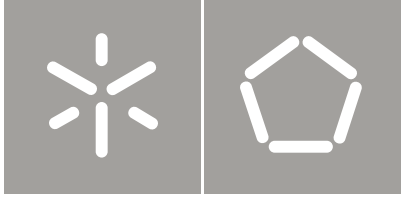
Universidade do Minho
Escola de Engenharia

Rui Fernando Martins Vaz

Estratégias de Filtragem *Anti-Spam* baseadas em Técnicas de Computação Evolucionária

Estratégias de Filtragem *Anti-Spam* baseadas em Técnicas de Computação Evolucionária
Rui Fernando Martins Vaz
UMinho | 2010

fevereiro de 2012



Universidade do Minho
Escola de Engenharia

Rui Fernando Martins Vaz

Estratégias de Filtragem *Anti-Spam* baseadas em Técnicas de Computação Evolucionária

Tese de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia de Comunicações

Trabalho efetuado sob a orientação do
Professor Doutor Paulo Alexandre Ribeiro Cortez

Professor Doutor Pedro Nuno Miranda de Sousa

fevereiro de 2012

Agradecimentos

Em primeiro lugar agradeço aos meus orientadores Professor Doutor Paulo Cortez e Professor Doutor Pedro Sousa e também ao Professor Doutor Miguel Rocha por toda a disponibilidade, orientação e colaboração no desenvolvimento deste trabalho.

Em segundo lugar agradeço à minha família, especialmente aos meus pais e às minhas irmãs por toda a dedicação e apoio.

Não posso também deixar de expressar o meu agradecimento a todas as pessoas que de alguma forma tornaram possível a realização desta dissertação, especialmente aquelas com quem mais convivi ao longo do meu percurso académico, entre as quais gostaria de destacar o Daniel Quinta, o João Veiga, o Francisco Silva e o Carlos Carneiro.

Por último, é de referir que este trabalho foi suportado por uma Bolsa de Investigação (BI), financiada pela Fundação para a Ciência e a Tecnologia (FCT), dentro do projecto de R&D PTDC/EIA/64541/2006 - "SPAM Telescope Miner: detecção a nível mundial de correio electrónico não solicitado via técnicas de data mining".

Resumo

O serviço de correio eletrónico é atualmente um serviço de comunicação essencial, que assume uma crescente importância na sociedade atual. No entanto, apesar dos vários esforços concentrados contra o correio eletrónico (*email*) não solicitado, designado também por *spam*, este continua a ser ainda um problema inerente a este serviço.

No quadro das soluções tecnológicas, os métodos de filtragem baseados no conteúdo das mensagens de *email*, que utilizam técnicas de *data mining*, são atualmente os mais populares e amplamente utilizados para combater este problema. No âmbito destes métodos, a seleção de atributos que melhor caraterizam as mensagens de *spam* (e.g., palavras mais correlacionadas com mensagens de *spam*), constitui um passo importante no desenvolvimento de filtros mais assertivos. Nesse sentido, é efetuado neste trabalho um estudo empírico da introdução de técnicas de computação evolucionária de seleção de atributos no contexto da filtragem *anti-spam*.

De forma a avaliar o método proposto foram desenvolvidos diversos filtros *anti-spam* que implementam, usando estratégias diferentes, técnicas de computação evolucionária de seleção de atributos. Uma das estratégias desenvolvida segue uma abordagem colaborativa que permite a troca de atributos relevantes entre filtros locais. O desempenho dos filtros *anti-spam* que utilizam técnicas de computação evolucionária de seleção de atributos são analisados. Posteriormente o desempenho do filtro colaborativo é comparado com um filtro padrão que utiliza apenas um método de seleção de atributos baseado num critério de informação.

Palavras Chave: *Spam*, Filtragem Baseada no Conteúdo, Seleção de Atributos, Aprendizagem Máquina, Algoritmos Evolucionários, Filtragem Colaborativa.

Abstract

Nowadays electronic mail (email) service assumes an increasing importance in modern society and is considered an essential communication service. However, despite the several efforts made against the unsolicited email (also known as spam), it remains an inherent problem which affects this service.

Within the existing technological solutions, Content-Based Filtering (CBF) methods, that use data mining techniques, are currently the most popular approaches to solve this issue. Feature selection techniques are essential in CBF methods. These techniques allow the selection of a reduced set of relevant attributes (e.g., words correlated with spam messages) that provides essential information to enhance the accuracy of anti-spam filters. Hence, in this work we perform an empirical study concerning the introduction of evolutionary computation techniques for feature selection in the scope of anti-spam filtering.

In order to evaluate the proposed method several anti-spam filters were developed. These filters implement, through different strategies, evolutionary computation techniques for feature selection. One of these strategies follows a collaborative approach which enables the exchange of relevant attributes between local filters. The performances of the developed filters that implement evolutionary computation techniques are evaluated. Afterwards, the performance of the collaborative filter is compared to a standard filter which uses a feature selection method based on an information criterion.

Keywords: Spam, Content-Based Filtering, Feature Selection, Machine Learning, Evolutionary Algorithms, Collaborative Filtering.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AGE	Algoritmo Genético e Evolucionário
ARFF	*Attribute-Relation File Format*
ARPANet	*Advanced Research Projects Agency Network*
AS	*Autonomous System*
AUC	*Area Under the Curve*
BCC	*Blind Carbon Copy*
BGP	*Border Gateway Protocol*
CBF	*Content-Based Filtering*
CGE	Computação Genética e Evolucionária
CSV	*Comma-Sepparated Values*
DHA	*Directory Harvest Attack*
DNS	*Domain Name System*
DNSBL	*Domain Name System based BlacklList*
DNSWL	*Domain Name System based Whitelist*
ESP	*Email Service Provider*
HTML	*Hypertext Markup Language*
IG	*Information Gain*
IMAP4	*Internet Message Access Protocol - Version 4*
IMP	*Interface Message Processors*
IP	*Internet Protocol*
ISP	*Internet Service Provider*

JECoLi	*Java Evolutionary Computation Library*
k-NN	*k-Nearest Neighbor*
MDA	*Mail Delivery Agent*
MPX	*Multiple Point Crossover*
MTA	*Mail Transfer Agent*
MUA	*Mail User Agent*
MVBNB	*Multivariate Bernoulli Naive Bayes*
OCR	*Optical Character Recognition*
P2P	*Peer to Peer*
POP3	*Post Office Protocol - Version 3*
rDNS	*reverse Domain Name System*
RFC	*Request for Comments*
ROC	*Receiver Operating Characteristic*
SMTP	*Simple Mail Transfer Protocol*
SPX	*Single Point Crossover*
SVM	*Support Vector Machine*
TCP	*Transmission Control Protocol*
TFP	Taxa de Falsos Positivos
TIC	Tecnologias da Informação e Comunicação
TPX	*Two Point Crossover*
TVP	Taxa de Verdadeiros Positivos
Usenet	*Unix User Network*
YALE	*Yet Another Learning Environment*

Capítulo 1

Introdução

1.1 Enquadramento e Motivação

A integridade e eficiência das aplicações e serviços, e a confiança dos utilizadores na sua utilização é fundamental para criar benefícios económicos e sociais que provêm das Tecnologias da Informação e Comunicação (TIC) [40]. No entanto, a integridade do serviço de correio eletrónico, e consequentemente a confiança dos utilizadores nesta tecnologia, é ameaçada pela intensificação das mensagens de *spam*, que prejudicam diariamente indivíduos, organizações e empresas.

Atualmente o *spam* deixou de ser apenas um fenómeno incómodo, para passar a ser um problema potencialmente mais perigoso, que põe em causa a segurança e privacidade dos utilizadores do serviço de *email*. O problema do *spam* é portanto um problema de segurança, como tal devem identificar-se os métodos de controlo do serviço de correio eletrónico contra esta atividade. Por outro lado, para melhor compreender o fenómeno é também necessário compreender as razões que levam os *spammers* a realizar esta atividade e quais as metodologias que utilizam.

A grande motivação para a proliferação de *spam* é sobretudo o marketing [50]. O envio em massa de mensagens de publicidade anunciam desde fármacos até esquemas de enriquecimento fácil. Através da colheita de endereços de correio eletrónico, recorrendo a *web crawlers* [41], ou à compra de listas já compiladas de endereços de correio eletrónico [44], os *spammers* obtêm uma grande quantidade de endereços de *email*. Isto possibilita-os a enviarem mensagens *emails* indiscriminadamente a um grande número de utilizadores do serviço de correio eletrónico. Este facto associado ao baixo custo do envio de mensagens de *email* atraiu um elevado número de opor-

tunistas [3]. Para os *spammers* esta continua a ser uma atividade que compensa pois os riscos de serem localizados são muito baixos. Além disso é uma prática que gera lucro aos *spammers* e às empresas que recorrem aos seus serviços [29].

A constante adaptação dos métodos utilizadas na proliferação de *spam*, e a própria evolução do conteúdo destas mensagens obriga ao desenvolvimento de novas técnicas mais robustas e mais adaptativas no combate ao *spam*. A utilização de técnicas *anti-spam* colaborativas mais estáticas, como *blacklists* ou *whitelists*, revelam-se insuficientes. Estas técnicas utilizam listas de endereços *Internet Protocol* (IP) para identificar os endereços IP associadas a fontes de *spam*, no caso das *blacklists*, ou os endereços IP de remetentes legítimos, nas *whitelists* (ver Secção 2.6.2). Estas listas têm de ser periodicamente atualizadas e os *spammers* utilizam técnicas que lhes permitem mudar constantemente de endereço IP, tornando assim difícil bloquear os *mail relays* que reencaminham *spam* (ver Secção 2.5).

As técnicas de filtragem de *spam* baseadas no conteúdos das mensagens de *email*, são dos métodos mais utilizados no combate ao *spam*. Estes métodos baseiam-se no conteúdo da mensagem para a classificarem como *spam* ou legítima. Uma desvantagem destes métodos é o facto de ser necessário treinar um filtro com instâncias prévias de mensagens de *email*, o que nem sempre é possível para caixas de correio eletrónico recentes. Por outro lado muitas vezes são necessários conjuntos de dados de grande dimensão para treinar os filtros de modo a estes serem eficazes. A seleção de um sub-grupo de atributos dos conjuntos de dados que representam as caixas de correio eletrónico é uma medida vantajosa neste contexto. Desta forma é possível reduzir o número de atributos que caraterizam as mensagens o que torna o processo de filtragem mais rápido. Por outro lado são eliminados atributos irrelevantes o que permite construir modelos de classificação mais assertivos [7].

A seleção dos atributos, que caraterizam as mensagens de *email*, utilizados para construir os modelos de classificação através dos algoritmos de aprendizagem máquina, constitui um passo importante no desenvolvimento de filtros mais fiáveis [26, 23, 52]. De uma forma geral existem dois métodos para seleção de atributos: filtros[1] e *wrappers*. Os métodos baseados em filtros são independentes do algorit-

[1]Não confundir filtros de seleção, utilizados para selecionar atributos durante a fase de pré-processamento, com filtros *anti-spam* que determinam se uma mensagem é *spam* ou legítima.

mos de aprendizagem máquina e são aplicados na fase de pré-processamento. Estes métodos utilizam uma medida de relevância (e.g., *Information Gain* (IG)[2], *Mutual Information*) para selecionar os melhores atributos. Por outro lado os métodos *wrapper* treinam um classificador com um sub-conjunto de atributos e calculam as métricas de avaliação (*Area Under the Curve* (AUC)[3], *precision*, *recall*) sobre amostras de treino. O sub-conjunto de atributos que atingir os melhores resultados de classificação é utilizado para construir o modelo de classificação final.

Neste trabalho é introduzido um método *wrapper* de seleção de atributos baseado em técnicas de computação evolucionária. As técnicas de computação evolucionária proporcionam o "cérebro"de otimização para seleção dos atributos mais relevantes que permitam maximizar os resultados de classificação do filtro *anti-spam*, reduzindo o número de atributos necessários para esta tarefa. Posteriormente foi também desenvolvido um filtro colaborativo *anti-spam* que permite a troca de atributos entre filtros locais que utilizam técnicas de computação evolucionária. Através desta troca de atributos, que consistem em palavras presentes nas mensagens de *email*, entre diferentes filtros pertencentes a diferentes utilizadores pretende-se melhorar a tarefa de classificação das mensagens de *email*.

1.2 Objetivos

Neste projeto pretende-se realizar um estudo empírico da utilização de técnicas de computação evolucionária para seleção de atributos de forma a construir, através de algoritmos de aprendizagem máquina, modelos de classificação mais assertivos no âmbito dos processos de filtragem *anti-spam*. Como segundo objetivo pretende-se desenvolver um filtro *anti-spam* colaborativo que aplique estas técnicas. Para isso os filtros *anti-spam* locais que utilizam técnicas de computação evolucionária vão partilhar atributos que consideram relevantes. Estes atributos partilhados são utilizados para alargar o domínio de procura do Algoritmo Genético e Evolucionário (AGE) local pela combinação de atributos que maximize a assertividade do filtro *anti-spam*. Os filtros baseiam-se no conteúdo da mensagem de *email* para a classificar como

[2]Critério de seleção de atributos muito utilizado no contexto da filtragem *anti-spam* (ver Secção 5.2.2).

[3]Métrica que permite avaliar o desempenho de classificadores (ver Secção 5.4).

spam ou legítima. Inicialmente pretende-se comparar e analisar o desempenho dos filtros que utilizam técnicas de computação evolucionária, incluindo o filtro colaborativo. Por fim pretende-se também analisar e comparar os resultados obtidos entre o filtro colaborativo e um filtro *anti-spam* que utiliza apenas um método de seleção de atributos baseado no IG.

Neste contexto os objetivos definidos para este trabalho são os seguintes:

- **Ambientação à área científica do projeto e estudo de bibliografia relacionada** - Estudo do panorama atual do problema do *spam* no serviço de correio eletrónico. Identificar as metodologias utilizadas nesta atividade, bem como os esforços no combate a esta prática. Efetuar um estudo mais aprofundado relativamente às propostas baseadas em filtros *anti-spam* que recorrem a técnicas de *data mining*.

- **Estudo e experimentação de tecnologias relevantes para o desenvolvimento** - Identificar as tecnologias que possibilitem o desenvolvimento dos filtros *anti-spam*, nomeadamente ferramentas de *data mining* e *frameworks* que permitam implementar AGEs. Familiarização com as tecnologias selecionadas efetuando alguma experimentação.

- **Desenvolvimento dos filtros locais** - Desenvolvimento dos filtros locais que integram técnicas de computação evolucionária de seleção dos atributos mais relevantes para construção de um modelo de classificação através de algoritmos de aprendizagem máquina. Implementar também um filtro padrão que utilize apenas um método de filtragem de seleção de atributos com o intuito de comparar esta abordagem tradicional com os filtros *anti-spam* que utilizam técnicas de computação evolucionária.

- **Desenvolvimento do filtro colaborativo** - Desenvolvimento de um filtro colaborativo que permita a troca de atributos entre os filtros locais que utilizam técnicas de computação evolucionária.

- **Experimentação e análise dos resultados obtidos** - Realizar experiências com os filtros *anti-spam* desenvolvidos sobre caixas de *email* pertencentes a vários utilizadores. Analisar e avaliar os resultados dos filtros *anti-spam* locais e colaborativo que utilizam técnicas de computação evolucionária. Por fim analisar os resultados do filtro colaborativo que utiliza técnicas de computação

evolucionária de seleção de atributos e do filtro simples que utiliza apenas o IG como método de seleção.

1.3 Organização

A estrutura deste documento está dividida em 6 capítulos principais organizados da seguinte forma:

- **Introdução** : Capítulo onde é efetuado um enquadramento introdutório à temática abordada neste trabalho. Além disso são apresentadas as motivações e os principais objetivos delineados para este trabalho.

- **A Problemática do *Spam*** : Descreve e contextualiza a problemática do *spam* que afeta atualmente o serviço de correio eletrónico. Inicialmente é apresentada a definição do conceito *spam*, e são explicados quais os inconvenientes para as pessoas e organizações afetadas por esta prática. Posteriormente são explicadas as metodologias e técnicas utilizadas pelos *spammers*. Por último são descritos os variados esforços no combate contra esta atividade, com maior ênfase para os métodos de filtragem baseados no conteúdo das mensagens de *email*.

- **Algoritmos Genéticos e Evolucionários** : Neste Capítulo são explicados os AGEs. É referida a sua inspiração em processos que ocorrem na natureza e como podem ser utilizados para resolver problemas relacionados com otimização, procura e aprendizagem. Ao longo do Capítulo é descrita a estrutura dos AGEs e os elementos que a constituem.

- **Soluções Desenvolvidas** : Os filtros *anti-spam* desenvolvidos são explicados neste Capítulo. Este filtros baseiam-se no conteúdo das mensagens de *email* e utilizam algoritmos de aprendizagem máquina para classificar as mensagens como *spam* ou legítimas. Três versões dos filtros desenvolvidos utilizam técnicas de computação evolucionária para procura de atributos, e uma versão utiliza apenas um critério de seleção de atributos, o IG. Esta última versão foi desenvolvida para comparar os resultados obtidos relativamente às versões que utilizam técnicas de computação evolucionária. No início deste Capítulo são também descritas as bibliotecas Java utilizadas no desenvolvimento, no-

meadamente a biblioteca *Java Evolutionary Computation Library* (JECoLi), e a biblioteca RapidMiner 4.6.

- **Experiências e Resultados Obtidos** : Inicialmente neste Capítulo é apresentada a forma como foram constituídas as caixas de *email* utilizadas para avaliar os filtros *anti-spam* desenvolvidos. Posteriormente são descritas as métricas de avaliação utilizadas. Finalmente são descritas as configurações das experiências realizadas e analisados os resultados obtidos.

- **Conclusão** : É apresentada uma síntese do trabalho realizado neste projeto e uma análise crítica aos resultados obtidos. Além disso são apresentadas algumas ideias de como seria possível melhorar os filtros *anti-spam* desenvolvidos, e de que forma estes poderiam ser implementados em ambiente real.

No final do documento são apresentados alguns anexos de natureza técnica mais detalhada que por motivos de facilidade de leitura não foram incluídos no corpo textual deste documento.

Capítulo 2

A Problemática do *Spam*

Neste Capítulo é efetuada uma contextualização do problema do *spam* no âmbito do serviço de correio eletrónico. Inicialmente é descrito de uma forma breve o serviço de correio eletrónico, nomeadamente o processo de entrega de uma mensagem de *email* e os protocolos envolvidos. Posteriormente são identificados métodos utilizados para a prática do *spam* que se podem dividir-se em dois processos. O primeiro corresponde à recolha de endereços de *email*. O segundo consiste no envio das mensagens de *spam* para os endereços obtidos. Por último são explicadas as diferentes abordagens existentes para minorar este problema, com especial ênfase para a filtragem *anti-spam* baseada no conteúdo das mensagens de *email*.

2.1 Introdução

Em 1975 foi identificado num *Request for Comments* (RFC) um problema no protocolo de interface *Host/Interface Message Processors (IMP)* da *Advanced Research Projects Agency Network* (ARPANet) que consistia na inexistência de um mecanismo que permitisse a um terminal recusar selectivamente a receção de uma mensagem [42]. Foi a primeira vez que surgiu na literatura o reconhecimento das mensagens indesejáveis no serviço de *email*. Conforme as suspeitas, na década de 90 surgiram os primeiros avisos na *Unix User Network* (Usenet). Utilizando o mecanismo de *crossposting* diversos *newsgroups* receberam mensagens em massa, abusivas sem consentimento [56].

Inicialmente os problemas relacionadas com o *spam*, na perspectiva do utilizador, eram sobretudo as questões afetas à produtividade, e ao facto de este ser um fator

de irritabilidade para os utilizadores do *email*. Em [53] foi efetuado um estudo que revela que os funcionários das empresas inquiridas perdem diariamente um total de 13 minutos para lidar com as mensagens de *spam*. De acordo com o mesmo estudo os inquiridos revelam que a redução da reputação do serviço de *email* é outra razão pela qual o *spam* é visto como um problema.

Atualmente o *spam* é também utilizado para outros fins além da publicidade comercial, nomeadamente como um veículo para a proliferação de *malware* e também como uma forma de atrair os utilizadores para esquemas fraudulentos [40]. Estas mensagens são utilizadas para obter indevidamente informação pessoal, financeira, e acesso a *passwords* [51]. O *spam* deixou de ser apenas um fenómeno incómodo, e é hoje em dia um problema potencialmente perigoso. Para os *Email Service Provider*s (ESPs) os custos estão associados à necessidade de implementar soluções *anti-spam*, de onde advém também custos de suporte técnico. Além disso existe a necessidade de expansão da infraestrutura para lidar com a grande quantidade de mensagens [40]. De acordo com [35], a percentagem de *emails* abusivos nos primeiros nove meses de 2011 variou entre 88% e 90%. Estes números sugerem que a industria deve continuar vigilante e ativa para combater este problema. No estudo efetuado em [34] 43% de utilizadores inquiridos admitem ter aberto mensagens que suspeitavam ser *spam*, 11% asseguram que clicaram em links, e 8% abriram anexos. Estes comportamentos de risco revelam que os objetivos dos *spammers* muitas vezes se concretizam.

Antes de serem explicadas as metodologias usadas pelos *spammers* nas suas atividades e as abordagens existentes no combate a este problema é efetuada uma breve explicação do serviço de correio eletrónico.

2.2 Serviço de Correio Eletrónico

Na Figura 2.1[1] está representado o processo de entrega de uma mensagem no serviço de correio eletrónico. Os principais elementos intervenientes neste processo são o *Mail User Agent* (MUA), o *Mail Transfer Agent* (MTA) e o *Mail Delivery Agent* (MDA). O MUA consiste na aplicação cliente de *email* que o utilizador

[1]Figura obtida em [51].

utiliza para compor a mensagem que é depois transmitida para o cliente *Simple Mail Transfer Protocol* (SMTP). Usualmente o MUA integra já o cliente SMTP (e.g., *Microsoft Outlook*, *Apple Mail*). Os MTAs são responsáveis pela transferência das mensagens de *email* entre o MUA e o MDA. Normalmente uma mensagem é transferida entre vários MTAs, que podem pertencer a organizações diferentes, até chegar ao destinatário. O protocolo utilizado pelos MTAs para transferirem as mensagens de *email* é o SMTP. Quando a mensagem chega a um MTA da organização do destinatário é transferida para um MDA que armazena a mensagem no servidor de *email* [51]. Por último o MUA do destinatário utiliza o protocolo *Post Office Protocol - Version 3* (POP3) ou o *Internet Message Access Protocol - Version 4* (IMAP4) para para aceder remotamente às suas mensagens de *email*.

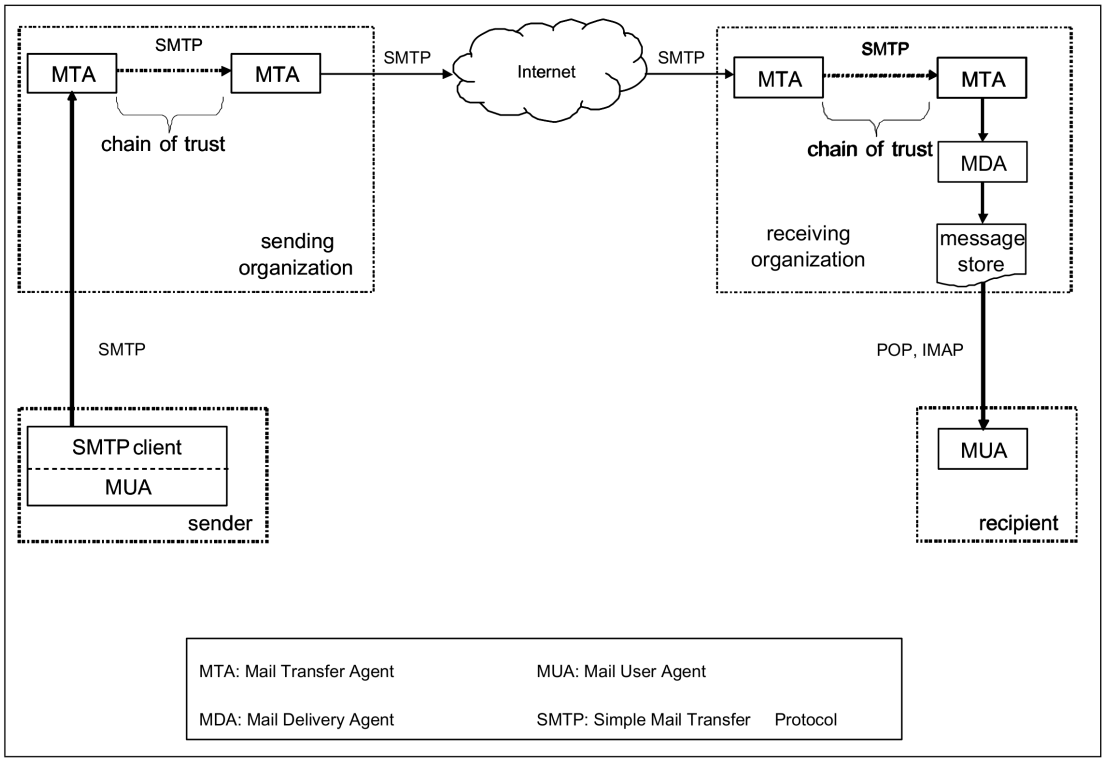


Figura 2.1: Processo de entrega de mensagem no serviço de correio eletrónico.

2.2.1 Protocolos de Envio/Receção

No serviço de correio eletrónico existem vários protocolos de comunicação envolvidos na transferência de uma mensagem até ao destinatário final. Na Secção anterior foram referidos três protocolos de aplicação o SMTP, o POP3 e o IMAP4

que são agora explicados.

O SMTP é um protocolo para transmissão de mensagens de *email* através de redes IP. É utilizado no momento de envio de uma mensagem entre o MUA e um MTA, e entre MTAs durante o percurso, antes de chegar ao destinatário. É um protocolo orientado à ligação que utiliza a porta *Transmission Control Protocol* (TCP) número 25. As mensagens são trocadas em texto simples. O procedimento do SMTP pode ser dividido em quatro fases [51]: a iniciação de sessão, a iniciação do cliente, a transação da mensagem de *email*, e a terminação de sessão. Um dos problemas associados a este protocolo está relacionado com o facto de não possuir um mecanismo de autenticação, que o torna vulnerável ao *email spoofing*[2] [5].

O POP3 é um protocolo de aplicação utilizado pelo MUA para transferir as mensagens do servidor de *email* para o computador local. Neste protocolo as mensagens de *email* são descarregadas e, normalmente, apagadas do servidor de *email* [38]. Assim o protocolo é apenas durante o tempo necessário para descarregar as mensagens de *email*. O POP3 utiliza a porta TCP número 110. O procedimento do POP3 pode ser dividido em três estados. Após o estabelecimento da ligação TCP e do envio de uma resposta positiva por parte do servidor POP3 a sessão entra num estado de autorização [38], durante a qual o cliente se deve autenticar. Após autenticação a sessão entra num estado de transação. Neste estado o cliente pode efetuar pedidos sobre o servidor POP3 (e.g., descarregar mensagens, listar as mensagens). No momento em que o cliente envia o comando *QUIT* a sessão entra no estado de atualização. Neste estado o POP3 liberta os recursos alocados durante a fase de transação [38]. Posteriormente a ligação TCP é terminada.

O IMAP4, tal como o POP3, é um protocolo que permite o acesso do MUA ao servidor de *email* para obter as mensagens de correio eletrónico. Ao contrário do POP3, o cliente permanece ligado ao servidor IMAP4 enquanto a interface do MUA está ativa. Assim o MUA pode aceder e manipular as caixas de *email* (pastas de mensagens remotas) diretamente no servidor, de forma equivalente a manipular pastas locais [14]. O IMAP4 possibilita efetuar operações CRUD[3] sobre as caixas de *email*, verificar se existem mensagens novas, remover permanentemente mensagens,

[2]Falsificação do endereço de *email* do remetente de forma a dissimular a origem da mensagem.

[3]CRUD - *Create, read, update and delete.*

realizar procuras, e obter seletivamente atributos, textos, ou partes de mensagens de *email* [14]. Este protocolo utiliza a porta TCP número 143.

2.2.2 Estrutura da Mensagem de *Email*

A estrutura de uma mensagem de *email* é constituída pelo cabeçalho e pelo corpo da mensagem. O cabeçalho é estruturado em campos, definidos no RFC 2822, que identificam a origem, o destinatário, a data, o assunto e outras informações adicionais sobre a mensagem de *email*. O corpo consiste no texto da mensagem. A título de exemplo são apresentados alguns dos campos definidos para o cabeçalho de uma mensagem de *email* [48]:

- ***Date:*** - Especifica a data e hora à qual a mensagem foi enviada;
- ***From:*** - Define o endereço *email* do emissor da mensagem;
- ***Reply-To:*** - Indica os endereços *email*, sugeridos pelo emissor, para onde as respostas devem ser enviadas. Na ausência deste campo as respostas são enviadas por omissão para o endereço *email* especificado no campo *From:*;
- ***To:*** - Define uma lista de um ou mais endereços *email* dos destinatários da mensagem;
- ***Cc:*** - Contém o endereço *email* de outros destinatários que irão receber a mensagem, mas cujo conteúdo pode não ser dirigido a eles;
- ***Bcc:*** - Especifica os destinatários da mensagem cujos endereços de *email* serão ocultados para outros destinatários da mesma mensagem de *email*;
- ***Subject:*** - Indica o assunto da mensagem definido pelo emissor.

Além dos campos especificados, está previsto no RFC 2822 a possibilidade de serem definidos novos campos para o cabeçalho das mensagens de *email*. Estes campos opcionais têm de seguir a sintaxe especificada, sendo que os nomes dos campos não podem ser idênticos a outros já definidos.

2.3 Definição de *Spam*

A definição de *spam* não é universalmente consensual, e varia de acordo com as leis locais. Por exemplo nos Estados Unidos da América o *spam* é baseado

numa abordagem *opt-out*. Desta forma *emails* associados a campanhas de marketing podem ser enviados sem consentimento prévio do destinatário. Apesar de tudo é necessário que exista um mecanismo de interrupção do envio desse tipo mensagens (e.g., uma hiperligação no corpo da mensagem). Por outro lado na Europa e no Canadá existe uma política *opt-in* relativamente aos emails de carácter comercial. Assim um *email* desta natureza é apenas enviado, legitimamente, para os destinatários que tenham prévia e explicitamente expressado o seu consentimento.

Além das diferentes legislações, é possível ainda encontrar uma grande variedade de interpretações do significado da palavra. Desde definições mais concisas como "*email* não solicitado enviado em massa"[2]. Outras mais específicas que definem a prática como o "Envio de *emails* não solicitados, em grandes quantidades, e em alguns casos repetidamente, para indivíduos com os quais o remetente não teve contacto prévio, e com os quais o endereço de *email* foi recolhido indevidamente"[51]. Noutras definições é também comum encontrar a palavra "comercial", na perspectiva de que o *spam* é usado para publicitar produtos e serviços. Apesar de não existir uma única e consensual definição de *spam*, é aceite que as mensagens de *spam* possuem as seguintes características:

- A mensagem de *spam* é uma mensagem de correio eletrónico.
- É enviada sem consentimento do destinatário.
- O *spam* é enviado em grandes quantidades, para um grande número de destinatários escolhidos indiscriminadamente.

Existem ainda outras características baseadas no conteúdo que podem ser consideradas para definir uma mensagem de *spam*. Por exemplo o facto de um *email* possuir conteúdos ilegais e ofensivos, ou mesmo fraudulentos. Esta forma de caraterização do *spam*, é mais relativa pois depende da forma como o destinatário ajuíza as mensagens de correio eletrónico.

2.4 Recolha de Endereços de *Email*

Segundo as estatísticas do projeto [43] os países onde mais se pratica a colheita de endereços de *email* são a Espanha (16.1%), China (14.9%), e os Estados Unidos da América (11.1%). No topo da lista de países responsáveis pelo maior envio de

spam encontra-se novamente a China (10%), Brasil (9.0%) , e em terceiro lugar os Estados Unidos da América (7.2%). Curiosamente a Espanha, apesar de liderar a lista de países que mais endereços de *email* recolhem, surge fora do *top* 10 de países que mais *spam* enviam. Se compararmos estes números com as estatísticas obtidas em 2005 pelo mesmo projeto [44], onde os Estados Unidos da América lideravam tanto na recolha de endereços *email* (32.1%) como no envio de mensagens de *spam* (38.4%), podemos constatar que ambos os processos estão mais uniformemente distribuídos à escala global. Outro dado interessante, segundo o mesmo projeto, revela que o tempo decorrido entre a recolha de um endereço *email* e o envio da primeira mensagem de *spam* para o mesmo é em média de duas semanas, 5 dias, 11 horas e 49 minutos.

Conhecer as metodologias utilizadas pelos *spammers* para obter endereços de *email* é um aspecto importante para saber quais as políticas e comportamentos a adotar de forma a reduzir a eficácias destes métodos. Estes são alguns dos métodos mais referidos e discutidos para obter endereços de *email*:

- ***Web crawlers*** - Os *web crawlers* são programas que navegam na *Internet* de uma forma automática com o objetivo de recolher alguma informação. Neste caso, são programados para procurar endereços de *email*. Os endereços de *email* estão disponíveis em páginas web, sobretudo em fóruns de discussão, *guestbooks* e blogs [51]. Os *newsgroups* da Usenet são também alvo destes *robots*, que procuram identificar endereços de *email* nos cabeçalhos e no corpo das mensagens deixadas pelos utilizadores [47].

- ***Email chain letters*** - As *chain letters* consistem em *emails* de conteúdo fraudulento cujo objetivo é convencer de alguma forma as pessoas a reencaminharem as mensagens para um grande número de contactos [51]. O conteúdo destas mensagens é diverso, podem ser petições, alertas para pedidos de ajuda, ou promessas de enriquecimento fácil caso a mensagem seja reencaminhada. No decorrer do processo é constituída uma lista de endereços válidos de todos os que re-encaminharam a mensagem.

- ***Directory Harvest Attack* (DHA)** - O DHA é um tipo de ataque que recorre à força bruta para identificar endereços de *email* válidos num domínio. O SMTP antes de enviar uma mensagem de *email* verifica se o endereço é válido, através do envio de um pedido *delivery attempt*. A este pedido o MTA

responde afirmativamente caso o endereço seja válido, caso contrário o servidor que enviou o pedido *delivery attempt* recebe uma mensagem de erro 550 SMTP. Para explorar esta vulnerabilidade os *spammers* escolhem inicialmente os domínios que pretendem atacar, e enviam milhares de mensagens para endereços de *email* gerados automaticamente [21]. Para isso os atacantes utilizam duas técnicas diferentes. A primeira consiste em gerar endereços de *email* tentando todas as combinações de carateres válidos num espaço de 1 a N. Outra forma menos exaustiva, recorre à utilização de listas de palavras, ou dicionários de nomes de utilizadores frequentes [6].

São ainda referidos na literatura outros métodos que consistem em atacar diretamente os computadores pessoais, através de vírus e *worms* que exploram pastas em busca de listas de contactos de endereços de *email* [51].

2.5 Metodologias de *Spamming*

Após identificados os vários métodos utilizados na recolha de endereços de *email*, é necessário perceber as metodologias e técnicas que os *spammers* adotam para conseguirem o envio massivo de mensagens de *email* não solicitadas a uma escala global. Algumas destas metodologias são utilizadas para superar as técnicas de deteção de *spam*. De acordo com a pesquisa efetuada em [46] os métodos, ou meios, mais comuns utilizados na proliferação de *spam* são:

- ***Direct spamming***.
- ***Open mail relays***.
- ***Spamming botnets***.
- ***Border Gateway Protocol* (BGP) *spectrum agility*** .

O *direct spamming* consiste na compra de serviços de conectividade a *Internet Service Provider*s (ISPs) que não controlam a atividade, ou com políticas *anti-spam* pouco rígidas. Caso sejam tomadas medidas por um ISP relativamente a esta pratica, os *spammers* simplesmente recorrem a outro ISP. Porém esta mudança obriga o *spammer* a alterar os endereços IP dos *mail relays*. Segundo [46] de modo a contornar este trabalho, os *spammers* obtêm uma *pool* de endereços *dial-up* que

utilizam em ligações de banda larga no tráfego *upstream*, através de IP *spoofing*, para envio de *spam*. Desta forma é também mais difícil localizar a proveniência do tráfego relativo ao *spam*.

Open mail relays são servidores SMTP configurados de modo a permitir aos utilizadores enviarem mensagens de *email*, sem ser necessário qualquer tipo de autenticação. Os *spammers* aproveitam estes recursos para reencaminharem o tráfego das suas mensagens de *spam* [27]. O método consiste em enviar para um *open mail relay* a mensagem de *spam* juntamente com uma extensa lista de *Blind Carbon Copy* (BCC). Assim o *open mail relay* reencaminha a mensagem *spam* para todos os contactos da lista. Este método caiu em desuso a partir do momento em que se começaram a adotar *blacklists* como uma medida de mitigação do *spam* [46]. Isto porque as *blacklists* permitem identificar através do endereço IP, os *open mail relays* utilizados para enviar *spam* [32].

Existe um grande consenso relativamente ao facto das mensagens de *spam* serem diretamente enviadas por um grande conjunto de máquinas comprometidas (*spamming botnets*) que são controladas por *spammers* [58, 32, 55]. As máquinas comprometidas estão infetadas com um *worm*, que é responsável por enviar uma pequena quantidade de *spam*. O resultado é uma rede distribuída de envio de *spam* que se torna muito difícil de ser detetada, ou bloqueada [32]. No estudo efetuado em [58] foram identificadas 7.721 campanhas de *spam* baseadas em *botnets*, bem como 340.050 endereços IP únicos de *botnets*. Estes dados foram obtidos analisando uma amostra, correspondente a três meses, de mensagens de *email* da *hotmail*. No mesmo estudo é demonstrado que os cinco *Autonomous Systems* (ASs) com o maior número de endereços IP únicos de *botnets* são ASs que fornecem *Internet* a clientes residenciais.

O BGP *spectrum agility* é um mecanismo de despistagem utilizado pelos *spammers*, que dificulta a identificação dos *mails relays* utilizados para reencaminhar *spam*. Consiste numa utilização de endereços IPs pertencentes a um grande bloco por curtos espaços de tempo. Ou seja os grandes blocos de endereços IP, conferem aos *mail relays* uma flexibilidade na utilização de endereços IP. Desta forma torna-se mais difícil identificar e bloquear, através de *blacklists*, os *mail relays* que reencaminham *spam*. Outra vantagem recorrente deste método prende-se com o facto de ser menos provável que os ISPs filtrem anúncios de rotas de prefixos de IP

mais baixos, que correspondem a maiores blocos de endereços IP [46].

Estas metodologias acima descritas referem sobretudo os métodos logísticos utilizados pelos *spammers* e também alguns mecanismos ao nível da rede usados para contornar os métodos de filtragem *anti-spam* que se baseavam em *blacklists* (ver Secção 2.6.2). Assim as metodologias *anti-spam* foram evoluindo para responder a estes novos métodos. Esta evolução conduziu as técnicas *anti-spam* a adotaram uma abordagem baseada na análise do conteúdo das mensagens. Estas técnicas, apelidadas de *Content-Based Filtering* (CBF), partem do pressuposto que as mensagens de *spam* podem ser distinguidas das mensagens legítimas (*ham*) a partir da análise das palavras utilizadas na mensagens (ver Secção 2.6.3). De acordo com este pressuposto existem palavras que estão mais correlacionadas com as mensagens de *spam*(e.g., viagra, rolex, lottery), permitindo classificar as mensagens de *email* como *spam* ou *ham*. Por conseguinte as técnicas de camuflagem de *spam* também evoluíram, e os *spammers* encontraram formas de induzir em erro os classificadores. Para isso recorriam à modificação da forma como as palavras são escritas (e.g., vi@gra, r0lex) [8]. Outra forma utilizada para superar as técnicas CBF, consistia na substituição dos textos nas mensagens de *spam*, por imagens. Perante esta situação surgiram soluções de *Optical Character Recognition* (OCR).

2.6 Metodologias *Anti-Spam*

2.6.1 Medidas Legislativas

Com a crescente atividade de *spamming* e os seus potenciais riscos, as autoridades de várias nações decidiram tratar esta questão recorrendo à legislação. Em 2003 a União Europeia emitiu uma diretiva que tinha de ser incluída na legislação de todos os estados membros. Nesta diretiva é regulamentada a forma como se pode efetuar marketing direto utilizando o serviço de correio eletrónico, de forma a proteger os interesses dos cidadãos. É referido que a abordagem para envio de *emails* publicitários deve seguir uma abordagem *opt-in*, e que deve existir um mecanismo que permita o subscritor desses *emails* cancelar a sua assinatura [1].

Na Áustria, por exemplo, o envio de uma mensagem *email* para mais de 50 destinatários, com o intuito de marketing direto, viola a lei Austríaca para as telecomunicações, exceto se o destinatário tenha previamente aceitado o envio [51].

Na Alemanha a legislação vai ainda mais longe, o próprio código penal visa um conjunto de leis que podem ser potencialmente infringidas na prática do envio de *spam*. Delitos como sabotagem de computadores e distúrbio do funcionamento correto dos sistemas de telecomunicações, execução de anexos de mensagens de *email* mal intencionados, como vírus, *worms*, ou *trojan horses* estão previstos no código penal. Até mesmo o conteúdo de um *email* pode ir contra a lei, se por exemplo o conteúdo for pornográfico [51].

Apesar destes esforços, segundo os estudos efetuados em [39], apenas 31 países confirmam que possuem uma legislação *anti-spam*. E em grandes partes do mundo, como em África, América Latina e Médio Oriente, não existe informação disponível sobre a legislação relativa a esta problemática [51]. Além disso a legislação que existe é heterogénea a um nível global, o que torna difícil o acompanhamento do problema uma vez que as mensagens de *spam* atravessam várias fronteiras internacionais desde o local onde são originadas até ao destinatário [51]. Outro problema consiste na falta de recursos e conhecimentos em várias partes do mundo, para a implementação de leis *anti-spam*. Por estes motivos a eficácia das medidas legislativas no combate ao *spam* são controversas e muitas vezes postas em causa.

2.6.2 Técnicas Colaborativas

Blacklists

No contexto das medidas *anti-spam* as *blacklists* são listas de endereços IP que se encontram identificados como fontes de envio de *spam*. É uma técnica de pré-aceitação que os ESPs utilizam para impedirem qualquer tentativa de comunicação proveniente desses endereços e dos seus servidores de *email*. É considerada como uma primeira medida de defesa contra o envio de *spam*. Segundo [54, 17], a utilização de *blacklists* pode filtrar até 85% das conexões SMTP, evitando nestas situações que mecanismos de deteção de *spam* mais exigentes a nível de processamento (e.g., filtragem baseada no conteúdo) sejam utilizados. Alguns ESPs mantêm as suas próprias *blacklists* para uso privado, e não as disponibilizam a outras organizações. Porém existem empresas que disponibilizam serviços de consulta de *blacklists* baseados no *Domain Name System* (DNS), chamados de *Domain Name System based Blacklist* (DNSBL). O serviço é semelhante ao DNS, mas em vez de serem resolvidos endereços IP a partir de nomes de domínios, são verificados os endereços IP que estão associados à atividade de *spamming*.

Cada endereço IP listado na DNSBL possui um registo DNS correspondente. O nome do registo é criado revertendo a ordem dos octetos do endereço IPv4, à semelhança do que acontece no *reverse Domain Name System* (rDNS), e acrescentando o nome do domínio da DNSBL. Assim caso fosse criado um registo para o endereço IPv4 192.0.2.88 na DNSBL dnsbl.exemplo.com, o nome do registo DNS seria 88.2.0.192.dnsbl.exemplo.com. Cada registo existente na DNSBL possui *A record* associado. Convencionalmente o conteúdo do *A record* é valor 127.0.0.2 [31]. Assim quando um servidor SMTP recebe uma nova ligação é enviado um pedido de *A record* para uma DNSBL. No pedido consta o IP revertido do cliente mais o domínio da DNSBL(e.g., 88.2.0.192.dnsbl.exemplo.com). Caso a resposta seja um *A record* com o valor 127.0.0.2, significa que o registo existe e consequentemente que o endereço IP está identificado como uma fonte de envio de*spam*. Segundo [51]

as DNSBLs conseguem identificar:

- Blocos de endereço IP associados a redes com historial de envio de *spam*.
- Endereços IP de *hosts* específicos com historial de envio de *spam*, que se encontram tipicamente infetados com software hostil.
- Blocos de endereços IP atribuídos a *hosts* cujos utilizadores utilizam geralmente os servidores de *emails* dos seus ISPs, e não são esperados que enviem *emails* diretamente.

De acordo com o mesmo autor as *blacklists* possuem algumas desvantagens que advêm do facto de os *spammers* utilizarem várias técnicas que lhes permitem mudar de endereços IP regularmente. Em certos casos endereços IP pertencentes a ESPs, podem ser utilizados inapropriadamente por *spammers* por apenas umas horas, resultando posteriormente na perda de milhares de *emails* legítimos.

Whitelists

Da mesma forma que existem *blacklists* para identificar fontes de envio de *spam*, existem também as *whitelists* que identificam os remetentes legítimos. A gestão das *whitelists* pode ser feita pelo utilizador a nível local, ou pode consistir num serviço disponível publicamente análogo ao DNSBL. No primeiro caso o utilizador é o responsável pela gestão da *whitelist*, assim este deve adicionar e atualizar os endereços de *email* que considera legítimos à *whitelist*. As *whitelists* como serviço,

Domain Name System based Whitelist (DNSWL), são utilizadas por organizações para identificar os servidores SMTP, ou domínios que são de confiança. A admissão de organizações com fins comerciais nas DNSWLs tem um custo associado, e nem todas estas organizações são elegíveis para constarem nas DNSWLs [11]. Estes serviços são especialmente apropriados para empresas que utilizam servidores de *email* de transações, como sistemas de reserva de voos *online*, serviços de *e-commerce*, serviços bancários *online* [11], que não querem que as suas mensagens de *email*, sejam classificadas como *spam*.

Filtragem Baseada em Assinaturas

Outro método colaborativo de filtragem de *spam* consiste na partilha de assinaturas de mensagens de *spam*. Uma assinatura de uma mensagem de *email*, pode ser obtida, por exemplo, através de uma função de *hash*, que utiliza o conteúdo da mensagem para produzir um número identificador. Assim se um utilizador receber uma mensagem de *spam* e a classificar como tal, é gerada uma assinatura que é depois utilizada para prevenir que essa mensagem chegue à caixa de correio de outros utilizadores. Perante isto os *spammers* passaram a efetuar pequenas alterações ao conteúdo das mensagens de *spam* para serem produzidas assinaturas diferentes. Logo surgiram novos métodos para detetarem a similaridade entre estas mensagens ligeiramente alteradas. Por exemplo, em [15] foi proposto um sistema colaborativo de filtragem baseado na partilha de informação entre servidores de *email* através de uma rede *Peer to Peer* (P2P). A informação partilhada consiste em *message digests* de 256 bit das mensagens classificadas pelos utilizadores como *spam*. Caso o *message digest* de duas mensagens varie no máximo até 74 bits então essas mensagens são consideradas como iguais. Outro método proposto em [59] baseia-se no calculo de várias assinaturas para a mesma mensagem. Para isso o conteúdo de uma mensagem é dividido em *substrings* de tamanho L. Assim para um documento com n carateres são criadas (n-L+1) *substrings*. Posteriormente é calculado o valor de *checksum* para cada *substring*, e são selecionados depois os N valores de *checksum* mais elevados para constituírem a assinatura da mensagem.

2.6.3 Filtragem Baseada no Conteúdo

As técnicas de filtragem baseadas no conteúdo visam classificar as mensagens de *email* como *spam* ou *ham*, utilizando para isso caraterísticas do conteúdo das men-

sagens como as palavras, frases ou anexos, que são correlacionadas com a classe das mensagens, *ham* ou *spam*. Para estabelecer essa correlação entre as características das mensagens e a sua classe, é necessário algum conhecimento prévio, que pode ser obtido através da utilização de técnicas de *data mining*. O sucesso destes métodos na categorização de textos, foi o que incentivou os investigadores a aplicarem algoritmos de aprendizagem máquina à filtragem de *spam* [3]. Nesse contexto é utilizado um classificador (e.g., *Naive Bayes*, *Support Vector Machine* (SVM), *k-Nearest Neighbor* (k-NN)) que, sobre um conjunto de dados de mensagens de *email* previamente classificadas, cria um modelo de classificação. Esse modelo é utilizado posteriormente para classificar as novas mensagens de *email*. Assim de uma forma geral a filtragem baseada no conteúdo aplica a seguinte função de decisão [8]:

$$f(m,\theta) = \begin{cases} c_{spam}, & \text{se a mensagem é considerada como } spam \\ c_{ham}, & \text{se a mensagem é considerada como } ham \end{cases} \tag{2.1}$$

sendo m a mensagem a classificar, θ o vetor de parâmetros utilizados para a classificação, e c_{ham} ou c_{spam} a classe da mensagem.

No contexto das técnicas de classificação utilizando algoritmos de aprendizagem máquina o vector de parâmetros θ é o resultado do treino do classificador com um conjunto de mensagens prévias [8]:

$$\theta = \Theta(M) \tag{2.2}$$

$$M = (m_1, y_1), (m_1, y_1), ...(m_n, y_n), y_i \in c_{spam}, c_{ham} \tag{2.3}$$

onde $m_1, m_2, ..., m_n$ são mensagens prévias, $y_1, y_2, ..., y_i$ as classes correspondentes e Θ a função de aprendizagem [8].

De acordo com [19], apesar da grande popularidade destes métodos na filtragem *anti-spam*, existem vários desafios a superar na aplicação destes métodos em ambientes reais. Estes consistem na inconstante distribuição de classes(*spam* ou *ham*), a incerteza em relação aos custos de erro de classificação, e a reatividade e adaptação dos *spammers* a estes métodos.

O primeiro desafio advêm do facto da proporção de mensagens de *ham* e *spam* não serem constantes. A quantidade de *spam* recebida depende do endereço *email*, da forma como este é exposto e também do tempo que passou desde que o *email* foi publicado [19]. A quantidade de correio legítimo também varia de utilizador para utilizador. Estas diferenças na proporção entre mensagens *spam* e *ham* torna mais difícil a avaliação de desempenho de diferentes filtros. Assim pode existir um filtro que tenha um melhor desempenho que outro num conjunto de dados com 20% de mensagens de *spam*, mas que o mesmo não aconteça num conjunto de dados com uma percentagem de *spam* de 80% [19].

Os erros de classificação consistem nos casos em que uma mensagem de *spam* é classificada como *ham*, ou vice-versa. Os filtros *anti-spam* classificam uma mensagem como uma instância positiva, se for *spam*, e negativa quando é legítima. A classificação de uma mensagem legítima como *spam* (falso positivo), pode ser bastante mais grave do que classificar uma mensagem de *spam* como legítima (falso negativo) [19, 3]. Senão vejamos, um falso negativo corresponde a receber uma mensagem de *spam* na caixa de correio legítimo, que pode ser uma situação inconveniente. Por outro lado um falso positivo pode resultar num cenário onde a mensagem de *ham* é enviada para a pasta de *spam*, obrigando o utilizador a perder tempo à procura de mensagens legítimas na caixa de *spam*. Ainda mais grave é o cenário em que a mensagem legítima é eliminada sem possibilidade de ser recuperada. Devido a estas diferenças existem autores que defendem a utilização de custos assimétricos para cada um dos casos [19, 3]. Mas quantificar o custo de um falso positivo em relação a um falso negativo é algo muito relativo, e em última instância é um fator que deve ser o utilizador a especificar [19].

A reatividade e capacidade de adaptação dos *spammers* às técnicas de filtragem é outro desafio que estes métodos enfrentam na deteção de *spam* em ambiente real. Exemplo disso é a distorção do conteúdo textual nas mensagens de *spam*, que consiste na alteração da forma como as palavras são escritas (e.g., Viagra →

Vi@gra) [8], ou na introdução de *tags Hypertext Markup Language* (HTML) no meio de palavras [3]. Outra técnica utilizada pelos *spammers* são os ataques de *poisoning* que recorrem à introdução de palavras "neutras"nas mensagens de *spam*, para desta forma tornar as classes *spam* e *ham* indistinguíveis ao filtro *anti-spam* [22, 19].

Classificador *Naive Bayes*

Existem vários algoritmos de aprendizagem máquina que são utilizados para implementar soluções de filtragem *anti-spam* (e.g., *Naive Bayes*, SVM, k-NN, *Boosting*). Entre eles o *Naive Bayes* é particularmente o mais popular e amplamente utilizado [3, 8, 36]. Segundo [36] isto deve-se ao facto de o *Naive Bayes* ser um algoritmo simples que torna fácil a sua implementação, além disso este possui uma complexidade linear e o seu desempenho pode ser comparável a outros algoritmos de aprendizagem máquina mais elaborados. O *Naive Bayes* é um filtro estatístico que se baseia no teorema probabilístico de *Bayes* para efetuar a classificação das mensagens de *email*. Assim a partir de um vetor com as caraterísticas destas mensagens, e outras probabilidades calculadas sobre um conjunto de dados de treino, é possível categorizar uma mensagem de *email* [51]. O vetor que carateriza a mensagem pode conter atributos como o número de ocorrências de cada termo ou o valor *tf-idf*[4] para cada palavra presente numa mensagem. Apesar de ser não ser convencional em [12] foi proposto também a utilização métricas de rede como atributos para caraterizar as mensagens.

Segundo o teorema de Bayes a probabilidade de uma mensagem com o vector $\vec{x} =< x_i ... x_n >$ pertencer à categoria c é dada por [36, 50]:

$$p(c|\vec{x}) = \frac{p(c) \cdot p(\vec{x}|c)}{p(\vec{x})} \tag{2.4}$$

O classificador *Naive Bayes* vai classificar cada mensagem na categoria que maximizar $p(c) \cdot p(\vec{x}|c)$, pois o denominador não depende da categoria [36]. Assim,

[4]tf-idf *(term frequency-inverse document frequency)* é uma medida estatística usada para avaliar a relevância de uma palavra para um documento.

no contexto da filtragem de *spam*, isto corresponde a classificar uma mensagem como *spam* sempre que [36]:

$$\frac{p(c_s) \cdot p(\vec{x}|c_s)}{p(c_s) \cdot p(\vec{x}|c_s) + p(c_h) \cdot p(\vec{x}|c_h)} > T \tag{2.5}$$

com $T = 0.5$, c_s representado a classe *spam*, e c_h a classe *ham*. O valor de T pode ser parametrizado para adaptar o filtro consoante as necessidades. Para valores de $T > 0.5$ opta-se por um maior número de verdadeiros negativos (mensagens de *ham* corretamente classificadas), à custa de menos verdadeiros positivos (instâncias de *spam* corretamente classificadas) [36]. Verifica-se exatamente o contrário para valores de $T < 0.5$. A probabilidade $p(c)$ pode ser calculada dividindo o número de mensagens de treino da categoria c pelo número total de mensagens utilizadas para treino. As probabilidades $p(\vec{x}|c)$ são obtidas consoante a versão do *Naive Bayes* [36].

O algoritmo *Naive Bayes* possui diferentes variantes. No trabalho efetuado em [36] são discutidas cinco versões distintas deste algoritmo. É importante perceber estas diferenças de modo a ser possível efetuar uma avaliação de desempenho rigorosa nas soluções *anti-spam*. Entre as diferentes variantes discutidas em [36] encontram-se o *Multi-variate Bernoulli Naive Bayes* (MVBNB) (ver Eq. (2.6)), *Multinomial NB* com atributos de frequência dos termos (tf-*term frequency*) (ver Eq. (2.7)) e o *Multinomial NB* com atributos booleanos.

O MVBNB utiliza um vetor binário $\vec{x} =< x_i...x_n >$ para representar uma mensagem d. Sendo $A = \{t_1...t_n\}$ o conjunto de atributos utilizados para a classificação, cada x_i indica se o atributo t_i está, ou não, presente na mensagem [36]. Além disso cada mensagem d da classe c é obtida depois de m ensaios de *Bernoulli*, a cada ensaio é decidido se o atributo t_i vai constar na mensagem d. A probabilidade de o termo t_i constar na mensagem d é $p(t_i|c)$ [36]. Assim $p(\vec{x}|c)$ pode ser obtido da seguinte forma [36]:

$$p(\vec{x}|c) = \prod_{i=1}^{m} p(t_i|c)^{x_i} \cdot (1 - p(t_i|c))^{(1-x_i)} \tag{2.6}$$

A versão *Multinomial NB* com atributos de frequência de termos, representa uma mensagem d através de um vetor $\vec{x} =< x_i...x_n >$ onde cada elemento x_i indica o número de ocorrências de cada atributo t_i na mensagem [36]. Além disso cada mensagem d da classe c é o resultado de selecionar de forma independente $|d|$ atributos de A com reposição com uma probabilidade de $p(t_i|c)$ para cada t_i [36]. Assim $p(\vec{x}|c)$ é obtido da seguinte forma [36]:

$$p(\vec{x}|c) = p(|d|) \cdot |d|! \cdot \prod_{i=1}^{m} \frac{p(t_i|c)^{x_i}}{x_i!} \tag{2.7}$$

Por último, a versão *Multinomial NB* com atributos booleanos, é em tudo semelhante à versão anterior exceto que agora em vez de representar uma mensagem por um vetor em que os elementos indicavam o número de ocorrências dos atributos, este têm um valor booleano e apenas denotam se o atributo ocorre, ou não na mensagem d. Esta variante difere do MVBNB, uma vez que o modelo não considera diretamente a ausência de atributos da mensagem, não existindo o fator $(1 - p(t_i|c))^{(1-x_i)}$ [36].

2.7 Sumário

Este Capítulo centrou-se na contextualização do problema do *spam* no serviço de correio eletrónico. Inicialmente foi efetuada uma descrição breve do serviço de correio eletrónico, e dos protocolos envolvidos na entrega das mensagens de *email*. De seguida foram apresentadas diferentes definições de *spam* que por vezes não são consensuais e dependem da legislação de cada país. Posteriormente foram descritos meios utilizados na recolha de endereços de *email* nomeadamente *web crawlers*, *email chain letters* e o método DHA. Os endereços de *email* obtidos por estes métodos são utilizados para constituir listas de endereços *email* que representam os destinatários alvo dos *spammers*. Numa segunda fase foram explicados os métodos de envio massivo de mensagens de *email* não solicitadas a uma escala global. Esses métodos consistem na utilização do *direct spamming*, *open mail relays* e *botnets*. Por fim são explicadas as varias vertentes de metodologias *anti-spam* nomeadamente medidas legislativas e tecnológicas. No contexto das medidas tecnológicas são descritas técnicas de filtragem colaborativas, métodos baseados em assinaturas

e por último os métodos CBF. No contexto dos métodos CBF é descrito com pormenor as várias versões existente do classificador Bayesiano uma vez que este foi utilizado no desenvolvimento das soluções propostas neste projeto.

Capítulo 3

Algoritmos Genéticos e Evolucionários

Os AGEs tiveram um papel central no desenvolvimento dos filtros *anti-spam* propostos, nomeadamente na forma como os atributos mais relevantes são selecionados para constituir o modelo de classificação. Portanto neste Capítulo é efetuada uma contextualização da lógica e dos processos subjacentes aos algoritmos genéticos e evolucionários. É demonstrado também como estes algoritmos podem ser utilizados para resolver problemas relacionados com otimização, procura e aprendizagem.

3.1 Introdução

A Computação Genética e Evolucionária (CGE) constitui uma família de modelos computacionais inspirados na teoria *Darwiniana* da evolução das espécies. Estes modelos utilizam processos de seleção natural e de codificação genética aplicados na resolução de problemas de procura, otimização e aprendizagem, em diversas áreas do conhecimento [49]. Os Algoritmos Genéticos e Evolucionários (AGEs) são um dos pilares da CGE, e a sua estrutura foi inicialmente proposta por *John Holland* na década de 60 [37]. O trabalho de *John Holland* consistiu sobretudo na análise formal dos processos de adaptação que ocorrem na natureza e na introdução desses mecanismos nos sistemas de computação [37].

No meio ambiente as diversas necessidades dos indivíduos, e as restrições impostas pelo meio que os rodeia, força-os a competirem entre si para produzirem descendentes cada vez mais aptos. Os indivíduos mais capazes têm uma maior pro-

babilidade de criarem uma descendência mais numerosa, tornando assim os seus genes predominantes na população [49]. No contexto dos AGEs os indivíduos representam soluções possíveis no âmbito de um problema. Assim, tal como no meio ambiente, os melhores têm maior probabilidade de serem selecionados como progenitores para a sua informação ser recombinada de forma a serem criados novos indivíduos. É através de operadores de seleção, recombinação e mutação que é possível evoluir de uma população para uma nova geração.

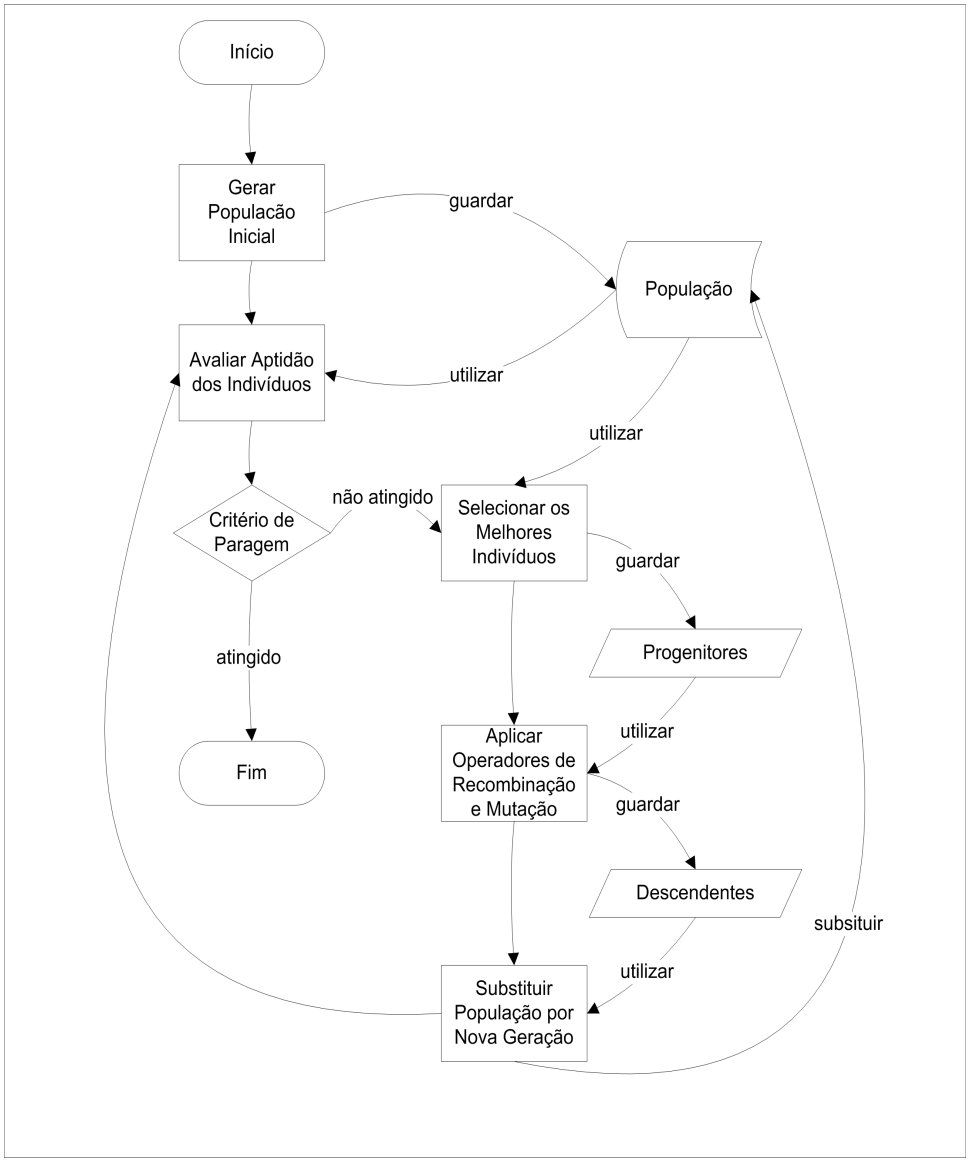


Figura 3.1: Fluxograma simples do algoritmo genético.

A estrutura simples de um AGE está representada na Figura 3.1. Inicialmente é gerada uma população inicial, tipicamente de forma aleatória. Posteriormente é atribuído um valor de aptidão a cada indivíduo da população, calculado através da função de avaliação definida. Os indivíduos mais aptos são selecionados (progenitores) e sujeitos a operadores de recombinação e mutação para assim gerarem novos indivíduos (descendentes). A nova geração de indivíduos vai substituir a população anterior e os valores de aptidão são novamente calculados. Este processo é cíclico e termina quando o valor de aptidão desejado é atingido, ou ao fim de um número definido de gerações.

Na resolução de um problema utilizando AGEs é necessário definir dois elementos fundamentais que dependem do problema em questão: a codificação das soluções e a função de avaliação. A codificação das soluções consiste na forma como as soluções são representadas no AGE. A função de avaliação define como é calculado o valor de aptidão para cada solução de acordo com a sua representação.

3.2 Codificação das Soluções

Segundo [37], a codificação das soluções é um fator central no sucesso de um AGE aplicado a problemas de procura ou aprendizagem. A representação de uma solução no contexto do AGE é também denominada de cromossoma cuja inspiração se deve à terminologia genética. Um cromossoma é formado por um conjunto de genes, e uma instância particular de um gene, ou seja o seu valor, é chamado de alelo. Os valores possíveis para os genes de um cromossoma dependem da representação que se utiliza para codificar as soluções:

- **Representação Binária** - O cromossoma é representado por uma sequência de genes, onde cada gene pode ter o valor 0 ou 1 (e.g., c=(1 0 0 0 1 0));
- **Representação Inteira** - O valor de cada gene do cromossoma pertence a um conjunto finito de valores inteiros, $\{1..N\}$ sendo N o limite superior definido para o problema. (e.g., c=(3 5 12 5 1));
- **Representação Real** - Na codificação real o valor de cada gene do cromossoma pertence a um conjunto finito de valores reais, com limites inferiores e superiores definidos para o problema. (e.g., c=(0.1 1.2 0.7 2.2));

- **Representação baseada em Permutações** - Neste tipo de representação a ordem dos genes no cromossoma é relevante. Além disso cada cromossoma é definido como uma permutação de todos os símbolos de um conjunto. Assim todos os indivíduos terão o mesmo comprimentos e este será igual à cardinalidade do conjunto. Por exemplo, para o conjunto $C = \{1..5\}$, os cromossomas têm de possuir todos os valores do conjunto sem repetições (e.g., c1=(2 4 1 3 5), c2=(1 5 4 3 2)).

- **Representação baseada em Conjuntos** - Este tipo de representação utiliza normalmente uma codificação inteira, a diferença está que o cromossoma de um individuo é um sub-conjunto de elementos de um dado conjunto, sendo que no mesmo cromossoma dois genes não podem ter o mesmo valor. Assim, para o conjunto $D = \{0..99\}$, os cromossomas c1=(1 5 23 76 98), c2=(0 5 23 26 43 99) são válidos, pois são sub-conjuntos de D. Convém também referir os tamanhos dos sub-conjuntos podem ser variáveis no mesmo AGE.

3.3 Função de Avaliação

A função de avaliação é responsável por atribuir a cada indivíduo um valor de aptidão. É com base neste valor que os melhores indivíduos são selecionados para criarem uma nova população, através de operadores de recombinação e mutação. Para avaliar um indivíduo é necessário primeiro descodificar a sua representação para uma solução que possa ser avaliada no contexto do problema. A função de avaliação pode ser uma função conhecida que se pretende maximizar ou minimizar, mas também pode estar implícita sem ser formalmente definida. De qualquer das formas o AGE realiza a otimização da função encarando-a como uma *caixa negra* onde são alimentados valores de entrada (cromossomas), e se obtêm valores de saída (aptidão dos cromossomas) [49].

3.4 Seleção

O operador de seleção determina quais os indivíduos da população que vão passar material genético para a próxima geração de indivíduos. A lógica é atribuir uma maior probabilidade de seleção, implícita ou explicita, aos cromossomas com melhor valor de aptidão. Alguns dos métodos utilizados para concretizar o processo de seleção são aqui descritos, nomeadamente:

- **Seleção por Torneio** - Na seleção por torneio são escolhidos k (tipicamente 2) indivíduos da população de forma aleatória segundo uma distribuição uniforme. Posteriormente os valores de aptidão são comparados, e o indivíduo vencedor é adicionado para a lista de progenitores. Neste método os valores absolutos de aptidão não desempenham qualquer papel. O que realmente importa é o valor relativo de aptidão entre os k indivíduos, ou seja se o valor de aptidão de um indivíduo é ou não superior ao(s) concorrente(s). Apesar de na seleção por torneio não estar explicita a atribuição de uma maior probabilidade de seleção aos melhores indivíduos implicitamente tal acontece. Analisemos o exemplo presente em [57] no qual se pretende selecionar por torneio n progenitores, onde n é igual ao tamanho da população, e $k = 2$. Assim cada indivíduo é em média selecionado duas vezes para participar no torneio. O indivíduo com maior valor de aptidão vai ganhar todas as competições em que participar, contribuindo assim com duas cópias para a lista de progenitores. No caso de um indivíduo mediano podemos considerar que este é melhor que 50% dos seus concorrentes, mas também perde o torneio em 50% dos casos. Assim, em média este indivíduo é aproximadamente adicionado uma vez à lista de progenitores. O pior indivíduo da população apenas pode passar para a lista de progenitores caso o seu concorrente seja ele próprio, o que acontece com uma probabilidade de apenas $\left(\frac{1}{n}\right)^2$.

- **Seleção por Classificação Linear** - Neste método de seleção os indivíduos são ordenados segundo o seu valor de aptidão. De seguida é atribuído um valor $i \in \{1, ..., N\}$ a cada indivíduo de acordo com a sua posição. O valor N é atribuído ao indivíduo mais apto, e o valor 1 ao pior. A probabilidade de seleção é depois atribuída linearmente de acordo com o valor i [9]:

$$p_i = \frac{1}{N} \cdot \left(\eta^- + (\eta^+ - \eta^-) \cdot \frac{i-1}{N-1}\right); i \in \{1, ..., N\} \tag{3.1}$$

Onde $\frac{\eta^-}{N}$ é a probabilidade do pior indivíduo ser selecionado, e $\frac{\eta^+}{N}$ é a probabilidade do melhor indivíduo ser selecionado [9]. As condições $\eta^+ = 2 - \eta^-$ e $\eta^+ \geq \eta^- \geq 0$ devem ser satisfeitas. Deve-se ainda referir que todos os indivíduos recebem um valor i diferente, mesmo que alguns possuam o mesmo

valor de aptidão.

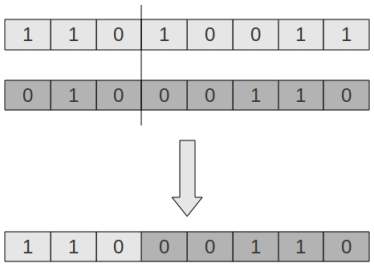
- **Seleção por Classificação Exponencial** - Este método de seleção é em tudo idêntico à seleção por classificação linear, diferenciando-se apenas na forma como as probabilidades de seleção são atribuídas aos indivíduos. Assim estas variam de uma forma exponencial de acordo com a posição do indivíduo, ao invés de uma variação linear. A base da exponencial é o parâmetro $0 < c < 1$, que permite controlar o nível de influência da posição do indivíduo na atribuição das probabilidades. Assim para valores de c próximos de 1 a exponencialidade da seleção é menor. Tal como no método linear, ao indivíduo mais apto é atribuído o valor N, e ao pior o valor 1. O cálculo das probabilidades é agora definido da seguinte forma [9]:

$$p_i = \frac{c-1}{c^N - 1} \cdot c^{N-i}; i \in \{1, ..., N\} \tag{3.2}$$

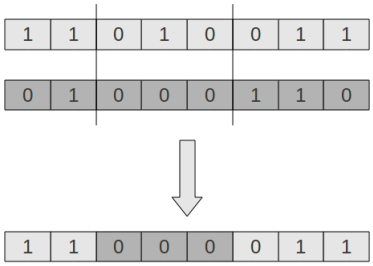
3.5 Recombinação

Após a seleção dos cromossomas progenitores, é necessário recombinar o seu material genético, à semelhança do que acontece na reprodução dos seres vivos, de modo a produzir novos indivíduos. Pretende-se desta forma preservar a informação genética dos indivíduos mais aptos para as próximas gerações. Nos AGEs os operadores que permitem esta recombinação são chamados operadores de cruzamento ou *crossover*.

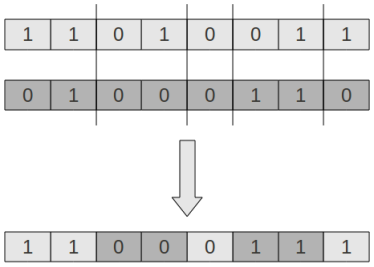
Através do *Single Point Crossover* (SPX) os cromossomas de ambos os progenitores são divididos num único ponto de cruzamento determinado aleatoriamente. O cromossoma do descendente é obtido anexando a primeira parte do primeiro progenitor à segunda parte do segundo progenitor, conforme está na Figura 3.2(a). No *Two Point Crossover* (TPX), a divisão dos cromossomas progenitores é efetuada em dois pontos de corte (ver Figura 3.2(b)). Assim o cromossoma do descendente é criado anexando a primeira e a terceira parte do primeiro progenitor com a segunda parte do segundo progenitor. A operação de cruzamento pode ainda ser efetuada em múltiplos pontos dos cromossomas progenitores *Multiple Point Crossover* (MPX) (ver Figura 3.2(c)).



(a) Cruzamento de Ponto Único (SPX).

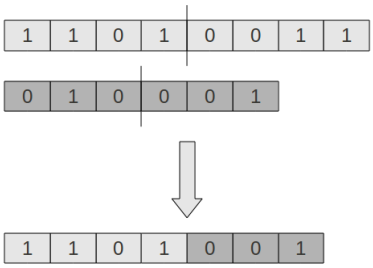


(b) Cruzamento de dois Pontos (TPX).

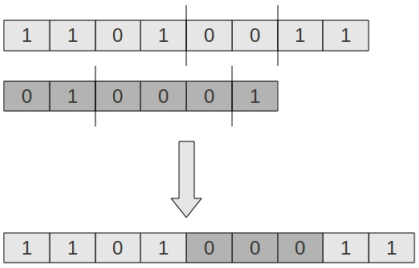


(c) Cruzamento de Múltiplos Pontos (MPX).

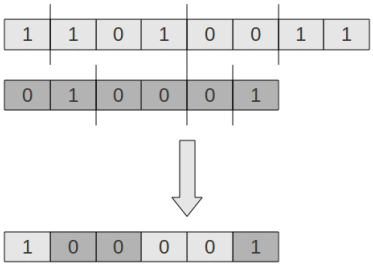
Figura 3.2: Operadores de cruzamento em cromossomas de tamanho igual.



(a) Cruzamento de Ponto Único.



(b) Cruzamento de dois Pontos.



(c) Cruzamento de Múltiplos Pontos.

Figura 3.3: Operadores de cruzamento em cromossomas de tamanhos variável.

Para cromossomas com tamanhos variáveis dentro da mesma população, o processo de cruzamento é idêntico. A única diferença está na determinação do ponto de cruzamento que é aleatoriamente determinado para cada progenitor. Como re-

sultado os pontos de cruzamento são, na maioria das vezes, diferentes para cada progenitor, como se mostra nas Figuras 3.3(a), 3.3(b) e 3.3(c).

3.6 Mutação

A mutação é um método importante para preservar a diversidade numa população de indivíduos, e consequentemente para explorar novos espaços de procura no âmbito do problema alvo [49, 57]. Imaginemos que numa dada população o valor de um dado gene de todos os indivíduos tem o mesmo valor, seria impossível através de operadores de cruzamento alterar esse valor. Assim através de um operador de mutação existe a possibilidade de um novo alelo ser recuperado [49]. A mutação é concretizada através da alteração aleatória do valor de um gene (alelo) no cromossoma (ver Figura 3.4). Para cromossomas com representação binária, essa alteração consiste em aplicar o complemento ao valor de um gene, assim se este for 1 passa para 0 ou vice versa.

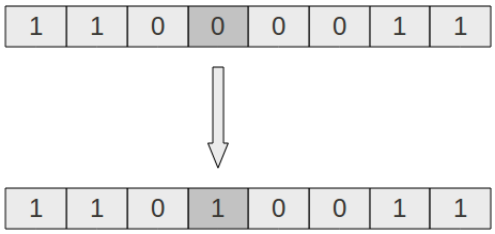


Figura 3.4: Mutação binária.

Quando estamos perante cromossomas com representação inteira existem dois tipos de mutação que podem ser aplicados. O primeiro consiste em alterar um dado gene por um valor aleatório selecionado entre os valores $\{1..N\}$. Na segunda abordagem o valor do gene é substituído pelo valor seguinte no intervalo, caso o valor a substituir no gene seja N este é substituído pelo primeiro [49].

Nas representações reais os operadores de mutação aplicam uma pequena perturbação no valor do gene. Esta perturbação traduz-se na adição ou subtração de valores aleatórios provenientes de uma distribuição estatística. Segundo [49], esta distribuição deve ser influenciada pela evolução do próprio algoritmo, para que a probabilidade de gerar estas pertubações diminua à medida que o AGE se aproxima do ponto de convergência.

3.7 Sumário

No decorrer deste Capítulo foi explicada a lógica subjacente aos AGEs, e de que forma estes podem ser utilizados para resolver problemas de otimização, procura e aprendizagem. Inicialmente foi explicada a origem dos AGEs, inspirada nos processos de adaptação que ocorrem na natureza. Nesse sentido foi descrita a estrutura e funcionamento dos AGEs referindo os vários componente que o constituem. De forma mais pormenorizada são descritos os operados de seleção, recombinação, mutação, e as várias representações que podem ser utilizadas na codificação das soluções de um problema.

Capítulo 4

Soluções Desenvolvidas

Neste Capítulo são inicialmente descritas na Secção 4.2 duas bibliotecas Java, nomeadamente a JECoLi e a RapidMiner 4.6, que tiveram um papel central no desenvolvimento dos filtros *anti-spam* deste projeto. Posteriormente na Secção 4.3 é explicado o classificador simples que não integra técnicas de computação evolucionária de procura de atributos. Este protótipo foi desenvolvido com o intuído de comparar os resultados de classificação obtidos com a abordagem que utiliza técnicas de computação evolucionária e partilha de atributos. De seguida na Secção 4.5 é apresentado o método utilizado para integrar as técnicas de computação evolucionária na procura de atributos mais relevantes . São apresentadas dois classificadores, que correspondem a filtros locais, que utilizam estas técnicas com abordagens diferentes. Por último é apresentado o classificador que além das da utilização das técnicas de computação evolucionária permite a partilha de atributos entre filtros locais. Neste Capítulo utiliza-se a designação de classificador para filtro *anti-spam*, uma vez que um filtro *anti-spam* não é mais que um classificador de mensagens de email com dois atributos objetivo, *spam* e *ham*.

4.1 Introdução

Neste trabalho foram desenvolvidos no total quatro classificadores. Todos os classificadores utilizam um algoritmo de aprendizagem máquina (e.g., *Naive Bayes*, SVM[1], árvores de decisão), que pode ser escolhido na parametrização. A diferença entre os classificadores está no método como a seleção dos atributos dos conjuntos

[1]SVM - *Support Vector Machine*.

de dados, que representam as mensagens de email, é efetuada nas amostras de treino para criar o modelo de classificação.

O primeiro classificador desenvolvido, apelidado de classificador simples, utiliza apenas um critério de seleção, o IG[2], para selecionar os atributos mais relevantes para classificação. Os m atributos mais relevantes de acordo com o IG são escolhidos para criar o modelo de classificação. Os restantes protótipos desenvolvidos utilizam técnicas de computação evolucionária, nomeadamente algoritmos genéticos, para resolver um problema de procura. O problem consiste em encontrar uma combinação de atributos que maximize o valor de AUC[3] obtido na classificação de mensagens de *email*. O domínio de procura do AGE limita-se aos atributos previamente selecionados pelo IG, e não a todos os atributos do conjunto de dados. Pretende-se assim reduzir o domínio de procura do AGE a atributos com alguma relevância. O método de procura utilizado é do tipo *wrapper* ou seja no contexto do problema um indivíduo do AGE é um sub-conjunto de dados do problema, e a função de avaliação consiste na função de classificação de novas mensagens de *email* em que o modelo de classificação é treinado com o sub-conjunto de dados representado pelo indivíduo.

As diferenças entre as três versões de classificadores que utilizam técnicas de computação evolucionária está relacionada com o método, e o domínio da procura por um sub-conjunto de atributos que melhore a assertividade do classificador. Para melhor perceber estas diferenças é necessário primeiro referir de que forma são avaliadas as estratégias de classificação das mensagens de *email*. Para classificar as mensagens de *email* tem de existir um sub-conjunto de mensagens utilizadas para criar um modelo de classificação (sub-conjunto de treino), e outro sub-conjunto utilizado para avaliar o modelo de classificação (sub-conjunto de teste). Para isso um conjunto de dados que representa na totalidade uma caixa de *email* de um utilizador tem de ser dividido. De forma a proceder a uma avaliação mais realista do desempenho da classificação foi adotado um método de treino incremental. Assim uma caixa de *email* é dividida em lotes $l_1 ... l_n$ com K mensagens contíguas ($|l_n|$ pode ser inferior a K), o filtro é treinado com $l_1 \cup ... \cup l_i$ e a avaliação é feita

[2]Critério de seleção de atributos muito utilizado no contexto da filtragem *anti-spam* (ver Secção 5.2.2).

[3]Métrica de avaliação do desempenho de um classificador (ver Secção 5.4).

com l_{i+1}. No primeiro protótipo, o algoritmo evolucionário é inicializado a cada lote de mensagens, desta forma a população inicial de indivíduos é aleatoriamente criada. Na segunda abordagem a população final de um lote é alimentada ao lote seguinte. Desta forma a população inicial de um lote possui informação relevante do passado dos lotes anteriores. Na terceira abordagem a população inicial de um lote pode receber indivíduos de lotes anteriores como também indivíduos de outros utilizadores. Existe assim uma partilha de atributos entre diferentes utilizadores. Na Secção 4.5 são explicadas de forma mais pormenorizada as diferenças entre estas abordagens.

4.2 Tecnologias de Desenvolvimento

Para desenvolver os protótipos de classificação de mensagens de *email* foi utilizada a linguagem Java. As bibliotecas RapidMiner 4.6 e JECoLi tiveram um papel central neste desenvolvimento. A biblioteca RapidMiner 4.6 disponibiliza um conjunto de operadores úteis para a implementação de processos de *data mining* e aprendizagem máquina. Estes operadores implementam métodos de pré-processamento e transformação de dados, métodos de cálculo da relevância de atributos e de seleção, algoritmos de aprendizagem máquina, funções para avaliação dos resultados de classificação ou procura, resumindo operadores muito úteis para a área da extração de conhecimento (ver Secção 4.2.1). A biblioteca JECoLi implementa algoritmos de otimização meta-heurísticos, com um destaque para os que utilizam métodos de computação evolucionária (ver Secção 4.2.2).

4.2.1 RapidMiner 4.6

O RapidMiner, anteriormente conhecido como *Yet Another Learning Environment* (YALE), é uma *framework open-source* desenvolvida em Java utilizada para processos de *data mining* e aprendizagem máquina. O projeto de desenvolvimento desta ferramenta começou em 2001 no Departamento de Ciências da Computação da Universidade de Dortmund. Desde então já foram registados mais de meio milhão de *downloads* deste software [25]. Com o amadurecimento deste projeto foram surgindo empresas interessadas neste produto e nos serviços e conhecimentos dos seus criadores. De forma a responder a estas necessidades foi criada a empresa Rapid-I que é atualmente responsável pelo desenvolvimento e suporte desta ferramenta [25].

O desenvolvimento de um processo de análise em RapidMiner 4.6 é bastante flexível e segue uma lógica de operadores modulares que são adicionados a uma árvore que vai representar o processo. A ordem de execução do processo é sequencial. Sendo que para um processo estar definido corretamente deve existir concordância entre os dados de saída e de entrada de dois operadores contíguos. Estes dados podem ser conjunto de dados, modelos de classificação, valores de relevância de atributos e vetores de desempenho.

Existem mais de 400 operadores disponíveis para realizar variadíssimas tarefas úteis para a análise de dados, (e.g., pré-processamento e transformação de dados, seleção de atributos, criação de modelos de aprendizagem, visualização de resultados, entre outros). Na Figura 4.1 está exemplificado um processo de classificação definido no RapidMiner 4.6. Este processo pode ser dividido em quatro fases:

- **Leitura do conjunto de dados** - Neste caso em formato esparso, desempenhado pelo operador de *input SparseFormatExampleSource*.

- **Seleção de atributos** - Esta fase é composta por dois operadores. Primeiro através do operador *InfoGainWeighting* é calculado o valor de IG para todos os atributos do conjunto de dados. De seguida, são selecionados um sub-conjunto de atributos através do operador *AttributeWeightSelection* em função do valor de IG calculado para cada atributo.

- **Construção do modelo de classificação** - Para obter o modelo de classificação é necessário um conjunto de dados de treino e um algoritmo de aprendizagem. O conjunto de dados de treino é obtido através do operador *SimpleValidation*. Este operador divide o conjunto de dados obtido da fase de seleção em dois sub-conjuntos um de treino e outro de teste. O algoritmo de aprendizagem, neste caso o *Multinominal NB*, é implementado pelo operador *W-NaiveBayesMultinomial*.

- **Avaliação do modelo** - O modelo de classificação obtido através do operador *W-NaiveBayesMultinomial* é aplicado às amostras de teste pelo operador *ModelApplier*. A avaliação do desempenho do modelo é feita pelo operador *BinominalClassificationPerformance*, que disponibiliza um conjunto de métricas utilizadas para esse efeito (e.g., AUC, *precision*, *recall*, etc).

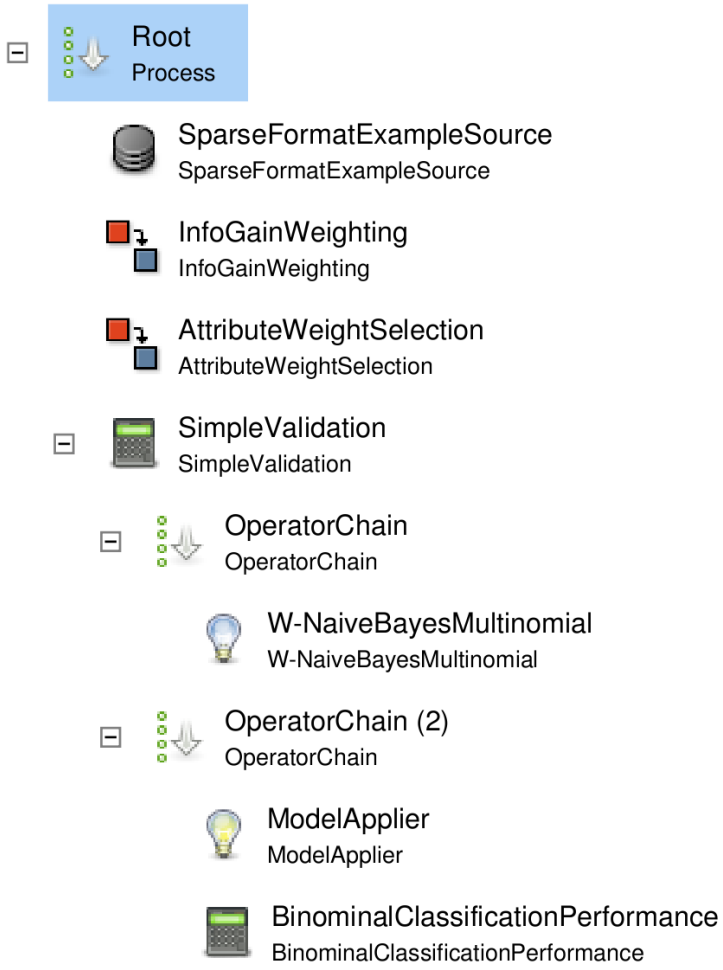


Figura 4.1: Processo de classificação no RapidMiner 4.6.

Integração do RapidMiner numa Aplicação Java

Uma vantagem do RapidMiner está relacionada com a sua filosofia *open-source* que possibilita a sua integração numa aplicação Java. Desta forma torna-se possível reutilizar implementações desta *framework* para desenvolver aplicações com contextos mais abrangentes. Além disso é promovido o desenvolvimento de extensões que podem ser posteriormente integradas nesta ferramenta.

Antes de utilizar RapidMiner numa aplicação este deve ser inicializado. Para isso tem de ser invocado o método *RapidMiner.init()*. Esta inicialização pode ser configurada através de várias opções, que são descritas em [24]. Após a inicialização podem ser criados operadores através do método *createOperator(...)* da

```
import com.rapidminer.tools.OperatorService;
import com.rapidminer.RapidMiner;
import com.rapidminer.Process;
import com.rapidminer.operator.Operator;
import com.rapidminer.operator.OperatorException;
import java.io.IOException;

public class ProcessCreator {

    public static Process createProcess() {
        try {
        // invoke init before using the OperatorService
            RapidMiner.init();
        } catch (IOException e) { e.printStackTrace(); }

        // create process
        Process process = new Process();
        try {
            // create operator
            Operator inputOperator =
                OperatorService.createOperator(ExampleSetGenerator.class);

            // set parameters
            inputOperator.setParameter("target_function", "sum␣classification");

            // add operator to process
            process.getRootOperator().addOperator(inputOperator);

            // add other operators and set parameters
            // [...]
        } catch (Exception e) { e.printStackTrace(); }
        return process;
    }

    public static void main(String[] argv) {
        // create process
        Process process = createProcess();
        // print process setup
        System.out.println(process.getRootOperator().createProcessTree(0));

        try {
            // perform process
            process.run();
            // to run the process with input created by your application use
        // process.run(new IOContainer(new IOObject[] { ... your objects ... });
        } catch (OperatorException e) { e.printStackTrace(); }
    }
}
```

Figura 4.2: Implementação em Java de um processo RapidMiner 4.6.

classe *com.rapidminer.tools.OperatorService*. Os operadores são adicionados a uma instância da classe *com.rapidminer.Process*, através do método *getRootOperator() .add(Operator op)*. Desta forma, a lógica é em tudo semelhante à forma como um processo é definido na interface gráfica do RapidMiner. Na Figura 4.2[4] é exemplificada a definição de um processo em RapidMiner implementado em Java (ver Anexo A para um exemplo da implementação do processo da Figura 4.1).

4.2.2 JECoLi

A JECoLi é uma *framework open-source* desenvolvida em Java que implementa algoritmos de otimização meta-heurísticos, com especial ênfase para abordagens baseadas em CGE [18]. Pretende ser uma plataforma de *software*, flexível, modular, adaptável e extensível que permita realizar dois tipos de tarefas [18]: (i) desenvolver componentes para outros sistemas com recurso aos algoritmos de otimização implementados; (ii) permitir uma análise e avaliação rápida de diferentes abordagens em tarefas específicas de otimização. Esta ferramenta disponibiliza um variado número de meta-heurísticas (e.g., algoritmos genéticos, *simulated annealing*, algoritmos evolucionários multi-objetivo, entre outros), bem como várias tipos de representações para codificar as soluções, nomeadamente: binárias, inteiras, reais, permutações, conjuntos e árvores.

Os passos básicos necessários para criar um algoritmo evolucionário para um dado problema na JECoLi são:

- **Definir a representação das soluções para o problema** - Um indivíduo é representado pela classe que implementa a interface *ISolution*. A classe que representa a solução deve retornar o cromossoma do indivíduo, e a sua aptidão. Para cada tipo de representação existem duas classes necessárias: a classe que implementa a interface *IRepresentation* cuja função é representar o cromossoma, e a classe responsável por gerar novas soluções que implementa a interface *ISolutionFactory*.

- **Definir a função de avaliação das soluções** - A função de avaliação define a forma como o valor de aptidão é calculado num problema específico. É

[4]Figura obtida em [24].

através desta função que efetivamente se integra a JECoLi com o problema. Para isso é necessário:

  - Implementar a classe Java que herda da classe *EvaluationFunction<R>*, em que *R* é a classe da representação utilizada.
  - Esta classe tem de implementar o método public *List<Double> evaluate(R solution)*. Este método implementa a função objetivo especifica do problema e retorna uma lista com o valor numérico de aptidão da solução. Dependentemente do problema pode ser necessário descodificar a solução antes de se calcular o valor de aptidão.

- **Configurar o algoritmo evolucionário** - A configuração de um algoritmo evolucionário na JECoLi é mantida numa instância da classe *Evolutionary-Configuration*. Existem diversos parâmetros que devem ser definidos antes de executar o algoritmo, nomeadamente:

  - A função de avaliação;
  - A *solution factory* a utilizar para gerar indivíduos;
  - O conjunto de operadores de reprodução;
  - O tamanho da população;
  - O método de seleção;
  - Os parâmetros de recombinação;
  - O critério de paragem do algoritmo;

No Anexo B está presente um exemplo de configuração de um algoritmo evolucionário na JECoLi. No exemplo em questão é utilizada uma representação de conjunto de *strings*. Esta representação foi desenvolvida especificamente para ser utilizada neste projeto (ver Secção 4.4).

4.3 Classificador Simples

O protótipo de classificação simples não integra técnicas de computação evolucionária no método de procura dos atributos mais relevantes. Foi desenvolvido com o intuito de ser versátil no formato de conjunto dados que suporta, no método de seleção de atributos e nos algoritmos de aprendizagem que possibilita utilizar.

Além disso, também é possível escolher diversas métricas de avaliação do desempenho da classificação. Na Figura 4.3 é apresentado o diagrama de classes do protótipo de classificação simples. Na classe *Dataset* é carregado o conjunto de dados, que pode estar em diversos formatos (e.g., *Comma-Sepparated Values* (CSV), *Attribute-Relation File Format* (ARFF), C4.5, modo esparso). Após esta operação o conjunto de dados é representado pelo objeto *ExampleSet* da biblioteca rapidminer. A seleção de atributos é feita na classe *AttributeSelector*. Para isso basta definir qual o método de seleção e o número de atributos a selecionar. O método de seleção pode ser qualquer um implementado na biblioteca rapidminer (e.g., IG, coeficiente de Gini, Chi-Quadrado, entre outros). O processo de treino do classificador e a classificação do conjunto de dados de teste é desempenhada na classe *Classifier*. Nesta classe pode ser definida a percentagem do conjunto de dados que é utilizada para treino, sendo que a restante é utilizada para classificação. O protótipo também suporta treino iterativo cumulativo, como explicado na Secção anterior. Para construir o modelo de classificação é possível optar por diversos algoritmos de aprendizagem máquina (e.g., *Naive Bayes*, SVM, àrvores de decisão), ou seja todos os algoritmos suportados na biblioteca rapidminer.

Para facilitar a utilização do classificador foi criado uma *shell script* para *bash*, que funciona como um interface entre o protótipo e o utilizador. Nesta *script* existem quatro argumentos obrigatórios para a classificação nomeadamente: o caminho para o ficheiro que contém o conjunto de dados, o formato dos dados, o algoritmo de aprendizagem máquina e o nome da pasta onde são guardados os resultados obtidos. Por exemplo:

```
$ classifier -l Y DataSet csv NaiveBayes Results
```

Note-se que neste exemplo foi também necessário especificar qual o atributo especial que identifica a classe de cada exemplo. Para isso utilizou-se a opção $-l$ Y, sendo Y o nome do atributo que identifica a classe. Existem várias opções que podem ser adicionadas à classificação consoantes as necessidades do problema:

```
-a --AUC                    Area Under Curve ( AUC );
-c --ColumnSeparator        Delimitador entre colunas;
-k --NSelectedAttributes    Número de atributos que se pretende seleccionar;
-l --LabelName              Nome do atributo que indica a classe;
-p --Precision              Precisão (métrica);
-r --SplitRatio             Percentagem de treino;
```

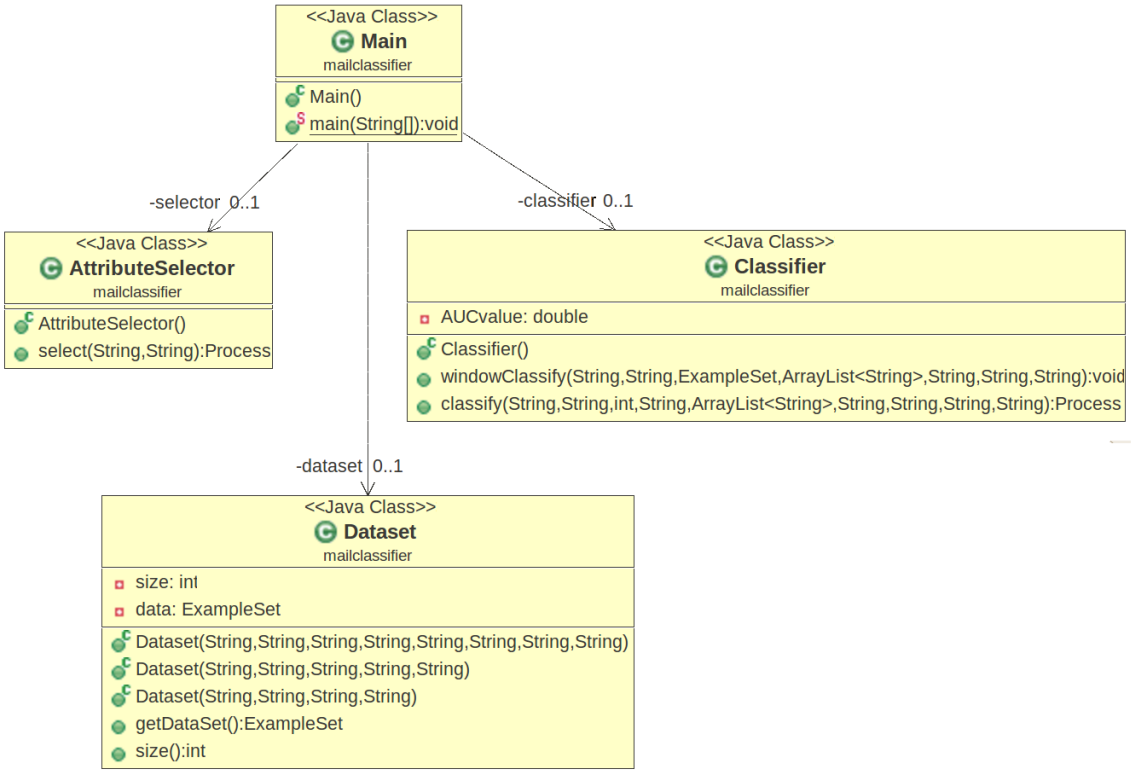


Figura 4.3: Diagrama de classes do classificador simples.

```
-s --AttributeWeighting    Critério de ponderação dos atributos;
-t --TVP@TFP               Métrica TVP@TFP;
-u --UseQuotes             Indica se os valores do data set estão entre aspas;
-w --WindowSize            Tamanho da janela para treino iterativo;
-x --Recall                Métrica recall.
```

Para efetuar uma classificação incremental cumulativa basta adicionar a opção $-w$ e especificar o tamanho da janela de treino/classificação. No treino incremental cumulativo uma caixa de *email* é dividida em lotes $l_1 ... l_n$ com K mensagens contíguas. O filtro é treinado com $l_1 \cup ... \cup l_i$ e a avaliação é feita com l_{i+1}.

```
$ classifier -w 100 -l Y DataSet csv LibSVMLeaner Result
```

No próximo exemplo é utilizada opção $-s$ para calcular o IG de cada atributo. São selecionados os 1000 atributos mais relevantes (opção $-k$). No caso da classificação incremental cumulativa, quando é selecionada a opção de seleção de atributos, esta é efetuada sobre $l_1 \cup ... \cup l_i$ a cada iteração.

```
$ classifier -s InfoGain -k 1000 SparseSet xy NaiveBayes Result
```

Na próxima Secção é descrita a nova representação de indivíduos de um AGE desenvolvida especificamente para este projeto. Posteriormente é explicada a forma como as técnicas de computação evolucionária de seleção de atributos foram integradas no classificador.

4.4 Representação Conjunto de *Strings*

De forma a facilitar a implementação dos protótipos de classificação que integram técnicas de computação evolucionária foi decidido desenvolver uma nova representação especificamente para este projeto. Nesta nova representação os indivíduos de um AGE são codificados num conjunto de *strings*. No contexto do problema de classificação, este conjunto de *strings* indica quais os atributos que devem ser utilizados para o processo de treino de um modelo e de classificação de mensagens de *email*.

Inicialmente optou-se por utilizar uma representação binária para codificar os indivíduos no AGE (ver Secção 3.2). Nesta representação um indivíduo é representado por uma sequência binária que codifica o sub-conjunto de dados. Por sua vez

o sub-conjunto de dados é representado por uma matriz onde as linhas representam as mensagens de *email* e as colunas as palavras que as caraterizaram. O atributo especial Y representa a classe a que cada mensagem pertence, spam (S) ou ham (H), A sequência de um indivíduo indica quais os atributos que se encontram presentes no sub-conjunto de dados que representa. Assim se o índice x da sequência tiver o valor 1, indica que o atributo com índice x no conjunto de dados faz parte do sub-conjunto representado pelo indivíduo. Se o valor for 0 significa que esse atributo não faz parte do sub-conjunto (ver Figura 4.4).

Conjunto de dados:

can	have	http	pills	the	their	they	Y
0	3	0	2	1	0	0	H
1	0	2	0	2	3	0	S
2	1	0	0	5	0	4	S

Indivíduo A:

1	1	0	1	0	0	1

Sub-conjunto de dados representado por A:

can	have
0	3
1	0
2	1

pills
2
0
0

they	Y
0	H
0	S
4	S

Figura 4.4: Representação de indivíduo com codificação binária.

A utilização desta representação tem duas desvantagens. Primeiro os indivíduos possuem todos o mesmo número de genes, isto não permite definir um número máximo e mínimo de atributos a utilizar por cada indivíduo. Por outro lado quando se pretende uma partilha de atributos entre utilizadores, através da troca de indivíduos, é necessário um processo de mapeamento dos indivíduos trocados. Este processo é necessário porque a sequência binária codifica um sub-conjunto de dados

baseando-se na posição dos atributos no conjunto de dados. Como os utilizadores possuem conjuntos de dados diferentes, o mesmo indivíduo com a mesma representação, codifica um sub-conjunto de dados diferentes para cada utilizador.

A utilização de uma representação de conjunto de *strings* resolve os dois problemas anteriormente enunciados. Primeiro esta representação permite definir os limites máximos e mínimos do tamanho dos indivíduos. Assim é possível definir um intervalo do número de atributos a selecionar pelo AGE. A outra vantagem está relacionada com o facto de a troca de indivíduos entre dois utilizadores poder ser concretizada de forma direta, sem a necessidade de um mapeamento, uma vez que nos cromossomas dos indivíduos está explicitamente representado o nome dos atributos. Na Figura 4.5 é demonstrado o cromossoma de dois indivíduos utilizando a representação binária e a representação por conjunto de *strings* equivalente.

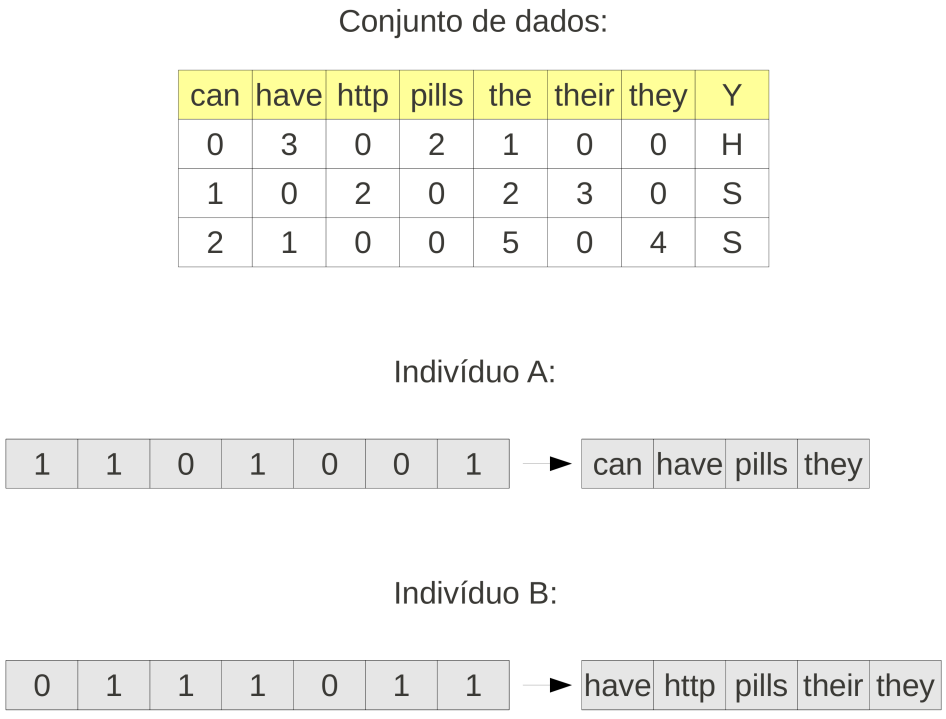


Figura 4.5: Exemplo de representação binária e representação por conjunto de *strings*.

Quando os utilizadores partilham indivíduos estão na realidade a partilhar atributos que são considerados relevantes. Estes atributos são considerados relevantes porque fazem parte dos cromossomas dos indivíduos mais aptos, que são os que pertencem à população final do AGE na fase de procura (ver Secção 4.5). Convém referir que informação partilhada entre utilizadores é constituída somente pelos nomes dos atributos, e não pelas frequências dos atributos em cada mensagem, ou pela correlação dos mesmos com a classe S ou H. A informação relativa a frequência dos atributos numa mensagem é depois obtida a nível local pelo utilizador que recebe o atributo consoante o seu conjunto de dados. Assim esta partilha pode ser considerada como uma forma dos utilizadores indicarem a outros quais os atributos que devem considerar na fase de procura dos atributos mais relevantes.

Como explicado na Secção 4.2.2 para cada tipo de representação presente na JECoLi existem duas classes necessárias. A classe cuja função é representar o cromossoma de uma solução, e a classe que é responsável por gerar novas soluções. Assim para esta nova representação foram implementadas as classes *StringSetRepresentation*, e *StringSetRepresentationFactory*. Para instanciar um objeto da classe *StringSetRepresentation* é necessário definir os seguintes parâmetros:

- O domínio das soluções (representa o conjunto de valores, neste caso *strings*, que os indivíduos podem conter no seu cromossoma;
- O tamanho mínimo e máximo do cromossoma de um indivíduo.;
- O número de objetivos do problema.

Como descrito na Secção 3.4 no AGE as melhores soluções são escolhidas por um operador de seleção. Após a seleção, os cromossomas dos indivíduos escolhidos como progenitores devem ser recombinados de forma a criarem uma nova geração de indivíduos. O método como a recombinação é concretizada depende da representação dos indivíduos. Para a representação de conjunto de *strings* optou-se por utilizar o processo de recombinação *random respectful recombination* [45] :

1. São criadas duas listas com os elementos presentes nos cromossomas dos progenitores. A primeira lista possui os elementos comuns aos cromossomas de ambos os progenitores, enquanto a segunda contem os elementos não comuns.
2. Os elementos da primeira lista são adicionados a ambos os descendentes.

3. De forma aleatória são adicionados elementos da segunda lista aos descendentes, sendo garantido o tamanho mínimo e máximo definido para os indivíduos.

A operação de mutação para esta representação consiste apenas na substituição de n elementos do cromossoma do indivíduo por n elementos que não estejam presentes no cromossoma. Sendo que n é definido na configuração do AGE.

4.5 Classificadores com Técnicas Evolucionárias de Procura

A integração das técnicas de computação evolucionária no protótipo de classificação visa resolver um problema de procura. O problema consiste em encontrar uma combinação de atributos que maximize o valor de AUC obtido na classificação de novas mensagens de *email*. O domínio da procura do AGE limita-se aos atributos mais relevantes selecionados de acordo com o critério IG. A razão de reduzir o domínio de procura de atributos numa primeira fase através do critério IG prende-se com o facto de os AGEs serem computacionalmente exigentes.

Na Figura 4.6 é apresentado o processo genérico de seleção de atributos com integração de técnicas de computação evolucionária. Inicialmente o conjunto de dados de treino inicial (não filtrado) é reduzido através da seleção dos melhores atributos de acordo com um critério de ponderação, calculado pelo IG. Esta primeira fase de seleção de atributos permite reduzir o domínio de procura do AGE. A segunda fase de seleção de atributos consiste num método *wrapper*. Isto significa que os processos de treino e classificação são utilizados, neste caso através de um AGE, para selecionar a melhor combinação de atributos de acordo com os resultados obtidos da classificação. As combinações de atributos que obtenham melhores resultados de classificação, neste caso valor de AUC mais elevado, são selecionadas para gerarem novas soluções através da recombinação do seu cromossoma (lista de atributos). A nova geração de soluções vai substituir a população de soluções anterior e os valores de AUC são novamente calculados. Este processo é cíclico e termina ao fim de um número de gerações definido como critério de paragem. Após o término do AGE é selecionado o melhor indivíduo, ou seja a combinação de atributos que obteve o valor mais elevado de AUC. O conjunto de dados representado pela melhor combinação de atributos é utilizado para constituir um modelo de classificação. O

modelo obtido através deste processo é o classificador final que vai ser utilizado para classificar novas mensagens de *email*.

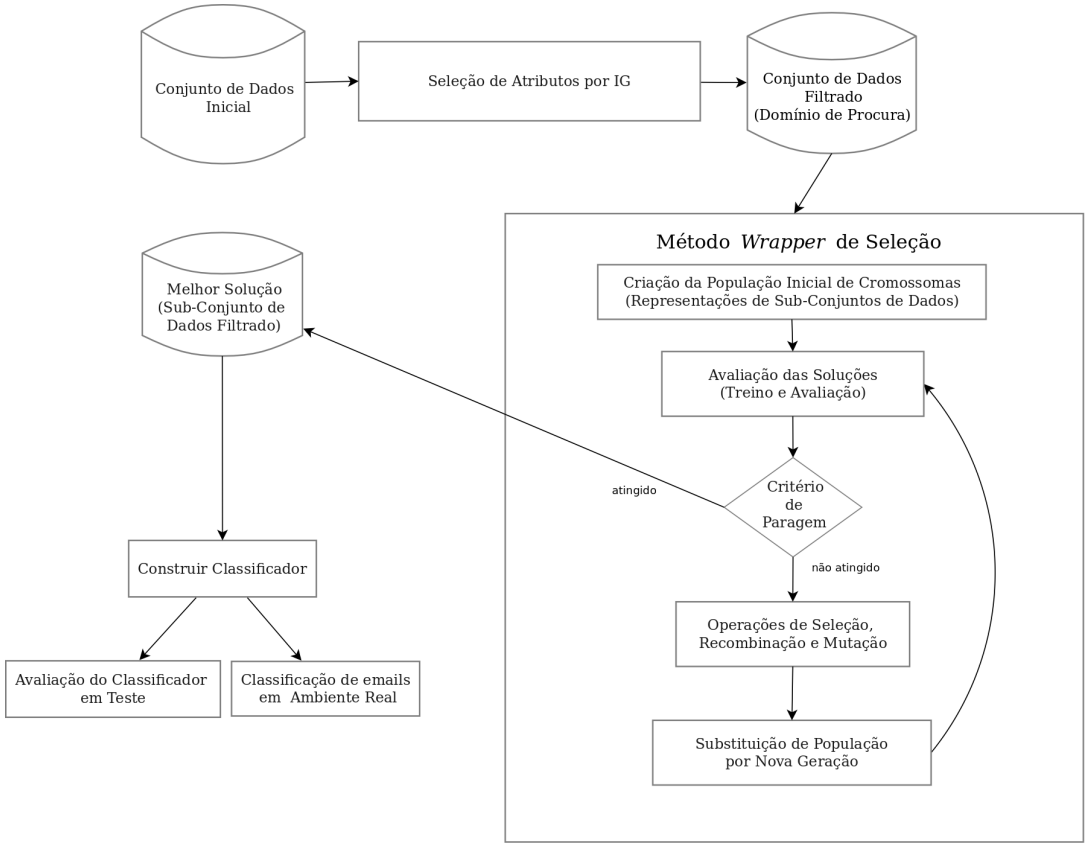


Figura 4.6: Componentes principais do protótipo de classificação desenvolvido com técnicas de computação evolucionária.

Desta forma o processo de classificação pode ser dividido em duas fases. A primeira corresponde à fase de procura, onde são utilizadas as técnicas de computação evolucionária para selecionar um conjunto de atributos. A segunda consiste na fase de teste, onde é utilizado o conjunto de atributos selecionado da fase de procura para constituir o modelo de classificação final. Este modelo é posteriormente utilizado para classificar novas mensagens de *email* em ambiente real, ou no caso deste trabalho para avaliar o desempenho do modelo de classificação (ver Figura 4.6). Note-se que o processo da fase de procura é apenas realizado sobre o conjunto de dados de treino, ou seja sobre mensagens de *email* já classificadas. Assim é necessário de alguma forma que parte das amostras de treino sejam utilizadas como amostras de avaliação para calcular os valores de AUC utilizados pelo AGE para selecionar as melhores soluções. Na Figura 4.7 é demonstrada a forma como o conjunto de

dados é dividido para a fase de procura e para a fase de teste, tendo em conta que é utilizada uma classificação com treino incremental cumulativo. Como referido na Secção 4.1, no treino incremental cumulativo uma caixa de *email* é dividida em lotes $l_1 ... l_n$ com K mensagens contíguas. O filtro é treinado com $l_1 \cup ... \cup l_i$ e a avaliação é feita com l_{i+1}.

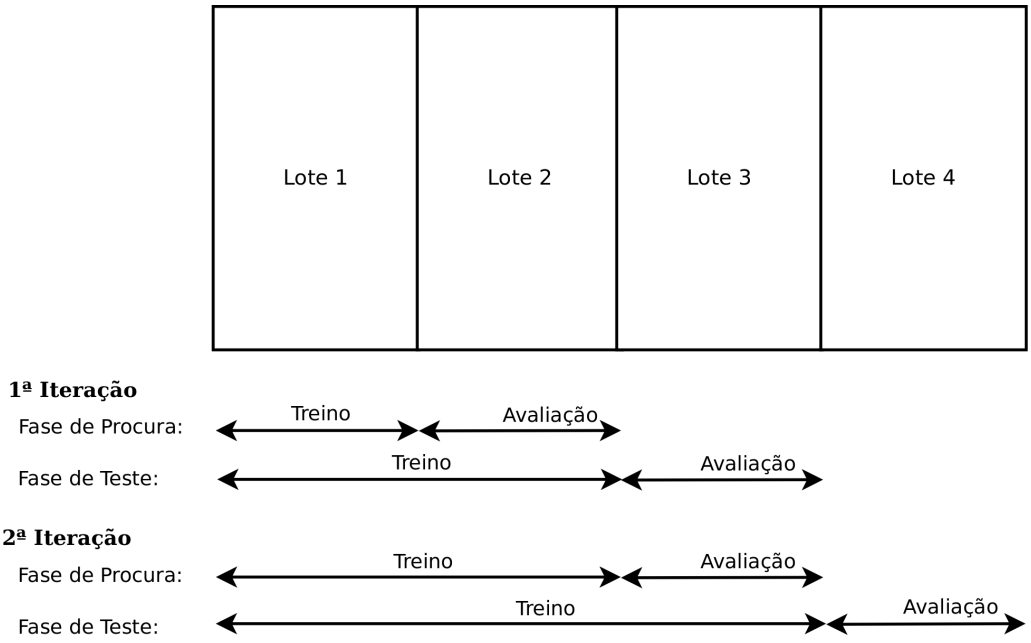


Figura 4.7: Método de divisão dos conjuntos de dados.

Na fase de procura da primeira iteração as soluções do AGE são avaliadas selecionando o lote 1 e o lote 2. O lote 1 é utilizado para treinar o modelo de classificação, e sobre as mensagens do lote 2 é efetuada a avaliação do modelo, utilizando os atributos representados na solução. O resultado desta avaliação é utilizado pelo AGE para selecionar as melhores soluções. Quando é atingido o critério de paragem do AGE, a melhor solução é selecionada para treinar o modelo de classificação na fase de teste, sendo que para isso são utilizados os lotes 1 e 2. A partir deste modelo são classificadas as novas mensagens de *email* presentes no lote 3. Resumindo os lotes $l_1 \cup ... \cup l_i$ são utilizados na fase de procura. Nesta fase $l_1 \cup ... \cup l_{i-1}$ são utilizados para treinar o filtro, e l_i para avaliar as soluções. Na fase de teste são utilizados os lotes $l_1 \cup ... \cup l_i$ para treinar o filtro e l_{i+1} para classificar as mensagens de *email*.

Como já foi referido na Secção 4.1 foram desenvolvidas três versões de classificadores que integram técnicas de computação evolucionária conforme foi explicado

nesta Secção. As diferenças entre estes classificadores é explicada de forma mais pormenorizada nas próximas Secções.

4.5.1 Classificador com Técnicas de Procura Evolucionária com Reinicialização

O classificador com técnicas de procura evolucionária com reinicialização foi o primeiro protótipo desenvolvido que integra estas técnicas. Neste protótipo é reinicializado um AGE na fase de procura a cada iteração. Isto significa que a cada iteração é criada uma população aleatória de indivíduos para definir a população inicial do AGE (ver Figura 4.8). É a partir desta população inicial aleatória que o AGE vai iniciar o processo de procura de uma combinação de atributos que maximize o valor de AUC. Essa procura é efetuada dentro do domínio de atributos previamente selecionados pelo critério IG.

Inicialmente o conjunto de dados é dividido em n lotes $l_1 \cup ... \cup l_n$. Os lotes $l_1 \cup ... \cup l_i$ correspondem aos lotes de mensagens já classificadas. O lote l_{i+1} representa o conjunto mais recente de mensagens que se pretendem classificar. A variável i é inicializada com o valor 2, pois são necessários no mínimo dois lotes para a fase de procura do AGE, o primeiro para construir o modelo de classificação e o segundo para avaliar as soluções. Numa primeira fase são utilizados os lotes $l_1 \cup ... \cup l_i$ para selecionar os N atributos mais relevantes de acordo com o critério IG. Os N atributos selecionados vão definir o domínio de procura do AGE. Quando o critério de paragem do AGE é atingido é selecionada a melhor solução. A melhor solução indica quais os atributos mais relevantes a utilizar no processo de treino e avaliação de um modelo. Os lotes $l_1 \cup ... \cup l_i$, utilizando os atributos da solução selecionada, vão ser usados para treinar o modelo de classificação. A classificação é efetuada sobre o lote l_{i+1}. A cada iteração o AGE é reinicializado, ou seja é criada uma população inicial de indivíduos aleatoriamente dentro do domínio definido pela seleção de atributos baseada no critério IG. Este processo termina quando o último lote de mensagens é classificado.

Na Figura 4.9 é apresentado o diagrama de classes para este classificador, no qual estão representadas as classes: $AGEWrapper$, $Dataset$ e $DatasetEvaluation$. Na classe $AGEWrapper$ é implementado o ciclo que define o processo de classificação iterativo conforme explicado anteriormente. Portanto é nesta classe que o conjunto

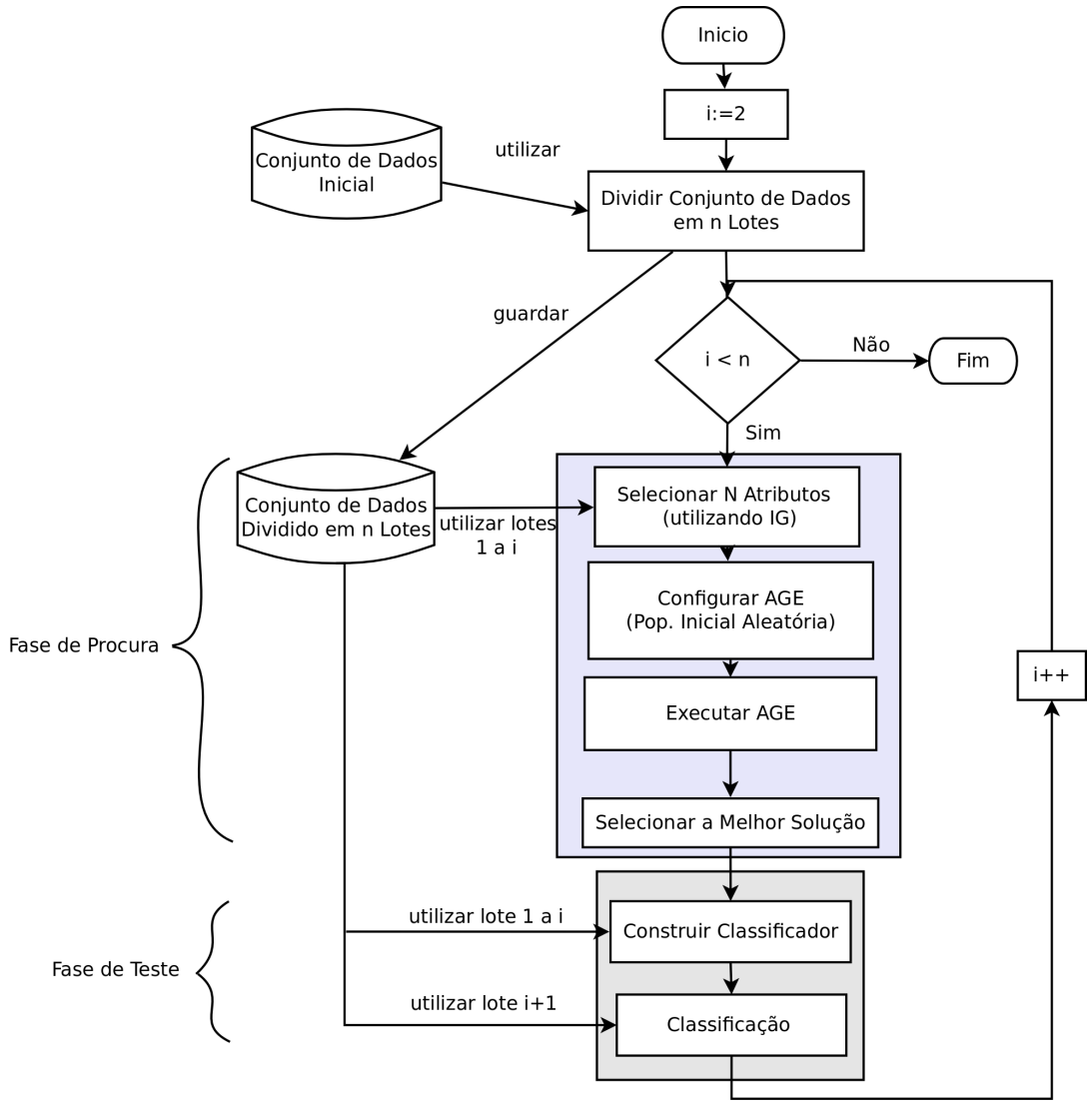


Figura 4.8: Fluxograma de classificação com técnicas de computação evolucionária com reinicialização.

de dados é divididos em n lotes. Posteriormente são selecionados os N atributos mais relevantes dos lotes de treino $l_1 \cup ... \cup l_i$, de acordo com o critério IG. A partir desta fase é efetuada a configuração do AGE com os parâmetros descritos na Secção 4.2.2. O AGE é executado nesta classe, e a cada iteração são guardadas as representações dos melhores indivíduos de cada geração, o correspondente valor de AUC da fase de procura, bem como a representação da melhor solução do AGE e o valor de AUC correspondente em teste. A classe *Dataset* permite efetuar a leitura do conjunto de dados, em vários formatos (e.g., ARFF, CSV, sparse). Além disso nesta classe é implementado o processo de classificação, que consiste na função de

avaliação dos indivíduos do AGE. Este processo é utilizado na fase de procura para avaliar os indivíduos do AGE, bem como na fase de teste para classificar o lote l_{i+1}, utilizando os atributos do melhor indivíduo do AGE.

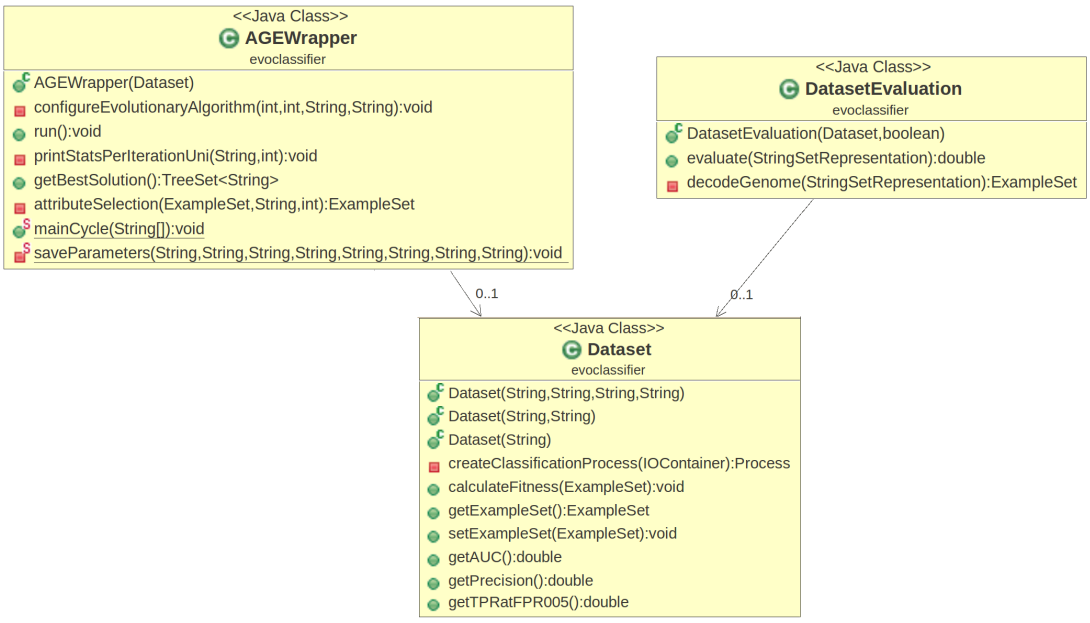


Figura 4.9: Diagrama de classes de classificador com técnicas de computação evolucionária integradas.

A classe *DatasetEvaluation* permite atribuir os valores de aptidão aos indivíduos de um AGE. Para isso o cromossomas dos indivíduos têm de ser descodificados para os sub-conjuntos de dados correspondentes. Após a descodificação do cromossoma de um indivíduo os lotes $l_1 \cup ... \cup l_{i-1}$ do sub-conjunto de dados correspondente são utilizados para construir um modelo de classificação, e o lote l_i para avaliar esse modelo de classificação. O processo de treino e avaliação do modelo de classificação é implementado na classe *Dataset* e é invocado pelo método *evaluate* da classe *DatasetEvaluation*. O valor de AUC obtido do processo de classificação é o valor de aptidão atribuído ao indivíduo. Existem outras duas métricas que são calculadas no processo de classificação, *precision*[5] e o TVP@TFP(para TFP=0.005)[6],

[5]Medida de relevância utilizada em recuperação de informação, corresponde ao rácio entre verdadeiros positivos e total de positivos na previsão obtida por um classificador.

[6]TVP@TFP(para TFP=0.005) - Taxa de Verdadeiros Positivos (TVP) quando o valor da Taxa de Falsos Positivos (TFP) é igual a 0.05 (ver Secção 5.4).

estas métricas são meramente informativas, sendo que posteriormente podem ser utilizadas para problemas de procura multi-objetivo, onde se pretendem maximizar duas ou mais métricas de classificação.

4.5.2 Classificador com Técnicas de Procura Evolucionária sem Reinicialização

A segunda versão do classificador com técnicas de computação evolucionária difere da anterior na forma como o domínio de procura é definido e na população inicial que é utilizada pelo AGE, sendo que estas diferenças estão relacionadas. Nesta versão a população final de um AGE vai constituir a população inicial do AGE na próxima iteração. Isto significa que é dada alguma relevância aos atributos selecionados pelo AGE na iteração anterior sobre os lotes $l_1 \cup ... \cup l_{i-1}$. Estes atributos podem estar fora do domínio de atributos selecionados na iteração atual pelo critério IG calculado sobre os lotes $l_1 \cup ... \cup l_i$. Assim o domínio de procura é alargado aos atributos presentes na população final do AGE da iteração anterior. A lógica desta abordagem é: se estes atributos fazem parte das soluções que obtiveram melhores resultados de classificação na iteração anterior, porque não garantir que são incluídos, apesar do *concept drift*[7] [16] que possa existir entre mensagens de lotes diferentes, no domínio de procura do AGE na presente iteração. Além disso o facto de a população inicial ser constituída pelas melhores soluções da iteração anterior pode traduzir-se numa vantagem no ponto de partida do AGE para a procura da combinação de melhores atributos.

Como se pode verificar na Figura 4.10 nesta abordagem o processo de classificação é semelhante à estratégia referida na Secção 4.5.1. A diferença está na configuração do AGE. A população inicial é preenchida com os indivíduos da população final do AGE da iteração anterior. Além disso os atributos representados nestes indivíduos que não foram selecionados pelo critério IG têm de ser acrescentados ao domínio de procura do AGE. Quando o critério de paragem do AGE é atingido a população final é guardada para possibilitar a sua utilização na iteração seguinte. O diagrama de classes deste classificador é em tudo idêntico ao anterior. Ao nível da implementação as únicas alterações que se efetuaram foram concretiza-

[7]No contexto da filtragem *anti-spam* as palavras mais correlacionadas com a classe *spam* ou *ham* variam ao longo do tempo.

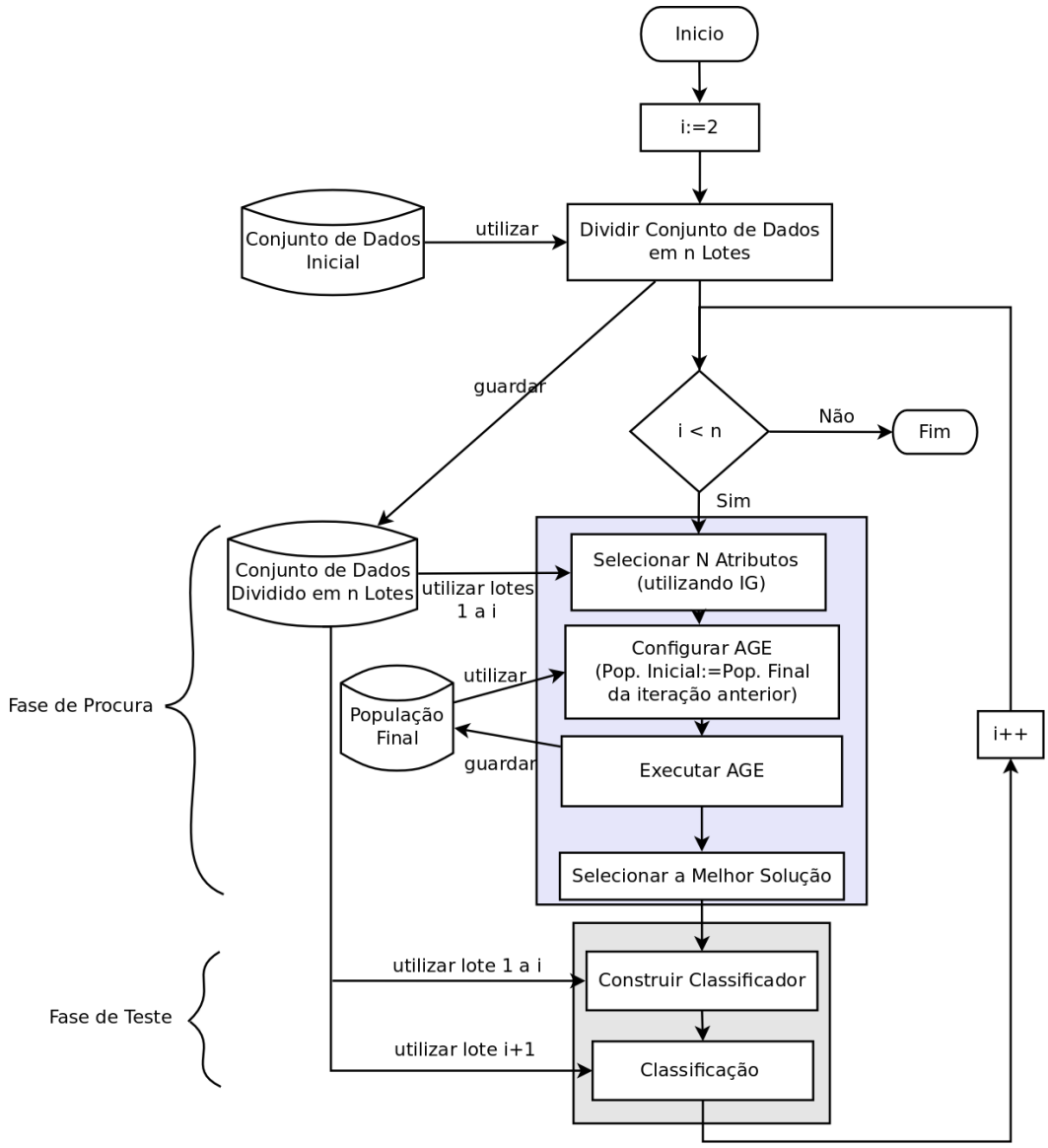


Figura 4.10: Fluxograma de classificação com técnicas de computação evolucionária sem reinicialização.

das no método de configuração do AGE.

4.5.3 Classificador com Técnicas de Procura Evolucionária e Partilha de Atributos

Nas duas versões de classificadores que utilizam técnicas de computação evolucionária, com reinicialização e sem reinicialização, a construção do modelo de classificação é efetuado utilizando um conjunto de dados do próprio utilizador sem

existir um mecanismo de colaboração entre os filtros pertencentes a diferentes utilizadores. Estes filtros designam-se de filtros locais. Esta terceira abordagem segue uma estratégia colaborativa de classificação. Desta forma é introduzida a possibilidade de os filtros locais trocarem entre si atributos que se considerem relevantes. Os atributos são trocados através da partilha de indivíduos da população final entre AGEs de utilizadores diferentes. A troca é feita à semelhança da abordagem referida na Secção 4.5.2. Os indivíduos recebidos de outros utilizadores são introduzidos na população inicial de um AGE local. Além disso o domínio de procura do AGE é alargado aos atributos representados nestes indivíduos que não constem nos atributos selecionados pelo critério IG. Com esta estratégia pretende-se diversificar o domínio de procura do AGE nos filtros locais, para eventualmente ser possível obter uma combinação de atributos que obtenha melhores resultados de classificação. Além disso os atributos considerados relevantes por outros filtros podem constituir uma mais valia para a classificação no filtro local.

Como referido anteriormente, nesta abordagem os indivíduos recebidos de outros utilizadores são introduzidos na população inicial do AGE. Mas de forma a manter atributos identificados como relevantes na fase de procura sobre lotes de mensagens anteriores, optou-se também por introduzir indivíduos do AGE da iteração anterior do próprio utilizador. Pretende-se assim manter informação sobre o histórico de classificação de lotes anteriores do próprio utilizador, acrescentando informação de classificação considerada relevante por outros utilizadores. A proporção entre estas duas fontes de informação é configurada pelo parâmetro p. O parâmetro p define a percentagem de indivíduos da população inicial que são provenientes de outros utilizadores. Esta percentagem é repartida equitativamente entre os utilizadores que fazem parte do grupo de partilha. O valor $1 - p$ corresponde à percentagem de indivíduos provenientes da população final do AGE da iteração anterior do próprio utilizador.

A partilha de atributos respeita a ordem cronológica pela qual as mensagens de *email* são recebidas. Imaginemos que na fase de procura um AGE vai utilizar o lote l_i do utilizador X para avaliar as soluções. Este AGE pode receber indivíduos do AGE do utilizador Y que utiliza o lote l_j para avaliar as soluções, desde que a data da última mensagem de l_j seja anterior à data da primeira mensagem de l_i. Por exemplo na Figura 4.11 o AGE 1 do utilizador X pode receber indivíduos da população final do AGE 1 do utilizador Y, pois o lote 2 de Y termina (data da

última mensagem recebida), antes do início (data da primeira mensagem recebida) do lote 2 de X. Segundo este critério o AGE 2 do utilizador X poderia receber indivíduos de ambos os AGEs do utilizador Y. Mas foi decidido que cada AGE local apenas recebe indivíduos de um AGE por utilizador. Esse AGE consiste no AGE mais recente que respeite a restrição temporal. Assim o AGE 2 de X apenas recebe indivíduos da população final do AGE 2 do utilizador Y.

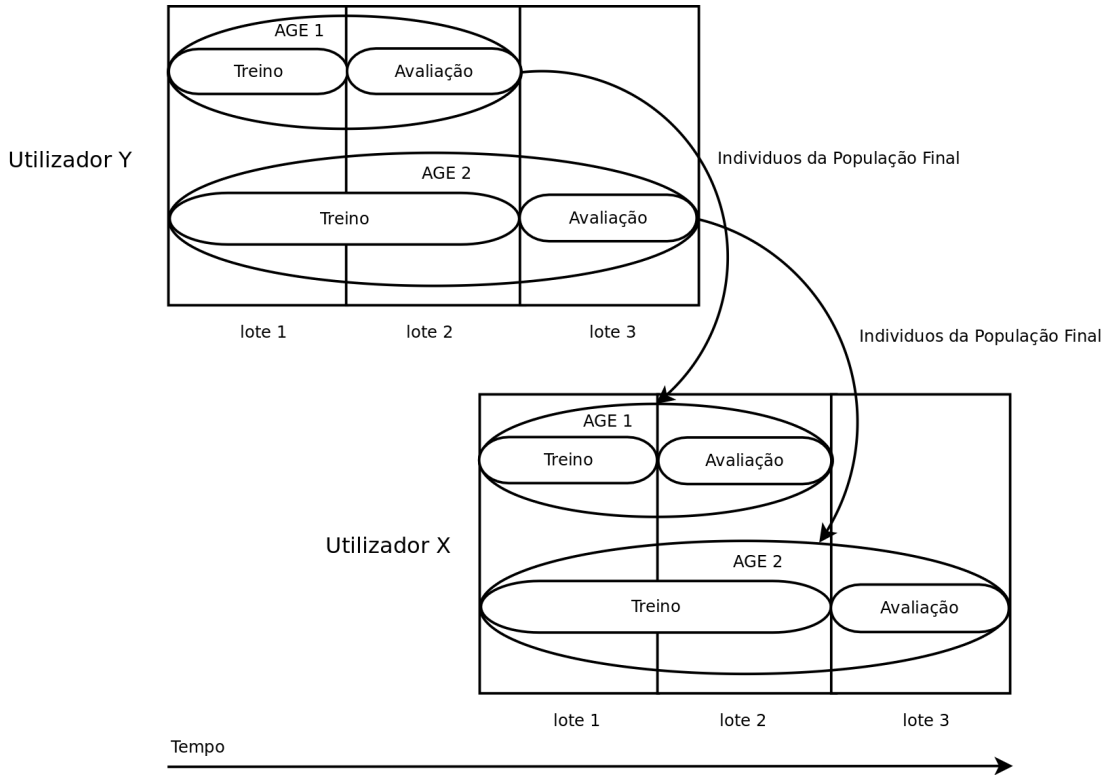


Figura 4.11: Partilha de indivíduos entre dois utilizadores.

Quando um AGE de um utilizador X recebe indivíduos de um AGE que pertencente a um utilizador Y existe a possibilidade de certos atributos representados nos indivíduos partilhados não constarem no conjunto de dados inicial de X. Nesta situação estes atributos podiam ser apenas ignorados pelo AGE local. Nesta abordagem optou-se por substituir estes atributos por outros atributos presentes no domínio de procura do AGE de Y, que não estavam representados nos indivíduos partilhados. Com esta estratégia aumenta-se a diversidade de atributos partilhados entre utilizadores.

Na Figura 4.12 é apresentado o diagrama de classes do classificador com técnicas

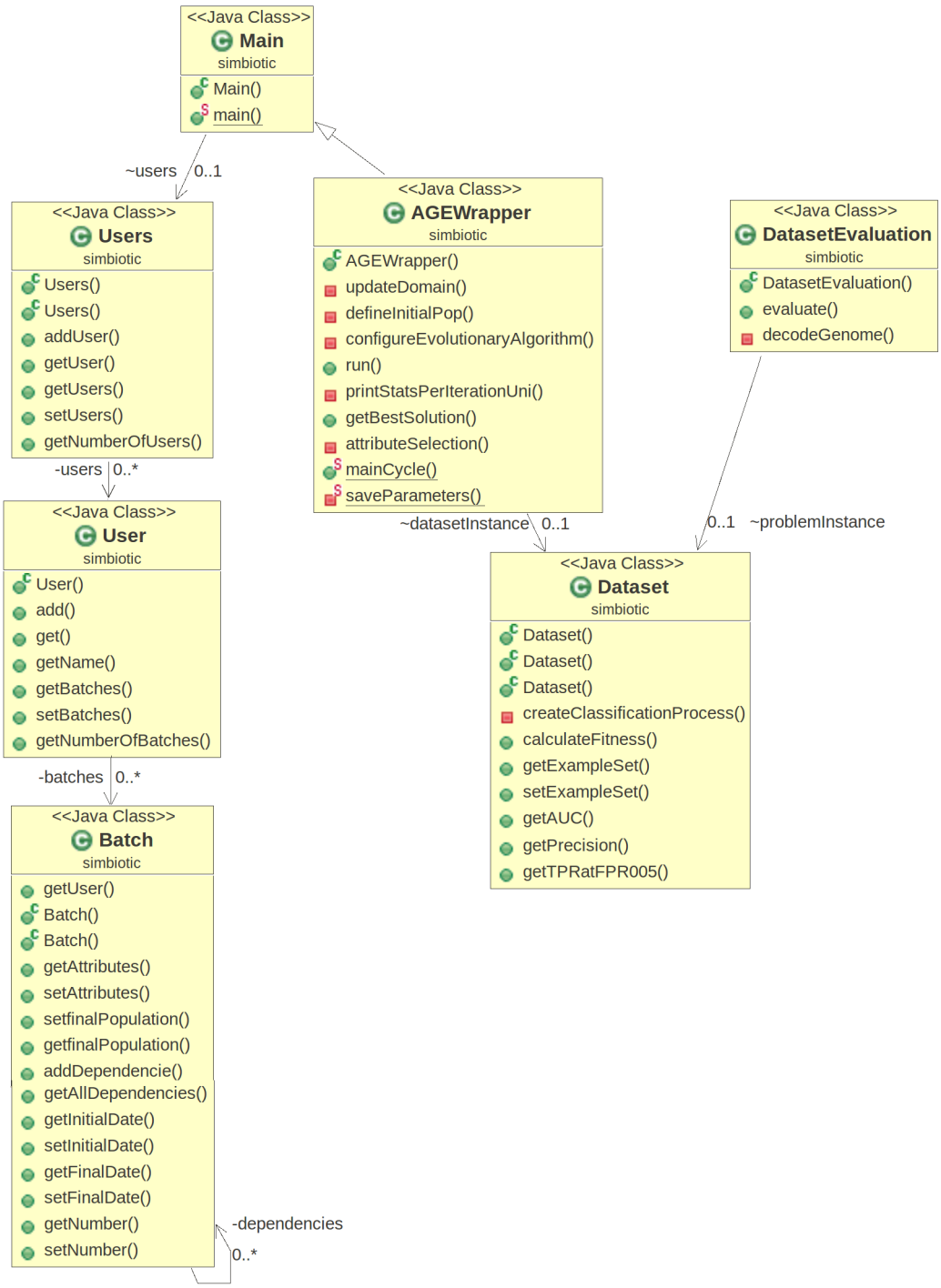


Figura 4.12: Diagrama de classes do classificador com técnicas de procura evolucionária e partilha de atributos.

de procura evolucionária e partilha de atributos. As classes *AGEWrapper*, *Dataset*, *DatasetEvaluation* foram reutilizadas do classificador descrito na Secção 4.5.1 com apenas algumas alterações a serem efetuadas na classe *AGEWrapper*. Esta classe implementa agora uma interface *runnable* que vai permitir criar um número de *threads* igual ao número de utilizadores presentes no grupo de partilha. Em cada *thread* é executado o processo de classificação iterativa sobre o conjunto de dados de um utilizador conforme explicado em 4.5.1. Inicialmente na classe *Main* é criada uma lista de dependências para cada lote de mensagens (*Batch*) de cada utilizador. As dependências são criadas através da leitura das datas de recepção de cada mensagem de *email* de cada utilizador que se encontram guardadas em ficheiro. O número de mensagem contido em cada lote é um parametrizável. Esta lista de dependências indica para um lote de mensagens quais são os lotes de mensagens de outros utilizadores cujos atributos podem ser partilhados de forma a respeitar a ordem cronológica das mensagens. Após este processo é criada uma instância da classe *Batch* para cada lote de mensagens, onde constam as datas inicial e final do lote, e a lista das instâncias *Batch* de outros utilizadores da qual pode receber atributos. Após cada iteração da fase de procura, a população final do AGE é guardada na instância *Batch* correspondente ao lote de mensagens utilizado pelo AGE na avaliação.

Antes da execução de um AGE, na classe *AGEWrapper* é verificado para o lote de mensagens que vai ser utilizado para avaliação na fase de procura a lista de dependências. Os indivíduos da população final dos lotes presentes na lista de dependências vão ser utilizados para constituir a população inicial do AGE, tendo em conta o valor do parâmetro p. Como já referido o parâmetro p vai definir o rácio, em relação ao tamanho da população definido, de indivíduos provenientes de outros utilizadores que vão constituir a população inicial do AGE. Os restantes indivíduos da população inicial são provenientes do AGE da iteração anterior do próprio utilizador. Para um AGE ser executado os AGEs de outros utilizadores executados sobre os lotes de mensagens presentes na lista de dependências têm de ter terminado. Só desta forma é possível obter a população final desses AGEs para constituir a população inicial do AGE local. É no método $defineInitialPop()$ da classe *AGEWrapper* que é aguardado o término dos AGEs de outros utilizadores. No método *updateDomain*() da mesma classe é alargado o domínio de procura do AGE a atributos que não foram selecionados pelo critério IG, mas que se encontram

representados nos indivíduos partilhados.

As classes ainda não referidas $Users$ e $User$ são utilizadas respetivamente para definir o grupo de partilha, e o utilizador. Na classe $Users$ vão ser mantidas as instâncias da classe $User$. Por sua vez na classe $User$ são mantidas as instâncias da classe $Batch$ que representam os lotes de mensagens presentes no conjunto de dados do utilizador. A classe $AGEWrapper$ herda da classe $Main$ precisamente para aceder à instância da classe $Users$. Desta forma na classe $AGEWrapper$ é possível aceder às instâncias da classe $Batch$ do respetivo utilizador no momento de guardar os indivíduos da população final de um AGE, e para obter os indivíduos de AGEs de outros utilizadores. Os atributos utilizados no domínio de procura de um AGE para avaliação de um dado lote são guardados também na instância $Batch$ correspondente ao lote de mensagens. Assim, como já foi explicado, quando um utilizador recebe um indivíduo de outro utilizador existe a possibilidade de esse indivíduo possuir um atributo que não conste no conjunto de dados local. Perante esta situação esse atributo é trocado por outro que pertença ao domínio de procura do AGE do utilizador que partilhou o indivíduo.

4.6 Sumário

Neste Capítulo foi descrito o desenvolvimento de diversos filtros *anti-spam*, designados aqui por classificadores. No inicio do Capítulo são apresentadas as bibliotecas RapidMiner4.6 e JECoLi utilizadas no desenvolvimento dos classificadores. A primeira biblioteca disponibiliza um elevado número de operadores que permite implementar processos de *data mining*. A JECoLi implementa algoritmos de otimização meta-heurísticos, com especial destaque para algoritmos de CGE.

De seguida foi explicada a implementação do classificador simples que utiliza apenas um método de filtragem[8] para selecionar atributos. Na mesma Secção é demonstrado um pequeno *how-to* de utilização deste classificador. Este classificador foi desenvolvido para possibilitar uma comparação dos resultados de classificação com os classificadores que utilizam técnicas de computação evolucionária de procura de atributos.

[8]Não confundir filtragem como método de seleção de atributos que se realiza durante a fase de pré-processamento, com filtragem *anti-spam*.

Posteriormente foi apresentada uma representação nova de conjuntos de *strings* desenvolvida especificamente de modo a facilitar o desenvolvimento das soluções desenvolvidas neste projeto. Esta nova representação permite definir cromossomas de tamanho variável dentro dos limites mínimos e máximos especificados. Além disso não é necessário um mapeamento dos conjuntos de dados de diferentes utilizador uma vez que os cromossomas dos indivíduos contêm explicitamente o nome dos atributos que são representados.

Por último é explicado como foram integradas as técnicas de computação evolucionária na procura de um conjunto de atributos que maximize o valor de AUC. Nesse sentido é utilizado um método *wrapper* onde se verifica para várias combinações de atributos, que são representados por indivíduos do AGE, os valores de AUC obtidos na classificação de mensagens de *email*. Sendo que a combinação de atributos que possuir um valor de AUC mais elevado na classificação sobre os lotes de treino é utilizada para constituir o modelo de classificação utilizado para classificar os *emails* dos lotes de teste. Foram desenvolvidos dois filtros locais que implementam estas técnicas com abordagens um pouco diferentes, sendo que uma das abordagens utiliza atributos considerados relevantes em lotes de mensagens anteriores para a fase de procura. Finalmente foi desenvolvido um filtro colaborativo *anti-spam* que permite a partilha de atributos considerados relevantes entre filtros locais pertencentes a diferentes utilizadores. Este filtro *anti-spam* foi designado de classificador com técnicas de procura evolucionária e partilha de atributos. Esta partilha permite aumentar o domínio de procura a atributos considerados relevantes pelos filtros locais de outros utilizadores. Os atributos considerados mais relevantes pelo filtro local para um dado lote de mensagens são aqueles representados na população final do AGE na fase de procura sobre esse mesmo lote. Para terminar é de referir que a partilha de atributos respeita a ordem cronológica das mensagens, desta forma é possível efetuar uma análise mais realista do comportamento do filtro *anti-spam* colaborativo.

Capítulo 5

Experiências e Resultados Obtidos

De forma a avaliar os filtros *anti-spam* desenvolvidos explicados no Capítulo anterior, foram realizadas diversas experiências. A partir destas experiências são obtidas métricas de classificação, nomeadamente o AUC e TVP@TFP(TFP=0.05), que permitem avaliar e comparar o desempenho dos filtros *anti-spam*. Inicialmente é descrita a forma como foram constituídos os conjuntos de dados utilizados na experimentação e o pré-processamento efetuado. Posteriormente são explicadas as métricas de avaliação utilizadas. Por último, são demonstrados e analisados os resultados obtidos de cada filtro sobre as caixas de *email* constituídas.

5.1 Introdução

Neste Capítulo são descritas as experiências realizadas com os protótipos de classificação desenvolvidos e analisados os resultados obtidos. Os resultados de classificação são comparados entre as três versões de classificadores que utilizam técnicas de computação evolucionária e o classificador simples. Em todas as experiências e para todos os protótipos de classificação, foi utilizado um algoritmo de aprendizagem *Bayesiano*, nomeadamente o *Multinomial Naive Bayes* com atributos de frequência de termos, descrito na Secção 2.6.3. Optou-se por este algoritmo de aprendizagem máquina por ser particularmente popular e amplamente utilizado em soluções de filtragem de *spam* [3, 8, 36]. O critério de seleção de atributos utilizado foi o IG (ver Secção 5.2.2). Este critério de seleção é também bastante utilizado em problemas de filtragem de *spam* [26]. Antes de realizar as experiências foram constituídos e selecionados dez conjuntos de dados que representam as caixas de *email* pertencentes a cinco utilizadores presentes no repositório público *Enron*.

Para cada um dos cinco utilizadores foram constituídas duas caixas de *email* que possuem mensagens de *spam* provenientes de fontes diferentes (ver Secção 5.2).

5.2 Conjunto de Dados

De forma a testar os protótipos de classificação desenvolvidos foi necessário previamente constituir caixas de correio eletrónico com mensagens de *spam* e *ham*. Os *emails* legítimos foram obtidos através do repositório público *Enron*. Este repositório possui 619.446 mensagens pertencentes a um grupo de 158 trabalhadores da já extinta companhia de energia norte-americana *Enron Corporation*. Para cada caixa de *email* deste conjunto de dados foram selecionados apenas os *emails* legítimos recebidos durante os anos 2000 e 2001 inclusive. No final deste processo sobraram 112 caixas de *email* de utilizadores que receberam durante esse período pelo menos 200 mensagens legítimas.

Como o *spam* evolui temporalmente [13, 30], optou-se por obter mensagens de *spam* mais recente de duas origens diferentes: do arquivo público *Bruce Guenter*[1], e do repositório *Spam Telescope Miner*, este último foi criado através de um projeto de investigação e desenvolvimento da Universidade do Minho [12]. Assim foram constituídos dois conjuntos de dados diferentes, ambos com as mesmas mensagens legítimas, mas com mensagens de *spam* provenientes de duas origens diferentes. Pretende-se assim que os resultados de classificação obtidos possam ser analisados de forma mais conclusiva e imparcial, pois são utilizados dois conjuntos de dados com misturas de *ham* e *spam* diferentes.

Para a primeira mistura, designada *Enron-Bg*, foram selecionadas do arquivo do *Bruce Guenter* 19.196 mensagens de *spam* de contas de *email* distintas, estas mensagens foram recebidas durante os anos de 2009 e 2010. O conjunto de dados foi criado adicionando de forma aleatória mensagens de *spam* do arquivo *Bruce Guenter* às caixas de *email* de cada utilizador do repositório *Enron*. As mensagens de cada caixa de *email* foram ordenadas temporalmente. Contudo devido ao facto das mensagens de *spam* serem mais recentes e de forma a garantir uma sequência realista de recepção de ham e *spam*, foi adicionado um período temporal de 9 anos

[1]Disponível em http://untroubled.org/spam/ .

às mensagens legítimas.

O processo para constituir a segunda mistura, apelidada de *Enron-Ts*, foi em tudo semelhante ao descrito anteriormente, sendo que as mensagens de *spam* foram obtidas a partir de uma única caixa de correio do repositório *Spam Telescope Miner*. Esta caixa de correio foi criada unicamente para receber mensagens de *spam*, e possui 15.597 mensagens deste tipo. Como resultado deste processo foram constituídas duas caixas de *email*, com mensagens de *spam* de origens diferentes, para cada um dos 112 utilizadores selecionados do arquivo *Enron*. A maioria das mensagens de *email* estão escritas na língua inglesa. Além disso para cada caixa de *email* existe um ficheiro que indica a data de recepção de cada mensagem. Este ficheiro é utilizado no protótipo de classificação com partilha de atributos para garantir que a partilha respeita a ordem temporal (ver Secção 4.5).

5.2.1 Seleção de Caixas de *Email*

Para realizar as experiências nos protótipos de classificação desenvolvidos decidiu-se selecionar as caixas de *email* de 5 utilizadores. As experiências foram realizadas sobre as duas misturas existentes, assim no total foram utilizadas 10 caixas de *email*. Como um dos protótipos desenvolvidos utiliza partilha de atributos entre utilizadores para efetuar a classificação das mensagens de *email*, optou-se por selecionar os utilizadores cuja combinação de caixas de correio possuísse o valor mais elevado de sobreposição temporal (ver Eq. (5.1)). Isto porque a partilha de atributos respeita a ordem temporal pela qual as mensagens são recebidas (ver Secção 4.5). Assim, assume-se que se o valor de sobreposição temporal for elevado entre as caixas de *email* de um grupo de utilizadores, estes possuem mais vantagem em trocar atributos para obterem melhores resultados de classificação. O cálculo da sobreposição temporal para duas caixas de *email* é dado pelo seguinte rácio:

$$\frac{t_p}{t_n}$$

onde t_p representa o período comum, e t_n o período não comum em milissegundos entre as duas caixas de *email*. Na Figura 5.1 é exemplificado o modo como é calculada a sobreposição temporal para três caixas de *email*.

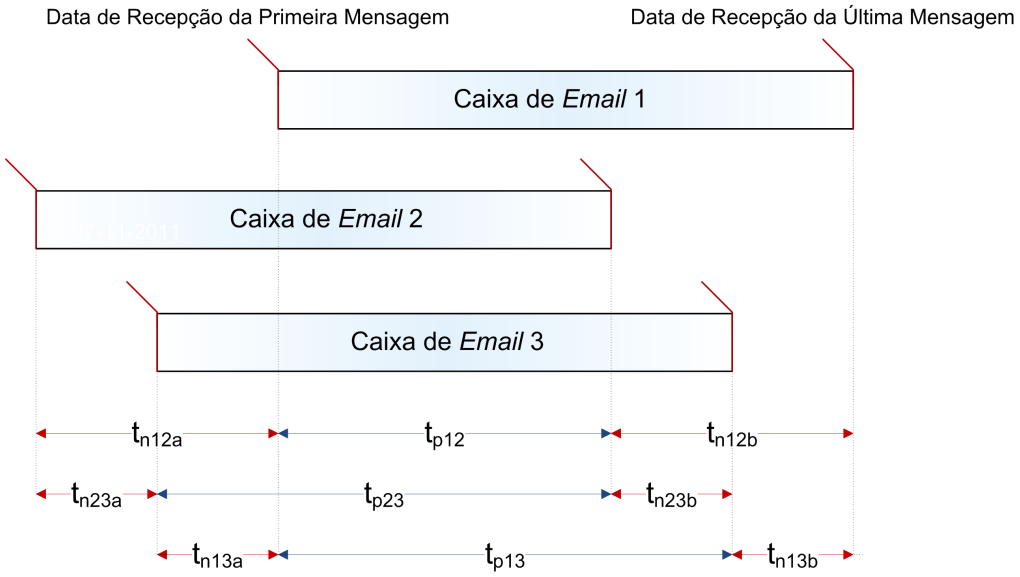


Figura 5.1: Exemplo de sobreposição temporal entre caixas de *email*.

Inicialmente são calculados os períodos temporais comuns e não comuns para cada par de caixas de *email*. O valor final da sobreposição temporal neste exemplo é dado por:

$$S_t = \frac{t_{p12} + t_{p23} + t_{p13}}{t_{n12a} + t_{n12b} + t_{n23a} + t_{n23b} + t_{n13a} + t_{n13b}} \tag{5.1}$$

A sobreposição temporal foi calculada para combinações de 5 utilizadores cujas caixas de *email* tivessem entre 500 e 20.000 mensagens. Convém referir que este cálculo foi efetuado para ambas as misturas, *Enron-Bg* e *Enron-Ts*. Sendo que o valor final, utilizado como critério de seleção, foi obtido para cada grupo de utilizadores através da soma das sobreposições temporais calculadas para cada mistura. Segundo este critério foi selecionado o seguinte grupo de 5 utilizadores em cada uma das mistura *Enron-Bg* e *Enron-Ts*: *martin-t*, *platter*-p, *saibi-e*, *scholtes-d*, *smith-m*. Nas Tabelas 5.1 e 5.2 são apresentadas algumas caraterísticas das caixas de *email* selecionadas.

Tabela 5.1: Caixas de *email* selecionadas da mistura *Enron-Bg*.

Utilizador	Nº de Mensagens	Nº de Atributos	Rácio *Spam*/*Ham*
martin-p	888	5.057	1,51
platter-p	672	2.303	2,15
saibi-e	1.688	4.476	1,38
scholtes-d	765	2.833	1,5
smith-m	941	3.460	0,94

Tabela 5.2: Caixas de *email* selecionadas da mistura *Enron-Ts*.

Utilizador	Nº de Mensagens	Nº de Atributos	Rácio *Spam*/*Ham*
martin-p	799	5.055	1,27
platter-p	558	2.364	1,62
saibi-e	1.622	4.747	1,29
scholtes-d	679	2.917	1,22
smith-m	996	3.775	1,05

Nas Figuras 5.2 e 5.3 verifica-se para as 10 caixas de correio dos 5 utilizadores selecionados, o período temporal desde a primeira até à última mensagem recebida está compreendido no ano de 2010.

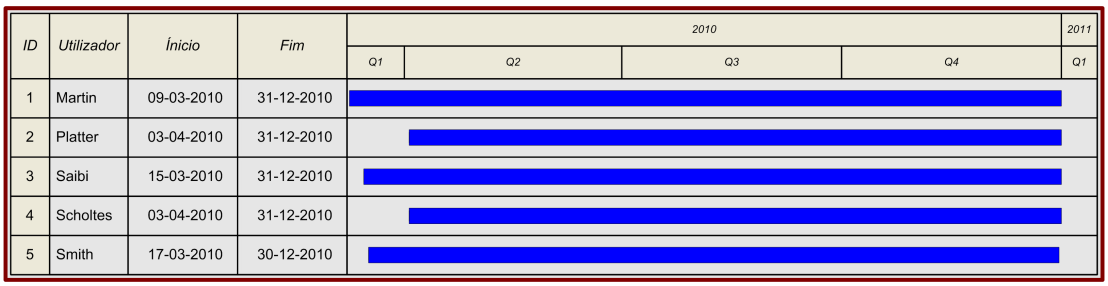


Figura 5.2: Períodos temporais das caixas de *email* selecionadas da mistura *Enron-Bg*.

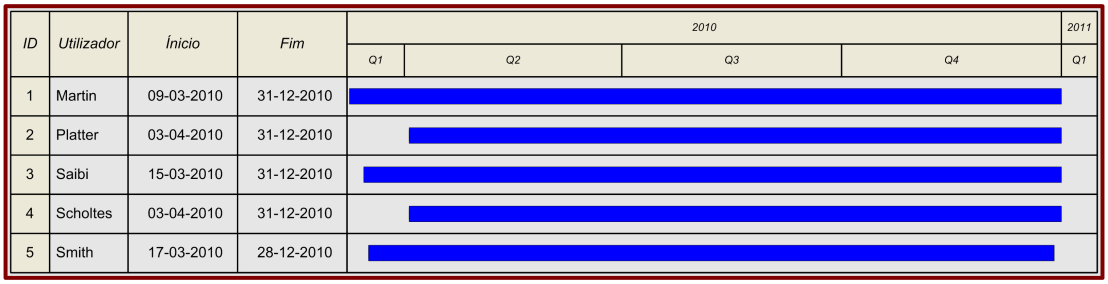


Figura 5.3: Períodos temporais das caixas de *email* selecionadas da mistura *Enron-Tel*.

5.2.2 Pré-processamento

A tarefa de pré-processamento das caixas de *email* é necessária para extrair os atributos de cada mensagem de *email* e estruturar os dados no formato a ser utilizado pelos protótipos de classificação. Cada caixa de *email* é representada num único ficheiro de texto onde estão presentes todas as mensagens. É a partir deste ficheiro, e após pré-processamento, que se obtêm os conjuntos de dados utilizados para classificação. Antes de ser pré-processada, uma mensagem de *email* contém diversos elementos que devem ser removidos de modo a serem extraídos apenas os atributos de classificação, que consistem nas palavras presentes no corpo de cada mensagem de *email*. Logo para cada mensagem de *email* foi necessário remover *tags* HTML, anexos e também vários campos do cabeçalho como *From*, *To*, *Message-ID*, entre outros. Além disso os acentos de algumas palavras que surgiram em língua portuguesa foram removidos e os caracteres maiúsculos foram passados para minúsculas.

Para efetuar este pré-processamento foram utilizadas *scripts* desenvolvidas em *perl*. As *scripts* em questão foram desenvolvidas pelos orientadores desta Dissertação. O pré-processamento foi realizado para todas as caixas de *email* das misturas *Enron-Bg* e *Enron-Ts*. No final deste processo foram constituídos dois ficheiros para cada caixa de *email*. Um ficheiro *.words*, que representa a caixa de *email* através de uma matriz em modo denso (ver Secção 5.2.2). O segundo ficheiro *.dat* possui informação relativamente à data de receção de cada mensagem.

Representação dos Conjunto de Dados

Inicialmente as caixas de *email* presentes nos conjuntos de dados *Enron-Bg* e *Enron-Ts* eram representadas num modo denso, através de um ficheiro que seguia o formato CSV. De acordo com este formato uma caixa de *email* pode ser vista na forma de uma matriz (ver Figura 5.4). Cada linha da matriz corresponde a uma mensagem de *email* que é representada por um vetor cujos elementos indicam o número de ocorrências de cada termo na mensagem. Uma das colunas indica a classe à qual a mensagem de *email* pertence, *spam* (S) ou *ham* (H).

Row No.	Y	can	have	http	pills	the	their	they	viagra	watches	with
2	H	1	2	0	0	8	0	0	0	0	0
3	H	1	0	0	0	1	0	0	0	0	0
4	S	0	0	8	0	1	0	0	0	0	0
5	S	0	0	8	0	1	0	0	0	0	0
6	H	1	4	0	0	6	0	0	0	0	1
7	S	0	1	0	0	2	0	0	0	0	2
8	S	0	0	0	0	0	0	0	0	0	2
9	H	1	1	0	0	6	0	0	0	0	2
10	S	0	0	3	0	1	0	0	2	0	0
11	H	0	1	2	0	6	0	0	0	0	0
12	S	0	0	3	0	1	0	0	0	0	0
13	S	0	0	0	0	0	0	0	0	0	2
14	S	1	2	2	0	4	0	1	0	0	0
15	H	0	2	0	0	23	0	0	0	0	0
16	S	0	0	8	0	2	0	0	0	0	0
17	H	0	0	0	0	1	0	0	0	0	0

Figura 5.4: Extrato de caixa de *email* em modo denso.

Na realidade constata-se que numa caixa de *email* existem alguns milhares de palavras diferentes presentes nos corpos das mensagens. Para cada *email* apenas uma pequena parte desse conjunto de palavras é utilizado. Assim numa representação em modo denso, o vetor que representa o número de ocorrências de cada termo na mensagem de *email* é em grande parte constituído por elementos com o valor 0. Em alternativa, a utilização de uma representação em modo esparso elimina os valores nulos, reduzindo substancialmente o tamanho dos conjuntos de dados. Esta redução reflete-se na diminuição dos requisitos de processamento necessários, especialmente no momento em que os conjuntos de dados são carregados para memória. Numa representação em modo esparso o vetor do número de ocorrências de cada mensagem de *email* é formado pelo índice do termo seguido do valor. Os termos cujos índices não estejam representados num dado vetor têm por omissão o valor

nulo. A título de exemplo nas Figuras 5.5 e 5.6 são demonstrados os dois tipos de representação para o mesmo sub-conjunto de dados.

```
Y can have http pills the their they
S  0   1     0     3    1    0    0
H  2   0     0     0    1    0    0
S  0   0     2     1    0    1    0
H  0   1     2     0    0    0    2
```

Figura 5.5: Exemplo de representação em modo denso.

```
Y can have http pills the their they
S  1:1  3:3  4:1
H  0:2  4:1
S  2:2 3:1  5:1
H  1:1 2:2  6:2
```

Figura 5.6: Exemplo de representação em modo esparso.

Nas caixas de *email* selecionadas, o número de atributos varia entre 2.303 e 5.057. Logo a redução dos sub-conjuntos de dados por utilização de uma representação esparsa é bem mais notória que no exemplo anterior. Nas Tabelas 5.3 e 5.4 são demonstrados os tamanhos das caixa de *email* em modo denso e esparso para cada utilizador.

Tabela 5.3: Tamanhos para representações densa e esparsa na mistura *Enron-Bg*.

Utilizador	Representação Densa (kB)	Representação Esparsa (kB)
martin-p	8.806	1.001
platter-p	3.072	454
saibi-e	14.745	1.231
scholtes-d	4.300	627
smith-m	6.348	818

Tabela 5.4: Tamanhos para representações densa e esparsa na mistura *Enron-Tel*.

Utilizador	Representação Densa (kB)	Representação Esparsa (kB)
martin-p	7.987	1.124
platter-p	2.560	488
saibi-e	15.052	1.375
scholtes-d	3.891	672
smith-m	7.475	951

Para converter os conjuntos de dados para um formato esparso foi desenvolvida uma pequena aplicação em linguagem Java com recurso à biblioteca RapidMiner versão 4.6. Esta aplicação é compatível com todos os formatos de ficheiros de dados suportados pelo RapidMiner (e.g., CSV, ARFF, c4.5). A conversão é realizada para um formato esparso próprio do RapidMiner. Neste formato, além do ficheiro de dados é também gerado um ficheiro de descrição dos atributos. Nesse ficheiro são definidas as propriedades dos atributos como por exemplo: o nome, se é um atributo regular ou define a classe, o índice no conjunto de dados, os valores que pode representar (e.g., inteiro, real, nominal).

Seleção de Atributos

A seleção de atributos pode ser bastante vantajosa no contexto de um problema de classificação de mensagens de *email*. Os atributos presentes nas amostras de treino devem ser selecionados de acordo com o seu contributo para a caraterização correta das mensagens de *ham* e *spam*. Existem duas vantagens para um modelo de classificação na utilização da seleção de atributos. Por um lado este processo permite reduzir o espaço dimensional dos atributos que caraterizam as mensagens. Esta redução permite que o processo de treino de um modelo de classificação seja mais rápido. A outra vantagem reside no facto de através da seleção de atributos poderem ser eliminados atributos irrelevantes, o que possibilita construir modelos de classificação mais assertivos [7].

Inicialmente foram removidos os atributos que correspondem a palavras cujo número de ocorrências numa caixa de *email* fosse inferior a 5. Assim, foi possível eliminar um grande número de atributos que devido à sua baixa ocorrência são pouco úteis para o processo de classificação [3]. Palavras constituídas por dois ou menos carateres foram também excluídas. Estas *stopwords* consistem normalmente em palavras muito comuns, que por este motivo são pouco informativas para a

distinção da classe das mensagens.

Numa segunda fase foi utilizado um método de seleção de atributos. O IG, informação mútua, e o coeficiente Chi-quadrado são alguns exemplos destes métodos que podem ser utilizados para este efeito [52]. Nas experiências realizadas optou-se por utilizar o IG (ver Eq. (5.4)). Este método é amplamente utilizado para seleção de atributos em problemas de filtragem de *spam* [26]. O IG mede a redução da entropia causada pela partição dos exemplos de acordo com os valores do atributo [28]. O valor da entropia para um conjunto de dados S relativamente ao atributo classe que pode ter c valores diferentes é dado por [4]:

$$H(S) = -\sum_{i=1}^{c} P(C_i, S) \cdot log_2(P(C_i, S)) \tag{5.2}$$

$P(C_i, S)$ representa a fração de exemplos do conjunto de dados S que têm classe C_i. Logo num problema de classificação onde existem apenas duas classes possíveis *spam*(c_s) e ham(c_h) o valor da entropia, relativamente à classe, é dado por:

$$H(S) = -P(c_h, S) \cdot log_2(P(c_h, S)) - P(c_s, S) \cdot log_2(P(c_s, S)) \tag{5.3}$$

Para um determinado atributo A no conjunto de dados S, o valor do IG o é dado por:

$$IG(S, A) = H(S) - \sum_{v \in A} \frac{|S_v|}{|S|} H(S_v) \tag{5.4}$$

v representa um valor do atributo A, S_v representa o sub-conjunto de dados de S em que A tem o valor de v, $|S_v|$ representa o número de instâncias em S_v. $|S|$ é o número de instâncias do conjunto de dados. Os valores de IG são calculados para todos os atributos do sub-conjunto de dados de treino. Os m atributos com valor de IG mais elevado são selecionados para o processo de treino do modelo de classificação.

5.3 *Receiver Operating Characteristic*

A curva *Receiver Operating Characteristic* (ROC) é uma representação gráfica muito utilizada para visualizar e avaliar o desempenho de um classificador (ver Figura 5.7). A sua utilização estende-se a diversas áreas de conhecimento como teoria de deteção de sinais, às ciências médicas, e na análise e avaliação do comportamento de sistemas de diagnósticos [20].

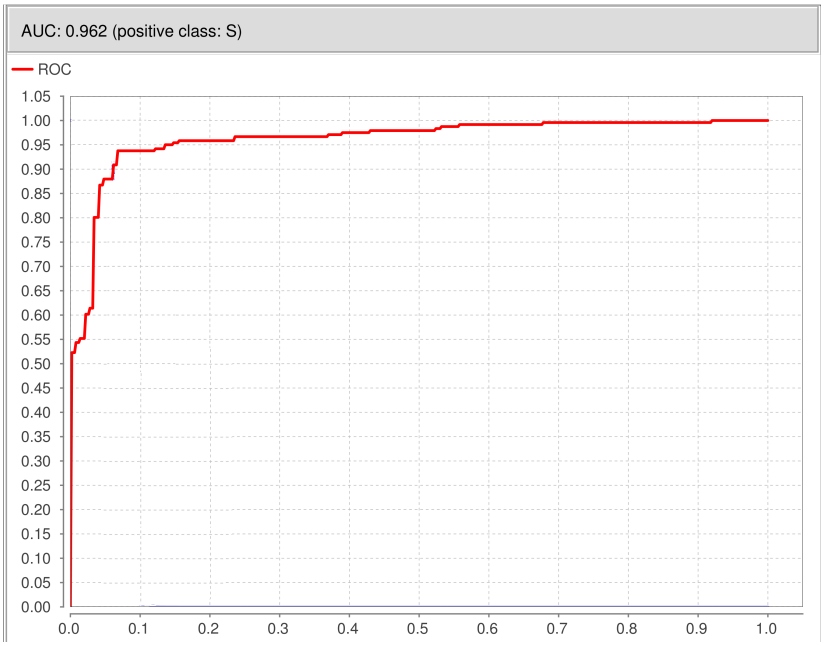


Figura 5.7: Exemplo de curva ROC gerada no *Rapideminer 4.6*.

As curvas ROC são representadas num gráfico de duas dimensões. No eixo das abcissas(x) é representa a Taxa de Falsos Positivos (TFP) e no eixo das ordenadas(y) é representada a Taxa de Verdadeiros Positivos (TVP). A TFP e a TVP são calculadas a partir da matriz de confusão. A matriz de confusão indica o número de instâncias corretamente e incorretamente classificadas para cada uma das duas classes, positiva (p) e negativa (n), de um problema de classificação binária (ver Figura 5.8). No problema de classificação de mensagens de *email* a classe positiva corresponde a mensagens de *spam*, e a classe negativa a mensagens legítimas. Assim para uma dada instância de classificação existem quatro resultados de classificação possíveis:

- **Verdadeiro Positivo (VP)**- A instância é positiva e é classificada como positiva.

- **Verdadeiro Negativo (VN)**- A instância é negativa e é classificada como negativa.
- **Falso Positivo (FP)**- A instância é negativa e é classificada positiva.
- **Falso Negativo (FN)**- A instância é positiva e é classificada como negativa.

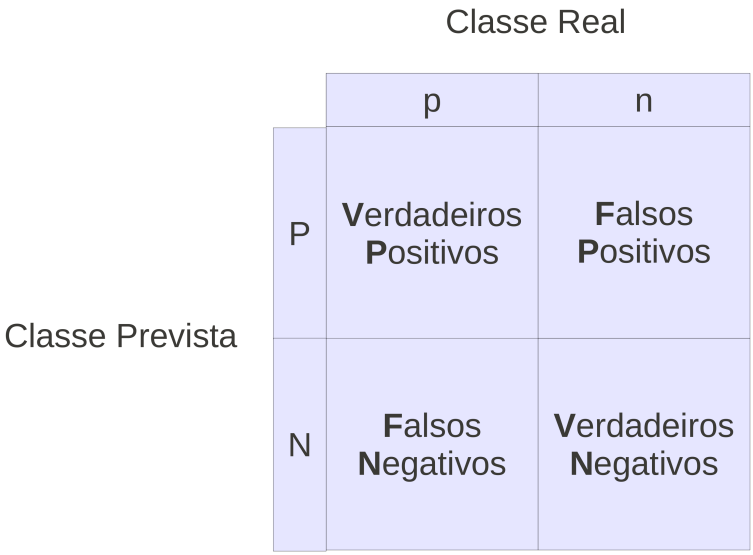


Figura 5.8: Matriz de confusão.

A partir da matriz de confusão a TVP e a TFP é calculada da seguinte forma:

$$TVP = \frac{VP}{TP} \tag{5.5}$$

$$TFP = \frac{FP}{TN} \tag{5.6}$$

sendo TP o número total de positivos, e TN o número total de negativos, dados por:

$$TP = VP + FN \tag{5.7}$$

$$TN = VN + FP \tag{5.8}$$

Para perceber de que forma uma curva ROC permite avaliar o desempenho de um classificador é necessário explicar alguns pontos presentes no gráfico da curva ROC. No ponto (0,0) pode-se interpretar que o classificador prevê uma instância sempre como negativa, ou seja não existem nem falsos positivos nem verdadeiros positivos. Por outro lado o ponto (0,1) indica que todas as instâncias foram corretamente classificadas. Senão vejamos, se $TVP = 1$ significa que $FN = 0$, logo $TP = VP$, o que significa que todas as instâncias positivas foram classificadas como verdadeiros positivos. Por outro lado se $TFP = 0$ significa que $FP = 0$, ou seja nenhuma instância negativa foi classificada como positiva. Assim pode-se concluir que quanto mais a curva ROC de um classificador se aproximar deste ponto melhor é o seu desempenho.

5.4 Métricas de Avaliação dos Filtros *Anti-Spam*

A avaliação do desempenho dos filtros *anti-spam* desenvolvidos sobre os conjuntos de dados *Enron-Ts* e *Enron-Bg* foi realizada analisando as métricas AUC (ver Eq. (5.9)) e TVP@TFP(com TFP=0.05) [10]. A métrica AUC mede a área sob a curva ROC que foi explicada na Secção anterior, assim o desempenho de uma curva ROC é transformado num valor escalar. Desta forma torna-se mais simples a avaliação e comparação entre o desempenho de classificadores. Estatisticamente o AUC representa a probabilidade de o valor em teste, atribuído pela função de decisão de um classificador, de uma instância positiva ,escolhida aleatoriamente, ser superior ao de uma instância negativa [33]. O valor de AUC é calculado pela seguinte expressão [33]:

$$AUC = \int_0^1 roc_x(t)\,\mathrm{d}t. \tag{5.9}$$

com $roc_x(t)$ a representar a função da curva ROC para um classificador x, e onde t representa a TFP.

O TVP@TFP(com TFP=0.05) mede o valor de TVP quando TFP=0.05. Os pontos da curva ROC posicionados na região mais a noroeste do gráfico, são considerados melhores, pois estão próximo do ponto (0,1). Nesta região a TVP possui valores superiores, e a TFP possui valores mais baixos. Assim é benéfico obter valores TVP@TFP elevados para valores baixos de TFP. Além disso, num problema de filtragem de *spam*, onde o custo de classificar uma mensagem legítima como *spam* (FP) é tipicamente superior a classificar uma mensagem de *spam* como legítima (FN), é mais vantajoso ter valores elevados de TVP para valores baixos de TFP. Desta forma pretende-se com esta métrica avaliar os classificadores nesta região específica da curva ROC.

5.5 Configuração das Experiências

Após o desenvolvimento dos filtros *anti-spam* e de constituídos os conjuntos de dados utilizados para avaliar o desempenho dos mesmos, foram realizadas as experiências de classificação para cada um dos utilizadores selecionados das misturas *Enron-Bg* e *Enron-Ts*. Os utilizadores selecionados para realizar a experimentação foram: *martin-p*, *platter-p*, *saibi-e*, *scholtes-d*,*smith-m*. Como já referido na Secção 5.1 para todos os filtros *anti-spam* o algoritmo de aprendizagem máquina utilizado foi o *Multinomial Naive Bayes* com atributos de frequência de termos, descrito na Secção 2.6.3. Como método de seleção de atributos foi utilizado o IG (ver Secção 5.2.2).

Para se realizar uma avaliação mais realista do desempenho dos filtros *anti-spam* adotou-se um método de treino incremental. Para isso as caixas de *email* foram divididas em n lotes de mensagens $l_1 ... l_n$ com K mensagens de *email* contíguas. Nas experiências realizadas definiu-se que $K = 100$. Os filtros são treinados com $l_1 \cup ... \cup l_i$ e avaliados com l_{i+1}. A cada iteração são selecionados os 500 atributos mais relevantes dos lotes de treino de acordo com o valor de IG.

De forma a obter resultados estatisticamente mais consistentes cada experiência foi realizada dez vezes e os resultados aqui apresentados representam os valores médios de AUC e TVP@TFP(TVP=0.05) obtidos em teste. Os respetivos intervalos de confiança também foram calculados de acordo com uma distribuição de *t-student*. Convém referir que na fase de procura foi otimizado apenas o valor de AUC. Portanto a melhor solução da fase de procura, tendo em conta o valor de

AUC, foi utilizada em teste como já explicado. O valor de TVP@TFP(TVP=0.05) foi obtido da mesma solução. Nos filtros que utilizam técnicas de computação evolucionária de procura de atributos foram utilizados os seguintes parâmetros de configuração para os AGEs:

- **Tamanho da população**: 20 indivíduos;
- **Representação dos indivíduos**: Representação de conjunto de *strings* (ver Secção 4.4);
- **Tamanho mínimo do cromossoma de um indivíduo**: 300;
- **Tamanho máximo do cromossoma de um indivíduo**: 400;
- **Método de Seleção**: Seleção por torneio com k=2 (ver Secção 3.4);
- **Operador de Recombinação**: *Random respectful recombination* (ver Secção 4.4);
- **Operador de Mutação**: Mutação aleatória para conjunto de *strings* (ver Secção 4.4);
- **Critério de paragem do AGE**: Número de gerações;
- **Número de gerações**: 100.

No filtro que utiliza partilha de atributos entre utilizadores definiu-se $p = 0.6$. O parâmetro p define a percentagem de indivíduos da população inicial que são provenientes de outros utilizadores (ver Secção 4.5).

De forma a facilitar a leitura nas Secções seguintes os nomes do filtros *anti-spam* são abreviados da seguinte forma: Classificador com Técnicas de Procura Evolucionária com Reinicialização (CTPEcR), Classificador com Técnicas de Procura Evolucionária sem Reinicialização (CTPEsR), Classificador com Técnicas de Procura Evolucionária e Partilha de Atributos (CTPEPA), e Classificador Simples (CS). Inicialmente são demonstrados alguns exemplos de resultados obtidos na fase de procura. De seguida são comparados os resultados de classificação das três versões de filtros *anti-spam* que utilizam técnicas de computação evolucionária. Posteriormente a melhor versão é utilizada para ser comparada com o classificador simples que não utiliza técnicas de computação evolucionária.

5.6 Exemplos de Resultados Obtidos na Fase de Procura

Nesta Secção são apresentados alguns exemplos da evolução dos valores de AUC obtidos no decorrer da fase de procura para os filtros que utilizam técnicas de computação evolucionária de seleção de atributos. Nas Figuras 5.9, 5.10 e 5.11 é demonstrada a evolução do valor de AUC para cada lote de mensagens na fase de procura para a caixa de *email* do utilizador *martin* da mistura *Enron-Bg*. Os valores de AUC representados correspondem ao valor de aptidão do melhor indivíduo de uma geração do AGE. Com o evoluir das gerações são obtidos indivíduos com valores de aptidão mais elevados. O processo de procura termina ao fim do número de gerações definido, que neste caso foi de 100. O indivíduo mais apto da última geração é utilizado, através da sua representação de atributos, para constituir o modelo de classificação final.

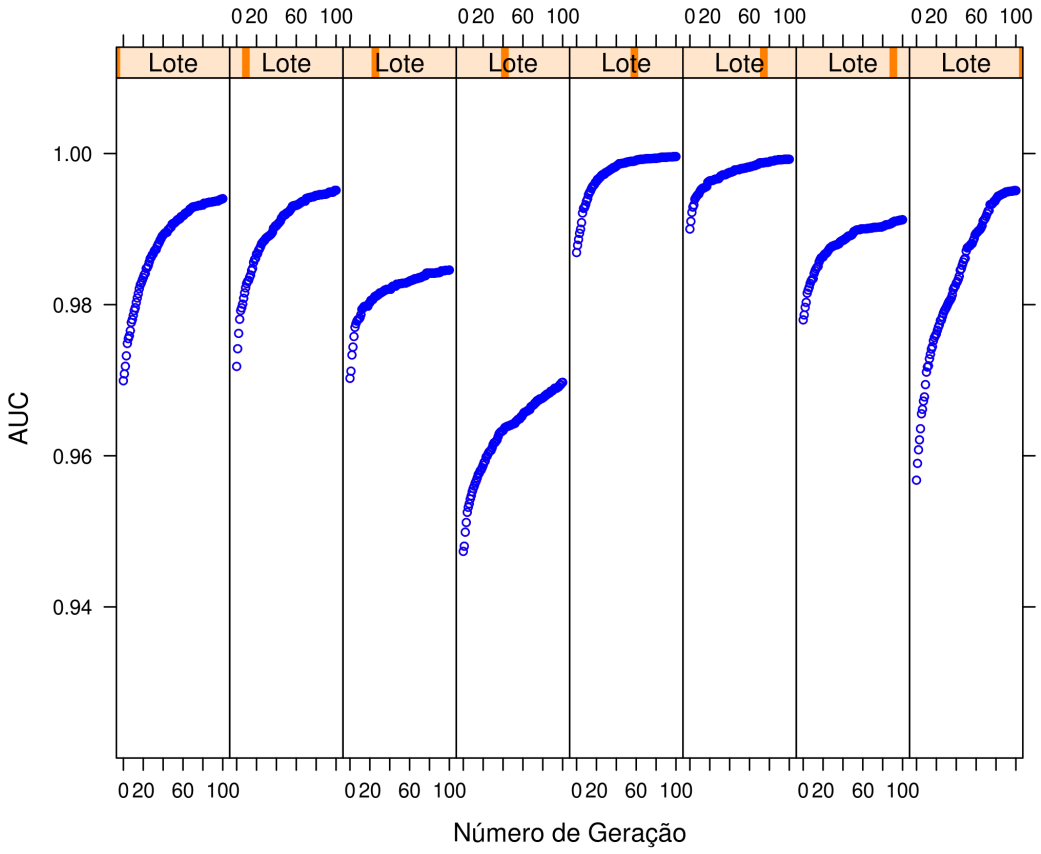


Figura 5.9: Evolução dos valores de AUC na fase de procura para CTPEsR.

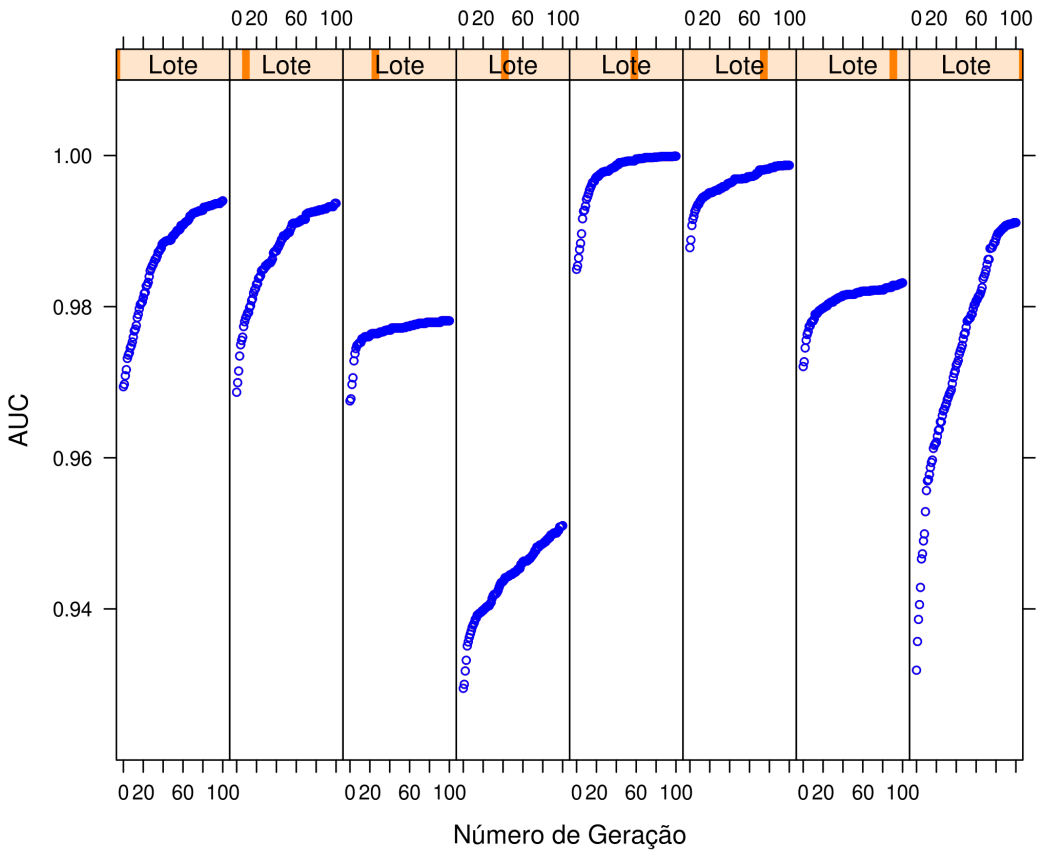


Figura 5.10: Evolução dos valores de AUC na fase de procura para CTPEcR.

Como explicado na Secção 4.5.2 o CTPEsR utiliza os indivíduos da população final de lotes de mensagens anteriores para constituir a população inicial da fase de procura sobre o lote seguinte. Nesse sentido, se observarmos a evolução do AUC entre os filtros CTPEcR (ver Figura 5.10) e CTPEsR (ver Figura 5.9) constata-se que o CTPEsR converge mais rápido, através de um menor número de gerações, para valores de AUC mais elevados. Este facto pode ser observado sobretudo nas primeiras gerações de cada lote de mensagens.

Analisando a Figura 5.11 verifica-se que para o CTPEPA, nas primeiras gerações do AGE, os valores de AUC obtidos são mais elevados quando comparados com as abordagem CTPEcR e CTPEsR. Além disso para esta abordagem, que utiliza partilha de atributos, observa-se que o AUC converge para valores muito elevados, superiores a 0,96 em todos os lotes da caixa de *email* analisada.

Pode-se verificar que os AGEs permitem, através dos seus métodos inspirados

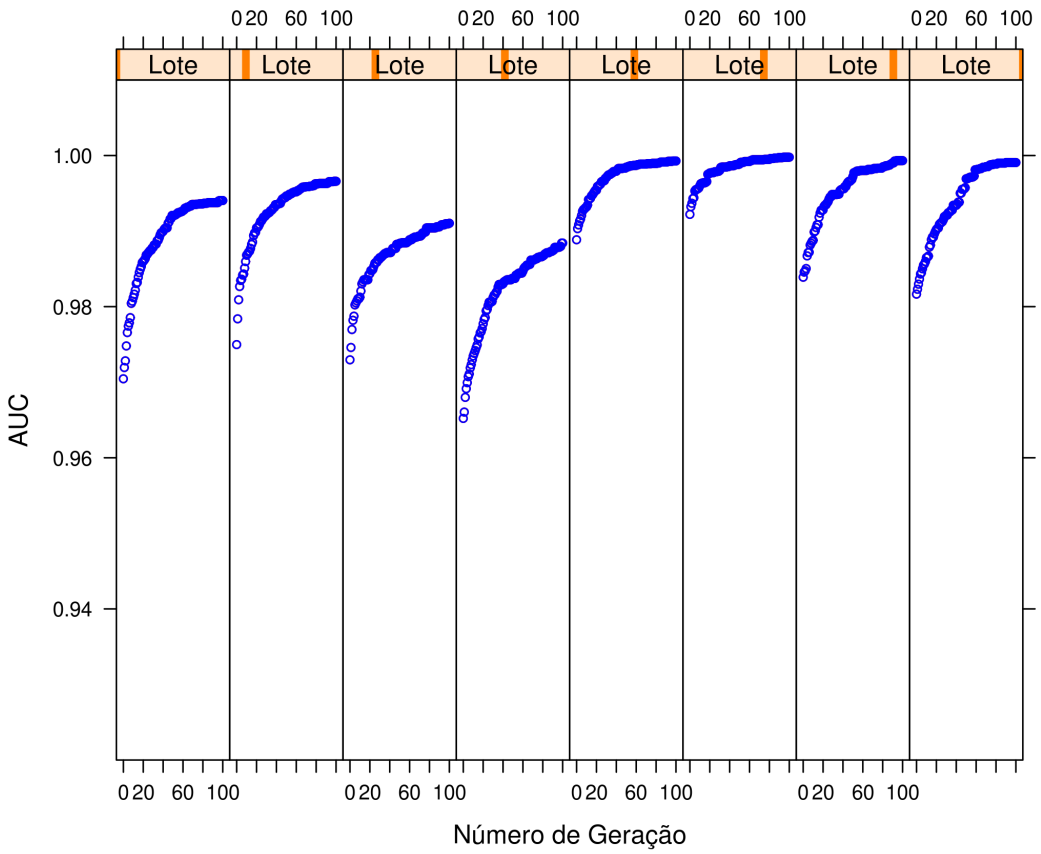


Figura 5.11: Evolução dos valores de AUC na fase de procura para CTPEPA.

em processos de adaptação que ocorrem na natureza, concretizar uma otimização do AUC através da seleção de atributos mais relevantes. Implicitamente também se constata que a seleção de atributos é uma passo importante na construção de filtros *anti-spam* mais fiáveis.

5.7 Resultados Obtidos nos Filtros com Técnicas de Procura Evolucionária

Nesta Secção são apresentados os resultados em teste obtidos para os filtros que utilizam técnicas de computação evolucionária para seleção de atributos. Os filtros *anti-spam* são testados em 10 caixas de *email* pertencentes a duas misturas diferentes: *Enron-Bg* e *Enron-Tel*. Para cada caixa de *email* é apresentado um gráfico com os valores de AUC médios, para cada lote de 100 *emails*, obtidos em cada versão dos filtros que utilizam técnicas de procura evolucionária. Como já

referido os valores médios foram calculados após dez repetições de cada experiência. Para uma melhor leitura dos valores obtidos é apresentada uma tabela que contém os valores de AUC e TVP@TFP(TFP=0.05) médios, e os respetivos intervalos de confiança. Na última linha de cada tabela são mostrado os valores médios de AUC e TVP@TFP(TFP=0.05) calculados sobre todos os lotes de teste.

5.7.1 Mistura *Enron-Bg*

Nesta mistura o classificador com melhores resultados foi o CTPEPA. Em 59%[2] dos lotes foi o que obteve melhores resultados de AUC, e em 41% dos lotes obteve valores de TVP@TFP(TFP=0.05) mais elevados que os classificadores CTPEcR e CTPEsR. Os resultados de classificação do CTPEPA destacam-se sobretudo nos utilizadores *martin*, (Figura 5.12 e Tabela 5.5), *platter*, (Figura 5.13 e Tabela 5.6) e *scholtes*, (Figura 5.15 e Tabela 5.8). Em segundo lugar destaca-se o CTPEcR que em 34% dos lotes superou o CTPEPA e o CTPEsR nos valores de AUC, e em 44% dos lotes obteve os valores mais elevados de TVP@TFP(TFP=0.05). Por último, o CTPEsR alcançou os melhores resultados de AUC em 14%, e de TVP@TFP(TFP=0.05) em 24% dos lotes.

Para o utilizador *saibi* o CTPEcR registou o valor de AUC médio mais elevado no total de classificação da caixa de *email*, apesar do CTPEPA possuir os valores de AUC mais elevados num maior número de lotes de teste (ver Figura 5.14 e Tabela 5.7). O maior equilíbrio entre as abordagens CTPEcR e CTPEPA, a nível do valor de AUC, verificou-se na caixa de *email* do utilizador *smith*. Cada abordagem obteve o melhor desempenho em 4 de 8 lotes, além disso o valor médio de AUC sobre o total de lotes é muito semelhante, 0,942 para o CTPEcR e 0,943 para o CTPEPA (ver Figura 5.16 e Tabela 5.9).

[2]Para alguns lotes os valores de AUC e de TVP@TFP(TFP=0.05) obtidos para cada classificador foram iguais e também os mais elevados, portanto a soma das percentagens referidas para cada métrica pode superar os 100%.

Martin

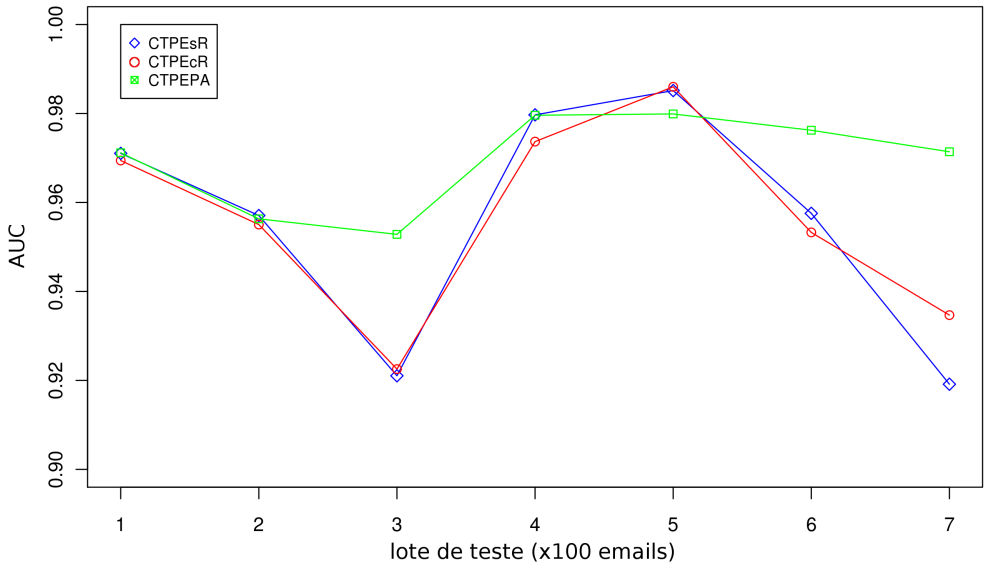


Figura 5.12: Gráfico de valores médios de AUC para *martin-Bg*.

Tabela 5.5: Valores médios de AUC e TPV@TFP(TFP=0.05) para *martin-Bg*.

	AUC ($IC_{95\%}$)			TVP@TFP(TFP=0,05) ($IC_{95\%}$)		
Lote	CTPEcR	CTPEsR	CTPEPA	CTPEcR	CTPEsR	CTPEPA
1	0,969 ± 0,006	**0,971** ± 0,006	**0,971** ± 0,005	0,886 ± 0,017	0,886 ± 0,014	**0,902** ± 0,011
2	0,955 ± 0,011	**0,957** ± 0,008	0,956 ± 0,019	**0,938** ± 0,016	0,932 ± 0,012	0,937 ± 0,024
3	0,923 ± 0,005	0,921 ± 0,006	**0,953** ± 0,012	0,831 ± 0,010	0,842 ± 0,011	**0,862** ± 0,016
4	0,974 ± 0,006	**0,980** ± 0,004	**0,980** ± 0,004	**0,955** ± 0,036	0,952 ± 0,021	0,933 ± 0,020
5	**0,986** ± 0,001	0,985 ± 0,004	0,980 ± 0,005	**0,957** ± 0,000	0,943 ± 0,021	**0,957** ± 0,000
6	0,953 ± 0,008	0,958 ± 0,008	**0,976** ± 0,010	0,771 ± 0,118	0,805 ± 0,090	**0,873** ± 0,120
7	0,935 ± 0,007	0,919 ± 0,009	**0,971** ± 0,007	0,847 ± 0,005	0,751 ± 0,083	**0,896** ± 0,015
Média	0,956 ± 0,006	0,956 ± 0,006	**0,970** ± 0,009	0,884 ± 0,029	0,873 ± 0,036	**0,909** ± 0,029

Platter

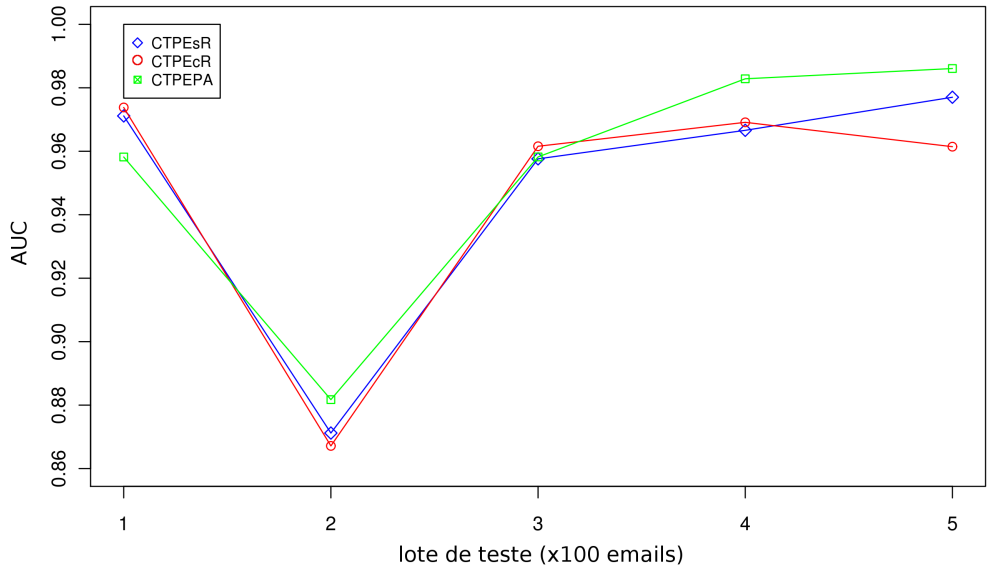


Figura 5.13: Gráfico de valores médios de AUC para *platter-Bg*.

Tabela 5.6: Valores médios de AUC e TPV@TFP(TFP=0.05) para *platter-Bg*.

	AUC ($IC_{95\%}$)			TVP@TFP(TFP=0,05) ($IC_{95\%}$)		
Lote	CTPEcR	CTPEsR	CTPEPA	CTPEcR	CTPEsR	CTPEPA
1	**0,974** ± 0,004	0,971 ± 0,006	0,958 ± 0,012	0,865 ± 0,064	**0,880** ± 0,038	0,803 ± 0,083
2	0,867 ± 0,012	0,871 ± 0,010	**0,882** ± 0,014	0,704 ± 0,061	0,751 ± 0,028	**0,775** ± 0,031
3	**0,962** ± 0,002	0,958 ± 0,008	0,958 ± 0,011	0,877 ± 0,013	**0,902** ± 0,017	0,771 ± 0,106
4	0,969 ± 0,005	0,967 ± 0,004	**0,983** ± 0,005	0,924 ± 0,011	0,890 ± 0,044	**0,938** ± 0,014
5	0,961 ± 0,014	0,977 ± 0,008	**0,986** ± 0,007	0,732 ± 0,098	0,872 ± 0,020	**0,919** ± 0,040
Média	0,947 ± 0,007	0,949 ± 0,007	**0,953** ± 0,010	0,820 ± 0,049	**0,859** ± 0,029	0,841 ± 0,055

Saibi

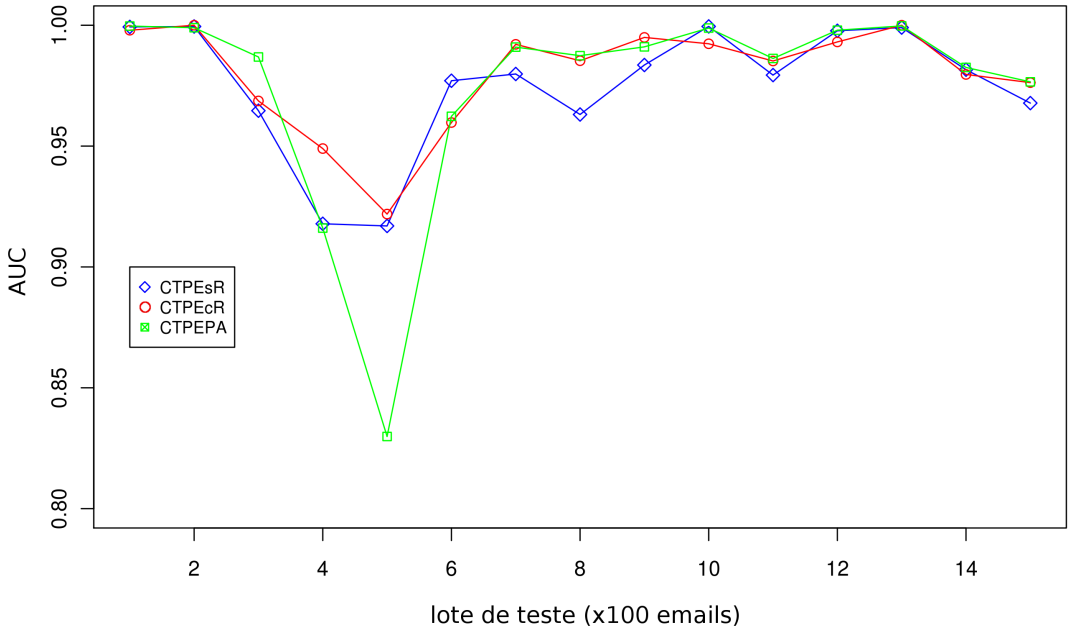


Figura 5.14: Gráfico de valores médios de AUC para *saibi-Bg*.

Tabela 5.7: Valores médios de AUC e TPV@TFP(TFP=0.05) para *saibi-Bg*.

	AUC ($IC_{95\%}$)			TVP@TFP(TFP=0,05) ($IC_{95\%}$)		
Lote	CTPEcR	CTPEsR	CTPEPA	CTPEcR	CTPEsR	CTPEPA
1	0,998 ± 0,002	0,999 ± 0,002	**1,000** ± 0,001	0,995 ± 0,005	**0,999** ± 0,002	**0,999** ± 0,002
2	**1,000** ± 0,000	0,999 ± 0,001	0,999 ± 0,002	**1,000** ± 0,000	0,999 ± 0,002	0,998 ± 0,003
3	0,969 ± 0,011	0,965 ± 0,010	**0,987** ± 0,006	0,969 ± 0,011	0,965 ± 0,010	**0,987** ± 0,006
4	**0,949** ± 0,011	0,918 ± 0,007	0,916 ± 0,029	**0,896** ± 0,022	0,836 ± 0,017	0,866 ± 0,017
5	**0,922** ± 0,010	0,917 ± 0,020	0,830 ± 0,064	0,689 ± 0,017	**0,700** ± 0,042	0,421 ± 0,159
6	0,960 ± 0,016	**0,977** ± 0,007	0,962 ± 0,020	0,919 ± 0,020	**0,939** ± 0,020	0,916 ± 0,041
7	**0,992** ± 0,002	0,980 ± 0,004	0,991 ± 0,003	0,923 ± 0,000	0,888 ± 0,009	**0,935** ± 0,023
8	0,985 ± 0,004	0,963 ± 0,007	**0,987** ± 0,005	**0,948** ± 0,013	0,870 ± 0,033	0,928 ± 0,033
9	**0,995** ± 0,002	0,984 ± 0,004	0,991 ± 0,005	**0,981** ± 0,013	0,925 ± 0,031	0,964 ± 0,025
10	0,992 ± 0,001	**1,000** ± 0,001	0,999 ± 0,001	0,971 ± 0,000	**1,000** ± 0,000	0,994 ± 0,009
11	0,985 ± 0,004	0,979 ± 0,007	**0,986** ± 0,007	**0,973** ± 0,019	0,912 ± 0,030	0,958 ± 0,021
12	0,993 ± 0,003	**0,998** ± 0,003	**0,998** ± 0,002	**1,000** ± 0,000	0,993 ± 0,008	**1,000** ± 0,000
13	**1,000** ± 0,000	0,999 ± 0,002	**1,000** ± 0,000	**1,000** ± 0,000	0,998 ± 0,004	0,998 ± 0,004
14	0,980 ± 0,004	0,982 ± 0,007	**0,983** ± 0,005	**0,978** ± 0,000	0,884 ± 0,087	0,971 ± 0,015
15	0,976 ± 0,005	0,968 ± 0,011	**0,977** ± 0,010	**0,937** ± 0,013	0,887 ± 0,037	0,905 ± 0,032
Média	**0,980** ± 0,005	0,975 ± 0,006	0,974 ± 0,011	**0,945** ± 0,009	0,920 ± 0,022	0,923 ± 0,025

Scholtes

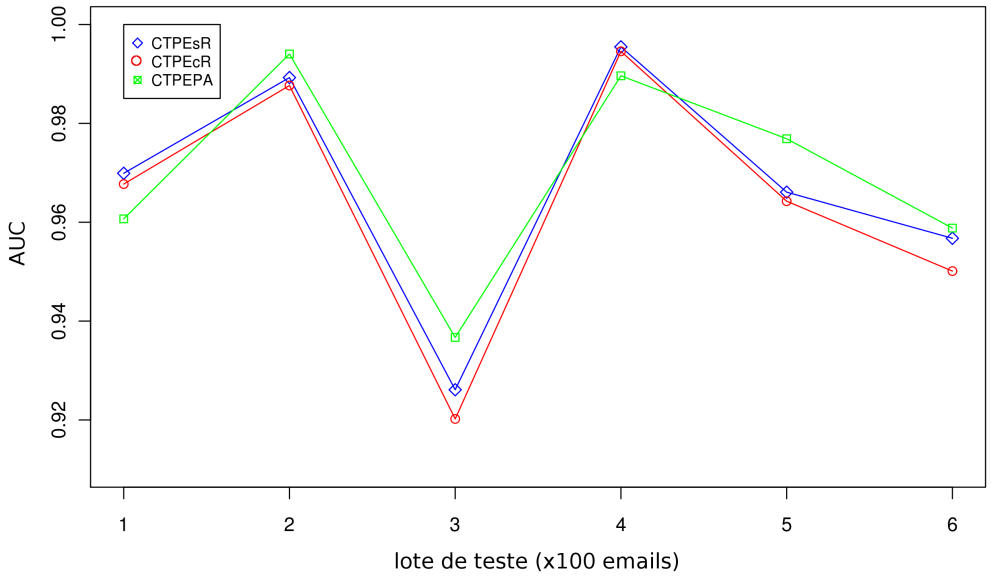


Figura 5.15: Gráfico de valores médios de AUC para *scholtes-Bg*.

Tabela 5.8: Valores médios de AUC e TPV@TFP(TFP=0.05) para *scholtes-Bg*.

	AUC ($IC_{95\%}$)			TVP@TFP(TFP=0,05) ($IC_{95\%}$)		
Lote	CTPEcR	CTPEsR	CTPEPA	CTPEcR	CTPEsR	CTPEPA
1	0,968 ± 0,005	**0,970** ± 0,007	0,961 ± 0,018	0,860 ± 0,019	0,866 ± 0,052	**0,868** ± 0,050
2	0,988 ± 0,005	0,989 ± 0,003	**0,994** ± 0,004	0,969 ± 0,013	0,979 ± 0,004	**0,985** ± 0,014
3	0,920 ± 0,004	0,926 ± 0,006	**0,937** ± 0,009	0,840 ± 0,031	**0,866** ± 0,020	0,792 ± 0,041
4	**0,995** ± 0,003	**0,995** ± 0,002	0,990 ± 0,006	0,948 ± 0,031	**0,983** ± 0,009	0,918 ± 0,053
5	0,964 ± 0,003	0,966 ± 0,005	**0,977** ± 0,007	0,930 ± 0,006	0,923 ± 0,008	**0,936** ± 0,018
6	0,950 ± 0,007	0,957 ± 0,007	**0,959** ± 0,016	0,776 ± 0,071	**0,854** ± 0,052	0,748 ± 0,134
Média	0,964 ± 0,005	0,967 ± 0,004	**0,970** ± 0,010	0,887 ± 0,023	**0,912** ± 0,024	0,875 ± 0,052

Smith

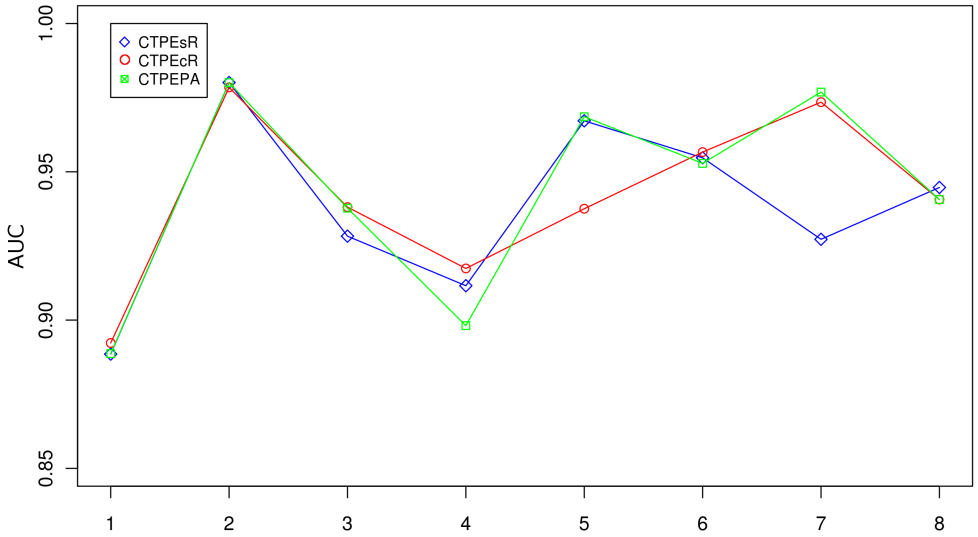


Figura 5.16: Gráfico de valores médios de AUC para *smith-Bg*.

Tabela 5.9: Valores médios de AUC e TPV@TFP(TFP=0.05) para *smith-Bg*.

	AUC ($IC_{95\%}$)			TVP@TFP(TFP=0,05) ($IC_{95\%}$)		
Lote	CTPEcR	CTPEsR	CTPEPA	CTPEcR	CTPEsR	CTPEPA
1	**0,892** ± 0,012	0,889 ± 0,018	0,889 ± 0,010	0,557 ± 0,040	**0,579** ± 0,038	**0,579** ± 0,034
2	0,978 ± 0,002	**0,980** ± 0,004	**0,980** ± 0,002	**0,967** ± 0,014	0,945 ± 0,016	0,945 ± 0,012
3	**0,938** ± 0,005	0,928 ± 0,009	**0,938** ± 0,010	**0,818** ± 0,028	0,776 ± 0,027	0,776 ± 0,051
4	**0,917** ± 0,008	0,912 ± 0,009	0,898 ± 0,017	**0,774** ± 0,021	0,727 ± 0,031	0,727 ± 0,139
5	0,938 ± 0,008	0,967 ± 0,003	**0,969** ± 0,010	0,717 ± 0,040	0,910 ± 0,019	**0,910** ± 0,062
6	**0,957** ± 0,007	0,955 ± 0,009	0,953 ± 0,009	**0,871** ± 0,030	0,868 ± 0,073	0,868 ± 0,091
7	0,973 ± 0,008	0,927 ± 0,016	**0,977** ± 0,009	**0,922** ± 0,025	0,643 ± 0,124	0,643 ± 0,027
8	0,941 ± 0,007	**0,945** ± 0,011	0,941 ± 0,016	**0,893** ± 0,012	0,888 ± 0,030	0,888 ± 0,020
Média	0,942 ± 0,006	0,938 ± 0,010	**0,943** ± 0,010	**0,815** ± 0,042	0,792 ± 0,045	0,792 ± 0,055

Síntese

Analisando o valor médio de AUC calculado sobre todos os lotes de teste verifica-se que o classificador CTPEPA obteve melhores resultados em 4 das 5 caixas de *email* dos utilizadores da mistura *Enron-Bg* (ver Tabela 5.10). Apenas na caixa de *email* do utilizador *saibi* o CTPEcR contrariou esta tendência. Apesar de tudo de

uma forma geral todos os classificadores obtiveram valores de AUC elevados. Relativamente à métrica TPV@TFP(TFP=0.05) o CTPEPA foi superior apenas para o utilizador *martin*. Para *saibi* e *smith* o CTPEcR obteve valores mais elevados. O CTPEsR superou as restantes abordagens nos utilizadores *platter* e *scholtes* a nível de TPV@TFP(TFP=0.05).

Após esta demonstração de resultados pode-se constatar que para a mistura *Enron-Bg* a partilha de atributos foi de facto uma medida benéfica. Os valores de AUC, que representam um critério de avaliação mais abrangente do que a métrica TPV@TFP(TFP=0.05), foram superiores na abordagem que utiliza partilha de atributos entre filtros locais.

Tabela 5.10: Valores médios de AUC e TPV@TFP(TFP=0.05) por caixa de *email* de cada utilizador da mistura *Enron-Bg*.

	AUC ($IC_{95\%}$)			TVP@TFP(TFP=0,05) ($IC_{95\%}$)		
Utilizador	CTPEcR	CTPEsR	CTPEPA	CTPEcR	CTPEsR	CTPEPA
martin	0,956 ± 0,006	0,956 ± 0,006	**0,970 ± 0,009**	0,884 ± 0,029	0,873 ± 0,036	**0,909** ± 0,029
platter	0,947 ± 0,007	0,949 ± 0,007	**0,953** ± 0,010	0,820 ± 0,049	**0,859** ± 0,029	0,841 ± 0,055
saibi	**0,980** ± 0,005	0,975 ± 0,006	0,974 ± 0,011	**0,945** ± 0,009	0,920 ± 0,022	0,923 ± 0,025
scholtes	0,964 ± 0,005	0,967 ± 0,004	**0,970** ± 0,010	0,887 ± 0,023	**0,912** ± 0,024	0,875 ± 0,052
smith	0,942 ± 0,006	0,938 ± 0,010	**0,943** ± 0,010	**0,815** ± 0,042	0,792 ± 0,045	0,792 ± 0,055

5.7.2 Mistura *Enron-Tel*

Nesta Secção são apresentados os resultados das experiências descritas na Secção 5.5, realizadas sobre as caixas de *email* pertencentes à mistura *Enron-Tel*. No total existem 38 lotes de teste nesta mistura, que correspondem ao somatório do número de lotes de teste de cada utilizador. Comparando os valores de AUC obtidos sobre cada lote de cada utilizador observa-se que o classificador CTPEPA obteve em 47% dos casos o valor mais elevado. O classificador CTPEcR alcançou igualmente uma percentagem elevada, em 42% dos lotes conseguiu os melhores valores. Por outro lado o CTPEsR conseguiu obter os resultados mais elevados em apenas 16% dos lotes. Os resultados de AUC obtidos pelo classificador CTPEPA destacam-se sobretudo no utilizador *platter* (ver Figura 5.18 e Tabela 5.12). Para os restantes utilizadores constata-se que existe um equilíbrio ao nível do valor de AUC médio alcançado nas abordagens CTPEPA e CTPEcR, exemplo disso são os resultados obtidos em *scholtes* (ver Figura 5.20 e Tabela 5.14) e *saibi* (ver Figura 5.19 e Tabela 5.13) onde a diferença nos valores de AUC é de apenas 0,001. Não obstante,

excetuando o caso do utilizador, *platter* o CTPEcR demonstrou ser tangencialmente superior. Para a caixa de *email* do utilizador *smith* verifica-se o maior equilíbrio entre as três abordagens com uma diferença de apenas 0,003 entre melhor e pior resultado de AUC. Analisando os valores de TVP@TFP(TFP=0.05) constata-se que o CTPEcR em 52% dos lotes alcançou os melhores resultados. O CTPEPA aparece em segundo lugar com 29% e o CTPEsR em terceiro com 24%.

Martin

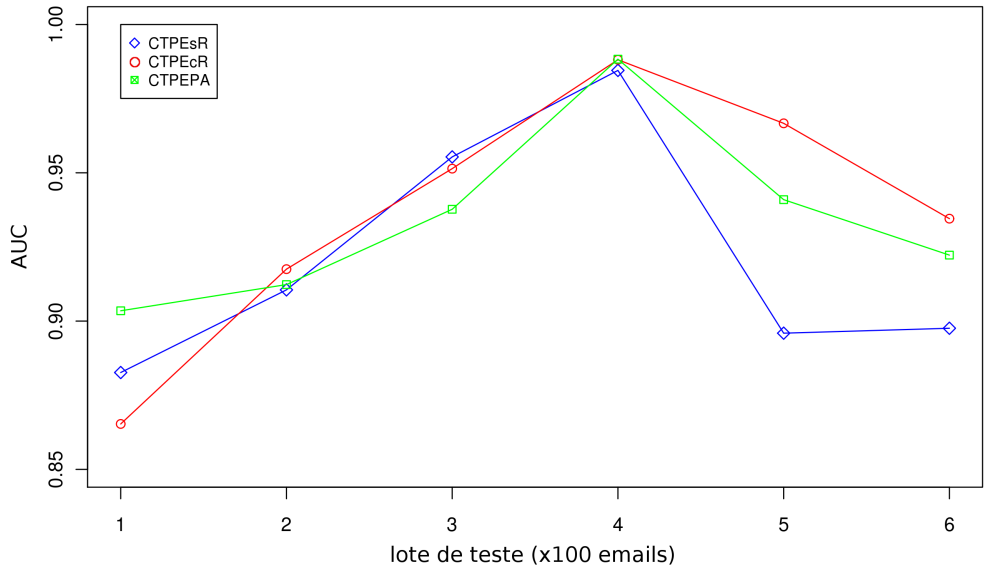


Figura 5.17: Gráfico de valores médios de AUC para *martin-Tel*.

Tabela 5.11: Valores médios de AUC e TPV@TFP(TFP=0.05) para *martin-Tel*.

	AUC ($IC_{95\%}$)			TVP@TFP(TFP=0,05) ($IC_{95\%}$)		
Lote	CTPEcR	CTPEsR	CTPEPA	CTPEcR	CTPEsR	CTPEPA
1	0,865 ± 0,014	0,883 ± 0,025	**0,903** ± 0,022	0,712 ± 0,022	0,734 ± 0,032	**0,764** ± 0,028
2	**0,918** ± 0,007	0,911 ± 0,018	0,912 ± 0,013	0,546 ± 0,045	**0,611** ± 0,081	0,511 ± 0,104
3	0,951 ± 0,012	**0,955** ± 0,013	0,938 ± 0,017	0,618 ± 0,149	**0,632** ± 0,120	0,614 ± 0,111
4	**0,988** ± 0,003	0,985 ± 0,004	**0,988** ± 0,005	0,907 ± 0,037	0,848 ± 0,068	**0,923** ± 0,050
5	**0,967** ± 0,005	0,896 ± 0,024	0,941 ± 0,028	**0,866** ± 0,078	0,443 ± 0,165	0,656 ± 0,207
6	**0,935** ± 0,019	0,898 ± 0,026	0,922 ± 0,033	**0,822** ± 0,071	0,675 ± 0,087	0,679 ± 0,233
Média	**0,937** ± 0,010	0,921 ± 0,018	0,934 ± 0,020	**0,745** ± 0,061	0,657 ± 0,079	0,691 ± 0,105

Platter

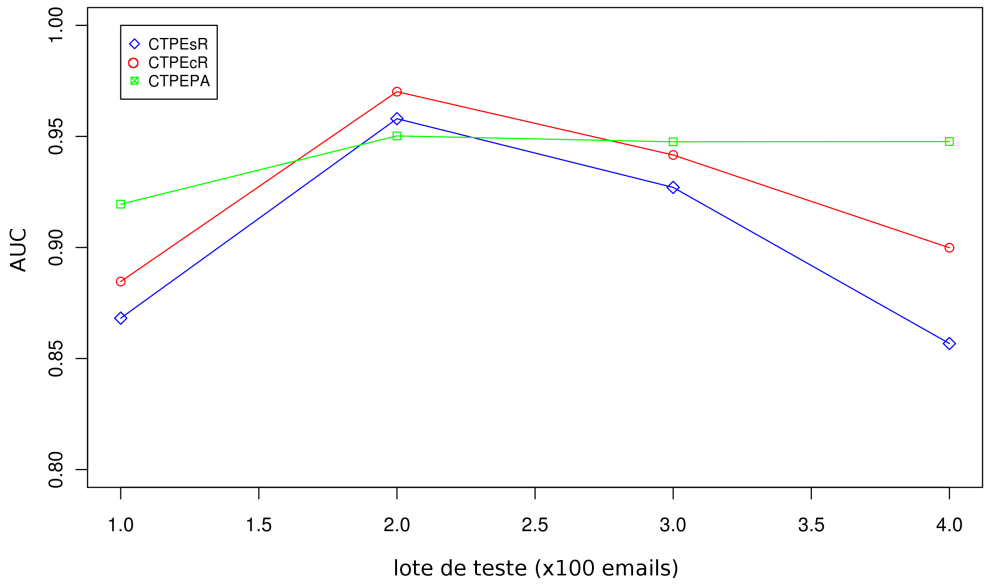


Figura 5.18: Gráfico de valores médios de AUC para *platter-Tel*.

Tabela 5.12: Valores médios de AUC e TPV@TFP(TFP=0.05) para *platter-Tel*.

	AUC ($IC_{95\%}$)			TVP@TFP(TFP=0,05) ($IC_{95\%}$)		
Lote	CTPEcR	CTPEsR	CTPEPA	CTPEcR	CTPEsR	CTPEPA
1	0,885 ± 0,015	0,868 ± 0,020	**0,919** ± 0,014	**0,713** ± 0,073	0,563 ± 0,109	0,699 ± 0,083
2	**0,970** ± 0,011	0,958 ± 0,012	0,950 ± 0,027	0,840 ± 0,046	**0,853** ± 0,053	0,745 ± 0,031
3	0,942 ± 0,005	0,927 ± 0,009	**0,948** ± 0,011	0,767 ± 0,019	0,748 ± 0,026	**0,804** ± 0,106
4	0,900 ± 0,015	0,857 ± 0,023	**0,948** ± 0,018	0,786 ± 0,037	0,733 ± 0,033	**0,833** ± 0,014
Média	0,924 ± 0,012	0,903 ± 0,011	**0,941** ± 0,018	**0,777** ± 0,044	0,724 ± 0,055	0,770 ± 0,059

Saibi

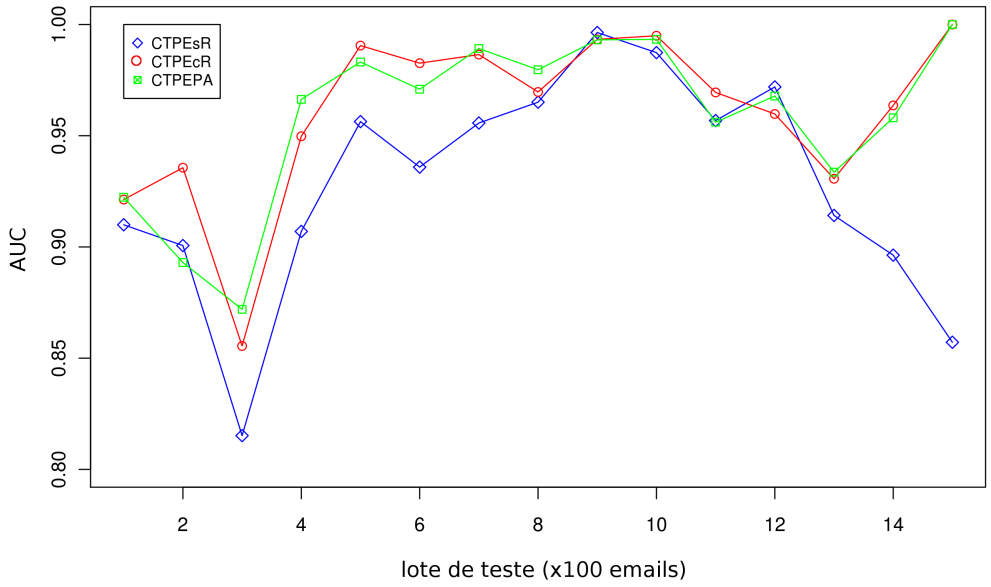


Figura 5.19: Gráfico de valores médios de AUC para *saibi-Tel*.

Tabela 5.13: Valores médios de AUC e TPV@TFP(TFP=0.05) para *saibi-Tel*.

	AUC ($IC_{95\%}$)			TVP@TFP(TFP=0,05) ($IC_{95\%}$)		
Lote	CTPEcR	CTPEsR	CTPEPA	CTPEcR	CTPEsR	CTPEPA
1	0,921 ± 0,031	0,910 ± 0,031	**0,922** ± 0,027	**0,829** ± 0,068	0,814 ± 0,068	0,828 ± 0,068
2	**0,936** ± 0,016	0,901 ± 0,032	0,893 ± 0,038	**0,568** ± 0,126	0,400 ± 0,045	0,418 ± 0,090
3	0,855 ± 0,011	0,815 ± 0,015	**0,872** ± 0,024	**0,629** ± 0,052	0,402 ± 0,064	0,427 ± 0,114
4	0,950 ± 0,021	0,907 ± 0,015	**0,966** ± 0,010	0,866 ± 0,073	0,742 ± 0,069	**0,893** ± 0,094
5	**0,991** ± 0,003	0,956 ± 0,007	0,983 ± 0,005	**0,953** ± 0,033	0,679 ± 0,075	0,869 ± 0,073
6	**0,983** ± 0,007	0,936 ± 0,007	0,971 ± 0,008	**0,806** ± 0,102	0,538 ± 0,080	0,779 ± 0,064
7	0,986 ± 0,004	0,956 ± 0,011	**0,989** ± 0,004	0,905 ± 0,055	0,564 ± 0,125	**0,944** ± 0,034
8	0,970 ± 0,006	0,965 ± 0,007	**0,980** ± 0,007	0,746 ± 0,037	0,722 ± 0,070	**0,880** ± 0,057
9	0,993 ± 0,004	**0,996** ± 0,003	0,993 ± 0,006	**0,994** ± 0,013	0,984 ± 0,029	0,976 ± 0,031
10	**0,995** ± 0,002	0,987 ± 0,003	0,993 ± 0,007	**0,975** ± 0,042	0,907 ± 0,046	**0,975** ± 0,023
11	**0,969** ± 0,007	0,957 ± 0,009	0,956 ± 0,018	**0,919** ± 0,018	0,848 ± 0,088	0,905 ± 0,040
12	0,960 ± 0,010	**0,972** ± 0,008	0,968 ± 0,015	0,923 ± 0,010	0,898 ± 0,050	**0,928** ± 0,037
13	0,931 ± 0,011	0,914 ± 0,017	**0,934** ± 0,015	**0,845** ± 0,014	0,779 ± 0,044	0,809 ± 0,050
14	**0,964** ± 0,010	0,896 ± 0,039	0,958 ± 0,014	0,863 ± 0,035	0,734 ± 0,087	0,817 ± 0,069
15	**1,000** ± 0,000	0,857 ± 0,059	**1,000** ± 0,000	**1,000** ± 0,000	0,650 ± 0,157	**1,000** ± 0,000
Média	**0,960** ± 0,010	0,928 ± 0,018	0,959 ± 0,013	**0,855** ± 0,045	0,711 ± 0,073	0,830 ± 0,056

Scholtes

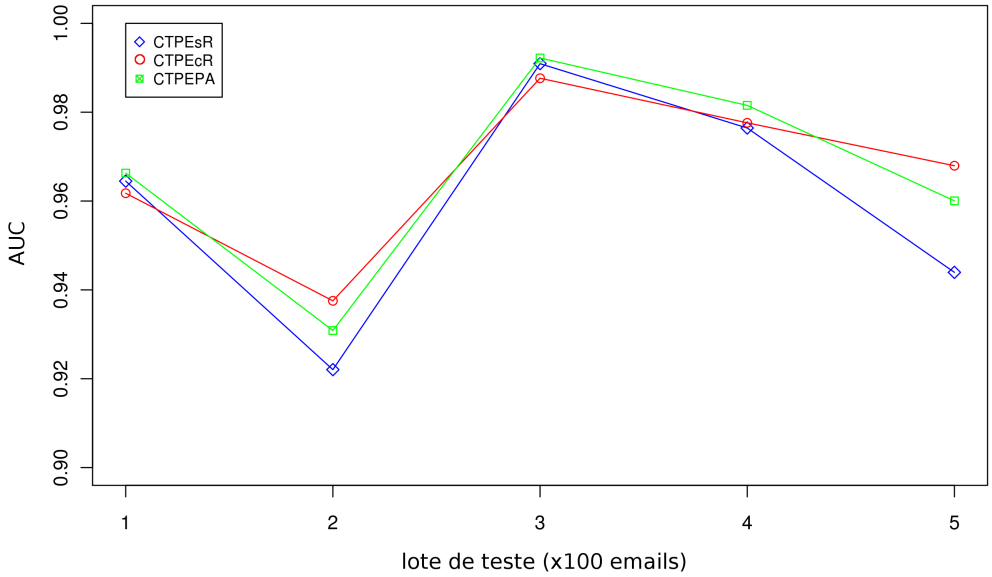


Figura 5.20: Gráfico de valores médios de AUC para *scholtes-Tel*.

Tabela 5.14: Valores médios de AUC e TPV@TFP(TFP=0.05) para *scholtes-Tel*.

	AUC ($IC_{95\%}$)			TVP@TFP(TFP=0,05) ($IC_{95\%}$)		
Lote	CTPEcR	CTPEsR	CTPEPA	CTPEcR	CTPEsR	CTPEPA
1	0,962 ± 0,005	0,964 ± 0,004	**0,966** ± 0,006	0,900 ± 0,026	**0,943** ± 0,017	0,931 ± 0,020
2	**0,938** ± 0,008	0,922 ± 0,008	0,931 ± 0,010	**0,730** ± 0,051	0,692 ± 0,118	0,671 ± 0,074
3	0,988 ± 0,003	0,991 ± 0,005	**0,992** ± 0,005	**0,943** ± 0,015	0,920 ± 0,046	0,917 ± 0,067
4	0,978 ± 0,002	0,976 ± 0,003	**0,982** ± 0,006	**0,941** ± 0,020	**0,941** ± 0,009	0,917 ± 0,044
5	**0,968** ± 0,014	0,944 ± 0,006	0,960 ± 0,016	**0,888** ± 0,051	0,877 ± 0,020	0,879 ± 0,078
Média	**0,967** ± 0,006	0,960 ± 0,005	0,966 ± 0,009	**0,880** ± 0,033	0,875 ± 0,042	0,863 ± 0,057

Smith

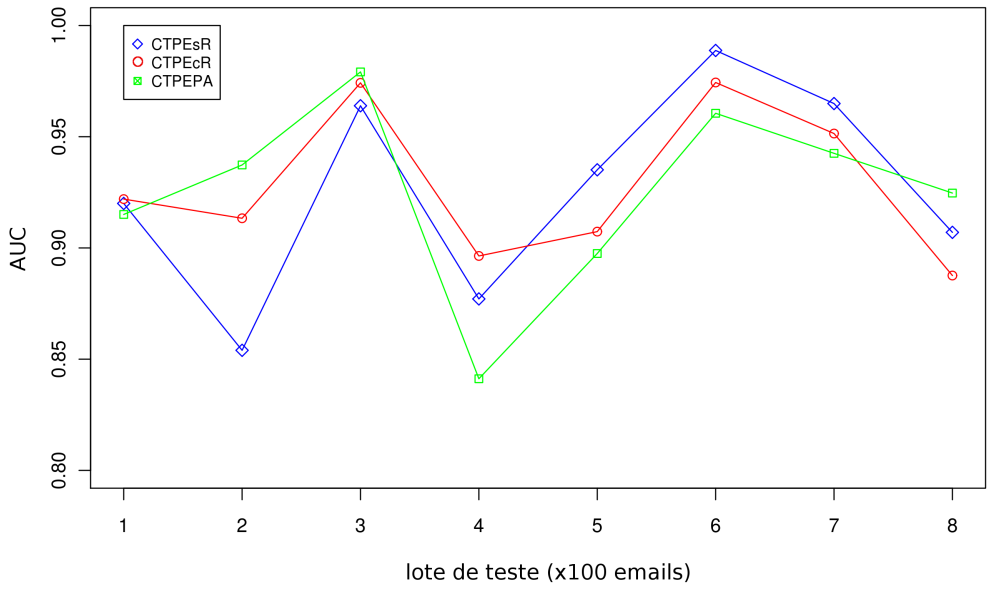


Figura 5.21: Gráfico de valores médios de AUC para *smith-Tel*.

Tabela 5.15: Valores médios de AUC e TPV@TFP(TFP=0.05) para *smith-Tel*.

	AUC ($IC_{95\%}$)			TVP@TFP(TFP =0,05) ($IC_{95\%}$)		
Lote	CTPEcR	CTPEsR	CTPEPA	CTPEcR	CTPEsR	CTPEPA
1	**0,922** ± 0,007	0,920 ± 0,013	0,915 ± 0,015	0,707 ± 0,075	**0,740** ± 0,076	0,667 ± 0,078
2	0,913 ± 0,049	0,854 ± 0,043	**0,937** ± 0,037	**0,725** ± 0,093	0,610 ± 0,074	0,706 ± 0,121
3	0,974 ± 0,011	0,964 ± 0,012	**0,979** ± 0,006	**0,915** ± 0,029	0,873 ± 0,040	0,909 ± 0,020
4	**0,896** ± 0,010	0,877 ± 0,010	0,841 ± 0,059	**0,422** ± 0,100	0,172 ± 0,027	0,250 ± 0,147
5	0,907 ± 0,011	**0,935** ± 0,019	0,898 ± 0,029	0,540 ± 0,057	**0,683** ± 0,076	0,625 ± 0,068
6	0,974 ± 0,005	**0,989** ± 0,005	0,961 ± 0,018	0,861 ± 0,094	**0,945** ± 0,041	0,790 ± 0,132
7	0,951 ± 0,008	**0,965** ± 0,005	0,943 ± 0,018	0,881 ± 0,049	**0,888** ± 0,036	0,715 ± 0,133
8	0,888 ± 0,015	0,907 ± 0,016	**0,925** ± 0,027	0,689 ± 0,034	0,695 ± 0,033	**0,764** ± 0,106
Média	**0,928** ± 0,015	0,926 ± 0,015	0,925 ± 0,026	**0,718** ± 0,066	0,701 ± 0,050	0,678 ± 0,101

Síntese

Apesar do classificador CTPEPA apresentar melhores resultados de AUC para uma maior percentagem do número de lotes (47% contra 42% do classificador CTPEcR), se compararmos os valores médios sobre o total de lotes para cada caixa de *email* constata-se que existe um equilíbrio entre as duas abordagens. Nos utilizadores *martin*, *saibi*, *scholtes* e *smith* a diferença entre o CTPEPA e o CTPEcR é

marginal, sobretudo quando analisados os intervalos de confiança (ver Tabela 5.16). Apenas no utilizador *platter* se verifica alguma vantagem para o CTPEPA. Relativamente aos valores de TVP@TFP(TFP=0,05) pode-se constatar que o CTPEcR foi a melhor abordagem. Assim pode-se concluir que a partilha de atributos entre filtros locais na mistura *Enron-Tel* não foi uma medida relevante para melhorar a assertividade do filtro *anti-spam*.

Tabela 5.16: Valores médios de AUC e TPV@TFP(TFP=0.05) por caixa de *email* de cada utilizador da mistura *Enron-Tel*.

	AUC ($IC_{95\%}$)			TVP@TFP(TFP=0,05) ($IC_{95\%}$)		
Utilizador	CTPEcR	CTPEsR	CTPEPA	CTPEcR	CTPEsR	CTPEPA
martin	**0,937** ± 0,010	0,921 ± 0,018	0,934 ± 0,020	**0,745** ± 0,061	0,657 ± 0,079	0,691 ± 0,105
platter	0,924 ± 0,012	0,903 ± 0,011	**0,941** ± 0,018	**0,777** ± 0,044	0,724 ± 0,055	0,770 ± 0,059
saibi	**0,960 ± 0,010**	0,928 ± 0,018	0,959 ± 0,013	**0,855** ± 0,045	0,711 ± 0,073	0,830 ± 0,056
scholtes	**0,967 ± 0,006**	0,960 ± 0,005	0,966 ± 0,009	**0,880** ± 0,033	0,875 ± 0,042	0,863 ± 0,057
smith	**0,928 ± 0,015**	0,926 ± 0,015	0,925 ± 0,026	**0,718** ± 0,066	0,701 ± 0,050	0,678 ± 0,101

5.8 Comparação de Filtro com Partilha de Atributos e Filtro Simples

Após análise dos resultados de avaliação do desempenho dos filtros que utilizam técnicas de computação evolucionária efetuadas na Secção anterior constata-se que o filtro baseado no classificador CTPEPA, que utiliza partilha de atributos, obteve de uma forma geral melhores resultados de classificação. Na mistura *Enron-Bg* em 59% dos lotes de teste o CTPEPA obteve o melhor valor de AUC contra 34% do CTPEcR. Analisando o AUC médio por utilizador constata-se que o CTPEPA foi a melhor abordagem para 4 de 5 caixas de *email*. Na mistura *Enron-Tel* de acordo com a mesma métrica o CTPEPA apresentou os melhores resultados em 47% dos lotes de teste contra 42% do CTPEcR. Apesar de nesta mistura o CTPEcR ter alcançado os melhores valores médios de AUC para 4 de 5 caixas de *email* a diferença foi tangencial quando comparada com o CTPEPA.

O CTPEcR apresentou melhores valores de TPV@TFP(TFP=0.05) para ambas as misturas *Enron-Bg* e *Enron-Tel* em relação ao CTPEPA, contudo optou-se por utilizar como critério de seleção a métrica AUC. Esta métrica representa um critério de avaliação do filtro mais abrangente que o TPV@TFP(TFP=0.05), uma vez que o AUC calcula a área total sob a curva ROC, enquanto o TPV@TFP(TFP=0.05)

apenas o valor de TPV num único ponto da mesma curva (no ponto TFP=0.05). Portanto nesta Secção são comparados os resultados de classificação entre o CTPEPA e o CS.

Pretende-se neste conjunto de experiências analisar a viabilidade da utilização de técnicas de computação evolucionária para seleção de atributos bem como a partilha de atributos entre filtros locais. Para isso os resultados de classificação do classificador CTPEPA são comparados com os resultados de uma abordagem padrão, o CS (ver Secção 4.3). O CS apenas utiliza o IG para seleção de atributos sobre os lotes de treino $l_1 \cup ... \cup l_i$. Os 500 atributos mais relevantes de acordo com o IG são selecionados para construir o modelo de classificação. A avaliação do classificador é efetuada sobre l_{i+1}.

Os resultados são apresentados de forma semelhante à da Secção anterior. É apresentado um gráfico por cada utilizador de cada mistura onde se comparam os valores de AUC nos filtros CTPEPA e CS obtidos para cada lote de teste. São demonstrados também numa tabela os valores de AUC e TVP@TFP(TFP=0.05) para cada lote de teste e os respetivos intervalos de confiança para o classificador CTPEPA. Como o CS é um filtro determinístico os valores de AUC e TVP@TFP(TFP=0.05) são sempre iguais para o mesmo lote de um conjunto de dados, logo não existe a necessidade do cálculo dos intervalos de confiança. Inicialmente são apresentados os resultados para a mistura *Enron-Bg* e posteriormente para a mistura *Enron-Tel*.

5.8.1 Mistura *Enron-Bg*

Comparando os resultados de classificação verifica-se que para esta mistura o CTPEPA obteve melhores resultados de classificação relativamente ao CS. Em 66% dos lotes de teste o CTPEPA obteve os melhores valores de AUC contra 37% do CS. O CTPEPA obteve melhores resultados nos utilizadores *martin*, *platter*, *scholtes* e *smith*. Para *martin* o CTPEPA obteve valores de AUC mais elevados para todos os lotes de teste (Figura 5.22 e Tabela 5.17). Em três de cinco lotes de teste o CTPEPA foi superior ao CS em *platter* (Figura 5.23 e Tabela 5.18). Para o utilizador *smith* em 6 de 8 lotes de teste o CTPEPA foi superior nos valores de AUC (Figura 5.26 e Tabela 5.21). Por último o CTPEPA obteve melhores resultados de AUC em 50% dos lotes para o utilizador *scholtes* (ver Figura 5.25 e Tabela 5.20). Analisando os valores de TVP@TFP(TFP=0.05) obtidos verifica-se que o CS

obteve os melhores resultados em 56% dos lotes de teste contra 46% do CTPEPA. O destaque vai sobretudo para o utilizador *saibi* no qual o CS obteve os valores de TVP@TFP(TFP=0.05) mais elevados em 11 de 15 lotes de teste (Figura 5.24 e Tabela 5.19).

Martin

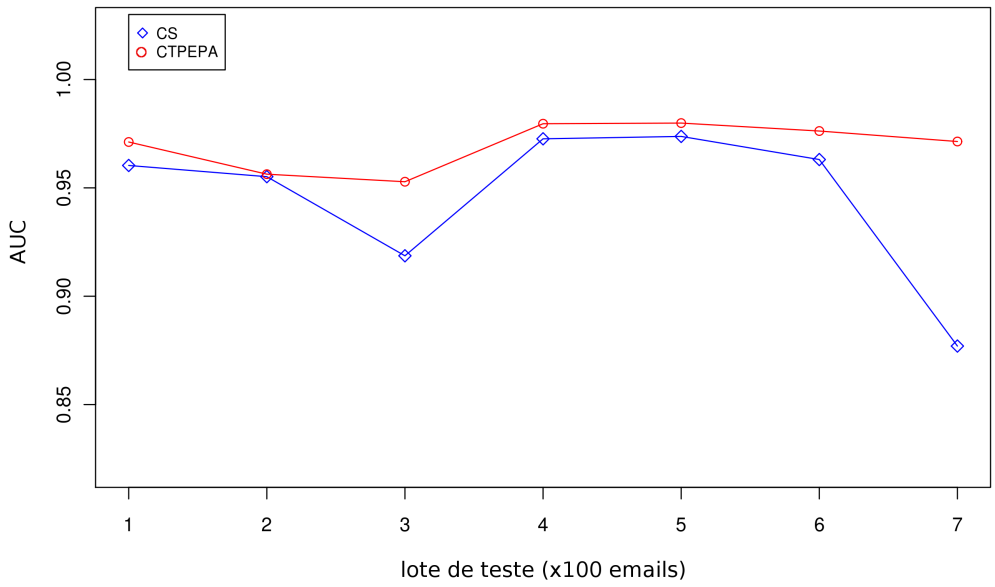


Figura 5.22: Gráfico de valores médios de AUC para *martin-Bg*.

Tabela 5.17: Valores médios de AUC e TPV@TFP(TFP=0.05) para *martin-Bg*.

	AUC		TVP@TFP(TFP=0,05)	
Lote	CTPEPA	CS	CTPEPA	CS
1	**0,971** ± 0,005	0,960	**0,902** ± 0,011	0,833
2	**0,956** ± 0,019	0,955	**0,937** ± 0,024	0,913
3	**0,953** ± 0,012	0,919	**0,862** ± 0,016	0,824
4	**0,980** ± 0,004	0,973	0,933 ± 0,020	**0,970**
5	**0,980** ± 0,005	0,974	**0,957** ± 0,000	**0,957**
6	**0,976** ± 0,010	0,963	0,873 ± 0,120	**0,903**
7	**0,971** ± 0,007	0,877	**0,896** ± 0,015	0,489
Média	**0,970** ± 0,009	0,946	**0,909** ± 0,029	0,841

Platter

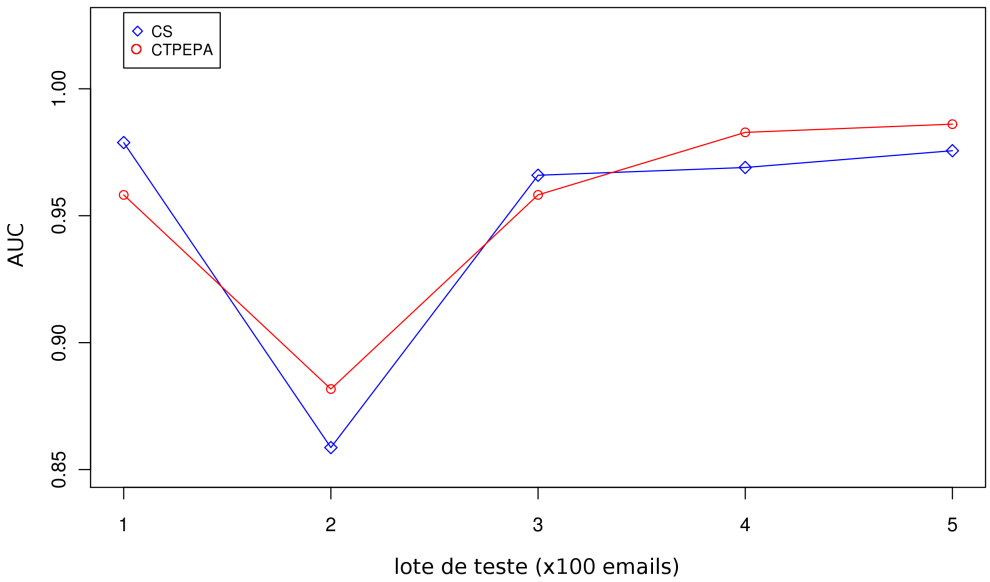


Figura 5.23: Gráfico de valores médios de AUC para *platter-Bg*.

Tabela 5.18: Valores médios de AUC e TPV@TFP(TFP=0.05) para *platter-Bg*.

	AUC		TVP@TFP(TFP=0,05)	
Lote	CTPEPA	CS	CTPEPA	CS
1	0,958 ± 0,012	**0,979**	0,803 ± 0,083	**0,941**
2	**0,882** ± 0,014	0,859	**0,775** ± 0,031	0,760
3	0,958 ± 0,011	**0,966**	0,771 ± 0,106	**0,896**
4	**0,983** ± 0,005	0,969	**0,938** ± 0,014	0,920
5	**0,986** ± 0,007	0,976	**0,919** ± 0,040	0,839
Média	**0,953** ± 0,010	0,950	0,841 ± 0,055	**0,871**

Saibi

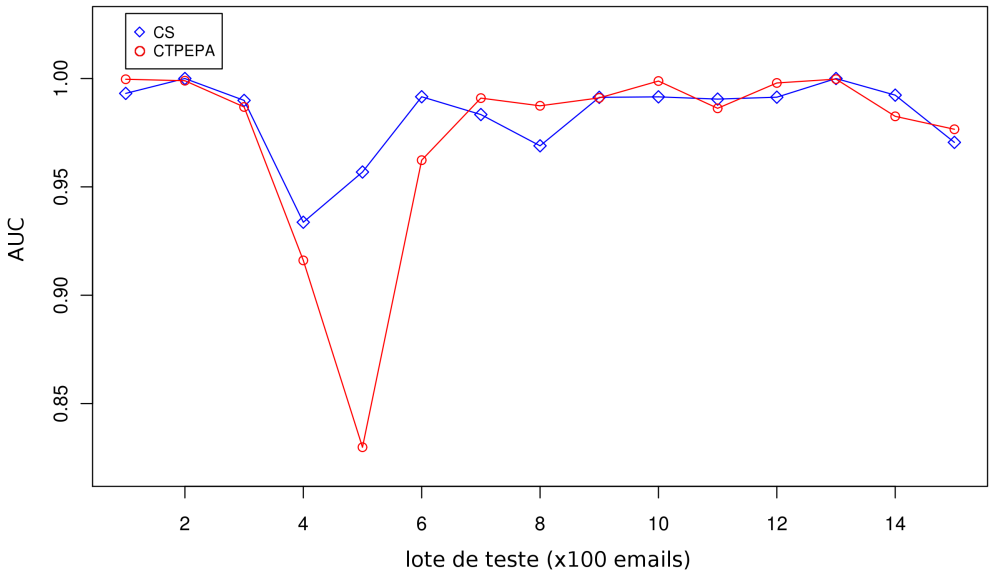


Figura 5.24: Gráfico de valores médios de AUC para *saibi-Bg*.

Tabela 5.19: Valores médios de AUC e TPV@TFP(TFP=0.05) para *smith-Bg*.

	AUC		TVP@TFP(TFP=0,05)	
Lote	CTPEPA	CS	CTPEPA	CS
1	**1,000** ± 0,001	0,993	**0,999** ± 0,002	0,990
2	0,999 ± 0,002	**1,000**	0,998 ± 0,003	**1,000**
3	**0,987** ± 0,006	**0,990**	**0,987** ± 0,006	**0,990**
4	0,916 ± 0,029	**0,934**	0,866 ± 0,017	**0,867**
5	0,830 ± 0,064	**0,957**	0,421 ± 0,159	**0,786**
6	0,962 ± 0,020	**0,992**	0,916 ± 0,041	**0,968**
7	**0,991** ± 0,003	0,983	**0,935** ± 0,023	0,923
8	**0,987** ± 0,005	0,969	**0,928** ± 0,033	0,913
9	0,991 ± 0,005	**0,991**	0,964 ± 0,025	**1,000**
10	**0,999** ± 0,001	0,992	**0,994** ± 0,009	0,971
11	**0,986** ± 0,007	**0,991**	0,958 ± 0,021	**1,000**
12	**0,998** ± 0,002	0,991	**1,000** ± 0,000	**1,000**
13	**1,000** ± 0,000	**1,000**	0,998 ± 0,004	**1,000**
14	**0,983** ± 0,005	**0,992**	0,971 ± 0,015	**0,978**
15	**0,977** ± 0,010	0,971	0,905 ± 0,032	**0,921**
Média	0,974 ± 0,011	**0,983**	0,923 ± 0,026	**0,954**

Scholtes

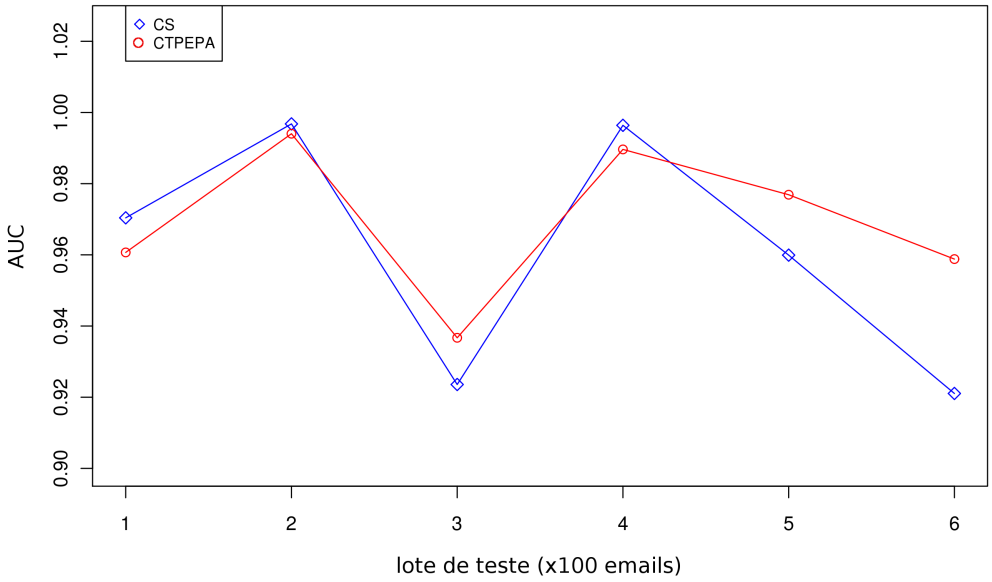


Figura 5.25: Gráfico de valores médios de AUC para *scholtes-Bg*.

Tabela 5.20: Valores médios de AUC e TPV@TFP(TFP=0.05) para *scholtes-Bg*.

	AUC		TVP@TFP(TFP=0,05)	
Lote	CTPEPA	CS	CTPEPA	CS
1	0,961 ± 0,018	**0,970**	0,868 ± 0,050	**0,880**
2	0,994 ± 0,004	**0,997**	**0,985** ± 0,014	0,981
3	**0,937** ± 0,009	0,924	0,792 ± 0,041	**0,878**
4	0,990 ± 0,006	**0,996**	0,918 ± 0,053	**0,979**
5	**0,977** ± 0,007	0,960	**0,936** ± 0,018	0,932
6	**0,959** ± 0,016	0,921	0,748 ± 0,134	**0,761**
Média	**0,970** ± 0,010	0,961	0,875 ± 0,052	**0,902**

Smith

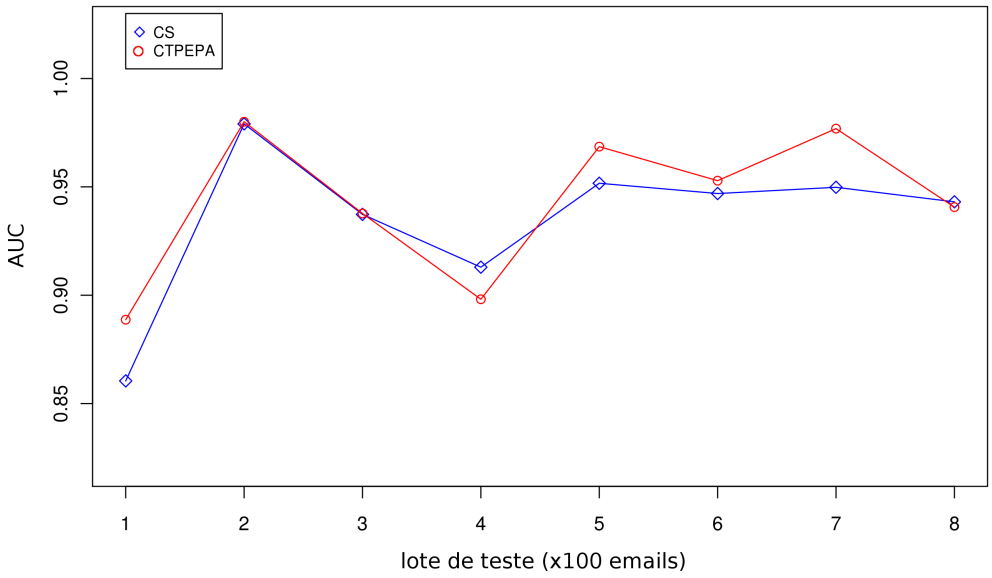


Figura 5.26: Gráfico de valores médios de AUC para *smith-Bg*.

Tabela 5.21: Valores médios de AUC e TPV@TFP(TFP=0.05) para *smith-Bg*.

	AUC		TVP@TFP(TFP=0,05)	
Lote	CTPEPA	CS	CTPEPA	CS
1	**0,889** ± 0,010	0,860	**0,579** ± 0,034	0,500
2	**0,980** ± 0,002	0,979	0,945 ± 0,012	**0,976**
3	**0,938** ± 0,010	0,937	0,776 ± 0,051	**0,848**
4	0,898 ± 0,017	**0,913**	0,727 ± 0,139	**0,791**
5	**0,969** ± 0,010	0,952	**0,910** ± 0,062	0,817
6	**0,953** ± 0,009	0,947	**0,868** ± 0,091	0,857
7	**0,977** ± 0,009	0,950	0,643 ± 0,027	**0,865**
8	0,941 ± 0,016	**0,943**	**0,888** ± 0,020	0,878
Média	**0,943** ± 0,010	0,935	0,792 ± 0,054	**0,817**

Síntese

Na Tabela 5.22 resumem-se os resultados de AUC e TVP@TFP(TFP=0.05) obtidos para os classificadores CTPEPA e CS para o total de lotes de teste de cada caixa de *email*. O CTPEPA alcançou melhores resultados de AUC do que o CS em 4 de 5 caixas de *email*, esta diferença é mais notória nos utilizadores *martin*,

scholtes e *smith*. O CS foi superior apenas para o utilizador *saibi*. Em contrapartida se examinarmos os valores de TVP@TFP(TFP=0.05) verifica-se que o CS obteve melhores resultados.

Através destes resultados comprova-se, para a mistura *Enron-Bg*, que a utilização das técnicas de computação evolucionária para seleção de atributos e também a a partilha de atributos possibilita melhorar a assertividade dos filtros *anti-spam*, nomeadamente o valor de AUC. Porém os valores de TVP@TFP(TFP=0.05) revelam-se inferiores aos obtidos na abordagem simples. Uma possível solução para melhorar esta métrica no CTPEPA seria realizar uma otimização multi-objetivo através de algoritmos evolucionários multi-objetivo (e.g., NSGA[3], SPEA[4]). Assim além do AUC poderia também ser maximizado o TVP@TFP(TFP=0.05).

Tabela 5.22: Valores médios de AUC e TPV@TFP(TFP=0.05) por caixa de *email* de cada utilizador da mistura *Enron-Bg*.

	AUC		TVP@TFP(TFP=0,05)	
Utilizador	CTPEPA	CS	CTPEPA	CS
martin	**0,970 ± 0,009**	0,946	0,909 ± 0,029	0,841
platter	**0,953** ± 0,010	0,950	0,841 ± 0,055	**0,871**
saibi	0,974 ± 0,011	**0,983**	0,923 ± 0,026	**0,954**
scholtes	**0,970** ± 0,010	0,961	0,875 ± 0,052	**0,902**
smith	**0,943** ± 0,010	0,935	0,792 ± 0,054	**0,817**

5.8.2 Mistura *Enron-Tel*

Para a mistura *Enron-Tel* o CS obteve os valores mais elevados de AUC em 55% dos lotes de teste contra 47% do CTPEPA. Contudo analisando os valores de AUC médios obtidos no conjunto de todos os lotes para cada utilizador verifica-se que o CTPEPA foi superior em 4 das 5 caixas de *email* (ver Tabela 5.28). O CS apenas obteve melhor desempenho na caixa de *email* do utilizador *martin* ver (Figura 5.27 e Tabela 5.23). Para os utilizadores *platter* (ver Figura 5.28 e Tabela 5.24) e *smith* (ver Figura 5.31 e Tabela 5.27) os valores de AUC obtidos pelo CTPEPA sobressaem em relação ao CS. Por outro lado para os utilizadores *saibi* (ver Figura 5.29 e Tabela 5.25) e *scholtes* (ver Figura 5.30 e Tabela 5.26) as duas abordagens

[3]NSGA - *Nondominated Sorting Genetic Algorithm.*

[4]SPEA - *Strenght Pareto Evolutionary Algorithm.*

obtêm resultados mais idênticos, apesar de tudo em termos de valor de AUC médio o CTPEPA é ligeiramente superior em ambas as caixas de *email*. Relativamente aos valores de TVP@TFP(TFP=0.05) o CS atinge melhores resultados em 55% dos lotes de teste em oposição a 45% do CTPEPA. Contudo no conjunto de todos os lotes para cada utilizador o CTPEPA é a melhor abordagem em 3 das 5 caixas de *email* considerando esta métrica (ver Tabela 5.28).

Martin

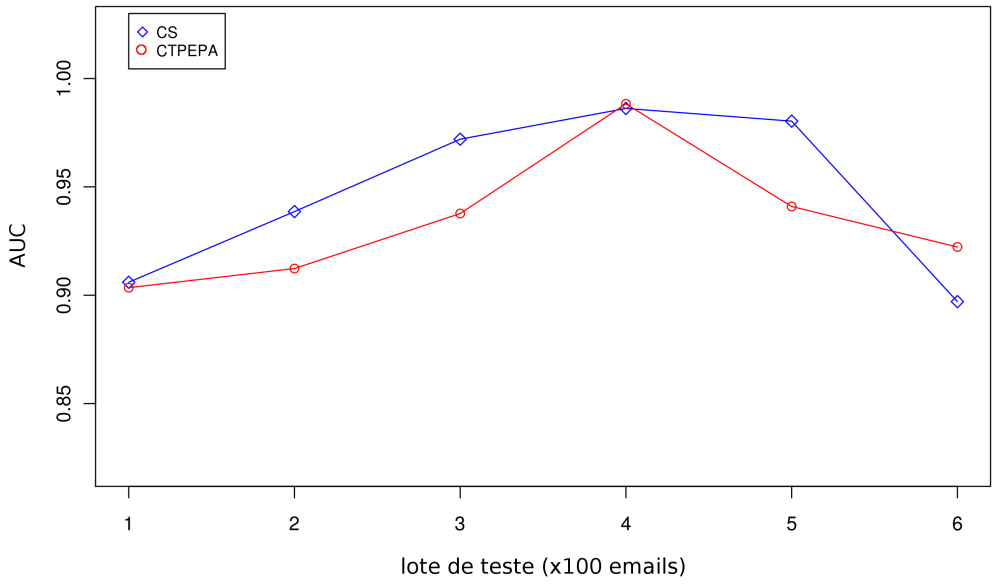


Figura 5.27: Gráfico de valores médios de AUC para *martin-Tel*.

Tabela 5.23: Valores médios de AUC e TPV@TFP(TFP=0.05) para *martin-Tel*.

	AUC		TVP@TFP(TFP=0,05)	
Lote	CTPEPA	CS	CTPEPA	CS
1	0,903 ± 0,022	**0,906**	0,764 ± 0,028	**0,793**
2	0,912 ± 0,013	**0,939**	0,511 ± 0,104	**0,789**
3	0,938 ± 0,017	**0,972**	0,614 ± 0,111	**0,840**
4	**0,988** ± 0,005	0,986	**0,923** ± 0,050	0,900
5	0,941 ± 0,028	**0,980**	0,656 ± 0,207	**0,962**
6	**0,922** ± 0,033	0,897	0,679 ± 0,233	**0,729**
Média	0,937 ± 0,020	**0,947**	0,691 ± 0,105	**0,836**

Platter

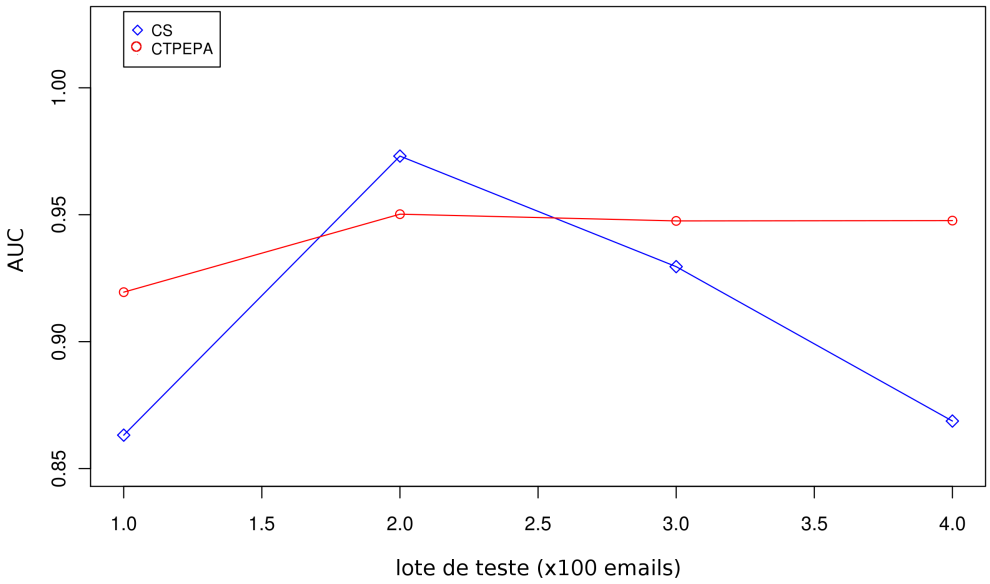


Figura 5.28: Gráfico de valores médios de AUC para *platter-Tel*.

Tabela 5.24: Valores médios de AUC e TPV@TFP(TFP=0.05) para *platter-Tel*.

	AUC		TVP@TFP(TFP=0,05)	
Lote	CTPEPA	CS	CTPEPA	CS
1	**0,919** ± 0,014	0,863	**0,699** ± 0,083	0,633
2	0,950 ± 0,027	**0,973**	0,745 ± 0,031	**0,851**
3	**0,948** ± 0,011	0,930	**0,804** ± 0,106	0,739
4	**0,948** ± 0,018	0,869	**0,833** ± 0,014	0,619
Média	**0,941** ± 0,018	0,909	**0,770** ± 0,059	0,711

Saibi

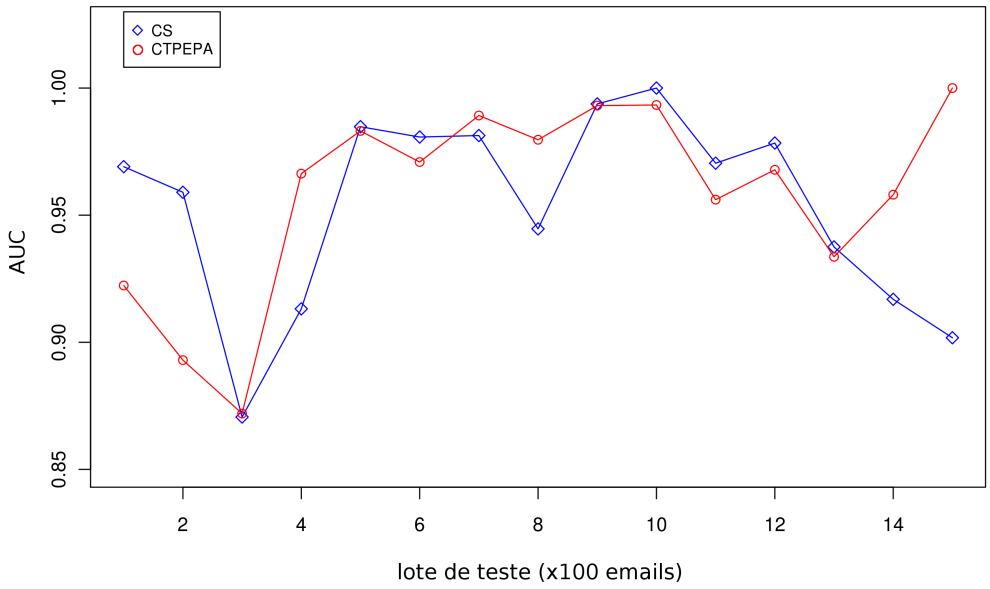


Figura 5.29: Gráfico de valores médios de AUC para *saibi-Tel*.

Tabela 5.25: Valores médios de AUC e TPV@TFP(TFP=0.05) para *saibi-Tel*.

	AUC		TVP@TFP(TFP=0,05)	
Lote	CTPEPA	CS	CTPEPA	CS
1	0,922 ± 0,027	**0,969**	0,828 ± 0,068	**0,928**
2	0,893 ± 0,038	**0,959**	0,418 ± 0,090	**0,500**
3	**0,872** ± 0,024	0,871	0,427 ± 0,114	**0,667**
4	**0,966** ± 0,010	0,913	**0,893** ± 0,094	0,732
5	0,983 ± 0,005	**0,985**	0,869 ± 0,073	**0,961**
6	0,971 ± 0,008	**0,981**	0,779 ± 0,064	**0,872**
7	**0,989** ± 0,004	0,981	**0,944** ± 0,034	0,833
8	**0,980** ± 0,007	0,945	**0,880** ± 0,057	0,634
9	0,993 ± 0,006	**0,994**	0,976 ± 0,031	**1,000**
10	0,993 ± 0,007	**1,000**	0,975 ± 0,023	**1,000**
11	0,956 ± 0,018	**0,970**	0,905 ± 0,040	**0,929**
12	0,968 ± 0,015	**0,978**	0,928 ± 0,037	**0,979**
13	0,934 ± 0,015	**0,937**	0,809 ± 0,050	**0,857**
14	**0,958** ± 0,014	0,917	**0,817** ± 0,069	0,787
15	**1,000** ± 0,000	0,902	**1,000** ± 0,000	0,714
Média	**0,959** ± 0,013	0,953	**0,830** ± 0,056	0,826

Scholtes

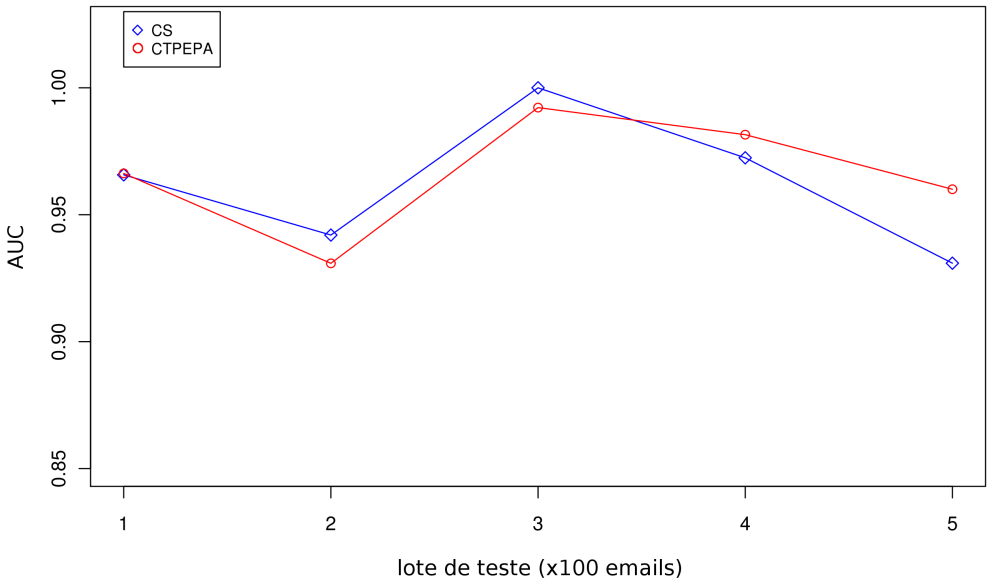


Figura 5.30: Gráfico de valores médios de AUC para *scholtes-Tel*.

Tabela 5.26: Valores médios de AUC e TPV@TFP(TFP=0.05) para *scholtes-Tel*.

	AUC		TVP@TFP(TFP=0,05)	
Lote	CTPEPA	CS	CTPEPA	CS
1	**0,966** ± 0,006	**0,966**	**0,931** ± 0,020	0,886
2	0,931 ± 0,010	**0,942**	0,671 ± 0,074	**0,770**
3	0,992 ± 0,005	**1,000**	0,917 ± 0,067	**1,000**
4	**0,982** ± 0,006	0,972	0,917 ± 0,044	**0,957**
5	**0,960** ± 0,016	0,931	**0,879** ± 0,078	0,769
Média	**0,966** ± 0,009	0,962	0,863 ± 0,057	**0,876**

Smith

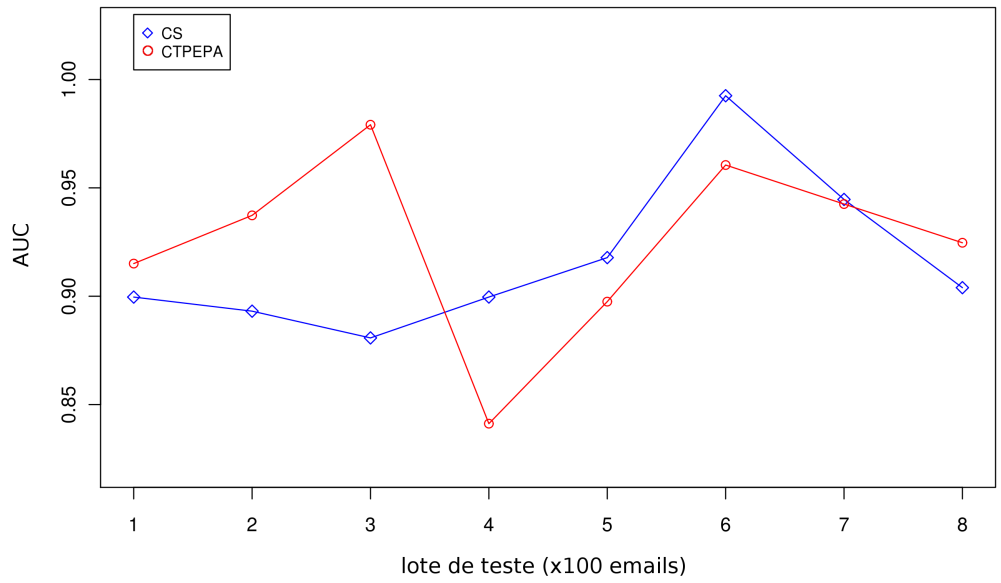


Figura 5.31: Gráfico de valores médios de AUC para *smith-Tel*.

Tabela 5.27: Valores médios de AUC e TPV@TFP(TFP=0.05) para *smith-Tel*.

	AUC		TVP@TFP(TFP=0,05)	
Lote	CTPEPA	CS	CTPEPA	CS
1	**0,915** ± 0,015	0,900	**0,667** ± 0,078	0,600
2	**0,937** ± 0,037	0,893	**0,706** ± 0,121	0,579
3	**0,979** ± 0,006	0,881	**0,909** ± 0,020	0,805
4	0,841 ± 0,059	**0,900**	**0,250** ± 0,147	0,102
5	0,898 ± 0,029	**0,918**	**0,625** ± 0,068	0,600
6	0,961 ± 0,018	**0,993**	0,790 ± 0,132	**0,968**
7	0,943 ± 0,018	**0,945**	0,715 ± 0,133	**0,892**
8	**0,925** ± 0,027	0,904	**0,764** ± 0,106	0,711
Média	**0,925** ± 0,026	0,917	**0,678** ± 0,101	0,657

Síntese

Examinando os valores médios de AUC obtidos para mistura *Enron-Tel* verifica-se mais uma vez que o CTPEPA foi superior ao CS em 4 de 5 caixas de *email* (ver Figura 5.28). A diferença é mais significante para os utilizadores *platter* e *smith*. Apenas para a caixa de *email* do utilizador *martin* o CS alcançou melhor valor de

AUC. Relativamente à métrica TPV@TFP(TFP=0.05) o CTPEPA foi superior ao CS em 3 de 5 caixas de *email*.

Novamente, tal como aconteceu para a mistura *Enron-Bg*, comprova-se que para mistura *Enron-Tel* a utilização das técnicas de computação evolucionária para seleção de atributos e também a partilha de atributos entre filtros locais permite melhorar a assertividade do filtro *anti-spam*.

Tabela 5.28: Valores médios de AUC e TPV@TFP(TFP=0.05) por caixa de *email* de cada utilizador da mistura *Enron-Tel*.

	AUC		TVP@TFP(TFP=0,05)	
Utilizador	CTPEPA	CS	CTPEPA	CS
martin	0,937 ± 0,020	**0,947**	0,691 ± 0,105	**0,836**
platter	**0,941** ± 0,018	0,909	0,770 ± 0,059	**0,711**
saibi	**0,959** ± 0,013	0,953	**0,830** ± 0,056	0,826
scholtes	**0,966** ± 0,009	0,962	0,863 ± 0,057	**0,876**
smith	**0,925** ± 0,026	0,917	**0,678** ± 0,101	0,657

5.9 Sumário

Neste Capítulo foram descritas as experiências realizadas de modo a comparar os diferentes filtros *anti-spam* desenvolvidos. Inicialmente foi descrita de que forma foram constituídos os conjuntos de dados, que representam as caixas de *email* utilizadas avaliar o desempenho dos filtros *anti-spam*, e quais as caixas de *email* selecionadas. No final desde processo foram selecionadas 10 caixas de *email* pertencentes a cinco utilizadores do repositório *Enron*. Os utilizadores selecionados foram: *martin*, *platter*, *saibi*, *scholtes* e *smith*. Cada utilizador possui duas caixas de *email* com misturas de *spam* provenientes de fontes diferentes. A utilização de duas misturas diferentes (*Enron-Bg* e *Enron-Tel*) permite efetuar uma análise mais conclusiva e imparcial dos filtros *anti-spam* desenvolvidos.

As métricas escolhidas para avaliar o desempenho dos filtros *anti-spam* foram o AUC e TVP@TFP(TFP=0.05), estas foram explicadas na Secção 5.4. Posteriormente foram comparados os três filtros que utilizam técnicas de computação evolucionária de procura de atributos. No domínio destes classificadores o que utiliza partilha de atributos (CTPEPA) foi de uma forma geral o que apresentou melhores

resultados, considerando a métrica AUC. Não obstante todos estes classificadores apresentaram resultados de classificação bastante satisfatórios.

A superioridade do CTPEPA foi mais evidente para as caixas de *email* da mistura *Enron-Bg*. Para esta mistura o CTPEPA obteve melhores resultados de AUC em 4 de 5 caixas de *email*. Neste sentido a partilha de atributos foi uma medida benéfica para os utilizadores da mistura *Enron-Bg*. Para a mistura *Enron-Tel* houve um equilíbrio entre o CTPEPA e o CTPEcR[5], os valores AUC obtidos em teste foram bastante semelhantes. Assim para a mistura *Enron-Tel* a partilha de atributos não foi relevante para melhorar a assertividade dos filtros *anti-spam* locais. No conjunto das experiências realizadas o CTPEsR[6] foi a abordagem que pior desempenho demonstrou.

De forma a analisar a viabilidade da utilização das técnicas de computação evolucionária de procura e também a partilha de atributos o CTPEPA foi comparado ao CS, um filtro *anti-spam* padrão que utiliza apenas o IG como método de seleção de atributos (ver Secção 5.8). Após analise dos resultados obtidos constatou-se que no total das duas misturas *Enron-Bg* e *Enron-Tel* o CTPEPA obteve valores de AUC superiores em 8 de 10 caixas de *email*. Relativamente aos valores de TVP@TFP(TFP=0.05) o CS foi superior em 6 de 10 caixas de *email*.

[5]CTPEcR - Classificador com Técnicas de Procura Evolucionária com Reinicialização. Este classificador ou filtro não utiliza partilha de atributos entre filtros locais.

[6]CTPsR - Classificador com Técnicas de Procura Evolucionária sem Reinicialização, uma abordagem diferente do CTPscR (ver Secção 4.5.2).

Capítulo 6

Conclusão

6.1 Síntese

De modo a contextualizar a problemática do *spam* que afeta atualmente o serviço de correio eletrónico, foi inicialmente efetuado neste documento um enquadramento sobre as motivações, metodologias e técnicas utilizadas pelos *spammers*. Por outro lado foram identificadas e explicadas as medidas e técnicas de controlo já existentes contra esta atividade.

Após realizado o estado da arte relativo ao fenómeno do *spam*, foi esclarecida a lógica e a metodologia subjacente aos AGEs, e de que forma estes algoritmos podem ser utilizados para resolver problemas relacionados com otimização, procura e aprendizagem. Estes algoritmos tiveram um papel central nas soluções desenvolvidas, nomeadamente na procura pelos atributos mais relevantes num conjunto de dados que permitam maximizar a assertividade de um filtro *anti-spam*.

No Capítulo 4 foram explicados os filtros *anti-spam* desenvolvidos, que se baseiam no conteúdo das mensagens de *email* e utilizam algoritmos de aprendizagem máquina para classificar as mensagens. Para facilitar a implementação dos filtros *anti-spam* neste projeto foi desenvolvida uma nova representação de soluções no contexto dos AGE designada de representação conjunto de *strings*. Três versões dos filtros desenvolvidos utilizam técnicas de computação evolucionária de seleção de atributos. Uma das versões utiliza a partilha de atributos entre filtros locais. Foi também desenvolvido um classificador simples que utiliza apenas o IG como método de seleção de atributos. Esta última versão foi desenvolvida com o intuito de comparar os resultado obtidos relativamente ao filtro que utiliza técnicas de procura

evolucionária e partilha de atributos.

Por último no Capítulo 5 são demonstrados e analisados os resultados de classificação obtidos através das métricas AUC e TPV@TFP(TFP=0.05) para cada um dos filtros *anti-spam* desenvolvidos. A experimentação foi realizada sobre dez caixas de *email* pertencentes a um grupo de cinco utilizadores, para cada utilizador foram constituídas duas caixas de *email* com misturas de *spam* provenientes de origens diferentes.

6.2 Discussão

No decorrer deste trabalho foi realizado um estudo empírico da utilização de técnicas de computação evolucionária para seleção de atributos com o objetivo de constituir, através de algoritmos de aprendizagem máquina, modelos de classificação mais assertivos no contexto dos processos de filtragem *anti-spam*. Nesse sentido foram desenvolvidos três filtros que utilizam abordagens diferentes de um método *wrapper* de seleção de atributos baseado em técnicas de computação evolucionária. Uma destas abordagens consiste num filtro colaborativo que permite a partilha de atributos entre filtros locais. De forma a analisar a viabilidade da utilização destas técnicas foi também desenvolvido um classificador padrão que utiliza apenas o IG como método de seleção de atributos. Todas as abordagens se baseiam-se no conteúdo da mensagem de *email* para a descriminarem como *spam* ou legítima, para isso foi utilizado um classificador *Bayesiano*, o *Multinomial Naive Bayes* (ver Secção 2.6.3).

Através das experiências realizadas verificou-se que no contexto dos filtros que utilizam técnicas de computação evolucionária a partilha de atributos pode ser uma medida benéfica para melhorar a assertividade do filtro *anti-spam*. Este melhoramento foi notório para as caixas de *email* da mistura *Enron-Bg*. Em 4 das 5 caixas de *email* testadas a abordagem colaborativa obteve melhores resultados que os filtros locais. Por outro lado, para a mistura *Enron-Tel* constatou-se que a partilha de atributos não foi uma medida relevante. Uma das razões associada a este facto pode estar relacionada com a origem desta mistura. A mistura *Enron-Tel* utiliza *spam* proveniente apenas de uma caixa de *email*, logo a partilha de atributos poderá não ser uma medida tão vantajosa. Por outro lado o *spam* proveniente do arquivo público *Bruce Guenter* é obtido a partir de diferentes caixas de *email*. Portanto

acreditamos que a mistura *Enron-Bg* é mais realista e adequa-se mais a avaliação dos filtros *anti-spam*, sobretudo em abordagens colaborativas.

Quando comparados os resultados de classificação do filtro colaborativo que utiliza técnicas de computação evolucionária de seleção de atributos e o classificador simples pode-se constatar que o filtro colaborativo obteve melhores resultados. Em 8 de 10 caixas de *email*, no total das duas misturas, esta abordagem obteve valores superiores de AUC. Assim comprova-se que a seleção de atributos é um passo importante para obter filtros mais assertivos, e que as técnicas de computação evolucionária podem ser utilizadas para este fim. Em acréscimo prova-se também que a partilha de atributos entre filtros locais pode ser uma medida benéfica.

Relativamente aos valores de TPV@TFP(TFP=0.05) obtidos observou-se que o filtro simples foi mais vantajoso que o filtro colaborativo. Uma solução possível para melhorar esta métrica seria a sua inclusão na função de avaliação de procura pelo melhor conjunto de atributos. Para isso poderia ser utilizado um algoritmo evolucionário multi-objetivo. Assim na fase de procura poderia ser maximizado o valor de TPV@TFP(TFP=0.05) além da métrica AUC.

O filtro colaborativo é computacionalmente mais exigente uma vez que são utilizados algoritmos evolucionários de procura através de uma abordagem *wrapper* na fase de procura dos melhores atributos. Esta é uma desvantagem relativamente ao classificador simples que utiliza apenas o IG como método de seleção de atributos.

A nível de segurança, a abordagem colaborativa proposta é menos sensível a problemas de privacidade quando comparada a outros métodos colaborativos, uma vez que são apenas trocados atributos entre utilizadores e não mensagens completas. Em acréscimo, no método proposto não existe a possibilidade de correlacionar os atributos partilhados com a classe a que as mensagens pertencem, ao contrário de uma abordagem que utiliza partilha de filtros entre utilizadores, como proposto em [13].

Por último, importa referir que no decorrer deste trabalho de mestrado, foi submetido o seguinte artigo: R. Vaz, P. Cortez, M. Rocha, M. Rio and P. Sousa. An Email Spam Filtering Colaborative Plataform with Evolutionary Algorithms for Feature Selection, 9th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Rome, INSTICC 2012.

6.3 Trabalho Futuro

Visto os filtros *anti-spam* desenvolvidos serem bastante parametrizáveis, como trabalho futuro aconselharia-se a fazer experiências mais exaustivas com os filtros desenvolvidos, utilizando algoritmos de aprendizagem diferentes e diversos critérios de seleção de atributos. Desta forma seria possível avaliar de forma mais abrangente as vantagens da introdução das técnicas de computação evolucionária na seleção de atributos para a filtragem *anti-spam*.

Outro passo importante para a consolidar o trabalho aqui apresentado seria a integração do filtro *anti-spam* desenvolvido que utiliza técnicas de computação evolucionaria e partilha de atributos numa aplicação cliente do serviço de correio eletrónico. Para isso poderia ser utilizado um modelo P2P, onde o cliente treinava o seu filtro *anti-spam* local e partilhava os atributos mais relevantes com outros utilizadores pertencentes ao mesmo grupo de partilha. Desta forma poderia ser avaliado o desempenho do filtro *anti-spam* em ambiente real.

Um aspecto interessante relacionado com a ideia anterior seria desenvolver uma forma inteligente de definir grupos de partilha. Desta forma os grupos de partilha seriam formados por utilizadores com perfis semelhantes, o que possibilitava uma redução do *concept drift* entre as caixas de *email* dos utilizadores que por sua vez tornaria mais vantajosa a partilha de atributos. A criação de diversos grupos de partilha teria também vantagens em termos de escalabilidade.

A exigência computacional dos filtros *anti-spam* que utilizam técnicas de computação evolucionária de procura pode ser considerável quando comparada com a abordagem simples. Seria importante que este esforço computacional fosse reduzido na aplicação cliente. Uma possibilidade seria disponibilizar um serviço designado para o efeito, que implementaria a fase de procura dos atributos mais relevantes, que corresponde à fase mais exigente do processo de filtragem que utiliza técnicas de computação evolucionária. Outra possibilidade seria a aplicação correr em *background* durante certos períodos de tempo ao longo do tempo de vida da *mailbox*.

Os métodos desenvolvidos neste trabalho poderiam também ser utilizadas numa ferramenta de análise da evolução e das tendências dos conceitos mais correlacionados com mensagens de *spam* de forma a melhor representar o perfil e as características destas mensagens.

Bibliografia

[1] Directive 2002/58/EC. Official journal of the european communities. Technical report, L 201/37, 2002.

[2] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, and Constantine D. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 160–167, New York, NY, USA, 2000. ACM.

[3] Ion Androutsopoulos, Georgios Paliouras, and Eirinaios Michelakis. Learning to filter unsolicited commercial e-mail. Technical report, National Centre for Scientific Research "Demokritos", 2004.

[4] Nf Ayan. Using information gain as feature weight. In *Proceedings of the 8th Turkish Symposium on Artificial*, 1999.

[5] T. Ayodele, C.A. Shoniregun, and G.A. Akmayeva. Security review of email summarization systems. In *2011 World Congress on Internet Security (WorldCIS)*, pages 269 –271, feb. 2011.

[6] Boldizsár Bencsáth and István Vajda. Efficient directory harvest attacks. In *Proceedings of the 2005 international conference on Collaborative technologies and systems*, CTS'05, pages 62–68, Washington, DC, USA, 2005. IEEE Computer Society.

[7] Michael Berry and Jacob Kogan. *Text Mining: Applications and Theory*. Wiley, 2010.

[8] Enrico Blanzieri and Anton Bryl. A survey of learning-based techniques of email spam filtering. *Artif. Intell. Rev.*, 29:63–92, March 2008.

[9] Tobias Blickle and Lothar Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evol. Comput.*, 4:361–394, December 1996.

[10] Ming-wei Chang, Wen-tau Yih, and Christopher Meek. Partitioned logistic regression for spam filtering. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 97–105, New York, NY, USA, 2008. ACM.

[11] The Spamhaus Whitelist Company. The spamhaus whitelist. http://www.spamhauswhitelist.com/en/eligibility.html, 2010.

[12] Paulo Cortez, André Correia, Pedro Sousa, Miguel Rocha, and Miguel Rio. Spam email filtering using network-level properties. In *Proceedings of the 10th industrial conference on Advances in data mining: applications and theoretical aspects*, ICDM'10, pages 476–489, Berlin, Heidelberg, 2010. Springer-Verlag.

[13] Paulo Cortez, Clotilde Lopes, Pedro Sousa, Miguel Rocha, and Miguel Rio. Symbiotic data mining for personalized spam filtering. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '09, pages 149–156, Washington, DC, USA, 2009. IEEE Computer Society.

[14] M. Crispin. INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1. RFC 3501 (Proposed Standard), March 2003. Updated by RFCs 4466, 4469, 4551, 5032, 5182, 5738.

[15] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. P2p-based collaborative spam detection and filtering. In *Proceedings of the Fourth International Conference on Peer-to-Peer Computing*, P2P '04, pages 176–183, Washington, DC, USA, 2004. IEEE Computer Society.

[16] Sarah Jane Delany, Pádraig Cunningham, Alexey Tsymbal, and Lorcan Coyle. A case-based technique for tracking concept drift in spam filtering. *Know.-Based Syst.*, 18:187–195, August 2005.

[17] Holly Esquivel, Aditya Akella, and Tatsuya Mori. On the effectiveness of ip reputation for spam filtering. In *Proceedings of the 2nd international conference on COMmunication systems and NETworks*, COMSNETS'10, pages 40–49, Piscataway, NJ, USA, 2010. IEEE Press.

[18] Pedro Evangelista, Paulo Maia, and Miguel Rocha. Implementing metaheuristic optimization algorithms with jecoli. In *Proceedings of the 9th International Conference on Intelligent Systems Design and Applications*, ISDA '09, pages 505–510, Washington, DC, USA, 2009. IEEE Computer Society.

[19] Tom Fawcett. "in vivo"spam filtering: a challenge problem for kdd. *SIGKDD Explor. Newsl.*, 5:140–148, December 2003.

[20] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27:861–874, June 2006.

[21] Postini Enterprise Spam Filtering. The silent killer: How spammers are stealing your email directory. Technical report, Postini, 2006.

[22] Michael Fong. Spam or ham. Technical report, Dept. of Computer Science, Iowa State University, 2008.

[23] George Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305, March 2003.

[24] Rapid-I GmbH. *RapidMiner 4.6 User Guide, Operator Reference, Developer Tutorial*, Outubro 2009.

[25] Rapid-I GmbH. *RapidMiner 5.0 Manual*, 2010.

[26] Thiago S. Guzella and Walmir M. Caminhas. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206–10222, September 2009.

[27] Jaeyeon Jung and Emil Sit. An empirical study of spam traffic and the use of dns black lists. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC '04, pages 370–375, New York, NY, USA, 2004. ACM.

[28] Ioannis Kanaris, Konstantinos Kanaris, Ioannis Houvardas, and Efstathios Stamatatos. Words versus character n-grams for anti-spam filtering. *International Journal on Artificial Intelligence Tools*, 16(6):1047, 2007.

[29] Chris Kanich, Nicholas Weaver, Damon McCoy, Tristan Halvorson, Christian Kreibich, Kirill Levchenko, Vern Paxson, Geoffrey M. Voelker, and Stefan Sa-

vage. Show me the money: Characterizing spam-advertised revenue. In *Proceedings of the 20th USENIX Security Symposium*, 2011.

[30] Ludmila Kuncheva. Classifier ensembles for changing environments. In Fabio Roli, Josef Kittler, and Terry Windeatt, editors, *Multiple Classifier Systems*, volume 3077 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-25966-4_1.

[31] J. Levine. Dns blacklists and whitelists, irtf anti-spam research group. internet Draft draft-irtf-asrg-dnsbl-08.txt., Nov 2008.

[32] Fulu Li and Mo-Han Hsieh. An empirical study of clustering behavior of spammers and group-based anti-spam strategies. In *Procedings of the Third Conference on Email and Anti-Spam, Mountain View, California, USA*, 2006.

[33] Honghu Liu and Gang Li. Testing statistical significance of the area under a receiving operating characteristics curve for repeated measures design with bootstrapping. *Journal of Data Science*, 3(3):257–278, 2005.

[34] MAAWG. Email security awareness and usage survey. Technical report, Messaging anti-abuse working group, 2010.

[35] MAAWG. Email metrics program:the network operators' perspective, report 15 – first, second and third quarter 2011. Technical report, Messaging Anti-Abuse Working Group, 2011.

[36] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes – which naive bayes? In *Third Conference on Email and AntiSpam CEAS*, pages 125–134, 2006.

[37] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.

[38] J. Myers and M. Rose. Post Office Protocol - Version 3. RFC 1939 (Standard), May 1996. Updated by RFCs 1957, 2449.

[39] OECD. Report on non-oecd countries' spam legislation. Technical report, OECD, 2004.

[40] OECD. Oecd guidelines for the security of information systems and networks: Towards a culture of security. Technical report, OECD, 2006.

[41] Iasonas Polakis, Georgios Kontaxis, Spiros Antonatos, Eleni Gessiou, Thanasis Petsas, and Evangelos P. Markatos. Using social networks to harvest email addresses. In *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society*, WPES '10, pages 11–20, New York, NY, USA, 2010. ACM.

[42] J. Postel. On the junk mail problem, rfc 706. Technical report, IETF Network Working Group., 1975.

[43] Project Honey Pot. Project honey pot statistics. `http://www.projecthoneypot.org/statistics.htm`, August 2011.

[44] M. Prince, B. Dahl, L. Holloway, A. Keller, and E. Langheinrich. Understanding how spammers steal your e-mail address: An analysis of the first six months of data from project honey pot. In *Second Conference on Email and Anti-Spam (CEAS 2005)*, 2005.

[45] Nicolas J Radcliffe. Genetic set recombination. In D Whitley, editor, *Foundations of Genetic Algorithms II*. M Kaufmann, 1992.

[46] Anirudh Ramachandran and Nick Feamster. Understanding the network-level behavior of spammers. *SIGCOMM Comput. Commun. Rev.*, 36:291–302, August 2006.

[47] Uri Raz. How do spammers harvest email addresses. http://www.private.org.il/harvest.html, Maio 2007.

[48] P. Resnick. Internet Message Format. RFC 2822 (Proposed Standard), 2001.

[49] Miguel Rocha and José Neves. Computação genética e evolucionária - apontamentos de apoio à disciplina de bioinformática. Universidade do Minho - Departamento de Informática, Novembro 2004.

[50] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk email. In *Learning for Text Categorization-Papers from the AAAI Workshop*, volume 98-05, pages 55–62, Madison, Wisconsin, 1998.

[51] Guido Schryen. *Anti-Spam Measures: Analysis and Design*. Springer, 2007.

[52] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34:1–47, March 2002.

[53] Mikko Siponen and Carl Stucke. Effective anti-spam strategies in companies: An international study. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences - Volume 06*, pages 127.3–, Washington, DC, USA, 2006. IEEE Computer Society.

[54] Spamhaus. Effective spam filtering. http://www.spamhaus.org/whitepapers/effective_filtering.html, 2011.

[55] Brett Stone-Gross, Thorsten Holz, Gianluca Stringhini, and Giovanni Vigna. The underground economy of spam: a botmaster's perspective of coordinating large-scale spam campaigns. In *Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats*, LEET'11, pages 4–4, Berkeley, CA, USA, 2011. USENIX Association.

[56] B Templeton. Origin of the term "spam" to mean net abuse. `http://www.templetons.com/brad/spamterm.html`, 2003.

[57] Thomas Weise. *Global Optimization Algorithms - Theory and Application*. it-weise.de (self-published): (Germany), 2009.

[58] Yinglian Xie, Fang Yu, Kannan Achan, Rina Panigrahy, Geoff Hulten, and Ivan Osipkov. Spamming botnets: signatures and characteristics. *SIGCOMM Comput. Commun. Rev.*, 38:171–182, August 2008.

[59] Feng Zhou, Li Zhuang, Ben Y. Zhao, Ling Huang, Anthony D. Joseph, and John Kubiatowicz. Approximate object location and spam filtering on peer-to-peer systems. In *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, Middleware '03, pages 1–20, New York, NY, USA, 2003. Springer-Verlag New York, Inc.

Apêndice A

Exemplo de Processo de Classificação do RapidMiner 4.6 em Java

```
import com.rapidminer.RapidMiner;
import com.rapidminer.operator.IOContainer;
import com.rapidminer.operator.ModelApplier;
import com.rapidminer.operator.Operator;
import com.rapidminer.operator.OperatorChain;
import com.rapidminer.operator.OperatorCreationException;
import com.rapidminer.operator.OperatorException;
import com.rapidminer.operator.features.selection.AttributeWeightSelection;
import com.rapidminer.operator.features.weighting.InfoGainWeighting;
import com.rapidminer.operator.io.SparseFormatExampleSource;
import com.rapidminer.operator.performance.BinominalClassificationPerformanceEvaluator;
import com.rapidminer.operator.performance.PerformanceVector;
import com.rapidminer.tools.OperatorService;
import com.rapidminer.Process;

public class RapidMinerProcess {

public static Process createProcess(String dataSetFilePath) throws OperatorCreationException
   {

   //Cria um novo processo
   Process process = new Process();

   //Cria operador de leitura do conjunto de dados no formato esparso
   Operator sparseFormatExampleSource =
   OperatorService.createOperator(SparseFormatExampleSource.class);

   //Definição dos parâmetros do operador
   //Define caminho para o ficheiro que representa o conjunto de dados
   sparseFormatExampleSource.setParameter("attribute_description_file",dataSetFilePath);
```

```
//Define o tipo de formato esparso em que o ficheiro se encontra
sparseFormatExampleSource.setParameter("format","xy");

//Cria operador para cálculo do Information Gain
Operator infoGainWeighting = OperatorService.createOperator(InfoGainWeighting.class);

//Cria Operador para seleção de atributos
Operator attributeWeightSelection =
OperatorService.createOperator(AttributeWeightSelection.class);

//Define o modo de seleção, neste caso os k atributos com os valores de IG mais elevado
attributeWeightSelection.setParameter("weight_relation", "top k");

//Define o número de atributos a selecionar
attributeWeightSelection.setParameter("k", "500");

//Cria Operador que divide o conjunto de dados em dois sub-conjuntos,
//um de treino outro de teste
OperatorChain fixedSplitValidation = (OperatorChain)
OperatorService.createOperator("SimpleValidation");

//Define a percentagem da amostra para o sub-conjunto de treino
fixedSplitValidation.setParameter("split_ratio", "0.6");
//Define uma seleção linear dos atributos
fixedSplitValidation.setParameter("sampling_type", "linear sampling" );

//Cria um OperatorChain
OperatorChain chain1 = (OperatorChain)
OperatorService.createOperator("OperatorChain");

//Cria o operador que implementa o algoritmo de aprendizagem máquina
Operator naiveBayesMultinomial =
OperatorService.createOperator("W-NaiveBayesMultinomial");


OperatorChain chain2 = (OperatorChain)
OperatorService.createOperator("OperatorChain");

//Cria operador que aplica o modelo de classificação às amostras de teste
Operator modelApplier = OperatorService.createOperator(ModelApplier.class);

//Cria operador utilizado para cálculo de métricas de avaliação da classificação
Operator binominalClassification =
OperatorService.createOperator(BinominalClassificationPerformanceEvaluator.class);

//Habilita dois critérios de avaliação da classificação AUC e precision
binominalClassification.setParameter("main_criterion", "AUC");
binominalClassification.setParameter("precision", "true");

//Adicionar os operadores criados ao processo.

fixedSplitValidation.addOperator(chain1);
fixedSplitValidation.addOperator(chain2);
```

```
   process.getRootOperator().addOperator(sparseFormatExampleSource);
   process.getRootOperator().addOperator(infoGainWeighting);
   process.getRootOperator().addOperator(attributeWeightSelection);

   chain1.addOperator(naiveBayesMultinomial);
   chain2.addOperator(modelApplier);
   chain2.addOperator(binominalClassification);

   process.getRootOperator().addOperator(fixedSplitValidation);

   return process;

   }

public static void main(String[] args) throws OperatorException, OperatorCreationException
   {
   //Definição da localização do rapidminer.home, é necessário quando
   //são utilizados classes do Weka.
   System.setProperty("rapidminer.home", "/usr/lib/rapidminer-4.6");

   //Invocar o init() antes de utilizar o objecto OperatorService
   RapidMiner.init(true,true,false,false);

   //Cria IOContainer que vai conter o output do processo
   IOContainer resultcontainer = new IOContainer();

   //Desempenha o processo
   resultcontainer=createProcess(args[0]).run();

   //Obtem o PerformanceVector que contém vários critérios de avaliação da classificação
   PerformanceVector pv = resultcontainer.get(PerformanceVector.class);
   //Obter o valor de AUC
   pv.getCriterion("AUC").getFitness();


   }

}
```

Apêndice B

Exemplo de Configuração de Algoritmo Evolucionário na JECoLi

```
public void configureEA()
{
//Tamanho máximo da solução
int maxSolutionSize = 50;
//Tamanho mínimo da solução
int minSolutionSize = 10;

int numberGenerations=300;

//Obter os atributos presentes no conjunto de dados
//Estes atributos constituem o dominio dos valores possiveis para um gene de um indivíduo
TreeSet<String> attributtes = ExampleSet.getAttributes();

//Instanciar um objecto da classe EvolutionaryConfiguration
// É necessário definir a representação das soluções
EvolutionaryConfiguration<StringSetRepresentation, StringSetRepresentationFactory> configuration
= new EvolutionaryConfiguration<StringSetRepresentation, StringSetRepresentationFactory>();

//Instanciar a solution factory
StringSetRepresentationFactory solutionFactory = new StringSetRepresentationFactory(
minSolutionSize,maxSolutionSize,attributtes,1);

configuration.setSolutionFactory(solutionFactory);


//Definir a classe de avaliação
IEvaluationFunction<StringSetRepresentation> evaluationFunction =
(IEvaluationFunction<StringSetRepresentation>) new SubsetEvaluation(ExampleS1et);
configuration.setEvaluationFunction(evaluationFunction);

//Definir o tamanho da população de indivíduos
```

```
configuration.setPopulationSize(populationSize);

//Expecificar o critério de paragem, neste caso definido pelo número de geraçõ1es
ITerminationCriteria terminationCriteria = new IterationTerminationCriteria(numberGenerations);
configuration.setTerminationCriteria(terminationCriteria);

//Definir os parametros da recombinação
RecombinationParameters recombinationParameters =
new RecombinationParameters(populationSize);
configuration.setRecombinationParameters(recombinationParameters);

//Definir os operadores de seleção
configuration.setSelectionOperator(new TournamentSelection<StringSetRepresentation>(1, 2));
configuration.setSurvivorSelectionOperator(new TournamentSelection<StringSetRepresentation>(1, 2));

configuration.setProblemBaseDirectory("nullDirectory");
configuration.setAlgorithmStateFile("nullFile");
configuration.setSaveAlgorithmStateDirectoryPath("nullDirectory");
configuration.setAlgorithmResultWriterList(
new ArrayList<IAlgorithmResultWriter<StringSetRepresentation>>());
configuration.setStatisticsConfiguration(new StatisticsConfiguration());
configuration.setRandomNumberGenerator(new DefaultRandomNumberGenerator());

//Definir os operadores de recombinação e mutação
ReproductionOperatorContainer<StringSetRepresentation,StringSetRepresentationFactory>
operatorContainer =
new ReproductionOperatorContainer<StringSetRepresentation,StringSetRepresentationFactory>();

operatorContainer.addOperator(0.5, new SetUniformCrossover());
operatorContainer.addOperator(0.5, new SetRandomMutation(Integer.parseInt(nMutation)));
configuration.setReproductionOperatorContainer(operatorContainer);

//Instanciar um algoritmo evolucionário com as configurações definidas
algorithm =
new EvolutionaryAlgorithm<StringSetRepresentation,StringSetRepresentationFactory>(configuration);

}
```