a tarefa de análise sintáctica de uma frase (em inglês denominada *parsing*) da seguinte forma: dadas uma gramática $G =< T,\ NT,\ S,\ P >$, e uma frase de entrada F, encontre-se uma derivação da frase F, a partir do símbolo inicial S, aplicando as regras de re-escrita definidas em P.

Por conseguinte, um analisador sintáctico automático (em inglês *parser*) é um sistema que, dados um léxico e uma gramática, consegue determinar qual a estrutura sintáctica de uma determinada frase, devolvendo-o como resultado numa estrutura de dados que toma geralmente a forma de árvore. Ou seja, perante o conjunto de regras gramaticais compatíveis com uma determinada situação, consegue escolher a mais adequada. Mais uma vez, trata-se de uma tarefa de desambiguação a qual é, substancialmente, simplificada se a ambiguidade lexical for, inicialmente, eliminada. Dalrymple ([Dalrymple, 2004]) mostrou que na presença de um marcador das PdD perfeito, a redução do nível de ambiguidade pode chegar aos 50%.

Os algoritmos utilizados para resolver esta tarefa são, geralmente, designados por algoritmos de *parsing*, termo adoptado da designação em inglês, e são pensados para classes de gramáticas, em vez de serem desenhados para gramáticas concretas. Existem várias propriedades que um algoritmo de *parsing* deve possuir, de forma a poder ter utilidade prática [Pulman, 1991]:

- Deverá ser consistente com uma determinada gramática e léxico; ou seja, não deverá atribuir a uma frase de entrada uma análise que não derive da gramática em questão.
- Deverá ser completo; ou seja, deverá atribuir a uma frase de entrada todas as análises que esta poderá ter com respeito à gramática e léxicos considerados.
- Idealmente deveria também ser eficiente, exigindo o mínimo de esforço computacional necessário para cumprir os dois requisitos anteriores.
- Finalmente, deverá ser robusto, comportando-se de forma razoável quando lhe é apresentada uma frase para a qual ele não consegue uma análise completa.

Em [Araujo, 2002*a*] os autores apresentam uma abordagem evolucionária ao problema da análise sintáctica de frases. O algoritmo é aplicado a uma frase de entrada, fornecida sob a forma de uma sequência de palavras, e devolve como solução uma análise sintáctica da frase sob a forma de árvore.

Neste trabalho, os indivíduos da população representam potenciais árvores sintácticas para a frase de entrada. Estas são obtidas de acordo com uma gramática independente

do contexto probabilistica. O conjunto de categorias gramaticais possíveis, para cada uma das palavras da frase, é obtido a partir da pesquisa a um dicionário. Um indivíduo da população, é representado à custa de um cromossoma, que por sua vez é definido à custa de uma estrutura de dados que guarda a seguinte informação:

- Desempenho do indivíduo;
- Uma lista de genes, cada um representando a análise de um conjunto diferente de palavras da frase;
- O número de genes presentes no cromossoma;
- A profundidade da árvore sintáctica correspondente.

Cada gene representa a análise de um conjunto de palavras consecutivas. Caso esta análise envolva símbolos não terminais, a análise das partições subsequentes do conjunto de palavras é dada em genes seguintes. Assim, a informação contida num gene inclui:

- A sequência de palavras consecutivas analisadas pelo gene;
- A regra da gramática usada para analisar as palavras correspondentes ao gene;
- A lista de referências dos genes onde é efectuada a análise de eventuais símbolos não terminais presentes no consequente da regra.
- A profundidade do nó, correspondente ao gene, na árvore sintáctica definida pelo cromossoma. Esta informação é usada na função de avaliação.

Para obter a população inicial a frase é dividida de forma aleatória, garantindo-se, no entanto, a existência de um verbo na segunda parte que irá corresponder à frase verbal (VP) principal. O grupo de palavras correspondente à frase nominal (NP) é analisado por uma das regras possíveis, definidas na gramática para este símbolo. A escolha da regra é realizada de forma aleatória. O mesmo processo é usado para o símbolo VP. De forma a melhorar o resultado do processo, os autores forçam a aplicação das regras capazes de analisar o número exacto de palavras abrangidas pelo gene. O processo continua até não existirem símbolos não terminais por analisar.

A função de avaliação definida tem em conta dois valores: f_{coher} e f_{prob}. O primeiro mede a capacidade do cromossoma em analisar a frase objectivo, enquanto o segundo mede a probabilidade das regras usadas na análise. O valor de f_{coher} baseia-se no número de genes coerentes do cromossoma. Os autores consideram que um gene é coerente se:

- Corresponde a uma regra cujo consequente é apenas composto por símbolos terminais, e estes correspondem às categorias das palavras a serem analisadas pela regra.

- Corresponde a uma regra com símbolos não terminais no consequente, sendo cada um destes analisado por um gene coerente.

O valor de f_{coher} é obtido através da Equação 4.4, onde g_coer representa o número de genes coerente, g_inco o número de genes incoerentes, e num_total, o número total de genes:

$$f_{coher} = \frac{g\_coer - \sum_{i \in g\_inco} \frac{penal}{profundidade(i)}}{num\_total} \tag{4.4}$$

A fórmula tem em consideração a relevância dos genes: quanto mais alto estiver situado na árvore sintáctica, definida pelo cromossoma, o gene incoerente, pior é a análise resultante. Assim, a fórmula introduz um factor de penalização que diminui com o nível de profundidade do gene. Por sua vez, f_{prob} é calculado por:

$$f_{prob} = \prod_{i=1}^{n} Prob(g_i) \tag{4.5}$$

onde $Prob(g_i)$ representa a probabilidade da regra gramatical correspondente ao gene g_i do cromossoma. O desempenho de um indivíduo é, assim, calculado através de uma combinação linear de ambos os valores:

$$Fitness = w_c \times f_{coher} + w_p \times f_{prob} \tag{4.6}$$

onde w_c e w_p são parâmetros que permitem afinar a computação ao longo do processo evolutivo. Os autores conseguem desta forma, fazer variar a importância dos dois componentes da função de avaliação, ao longo da evolução. Inicialmente, nas primeiras gerações, ao atribuírem a w_c um valor mais elevado, conseguem privilegiar indivíduos que correspondem a árvore sintácticas possíveis. Mais tarde, fazem variar os pesos de forma a aumentar o valor de w_p e dar preferência às árvores sintácticas mais prováveis.

Os operadores genéticos utilizados foram a recombinação e a mutação. O operador de recombinação combina duas árvores sintácticas de forma a obter uma nova árvore. A forma como foi implementado é uma adaptação da recombinação típica com um ponto de corte. Resumidamente, o processo pode ser descrito da seguinte forma:

- Para cada par de indivíduos, selecciona-se, de forma aleatória, uma palavra da sequência de entrada, e identifica-se em cada um dos progenitores, o gene mais interior, correspondente a essa palavra.
- Caso os genes correspondam a conjuntos diferentes de palavras, selecciona-se o próximo gene mais interior.
- O processo continua até que a sequência de palavras das análises que se pretendem trocar sejam iguais, ou até que NP e VP sejam encontradas.
- Quando os dois genes seleccionados analisam a mesma sequência de palavras, a recombinação é concretizada. Caso contrário, são gerados, aleatoriamente, genes adequados para a troca.

O operador de mutação é aplicado aos indivíduo resultantes da recombinação, com uma determinada probabilidade a qual é estabelecida como parâmetro de entrada do algoritmo. A mutação foi implementada de forma a que, caso ocorra num indivíduo, seja escolhido, aleatoriamente, um dos seus genes, para o qual será gerada uma nova análise (árvore).

O melhor dos dois descendentes é escolhido para fazer parte da nova geração. A nova geração é obtida a partir da anterior, com os melhores descendentes a substituir os piores indivíduos da população anterior. A selecção é feita tendo em conta o desempenho relativo dos indivíduos: um indivíduo com um desempenho inferior à média dos desempenhos, tem uma maior probabilidade de ser substituído.

Os autores fazem uma implementação paralela do algoritmo evolucionário, adoptando o modelo de ilha, em que os processadores trocam de populações assincronamente numa sequência *round-robin*. As experiências realizadas mostraram ser necessário trocar uma percentagem significativa da população. Relativamente ao intervalo entre trocas, os autores concluíram que as trocas não devem ser nem muito frequentes, nem muito esporádicas.

Em [Araujo, 2004; Gelbukh and Araujo, 2004] encontramos outra abordagem evolucionária ao problema da análise sintáctica de frases. O resultado é um *Chart Parser* estocástico obtido a partir de um algoritmo evolucionário. Neste caso, cada indivíduo da população corresponde a uma análise de um segmento da frase de entrada, sendo-lhe atribuída a categoria sintáctica, correspondente ao símbolo não terminal presente no antecedente da regra principal que determina a análise. Para cada indivíduo são também registadas:

- A primeira palavra do segmento analisado pela árvore;
- O número de nós da árvore;

- O número de palavras da sequência;
- A probabilidade da regra, definida na GIC probabilistica, que determina a análise representada pelo indivíduo.

A população inicial foi gerada utilizando dois mecanismos diferentes. Primeiro, para cada palavra da frase foram gerados indivíduos que representam, simplesmente, uma atribuição a essa palavra de uma das categorias lexicais possíveis. O segundo mecanismo consistiu em gerar indivíduos a partir da aplicação de regras da gramática, desde que todas as categorias, definidas no consequente da regra, fossem entradas lexicais.

Os autores utilizam dois operadores genéticos: um operador recombinação e um operador de *corte* (do inglês *cut*). Cada indivíduo tem uma probabilidade p_r de sofrer recombinação. Caso esta ocorra, o processo definido consiste em procurar na gramática regras cujo consequente tenha, como primeira categoria sintáctica, a categoria que caracteriza o indivíduo seleccionado. De seguida, para cada uma das regras encontradas, o algoritmo irá procurar na população indivíduos que permitam completar o consequente da regra. Caso estes sejam encontrados, são formados novos indivíduos, para cada combinação possível, e adicionados à população.

O operador de *corte* permite a extracção de uma parte da análise representada por um determinado indivíduo. A probabilidade de aplicação do operador aumenta com o tamanho do indivíduo, ao contrário da recombinação, cuja probabilidade é fixa. Desta forma, no início do processo evolutivo a recombinação é aplicada quase em exclusivo, levando ao crescimento dos indivíduos. Mais tarde, quando o tamanho é considerado suficiente, os operadores de *corte* e recombinação são aplicados em conjunto. A aplicação do operador depende, assim, de dois parâmetros: *per_cut* e o *threshold_cut*. O primeiro define a percentagem de aplicação, enquanto o segundo define o número mínimo de palavras que um indivíduo deve ter para que seja permitida a aplicação do operador. Caso as condições necessárias à aplicação do operador sejam satisfeitas, a sua aplicação consiste em seleccionar e cortar, de forma aleatória, uma sub-árvore do indivíduo. Esta assumirá o papel de um novo indivíduo sendo, por isso, adicionada à população.

Em cada geração, os operadores genéticos produzem novos indivíduos, os quais são adicionados à população, aumentando-a. O processo de selecção é responsável por reduzir o tamanho da população para o número pré-estabelecido, o qual é fornecido como parâmetro de entrada. A selecção é realizada tendo em conta o desempenho relativo dos indivíduos. Porém, os autores consideram também outros factores para assegurar a presença, na população, de árvores que contenham palavras que possam vir a ser necessárias em gerações futuras. Assim, antes da eliminação de um indivíduo, o algoritmo verifica primeiro se as palavras representadas nessa análise estão presentes em pelo menos um outro indivíduo. Além disso, de forma a incentivar a diversidade

da população, indivíduos duplicados são também eliminados.

A função de desempenho escolhida é a média das probabilidades das regras gramaticais que determinam a análise representada pelo indivíduo:

$$fitness = \frac{\sum_{\forall s_i \in T} prob(s_i)}{nn(T)} \tag{4.7}$$

Na Equação 4.7, T representa a árvore a avaliar, $nn(T)$ representa o número de nós da árvore T e s_i cada um dos nós de T. Para as categorias lexicais, as probabilidades correspondem à frequência relativa da classe gramatical escolhida.

A gramática, e as frases usadas para avaliar o analisador, foram extraídas do *corpus Susanne*. Os resultados obtidos, medidos pela precisão, cobertura e exactidão, foram comparados com um analisador clássico, um *Chart Parser,* tendo revelado melhores resultados. Os autores concluem, também, que o *corpus Susanne* não é o mais apropriado para este tipo de estudo experimental, dado que o conjunto de etiquetas lexicais e sintácticas usado é demasiado vasto para se conseguirem dados estatísticos significativos.

Em [Maiti et al., 2008] os autores apresentam um analisador sintáctico baseado no algoritmo de optimização por colónias de formigas [Dorigo and Blum, 2005]. O grafo que irá ser percorrido pelas formigas obtém-se pela adição de um nó para cada regra de produção da gramática usada. Cada nó está ligado a todos os outros, por uma aresta, o que significa que é possível, a partir de um nó, transitar para qualquer outro.

As regras de produção são armazenada no nó em duas partes: o antecedente e o consequente. Suponhamos que $S \longrightarrow aAcBe$ é uma das produções da gramática, neste caso a regra seria partida em duas partes, de forma a armazenar S e $aAcBe$ separadamente. Assim, cada nó possuí duas pilhas, onde cada uma das partes é armazenada.

Às formigas são atribuídos alguns atributos que lhes vão permitir movimentarem-se através do grafo ligado. Cada formiga tem disponível a cadeia de entrada w, a qual é armazena sob a forma de uma pilha na sua memória interna. Inicialmente, as formigas são dispostas de forma aleatória pelo grafo. Cada uma delas irá tentar usar a informação armazenada no nó onde estão colocadas, de forma a modificar a expressão que têm em memória.

Vamos supor que é fornecida às formigas a cadeia de entrada $w = abbcde$, e que uma das regras da gramática é: $B \longrightarrow d$. Uma formiga, posicionada no nó correspondente a esta regra, irá verificar se alguma subcadeia da cadeia que tem em memória, unifica com o consequente da regra armazenada no nó, neste caso d. Caso seja possível esta unificação, a formiga irá modificar a sua cadeia, substituindo-a pela expressão presente no antecedente da regra. Neste caso em particular, a formiga iria substituir a cadeia

em memória $w = abbcde$ pela expressão "$abbcBe$", já que descobre que a subcadeia "d" unifica com o consequente da regra representada pelo nó, cujo antecedente é B.

As transições de nó para nó são guiadas de forma probabilística, sendo o valor da probabilidade dado pela Equação 4.8.

$$P_i^k(j) = \begin{cases} \dfrac{\tau_{ij}}{\sum\limits_{k:k\in N_i^k} \tau_{ik}}, & \text{se } q < q_0; \\ 1, & \text{se } \tau_{ij} = max\{\tau_{ik} : k \in N_i^k\} \text{ com } q > q_0; \\ 0, & \text{se } \tau_{ij} \neq max\{\tau_{ik} : k \in N_i^k\} \text{ com } q > q_0. \end{cases} \tag{4.8}$$

No início, a concentração de feromona no espaço de procura é uniforme. Isto faz com que as formigas, inicialmente, não tenham nenhuma ideia de como se movimentar através do grafo. Escolhem, por isso, o próximo nó de forma aleatória. Quando uma formiga chega a um nó, tenta unificar a cadeia (já modificada) que armazena em memória, com as cadeias armazenadas no nó. Caso consiga unificar, modifica a sua cadeia interna de acordo com a expressão do nó e, tal como anteriormente, movimenta-se para o nó seguinte. Caso a informação presente no nó não seja útil, i.e. não seja possível unificar as cadeias da formiga e do nó, a formiga pára de se movimentar, tornando-se inactiva.

Sempre que uma formiga actualiza a sua posição, ou seja quando transita de nó, compara a sua expressão interna com a expressão objectivo (que é o símbolo inicial da gramática usada). Caso unifiquem, isso significa que a formiga descobriu o caminho que permite obter S a partir da cadeia de entrada w, e pára.

Não existe nenhuma restrição que impeça a formiga de visitar um nó mais do que uma vez, já que o processo de derivação da expressão pode implicar a utilização repetida de uma mesma regra de produção. No entanto, cada formiga vai contando o número de saltos que precisou para se deslocar desde o nó inicial até ao nó final, e deposita sobre as ligações percorridas uma quantidade de feromona cuja proporção é inversa ao número de saltos realizados. Isto significa que uma formiga que atinja S, num número mínimo de saltos, deposita uma quantidade máxima de feromona nas ligações que atravessou. À medida que o algoritmo progride, apenas um conjunto restrito de ligações (as que permitem guiar a formiga à solução óptima), recebe uma quantidade crescente de feromona, o que eventualmente conduz à exploração do conjunto mais pequeno de passos que permitem verificar a validade de uma cadeia de entrada.

4.4 Geração de Gramáticas

A indução de gramáticas é uma tarefa para a qual têm sido propostas várias abordagens usando algoritmos evolucionários. Trata-se de induzir as regras da gramática de uma língua a partir de exemplos positivos e negativos.

Como vimos, a análise sintáctica de frases de uma determinada língua requer a utilização de uma gramática para a mesma. Sabemos, também, que uma gramática para uma determinada língua, corresponde a um conjunto exaustivo de regras que definem quais as palavras e sintaxe que devem ser aceites por essa língua. A abordagem manual implica a construção de milhares de regras e entradas lexicais, por linguistas que devem escrever regras que traduzam o seu conhecimento linguístico. Estas gramáticas tendem a ter uma cobertura limitada, não sendo, por isso, muito apropriadas para a análise sintáctica automática de frases reais. Existe, por isso, outra abordagem que utiliza o conhecimento dos linguistas, não para a definição das regras, mas para a construção de um *corpus* anotado de frases com a correspondente estrutura sintáctica. A tarefa que se propõe aos linguistas é, neste caso, a da análise sintáctica de frases concretas, e não a da definição de regras abstractas. Construído o *corpus*, o passo seguinte é a extracção das regras gramaticais a partir dos exemplos nele definidos.

Em [Smith and Witten, 1995] encontramos a aplicação de um algoritmo genético para indução de uma GIC. Os autores admitem apenas um domínio limitado da língua natural. Assim, apenas são consideradas regras não-recursivas, e as experiências foram conduzidas para um conjunto limitado de frases simples do tipo *"The dog saw a cat"*. Neste trabalho, as partes do discurso são deduzidas, automaticamente, em simultâneo com a gramática. Ou seja, não se assume que as palavras tenham sido anotadas previamente com as respectivas classes gramaticais.

Os indivíduos são gramáticas representados por expressões $AND-OR$ na linguagem de programação LISP. Os indivíduos são seleccionados para reprodução e mutação proporcionalmente ao tamanho das gramáticas que codificam. Indivíduos que correspondam a gramáticas com menos nós têm maior probabilidade de se reproduzir e sobreviver do que aqueles que correspondem a gramáticas com um número superior de nós. Esta avaliação é baseada na assunção de que gramáticas mais pequenas são mais eficazes.

O desempenho de um indivíduo é medido com base na análise sintáctica conseguida pela gramática que codifica no conjunto de teste, o qual é gradualmente incrementado. O sistema resultante foi capaz de inferir gramáticas correctas para um conjunto pequeno de frases em língua Inglesa.

[Losee, 1996] propõe um algoritmo que designa por algoritmo genético para aprender de raiz regras sintácticas e etiquetas. Ou seja, não assume de início um conjunto específico de etiquetas. Os indivíduos da população representam gramáticas, constituídas por

um conjunto de regras sintácticas. Cada símbolo não-terminal, no antecedente de uma regra, pode ter diferentes consequentes que descrevem a sua composição, com um valor por defeito igual a cinco. A população é apenas composta por três indivíduos, dois deles são os progenitores e o terceiro, o descendente, gerado pelos dois anteriores. Em cada iteração os três indivíduos são ordenados pelo valor do seu desempenho, os dois melhores indivíduos são definidos como progenitores. A população inicial é composta por cópias de uma gramática inicial, gerada aleatoriamente. O operador de recombinação foi definido de forma a combinar um subconjunto das regras dos dois progenitores. Por sua vez, o operador de mutação produz novas regras por combinação de fragmentos de consequentes de regras com o mesmo antecedente. Pode também produzir fragmentos com conteúdo gerado de forma aleatória.

A função de avaliação definida mede o desempenho da gramática codificada por cada indivíduo em duas tarefas: análise sintáctica e recuperação de informação. Para medir a análise sintáctica os autores utilizam a média do tamanho máximo das análises (o número máximo de termos da análise de cada frase). A análise de cada uma das frases é realizada à custa de um *Chart Parser*.

O sistema foi testado numa colecção de resumos de documentos. De forma a simplificar a gramática, foram apenas usadas 10 partes do discurso e 20 símbolos não-terminais. Os resultados apresentados mostram que o algoritmo é capaz de melhorar a qualidade das regras sintácticas, e partes do discurso inicialmente geradas de forma aleatória.

Em [Korkmaz and Üçoluk, 2001] encontramos a indução de gramáticas independentes do contexto, através da aplicação de programação genética. No entanto, os autores decidem utilizar uma representação alternativa, argumentando que a utilização directa do algoritmo de programação genética clássico falha. Os autores apresentam como explicação a elevada interdependência entre as sub-partes da GIC, as quais tendem a ser destruídas pelos operadores tradicionalmente usados na programação genética.

A representação tradicional em árvore é transformada numa representação vectorial, de forma que cada indivíduo da população represente um ponto num espaço com n dimensões. Esta representação permitiu a introdução de um módulo de controlo que corre um algoritmo de classificação, para determinar cromossomas válidos e inválidos. Este módulo selecciona também a aplicação dos operadores genéticos.

O sistema foi testado numa pequena amostra da língua Inglesa. O conjunto de treino é composto por cerca de 20 frases simples em inglês, compostas por frases nominais e verbais, e onde o verbo na frase verbal pode ser transitivo ou intransitivo. As experiências realizadas mostraram que a versão adaptada do algoritmo, proposta pelos autores, permite obter melhores resultados do que a aplicação directa do algoritmo clássico de programação genética.

[De Pauw, 2003] desenvolveu o sistema GRAEL, uma abordagem evolucionária baseada

em agentes, para indução e optimização de gramáticas para língua natural. O sistema trabalha com uma população de agentes. Cada agente guarda um número de estruturas gramaticais que lhe permite gerar frases, assim como analisar as frases geradas por outros agentes. Estas gramáticas são actualizadas por uma série intensiva de interacções entre agentes, que adopta uma aprendizagem orientada pelo erro. Os agentes de uma população comunicam entre si pela troca de frases. Um agente, $ag1$, apresenta uma frase a outro agente, $ag2$. Se $ag2$ consegue analisar correctamente a frase fornecida por $ag1$, a comunicação tem sucesso. Por outro lado, se $ag2$ tem falta de informação gramatical necessária para analisar a frases correctamente, o agente $ag1$ partilha com $ag2$ a informação necessária para que este consiga chegar à solução correcta.

O autores desenvolveram diferentes instâncias deste ambiente. O sistema GRAEL-1 é a instanciação mais simples do sistema GRAEL, sendo direccionado para a optimização de gramáticas probabilísticas. Foram conduzidas experiências usando parte do *corpus* do *Penn Treebank* para a língua Inglesa. O sistema melhora, significativamente, os resultados da análise conseguidos pelo sistema básico, que consiste em usar como valores de probabilidades das regras a sua frequência no conjunto de treino.

Qualquer tipo de gramática, quer seja induzida a partir de um *corpus* anotado, ou definida manualmente, não consegue cobrir todas as frases de uma língua. De facto, algumas frases irão exigir uma regra que não está disponível na gramática. Com GRAEL-2 os autores pretendem conseguir um método que consiga pegar numa gramática, e melhorar a sua cobertura gerando novas regras. O sistema deverá conseguir isso adoptando um mecanismo guiado, de forma a evitar o crescimento explosivo do número de novas regras, muitas delas sem sentido. A gramática original é distribuída por um grupo de agentes. Cada um destes agentes pode mutar, de forma aleatória, as estruturas gramaticais que lhe foram atribuídas. A nova informação gramatical, criada desta forma, é aplicada e testada pela interacção entre os diferentes agentes. A ideia é tentar reter pela população, qualquer informação gramatical nova, enquanto o ruído é filtrado ao longo do tempo. Com este método os autores conseguem criar novas estruturas gramaticais, anteriormente indisponíveis no *corpus*, enquanto, simultaneamente, as avaliam num contexto prático, sem necessidade de uma fonte de informação externa.

Os melhores resultados são obtidos quando o GRAEL-2 é usado para induzir novas regras, e num segundo passo o GRAEL-1 é usado para optimizar as probabilidades da nova gramática.

Por fim, os autores apresentam o GRAEL-3, o qual se destina à indução de gramáticas, mas agora de forma não-supervisionada. Os resultados ficam aquém dos conseguidos com sistemas supervisionados. No entanto, as experiências mostraram que o ambiente GRAEL, mais uma vez, consegue partir de uma colecção de gramáticas deficientes, e melhorá-las através de um processo longo de interacções entre agentes.

Em [Serrano and Araujo, 2005] encontramos uma abordagem evolucionária ao problema da inferência de uma gramática, mas neste caso apenas para estruturas gramaticais específicas, mais concretamente para frases nominais. Os autores apresentam um método, que caracterizam como flexível, para a identificação de frases nominais básicas (não-recursivas), em textos arbitrários de língua Inglesa.

O sistema gera um autómato de estados finito (AEF) probabilístico capaz de reconhecer a sequência de etiquetas que formam uma frase nominal. O AEF é gerado com o algoritmo de inferência gramatical com correcção de erros (do inglês *Error Correcting Grammatical Inference*), sendo as probabilidades iniciais atribuídas a partir das probabilidades de bi-gramas, recolhidas anteriormente.

Os autores defendem que as probabilidade do AEF permitem um reconhecimento flexível de cadeias de entrada que são consideradas frases nominais. Isto porque permitem reconhecer uma cadeia de entrada, mesmo que esta não seja aceite pelo o autómato, desde que seja suficientemente semelhante a uma que o seja. Desta forma, o sistema é capaz de reconhecer frases nominais que não estão presentes nos exemplos de treino, o que provou ser vantajoso para o desempenho geral do sistema.

Neste trabalho, o algoritmo evolucionário é usado para optimizar as probabilidades do AEF. O algoritmo evolucionário utiliza exemplos de treino positivos e negativos que contribuem para melhorar a cobertura do sistema enquanto mantém uma precisão elevada. Os autores comparam os resultados obtidos, com os de outros sistemas testados no mesmo conjunto de textos, e concluem que o método adoptado melhora a cobertura do sistema, mantendo um elevado nível de precisão, o que se traduz numa melhoria geral do desempenho.

Em [Araujo and Santamaria, 2010] o problema da geração não supervisionada de gramáticas para língua natural é novamente abordado, fazendo-se uso de um algoritmo evolucionário. O algoritmo apresentado constrói, de forma incremental, a gramática que representa a língua. O algoritmo considera apenas uma frase de cada vez. Para uma determinada frase de entrada, o algoritmo constrói uma população de indivíduos que representam possíveis árvores sintácticas para a frase de entrada. Ou seja, árvores que possuam como folhas a sequência completa de etiquetas de partes do discurso da frase de entrada.

A população inicial é gerada de forma aleatória. No entanto, os autores garantem que apenas são consideradas árvores possíveis para a frase. São utilizados quatro operadores genéticos: um de recombinação e três tipos diferentes de mutação. O operador de recombinação foi adaptado de forma a poder ser aplicado a estruturas de dados tipo árvore. Os operadores de mutação foram desenhados de forma a permitir *desagrupar* um grupo de etiquetas (do inglês *ungrouping)*, *agrupar* uma sequência de etiquetas (do inglês *grouping)*, e *reagrupar* um grupo de sequências (do inglês *group-ungroup*),

permitindo alterar a forma do agrupamento.

A função de avaliação desenhada pelos autores tem em consideração os contextos em que cada sequência de etiquetas surge no *corpus* de treino, assim como a frequência dessas sequências e contextos. Os autores avaliaram o sistema, comparando os resultados com as árvores, construídas manualmente, disponíveis no *corpus* do *Penn Treebank*. Os resultados publicados indicam que o algoritmo proposto é capaz de melhorar os resultados de um algoritmo de optimização clássico, como o algoritmo de maximização de expectativas (do inglês *Expectation Maximization*), para constituintes curtos da gramática, embora piorem para constituintes mais longos.

4.5 Considerações Finais

O estudo dos trabalhos apresentados neste capítulo permitiu retirar um conjunto de conclusões que contribuíram para a escolha da direcção seguida no trabalho de investigação apresentado nesta tese. De entre essas conclusões destacamos as seguintes:

- Existe um número substancial de abordagens evolucionárias a problemas de PLN, embora, atendendo às potencialidades destes métodos, se pudesse esperar serem em maior número.
- As abordagens descritas são geralmente complexas, tanto ao nível das representações utilizadas, como das alterações aos algoritmos base, como ainda pelas funções de avaliação utilizadas.
- Muitos dos artigos descrevem diversas dificuldades encontradas durante o desenvolvimento da abordagem, desde a complexidade computacional até à impossibilidade de obter resultados competitivos com abordagens anteriores. Estes dois últimos pontos ajudam talvez a compreender a inexistência de mais abordagens evolucionárias a este tipo de problemas.
- Ainda assim, muitos métodos permitem obter resultados competitivos e mesmo superiores a abordagens clássicas, além de que o próprio desenvolvimento de novas representações e funções de avaliação, contribuem para a melhor compreensão dos problemas em estudo.
- Existem abordagens para diferentes problemas, indicando assim a aplicabilidade versátil dos algoritmos evolucionários na área de PLN, onde muitos dos problemas implicam a procura de soluções em espaços complexos.
- A vasta maioria dos métodos descritos baseiam-se na utilização de algoritmos genéticos. Métodos de inteligência de enxame, mais especificamente os baseados em enxames de partículas, quase ainda não foram explorados.

Em resumo, a aplicação de computação evolucionária na área de processamento de língua natural encontra-se ainda numa fase inicial, sendo as dificuldades encontradas em cada nova aplicação, frequentemente recompensadas por melhores resultados e uma melhor compreensão de diversos aspectos dos problemas. A utilização de algoritmos menos explorados, como a programação genética, ou a optimização por enxame de partículas, abre novas possibilidades de aplicação e a promessa de novos contributos para esta área emergente.

Capítulo 5

Descoberta de Regras de Desambiguação

Neste capítulo abordamos a estratégia adoptada para representar e recolher a informação utilizada para resolver o problema das partes do discurso. Como iremos ver o caminho seguido passou pela generalização da informação tipicamente usada, à custa de um algoritmo de classificação para descoberta de regras. De forma a melhor contextualizar o trabalho apresentado, começamos por fazer uma pequena introdução à descoberta de regras de classificação. Seguem-se a fundamentação da estratégia adoptada e a descrição do algoritmo de classificação utilizado. Como iremos ver este algoritmo foi implementado de duas formas distintas, as quais permitiram estudar a aplicação de dois algoritmos representativos da área da computação evolucionária para a descoberta de regras de desambiguação. É com a descrição da implementação de cada um destes algoritmos, um algoritmo genético e um optimizador baseado em enxames de partículas, que terminamos o capítulo.

5.1 Descoberta de Regras de Classificação

Num problema de classificação, o objectivo consiste em definir um mecanismo capaz de atribuir, automaticamente, a um objecto descrito por vários atributos mensuráveis, uma de várias classes possíveis. Um exemplo clássico consiste em atribuir a um animal a classe a que pertence (mamífero, ave, réptil, peixe...) com base em atributos como a presença de penas, se produz ou não leite, etc... Tipicamente, o mecanismo de classificação é construído a partir de um conjunto de exemplos de objectos previamente classificados, numa forma de aprendizagem supervisionada. Este conjunto é denominado conjunto de treino.

De forma genérica, um conjunto de treino T com n exemplos tem a forma $T = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, em que cada par $(\mathbf{x}_i, y_i)$ corresponde a um exemplo de treino individual. Cada exemplo de treino i é constituído por um vector de m características utilizadas para descrever uma instância em particular do objecto a classificar. Para a instância $\mathbf{x}_i$, cada valor $x_{ij} \in X_i$ corresponde a um atributo mensurável do objecto, cujo espaço de valores possíveis é X_i. Este espaço tanto pode ser um subconjunto de $\mathbb{R}$ como um conjunto de valores discretos, os quais por sua vez podem ser numéricos ou simbólicos. O valor de $y_i \in Y$ corresponde à classificação da instância $\mathbf{x}_i$, podendo assumir uma gama de valores discretos (no caso mais simples, dois) representada por Y.

O objectivo de um algoritmo de classificação é, a partir do conjunto de treino T, construir uma função $f : X_1 \times X_2 \times \cdots \times X_m \longrightarrow Y$, capaz de mapear cada instância possível $\mathbf{x}$ na correspondente classe y, i.e. f(x)=y. A qualidade da função f é normalmente avaliada utilizando um conjunto de exemplos S, disjunto do conjunto de treino T, denominado conjunto de teste. A precisão com que f é capaz de classificar estas novas instâncias, que não estiveram envolvidas na sua construção, permite estimar a sua capacidade de generalização, ou seja a precisão com que irá classificar futuras instâncias que lhe sejam apresentadas e que não fizeram parte do conjunto de treino.

A função f pode assumir muitas formas, desde modelos estatísticos a árvores de decisão, passando por funções matemáticas e algoritmos de variados níveis de complexidade. É de especial interesse para o trabalho apresentado nesta tese a definição de f como um conjunto de regras de classificação com a forma **Se** $z_1 = a$ **E** $z_2 = b$ **E** ..... **E** $z_3 = c$ **Então** $y \leftarrow d$, em que os z_i são atributos da instância $\mathbf{z}$, a, b, e c são valores possíveis para cada um desses atributos, e d é uma classe possível de y. Na realidade, as regras de classificação não são mais do que cláusulas condicionais envolvendo duas partes: o antecedente e o consequente. O primeiro toma a forma de uma conjunção de testes lógicos, envolvendo os atributos mensuráveis do objecto que se pretende classificar, e o último revela a classe a que pertencem as instâncias cobertas pela regra.

Concretizando com um pequeno exemplo, admitamos que se pretende decidir quando é que a generalidade das pessoas pratica desporto ao ar livre. Tipicamente, a generalidade das pessoas toma esta decisão com base nas condições meteorológicas. Limitemos essa decisão às seguintes características atmosféricas: Humidade, Vento e Temperatura. Neste caso, o conjunto de atributos de predição é $A = \{Humidade, Vento, Temp\}$ e $X_{Humidade} = \{baixa, normal, elevada\}$, $X_{Vento} = \{fraco, forte, intenso\}$, $X_{Temp} = \{baixa, moderada, elevada\}$. O objecto a classificar é o objecto Praticar_Desporto, sendo que as classes possíveis pertencem a um conjunto de valores discretos de apenas dois elementos: **Sim** e **Não**, $Y = \{Sim, Não\}$. Uma eventual regra de classificação para este problema poderia ser,

Se Humidade=normal **E** Vento=fraco **E** Temp=moderada **Então**
Praticar_Desporto← Sim

Considera-se que uma instância em particular é coberta por uma regra de classificação, se os valores que apresenta para os atributos de predição satisfazem a expressão lógica definida no antecedente da regra. Ou seja, se o valor lógico resultante corresponde ao valor de verdade *Verdadeiro*.

As regras de classificação são particularmente populares por constituírem uma das formulações mais compreensíveis, sobretudo para utilizadores não familiarizados com a área. Um algoritmo de classificação baseado em regras utiliza o conjunto de exemplos T para construir uma lista de regras, com o formato atrás referido, para cada uma das classes possíveis. Novas instâncias são geralmente classificadas percorrendo, sequencialmente, o conjunto de regras até que a conjunção de condições duma regra seja satisfeita pelos valores dos atributos da instância. Esta recebe a classe associada ao consequente da regra cuja parte condicional foi satisfeita, e diz-se que a instância foi coberta pela regra.

Existem, em especial na área da aprendizagem, inúmeros algoritmos que permitem construir regras a partir de um conjunto de exemplos de treino. Abordagens clássicas incluem algoritmos de cobertura como o PRISM, que gera as regras directamente, ou algoritmos como o C4.5 que geram árvores de decisão que podem, posteriormente, ser convertidas em conjuntos de regras. Muitos destes algoritmos utilizam, no entanto, estratégias gananciosas na procura das regras, já que a vastidão dos espaço de procura torna outras abordagens pouco eficientes. Estas técnicas podem assim facilmente ficar presas em soluções sub-óptimas. É neste contexto que, mais recentemente, têm surgido várias soluções que permitem encontrar regras de classificação utilizando algoritmos de optimização global, em especial os provenientes da computação evolucionária [Freitas, 2003].

Os primeiros algoritmos da área da computação evolucionária a serem utilizados na procura de regras de classificação foram os algoritmos genéticos. A facilidade com que as regras podem ser codificadas como vectores binários foi talvez responsável pela proliferação de abordagens baseadas em AG, as quais podem ser divididas em dois grandes grupos segundo a abordagem utilizada para a representação das regras. Nas abordagens do tipo Michigan uma regra é codificada à custa de um indivíduo (e.g. [Greene and Smith, 1993; Giordana and Neri, 1995]). Quando um indivíduo representa todo o conjunto de regras, a bordagem é normalmente designada como sendo do tipo Pittsburgh (e.g. [de Jong et al., 1993; Janikow, 1993]). A escolha da abordagem depende normalmente do problema, em especial do facto de ser ou não possível avaliar a qualidade de uma regra de forma isolada, ou se essa avaliação só faz sentido para um conjunto de regras no seu todo.

A avaliação das regras é geralmente uma questão complexa que envolve diversos factores. Alguns destes estão directamente ligados ao desempenho em termos de classificação, incluindo factores como a precisão, ou o número de exemplos de treino cobertos pela(s) regra(s), embora também possam ser utilizados elementos adicionais como a parsimónia, i.e. o quão compacto é o classificador em termos de número de regras ou de testes. Os classificadores mais compactos são, de uma forma geral, mais desejáveis, já que tendem a apresentar uma maior capacidade de generalização.

Além dos algoritmos genéticos, também a programação genética foi utilizada com sucesso na evolução de regras de classificação, tanto directamente, como a partir de árvores de decisão, as quais se adequam mais naturalmente à representação utilizada em PG (e.g. [Eggermont et al., 1999; Wong and Leung, 2000]). Os algoritmos baseados em inteligência de enxame, tendo surgido mais tarde, só mais recentemente foram aplicados aos problemas de classificação e, mais especificamente, à procura de regras de classificação.

Em trabalho anterior, apresentámos uma das primeiras propostas de utilização de algoritmos de optimização por enxames de partículas em tarefas de classificação [Sousa et al., 2003, 2004]. Nesse trabalho, utilizando vários problemas típicos da área, comparámos vários classificadores, baseados em OEP, com um algoritmo genético e uma implementação do C4.5. Os resultados obtidos permitiram estabelecer que a abordagem utilizada era não só viável, mas também bastante competitiva, em termos de precisão da classificação, com os restantes algoritmos. Foi, também, possível verificar que os classificadores baseados em inteligência de enxame conseguiam frequentemente obter resultados semelhantes, ou melhores, do que os do classificador baseado em AG, necessitando para isso, em alguns dos problemas, de significativamente menos partículas/indivíduos, com óbvios ganhos computacionais.

Posteriormente realizámos uma comparação mais extensa, utilizando mais conjuntos de dados e dois dos algoritmos clássicos mais populares (PRISM, C4.5), além de várias versões de classificadores baseadas em OEP [Sousa et al., 2005]. De acordo com as expectativas teóricas, foi possível obter melhores resultados de precisão de classificação, com os algoritmos de enxame, em vários dos problemas utilizados na comparação. Adicionalmente, pôde-se constatar que os modelos de classificação construídos pelos classificadores baseados em OEP eram substancialmente menos complexos, tanto em termos de número de regras, como de número médio de testes por regra, do que os classificadores produzidos pelos algoritmos clássicos. Esta qualidade de parsimónia é bastante valiosa em muitas aplicações (por exemplo, na análise inteligente de dados), e acreditamos que também o possa ser no problema objecto desta tese. Refira-se ainda que a inteligência de enxame tem continuado a ser aplicada com sucesso nesta área, tendo inclusivamente surgido abordagens também baseadas em optimização por colónias de formigas [Holden and Freitas, 2005; Falco et al., 2007]. Um resumo recente

da utilização de inteligência de enxame em problemas de classificação, no contexto mais lato da análise inteligente de dados, pode ser encontrada em [Martens et al., 2011].

Os resultados obtidos neste trabalho anterior, aliado ao sucesso obtido por outras abordagens baseadas em inteligência de enxame, levam-nos a considerar a possibilidade de aplicação destes algoritmos à tarefa da marcação das PdD, que, embora diferente do problema de classificação, tem com ele vários pontos de contacto, discutidos na próxima secção. A experiência ganha em termos de representação e avaliação das regras, bem como na construção dos próprios algoritmos, também se mostrará útil no desenvolvimento de abordagens evolucionárias a este novo problema.

5.2 Definição da Estratégia

A partir do estudo da arte realizado pudemos observar que independentemente do tipo de abordagem utilizada, a tarefa de determinar qual a classe gramatical que uma determinada palavra assume numa frase, tem em consideração a classe gramatical a que pertencem as palavras que a rodeiam. A questão está em como representar e usar esta informação. Nos sistemas baseados em regras, a informação sobre o contexto está representada sob a forma de regras, geralmente regras de transformação orientadas para a correcção de erros. Nos sistemas baseados em modelos estatísticos, essa informação é usada sob a forma de valores de probabilidades.

A aplicação de algoritmos da área da computação evolucionária a este problema, tem até agora sido concretizada por utilização de algoritmos evolucionários, mais concretamente algoritmos genéticos. Nestas abordagens, o algoritmo genético é utilizado de forma diversa, consoante se trate de um marcador baseado em regras ou dum marcador estocástico. Nos primeiros, a aplicação do algoritmo evolucionário serve o propósito de descobrir o conjunto de regras de transformação que servirão para realizar a marcação automática das PdD. Porém, a marcação é efectuada de forma não evolucionária, geralmente por utilização sistemática das regras de transformação. Nos marcadores estocásticos, o algoritmo genético é usado tipicamente para descobrir a sequência de etiquetas mais prováveis para marcar uma frase. Neste caso, a informação estatística previamente recolhida vai servir para avaliar os indivíduos da população.

Podemos identificar alguns paralelismos entre o problema da marcação das PdD e o problema de classificação. Neste último, como já vimos, o objectivo consiste em definir um mecanismo que permita atribuir, automaticamente, a um objecto descrito por vários atributos mensuráveis, geralmente designados atributos de predição, uma de várias classes possíveis. Uma das forma de resolver o problema é à custa de um conjunto de regras de classificação.

No problema da marcação das PdD, o objectivo é inferir a classe gramatical com que uma determinada palavra está a ser usada numa frase, a partir de um conjunto de características que englobam diversos aspectos. Estes dizem respeito, tipicamente, às classes gramaticais das palavras vizinhas, características morfológicas da palavra, características morfológicas das palavras vizinhas, e as próprias palavras. O objecto a classificar é uma palavra dentro do contexto de uma frase, usando como atributos mensuráveis todas, ou algumas, das características mencionadas. As classes possíveis são neste caso as classes gramaticais possíveis para a palavra objecto de classificação. Acresce que a marcação das PdD tem como objectivo classificar, automaticamente, todas as palavras de uma frase, pelo que a classificação de cada palavra não pode ser encarada como um acto isolado. Revela-se antes como uma cadeia de decisões com relações de dependência muito fortes. Daqui decorre que ao contrário do problema de classificação, na marcação das PdD o problema não se resolve à custa de uma função $f : X_1 \times X_2 \times \cdots \times X_n \longrightarrow Y$.

A realidade actual caracterizada por um número cada vez maior de recursos linguísticos, em particular *corpora* anotados em várias línguas, constituídos por milhares de palavras marcadas manualmente, possibilita a construção de conjuntos de treino, como os que identificámos para o problema de classificação. Cada tuplo $\langle palavra, etiqueta \rangle$ do *corpus* anotado determina um exemplo específico $\langle \mathbf{x}_i, etiqueta \rangle$, com $\mathbf{x}_i$ a reunir um conjunto de características específicas sobre a palavra. A perspectiva de aplicação de um algoritmo de classificação para descoberta de regras de classificação a partir dos *corpora* disponíveis apresenta-se, por conseguinte, viável, não obstante o resultado que se possa retirar destas não ser o tipicamente esperado.

Na eventualidade de possuirmos um conjunto de regras para classificar uma palavra relativamente à classe a que pertence, no contexto de uma frase, a sua aplicação para resolver o problema da marcação das PdD, não é de todo óbvia. Vamos admitir essa hipótese, considerando que temos ao nosso dispor um conjunto de regras de classificação de palavras, e que os atributos de predição contemplam informação determinante para a classificação, como sejam as classes das palavras vizinhas (à esquerda e à direita). Facilmente percebemos que as regras não podem ser usadas directamente como regras de classificação, dado que os valores dos atributos de predição, numa determinada situação, podem ainda ser desconhecidos, e sobretudo porque esses próprios valores dependem da predição para essa mesma posição. Não obstante, se considerarmos uma situação isolada, e assumindo que são conhecidas as etiquetas das palavras vizinhas, a utilização das regras de classificação é possível. A questão óbvia é como aproveitar o conhecimento implícito nestas regras para resolver o problema da marcação das PdD.

Sabemos que um algoritmo para marcação das PdD deve receber como entrada uma frase, $\mathbf{p}$, não anotada, definida à custa de n palavras, p_i, e devolver a mesma frase, mas agora com todas as p_i palavras marcadas com a classe adequada. Na hipótese

de conhecermos as possibilidades de marcação, e, de cada uma das palavras, p_i, com $e \in P_i$, da frase de entrada, o espaço de procura do problema pode ser definido pelo conjunto $P_1 \times P_2 \times \cdots \times P_m$. Neste caso a solução poderá ser encontrada por procura no espaço de estados do problema. Temos no entanto que definir qual é o objectivo da procura, em suma que solução é que procuramos. Nas abordagens evolucionárias que destacámos no estado da arte, o objectivo é optimizar uma medida de probabilidade definida à custa da função de avaliação usada para medir o desempenho dos indivíduos. A informação usada para o cálculo das probabilidades é armazenada sob a forma das já mencionadas tabelas de treino. A solução óptima, neste caso, é a que atribui a cada uma das p_i a $e \in P_i$ mais provável de acordo com a tabela de treino.

É nossa convicção que a informação armazenada nas tabelas de treino pode ser interpretada como um conjunto de instâncias, descritas à custa de um conjunto de atributos mensuráveis, relacionados com as etiquetas das palavras vizinhas, às quais está associado um valor numérico que identifica o número de vezes em que cada uma ocorre no *corpus* de treino. Naturalmente, esta informação é específica para o *corpus* de onde foi retirada e não apresenta qualquer grau de generalização, sendo antes uma recolha extensa e exaustiva de informação. Será, portanto, oportuno investigar a possibilidade de generalizar esta informação à custa de um algoritmo de classificação, reduzindo não só a quantidade de informação necessária para resolver o problema, mas também contribuindo para uma expectável melhoria da qualidade da marcação. O conjunto de regras obtido, poderá ser usado, à semelhança da tabela de treino, para guiar a procura da solução no espaço de estados do problema de marcação. Estas regras não visam classificar uma determinada palavra, mas antes avaliar a qualidade de uma classificação em particular.

Independentemente do algoritmo de classificação usado para aprendizagem de regras de classificação, as regras são avaliadas sob vários aspectos, tipicamente, relacionados com o desempenho em termos de classificação. Aqui destacam-se, como já observámos na secção anterior, factores como a precisão e a cobertura, ou seja o número de exemplos de treino cobertos pela regra. Decorre deste facto que, cada regra r_i, obtida pelo algoritmo de classificação poderá ter associado um valor numérico, q_i, que reflecte a sua qualidade e que determinou a sua descoberta pelo mecanismo de aprendizagem. O conjunto formado pelos tuplos $< r_i, q_i >$, poderá deste modo consubstanciar-se numa heurística que poderá ser usada para guiar a procura da solução para o problema da marcação das PdD, e não como um classificador, razão pela qual optámos por designar as regras por regras de desambiguação.

Outro aspecto que observámos do estudo das abordagens estatísticas ao problema, quer por aplicação de modelos tradicionais como o de Markov, ou de abordagens evolucionárias, é o facto da informação usada se limitar às etiquetas das palavras de contexto. Existem, contudo, outros aspectos que podem ser usados para determinar a classe gra-

matical com que uma palavra está a ser usada numa frase, para além do contexto [Steven Bird and Loper, 2009]. A estrutura interna das palavras pode ser uma pista bastante útil. Por exemplo, na língua Inglesa, geralmente o sufixo *-ness* combinado com um adjectivo conduz a um substantivo, tal como *happy* → *happiness*, *ill* → *illness*. Desta forma, se encontramos uma palavra que termine com *-ness*, é muito provável que esta seja um nome. De forma semelhante, *-ing* é um sufixo geralmente associado com o gerúndio, como *walking*, *talking*, *thinking*, *listening*. Podemos também admitir que há grande probabilidade de uma palavra que termine em *-ed* ser um verbo no particípio passado, e que qualquer palavra que termine em *'s* seja um nome possessivo. A forma como está escrita a palavra poderá também revelar-se útil. O facto de sabermos se a palavra começa, ou não, por letra maiúscula, poderá ajudar a decidir se se trata de um nome próprio, sobretudo se soubermos se isso se deve, ou não, ao facto de a palavra iniciar a frase.

Do que já foi dito sobre a representação típica das regras de classificação, a consideração desta informação na resolução do problema da marcação das PdD, é facilmente conseguida por inclusão dos atributos de predição que considerarmos relevantes.

A oportunidade de investigação que acabámos de descrever, aliada à experiência adquirida no trabalho exploratório realizado no âmbito da aplicação de algoritmos da área de computação evolucionária, ao problema de classificação, conduziu-nos ao trabalho que apresentamos nesta tese. Nele decidimos investigar a possibilidade de dividir o problema da marcação das PdD em dois subproblemas. O primeiro diz respeito à aplicação de um algoritmo de classificação para descoberta de um conjunto de regras que representarão não um classificador, mas sim uma heurística que deverá ser usada numa segunda fase com vista à marcação efectiva das palavras de uma frase. O segundo problema prende-se com o desenvolvimento de um algoritmo de procura que permita resolver o problema de optimização combinatória, inerente ao problema de marcação das PdD, usando para tal a heurística consubstanciada pelo conjunto de regras de desambiguação descoberto na primeira fase.

A consolidação do sucesso dos algoritmos de computação evolucionária em problemas de classificação, na qual o trabalho anterior desenvolvido, e mencionado atrás, teve um contributo relevante, levou-nos a seguir esta linha de investigação para a descoberta das regras de desambiguação. Escolhemos aplicar um algoritmo de cada uma das sub áreas da computação evolucionária. De entre os algoritmos evolucionários a nossa escolha foi para os algoritmos genéticos, e dentro da inteligência de enxame optámos por implementar um optimizador baseado em enxames de partículas. Ambas as escolhas são justificadas pelas provas anteriormente dadas na resolução de problemas de classificação.

Nas secções seguintes iremos delinear os aspectos fundamentais da abordagem seguida. Começamos por apresentar os aspectos gerais do algoritmo utilizado, e terminamos

com a exposição dos aspectos específicos da implementação do algoritmo genético e do optimizador baseado em enxame de partículas.

5.3 Algoritmo Genérico

O mecanismo adoptado para a descoberta das regras de desambiguação, divide o problema em n problemas de classificação distintos, sendo n o número de etiquetas diferentes usadas na anotação do *corpus*, a partir do qual serão aprendidas as regras, e que definem o conjunto de etiquetas E. Cada etiqueta $e \in E$ presente no *corpus* determina um objecto a classificar, sendo que as classes possíveis pertencem ao conjunto discreto $Y = \{Sim, Não\}$.

O conjunto de exemplos de treino para cada um dos problemas de classificação foram extraídos a partir de um *corpus* anotado. Estes repositórios linguísticos armazenam um conjunto extenso de textos, onde cada palavra está marcada com uma determinada etiqueta, a qual representa a classe gramatical com que aquela está a ser usada na frase, formando, como já dissemos, um tuplo $\langle palavra_i, etiqueta_i \rangle$.

A experiência anterior na área levou-nos a definir o algoritmo de classificação com base num algoritmo de cobertura. As linhas gerais do algoritmo encontram-se definidas no Algoritmo 5.1. Como podemos observar, o conjunto final de regras é conseguido à custa de m execuções do algoritmo de procura responsável por determinar a melhor regra de classificação, para o conjunto de exemplos de treino que recebe como entrada. De cada vez que o algoritmo de procura é executado, a regra obtida é guardada, e o conjunto de exemplos positivos é actualizado. Esta actualização consiste em eliminar do conjunto todas as instâncias cobertas pela regra. O algoritmo de procura será executado tantas vezes quantas as necessárias para que se consigam cobrir todos os exemplos de treino, ou seja, até que o conjunto de exemplos positivos seja igual ao conjunto vazio.

Algoritmo 5.1 Algoritmo de Cobertura

Entrada: ExemplosPos, ExemplosNeg
Saída: ConjuntoRegras
  enquanto ExemplosPos $\neq \varnothing$ **faz**
    ⟨MelhorRegra, Qualidade⟩ ← AlgoritmoProcura(ExemplosPos, ExemplosNeg)
    ExemplosPos ← RemoveExemplos(ExemplosPos,MelhorRegra)
    ConjuntoRegras ← Adiciona(ConjuntoRegras, ⟨MelhorRegra, Qualidade⟩)
  fim enquanto

Este mecanismo foi implementado de duas formas diferentes. Uma implicou a definição do algoritmo de procura à custa de um algoritmo genético e a segunda resultou na aplicação de OEP. Nas secções seguintes, iremos debater os aspectos relacionados com

a implementação de cada um destes algoritmos. Não obstante, existem determinadas opções de modelação que são comuns a ambos, como sejam a escolha dos atributos de predição, a escolha da representação dos indivíduos, a definição de rotinas para extracção dos exemplos de treino, e por fim questões relacionadas com a avaliação das regras. Naturalmente, todos estes aspectos serão tratados de forma comum, e será com eles que iniciaremos a exposição que se segue, terminando a mesma com os aspectos específicos de implementação de cada um dos algoritmos.

5.4 Selecção de Atributos de Predição

Os atributos de predição contemplados no antecedente das regras devem formar um subconjunto do conjunto de atributos possíveis para descrever o objecto alvo de classificação. Estes devem ser seleccionados de acordo com a sua relevância na resolução do problema de classificação.

Já tivemos oportunidade de enunciar os aspectos que tipicamente são considerados relevantes para desambiguar a tarefa de marcação das PdD. Os mais consensuais dizem respeito ao contexto da palavra, o qual contempla as etiquetas com que estão marcadas as palavras vizinhas. Esta vizinhança estende-se pelas palavras à esquerda e/ou à direita da palavra alvo, sendo que o número de palavras abrangidas é diverso de abordagem para abordagem. O número de palavras vizinhas à esquerda e à direita define a forma do contexto.

Nos marcadores estocásticos, baseados em modelos de Markov, o contexto é formado, apenas, por palavras à esquerda de uma palavra. No entanto, nos marcadores estocásticos, baseados em algoritmos evolucionários [Araujo, 2002*b*], assim como nos baseados em regras de transformação (por exemplo, [Brill, 1995] e [Wilson and Heywood, 2005]), são contempladas palavras à esquerda e à direita. Relativamente ao número de palavras contempladas, as conclusões retiradas do trabalho apresentado em [Araujo, 2002*b*], são de que contextos com mais do que três palavras à esquerda e à direita, se mostram irrelevantes.

Foi assim para nós, evidente que o contexto da palavra deveria ser representado nas regras de desambiguação sob a forma de atributos de predição. Foram contemplados os seguintes elementos de informação:

- A etiqueta da terceira palavra à esquerda;
- A etiqueta da segunda palavra à esquerda;
- A etiqueta da primeira palavra à esquerda;

- A etiqueta da primeira palavra à direita;
- A etiqueta da segunda palavra à direita;
- A etiqueta da terceira palavra à direita;

Estes atributos coincidem com os que foram usados nos trabalhos de [Brill, 1995] e [Wilson and Heywood, 2005].

Além do contexto, foram tidas em consideração outras características das palavras. Como já vimos, a estrutura interna da própria palavra, pode contribuir com pistas úteis [Steven Bird and Loper, 2009] sobre a classe gramatical a que esta pertence. A forma como está escrita a palavra, se começa ou não com letra maiúscula, se inicia ou não a frase, são considerações que podem ser relevantes para a identificação de determinada classe gramatical. Evidentemente, alguns aspectos desta informação dependem da língua com que se pretende trabalhar, como por exemplo a estrutura interna das palavras. Quando definimos os *corpora* a usar no trabalho experimental optámos por dois *corpora* em língua Inglesa, os quais coincidem com os utilizados nas abordagens evolucionárias ao problema, mencionadas no estado da arte. O objectivo foi poder comparar os resultados conseguidos com os publicados em trabalhos semelhantes. Assim, assumindo que pretendemos realizar a marcação das PdD para a língua Inglesa considerámos as seguintes observações sobre as palavras (retiradas de [Steven Bird and Loper, 2009]):

- Palavras que terminam em -ed têm grande probabilidade de serem verbos no tempo passado;
- Palavras que terminam em -ing têm grande probabilidade de serem verbos no particípio presente;
- Palavras que terminam em -es têm grande probabilidade de serem verbos na 3ª pessoa do singular;
- Palavras que terminam em -ould têm grande probabilidade de serem um verbo modal;
- Palavras que terminam em -'s têm grande probabilidade de serem nomes possessivos;
- Palavras que terminam em -s têm grande probabilidade de serem nomes no plural;

Além destas características, específicas para as palavras da língua Inglesa, considerámos ainda os seguintes aspectos possíveis de aplicar a várias línguas:

- Palavras que contenham algarismos e/ou '.' e algarismos, têm grande probabilidade de representarem números;
- Palavras que comecem por letra maiúscula e não sejam a primeira palavra da frase, têm grande probabilidade de serem nomes próprios;

O conjunto completo de atributos de predição adoptado, não relacionados com o contexto, foi o seguinte:

- Se a palavra começa, ou não, por letra maiúscula:
- Se a palavra é a primeira palavra da frase;
- Se a palavra termina em **ed**;
- Se a palavra termina em **ing**;
- Se a palavra termina em **es**;
- Se a palavra termina em **ould**;
- Se a palavra termina em **'s**;
- Se a palavra termina em **s**;
- Se a palavra contém algarismos ou . e algarismos.

O conjunto de valores possíveis para cada um dos atributos mensuráveis relacionados com o contexto, são as etiquetas presentes no conjunto E. Os restantes atributos tomam valores lógicos, assumindo assim os valores de verdade *Verdadeiro*, e *Falso*.

5.5 Codificação das Regras

Como atrás observámos, a aplicação de algoritmos da área da computação evolucionária para descoberta de regras, usa, tipicamente, uma de duas abordagens para codificação das regras: a abordagem de Michigan ou a abordagem de Pittsburgh. Na abordagem de Michigan, cada indivíduo/partícula codifica uma única regra, enquanto que na abordagem de Pittsburgh, cada indivíduo/partícula representa um conjunto de regras. Em ambos os casos, as regras são tradicionalmente representadas à custa de vectores binários.

A escolha da abordagem depende, fortemente, do tipo de regras que pretendemos descobrir, o que por sua vez está relacionado com o tipo de tarefa de aprendizagem

que nos propomos resolver. No caso do problema de classificação estamos, geralmente, interessados em avaliar o conjunto de regras como um todo, em oposição à apreciação individual das regras. Ou seja, a interacção entre as regras é importante, e por esse motivo, para tarefas de classificação, a abordagem de Pittsburgh apresenta-se mais natural [Freitas, 2003].

Neste caso estamos interessados num conjunto de regras que se pretende que sejam usadas, não como um classificador, mas como uma heurística que permita resolver o problema da marcação das PdD. Apesar disso, a abordagem de Pittsburgh continua a parecer mais natural, dado que nos permite avaliar aspectos do conjunto de regras como a sua parsimónia. No entanto, ela levanta uma dificuldade acrescida relacionada com o tamanho dos indivíduos. Se assumimos que um indivíduo codifica um conjunto de regras precisamos de decidir como lidar com a cardinalidade desse conjunto. São válidas duas opções: ou se estabelece um número fixo de regras, ou se adopta uma representação de tamanho variável. No primeiro caso, todos os indivíduos são compostos pelo mesmo número de regras, viabilizando a adopção de uma representação tradicional de tamanho fixo. Na segunda opção, a admissão de indivíduos de tamanho diverso, permite explorar conjuntos de regras com diferentes graus de parsimónia. Neste caso, o algoritmo de procura tem não só a responsabilidade de descobrir as regras, como pode ser levado a descobrir o conjunto de regras mais compacto.

Na representação de tamanho fixo, o problema está em escolher o número de regras que queremos que cada indivíduo codifique, já que geralmente não sabemos quantas regras são necessárias para resolver um particular problema de classificação. Por outro lado, com uma representação de tamanho variável, enfrentamos um problema de difícil resolução relacionado com o controlo do tamanho dos indivíduos. É um facto estabelecido, que nas representações de tamanho variável os indivíduos têm tendência a crescer desmesuradamente ao longo do processo de evolução. Trata-se de um problema bem conhecido, denominado em inglês por *bloat problem* [Poli, 2003]. Esta explosão no crescimento dos indivíduos leva a óbvios custos computacionais.

Após ponderadas as vantagens e desvantagens de cada uma das abordagens optámos pela abordagem de Michigan. Esta decisão foi sustentada pelo facto dos conjuntos de treino serem de dimensão considerável, sendo por isso expectável um tempo de execução significativo para o algoritmo de classificação, o qual seria agravado se não conseguíssemos resolver, de forma eficaz, o problema da explosão do tamanho dos indivíduos no algoritmo de procura. Os resultados conseguidos no trabalho anterior, usando a abordagem de Michigan, condicionaram também esta decisão.

Na estratégia adoptada, cada indivíduo representa uma regra da forma **Se** *Antecedente* **Então** *Consequente*, onde *Antecedente* representa uma conjunção dos atributos de predição e *Consequente* um dos valores possíveis do conjunto $Y = \{Sim, Não\}$.

5.5.1 Representação do antecedente

Uma forma natural de codificar o antecedente de uma regra (uma conjunção de condições) num indivíduo é utilizando uma representação binária. Vamos assumir que um determinado atributo pode tomar um de k valores discretos . Estes valores podem ser codificados usando k *bits*. O valor do atributo na posição i, a_i, com $i = 1, ..., k$, faz parte da condição da regra, se e só se o *bit* correspondente é igual a 1.

A título ilustrativo, consideremos o seguinte pequeno exemplo. Pretendemos representar o antecedente de uma regra que tem apenas em consideração um atributo de predição, por exemplo *Tempo*, cujos valores possíveis são *Sol*, *Chuva*, *Nevoeiro* e *Vento*. Neste caso, a codificação deste antecedente numa representação binária, implicaria definir um determinado alinhamento para os valores do atributo *Tempo*. Este alinhamento pode ser estabelecido à custa do seguinte tuplo, $\langle Sol, Chuva, Nevoeiro, Vento\rangle$. A dimensão do tuplo define a dimensão do vector binário necessário para codificar o antecedente da regra. Por sua vez, o ordenamento do tuplo determina de que forma cada valor binário deve ser interpretado. No caso particular deste exemplo, teríamos necessidade de adoptar um vector binário constituído por 4 *bits*, sabendo que o primeiro *bit* se refere ao valor Sol, o segundo ao valor $Chuva$, e assim sucessivamente. Concluindo, uma sequência do tipo $\langle 1001\rangle$ permitiria codificar o seguinte antecedente, "**Se** *Tempo=Sol* **OU** *Tempo=Vento* **Então**...".

Como podemos ver, este tipo de representação permite definir conjunções de disjunções lógicas. A inclusão de outro atributo implicaria apenas concatenar a sequência de *bits* necessária para codificar os respectivos valores. Daqui decorre que este tipo de codificação pode ser estendida de forma a incluir um número qualquer de atributos, assumindo que todos estão ligados pela conjunção lógica. Uma situação particularmente importante deste tipo de representação surge quando todos os *bits* de um determinado atributo são iguais a 1. A leitura directa desta situação indica que qualquer valor é possível para esse atributo, o que em termos da interpretação lógica da conjunção definida no antecedente da regra, significa que esse atributo em particular pode ser ignorado.

Como vimos na secção anterior, os valores possíveis para os atributos de predição, relacionados com a informação de contexto, pertencem ao conjunto de etiquetas adoptado para o *corpus* usado para descobrir as regras. A cardinalidade deste conjunto varia de *corpus* para *corpus*, no entanto, os valores mínimos usados situam-se entre as 20 etiquetas e os máximos ultrapassam as 80. Uma representação como a que acabámos de descrever conduziria a indivíduos demasiado longos. Por este motivo optámos por uma representação ligeiramente diferente, inspirada na representação usada por Wilson em [Wilson and Heywood, 2005].

Para cada um dos seis atributos relacionados com o contexto usámos $n + 1$ *bits*. O

primeiro *bit* indica se o atributo deve ou não ser contemplado no antecedente da regra, enquanto os restantes n *bits* codificam o valor para o atributo em questão. O número, m, de etiquetas usadas na marcação do *corpus* de treino determina o valor de n. Assim, n deve tomar o valor do menor inteiro que permite definir o maior binário, b, através da sequência de n *bits*, de forma a que $b \geq m$. O valor codificado pelos n *bits* é usado para indexar uma tabela de m entradas, que guardam as etiquetas do conjunto de etiquetas considerado. Se o valor decimal, i, correspondente ao binário b ultrapassar o valor de m, então o resto da divisão inteira de i por m é usado. O *bit* extra adoptado para cada atributo permite ignorar esse atributo no antecedente da regra, tal como acontece na representação ilustrada em cima, quando todos os *bits* são iguais a 1 (ver Figura 5.1).

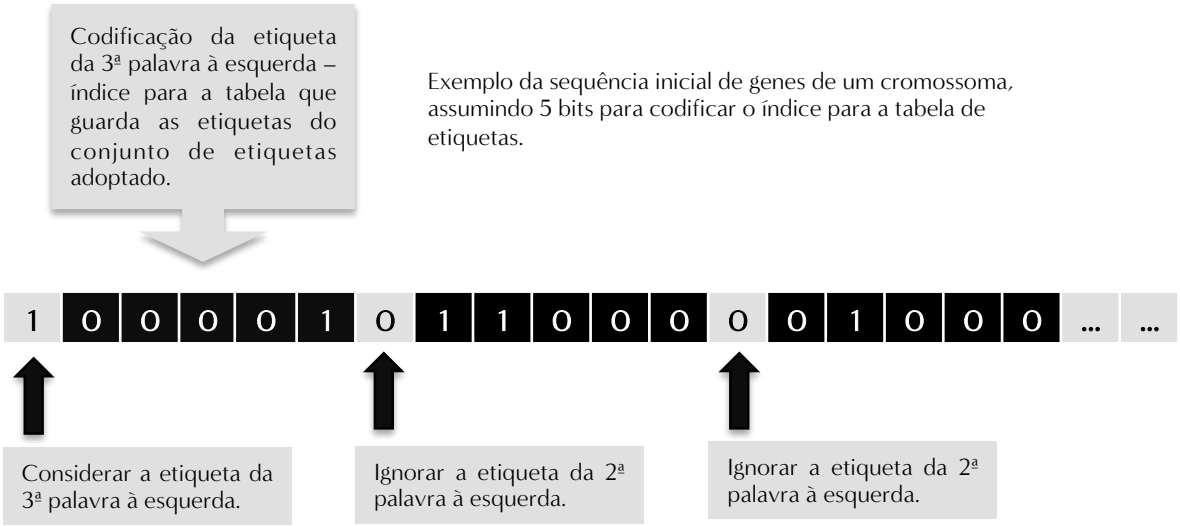


Figura 5.1: Representação binária dos atributos relacionados com o contexto.

Os restantes nove atributos foram codificados à custa de 18 *bits*. Cada par de dois *bits* indica se o atributo deve ou não ser considerado, e se a propriedade correspondente está, ou não, presente (ver Figura 5.2). Assim, cada indivíduo é constituído por $6 \times (n + 1) + 2 \times 9$ *bits*.

Tal como na representação típica, os atributos estão ligados por conjunções lógicas. No entanto, a representação escolhida não permite contemplar disjunções entre diferentes valores possíveis, para um determinado atributo. Não obstante, este conhecimento pode ser expresso por diferentes regras para a mesma classe (ver exemplo da Figura 5.3).

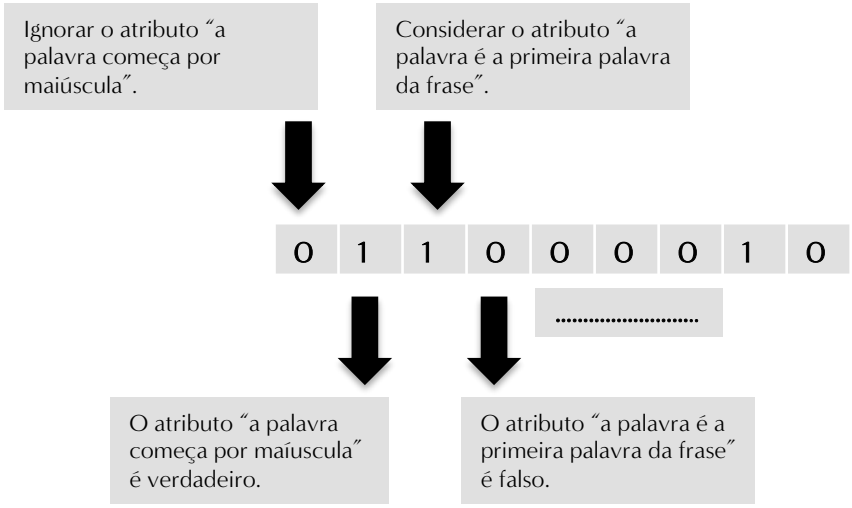


Figura 5.2: Representação binária dos atributos relacionados com as características da palavra.

5.5.2 Representação do consequente

Em geral, existem três formas distintas de representar o valor da classe do objecto alvo de classificação num algoritmo evolucionário [Freitas, 2003]. Uma possibilidade passa por codificar a classe no genoma do indivíduo, submetendo-a ao processo de evolução ([de Jong et al., 1993], [Greene and Smith, 1993]). A segunda, estabelece que todos os indivíduos da população pertencem à mesma classe, a qual se mantém inalterada ao longo da execução do algoritmo. Esta opção determina que, se pretendemos encontrar um conjunto de regras de classificação para predizer a classe de um objecto com k valores distintos, precisamos de correr o algoritmo pelo menos k vezes. Em cada uma das i-nésimas execuções, o algoritmo apenas descobre regras que predizem o i-nésimo valor ([Janikow, 1993]. A terceira alternativa consiste em escolher o valor para a classe de forma mais ou menos determinística. O valor escolhido poderá ser aquele que possui mais exemplos representativos, no conjunto de exemplos que satisfazem o antecedente da regra ([Giordana and Neri, 1995]), ou o valor que maximiza o desempenho do indivíduo ([Noda et al., 1999]).

Como começámos por referir no início desta secção, na estratégia adoptada no trabalho realizado no âmbito desta tese, o problema de classificação foi dividido em n problemas distintos. Cada um deles responsável pela descoberta de regras para cada um dos n objectos de classificação. Determinando, assim, que os valores possíveis das classes pertencem ao conjunto discreto de apenas dois elementos, $Y = \{Sim, Não\}$. Por conseguinte, optámos por seguir a segunda alternativa, já que apenas nos interessa descobrir regras positivas. Desta decisão decorre que não tivemos necessidade de codificar o valor da classe no genoma dos indivíduos.

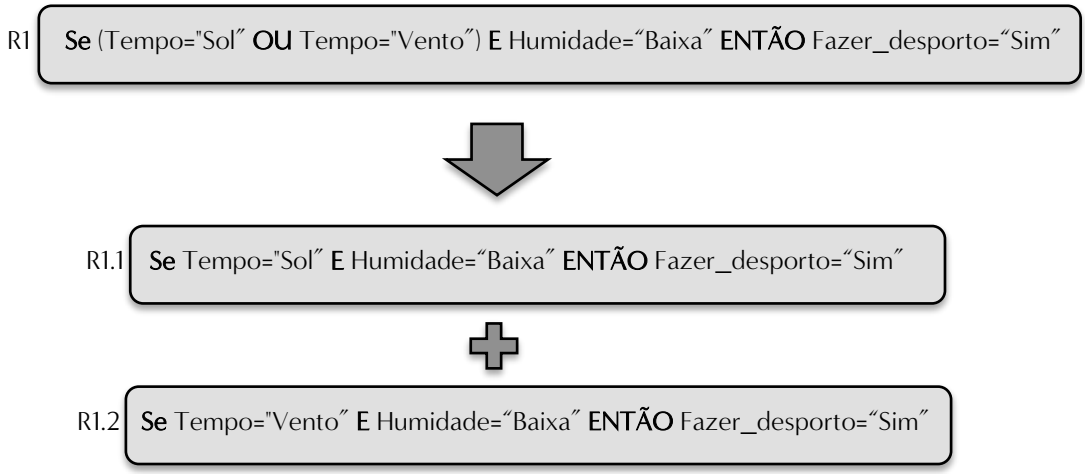


Figura 5.3: Exemplo de desdobramento de uma regra, que contempla conjunções de disjunções de valores de atributos de predição, num conjunto de regras equivalente.

5.6 Construção dos Exemplos de Treino

De forma a ser possível aplicar o algoritmo de classificação foi necessário construir os exemplos de treino a partir do *corpus* escolhido para o efeito. Inicialmente, a informação disponível encontra-se sob a forma de frases, onde cada uma das palavras tem associada uma determinada etiqueta. É por isso necessário processar esta informação de forma a obter um conjunto de instâncias descritas à custa dos atributos de predição previamente definidos.

Para cada uma das palavras do *corpus* de treino foram reunidos os valores dos atributos de predição escolhidos criando assim a partir de cada uma, um exemplo específico. A construção dos exemplos foi feita frase a frase. O contexto de uma determinada palavra deve apenas conter informação sobre as etiquetas das palavras vizinhas, dentro dos limites da frase [Steven Bird and Loper, 2009]. Por este motivo, adoptámos um valor extra para os atributos relacionados com o contexto da palavra: *None*. Este valor deve ser usado para todos os atributos relacionados com o contexto, para os quais não exista a palavra correspondente na frase. Consideremos como exemplos a frase anotada:

$$\langle\langle\ \textit{The}, DET\rangle, \langle\ \textit{cat}, N\rangle, \langle\ \textit{eat}, V\rangle, \langle\ \textit{the}, DET\rangle, \langle\ \textit{fish}, N\rangle\rangle$$

Onde cada palavra tem associada uma etiqueta, representativa da parte do discurso com que está a ser usada na frase ⟨ Palavra, Etiqueta⟩. O exemplo de treino correspondente à primeira palavra, *The*, deveria conter os seguintes valores para os atributos adoptados:

1. Etiqueta da terceira palavra à esquerda = *None*

2. Etiqueta da segunda palavra à esquerda = *None*
3. Etiqueta da primeira palavra à esquerda = *None*
4. Etiqueta da primeira palavra à direita = *N*
5. Etiqueta da segunda palavra à direita = *V*
6. Etiqueta da terceira palavra à direita = *DET*
7. A palavra começa com maiúscula = *Verdadeiro*
8. A palavra é a primera da frase? = *Verdadeiro*
9. A palavra termina em **ed**? = *Falso*
10. A palavra termina em **ing**? = *Falso*
11. A palavra termina em **es**? = *Falso*
12. A palavra termina em **ould**? = *Falso*
13. A palavra termina em **'s**? = *Falso*
14. A palavra termina em **s**? = *Falso*
15. A palavra contém números ou . e números? = *Falso*

A frase em causa estabelece cinco exemplos de treino, um exemplo para cada palavra que a compõe. Generalizando, podemos estabelecer que uma frase com n palavras, determinará n exemplos de treino. No exemplo anterior, usámos as etiquetas N para representar a classe gramatical *Substantivo*, V para representar a classe gramatical *Verbo*, e, finalmente, DET para representar a classe gramatical *Determinante*. Como já dissemos, o valor *None* deve ser usado sempre que o atributo correspondente não se aplique. Esta situação ocorre nas primeiras, e últimas, três palavras de uma frase. A última palavra da frase, *fish*, estabelece o seguinte exemplo de treino:

1. Etiqueta da terceira palavra à esquerda = *N*
2. Etiqueta da segunda palavra à esquerda = *V*
3. Etiqueta da primeira palavra à esquerda = *DET*
4. Etiqueta da primeira palavra à direita = *None*
5. Etiqueta da segunda palavra à direita = *None*

6. Etiqueta da terceira palavra à diretita = *None*
7. A palavra começa com maiúscula = *Falso*
8. A palavra é a primera da frase? = *Falso*
9. A palavra termina em **ed**? = *Falso*
10. A palavra termina em **ing**? = *Falso*
11. A palavra termina em **es**? = *Falso*
12. A palavra termina em **ould**? = *Falso*
13. A palavra termina em **'s**? = *Falso*
14. A palavra termina em **s**? = *Falso*
15. A palavra contém números ou . e números? = *Falso*

Para cada um dos exemplos foram também guardadas a palavra que lhe deu origem, e a etiqueta a ela associada. Um tuplo de tamanho 17 foi usado para guardar esta informação. As primeiras 15 posições correspondem aos atributos de predição, pela ordem em que aparecem na lista anterior, e as duas últimas correspondem à palavra e à etiqueta. Os exemplos anteriores seriam armazenados à custa dos seguintes tuplos:

⟨ *None, None, None, N, V, DET, Verdadeiro, Verdadeiro, Falso, Falso, Falso, Falso, Falso, Falso, Falso, The, DET* ⟩

⟨ *N, V, DET, None, None, None, Falso, Falso, Falso, Falso, Falso, Falso, Falso, Falso, Falso, fish, N* ⟩

Após determinados os exemplos, foi necessário organizá-los em conjuntos diferentes de acordo com o problema de classificação a que dizem respeito. Para cada um dos objectos a classificar foi necessário construir o respectivo conjunto de exemplos. Como os valores possíveis para a classe são *Sim* e *Não*, vamos designar por conjunto de exemplos positivos, o conjunto composto por exemplos classificados com o valor *Sim*, e por conjunto de exemplos negativos, o conjunto formado pelos exemplos classificados com o valor *Não*.

No exemplo anterior foram usadas três etiquetas distintas para marcar as palavras da frase. Vamos admitir que o problema que estamos a tentar resolver consiste apenas em determinar conjuntos de regras para os objectos determinados pelo conjunto de etiquetas $E = \{DET, V, N\}$. Para cada uma das instâncias do problema de classificação, terão que ser construídos os conjuntos de exemplos positivos e negativos. O conjunto de exemplos para o problema de classificação determinado pelo objecto *DET*, deverá

ser composto por exemplos determinados por palavras marcadas com a etiqueta *DET* (exemplos positivos), e por exemplos determinados por palavras marcadas com uma etiqueta diferente de *DET*.

Referimos atrás que a dimensão do conjunto de etiquetas usado na marcação dos diferentes *corpora* anotados, varia de *corpus* para *corpus*. Contudo, os valores típicos encontram-se entre as 20 e as 80 etiquetas. Este facto levanta uma dificuldade na construção dos conjuntos de exemplos positivos e negativos para cada etiqueta, relacionada com a construção dos conjuntos de exemplos negativos. Na situação mais favorável, admitindo um conjunto de 20 etiquetas, o conjunto de exemplos negativos para uma das etiquetas e seria constituído por todos os exemplos determinados por palavras que não estivessem marcadas no *corpus* com a etiqueta e. O resultado seria um conjunto com uma elevada cardinalidade. Uma forma de resolver esta situação seria limitar o conjunto de exemplos negativos às situações mais relevantes para o problema que se pretende resolver.

Interessa neste ponto relembrar que o objectivo deste trabalho não é obter um classificador, mas sim encontrar um conjunto de regras que ajudem a escolher a melhor etiqueta para uma palavra, a partir de um conjunto de etiquetas possíveis. Este conjunto não é, tipicamente, composto por todas as etiquetas do conjunto de etiquetas do *corpus*, mas sim um subconjunto deste último, composto geralmente, por um número pequeno de elementos. Durante o processo de marcação das PdD a marcação das palavras que apenas podem ser usadas com uma etiqueta, é óbvia. As palavras problemáticas são aquelas que são ambíguas. Ou seja, aquelas que podem ser usadas de diferentes formas numa frase. Estamos a falar de todas as palavras para as quais o conjunto de etiquetas possíveis tem cardinalidade superior a 1.

As considerações que acabámos de tecer conduziram-nos à adopção de um mecanismo para a construção dos exemplos de treino diverso do que seria habitual. Este mecanismo baseia-se na assunção de que os exemplos mais relevantes para o problema que pretendemos resolver, são os determinados por palavras ambíguas. Daqui decorre que o processo de construção dos conjuntos de exemplos positivos e negativos deveria apenas ser composto por exemplos derivados de palavras ambíguas. Este processo permite reduzir o tamanho dos conjuntos de exemplos de treino de cada uma das etiquetas, contribuindo assim para uma redução do tempo de execução do algoritmo de classificação.

O algoritmo concebido para definir o conjunto de exemplos positivos, e negativos, para cada uma das etiquetas, foi o seguinte (ver Algoritmo 5.2):

- Para cada um dos exemplos $\langle \mathbf{x}_i, e_i \rangle$ do conjunto total de exemplos, T, determinado pela palavra p_i marcada com a etiqueta $e_i \in E$, o algoritmo investiga se p_i é uma palavra ambígua, verificando se esta surge com mais do que uma hipótese

de marcação no dicionário.

- Se p_i é uma palavra ambígua, e P_i é o conjunto de etiquetas possíveis para essa palavra, então:
  - Coloque-se $\mathbf{x}_i$ no conjunto de exemplos positivos da etiqueta e.
  - Coloque-se $\mathbf{x}_i$ no conjunto de exemplos negativos de todas as etiquetas $t \in P_i$ com $t \neq e$.

Algoritmo 5.2 Algoritmo para construção dos exemplos de treino

```
Entrada: Exemplos, Dicionário
Saída: ExemplosPos, ExemplosNeg
  para e em Exemplos faz
    dim ← dimensão(e)
    palavra ← e[dim−2]
    etiqueta ← e[dim−1]
    se e[:dim−2] ∈ ExemplosPos[etiqueta] então
      ExemplosPos[etiqueta][e[:dim-2]] ← ExemplosPos[etiqueta][e[:dim−2]] +1
    senão
      ExemplosPos[etiqueta][e[:dim-2]] ← 1
    fim se
    possibilidades ← Dicionário[palavra]
    possibilidades.remove(etiqueta)
    para c em possibilidades faz
      se e[:dim−2] ∈ ExemplosNeg[c] então
        ExemplosNeg[c][e[:dim-2]] ← ExemplosNeg[c][e[:dim-2]] +1
      senão
        ExemplosNeg[c][e[:dim-2]] ← 1
      fim se
    fim para
  fim para
```

Como podemos observar no Algoritmo 5.2, a informação que passa a ser guardada em cada um dos exemplos é limitada às primeira 15 posições do 17-tuplo que forma o exemplo original, passando a dizer apenas respeito aos valores dos atributos de predição, ignorando-se a palavra e a etiqueta que lhe deu origem. Como os exemplos positivos e negativos foram organizados em dois conjuntos distintos não houve necessidade de classificar cada um com a classe correspondente (*Sim* ou *Não*), formando os pares $(\mathbf{x}_i, y_i)$ mencionados no início do capítulo. Porém, foram formados pares $(\mathbf{x}_i, o_i)$ com o_i a representar o número de ocorrências da instância $\mathbf{x}_i$. Com esta associação a compactação dos conjunto de exemplos é reforçada, sem se perder no entanto significado estatístico.

5.7 Avaliação das Regras

As regras devem ser avaliadas durante o processo de aprendizagem de forma a estabelecer pontos de referência para o algoritmo de procura. A função usada para avaliar as regras deve considerar, não só as instâncias correctamente classificadas, mas também as que ficaram por classificar e as que foram classificadas erradamente. A fórmula definida para avaliar as regras, e consequentemente determinar a sua qualidade está expressa na Equação 5.1. Na sua definição quisemos destacar a importância dos atributos relacionados com o contexto. Por este motivo, a fórmula penaliza uma solução que codifica uma regra que ignora os primeiros seis atributos de predição (relacionados com o contexto) (Equação 5.2). As restantes são avaliadas por uma função bem conhecida, a *Medida-F*, F_β (5.3). Esta função pode ser interpretada como a média pesada da precisão (5.4) e cobertura (5.5) da regra.

$$Q(X) = \begin{cases} F_\beta(X) & \text{Se contexto(X) = Verdadeiro} \\ -1 & \text{Caso contrário} \end{cases} \tag{5.1}$$

$$contexto(X) = \begin{cases} Verdadeiro & \text{Se } X \text{ testa pelo menos um dos 6 primeiro atributos} \\ Falso & \text{Caso contrário} \end{cases} \tag{5.2}$$

$$F_\beta(X) = (1 + \beta^2) \times \frac{precisao(X) \times cobertura(X)}{\beta^2 \times precisao(X) + cobertura(X)} \tag{5.3}$$

$$precisao(X) = \frac{VP}{VP + FP} \tag{5.4}$$

$$cobertura(X) = \frac{VP}{VP + FN} \tag{5.5}$$

onde:

- VP - Verdadeiros Positivos = número de instâncias cobertas pela regra que são classificadas correctamente, i.e., a classe predita corresponde à classe alvo;
- FP - Falsos Positivos = número de instâncias cobertas pela regra que são classificados erradamente, i.e., a classe predita não coincide com a classe alvo;
- FN - Falsos Negativos = número de instâncias que não são cobertas pela regra, mas cuja classe corresponde à classe alvo.

5.8 Algoritmo Genético

O algoritmo de procura implementado, baseado num algoritmo genético (ver Algoritmo 5.3), recebe como parâmetros de entrada o conjunto de exemplos positivos e negativos do objecto a classificar, assim como o tamanho da população e o número de gerações. A representação escolhida para os indivíduos é a que foi detalhada atrás. Cada indivíduo da população é representado à custa de um cromossoma, o qual codifica uma regra através de um vector binário de tamanho igual a $6 \times (n + 1) + 9 \times 2$, sendo n o número de *bits* necessário para representar o inteiro mais pequeno, maior ou igual à cardinalidade do conjunto de etiquetas usado no *corpus*, a partir do qual se extraíram os exemplos de treino.

Algoritmo 5.3 Algoritmo genético para evolução de regras

Entrada: TamPop, NumGerações, ExemplosPos, ExemplosNeg
Saída: MelhorRegra

```
População ← GeraPopInicial(TamPop, ExemplosPos)
para i←1 to NumGerações faz
  Avalia(População, ExemplosPos, ExemplosNeg)
  Melhor ← MelhorIndivíduo(População)
  PopTemporária ← SelecçãoTorneio(pop)
  PopDescendentes ← AplicaRecombinação(PopTemporária)
  PopDescendentes ← AplicaMutação(NovaPop)
  Pior ← PiorIndivíduo(PopDescendentes )
  População ← Substituir(Pior, Melhor, PopDescendentes )
  i ← i+1
fim para
Melhor ← MelhorIndivíduo(População)
MelhorRegra ← Fenótipo(Melhor)
```

Os indivíduos são avaliados à custa da função de avaliação desenhada para o problema e definida pela Equação 5.1. O esquema de substituição adoptado foi a substituição geracional. Porém, por forma a não ser perdido o melhor indivíduo, de geração para geração, usou-se uma forma de elitismo que garante que o melhor indivíduo de uma geração substitui sempre o pior indivíduo da geração seguinte.

Os aspectos específicos deste algoritmo tais como a geração da população inicial, operadores genéticos e mecanismo de selecção são debatidos nas secções seguintes.

5.8.1 Geração da população inicial

A geração da população inicial segue os passos apresentados no Algoritmo 5.4. Assumindo que a população deverá ser constituída por um total de m indivíduos, metade dos genótipos, $\frac{m}{2}$, são gerados aleatoriamente, atribuindo 0 ou 1 a cada uma das posições do respectivo vector binário, e os restantes são obtidos a partir do conjunto de exemplos positivos. Os exemplos escolhidos são codificados, na representação binária estabelecida, e adicionados à população. A escolha dos exemplos é feita de forma aleatória. Este processo foi adoptado de forma a acelerar o processo de aprendizagem. Pretende-se, assim, saltar um número significativo de primeiras gerações, cujas populações são compostas por indivíduos com desempenho muito pobre.

Algoritmo 5.4 Geração da população inicial

Entrada: TamPop, ExemplosPos
Saída: População

```
k ← Arredonda(TamPop ÷ 2)
para i←1 to k faz
  Indivíduo ← GeraAleatoriamenteInd()
  População ← Adiciona(Indivíduo, População)
  i ← i+1
fim para
para i←1 to TamPop faz
  Exemplo ← EscolheAleatoriamente(ExemplosPos)
  Indivíduo ← Codifica(Exemplo)
  População ← Adiciona(Indivíduo, População)
  i ← i+1
fim para
```

5.8.2 Operadores genéticos

Uma vez que que a representação escolhida para a codificação das regras é uma representação binária típica, não houve necessidade de utilizar operadores de variação específicos. Foram adoptados os operadores tradicionais de mutação binária e de recombinação com dois pontes de corte. A mutação binária utilizada segue o procedimento típico: cada gene de um cromossoma tem uma determinada probabilidade, p_m, de ser mutado; no caso de ocorrer mutação, se o gene possuí o alelo 1, é mutado para 0, e *vice versa* (ver Figura 5.4).

A recombinação com dois pontes de cortes foi também ela implementada seguindo o esquema tradicional:

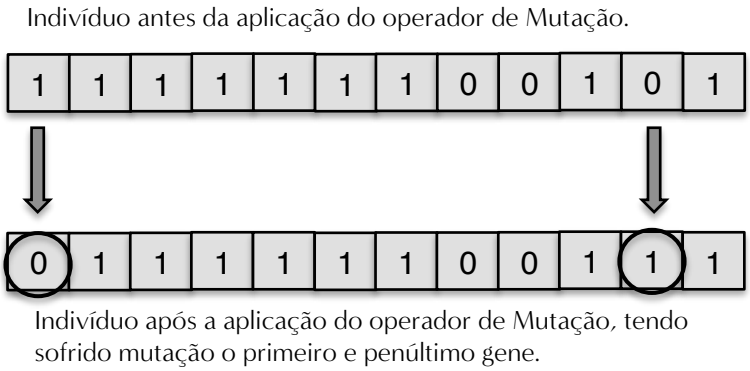


Figura 5.4: Exemplo de aplicação do operador de mutação binária.

- Cada par de indivíduos da população temporária tem uma determinada probabilidade, p_r, de ser recombinado.
- Caso a recombinação aconteça são escolhidos de forma aleatória dois pontos de corte.
- O segmento interior, definido pelos dois pontos de corte, nos cromossomas dos indivíduos, é trocado, dando origem a dois novos cromossomas (ver Figura 5.5), ou seja a dois novo indivíduos.
- Quando a recombinação não ocorre, são colocadas na população de descendentes cópias dos dois indivíduos.

5.8.3 Esquema de selecção

Tendo em conta a sua simplicidade, o mecanismo de selecção adoptado foi a selecção por torneio. Neste esquema de selecção, os indivíduos são escolhidos para a população temporária através da realização de torneios, onde competem uns com os outros. De cada torneio apenas sai um vencedor. Por conseguinte, se pretendemos seleccionar m progenitores para a população temporária, então devem ser realizados m torneios.

O número de indivíduos que competem no torneio pode variar, dependendo da pressão de selecção que se pretende exercer. Quanto maior é o torneio, ou seja quanto maior é o número de indivíduos que competem, maior é a pressão de selecção exercida. A implementação do processo, para torneios com mais que dois competidores, é simples e pode ser descrita pelos seguintes passos:

- Para cada torneio realizado são escolhidos, aleatoriamente, i indivíduos, sendo i o tamanho do torneio, definido previamente.

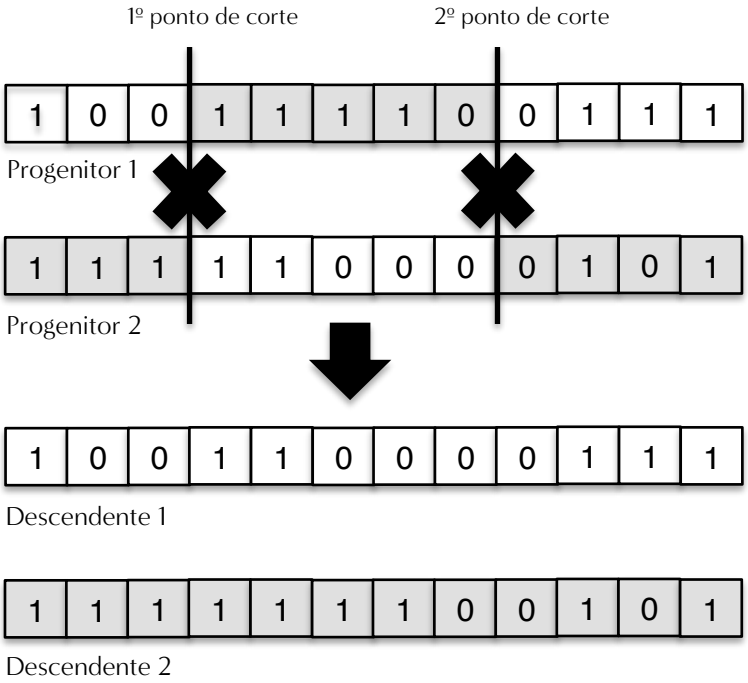


Figura 5.5: Esquema da recombinação com dois pontos de corte.

- O indivíduo que vence o torneio deverá ser o que apresenta melhor valor de desempenho.

Quando $i = 2$, este processo varia ligeiramente. Neste caso, uma constante k é adoptada, e o seu valor irá permitir controlar a pressão de selecção exercida, uma vez que define a probabilidade do melhor indivíduo vencer o torneio. Quanto maior é o valor de k, maior é a pressão de selecção:

- Para cada torneio realizado são escolhidos, aleatoriamente, 2 indivíduos.
- A escolha do vencedor depende de um número $al \in [0, 1]$, gerado aleatoriamente.
- Se $al \leq k$ o indivíduo com melhor desempenho é escolhido como vencedor.
- Se $al > k$ o indivíduo com pior desempenho ganha o torneio.

Neste trabalho optou-se por torneios de tamanho 2 com um valor de $k = 0.8$. O processo utilizado é apresentado no Algoritmo 5.5.

Algoritmo 5.5 Selecção por torneio

```
Entrada: NumProgenitores, População, k
Saída: PopulaçãoTemporária
  para i←1 to NumProgenitores faz
    Indivíduo1 ← SeleccionaAleatoriamente(População)
    Indivíduo2 ← SeleccionaAleatoriamente(População)
    j ← GeraAleatórios(1)
    se j ≤ k então
      Progenitor ← MelhorDesempenho(Indivíduo1, Indivíduo2)
    senão
      Progenitor ← PiorDesempenho(Indivíduo1, Indivíduo2)
    fim se
    PopulaçãoTemporária ← Adiciona(PopTemporária, Progenitor)
    i ← i+1
  fim para
```

5.9 Optimizador Baseado em Partículas de Enxame

Nesta secção iremos apresentar os aspectos específicos da implementação do algoritmo de procura baseado num OEP. O algoritmo implementado foi o da versão binária do optimizador baseado em enxame de partículas, proposto por Kennedy em [Kennedy and Eberhart, 2001]. A estrutura genérica do optimizador encontra-se descrita no Algoritmo 5.6.

Algoritmo 5.6 Algoritmo genérico do optimizador baseado em enxames de partículas

```
Entrada: NumPartículas, NumDimensões, CritérioParagem, k
  Enxame←InicializarEnxame(NumPartículas, NumDimensões)
  repete
    para partícula ∈ Enxame faz
      Avalia(partícula)
      ActualizaExperiênciaPassada(partícula)
      ActualizaMelhorVizinhança(partícula, k)
      para d←1 to NumDimensões faz
        Move(partícula, d)
      fim para
    fim para
  até CritérioParagem
```

Na variante binária do optimizador, cada *bit* é considerado uma dimensão com dois valores possíveis, 0 ou 1. O movimento das partículas é apenas definido pela alternância entre entes dois valores.

Existe um valor de velocidade (v_{id}) associado a cada dimensão/*bit*. Este valor é gerado aleatoriamente a partir do conjunto $[-4.0, 4.0]$ aquando da criação da partícula, e é actualizado iterativamente de acordo com a sua melhor posição (p_{id}) e a melhor posição obtida na vizinhança (p_{gd}), segundo a Equação 5.6. Os valores φ_1 e φ_2 são pesos aleatórios, cujo papel é permitir equilibrar a aprendizagem individual e a influência social.

$$v_{id}(t) = v_{id}(t-1) + \varphi_1(p_{id} - x_{id}(t-1)) + \varphi_2(p_{gd} - xi, d(t-1)) \tag{5.6}$$

Para determinar se um *bit* é alterado, um número aleatório ρ é gerado a partir de uma distribuição uniforme entre 0 e 1, e comparado com o valor normalizado da velocidade associada a essa dimensão, segundo a Equação 5.7.

$$x_{i,d}(t) = \begin{cases} 1 & \text{Se } \rho < S(v_{id}(t)) \\ 0 & \text{Caso contrário} \end{cases} \tag{5.7}$$

A função sigmoide (Equação 5.8) é usada de forma a assegurar que o valor da velocidade é mantido no intervalo $[0.0, 1.0]$.

$$S(v_{id}) = \frac{1}{1 + exp(-v_{id})} \tag{5.8}$$

No optimizador implementado, cada partícula representa uma regra codificada segundo uma representação binária, tal como atrás se referiu. Cada *bit* representa uma dimensão da partícula. O enxame inicial é gerado de forma semelhante à geração da população inicial do algoritmo genético, ou seja, metade das partículas são geradas de forma aleatória, sendo as restantes codificações binárias de exemplos positivos escolhidos aleatoriamente.

Capítulo 6

Algoritmo de Procura para Marcação das PdD

Neste capítulo começamos por apresentar a ideia subjacente à concepção do marcador das PdD, revelando de que forma as regras de desambiguação poderão ser usadas para concretizar a tarefa de marcação. Como iremos ver, foram exploradas duas abordagens distintas para a construção do marcador. Uma resultou da aplicação de um algoritmo genético e a outra de um optimizador baseado em enxames de partículas. A discussão dos aspectos específicos de cada uma das duas abordagens desenrolar-se-à em secções distintas.

6.1 Como Aplicar as Regras de Desambiguação?

Comecemos por relembrar as características de um algoritmo para marcação das PdD. Trata-se de um processo automático, que recebe como entrada uma frase não anotada, definida à custa de um vector de n palavras, $\mathbf{p}$, e deve devolver como saída a mesma frase, mas agora com todas as p_i marcadas com uma etiqueta $e \in E$ adequada.

No capítulo anterior, apresentámos uma abordagem baseada em computação evolucionária para a descoberta de regras de desambiguação, para um conjunto pré-definido de etiquetas E. A finalidade destas regras seria a de auxiliar a resolução do problema de marcação das PdD.

Para cada $e \in E$, foi construído um conjunto de regras de desambiguação, $D = \{D_e \mid e \in E\}$, por aplicação de um algoritmo de classificação baseado em computação evolucionária. Os conjuntos D_e são constituídos por pares $\langle regra_j, valor_j \rangle$, onde $valor_j$ representa a qualidade estimada da regra $regra_j$. Este valor foi determinado pela função usada pelo algoritmo de classificação, para avaliar as regras durante o processo de

aprendizagem (ver Equação 5.1).

Como dissemos no capítulo anterior, o problema da marcação das PdD pode ser visto como um problema de optimização combinatória, o qual, como sabemos, pode ser resolvido à custa de um algoritmo de procura. A hipótese que se defende nesta tese é de que os conjuntos D_e podem servir de base a uma função heurística capaz de orientar um algoritmo de procura no espaço de soluções candidatas para o problema. Este pode ser definido pelo conjunto $P_1 \times P_2 \times \cdots \times P_n$, na assunção de que conhecemos as possibilidades de marcação, P_i, de cada uma das n palavras palavras, p_i, da frase de entrada, $\mathbf{p}$.

As regras presentes nos conjuntos D_e irão ser usadas, não para descobrir a marcação adequada de uma palavra, mas sim para avaliar uma marcação em particular. Vamos admitir que se pretende avaliar a marcação de uma palavra p_i com a etiqueta $e \in P_i$. Assuma-se ainda que $\mathbf{x}_i$, representa o vector de características que define a instância determinada por p_i. Com base nestas premissas, podemos estabelecer que a qualidade da marcação $\langle p_i, e_i \rangle$, poderá ser estimada pelo valor de qualidade *valor* associado a uma eventual regra *regra* tal que $\langle regra, valor \rangle \in D_e$ e a instância $\mathbf{x}_i$ é coberta pela regra *regra*. Este raciocínio pode ser alargado a todas as marcações das palavras de $\mathbf{p}$, donde decorre que o conjunto D, pode ser usado para avaliar cada um dos pontos $\mathbf{z}_i \in P_1 \times P_2 \times \cdots \times P_n$ do espaço de estados do problema de marcação, computando-se para isso o somatório de cada uma das avaliações individuais.

A decisão de aplicar algoritmos de computação evolucionária a este problema foi influenciada pelo facto destes algoritmos mostrarem resultados bastante promissores noutras tarefas de optimização combinatória, incluindo o problema da marcação das PdD ([Araujo, 2002*b*], [Alba et al., 2006]).

Também neste caso investigámos a possibilidade de aplicação de dois algoritmos diferentes: um algoritmo evolucionário, mais concretamente um algoritmo genético, e um optimizador baseado em enxames de partículas. Cada uma das abordagens será detalhada nas próximas secções.

6.2 Algoritmo Genético para Marcação das PdD

O marcador desenvolvido à custa de um algoritmo genético, daqui em diante denominado marcador genético, foi desenhado de forma a receber como entrada uma frase não anotada, sob a forma de um vector de palavras $\mathbf{p}$, um conjunto, D, de conjuntos de regras de desambiguação, D_e, com $e \in E$, e um dicionário DI. O resultado final, retornado pelo marcador, deverá consistir numa frase anotada, **anotada**, derivada da marcação adequada, $\langle p_i, e \rangle$, de cada uma das palavras p_i. O processo utilizado para

efectuar a marcação envolve os seguintes passos (ver Algoritmo 6.1):

- O algoritmo começa por criar um novo vector de palavras, **np**, com todas as palavras de **p**, com excepção da pontuação.

- De seguida, o algoritmo genético evolui um conjunto de soluções candidatas que representam sequências possíveis de etiquetas, $e \in E$, para as palavras np_i. O processo de evolução será guiado pelas regras dos conjuntos D_e.

- Concluído um número pré-estabelecido de gerações, o melhor indivíduo da população, correspondente à ultima geração, estabelece a marcação das PdD que será retornada pelo marcador.

- Após a execução do processo evolutivo, o marcador constrói a frase anotada, **anotada**, atribuindo às palavras np_i as etiquetas codificadas pelo melhor indivíduo da população, assim como a pontuação original devidamente etiquetada. A marcação da pontuação é realizada por simples consulta de uma tabela, onde são armazenadas as etiquetas usadas no *corpus*, para etiquetar os sinais de pontuação.

Algoritmo 6.1 Marcador genético

Entrada: TamPop, NumGerações, Regras, Dicionário, Frase
Saída: Anotada

```
Nova ← RetiraPontuação(Frase)
População ← GeraPopInicial(TamPop, Regras, Dicionário, Nova)
para i ← 1 to NumGerações faz
  Avalia(População, Regras)
  Melhor ← MelhorIndivíduo(População)
  PopTemporária ← SelecçãoTorneio(População)
  PopDescendentes ← AplicaRecombinação(PopTemporária)
  PopDescendentes ← AplicaMutação(PopDescendentes)
  Pior ← PiorIndivíduo(PopDescendentes)
  População ← Substitui(Pior, Melhor, PopDescendentes)
  i ← i+1
fim para
Melhor ← MelhorIndivíduo(População)
Anotada ← IncluiAnotação(Frase, Melhor)
```

Nas secções seguintes iremos debater os aspectos relacionados com a representação, operadores genéticos, esquema de selecção e função de avaliação do algoritmo genético implementado.

6.2.1 Representação

Para a representação dos indivíduos foram testadas duas soluções. A primeira, que passamos a explicar, é semelhante à usada em [Araujo, 2002*b*] e [Alba et al., 2006]. Neste caso, um indivíduo é representado à custa de um cromossoma, **g**, composto por uma sequência de genes. O número de genes de um cromossoma deverá ser igual ao número de palavras da frase de entrada. Cada gene propõe uma etiqueta candidata para a palavra na posição homóloga na frase. Por exemplo, consideremos a frase *"The cat sat on the mat."*. Uma possível representação de um indivíduo, consistiria num cromossoma composto por seis genes, tal como o da figura Figura 6.1.

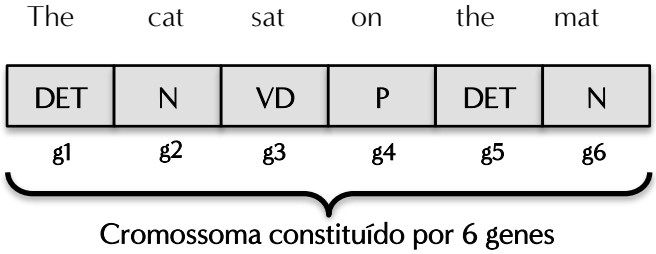


Figura 6.1: Exemplo de um indivíduo para a marcação da frase *"The cat sat on the mat."*.

Na segunda representação considerámos que um cromossoma é apenas composto por um número de genes, igual ao número de palavras ambíguas, e desconhecidas, presentes na frase. O que significa que todas as palavras que apenas têm uma possibilidade de marcação são automaticamente marcadas numa fase inicial. As restantes recebem a etiqueta *'Unkown'*, e vão definir os genes que cada cromossoma deverá ter.

Nos dois casos, para a codificação dos cromossomas adoptámos um representação simbólica. Ou seja, os cromossomas são codificados à custa de um vector cujos valores pertencem a um conjunto finito de valores não numéricos. Os alelos possíveis dos genes que constituem o cromossoma de um indivíduo, correspondem às etiquetas $e \in E$. Todavia, para um particular gene, g_i, os alelos permitidos correspondem, apenas, às etiquetas $e \in P_i$ possíveis para a palavra, np_i na posição homóloga na frase, as quais são obtidas a partir do dicionário DI.

A avaliação dos indivíduos é realizada à custa das regras de desambiguação D. Como tivemos oportunidade de ver no capítulo anterior, estas regras têm 15 atributos de predição: seis relacionados com o contexto, e nove relacionados com determinadas propriedades da palavra. Por conseguinte, de forma a permitir a avaliação dos indivíduos, por aplicação das regras, foi necessário proceder à construção das instâncias $\mathbf{x}_i$, determinadas por cada par $\langle np_i, g_i \rangle$. As instâncias foram assim determinadas a partir da informação da frase representada por **np**, e do genótipo do indivíduo, **g**. Cada par $\langle np_i, g_i \rangle$ determina um 15-tuplo de características com o seguinte alinhamento:

1. A etiqueta proposta pelo terceiro gene à esquerda;
2. A etiqueta proposta pelo segundo gene à esquerda;
3. A etiqueta proposta pelo gene imediatamente à esquerda;
4. A etiqueta proposta pelo gene imediatamente à direita;
5. A etiqueta proposta pelo segundo gene à direita;
6. A etiqueta proposta pelo terceiro gene à direita;
7. $Verdadeiro$ se a palavra np_i começa com letra maiúscula, $Falso$ caso contrário;
8. $Verdadeiro$ se a palavra np_i é a primeira palavra da frase, $Falso$ caso contrário;
9. $Verdadeiro$ se a palavra np_i termina com ***ed***, $Falso$ caso contrário;
10. $Verdadeiro$ se a palavra np_i termina com ***ing***, $Falso$ caso contrário;
11. $Verdadeiro$ se a palavra np_i termina com ***es***, $Falso$ caso contrário;
12. $Verdadeiro$ se a palavra np_i termina com ***ould***, $Falso$ caso contrário;
13. $Verdadeiro$ se a palavra np_i termina com ***'s***, $Falso$ caso contrário;
14. $Verdadeiro$ se a palavra np_i termina com ***s***, $Falso$ caso contrário;
15. $Verdadeiro$ se a palavra np_i contem números, ou '.' e números, $Falso$ caso contrário;

Quando não existe nenhum gene (nenhuma palavra) numa das posições contempladas nos primeiros seis atributos, o atributo correspondente fica com o valor *'None'*. Isto pode acontecer nos primeiros e últimos três genes do cromossoma.

Para a representação em que cada indivíduo é constituído por um cromossoma que apenas propõe etiquetas para as palavras ambíguas, as instâncias são determinadas tendo em conta a marcação inicial e a que é proposta por cada gene.

A representação adoptada determina, portanto, que o genótipo de um indivíduo é constituído pela sequência dos seus genes, os quais sugerem etiquetas $e \in P_i$ para cada uma das palavras, np_i da frase **np**. Numa das opções de representação, o genótipo é constituído por tantos genes quantas as palavras da frase, noutra, apenas por um número de genes igual ao total de palavras desconhecidas e ambíguas. Independentemente deste detalhe de representação, em ambos os casos o fenótipo dos indivíduos será constituído pelo conjunto $F = \{\langle \mathbf{x}_i, e_i \rangle \mid \mathbf{x}_i \in X\}$ de exemplos, de cardinalidade igual a $\mid \mathbf{np} \mid$. A avaliação de um indivíduo será determinada pela avaliação do seu fenótipo.

A necessidade de incluir, em ambos os casos, as etiquetas de todas as palavras da frase (mesmo as das não ambíguas) prende-se pelo facto de a avaliação das instâncias por elas determinadas poder ser influenciada pelas escolhas de etiquetas das suas eventuais vizinhas ambíguas. Deste modo, esta avaliação poderá contribuir para a escolha da etiqueta correcta para as palavras ambíguas, as quais interessa, evidentemente, marcar.

6.2.2 Geração da população inicial

A população inicial, **Pop**, é gerada escolhendo para cada gene g_i do genótipo, $\mathbf{g}$, de um indivíduo, uma das etiquetas $e \in P_i$ possíveis para a palavra np_i na posição homóloga a g_i. O processo utilizado varia consoante np_i está, ou não, presente em DI:

- Se $np_i \in DI$ então P_i é determinado por simples consulta ao dicionário DI, o qual contém, para todas as palavras do *corpus*, as respectivas hipóteses de marcação.
- Se $np_i \notin DI$, o conjunto de etiquetas possíveis P_i será constituído por todas as etiquetas e que indexem um conjunto de regras de desambiguação do conjunto D que contenha pelo menos uma regra que cubra a instância $\mathbf{x}_i$ determinada pela palavra p_i.
- Determinado o conjunto P_i, o algoritmo escolhe, de forma aleatória, uma das etiquetas deste conjunto.
- Se nenhuma regra, de nenhum conjunto, cobre a instância $\mathbf{x}_i$, o alelo do gene g_i será determinado escolhendo, de forma aleatória, uma das etiquetas do conjunto E.

6.2.3 Operadores genéticos e esquema de selecção

Como operadores de variação, foram usados o operador de recombinação com um ponto de corte, e um operador de mutação. O operador de recombinação segue o esquema tradicional da recombinação com um ponto de corte (ver Figura 6.2).

Porém, o operador de mutação foi definido de forma a funcionar de acordo com a representação escolhida. Por conseguinte, a mutação de um gene, g_i, consiste em escolher de forma aleatória um novo alelo para g_i, de entre o conjunto de alelos possíveis. Ou seja, de entre o conjunto de etiquetas possíveis P_i, para a palavra np_i na posição homóloga a g_i (ver Figura 6.3).

Caso np_i não conste do dicionário de entrada DI, o conjunto de alelos possíveis é determinado pela aplicação das regras de desambiguação, tal como definido para a situação idêntica aquando da definição da população inicial.

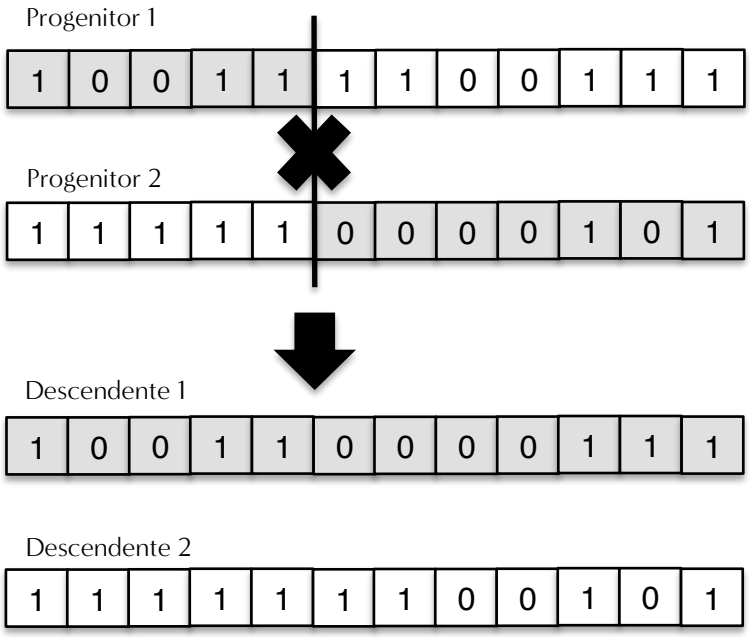


Figura 6.2: Esquema de funcionamento do operador de recombinação com um ponto de corte.

O mecanismo de selecção adoptado foi a selecção por torneio, com torneios de tamanho dois e $k = 0.8$. O esquema de substituição definido, para obter a nova população a partir da população actual e da população de descendentes, seguiu a tradicional substituição geracional. Não obstante, optou-se por usar uma forma de elistimo que consistiu em preservar o melhor indivíduo de cada geração (ver Algoritmo 6.1).

6.2.4 Função de avaliação

O desempenho de um indivíduo da população é dado pela avaliação do seu fenótipo, F, a qual corresponderá à soma da qualidade dos exemplos $\langle \mathbf{x}_i, e_i \rangle \in F$. Consideremos e a etiqueta proposta pelo gene g_i para a palavra np_i, e $\mathbf{x}_i$ o 15-tuplo de características imposto por np_i. Seja D_e o conjunto de regras de desambiguação para a etiqueta e, e $\langle r_k, v_k \rangle \in D_e$ um par ordenado cujo primeiro elemento identifica uma regra de desambiguação que cobre a instância $\mathbf{x}_i$. A avaliação de $\langle \mathbf{x}_i, e \rangle$, é definida pela Equação 6.1.

$$h(\langle \mathbf{x}_i, e \rangle) = \begin{cases} v_k & \text{Se } \langle r_k, v_k \rangle \in D_e \text{ e } r_k \text{ cobre } \mathbf{x}_i \\ 0 & \text{Caso contrário} \end{cases} \tag{6.1}$$

Por conseguinte, a Equação 6.2, define a função que permite avaliar o desempenho de

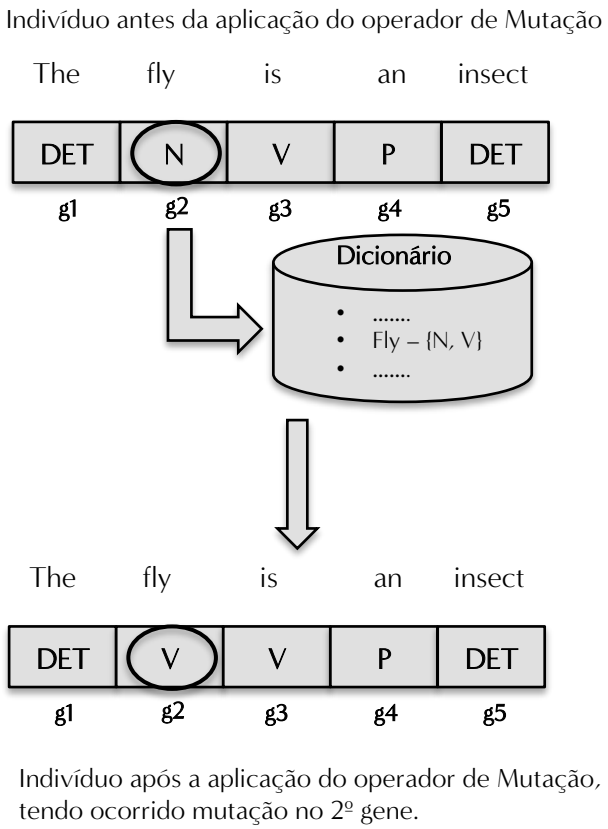


Figura 6.3: Esquema de funcionamento do operador de mutação. O exemplo mostra um caso em que o segundo gene de um indivíduo foi alvo de mutação.

um indivíduo, à custa da avaliação do seu fenótipo F.

$$Desempenho(F) = \sum_{j=1}^{n} h(F_j) \tag{6.2}$$

6.3 OEP para Marcação das PdD

O marcador baseado em optimização por enxames de partículas, daqui em diante designado por marcador OEP, foi desenhado à semelhança do marcador genético. Assim, tal como antes, o marcador recebe como entrada uma frase não anotada, **p**, um conjunto de conjuntos de regras de desambiguação, $D = \{D_e \mid e \in E\}$, e um dicionário DI. O resultado final, retornado pelo marcador, deverá consistir na anotação da frase de entrada, **anotada**, que resulta da marcação adequada, $\langle p_i, e\rangle$ com $e \in E$, de cada uma das n palavras p_i de **p**. Os seguintes passos permitem descrever o comportamento do algoritmo definido:

- O algoritmo começa por retirar da frase de entrada os sinais de pontuação, obtendo uma nova frase, **np**.
- De seguida investiga quais as palavras não ambíguas de **np**, marcando-as com a respectiva etiqueta. Para isso, simplesmente consulta o dicionário, DI, que recebe como entrada, marcando cada uma delas com a única etiqueta com que aparece associada no dicionário. As restante palavras recebem a etiqueta *'Unkown'*.
- O algoritmo prossegue com a execução do optimizador baseado em enxames de partículas, de forma a determinar as etiquetas das palavras que ficaram por marcar. A procura da solução, no espaço de estados do problema, é guiada, tal como no marcador genético, pelas regras de desambiguação presentes no conjunto de regras de entrada, D.

A versão utilizada do optimizador baseado em enxames de partículas foi a versão binária do algoritmo. O optimizador evolui um enxame de partículas que codificam, cada uma delas, uma sequência de etiquetas para marcar as palavras ambíguas da frase de entrada, **p**. A aplicação do OEP binário condicionou a representação escolhida para as partículas, tendo por isso sido adoptada uma codificação diferente da usada no algoritmo genético. Nas secções seguintes iremos, precisamente, discutir os aspectos relacionados com a representação e com a forma como foram avaliadas as partículas.

6.3.1 Representação

A aplicação do OEP binário conduziu à necessidade de adoptar uma representação binária para as partículas. A codificação de cada uma das $e \in E$, foi conseguida à custa de k *bits*, com k a representar o número de *bits* necessário para codificar o binário mais pequeno, maior ou igual à cardinalidade do conjunto E. Daqui decorre que uma partícula que proponha a anotação de uma frase com m palavras ambíguas, deverá ser composta por $m \times k$ dimensões.

Cada sequência de k dimensões de uma partícula codifica um número binário, b, cujo correspondente valor decimal, d, indexa uma tabela t com tantas entradas quantas as etiquetas possíveis, $e \in P_i$, da palavra p_i para a qual está a propor a marcação, ou seja a dimensão de t é $dim =\mid P_i \mid$. Caso $d > dim$, o resto da divisão inteira de d por dim é adoptado como índice.

6.3.2 Avaliação das partículas

A qualidade das partículas é determinada pela marcação que sugerem para as palavras ambíguas da frase de entrada. Tal como no marcador genético, a avaliação é realizada

à custa das regras de desambiguação presentes em D. A avaliação de uma partícula pressupõe a sua prévia descodificação num vector de etiquetas, $\mathbf{v}$. As etiquetas v_i serão usadas para etiquetar as palavras da frase **np** que aparecem marcadas com a etiqueta *'Unkown'*, segundo a ordem em que ocorrem em **np**.

Após a anotação completa da frase **np** o conjunto $T = \{\langle \mathbf{x}_i, e\rangle \mid \mathbf{x}_i \in X\}$ de exemplos é construído. Cada anotação $\langle p_i, e\rangle$ em **np**, estabelece um vector de característica $\mathbf{x}_i \in X$ com o seguinte alinhamento:

1. A etiqueta da terceira palavra à esquerda de np_i;
2. A etiqueta da segunda palavra à esquerda de np_i;
3. A etiqueta da palavra imediatamente à esquerda de np_i;
4. A etiqueta da palavra imediatamente à direita de np_i;
5. A etiqueta da segunda palavra à direita de np_i;
6. A etiqueta da terceira palavra à direita de np_i;
7. $Verdadeiro$ se a palavra np_i começa com letra maiúscula, $Falso$ caso contrário;
8. $Verdadeiro$ se a palavra np_i é a primeira palavra da frase, $Falso$ caso contrário;
9. $Verdadeiro$ se a palavra np_i termina com ***ed***, $Falso$ caso contrário;
10. $Verdadeiro$ se a palavra np_i termina com ***ing***, $Falso$ caso contrário;
11. $Verdadeiro$ se a palavra np_i termina com ***es***, $Falso$ caso contrário;
12. $Verdadeiro$ se a palavra np_i termina com ***ould***, $Falso$ caso contrário;
13. $Verdadeiro$ se a palavra np_i termina com ***'s***, $Falso$ caso contrário;
14. $Verdadeiro$ se a palavra np_i termina com ***s***, $Falso$ caso contrário;
15. $Verdadeiro$ se a palavra np_i contem números, ou '.' e números, $Falso$ caso contrário;

Tal como no marcador genético, quando não existe palavra na posição contemplada no contexto, adoptamos um valor extra igual a *'None'*.

A avaliação de uma qualquer partícula do enxame de partículas é estabelecida de forma semelhante à usada para avaliar o fenótipo de um indivíduo no marcador genético (ver Equação 6.4). Neste caso, a qualidade da partícula é calculada a partir do vector de

etiquetas **v** que codifica, e resulta do somatória das qualidades estimadas de cada um dos exemplos $\langle \mathbf{x}_i, e \rangle \in T$

$$h(\langle \mathbf{x}_i, e \rangle) = \begin{cases} v_k & \text{Se } \langle r_k, v_k \rangle \in D_e \text{ e } r_k \text{ cobre } \mathbf{x}_i \\ 0 & \text{Caso contrário} \end{cases} \tag{6.3}$$

$$Qualidade(T) = \sum_{j=1}^{n} h(T_j) \tag{6.4}$$

O trabalho experimental desenvolvido para testar os marcadores das PdD foi realizado para a língua Inglesa e será apresentado com detalhe no próximo capítulo. As experiências realizadas envolveram duas fases distintas. A primeira parte centrou-se na descoberta de regras de desambiguação por aplicação do algoritmo de classificação descrito no capítulo anterior. Na segunda, testaram-se os marcadores em dois *corpora* usando as regras descobertas na primeira fase. Apresentamos ainda os resultados obtidos num caso de estudo de aplicação da abordagem à língua Portuguesa.

Capítulo 7

Resultados Experimentais

Neste capítulo, iremos apresentar os resultados experimentais obtidos com a execução do algoritmo de extracção de regras de desambiguação, assim como os alcançados com o marcador das PdD, quando alimentado com as regras previamente extraídas. Antes de discutir estes resultados, apresentaremos uma descrição breve dos *corpora* utilizados nas referidas experiências, bem como do ambiente experimental no qual os algoritmos foram implementados e as experiências realizadas.

7.1 Ambiente Experimental

Os algoritmos descritos nos capítulos anteriores, tanto para extracção de regras de desambiguação, como para marcação das PdD, foram implementados na linguagem de programação *Python*. Foram também utilizados os recursos disponíveis no pacote de software NLTK (*Natural Language Toolkit*). Este pacote disponibiliza para a língua Inglesa, entre outros, o *corpus* de *Brown* e uma amostra de 10% do *corpus* do *Wall Street Journal* (WSJ) do projecto *Penn Treebank*. Estes *corpora* são os mais, frequentemente, utilizados no desenvolvimento de marcadores. Além de disponibilizar *corpora* para a língua Inglesa, o NLTK, oferece *corpora* noutras línguas, em particular a língua Portuguesa, como o *corpus Mac-Morpho*.

7.2 *Corpora* Utilizados no Trabalho Experimental

Como *corpora* de teste para os nossos algoritmos, para a língua inglesa foram escolhidos o *corpus* de *Brown* [F.W. Nelson, 1979] e o *corpus* do WSJ do *Penn Treebank* [M.P. Marcus, 1994]. Para a língua Portuguesa o *corpus* escolhido foi o *corpus Mac-*

Morpho [Aluísio et al., 2003].

O *corpus Brown University Standard Corpus of Present-Day American English*, ou simplesmente *corpus* de *Brown*, foi compilado em 1960 por Henry Kucera e W. Nelson Francis da Universidade de *Brown*, em *Rhode Island*. Foi o primeiro *corpus* de língua Inglesa a ser desenvolvido para análise computacional. Contém 500 exemplos de textos em prosa, compilados de trabalhos publicados nos Estados Unidos da América em 1961.

Cada amostra contém 2000 ou mais palavras, terminando na frase que perfaz, ou ultrapassa, as 2000 palavras. Garante, assim, que as amostras apenas contêm frases completas. Este *corpus* incluí cerca de um milhão de palavras.

Inicialmente o *corpus* era constituído apenas pelas palavras, tendo as PdD sido introduzidas mais tarde. De facto o *corpus* tem sido alvo de numerosas tentativas de marcação, tendo sido a primeira tentativa realizada pelo programa *Greene and Rubin*. Este programa conseguiu na altura uma exactidão (dada pela percentagem de palavras marcadas correctamente) de cerca de 70%. Os resultados conseguidos pelo programa foram repetidamente corrigidos e revistos manualmente. No final da década de 70, a marcação foi considerada quase perfeita.

O *corpus* tem sido usado em vários trabalhos que estudam a frequência de palavras e a marcação das PdD, e inspirou o desenvolvimento de *corpora* semelhantes em várias outras línguas. Os dados estatísticos recolhidos da sua análise estiveram na base de vários dos sistemas de marcação das PdD que surgiram mais tarde.

A versão anotada do *corpus* de *Brown* usa uma selecção de cerca de 80 etiquetas, assim como indicadores especiais para formas compostas, contracções, palavras estrangeiras, e outros fenómenos.

O *corpus* continha, originalmente, 1014312 palavras reunidas a partir de 15 categorias de textos:

- Imprensa - Reportagem: categoria constituída por 44 textos de áreas tão diversas como política, desporto, sociedade, notícias locais, finança e cultura;
- Imprensa - Editorial: categoria constituída por 27 textos de secções várias como institucional, classificados e cartas ao editor;
- Imprensa - Crítica: categoria constituída por 17 textos de criticas em diversas áreas como teatro, livros, música e dança;
- Religião: categoria constituída por 17 textos retirados de livros, periódicos e brochuras;
- Passatempos e habilidades: categoria constituída por 36 textos retirados de livros e periódicos;

- Sabedoria popular: categoria constituída por 48 textos retirados de livros e periódicos;
- Literatura - Biografia e Memórias : categoria constituído por 75 textos;
- Miscelâneas: categoria constituída por 30 textos, retirados de documentos do governo, relatórios de fundações, relatórios da indústria, catálogos de colégios e periódicos da indústria;
- Aprendizagem: categoria constituída por 80 textos, retirados de áreas como ciências naturais, medicina, matemática, ciências sociais, ciências políticas, direito, educação, humanidades, tecnologia e engenharia.
- Ficção genérica: categoria constituída por 6 textos retirados de novelas e histórias curtas (contos).
- Ficção mistério e detectives: categoria constituída por 24 textos retirados de novelas e histórias curtas (contos).
- Ficção aventura e *western*: categoria constituída por 29 textos retirados de novelas e histórias curtas (contos).
- Ficção romance: categoria constituída por 29 textos retirados de novelas e histórias curtas (contos).
- Humor: categoria constituída po 9 textos retirados de novelas, ensaios, etc.

O segundo *corpus* que utilizámos, o *corpus* do *Penn Treebank*, reúne cerca de 4.5 milhões de palavras em língua Inglesa. Durante os primeiros três anos do projecto (1989-1992), o *corpus* foi anotado com as PdD respectivas. Embora baseada no *corpus* de *Brown*, a marcação do *corpus* do *Penn treebank* adoptou um conjunto de etiquetas consideravelmente reduzido. O conjunto adoptado é constituído por 36 etiquetas relacionadas com as PdD, e 12 etiquetas adicionais para marcação da pontuação e símbolos monetários. A versão anotada do *corpus* foi produzida em duas fases, usando uma combinação de marcação automática e de correcções manuais.

O *corpus* do *Penn Treebank* reúne um conjunto vasto de textos de várias fontes, tais como:

- O *corpus* de *Brown*, com uma nova marcação. O *corpus* foi novamente marcado pelo projecto *Penn Treebank*, usando como ponto de partida a versão original (1964), não anotada do *corpus*;
- O *corpus* do *Wall Street Journal*, constituído por noticias do *Wall Street Journal*;

- Resumos do Departamento de Energia, os quais reúnem resumos científicos de várias disciplinas do conhecimento;
- Boletins do Departamento da Agricultura, incluindo boletins curtos sobre vários tópicos que vão, desde quando plantar determinadas plantas, até como enlatar determinados frutos e vegetais;
- Textos da Biblioteca da América, constituído por passagens de 5000-10000 palavras, predominantemente, capítulos de livros de uma variedade de autores americanos, incluindo Mark Twain, Henry Adams, Willa Cather, Herman Melville, W. E. B. Dubois, e Ralph Waldo Emerson;
- Textos da MUC-3, constituído por textos de notícias do Serviço Nacional de Notícias acerca de actividades terroristas na América do Sul. Alguns destes textos são traduções de noticias em Espanhol, ou transcrições de transmissões de rádio. Foram retirados do material de treino disponível para a *Third Message Understanding Conference (MUC-3)*;
- Frases retiradas dos manuais de computador da IBM. Estes textos foram escolhidos de forma a conterem um vocabulário de cerca de 3000 palavras;
- Frases da ATIS, consistem em versões transcritas de frases espontâneas reunidas como material de treino para o projecto DARPA *Air Travel Information System (ATIS).*

Finalmente, o *corpus Mac-Morpho* é um *corpus* em Português do Brasil que reúne cerca de 1.1 milhões de palavras do jornal Folha de São Paulo. Este *corpus* resulta de um projecto de dois anos, com o nome *Lacio-Web*, que teve início em 2002. O conjunto de etiquetas adoptado pelo projecto *Lacio-Web* tem sofrido várias revisões. No entanto, a versão usada nas experiências realizadas no âmbito deste trabalho, diz respeito à nona revisão. Nesta versão, o conjunto de etiquetas regulares é composto por 22 etiquetas (ver Tabela 7.1), em conjunto com 9 etiquetas complementares. Estas últimas complementam a informação das etiquetas regulares, sendo concatenadas a estas usando como separador o símbolo '|'.

7.3 Aprendizagem de Regras de Desambiguação

Nesta secção iremos apresentar os resultados obtidos com o algoritmo de classificação usado para a descoberta de regras de desambiguação. Como foram implementados dois algoritmos de procura distintos para a aprendizagem das regras, iremos apresentar os resultados correspondentes em duas secções distintas. Começaremos por apresentar os

Tabela 7.1: Conjunto de etiquetas regulares do *copus Mac-Morpho*

Etiqueta	Definição
ADJ	Classe aberta para modificador de nome
ADV-KS-REL	Advérbio relativo subordinativo
ADV-KS	Advérbio não relativo subordinativo
ADV	Advérbio não subordinativo
ART	Artigo
KC	Conjunções coordenativas
KS	Conjunções coordenativas
IN	Interjeição
N	Nome nuclear da frase
NPROP	Nome próprio
NUM	Numeral como modificador de nome
PCP	Particípio passado ou adjectivo
PDEN	Enfatização
PREP	Preposição
PROPESS	Pronome pessoal
PRO-KS-REL	Pronome relativo subordinativo
PRO-KS	Pronome não relativo subordinativo
PROSUB	Pronome não subordinativo como nome nuclear da frase
PROADJ	Pronome não subordinativo como modificador
VAUX	Verbo auxiliar
V	Verbo não auxiliar
CUR	Símbolo monetário

resultados obtidos com a aplicação do algoritmo genético, e de seguida os conseguidos com o optimizador baseado em enxames de partículas.

Como vimos na secção anterior, os *corpora* disponíveis usam de uma forma geral diferentes formas de marcar as palavras com as respectivas PdD. Estas diferenças reflectem-se, não só ao nível do conjunto de etiquetas, mas também ao nível do formato adoptado para representar a anotação. Neste sentido, o pacote de *software* NLTK revelou-se bastante útil, dado que disponibiliza vários módulos em *Python* para processar os diferentes *corpora*, permitindo deste modo a utilização de uma *interface* uniforme.

O facto do conjunto de etiquetas adoptado nos diferentes *corpora* ser diverso, condiciona a utilização das regras de desambiguação descobertas pelo algoritmo de classificação. Como sabemos, a estratégia adoptada estabelece que o algoritmo deve ser executado para cada uma das etiquetas do conjunto de etiquetas considerado no *corpus* de treino. A marcação das PdD realizada pelos marcadores desenvolvidos fica assim limitada às etiquetas admitidas pelo algoritmo de classificação. No entanto, se pretendermos testar os marcadores noutro *corpus*, distinto do usado para obter as regras de desambiguação, não teremos forma de avaliar os resultados obtidos, sem sermos obrigados a aprender novas regras. Porém, um dos objectivos deste trabalho é, precisamente, averiguar se as regras obtidas são suficientemente genéricas, para poderem ser utilizadas na marcação das PdD de frases pertencentes a diversos *corpora*.

No sentido de evitar a situação descrita, decidimos usar a opção *simplify_tags=True* do módulo *tagged_sentence* do *corpus reader* do pacote de *software* NLTK. Quando esta opção é definida com o valor *True*, o NLTK converte o respectivo conjunto de etiquetas do *corpus* num conjunto uniforme e simplificado de etiquetas, composto por 20 elementos (ver Tabela 7.2). Este foi o conjunto que usámos no trabalho experimental e que determinou a definição dos problemas de classificação.

As experiências executadas no âmbito da aplicação do algoritmo de classificação para as descoberta de regras de desambiguação, foram conduzidas usando como recurso linguístico o *corpus* de *Brown*. Os exemplos de treino considerados foram construídos a partir de 90% do *corpus*. A definição dos conjuntos de exemplos, positivos e negativos, para cada uma das etiquetas da Tabela 7.2, foi conseguida à custa da aplicação do Algoritmo 5.2, o qual foi descrito no Capítulo 5. O total de exemplos, positivos e negativos, conseguido para cada uma das etiquetas pode ser consultado na Tabela 7.3.

Ponderando o número elevado de exemplos de treino, tomámos a decisão de limitar a cardinalidade dos conjuntos a valores pré-estabelecidos que considerámos razoáveis. Esta decisão pesou sobretudo os custos computacionais impostos pela utilização de conjuntos de cardinalidade elevada nos algoritmos de classificação. Foram assim adoptados conjuntos de diferentes cardinalidades, construídos a partir dos $3E4$, $4E4$, $5E4$, $6E4$, $7E4$ e $8E4$ primeiros exemplos de treino (ver Tabela 7.4). A cardinalidade dos

Tabela 7.2: Conjunto simplificado de etiquetas utilizado para os *corpora* em língua Inglesa.

Etiqueta	Significado	Exemplos de palavras
ADJ	Adjectivo	*new, good, high, special, big, local*
ADV	Advérbio	*really, already, still, early, now*
CNJ	Conjunção	*and, or, but, if, while, although*
DET	Determinante	*the, a, some, most, every, no*
EX	Existencial	*there, there's*
FW	Palavra estrangeira	*dolce, ersatz, esprit, quo, maitre*
MOD	Verbo modal	*will, can, would, may, must, should*
N	Nome	*year, home, costs, time, education*
NP	Nome próprio	*Alison, Africa, April, Washington*
NUM	Número	*twenty-four, fourth, 1991, 14:24*
PRO	Pronome	*he, their, her, its, my, I, us*
P	Preposição	*on, of, at, with, by, into, under*
TO	A palavra to	*to*
UH	Interjeição	*ah, bang, ha, whee, hmpf, oops*
V	Verbo	*is, has, get, do, make, see, run*
VBZ	Verbo na 3ª pessoa do singular	*gives, likes, sees*
VD	Verbo no tempo passado	*said, took, told, made, asked*
VG	Verbo no particípio presente	*making, going, playing, working*
VN	Verbo no particípio passado	*given, taken, begun, sung*
WH	Determinador wh	*who, which, when, what, where, how*

Tabela 7.3: Número de exemplos, positivos e negativos, obtido para cada uma das etiquetas através do Algoritmo 5.2, quando aplicado às palavras que constituem 90% do *corpus* de *Brown*.

Etiqueta	Exemplos Positivos	Exemplos Negativos
ADJ	35212	22704
ADV	26298	58093
CNJ	34546	16600
DET	53926	30355
EX	1091	431
FW	978	15533
MOD	8029	514
N	112434	60294
NP	26984	7300
NUM	11881	758
PRO	34562	3628
P	48517	26142
TO	8330	7454
UH	348	8492
V	50636	37747
VBZ	6005	6841
VD	15880	14992
VG	11814	2696
VN	17165	14817
WH	9696	6846

Tabela 7.4: Designação escolhida para os conjuntos de exemplos de treino usados no trabalho experimental.

Designação do conjunto	Cardinalidade
A	30000
B	40000
C	50000
D	60000
E	70000
F	80000

conjuntos de exemplos positivos e negativos, para cada uma das etiquetas consideradas, para cada um dos conjuntos referidos, pode ser consultada nas Tabelas 7.5, 7.6, e 7.7. Importa relembrar que os valores apresentados se referem ao número de exemplos distintos considerados. Isto porque o algoritmo concebido para a definição dos exemplos positivos e negativos guarda, para cada exemplo encontrado, o número de vezes que este ocorre no conjunto total de exemplos considerado.

Tabela 7.5: Número de exemplos, positivos e negativos, obtidos através do Algoritmo 5.2, para cada uma das etiquetas, quando aplicado aos primeiros $3E4$ (conjunto A) e $4E4$ (conjunto B) exemplos de treino.

	Conjunto A		Conjunto B	
Etiqueta	Exemplos+	Exemplos−	Exemplos+	Exemplos−
ADJ	1508	869	1915	1113
ADV	832	2874	1099	3600
CNJ	1229	727	1623	919
DET	3182	961	3945	1191
EX	48	10	61	14
FW	8	794	14	999
MOD	419	24	540	32
N	6414	2911	8269	3749
NP	2233	307	3332	429
NUM	854	11	1053	13
PRO	1047	136	1341	165
P	2928	1055	3770	1305
TO	430	277	503	360
UH	3	208	6	248
V	2034	1870	2607	2388
VBZ	175	284	235	339
VD	941	644	1136	830
VG	502	90	635	124
VN	712	910	954	1091
WH	360	239	475	302

Tabela 7.6: Número de exemplos, positivos e negativos, obtidos através do Algoritmo 5.2, para cada uma das etiquetas, quando aplicado aos primeiros $5E4$ (conjunto C) e $6E4$ (conjunto D) exemplos de treino.

	Conjunto C		Conjunto D	
Etiqueta	Exemplos+	Exemplos−	Exemplos+	Exemplos−
ADJ	2283	1380	2767	1654
ADV	1333	4302	1601	5168
CNJ	2005	1101	2436	1335
DET	4641	1411	5394	1699
EX	70	15	78	21
FW	19	1236	21	1449
MOD	608	37	690	41
N	10352	4475	12365	5348
NP	4139	587	4709	687
NUM	1318	14	1559	20
PRO	1593	193	1922	215
P	4658	1545	5542	1856
TO	598	455	697	556
UH	6	292	7	369
V	3132	2868	3759	3391
VBZ	259	403	342	499
VD	1479	1052	1671	1223
VG	782	170	934	215
VN	1227	1404	1445	1572
WH	578	358	673	431

Nas próximas secções apresentaremos os resultados obtidos pelas duas implementações do algoritmo de classificação. Ambas resultaram da adopção de um algoritmo de procura diferente para a descoberta das regras de desambiguação. Numa a procura foi realizada à custa do algoritmo genético descrito no Capítulo 5, e na outra à custa de um optimizador baseado em enxames de partículas, também já discutido no mesmo capítulo. As experiências realizadas foram conduzidas usando os mesmos conjuntos de exemplos de treino, sendo estes os que acabámos de definir. Os resultados serão apresentados sob a forma do número de regras descobertas para cada uma das etiquetas. A eficiência de cada um destes conjuntos, para guiar a procura realizada pelos marcadores das PdD definidos, será discutida à custa dos resultados alcançados pelos mesmos.

7.3.1 Resultados obtidos com o algoritmo genético

Foram realizados dois *runs* para cada um dos conjuntos de exemplos de treino apresentados nas Tabelas 7.5, 7.6 e 7.7. A decisão de efectuar apenas dois *runs* baseou-se sobretudo nos custos computacionais elevados do processo de aprendizagem. Contudo, quisemos investigar se os resultados apresentavam diferenças significativas entre execuções distintas, tendo por isso optado por repetir cada uma das experiências realizadas.

O algoritmo genético foi executado com uma população de 200 indivíduos durante 80 gerações. As probabilidades estabelecidas para o operador de recombinação e mutação foram de 0.75 e 0.01, respectivamente. O mecanismo de selecção por torneio foi implementado com um valor de $k = 0.8$. Este valores foram determinados a partir de um conjunto preliminar de experiências. O número de regras obtido, para cada uma das etiquetas, em cada um dos *runs* realizados pode ser consultado nas Tabelas 7.8, 7.9 e 7.10.

7.3.2 Resultados obtidos com o OEP

O algoritmo de classificação definido com o optimizador baseado em enxame de partículas, para efectuar a procura das regras de desambiguação, foi executado duas vezes para cada um dos conjuntos A, B, C, D, E e F. O OEP foi configurado com um enxame de 20 partículas, e com um número máximo de gerações igual a 200. Estes valores foram determinados à custa de um conjunto preliminar de experiências. O número total de regras descobertas em cada um dos *runs*, para cada uma das etiquetas, encontram-se nas Tabelas 7.11, 7.12 e 7.13.

Tabela 7.7: Número de exemplos, positivos e negativos, obtidos através do Algoritmo 5.2, para cada uma das etiquetas, quando aplicado aos primeiros $7E4$ (conjunto E) e $8E4$ (conjunto F) exemplos de treino.

	Conjunto E		Conjunto F	
Etiqueta	Exemplos+	Exemplos−	Exemplos+	Exemplos−
ADJ	3357	1885	3906	2169
ADV	1883	6000	2245	6891
CNJ	2841	1538	3277	1784
DET	6299	2037	7260	2426
EX	103	22	113	26
FW	59	1682	68	1931
MOD	804	50	877	54
N	14085	6119	15686	6999
NP	5188	878	5827	987
NUM	1715	25	2012	31
PRO	2251	256	2654	310
P	6268	2159	7019	2514
TO	816	648	948	748
UH	13	447	13	514
V	4392	3894	5031	4415
VBZ	403	569	472	652
VD	1832	1403	2127	1586
VG	1052	275	1199	309
VN	1674	1723	1869	1988
WH	788	534	912	619

Tabela 7.8: Número de regras descobertas pelo algoritmo de classificações quando a procura foi realizada à custa do algoritmo genético, usando os conjuntos de exemplos de treino da Tabela 7.5 como parâmetros de entrada. Os resultados apresentados correspondem aos dois *runs* realizados para cada um dos conjuntos (A e B).

	Conjunto A		Conjunto B	
Etiqueta	*Run* AG A.1	*Run* AG A.2	*Run* AG B.1	*Run* AG B.2
ADJ	227	227	207	253
ADV	284	270	325	331
CNJ	232	228	263	255
DET	220	217	220	265
EX	8	7	10	9
FW	4	3	10	7
MOD	14	14	18	19
N	458	446	519	501
NP	194	177	199	193
NUM	42	44	51	54
PRO	114	123	119	116
P	158	162	208	200
TO	6	5	10	7
UH	3	3	5	5
V	252	248	289	291
VBZ	86	78	93	99
VD	140	140	151	148
VG	68	72	86	73
VN	138	140	166	172
WH	92	94	110	105
Total	2740	2698	3059	3103
Média	2719		3081	

Tabela 7.9: Número de regras descobertas pelo algoritmo de classificação quando a procura foi realizada à custa do algoritmo genético, usando os conjuntos de exemplos de treino da Tabela 7.6 como parâmetros de entrada. Os resultados apresentados correspondem aos dois *runs* realizados para cada um dos conjuntos (C e D).

	Conjunto C		Conjunto D	
Etiqueta	*Run* AG C.1	*Run* AG C.2	*Run* AG D.1	*Run* AG D.2
ADJ	266	269	302	306
ADV	352	374	411	426
CNJ	269	266	305	317
DET	275	277	295	315
EX	12	10	12	10
FW	12	12	11	11
MOD	17	18	19	19
N	557	555	592	625
NP	232	217	262	221
NUM	54	46	54	58
PRO	133	127	133	138
P	211	213	233	203
TO	8	8	12	11
UH	5	5	6	6
V	317	323	364	336
VBZ	98	81	101	96
VD	139	172	169	190
VG	83	87	110	103
VN	191	188	207	204
WH	124	114	144	134
Total	3355	3362	3742	3729
Média	3358.5		3735.5	

Tabela 7.10: Número de regras descobertas pelo algoritmo de classificação quando a procura foi realizada pelo algoritmo genético, usando os conjuntos de exemplos de treino da Tabela 7.7 como parâmetro de entrada. Os resultados apresentados correspondem aos dois *runs* realizados para cada um dos conjuntos (E e F).

	Conjunto C		Conjunto D	
Etiqueta	*Run* AG E.1	*Run* AG E.2	*Run* AG F.1	*Run* AG F.2
ADJ	309	301	361	345
ADV	472	476	526	490
CNJ	331	337	352	335
DET	354	345	381	390
EX	11	12	13	13
FW	19	19	19	22
MOD	19	17	22	24
N	657	638	670	690
NP	286	263	290	276
NUM	61	66	73	67
PRO	152	145	246	231
P	267	230	246	231
TO	10	10	10	11
UH	12	10	10	11
V	397	387	401	410
VBZ	114	119	136	123
VD	199	204	212	202
VG	120	128	120	130
VN	216	233	246	246
WH	160	168	188	179
Total	4166	4108	4440	4358
Média	4137		4399	

Tabela 7.11: Número de regras obtidas pelo algoritmo de classificação quando a procura foi realizada à custa do OEP usando os conjuntos de exemplos de treino da Tabela 7.5 como parâmetros de entrada. Os resultados apresentados correspondem aos dois *runs* realizados para cada um dos conjuntos (A e B), identificados na tabela por OPE A.1, OPE A.2, OPE B.1, OPE B.2.

	Conjunto A		Conjunto B	
Etiqueta	*Run* OPE A.1	*Run* OPE A.2	*Run* OPE B.1	*Run* OPE B.2
ADJ	188	199	235	255
ADV	306	324	360	338
CNJ	220	218	259	247
DET	218	216	248	230
EX	11	10	10	9
FW	6	6	13	13
MOD	13	12	18	13
N	438	411	500	457
NP	150	158	183	194
NUM	44	40	42	50
PRO	114	114	110	120
P	161	193	182	215
TO	7	7	8	7
UH	3	3	5	5
V	249	271	302	305
VBZ	96	85	100	103
VD	142	153	181	167
VG	73	78	94	81
VN	146	144	162	170
WH	110	94	136	122
Total	2695	2736	3148	3101
Média	2715.5		3124.5	

Tabela 7.12: Número de regras obtidas pelo algoritmo de classificação quando a procura foi realizada à custa do OEP, usando os conjuntos de exemplos da Tabela 7.6 como parâmetros de entrada. Os resultados apresentados correspondem aos dois *runs* realizados para cada um dos conjuntos (C e D), identificados na tabela por OPE C.1, OPE C.2, OPE D.1, OPE D.2.

	Conjunto C		Conjunto D	
Etiqueta	*Run* OPE C.1	*Run* OPE C.2	*Run* OPE D.1	*Run* OPE D.2
ADJ	250	256	290	306
ADV	363	392	429	426
CNJ	268	285	318	317
DET	267	224	287	315
EX	17	16	20	10
FW	13	14	21	11
MOD	12	9	14	19
N	505	509	566	625
NP	223	203	221	221
NUM	48	55	51	58
PRO	126	112	112	138
P	214	171	175	203
TO	7	9	8	11
UH	5	5	6	6
V	343	329	367	336
VBZ	112	109	112	96
VD	178	181	196	190
VG	105	78	100	103
VN	197	176	222	204
WH	132	136	149	134
Total	3385	3269	3664	3729
Média	3327		3696,5	

7.3.3 Análise dos resultados

Quando estamos perante um problema de classificação, a avaliação dos conjuntos de regras obtidos, é geralmente efectuada usando as métricas típicas da cobertura e precisão. Contudo, como sabemos, as regras aqui obtidas não têm, no contexto do trabalho desenvolvido nesta tese, como objectivo resolver um problema de classificação. O seu propósito é servirem como heurística do algoritmo de procura, responsável por resolver o problema de optimização combinatória, que caracteriza o problema da marcação das PdD. Daqui decorre que a qualidade de cada um dos conjuntos depende dos resultados a que conduzem na resolução do problema da marcação das PdD.

A avaliação do sucesso da marcação das PdD é, tipicamente, realizada à custa do cálculo da percentagem de palavras marcadas correctamente. Porém, neste trabalho, não pretendemos apenas investigar a viabilidade de aplicação das regras de desambiguação, na construção de um marcador das PdD que se mostre competitivo, no que concerne a exactidão da marcação. A utilização de regras de desambiguação, em substituição da tabela de treino que é utilizada nas abordagens evolucionárias descritas no estado da arte, pretende reduzir e tornar mais legível a informação necessária para realizar a referida marcação. Esta redução, conseguida à custa da generalização da informação, tipicamente armazenada nas tabelas de treino, tem como objectivo contribuir para uma redução dos custos computacionais inerentes à utilização das tabelas de treino, e sobretudo substanciar-se numa melhoria de desempenho do marcador quando aplicado a *corpora* diverso do *corpus* usado no treino.

A análise que aqui fazemos, dos resultados obtidos pelo algoritmo de classificação, pode ser dividida em duas partes distintas, de acordo com os aspectos analisados:

- Redução no volume de informação normalmente utilizada pelos métodos probabilísticos para resolver o problema.
- Avaliação da qualidade das heurísticas resultantes.

Redução no volume de informação a usar pelo marcador

Iremos começar por tentar perceber o nível de redução que foi conseguido no volume de informação a fornecer ao marcador. Neste ponto é oportuno recordar que as tabelas de treino mencionadas em cima armazenam, para cada etiqueta, os diferentes contextos em que estas ocorrem no *corpus* de treino. Os marcadores definidos com base neste tipo de tabelas, definem à priori tabelas que contemplam diferentes formas de contexto. Como sabemos, a forma do contexto é definida pelo número de palavras observadas à esquerda, y, e à direita, z, de uma palavra, e é tradicionalmente denotada por $y-z$. Nos

Tabela 7.13: Número de regras obtidas pelo algoritmo de classificação quando a procura foi realizada à custa do OEP, usando os conjuntos de exemplos da Tabela 7.7 como parâmetros de entrada. Os resultados apresentados correspondem aos dois *runs* realizados para cada um dos conjuntos (E e F), identificados na tabela por OEP E.1, OEP E.2, OEP F.1, OEP F.2.

	Conjunto E		Conjunto F	
Etiqueta	*Run* OEP E.1	*Run* OEP E.2	*Run* OEP F.1	*Run* OEP F.2
ADJ	273	301	332	326
ADV	461	476	518	521
CNJ	308	337	361	300
DET	324	345	352	344
EX	14	12	27	15
FW	15	19	22	28
MOD	17	17	17	21
N	630	638	621	612
NP	245	263	254	250
NUM	55	66	75	63
PRO	115	145	153	156
P	246	230	255	242
TO	9	10	9	8
UH	11	10	11	11
V	394	387	406	439
VBZ	112	119	139	124
VD	208	204	192	244
VG	139	128	137	104
VN	225	233	239	269
WH	157	168	189	191
Total	3958	4108	4309	4268
Média	4033		4288.5	

trabalhos apresentados em [Araujo, 2002*b*] e [Alba et al., 2006], são exibidos resultados para diferentes formas de contexto, nomeadamente $1-0$, $2-0$, $1-1$ e $2-1$.

Uma das limitações que se infere dos trabalhos mencionados, é a imposta ao marcador pelo tipo de tabela que é fornecida como parâmetro de entrada, a qual determina a forma do contexto que é usada. Esta limitação reflecte-se em diferentes resultados de exactidão. Na implementação menos limitadora, são consultadas diferentes tabelas de contexto. Na avaliação de uma situação em particular, a tabela correspondente ao contexto mais abrangente admitido é consultada. Se a tabela não contém nenhuma entrada que permita efectuar a avaliação, o algoritmo investiga sucessivamente as entradas de tabelas de treino de contextos menos abrangentes.

Na abordagem aqui apresentada, a forma do contexto varia consoante a regra que está a ser aplicada. O algoritmo de classificação tem a responsabilidade de descobrir as regras que melhor classificam os exemplos de treino, e por conseguinte, inerentemente, descobrir as formas de contexto mais adequadas para determinadas situações. O marcador das PdD pode assim ser concebido sem ter que ponderar as formas de contexto mais adequadas.

Embora o processo para obtenção das tabelas de treino seja simples, quando comparado com a aplicação de um algoritmo de classificação como o proposto neste trabalho, o resultado são tabelas com milhares de entradas. Estas irão condicionar posteriormente o algoritmo de procura responsável pela marcação, não só no que concerne a óbvios custos computacionais, mas também à exactidão exibida.

Não querendo menosprezar os custos computacionais inerentes à aplicação do algoritmo de classificação definido neste trabalho, salienta-se o facto de este processo apenas ter que ser realizado uma vez. Determinadas as regras, estas podem alimentar o marcador evolucionário sempre que necessária a sua execução, traduzindo-se num volume de informação significativamente mais compacto do que o das tabelas de treino.

Na Tabela 7.14 podemos observar o número total de exemplos positivos diferentes considerados em cada um dos conjuntos usados pelo algoritmo de classificação. Este número é a soma de todos os exemplos de todas as etiquetas consideradas. Concernem-se, no entanto, apenas aos exemplos definidos por palavras ambíguas, e foram retirados, como já dissemos, de conjuntos com cardinalidade que varia entre os $3E4$ e os $8E4$ exemplos. Não obstante, estes valores permitem-nos ter uma ideia da dimensão das tabelas de treino das abordagens mencionadas, dado que o tipo de informação armazenada é semelhante. Contudo, a cardinalidade do conjunto de treino usado, nesses casos, é regra geral superior às $1.5E5$ palavras.

Como podemos ver na Tabela 7.14, as regras obtidas por ambas as implementações, permitem reduzir de forma significativa (na ordem dos 90%) o número de exemplos positivos considerados. Embora as publicações a que tivemos acesso, que descrevem as

Tabela 7.14: Número médio de regras obtidas pelo algoritmo de classificação, nos dois *runs* realizados para cada um dos conjuntos de A a F, quando executado com o algoritmo genético (coluna AG) e com o OEP (coluna OEP).

		Número médio de regras			
Conjunto	Exemplos positivos	AG	Redução	OEP	Redução
A	25859	2719	89.49%	2715.5	89.49%
B	33513	3081	90.81%	3124.5	90.68%
C	41080	3358.5	91.82%	3327.0	91.90%
D	48612	3735.5	92.32%	3696.5	92.39%
E	55823	4137	92.59%	4033.0	92.78%
F	63515	4399	93.07%	4288.5	93.25%

abordagens evolucionárias ao problema das PdD, não indiquem claramente o número de entradas das tabelas de treino usadas, a dimensão destas é referida explicitamente como ponto sensível do marcador, no que diz respeito ao tempo de execução do algoritmo.

De salientar que a informação contida nas regras de desambiguação não diz apenas respeito à informação de contexto. Uma das vantagens da utilização de regras de classificação é, precisamente, a possibilidade de facilmente se poder incluir diferentes características, para além da típica informação de contexto. Podemos assim concluir que a abordagem adoptada permitiu reduzir substancialmente a informação a fornecer ao marcador, e além disso permitiu a inclusão de outro tipo de informação, para além da tipicamente usada.

Avaliação da qualidade das heurísticas

De forma a avaliar a qualidade da heurística representada por cada um dos diferentes conjuntos de regras, foi necessário determinar a que resultados de marcação cada um deles conduziu. Para isso foram realizadas experiências onde cada um dos conjuntos de regras foi fornecido como parâmetro de entrada dos marcadores desenvolvidos. Quisemos com estas experiências investigar os seguintes aspectos:

- Perceber se a abordagem proposta é admissível.
- Averiguar se a informação obtida pelo algoritmo de classificação conduz os marcadores a resultados admissíveis.
- Comparar os resultados obtidos pelas duas implementações do algoritmo de classificação.

Antes de passarmos à análise desses resultados, podemos desde já observar, que o número de regras obtido pelo marcador desenvolvido com base num OEP é, de uma forma geral, semelhante ao número de regras obtido pelo algoritmo genético (ver Tabela 7.14). Contudo, o tempo de execução do algoritmo de classificação definido à custa do algoritmo genético, foi significativamente superior ao do algoritmo baseado em partículas de enxame.

Como foi observado no capítulo anterior, decidiu-se investigar o resultado da implementação do marcador das PdD à custa de dois algoritmos de procura distintos: um baseado num algoritmo genético (daqui em diante denominado marcador genético), e outro definido à custa de um optimizador baseado em enxames de partículas (daqui em diante denominado marcador OEP). De forma a perceber a qualidade da marcação das PdD, proporcionada pelas diferentes heurísticas determinadas por cada um dos conjuntos de regras, submeteu-se cada um deles aos diferentes marcadores implementados.

Relativamente ao marcados AG foram testadas as duas representações definidas. A representação identificada nas tabelas como representação do tipo **A**, corresponde àquela em que se evolui apenas as etiquetas respeitantes às palavras ambíguas. Por representação do tipo **B** designa-se a representação que considera tantos genes quantas as palavras da frase de entrada.

As experiências foram realizadas usando como conjunto de teste um subconjunto do *corpus* de *Brown*, distinto do conjunto utilizado para descobrir as regras de desambiguação. Todas as experiências foram executadas sobre o mesmo conjunto de frases, tendo sido realizadas utilizando um Mac Pro com 2 processadores (2.4 GHz), cada um com 4 *cores*. Todos os tempos de execução apresentados nas tabelas que se seguem encontram-se em segundos.

Nas Tabelas 7.15 e 7.16 encontram-se os resultados obtidos pelo marcador genético quando executado com 50 indivíduos, durante 10 gerações, num conjunto de 10 *runs* para cada conjunto de regras descoberto pelo algoritmo de classificação baseado num algoritmo genético. Os resultados apresentados permitem identificar o conjunto de regras, designado na tabela por *Run* AG F.1, como o que permite alcançar uma melhor exactidão na marcação das PdD. Recorde-se que *Run* AG F.1 identifica o conjunto de regras descoberto pelo algoritmo de classificação, baseado num algoritmo genético, a partir dos exemplos de treino retirados dos $8E4$ primeiros exemplos.

Os resultados obtidos pelo marcador OEP, quando executado com 10 partículas durante 50 gerações, para o mesmo *corpus* de teste encontram-se na Tabela 7.17. O tamanho do enxame e o número de gerações foram escolhidos de forma a resultarem no mesmo número de avaliações realizadas pelos marcadores genéticos. Foram, no entanto, adaptados às características do algoritmo que tipicamente é executado com enxames com um número reduzido de partículas.

Tabela 7.15: Resultados obtidos pelo marcador genético, com a representação do tipo **A**, quando executado com 50 indivíduos, durante 10 gerações, num conjunto de 10 *runs* para cada conjunto de regras descoberto pelo algoritmo de classificação definido à custa de um algoritmo genético. O tempo apresentado corresponde à execução dos 10 *runs* e encontra-se em segundos.

Conjunto	Nº de regras	Média	Melhor	Desvio Padrão	Tempo
Run AG A.1	2740	0.9652158	0.9655172	$1.854E-4$	2250.33
Run AG A.2	2698	0.9643826	0.9645865	$1.466E-4$	2220.37
Run AG B.1	3059	0.9650474	0.9654286	$1.450E-4$	2190.29
Run AG B.2	3103	0.9644446	0.9646751	$1.602E-4$	2320.31
Run AG C.1	3355	0.9665943	0.9669356	$2.250E-4$	2260.31
Run AG C.2	3362	0.9652336	0.9654286	$1.607E-4$	2150.29
Run AG D.1	3742	0.9660846	0.9663594	$1.819E-4$	2357.98
Run AG D.2	3362	0.9658231	0.96609341	$1.393E-4$	2340.33
Run AG E.1	4166	0.9669932	0.9672458	$1.1736E-4$	2370.32
Run AG E.2	4108	0.9666696	0.9668026	$1.3859E-4$	2380.32
Run AG F.1	4440	**0.9672813**	**0.9675117**	$1.843E-4$	2430.32
Run AG F.2	4358	0.9668336	0.9669799	$1.619E-4$	2390.33

Tabela 7.16: Resultados obtidos pelo marcador genético, com a representação do tipo **B**, quando executado com 50 indivíduos, durante 10 gerações, num conjunto de 10 *runs* para cada conjunto de regras obtido pelo algoritmo de classificação definido à custa de um algoritmo genético. O tempo apresentado, corresponde à execução dos 10 *runs*, e encontra-se em segundos.

Conjunto	Nº de regras	Média	Melhor	Desvio Padrão	Tempo
Run AG A.1	2740	0.9652646	0.9655616	$2.121E-4$	2240.20
Run AG A.2	2698	0.9644491	0.9648081	$2.146E-4$	2210.20
Run AG B.1	3059	0.9651848	0.9654286	$1.726E-4$	2200.28
Run AG B.2	3103	0.9644048	0.9646751	$1.5275E-4$	2320.29
Run AG C.1	3355	0.9664657	0.9667583	$2.2812E-4$	2270.28
Run AG C.2	3362	0.9653001	0.9655616	$1.9650E-4$	2300.29
Run AG D.1	3742	0.9661200	0.9663594	$1.8828E-4$	2340.34
Run AG D.2	3362	0.9657832	0.9660048	$1.5067E-4$	2260.39
Run AG E.1	4166	0.9669223	0.9670242	$1.0867E-4$	2460.37
Run AG E.2	4108	0.9666076	0.96689126	$1.6345E-4$	2370.35
Run AG F.1	4440	**0.9672547**	**0.9675561**	$2.0236E-4$	2430.37
Run AG F.2	4358	0.9669134	0.9671128	$1.8245E-4$	2360.35

Tabela 7.17: Resultados obtidos pelo marcador OEP quando executado com 10 partículas, durante 50 gerações, num conjunto de 10 *runs* para cada conjunto de regras obtido pelo algoritmo de classificação definido à custa de um algoritmo genético. O tempo apresentado, corresponde à execução dos 10 *runs*, e encontra-se em segundos.

Conjunto	Nº de regras	Média	Melhor	Desvio Padrão	Tempo
Run AG A.1	2740	0.9655128	0.9659605	$2.3120E-4$	2080.33
Run AG A.2	2698	0.9647239	0.9652956	$3.7227E-4$	2080.34
Run AG B.1	3059	0.9651449	0.9654286	$2.6196E-4$	2070.34
Run AG B.2	3103	0.9644358	0.9649854	$2.4651E-4$	2190.29
Run AG C.1	3355	0.9664569	0.9667583	$2.6329E-4$	2150.32
Run AG C.2	3362	0.9654596	0.9658718	$2.6350E-4$	2212.34
Run AG D.1	3742	0.9664258	0.9667139	$1.7882E-4$	2280.39
Run AG D.2	3362	0.9661023	0.9664480	$3.2624E-4$	2240.59
Run AG E.1	4166	0.9669666	0.9672458	$1.7858E-4$	2414.54
Run AG E.2	4108	0.9666209	0.9669356	$2.4496E-4$	2320.58
Run AG F.1	4440	**0.9672369**	**0.9677334**	$2.3248E-4$	2463.65
Run AG F.2	4358	0.9671128	0.9677334	$2.7878E-4$	2340.35

À semelhança do conjunto de experiências anteriores foram realizados 10 *runs* para cada conjunto de regras de desambiguação. Também com o marcador OEP, o conjunto de regras que conduziu aos melhores resultados é o conjunto designado por *Run*AG F.1

A mesma sequência de experiências foi realizada para as regras obtidas pelo algoritmo de classificação baseado em optimização por enxame de partículas. Os resultados obtidos encontram-se nas Tabelas 7.18, 7.19 e 7.20. Os valores de exactidão conseguidos pelos três marcadores, permitem concluir que a melhor heurística é a consubstanciada nas regras do conjunto identificado por *Run* OEP C.1.

O conjunto de experiências cujos resultados acabámos de apresentar, permitiu-nos identificar o conjunto de regras que se traduziu na melhor heurística para guiar o marcador das PdD. No entanto, possibilitou-nos sobretudo confirmar que a abordagem proposta é possível, e que a informação assim recolhida conduz o marcador a resultados no mínimo admissíveis. De facto, como iremos ver na próxima secção, os resultados apresentados são, não só admissíveis como, competitivos com os melhores resultados publicados.

A Tabela 7.21 faz um apanhado dos melhores resultados obtidos por cada um dos marcadores. Como podemos observar, a melhor exactidão conseguida resultou da utilização do conjunto de regras designado por *Run* AG F.1. De facto, de uma forma geral,

Tabela 7.18: Resultados obtidos pelo marcador genético, com representação do tipo **A**, quando executado com 50 indivíduos, durante 10 gerações, num conjunto de 10 *runs* para cada conjunto de regras obtido pelo algoritmo de classificação baseado em OEP. O tempo apresentado, corresponde à execução dos 10 *runs*, e encontra-se em segundos

Conjunto	Nº de regras	Média	Melhor	Desvio Padrão	Tempo
Run OEP A.1	2695	0.9637842	0.9641432	$2.9948E-4$	2160.62
Run OEP A.2	2736	0.9630529	0.9633454	$2.2812E-4$	2060.33
Run OEP B.1	3148	0.9630839	0.9633898	$1.9539E-4$	2050.29
Run OEP B.2	3101	0.9646397	0.9648967	$1.6292E-4$	2000.27
Run OEP C.1	3385	**0.9665987**	**0.9668026**	$1.9399E-4$	2110.28
Run OEP C.2	3269	0.9648037	0.9650740	$1.7032E-4$	1940.26
Run OEP D.1	3664	09653134	0.9656502	$1.8828E-4$	2060.28
Run OEP D.2	3729	0.9655882	0.9659161	$2.5607E-4$	2040.28
Run OEP E.1	3958	0.9646441	0.9648524	$1.8926E-4$	2020.27
Run OEP E.2	4108	0.9654197	0.9656502	$2.1993E-4$	2060.29
Run OEP F.1	4309	0.9656280	0.9659605	$2.5696E-4$	2070.27
Run OEP F.2	4268	0.9656945	0.9661378	$1.9488E-4$	2000.26

os resultados conseguidos pelos três marcadores, mostraram uma exactidão superior quando usaram como heurística as regras descobertas pelo algoritmo de classificação definido à custa do algoritmo genético.

Os valores de exactidão apresentados permitem-nos, também, observar uma tendência dos resultados obtidos pelo algoritmo de procura de regras, definido à custa de um algoritmo genético. Podemos constatar que, tendencialmente, os conjuntos de regras obtidos a partir de conjuntos de exemplos de treino de cardinalidade superior, conduzem a valores superiores de exactidão. Sendo, por isso, admissível esperar que os resultados obtidos pelos marcadores possam ser melhorados com a utilização de conjuntos de regras obtidos a partir de um conjunto mais alargado de exemplos.

Por outro lado, a marcação sugerida pelos marcadores, quando estes são alimentados com as regras obtidas pelo algoritmo de procura baseado em OEP, parecem mostrar que não será expectável uma melhoria dos resultados, usando conjuntos de exemplos mais vastos. Isto porque os melhores valores de exactidão são conseguidos com regras obtidas a partir de conjuntos de exemplos de cardinalidade intermédia.

Na Tabela 7.22 podemos observar os conjuntos de regras, descobertos por cada uma das implementações do algoritmo de classificação, que resultaram nas melhores heurísticas. Neste caso a medida usada para avaliar a qualidade da heurística traduziu-se no valor

Tabela 7.19: Resultados obtidos pelo marcador genético, com a representação do tipo **B**, quando executado com 50 indivíduos, durante 10 gerações, num conjunto de 10 *runs* para cada conjunto de regras obtido pelo algoritmo de classificação baseado em OEP. O tempo apresentado, corresponde à execução dos 10 *runs*, e encontra-se em segundos

Conjunto	Nº de regras	Média	Melhor	Desvio Padrão	Tempo
Run OEP A.1	2695	0.9638020	0.9640546	$2.0689E-4$	1970.00
Run OEP A.2	2736	0.9629643	0.9633454	$2.2328E-4$	2020.29
Run OEP B.1	3148	0.9631283	0.9634341	$1.6511E-4$	2040.33
Run OEP B.2	3101	0.9646707	0.9651183	$2.5882E-4$	1970.31
Run OEP C.1	3385	**0.9666519**	**0.9670685**	$1.9172E-4$	2060.31
Run OEP C.2	3269	0.9648701	0.9652070	$1.7754E-4$	1920.30
Run OEP D.1	3664	0.9653710	0.9656945	$2.2605E-4$	2040.29
Run OEP D.2	3729	0.9656635	0.9659605	$1.6056E-4$	2050.31
Run OEP E.1	3958	0.9646928	0.9650297	$2.6029E-4$	2010.34
Run OEP E.2	4108	0.9653754	0.9659161	$2.4078E-4$	2070.30
Run OEP F.1	4309	0.9656990	0.9660934	$2.4849E-4$	2050.30
Run OEP F.2	4268	0.9656857	0.9660048	$1.7704E-4$	2000.28

de exactidão a que conduziu.

Quisemos ainda estudar a influência dos diferentes conjuntos de regras, no tempo de execução dos marcadores. Para isso, compararam-se, para cada um, os tempos conseguidos com as regras obtidas pelas duas implementações do algoritmo de classificação, para cada um dos conjuntos de exemplos de treino considerados (de A.1 a F.2).

As Tabelas 7.23, 7.24 e 7.25, apresentam os tempos conseguidos pelos marcadores genéticos **A** e **B**, e pelo marcador OEP, respectivamente. A última coluna de cada tabela permite observar a percentagem de redução de tempo conseguida pelas regras descobertas pelo algoritmo baseado em OEP.

Como podemos ver, cada um dos marcadores consegue realizar a marcação do texto fornecido, em menos tempo, usando como heurística as regras descobertas pelo algoritmo de classificação baseado em OEP. Embora a exactidão conseguida pelos três marcadores seja inferior quando são usadas estas regras, como veremos na próxima secção os resultados exibidos são ainda assim competitivos, quando comparados com os obtidos pelas outras abordagens evolucionárias.

Tabela 7.20: Resultados obtidos pelo marcador OEP, quando executado com 10 partículas, durante 50 gerações, num conjunto de 10 *runs* para cada conjunto de regras obtido pelo algoritmo de classificação baseado em OEP. O tempo apresentado, corresponde à execução dos 10 *runs*, e encontra-se em segundos

Conjunto	Nº de Regras	Média	Melhor	Desvio Padrão	Tempo
Run OEP A.1	2695	0.9635227	0.9641876	$2.5759E-4$	1630.21
Run OEP A.2	2736	0.9629022	0.9633011	$2.8265E-4$	1700.23
Run OEP B.1	3148	0.9628668	0.9631682	$2.6163E-4$	1640.21
Run OEP B.2	3101	0.9649366	0.9651183	$1.5132E-4$	1730.22
Run OEP C.1	3385	**0.9669356**	**0.9673345**	$2.3172E-4$	1630.21
Run OEP C.2	3269	0.9650962	0.9654286	$2.3937E-4$	1650.22
Run OEP D.1	3664	0.9655749	0.9660048	$2.3643E-4$	1660.21
Run OEP D.2	3729	0.9661378	0.9664923	$2.4005E-4$	1590.21
Run OEP E.1	3958	0.9650740	0.9654286	$2.0365E-4$	1490.19
Run OEP E.2	4108	0.9654286	0.9658275	$2.4809E-4$	1710.22
Run OEP F.1	4309	0.9655394	0.9658718	$2.1435E-4$	1580.20
Run OEP F.2	4268	0.9656901	0.9660934	$2.0836E-4$	1680.24

Tabela 7.21: Melhores resultados, relativamente à exactidão da marcação, alcançados por cada um dos marcadores.

Marcador	Regras	Média	Melhor	Desvio Padrão	Tempo
Genetico A	*Run* AG F.1	**0.9672813**	**0.9675117**	$1.843E-4$	2430.32
Genetico B	*Run* AG F.1	**0.9672547**	**0.9675561**	$2.0236E-4$	2430.37
OEP	Run AG F.1	**0.9672369**	**0.9677334**	$2.3248E-4$	2463.65

Tabela 7.22: Melhores conjuntos de regras obtidos por cada um dos algoritmos de descoberta de regras implementado.

Regras	Marcador	Média	Melhor	Desvio Padrão	Tempo
Run AG F.1	OEP	0.9672369	0.9677334	$2.3248E-4$	2463.65
Run OEP C.1	OEP	0.9669356	0.9673345	$2.3172E-4$	1630.21

Tabela 7.23: Tempos obtidos pelo marcador genético, com representação do tipo **A**, usando como heurística os conjuntos de regras descobertas pelo algoritmo de classificação definido à custa de algoritmos genéticos (Tempo *Run* AG), e de OEP (Tempo *Run* OEP).

	Tempo *Run* AG (TAG)	Tempo *Run* OEP (TOEP)	$(1 - \frac{TOEP}{TAG}) \times 100$
A.1	2250.34	2160.62	3.99
A.2	2220.37	2060.33	7.21
B.1	2190.29	2050.29	6.39
B.2	2320.32	2000.27	13.79
C.1	2260.31	2110.28	6.64
C.2	2150.29	1940.26	9.77
D.1	2357.98	2060.28	12.63
D.2	2340.33	2040.28	12.82
E.1	2370.32	2020.27	14.77
E.2	2380.32	2060.29	13.44
F.1	2430.32	2070.27	14.81
F.2	2390.33	2000.26	16.32

Tabela 7.24: Tempos obtidos pelo marcador genético, com representação do tipo **B**, usando como heurística as regras descobertas pelas duas implementações do algoritmo de classificação: baseada em algoritmos genéticos (Tempo *Run* AG), e em optimização de partículas de enxame (Tempo *Run* OEP).

	Tempo *Run* AG (TAG)	Tempo *Run* OEP (TOEP)	$(1 - \frac{TOEP}{TAG}) \times 100$
A.1	2240.20	1970.00	12.06
A.2	2210.29	2020.29	8.59
B.1	2200.28	2040.33	7.27
B.2	2320.29	1970.31	15.08
C.1	2270.28	2060.31	9.25
C.2	2300.29	1920.30	16.52
D.1	2340.34	2040.29	12.82
D.2	2260.39	2050.31	9.29
E.1	2460.37	2010.34	18.29
E.2	2370.35	2070.30	12.66
F.1	2430.37	2050.30	15.64
F.2	2360.35	2000.28	15.25

Tabela 7.25: Tempos obtidos pelo marcador OEP usando como heurística os conjuntos de regras descobertas pelas duas implementações do algoritmo de classificação: baseada em algoritmos genéticos (Tempo *Run* AG), e em OEP (Tempo *Run* OEP).

	Tempo *Run* AG (TAG)	Tempo *Run* OEP (TOEP)	$(1 - \frac{TOEP}{TAG}) \times 100$
A.1	2250.34	1630.21	27.56
A.2	2220.37	1700.23	23.43
B.1	2190.29	1640.21	25.11
B.2	2320.32	1730.22	25.43
C.1	2260.31	1630.21	27.88
C.2	2150.29	1650.22	23.26
D.1	2357.98	1660.21	29.59
D.2	2340.33	1590.21	32.05
E.1	2370.32	1490.19	37.13
E.2	2380.32	1710.22	28.15
F.1	2430.32	1580.20	34.98
F.2	2390.33	1680.24	29.71

7.3.4 Exemplos de algumas das regras obtidas

Nesta secção apresentamos, apenas a título ilustrativo, algumas das regras descobertas pelo algoritmo de classificação. No entanto, uma amostra mais alargada pode ser encontrada nos anexos deste documento. As regras são descritas usando lógica de predicados. A definição dos predicados, símbolos funcionais e constantes individuais usados na representação das regras, encontra-se na Tabela 7.26.

Tabela 7.26: Definição dos predicados, símbolos funcionais e constantes individuais usados na representação das regras.

Predicado	Significado
Maiúscula(x)	x é maiúscula
Primeira(x)	x é a primeira palavra da frase
Termina(x,y)	x termina com y
Especiais(x)	x contem números ou '.' e números
Classe(x,y)	x é da classe gramatical y

Símbolo Funcional	Significado
etiqueta(x,y)	a etiqueta da x^{o} palavra na posição y

Constantes Individuais	Significado
esquerda	posição à esquerda
direita	posição à direita
ed	terminação *ed*
ing	terminação *ing*
ould	terminação *ould*
's	terminação *'s*
s	terminação *s*
1, 2 e 3	1, 2 e 3

Tabela 7.27: Exemplos de regras obtidas para a etiqueta **ADJ** (*Adjectivo*).

Exemplos de regras obtidas para a etiqueta **ADJ** (*Adjectivo*)
SE etiqueta(2, esquerda)=DET ∧ etiqueta(1, direita)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, ADJ) com $f_\beta = 0.8299$
SE etiqueta(1, esquerda)=DET ∧ etiqueta(1, direita)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, ADJ) com $f_\beta = 0.9078$
SE etiqueta(1, esquerda)=P ∧ etiqueta(1, direita)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, ADJ) com $f_\beta = 0.8867$

Tabela 7.28: Exemplos de regras obtidas para a etiqueta **ADV** (*Advérbio*).

Exemplos de regras obtidas para a etiqueta **ADV** (*Advérbio*)
SE etiqueta(1, direita)=VD ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, ADV) com $f_\beta = 0.7755$
SE etiqueta(1, esquerda)=N ∧ etiqueta(1, direita)=None ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, ADV) com $f_\beta = 6840$
SE etiqueta(1, esquerda)=MOD ∧ etiqueta(1, direita)=V ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, ADV) com $f_\beta = 0.6801$

Tabela 7.29: Exemplos de regras obtidas para a etiqueta **DET** (*Determinante*).

Exemplos de regras obtidas para a etiqueta **DET** (*Determinante*)
SE etiqueta(2, esquerda)=N ∧ etiqueta(1, direita)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, DET) com $f_\beta = 0.9304$
SE etiqueta(1, esquerda)=P ∧ etiqueta(2, direita)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, DET) com $f_\beta = 0.9348$
SE etiqueta(1, esquerda)=P ∧ etiqueta(1, direita)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, DET) com $f_\beta = 0.8905$

Tabela 7.30: Exemplos de regras obtidas para a etiqueta **N** (*Nome*).

Exemplos de regras obtidas para a etiqueta **N** (*Nome*)
SE etiqueta(1, esquerda)=DET ∧ etiqueta(1, direita)=P ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, N) com $f_\beta = 0.8890$
SE etiqueta(1, esquerda)=ADJ ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, N) com $f_\beta = 0.8908$
SE etiqueta(2, esquerda)=P ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, N) com $f_\beta = 0.8368$

Como podemos constatar a informação recolhida pelo algoritmo de classificação, sob a forma de regras de desambiguação, é facilmente traduzida num formato compreensível, permitindo assim a sua utilização noutros contextos.

7.3.5 Síntese dos resultados

Nesta secção pretendemos fazer uma síntese da análise dos resultados apresentados na secção anterior, os quais dizem respeito às regras de desambiguação obtidas pelo algoritmo de classificação descrito no Capítulo 5. Como tivemos oportunidade de ver, esta análise envolveu o estudo dos resultados exibidos pelos marcadores das PdD quando executados com a informação obtida. No entanto, as experiências apresentadas não permitiram estudar todos os aspectos que pretendíamos, nomeadamente o grau de generalização conseguido. O qual esperamos se venha a traduzir numa maior independência do marcador, relativamente ao domínio do *corpus* onde foi efectuado o treino. Este aspecto será desenvolvido na próxima secção, quando apresentarmos o trabalho experimental realizado para estudar o comportamentos dos marcadores das PdD desenvolvidos.

Durante a análise dos resultados, obtidos pelo algoritmo de classificação, tivemos a oportunidade de apresentar várias conclusões que gostaríamos aqui de sintetizar:

- A viabilidade da abordagem proposta neste trabalho de investigação foi confirmada, tendo sido mostrado que é possível descobrir regras a partir da informação disponível nos *corpora* anotados, e que estas conseguem guiar a procura de uma solução, no espaço de estados do problema da marcação das PdD.

- A informação recolhida sob a forma de regras de desambiguação permitiu reduzir, substancialmente, o volume de informação usada pelo marcador das PdD.

- A informação pode ser facilmente traduzida num formato de fácil compreensão, como o aqui apresentado, fazendo uso de lógica de predicados.

- Os valores de exactidão exibidos pelos marcadores são no mínimo admissíveis, situando-se numa gama de valores perto, e, em alguns casos, superior a outros publicados por abordagens semelhantes, como teremos oportunidade de observar com maior detalhe na próxima secção.

- As regras obtidas pelo algoritmo genético traduzem-se na melhor heurística para guiar os marcadores desenvolvidos.

- É possível identificar uma tendência nos conjuntos de regras obtidos pelo algoritmo de classificação definido à custa de um algoritmo genético. Podemos

Tabela 7.31: Exemplos de regras obtidas para a etiqueta **NP** (*Nome Próprio*).

Exemplos de regras obtidas para a etiqueta **NP** (*Nome Próprio*)
SE etiqueta(1, esquerda)=NP ∧ ∧ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, NP) com $f_\beta = 0.9304$
SE etiqueta(1, direita)=NP ∧ ∧ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, NP) com $f_\beta = 0.9121$
SE etiqueta(1, direita)=NP ∧ ∧ Maiúscula(x) ∧ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, NP) com $f_\beta = 0.8835$

observar que conjuntos de regras obtidos a partir de conjuntos de exemplos de cardinalidade crescente, conduzem tendencialmente a valores crescentes de exactidão. Esta observação leva-nos a admitir que a qualidade da marcação pode ainda ser melhorada.

- Os resultados obtidos pelo algoritmo de procura baseado em OEP mostram que, a partir de um determinado número de exemplos de treino, não houve melhoria na qualidade dos conjuntos de regras obtidos. Isto porque, não se observam melhorias significativas na marcação efectuada à custa de conjuntos de regras obtidos a partir de conjuntos de exemplos de treino de cardinalidade superior.

- Embora conduzindo a valores de exactidão inferiores, as regras obtidas pelo algoritmo de classificação baseado em OEP, resultam numa marcação significativamente mais rápida do que a proporcionada pelas regras conseguidas à custa do algoritmo genético.

7.4 Marcação das PdD

Nesta secção iremos apresentar o trabalho experimental desenvolvido na afinação dos marcadores das PdD implementados, assim como os resultados conseguidos na aplicação dos mesmos ao *corpus* do WSJ, e finalmente situar os valores de exactidão exibidos entre os publicados em trabalhos semelhantes.

Todas as experiências realizadas para a afinação dos marcadores foram executadas recebendo como entrada o mesmo conjunto de frases. Estas frases foram retiradas da parte do *corpus* de *Brown* que não foi usada para a descoberta de regras de desambiguação. O conjunto usado é composto por 22562 palavras. O estudo apresentado na secção anterior resultou da utilização do mesmo conjunto de frases.

Na secção anterior apresentámos um primeiro estudo do comportamento dos marcadores com a utilização de diferentes conjuntos de regras de desambiguação. Este estudo permitiu identificar o conjunto de regras que se traduziu na melhor heurística para o algoritmo de procura, proporcionando os melhores resultados ao nível da exactidão, e aquele que conseguiu o melhor desempenho relativamente ao tempo de execução do marcador.

Destacaram-se dois conjuntos de regras, um obtido pela implementação do algoritmo de classificação baseada num algoritmo genético (*Run* AG F.1), e o outro obtido pela implementação baseada em OEP (*Run* OEP C.1). O primeiro conduziu os marcadores aos melhores resultados de exactidão, o segundo permitiu obter as marcações nos menores tempos de execução. Decidimos, por isso realizar o estudo de afinação dos parâmetros dos marcadores para cada um dos conjuntos de regras referidos.

Tabela 7.32: Resultados de 20 *runs* do marcador OEP, quando executado com 10 e 20 partículas, durante 50 e 100 gerações, com as regras *Run* OEP C.1 e *Run* AG F.1, no *corpus* de teste definido a partir do corpus de *Brown*.

Regras	Partículas	Gerações	Média	Melhor	Desvio Padrão
Run OEP C.1	10	50	0.9668092	0.9672901	$2.2244E-4$
		100	0.9667339	0.9672458	$2.8159E-4$
	20	**50**	**0.9669533**	**0.9675117**	$\mathbf{2.4336E-4}$
		100	0.9669267	0.9673345	$2.4429E-4$
Run AG F.1	10	50	0.9672658	0.9679550	$2.6534E-4$
		100	0.9673123	0.9676004	$1.9373E-4$
	20	**50**	**0.9674896**	**0.9678220**	$1.9158E-4$
		100	0.9673921	0.9678663	$2.1479E-4$

Uma das conclusões que foi possível retirar do estudo prévio, apresentado na secção anterior, foi o de não existirem diferenças significativas entre os marcadores genéticos, implementados com as representações do tipo **A** e do tipo **B**. Por este motivo, nesta secção apresentaremos apenas o estudo experimental realizado com a utilização do marcador genético com a representação do tipo **B**, dado que é semelhante à representação das abordagens evolucionárias apresentadas no estado da arte.

Deste primeiro estudo foi, também, possível identificar o marcador OEP como sendo o marcador que, de uma forma geral, conseguiu obter os melhores resultados. Contudo, os resultados conseguidos pelo marcador genético, como iremos ter oportunidade de ver, revelam-se competitivos com os publicados em trabalhos que apresentam abordagens semelhantes.

Ponderando os aspectos apresentados, optámos por realizar o estudo experimental, para afinação de parâmetros, para o marcador genético do tipo **B**, daqui em diante simplesmente identificado por marcador genético, e para o marcador OEP.

Para o marcador OEP foram testadas as combinações de 10 e 20 partículas com um número de gerações igual a 50 e 100. Para cada uma das combinações de parâmetros foram realizados 20 *runs* independentes, para os dois conjuntos de regras considerados. A Tabela 7.32 apresenta os valores da média, desvio padrão e melhor exactidão conseguida no total de cada um dos 20 *runs*.

Para ambos os conjuntos de regras a combinação que exibiu a melhor exactidão consistiu na execução do marcador com um enxame de 20 partículas durante 50 gerações.

Tabela 7.33: Resultados de 20 *runs* do marcador genético quando executado com 50 e 100 indivíduos, durante 10 e 20 gerações, com as regras Run OEP C.1 e Run AG F.1, no *corpus* de teste definido a partir do *corpus* de *Brown*.

Regras	Indivíduos	Gerações	Média	Melhor	Desvio Padrão
Run OEP C.1	50	10	0.9666896	0.9671572	$2.1255E-4$
		20	0.9667538	0.9669799	$1.6006E-4$
	100	10	0.9666807	0.9671128	$1.8125E-4$
		20	**0.9668124**	**0.9672015**	$1.6422E-4$
Run AG F.1	50	10	0.9672170	0.9675561	$1.9200E-4$
		20	0.9672968	0.9674231	$1.1707E-4$
	100	10	0.9672591	0.9675561	$1.4097E-4$
		20	**0.9672835**	**0.9675117**	$1.0978E-4$

O estudo da influência dos parâmetros no marcador genético foi realizado para combinações de 50 e 100 indivíduos, para um número de 10 e 20 gerações. O objectivo foi testar o marcador genético para valores que resultassem num esforço computacional, equivalente ao exigido nas experiências realizadas com o marcador OEP. Aqui o esforço computacional considerado diz respeito ao número de avaliações, determinadas pela combinação de parâmetros usada.

Nas experiências definidas para o marcador genético, consideraram-se populações superiores aos enxames definidos para o marcador OEP, no entanto assumiu-se um número inferior de gerações. Isto porque, tipicamente, os algoritmo genéticos necessitam de evoluir um número superior de soluções candidatas do que os optimizadores por partículas de enxame.

Os resultados obtidos pelo marcador genético, para as experiências descritas em cima, podem ser observados na Tabela 7.33. A análise dos resultados permite concluir que a melhor exactidão conseguida pelo marcador genético, resulta da execução do marcador com uma população constituída por 100 indivíduos durante 20 gerações.

Na Tabela 7.34 reunimos os melhores resultados conseguidos por ambos os marcadores para cada um dos conjuntos de regras. Como podemos observar, o marcador OEP consegue em ambos os casos os melhores resultados. Além disso, sublinha-se o facto dos resultados exibidos pelo marcador OEP implicarem metade das avaliações realizadas pelo marcador genético. De notar, porém, que o desvio padrão que caracteriza o conjunto de experiências do marcador genético ser, sistematicamente, inferior ao do marcador OEP, o que permite concluir uma maior constância dos resultados exibidos.

Tabela 7.34: Melhores resultados conseguidos por cada um dos marcadores.

Marcador	Regras	Avaliações	Média	Melhor	Desvio Padrão
Genético	*Run* OEP C.1	$2E3$	0.9668124	0.9672015	$1.6422E-4$
Genético	*Run* AG F.1	$2E3$	0.9672835	0.9675117	$1.0978E-4$
OEP	*Run* OEP C.1	$1E3$	**0.9669533**	**0.9675117**	$2.4336E-4$
OEP	*Run* AG F.1	$1E3$	**0.9674896**	**0.9678220**	$1.9158E-4$

Tabela 7.35: Resultados de 20 *runs* do marcador OEP, quando executado com 10 e 20 partículas, durante 50 e 100 gerações, com as regras Run OEP C.1 e Run AG F.1, no *corpus* de teste definido a partir do *corpus* do WSJ.

Regras	Partículas	Gerações	Média	Melhor	Desvio Padrão
Run OEP C.1	10	50	0.9645291	0.9651366	$2.9968E-4$
		100	0.9646204	0.9651763	$3.7249E-4$
	20	50	0.9644735	0.9651763	$3.6938E-4$
		100	**0.9646323**	**0.9652160**	$4.1408E-4$
Run AG F.1	10	50	0.9659566	0.9666058	$3.8719E-4$
		100	0.9659466	0.9664470	$2.8674E-4$
	20	50	**0.9659943**	**0.9668837**	$3.1277E-4$
		100	0.9659129	0.9664470	$3.5457E-4$

Contudo, de uma forma geral, o desvio padrão resultante das experiências efectuadas, é consideravelmente baixo.

Com o objectivo de investigar o grau de generalização das regras obtidas, dado que esse é um dos contributos expectáveis da utilização desta abordagem, testámos os marcadores num conjunto de teste obtido a partir do *corpus* do WSJ. Importa ressalvar que os marcadores foram executados usando como heurística as regras obtidas a partir do *corpus* de *Brown*. O objectivo é perceber se as regras descobertas são suficientemente genéricas para permitir a sua aplicação na marcação de frases de *corpora* diferentes na mesma língua. As Tabelas 7.35 e 7.36 mostram os resultados obtidos num conjunto de experiências, semelhantes às anteriores, mas neste caso com um conjunto de teste constituído por 1000 frases do *corpus* do WSJ, num total de 25184 palavras.

Como seria de esperar os resultados conseguidos pelos dois marcadores no *corpus* do WSJ, usando como heurística as regras de desambiguação aprendidas a partir do *corpus* de *Brown*, ficam aquém dos obtidos pelos marcadores na marcação do *corpus* de

Tabela 7.36: Resultados de 20 *runs* do marcador genético, quando executado com 50 e 100 indivíduos, durante 10 e 20 gerações, com as regras Run OEP C.1 e Run AG F.1, no *corpus* de teste definido a partir do *corpus* do WSJ.

Regras	Indivíduos	Gerações	Média	Melhor	Desvio Padrão
Run OEP C.1	50	10	0.9645092	0.9650175	$3.3237E-4$
		20	0.9647256	0.9652160	$2.5451E-4$
	100	10	**0.9647931**	**0.9654543**	$2.9526E-4$
		20	0.9644477	0.9650175	$2.8467E-4$
Run AG F.1	50	10	0.9659268	0.9668440	$3.2076E-4$
		20	0.9658374	0.9662484	$2.7997E-4$
	100	10	0.9658275	0.9662484	$2.7219E-4$
		20	**0.9660598**	**0.9663675**	$2.4541E-4$

Brown. No entanto, julgamos que nos permitem concluir que as regras obtidas são suficientemente genéricas, para poderem ser usadas em *corpora* diferentes. Esta ilação resulta da comparação dos resultados obtidos com os publicados por outras abordagens (Tabela 7.37) evolucionárias. Na realidade, verificamos que a exactidão conseguida se situa em valores não muito distantes do melhor resultado publicado, sendo necessário vincar que esta é a única abordagem não treinada no *corpus*.

Como já referimos, o pacote de *software* NLTK, disponibiliza um conjunto de frases do *corpus* do WSJ que corresponde a cerca de 10% do *corpus* do *Penn Treebank*. Este conjunto de frases contém cerca de 100676 palavras. Os resultados apresentados nas Tabelas 7.35 e 7.36 foram obtidos na marcação das primeiras 1000 frases, num total de 25184 palavras. A opção de usar esta amostra neste conjunto de experiências teve em vista apenas uma redução no tempo de execução das mesmas. No entanto, quisemos testar os marcadores para o conjunto de frases completo, usando para cada um o número de avaliações que conduziram aos melhores resultados nas experiências anteriores (ver Tabela 7.37).

Na Tabela 7.37 fazemos um apanhado dos valores de exactidão exibidos pelos marcadores, nos dois *corpora* em língua Inglesa utilizados, juntamente com os resultados publicados por trabalhos que seguem abordagens semelhantes, e que foram descritos no capítulo dedicado ao estado da arte. Os resultados apresentados na tabela apenas permitem afirmar que os valores obtidos são competitivos com os resultados publicados. Isto porque não podemos comparar directamente os valores obtidos com os publicados dado que não temos acesso ao conjunto de teste usado nas experiências realizadas nos trabalhos citados. No entanto, podemos concluir que para conjuntos de palavras

Tabela 7.37: Resultados obtidos pelos marcadores OEP e genético em dois *corpus* diferentes em língua Inglesa, assim como os resultados publicados em abordagens semelhantes (Araujo - [Araujo, 2002*b*]; GA - [Alba et al., 2006]; PGA - [Alba et al., 2006]; Wilson - [Wilson and Heywood, 2005]; Brill - [Brill, 1995]; Alba - [Alba et al., 2006]).

Corpus	Marcador	Treino	Teste	Melhor
Brown	OEP	80000	22562	**96.78**
	Genético	80000	22562	96.76
	Araujo	185000	2500	95.40
	GA	165276	17303	96.67
	PGA	165276	17303	96.75
WSJ	OEP	∅	100676	96.67
	Genético	∅	100676	96.66
	Wilson	600000	=Treino	89.80
	Brill	600000	150000	**97.20**
	Alba	554923	2544	96.63

de dimensão comparável (no caso das abordagens evolucionárias), retirados do mesmo *corpus*, os resultados conseguidos neste trabalho situam-se entre os melhores publicados. Os valores apresentados foram convertidos para percentagem, e arredondados à segunda casa decimal, de forma a serem mais facilmente comparados com os valores descritos nas publicações citadas.

Antes de terminarmos esta secção achamos que é importante fazer, também aqui, uma síntese das várias conclusões que foram sendo retiradas ao longo da análise dos resultados obtidos pelos marcadores das PdD:

- O trabalho experimental desenvolvido para o estudo da influência dos parâmetros de entrada dos marcadores, nomeadamente para o número de gerações e número de partículas/indivíduos, permitiu-nos concluir que os algoritmos conseguem obter resultados com recursos consideravelmente reduzidos. De facto não se observou uma clara tendência de melhoria dos resultados para valores crescentes de avaliações.

- O marcador definido à custa de OEP foi o que exibiu melhores resultados de exactidão. Estes resultados permitem-nos concluir que os algoritmos baseados em inteligência de enxame, também, em problemas de PLN mostram bons resultados.

- Gostaríamos de destacar que tanto quanto é do nosso conhecimento esta foi a

primeira tentativa de aplicação de um OEP para resolver o problema da marcação das PdD, sendo que, em geral, são poucas as abordagens baseadas em inteligência de enxames para resolver tarefas de PLN.

- Pudemos avaliar o grau de generalização da informação obtida pelo algoritmo de classificação adoptado. As experiências realizadas no *corpus* do WSJ, usando, como heurística, as regras obtidas a partir do *corpus* de *Brown*, permitiram-nos confirmar que as regras obtidas são, suficientemente, genéricas para poderem ser usadas em diferentes *corpora*.

7.5 Aplicação ao *Corpus Mac-Morpho*

A abordagem desenvolvida pode ser aplicada a qualquer língua, desde que estejam disponíveis recursos linguísticos que permitam fazer a descoberta das regras de desambiguação. Como exemplo, apresentamos nesta secção os resultados obtidos no *corpus Mac-Morpho*, descrito no início do capítulo.

Apesar de existir uma quantidade substancial de trabalhos dedicados à marcação das PdD para a língua Inglesa, o mesmo não acontece para o Português. Uma das principais razões está relacionada com o número, ainda relativamente escasso de recursos disponíveis. Embora tenham vindo a ser realizados avanços significativos para colmatar esta lacuna, o volume de trabalho desenvolvido nesta área para a língua Inglesa é significativamente superior. Não obstante, é possível encontrar na literatura trabalhos que aplicam algumas das técnicas conhecidas à marcação de frases em Português: Aprendizagem por regras de transformação (do Inglês *transformation based learning*) [Aires et al., 2000], modelos de Markov [Kepler and Finger, 2006], aprendizagem por transformação guiada por entropia (do Inglês *entropy guided transformation learning*) [Nogueira Dos Santos et al., 2008]. Porém, tanto quanto é do nosso conhecimento, esta é a primeira tentativa de aplicação de uma abordagem evolucionária ao Português.

A aplicação do marcador ao *corpus Mac-Morpho* tem como requisito prévio a existência de regras de desambiguação para o Português. Assim, numa primeira fase aplicámos o algoritmo de classificação para a descoberta das regras, utilizando para isso parte do *corpus Mac-Morpho*.

Como já tivemos oportunidade de referir, o *corpus Mac-Morpho* é um *corpus* em língua Portuguesa que contém $1.2E6$ de palavras marcadas manualmente. O conjunto de etiquetas adoptado contém 22 elementos. São ainda utilizadas 10 etiquetas suplementares que representam aspectos semânticos adicionais. Estas são utilizadas em conjunto com as 22 etiquetas principais, concatenando os acrónimos, com o símbolo '|' a fazer a separação. Nas experiências realizadas adoptaram-se as etiquetas simples,

sem a informação adicional

O processamento do *corpus* foi efectuado de forma semelhante ao realizado para o *corpus* de *Brown*. Porém, neste caso, utilizámos 60% das frases para construir os exemplos considerados na definição dos conjuntos de exemplos positivos e negativos, para cada uma das etiquetas. Também aqui se utilizou uma representação compactada dos exemplos. No entanto, de forma a reduzir o número de exemplos de cada um dos conjuntos, foram eliminados aqueles que surgiam apenas uma, duas ou três vezes, dependendo da cardinalidade do conjunto de exemplos positivos e negativos em causa. O total de exemplos considerados é igual a 107200 exemplos diferentes. Este número inclui os exemplos positivos e negativos de todas as etiquetas.

Os atributos mensuráveis considerados no antecedente das regras de desambiguação, reúnem características relacionadas com a informação de contexto. Tal como para a língua Inglesa, considerámos as etiquetas das três palavras à esquerda e à direita de uma determinada palavra:

- A etiqueta da terceira palavra à esquerda;
- A etiqueta da segunda palavra à esquerda;
- A etiqueta da primeira palavra à esquerda;
- A etiqueta da primeira palavra à direita;
- A etiqueta da segunda palavra à direita;
- A etiqueta da terceira palavra à direita.

Além da informação de contexto, foram ainda considerados os seguintes atributos:

- Se a palavra começa, ou não, por letra maiúscula:
- Se a palavra é a primeira palavra da frase.

As experiências foram conduzidas usando a versão do algoritmo de classificação definida à custa de OEP, e o marcador OEP. Esta decisão baseou-se apenas no tempo de execução de ambos os algoritmos. O algoritmo de classificação foi executado com 20 partículas, durante 200 gerações. Foram realizados vários *runs*, sendo que aquele que deu origem ao conjunto de regras que conduziu à melhor marcação das PdD, é constituído por 5988 regras.

Descobertas as regras de desambiguação, testámos o marcador OEP num conjunto de frases do *corpus Mac-Morpho*, diferentes das utilizadas para a descoberta de regras.

Tabela 7.38: Resultados obtidos pelo marcador OEP, após 20 *runs* com um enxame de 10 e 20 partículas, durante 50 e 100 gerações.

Particulas	Gerações	Média	Melhor	Desvio Padrão
10	50	0.9682009	0.9686947	$2.8566E-4$
	100	**0.9683406**	0,9686481	$2.8985E-4$
20	50	0.9682568	**0.9689742**	$3.2588E-4$
	100	0.9681962	0.9686481	$2.5521E-4$

Tabela 7.39: Melhor resultado conseguido com o marcador OEP para o *corpus* de *Mac-Morpho*, juntamente com os resultados publicados em [Nogueira Dos Santos et al., 2008], aqui identificados por *ETL-Tagger*. A média neste caso não é apresentada já que o seu valor não é indicado na referida publicação. A cardinalidade do conjunto de teste, usado nas experiências executados para o marcador OEP, é dado em número de palavras.

Marcador	Conjunto de treino	Teste	Média	Melhor
Marcador OEP	107200	21466	96.83	**96.86**
ETL-Tagger	1M	200K	-	96.75

No total de frases utilizadas foram marcados 21466 palavras. O marcador foi testado com diferentes parâmetros. Foram executados 20 *runs*, com um enxame de 10 e 20 partículas, durante 50 e 100 gerações. Os resultados obtidos podem ser consultados na Tabela 7.38.

Da análise dos resultados podemos concluir, que a melhor média, 96.83%, foi conseguida com um enxame de 10 partículas, durante 100 gerações. Contudo, a melhor exactidão, 96.89, foi obtida num *run* com um enxame de 20 partículas durante 50 gerações.

Na Tabela 7.39, apresentamos o melhor resultado conseguido com o marcador OEP, assim como os resultados apresentados em [Nogueira Dos Santos et al., 2008] para o *corpus Mac-Morpho*. Os valores apresentados foram convertidos para percentagem, e arredondados à segunda casa decimal, de forma a apresentarem o formato dos resultados publicados em [Nogueira Dos Santos et al., 2008].

Os resultados obtidos permite-nos confirmar a versatilidade da abordagem proposta, ao nível da sua aplicação a diferentes línguas. De facto, desde que estejam disponíveis *corpora* anotados, para que possam ser descobertas as regras de desambiguação, o marcador consegue, independentemente da língua, obter resultados competitivos.

Capítulo 8

Conclusões

Este último capítulo reúne sumariamente as principais contribuições desta tese, bem como os resultados obtidos no trabalho experimental realizado. Apresentamos, também, as linhas de investigação futura que emergem do trabalho apresentado.

8.1 Resultados Obtidos

O trabalho desenvolvido no âmbito desta tese explorou a aplicação de algoritmos da área da computação evolucionária à resolução do problema da marcação das partes do discurso, um problema bem conhecido da área de processamento de língua natural. Tanto quanto é do nosso conhecimento, esta é a primeira tentativa de utilização de técnicas da área da computação evolucionária para tratar todos os aspectos desta tarefa. Foram várias as conclusões que retirámos ao longo deste trabalho as quais tivemos oportunidade de apresentar nos capítulos respectivos. Fazemos agora aqui um resumo daquelas que, no nosso entender, assumem uma maior relevância. No entanto, antes de passarmos à síntese das conclusões, gostaríamos de relembrar as hipóteses que estabelecemos na introdução desta tese, e nas quais assenta todo o trabalho apresentado:

- É possível, a partir de recursos linguísticos específicos para a tarefa que se pretende resolver, generalizar a informação tipicamente utilizada, aprendendo regras de desambiguação, por aplicação de um algoritmo de classificação. Estas regras não terão como objectivo desempenhar o papel de um classificador, mas antes servirem como heurística para resolver a tarefa em causa.
- É possível formalizar o problema como um problema de procura e usar as regras descobertas na primeira fase, como heurística, para guiar a procura de uma solução no espaço de estados do problema.

Estas duas hipóteses resumem a metodologia proposta, a qual foi definida com a perspectiva de poder ser extensível a outras tarefas de processamento de língua natural, com características semelhantes às da marcação das PdD. Esta metodologia estabelece uma divisão do problema em duas tarefas distintas (ver Figuras 8.1 e 8.2): uma de aprendizagem e outra de optimização. A utilização de algoritmos da área da computação evolucionária nesta abordagem assentou principalmente em duas constatações: o manifesto sucesso conseguido por estas técnicas em ambas as áreas (aprendizagem e optimização), e sobretudo a sua adaptabilidade a diferentes tarefas e representações.

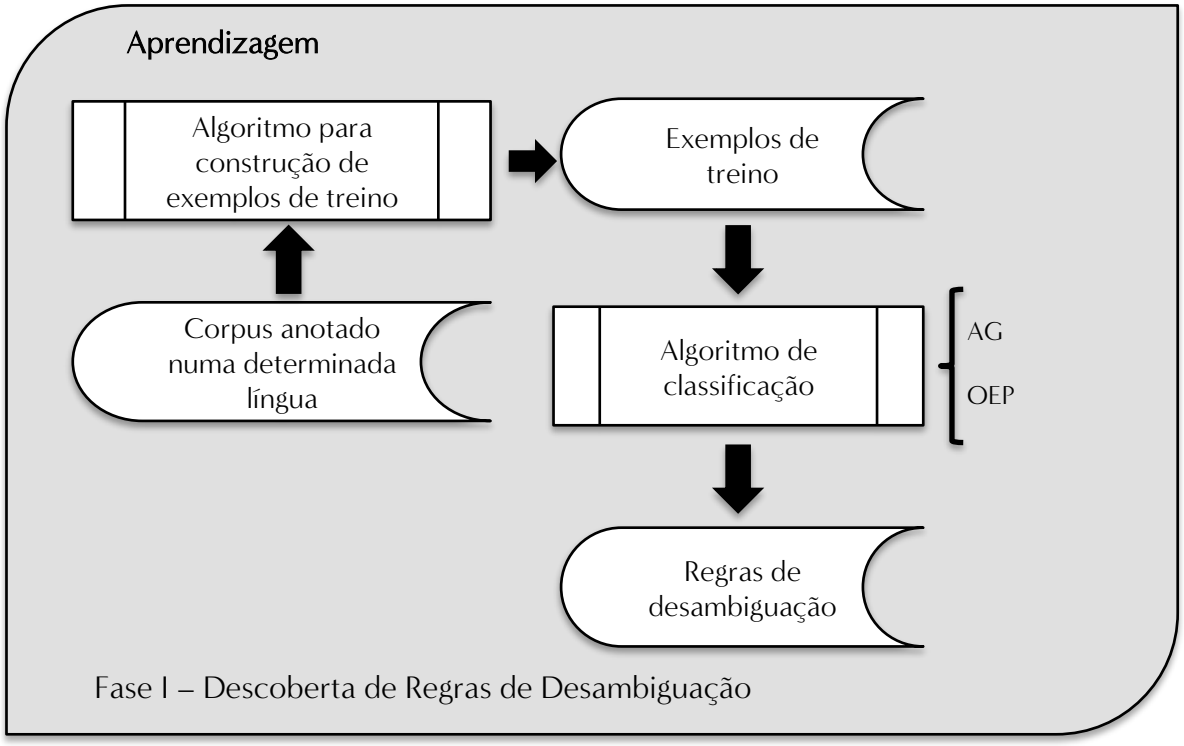


Figura 8.1: Estrutura da primeira fase da abordagem proposta.

O trabalho desenvolvido, e que apresentámos ao longo dos Capítulos 5, 6 e 7, permitiu comprovar, por aplicação ao problema concreto da marcação das partes do discurso, a validade das duas assunções inicialmente colocadas, e consequentemente a viabilidade da abordagem proposta. As principais contribuições no que diz respeito à marcação das PdD são:

- A desejada generalização da informação foi conseguida, como se pode comprovar com os resultados conseguidos na marcação do *corpus* do WSJ, usando como heurística as regras descobertas a partir de informação obtida a partir do *corpus* de *Brown*. Esta generalização fez-se sentir, não só numa maior independência do marcador em relação ao domínio de onde são obtidas as regras de desambiguação

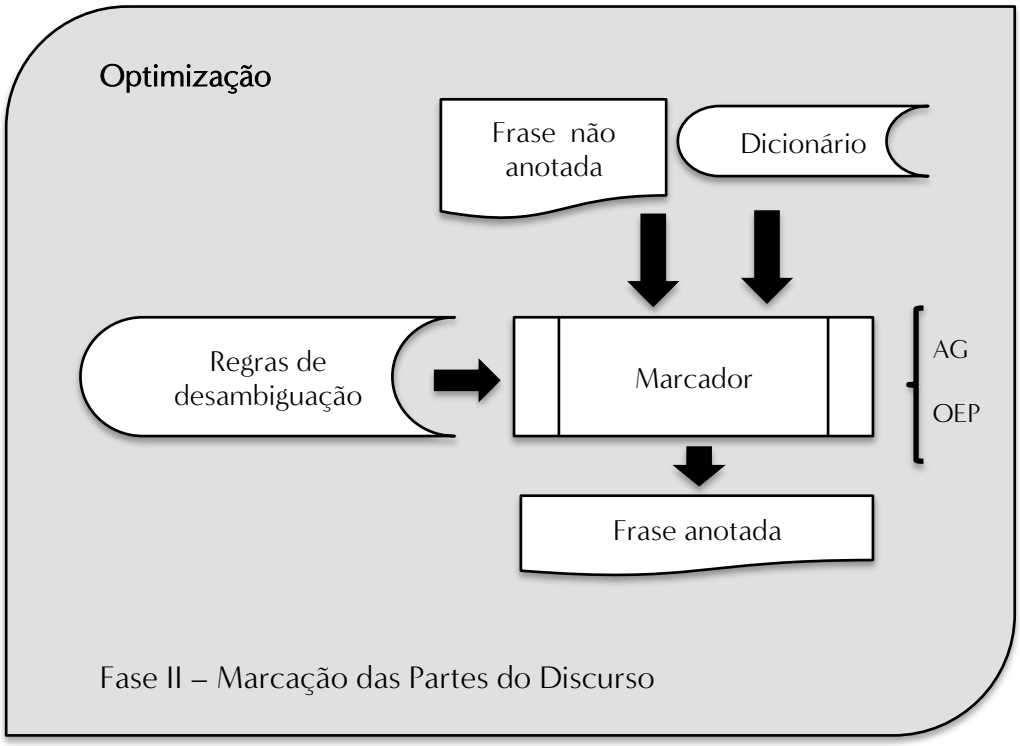


Figura 8.2: Estrutura da segunda fase da abordagem proposta.

(ver Tabela 8.1), mas também numa redução substancial da informação usada pelo marcador (ver Tabela 8.2).

Tabela 8.1: Comparação dos resultados obtidos no *corpus* do WSJ usando as regras de desambiguação descobertas a partir do *corpus* de *Brown*, assim como os resultados publicados em abordagens semelhantes (Araujo - [Araujo, 2002*b*]; GA - [Alba et al., 2006]; PGA - [Alba et al., 2006]; Wilson - [Wilson and Heywood, 2005]; Brill - [Brill, 1995]; Alba - [Alba et al., 2006]) quando treinadas no *corpus* do WSJ.

Corpus Teste	Marcador	Conjunto de Teste	Melhor	*Corpus* Treino
WSJ	OEP	100676	96.67	*Brown*
	Genético	100676	96.66	*Brown*
	Wilson	=Treino	89.80	WSJ
	Brill	150000	**97.20**	**WSJ**
	Alba	2544	96.63	WSJ

Tabela 8.2: Comparação dos resultados obtidos com os publicados por outras abordagens, tendo em conta o tamanho do *corpus* de treino (Araujo - [Araujo, 2002*b*]; GA - [Alba et al., 2006]; PGA - [Alba et al., 2006]; Wilson - [Wilson and Heywood, 2005]; Brill - [Brill, 1995]; Alba - [Alba et al., 2006]).

Corpus Teste	Marcador	Conjunto Treino	Melhor
Brown	OEP	**80000**	**96.78**
	Genético	80000	96.76
	Araujo	185000	95.40
	GA	165276	96.67
	PGA	165276	96.75

- A informação recolhida pode ser facilmente traduzida num formato de fácil compreensão, o que viabiliza a sua utilização em outros contextos (ver Tabela 8.3).

Tabela 8.3: Exemplo de regra obtida para a etiqueta **ADJ** (*Adjectivo*).

SE etiqueta(2, esquerda)=DET ∧ etiqueta(1, direita)=N ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, ADJ) com $f_\beta = 0.8299$

- A representação da informação sob a forma de regras de desambiguação permitiu a inclusão de outros aspectos para além dos tipicamente usados nas abordagens tradicionais, abrindo assim caminho à exploração de outras fontes de conhecimento que possam vir a revelar-se úteis na escolha da marcação adequada das palavras (ver Tabela 8.3).
- Os resultados obtidos permitiram identificar, como melhor marcador, o marcador definido à custa de um optimizador baseado em enxames de partículas. A

importância destes resultados assume um maior destaque pelo facto desta, tanto quanto sabemos, ser a primeira tentativa de aplicação de inteligência de enxame a este problema. O caminho à exploração de novas aplicações destes algoritmos a outras tarefas de processamento de língua natural fica, assim, aberto.

- Os valores de exactidão obtidos pelos marcadores desenvolvidos, com especial destaque para os obtidos pelo marcador OEP, encontram-se entre os melhores resultados publicados para os *corpora* usados no trabalho experimental (ver Tabela 8.4).

Tabela 8.4: Resultados obtidos em dois *corpus* em língua Inglesa, assim como os resultados publicados em abordagens semelhantes (Araujo - [Araujo, 2002*b*]; GA - [Alba et al., 2006]; PGA - [Alba et al., 2006]; Wilson - [Wilson and Heywood, 2005]; Brill - [Brill, 1995]; Alba - [Alba et al., 2006]).

Corpus Teste	Marcador	*Corpus* Treino	Conjunto de Teste	Melhor
Brown	OEP	Brown	22562	**96.78**
	Genético	Brown	22562	96.76
	Araujo	Brown	2500	95.40
	GA	Brown	17303	96.67
	PGA	Brown	17303	96.75
WSJ	OEP	Brown	100676	96.67
	Genético	Brown	100676	96.66
	Wilson	WSJ	=Treino	89.80
	Brill	WSJ	150000	**97.20**
	Alba	WSJ	2544	96.63

- Uma consequência directa da metodologia definida, é a de esta poder ser extensível a qualquer língua, desde que estejam disponíveis recursos linguísticos adequados para se poder fazer a aprendizagem das regras de desambiguação. As experiências realizadas para um *corpus* em língua Portuguesa permitiram comprovar a independência do marcador em relação à língua (ver Tabela 8.5).

Demonstrada a aplicabilidade da abordagem ao problema da marcação das PdD, julgamos estarem criadas condições para explorar a sua viabilidade no tratamento de outras tarefas de processamento de língua natural que satisfaçam as hipóteses matrizes

Tabela 8.5: Resultados obtidos para o *corpus Mac-Morpho*, em língua Portuguesa, juntamente com os resultados publicados em [Nogueira Dos Santos et al., 2008], aqui identificados por *ETL-Tagger*.

Marcador	Conjunto de treino	Conjunto de Teste	Melhor
Marcador OEP	107200	21466	**96.86**
ETL-Tagger	1M	200K	96.75

desta abordagem. Acreditamos, por isso, que a metodologia proposta, assim como os algoritmos que a integram, permitem estabelecer um novo paradigma para resolver um conjunto alargado de tarefas da área de processamento de língua natural.

8.2 Trabalho Futuro

Dentro da área de processamento de língua natural encontramos várias tarefas com características semelhantes às da marcação das PdD. Estas similaridades encontram-se na necessidade comum de realizar um determinado tipo de marcação num texto, a qual implica um qualquer processo de desambiguação. Como tivemos oportunidade de ver no problema da marcação das PdD, o objectivo, neste caso, é marcar todas as palavras de um texto. Contudo, existem outras tarefas que implicam, numa primeira fase, a identificação na frase da palavra, ou sequência de palavras, e, numa segunda, a associação das mesmas a uma determinada etiqueta. É o caso do problema da marcação de entidades mencionadas. Aqui o objectivo é identificar, e marcar numa frase, eventuais entidades que possam ocorrer na mesma. As entidades mencionadas abrangem diferentes categorias desde organizações, pessoas, referências geográficas, referências temporais, etc.

Um passo prévio à marcação de entidades mencionadas é a identificação na frase de frases nominais. Esta tarefa é também útil noutros aspectos do processamento de língua natural, como são os casos da análise sintáctica e tradução automática, assumindo um papel de especial relevo em várias tarefas de recuperação de informação, e.g. nos sistemas de pergunta-resposta.

As frases nominais são subfrases de uma frase principal, que são encabeçadas por um nome, ou cuja palavra principal é um nome. Estas unidades são as principais depositárias da informação relevante para a identificação do conteúdo de um texto. Uma das formas de marcar numa frase as suas frases nominais, consiste na utilização das etiquetas **I** (do inglês *Inside*, dentro), **O** (do inglês *outside*, fora) e **B** (do inglês *begin*, inicio). À semelhança da tarefa de marcação das PdD, também aqui é necessário aplicar um processo de desambiguação, já que há várias possibilidades de agrupar as

palavras de uma frase em subfrases. Esta tarefa obriga assim à identificação de frases nominais válidas numa frase e subsequente escolha da frase nominal mais adequada.

As características desta tarefa, resumidamente descritas em cima, assumem uma particular adequabilidade à aplicação da metodologia proposta nesta tese, na medida em que também aqui é expectável a validade das hipóteses matrizes:

- A identificação das frases nominais pode ser realizada à custa de uma gramática, que reúne um conjunto de regras cujo corpo pode ser definido por expressões regulares compostas pelas etiquetas que designam as diferentes PdD. Tal como para as regras de desambiguação também aqui é admissível descobrir esta gramática usando computação evolucionária. Os recursos linguísticos disponíveis permitem aprender estas regras a partir de *corpus* anotados com as etiquetas referidas. Cada uma das regras deverá ter também associado um valor de qualidade. O propósito não será apenas o de reconhecer uma frase nominal, mas o de orientar a procura no espaço de estados do problema de marcação, ajudando na escolha da melhor combinação.
- À semelhança da marcação das PdD, também aqui o problema da marcação pode ser formalizado como um problema de optimização combinatória, sendo que, neste caso, as hipóteses de marcação de cada uma das palavras tomam valores no conjunto $\{I, O, B\}$. A dimensão do espaço de estados do problema é igual a 3^n, sendo n o número de palavras da frase a marcar. Mais uma vez, a procura neste espaço implica a utilização de uma heurística adequada.

Na sequência da investigação realizadas para a tarefa das PdD, pretendemos estender a abordagem proposta a outras tarefas de processamento de língua natural, como é o caso da marcação de frases nominais, e, posteriormente, à marcação de entidades mencionadas.

Apêndice A

Regras de Desambiguação

Apresentam-se aqui exemplos de três regras de desambiguação para cada uma das etiquetas consideradas no conjunto de etiquetas adoptado no trabalho experimental.

Tabela A.1: Exemplos de regras obtidas para a etiqueta **ADJ** (*Adjectivo*).

Exemplos de regras obtidas para a etiqueta **ADJ** (*Adjectivo*)
SE etiqueta(2, esquerda)=DET ∧ etiqueta(1, direita)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, ADJ) com $f_\beta = 0.8299$
SE etiqueta(1, esquerda)=DET ∧ etiqueta(1, direita)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, ADJ) com $f_\beta = 0.9078$
SE etiqueta(1, esquerda)=P ∧ etiqueta(1, direita)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, ADJ) com $f_\beta = 0.8867$

Tabela A.2: Exemplos de regras obtidas para a etiqueta **ADV** (*Advérbio*).

Exemplos de regras obtidas para a etiqueta **ADV** (*Advérbio*)
SE etiqueta(1, direita)=VD ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, ADV) com $f_\beta = 0.7755$
SE etiqueta(1, esquerda)=N ∧ etiqueta(1, direita)=None ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, ADV) com $f_\beta = 6840$
SE etiqueta(1, esquerda)=MOD ∧ etiqueta(1, direita)=V ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, ADV) com $f_\beta = 0.6801$

Tabela A.3: Exemplos de regras obtidas para a etiqueta **CNJ** (*Conjunção*).

Exemplos de regras obtidas para a etiqueta **CNJ** (*Conjunção*)
SE etiqueta(1, esquerda)=NP ∧ etiqueta(1, direita)=NP ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, CNJ) com $f_\beta = 0.99$
SE etiqueta(1, esquerda)=ADJ ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, CNJ) com $f_\beta = 0.7823$
SE etiqueta(3, esquerda)=P ∧ etiqueta(1, esquerda)=N ∧ etiqueta(1, direita)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, CNJ) com $f_\beta = 0.7483$

Tabela A.4: Exemplos de regras obtidas para a etiqueta **DET** (*Determinante*).

Exemplos de regras obtidas para a etiqueta **DET** (*Determinante*)
SE etiqueta(2, esquerda)=N ∧ etiqueta(1, direita)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, DET) com $f_\beta = 0.9304$
SE etiqueta(1, esquerda)=P ∧ etiqueta(2, direita)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, DET) com $f_\beta = 0.9348$
SE etiqueta(1, esquerda)=P ∧ etiqueta(1, direita)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, DET) com $f_\beta = 0.8905$

Tabela A.5: Exemplos de regras obtidas para a etiqueta **EX** (*Existencial*).

Exemplos de regras obtidas para a etiqueta **EX** (*Existencial*)
SE etiqueta(1, direita)=V ∧
∧ Maiúscula(x) ∧ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, EX) com $f_\beta = 0.9780$
SE etiqueta(2, direita)=V ∧ etiqueta(3, direita)=DET ∧
∧ Maiúscula(x) ∧ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, EX) com $f_\beta = 0.8967$
SE etiqueta(1, direita)=VD∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, EX) com $f_\beta = 0.9521$

Tabela A.6: Exemplos de regras obtidas para a etiqueta **FW** (Palavra estrangeira).

Exemplos de regras obtidas para a etiqueta **FW** (*Palavras estrangeiras*)
SE etiqueta(1, esquerda)=FW ∧ ∧ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, FW) com $f_\beta = 0.9829$
SE etiqueta(2, direita)=FW ∧ etiqueta(3, direita)=P ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, FW) com $f_\beta = 0.7567$
SE etiqueta(1, direita)=FW ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, FW) com $f_\beta = 0.9635$

Tabela A.7: Exemplos de regras obtidas para a etiqueta **MOD** (Verbo modal).

Exemplos de regras obtidas para a etiqueta **MOD** (Verbo modal)
SE etiqueta(1, direita)=V ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, MOD) com $f_\beta = 0.9924$
SE etiqueta(1, esquerda)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, MOD) com $f_\beta = 0.9966$
SE etiqueta(1, direita)=V ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, MOD) com $f_\beta = 0.9966$

Tabela A.8: Exemplos de regras obtidas para a etiqueta **N** (*Nome*).

Exemplos de regras obtidas para a etiqueta **N** (*Nome*)
SE etiqueta(1, esquerda)=DET ∧ etiqueta(1, direita)=P ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, N) com $f_\beta = 0.8890$
SE etiqueta(1, esquerda)=ADJ ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, N) com $f_\beta = 0.8908$
SE etiqueta(2, esquerda)=P ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, N) com $f_\beta = 0.8368$

Tabela A.9: Exemplos de regras obtidas para a etiqueta **NP** (*Nome próprio*).

Exemplos de regras obtidas para a etiqueta **NP** (*Nome próprio*)
SE etiqueta(1, esquerda)=NP ∧ ∧ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, NP) com $f_\beta = 0.9304$
SE etiqueta(1, direita)=NP ∧ ∧ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, NP) com $f_\beta = 0.9121$
SE etiqueta(1, direita)=NP ∧ ∧ Maiúscula(x) ∧ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, NP) com $f_\beta = 0.8835$

Tabela A.10: Exemplos de regras obtidas para a etiqueta **NUM** (*Número*).

Exemplos de regras obtidas para a etiqueta **NUM** (*Número*)
SE etiqueta(2, direita)=N ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ Especiais(x)
ENTÃO Classe(x, NUM) com $f_\beta = 0.9418$
SE etiqueta(1, esquerda)=P ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ Especiais(x)
ENTÃO Classe(x, NUM) com $f_\beta = 0.9752$
SE etiqueta(1, esquerda)=NP ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ Especiais(x)
ENTÃO Classe(x, NUM) com $f_\beta = 0.9725$

Tabela A.11: Exemplos de regras obtidas para a etiqueta **PRO** (*Pronome*).

Exemplos de regras obtidas para a etiqueta **PRO** (*Pronome*)
SE etiqueta(1, direita)=V ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, PRO) com $f_\beta = 0.9448$
SE etiqueta(2, direita)=N ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, PRO) com $f_\beta = 9458$
SE etiqueta(2, direita)=V ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, PRO) com $f_\beta = 0.9368$

Tabela A.12: Exemplos de regras obtidas para a etiqueta **P** (*Preposição*).

Exemplos de regras obtidas para a etiqueta **P** (*Preposição*)
SE etiqueta(1, direita)=ADJ ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, P) com $f_\beta = 9086$
SE etiqueta(1, esquerda)=N ∧ etiqueta(1, direita)=DET ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, P) com $f_\beta = 0.9550$
SE etiqueta(1, esquerda)=N ∧ etiqueta(1, direita)=N ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, P) com $f_\beta = 0.9447$

Tabela A.13: Exemplos de regras obtidas para a etiqueta **TO** (*a palavra TO*).

Exemplos de regras obtidas para a etiqueta **TO** (*palavra to*)
SE etiqueta(1, direita)=V ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, TO) com $f_\beta = 9987$
SE etiqueta(1, direita)=V ∧
∧ Maiúscula(x) ∧ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, TO) com $f_\beta = 0.9894$
SE etiqueta(1, direita)=ADV ∧ etiqueta(2, direita)=V ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, TO) com $f_\beta = 9888$

Tabela A.14: Exemplos de regras obtidas para a etiqueta **VBZ** (*Verbo, 3ª pessoa do singular*).

Exemplos de regras obtidas para a etiqueta **VBZ** (*Verbo, 3ª pessoa do singular*)
SE etiqueta(1, esquerda)=WH ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, VBZ) com $f_\beta = 8555$
SE etiqueta(2, esquerda)=N ∧ etiqueta(1, esquerda)=WH ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, VBZ) com $f_\beta = 8606$
SE etiqueta(1, direita)=DET ∧ etiqueta(2, direita)=ADJ ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, V) com $f_\beta = 0.8353$

Tabela A.15: Exemplos de regras obtidas para a etiqueta **VD** (Verbo no tempo passado).

Exemplos de regras obtidas para a etiqueta **VD** (Verbo no tempo passado)
SE etiqueta(1, esquerda)=NP ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, VD) com $f_\beta = 0.9059$
SE etiqueta(1, esquerda)=NP ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, VD) com $f_\beta = 0.9116$
SE etiqueta(2, esquerda)=DET ∧ etiqueta(1, esquerda)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, VD) com $f_\beta = 0.8173$

Tabela A.16: Exemplos de regras obtidas para a etiqueta **VG** (Verbo no particípio presente).

Exemplos de regras obtidas para a etiqueta **VG** (Verbo no particípio presente).
SE etiqueta(1, esquerda)=P ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬ Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, VG) com $f_\beta = 0.9563$

SE etiqueta(2, direita)=N ∧ etiqueta(3, direita)=P ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, VG) com $f_\beta = 0.8572$

SE etiqueta(1, esquerda)=V ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, VG) com $f_\beta = 0.9515$

Tabela A.17: Exemplos de regras obtidas para a etiqueta **VN** (Verbo no particípio passado).

Exemplos de regras obtidas para a etiqueta **VN** (Verbo no particípio passado).
SE etiqueta(1, esquerda)=V ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬ Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, VN) com $f_\beta = 0.9867$
SE etiqueta(1, esquerda)=V ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ ¬Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, VN) com $f_\beta = 0.9594$
SE etiqueta(1, esquerda)=DET ∧ etiqueta(2, esquerda)=N ∧ ∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧ ∧ Termina(x,'ed') ∧ ∧ ¬Termina(x,'ing') ∧ ∧ ¬Termina(x,'ould') ∧ ∧ ¬Termina(x,' 's') ∧ ∧ ¬Termina(x,'s') ∧ ∧ ¬Especiais(x) **ENTÃO** Classe(x, VN) com $f_\beta = 0.9144$

Tabela A.18: Exemplos de regras obtidas para a etiqueta **WH** (determinador *wh*).

Exemplos de regras obtidas para a etiqueta **WH** (determinador *wh*)
SE etiqueta(1, direita)=V ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬ Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, WH) com $f_\beta = 0.9292$
SE etiqueta(1, direita)=VD ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, WH) com $f_\beta = 0.9485$
SE etiqueta(1, direita)=MOD ∧ etiqueta(2, direita)=V ∧
∧ ¬ Maiúscula(x) ∧ ¬ Primeira(x) ∧
∧ ¬Termina(x,'ed') ∧
∧ ¬Termina(x,'ing') ∧
∧ ¬Termina(x,'ould') ∧
∧ ¬Termina(x,' 's') ∧
∧ ¬Termina(x,'s') ∧
∧ ¬Especiais(x)
ENTÃO Classe(x, WH) com $f_\beta = 0.9047$

Bibliografia

Aires, R. V. X., Aluísio, S. M., Kuhn, D. C. e. S., Andreeta, M. L. B. and Oliveira, Jr., O. N. [2000], Combining classifiers to improve part of speech tagging: A case study for brazilian portuguese, *in* 'Proceedings of the Brazilian AI Symposium', SBIA'2000, ICMC/USP, pp. 20–22.

Alba, E., Luque, G. and Araujo, L. [2006], 'Natural language tagging with genetic algorithms', *Information Processing Letters* **100**(5), 173–182.

Aluísio, S., Pelizzoni, J., Marchi, A. R., de Oliveira, L., Manenti, R. and Marquiafável, V. [2003], An account of the challenge of tagging a reference corpus for brazilian portuguese, *in* 'Proceedings of the 6th International Conference on Computational Processing of the Portuguese Language', PROPOR'03, Springer-Verlag, Berlin, Heidelberg, pp. 110–117.

Araujo, L. [2002*a*], A parallel evolutionary algorithm for stochastic natural language parsing, *in* 'Proceedings of the 7th International Conference on Parallel Problem Solving from Nature', PPSN VII, Springer-Verlag, London, UK, pp. 700–709.

Araujo, L. [2002*b*], Part-of-speech tagging with evolutionary algorithms, *in* A. Gelbukh, ed., 'Computational Linguistics and Intelligent Text Processing', Vol. 2276 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 187–203.

Araujo, L. [2003], Studying the advantages of a messy evolutionary algorithm for natural language tagging, *in* 'Genetic and Evolutionary Computation — GECCO 2003', Vol. 2724 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 206–206.

Araujo, L. [2004], Genetic programming for natural language parsing, *in* M. Keijzer, U.-M. O'Reilly, S. M. Lucas, E. Costa and T. Soule, eds, 'Genetic Programming, 7th European Conference, EuroGP2004', Vol. 3003 of *Lecture Notes in Computer Science*, Springer, pp. 230–239.

Araujo, L., Luque, G. and Alba, E. [2004], Metaheuristics for natural language tagging, *in* 'Genetic and Evolutionary Computation - GECCO 2004, Genetic and Evolutionary Computation Conference', Vol. 3102 of *Lecture Notes in Computer Science*, Springer, pp. 889–900.

Araujo, L. and Santamaria, J. [2010], Evolving natural language grammars without supervision, *in* 'IEEE Congress on Evolutionary Computation', CEC'10, IEEE, pp. 1–8.

Banks, A., Vincent, J. and Anyakoha, C. [2007], 'A review of particle swarm optimization. part i: background and development', *Natural Computing* **6**, 467–484.

Banks, A., Vincent, J. and Anyakoha, C. [2008], 'A review of particle swarm optimization. part ii: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications', *Natural Computing* **7**, 109–124.

Banzhaf, W., Nordin, P., Keller, R. E. and Francone, F. D. [1997], *Genetic Programming : An Introduction : On the Automatic Evolution of Computer Programs and Its Applications (The Morgan Kaufmann Series in Artificial Intelligence)*, Morgan Kaufmann Publishers.

Beyer, H.-G. and Schwefel, H.-P. [2002], 'Evolution strategies – a comprehensive introduction', *Natural Computing* **1**, 3–52.

Bonabeau, E., Dorigo, M. and Theraulaz, G. [1999], *Swarm Intelligence - From Natural to Artificial Systems*, Oxford University Press.

Brill, E. [1995], 'Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging', *Computational Linguistics* **21**, 543–565.

Clerc, M. [2010], *Particle Swarm Optimization*, Wiley-ISTE.

Dalrymple, M. [2004], How much can tagging help parsing?, Technical report, Department of Computer Science, King's College London.

Darwin, C. [1872], *The Origin of Species by means of Natural Selection; or, the Preservation of Favoured Races in the Struggle for Life*, 6th edn, Modern Library.

de Jong, K. A., Spears, W. M. and Gordon, D. F. [1993], 'Using genetic algorithms for concept learning', *Machine Learning* **13**, 161–188.

De Pauw, G. [2003], Evolutionary computing as a tool for grammar development, *in* 'Genetic and Evolutionary Computation — GECCO 2003', Vol. 2723 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 202–202.

De Rose, S. J. [1988], 'Grammatical category disambiguation by statistical optimization', *Computational Linguistics* **14**, 31–39.

Dorigo, M. [1992], 'Optimization, learning and natural algorithms', *Ph.D. Thesis, Politecnico di Milano, Italy* .

Dorigo, M. and Blum, C. [2005], 'Ant colony optimization theory: A survey', *Theoretical Computer Science* **344**(2-3), 243–278.

Dorigo, M., Bonabeau, E. and Theraulaz, G. [2000], 'Ant algorithms and stigmergy', *Future Generation Computer Systems* **16**, 851–871.

Dorigo, M. and Stützle, T. [2004], *Ant Colony Optimization*, Bradford Company, Scituate, MA, USA.

Dréo, J. and Siarry, P. [2002], A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions, *in* M. Dorigo, G. Di Caro and M. Sampels, eds, 'Ant Algorithms', Vol. 2463 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 216–221.

Eberhart, R. C., Kennedy, J. and Shi, Y. [2001], *Swarm Intelligence*, Morgan Kaufmann series in evolutionary computation, Elsevier, Burlington, MA.

Eggermont, J., Eiben, A. E. and Hemert, J. I. [1999], A comparison of genetic programming variants for data classification, *in* D. Hand, J. Kok and M. Berthold, eds, 'Advances in Intelligent Data Analysis', Vol. 1642 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 281–290.

Engelbrecht, A. P. [2007], *Computational Intelligence: An Introduction*, 2nd edn, Wiley Publishing.

Eshelman, L. J. [1991], The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination, *in* 'Proceedings of the First Workshop on Foundations of Genetic Algorithms', Morgan Kaufmann, pp. 265–283.

Falco, I. D., Cioppa, A. D. and Tarantino, E. [2007], 'Facing classification problems with particle swarm optimization', *Applied Soft Computing* **7**(3), 652–658.

Fogel, D. B. [1999], *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*, 2nd edn, Wiley-IEEE Press.

Fogel, L. J., Owens, A. J. and Walsh, M. J. [1966], *Artificial Intelligence through Simulated Evolution*, John Wiley, New York, USA.

Forney, G.D., J. [1973], 'The viterbi algorithm', *Proceedings of the IEEE* **61**(3), 268 – 278.

Freitas, A. A. [2003], 'A survey of evolutionary algorithms for data mining and knowledge discovery', *Advances in Evolutionary Computing* pp. 819–845.

F.W. Nelson, H. K. [1979], Manual of information to accompany a standard corpus of present-day edited american english, for use with digital computers, Technical report, Dep. of Linguistics, Brown University.

G. Gazdar, E. Klein, G. P. and Sag, I. [1985], *Generalized Phrase Structure Grammar*, Harvard University Press.

Garnier, S., Gautrais, J. and Theraulaz, G. [2007], 'The biological principles of swarm intelligence', *Swarm Intelligence* **1**, 3–31.

Gelbukh, A. and Araujo, L. [2004], A probabilistic chart parser implemented with an evolutionary algorithm, *in* 'Computational Linguistics and Intelligent Text Processing', Vol. 2945 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 81–92.

Giordana, A. and Neri, F. [1995], 'Search-intensive concept induction', *Evolutionary Computation* **3**(4), 375–416.

Goldberg, D. E. [1989], *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.

Greene, D. P. and Smith, S. F. [1993], 'Competition-based induction of decision models from examples', *Machine Learning* **13**, 229–257.

Hansen, N. and Kern, S. [2004], *Evaluating the CMA Evolution Strategy on Multimodal Test Functions*, Vol. 3242 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 282–291.

Haupt, R. L. and Haupt, S. E. [2004], *Practical Genetic Algorithms with CD-ROM*, Wiley-Interscience.

Holden, N. and Freitas, A. A. [2005], A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data, *in* 'Proceedings of the IEEE Swarm Intelligence Symposium', SIS'2005, pp. 100–107.

Holland, J. [1975], *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

Holland, J. [1992], 'Genetic Algorithms', *Scientific American* **267**(1), 66–72.

Janikow, C. Z. [1993], 'A knowledge-intensive genetic algorithm for supervised learning', *Machine Learning* **13**, 189–228.

Joosse, W. [2006], The application of genetic algorithms in part-of-speech tagging for dutch corpora, *in* '4th Twente Student Conference on IT'.

Karaboga, D. and Akay, B. [2009], 'A survey: algorithms simulating bee swarm intelligence', *Artificial Intelligence Review* **31**, 61–85.

Kazakov, D., Page, D. and Manandhar, S. [1998], A hybrid approach to word segmentation, *in* 'Inductive Logic Programming', Vol. 1446 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 125–134.

Kennedy, J. and Eberhart, R. [1995], Particle swarm optimization, *in* 'Proceedings of the IEEE International Conference on Neural Networks', Vol. 4, pp. 1942–1948.

Kennedy, J. and Eberhart, R. [1997], A discrete binary version of the particle swarm algorithm, *in* 'Proceedings of the IEEE International Conference on Systems, Man and Cybernetics', Vol. 5, pp. 4104–4108.

Kennedy, J. and Eberhart, R. C. [2001], *Swarm intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Kepler, F. and Finger, M. [2006], Comparing two markov methods for part-of-speech tagging of portuguese, *in* J. Sichman, H. Coelho and S. Rezende, eds, 'Advances in Artificial Intelligence - IBERAMIA-SBIA 2006', Vol. 4140 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 482–491.

Korkmaz, E. E. and Üçoluk, G. [2001], Genetic programming for grammar induction, *in* G. E.D., ed., 'Genetic and Evolutionary Computation Conference Late Breaking Papers', GECCO'01, pp. 245–251.

Koza, J. [1994], *Genetic programming II: Automatic discovery of reusable programs*, Massachusetts Institute of Technology, Cambridge, MA.

Koza, J. R. [1992], *Genetic programming: on the programming of computers by means of natural selection*, MIT Press, Cambridge, MA, USA.

Koza, J. R. [2003], *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, Kluwer Academic Publishers, Norwell, MA, USA.

Langdon, W. B. and Poli, R. [2002], *Foundations of Genetic Programming*, Springer-Verlag.

Lankhorst, M. [1995], Automatic word categorization with genetic algorithms, *in* 'Proceedings of the ECAI'94, Workshop on Applied Genetic and Other Evolutionary Algorithms', Springer, Amsterdam.

Lankhorst, M. M. and Moddemeijer, R. [1993], Automatic word categorization: An information-theoretic approach, *in* I. K. S. Immink and e. P.G.M. de Bot, eds, 'Proceedings of the Fourteenth Symposium on Information Theory in the Benelux', pp. 62–69.

Losee, R. M. [1996], 'Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering: An empirical basis for grammatical rules', *Information Processing and Management* **32**(2), 185–197.

Maiti, D., Acharya, A., Konar, A. and Ramadoss, J. [2008], A novel parser design algorithm based on artificial ants, *in* 'Proceedings of the 4th International Conference on Information and Automation for Sustainability', ICIAFS 2008, pp. 247–250.

Martens, D., Baesens, B. and Fawcett, T. [2011], 'Editorial survey: swarm intelligence for data mining', *Machine Learning* **82**, 1–42.

Mitchell, M. [1996], *An introduction to genetic algorithms*, MIT Press, Cambridge, MA, USA.

M.P. Marcus, B. Santorini, M. M. [1994], 'Building a large annotated corpus of english: The penn treebank', *Computational Linguistics* **19**(2), 313–330.

Noda, E., Freitas, A. and Lopes, H. [1999], Discovering interesting prediction rules with a genetic algorithm, *in* 'Proceedings of the 1999 Congress on Evolutionary Computation', Vol. 2 of *CEC 99*.

Nogueira Dos Santos, C., Milidiú, R. L. and Rentería, R. P. [2008], Portuguese part-of-speech tagging using entropy guided transformation learning, *in* 'Proceedings of the 8th International Conference on Computational Processing of the Portuguese Language', PROPOR '08, Springer-Verlag, Berlin, Heidelberg, pp. 143–152.

Piasecki, M. and Gaweł, B. [2005], A rule-based tagger for polish based on genetic algorithm, *in* 'Intelligent Information Processing and Web Mining', Vol. 31 of *Advances in Soft Computing*, Springer Berlin / Heidelberg, pp. 247–255.

Poli, R. [2003], A simple but theoretically-motivated method to control bloat in genetic programming, *in* 'Genetic Programming', Vol. 2610 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 43–76.

Poli, R., Kennedy, J. and Blackwell, T. [2007], 'Particle swarm optimization', *Swarm Intelligence* **1**, 33–57.

Poli, R., Langdon, W. B. and McPhee, N. F. [2008], *A field guide to genetic programming*, Published via http://lulu.com.

Pulman, S. G. [1991], Basic parsing techniques: an introductory survey, Technical report, University of Cambridge Computer Laboratory and SRI International, Cambridge.

Rechenberg, I. [1973], *Evolutionsstrategie – Optimierung technisher systeme nach prinzipien der biologischen evolution*, Frommann-Holzboog, Stuttgart, GER.

Schwefel, H. P. [1974], *Numerische optimierung von computer-modellen*, Birkhäuser.

Schwefel, H. P. [1981], *Numerical optimization of computer models*, Wiley & Sons, Chichester.

Serrano, J. I. and Araujo, L. [2005], Evolutionary algorithm for noun phrase detection in natural language processing, *in* 'Congress on Evolutionary Computation', CEC'05, IEEE, pp. 640–647.

Silva, A. P. and Rodrigues, I. [2013], Discovery of disambiguation rules for the pos problem using genetic algorithms, *in* 'Actas das Terceiras Jornadas de Informática da Universidade de Évora', JIUE 2013, Évora.

Silva, A. P., Silva, A. and Rodrigues, I. [2012*a*], Biopos: Biologically inspired algorithms for pos tagging, *in* 'Proceedings of the 1st International Workshop on Optimization Techniques for Human Language Technology', OPTHLT 2012 / COLING 2012, Mumbai, India, pp. 1–16.

Silva, A. P., Silva, A. and Rodrigues, I. [2012*b*], Tagging with disambiguation rules: A new evolutionary approach to the part-of-speech problem, *in* 'Proceedings of the 4th International Conference on Evolutionary Computation Theory and Applications', IJCCI 2012, Barcelona, pp. 5–14.

Silva, A. P., Silva, A. and Rodrigues, I. [2013*a*], An approach to the pos tagging problem using genetic algorithms, *in* 'Studies in Computational Intelligence', Springer-Verlag.

Silva, A. P., Silva, A. and Rodrigues, I. [2013*b*], Pso-tagger: A new biologically inspired approach to the part-of-speech tagging problem, *in* 'Proceedings of the 11th International Conference on Adaptive and Natural Computing Algorithms', Lecture Notes in Computer Science, Springer, Lausanne, Switzerland.

Smith, T. C. and Witten, I. H. [1995], A genetic algorithm for the induction of natural language grammars, *in* 'Proceedings of the Workshop on New Approaches to Learning for Natural Language Processing', IJCAI'95, pp. 17–24.

Socha, K. and Dorigo, M. [2008], 'Ant colony optimization for continuous domains', *European Journal of Operational Research* **185**(3), 1155–1173.

Sousa, T., Neves, A. and Silva, A. [2003], Swarm optimisation as a new tool for data mining, *in* 'Proceedings of the 17th International Symposium on Parallel and Distributed Processing', IPDPS '03, IEEE Computer Society, Washington, DC, USA, p. 144.

Sousa, T., Silva, A. and Neves, A. [2004], 'Particle swarm based data mining algorithms for classification tasks', *Parallel Computing* **30**(5), 767–783.

Sousa, T., Silva, A., Neves, A. and Costa, E. [2005], Bio-inpired datamining, *in* S. Olariu and A. Zomaya, eds, 'Handbook of Bioinspired Algorithms and Applications', CRC Press, USA, p. 467.

Steven Bird, E. K. and Loper, E. [2009], *Natural Language Processing with Python*, O'Reilly Media.

Suresh Manandhar, S. D. and Erjavec, T. [1998], Learning multilingual morphology with CLOG, *in* 'The Eight International Conference on Inductive Logic Programming', Vol. 1446 of *Lecture Notes in Computer Science*, Springer, pp. 135–144.

Tanaka, H. [1993], Current trends on parsing - a survey, Technical Report TR 93 003, TITCS.

Weise, T. [2008], *Global Optimization Algorithms – Theory and Application*, Published via www.it-weise.de.

Wilson, G. and Heywood, M. [2005], Use of a genetic algorithm in Brill's transformation-based part-of-speech tagger, *in* 'Proceedings of the 2005 Conference on Genetic and Evolutionary Computation', GECCO'05, ACM, New York, USA, pp. 2067–2073.

Wong, M. and Leung, K. [2000], *Data mining using grammar based genetic programming and applications*, Kluwer Academic.

Contactos:
Universidade de Évora
Instituto de Investigação e Formação Avançada - IIFA
Palácio do Vimioso | Largo Marquês de Marialva, Apart. 94
7002-554 Évora | Portugal
Tel: (+351) 266 706 581
Fax: (+351) 266 744 677
email: iifa@uevora.pt