Pós-Graduação em Ciência da Computação

Roberto Hugo Wanderley Pinheiro

COMBINAÇÃO DE CLASSIFICADORES EM DIFERENTES ESPAÇOS DE CARACTERÍSTICAS PARA CLASSIFICAÇÃO DE DOCUMENTOS

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2017

Roberto Hugo Wanderley Pinheiro

COMBINAÇÃO DE CLASSIFICADORES EM DIFERENTES ESPAÇOS DE CARACTERÍSTICAS PARA CLASSIFICAÇÃO DE DOCUMENTOS

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientador: *Prof. Dr. George Darmiton da Cunha Cavalcanti*
Co-Orientador: *Prof. Dr. Tsang Ing Ren*

RECIFE
2017

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

P654c Pinheiro, Roberto Hugo Wanderley
Combinação de classificadores em diferentes espaços de características para classificação de documentos / Roberto Hugo Wanderley Pinheiro. – 2017.
115 f.: il., fig., tab.

Orientador: George Darmiton da Cunha Cavalcanti.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências e apêndice.

1. Inteligência artificial. 2. Recuperação da informação. I. Cavalcanti, George Darmiton da Cunha (orientador). II. Título.

006.31 CDD (23. ed.) UFPE- MEI 2017-83

Roberto Hugo Wanderley Pinheiro

Combinação de Classificadores em Diferentes Espaços de Características para Classificação de Documentos

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutora em Ciência da Computação

Aprovado em: 17/02/2017.

_____

Orientador: Prof. Dr. George Darmiton da Cunha Cavalcanti

BANCA EXAMINADORA

_____

Prof. Dr. Ricardo Bastos Cavalcante Prudêncio
Centro de Informática / UFPE

_____

Prof. Dr. Renato Vimieiro
Centro de Informática / UFPE

_____

Prof. Dr. Luciano de Andrade Barbosa
Centro de Informática / UFPE

_____

Profa. Dra. Eulanda Miranda dos Santos
Departamento de Ciência da Computação/ UFAM

_____

Prof. Dr. Rafael Ferreira Leite de Mello
Departamento de Estatística e Informática/UFRPE

Em Tempo: onde se lê: Aprovado em: 17/02/2017, leia-se: Aprovado em: 20/02/2017.

Agradecimentos

Essa tese foi possível apenas devido aos fundos da FACEPE (IBPG-0613-1.03/12) e dos servidores da Laboratório de Computação de Auto-Desempenho (LCAD) da Universidade Federal de Integração Latino-Americana (UNILA).

Resumo

Classificação de Documentos é um problema no qual um documento em linguagem natural deve ser designado como pertencente à uma das classes pré-estabelecidas. A Classificação de Documentos, com vetores de características gerados pela *Bag-of-Words*, possui duas dificuldades notáveis: alta dimensionalidade e matriz de dados esparsa. Seleção de características reduzem essas dificuldades, mas descarta informação no processo. Uma alternativa é realizar transformações sobre as características, pois ao alterar as características é possível trabalhar sem descartar informações, possibilitando uma melhoria nas taxas de reconhecimento e, em alguns casos, redução da dimensionalidade e esparsidade. Dentre essas transformações, duas pouco utilizadas na literatura são: *Dissimilarity Representation* (DR), no qual cada documento é representado por um vetor composto de distâncias calculadas com relação a um conjunto de documentos referência; e *Dichotomy Transformation* (DT), no qual o problema original é transformado em um problema binário e cada documento é transformado em vários vetores com características obtidas pelo valor absoluto da diferença para os documentos de um subconjunto do conjunto original. A utilização da DR pode reduzir tanto a alta dimensionalidade quanto a esparsidade. Enquanto que a utilização da DT, apesar de não reduzir a dimensionalidade ou esparsidade, melhora as taxas de reconhecimento do classificador, pois trabalha com uma quantidade maior de documentos sobre um problema transformado para duas classes. Neste trabalho, são propostos dois sistemas de múltiplos classificadores para Classificação de Documentos: *Combined Dissimilarity Spaces* (CoDiS) e *Combined Dichotomy Transformations* (CoDiT), cada um baseado em uma das transformações citadas acima. Os múltiplos classificadores se beneficiam da necessidade de encontrar um conjunto para as transformações, pois utilizando diferentes conjuntos possibilita a criação de um sistema diverso e robusto. Experimentos foram realizados comparando as arquiteturas propostas com métodos da literatura usando até 47 bancos de dados públicos e os resultados mostram que as propostas atingem desempenho superior na maioria dos casos.

Palavras-chave: Classificação de Documentos. *Dissimilarity Representation*. *Dichotomy Transformation*. Combinação de Classificadores.

Abstract

Text Classification is a problem in which a natural language document is assigned to one of the pre-established classes. Text Classification, with feature vectors generated by Bag-of-Words, has two notable difficulties: high dimensionality and sparse data matrix. Feature selection reduces these difficulties, but discards information in the process. An alternative is to perform transformations over the features, because by altering the features it is possible to work without discarding information, allowing improvement of recognition rates and, in some cases, reduction of dimensionality and sparseness. Among these transformations, two underused in literature are: Dissimilarity Representation (DR), where each document is represented by a vector composed of distances calculated relative to a set of reference documents; and Dichotomy Transformation (DT), where the original problem is transformed into a binary problem and each document is transformed into several vectors with features obtained by the absolute value of the difference for the documents of a subset of the original set. The use of DR can reduce both the high dimensionality and sparseness. Whereas the use of DT, despite not reducing dimensionality or sparseness, improves the recognition rates of the classifier, since it works with a larger amount of documents on a problem transformed into two classes. In this work, two multiple classifiers systems for Text Classification are proposed: Combined Dissimilarity Spaces (CoDiS) and Combined Dichotomy Transformations (CoDiT), each one based on the transformations mentioned above. The multiple classifiers benefits from the need to find a set for the transformations, because using different sets allows the creation of a diverse and robust system. Experiments were performed comparing the proposed architectures with literature methods using up to 47 public databases and the results show that the proposals achieve superior performance in most cases.

Keywords: Text Classification. Dissimilarity Representation. Dichotomy Transformation. Multiple Classifiers System.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

ALOFT *At Least One FeaTure* 38

All-kNN *All k Nearest Neighbor*

ATISA2 *Adaptive Threshold-based Instance Selection Algorithm 2* 47

BNS *Bi-Normal Separation* 36

BoW *Bag-of-Words* 15

CDM *Class Discriminanting Measure* 37

CHI *Chi-Squared Statistics* 36

cMFDR *Category-dependent Maximum f Features per Document - Reduced* 39

CNN *Condensation Nearest Neighbor* 44

CoDiS *Combined Dissimilarity Spaces* 18

CoDiT *Combined Dichotomy Transformations* 18

CR *Cosine Representation* 87

DR *Dissimilarity Representation* 16

DROP3 *Decremental Reduction Optimization Procedure 3* 46

DT *Dichotomy Transformation* 16

ENN *Edited Nearest Neighbor* 44

FEF *Feature Evaluation Function* 35

IB2 *Instance-Based Learning Algorithm 2* 45

IG *Information Gain* 37

MFD *Maximum f Features per Document* 38

MOR *Multi-class Odds Ratio* 37

RNN *Reduced Nearest Neighbor* 44

TF *Term Frequency* 27

TF-IDF *Term Frequency - Inverse Document Frequency* 27

SVM *Support Vector Machine* 28

VR *Variable Ranking* 34

Sumário

1

Introdução

A quantidade de informação em forma de texto está em crescimento desde a difusão da *Internet*, devido à facilidade de produzir e divulgar conteúdo. Dado que o interesse em obter informações de modo rápido e prático cresceu, a comunidade científica iniciou o desenvolvimento de aplicações voltadas para documentos digitais. Várias ramificações de aplicações foram criadas para superar os problemas emergentes, sendo uma das mais importantes a Classificação de Documentos [2].

A Classificação de Documentos é a tarefa de atribuir uma classe, pertencente a um universo finito e pré-estabelecido, a um determinado texto em linguagem natural. Aplicações capazes de lidar com esse problema estão tornando-se cada vez mais difundidas, seja para detecção de *spam* [3], removendo as mensagens de conteúdo suspeito ou propagandas, seja de *e-mails*, comentários do *twitter* ou *youtube*; organização de documentos em tópicos hierárquicos, facilitando pesquisas em diversas áreas; análise de sentimento [4], facilitando a compreensão de dados massivos de empresas em redes sociais; ou até mesmo como sistemas de recomendação, indicando produtos de interesse para um usuário baseado em seu histórico de compras e acessos.

Esta tese é focada em problemas de Classificação de Documentos, um problema de Aprendizagem de Máquina, que como qualquer problema possui suas particularidades e desafios. Em primeiro lugar, para realizar a Classificação de Documentos, é necessário converter os documentos com texto em vetores de características. Sendo a representação tradicional mais utilizada na área conhecida por *Bag-of-Words* (BoW) [2], devido aos bons resultados, facilidade de implementação e baixo custo computacional. Esta representação traz consigo dois grandes desafios: i) Alta dimensionalidade [5]: Cada documento é representado por um vetor de características, no qual cada uma dessas características é um termo[1] que aparece em qualquer documento do *Corpus*[2]. Como todos os termos do *Corpus* são considerados, é comum os vetores possuírem dezenas de milhares de características; ii) Matriz de dados esparsa [6]: Termos que não aparecem em um determinado documento possuem a característica com o valor zero. Como cada documento possui apenas um percentual pequeno de todos os termos existentes no *Corpus*, a quantidade de zeros é imensa e, consequentemente, os dados são esparsos.

[1] Sequência de caracteres alfanuméricos separados por delimitadores como espaços, vírgulas e pontos.
[2] Coletânea ou conjunto de documentos sobre determinado tema, isto é, a base de dados utilizada.

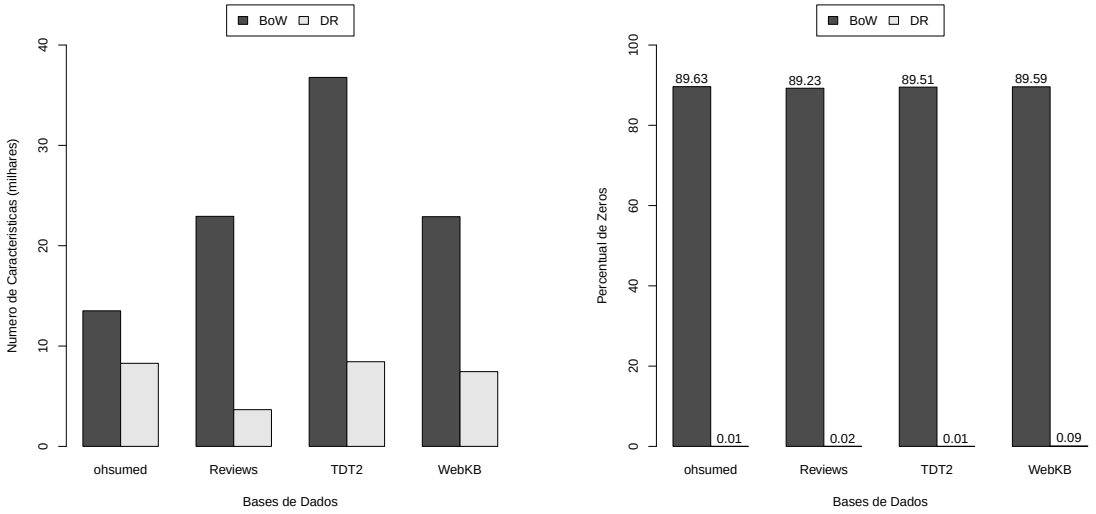
Problemas com uma grande quantidade de características podem sofrer da maldição da dimensionalidade [7], isto é, o volume do espaço cresce tão rapidamente, com o crescimento das características, que os dados distribuem-se de modo esparso. Nesses casos, é necessário aumentar a quantidade de dados ou diminuir a quantidade de características, para facilitar o processo de classificação. Tendo em vista que independentemente da alta dimensionalidade, problemas de Classificação de Documentos possuem uma matriz de dados esparsa, tornando-o um problema bastante específico e desafiador [6].

Portanto, é importante reduzir a dimensionalidade do vetor de características quando BoW é usado. O uso de métodos de seleção de características é a opção mais adotada para lidar com a alta dimensionalidade. Esses métodos diminuem a quantidade de zeros na matriz de dados e removem as características irrelevantes ou redundantes, consequentemente reduzindo o custo computacional das técnicas de Aprendizagem de Máquina usadas na classificação. Diversos métodos de seleção de características foram propostos especificamente para Classificação de Documentos [8, 9, 10, 11, 12, 13, 14, 15]. Entretanto, é necessária uma seleção de características agressiva para reduzir a alta dimensionalidade e, principalmente, para reduzir o percentual de zeros na matriz de dados. Uma seleção agressiva pode degradar o desempenho do sistema [16, 17], pois informações importantes são descartadas no processo.

Uma alternativa para lidar com certos desafios, incluindo alta dimensionalidade e esparsidade, está no uso de transformações sobre as características [18, 19, 20, 21, 22]. As transformações podem diminuir as dificuldades existentes nos dados originais ou melhorar o poder discriminatório entre as classes do problema, visando facilitar o processo de classificação para melhorar as taxas de reconhecimento. Dentre essas transformações, duas foram utilizadas neste trabalho: i) *Dissimilarity Representation* (DR), no qual cada documento é representado por um vetor de distâncias, calculadas com relação a um conjunto de documentos referência; ii) *Dichotomy Transformation* (DT), no qual o problema original é transformado em um problema binário (duas classes) e cada documento é transformado em vários vetores com características obtidas pelo valor absoluto da diferença para os documentos de um subconjunto do conjunto original.

A *Dissimilarity Representation* proposta por Duin e Pekalska [23] pode ser utilizada para transformar os dados obtidos pelo BoW visando reduzir a dimensionalidade e a quantidade de zeros na matriz de dados (esparsidade). Na *Dissimilarity Representation*, cada documento é transformado em um vetor de distâncias, de modo que tanto a quantidade de características quanto a quantidade de zeros é inferior à original. A Figura 1.1 mostra dois gráficos que exibem bem as vantagens da *Dissimilarity Representation* sobre a representação *Bag-of-Words* em quatro bases de dados diferentes, tanto com relação à redução da dimensionalidade (Figura 1.1(a)), no qual a utilização de DR sempre reduz a quantidade de características; como com relação ao percentual de zeros na matriz de dados (Figura 1.1(b)), no qual a utilização de DR reduz drasticamente a esparsidade dos dados. Espera-se que a dissimilaridade entre documentos reduza a perda de informação em comparação com uma seleção de características agressiva,

pois na seleção de características vários termos são descartados, enquanto que na *Dissimilarity Representation* todos os termos são usados no cálculo da distância entre os documentos.



(a) Dimensionalidade (b) Esparsidade

Figura 1.1: Gráficos mostrando a diferença entre a utilização de *Bag-of-Words* (BoW) e *Dissimilarity Representation* (DR) em quatro bases de dados reais.

Diferentemente da *Dissimilarity Representation*, a *Dichotomy Transformation* proposta por Cha e Srihari [24] não reduz a quantidade de características, a abordagem utilizada pela DT é tratar diretamente a maldição da dimensionalidade. Para tal, a *Dichotomy Transformation* aumenta a quantidade de exemplos disponíveis para treinamento, evitando dados esparsos no espaço de características. A ideia da DT é facilitar a discriminação as classes para melhorar o desempenho do classificador, isto é, criar um classificador forte, que atinja altas taxas de desempenho, e robusto, capaz de lidar com diversas bases de dados. A *Dichotomy Transformation* transforma um problema multi-classe em um problema binário (duas classes) e aumenta exponencialmente a quantidade de documentos. Portanto, espera-se que a *Dichotomy Transformation* melhore as taxas de reconhecimento do classificador, pois aumentar a quantidade de documentos facilita a generalização do classificador e evita a maldição da dimensionalidade, sendo possível treinar um classificador mais robusto; e a classificação de um problema binário é, geralmente, mais simples de ser resolvida em comparação com problemas multi-classe [25].

Entretanto, apesar das vantagens das transformações, ambas requerem certos parâmetros de entrada, mais especificamente conjuntos de dados, para serem utilizadas. Deste modo, o desempenho do classificador treinado a partir dos dados transformado é influenciado diretamente pelo conjunto de dados utilizado na transformação. Encontrar o melhor conjunto para aplicar essas transformações é uma tarefa não trivial. Assim, a Combinação de Classificadores pode ser utilizada para evitar a busca pelo melhor conjunto, pois é possível superar o problema do ótimo local ao combinar condições iniciais diferentes [26], isto é, variando o conjunto de dados utilizado nas transformações. Como o sucesso da combinação não depende de classificadores

fortes para atingir altas taxas de acerto [27] e classificadores gerados por conjuntos diferentes conseguem uma boa performance [28], espera-se que a combinação de classificadores treinados em diferentes espaços de características transformadas incrementem o desempenho de todo o sistema, pois essa variação dos dados é a metodologia usada pela maioria dos métodos de combinação [1]. Portanto, a Combinação de Classificadores se beneficia da necessidade de encontrar o melhor conjunto, pois utilizando diferentes conjuntos nas transformações possibilita a criação de um sistema diverso, fator importante em qualquer combinação de classificadores [29]; e mais robusto, pois apesar da combinação não resolver os problemas de todas as bases de dados, um sistema diverso tem uma maior capacidade do que um classificador isolado [30].

1.1 Objetivo

Nessa tese são propostas duas arquiteturas para Classificação de Documentos que se utilizam de transformação das características para geração de classificadores diversos e um sistema robusto, isto é, capazes de obter bons resultados na maioria das bases de dados. As arquiteturas propostas tratam-se de sistemas de múltiplos classificadores gerados com base na *Dissimilarity Representation*, chamado de *Combined Dissimilarity Spaces* (CoDiS) e na *Dichotomy Transformation*, chamado de *Combined Dichotomy Transformations* (CoDiT). As arquiteturas propostas devem melhorar as taxas de reconhecimento quando comparadas a outras metodologias de combinação de classificadores, o CoDiS ao tratar as dificuldades obtidas pelo BoW e o CoDiT ao expandir a quantidade de informação disponível para o treinamento do classificador.

1.2 Contribuições

Os estudos realizados nesta tese proporcionaram as seguintes contribuições:

i) Proposta do *Combined Dissimilarity Spaces* (CoDiS), um sistema de Classificação de Documentos que combina classificadores treinados em diferentes espaços de dissimilaridades obtidos pela *Dissimilarity Representation* (Capítulo 3);

ii) Proposta do *Combined Dichotomy Transformations* (CoDiT), um sistema de Classificação de Documentos que combina classificadores treinados em diferentes espaços de características transformadas pela *Dichotomy Transformation* (Capítulo 4). Parte desta proposta foi publicada em evento [22].;

iii) Uma métrica inspirada em similaridade para a *Dichotomy Transformation* (Seção 4.5). Parte desta proposta foi publicada em evento [22];

iv) Utilização da similaridade do cosseno na *Dissimilarity Representation* em uma arquitetura sem combinação de classificadores (Seção 3.4 e Seção 5.5). Parte desta proposta foi publicada em evento [19];

1.3 Trabalhos Relacionados

Nesta seção são discutidos alguns trabalhos de Classificação de Documentos relacionados com os estudos realizados nesta tese. Os trabalhos são mostrados de forma resumida na Tabela 1.1 e são focados em três aspectos: i) redução de características (coluna Característica), indicando que o trabalho utiliza seleção de características, extração de características ou alguma transformação nas características de modo à reduzir a dimensionalidade; ii) tratamento de esparsidade (coluna Esparsidade) indicando que o trabalho reduz a esparsidade dos dados; e, iii) combinação de classificadores (coluna Combinação) indicando trabalhos que usam combinação de classificadores. A última coluna da Tabela 1.1 mostra um resumo do método proposto em cada artigo.

Tabela 1.1: Related works

Método	Característica	Esparsidade	Combinação	Resumo
Prabowo e Thelwall [31]	✓		✓	Combina diferentes tipos de classificadores
Ozgur e Gungor [32]	✓			Extensão do *Bag of Words* baseada em conceitos de poda
Xia et al. [33]			✓	Integra diferentes tipos de classificadores, conjuntos de características e abordagens de combinação
Pinheiro et al. [10]	✓			Método de seleção de características
De Silva et al. [34]	✓		✓	Combina diferentes representações de características com diferentes tipos de classificadores
Wang et al. [4]	✓		✓	Compara métodos de combinação de classificadores
Wu et al. [35]	✓		✓	Método híbrido (*Support Vector Machines* e *Random Forest*) para bases desbalanceadas de texto
Jun et al. [18]	✓	✓		Agrupamento de documentos com vetores de suporte em dimensão reduzida
Zhang and He [20]		✓	✓	Combinação de dois classificadores usando *Bag-of-Words* aprimorado
Altinel et al. [21]	✓	✓		*Kernel* com peso para as classes
Onan et al. [36]	✓		✓	Extração de palavras-chave estatísticas
CR (proposta)	✓	✓		Representação do cosseno com seleção de protótipos
CoDiS (proposta)	✓	✓	✓	Combina diferentes espaços de dissimilaridade
CoDiT (proposta)		✓	✓	Combina classificadores binários com transformações diferentes

Em [32], Ozgur e Gungor propõem um método de seleção de características pra reduzir a alta-dimensionalidade gerada pelo *Bag-of-Words*. Entretanto, em contraste aos demais métodos de seleção de características que precisam de uma entrada indicando a quantidade de características, nesse trabalho foi utilizado um parâmetro chamado nível de poda (*pruning level*) que é utilizado para encontrar o número final de características de um modo guiado pelos dados. Esta estratégia de encontrar o número de características por base de dados foi aplicado em trabalhos mais recentes de seleção de características [10, 13]. Altinel et al. [21] introduz *kernel* semântico para *Support Vector Machines* (SVM) que suavisa a representação dos documentos ao apresentar uma dependência baseada nas classes dos termos, reduzindo a dimensionalidade e esparsidade. O *kernel* proposto é rápido e melhora os resultados em problemas de texto. Jun et al. [18] também propos uma técnica para reduzir a alta dimensionalidade e esparsidade em problemas de texto, realizando uma redução de dimensionalidade combinando *singular value decomposition* e *principal component analysis*. A maior dificuldade desta abordagem é definir o número de agrupamentos, e os autores afirmam que um modelo combinado poderia melhorar os resultados. Uma ideia similar foi proposta nesta tese [19], no qual a representação do cosseno e algoritmos de seleção de protótipos foram combinados para reduzir a esparsidade. Ambas as propostas não utilizam combinação de classificadores e poderiam se beneficiar para construção de um método mais robusto.

Combinação de classificadores foi aplicada em texto por Prabowo e Thelwall [31]. O

método utiliza uma quantidade pequena de classificadores (dois à quatro) que são combinados em série usando classificadores baseados em regras e SVM. Apenas um dos classificadores da combinação, o classificador baseado em estatísticas, realiza seleção de características, sendo resposável pelos melhores resultados dentre as configurações avaliadas. Xia et al. [33] propos um *framework* que integra diferentes tipos de classificadores (*Naive Bayes*, *maximum entropy*, e SVM) usando diferentes conjuntos de características extraídas da representação *Bag-of-Words*. Esses classificadores heterogeneos foram combinados usando diferentes regras de combinação, como fixa, pondera, e meta-classificador. No final, várias configurações diferentes da proposta foi avaliada em cinco bases de dados, exibindo resultados promissores para combinação de diferentes vetores de características. De Silva et al. [34] também utiliza uma combinação pequena e heterogenea. Essa combinação é composta por quatro classificadores: *multinomial Naive Bayes*, SVM, *Random Forest*, e *logistic regression*; e voto majoritário foi utilizado como regra de combinação. Nesta proposta foi utilizado *hashing* de características para reduzir o número de características; entretanto, *Bag-of-Words* sem redução alcançou os melhores resultados. Os autores declararam que um maior esforço deveria ser devotado no aumento da diversidade dos classificadores para melhorar as taxas de acerto. Wang et al. [4] avaliou a eficácia de três métodos de geração de combinações (*Bagging*, *Boosting*, e *Random Subspace*) usando uma combinação composta por cinco classificadores (*Naive Bayes*, *maximum entropy*, árvore de decisão, *k Nearest Neighbors*, e SVM). A melhor configuração foi a combinação com *Random Subspace* e SVM como classificadores base. É importante enfatizar que, diferente dos outros dois métodos de geração de combinações, o *Random Subspace* lida com dados de alta dimensionalidade, pois cada classificador é treinado usando um subconjunto aleatório de características. Wu et al. [35] propos um método híbrido para bases desbalanceadas de texto combinando conceitos de SVM e *Random Forest*, chamado *ForestTexter*. *ForestTexter* lida com a alta-dimensionalidade dos dados ao realizar amostragem das características, de modo similar ao *Random Subspace*. Cada nó da árvore é um classificador SVM. Onan et al. [36] utilizou extração de palavras-chave para reduzir o número de características e realizou diversos experimentos usando quatro classificadores base (*Naive Bayes*, SVM, *logistic regression*, e *Random Forest*) com cinco métodos de combinação (*AdaBoost*, *Bagging*, *Dagging*, *Random Subspace*, e voto majoritário). Apesar das diversas representações com a extração de palavras-chave, essas extrações nunca são combinadas no trabalho, uma oportunidade perdida. Em geral, uma combinação de *Bagging*, *Random Subspace* e *Random Forest* mostrou-se promissora. Zhang and He [20] utilizou *latent topics* e *related words* para enriquecer a representação *Bag-of-Words*, que proveem maneiras de tornar o texto esparso mais relacionado e focado em tópicos. Ambas representações são usadas para criar uma combinação com dois classificadores: *Naive Bayes* e *maximum entropy*.

Dado a apresentação destes trabalhos, alguns pontos se destacam: i) *Bag-of-Words* ainda é uma representação bastante utilizada, apesar dos seus problemas, normalmente sendo utilizada como uma primeira camada para uma representação mais complexa; ii) SVM é bastante utilizado em problemas de texto e é uma escolha bastante promissora, atingindo boas taxas

individualmente e em combinações; iii) alguns trabalhos utilizam combinação de classificadores, sendo que a quantidade de classificadores utilizados é normalmente pequena. Uma quantidade maior de classificadores podem levar a um aumento de diversidade, que tem um importante papel em combinações; iv) a esparsidade é tratada normalmente utilizando alguma transformação sobre as características.

Com tais pontos em mente, nossas propostas se encaixam na literatura de diferentes maneiras. A proposta *Cosine Representation* (Seção 4.5), variante sem combinação do CoDiS, trata os problemas do *Bag-of-Words*, pois reduz a dimensionalidade e a esparsidade; A proposta *Combined Dissimilarity Spaces* (Capítulo 3) possui os três aspectos citados, isto é, reduz a quantidade de características, esparsidade e utiliza combinação de classificadores. Nenhum outro trabalho lida com todos os aspectos, pois ou lidam com dimensionalidade e esparsidade, mas não usufruem da combinação de classificadores [21, 18, 19]; ou utilizam combinação de classificadores mas falham com a dimensionalidade ou esparsidade [31, 33, 34, 4, 35, 20, 36]. Por fim, a proposta *Combined Dichotomy Transformations* (Capítulo 4) utiliza combinação de classificadores e lida com a esparsidade tratando a maldição da dimensionalidade, sendo essa configuração vista apenas em um dos trabalhos estudos [20]. Em suma, tendo em vista os três aspectos verificados nesse estudo, as propostas são mais abrangentes do que os demais trabalhos na literatura ou equivalentes.

1.4 Organização do Texto

Este trabalho está organizado em cinco capítulos. No Capítulo 2 está a fundamentação teórica deste trabalho, no qual são apresentados os conceitos necessários para compreensão das arquiteturas propostas: Classificação de Documentos, apresentando a arquitetura de um sistema, classificador utilizado e medidas de avaliação; Seleção de Características, apresentando o algoritmo mais utilizado para seleção no domínio estudado e algoritmos estado-da-arte utilizados nos experimentos, bem como diversas funções de avaliação de características; *Dissimilarity Representation*, apresentando a representação que se baseia em dissimilaridade entre documentos; *Dichotomy Transformation*, apresentando a transformação que gera um problema binário com base em um problema multi-classe; Combinação de Classificadores, apresentando como calcular a diversidade, como gerar o *pool* de classificadores e como combinar as respostas dos classificadores. No Capítulo 3 é apresentada uma das arquiteturas propostas: *Combined Dissimilarity Spaces* (CoDiS). No Capítulo 4 é apresentada outra arquitetura prosposta: *Combined Dichotomy Transformations* (CoDiT). No Capítulo 5 são relatados e discutidos os resultados obtidos nos experimentos. Finalmente, o Capítulo 6 encerra este trabalho com as conclusões.

2

Fundamentação Teórica

Este capítulo apresenta os conceitos necessários para compreensão das abordagens propostas. Na Seção 2.1 é descrita a classificação de documentos. A Seção 2.2 descreve algumas técnicas de seleção de características. A Seção 2.3 apresenta a *Dissimilarity Representation*. A Seção 2.4 apresenta a *Dichotomy Transformation*. A Seção 2.5 descreve alguns algoritmos de seleção de protótipos. A Seção 2.6 apresenta a arquitetura geral de um sistema de múltiplos classificadores e quais metodologias são utilizadas para combinar. A Seção 2.7 traz as considerações finais deste capítulo.

2.1 Classificação de Documentos

Classificação de Documentos é a tarefa de designar uma determinada classe $class(y_j) \in \mathbf{C} = \{c_1, \ldots, c_C\}$ para todo documento $y_j \in \mathbf{Y}$. No qual $\mathbf{Y}$ é o conjunto de teste e $\mathbf{C}$ é o conjunto pré-definido com C categorias. Essa é uma tarefa que pode ser naturalmente modelada como um problema de aprendizado supervisionado. Técnicas de Aprendizagem de Máquina são costumeiramente utilizadas em problemas de Classificação de Documentos, tais como Árvores de Decisão [37, 38], Redes Neurais [39, 40], k vizinhos mais próximos [41, 42], *AdaBoost* [43], Lógica *Fuzzy* [44], Máquina de Vetor Suporte (SVM, do inglês *Support Vector Machine*) [45, 46] e *Naïve Bayes* (NB) [47, 48]. Recentemente, alguns modelos de *deep learning* estão sendo utilizados para Classificação de Documentos [49, 50, 51, 52].

Dentre as aplicações de Classificação de Documentos, destacam-se seu uso como etapa de pré-processamento para indexação de documentos em sistemas de recuperação de informação [53], detecção de *spam* em *e-mails* [54, 3], e análise de sentimentos [4, 31]. Todo problema que envolva texto e exista a necessidade de distribuir os documentos em categorias é uma aplicação de Classificação de Documentos. Portanto, é uma área ampla em aplicações.

A Figura 2.1 mostra uma arquitetura de um sistema de Classificação de Documentos. Inicialmente, todos os documentos de uma base de dados usada em um sistema de Classificação de Documentos são escritos em linguagem natural. Essa base de dados em linguagem natural (chamada de *Corpus*) é dividida em conjunto de treinamento e conjunto de teste, o primeiro será utilizado para treinar o classificador e o segundo será utilizado para avaliar o classificador. A

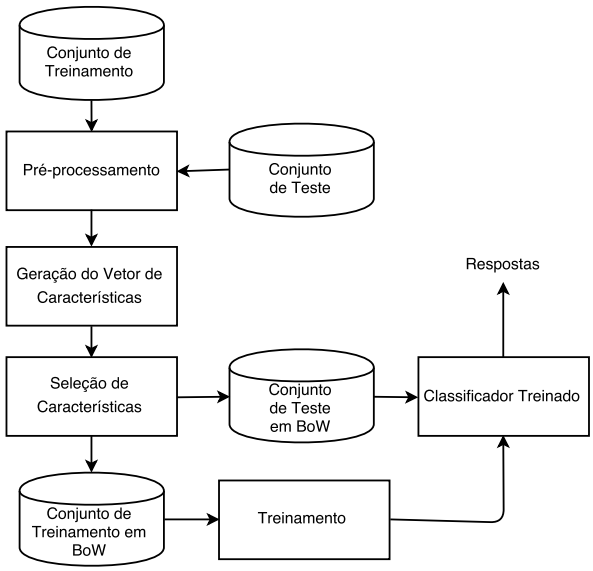


Figura 2.1: Arquitetura de um sistema de Classificação de Documentos.

primeira etapa é tratar esses documentos por diversas rotinas de pré-processamento (Seção 2.1.1). Após o pré-processamento, é necessário modificar a representação original (linguagem natural) para alguma representação aplicável em Aprendizagem de Máquina, sendo a abordagem mais comum a geração de um vetor de características (Seção 2.1.2). Esse vetor pode ser reduzido por uma rotina de seleção de características (Seção 2.2), criando uma representação reduzida dos documentos. Por fim, essa representação é armazenada numa memória principal ou secundária, para ser usada na construção do classificador. Esses mesmos procedimentos são realizados sobre o conjunto de teste, pois este deve possuir a mesma estrutura do conjunto de treinamento.

O conjunto de treinamento na representação *Bag-of-Words* serve como entrada para o módulo de treinamento que irá treinar um classificador com alguma técnica de aprendizado supervisionado. Após o treinamento, os dados do conjunto de teste são apresentados ao classificador treinado e, por fim, os documentos são etiquetados. Essa etiquetagem indica à qual classe o documento pertence.

Todas as etapas da arquitetura (Figura 2.1) são apresentadas a seguir. Adicionalmente, a Seção 2.1.4 mostra as medidas de avaliação usadas em Classificação de Documentos.

2.1.1 Pré-processamento

O pré-processamento contém as primeiras operações realizadas sobre o *Corpus*, sendo executado antes de qualquer etapa. Seu objetivo principal é organizar as informações presentes nos documentos, preparando-os para a geração do vetor de características. Existem diversas rotinas de pré-processamento de texto. A utilização, ou não, dessas rotinas depende do domínio de aplicação do *Corpus*, pois certas rotinas podem contribuir em alguns domínios, enquanto que outras devem ser completamente evitadas. As principais rotinas são:

- **Análise Léxica**

  Converte o texto original para uma lista de palavras. Normalmente removem pontuação, hífens e dígitos, além de transformar letras maiúsculas em minúsculas. Entretanto, algumas remoções podem alterar o sentido do texto ou, até mesmo, eliminar informações relevantes (números em *Corpus* com datas relevantes ou hífens em *Corpus* com muitas palavras compostas). Portanto, dependendo do *Corpus*, algumas remoções devem ser evitadas.

- **Stopwords**

  Stopwords é uma lista de palavras, composta por termos que não possuem um significado semântico relevante, como artigos, conjunções e preposições. Essas palavras serão removidas de todos os documentos do *Corpus*. Normalmente uma lista padrão é utilizada dependendo do idioma, mas essa lista também pode ser elaborada para atender especificamente algum *Corpus*.

- **Stemming**

  A tarefa do *stemming* é reduzir as palavras ao seu radical, pois é irrelevante manter termos de mesmo radical[1] separados (plural, mudança de gênero e tempo verbal). Deste modo, a quantidade de termos é reduzida e unida para compor termos mais relevantes.

  Vários algoritmos de *stemming* foram propostos, normalmente usando gramáticas [55] ou regras condicionais [56]. Uma revisão geral dos algoritmos de *stemming* é feita no trabalho de Viera e Virgil [57]. Dado que a construção dos prefixos, sufixos e radicais variam para cada idioma, a maioria dos algoritmos é criada para solucionar o problema em um determinado idioma. Uma proposta robusta foi a utilização de *n*-grams para gerar pseudo-radicais [58]. Com essa abordagem, é possível realizar o *stemming* independente do idioma, apesar dos resultados serem inferiores com relação aos algoritmos criados para um idioma específico.

- **Tesauro**

  Tesauro é um dicionário de sinônimos de um idioma. Sua função é similar ao *stemming*, unir vários termos em um único termo, mas nesse caso utilizando sinônimos em vez do radical. Essa rotina de pré-processamento pode ser implementada com o apoio do *WordNet* [59, 60], um grande Tesauro *online*.

- **Lemmatization**

  Lemmatization é um processo de agrupar certos termos em um mais abrangente [61], como no *stemming*. Entretanto, o *lemmatization* é um procedimento semântico, pois

[1] O radical, ou raiz, é a parte do verbo ou substantivo que exprime a ideia geral da palavra, ou seja, seu significado mesmo sem o prefixo ou sufixo. É a parte invariável do vocábulo.

leva em consideração o significado das palavras, assim como a vizinhança dos termos próximos ou até mesmo em todo o documento.

- **Grupos Nominais**

  Grupos nominais são termos compostos (por exemplo, "Inteligência Artificial"). Sua função também é similar ao *stemming*, pois visa unir vários termos em um único termo, mas nesse caso é a união de blocos de termos em sequência.

- **Corretor Ortográfico**

  Um dos desafios de quando se trabalha com texto são os erros ortográficos. Esses erros criam características irreais, pois a mesma palavra é escrita de várias maneiras diferentes. Um corretor ortográfico pode reduzir esses problemas, mas também pode criar outros problemas, seja ao substituir palavras incorretas por uma palavra indesejada ou ao substituir palavras corretas desconhecidas no vocabulário do corretor (por exemplo, sigla "ONU" se tornar "ônus").

2.1.2 Vetor de Características

Os documentos em sua estrutura original (diversos caracteres compondo palavras e frases) não são bons objetos de entrada para um sistema de Classificação de Documentos, tanto por sua complexidade como por seu tamanho. Portanto, todos os documentos são convertidos para uma forma de representação mais compacta: um vetor de características.

Segundo Xue e Zhou [62], quando se trata de Classificação de Documentos, a palavra "característica" possui dois significados. Um é referente à unidade que será utilizada para representar o documento (chamado de unidade da característica). O outro é referente a qual valor traz melhor representatividade às características (chamado de valor da característica).

Com relação às unidades das características, podem ser citadas:

- **Palavras isoladas**

  Palavras isoladas são sequências de caracteres que iniciam em um caractere alfanumérico e são finalizados por algum símbolo delimitador (espaço, ponto, vírgula) especificado pelo desenvolvedor. Essa abordagem é conhecida como *Bag-of-Words* (BoW), sendo a mais difundida e utilizada em toda literatura [2]. Apesar da existência de unidades de características mais complexas, os resultados obtidos pela abordagem BoW são bem estabelecidos e esta permanece sendo utilizada em trabalhos recentes [13]. Adicionalmente, BoW requer menos tempo computacional, devido sua simplicidade.

- **Sentenças**

  São cadeias de palavras, como: "o cachorro atravessou a rua" ou "enviou uma carta". A definição do que é uma sentença depende do limite imposto pelas especificações do algoritmo utilizado.

Como diversas pesquisas foram realizadas sobre o uso de sentenças como unidades das características, algoritmos foram propostos e estudados visando formar sentenças mais representativas. Uma das primeiras propostas, para melhorar a composição das sentenças, foram as sentenças sintáticas [63, 64, 65], que apesar de serem mais sofisticadas por usar a gramática do idioma, não trouxeram melhorias significativas. Outra proposta foram as sentenças estatísticas [66, 67, 68], chamadas de *n*-grams, que coleta trechos de *n* em *n* palavras. Melhorias foram reportadas, mas com muita dependência ao valor utilizado no parâmetro *n*. Uma combinação entre as sentenças sintáticas e estatísticas foi proposta [69], mas foram notados diversos problemas e custo computacional elevado.

Um dos problemas no uso de sentenças é conhecido como a má vizinhança de termos [70], que afirma ser difícil perceber quando uma ou mais palavras devem ser unidas ou não. Um exemplo pode ser dado pelas sentenças "professor assistente" e "professor alto". A primeira é uma sentença interessante por abordar o posto de um determinado professor e consequentemente não confundir com "professor associado", enquanto a segunda refere-se apenas a uma característica física do professor.

- ***n*-grams**

  A unidade *n*-grams possui dois significados, podendo *n* ser a quantidade de palavras em sequência para compor uma sentença – como citado no item anterior – ou uma quantidade de caracteres em sequência, como é mostrado no presente tópico. A palavra "teste", por exemplo, em 3-grams seria representada pelas cadeias "tes", "est", "ste".

  Em algumas situações, não é uma boa prática utilizar uma representação baseada em palavras ou sentenças. Idiomas como chinês ou japonês não fazem uso do espaço, não sendo possível segmentar este tipo de documento por palavras, assim como documentos que utilizem muitos símbolos, como nas linguagens de programação.

  Bons resultados foram atingidos com a abordagem *n*-grams [71]. Apesar das possibilidades de características crescerem exponencialmente com o aumento do *n*, apenas uma pequena fração é utilizada pelo conjunto de treinamento e algumas destas ainda podem ser removidas por métodos de seleção de características.

Com relação aos valores das características [72], podem ser citados:

- **Binário**

  Um valor binário atribui em cada característica uma representação *true* ou *false*. Esse valor é usado para indicar a presença ou ausência desta característica em um determinado documento.

É o modelo mais simples, sendo facilmente implementado, pouco custoso em termos de memória e processamento. Entretanto, tamanha simplicidade remete a uma baixa representatividade.

- *Term Frequency* (TF)

  É um modelo mais completo que o binário, pois não indica apenas a presença ou ausência da característica. O valor da frequência do termo será um inteiro representando quantas vezes a característica aparece em um dado documento. Em alguns casos este valor inteiro é normalizado. Obviamente, sua implementação é mais custosa e mais complexa que a do valor binário. A Figura 2.2 exemplifica essa representação.

- *Term Frequency - Inverse Document Frequency* (TF-IDF)

  É o modelo mais popular, bastante utilizado em conjunto com a abordagem BoW e, normalmente, usado em comparação de desempenho com outras representações propostas [62, 73]. Sua implementação é mais complexa que as anteriores, pois necessita de cálculos menos intuitivos.

  O valor do TF-IDF$_{x_i,w_h}$ representa a importância do termo w_h no documento x_i. Seu calculo é dado pela equação:

$$\text{TF-IDF}_{x_i,w_h} = tf_{x_i,w_h} \times idf_{w_h}, \tag{2.1}$$

  sendo o valor *Term Frequency* definido por:

$$tf_{x_i,w_h} = \frac{w_h^{x_i}}{\sum_a^V w_a^{x_i}}, \tag{2.2}$$

  que representa a quantidade de ocorrências do termo h no documento i, sendo o denominador um fator de normalização dado pelo somatório de todos os termos existentes no documento x_i $(w_1^{x_i}, w_2^{x_i}, \ldots, w_V^{x_i})$.

  Enquanto que o valor *Inverse Document Frequency* é definido pela equação:

$$idf_{w_h} = \ln \frac{T}{T_{w_h}}, \tag{2.3}$$

  cujo cálculo é realizado baseado na quantidade de documentos no *Corpus* (T) e na quantidade de documentos no qual o termo w_h aparece (T_{w_h}).

2.1.3 Classificador

Diversas técnicas de Aprendizagem de Máquina são utilizadas em problemas de Classificação de Documentos. Entretanto, o foco deste trabalho não é aprofundar um estudo sobre esses

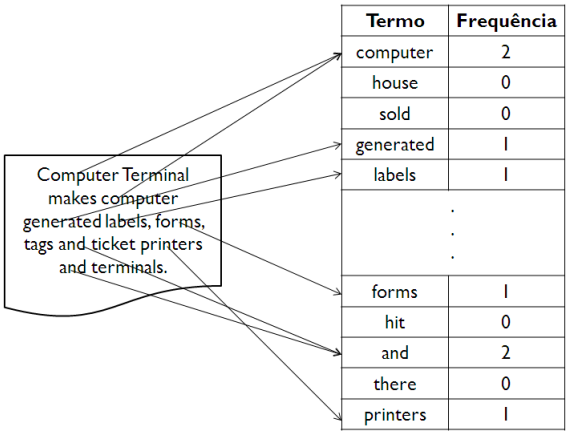


Figura 2.2: Exemplo da representação *Bag-of-Words* (palavras isoladas como unidade das características) com valores das características usando Frequência do Termo.

classificadores. Os classificadores serão apenas uma ferramenta para avaliar o desempenho dos métodos propostos. Portanto, nesta seção será apresentado o classificador estado-da-arte *Support Vector Machine* [45, 46].

Classificação binária pode ser vista como uma tarefa de separar duas classes no espaço de características com um hiperplano. Os pontos acima do hiperplano pertencem à uma classe e os pontos abaixo do hiperplano pertencem à outra classe. Se o problema for linearmente separável, infinitos hiperplanos podem ser utilizados para separar as duas classes. A dificuldade em um problema assim está em encontrar o hiperplano que separa melhor as duas classes.

O *Support Vector Machine* (SVM) é um classificador binário que se baseia na premissa de que o melhor hiperplano separador é aquele que cria a maior margem entre os dados das duas classes. Esta seção apresenta, superficialmente, o *Rigid Margin SVM*, o *Soft Margin SVM* e o SVM não-Linear. Para um estudo mais abrangente sobre SVM, é recomendada a leitura do livro de Vapnik [74] ou do trabalho de Lorena e Carvalho [75].

Os *Rigid Margin SVM* possuem esse nome, pois trabalham com a restrição de não poder existir dados de treinamento entre as margens. Então, dado um conjunto de treinamento $\mathbf{X}$ com N documentos x com algum rótulo $class(x) \in \mathbf{C} = \{-1, +1\}$, existe um hiperplano $f(x) = a^{\mathrm{T}}x + b$ que divide o espaço em duas regiões $a^{\mathrm{T}}x + b > 0$ e $a^{\mathrm{T}}x + b < 0$. A função de classificação é definida por:

$$g(x) = \mathrm{sgn}\big(f(x)\big) = \mathrm{sgn}(a \cdot x + b) \tag{2.4}$$

a função $\mathrm{sgn}(\cdot)$ retorna +1 se o resultado for positivo e -1 se o resultado for negativo. Por se tratar de um *Rigid SVM*, $f(x)$ precisa satisfazer duas condições:

i) Não existir nenhum dado dentro da margem.

ii) A margem seja a maior possível.

Dado que infinitos hiperplanos podem ser obtidos por $f(x)$, todo ponto mais próximo

do hiperplano (x_i) deve satisfazer a condição de $|a \cdot x_i + b| = 1$. Deste modo, encontram-se as inequações:

$$g(x) = \text{sgn}\big(f(x)\big) = \begin{cases} a \cdot x_i + b \geqslant +1 & \text{se } class(x_i) = +1 \\ a \cdot x_i + b \leqslant -1 & \text{se } class(x_i) = -1 \end{cases}$$

que podem ser resumidas por

$$class(x_i)(a \cdot x_i + b) - 1 \geqslant 0, \forall x_i \in \mathbf{X} \tag{2.5}$$

que representa a restrição de não existir nenhum documento dentro da margem.

Dado que a distância entre os dois hiperplanos das margens é $\frac{2}{\|a\|}$ [75], resta maximizar a distância, chegando no problema de otimização:

$$\min_{a,b} \frac{1}{2}\|a\|^2, \tag{2.6}$$

com as restrições da Equação (2.5). Como a função possui apenas um mínimo global, pode ser resolvida pela função Lagrangiana:

$$L(a,b,\alpha) = \frac{1}{2}\|a\|^2 - \sum_i^N \alpha_i \big(class(x_i)(a \cdot x_i + b) - 1\big), \tag{2.7}$$

a função Lagrangiana deve ser minimizada para a e b:

$$\frac{\partial L}{\partial a} = 0 \quad \text{e} \quad \frac{\partial L}{\partial b} = 0. \tag{2.8}$$

Com a resolução dessas equações chega-se em

$$\sum_i^N \alpha_i class(x_i) = 0 \quad \text{e} \quad a = \sum_i^N \alpha_i class(x_i) x_i, \tag{2.9}$$

substituindo esses valores na Equação (2.7), é gerada a equação:

$$\max_{\alpha} \sum_i^N \alpha_i - \frac{1}{2} \sum_i^N \sum_{i'}^N \alpha_i \alpha_{i'} class(x_i) class(x_{i'})(x_i \cdot x_{i'}) \tag{2.10}$$

com as restrições:

$$\begin{cases} \alpha_i \geqslant 0, \forall i = 1, \ldots, N \\ \sum_i^N \alpha_i class(x_i) = 0 \end{cases}$$

Dado a função de decisão da Equação (2.4), chega-se à função de decisão final:

$$g(x) = \text{sgn}\big(f(x)\big) = \text{sgn}\left(\sum_{x_i \in SV} class(x_i) \alpha_i^* x_i \cdot x + b^* \right), \tag{2.11}$$

no qual SV são os documentos que são vetores suporte e b^* é calculado por:

$$b^* = \frac{1}{|SV|} \sum_{x_{i'} \in SV} \left(\frac{1}{class(x_{i'})} - \sum_{x_i \in SV} \alpha_i^* class(x_i) x_i \cdot x_{i'} \right). \tag{2.12}$$

Entretanto, problemas reais dificilmente são linearmente separáveis, devido a ruídos, *outliers* e complexidade. Por isso foi proposta a *Soft Margin SVM*, que introduz as variáveis de folga $\xi_i \in (0,1]$, modificando a Equação (2.5) para:

$$class(x_i)(a \cdot x_i + b) \geqslant 1 - \xi_i, \forall i = 1, \ldots, N. \tag{2.13}$$

Essas variáveis de folga relaxam as restrições impostas, permitindo que alguns documentos de treinamento fiquem entre as margens. Sendo que, assim como a *Rigid Margin SVM*, ainda é necessário maximizar as margens:

$$\min_{a,b,\xi} \frac{1}{2}\|a\|^2 + F\left(\sum_i^N \xi_i\right), \tag{2.14}$$

no qual F é uma constante de peso para a folga. Chega-se novamente na Equação (2.10), mas com as seguintes restrições:

$$\begin{cases} 0 \leqslant \alpha_i \leqslant F, \forall i = 1, \ldots, N \\ \sum_i^N \alpha_i class(x_i) = 0 \end{cases}$$

ξ_i é encontrado isolando-o na Equação (2.13):

$$\xi_i = \max\left\{0, 1 - class(x_i) \sum_{i'}^{N} class(x_{i'}) \alpha_{i'}^* x_{i'} \cdot x_i + b^*\right\}, \tag{2.15}$$

no qual $\max\{\cdot,\cdot\}$ retorna o maior valor dentre os dois parâmetros. Deste modo, o valor de x_i é mantido entre 0 e 1.

Apesar da versatilidade do *Soft Margin SVM*, certos problemas não conseguem ser divididos satisfatoriamente com um hiperplano. Para tal, utiliza-se o SVM não-Linear que trabalha com o chamado *kernel trick*. O uso do *kernel trick* faz com o que o SVM tenha a capacidade de traçar um hiperplano em uma dimensionalidade muito superior à dimensionalidade atual, de modo que ao "retornar"para a dimensionalidade original, existam fronteiras cercando áreas que antes não seria possível. A Figura 2.3 exemplifica esse processo em sequência com os dados originais em duas dimensões da Figura 2.3(a) sendo transformados para a terceira dimensão e nesta nova dimensão um hiperplano é possível de ser traçado para separar as duas classes, vide Figure 2.3(b), e assim ao retornar para a dimensão original na Figure 2.3(c) as classes estão separadas.

O SVM não-Linear utiliza-se das mesmas Equações apresentadas anteriormente, no qual todas as multiplicações entre dois padrões são realizadas com uso de um *kernel*. Isto é, $x_i \cdot x_j$

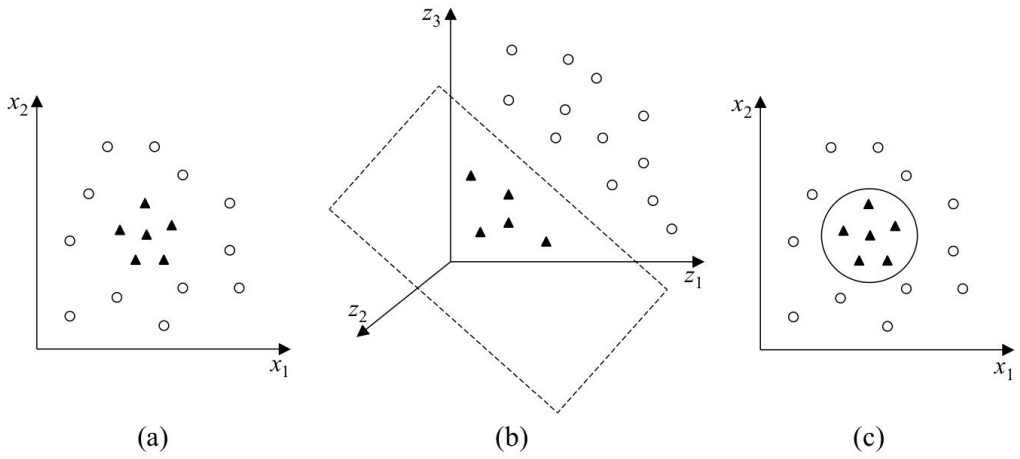


Figura 2.3: Exemplo do SVM não-Linear com *kernel trick*.

se tornará $\Phi(x_i) \cdot \Phi(x_j)$. No qual Φ é uma função de *kernel* que realiza um mapeamento dos dados atuais para um novo espaço de características, com uma quantidade de características normalmente bastante superior ao original. Por exemplo, a Equação (2.10) é alterada para:

$$\max_{\alpha} \sum_{i}^{N} \alpha_i - \frac{1}{2} \sum_{i}^{N} \sum_{i'}^{N} \alpha_i \alpha_{i'} class(x_i) class(x_{i'}) \left(\Phi(x_i) \cdot \Phi(x_{i'}) \right), \tag{2.16}$$

assim como todos os cálculos de produtos vetoriais entre dois padrões, como as Equações (2.11) e (2.12).

2.1.4 Avaliação de Performance

A avaliação de um sistema de Classificação de Documentos é experimental, pois para realizar uma avaliação analítica seria necessário formalizar um problema que é naturalmente informal [2].

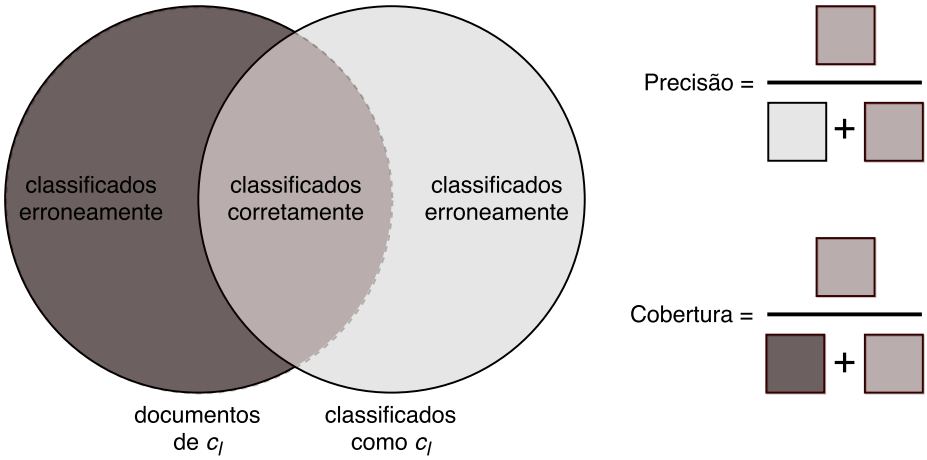


Figura 2.4: Calculo de precisão e cobertura para uma classe c_l.

Existem diversas medidas de avaliação para verificar o desempenho de um classificador. Em Classificação de Documentos, a metodologia mais utilizada baseia-se nas medidas de precisão (do inglês *precision*) e a cobertura (do inglês *recall*) [76], cujos valores são calculados, respectivamente, pelas equações:

$$p_l = \frac{TP_l}{TP_l + FP_l} \qquad \text{e} \qquad r_l = \frac{TP_l}{TP_l + FN_l}, \tag{2.17}$$

no qual as equações são calculadas por classe e as variáveis das equações são de acordo com a Tabela 2.1. A Figura 2.4 ilustra como as medidas são obtidas para uma determinada classe c_l. O valor máximo da precisão ($p_l = 1$) ocorre quando todos os documentos classificados como pertencentes à classe c_l são de fato desta classe. Enquanto que o valor máximo da cobertura ($r_l = 1$) ocorre quando todos os documentos da classe c_l são classificados como pertencente à classe c_l.

Tabela 2.1: Tabela de Contingência da classe c_l

Classe c_l		Resposta Real	
		Sim	Não
Resposta do Classificador	Sim	TP_l	FP_l
	Não	FN_l	TN_l

Encontrar um valor máximo de cobertura é trivial, basta classificar todos os documentos como pertencentes à classe c_l. Portanto, o valor de cobertura sozinho não é suficiente para indicar o desempenho de um sistema de Classificação de Documentos. Portanto, é utilizada a *F-Measure* (ou F1), que unifica os valores de cobertura e precisão pela equação:

$$F1_l = \frac{2 \times p_l \times r_l}{(p_l + r_l)}. \tag{2.18}$$

Como os sistemas de Classificação de Documentos lidam, normalmente, com diversas classes, apenas o valor isolado da $F1$ de cada classe não mede o desempenho do sistema completo. Portanto, para avaliar o desempenho médio do sistema, são utilizadas a macro-média e a micro-média. Sendo estas as medidas de avaliação mais utilizadas na literatura [13, 77, 32, 78, 62, 79, 9, 41, 80]. A macro-média (macro-F1) é obtida com uma média direta dos valores $F1$ de cada classe:

$$\text{Macro-F1} = \sum_{l}^{N} \frac{F1_l}{N}, \tag{2.19}$$

e a micro-média é calculada substituindo os valores presentes na Tabela 2.1 pelos valores da tabela de contingência global (Tabela 2.2) e utilizando-os nas Equações (2.17), e (2.18). Deste modo, serão obtidos os valores de *micro-precision*, *micro-recall* e micro-F1, sendo este último uma medida de avaliação do sistema.

Tabela 2.2: Tabela de Contingência Global

Conjunto de Classes $\mathscr{C} = \{c_1, \ldots, c_C\}$		Resposta Real	
		Sim	Não
Resposta do	Sim	$TP = \sum_{l=1}^{N} TP_l$	$FP = \sum_{l=1}^{N} FP_l$
Classificador	Não	$FN = \sum_{l=1}^{N} FN_l$	$TN = \sum_{l=1}^{N} TN_l$

Em resumo, a macro-F1 é uma média não ponderada pela quantidade de documentos presentes nas classes, enquanto a micro-F1 é uma média ponderada pela quantidade de documentos presentes nas classes. Sendo assim, um sistema de Classificação de Documentos é avaliado por ambas as medidas, pois apenas uma delas não descreve totalmente o desempenho de um sistema que possua classes desbalanceadas (diferentes quantidades de documentos). Os valores são todos no intervalo $[0,1]$, e quanto maior, melhor a classificação.

2.2 Seleção de Características

A redução de dimensionalidade é, normalmente, obtida por duas maneiras distintas: extração e seleção [2, 5]. Ambas as abordagens têm como objetivo obter um conjunto de características $\mathbf{F}'$ baseado num conjunto de características original $\mathbf{F}$ de modo que $|\mathbf{F}'| \ll |\mathbf{F}|$. Sendo que a extração de características gera um novo conjunto baseado nas características originais, enquanto que a seleção de características encontra um subconjunto das características originais.

As características do novo conjunto $\mathbf{F}'$, obtido pela extração de características, são sintéticas, isto é, são unidades desconhecidas (não mais palavras, frases ou letras) geradas pelo processo de extração. Semântica latente [81, 39, 82], *clustering* de termos [83, 64, 84, 78] e Análise de Componentes Principais [82, 39] são técnicas de extração de características aplicadas em Classificação de Documentos.

Enquanto que as características do novo conjunto $\mathbf{F}'$, obtido pela seleção de características, possuem o mesmo sentido semântico do conjunto original. Portanto, nenhuma informação nova é gerada, há simplesmente uma redução do conjunto original por meio da seleção.

A seleção de características pode ser global ou local. Na seleção local, é realizada uma seleção para cada classe existente, cada seleção irá considerar uma classe como positiva e as demais como negativas. Portanto, a seleção local terá um subconjunto de características $\mathbf{F}'_l$ para cada classe c_l [85] e estes subconjuntos serão utilizados por classificadores binários e combinados posteriormente. Já a seleção global considera todas as classes de uma única vez, possuindo apenas um subconjunto de características $\mathbf{F}'$ [9], que pode ser utilizado para classificações multi-classe.

O processo de seleção de características pode ser baseado, de modo geral, em métodos de filtragem ou métodos *wrappers*. Métodos de filtragem [86] selecionam os termos baseados em algum algoritmo ou propriedade estatística presente nos dados. Enquanto que os métodos

wrapper [87] são mais complexos e custosos, pois utilizam classificadores para avaliar diversos subconjuntos de $\mathbf{F}'$, de modo a escolher o subconjunto com melhor resultado em um conjunto de validação. Os métodos *wrappers* conseguem, na maioria dos casos, melhor desempenho, comparado com os métodos de filtragem. Entretanto, em Classificação de Documentos, os métodos de filtragem são mais utilizados, devido ao custo computacional elevado dos métodos *wrapper* [88].

Vários algoritmos de filtragem foram propostos ao longo dos anos. Dois dos primeiros foram o FOCUS [89, 90, 91] e o RELIEF [8, 92], que não conseguiram ganhar espaço em problemas de Classificação de Documentos, por seu elevado custo computacional ao lidar com alta dimensionalidade [8]. A abordagem mais utilizada em Classificação de Documentos é a *Variable Ranking* [16, 79, 93, 94, 62, 9], apresentada na seção a seguir (Seção 2.2.1).

2.2.1 *Variable Ranking*

Até o início dos anos 90, a abordagem típica para seleção de características [8] era utilizar uma função $J(\mathbf{F}', \mathbf{X})$. Esta função avalia a qualidade do subconjunto de características $\mathbf{F}'$ com relação ao conjunto de treinamento $\mathbf{X}$ [95]. Um subconjunto $\mathbf{F}'_1$ é considerado melhor que o subconjunto $\mathbf{F}'_2$ se $J(\mathbf{F}'_1, \mathbf{X}) > J(\mathbf{F}'_2, \mathbf{X})$. Portanto, é necessário avaliar diversos conjuntos $\mathbf{F}'$ com a função $J(\mathbf{F}', \mathbf{X})$ para encontrar uma boa seleção.

A abordagem modificou-se com o trabalho de Lewis e Ringuette [37], que utilizou *Information Gain* para selecionar as características. Mais tarde, foram usadas *Mutual Information* e *Chi-Squared Statistics* [82]. A metodologia tornou-se amplamente utilizada e métricas foram comparadas em alguns trabalhos [16, 79]. Devido a sua simplicidade, esta metodologia sempre foi chamada pelo nome da métrica utilizada para realizar a seleção. Neste trabalho, essa metodologia será intitulada de *Variable Ranking* (VR) [96].

O Algoritmo 1 descreve o funcionamento do método VR. Nele há um conjunto de treinamento $\mathbf{X}$, composto por documentos, cada documento x_i é representado por V características, sendo w_h a h-ésima característica do vocabulário. O vetor S representa a importância de cada característica e o vetor SN é uma cópia do vetor S, utilizado apenas para não corromper os dados do vetor original.

Apesar das 17 linhas do algoritmo, seu funcionamento é bastante simples. O algoritmo requer uma entrada de quantas características serão selecionadas (m). Uma FEF (do inglês *Feature Evaluation Function*, detalhes na Seção 2.2.2) [66, 16] para calcular a importância de cada termo (linhas 2-4). Então, são selecionados os m termos com os maiores valores na FEF (linhas 5-16). Por fim, o algoritmo retorna um vetor com os índices das características selecionadas (linha 17).

Outras implementações deste método modificam o parâmetro de entrada m por um limiar ponto flutuante. Esse limiar definirá não mais a quantidade de características, mas sim qual será o ponto de corte – com base na FEF – para que uma característica seja inserida, ou não, no novo

Algoritmo 1 *Variable Ranking*

Entrada: Integer $m > 0$

Load training set **X**
for $h = 1$ to V **do** // Calcula FEF para cada termo
  $S_h = \text{FEF}(w_h)$
end for
for $i = 1$ to m **do** // Seleciona as m características de maior FEF.
  $SN = S$
  $bestscore = 0.0$
  for $h = 1$ to V **do**
    if $SN_h > bestscore$ **then**
      $bestscore = SN_h$
      $bestfeature = h$
    end if
  end for
  $SN_{bestfeature} = 0$
  $FS_i = bestfeature$
end for
return FS

subconjunto. Esse tipo de implementação é interessante caso o desenvolvedor conheça bem a FEF utilizada, escolhendo um limiar capaz de selecionar apenas as características relevantes.

Apesar do método VR (Seção 2.2.1) ser amplamente utilizado, essa abordagem possui uma grande dificuldade: encontrar o melhor valor para m. Todos os subconjuntos de características encontrados com diferentes valores de m devem ser testados para encontrar o melhor subconjunto. Entretanto, isso transforma o método numa busca exaustiva, sendo necessário avaliar cada subconjunto com um classificador. Deste modo, um método de filtragem passa a funcionar como um método *wrapper*, perdendo a vantagem do baixo custo computacional. Portanto, para evitar a busca exaustiva, apenas alguns valores de m (definidos pelo usuário) são avaliados, e o melhor selecionado, sendo improvável obter o melhor subconjunto.

Outra dificuldade é utilizar as FEFs como verdade absoluta. O método faz sua seleção com base apenas nos valores da FEF escolhida, se esta não for adequada, a seleção sofrerá as consequências. A utilização de uma FEF inadequada, juntamente com um baixo valor para m pode acarretar em documentos sem representação, isto é, o documento possui todas as características com valor zero. Esse é um problema grave, pois um documento sem representação não poderá ser avaliado corretamente para qual classe pertence e será, provavelmente, classificado de modo aleatório.

2.2.2 *Feature Evaluation Function*

Diversas *Feature Evaluation Function*s (FEFs) foram propostas no decorrer dos anos e estudos comparativos foram realizados sobre elas na literatura [16, 79, 93, 94]. Essas funções são baseadas em medidas estatísticas que utilizam as probabilidades de um termo w com relação aos

documentos de uma classe c_l, no qual $l = 1, 2, \ldots, C$. As seguintes probabilidades são utilizadas nestes cálculos:

- $P(w|c_l)$ é a probabilidade do termo w estar presente em documentos da classe c_l;
- $P(w|\bar{c}_l)$ é a probabilidade do termo w não estar presente em documentos da classe c_l;
- $P(\bar{w}|c_l)$ é a probabilidade de todos os termos – exceto o termo w – estarem presentes nos documentos da classe c_l;
- $P(\bar{w}|\bar{c}_l)$ é a probabilidade de todos os termos – exceto o termo w – não estarem presentes nos documentos da classe c_l;
- $P(c_l)$ e $P(w)$ são as probabilidades *a priori* da classe c_l e do termo w, respectivamente.

Algumas das *feature evaluation function* são:

- ***Bi-Normal Separation***

  Proposta por George Forman [93, 97], a *Bi-Normal Separation* (BNS) é uma métrica definida pela equação:

  $$\mathrm{BNS}(w) = \sum_{l=1}^{C} \left| F^{-1}(P(w|c_l)) - F^{-1}(P(w|\bar{c}_l)) \right|, \tag{2.20}$$

  na qual $F^{-1}(\cdot)$ é função de probabilidade acumulativa inversa de uma distribuição normal padrão, também chamada de *z-score*. A BNS, portanto, mede a separação entre os dois limiares encontrados pelas funções.

  Para evitar o valor indefinido de $F^{-1}(0)$, Forman recomenda que o valor 0 seja substituído por $0,0005$ [93].

- ***Chi-Squared Statistics***

  Usada constantemente em problemas de Classificação de Documentos [66, 16], apresentou bons resultados ao ser comparada com outras métricas clássicas [79]. A *Chi-Squared Statistics* (CHI) é definida por:

  $$\mathrm{CHI}(w) = \sum_{l=1}^{C} \frac{\left[P(w|c_l)P(\bar{w}|\bar{c}_l) - P(w|\bar{c}_l)P(\bar{w}|c_l)\right]^2}{P(w)P(\bar{w})P(c_l)P(\bar{c}_l)}, \tag{2.21}$$

  na qual as probabilidades seguem a mesma descrição apresentada no início desta seção.

  O valor obtido pela Equação (2.21) é referente à independência do termo w com relação às classes existentes. Portanto, quanto menor o valor retornado pela CHI, mais

independente é aquele termo. Como uma boa seleção se caracteriza por encontrar termos que dependam da classe, isto é, que possuam uma forte relação com certas classes, serão selecionadas as características com elevado valor na CHI [98].

- ***Odds Ratio***

  A *Odds Ratios* [66, 9, 99] é uma função de domínio binário, descrita por:

$$\mathrm{OR}(w) = \log \frac{P(w|c_l)(1-P(w|\bar{c}_l))}{P(w|\bar{c}_l)(1-P(w|c_l))}. \tag{2.22}$$

  Neste trabalho será usada a *Multi-class Odds Ratio* (MOR) [9], sua versão multi-classe, descrita por:

$$\mathrm{MOR}(w) = \sum_{l=1}^{C} \left| \log \frac{P(w|c_l)(1-P(w|\bar{c}_l))}{P(w|\bar{c}_l)(1-P(w|c_l))} \right|. \tag{2.23}$$

- ***Class Discriminanting Measure***

  Proposta por Chen et al. [9], a *Class Discriminanting Measure* (CDM) é calculada por:

$$\mathrm{CDM}(w) = \sum_{l=1}^{C} \left| \log \frac{P(w|c_l)}{P(w|\bar{c}_l)} \right|. \tag{2.24}$$

  É notável sua semelhança com a MOR, pois a CDM é sua versão simplificada [9].

- ***Information Gain***

  Juntamente com a CHI, a *Information Gain* (IG) foi reportada como uma das melhores medidas em problemas de multi-classe [16]. Seu uso permanece frequente em trabalhos recentes [9, 10].

  Sua equação é descrita de várias formas na literatura, podendo ser descrita como:

$$\mathrm{IG}(w) = \sum_{l=1}^{C} P(w|c_l) \log \frac{P(w|c_l)}{P(c_l)} + \sum_{l=1}^{C} P(\bar{w}|c_l) \log \frac{P(\bar{w}|c_l)}{P(c_l)}. \tag{2.25}$$

  A Equação (2.25) terá resultado 0 quando for completamente independente da classe e crescerá monotonicamente de acordo com sua dependência [100].

- **Outras Métricas**

  Além das já listadas, outras FEFs são: *Accuracy* [93] e sua versão balanceada [94], *Mutual Information* [101, 78, 16], *Gini index* e suas diversas reformulações [102, 103, 77], *Term Strength* proposta por Wilbur e Sirotkin [104] e utilizada em diversos outros trabalhos [105, 106, 16].

2.2.3 *Maximum f Features per Document*

Como foi citado na Seção 2.2.1, o método VR possui algumas dificuldades. O *At Least One FeaTure* (ALOFT) [10] foi proposto para lidar com essas dificuldades. Sua ideia central é selecionar pelo menos uma característica por documento, evitando o problema de documentos sem representação; e encontrar a quantidade de características no decorrer da execução, em vez de ser um parâmetro do algoritmo. A diferença do *Maximum f Features per Document* (MFD) para o ALOFT é a inserção do parâmetro f, para que sejam selecionadas f características por documento, em vez de apenas uma. Nesta seção é descrito o funcionamento do MFD [13], por ser uma versão mais geral do ALOFT.

Algoritmo 2 *Maximum f Features per Document* (MFD)

Entrada: $f \geq 1$
1: Load training set $\mathbf{X}$
2: **for** $h = 1$ to V **do** // Calcula FEF para cada termo
3: $S_h = \text{FEF}(w_h)$
4: **end for**
5: $m = 0$
6: FS an empty vector
7: **for all** $x_i \in \mathbf{X}$ **do** // Varre todos os documentos
8: **for** $n = 1$ to f **do** // Seleciona as f características valoradas de maior FEF.
9: $bestscore = 0.0$
10: **for** $h = 1$ to V **do**
11: **if** $w_h^{x_i} > 0$ and $S_h > bestscore$ **then**
12: $bestscore = S_h$
13: $bestfeature = h$
14: **end if**
15: **end for**
16: **if** $bestfeature \notin FS$ **then** // Insere apenas se não tiver sido selecionada ainda
17: $m = m + 1$
18: $FS_m = bestfeature$
19: $S_{bestfeature} = -1 \times S_{bestfeature}$
20: **end if**
21: **end for**
22: **for** $h = 1$ to V **do**
23: $S_h = |S_h|$
24: **end for**
25: **end for**
26: **return** FS

O Algoritmo 2 descreve o método MFD. O conjunto de treinamento $\mathbf{X}$ é composto por N documentos, no qual cada documento x_i é representado por V características indexadas como $w_h^{x_i}$, sendo $h = 1, \ldots, V$. O algoritmo é dividido em três passos:

i) *Input*: requer um parâmetro f, inteiro positivo, que define a quantidade de características que serão selecionadas em cada documento;

ii) Linhas 2-4: calcula S (vetor de tamanho V), para cada característica, com alguma FEF (descritas na Seção 2.2.2). O valor de S_h representa a importância da h-ésima característica (w_h). A FEF utilizada influencia diretamente no quão efetiva será a seleção;

iii) Linhas 5-6: inicializa $m = 0$ e o vetor FS como vazio;

iv) Linhas 7-21: gera o vetor FS, que representa o novo conjunto de características. São selecionadas f características por documento, baseado nos maiores valores de S dentre todas as características com valor diferente de zero. As características selecionadas que já estiverem no vetor FS são ignoradas. No final desta etapa, FS terá tamanho m, os valores em FS representam os índices das características selecionadas do conjunto original;

v) Linha 22: retorna o vetor FS com as características selecionadas.

O algoritmo possui alguns trechos inseridos a título de implementação:

i) Linha 19: o valor que representa a importância do termo selecionado se torna negativo no vetor FS. Com o valor negativo, a característica não será escolhida novamente;

ii) Linhas 22-24: os valores do vetor S são restaurados ao seu valor original (positivo).

Essas linhas adicionadas no algoritmo incrementam um pouco mais o tempo de processamento do método. Uma otimização seria restaurar (linhas 22-24) os valores apenas dos índices presentes no vetor FS, pois apenas eles teriam se tornado negativos na operação efetuada na linha 19.

No caso de $f = 1$ o subconjunto de características selecionado será o mesmo do algoritmo ALOFT [10].

Trabalhos mais recentes, inspirados no ALOFT, melhoram o desempenho do MFD, tais como o *Category-dependent Maximum f Features per Document - Reduced* (cMFDR) [14] e acafsa [15].

2.3 *Dissimilarity Representation*

A *Dissimilarity Representation* (DR), em português Representação por Dissimilaridade, proposta por Pekalska e Duin [23] possui três abordagens: pré-topológica, espaço de dissimilaridade, e *embedding*. A abordagem pré-topológica utiliza de modo direto ou indireto o espaço obtido pela dissimilaridade entre os exemplos. Deste modo, é possível realizar a classificação verificando vizinhança e proximidade entre os exemplos. O k vizinhos mais próximos pode ser considerado um classificaor que utiliza-se dessa abordagem. A abordagem do espaço de dissimilaridade será explicada em detalhes nessa seção, pois é a abordagem utilizada em todo

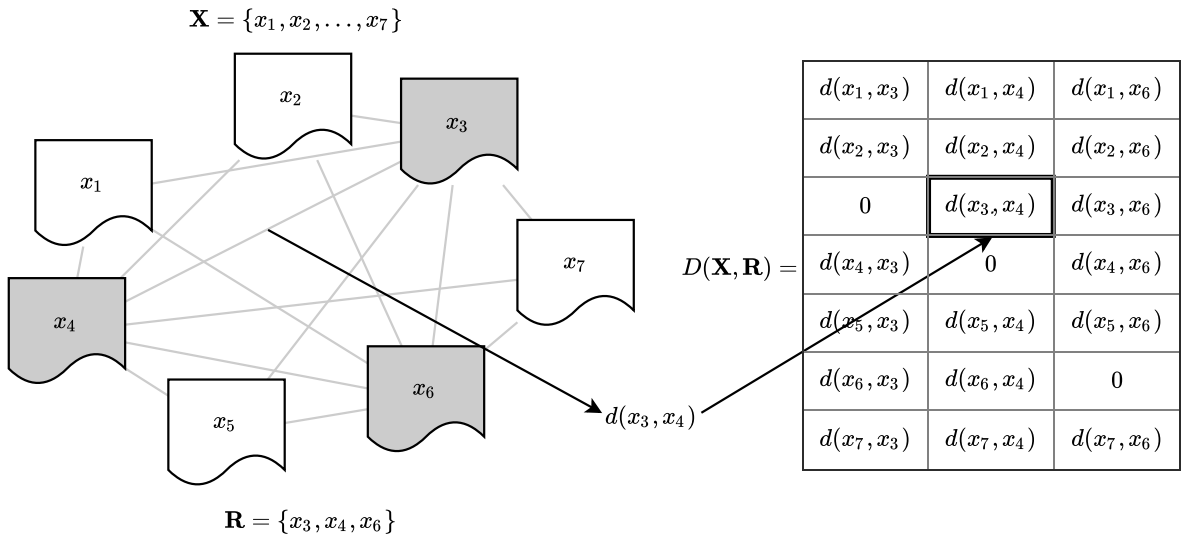


Figura 2.5: Criação da matriz de dissimilaridade. **X** é o conjunto de treinamento, **R** é o conjunto de representação e $D(\mathbf{X}, \mathbf{R})$ é a matriz de dissimilaridade resultante.

este trabalho. Já a abordagem *embedding* requer uma série de etapas de modo que o classificador é embutido nas dissimilaridade. O processo é iniciado com uma representação simétrica utilizando parte dos exemplos e depois, os exemplos restantes são projetados no plano obtido. O motivo de utilizar espaço de dissimilaridade em vez das outras duas abordagens é devido a sua universalidade, isto é, a capacidade de usar a *Dissimilarity Representation* com qualquer métrica (até mesmo de similaridade) e com qualquer classificador que use vetores de características como entrada. Essa universalidade não ocorre com nenhuma das outras duas abordagens, pois a abordagem pré-topologica utiliza especificamente apenas as distância como base para a classificação e a abordagem *embedding* requer modificações no classificador dependendo da métrica utilizada, inclusive permitindo apenas a utilização de métricas bastante específicas. Detalhamento sobre as outras duas abordagens podem ser encontrados na tese de Pekalska [107]. Portanto, toda e qualquer referência à *Dissimilarity Representation* nesta proposta refere-se à segunda abordagem proposta por Pekalska e Duin.

A *Dissimilarity Representation* é um mapeamento definido pelo cálculo de dissimilaridade de um documento (ou conjunto de documentos) para todos os documentos de um conjunto de representação. O conjunto de representação **R** é um subconjunto do conjunto de treinamento ($\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$) composto por Q protótipos ($\mathbf{R} = \{p_1, \ldots, p_Q\}$). O conjunto **R** pode ser obtido por meio de qualquer algoritmo de seleção de protótipos executado sobre o conjunto **X**.

A Figura 2.5 mostra um exemplo de como a matriz de dissimilaridade é obtida. Neste exemplo temos o conjunto de treinamento **X** de tamanho $N = 7$ e seu subconjunto **R** (conjunto de representação) de tamanho $Q = 3$. É calculada a distância $d(\cdot, \cdot)$ de todos os documentos de **X** para todos os documentos de **R**, resultando em uma matriz de dissimilaridade $D(\mathbf{X}, \mathbf{R})$. Generalizando, o resultado é uma matriz de dimensões $N \times Q$:

$$D(\mathbf{X},\mathbf{R}) = \begin{bmatrix} d(x_1,p_1) & d(x_1,p_2) & \dots & d(x_1,p_Q) \\ d(x_2,p_1) & d(x_2,p_2) & \dots & d(x_2,p_Q) \\ \vdots & \vdots & \ddots & \vdots \\ d(x_N,p_1) & d(x_N,p_2) & \dots & d(x_N,p_Q) \end{bmatrix},$$

no qual cada linha representa um documento, cada coluna representa uma característica e $d(\cdot,\cdot)$ é a dissimilaridade entre dois documentos. Portanto, essa matriz de dissimilaridade é a própria matriz de dados. Deste modo, a *Dissimilarity Representation* pode ser considerada uma transformação do espaço V (quantidade de termos original) para o espaço Q (quantidade de protótipos).

A principal razão de utilizar *Dissimilarity Representation* é obter vantagem das propriedades do cálculo de distâncias. Distâncias entre documentos de mesma classe devem ser pequenas, enquanto que distâncias entre documentos de classes diferentes devem ser grandes [108]. Portanto, ter um protótipo p_k, que represente bem uma classe específica, gera uma característica poderosa, pois as distâncias para p_k provavelmente terão um bom poder de discriminação entre as classes.

Outra razão para o uso da *Dissimilarity Representation* é para reduzir o tamanho do espaço de características original. A representação *Bag-of-Words* sofre de matriz de dados esparsa, alta dimensionalidade e alta taxa de característica-por-instância. No espaço de dissimilaridade todos esses problemas são reduzidos. Matriz de dados esparsa, pois $d(x_i,p_k)$ é zero apenas quando ambos os documentos são iguais, esse caso raramente ocorre; alta dimensionalidade, pois Q é normalmente menor do que V em problemas de Classificação de Documentos; e taxa de característica-por-instância, pois $Q \leqslant N$. Algumas dessas vantagens podem ser observadas na Figura 1.1.

2.4 *Dichotomy Transformation*

A *Dichotomy Transformation* (DT) proposta por Cha e Srihari [24] transforma um problema multi-classe em um problema binário (duas classes). Sua proposta inicial foi para resolver um problema de identificação de caligrafia, de modo que o problema original possuía várias classes (autores) e foi transformado em um único problema binário (autoria ou não-autoria). Assim, em vez de verificar se uma determinada caligrafia era de um dos autores disponíveis, a verificação era feita com um par de caligrafias e classificado como sendo ambas do mesmo autor ou não. DT tem sido usado não apenas para identificação de caligrafias [109, 110, 111] mas também em verificação de voz [112], biometria de íris [113], e biometria de digitação [114]. Nesta seção é descrito seu funcionamento utilizando a nomenclatura de Classificação de Documentos.

A utilização da DT requer dois conjuntos de documentos como entrada, obtendo como saída um novo conjunto de documentos com duas classes (positiva e negativa). Dados os seguintes conjuntos de entrada: conjunto **A** e conjunto **B**. Cada documento $a_i \in \mathbf{A} = \{a_1, a_2, \dots, a_A\}$ é

composto por um vetor de características $w^{a_i} = [w_1^{a_i}, w_2^{a_i}, \ldots, w_V^{a_i}]^{\mathrm{T}}$ com termos obtidos via *Bag-of-Words*; e a classe do documento $class(a_i) \in \mathbf{C} = \{c_1, c_2, \ldots, c_C\}$. Enquanto que cada documento $b_j \in \mathbf{B} = \{b_1, b_2, \ldots, b_B\}$ utiliza também o *Bag-of-Words*, possuindo o vetor de características $w^{b_j} = [w_1^{b_j}, w_2^{b_j}, \ldots, w_V^{b_j}]^{\mathrm{T}}$. A saída da DT é o conjunto $\mathbf{Z}$.

O conjunto $\mathbf{Z}$ é composto de $A \times B$ documentos, um para cada par $\langle a_i, b_j \rangle$ dos documentos de $\mathbf{A}$ e $\mathbf{B}$, no qual cada par de documentos é considerado positivo ($class(a_i) = class(b_j)$) ou negativo ($class(a_i) \neq class(b_j)$). Cada documento $z_{i,j} \in \mathbf{Z}$ é obtido calculando $dt(a_i, b_j)$ para cada par $a_i \in \mathbf{A}$ e $b_j \in \mathbf{B}$. A metodologia mais comum para calcular esta função é usar o valor absoluto da diferença:

$$dt(a_i, b_j) = \begin{bmatrix} |w_1^{a_i} - w_1^{b_j}| \\ |w_2^{a_i} - w_2^{b_j}| \\ \vdots \\ |w_V^{a_i} - w_V^{b_j}| \end{bmatrix} \tag{2.26}$$

no qual $w_h^{a_i}$ e $w_h^{b_j}$ são as h-ésimas características dos documentos a_i e b_j respectivamente, e V é o número de características.

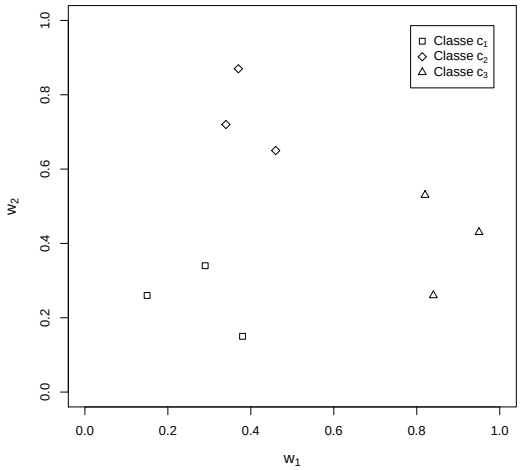
A Figura 2.6 ilustra um exemplo da *Dichotomy Transformation* que utiliza os conjuntos de entrada: $\mathbf{A}$ (Figura 2.6(a)) com 9 documentos ($A = 9$), e $\mathbf{B}$ (Figura 2.6(b)) com 6 documentos ($B = 6$), ambos possuindo 3 classes ($C = 3$). Figura 2.6(c) apresenta conjunto resultante $\mathbf{Z}$ que possui 54 documentos ($A \times B$); esta abordagem possui uma propriedade interessante, pois aumenta a quantidade de documentos disponíveis. Os documentos de $\mathbf{Z}$ são divididos em duas classes: positiva (•), são os 18 documentos próximos da origem, obtidos de pares que possuem a mesma classe; e negativa (○), são os 36 documentos mais afastados da origem, obtidos de pares que possuem classes diferentes. Todos os conjuntos são representados por 2 características (V) exibidas no eixo x (w_1) e no eixo y (w_2).

Um classificador pode ser treinado com os documentos do conjunto $\mathbf{Z}$ para verificar se um par de documentos pertence à mesma classe (positivo) ou não (negativo).

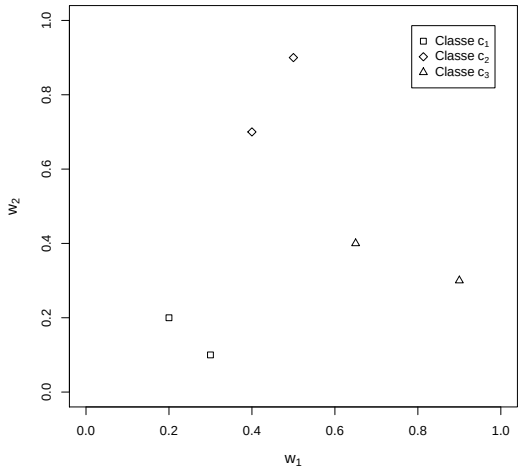
2.5 Seleção de Protótipos

Algoritmos de seleção de protótipos reduzem o tamanho original do conjunto de treinamento. Essa redução é obtida ao selecionar um subconjunto dos documentos de treinamento que melhorem, ou pelo menos não prejudiquem, o desempenho do classificador. A seleção de protótipos tem a capacidade de melhorar o desempenho, reduzir o armazenamento e diminuir o tempo de treinamento. Existem três tipos de algoritmos de seleção de protótipos: filtro, condensação e híbrido [115].

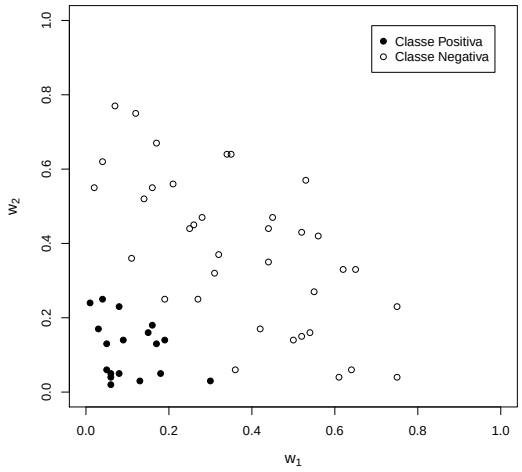
Os algoritmos de seleção de protótipos do tipo filtro removem documentos ruidosos. Documentos ruidosos são aqueles documentos com dados incorretos ou que simplesmente são



(a) Conjunto de entrada **A**



(b) Conjunto de entrada **B**



(c) Conjunto de saída **Z**

Figura 2.6: Examplo da *Dichotomy Transformation* com os conjuntos de entrada **A** (a) e **B** (b), e o conjunto de saída **Z** (c). Todos os conjuntos são representados por duas características (w_1 e w_2).

diferentes demais dos outros (*outliers*). Com a remoção desses documentos é possível aumentar o desempenho, pois esses documentos podem atrapalhar o treinamento do classificador. Métodos de condensação removem documentos irrelevantes. Documentos irrelevantes são aqueles que possuem informação redundante ou simplesmente não impactam na classificação. Com a remoção desses documentos é possível reduzir o tamanho do conjunto de treinamento sem dificultar o treinamento do classificador, em alguns casos facilitando esse treinamento. Finalmente, métodos híbridos visam tanto remover documentos ruidosos como documentos irrelevantes. O resultado esperado é um subconjunto menor com alguma melhora no desempenho.

As próximas subseções descrevem os algoritmos de seleção de protótipos usados nos experimentos. Tendo em vista que k foi utilizado como outra variável neste trabalho, os vizinhos

do *k*NN serão tratados pela variável v. Portanto, a função $NN(d, \mathbf{B}, v)$ retorna v vizinhos mais próximos do documento d no conjunto $\mathbf{B}$. Enquanto que $class(NN(\cdot,\cdot,\cdot))$ retorna a classe da maioria dos vizinhos.

2.5.1 *Condensation Nearest Neighbor*

Proposto por Hart [116], o *Condensation Nearest Neighbor* (CNN) é um método de condensação [115] que prioriza manter exemplos das regiões limitantes das classes, isto é, exemplos da borda. O Algoritmo 3 mostra o funcionamento do método. O conjunto de protótipos $\mathbf{R}$ inicia com um exemplo aleatório e crescerá enquanto for modificado no laço das Linhas 3-7. Todos os documentos do conjunto de treinamento, excluindo os documentos já selecionado para $\mathbf{R}$, serão verificados. O documento é selecionado se for classificado incorretamente com base no atual conjunto $\mathbf{R}$.

Algoritmo 3 *Condensation Nearest Neighbor* (CNN)

Entrada: $v \geqslant 1$ e conjunto de treinamento $\mathbf{X}$

1: $\mathbf{R} = \{x_{rand(N)}\}$
2: **while** $\mathbf{R}$ for modificado **do**
3: **for all** $x_i \in \mathbf{X} - \mathbf{R}$ **do**
4: **if** $class(x_i) \neq class(NN(x_i, \mathbf{R}, v))$ **then**
5: $\mathbf{R} = \mathbf{R} \cup \{x_i\}$
6: **end if**
7: **end for**
8: **end while**
9: retorna $\mathbf{R}$

2.5.2 *Reduced Nearest Neighbor*

Proposto por Gates [117] como uma melhoria do CNN, o *Reduced Nearest Neighbor* (RNN) executa o CNN para gerar um conjunto inicial e remove exemplos desnecessários para consistência da vizinhança. O RNN é um método de condensação [115] e seu funcionamento é mostrado no Algoritmo 4. O RNN funciona de modo à reduzir o conjunto $\mathbf{R}$ obtido por execução inicial do CNN (Linha 1). Então, um laço é executado para verificar cada documento em $\mathbf{R}$ (Linhas 2-10). O documento é atual é removido e é verificado se existe alguma classificação incorreta, caso exista, o documento atual é mantido fora do conjunto $\mathbf{R}$; caso contrário, o documento é recolocado. Assim o RNN sempre seleciona menos que o CNN.

2.5.3 *Edited Nearest Neighbor*

Proposto por Wilson [118], o *Edited Nearest Neighbor* (ENN) inicializa com todos os documentos e remove os exemplos classificados incorretamente com relação ao conjunto de treinamento. O ENN é um método de filtro [115] e seu funcionamento é mostrado no Algoritmo 5.

Algoritmo 4 *Reduced Nearest Neighbor* (RNN)

Entrada: $v \geqslant 1$ e conjunto de treinamento $\mathbf{X}$

1: $\mathbf{R} = CNN(v, \mathbf{X})$
2: **for all** $p_k \in \mathbf{R}$ **do**
3: $\quad \mathbf{R} = \mathbf{R} - \{p_k\}$
4: $\quad$ **for all** $x_i \in \mathbf{X}$ **do**
5: $\quad\quad$ **if** $class(x_i) \neq class(NN(x_i, \mathbf{R}, v))$ **then**
6: $\quad\quad\quad \mathbf{R} = \mathbf{R} \cup \{p_k\}$
7: $\quad\quad\quad$ encerra **for all** atual
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: **end for**
11: retorna $\mathbf{R}$

Ao contrário do CNN e RNN, o ENN inicia o conjunto $\mathbf{R}$ com todo o conjunto de treinamento e irá reduzir a medida que o algoritmo é executado. Deste modo, cada protótipo será verificado (Linhas 2-6) e removido caso sua classificação seja incorreta com relação à todo o conjunto de treinamento, como se ele fosse considerado um protótipo que não representa bem a identidade de classes atuais do problema.

Algoritmo 5 *Edited Nearest Neighbor* (ENN)

Entrada: $v \geqslant 1$ e conjunto de treinamento $\mathbf{X}$

1: $\mathbf{R} = \mathbf{X}$
2: **for all** $p_k \in \mathbf{R}$ **do**
3: $\quad$ **if** $class(p_k) \neq class(NN(p_k, \mathbf{X}, v))$ **then**
4: $\quad\quad \mathbf{R} = \mathbf{R} - \{p_k\}$
5: $\quad$ **end if**
6: **end for**
7: retorna $\mathbf{R}$

2.5.4 *All k Nearest Neighbor*

Proposto por Tomek [119], o All-kNN é um método de filtro [115] e seu funcionamento é mostrado no Algoritmo 6. Assim como no ENN, o All-kNN inicia o conjunto $\mathbf{R}$ com todos os documentos do conjunto de treinamento. Sua avaliação é feita de modo de vizinhança incremental, e se em qualquer vizinhança o documento for classificador incorretamente, ele é removido.

2.5.5 *Instance-Based Learning Algorithm 2*

Proposto por Aha et al. [120], o *Instance-Based Learning Algorithm 2* (IB2) funciona bastante similar ao CNN, sem realizar múltiplas iterações sempre que houver modificação. O Algoritmo 7 mostra o funcionamento do método. IB2 age de modo incremental, assim o

Algoritmo 6 *All k Nearest Neighbor* (All-kNN)

Entrada: $v \geqslant 1$ e conjunto de treinamento $\mathbf{X}$

1: $\mathbf{R} = \mathbf{X}$
2: **for** $a = 1$ até v **do**
3: **for all** $p_k \in \mathbf{R}$ **do**
4: **if** $class(p_k) \neq class(NN(p_k, \mathbf{X}, a))$ **then**
5: marque p_k
6: **end if**
7: **end for**
8: **end for**
9: retorna $\mathbf{R}$ removendo quaisquer p_k marcado

conjunto final é iniciado vazio, sendo preenchido com os exemplos que forem classificadores incorretamente pelos seus vizinhos mais próximos. Funcionamento bastante similar ao CNN.

Algoritmo 7 *Instance-Based Learning Algorithm 2* (IB2)

Entrada: $v \geqslant 1$ e conjunto de treinamento $\mathbf{X}$

1: $\mathbf{R} = \emptyset$
2: **for all** $x_i \in \mathbf{X}$ **do**
3: **if** $class(x_i) \neq class(NN(x_i, \mathbf{R}, v))$ **then**
4: $\mathbf{R} = \mathbf{R} \cup \{x_i\}$
5: **end if**
6: **end for**
7: retorna $\mathbf{R}$

2.5.6 *Decremental Reduction Optimization Procedure 3*

Proposto por Wilson e Martinez [121], o *Decremental Reduction Optimization Procedure 3* (DROP3) faz parte de uma série de cinco algoritmos: DROP1, DROP2, DROP3, DROP4 e DROP5, criados para serem tolerantes a ruídos e com altas taxas de redução. O DROP3 é um método híbrido [115], isto é, opera tanto com condensação quanto com edição. O Algoritmo 8 mostra o funcionamento do método, no qual $viz(d)$ são todos os vizinhos de d e $ass(d)$ são todos os documentos que possuem d como vizinho. Inicialmente, o DROP3 cria uma lista de associados (Linhas 2-8) e aplica o filtro ENN em seguida (Linha 9), eliminando exemplos ruidosos e próximos às fronteiras de decisão. Após essa filtragem, é feita uma ordenação com base no inimigo mais próximo (Linha 10). Depois parte-se para a etapa de maior redução do algoritmo, um laço (Linhas 11-30) que irá remover documentos levando em consideração o desempenho dos associados. O exemplo é descartado se os associados conseguirem serem classificados corretamente sem esse exemplo (Linha 12-20). A lista de associados dos vizinhos mais próximos é atualizada (Linhas 23-28) sempre após uma remoção de exemplo (Linha 22).

Algoritmo 8 *Decremental Reduction Optimization Procedure 3* (DROP3)

Entrada: $v \geqslant 1$ e conjunto de treinamento $\mathbf{X}$

1: $\mathbf{R} = \mathbf{X}$
2: $ass(\mathbf{R}) = \emptyset$
3: **for all** $p_k \in \mathbf{R}$ **do**
4: $viz(p_k) = NN(p_k, \mathbf{R}, v+1)$
5: **for all** $d \in viz(p_k)$ **do**
6: $ass(d) = ass(d) \cup \{p_k\}$
7: **end for**
8: **end for**
9: $\mathbf{R} = ENN(v, \mathbf{X})$
10: ordena descendentemente $\mathbf{R}$ pela distância do inimigo mais próximo
11: **for all** $p_k \in \mathbf{R}$ **do**
12: $com = 0$ e $sem = 0$
13: **for all** $d \in ass(p_k)$ **do**
14: **if** $class(d) = class(viz(d))$ **then**
15: $com = com + 1$
16: **end if**
17: **if** $class(d) = class(viz(d) - \{p_k\})$ **then**
18: $sem = sem + 1$
19: **end if**
20: **end for**
21: **if** $sem \geqslant com$ **then**
22: $\mathbf{R} = \mathbf{R} - \{p_k\}$
23: **for all** $d \in ass(p_k)$ **do**
24: $viz(d) = viz(d) - \{p_k\}$
25: encontre o próximo vizinho de d (nv)
26: $viz(d) = viz(d) \cup \{nv\}$
27: $ass(nv) = ass(nv) \cup \{d\}$
28: **end for**
29: **end if**
30: **end for**
31: retorna $\mathbf{R}$

2.5.7 *Adaptive Threshold-based Instance Selection Algorithm 2*

Proposto por Cavalcanti et al. [122], o *Adaptive Threshold-based Instance Selection Algorithm 2* (ATISA2) é de uma série de três algoritmos: ATISA1, ATISA2 e ATISA3. O Algoritmo 9 mostra o funcionamento do método, no qual $NE(d, \mathbf{B}, v)$ retorna v documentos mais próximos do documento d no conjunto $\mathbf{B}$ que possuem classe diferente de d. Assim como alguns outros algoritmos, o ATISA2 inicia executando o ENN (Linha 1) e depois calcula a distância de cada documento desse conjunto para seu inimigo mais próximo, do inglês *nearest enemy*, nas Linhas 2-4. Ordena-se esse conjunto temporário ($\mathbf{X}_f$) de acordo com a distância para os inimigos mais próximos (Linha 5). Essa distância será o limiar que define se um documento será inserido ou não no conjunto $\mathbf{R}$, que inicia vazio (Linha 6). A ideia do algoritmo é gerar o conjunto $\mathbf{R}$ com

base no conjunto $\mathbf{X}_f$, isto é, $|\mathbf{R}| < |\mathbf{X}_f|$ (Linhas 7-12). Deste modo, os documentos de $\mathbf{X}_f$ são inseridos em $\mathbf{R}$ se sua classificação for incorreta com relação aos seus vizinhos ou a distância para seus vizinhos for maior que para seu inimigo mais próximo (Linhas 9-11).

Algoritmo 9 *Adaptive Threshold-based Instance Selection Algorithm 2* (ATISA2)

Entrada: $v \geqslant 1$ e conjunto de treinamento $\mathbf{X}$

1: $\mathbf{X}_f = ENN(v, \mathbf{X})$
2: **for all** $d \in \mathbf{X}_f$ **do**
3: $\quad \theta(d) = dist(d, NE(d, \mathbf{X}_f, 1))$
4: **end for**
5: ordena descendentemente $\mathbf{X}_f$ pela distância do inimigo mais próximo
6: $\mathbf{R} = \emptyset$
7: **for all** $d \in \mathbf{X}_f$ **do**
8: $\quad viz = NN(d, \mathbf{R}, v)$
9: $\quad$ **if** $class(d) \neq class(viz)$ ou $dist(d, viz) > \theta(d)$ **then**
10: $\quad\quad \mathbf{R} = \mathbf{X} \cup \{d\}$
11: $\quad$ **end if**
12: **end for**
13: retorna $\mathbf{R}$

2.6 Combinação de Classificadores

Nenhum classificador é capaz de resolver todos os problemas em qualquer base de dados [30]. Portanto, várias aplicações focam seus esforços em combinar múltiplos classificadores, resolvendo problemas complexos de modo mais robusto e eficiente [123]. A combinação de classificadores baseia-se na premissa de unir classificadores com características diferentes, se possível complementares, pois a união de diferentes classificadores traz as vantagens destes e dispersa suas fraquezas. Logo, espera-se que uma combinação melhore o desempenho do sistema [124].

Sendo que para um sistema de múltiplos classificadores obter os resultados esperados, isto é, os classificadores trabalhando em conjunto para melhorar o desempenho global, é necessário que exista diversidade na combinação [125]. O conceito de diversidade será discutido na Seção 2.6.1.

A Figura 2.7 mostra a arquitetura de um sistema de múltiplos classificadores. Um sistema de múltiplos classificadores inicia com a geração do *pool* de classificadores, isto é, alguma metodologia para criar classificadores diferentes para serem combinados; em seguida todo o *pool* de classificadores é treinado; alguns desses classificadores podem ser removidos, com base em métodos de poda, que utilizam variadas abordagens, seja com base na diversidade, performance ou algum outro pré-requisito, para realizar a poda utiliza-se um conjunto de validação; com os classificadores treinado é possível iniciar a fase de teste, isto é, apresentar exemplos ainda não classificados para verificar a performance do sistema como um todo; durante a fase de teste é possível existir uma seleção dinâmica do *ensemble*, para utilizar apenas classificadores

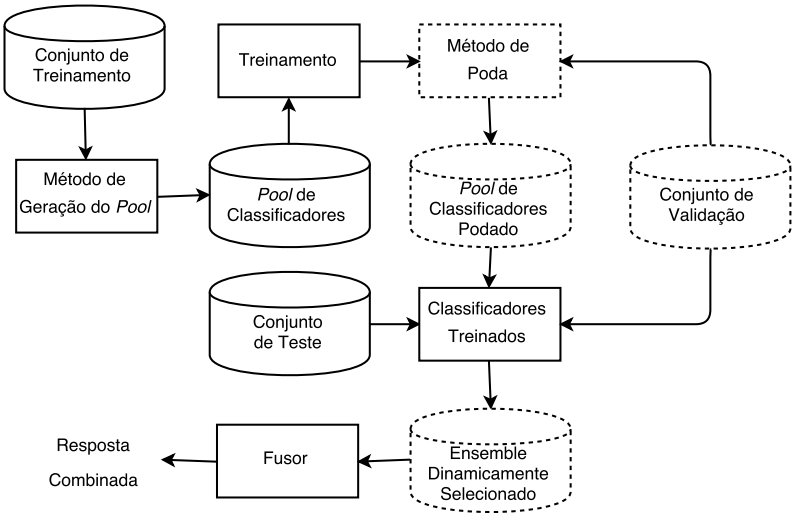


Figura 2.7: Arquitetura de um sistema de múltiplos classificadores.

especializados naquele exemplo que será classificado, para tal é utilizado um conjunto de validação; no final, o fusor que irá gerar uma única resposta a partir das respostas dadas por todos os classificadores. As etapas com caixa em linhas tracejadas indicam módulos opcionais, isto é, módulos que podem ser utilizados ou não dependendo dos interesses e necessidades da aplicação.

Na Seção 2.6.2 são apresentados métodos de geração do *pool* de classificadores e na Seção 2.6.3 são apresentadas maneiras de combinar as respostas dos vários classificadores (Fusor). Como esta proposta não utiliza poda nem seleção dinâmica, estes tópicos não serão abordados, mas uma revisão da literatura foi realizada nos trabalhos de Soto et al. [126] e Britto et al. [127], respectivamente.

2.6.1 Diversidade

Estudos empíricos mostram que existe uma relação positiva entre o desempenho do *ensemble* e a diversidade nos classificadores do *pool* [128]. Entretanto, não existe uma definição formal do que é diversidade [129]. Portanto, apesar da diversidade ser um dos fatores para o sucesso em uma Combinação de Classificadores, não é possível afirmar que seja um fator definitivo. Nesta seção, são revisadas algumas medidas de diversidade mais utilizadas, começando pelas medidas pareadas, seguindo para as não pareadas. É recomendado o trabalho de Tang et. al [130] para um estudo mais aprofundado sobre diversidade.

Os cálculos de todas essas medidas devem ser realizados sobre as respostas de um conjunto de validação ou as respostas de um conjunto de teste. A nomenclatura utilizada considera que as medidas estão sendo calculadas sobre as respostas do conjunto de teste.

As medidas de diversidade pareadas calculam a diversidade entre dois classificadores. A Tabela 2.3 mostra as variáveis essenciais para os cálculos das medidas de diversidade pareadas a seguir:

Tabela 2.3: Tabela de Relação entre classificadores α e β. Os valores a, b, c e d são contadores baseados no conjunto de validação ou teste.

	Classificador α acerta	Classificador α erra
Classificador β acerta	a	b
Classificador β erra	c	d

- ***Q Statistics*** [131]

$$\mathrm{Q}_{\alpha,\beta} = \frac{ad - bc}{ad + bc} \tag{2.27}$$

$\mathrm{Q}_{\alpha,\beta}$ varia entre -1 e $+1$. Quando os classificadores reconhecem os mesmos padrões, tende à $+1$, quando os classificadores cometem erros diferentes tende à -1. Em classificadores estatisticamente independentes $\mathrm{Q}_{\alpha,\beta} = 0$.

- ***Correlation Coefficient*** ρ [132]

$$\rho_{\alpha,\beta} = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}}. \tag{2.28}$$

Similar à *Q Statistics*. Ambas terão o mesmo sinal, mas $\mathrm{Q}_{\alpha,\beta} \geqslant \rho_{\alpha,\beta}$.

- ***Disagreement Measure*** [133]

$$\mathrm{Dis}_{\alpha,\beta} = \frac{b+c}{a+b+c+d}. \tag{2.29}$$

Representa a probabilidade dos classificadores α e β discordarem da decisão, seja esta correta ou incorreta. Existe também a medida inversa chamada de *Agreement Measure*.

- ***Double-Fault Measure*** [134]

$$\mathrm{DF}_{\alpha,\beta} = \frac{d}{a+b+c+d}. \tag{2.30}$$

Representa a probabilidade dos classificadores α e β errarem juntos.

Enquanto que as medidas de diversidade pareadas calculam a diversidade apenas entre dois classificadores, as medidas não-pareadas calculam a diversidade de todo o *pool* de classificadores. É possível, entretanto, utilizar as medidas pareadas para calcular a diversidade de todo o *pool* pela seguinte equação:

$$\mathrm{Medida}_{\mathrm{av}} = \frac{2}{L(L-1)} \sum_{a}^{L-1} \sum_{b}^{L} \mathrm{Medida}_{\mathrm{a,b}}, \tag{2.31}$$

no qual L é a quantidade de classificadores do *pool* e $\text{Medida}_{\text{a,b}}$ pode ser qualquer medida de diversidade pareada.

Algumas medidas de diversidade não pareadas utilizam a função binária $or(s, j)$, que indica se o classificador s acertou (1) ou errou (0) a classe do documento j; e também necessitam da quantidade de documentos do conjunto de teste (M). Algumas medidas de diversidade são apresentadas a seguir:

- ***Entropy Measure E*** [135]

$$E = \frac{1}{M}\frac{2}{L-1}\sum_{j}^{M}\min\left\{\left(\sum_{s}^{L} or(s,j)\right), \left(L - \sum_{s}^{L} or(s,j)\right)\right\}, \tag{2.32}$$

  varia entre 0 (nenhuma diferença) e $+1$ (maior diversidade possível).

  Intuitivamente, é possível afirmar que um *ensemble* é diverso quando os $L/2$ classificadores acertam e os outros $L/2$ erram. Afinal, se todos acertarem ou errarem tudo, não há desacordo. Logo, não é efetivo combiná-los.

- ***Kohavi-Wolpert Variance*** [136]

$$KW = \frac{1}{NL^2}\sum_{j}^{M}\left(\sum_{s}^{L} or(s,j)\right)\left(L - \sum_{s}^{L} or(s,j)\right), \tag{2.33}$$

  varia entre 0 (nenhuma diferença) e 0.25 (maior diversidade possível).

  Ideia parecida com a *Entropy Measure*, isto é, a máxima diversidade é atingida quando existe uma divisão equalitária de acertos e erros entre os classificadores.

- ***Measure of Interrater Agreement*** [137]

$$\kappa = 1 - \frac{\frac{1}{L}\sum_{j}^{M}\left(\sum_{s}^{L} or(s,j)\right)\left(L - \sum_{s}^{L} or(s,j)\right)}{N(L-1)\bar{p}(1-\bar{p})}, \tag{2.34}$$

  no qual $\bar{p}$ é a taxa de acerto média dos classificadores do *pool*.

  κ varia entre $+1$ (nenhuma diferença) e -1 (maior diversidade possível) e possui uma relação com outras medidas de diversidade:

$$\kappa = 1 - \frac{L}{(L-1)\bar{p}(1-\bar{p})}KW = 1 - \frac{1}{2\bar{p}(1-\bar{p})}\text{Dis}_{\text{av}}, \tag{2.35}$$

  no qual Dis_{av} é a *Disagreement Measure* transformada em medida de diversidade não pareada pela Equação (2.31).

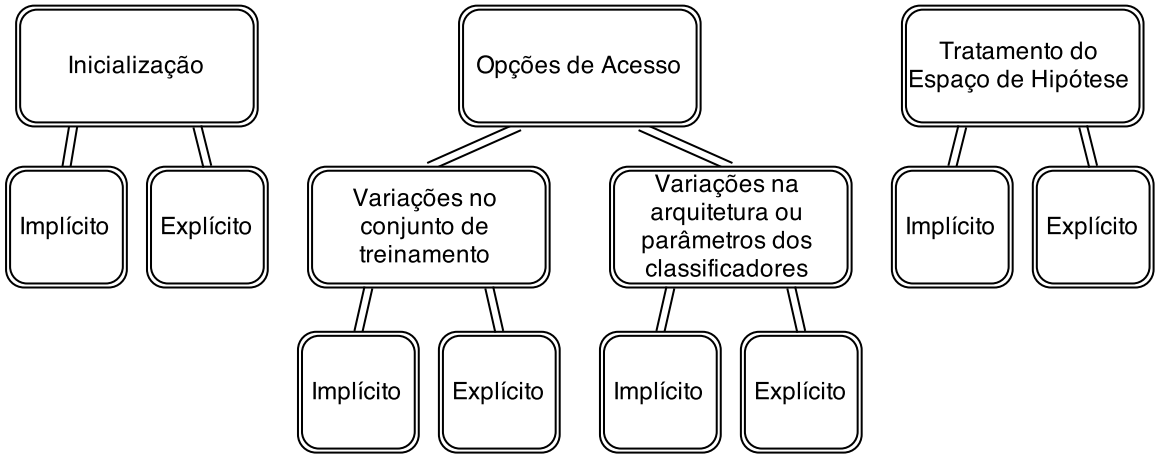


Figura 2.8: Taxonomia de Brown et. al [1].

2.6.2 Geração do *Pool*

A geração do *pool* é a etapa mais importante de um sistema de múltiplos classificadores, pois se o *pool* iniciou de modo insatisfatório – por exemplo, composto por classificadores similares – não existe nenhuma poda, seleção dinâmica ou fusor que seja capaz de melhorar o seu desempenho. Apesar de não existir um consenso formal da definição de diversidade [130], é notável a necessidade dos classificadores serem complementares para a combinação ser bem sucedida [1]. Por isso, os métodos de geração de *pool* de classificadores sempre utilizam alguma metodologia visando, por fim, um *pool* diverso.

A Figura 2.8 mostra a taxonomia proposta por Brown [1] para expor os diversos métodos de geração de *pool* de classificadores. A taxonomia abrange duas distinções. A primeira distinção é se o método é explícito ou implícito em sua busca pela diversidade. Um método explícito utiliza alguma metodologia determinística, talvez utilizando medidas de diversidade ou outros conceitos para atingir a diversidade. Um método implícito usa aleatoriedade na tentativa de gerar diversidade, espera-se que a diversidade exista devido à aleatoriedade. Dado que a definição de diversidade ainda não é bem estabelecida, as abordagens mais bem sucedidas são implícitas [138]. A segunda distinção é dividida em três tipos: i) inicialização, variação do ponto inicial do espaço de hipótese influenciando na convergência; ii) opções de acesso, variação dos espaços de hipóteses disponíveis, podendo existir variações no conjunto de treinamento ou na arquitetura do classificador; iii) tratamento do espaço de hipótese, estes alteram o maneira que o espaço é tratado levando diferentes classificadores para uma convergência diferente do *ensemble*.

É importante destacar o *Bagging*, o *Random Subspace* e o *Random Forest*, pois os três são utilizados nos experimentos. *Bagging* é uma sigla extraída de *Bootstrap AGGregatING* [139]. É um método de geração de *pool* de classificadores implícito que varia suas opções de acesso de modo que cada classificador trabalhe com um conjunto de treinamento distinto. Esses conjuntos de treinamento são subconjuntos de mesmo tamanho do conjunto de treinamento original, sendo

obtido por uma amostragem aleatória com reposição. Deste modo, alguns documentos aparecem múltiplas vezes para um determinado classificador, enquanto não aparecem nenhuma vez para outro. *Bagging* é um método clássico bastante utilizado na literatura, principalmente para avaliar o desempenho em comparação com métodos propostos [140, 141, 142].

Random Subspace [143] é outro método de geração de *pool* de classificadores implícito que varia suas opções de acesso, de modo que cada classificador trabalhe em um espaço de características diferente. Isto é, o conjunto de treinamento de cada classificador possui os mesmos documentos, mas cada um deles utilizando apenas uma parte das características selecionadas aleatoriamente. Para tanto, é definido um parâmetro de quantas características deverão ser selecionadas. Quanto menor a quantidade de características, pior será a performance de cada classificador individual mas a variedade entre os classificadores será maior. O *Random Subspace* é normalmente utilizado para bases de dados com muitas características [144] e serviu de inspiração para métodos voltados especificamente para lidar com alta dimensionalidade [145, 146, 147].

Random Forest [148] é um método de combinação de classificadores bastante robusto [149], capaz de lidar com diferentes bases de dados de diferentes domínios [150, 151, 152], incluindo Classificação de Documentos [35]. Seu nome é proveniente do uso das árvores de decisão como classificadores base, portanto floresta seria uma combinação de árvores. Esses classificadores são treinados usando um procedimento similar a uma união do *Bagging* com *Random Subspace*, isto é, os dados utilizados para treinamento de cada classificador do *pool* varia tanto nos padrões quanto nas características. Deste modo, o *Random Forest* inicia particionando o conjunto de treinamento utilizando uma amostragem aleatória com reposição. Em seguida, cada novo conjunto será utilizado para construir uma árvore de decisão. Durante o treinamento de cada árvore de decisão, um espaço de características menor que o original é selecionado aleatoriamente. Por fim, voto majoritário é utilizado para combinar as saídas dos classificadores.

2.6.3 Fusor

Após a geração do *pool* de classificadores e do treinamento dos classificadores, o conjunto de teste é apresentado ao *ensemble*. A resposta será baseada na combinação (fusão) dos resultados de todos os classificadores ou parte deles, caso seleção dinâmica seja utilizada.

A maneira mais comum de combinar as respostas dos classificadores é utilizando voto majoritário [153]. Esta combinação pode funcionar de três modos [154]:

i) A classe será aquela que todos os classificadores votaram nela;

ii) A classe será aquela que conseguir uma quantidade de votos maior que a metade dos classificadores do pool;

iii) A classe será aquela que conseguir a maior quantidade de votos em comparação com as outras.

Caso a condição não seja atingida, ou existir empate, o documento pode ser considerado de uma classe indefinida ou escolher uma classe aleatoriamente (dentre as melhores). Neste trabalho é utilizada a terceira opção, pois tanto a primeira como a segunda opção são casos mais difíceis de ocorrer e apenas restringem a combinação.

Outra maneira, de combinar as respostas dos classificadores, é utilizar um classificador que dará como resposta a classe final de um determinado documento. Este classificador deve ser treinado, após o treinamento dos classificadores do *ensemble*, usando como entrada as respostas do *pool* de classificadores dos documentos de um conjunto de validação.

2.7 Considerações Finais

Neste capítulo foram apresentadas as etapas de um sistema de Classificação de Documentos: pré-processamento, que utiliza diversas rotinas para transformar os documentos originais em documentos mais simplificados, visando facilitar o trabalho das etapas subsequentes; geração do vetor de características, que transforma os textos em linguagem natural para um vetor de representação possível de ser utilizado em classificadores; e a construção do classificador, responsável pela classificação dos novos documentos. Adicionalmente, foram apresentados métodos de seleção de características que são utilizados em Classificação de Documentos. Por fim, este capítulo apresentou conceitos e métodos relevantes para a compreensão das abordagens propostas, como: *Dissimilarity Representation*, *Dichotomy Transformation*, Seleção de Protótipos e Combinação de Classificadores.

3

Combined Dissimilarity Spaces (CoDiS)

Neste capítulo é apresentada a primeira arquitetura proposta: CoDiS, do inglês *Combined Dissimilarity Spaces*. Essencialmente, trata-se de um sistema de Classificação de Documentos (Seção 2.1) que combina classificadores treinados em diferentes espaços de dissimilaridades obtidos pela Representação por Dissimilaridade (DR, apresentad na Seção 2.3). As seções seguintes descrevem a notação da arquitetura proposta (Seção 3.1); suas duas etapas, treinamento e teste em linhas gerais (Seção 3.2); detalhamento de alguns módulos da arquitetura (Seção 3.3 e Seção 3.3); um exemplo *toy*, mostrando como a arquitetura proposta funciona com dados hipotéticos e gráficos (Seção 3.4.1); e por fim, considerações finais sobre a CoDiS (Seção 3.5).

3.1 Notação

CoDiS trabalha com diversos conjuntos diferentes durante sua execução, sejam conjuntos de entrada, de saída, ou gerados durante a fase de treinamento. Segue notações desses conjuntos:

- O conjunto de treinamento $\mathbf{X}$ é composto de N documentos $x_i \in \mathbf{X} = \{x_1, x_2, \ldots, x_N\}$, cada um sendo representado pelo par $\langle w^{x_i}, class(x_i)\rangle$, no qual $w^{x_i} = [w_1^{x_i}, w_2^{x_i}, \ldots, w_V^{x_i}]^{\mathrm{T}}$ é um vetor de termos obtido com *Bag-of-Words* e $class(x_i) \in \mathbf{C} = \{c_1, c_2, \ldots, c_C\}$ é a classe de um determinado documento x_i.
- O conjunto de teste $\mathbf{Y}$ é composto de M documentos $y_j \in \mathbf{Y} = \{y_1, y_2, \ldots, y_M\}$ e usa a mesma representação *Bag-of-Words* do conjunto de treinamento.
- Os conjuntos $\mathbf{X}_s$ são subconjuntos do conjunto de treinamento $\mathbf{X}$, cada subconjunto de treinamento $\mathbf{X}_s$ é composto de N' documentos ($N' \leq N$).
- Os conjuntos de representação $\mathbf{R}_s$ são compostos de Q documentos $p_k \in \mathbf{R}_s = \{p_1, p_2, \ldots, p_Q\}$.

3.2 Treinamento e Teste

A Figura 3.1 mostra um fluxograma da fase de treinamento. Nesta fase, CoDiS recebe como entrada: o conjunto de treinamento $\mathbf{X}$, o tamanho do *ensemble* (L) e o tamanho

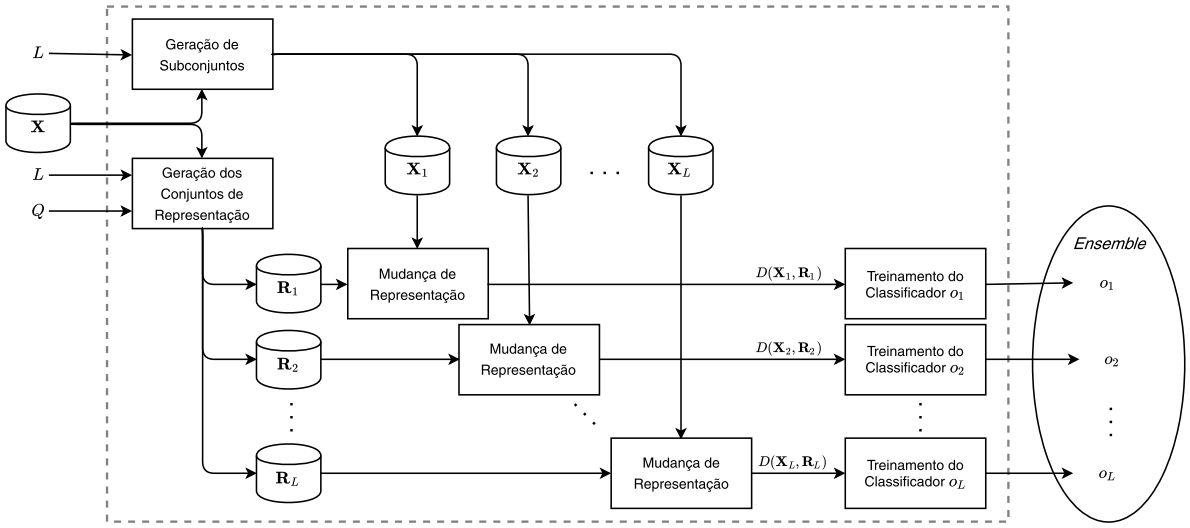


Figura 3.1: Fase de treinamento do CoDiS. $\mathbf{X}$ é o conjunto de treinamento, $\mathbf{X}_s$ são os subconjuntos do conjunto de treinamento, $\mathbf{R}_s$ são os conjuntos de representação, L é a quantidade de classificadores no *pool*, Q é a quantidade de documentos do conjunto de representação, $D(\mathbf{X}_s, \mathbf{R}_s)$ são as matrizes de dissimilaridade obtidas usando $\mathbf{X}_s$ e $\mathbf{R}_s$, e o_s são os classificadores do *pool*. Note que $s = \{1, 2, \ldots, L\}$.

dos conjuntos de representação (Q); e, como saída, L classificadores. Quatro módulos distintos são utilizados durante essa fase: Geração dos Subconjuntos, Geração dos Conjuntos de Representação, Mudança de Representação e Treinamento do Classificador.

O módulo Geração de Subconjuntos recebe como entrada o conjunto de treinamento $\mathbf{X}$. Este módulo trabalha com *Bootstrapping*, uma amostragem com reposição, tal como no *Bootstrap AGGregatING* [139]. Todos os subconjuntos terão o mesmo tamanho do conjunto de treinamento original (N) com alguns documentos aparecendo múltiplas vezes e outros nenhuma vez. As saídas deste módulo serão L subconjuntos de treinamento ($\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_L}$). A utilização de subconjuntos diferentes, em vez da replicação do conjunto de treinamento, serve para aumentar a diversidade obtida no *pool* de classificadores.

Em paralelo ao módulo Geração de Subconjuntos, existe o módulo Geração dos Conjuntos de Representação que recebe como entrada o conjunto de treinamento $\mathbf{X}$ e o tamanho do *pool* de classificadores (L). A saída deste módulo serão L subconjuntos do conjunto de treinamento ($\mathbf{R_1}, \mathbf{R_2}, \ldots, \mathbf{R_L}$) cada um com Q documentos. É desejável que $Q \ll N$, pois esse conjunto irá definir a quantidade de características na mudança de representação. Apesar da similaridade com o módulo Geração de Subconjuntos, este módulo é mais complexo e possui algumas variações. Detalhamento na Seção 3.3.

Até o momento, foram gerados $2 \times L$ subconjuntos do conjunto de treinamento. Na primeira metade ($\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L$) cada subconjunto contém N documentos; e na segunda metade ($\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L$) cada subconjunto contém Q documentos. Então, o módulo Mudança de Representação utiliza esses subconjuntos para gerar L matrizes de dados de tamanho $N \times Q$, representando os N documentos com Q características. Detalhamento na Seção 3.4.

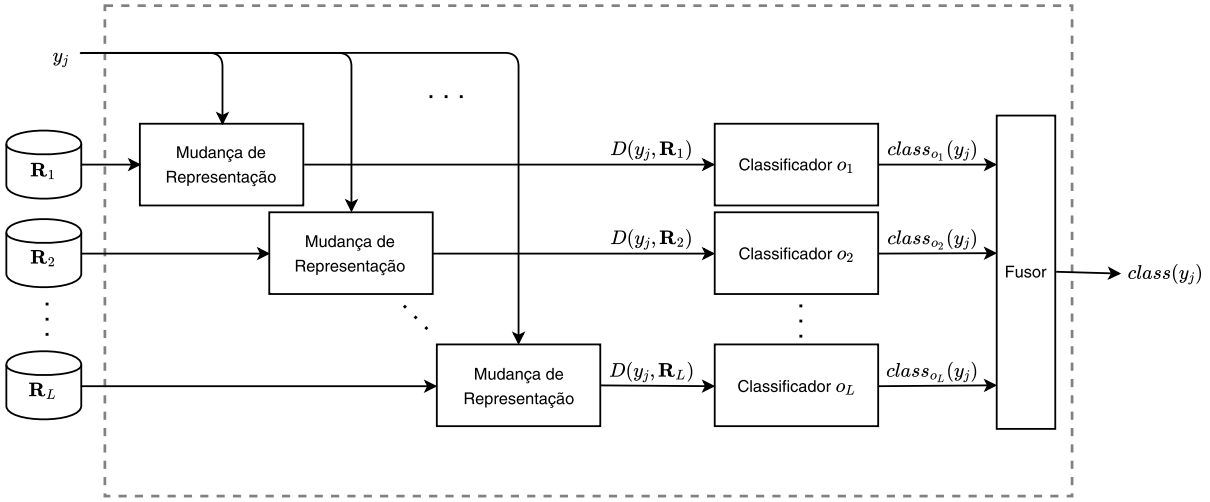


Figura 3.2: Fase de teste do CoDiS. $y_j \in \mathbf{Y}$ é um documento do conjunto de teste, $\mathbf{R}_s$ são os conjuntos de representação, L é o número de classificadores no *ensemble*, $D(y_j, \mathbf{R}_s)$ é o vetor de dissimilaridade obtido usando y_j e $\mathbf{R}_s$, $class_{o_s}(y_j)$ são as respostas de cada classificador o_s e $class(y_j)$ é a resposta final. Note que $s = \{1, 2, \ldots, L\}$.

Finalmente, cada uma das L matrizes é apresentada a um módulo Treinamento do Classificador distinto. Deste modo, a saída final da fase de treinamento que são os L classificadores treinados $(o_1, o_2, \ldots, o_L)$.

Após a conclusão da fase de treinamento, inicia-se a fase de teste. A Figura 3.2 mostra a fase de teste da arquitetura proposta. Nesta fase temos, como entrada, um documento do conjunto de teste $y_j \in \mathbf{Y}$, L conjuntos de representação $(\mathbf{R_1}, \mathbf{R_2}, \ldots, \mathbf{R_L})$ e L classificadores; e, como saída, uma resposta $class(y_j)$. São utilizados três módulos distintos nessa fase: Mudança de Representação, Classificador e Fusor.

O módulo Mudança de Representação deve ser o mesmo utilizado na fase de treinamento. Entretanto, os dados de entrada do módulo são: um documento do conjunto de teste $\mathbf{Y}$ e os conjuntos de representação obtidos na fase de treinamento. Deste modo, cada classificador (Seção 2.1.3) recebe um vetor de tamanho Q. As respostas dos L classificadores são combinadas pelo fusor (Seção 2.6.3) que dará a resposta $class(y_j)$.

3.3 Geração dos Conjuntos de Representação

O módulo Geração dos Conjuntos de Representação (Figura 3.1) recebe o conjunto de treinamento $\mathbf{X}$ e gera L subconjuntos $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L$, chamados de conjuntos de representação. O conjunto de representação é um conjunto formado por documentos de referência utilizados para a mudança de representação (Seção 3.4).

Diversas abordagens podem ser utilizadas para gerar um conjunto de representação: utilizar todos os documentos do conjunto de treinamento, selecionar alguns documentos aleatoriamente ou selecionar os documentos mais dissimilares [23]. Cada uma dessas metodologias possui vantagens e desvantagens. Neste trabalho foram utilizadas duas abordagens: seleção

aleatória, pela simplicidade e capacidade de atingir diversidade na combinação de classificadores; e seleção de protótipos, pela possibilidade de obter conjuntos de representação compostos por exemplos capazes de descrever bem todos o conjunto de treinamento original.

A abordagem com seleção aleatória funciona similar ao *Bootstrapping*. Entretanto, a amostragem é sem reposição, isto é, nenhum documento é selecionado duas vezes, e a quantidade de documentos selecionados é menor que o número de documentos no conjunto de treinamento original ($Q < N$). Apesar da imprevisibilidade dessa abordagem, é possível atingir bons resultados mesmo sem Combinação de Classificadores [28].

A abordagem com seleção de protótipos utiliza algum algoritmo para selecionar os documentos. O algoritmo de seleção de protótipos define um procedimento determinístico, normalmente baseado nos vizinhos mais próximos, para que selecione apenas documentos relevantes. A quantidade de documentos do conjunto de representação (Q) vai depender do algoritmo escolhido, pois Q é definido em tempo de execução. Portanto, com essa abordagem, não é necessário definir *Q a priori*, mas é necessário definir a quantidade de vizinhos levados em consideração (v). Espera-se que esses algoritmos reduzam e melhorem a representação ao remover documentos irrelevantes ou redundantes [115]. Detalhamento de alguns algoritmos de seleção de protótipos na Seção 2.5.

3.4 Mudança de Representação

O módulo Mudança de Representação recebe dois conjuntos de dados no mesmo espaço de representação e gera uma matriz de dados que é a nova representação dos dados de entrada. Na arquitetura proposta, esse módulo está presente na fase de treinamento (Figura 3.1) recebendo $\mathbf{X}_s$ e $\mathbf{R}_s$ ($s = 1,2,\ldots,L$) que geram matrizes de tamanho $N \times Q$, e na fase de teste (Figura 3.2) recebendo y_j e $\mathbf{R}_s$ ($s = 1,2,\ldots,L$) que gera vetores de tamanho $1 \times Q$. Essas matrizes e vetores são baseados na Representação por Dissimilaridade proposta por Pekalska e Duin [23], mais especificamente a abordagem de espaço de dissimilaridade, apresentada na Seção 2.3.

Normalmente, a Representação por Dissimilaridade utiliza distância Euclidiana que é definida por:

$$d(x_a,x_b) = \sqrt{\sum_{h=1}^{V}(w_{a,h}-w_{b,h})^2}, \tag{3.1}$$

no qual x_a e x_b são dois documentos quaisquer, $w_{a,h}$ e $w_{b,h}$ são as h-ésimas características dos documentos x_a e x_b respectivamente, e V é a quantidade de características.

Entretanto, o espaço de dissimilaridade pode utilizar qualquer métrica de relação, seja distância ou similaridade. Uma possibilidade é a similaridade do cosseno, definida por:

$$sim(x_a,x_b) = \frac{\sum_{h=1}^{V} w_{a,h}w_{b,h}}{\sqrt{\sum_{h=1}^{V} w_{a,h}^2}\sqrt{\sum_{h=1}^{V} w_{b,h}^2}}. \tag{3.2}$$

A similaridade do cosseno possui resultados bem estabelecidos em problemas de Classificação de Documentos [10, 155], portanto é interessante verificar seu desempenho na arquitetura proposta. Tendo em vista que a matriz de dados é esparsa e de alta dimensionalidade, a distância Euclidiana entre documentos será elevada mesmo para documentos similares da mesma classe, enquanto que a similaridade do cosseno verifica apenas as características com valor diferente de zero, evitando o problema dos dados esparsos.

Considere um exemplo com os documentos de treinamento $x_1 = \{2, 2, 1, 4, 0, 0, 0, 0, 0, 0\}$ da classe A e $x_2 = \{0, 0, 0, 0, 0, 0, 0, 3, 2, 0\}$ da classe B e um documento de teste $y_1 = \{2, 0, 1, 0, 0, 0, 0, 0, 0, 0\}$, que será classificado na fase de teste, da classe A. Supondo um classificador simples, como o 1NN (*Nearest Neighbor*), no qual a classe do documento de teste será igual a classe do documento de treino mais próximo (menor distância) ou mais similar (maior similaridade). É calculada a distância para os dois documentos de treinamento, obtendo $d(y_1,x_1) = 4,47$ e $d(y_1,x_2) = 4,24$. A classificação seria incorreta, pois $d(y_1,x_2) < d(y_1,x_1)$, logo y_1 é classificado com a mesma classe de x_2 (B) em vez da classe A. Enquanto que com a similaridade do cosseno os resultados são $sim(y_1,x_1) = 0,44$ e $sim(y_1,x_2) = 0$. A classificação seria correta, pois $sim(y_1,x_1) > sim(y_1,x_2)$, logo y_1 é classificado como pertencente a classe A.

Deste modo, na fase de treinamento (Figura 3.1), o conjunto de treinamento $\mathbf{X}$ e um conjunto de representação $\mathbf{R}$ são utilizados para gerar a nova representação com a similaridade do cosseno:

$$S(\mathbf{X},\mathbf{R}) = \begin{bmatrix} sim(x_1,p_1) & sim(x_1,p_2) & \dots & sim(x_1,p_Q) \\ sim(x_2,p_1) & sim(x_2,p_2) & \dots & sim(x_2,p_Q) \\ \vdots & \vdots & \ddots & \vdots \\ sim(x_N,p_1) & sim(x_N,p_2) & \dots & sim(x_N,p_Q) \end{bmatrix} \tag{3.3}$$

de dimensões $N \times Q$, no qual cada linha representa um documento e cada coluna representa uma característica.

Assim como na fase de teste (Figura 3.2), um documento do conjunto de teste $y_j \in \mathbf{Y}$ e um conjunto de representação $\mathbf{R}$ geram um vetor de similaridade: $S(y_j,\mathbf{R}) = [sim(y_j,p_1), sim(y_j,p_2), \dots, sim(y_j,p_Q)]^{\mathrm{T}}$. Por fim, $y_j \in \mathbf{Y}$ será classificado para uma classe $class(y_j)$.

3.4.1 Exemplo *toy*

Nesta seção, é apresentado um exemplo hipotético (Tabela 3.1) para mostrar o funcionamento do CoDiS. A Tabela 3.1 mostra o subconjunto de treinamento $\mathbf{X}_1$ com 10 documentos ($N' = 10$), o conjunto de representação $\mathbf{R}_1$ com 4 documentos ($Q = 4$), e um exemplo de teste $y_j \in \mathbf{Y}$. Todos os documentos são representados por 9 características e divididos em duas classes ($\mathbf{C} = \{A,B\}$).

O conjunto de treinamento $\mathbf{X}_1$ é gerado usando *Bagging*. Deste modo, alguns documentos do conjunto original não estão presentes, enquanto que outros aparecem mais do que uma vez,

por exemplo, $x_5 = x_6$. O conjunto de representação $\mathbf{R}_1$ também é obtido aleatoriamente com base no conjunto de treinamento original, assim é possível compartilhar alguns dos documentos presentes no subconjunto de representação, por exemplo, $p_1 = x_2$, $p_2 = x_4$ e $p_4 = x_9$.

O próximo passo é calcular a matriz de dissimilaridade $D(\mathbf{X}_1, \mathbf{R}_1)$, com base na Equação (3.1), mostrado na Tabela 3.2.

CoDiS é uma combinação de classificadores, sendo assim suponha que são utilizados $L = 5$ classificadores $\{o_1, o_2, o_3, o_4, o_5\}$. Cada classificador o_s é treinado usando sua respectiva matriz de dissimilaridade, obtida pela pelo conjunto de representação $\mathbf{R}_s$. Então, o classificadoro_1 é treinado com os dez documentos obtidos com a *Dissimilarity Representation* sendo cada um representado por quatro características como exibido na Tabela 3.2, no qual cada característica é a distância entre o documento no subconjunto de treinamento $\mathbf{X}_1$ e um documento do conjunto de representação $\mathbf{R}_1$.

Após o treinamento, o examplo de teste $y_j \in \mathbf{Y}$ (Tabela 3.1) é apresentado aos 5 classificadores. A entrada de cada classificador o_s é $D(y_j, \mathbf{R}_s)$, no qual $s = \{1,2,3,4,5\}$; por exemplo instance, a resposta de y_j com o_1 é baseado no vetor obtido por $D(y_j, \mathbf{R}_1) = [8.185353, 5.916080, 4.123106, 6.782330]^{\mathrm{T}}$. Suponha que o classificador o_1 atribui a classe A para y_j, isto está incorreto. Entretanto, a resposta final do CoDiS é dada pela combinação de todos os classificadores. Assim, assuma que o_2, o_4, e o_5 respondem classe B e o_3 responde classe A. A resposta final para y_j é para a classe B, pois foi a classe que obteve a maioria dos votos (3 contra 2).

É importante notar que a matriz de dados original ($\mathbf{X}_1$ in Table 3.1) possui 54 zeros ($54/90 = 60\%$), enquanto que a matriz de dissimilaridade (Table 3.2) possui apenas 3 zeros ($3/40 = 7.5\%$). Isso é uma grande redução na esparsidade, no qual não pode ser alcançada com seleção de características. A menor esparsidade obtida por uma seleção de características com 4 características é 50% (selecionando w_4, w_7, w_1, e w_5). Portanto, CoDiS trabalha com um número reduzido de características e de esparsidade.

3.5 Considerações Finais

Neste capítulo foi apresentada a primeira arquitetura proposta: CoDiS, do inglês *Combined Dissimilarity Spaces*. Trata-se de um sistema de Classificação de Documentos que utiliza Representação por Dissimilaridade em conjunto com Combinação de Classificadores, para lidar com as dificuldades encontradas na representação usual BoW. A arquitetura é modular, possibilitando a substituição dos métodos utilizados em cada módulo, seja com relação à geração dos subconjuntos de treinamento, geração dos conjuntos de representação, classificador ou fusor. Além da variação de parâmetros, como L, Q e métrica da mudança de representação. No Capítulo 5 serão apresentados os experimentos com as comparações do desempenho de algumas das configurações da arquitetura proposta com relação a outros métodos da literatura.

Tabela 3.1: Exemplo *toy* do CoDiS. Primeira coluna identifica o conjunto do documento na segunda coluna, a última coluna ($class(\cdot)$) mostra a classe de cada documento, e as colunas no meio ($w_1, w_2, \ldots, w_9$) mostram as características. Cada linha representa um documento.

Conjunto	Documentos	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	$class(\cdot)$
$\mathbf{X}_1$	x_1	2	1	5	0	0	0	1	5	0	A
	x_2	3	0	0	3	2	0	0	6	4	A
	x_3	0	6	0	0	0	0	0	0	0	A
	x_4	0	2	0	0	0	0	1	1	0	A
	x_5	0	0	0	3	0	3	0	0	6	B
	x_6	0	0	0	3	0	3	0	0	6	B
	x_7	1	0	1	4	2	0	0	0	0	B
	x_8	0	0	0	2	0	2	2	0	7	B
	x_9	5	0	1	3	1	0	5	0	0	B
	x_{10}	0	0	0	4	2	0	0	0	3	B
$\mathbf{R}_1$	p_1	3	0	0	3	2	0	0	6	4	A
	p_2	0	2	0	0	0	0	1	1	0	A
	p_3	0	0	0	6	0	6	0	0	0	B
	p_4	5	0	1	3	1	0	5	0	0	B
$\mathbf{Y}$	y_j	2	0	0	4	0	3	0	0	0	B

Tabela 3.2: Matriz de dissimilaridade $D(\mathbf{X}_1, \mathbf{R}_1)$ e a classe de cada documento ($class(x_i)$).

$D(\mathbf{X}_1, \mathbf{R}_1)$	p_1	p_2	p_3	p_4	$class(x_i)$
$D(x_1, \mathbf{R}_1)$	7.615773	6.782330	11.313708	8.7749644	A
$D(x_2, \mathbf{R}_1)$	0.000000	8.246211	10.488088	9.110434	A
$D(x_3, \mathbf{R}_1)$	10.488088	4.242641	10.392305	9.848858	A
$D(x_4, \mathbf{R}_1)$	8.246211	0.000000	8.831761	7.549834	A
$D(x_5, \mathbf{R}_1)$	7.874008	7.745967	7.348469	9.848858	B
$D(x_6, \mathbf{R}_1)$	7.874008	7.745967	7.348469	9.848858	B
$D(x_7, \mathbf{R}_1)$	7.615773	5.291503	6.782330	6.557439	B
$D(x_8, \mathbf{R}_1)$	8.185353	7.937254	9.219544	9.486833	B
$D(x_9, \mathbf{R}_1)$	9.110434	7.549834	9.848858	0.000000	B
$D(x_{10}, \mathbf{R}_1)$	6.855655	5.916080	7.280110	7.874008	B

4

Combined Dichotomy Transformations (CoDiT)

Neste capítulo é apresentada a segunda arquitetura proposta: CoDiT, do inglês *Combined Dichotomy Transformations*. Essencialmente, trata-se de um sistema de Classificação de Documentos (Seção 2.1) que combina classificadores binários em diferentes espaços transformados pela *Dichotomy Transformation* (DT, apresentada na Seção 2.4). As seções seguintes descrevem a notação da arquitetura proposta (Seção 4.1); suas duas fases principais, treinamento (Seção 4.2), no qual L classificadores o_s são treinados; e teste (Seção 4.3), no qual o sistema é avaliado utilizando o conjunto de teste $\mathbf{Y}$; e por fim, um exemplo *toy*, mostrando como a arquitetura proposta funciona com dados hipotéticos e gráficos (Seção 4.4).

4.1 Notação

CoDiT trabalha com diversos conjuntos diferentes durante sua execução, sejam conjuntos de entrada, de saída, ou gerados durante a fase de treinamento. Segue notações desses conjuntos:

- O conjunto de treinamento $\mathbf{X}$ é composto de N documentos $x_i \in \mathbf{X} = \{x_1, x_2, \ldots, x_N\}$, cada um sendo representado pelo par $\langle w^{x_i}, class(x_i)\rangle$, no qual $w^{x_i} = [w_1^{x_i}, w_2^{x_i}, \ldots, w_V^{x_i}]^{\mathrm{T}}$ é um vetor de termos obtido com *Bag-of-Words* e $class(x_i) \in \mathbf{C} = \{c_1, c_2, \ldots, c_C\}$ é a classe de um determinado documento x_i.

- O conjunto de teste $\mathbf{Y}$ é composto de M documentos $y_j \in \mathbf{Y} = \{y_1, y_2, \ldots, y_M\}$ e usa a mesma representação *Bag-of-Words* do conjunto de treinamento.

- Os conjuntos $\mathbf{X}_s$ são subconjuntos do conjunto de treinamento $\mathbf{X}$, cada subconjunto de treinamento $\mathbf{X}_s$ é composto de N' documentos ($N' \leq N$).

- Os conjuntos de representação $\mathbf{R}_s$ são compostos de Q documentos $p_k \in \mathbf{R}_s = \{p_1, p_2, \ldots, p_Q\}$.

- Os conjuntos $\mathbf{Z}_s$ são compostos de $N' \times Q$ documentos gerados pela DT usando $\mathbf{X}_s$ e $\mathbf{R}_s$; e os conjuntos $\mathbf{Z}'_s$ são compostos de Q documentos gerados pela DT usando y_j e $\mathbf{R}_s$.

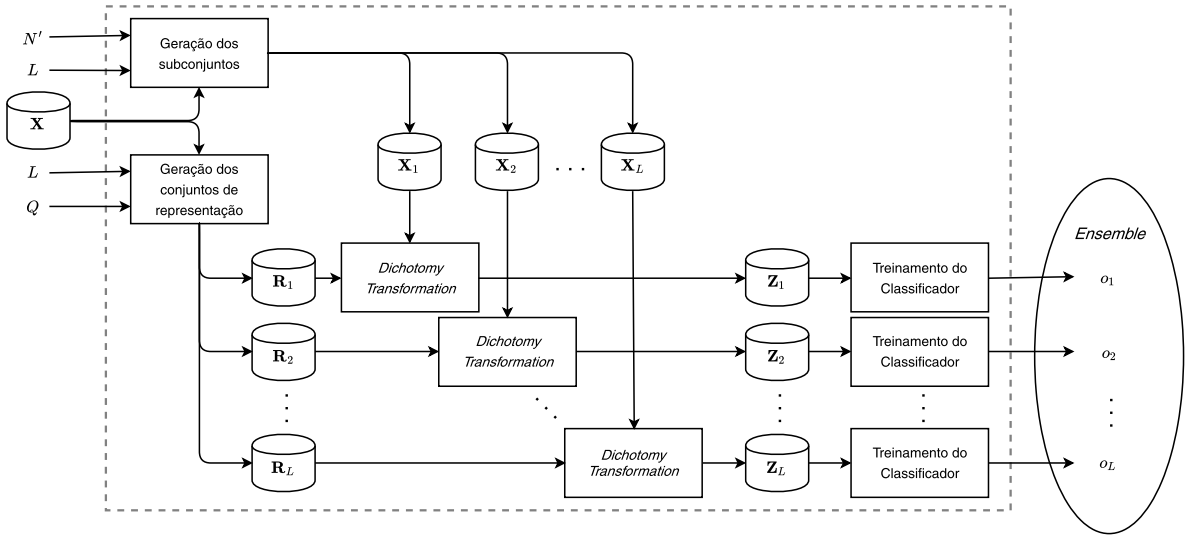


Figura 4.1: Fase de treinamento da CoDiT. $\mathbf{X}$ é o conjunto de treinamento, $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L$ são os subconjuntos de treinamento, $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L$ são os conjuntos de representação, $\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_L$ são os conjuntos obtidos pela DT, L é o tamanho do *ensemble*, N' é o número de documentos dos subconjuntos de treinamento, Q é o número de documentos no conjunto de representação e $o_1, o_2, \ldots, o_L$ são os classificadores do *ensemble*.

4.2 Fase de Treinamento

A Figura 4.1 mostra um fluxograma da fase de treinamento. Nesta fase, o CoDiT recebe como entrada: o conjunto de treinamento $\mathbf{X}$, o tamanho do *ensemble* (L), o tamanho dos subconjuntos de treinamento (N') e o tamanho dos conjuntos de representação (Q); e, como saída, L classificadores. Quatro módulos distintos são utilizados durante essa fase: Geração dos Subconjuntos, Geração dos Conjuntos de Representação, *Dichotomy Transformation* e Treinamento do Classificador.

Durante a fase de treinamento, cada um dos L módulos da *Dichotomy Transformation* recebem como entrada: um subconjunto de treinamento $\mathbf{X}_s$ e um conjunto de representação $\mathbf{R}_s$, no qual $s = \{1, 2, \ldots, L\}$. Dado o funcionamento da CoDiT, algumas características são esperadas dos conjuntos de entrada ($\mathbf{X}_s$ e $\mathbf{R}_s$): i) como o módulo DT aumenta o número de documentos disponíveis para treinamento, é interessante os conjuntos de entrada possuirem uma quantidade de documentos menor do que a original, para evitar aumento desnecessário no custo computacional; ii) CoDiT trata-se de um sistema de múltiplos classificadores e, nestes sistemas, diversidade é um fator importante [125], logo, os vários $\mathbf{X}_s$ e $\mathbf{R}_s$ devem ter documentos diferentes para aumentar a diversidade do *ensemble*; iii) devido à regra de classificação da CoDiT (apresentado na Seção 4.3), o conjunto de representação deve ter a mesma quantidade de documentos por classe. Os módulos de Geração de Subconjuntos e Geração dos Conjuntos de Representação são descritos a seguir e possuem metodologias para gerar conjuntos que possuam essas características.

O módulo de Geração de Subconjuntos recebe o conjunto de treinamento $\mathbf{X}$ como entrada, e gera L subconjuntos de treinamento $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L)$ como saída, cada um com N' documentos

($N' \leq N$). Este módulo utiliza uma subamostragem aleatória, isto é, uma amostragem aleatória, sem reposição, de N' documentos para cada subconjunto de treinamento $\mathbf{X}_s$. Essa metodologia colabora para obter as duas primeiras características listadas anteriormente: i) e ii).

O módulo de Geração dos Conjuntos de Representação recebe o conjunto de treinamento $\mathbf{X}$, o tamanho do *ensemble* (L), e o tamanho do conjunto de representação (Q) como entrada, e gera L conjuntos de representação ($\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L$), cada um com Q documentos. Diversas abordagens podem ser utilizadas para definir quais documentos deverão ser selecionados para gerar um conjunto de representação. Como, por exemplo, um conjunto de representação com todos os documentos do conjunto de treinamento, mas esse procediumento consome muito tempo e é desnecessário [23]. Outras alternativas visam gerar um conjunto de representação menor do que o conjunto de treinamento original, tais como: selecionar os documentos mais dissimilares [23]; algoritmos de seleção de protótipos, como *condensed nearest neighbor*, *edition nearest neighbor*, dentre outros [156, 19]; ou selecionar alguns documentos aleatoriamente. CoDiT utiliza uma seleção aleatória por classe para gerar seus conjuntos de representação. Este processo de seleção não parece interessante a priori, entretanto, em um trabalho anterior [28] foi mostrado que conjuntos de representação obtidos de modo aleatório podem atingir bons resultados. A seleção aleatória por classe é uma amostragem sem reposição, i.e., nenhum documento é selecionado duas vezes; com a restrição de cada classe possuir o mesmo número de documentos. Essa restrição é necessária para evitar viés para algumas classes durante a fase de teste. Portanto, Q documentos ($Q \leq N$) são selecionados para compor cada conjunto de representação e cada classe terá Q/C documentos, atendendo todas as características listadas anteriormente: i), ii) e iii).

Assim, com a geração dos subconjuntos de treinamento $\mathbf{X}_s$ e dos conjuntos de representação $\mathbf{R}_s$, utiliza-se a *Dichotomy Transformation* para geração dos conjuntos $\mathbf{Z}_s$:

$$\begin{aligned}\mathbf{Z}_s = \{ & dt(x_1, p_1), dt(x_1, p_2), \ldots, dt(x_1, p_Q), \\ & dt(x_2, p_1), dt(x_2, p_2), \ldots, dt(x_2, p_Q), \ldots \\ & dt(x_{N'}, p_1), dt(x_{N'}, p_2), \ldots, dt(x_{N'}, p_Q)\},\end{aligned} \tag{4.1}$$

no qual cada função $dt(x_i, p_k)$[1] gera um novo documento, como mostrado na Equação (2.26). Deste modo, cada conjunto $\mathbf{Z}_s$ tem $N' \times Q$ documentos, no qual cada documento pertence à classe positiva ($class(x_i) = class(p_k)$) ou à classe negativa ($class(x_i) \neq class(p_k)$). Substituindo

[1]Tendo em vista que outras métricas podem ser aplicadas com a *Dichotomy Transformation*, uma métrica inspirada na similaridade do cosseno foi proposta nesta tese. O CoDiT funciona da mesma maneira, modificando apenas o uso da Equação (2.26) pela Equação (4.7). Detalhes do funcionamento da métrica proposta na Seção 4.5.

a Equação (2.26) na Equação (4.1), temos:

$$\mathbf{Z}_s = \left\{ \begin{array}{l} \left[\begin{array}{c} |w_1^{x_1} - w_1^{p_1}| \\ |w_2^{x_1} - w_2^{p_1}| \\ \vdots \\ |w_V^{x_1} - w_V^{p_1}| \end{array} \right], \left[\begin{array}{c} |w_1^{x_1} - w_1^{p_2}| \\ |w_2^{x_1} - w_2^{p_2}| \\ \vdots \\ |w_V^{x_1} - w_V^{p_2}| \end{array} \right], \cdots, \left[\begin{array}{c} |w_1^{x_1} - w_1^{p_Q}| \\ |w_2^{x_1} - w_2^{p_Q}| \\ \vdots \\ |w_V^{x_1} - w_V^{p_Q}| \end{array} \right], \\ \left[\begin{array}{c} |w_1^{x_2} - w_1^{p_1}| \\ |w_2^{x_2} - w_2^{p_1}| \\ \vdots \\ |w_V^{x_2} - w_V^{p_1}| \end{array} \right], \left[\begin{array}{c} |w_1^{x_2} - w_1^{p_2}| \\ |w_2^{x_2} - w_2^{p_2}| \\ \vdots \\ |w_V^{x_2} - w_V^{p_2}| \end{array} \right], \cdots, \left[\begin{array}{c} |w_1^{x_2} - w_1^{p_Q}| \\ |w_2^{x_2} - w_2^{p_Q}| \\ \vdots \\ |w_V^{x_2} - w_V^{p_Q}| \end{array} \right], \cdots \\ \left[\begin{array}{c} |w_1^{x_{N'}} - w_1^{p_1}| \\ |w_2^{x_{N'}} - w_2^{p_1}| \\ \vdots \\ |w_V^{x_{N'}} - w_V^{p_1}| \end{array} \right], \left[\begin{array}{c} |w_1^{x_{N'}} - w_1^{p_2}| \\ |w_2^{x_{N'}} - w_2^{p_2}| \\ \vdots \\ |w_V^{x_{N'}} - w_V^{p_2}| \end{array} \right], \cdots, \left[\begin{array}{c} |w_1^{x_{N'}} - w_1^{p_Q}| \\ |w_2^{x_{N'}} - w_2^{p_Q}| \\ \vdots \\ |w_V^{x_{N'}} - w_V^{p_Q}| \end{array} \right] \end{array} \right\} \quad (4.2)$$

no qual $w_h^{x_i}$ e $w_h^{p_k}$ são as h-ésimas características dos documentos x_i e p_k respectivamente, N' é o número de documentos no subconjunto de treinamento $\mathbf{X}_s$, Q é o número de documentos no conjunto de representação $\mathbf{R}_s$ e V é o número de características.

Finalmente, cada um dos L conjuntos $\mathbf{Z}_s$ é apresentado para seu respectivo módulo de Treinamento do Classificador para treinar o classificador o_s. Cada classificador o_s é binário, pois a DT reduziu todos os problemas para apenas duas classes, independente da quantidade de classes orginais (C). Assim, a saída final da fase de treinamento são L classificadores binários $(o_1, o_2, \ldots, o_L)$.

4.3 Fase de teste

A Figura 4.2 mostra o fluxograma da fase de teste. Nesta fase, cada documento do conjunto de teste $y_j \in \mathbf{Y}$ é apresentado, um de cada vez, para classificação. A fase de teste também possui, como entrada: L conjuntos de representação $(\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L)$ e L classificadores; e, como saída, a resposta $class(y_j)$ para cada documento do conjunto de teste. Três módulos distintos são utilizados durante essa fase: *Dichotomy Transformation*, Classificação, e Fusor.

O módulo *Dichotomy Transformation* recebe, como entrada, um documento $y_j \in \mathbf{Y}$ e um conjunto de representação $\mathbf{R}_s$, obtido durante a fase de treinamento; e gera um conjunto $\mathbf{Z}'_s$:

$$\mathbf{Z}'_s = \left\{ z'_{y,1}, z'_{y,2}, \ldots, z'_{y,Q} \right\} \quad (4.3)$$

$$= \left\{ dt(y_j, p_1), dt(y_j, p_2), \ldots, dt(y_j, p_Q) \right\}, \quad (4.4)$$

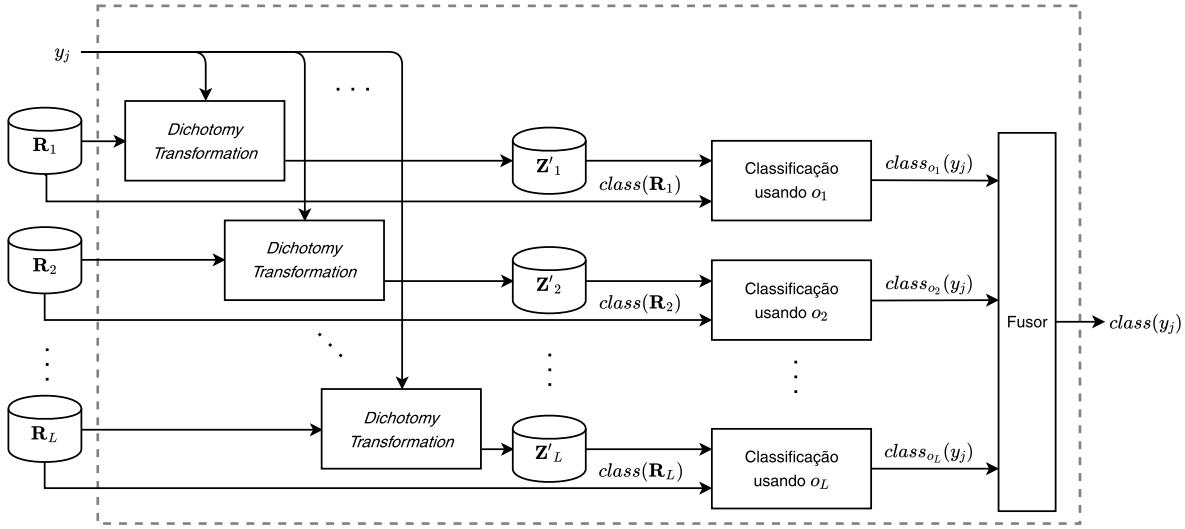


Figura 4.2: Fase de teste da arquitetura proposta. $y_j \in \mathbf{Y}$ é um documento do conjunto de teste, $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L$, são os conjuntos de representação, $\mathbf{Z}'_1, \mathbf{Z}'_2, \ldots, \mathbf{Z}'_L$ são os conjuntos dicotomizados baseados na função $dt(y_j, \mathbf{R}_s$, $class(\mathbf{R}_s)$ são as classes de todos os documentos $p_k \in \mathbf{R}_s$, $class(y_j)$ é a classe resultante de $y_j \in \mathbf{Y}$ e L é o tamanho do *ensemble*.

no qual o documento $z'_{j,k}$ é obtido com a função $dt(y_j, p_k)$ como mostrado na Equação (2.26), sendo $p_k \in \mathbf{R}_s$ ($k = \{1, 2, \ldots Q\}$). Deste modo, cada módulo DT gera Q documentos. Substituindo a Equação (2.26) na Equação (4.4), temos:

$$\mathbf{Z}'_s = \left\{ \begin{bmatrix} |w_1^{y_j} - w_1^{p_1}| \\ |w_2^{y_j} - w_2^{p_1}| \\ \vdots \\ |w_V^{y_j} - w_V^{p_1}| \end{bmatrix}, \begin{bmatrix} |w_1^{y_j} - w_1^{p_2}| \\ |w_2^{y_j} - w_2^{p_2}| \\ \vdots \\ |w_V^{y_j} - w_V^{p_2}| \end{bmatrix}, \cdots, \begin{bmatrix} |w_1^{y_j} - w_1^{p_Q}| \\ |w_2^{y_j} - w_2^{p_Q}| \\ \vdots \\ |w_V^{y_j} - w_V^{p_Q}| \end{bmatrix} \right\} \tag{4.5}$$

no qual $w_h^{y_j}$ e $w_h^{p_k}$ são as h-ésimas características dos documentos y_j e p_k respectivamente, e V é o número de características.

A classificação é realizada usando a saída do módulo DT ($\mathbf{Z}'_s$) e as classes de cada documento do conjunto de representação ($class(\mathbf{R}_s)$). Por uma questão de clareza, a Figura 4.3 expande um dos módulos "Classificação usando o_s" mostrado na Figura 4.2. Logo, cada documento $z'_{j,k} \in \mathbf{Z}'_s$ é usado como entrada do classificador o_s, gerando como saída a probabilidade de $z'_{j,k}$ ser classificado como pertencente à classe positiva ($p(z'_{j,k}|c_\bullet)$). As probabilidades são combinadas no módulo Fusão Interna (Figure 4.3), resultando em:

$$class_{o_s}(y_j) = \underset{l=\{1,2,\ldots,C\}}{\operatorname{argmax}} \sum_{\substack{p_k \in \mathbf{R}_s \\ class(p_k)=c_l}} p(z'_{j,k}|c_\bullet). \tag{4.6}$$

Em outras palavras, a saída da Fusão Interna é a classe com o maior somatório das probabilidades de pertencer à classe positiva ($p(z'_{j,k}|c_\bullet)$) dentre as classes dos documentos do conjunto de representação. Por conta desse somatório por classe é que o conjunto de

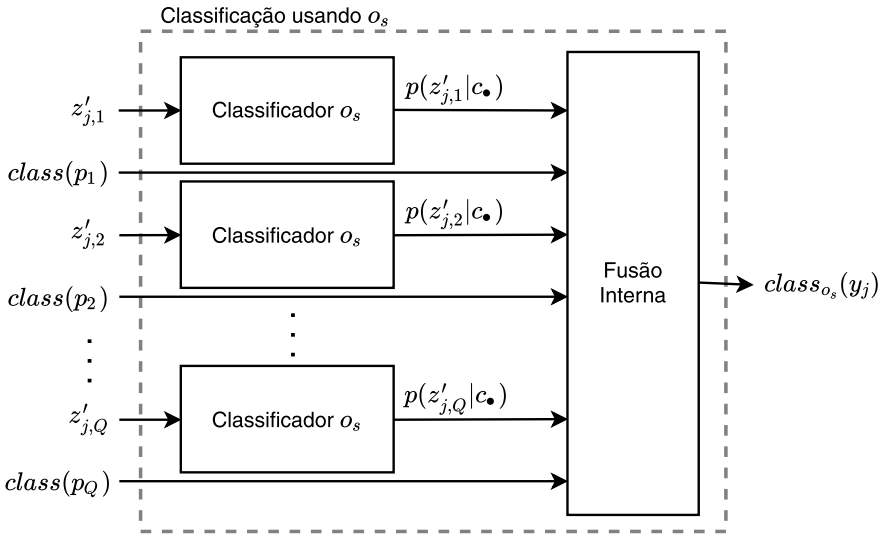


Figura 4.3: Expansão do módulo "Classificação usando o_s" mostrado na Figura 4.2. $z'_{j,k}$, $class(p_k)$ e $p(z'_{j,k}|c_\bullet)$ ($k = \{1,2,\ldots,Q\}$ são, respectivamente, os documentos do conjunto $\mathbf{Z}_s$, as classes dos documentos do conjunto de representação $\mathbf{R}_s$ e a probabilidade de classificar o documento $z'_{j,k}$ como positivo. A saída $class_{o_s}(y_j)$ é a classe resultante em cada módulo de classificação.

representação precisa ter a mesma quantidade de documentos por classe. Deste modo, a soma é justa para todas as classes, pois se uma classe tivesse mais documentos presentes do que outra no conjunto de representação seria mais fácil para essa classe de atingir um maior somatório das probabilidades.

A saída final $class(y_j)$, será obtida combinando as saídas $class_{o_s}(y_j)$ dos L módulos "Classificação usando o_s" presentes na Figura 4.2. Para tal, a combinação dessas L respostas é realizada com voto majoritário [153], escolhido por sua simplicidade e ampla utilização em toda a literatura [157, 154].

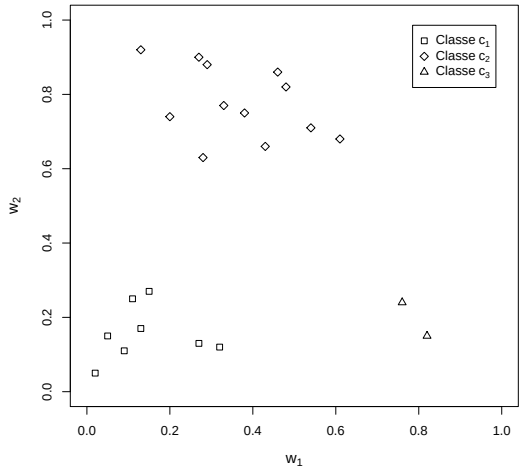
4.4 Exemplo *Toy*

Essa seção apresenta um passo-a-passo da CoDiT com enfoque nos detalhes do classificador o_1. A Figura 4.4(a) mostra o conjunto de treinamento $\mathbf{X}$ com 22 documentos ($N = 22$) distribuídos entre 3 classes ($\mathbf{C} = \{c_1, c_2, c_3\}$). O restante dos parâmetros de entrada da CoDiT são: o tamanho do *ensemble* ($L = 3$), o tamanho dos subconjuntos de treinamento ($N' = 6$) e o tamanho dos conjuntos de representação ($Q = 6$).

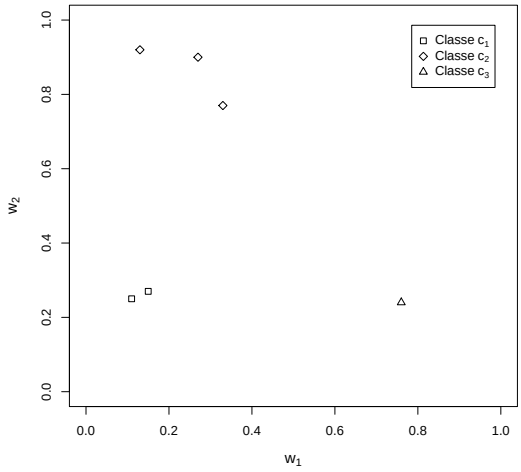
CoDiT inicia com os módulos de Geração de Subconjuntos e Geração dos Conjuntos de Representação. Ambos os módulos geram $L = 3$ subconjuntos do conjunto de treinamento, sendo que cada módulo realiza uma seleção de documentos de modo diferente. O módulo de Geração de Subconjuntos gera os subconjuntos de treinamento $\mathbf{X}_1$, $\mathbf{X}_2$ e $\mathbf{X}_3$, cada um com $N' = 6$ documentos selecionados aleatoriamente do conjunto de treinamento. A Figura 4.4(b) mostra os documentos de $\mathbf{X}_1$. Enquanto que o módulo de Geração dos Conjuntos de Representação

gera os conjuntos de representação $\mathbf{R}_1$, $\mathbf{R}_2$ e $\mathbf{R}_3$, cada um com $Q = 6$ documentos selecionados aleatoriamente com a restrição de selecionar o mesmo número de documentos por classe, logo, $Q/C = 2$ documentos são selecionados para cada classe. A Figura 4.4(c) mostra os documentos de $\mathbf{R}_1$. Depois, o conjunto $\mathbf{Z}_1$ é obtido com a função $dt(\mathbf{X}_1, \mathbf{R}_1)$ usando a Equação (2.26). A Figura 4.4(d) apresenta os documentos transfomados de $\mathbf{Z}_1$, divididos em 2 classes, a classe positiva (•) e a classe negativa (○).

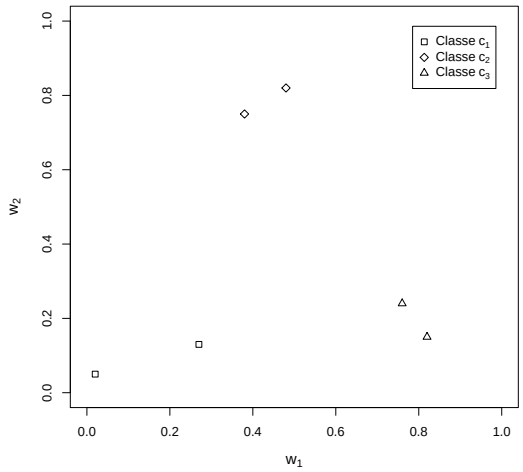
O passo final da fase de treinamento é treinar os $L = 3$ classificadores, utilizando os documentos de cada conjunto $\mathbf{Z}_s$, no qual $s = \{1, 2, 3\}$. A Figura 4.5(a) mostra o hiperplano separador do classificador o_1, obtido pelo treinamento com os documentos de $\mathbf{Z}_1$.



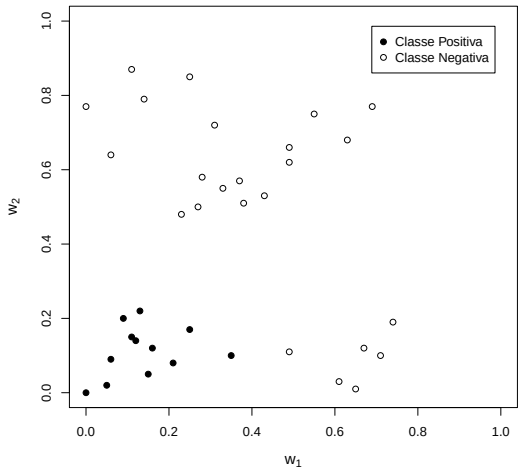
(a) Conjunto de treinamento $\mathbf{X}$



(b) Subconjunto de treinamento $\mathbf{X}_1$



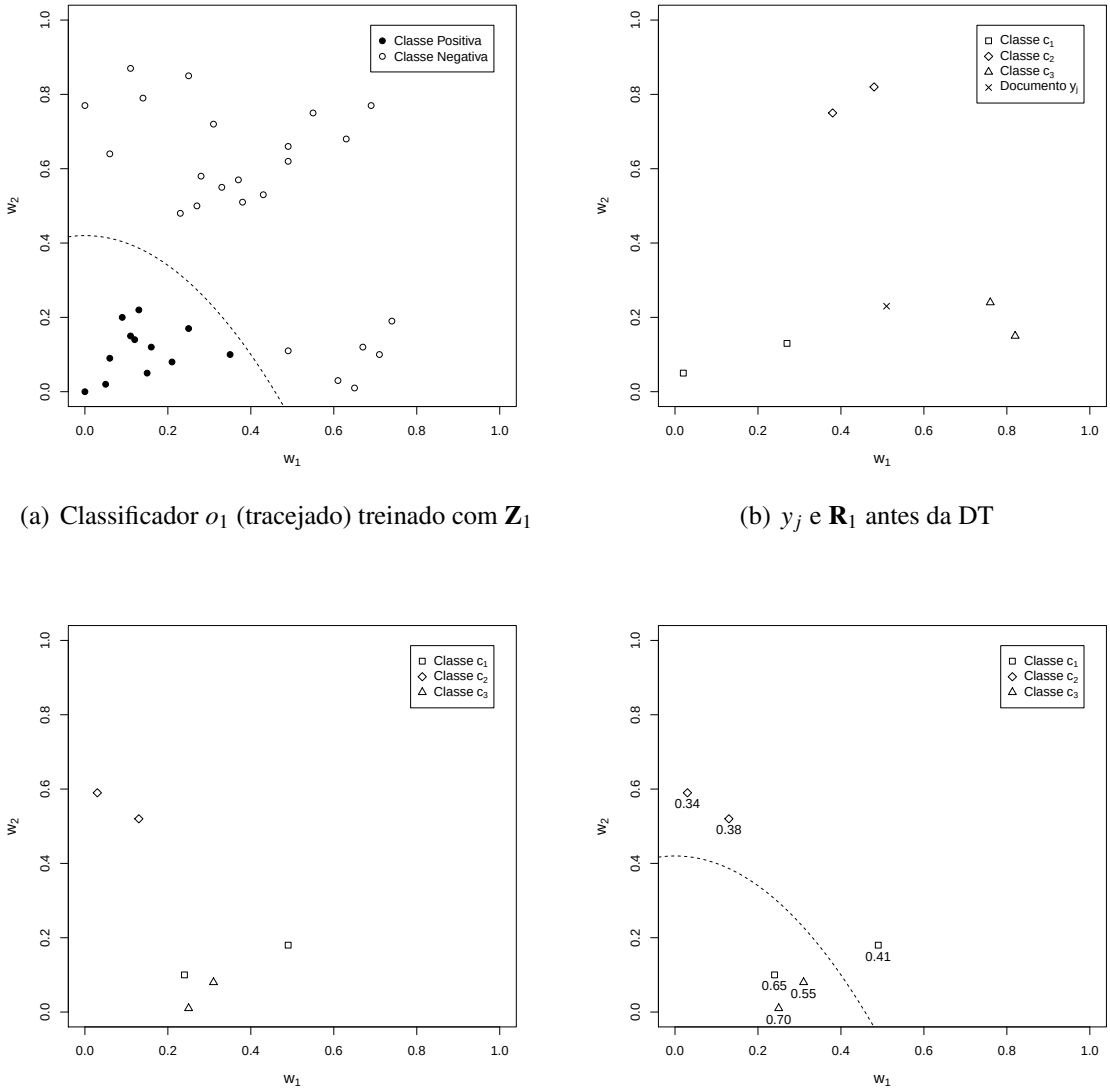
(c) Conjunto de representação $\mathbf{R}_1$



(d) Conjunto $\mathbf{Z}_1$

Figura 4.4: Examplo da *Dichotomy Transformation* com conjuntos de entrada $\mathbf{X}_1$ (b) e $\mathbf{R}_1$ (c), ambos obtidos com base no conjunto de treinamento original (a), e o conjunto de saída $\mathbf{Z}_1$ (d). Todos os conjuntos representados por duas características (w_1 e w_2).

A Figura 4.5(b) mostra um documento de teste y_j juntamente com os documentos do conjunto de representação $\mathbf{R}_1$, simplesmente para visualizar os documentos antes da DT. A Figura 4.5(c) mostra os 6 documentos obtidos com a *Dichotomy Transformation* $dt(y_j, \mathbf{R}_1)$ usando a Equação (2.26). A Figura 4.5(d) mostra as probabilidades, dadas pelos classificador o_1, de cada documento transformado pertencer à classe positiva. Então, o documento y_j é classificado por o_1 como pertencente à classe c_3, por conta do somatório das probabilidades $p(z'_{j,k}|c_\bullet)$ de modo que $class(p_k) = c_3$ possui o maior somatório em comparação com as demais classes. Mais especificamente, os somatórios são: 1.06 $(0.65 + 0.41)$ para classe c_1, 0.72 $(0.34 + 0.38)$ para classe c_2 e 1.25 $(0.7 + 0.55)$ para classe c_3.



(a) Classificador o_1 (tracejado) treinado com $\mathbf{Z}_1$

(b) y_j e $\mathbf{R}_1$ antes da DT

(c) $\mathbf{Z}'_1 = \{z'_{j,1}, z'_{j,2}, \ldots, z'_{j,Q}\}$

(d) Probabilidades $p(z'_{j,k}|c_\bullet)$

Figura 4.5: Com o classificador treinado (a), o documento y_j é apresentado à arquitetura (b), passando pela *Dichotomy Transformation* e gerando $Q = 6$ documentos (c). Os números (d) são as probabilidades $p(z'_{j,k}|c_\bullet)$. Deste modo, $class_{o_1}(y_j) = c_3$. Todos os documentos deste exemplo são representados por duas características (w_1 e w_2)

Dado que $L = 3$, outros dois classificadores darão suas respostas. Suponha que y_j é classificado como c_1 pelo classificador o_2 e como c_3 pelo classificador o_3. Assim, com voto majoritário, a resposta final é $class(y_j) = c_3$, pois c_3 tem dois votos (o_1 e o_3), enquanto que c_1 tem apenas um voto (o_2) e c_2 não tem votos.

4.5 Função de Similaridade

A *Dichotomy Transformation*, como apresentada na Seção 2.4, usa o valor absoluto da diferença vide Equação (2.26), que possui um comportamento similar à uma métrica de distância, como distância Euclidiana. Distância Euclidiana, quando utilizada para verificar vizinhança, não gera um classificador de k vizinhos mais próximos tão efetivo quanto a similaridade do cosseno [158], que possui um desempenho assintoticamente superior com vetores esparsos [159, 160], fator presente devido à representação *Bag-of-Words*.

Tendo em vista as qualidades da similaridade do cosseno, uma função inspirada na similaridade do cosseno para a *Dichotomy Transformation* foi proposta. Essa função está localizada nessa Seção para facilitar a contextualização da proposta com relação à *Dichotomy Transformation*. Assim, dado um par de documentos a_i e b_j, a *Dichotomy Transformation* é calculada como:

$$dts(a_i, b_j) = \begin{bmatrix} \frac{min(w_1^{a_i}, w_1^{b_j})}{max(w_1^{a_i}, w_1^{b_j})} \\ \frac{min(w_2^{a_i}, w_2^{b_j})}{max(w_2^{a_i}, w_2^{b_j})} \\ \vdots \\ \frac{min(w_V^{a_i}, w_V^{b_j})}{max(w_V^{a_i}, w_V^{b_j})} \end{bmatrix}, \tag{4.7}$$

no qual $w_h^{a_i}$ e $w_h^{b_j}$ são as h-ésimas características dos documentos a_i e b_j respectivamente, e V é o número de características. Para evitar valores incalculáveis, em caso do valor zero na função $max(\cdot,\cdot)$, é assumido o valor zero para a característica. Deste modo, após a *Dichotomy Transformation*, as características estarão no intervalo $[0,1]$, no qual características próximas à 1 indicam que as características são similares.

A Tabela 4.1 mostra um exemplo ilustrativo do funcionamento desta nova métrica e por qual motivo esta poderia funcionar em problemas de Classificação de Documentos. Neste exemplo, a *Dichotomy Transformation* recebe como entrada os conjuntos: $\mathbf{A} = \{a_1, a_2\}$ e $\mathbf{B} = \{b_1, b_2, b_3\}$, cada documento possuindo 9 características ($V = 9$). Para simplificar o processo, o problema original possui apenas 2 classes (c_1 and c_2), sendo esta quantidade o suficiente para simular e observar o comportamento das duas funções. DT gera um novo conjunto com 2 classes: positiva ($c_\bullet$) e negativa ($c_\circ$). Usando a Equação (2.26), é esperado que documentos da classe $c_\bullet$ tenha características com valores pequenos (próximos de zero), enquanto que usando a Equação

Tabela 4.1: Exemplo da *Dichotomy Transformation* usando as funções $dt(\cdot,\cdot)$ e $dts(\cdot,\cdot)$. No primeiro terço das linhas estão os documentos originais, no segundo terço os documentos obtidos com a função $dt(\cdot,\cdot)$, e o último terço os documentos obtidos com a função $dts(\cdot,\cdot)$.

Doc.	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	**C**	$\sum w_h$
a_1	0	2	0	3	0	0	2	0	0	c_1	
a_2	4	0	2	0	2	0	0	4	0	c_2	
b_1	0	0	0	4	0	1	1	0	0	c_1	n.a.
b_2	0	0	3	0	1	0	3	0	0	c_2	
b_3	0	3	2	0	2	2	0	0	2	c_2	
$dt(a_1,b_1)$	0	2	0	1	0	1	1	0	0	$c_\bullet$	5
$dt(a_1,b_2)$	0	2	3	3	1	0	1	0	0	$c_\circ$	10
$dt(a_1,b_3)$	0	1	2	3	2	2	2	0	2	$c_\circ$	14
$dt(a_2,b_1)$	4	0	2	4	2	1	1	4	0	$c_\circ$	18
$dt(a_2,b_2)$	4	0	2	0	2	0	0	4	0	$c_\bullet$	12
$dt(a_2,b_3)$	4	3	0	0	0	2	0	4	2	$c_\bullet$	15
$dts(a_1,b_1)$	0	0	0	0,75	0	0	0,5	0	0	$c_\bullet$	1,25
$dts(a_1,b_2)$	0	0	0	0	0	0	0,66	0	0	$c_\circ$	0,66
$dts(a_1,b_3)$	0	0,66	0	0	0	0	0	0	0	$c_\circ$	0,66
$dts(a_2,b_1)$	0	0	0	0	0	0	0	0	0	$c_\circ$	0,00
$dts(a_2,b_2)$	0	0	0,66	0	0,5	0	0	0	0	$c_\bullet$	1,16
$dts(a_2,b_3)$	0	0	1	0	1	0	0	0	0	$c_\bullet$	2,00

(4.7) é esperado que os documentos da classe $c_\bullet$ tenha características com valores elevados (próximos de um). O propósito da última coluna ($\sum w_h$) é para verificar se as características obtidas pela *Dichotomy Transformation* comportam-se como o esperado. Por exemplo, usando a Equação (2.26), documentos da classe positiva ($dt(a_2,b_2)$ and $dt(a_2,b_3)$) possuem um valor maior de $\sum w_h$ do que documentos da classe negativa ($dt(a_1,b_2)$ e $dt(a_1,b_3)$), sendo que era esperado que eles possuíssem valores menores. Enquanto que com a função inspirada no cosseno, todos os $\sum w_h$ são maiores na classe positiva do que na classe negativa, como é o esperado dado a Equação (4.7). Este comportamento mostra que $dts(\cdot,\cdot)$, neste exemplo, é mais robusto que a função $dt(\cdot,\cdot)$.

4.6 Considerações Finais

Neste capítulo foi apresentada a segunda arquitetura proposta: CoDiT, do inglês *Combined Dichotomy Transformations*. Trata-se de um sistema de Classificação de Documentos que utiliza *Dichotomy Transformation* em conjunto com Combinação de Classificadores, para gerar um sistema robusto composto por classificadores fortes. A arquitetura é modular, mas a metodologia de diversos módulos foi especificada devido às restrições impostas pelos dados obtidos após a *Dichotomy Transformation* e também visando a diversidade do *ensemble*. No Capítulo 5 serão apresentados os experimentos com as comparações do desempenho de algumas das configurações da arquitetura proposta com relação a outros métodos da literatura.

5

Experimentos

Neste capítulo são apresentados os experimentos realizados para verificar a performance das propostas. Nas próximas seções são apresentadas as bases de dados utilizadas nos experimentos (Seção 5.1) e as configurações adotadas nos experimentos (Seção 5.2). As demais seções foram divididas de acordo com as propostas, iniciando com experimentos preliminares da arquitetura CoDiS (Seção 5.3), para definição de alguns parâmetros importantes e também compreensão de alguns pontos importantes. Após os experimentos preliminares são apresentados os experimentos do CoDiS (Seção 5.4). Tendo em vista alguns resultados interessantes obtidos nos experimentos preliminares, foi adotada uma abordagem sem combinação baseada no CoDiS (Seção 5.5). Na Seção 5.6 são apresentados os experimentos do CoDiT, utilizando as duas métricas para a *Dichotomy Transformation*. Após a apresentação de todas as propostas individualmente, todas são comparadas na Seção 5.7. Finalmente, as considerações finais são realizadas na Seção 5.8.

5.1 Bases de Dados

Foram utilizadas 47 bases de dados diferentes nos experimentos. A Tabela 5.1 mostra as características das bases de dados utilizadas. Todas as bases de dados podem ser encontradas ou geradas com arquivos obtidos na *Internet*. A base de dados TDT2T30 em `http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html` e todas as outras em `http://sites.labic.icmc.usp.br/text_collections/`. As bases de dados são dos mais diferentes domínios, visando avaliação da robustez das propostas, isto é, a capacidade das propostas atingirem bons resultados em domínios diferentes na mesma aplicação. As bases de dados com Domínio especificado como TREC são provenientes da Text REtrieval conference (`http://trec.nist.gov`). Classic4 e WebKB foram manipuladas para gerar duas outras bases de dados: Classic4 para gerar a Classic3, removendo a classe com mais documentos e, WebKB para gerar WebKB4, utilizando apenas 4 categorias: *course*, *faculty*, *project* and *student*. As bases de dados passaram por uma remoção de *stopwords* (Lista usada no Anexo **??**) e *stemming* com o algoritmo de Porter [55].

Tabela 5.1: Base de dados utilizadas nos experimentos.

Bases	Domínio	Documentos	Características	Categorias
20 Newsgroup (20Ng)	*E-mails*	18828	45433	20
ACM	Artigos Científicos	3493	60768	40
Classic3	Resumos	3891	7749	3
Classic4	Resumos	7095	7749	4
CSTR	Artigos Científicos	299	1726	4
Dmoz Computers 500 (DmComp)	*Web Pages*	9500	5011	19
Dmoz Health 500 (DmHealth)	*Web Pages*	6500	4217	13
Dmoz Science 500 (DmScience)	*Web Pages*	6000	4821	12
Dmoz Sports 500 (DmSport)	*Web Pages*	13500	5682	27
Enron Top20 (Enron20)	*E-mails*	13199	18194	20
FBIS	Notícias	2463	2001	17
Hitech	Notícias	2301	12942	6
Irish Sentiment (Irish)	Análise de Sentimento	1660	8659	3
La1s	Notícias	3204	13196	6
La2s	Notícias	3075	12433	6
LATimes	Notícias	6278	10020	6
Multi Domain Sentiment (MDS)	Análise de Sentimento	8000	13360	2
New 3	Notícias	9558	26833	44
NFS	Artigos Científicos	10524	3888	16
Oh0	Documentos Médicos	1003	3183	10
Oh5	Documentos Médicos	918	3013	10
Oh10	Documentos Médicos	1050	3239	10
Oh15	Documentos Médicos	913	3103	10
Ohscal	Documentos Médicos	11162	11466	10
Ohsumed-400 (Ohsumed)	Documentos Médicos	9200	13512	23
Opinosis	Análise de Sentimento	6457	2693	51
Re0	Notícias	1504	2887	13
Re1	Notícias	1657	3759	25
Re8	Notícias	7674	8901	8
Reuters-21578	Notícias	14175	14035	120
Review Polarity (Polarity)	Análise de Sentimento	2000	15697	2
Reviews	Notícias	4069	22927	5
SpamAssassin (SpamAss)	*E-mails*	9348	97851	2
SpamTrec-3000 (Spam3000)	*E-mails*	5982	100464	2
SyskillWebert (SsW)	*Web Pages*	334	4340	4
TDT2T30	Notícias	9394	36771	30
Tr11	TREC	414	6430	9
Tr12	TREC	313	5805	8
Tr21	TREC	335	7903	6
Tr23	TREC	204	5833	6
Tr31	TREC	927	10129	7
Tr41	TREC	878	7455	10
Tr45	TREC	690	8262	10
WAP	*Web Pages*	1560	8461	20
WebACE	*Web Pages*	3900	8880	21
WebKB	*Web Pages*	8282	22892	7
WebKB4	*Web Pages*	4199	22892	4

5.2 Configurações dos Experimentos

Todos os experimentos utilizaram as seguintes configurações:

- Validação cruzada estratificada com *10-folds*. Isto é, dez conjuntos (*folds*) com documentos divididos de modo proporcional às classes. Estes *folds* são unidos para gerar diferentes conjuntos de treinamento e de teste. O conjunto de treinamento é composto por nove *folds* (90%) e o conjunto de teste por um *fold* (10%). Portanto, cada experimento é realizado dez vezes, cada um utilizando um conjunto de teste diferente e, consequentemente, um conjunto de treinamento diferente. Os resultados apresentados nas tabelas são as médias desses dez resultados seguidos, em alguns casos, pelo desvio padrão entre parênteses;

- Vetor de características foi gerado seguindo a abordagem BoW com valores da Frequência do Termo (detalhes na Seção 2.1.2);

- O classificador base (o_s) utilizado foi o SVM com *kernel* linear e demais parâmetros em *default* do LIBSVM[1], isto é, tipo utilizado foi C-SVC e $C = 1$. No caso do CoDiT, como é necessário a resposta em probabilidade, o parâmetro *probability estimates* foi 1;

- Para avaliar o desempenho dos algoritmos foram utilizadas as medidas de avaliação macro-F1 e micro-F1 descritas na Seção 2.1.4. Entretanto, os valores originais destas medidas estão no intervalo $[0, 1]$, para facilitar a visualização, os valores foram multiplicados por 100;

- As comparações entre os métodos foram realizadas usando dois diferentes testes estatísticos: teste *Wilcoxon signed rank* [161] e teste de *Friedman* [162]. Os valores comparados foram a micro-F1 e a macro-F1 de cada *fold*. Os testes de *Wilcoxon* mais relevantes passaram por um ajuste do *p-value* realizado pelo método de *Holm-Bonferroni* [163]. A seguinte iconografia foi utilizada em alguns momentos para facilitar a visualização das comparações: *p-value* menor ou igual a 0,01, representa uma evidência forte de que um método possui uma performance superior ($\gg$) ou inferior ($\ll$) ao outro; *p-value* maior que 0,01 e menor que 0,05, representa uma evidência fraca de que um método possui uma performance superior ($>$) ou inferior ($<$) a outro; e *p-value* maior que 0,05 representa que a diferença entre os métodos não é significante ($\sim$ ou $=$) estatisticamente. Por fim, o teste de *Friedman*, também utilizado globalmente, compara todos os métodos envolvidos na comparação utilizando dois diagramas CD (*critical difference*) [162], um para macro-F1 e outro para micro-F1, que utiliza como base os *ranks* médios;

[1] `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`

- A combinação dos classificadores foi realizada usando voto majoritário [164]. Portanto, um documento $y_j \in \mathbf{Y}$ é classificado como pertencente à classe com maior número de votos dentre os L classificadores. Foram utilizados 10 valores para a quantidade de classificadores do *pool*: $L = \{10, 20, \ldots, 100\}$.

- Todos os algoritmos de seleção de protótipos baseados em vizinhança utilizaram $v = 5$ (quantidade de vizinhos) e similaridade do cosseno para encontrar os vizinhos mais similares. Os algoritmos de seleção de protótipos foram escolhidos devido à variação entre eles que, de acordo com uma taxonomia da literatura [115], são métodos de condensação, filtro decremental, filtro *batch* e híbrido. ATISA2 foi proposto após a publicação da taxonomia mas seria classificado como híbrido. ATISA2 foi utilizado por ser a versão mais balanceada (redução x performance) dentre os três algoritmos ATISA [122]. Além da variação pela taxonomia, os algoritmos escolhidos variam em seu poder de redução, alguns selecionando muitos protótipos (ENN e All-kNN) e outros pouquíssimos protótipos (ATISA2 e DROP3). Todos os algoritmos de seleção de protótipos são descritos na Seção 2.5.

- Os experimentos foram realizados em um computador pessoal com Windows 8 (64 bits) com processador Intel Core i5 de 3.0 GHz e 4096 MB de memória RAM DDR3.

Quaisquer exceções ou variações serão destacadas nos respectivos experimentos.

5.3 Experimentos Preliminares - CoDiS

O principal objetivo dos experimentos preliminares é reduzir a quantidade de parâmetros utilizados nos experimentos. Adicionalmente, os experimentos preliminares permitem a observação de algumas particularidades dos parâmetros. A análise dos parâmetros seguiu uma metodologia *top-down*, isto é, os testes iniciam com diversas configurações diferentes do CoDiS para terminar com apenas uma configuração. As seleções das configurações foram realizadas de acordo com alguma medida de avaliação, que na maioria dos casos foi a micro-F1.

Nesta etapa, são utilizadas seis bases de dados: CSTR, Irish, Oh0, Polarity, Tr23 e WAP (Tabela 5.1). A seleção das bases de dados para esta etapa foi baseada em seu tamanho, selecionando as menores bases para execução dos experimentos em tempo hábil; e origem, pois diversas bases de dados possuem um comportamento similar por serem provenientes da mesma fonte. Por exemplo, as bases Oh5 ou Tr11 não foram selecionadas, pois as bases Oh0 e Tr23 foram selecionadas e possuem a mesma procedência.

As próximas seções focam em responder as seguintes perguntas: Como escolher os protótipos dos L conjuntos de representação? Isto é, definir o algoritmo de seleção de protótipos para obter os conjuntos de representação; qual a métrica para mudança de representação é melhor aproveitada em uma combinação de classificadores? Isto é, definir a métrica utilizada na mudança de representação (similaridade do cosseno ou distância Euclidiana); e, quantos documentos no

Tabela 5.2: Comparação da performance (micro-F1) dos classificadores individuais obtidos com algoritmos de seleção de protótipos combinados (colunas Classificadores Individuais), combinação dos classificadores obtidos com seleção de protótipos (coluna Algoritmo de Classificadores Combinados) e combinação dos classificadores obtidos com seleção aleatória de protótipos (coluna Aleatória de Classificadores Combinados). Em negrito está o melhor resultado absoluto para cada base de dados e entre parênteses está o desvio padrão.

		Classificadores Individuais							Classificadores Combinados	
Bases	Métrica	CNN	RNN	ENN	All-kNN	IB2	Drop3	ATISA2	Algoritmo	Aleatória
CSTR	cosseno	79,92 (5,45)	78,99 (7,19)	82,61 (6,52)	**83,33** (6,60)	77,64 (8,07)	71,32 (8,48)	77,69 (8,63)	80,68 (7,42)	79,96 (7,65)
	Euclidiana	78,31 (7,52)	77,71 (7,77)	74,69 (6,72)	75,86 (5,67)	79,26 (7,12)	75,27 (7,04)	76,15 (4,29)	80,31 (5,70)	81,25 (4,62)
Irish	cosseno	**68,55** (3,52)	67,64 (4,01)	68,74 (3,92)	66,14 (2,74)	67,04 (3,77)	61,63 (3,82)	66,75 (3,52)	67,29 (3,83)	67,11 (2,88)
	Euclidiana	64,27 (3,72)	63,79 (3,15)	60,96 (2,37)	58,67 (3,21)	62,41 (4,17)	60,72 (4,84)	60,78 (2,97)	65,17 (3,83)	65,05 (4,07)
Oh0	cosseno	86,35 (3,27)	85,46 (3,83)	**86,57** (3,68)	85,26 (4,10)	84,86 (3,02)	78,09 (4,24)	82,95 (3,01)	85,77 (3,12)	86,37 (3,71)
	Euclidiana	83,79 (3,37)	84,25 (3,87)	81,19 (2,84)	81,38 (2,75)	83,28 (3,21)	78,02 (3,06)	80,50 (2,86)	84,79 (3,75)	85,68 (4,12)
Polarity	cosseno	82,35 (2,70)	82,00 (2,83)	**83,35** (3,15)	80,90 (2,59)	81,50 (2,37)	66,50 (4,15)	81,40 (2,82)	82,05 (2,95)	82,40 (3,04)
	Euclidiana	81,80 (2,50)	80,00 (2,60)	78,80 (3,02)	78,30 (2,35)	80,50 (2,74)	68,25 (3,51)	79,20 (3,38)	82,90 (1,91)	82,00 (2,46)
Tr23	cosseno	75,74 (5,92)	72,85 (6,21)	86,09 (8,92)	83,20 (9,44)	72,80 (7,34)	55,81 (10,18)	72,89 (8,77)	74,07 (6,96)	75,29 (7,64)
	Euclidiana	82,66 (7,08)	83,67 (8,94)	73,37 (8,90)	75,64 (8,87)	82,79 (6,47)	64,08 (9,45)	82,52 (9,20)	**86,41** (8,24)	83,27 (12,12)
WAP	cosseno	84,49 (2,84)	84,66 (3,04)	84,19 (1,60)	82,86 (1,84)	84,03 (2,84)	74,71 (4,03)	81,07 (3,25)	83,45 (2,83)	83,70 (2,46)
	Euclidiana	84,01 (2,60)	83,95 (2,22)	82,69 (2,19)	82,68 (1,39)	83,81 (2,48)	80,28 (1,54)	83,90 (2,85)	**85,28** (2,41)	**85,28** (2,71)

conjunto de representação conseguem o melhor balanço de diversidade e performance? Isto é, definir o tamanho do conjunto de representação (Q). Por fim, a última pergunta: qual abordagem de seleção de características tem melhor performance com *Bagging*? Visando definir qual método será utilizado para comparação com o CoDiS.

5.3.1 Algoritmo de Seleção de Protótipos

A primeira avaliação é com relação ao algoritmo de seleção de protótipos utilizado no módulo Geração dos Conjuntos de Representação (Seção 3.3) do CoDiS. A Tabela 5.2 mostra os resultados de classificadores individuais e classificadores combinados, utilizando duas métricas (similaridade do cosseno e distância Euclidiana). A coluna "Algoritmo" da Tabela 5.2 é a combinação dos 7 classificadores individuais obtidos com algoritmos de seleção de protótipos (CNN, RNN, ENN, All-kNN, IB2, Drop3, ATISA2) e a coluna "Aleatória" é a combinação de 7 classificadores com seleção aleatória de protótipos.

Apesar do número de protótipos selecionados serem uma informação relevante na maioria dos casos, essa informação foi omitida, pois a ideia é comparar a performance (micro-F1) das combinações. Para uma comparação justa, o número de protótipos selecionados pela seleção aleatória é o mesmo de cada algoritmo de seleção de protótipos.

A diferença de performance entre a seleção aleatória e os algoritmos de seleção de protótipos é estatisticamente insignificante, isto é, além do custo computacional para execução dos algoritmos de seleção de protótipos, não existe incremento na performance. A baixa performance dos algoritmos de seleção de protótipos é, provavelmente, devido à baixa diversidade, pois os algoritmos funcionam de modo similar e todos usam a ideia de vizinhança. Assim, a seleção aleatória mesmo não possuindo nenhuma metodologia consegue atingir performance similar. Além do tempo de execução, a seleção aleatória tem outra vantagem sobre os algoritmos de

seleção de protótipos: não é necessário restringir a quantidade de classificadores à quantidade de algoritmos de seleção de protótipos. Portanto, nos próximos experimentos de combinação de classificadores, será utilizada apenas seleção aleatória de protótipos.

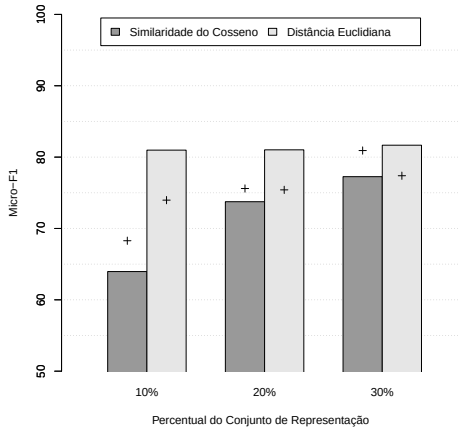
5.3.2 Métrica para Mudança de Representação

A segunda avaliação é com relação à métrica utilizada no módulo Mudança de Representação (Seção 3.4) do CoDiS. Dado que na primeira avaliação foi escolhida a seleção aleatória, para essa avaliação foram estabelecidos três valores para a quantidade de documentos do conjunto de representação: 10%, 20% ou 30% do conjunto de treinamento. A Figura 5.1 mostra a performance (micro-F1) das combinações e também do melhor classificador individual de cada combinação (*single best*).

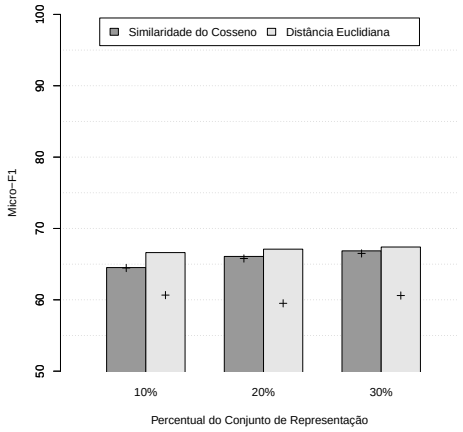
De modo geral, a distância Euclidiana é pouco utilizada para relacionar ou comparar documentos [155], pois a similaridade do cosseno atinge melhores resultados em classificadores baseados em distância, como *k*-NN [2]. De acordo com os gráficos da Figura 5.1, esse não é o caso no CoDiS, pois resultados do *single best* utilizando similaridade do cosseno são melhores ou próximos do resultado da combinação. Esse comportamento, do *single best* com relação à combinação, é observado em *ensembles* com baixa diversidade [125], um forte indicativo de que a combinação desses classificadores não é interessante. Em comparação, a distância Euclidiana atinge melhores resultados, se comparado com a similaridade do cosseno, e o *single best* é inferior à combinação em todos os casos. Esses resultados podem estar relacionados ao fato da similaridade do cosseno gerar características com baixa variância, enquanto que a distância Euclidiana gera características com alta variância. Por exemplo, na base de dados Irish, após a mudança de representação, o valor médio (e desvio padrão entre parênteses) das características é de 0,068 (0,003) com similaridade do cosseno e 29,782 (101,758) com distância Euclidiana. Portanto, nos próximos experimentos de combinação de classificadores será utilizada apenas distância Euclidiana.

5.3.3 Tamanho do Conjunto de Representação

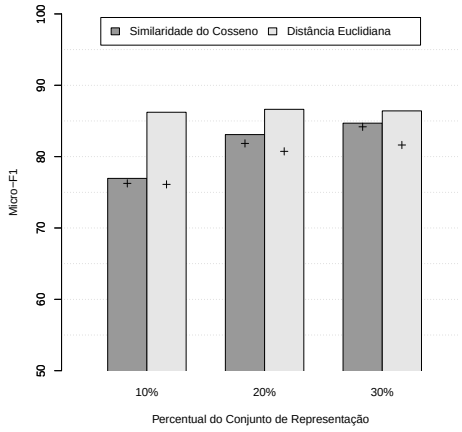
Na avaliação anterior (Seção 5.3.2) foram estabelecidos três valores para a quantidade de documentos do conjunto de representação: 10%, 20% ou 30% do conjunto de treinamento. Portanto, na terceira avaliação será definido qual o percentual do conjunto de treinamento será utilizado para compor os conjuntos de representação. Tendo em vista que uma combinação de classificadores não requer apenas performance, serão verificados tanto a performance como a diversidade para essa avaliação. A Figura 5.2 mostra um comparativo, utilizando o teste de *Wilcoxon*, de três valores do parâmetro Q (10%, 20% e 30%) com relação a duas medidas (diversidade e performance). A medida de diversidade utilizada foi a *Double-Fault Measure*, calculada pela Equação (2.30) apresentada na Seção 2.6.1, no qual quanto menor seu valor, mais diversidade; e como medida de performance foi utilizado a micro-F1, no qual quanto maior seu



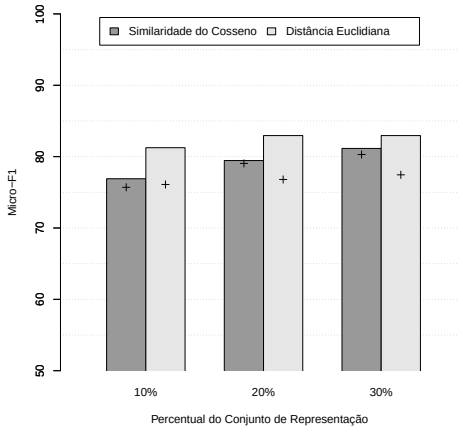
(a) CSTR



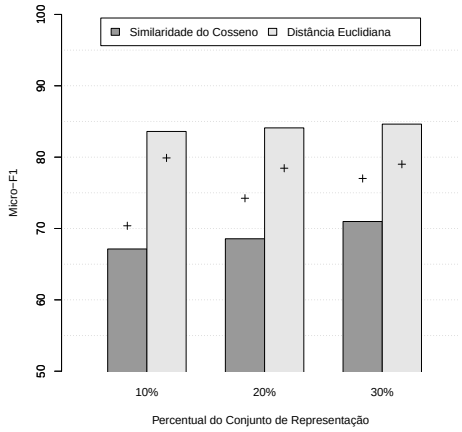
(b) Irish



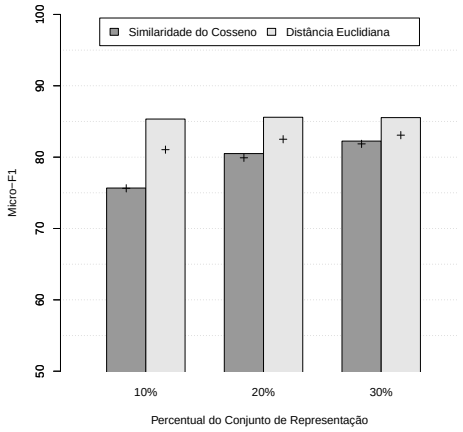
(c) Oh0



(d) Polarity



(e) Tr23



(f) WAP

Figura 5.1: Comparação da performance de duas métricas para Mudança de Representação: similaridade do cosseno e distância Euclidiana. As barras representam a micro-F1 dos *ensembles* com 100 classificadores ($L = 100$), e o símbolo de adição (+) são os resultados dos respectivos *single best*.

valor, mais preciso. O percentual de 10% foi igual ao 20% melhor, com relação à diversidade, sendo as melhores alternativas em 3 das 6 bases de dados, sendo 30% a pior com relação a diversidade. Entretanto, com relação à performance, 30% foi melhor em todas as bases, seguido dos 20% que conseguiu 2 das 6 bases com relação à performance. O parâmetro foi definido como 20%, pois é a opção mais balanceada em termos de performance e diversidade.

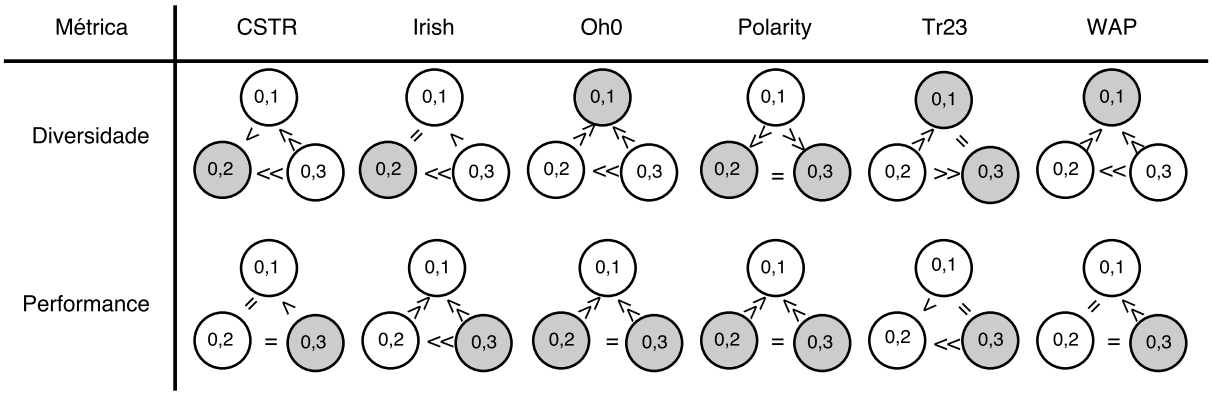


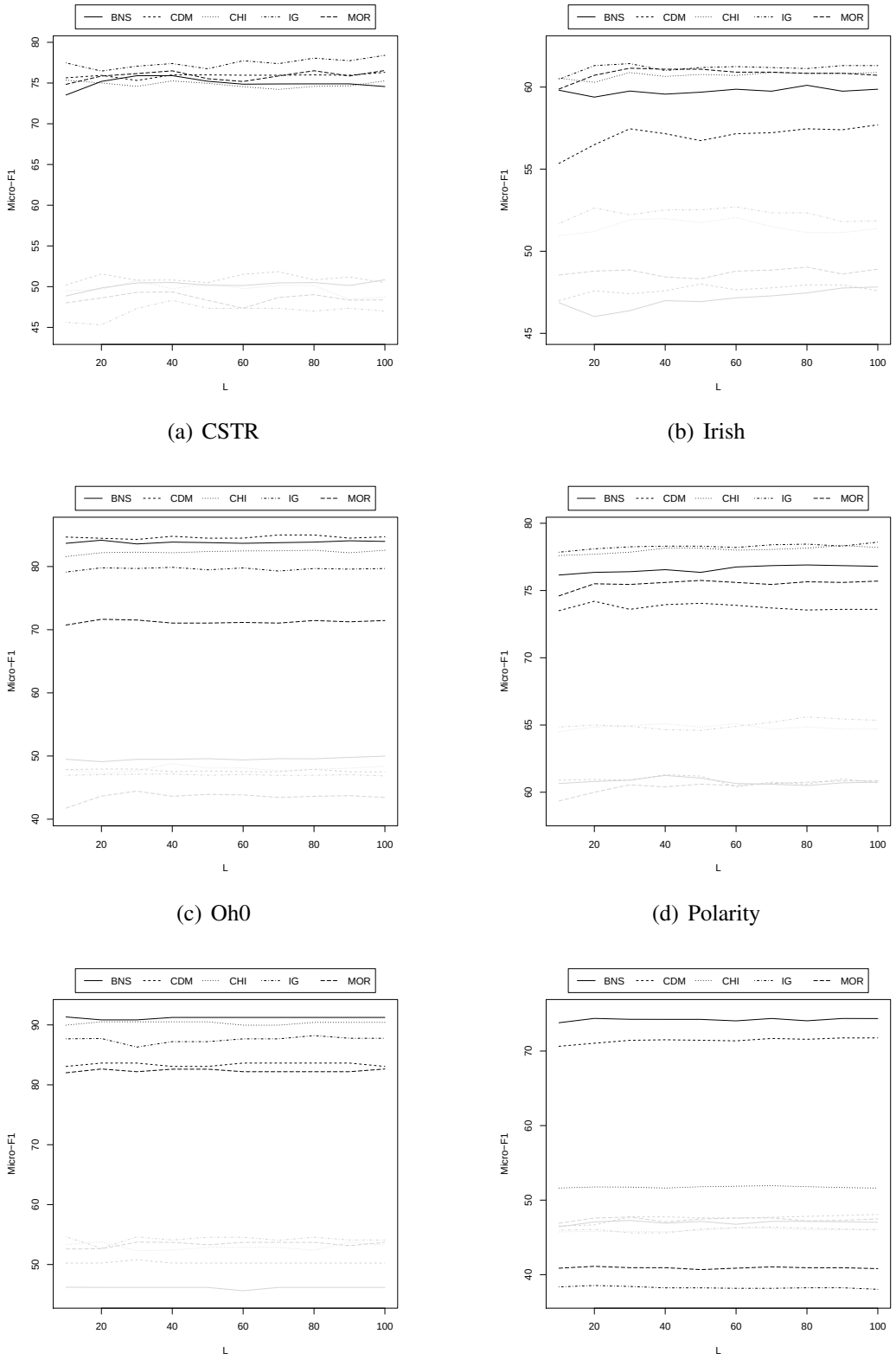
Figura 5.2: Comparação de três configurações de Q (10%, 20% e 30% do conjunto de treinamento) com relação à diversidade (*Double-Fault Measure*) e performance (micro-F1). Todas as comparações foram realizadas com o teste de *Wilcoxon* (simbologia na Seção 5.2). Em cinza as melhores opções para cada caso.

5.3.4 Outros métodos

Para finalizar, foram comparados os métodos ALOFT [10] e VR (descritos na Seção 2.2) para definir qual será utilizado nas comparações com o CoDiS. Foi utilizado *Bagging* para gerar o *pool* de classificadores, sendo a seleção de características aplicada como pré-processamento. A quantidade de características selecionadas pelo ALOFT é definida durante a execução do algoritmo, logo não pode ser controlada por um parâmetro, enquanto que o VR requer a definição de quantas características devem ser selecionadas. Portanto, foi definido para o VR selecionar mesma quantidade da seleção de protótipos do CoDiS (Q) de 20% do tamanho do conjunto de treinamento ($N \times 0,2$).

Nessa comparação foram utilizadas cinco FEFs: *Bi-Normal Separation* (BNS), *Chi-Squared Statistics* (CHI), *Class Discriminanting Measure* (CDM), *Information Gain* (IG) e *Multi-class Odds Ratio* (MOR). Todas apresentadas na Seção 2.2.2.

Os resultados são apresentados na Figura 5.3. ALOFT consegue melhores resultados que o VR em 28 dos 30 casos. Dado o o resultado superior do ALOFT, ele será utilizado para comparar com o CoDiS. Com relação à FEF, BNS e CDM foram escolhidas, pois foram as únicas a manter superioridade aos resultados do VR. Adicionalmente, as duas se comportam diferentemente: BNS varia bastante, dependendo da base de dados, capaz de atingir ótimos resultados ou os piores resultados com ALOFT [13], enquanto que CDM consegue resultados mais estáveis por ter a tendência de selecionar mais características com ALOFT [10, 13, 14].



(a) CSTR

(b) Irish

(c) Oh0

(d) Polarity

(e) Tr23

(f) WAP

Figura 5.3: Comparação de dois métodos de seleção de características: ALOFT (linhas pretas) e VR (linhas cinzas) considerando tamanho do *pool* (L) e FEFs (BNS, CDM, CHI, IG e MOR) em 6 bases de dados.

5.4 Experimentos - CoDiS

Baseado nos experimentos preliminares, alguns parâmetros dos módulos do CoDiS (Figura 3.1) foram definidos para os experimentos desta seção:

i) Seleção aleatória de protótipos no módulo Geração dos Conjuntos de Representação (Seção 3.3);

ii) Distância Euclidiana, definida pela Equação (3.1), é a métrica utilizada no módulo Mudança de Representação (Seção 3.4);

iii) Quantidade de documentos no módulo Geração dos Conjuntos de Representação foi definido por $Q = N \times 0,2$, no qual N é a quantidade de documentos do conjunto de treinamento.

Nos experimentos desta seção foram utilizados 10 valores para definição da quantidade de classificadores do *pool* ($L = \{10, 20, \ldots, 100\}$). Entretanto, a Tabela 5.3 mostra apenas os melhores resultados, e seu respectivo L, em quatro diferentes abordagens: CoDiS, *Bagging* utilizando ALOFT com BNS e CDM, e *Random Subspace*. Também foi comparado com SVM Individual e Random Forest (sempre com $L = 100$). É interessante verificar os resultados do *Random Subspace*, pois a quantidade de características em cada classificador é a mesma do CoDiS.

A Tabela 5.3 mostra os melhores resultados, e seus respectivos L, para cada um dos quatro métodos: CoDiS, *Bagging* usando ALOFT com BNS, *Bagging* usando ALOFT com CDM and *Random Subspace*. SVM Individual trata-se apenas de um classificador, por isso não exibe valor de L; e *Random Forest* utiliza sempre um *ensemble* com 100 classificadores ($L = 100$). A linha "Média" apresenta as médias dos valores da tabela de L, macro-F1, e micro-F1. As últimas duas linhas comparam os resultados (macro-F1 e micro-F1) do CoDiS com os demais métodos da literatura. No final da tabela, na linha "*Wilcoxon p-value*" é mostrado o *p-value* obtido pelo teste de *Wilcoxon* e, logo em seguida, na linha "*Holm p-value*" é mostrado o *p-value* corrigido pelo método de *Holm*. Note que existe um resultado do Bagging (ALOFT-BNS) bastante inferior na base WebACE, isto ocorre, pois foi um raro caso que o ALOFT com BNS selecionou apenas 2 características, quantidade extremamente insuficiente para uma boa representação dos dados.

Na Tabela 5.1 são mostrados os domínios de cada base de dados. Verificando esses domínios com os resultados da Tabela 5.3, não é possível estabelecer uma conexão entre alta ou baixa performance do CoDiS com relação a um domínio específico. Por exemplo, em *Web Pages* o CoDiS atinge tanto resultados inferiores em comparação com os demais métodos (DmComp, DmHealth, DmScience e DmSport) como resultados superiores aos demais métodos (WAP, WebAce, WebKB e WebKB4). Um ponto que é possível estabelecer alguma conexão é a alta performance obtida, geralmente, pelo CoDiS em bases com mais características do que bases com menos características, pois se considerarmos o mesmo exemplo das *Web Pages*, o subgrupo

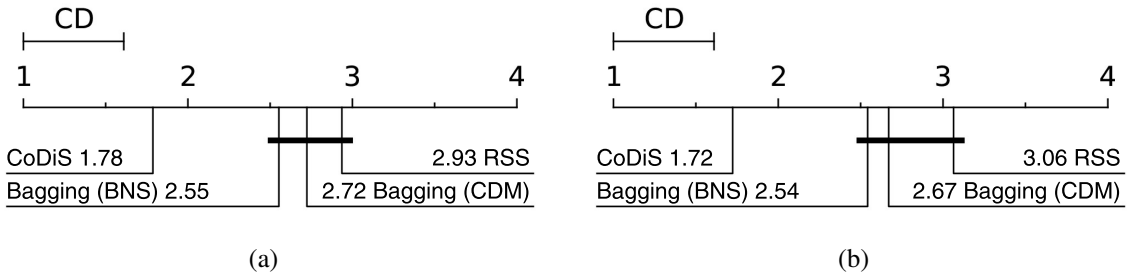
Tabela 5.3: Melhores resultados do CoDiS comparados com SVM individual, *Random Forest*, *Bagging* (usando ALOFT com duas FEFs) e *Random Subspace*. Em negrito os melhores resultados em cada base de dados. Desvio padrão de cada resultado entre parênteses. Nas últimas linhas estão, em ordem, média dos valores da tabela, *p-value* obtido pelo teste de *Wilcoxon* e *p-value* corrigido pelo método de *Holm*.

Bases	CoDiS			SVM Individual		*Random Forest*		*Bagging* (ALOFT-BNS)			*Bagging* (ALOFT-CDM)			*Random Subspace*		
de dados	*L*	macro-F1	micro-F1	macro-F1	micro-F1	macro-F1	micro-F1	*L*	macro-F1	micro-F1	*L*	macro-F1	micro-F1	*L*	macro-F1	micro-F1
20Ng	80	84,20 (0,66)	84,30 (0,63)	87,45 (0,53)	**87,45 (0,53)**	79,71 (0,87)	79,70 (0,86)	100	79,90 (1,00)	80,05 (0,98)	70	81,05 (1,00)	81,15 (1,01)	90	**88,00 (0,49)**	85,59 (0,50)
ACM	100	73,65 (1,95)	72,93 (2,10)	74,42 (2,77)	73,78 (2,82)	**87,30 (0,94)**	**86,74 (0,93)**	70	39,04 (3,52)	39,06 (3,77)	80	64,99 (2,10)	64,19 (2,21)	100	79,38 (1,70)	76,77 (1,81)
Classic3	20	98,75 (0,51)	98,77 (0,48)	**98,88 (0,66)**	**98,90 (0,66)**	96,32 (1,04)	96,45 (1,03)	80	98,18 (1,12)	98,20 (1,08)	60	98,43 (0,80)	98,46 (0,77)	90	97,17 (1,01)	97,15 (1,05)
Classic4	20	**96,64 (0,43)**	**96,57 (0,39)**	95,70 (0,90)	95,52 (0,88)	94,12 (0,54)	94,28 (0,51)	70	95,65 (0,81)	95,48 (0,86)	40	95,87 (0,65)	95,70 (0,64)	100	95,51 (1,02)	95,19 (1,08)
CSTR	90	**83,89 (8,40)**	**81,68 (7,66)**	80,56 (10,69)	79,21 (8,63)	75,60 (9,14)	77,67 (6,66)	40	72,95 (11,95)	75,91 (4,60)	100	74,76 (10,79)	76,31 (7,36)	10	40,90 (14,07)	49,93 (6,85)
DmComp	100	64,76 (1,95)	64,41 (2,05)	66,73 (1,15)	66,58 (1,22)	61,83 (0,83)	61,71 (0,85)	90	63,99 (2,06)	63,66 (2,03)	100	64,01 (1,86)	63,71 (1,85)	100	**71,00 (1,34)**	**71,32 (1,33)**
DmHealth	50	79,04 (1,81)	78,60 (1,85)	82,00 (1,71)	81,65 (1,72)	78,23 (1,36)	77,12 (1,50)	40	79,85 (1,23)	79,23 (1,25)	20	79,32 (1,21)	78,66 (1,25)	100	**84,12 (1,07)**	**83,77 (0,98)**
DmScience	100	66,64 (1,98)	66,38 (1,92)	68,69 (1,40)	68,43 (1,39)	63,11 (1,88)	62,52 (1,95)	90	67,73 (1,38)	67,28 (1,40)	100	66,29 (2,11)	65,95 (2,23)	90	**70,34 (1,36)**	**69,12 (1,43)**
DmSport	40	84,38 (0,75)	84,19 (0,82)	87,02 (0,82)	86,79 (0,81)	86,34 (0,96)	85,84 (0,99)	40	84,88 (0,88)	84,51 (0,96)	90	84,92 (0,98)	84,64 (1,06)	100	**89,52 (0,66)**	**89,24 (0,64)**
Enron20	90	65,53 (1,33)	67,26 (1,07)	65,16 (1,24)	66,55 (1,01)	65,99 (1,02)	**67,80 (0,99)**	70	65,13 (0,69)	67,20 (0,54)	90	65,65 (1,16)	67,69 (1,02)	100	**66,45 (0,80)**	67,49 (0,63)
FBIS	60	78,64 (4,48)	84,45 (1,92)	81,23 (3,85)	83,79 (2,19)	81,01 (3,26)	85,98 (2,41)	70	72,40 (4,13)	80,45 (2,67)	80	66,25 (3,45)	77,30 (2,30)	70	**83,96 (2,78)**	**86,70 (1,20)**
Hitech	80	**72,16 (3,50)**	**75,41 (2,67)**	69,38 (3,24)	72,06 (2,51)	68,11 (3,21)	72,62 (1,76)	50	68,81 (3,92)	71,74 (2,63)	100	61,92 (3,99)	68,88 (2,99)	60	53,35 (5,18)	61,50 (3,27)
Irish	60	**65,68 (4,28)**	**67,28 (4,09)**	63,17 (4,33)	64,09 (4,22)	61,10 (4,49)	63,07 (4,27)	80	58,77 (3,77)	60,11 (3,70)	100	57,50 (4,11)	57,70 (3,92)	80	59,56 (4,98)	60,06 (1,43)
La1s	100	**88,02 (1,70)**	**89,64 (1,81)**	86,02 (1,72)	87,95 (1,57)	86,55 (1,76)	88,01 (1,49)	20	62,36 (3,12)	67,07 (1,72)	80	78,17 (2,14)	81,87 (1,55)	80	75,54 (3,10)	76,80 (2,56)
La2s	80	**89,43 (2,27)**	**91,10 (1,75)**	87,74 (1,75)	89,50 (1,30)	86,38 (2,49)	88,17 (2,00)	50	61,55 (2,66)	65,10 (1,95)	70	80,55 (2,57)	83,71 (2,29)	60	78,77 (1,67)	80,49 (1,58)
LATimes	70	**81,73 (1,41)**	**84,38 (1,51)**	76,71 (1,19)	78,94 (1,11)	77,75 (1,94)	80,57 (1,97)	20	61,84 (2,74)	66,71 (2,28)	90	76,78 (1,72)	79,93 (1,69)	90	74,08 (2,14)	73,91 (2,23)
MDS	100	**81,75 (1,17)**	**81,74 (1,16)**	76,97 (1,18)	76,95 (1,17)	79,00 (1,43)	78,97 (1,42)	80	81,00 (1,55)	80,96 (1,57)	90	80,57 (1,55)	80,54 (1,57)	70	79,30 (1,05)	78,99 (1,09)
New 3	90	86,63 (1,24)	87,26 (1,08)	86,73 (1,22)	86,22 (1,14)	**87,63 (1,17)**	**87,44 (1,02)**	80	54,81 (1,72)	57,80 (1,35)	100	67,27 (1,11)	71,43 (0,96)	100	85,49 (1,56)	84,49 (1,64)
NFS	100	78,30 (0,76)	79,42 (0,89)	82,47 (1,02)	82,66 (0,90)	74,75 (1,14)	75,45 (1,12)	40	77,26 (1,05)	78,01 (1,13)	100	76,68 (1,27)	77,63 (1,49)	100	**84,49 (0,71)**	**84,20 (0,65)**
Oh0	100	85,04 (3,84)	86,63 (3,95)	85,64 (2,77)	86,55 (2,55)	**88,00 (3,25)**	**87,07 (3,40)**	20	83,59 (3,04)	84,17 (2,41)	70	85,08 (3,09)	85,00 (2,61)	70	74,82 (4,12)	77,86 (2,84)
Oh5	80	85,94 (4,79)	86,17 (4,51)	86,55 (3,41)	86,73 (3,45)	**88,88 (3,80)**	**89,06 (3,64)**	70	85,56 (5,38)	86,10 (4,63)	30	83,00 (2,99)	83,32 (2,43)	90	71,72 (5,28)	70,90 (4,25)
Oh10	70	74,65 (3,12)	77,91 (2,95)	74,91 (3,78)	77,59 (3,77)	**84,00 (3,58)**	**84,34 (2,90)**	20	76,38 (4,08)	77,46 (3,42)	20	74,64 (4,61)	75,89 (4,79)	70	69,61 (4,12)	72,77 (4,21)
Oh15	40	77,62 (4,75)	78,49 (4,94)	80,56 (3,95)	81,00 (4,05)	**84,40 (3,61)**	**83,70 (4,23)**	30	78,87 (5,18)	77,87 (6,04)	70	78,69 (3,79)	78,02 (4,91)	100	66,35 (7,60)	65,86 (4,07)
Ohscal	90	**78,31 (1,08)**	**79,22 (0,79)**	76,47 (0,97)	77,39 (0,91)	78,11 (1,89)	78,81 (1,74)	100	77,26 (1,42)	78,06 (1,36)	90	76,38 (1,64)	77,34 (1,48)	100	77,53 (1,39)	78,38 (1,31)
Ohsumed	50	31,72 (1,20)	31,30 (1,23)	34,28 (1,11)	33,83 (1,18)	34,76 (1,51)	34,32 (1,42)	100	34,34 (0,93)	33,83 (1,02)	60	33,42 (1,12)	32,91 (1,03)	100	**35,96 (1,05)**	**36,03 (1,14)**
Opinosis	100	61,07 (2,16)	60,97 (1,15)	**62,92 (1,76)**	61,63 (1,34)	59,67 (1,86)	58,92 (1,10)	10	62,35 (2,19)	**62,70 (1,30)**	30	60,91 (2,20)	61,28 (1,62)	70	62,59 (1,87)	62,50 (1,42)
Re0	50	64,81 (8,82)	82,03 (2,29)	**78,03 (7,05)**	83,65 (3,24)	75,07 (8,34)	**84,31 (2,19)**	40	51,35 (7,92)	76,40 (3,46)	40	71,96 (6,24)	83,03 (2,56)	100	47,46 (10,02)	67,86 (4,27)
Re1	90	64,60 (4,12)	81,90 (2,13)	74,43 (6,57)	83,99 (2,45)	74,96 (4,04)	86,30 (1,40)	70	**78,19 (4,69)**	**86,94 (2,04)**	40	56,46 (5,70)	78,97 (2,25)	100	40,55 (5,11)	61,83 (2,50)
Re8	60	89,98 (2,28)	96,32 (0,59)	**92,11 (2,54)**	**96,65 (0,60)**	88,23 (3,98)	95,44 (0,71)	60	90,58 (2,23)	95,41 (0,42)	80	89,97 (2,41)	95,69 (0,86)	100	89,09 (2,36)	95,28 (0,52)
Reuters-21578	10	15,63 (0,89)	57,44 (1,51)	**18,78 (1,46)**	57,27 (1,18)	15,93 (0,96)	56,56 (1,62)	10	13,55 (0,94)	**59,36 (1,08)**	20	11,47 (0,83)	58,12 (1,32)	100	17,74 (1,21)	59,02 (1,05)
Polarity	80	**83,13 (1,69)**	**83,05 (1,62)**	81,56 (2,36)	81,50 (2,37)	79,38 (3,09)	79,30 (3,07)	80	76,94 (2,10)	76,90 (2,11)	20	74,32 (2,04)	74,20 (2,02)	50	77,07 (2,26)	76,10 (2,50)
Reviews	90	**93,96 (2,42)**	**95,89 (1,52)**	92,43 (3,03)	94,91 (1,48)	91,07 (2,46)	94,17 (1,17)	30	90,08 (2,54)	93,66 (1,27)	80	89,43 (3,00)	93,66 (1,57)	80	82,36 (1,25)	86,31 (1,27)
SpamA	100	98,79 (0,49)	99,08 (0,37)	98,83 (0,57)	99,11 (0,43)	**98,86 (0,37)**	**99,13 (0,28)**	100	96,68 (0,84)	97,47 (0,63)	80	97,62 (0,56)	98,18 (0,42)	10	74,89 (1,55)	81,68 (0,98)
Spam3000	40	98,58 (0,40)	98,58 (0,40)	98,43 (0,62)	98,43 (0,62)	**98,75 (0,41)**	**98,75 (0,41)**	10	97,76 (0,61)	97,76 (0,60)	50	98,52 (0,45)	98,51 (0,45)	10	86,53 (1,58)	85,44 (1,78)
SsW	20	93,04 (3,34)	93,06 (3,29)	92,73 (4,58)	92,77 (5,11)	**95,48 (4,64)**	**95,46 (4,80)**	60	94,14 (3,40)	94,00 (3,55)	10	93,07 (4,20)	92,74 (4,53)	90	39,38 (9,64)	51,05 (6,38)
TDT2T30	90	95,87 (1,01)	97,19 (0,47)	**96,81 (0,47)**	**97,46 (0,31)**	96,58 (0,58)	97,39 (0,42)	50	93,91 (1,17)	96,28 (0,66)	100	92,93 (1,72)	96,10 (0,62)	50	84,14 (1,71)	86,23 (0,97)
Tr11	50	72,07 (8,12)	88,01 (4,73)	73,91 (9,34)	87,20 (5,70)	78,80 (4,45)	**91,82 (2,88)**	20	73,99 (6,82)	86,84 (5,27)	80	**78,98 (9,62)**	88,75 (6,01)	20	25,78 (4,72)	49,83 (4,49)
Tr12	60	82,48 (8,01)	85,58 (3,52)	83,21 (7,82)	87,23 (5,84)	**87,65 (7,90)**	**91,44 (5,91)**	70	81,47 (9,54)	87,97 (5,55)	70	86,21 (6,64)	87,72 (4,94)	10	15,02 (7,71)	33,22 (5,61)
Tr21	70	65,37 (11,35)	92,10 (3,65)	65,53 (16,88)	89,92 (4,86)	54,44 (9,67)	89,61 (3,00)	20	**76,12 (14,94)**	**92,52 (3,74)**	30	47,50 (14,03)	81,48 (7,38)	10	13,58 (0,41)	68,82 (3,53)
Tr23	70	66,50 (15,20)	84,13 (8,85)	74,00 (12,43)	84,19 (4,60)	**82,50 (12,17)**	**94,46 (5,40)**	10	80,66 (10,09)	91,33 (4,39)	20	60,90 (12,06)	83,63 (7,45)	10	13,99 (5,58)	46,40 (4,17)
Tr31	40	81,24 (1,52)	96,77 (1,11)	83,36 (1,73)	97,74 (1,49)	**83,71 (1,67)**	**98,40 (0,88)**	30	81,91 (2,48)	97,53 (1,33)	70	81,65 (2,09)	97,19 (1,30)	100	59,22 (8,71)	85,26 (4,56)
Tr41	100	84,86 (5,09)	94,61 (1,96)	**89,04 (5,47)**	**96,02 (1,98)**	86,37 (5,90)	95,83 (1,36)	60	72,14 (6,67)	89,78 (2,49)	50	82,41 (6,30)	93,68 (2,19)	60	63,85 (7,21)	78,92 (5,97)
Tr45	60	85,80 (7,10)	91,89 (3,20)	85,97 (6,45)	91,59 (2,67)	**87,81 (4,42)**	**94,69 (1,98)**	100	83,12 (5,66)	90,10 (3,09)	90	78,37 (3,47)	89,60 (2,28)	70	64,04 (3,07)	74,90 (2,63)
WAP	70	68,30 (4,59)	85,67 (2,83)	**74,08 (5,35)**	**85,69 (2,57)**	61,86 (2,90)	81,00 (1,96)	20	58,16 (4,73)	74,38 (3,09)	90	50,69 (4,72)	71,76 (3,94)	100	51,20 (3,28)	70,21 (3,62)
WebACE	100	67,44 (2,61)	89,05 (1,69)	**74,14 (5,36)**	**89,62 (2,03)**	61,27 (2,67)	86,77 (1,05)	20	7,10 (1,28)	44,70 (1,07)	50	49,81 (2,76)	76,74 (1,95)	90	52,79 (3,39)	80,70 (1,80)
WebKB	50	**72,32 (2,02)**	**80,38 (1,50)**	65,13 (2,39)	72,40 (1,25)	68,01 (2,38)	79,04 (1,77)	60	63,17 (2,60)	73,72 (1,50)	90	56,53 (3,08)	69,79 (0,88)	30	36,72 (1,49)	55,09 (1,10)
WebKB4	70	**90,34 (1,32)**	**91,05 (1,21)**	87,02 (1,70)	88,26 (1,73)	87,64 (1,82)	88,76 (1,41)	20	88,27 (1,46)	89,36 (1,30)	40	85,62 (2,11)	87,78 (1,66)	30	60,93 (2,43)	60,58 (2,10)
Média	70,85	77,64 (15,71)	82,98 (12,81)	78,67 (14,75)	82,70 (12,54)	77,93 (15,84)	83,08 (13,04)	52,55	72,20 (19,52)	78,10 (15,49)	66,59	73,35 (17,20)	79,71 (13,07)	72,55	64,08 (22,64)	72,16 (14,82)
Wilcoxon p-value	-	n,a,	n,a,	9,713e-01	3,355e-01	3,471e-01	2,874e-01	-	8,303e-04	2,257e-04	-	1,297e-05	1,57e-06	-	1,803e-06	4,284e-07
Holm p-value	-	n,a,	n,a,	1,000e+00	1,000e+00	1,000e+00	1,000e+00	-	4,151e-03	1,354e-03	-	9,079e-05	1,413e-05	-	1,442e-05	4,284e-06

que CoDiS é inferior possui menos de 6 mil características, enquanto que o subgrupo que CoDiS é superior possui mais de 8 mil características.

O método proposto, CoDiS, é considerado similar ao SVM Individual e ao *Random Forest*. Entretanto, é considerado significativamente superior do que os métodos similares (*Bagging* e *Random Subspace*) devido ao baixo *p-value* com *Wilcoxon*. Para verificar se o CoDiS é realmente superior, aos métodos de *Bagging* e *Random Subspace*, foi realizado um teste de *Friedman* verificando o *rank* médio de todos os métodos nas 47 bases de dados. A Figura 5.4 mostra dois diagramas CD [162], um para a macro-F1 e outro para a micro-F1. CoDiS alcançou o menor *rank* médio em ambos os casos, sendo considerado estatisticamente superior aos demais métodos.

Ao contrário do CoDiS, o *Random Subspace* depende muito da quantidade de características, e como a quantidade de características depende da quantidade de documentos do conjunto de treinamento (N), o comportamento do *Random Subspace* tem uma forte relação com N. A Tabela 5.1 mostra os tamanhos da base, como os experimentos foram realizados utilizando 10-*folds*, N é 90% da quantidade de documentos da base. Comparando o tamanho das bases de dados (Tabela 5.1) com os resultados do *Random Subspace* (Tabela 5.3) é possível observar que bases de dados com $N < 2000$ são os piores resultados do *Random Subspace*. Já os melhores resultados são atingidos quando a quantidade de documentos ($N \times 0,9$) é próxima à quanti-



(a) (b)

Figura 5.4: Diagramas CD para macro-F1 (a) e micro-F1 (b) comparando todos os métodos usados nos experimentos. *Rank* médio é apresentado para cada método e CD = 0.6102 em ambos os diagramas.

dade de características originais (V), principalmente quando a base possui muitos documentos ($N > 6000$).

Ambas as abordagens com *Bagging*, tiveram seu comportamento dependente da seleção de características aplicada antes da geração do *pool*. Isto é, ALOFT com BNS atinge bons resultados, mas é bastante dependente da base de dados. Já o ALOFT com CDM atinge resultados medianos sem depender tanto da base de dados.

Os resultados com SVM Individual demonstram que o classificador é comparável à combinações, tendo resultados superiores com unanimidade (ambas as métricas) em 6 bases de dados. Enquanto que o *Random Forest* mostrou-se uma boa combinação, sendo superior com unanimidade em 13 bases de dados, empatando com o CoDiS que também mostrou-se superior com unanimidade em 13 bases de dados.

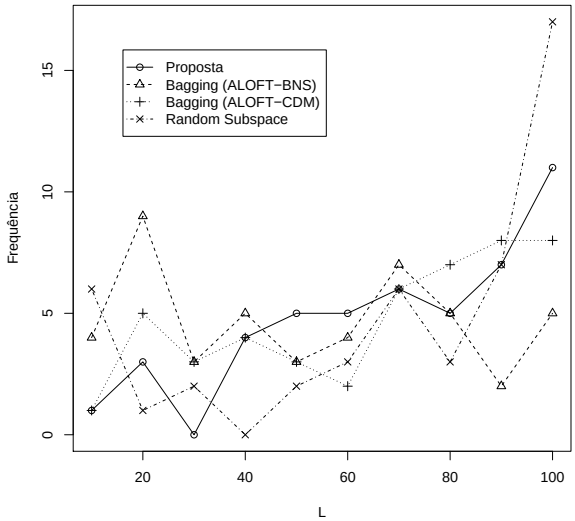
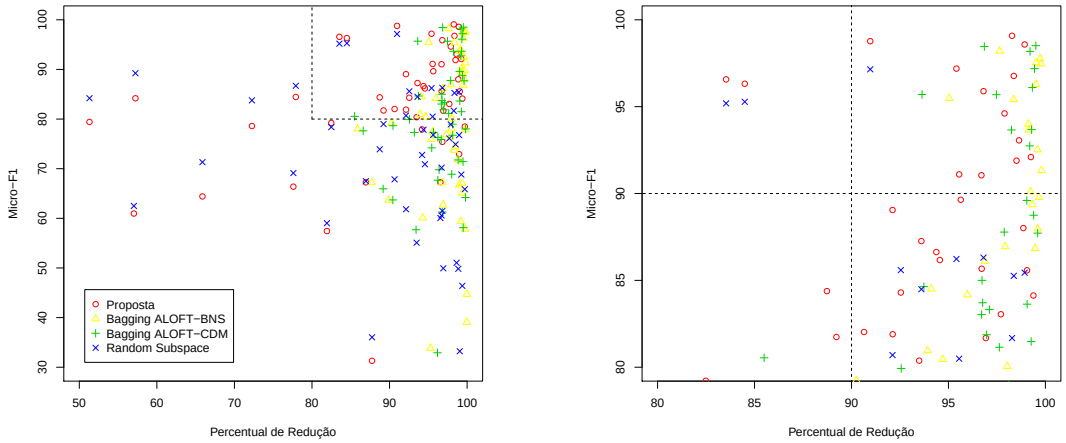


Figura 5.5: Gráfico exibindo a frequência dos diferentes valores de L dentre os melhores resultados (Tabela 5.3).

Não foi encontrada nenhuma relação do melhor L para os resultados na Tabela 5.3. Entretanto, a Figura 5.5 facilita a observação de que tanto o CoDiS quanto o *Random Subspace*

atingem os melhores resultados com grandes *ensembles* ($L = 100$). O SVM Individual e *Random Forest* foram ignorados nessa Figura, pois não possuem variação em L.



(a) Todos os casos. (b) Aproximação do canto superior-direito.

Figura 5.6: Redução do número de características em relação à performance (micro-F1). Cada ponto é um resultado da Tabela 5.3.

A Figura 5.6 explora a relação da performace com a redução das características dos métodos focados nesse aspecto: CoDiS, *Bagging* com ALOFT e *Random Subspace*. Cada ponto do gráfico representa o resultado de um determinado método em uma base de dados, assim, existem 47 pontos por método, totalizando 188 pontos para os quatro métodos (47×4). É observado que o *Random Subspace* comporta-se melhor quando a redução é inferior a 80%. O *Bagging* com ALOFT seleciona as características automaticamente, sua redução é superior à 85% em todos os casos, independente da FEF utilizada. A Figura 5.6(b) mostra apenas os experimentos com redução superior à 80% e micro-F1 superior à 80, pois muitos resultados ficaram localizados no canto superior-direito da 5.6(a). Então, na Figura 5.6(b) observa-se uma avaliação mais pontual. Considere apenas os pontos com micro-F1 superior à 90 e redução superior à 90%, quadrante superior-direito da 5.6(b). Nesta região, o CoDiS aparece 12 vezes, *Bagging* usando ALOFT-BNS também aparece 12 vezes, *Bagging* usando ALOFT-CDM com 9 aparições e *Random Subspace* com apenas 1 aparição. Ampliando um pouco a região para micro-F1 acima de 80 e redução acima de 80%, toda a 5.6(b), a superioridade do CoDiS fica mais visível com 31 aparições, seguido pelo *Bagging* usando ALOFT-CDM com 25 aparições, depois *Bagging* usando ALOFT-BNS com 23 e, novamente com a menor contagem, *Random Subspace* com 12 aparições. Essa contagem demonstra que CoDiS possui o melhor benefício com relação à redução e performance com relação aos outros três métodos apresentados, pois consegue reduzir a quantidade de características sem degradar tanto a performance.

Entretanto, do ponto de vista do tempo de execução do CoDiS sofre algumas dificuldades. A Tabela 5.4 mostra o tempo de execução médio de cada método com relação ao tempo de processamento antes do classificador (TPre), seria o tempo gasto pelo método para transformação

Tabela 5.4: Tempo médio, em segundos, das execuções antes do classificador (TPre), do classificador no treinamento (TTReino) e do classificador no teste (TTeste) do CoDiS, *Bagging* (usando ALOFT) e *Random Subspace*. Última linha é um comparativo das médias com relação ao mais rápido (*Bagging*).

	CoDiS			*Bagging* (ALOFT)			*Random Subspace*		
Bases	TPre	TTreino	TTeste	TPre	TTreino	TTeste	TPre	TTreino	TTeste
20Ng	115,662	930,731	129,251	0,118	6,693	1,00	0,943	26,919	1,837
ACM	24,544	8,972	2,011	0,162	0,290	0,096	0,096	44,752	0,164
Classic3	4,395	3,148	0,326	0,031	0,196	0,028	0,065	0,483	0,037
Classic4	13,515	26,650	2,589	0,030	0,576	0,053	0,145	2,113	0,105
CSTR	0,044	0,090	0,021	0,021	0,010	0,020	0,015	0,012	0,017
DmComp	11,660	197,368	26,212	0,023	1,783	0,261	0,224	6,196	0,447
DmHealth	5,553	49,861	6,713	0,039	0,804	0,099	0,116	2,306	0,154
DmScience	4,614	63,482	7,056	0,028	0,692	0,099	0,106	2,287	0,137
DmSport	27,870	291,000	49,736	0,034	3,574	0,695	0,479	8,385	0,909
Enron20	41,745	580,609	53,776	0,083	2,866	0,514	0,459	11,724	0,911
FBIS	3,239	4,471	0,483	0,039	0,134	0,030	0,078	1,335	0,168
Hitech	2,734	13,083	0,432	0,030	0,088	0,015	0,044	1,107	0,039
Irish	1,230	26,764	0,200	0,033	0,042	0,018	0,037	0,465	0,026
La1s	5,071	8,604	0,677	0,041	0,150	0,024	0,060	0,970	0,051
La2s	4,720	7,898	0,574	0,043	0,143	0,027	0,078	1,080	0,064
LATimes	10,454	63,275	4,519	0,034	0,559	0,062	0,126	4,601	0,111
MDS	21,975	486,903	6,583	0,046	0,628	0,062	0,332	49,049	0,189
New 3	61,138	110,495	20,846	0,139	2,170	0,563	0,409	10,415	1,646
NFS	18,319	196,595	27,556	0,022	1,690	0,277	0,271	4,883	0,396
Oh0	0,323	0,517	0,090	0,024	0,024	0,017	0,020	0,095	0,0271
Oh5	0,287	0,433	0,081	0,021	0,023	0,016	0,016	0,089	0,016
Oh10	0,368	0,620	0,098	0,020	0,035	0,015	0,020	0,132	0,022
Oh15	0,287	0,508	0,084	0,021	0,024	0,017	0,018	0,098	0,018
Ohscal	38,608	228,454	30,185	0,053	2,129	0,208	0,378	18,700	0,993
Ohsumed	25,478	300,387	30,973	0,049	1,710	0,289	0,270	15,982	0,702
Opinosis	4,918	38,919	10,074	0,019	0,767	0,317	0,114	1,793	0,405
Re0	0,643	1,197	0,157	0,021	0,041	0,015	0,029	0,250	0,029
Re1	0,815	1,867	0,231	0,048	0,074	0,036	0,032	0,355	0,053
Re8	14,006	30,079	3,157	0,063	0,530	0,079	0,168	2,171	0,149
Reuters-21578	53,295	858,744	87,020	0,062	2,796	2,847	0,588	29,676	3,961
Polarity	2,777	47,176	0,213	0,041	0,051	0,033	0,045	0,434	0,027
Reviews	9,974	7,300	0,685	0,061	0,218	0,031	0,078	1,364	0,069
SpamAss	33,098	148,985	2,765	0,071	0,647	0,046	0,227	7,635	0,081
Spam3000	22,131	48,471	0,726	0,105	0,369	0,044	0,115	1,797	0,048
SsW	0,059	0,051	0,017	0,022	0,010	0,024	0,023	0,010	0,018
TDT2T30	38,522	51,458	6,652	0,101	1,917	0,378	0,283	4,496	0,545
Tr11	0,148	0,153	0,043	0,020	0,024	0,017	0,015	0,032	0,022
Tr12	0,094	0,471	0,021	0,021	0,012	0,019	0,016	0,028	0,019
Tr21	0,134	0,922	0,020	0,023	0,010	0,020	0,015	0,026	0,017
Tr23	0,062	0,864	0,014	0,022	0,011	0,019	0,019	0,011	0,021
Tr31	0,654	0,692	0,061	0,030	0,036	0,017	0,019	0,114	0,017
Tr41	0,495	0,318	0,060	0,023	0,023	0,019	0,021	0,082	0,014
Tr45	0,386	0,754	0,067	0,031	0,025	0,018	0,021	0,097	0,016
WAP	1,278	1,316	0,201	0,031	0,059	0,023	0,035	0,230	0,032
WebACE	4,256	9,932	1,371	0,031	0,208	0,060	0,057	0,755	0,070
WebKB	26,584	552,298	11,578	0,062	0,758	0,080	0,246	28,210	0,312
WebKB4	6,336	34,170	1,152	0,038	0,231	0,030	0,081	2,153	0,049
Média	14,138	115,682	11,220	0,045	0,762	0,184	0,150	6,295	0,322
Comparação	314,17x	151,81x	60,97x	x	x	x	3,33x	8,26x	1,75x

dos dados ou gerar os novos conjuntos baseados nas amostragens aleatórias e salvá-los em disco, esse tempo inclui tanto o tempo antes do treinamento quanto antes do teste; tempo de treinamento (TTreino), seria o tempo necessário para treinar o classificador; tempo de teste (TTeste), seria o tempo necessário para testar o classificador com todo o conjunto de teste, dado que o SVM utilizado não foi implementação próprio, tanto o TTreino como o TTest inclui todos os procedimentos de leitura e escrita de arquivos. Note que todos os tempos estão em segundos e indicam o tempo de apenas um dos classificadores do *ensemble*. Observando a linha final, que exibe a média para cada tempo, é verificado que o CoDiS exige muito mais tempo do que os demais, em especial no treinamento. Sendo o *Bagging* (ALOFT) o mais rápido, pois o ALOFT é uma seleção bastante rápida e reduz a quantidade de característica mais do que o CoDiS e o *Random Subspace*, o que agiliza o TTreino e TTeste. Talvez fosse esperado que o CoDiS tivesse um tempo de treinamento e de teste similar aos demais métodos, pois a quantidade de características é a mesma utilizada no *Random Subspace*. Entretanto, como esclarecido anteriormente, nos TTreino e TTeste estão incluídos a leitura de arquivos, provavelmente uma das partes mais custosas do processo. Tendo em vista que a construção do arquivo para leitura é feita de modo esparso, isto é, características de valor zero não fazem parte do arquivo; e que a *Dissimilarity Representation* reduz à quase 0% a esparsidade dos dados, a leitura possui 900 à 9000 vezes mais dados (Figura 1.1(b)). Portanto, é perfeitamente esperado esse aumento no TTreino e TTeste, mas isso não indica que o classificador em si demora tanto quanto os tempos indicam. Deste modo, muito provavelmente, o TPre deve ser a maior diferença de tempo para o CoDiS, pois a Representação por Dissimilaridade é de fato custosa, tendo em vista que calcula a distância Euclidiana uma quantidade de vezes igual ao número de documentos de um subconjunto de treinamento vezes a quantidade de documentos de um conjunto de representação, em cada classificador. O SVM Individual e *Random Forest* foram ignorados nessa Tabela, pois o SVM Individual trata-se apenas de um classificador e não foi possível medir o tempo de apenas um classificador no *Random Forest*.

5.5 Experimentos - Variante do CoDiS

Nos experimentos preliminares foram verificadas várias configurações para o CoDiS. A análise foi primordialmente voltada para o comportamento das configurações em combinação de classificadores, no qual vários parâmetros foram avaliados. Entretanto, se o CoDiS utilizasse apenas um classificador, em vez de uma combinação de classificadores, algumas configurações descartadas poderiam atingir uma boa performance. Observe na Tabela 5.2 que classificadores individuais utilizando similaridade do cosseno podem conseguir resultados superiores à combinação, dependendo da seleção de protótipos. A mesma conclusão pode ser extraída da Figura 5.1, com os classificadores *single best* utilizando similaridade do cosseno sendo superiores à combinação, em alguns casos apenas, pois a seleção é aleatória. Portanto, apesar da similaridade do cosseno e seleção de protótipos não gerarem classificadores interessantes para combinação, essa

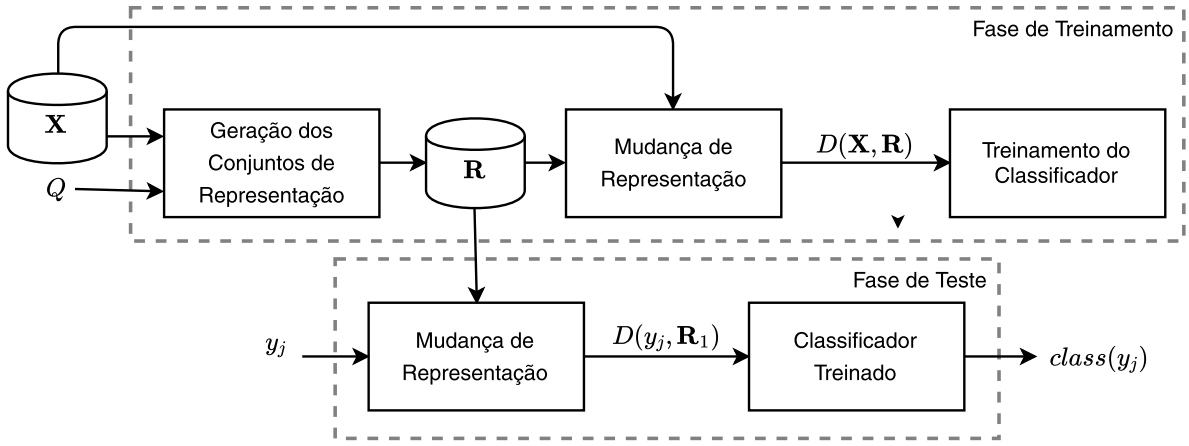


Figura 5.7: Fase de treinamento e teste do CR. $\mathbf{X}$ é o conjunto de treinamento, $\mathbf{R}$ é o conjunto de representação, Q é o número de documentos no conjunto de representação, $D(\mathbf{X},\mathbf{R})$ é a matriz de similaridade obtidas usando $\mathbf{X}$ e $\mathbf{R}$, $y_j \in \mathbf{Y}$ é um documento do conjunto de teste, $D(y_j,\mathbf{R})$ é o vetor de dissimilaridade obtido usando y_j e $\mathbf{R}$ e $class(y_j)$ é a resposta do classificador.

configuração pode ser capaz de gerar classificadores individuais poderosos.

Nesta seção, são apresentados e discutidos os experimentos com essa configuração: uma arquitetura baseada no CoDiS (Capítulo 3) sem Combinação de Classificadores, usando similaridade do cosseno como métrica para o módulo Mudança de Representação (Seção 3.4) e algoritmos de seleção de protótipos no módulo Geração dos Conjuntos de Representação (Seção 3.3). Essa configuração será chamada, de agora em diante, como *Cosine Representation* (CR). Note que apesar de ser baseada no CoDiS, o CR não utiliza uma arquitetura identica ao CoDiS, a Figura 5.7 apresenta a arquitetura do CR.

Nesses experimentos são utilizadas vinte base de dados: CSTR, Irish, La1s, La2s, Oh0, Oh5, Oh10, Oh15, Re0, Re1, Polarity, SsW, Tr11, Tr12, Tr21, Tr23, Tr31, Tr41, Tr45 e WAP. A Tabela 5.1 mostra as características das bases de dados. Foram utilizados cinco algoritmos de seleção de protótipos: CNN [116], ENN [118], All-kNN [119], DROP3 [121] e ATISA2 [122]. Os algoritmos IB2 e RNN, utilizados nos experimentos preliminares, foram removidos por serem considerados variações do CNN e terem atingido resultados, em sua maioria, inferiores (Tabela 5.2).

A Tabela 5.5 mostra os resultados do CR e do BoW (SVM sem nenhum processamento diferenciado). Baseado no teste de *Wilcoxon*, foi inferido a seguinte situação para a macro-F1: semPS $>$ BoW $\sim$ ENN $\gg$ All-kNN $\sim$ CNN $\gg$ ATISA2 $\gg$ DROP3; e para a micro-F1: semPS $\gg$ ENN $>$ BoW $\sim$ All-kNN $\sim$ CNN $\gg$ ATISA2 $\gg$ DROP3. Indicando que o CR é superior ao BoW sem seleção de protótipos (semPS) e, neste caso, ainda possui uma redução média nas características de 82,3%. Quanto mais agressiva for a seleção de protótipos, pior é a performance do CR. A Tabela 5.6 exibe os *p-value* de todos os testes estatísticos.

Tendo em vista a boa performance do *Cosine Representation*, experimentos foram realizados para verificar se a mesma arquitetura usando uma métrica diferente (Distância Euclidiana) é igualmente válida. A Tabela 5.7 compara o CR sem seleção de protótipos (melhor resultados

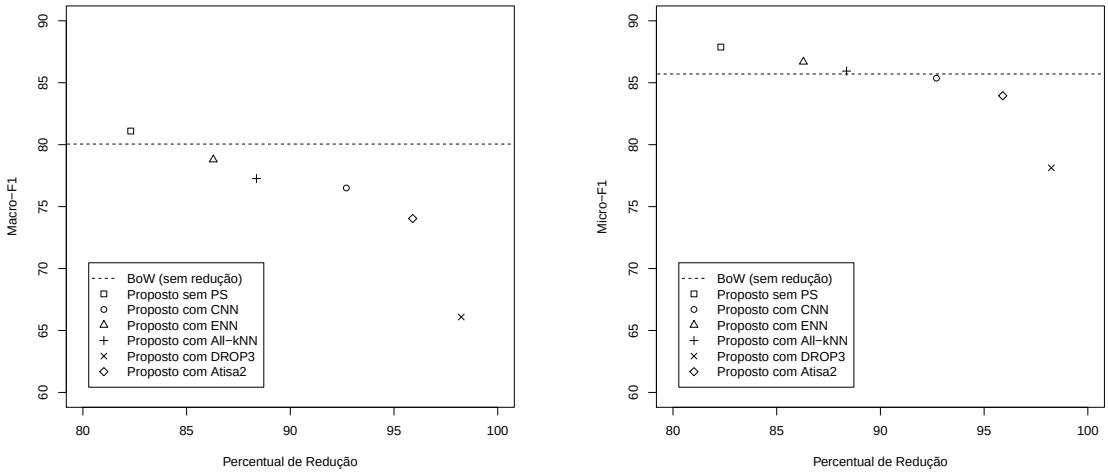
Tabela 5.5: Resultados de BoW e CR, sem combinação, em vinte bases de dados. Nomes em negrito indicam o método que atingiu a melhor performance (valor absoluto) para cada base de dados. A coluna de Redução indica o quanto o vetor de características foi reduzido com relação ao BoW. CR sem PS é quando o conjunto de representação é todo o conjunto de treinamento, isto é, sem algoritmo de seleção de protótipos.

Bases	Método	Redução	macro-F1	micro-F1	Bases	Método	Redução	macro-F1	micro-F1
	BoW	0,00%	80,56 (10,69)	79,21 (8,63)		BoW	0,00%	81,56 (2,36)	81,50 (2,37)
	CR sem PS	84,41%	86,63 (4,91)	84,94 (5,78)		**CR sem PS**	88,53%	83,71 (2,73)	83,65 (2,71)
	CR com CNN	93,44%	80,72 (8,72)	79,92 (5,45)		CR com CNN	94,04%	82,41 (2,70)	82,35 (2,70)
CSTR	CR com ENN	87,65%	83,35 (8,80)	82,61 (6,52)	Polarity	CR com ENN	92,23%	83,40 (3,18)	83,35 (3,15)
	CR com All-kNN	89,56%	83,03 (8,68)	83,33 (6,60)		CR com All-kNN	94,67%	80,94 (2,62)	80,90 (2,59)
	CR com DROP3	97,85%	68,28 (12,14)	71,32 (8,48)		CR com DROP3	99,13%	66,77 (4,20)	66,50 (4,15)
	CR com ATISA2	96,19%	75,94 (10,52)	77,69 (8,63)		CR com ATISA2	95,67%	81,44 (2,81)	81,40 (2,82)
	BoW	0,00%	63,17 (4,33)	64,09 (4,22)		BoW	0,00%	92,73 (4,58)	92,77 (5,11)
	CR sem PS	82,75%	68,79 (3,27)	70,18 (3,14)		CR sem PS	93,09%	95,67 (3,10)	95,77 (2,96)
	CR com CNN	89,36%	66,95 (3,71)	68,55 (3,52)		CR com CNN	98,68%	93,77 (2,47)	93,67 (2,76)
Irish	CR com ENN	89,84%	67,03 (3,73)	68,74 (3,92)	SsW	**CR com ENN**	93,64%	96,08 (2,05)	96,08 (2,06)
	CR com All-kNN	93,26%	64,30 (2,76)	66,14 (2,74)		CR com All-kNN	93,79%	95,27 (1,65)	95,17 (1,61)
	CR com DROP3	98,81%	59,06 (3,74)	61,63 (3,82)		CR com DROP3	99,51%	93,97 (2,61)	93,40 (3,04)
	CR com ATISA2	95,81%	64,96 (3,62)	66,75 (3,52)		CR com ATISA2	98,78%	92,55 (4,80)	92,72 (4,75)
	BoW	0,00%	86,02 (1,72)	87,95 (1,57)		BoW	0,00%	73,91 (9,34)	87,20 (5,70)
	CR sem PS	78,15%	88,30 (1,69)	89,98 (1,68)		**CR sem PS**	94,21%	72,21 (4,41)	88,19 (3,13)
	CR com CNN	91,98%	88,04 (2,08)	89,66 (2,08)		CR com CNN	98,01%	67,73 (7,21)	86,48 (4,26)
La1s	CR com ENN	82,30%	88,11 (1,28)	89,73 (1,54)	Tr11	CR com ENN	95,10%	70,31 (5,67)	86,26 (2,31)
	CR com All-kNN	84,66%	87,74 (1,53)	89,39 (1,46)		CR com All-kNN	95,77%	63,10 (5,16)	84,79 (3,29)
	CR com DROP3	98,21%	84,63 (2,37)	86,48 (2,24)		CR com DROP3	99,43%	49,07 (8,57)	78,09 (4,56)
	CR com ATISA2	94,71%	87,42 (1,78)	89,23 (1,94)		CR com ATISA2	98,70%	61,71 (5,07)	84,02 (4,10)
	BoW	0,00%	87,74 (1,75)	89,50 (1,30)		BoW	0,00%	83,21 (7,82)	87,23 (5,84)
	CR sem PS	77,74%	89,88 (2,08)	91,58 (1,45)		**CR sem PS**	95,15%	83,84 (6,02)	86,50 (5,42)
	CR com CNN	92,48%	89,60 (2,44)	91,48 (1,71)		CR com CNN	98,03%	77,77 (7,80)	83,17 (3,94)
La2s	**CR com ENN**	81,50%	90,20 (2,02)	91,71 (1,48)	Tr12	CR com ENN	96,03%	78,31 (8,24)	83,92 (5,44)
	CR com All-kNN	83,86%	89,59 (1,95)	91,19 (1,49)		CR com All-kNN	96,71%	78,77 (8,01)	84,83 (4,98)
	CR com DROP3	98,07%	85,64 (2,48)	88,24 (1,89)		CR com DROP3	99,30%	61,46 (9,20)	74,60 (6,58)
	CR com ATISA2	94,86%	88,79 (2,53)	90,74 (1,89)		CR com ATISA2	98,84%	71,21 (6,63)	80,52 (3,85)
	BoW	0,00%	85,64 (2,77)	86,55 (2,55)		BoW	0,00%	65,53 (16,88)	89,92 (4,86)
	CR sem PS	71,64%	87,41 (3,53)	88,35 (3,53)		**CR sem PS**	96,18%	68,26 (15,89)	93,51 (3,51)
	CR com CNN	87,52%	84,88 (3,76)	86,35 (3,27)		CR com CNN	98,99%	50,36 (6,15)	88,43 (3,43)
Oh0	CR com ENN	78,03%	85,61 (3,97)	86,57 (3,68)	Tr21	CR com ENN	96,68%	54,62 (7,91)	90,26 (4,31)
	CR com All-kNN	81,14%	84,09 (4,58)	85,26 (4,10)		CR com All-kNN	96,94%	55,77 (8,03)	90,29 (4,75)
	CR com DROP3	96,82%	76,22 (5,46)	78,09 (4,24)		CR com DROP3	99,81%	30,27 (9,19)	76,93 (6,90)
	CR com ATISA2	93,12%	81,59 (3,15)	82,95 (3,01)		CR com ATISA2	99,31%	47,34 (5,54)	87,89 (3,76)
	BoW	0,00%	86,55 (3,41)	86,73 (3,45)		BoW	0,00%	74,00 (12,43)	84,19 (4,60)
	CR sem PS	72,58%	87,84 (3,27)	87,92 (3,11)		**CR sem PS**	96,85%	74,87 (17,74)	88,88 (7,34)
	CR com CNN	88,07%	84,74 (4,35)	84,88 (3,92)		CR com CNN	99,05%	50,57 (15,32)	75,74 (5,92)
Oh5	CR com ENN	79,03%	86,36 (3,44)	86,61 (3,22)	Tr23	CR com ENN	97,28%	67,08 (17,32)	86,09 (8,92)
	CR com All-kNN	82,66%	86,05 (3,04)	86,15 (3,29)		CR com All-kNN	97,62%	60,77 (17,59)	83,20 (9,44)
	CR com DROP3	96,39%	80,68 (6,28)	80,55 (6,02)		CR com DROP3	99,82%	28,00 (8,48)	55,81 (10,18)
	CR com ATISA2	92,62%	85,55 (3,77)	85,41 (3,89)		CR com ATISA2	99,36%	45,39 (15,85)	72,89 (8,77)
	BoW	0,00%	74,91 (3,78)	77,59 (3,77)		BoW	0,00%	83,36 (1,73)	97,74 (1,49)
	CR sem PS	70,82%	77,71 (4,59)	80,74 (4,39)		**CR sem PS**	91,76%	83,61 (2,01)	98,17 (1,26)
	CR com CNN	84,22%	77,61 (2,98)	80,42 (3,71)		CR com CNN	98,50%	80,28 (2,93)	95,48 (2,08)
Oh10	CR com ENN	80,48%	77,30 (3,51)	79,95 (3,79)	Tr31	CR com ENN	92,21%	82,19 (3,08)	97,74 (1,48)
	CR com All-kNN	83,98%	77,49 (3,10)	80,05 (3,72)		CR com All-kNN	92,62%	82,24 (3,05)	97,74 (1,48)
	CR com DROP3	97,11%	69,40 (4,09)	73,12 (4,42)		CR com DROP3	99,28%	77,93 (3,08)	93,33 (2,26)
	CR com ATISA2	93,72%	76,20 (3,44)	78,32 (3,57)		CR com ATISA2	98,67%	80,69 (3,27)	95,81 (2,40)
	BoW	0,00%	80,56 (3,95)	81,00 (4,05)		**BoW**	0,00%	89,04 (5,47)	96,02 (1,98)
	CR sem PS	73,52%	80,62 (3,80)	81,61 (3,91)		CR sem PS	89,40%	86,21 (5,20)	95,31 (1,90)
	CR com CNN	86,45%	78,86 (3,64)	79,94 (3,90)		CR com CNN	97,56%	84,19 (5,97)	94,82 (2,82)
Oh15	CR com ENN	81,65%	78,75 (3,33)	79,58 (3,23)	Tr41	CR com ENN	90,26%	86,94 (5,59)	95,76 (2,23)
	CR com All-kNN	85,08%	78,23 (3,57)	78,81 (3,36)		CR com All-kNN	90,68%	87,09 (5,28)	95,86 (2,32)
	CR com DROP3	96,84%	73,34 (3,32)	73,63 (3,90)		CR com DROP3	98,85%	78,07 (6,96)	91,74 (3,24)
	CR com ATISA2	92,98%	76,44 (3,08)	77,22 (3,48)		CR com ATISA2	98,08%	83,97 (4,52)	94,72 (2,52)
	BoW	0,00%	78,03 (7,05)	83,65 (3,24)		BoW	0,00%	85,97 (6,45)	91,59 (2,67)
	CR sem PS	53,11%	74,83 (9,24)	85,89 (2,58)		**CR sem PS**	92,48%	88,56 (4,69)	93,06 (1,72)
	CR com CNN	82,44%	72,59 (7,44)	85,04 (2,36)		CR com CNN	97,90%	82,62 (5,59)	90,74 (2,16)
Re0	CR com ENN	62,18%	74,59 (7,54)	85,83 (2,46)	Tr45	CR com ENN	93,36%	86,86 (5,33)	92,31 (2,07)
	CR com All-kNN	68,53%	69,75 (7,79)	85,37 (2,58)		CR com All-kNN	93,82%	85,43 (5,10)	91,35 (2,21)
	CR com DROP3	96,40%	57,42 (8,29)	80,59 (3,99)		CR com DROP3	99,13%	75,44 (5,06)	86,53 (3,42)
	CR com ATISA2	89,65%	71,79 (5,79)	85,25 (2,22)		CR com ATISA2	98,45%	80,51 (5,51)	89,87 (2,71)
	BoW	0,00%	74,43 (6,57)	83,99 (2,45)		**BoW**	0,00%	74,08 (5,35)	85,69 (2,57)
	CR sem PS	60,33%	73,67 (7,12)	86,90 (2,00)		**CR sem PS**	83,41%	69,38 (4,20)	86,44 (2,23)
	CR com CNN	84,69%	71,20 (6,30)	85,77 (1,80)		CR com CNN	92,61%	65,25 (4,20)	84,49 (2,84)
Re1	CR com ENN	68,19%	72,46 (5,33)	86,49 (1,32)	WAP	CR com ENN	88,09%	66,34 (3,72)	84,19 (1,60)
	CR com All-kNN	72,06%	71,43 (6,44)	86,12 (2,15)		CR com All-kNN	89,95%	64,24 (3,65)	82,86 (1,84)
	CR com DROP3	96,02%	54,06 (4,46)	77,42 (3,03)		CR com DROP3	98,05%	52,06 (4,67)	74,71 (4,03)
	CR com ATISA2	91,75%	67,32 (4,08)	84,52 (2,07)		CR com ATISA2	96,74%	59,94 (3,84)	81,07 (3,25)

Tabela 5.6: *P-value* do teste estatístico *Wilcoxon signed rank* realizados sobre os resultados apresentados na Tabela 5.5. O primeiro valor de cada célula é o *p-value* com relação a macro-F1, enquanto que o segundo é o *p-value* com relação a micro-F1. Todos os *p-values* foram corrigidos de acordo com o método de *Holm*.

Teste	BoW	CR sem PS	CR com CNN	CR com ENN	CR com All-kNN	CR com DROP3	CR com ATISA2
BoW > ?	-	1,000e+00 / 1,000e+00	1,115e-04 / 1,000e+00	1,000e+00 / 1,000e+00	4,412e-03 / 1,000e+00	1,341e-29 / 9,802e-27	5,103e-13 / 9,200e-04
CR sem PS > ?	6,028e-02 / 3,445e-10	-	3,999e-17 / 4,121e-17	3,267e-08 / 4,814e-11	3,999e-18 / 6,281e-17	1,508e-32 / 5,330e-32	7,030e-25 / 3,460e-24
CR com CNN > ?	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00	-	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00	4,646e-30 / 1,691e-29	1,630e-09 / 6,927e-08
CR com ENN > ?	1,000e+00 / 1,477e-02	1,000e+00 / 1,000e+00	3,936e-06 / 1,212e-05	-	2,484e-09 / 4,525e-06	1,312e-32 / 8,080e-32	2,622e-21 / 8,017e-20
CR com All-kNN > ?	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00	-	7,631e-32 / 4,121e-31	3,070e-10 / 9,305e-12
CR com DROP3 > ?	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00	-	1,000e+00 / 1,000e+00
CR com ATISA2 > ?	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00	2,399e-26 / 2,522e-26	-

de acordo com as Tabelas 5.5 e 5.6) com a mesma arquitetura utilizando Distância Euclidiana também sem seleção de protótipos. A Distância Euclidiana obteve resultados inferiores na maioria das bases de dados. Portanto, a presente arquitetura sem combinação é mais interessante com Similaridade do Cosseno.



(a) (b)

Figura 5.8: Relação da redução das características com as métricas de performance: macro-F1 (a) e micro-F1 (b). Cada ponto representa a média daquele método usando os valores da Tabela 5.5.

A Figura 5.8 mostra médias, de todas as bases, com relação à redução e performance. CNN, DROP3 e ATISA2 foram os algoritmos com redução mais agressiva. A redução ocorre mesmo sem a seleção de protótipos, pois na mudança de representação, a quantidade máxima de características é igual à quantidade de documentos no conjunto de treinamento. Como todas as bases de dados, utilizadas nesses experimentos, possuem um número de documentos inferior à quantidade de características, o número de características é menor que ao utilizar BoW.

A melhor configuração (Figura 5.8) depende da relação desejada entre a performance e a redução. Caso seja necessário mais performance, CR funciona melhor sem algoritmos de seleção de protótipos, isto é, usando todo o conjunto de treinamento igual ao conjunto de representação ($\mathbf{X} = \mathbf{R}$); mas caso mais redução seja necessária, ATISA2 provavelmente é a melhor configuração, pois consegue uma redução média superior à 95% sem degradar tanto os resultados, como ocorre com o DROP3. De modo geral, quanto maior a redução do número de características, menor a

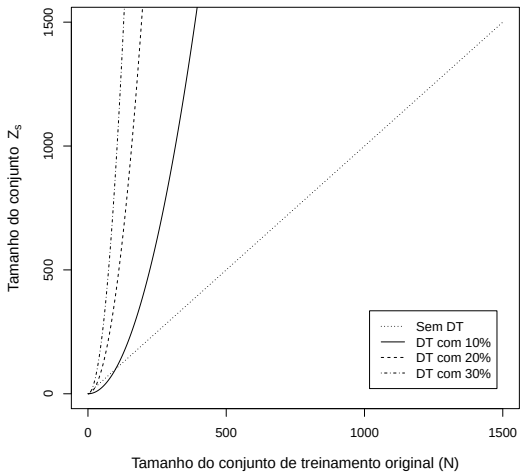


Figura 5.9: Número de documentos após a *Dichotomy Transformation* (DT) usando três diferentes tamanhos (10%, 20% e 30% do conjunto de treinamento) para os conjuntos de entrada ($\mathbf{X}_s$ and $\mathbf{R}_s$). A linha pontilhada (documentos sem DT) serve apenas como base de comparação.

performance.

5.6 Experimentos - CoDiT

Apesar da Tabela 5.1 mostrar todas bases utilizadas nos experimentos, essa informação é replicada na Tabela 5.8 para facilitar a visualização, pois são apresentados os parâmetros de entrada do CoDiT nesta mesma Tabela. Esses parâmetros são: média do número de documentos nos subconjuntos de treinamento ($\bar{N}'$) e média do número de documentos dos conjuntos de representação ($\bar{Q}$). Ambos N' e Q são apresentados como média, pois os valores mudam de acordo com os diferentes *folds*. A quantidade de bases de dados foi reduzida nesses experimentos por conta do tempo de execução, mesmo bases de dados pequenas como a Irish demoravam 10 horas para cada classificador, como é necessária a execução de 100 classificadores em 10 *folds* (mil execuções), tornou-se inviável a execução da maioria das bases de dados.

A Figura 5.9 mostra o número de documentos após a *Dichotomy Transformation* (DT), que é baseado nos dois conjuntos de entrada: subconjunto de treinamento e conjunto de representação. Dado o aumento significativo do número de documentos após a DT, tanto o tamanho dos subconjuntos de treinamento (N'), como o tamanho dos conjuntos de representação (Q) foram definidos como 10% do tamanho do conjunto de treinamento original ($0.1 \times N$). Entretanto, o conjunto de representação deve ter o mesmo número de documentos por classe, em algumas bases isto não é possível de ser obtido se $Q = 0,1 \times N$. Por exemplo, a base de dados Tr31 (Tabela 5.1) possui 7 classes. Em um dos *folds* usados nos experimentos, o número de documentos do conjunto de treinamento é de 834. Assim, o valor de Q deveria ser 83 ($834 \times 0,1 = 83,4$), mas não pode ser utilizado, pois a classe minoritária possui apenas 2 documentos. Neste caso, Q é igual à quantidade de documentos da classe minoritária multiplicado por C ($2 \times 7 = 14$).

Tabela 5.7: Resultados do CR (extraídos da Tabela 5.5) em comparação com a arquitetura do CR usando Distância Euclidiana (indicado por DE abaixo) em vez da Similaridade do Cosseno. Usado apenas a configuração sem seleção de protótipos (sem PS), pois foi a configuração que alcançou melhor performance. Melhores resultados para cada base de dados em negrito.

Bases	Método	macro-F1	micro-F1	Bases	Método	macro-F1	micro-F1
CSTR	CR sem PS	**86,63 (4,91)**	**84,94 (5,78)**	Polarity	CR sem PS	**83,71 (2,73)**	**83,65 (2,71)**
	DE sem PS	80,15 (11,43)	79,33 (8,65)		DE sem PS	83,22 (2,77)	83,15 (2,73)
Irish	CR sem PS	**68,79 (3,27)**	**70,18 (3,14)**	SsW	CR sem PS	**95,67 (3,10)**	**95,77 (2,96)**
	DE sem PS	67,21 (3,78)	68,43 (3,54)		DE sem PS	93,63 (4,61)	93,66 (4,90)
La1s	CR sem PS	**88,30 (1,69)**	**89,98 (1,68)**	Tr11	CR sem PS	72,21 (4,41)	**88,19 (3,13)**
	DE sem PS	87,48 (2,19)	89,01 (2,06)		DE sem PS	**73,72 (9,75)**	87,19 (5,22)
La2s	CR sem PS	**89,88 (2,08)**	**91,58 (1,45)**	Tr12	CR sem PS	**83,84 (6,02)**	**86,50 (5,42)**
	DE sem PS	88,48 (2,19)	90,38 (1,65)		DE sem PS	80,62 (6,67)	84,64 (3,27)
Oh0	CR sem PS	**87,41 (3,53)**	**88,35 (3,53)**	Tr21	CR sem PS	**68,26 (15,89)**	**93,51 (3,51)**
	DE sem PS	84,55 (3,68)	85,33 (3,83)		DE sem PS	56,23 (11,76)	89,31 (3,26)
Oh5	CR sem PS	**87,84 (3,27)**	**87,92 (3,11)**	Tr23	CR sem PS	**74,87 (17,74)**	**88,88 (7,34)**
	DE sem PS	86,55 (3,69)	86,41 (3,81)		DE sem PS	70,46 (14,43)	85,84 (5,72)
Oh10	CR sem PS	**77,71 (4,59)**	**80,74 (4,39)**	Tr31	CR sem PS	**83,61 (2,01)**	**98,17 (1,26)**
	DE sem PS	75,45 (3,88)	78,61 (4,08)		DE sem PS	81,52 (1,86)	96,66 (1,95)
Oh15	CR sem PS	**80,62 (3,80)**	**81,61 (3,91)**	Tr41	CR sem PS	**86,21 (5,20)**	**95,31 (1,90)**
	DE sem PS	79,65 (3,11)	80,23 (3,85)		DE sem PS	85,37 (5,41)	94,29 (1,25)
Re0	CR sem PS	**74,83 (9,24)**	**85,89 (2,58)**	Tr45	CR sem PS	**88,56 (4,69)**	**93,06 (1,72)**
	DE sem PS	70,26 (6,72)	82,71 (2,73)		DE sem PS	84,39 (7,20)	90,58 (3,52)
Re1	CR sem PS	**73,67 (7,12)**	**86,90 (2,00)**	WAP	CR sem PS	**69,38 (4,20)**	**86,44 (2,23)**
	DE sem PS	67,52 (5,90)	83,22 (2,54)		DE sem PS	68,28 (3,62)	84,65 (2,27)

Tabela 5.8: Informações das bases de dados e parâmetros de entrada do CoDiT

id	Bases	Categorias	Documentos	Características	$\bar{N}'$	$\bar{Q}$	$\bar{Q}/C$
1	CSTR	4	299	1726	26.4	24	6
2	Oh0	10	1003	3183	89.9	89	8.9
3	Oh5	10	918	3013	82.1	80	8
4	Oh10	10	1050	3239	94	90	9
5	Oh15	10	913	3103	81.6	80	8
6	Re0	13	1504	2887	135.1	128.7	9.9
7	Re1	25	1657	3759	148.6	127.5	5.1
8	SsW	4	334	4340	29.5	28	7
9	Tr11	9	414	6430	36.9	36	4
10	Tr12	8	313	5805	27.7	24	3
11	Tr31	7	927	10129	82.9	12.6	1.8
12	Tr41	10	878	7455	78.6	70	7
13	Tr45	10	690	8262	61.6	60	6
14	WAP	20	1560	8461	140	90	4.5

Tabela 5.9: Macro-F1 e micro-F1 dos melhores resultados para as duas funções do CoDiT e seu respectivo tamanho do *ensemble* (*L*). Em negrito estão marcados os melhores resultados em cada base de dados e entre parênteses os desvios padrão.

	CoDiT com $dt()$			CoDiT com $dts()$		
Bases	L	macro-F1	micro-F1	L	macro-F1	micro-F1
CSTR	90	80,92 (6,82)	**80,73 (5,34)**	70	**81,78 (8,69)**	80,29 (5,50)
Oh0	80	92,26 (2,58)	**92,45 (2,52)**	90	**92,47 (1,44)**	92,40 (1,51)
Oh5	100	**91,72 (3,51)**	**91,91 (3,20)**	100	89,47 (3,81)	89,74 (3,62)
Oh10	100	**84,68 (4,07)**	**86,43 (3,17)**	60	83,64 (2,70)	84,58 (2,31)
Oh15	70	**88,60 (2,70)**	**88,44 (3,03)**	80	86,69 (3,26)	86,01 (3,09)
Re0	90	**77,21 (8,70)**	**78,71 (2,52)**	100	69,37 (8,29)	75,22 (2,97)
Re1	50	**79,40 (3,33)**	**87,52 (1,04)**	80	74,13 (4,24)	83,98 (1,45)
SsW	30	**94,97 (3,68)**	**94,88 (3,60)**	40	92,69 (3,11)	92,46 (3,29)
Tr11	90	**81,05 (9,38)**	**89,18 (5,04)**	60	73,98 (9,27)	88,10 (4,51)
Tr12	90	**88,09 (6,34)**	**89,15 (5,00)**	50	82,69 (8,00)	85,23 (7,02)
Tr31	90	**82,96 (1,46)**	97,08 (1,55)	60	80,76 (4,76)	**97,13 (2,02)**
Tr41	90	90,55 (6,95)	96,25 (1,97)	60	**91,91 (5,86)**	**96,86 (2,15)**
Tr45	50	**93,30 (5,41)**	94,60 (2,25)	50	90,09 (6,04)	**95,55 (2,20)**
WAP	90	**72,06 (5,63)**	**85,42 (2,54)**	70	66,90 (6,63)	82,40 (2,04)
Média	79,28	85,55 (6,85)	89,48 (5,52)	69,28	82,61 (8,67)	87,85 (6,53)

Antes de comparar CoDiT com outras metodologias, foram comparadas as duas funções utilizadas na *Dichotomy Transformation* do CoDiT: $dt(\cdot,\cdot)$ apresentada na Equação (2.26), função com valor absoluto da diferença; e $dts(\cdot,\cdot)$ apresentada na Equação (4.7), função proposta inspirada na similaridade do cosseno. A Tabela 5.9 mostra essa comparação claramente, com todas as bases de dados utilizadas nesses experimentos. Apesar da motivação por trás do uso de uma medida inspirada na similaridade do cosseno (Seção 4.5), foi realizado o teste *Wilcoxon* para comparar as duas versões do CoDiT. Tendo em vista a superioridade do CoDiT com $dt()$, o teste utilizado foi para verificar se ele é de fato superior ou estatisticamente similar. O *p-value* obtido para macro-F1 foi de: $5,501e-07$ e o *p-value* obtido para micro-F1 foi de: $1,271e-07$, isto é, em ambas as medidas a diferença foi significante com um nível de confiança superior à 99%. Apesar da função $dts()$ ter vários indicativos para uma boa performance em dados esparsos, existe uma perda de informação em comparação com a $dt()$, pois quando qualquer uma das características é zero, o valor da outra é ignorado e toda a característica tem o valor igual a zero. Por conta dessa perda de informação, $dts()$ tem um funcionamento similar à uma seleção de características, o que acarreta numa redução média de $14,6$ vezes no tempo do $dts()$ com relação ao $dt()$ durante o treinamento e teste. Apesar da diferença exibida no teste estatístico, $dts()$ não possui um desempenho tão inferior se comparado em valores absolutos, tendo $dt()$ um ganho médio de apenas 3,5%. Portanto, se a aplicação necessitar de um menor tempo de execução, a aplicação do $dts()$ é válida. Entretanto, como o enfoque deste trabalho está em obter o sistema mais robusto e de melhor performance, deste momento em diante, as demais análises realizadas nesta seção serão apenas sobre o CoDiT utilizando a Equação (2.26).

Tabela 5.10: Resultados da macro-F1 e micro-F1 para cada método. Em negrito são os melhores resultados de cada base de dados. As últimas linhas apresentam a média de cada coluna de resultados, o *p-value* do teste de *Wilcoxon* comparando o método CoDiT com os demais e o *p-value* corrigido pelo método de *Holm*.

	SVM individual		*Random Forest*		*Bagging*			*Random Subspace*			CoDiT		
Bases	macro-F1	micro-F1	macro-F1	micro-F1	*L*	macro-F1	micro-F1	*L*	macro-F1	micro-F1	*L*	macro-F1	micro-F1
CSTR	80,56	79,21	75,60	77,67	90	79,09	77,90	50	**84,39**	**80,92**	90	80,92	80,73
Oh0	85,64	86,55	88,00	87,07	50	86,38	87,41	70	87,37	88,42	80	**92,26**	**92,45**
Oh5	86,55	86,73	88,88	89,06	60	86,78	86,83	90	87,09	86,9	100	**91,72**	**91,91**
Oh10	74,91	77,59	84,00	84,34	30	77,69	79,97	100	80,26	81,82	100	**84,68**	**86,43**
Oh15	80,56	81,00	84,40	83,70	60	80,76	81,01	100	82,67	82,67	70	**88,60**	**88,44**
Re0	78,03	83,65	75,07	**84,31**	100	**79,18**	83,97	100	71,13	79,90	90	77,21	78,71
Re1	74,43	83,99	74,96	86,30	10	73,64	84,54	80	67,90	81,38	50	**79,40**	**87,52**
SsW	92,73	92,77	**95,48**	**95,46**	10	93,39	93,05	100	92,73	92,98	30	94,97	94,88
Tr11	73,91	87,20	78,80	**91,82**	10	74,23	88,12	10	77,50	90,32	90	**81,05**	89,18
Tr12	83,21	87,23	87,65	**91,44**	30	84,30	88,16	100	84,12	86,96	90	**88,09**	89,15
Tr31	83,36	97,74	**83,71**	**98,40**	60	83,38	97,74	80	83,46	97,95	20	82,96	97,08
Tr41	89,04	96,02	86,37	95,83	40	87,21	95,21	50	89,14	**96,32**	90	**90,55**	96,25
Tr45	85,97	91,59	87,81	**94,69**	50	85,34	90,86	100	88,10	93,26	50	**93,30**	94,60
WAP	**74,08**	**85,69**	61,86	81,00	60	68,45	84,24	70	71,61	85,65	90	72,06	85,42
Média	81,64	86,92	82,32	88,65	47,14	81,41	87,07	78,57	81,96	87,53	79,28	85,55	89,48
Wilcoxon p-value	2,25e-10	8,61e-09	4,36e-09	7,96e-03	-	1,03e-11	1,98e-08	-	2,64e-10	2,59e-07	-	-	-
Holm p-value	1,57e-09	3,44e-08	2,18e-08	7,96e-03	-	8,24e-11	5,94e-08	-	1,58e-09	5,18e-07			

CoDiT foi comparado com 4 métodos: i) SVM, o classificador individual mais utilizado em Classificação de Documentos [165, 166]; ii) *Bootstrap AGGregatING* [139], também conhecido como *Bagging*, comumente utilizado como base comparativa para sistemas de múltiplos classificadores [140, 141, 142]; iii) *Random Subspace* [143], por conta de sua capacidade de lidar com problemas de alta dimensionalidade [147, 145]; iv) *Random Forest* [148], devido à sua robustez [149].

Como exposto na Seção 5.2, os métodos de combinação utilizaram 10 valores para definir o tamanho do *pool* de classificadores: $L = \{10, 20, \ldots, 100\}$. Entretanto, o *Random Forest* foi uma exceção que utilizou apenas uma configuração com $L = 100$ e o número de características para cada classificador foi de um décimo do original ($V/10$). Essa configuração foi estabelecida em trabalhos anteriores [167, 35]. Foi utilizada a implementação do *Random Forest* em *Python* com a função *RandomForestClassifier* do pacote *scikit-learn*[2].

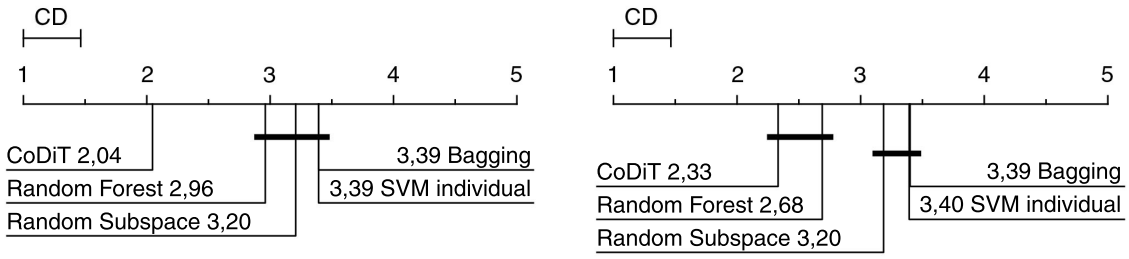
Com relação ao *Random Subspace*, cada classificador foi treinado com 20% das características originais, seguindo a configuração utilizada em outros trabalhos [168, 144].

A Tabela 5.10 mostra o tamanho do *ensemble* (L), macro-F1 e micro-F1 obtida por cada um dos métodos. Todos os resultados foram avaliados, mas apenas os melhores são exibidos, juntamente com seu respectivo L. O SVM utiliza apenas um classificador, logo não há combinação, e *Random Forest* foi executado utilizando apenas uma configuração ($L = 100$), portanto ambos apresentam apenas suas performances. Os melhores resultados de cada base de dados são destacados em negrito. A última linha mostra o *p-value* do teste estatístico *Wilcoxon signed rank*, para verificar se CoDiT é superior aos demais métodos. Baseado no teste de *Wilcoxon*, CoDiT é considerado estatisticamente superior em comparação com todos os outros métodos com um nível de confiança de 99%.

Os métodos também foram comparados utilizando o teste estatístico do *Friedman*. A

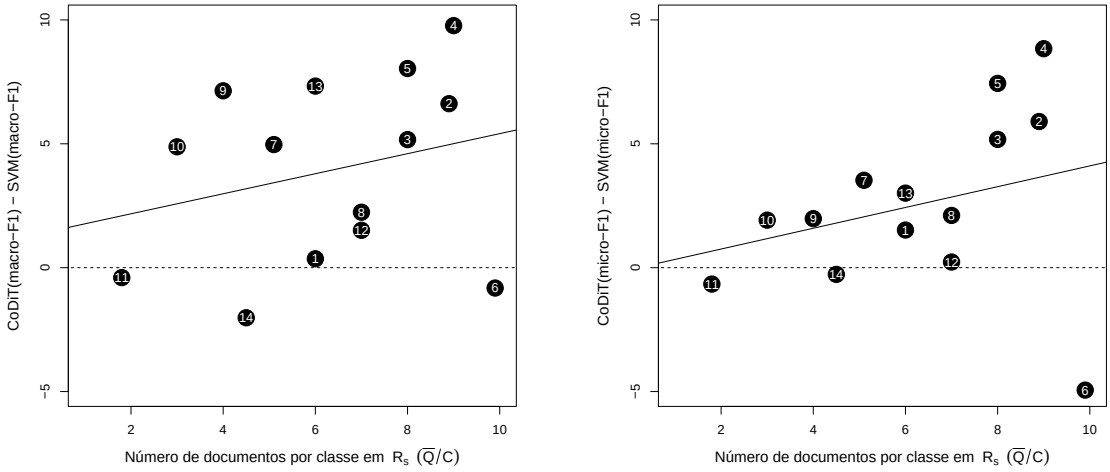
[2]Disponível em `http://scikit-learn.org/stable/`

Figura 5.10 mostra dois diagramas CD [162], um para cada medida de desempenho. CoDiT obteve o menor *rank* médio em ambas as medidas. Com a macro-F1, o CoDiT resultados estatisticamente superiores se comparado com os demais métodos, enquanto que com a micro-F1, CoDiT não foi considerado melhor que o *Random Forest*, mas foi superior aos demais.



(a) (b)

Figura 5.10: Diagramas CD para macro-F1 (a) e micro-F1 (b) comparando os métodos. *Rank* médio exibido para cada método e CD = 0.4648 nos dois diagramas.



(a) (b)

Figura 5.11: Influência da média dos documentos por classe no conjunto de representação $\mathbf{R}_s$ $(\bar{Q}/C)$ sobre a macro-F1 (a) e micro-F1 (b) do CoDiT menos a macro-F1 e micro-F1 do SVM (base comparativa). Cada ponto é uma base de dados da Tabela 5.1, o número é uma referência direta à coluna 'id'. A linha tracejada indica quando não há diferenças entre CoDiT e SVM, e a linha continua é a correlação entre os eixos dos gráficos.

A Figura 5.11 mostra os resultados do CoDiT menos os resultados do SVM com relação ao número de documentos por classe no conjunto de representação (Q/C) para cada base de dados. O valor de Q é a média dentre todos os *folds*, pois seu valor é modificado de acordo com a distribuição das classes naquele *fold*. Cada ponto dos gráficos da Figura 5.11 representa o resultado de uma base de dados, deste modo cada gráfico possui 14 pontos. Pontos abaixo da linha pontilhada indicam que o SVM foi melhor que o CoDiT, enquanto que pontos acima

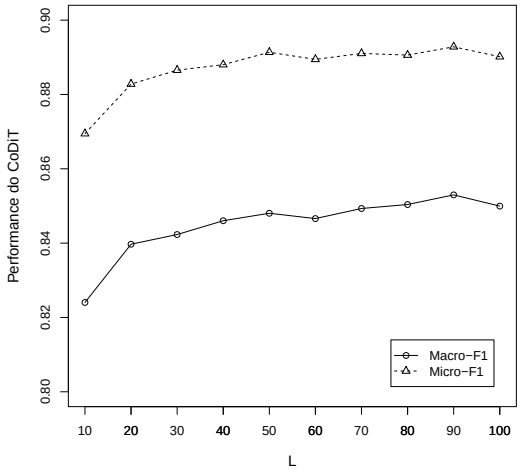


Figura 5.12: Performance do CoDiT (macro-F1 e micro-F1) quando diferentes tamanhos de *ensemble* (L) são utilizados.

indicam que o CoDiT foi superior ao SVM. A melhoria do CoDiT aumenta com $\bar{Q}/C$, como mostrado pela correlação (linha contínua). Esse é o comportamento esperado, pois quanto mais documentos por classe no conjunto de representação, maior é a quantidade de informação usada para classificar um documento de teste. Entretanto, algumas bases de dados não apresentam essa correlação, sendo os exemplos mais notáveis com Re0 (6) e WAP (14). No caso da base Re0, o CoDiT possui o pior resultado em comparação com qualquer outro método na micro-F1, dado o enfoque equilibrado para todas as classes, provavelmente fazendo a classe majoritária (40.43% de toda a base) ser negligenciada; e no caso da WAP, foi o melhor resultado do SVM em comparação com qualquer outro método, logo, é esperado que CoDiT seja inferior quando comparado com SVM.

A Figura 5.12 baseia-se nos resultados dos experimentos para mostrar uma avaliação da variação da performance (macro-F1 e micro-F1) sobre o tamanho do *ensemble* (L) para o CoDiT. O teste estatístico de *Wilcoxon* foi usado para comparar cada incremento de L ($L = 10$ contra $L = 20$, $L = 20$ contra $L = 30$, e assim por diante). O teste de *Wilcoxon* mostrou que o CoDiT continua melhorando sua performance, com um nível de confiança de 90%, até $L = 50$. Deste momento em diante, não existe diferença estatística significativa entre $L = 50$ e $L = \{60, 70, 80, 90, 100\}$. Sendo a única exceção a comparação de $L = 50$ com $L = 90$ em comparação com a macro-F1, no qual $L = 90$ é considerado estatisticamente superior. Portanto, a melhor configuração em relação à performance é $L = 90$, mas a melhor configuração levando em consideração o balanço entre tamanho do *ensemble* L e performance é $L = 50$.

5.7 Comparação das Propostas

Nesta seção serão comparadas todas as propostas e suas diferentes configurações, isto é, CR, CoDiS, CoDiT com $dt()$ e CoDiT com $dts()$. Tendo em vista que CR possui configurações

Tabela 5.11: Macro-F1 e micro-F1 dos melhores resultados para as propostas: CR, CoDiS e CoDiT com suas duas funções. Junto com os métodos de combinação de classificadores está o respectivo tamanho do *ensemble* (L). Os resultados do CR são apenas os sem seleção de protótipos. Em negrito estão marcados os melhores resultados em cada base de dados.

	CR		CoDiS			CoDiT com $dt()$			CoDiT com $dts()$		
Bases	macro-F1	micro-F1	L	macro-F1	micro-F1	L	macro-F1	micro-F1	L	macro-F1	micro-F1
CSTR	**86,63 (4,91)**	**84,94 (5,78)**	90	83,89 (8,40)	81,68 (7,66)	90	80,92 (6,82)	80,73 (5,34)	70	81,78 (8,69)	80,29 (5,50)
Oh0	87,41 (3,53)	88,35 (3,53)	100	85,04 (3,84)	86,63 (3,95)	80	92,26 (2,58)	**92,45 (2,52)**	90	**92,47 (1,44)**	92,40 (1,51)
Oh5	87,84 (3,27)	87,92 (3,11)	80	85,94 (4,79)	86,17 (4,51)	100	**91,72 (3,51)**	**91,91 (3,20)**	100	89,47 (3,81)	89,74 (3,62)
Oh10	77,71 (4,59)	80,74 (4,39)	70	74,65 (3,12)	77,91 (2,95)	100	**84,68 (4,07)**	**86,43 (3,17)**	60	83,64 (2,70)	84,58 (2,31)
Oh15	80,62 (3,80)	81,61 (3,91)	40	77,62 (4,75)	78,49 (4,94)	70	**88,60 (2,70)**	**88,44 (3,03)**	80	86,69 (3,26)	86,01 (3,09)
Re0	74,83 (9,24)	**85,89 (2,58)**	50	64,81 (8,82)	82,03 (2,29)	90	**77,21 (8,70)**	78,71 (2,52)	100	69,37 (8,29)	75,22 (2,97)
Re1	73,67 (7,12)	86,90 (2,00)	90	64,60 (4,12)	81,90 (2,13)	50	**79,40 (3,33)**	**87,52 (1,04)**	80	74,13 (4,24)	83,98 (1,45)
SsW	**95,67 (3,10)**	**95,77 (2,96)**	20	93,04 (3,34)	93,06 (3,29)	30	94,97 (3,68)	94,88 (3,60)	40	92,69 (3,11)	92,46 (3,29)
Tr11	72,21 (4,41)	88,19 (3,13)	50	72,07 (8,12)	88,01 (4,73)	90	**81,05 (9,38)**	**89,18 (5,04)**	60	73,98 (9,27)	88,10 (4,51)
Tr12	83,84 (6,02)	86,50 (5,42)	60	82,48 (8,01)	85,58 (3,52)	90	**88,09 (6,34)**	**89,15 (5,00)**	50	82,69 (8,00)	85,23 (7,02)
Tr31	**83,61 (2,01)**	**98,17 (1,26)**	40	81,24 (1,52)	96,77 (1,11)	90	82,96 (1,46)	97,08 (1,55)	60	80,76 (4,76)	97,13 (2,02)
Tr41	86,21 (5,20)	95,31 (1,90)	100	84,86 (5,09)	94,61 (1,96)	90	90,55 (6,95)	96,25 (1,97)	60	**91,91 (5,86)**	**96,86 (2,15)**
Tr45	88,56 (4,69)	93,06 (1,72)	60	85,80 (7,10)	91,89 (3,20)	50	**93,30 (5,41)**	94,60 (2,25)	50	90,09 (6,04)	**95,55 (2,20)**
WAP	69,38 (4,20)	**86,44 (2,23)**	70	68,30 (4,59)	85,67 (2,83)	90	**72,06 (5,63)**	85,42 (2,54)	70	66,90 (6,63)	82,40 (2,04)
Média	82,01	88,55	65,71	78,88	86,45	79,28	85,55	89,48	69,28	82,61	87,85

dependentes da seleção de protótipos utilizada, nesta comparação é apenas utilizada a configuração sem seleção de protótipos, pois possui a melhor performance. A Tabela 5.11 mostra os melhores resultados dos métodos já citados, e baseado em um teste de *Wilcoxon* é verificado a seguinte situação para a macro-F1: CoDiT com $dt() \gg$ CoDiT com $dts() \sim$ CR $\gg$ CoDiS; e para a micro-F1: CoDiT com $dt() \sim$ CR $>$ CoDiT com $dts() >$ CoDiS. A Tabela 5.12 exibe os *p-value* de todos os testes estatísticos. De modo geral o CoDiS foi o pior método, sendo o CoDiT $dt()$ o melhor método. Entretanto, essa comparação não pode ser tão simplificada. Primeiro ponto é com relação à métrica: macro-F1 e micro-F1. Claramente, o CoDiT com $dt()$ foi superior a todos os outros métodos na macro-F1, um forte indicativo de que o CoDiT trabalha melhor com bases desbalanceadas. Enquanto que na micro-F1 todos os métodos foram considerados estatisticamente similares, com exceção do CoDiS que foi considerado inferior independente da métrica. Em segundo ponto está no fato do CR ser a única proposta que utiliza apenas um classificador e possui resultados bastante competitivos. Inclusive, o CR foi considerado estatisticamente similar ao CoDiT com $dts()$.

Tabela 5.12: *P-value* do teste estatístico *Wilcoxon signed rank* realizados sobre os resultados apresentados na Tabela 5.11. O primeiro valor de cada célula é o *p-value* com relação a macro-F1, enquanto que o segundo é o *p-value* com relação a micro-F1. Todos os *p-values* foram corrigidos de acordo com o método de *Holm*.

Teste	CR	CoDiS	CoDiT $dt()$	CoDiT $dts()$
CR > ?	-	4,199e-09 / 6,271e-10	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00
CoDiS > ?	1,000e+00 / 1,000e+00	-	1,000e+00 / 1,000e+00	1,000e+00 / 1,000e+00
CoDiT $dt()$ > ?	2,338e-08 / 1,991e-01	5,228e-16 / 1,100e-08	-	9,350e-06 / 2,416e-06
CoDiT $dts()$ > ?	1,000e+00 / 1,000e+00	2,945e-06 / 5,493e-02	1,000e+00 / 1,000e+00	-

Enquanto isso, a Tabela 5.13 avalia os métodos propostos sobre uma perspectiva de custo computacional (tempo). Foram avaliados os tempos para montagem do conjunto utilizado no treinamento e no teste (TPre), bem como os tempos gastos pelo classificador para treinar (TTreino) e testar (TTeste). Todos os tempos são com relação a apenas um classificador. Claramente,

Tabela 5.13: Tempo médio, em segundos, das execuções antes do classificador (TPre), do classificador no treinamento (TTReino) e do classificador no teste (TTeste) do CR, CoDiS e CoDiT com suas duas funções. Última linha é um comparativo das médias com relação ao classificador individual mais rápido (CoDiS).

	CR			CoDiS			CoDiT com $dt()$			CoDiT com $dts()$		
Bases	TPre	TTreino	TTeste	TPre	TTreino	TTeste	TPre	TTreino	TTeste	TPre	TTreino	TTeste
CSTR	0,104	0,163	0,048	0,044	0,090	0,021	0,099	2,138	0,233	0,032	0,128	0,040
Oh0	1,137	2,133	0,452	0,323	0,517	0,090	0,656	293,716	11,192	0,110	12,779	0,561
Oh5	0,966	1,677	0,377	0,287	0,433	0,081	0,593	193,213	8,230	0,057	8,067	0,441
Oh10	1,272	2,399	0,532	0,368	0,620	0,098	0,731	426,128	14,908	0,091	15,510	0,732
Oh15	1,001	1,828	0,414	0,287	0,508	0,084	0,604	264,224	10,051	0,073	9,158	0,589
Re0	2,439	3,894	0,752	0,643	1,197	0,157	1,262	815,691	36,616	0,130	77,353	2,593
Re1	3,020	7,146	1,321	0,815	1,867	0,231	1,379	404,456	23,544	0,108	18,681	1,295
SsW	0,156	0,125	0,031	0,059	0,051	0,017	0,143	2,467	0,313	0,023	0,144	0,032
Tr11	0,455	0,265	0,062	0,148	0,153	0,043	0,575	12,923	1,829	0,076	1,119	0,221
Tr12	0,250	0,172	0,046	0,094	0,471	0,021	0,243	2,788	0,530	0,048	0,328	0,087
Tr31	2,257	1,031	0,205	0,654	0,692	0,061	0,248	0,371	0,454	0,074	0,040	0,071
Tr41	1,673	1,064	0,236	0,495	0,318	0,060	1,529	77,350	10,229	0,175	8,045	1,316
Tr45	1,314	0,673	0,140	0,386	0,754	0,067	1,414	68,406	8,711	0,178	6,693	1,105
WAP	4,568	6,166	1,235	1,278	1,316	0,201	1,990	182,541	28,006	0,324	17,863	3,896
Média	1,472	2,052	0,417	0,420	0,641	0,088	0,819	196,172	11,060	0,107	12,564	0,927
Comparação	3,50x	3,20x	4,73x	x	x	x	1,95x	306,04x	125,68x	0,25x	19,60x	10,53x

CoDiT demanda mais tempo que as demais propostas, especialmente a configuração com $dt()$, isso ocorre simplesmente por conta do aumento exponencial da quantidade de características, tanto no treinamento quanto no teste o procedimento aumenta a quantidade de documentos. A configuração com $dts()$ tem um tempo bastante inferior à $dt()$, pois a métrica inspirada na similaridade do cosseno funciona quase como uma seleção de características devido ao seu funcionamento de zerar características cujo qualquer um dos pares possuir um zero. Entretanto, apesar da melhoria no tempo do CoDiT com $dts()$, tanto CoDiS como o CR demandam menos tempo. Isso ocorre devido à redução da dimensionalidade. Mais notavelmente, CR não é um sistema de múltiplos classificadores, portanto apesar de na média seu tempo ser superior ao CoDiS, se o CoDiS utilizar 10 classificadores (L mínimo utilizado nos experimentos) o tempo gasto pelo CoDiS já será superior ao CR.

5.8 Considerações Finais

Neste capítulo foram apresentados os experimentos, iniciando com a apresentação das bases de dados e as configurações gerais dos experimentos, seguindo para os resultados nas diversas configurações dos métodos propostos. Primeiramente, foram realizados experimentos preliminares para delimitar alguns parâmetros e definir configurações para uma das arquiteturas propostas (CoDiS). Os experimentos preliminares trouxeram resultados interessantes e foram elaboradas duas seções para as configurações obtidas: CR, uma arquitetura sem combinação de classificadores que utiliza similaridade do cosseno como métrica para a mudança de representação e algoritmos de seleção de protótipos para a geração do conjunto de representação; e CoDiS, a arquitetura proposta de fato, uma combinação de classificadores que utiliza distância Euclidiana como métrica para a mudança de representação, seleção aleatória de protótipos para geração dos

conjuntos de representação, compostos por 20% dos documentos do conjunto de treinamento. Essas duas configurações foram comparadas com métodos equivalentes. CR, além de reduzir a dimensionalidade, atingiu resultados estatisticamente superiores, quando não utiliza seleção de protótipos, em comparação com BoW base; enquanto que CoDiS foi superior estatisticamente ao *Bagging* com ALOFT e ao *Random Subspace*. Em um segundo momento, foi avaliado o CoDiT, outra arquitetura proposta. Devido às restrições impostas pelo uso da *Dichotomy Transformation*, as configurações do CoDiT foram estabelecidas rapidamente e seus resultados foram comparados com SVM individual, *Random Forest* e *Random Subspace*. CoDiT mostrou-se superior aos demais métodos com relação à macro-F1, mostrando seu potencial para lidar com bases desbalanceadas; e foi equivalente ao *Random Forest* e superior aos demais com relação à micro-F1. O terceiro momento foi para comparação dos métodos propostos em suas diversas configurações. CoDiT mostrou-se a melhor arquitetura tanto para macro-F1 como para micro-F1 quando utiliza $dt()$ como métrica para transformação. Entretanto, essa configuração requer muito mais tempo que as demais. Destacou-se o CR devido seu bom desempenho e também seu baixo custo computacional em comparação com as demais configurações.

6

Conclusões

O principal objetivo deste trabalho foi de elaborar arquiteturas capazes de beneficiar-se de transformações de características em sistemas de múltiplos classificadores. Para tal, foram propostas duas arquiteturas: CoDiS (*Combined Dissimilarity Spaces*), que utiliza *Dissimilarity Representation*, e CoDiT (*Combined Dichotomy Transformations*), que utiliza *Dichotomy Transformation*. As duas transformações, de acordo com nosso conhecimento, nunca foram utilizadas em problemas de Classificação de Documentos. A proposta do CoDiS foi baseada na premissa de diminuir os dois problemas da representação *Bag-of-Words*: alta dimensionalidade e esparsidade dos dados. Enquanto que a proposta do CoDiT ataca diretamente a maldição da dimensionalidade, aumentando a quantidade de documentos presente no treinamento; e transforma o problema multi-classe em um problema binário, aumentando a performance do classificador. A união dessas transformações com Combinação de Classificadores permitiu a criação de sistemas robustos, pois ao criar diferentes conjuntos de representação (importante parâmetro nas duas transformações) é possível obter *ensembles* diversos e, consequentemente, melhorar o sistema como um todo. Através dos experimentos, foi observado que as arquiteturas propostas atingiram bons resultados em comparação com métodos equivalentes, sendo estatisticamente superior na maioria dos casos estudados.

Apesar da proposta das duas arquiteturas, ao final deste trabalho duas configurações distintas para cada proposta foram obtidas, totalizando as quatro configurações a seguir:

i) *Combined Dissimilarity Spaces* (CoDiS): proposta apresentada no Capítulo 3, que utiliza entradas diferentes para as várias *Dissimilarity Representations* usadas no processo. Experimentos preliminares mostraram que essa arquitetura se beneficia do uso da distância Euclidiana e seleção aleatória para atingir o melhor balanço entre diversidade e performance;

ii) *Cosine Representation* (CR): esta configuração utiliza a *Dissimilarity Representation* com a métrica de similaridade do cosseno para realizar a transformação, sendo uma variação do CoDiS sem combinação de classificadores. A similaridade do cosseno foi capaz de atingir ótima performance com classificadores individuais. Tendo em vista que não há combinação no CR, também foi verificado o comportamento do conjunto

de representação obtido por algoritmos de seleção de protótipos em vez da seleção aleatória usada em todos as propostas que utilizando combinação de classificadores;

iii) *Combined Dichotomy Transformations* (CoDiT) com Diferença: proposta apresentada no Capítulo 4, que utiliza-se de entradas diferentes para as várias *Dichotomy Transformations* usadas no processo. Devido às restrições impostas pela transformação usada no CoDiT, certos parâmetros foram estabelecidos de modo a evitar o crescimento exponencial dos exemplos e para funcionamento correto da regra de classificação. Deste modo, foi utilizada seleção aleatória por classe para obter o conjunto de representação, isto é, cada classe ter a mesma quantidade de documentos no conjunto de representação usado no CoDiT. Para a transformação, foi utilizada a Equação (2.26), no qual a característica transformada é o valor absoluto da diferença com relação as características dos documentos do conjunto de representação.

iv) *Combined Dichotomy Transformations* (CoDiT) com Similaridade: exatamente a mesma abordagem do CoDiT com Diferença, mas utilizando a Equação (4.7) para realizar a transformação. A Equação (4.7) foi proposta neste trabalho e é inspirada na similaridade do cosseno. Apenas a modificação da função usada na *Dichotomy Transformations* foi capaz de obter resultados bastante diferentes, bem como o tempo de execução desta configuração.

Comparativamente, cada configuração possui seus méritos. CoDiS consegue tratar os problemas de alta dimensionalidade e esparsidade de modo extremamente eficaz, se comparado com outras metodologias que tratam os mesmos problemas. Dado a redução no número de características, CoDiS possui os classificadores mais rápidos dentre as propostas. Entretanto, se comparada com as demais configurações propostas, CoDiS tem a pior performance. Enquanto isso, a CR, consegue uma performace competitiva entre as demais propostas e por se tratar de um único classificador (não realiza combinação) é a configuração mais rápida. Ambas as configurações do CoDiT possuem um alto custo computacional, devido ao aumento exponencial do número de exemplos usados para treinamento e teste. Entretanto, a performance do CoDiT com Diferença é a melhor dentre todas as propostas, sendo superior estatisticamente. Por outro lado, a CoDiT com Similaridade é uma configuração inferior em performance, mas com tempo de execução médio 15 vezes mais rápido se comparada com a CoDiT com Diferença. Tanto a perda de performance como a diminuição no tempo de execução ocorrem no CoDiT com Similaridade, pois a função utilizada na transformação tem um comportamento similar a uma seleção de características, isto é, a quantidade de informação é reduzida. Portanto, CR e CoDiT com Diferença foram as melhores configurações das propostas, uma por seu tempo de execução aliado a uma boa performance e a outra pela sua superioridade de performance. Sendo CoDiT com Similaridade uma alternativa viável, caso o tempo necessário para a execução do CoDiT com Diferença seja inviável para a aplicação em questão.

6.1 Resultados Alcançados

Como resultados alcançados, destacam-se:

i) Duas propostas de arquiteturas baseadas em transformação de características;

ii) CoDiS, uma das arquiteturas propostas, utiliza-se da *Dissimilarity Representation* para reduzir a dimensionalidade e esparsidade dos dados;

iii) CoDiT, uma das arquiteturas propostas, utiliza-se da *Dichotomy Transformation* para construir um *ensemble* robusto;

iv) Aplicação tanto da *Dissimilarity Representation*, quanto da *Dichotomy Transformation*, nunca antes utilizadas para Classificação de Documentos;

v) Estudo de uma variação do CoDiS, que utiliza similaridade do cosseno na *Dissimilarity Representation* sem combinação de classificadores;

6.2 Trabalhos Futuros

Durante o desenvolvimento deste trabalho, surgiram alguns ideias fora do escopo da tese que podem ser investigados em trabalhos futuros:

i) Verificar o impacto de outras métricas na *Dissimilarity Representation*;

ii) Verificar performance do CoDiT em outros problemas, em especial problemas com poucos exemplos e bases desbalanceadas;

iii) Verificar performance das proposta substituindo o classificador atual (SVM) por algum classificador fraco;

iv) Substituir o Classificador utilizado no CoDiT por alguma heurística simples, baseado na *Dichotomy Transformation*, para diminuir o tempo de execução da arquitetura;

v) Modificar o módulo de Geração dos Conjuntos de Representação do CoDiT para tratar bases de dados desbalanceadas, visando aumentar o número de documentos por classe no conjunto de representação;

vi) Verificar se o desempenho da similaridade do cosseno no CoDiS e função inspirada na similaridade do cosseno no CoDiT atingem melhores resultados utilizando TF-IDF.

vii) Verificar o quão efetiva seria uma combinação do CoDiS e CoDiT em uma única arquitetura.

Referências

[1] BROWN, G. et al. Diversity creation methods: a survey and categorisation. *Information Fusion*, Elsevier, v. 6, n. 1, p. 5–20, 2005.

[2] SEBASTIANI, F. Machine learning in automated text categorization. *ACM Computing Surveys*, ACM, v. 34, n. 1, p. 1–47, 2002. ISSN 0360-0300.

[3] LI, C. H.; HUANG, J. X. Spam filtering using semantic similarity approach and adaptive bpnn. *Neurocomputing*, Elsevier, v. 92, p. 88–97, 2012.

[4] WANG, G. et al. Sentiment classification: The contribution of ensemble learning. *Decision support systems*, Elsevier, v. 57, p. 77–93, 2014.

[5] LEWIS, D. D. Feature selection and feature extraction for text categorization. In: *Workshop on Speech and Natural Language*. [S.l.: s.n.], 1992. p. 212–217.

[6] SU, J.; SAYYAD-SHIRABAD, J.; MATWIN, S. Large scale text classification using semi-supervised multinomial naive bayes. In: *Proceedings of the International Conference on Machine Learning*. [S.l.: s.n.], 2011. p. 97–104.

[7] BELLMAN, R.; KALABA, R. Reduction of dimensionality, dynamic programming, and control processes. *Journal of Fluids Engineering*, American Society of Mechanical Engineers, v. 83, n. 1, p. 82–84, 1961.

[8] KIRA, K.; RENDELL, L. A. The feature selection problem: Traditional methods and a new algorithm. In: JOHN WILEY & SONS LTD. *Proceedings of the National Conference on Artificial Intelligence*. [S.l.], 1992. p. 129–129.

[9] CHEN, J. et al. Feature selection for text classification with naive bayes. *Expert Systems with Applications*, Elsevier, v. 36, n. 3, p. 5432–5435, 2009.

[10] PINHEIRO, R. H. W. et al. A global-ranking local feature selection method for text categorization. *Expert Systems with Applications*, v. 39, n. 17, p. 12851–12857, 2012.

[11] BACCIANELLA, S.; ESULI, A.; SEBASTIANI, F. Using micro-documents for feature selection: The case of ordinal text classification. *Expert Systems with Applications*, Elsevier, v. 40, n. 11, p. 4687–4696, 2013.

[12] YANG, J. et al. A new feature selection based on comprehensive measurement both in inter-category and intra-category for text categorization. *Information Processing and Management*, v. 48, n. 4, p. 741–754, 2012.

[13] PINHEIRO, R. H. W.; CAVALCANTI, G. D. C.; REN, T. I. Data-driven global-ranking local feature selection methods for text categorization. *Expert Systems with Applications*, v. 42, n. 4, p. 1941–1949, 2015.

[14] FRAGOSO, R. C. P.; PINHEIRO, R. H. W.; CAVALCANTI, G. D. C. Class-dependent feature selection algorithm for text categorization. In: IEEE. *International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2016. p. 3508–3515.

[15] FRAGOSO, R. C. P.; PINHEIRO, R. H. W.; CAVALCANTI, G. D. C. A method for automatic determination of the feature vector size for text categorization. In: *Brazilian Conference on Intelligent Systems*. [S.l.: s.n.], 2016. p. –.

[16] YANG, Y.; PEDERSEN, J. O. A comparative study on feature selection in text categorization. In: *Proceedings of International Conference on Machine Learning*. [S.l.: s.n.], 1997. p. 412–420.

[17] BRANK, J. et al. Interaction of feature selection methods and linear classification models. In: *Proceedings of International Conference on Machine Learning*. [S.l.]: Forthcoming, 2002.

[18] JUN, S.; PARK, S.-S.; JANG, D.-S. Document clustering method using dimension reduction and support vector clustering to overcome sparseness. *Expert Systems with Applications*, Elsevier, v. 41, n. 7, p. 3204–3212, 2014.

[19] PINHEIRO, R. H. W.; CAVALCANTI, G. D. C.; REN, T. I. Text categorization based on dissimilarity representation and prototype selection. In: *Brazilian Conference on Intelligent Systems*. [S.l.: s.n.], 2015. p. 163–168.

[20] ZHANG, P.; HE, Z. Using data-driven feature enrichment of text representation and ensemble technique for sentence-level polarity classification. *Journal of Information Science*, SAGE Publications, v. 41, n. 4, p. 531–549, 2015.

[21] ALTINEL, B.; DIRI, B.; GANIZ, M. C. A novel semantic smoothing kernel for text classification with class-based weighting. *Knowledge-Based Systems*, Elsevier, v. 89, p. 265–277, 2015.

[22] PINHEIRO, R. H. W.; CAVALCANTI, G. D. C.; REN, T. I. Combinando classificadores binários no espaço de dissimilaridade para categorização de documentos. In: *Encontro Nacional de Inteligência Artificial e Computacional*. [S.l.: s.n.], 2016. p. 253–264.

[23] PEKALSKA, E.; DUIN, R. P. Dissimilarity representations allow for building good classifiers. *Pattern Recognition Letters*, Elsevier, v. 23, n. 8, p. 943–956, 2002.

[24] CHA, S.-H.; SRIHARI, S. N. Writer identification: statistical analysis and dichotomizer. In: *Advances in Pattern Recognition*. [S.l.]: Springer, 2000. p. 123–132.

[25] ALLWEIN, E. L.; SCHAPIRE, R. E.; SINGER, Y. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, v. 1, n. Dec, p. 113–141, 2000.

[26] DIETTERICH, T. G. Ensemble methods in machine learning. In: *Multiple Classifier Systems*. [S.l.: s.n.], 2000. p. 1–15.

[27] JI, C.; MA, S. Combinations of weak classifiers. *IEEE Transactions on Neural Networks*, IEEE, v. 8, n. 1, p. 32–42, 1997.

[28] PEKALSKA, E.; DUIN, R. P.; PACLIK, P. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, Elsevier, v. 39, n. 2, p. 189–208, 2006.

[29] LAM, L. Classifier combinations: implementations and theoretical issues. In: SPRINGER. *International Workshop on Multiple Classifier Systems*. [S.l.], 2000. p. 77–86.

[30] WOLPERT, D. H. The supervised learning no-free-lunch theorems. In: *Soft Computing and Industry*. [S.l.]: Springer, 2002. p. 25–42.

[31] PRABOWO, R.; THELWALL, M. Sentiment analysis: A combined approach. *Journal of Informetrics*, Elsevier, v. 3, n. 2, p. 143–157, 2009.

[32] ÖZGüR, L.; GüNGöR, T. Text classification with the support of pruned dependency patterns. *Pattern Recognition Letters*, Elsevier, v. 31, n. 12, p. 1598–1607, 2010. ISSN 0167-8655.

[33] XIA, R.; ZONG, C.; LI, S. Ensemble of feature sets and classification algorithms for sentiment classification. *Information Sciences*, Elsevier, v. 181, n. 6, p. 1138–1152, 2011.

[34] SILVA, N. F. D.; HRUSCHKA, E. R.; HRUSCHKA, E. R. Tweet sentiment analysis with classifier ensembles. *Decision Support Systems*, Elsevier, v. 66, p. 170–179, 2014.

[35] WU, Q. et al. Forestexter: an efficient random forest algorithm for imbalanced text categorization. *Knowledge-Based Systems*, Elsevier, v. 67, p. 105–116, 2014.

[36] ONAN, A.; KORUKOĞLU, S.; BULUT, H. Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications*, Elsevier, v. 57, p. 232–247, 2016.

[37] LEWIS, D. D.; RINGUETTE, M. A comparison of two learning algorithms for text categorization. In: *Proceedings of Symposium on Document Analysis and Information Retrieval*. [S.l.: s.n.], 1994. v. 33, p. 81–93.

[38] APTE, C.; DAMERAU, F.; WEISS, S. M. Text mining with decision trees and decision rules. In: *Conference on Automated Learning and Discovery*. [S.l.: s.n.], 1998.

[39] WIENER, E.; PEDERSEN, J. O.; WEIGEND, A. S. A neural network approach to topic spotting. In: LAS VEGAS, NV, USA: UNIVERSITY OF NEVADA. *Proceedings of Symposium on Document Analysis and Information Retrieval*. [S.l.], 1995. v. 332, p. 317–332.

[40] SOUZA, A. F. D. et al. Automated multi-label text categorization with vg-ram weightless neural networks. *Neurocomputing*, Elsevier, v. 72, n. 10-12, p. 2209–2217, 2009.

[41] TAN, S. An effective refinement strategy for knn text classifier. *Expert Systems with Applications*, Elsevier, v. 30, n. 2, p. 290–298, 2006.

[42] LAM, W.; HO, C. Y. Using a generalized instance set for automatic text categorization. In: ACM. *Proceedings of International Conference on Research and Development in Information Retrieval*. [S.l.], 1998. p. 81–89.

[43] SCHAPIRE, R. E.; SINGER, Y. Boostexter: A boosting-based system for text categorization. *Machine Learning*, Springer, v. 39, n. 2, p. 135–168, 2000. ISSN 0885-6125.

[44] ZADROZNY, S.; KACPRZYK, J. Computing with words for text processing: an approach to the text categorization. *Information Sciences*, Elsevier, v. 176, n. 4, p. 415–437, 2006. ISSN 0020-0255.

[45] GODBOLE, S.; SARAWAGI, S.; CHAKRABARTI, S. Scaling multi-class support vector machines using inter-class confusion. In: ACM. *Proceedings of International Conference on Knowledge Discovery and Data mining*. [S.l.], 2002. p. 513–518.

[46] LEE, K. S.; KAGEURA, K. Virtual relevant documents in text categorization with support vector machines. *Information Processing & Management*, Elsevier, v. 43, n. 4, p. 902–913, 2007.

[47] MCCALLUM, A.; NIGAM, K. A comparison of event models for naive bayes text classification. In: AAAI PRESS. *Proceedings of Workshop on Learning for Text Categorization*. [S.l.], 1998. p. 41–48.

[48] LEWIS, D. Naive (bayes) at forty: the independence assumption in information retrieval. *Proceedings of European Conference on Machine Learning*, Springer, p. 4–15, 1998.

[49] ZHANG, X.; ZHAO, J.; LECUN, Y. Character-level convolutional networks for text classification. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 649–657.

[50] ZHOU, C. et al. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015.

[51] LEE, J. Y.; DERNONCOURT, F. Sequential short-text classification with recurrent and convolutional neural networks. *arXiv preprint arXiv:1603.03827*, 2016.

[52] IYYER, M. et al. Deep unordered composition rivals syntactic methods for text classification. In: *ACL (1)*. [S.l.: s.n.], 2015. p. 1681–1691.

[53] LI, H. Text classification retrieval based on complex network and ica algorithm. *Journal of Multimedia*, v. 8, n. 4, p. 372–378, 2013.

[54] DRUCKER, H.; WU, D.; VAPNIK, V. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, Institute of Electrical and Electronics Engineers, Inc, 445 Hoes Ln, Piscataway, NJ, 08854-1331, USA,, v. 10, n. 5, p. 1048–1054, 1999. ISSN 1045-9227.

[55] PORTER, M. F. An algorithm for suffix stripping. *Program: electronic library and information systems*, Emerald Group Publishing Limited, v. 40, n. 3, p. 211–218, 2006. ISSN 0033-0337.

[56] PAICE, C. D. Another stemmer. *ACM Sigir Forum*, v. 24, n. 3, p. 56–61, 1990. ISSN 0163-5840.

[57] VIERA, A. F. G.; VIRGIL, J. Uma revisão dos algoritmos de radicalização em língua portuguesa. *Information Research*, v. 12, n. 3, 2007.

[58] MAYFIELD, J.; MCNAMEE, P. Single n-gram stemming. In: ACM. *Proceedings of International Conference on Research and Development in Information Retrieval*. [S.l.], 2003. p. 415–416. ISBN 1581136463.

[59] MILLER, G. Wordnet: an on-line lexical database. *International Journal of Lexicography*, v. 4, n. 3, 1990.

[60] FELLBAUM, C. *WordNet: An Electronic Lexical Database*. [S.l.]: The MIT Press, 1998.

[61] LIU, H. et al. Biolemmatizer: a lemmatization tool for morphological processing of biomedical text. *Journal of biomedical semantics*, BioMed Central, v. 3, n. 1, p. 3, 2012.

[62] XUE, X. B.; ZHOU, Z. H. Distributional features for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, v. 21, n. 3, p. 428–442, 2009.

[63] DUMAIS, S. et al. Inductive learning algorithms and representations for text categorization. In: ACM. *Proceedings of International Conference on Information and Knowledge Management*. [S.l.], 1998. p. 148–155. ISBN 1581130619.

[64] LEWIS, D. D. An evaluation of phrasal and clustered representations on a text categorization task. In: ACM. *Proceedings of International Conference on Research and Development in Information Retrieval*. [S.l.], 1992. p. 37–50. ISBN 0897915232.

[65] SCOTT, S.; MATWIN, S. Feature engineering for text classification. In: *Proceedings of International Conference on Machine Learning*. [S.l.: s.n.], 1999. p. 379–388.

[66] CAROPRESO, M. F.; MATWIN, S.; SEBASTIANI, F. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. *Text Databases and Document Management: Theory and Practice*, IGI Publishing, p. 78–102, 2001.

[67] FüRNKRANZ, J. A study using n-gram features for text categorization. *Austrian Research Institute for Artifical Intelligence*, v. 3, 1998.

[68] MLADENIC, D.; GROBELNIK, M. Word sequences as features in text-learning. In: *Proceedings of Electrotechnical and Computer Science Conference*. [S.l.: s.n.], 1998. p. 145–148.

[69] TZERAS, K.; HARTMANN, S. Automatic indexing based on bayesian inference networks. In: ACM. *Proceedings of International Conference on Research and Development in Information Retrieval*. [S.l.], 1993. p. 22–35. ISBN 0897916050.

[70] KAGEURA, K.; UMINO, B. Methods of automatic term recognition: A review. *Terminology*, John Benjamins Publishing Company, v. 3, n. 2, p. 259–289, 1996. ISSN 0929-9971.

[71] LODHI, H. et al. Text classification using string kernels. *The Journal of Machine Learning Research*, JMLR. org, v. 2, p. 419–444, 2002. ISSN 1532-4435.

[72] SALTON, G.; BUCKLEY, C. Term-weighting approaches in automatic text retrieval. *Information processing & management*, Elsevier, v. 24, n. 5, p. 513–523, 1988. ISSN 0306-4573.

[73] WANG, P.; DOMENICONI, C. Building semantic kernels for text classification using wikipedia. In: ACM. *Proceedings of International Conference on Knowledge Discovery and Data mining*. [S.l.], 2008. p. 713–721.

[74] VAPNIK, V. N. *The nature of statistical learning theory*. [S.l.]: Berlin: Springer-Verlag, 1995. ISBN 0387945598.

[75] LORENA, A. C.; CARVALHO, A. C. de. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007.

[76] RIJSBERGEN, V. *Information Retrieval*. [S.l.]: London: Butterworth, 1979.

[77] SHANG, W. et al. A novel feature selection algorithm for text categorization. *Expert Systems with Applications*, Elsevier, v. 33, n. 1, p. 1–5, 2007. ISSN 0957-4174.

[78] BEKKERMAN, R. et al. Distributional word clusters vs. words for text categorization. *The Journal of Machine Learning Research*, JMLR. org, v. 3, p. 1183–1208, 2003. ISSN 1532-4435.

[79] ROGATI, M.; YANG, Y. High-performing feature selection for text classification. In: ACM. *Proceedings of International Conference on Information and Knowledge Management*. [S.l.], 2002. p. 659–661.

[80] MANNING, C.; RAGHAAN, P.; SCHüTZE, H. *Introduction to Information Retrieval*. [S.l.]: Cambridge University Press, 2008.

[81] WEIGEND, A. S.; WIENER, E. D.; PEDERSEN, J. O. Exploiting hierarchy in text categorization. *Information Retrieval*, Springer, v. 1, n. 3, p. 193–216, 1999. ISSN 1386-4564.

[82] SCHÜTZE, H.; HULL, D. A.; PEDERSEN, J. O. A comparison of classifiers and document representations for the routing problem. In: ACM. *Proceedings of International Conference on Research and Development in Information Retrieval*. [S.l.], 1995. p. 229–237. ISBN 0897917146.

[83] BAKER, L. D.; MCCALLUM, A. K. Distributional clustering of words for text classification. In: ACM. *Proceedings of International Conference on Research and Development in Information Retrieval*. [S.l.], 1998. p. 96–103. ISBN 1581130155.

[84] SLONIM, N.; TISHBY, N. The power of word clusters for text classification. In: *Proceedings of European Colloquium on Information Retrieval Research (ECIR)*. [S.l.: s.n.], 2001.

[85] LI, Y.; DONG, M.; HUA, J. Localized feature selection for clustering. *Pattern Recognition Letters*, Elsevier, v. 29, n. 1, p. 10–18, 2008. ISSN 0167-8655.

[86] YU, L.; LIU, H. Feature selection for high-dimensional data: A fast correlation-based filter solution. *Proceedings of International Conference on Machine Learning*, v. 20, n. 2, p. 856–863, 2003.

[87] KOHAVI, R.; JOHN, G. H. Wrappers for feature subset selection. *Artificial Intelligence*, Elsevier, v. 97, n. 1-2, p. 273–324, 1997.

[88] TANG, B. et al. Comparing and combining dimension reduction techniques for efficient text clustering. *Feature Selection for Data Mining*, p. 17–26, 2005.

[89] ALMUALLIM, H.; DIETTERICH, T. G. Learning with many irrelevant features. *Proceedings of the National Conference of Artificial Intelligence*, Oregon State University Corvallis, OR, USA, p. 547–552, 1991.

[90] ALMUALLIM, H.; DIETTERICH, T. G. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, Elsevier, v. 69, n. 1-2, p. 279–305, 1994. ISSN 0004-3702.

[91] ARAUZO, A.; BENITEZ, J. M.; CASTRO, J. L. C-focus: a continuous extension of focus. *Advances in Soft Computing: Engineering Design and Manufacturing*, Springer Verlag, p. 225–232, 2003.

[92] KONONENKO, I. Estimating attributes: Analysis and extensions of relief. In: SPRINGER. *Proceedings of European Conference on Machine Learning*. [S.l.], 1994. p. 171–182.

[93] FORMAN, G. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, JMLR. org, v. 3, p. 1289–1305, 2003.

[94] MLADENIC, D.; GROBELNIK, M. Feature selection for unbalanced class distribution and naive bayes. In: *Proceedings of International Conference on Machine Learning*. [S.l.: s.n.], 1999. p. 258–267.

[95] YAO, Y. Information-theoretic measures for knowledge discovery and data mining. *Entropy Measures, Maximum Entropy Principle and Emerging Applications*, p. 115–136, 2003.

[96] GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, JMLR. org, v. 3, p. 1157–1182, 2003. ISSN 1532-4435.

[97] FORMAN, G. Feature selection for text classification. In: *Computational Methods of Feature Selection*. [S.l.]: Chapman and Hall/CRC Press, 2007.

[98] DEBOLE, F.; SEBASTIANI, F. Supervised term weighting for automated text categorization. In: ACM. *Proceedings of the Symposium on Applied computing*. [S.l.], 2003. p. 784–788.

[99] MLADENIC, D. Feature subset selection in text-learning. *Proceedings of European Conference on Machine Learning*, Springer, p. 95–100, 1998.

[100] COVER, T. M.; THOMAS, J. A.; MYILIBRARY. *Elements of information theory*. [S.l.]: Wiley Online Library, 1991. v. 6.

[101] CHANG, Y. C.; CHEN, S. M.; LIAU, C. J. Multilabel text categorization based on a new linear classifier learning method and a category-sensitive refinement method. *Expert Systems with Applications*, Elsevier, v. 34, n. 3, p. 1948–1953, 2008. ISSN 0957-4174.

[102] PARK, H.; KWON, S.; KWON, H. C. Complete gini-index text (git) feature-selection algorithm for text classification. In: IEEE. *Proceedings of International Conference on Software Engineering and Data Mining*. [S.l.], 2010. p. 366–371.

[103] OGURA, H.; AMANO, H.; KONDO, M. Feature selection with a measure of deviations from poisson in text categorization. *Expert Systems with Applications*, Elsevier, v. 36, n. 3, p. 6826–6832, 2009. ISSN 0957-4174.

[104] WILBUR, W. J.; SIROTKIN, K. The automatic identification of stop words. *Journal of Information Science*, Sage Publications, v. 18, n. 1, p. 45–55, 1992. ISSN 0165-5515.

[105] YANG, Y. Noise reduction in a statistical approach to text categorization. In: ACM. *Proceedings of International Conference on Research and Development in Information Retrieval*. [S.l.], 1995. p. 256–263. ISBN 0897917146.

[106] YANG, Y.; WILBUR, J. Using corpus statistics to remove redundant words in text categorization. *Journal of the American Society for Information Science*, v. 47, n. 5, p. 357–369, 1996. ISSN 0002-8231.

[107] PEKALSKA, E. *Dissimilarity representations in pattern recognition. Concepts, theory and applications.* Tese (Doutorado) — The University of Wroc?aw, 2005.

[108] DUIN, R. P. et al. Feature-based dissimilarity space classification. *Recognizing Patterns in Signals, Speech, Images and Videos*, Springer, p. 46–55, 2010.

[109] SRIHARI, S. N.; XU, A.; KALERA, M. K. Learning strategies and classification methods for off-line signature verification. In: IEEE. *International Workshop on Frontiers in Handwriting Recognition*. [S.l.], 2004. p. 161–166.

[110] BERTOLINI, D. et al. Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers. *Pattern Recognition*, Elsevier, v. 43, n. 1, p. 387–396, 2010.

[111] RIVARD, D.; GRANGER, E.; SABOURIN, R. Multi-feature extraction and selection in writer-independent off-line signature verification. *International Journal on Document Analysis and Recognition*, Springer, v. 16, n. 1, p. 83–103, 2013.

[112] TRILOK, N. P.; CHA, S.-H.; TAPPERT, C. C. Establishing the uniqueness of the human voice for security applications. *Proceedings CSIS Research Day, Pace University, NY, May*, 2004.

[113] YOON, S. et al. On the individuality of the iris biometric. In: SPRINGER. *International Conference Image Analysis and Recognition*. [S.l.], 2005. p. 1118–1124.

[114] TAPPERT, C. C.; VILLANI, M.; CHA, S.-H. Keystroke biometric identification and authentication on long-text input. *Behavioral biometrics for human identification: Intelligent applications*, p. 342–367, 2009.

[115] GARCIA, S. et al. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 34, n. 3, p. 417–435, 2012.

[116] HART, P. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, v. 14, n. 3, p. 515–516, 1968. ISSN 0018-9448.

[117] GATES, G. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, v. 18, n. 3, p. 431–433, 1972. ISSN 0018-9448.

[118] WILSON, D. L. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, IEEE, v. 2, n. 3, p. 408–421, 1972.

[119] TOMEK, I. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, IEEE, v. 6, n. 6, p. 448–452, 1976.

[120] AHA, D. W.; KIBLER, D.; ALBERT, M. K. Instance-based learning algorithms. *Machine Learning*, Springer, v. 6, n. 1, p. 37–66, 1991.

[121] WILSON, D. R.; MARTINEZ, T. R. Instance pruning techniques. In: *Proceedings of the International Conference on Machine Learning*. [S.l.: s.n.], 1997. v. 97, p. 403–411.

[122] CAVALCANTI, G. D. C.; REN, T. I.; PEREIRA, C. L. Atisa: Adaptive threshold-based instance selection algorithm. *Expert Systems with Applications*, Elsevier, v. 40, n. 17, p. 6894–6900, 2013.

[123] YANG, Y.; AULT, T.; PIERCE, T. Combining multiple learning strategies for effective cross validation. In: CITESEER. *Proceedings of International Conference on Machine Learning*. [S.l.], 2000. p. 1167–1174.

[124] BI, Y. et al. Combining multiple classifiers using dempster's rule of combination for text categorization. In: *Modeling Decisions for Artificial Intelligence*. [S.l.]: Springer, 2004. p. 127–138.

[125] SHIPP, C. A.; KUNCHEVA, L. I. Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion*, Elsevier, v. 3, n. 2, p. 135–148, 2002.

[126] SOTO, V. et al. A double pruning scheme for boosting ensembles. *IEEE Transactions on Cybernetics*, IEEE, v. 44, n. 12, p. 2682–2695, 2014.

[127] JR, A. S. B.; SABOURIN, R.; OLIVEIRA, L. E. Dynamic selection of classifiers - a comprehensive review. *Pattern Recognition*, Elsevier, v. 47, n. 11, p. 3665–3680, 2014.

[128] DIETTERICH, T. G. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, Springer, v. 40, n. 2, p. 139–157, 2000.

[129] KUNCHEVA, L. I.; WHITAKER, C. J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, Springer, v. 51, n. 2, p. 181–207, 2003.

[130] TANG, E.; SUGANTHAN, P.; YAO, X. An analysis of diversity measures. *Machine Learning*, Springer, v. 65, n. 1, p. 247–271, 2006.

[131] YULE, G. U. On the association of attributes in statistics: with illustrations from the material of the childhood society. *Philosophical Transactions of the Royal Society of London*, JSTOR, p. 257–319, 1900.

[132] SNEATH, P. H.; SOKAL, R. R. et al. *Numerical taxonomy. The principles and practice of numerical classification*. [S.l.]: San Francisco: Freeman, 1973.

[133] SKALAK, D. B. The sources of increased accuracy for two proposed boosting algorithms. In: CITESEER. *Proceedings of Workshop on Integrating Multiple Learned Models*. [S.l.], 1996. v. 1129, p. 1133.

[134] RUTA, D.; GABRYS, B. Analysis of the correlation between majority voting error and the diversity measures in multiple classifier systems. In: *Proceedings of Soft Computing and Intelligent Systems for Industry*. [S.l.]: ICSC-NAISO Academic Press, 2001.

[135] CUNNINGHAM, P.; CARNEY, J. Diversity versus quality in classification ensembles based on feature selection. In: *European Conference on Machine Learning*. [S.l.]: Springer, 2000. p. 109–116.

[136] KOHAVI, R.; WOLPERT, D. H. et al. Bias plus variance decomposition for zero-one loss functions. In: *Proceedings of the International Conference on Machine Learning*. [S.l.: s.n.], 1996. p. 275–283.

[137] FLEISS, J. L.; LEVIN, B.; PAIK, M. C. *Statistical methods for rates and proportions*. [S.l.]: John Wiley & Sons, 2013.

[138] RODRIGUEZ, J. J.; KUNCHEVA, L. I.; ALONSO, C. J. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 28, n. 10, p. 1619–1630, 2006.

[139] BREIMAN, L. Bagging predictors. *Machine Learning*, Springer, v. 24, n. 2, p. 123–140, 1996.

[140] QIAN, Y. et al. A resampling ensemble algorithm for classification of imbalance problems. *Neurocomputing*, Elsevier, v. 143, p. 57–67, 2014.

[141] ISLAM, M. M.; YAO, X.; MURASE, K. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks*, IEEE, v. 14, n. 4, p. 820–834, 2003.

[142] KIM, M.-J.; KANG, D.-K. Classifiers selection in ensembles using genetic algorithms for bankruptcy prediction. *Expert Systems with Applications*, Elsevier, v. 39, n. 10, p. 9308–9314, 2012.

[143] HO, T. K. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 20, n. 8, p. 832–844, 1998.

[144] GANGEH, M. J.; KAMEL, M. S.; DUIN, R. P. Random subspace method in text categorization. In: *Proceedings of the International Conference on Pattern Recognition*. [S.l.: s.n.], 2010. p. 2049–2052.

[145] YU, H.; NI, J. An improved ensemble learning method for classifying high-dimensional and imbalanced biomedicine data. *IEEE Transactions on Computational Biology and Bioinformatics*, IEEE, v. 11, n. 4, p. 657–666, 2014.

[146] PATHICAL, S.; SERPEN, G. Comparison of subsampling techniques for random subspace ensembles. In: IEEE. *International Conference on Machine Learning and Cybernetics*. [S.l.], 2010. v. 1, p. 380–385.

[147] PIAO, Y. et al. Ensemble method for classification of high-dimensional data. In: IEEE. *International Conference on Big Data and Smart Computing*. [S.l.], 2014. p. 245–249.

[148] BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001.

[149] FERNÁNDEZ-DELGADO, M. et al. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, v. 15, p. 3133–3181, 2014.

[150] WU, Q. et al. Snp selection and classification of genome-wide snp data using stratified sampling random forests. *IEEE Transactions on Nanobioscience*, IEEE, v. 11, n. 3, p. 216–227, 2012.

[151] MOOSMANN, F.; NOWAK, E.; JURIE, F. Randomized clustering forests for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 30, n. 9, p. 1632–1646, 2008.

[152] LEPETIT, V.; LAGGER, P.; FUA, P. Randomized trees for real-time keypoint recognition. In: IEEE. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.], 2005. v. 2, p. 775–781.

[153] LI, Y. H.; JAIN, A. K. Classification of text documents. *The Computer Journal*, Br Computer Soc, v. 41, n. 8, p. 537–546, 1998.

[154] WOŹNIAK, M.; GRAÑA, M.; CORCHADO, E. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, Elsevier, v. 16, p. 3–17, 2014.

[155] QAMAR, A. M. et al. Similarity learning for nearest neighbor classification. In: IEEE. *International Conference on Data Mining*. [S.l.], 2008. p. 983–988.

[156] DUIN, R. P.; PEKALSKA, E. The dissimilarity representation for structural pattern recognition. In: SPRINGER. *Iberoamerican Congress on Pattern Recognition*. [S.l.], 2011. p. 1–24.

[157] WHITAKER, C.; KUNCHEVA, L. Examining the relationship between majority vote accuracy and diversity in bagging and boosting. *School of Informatics, University of Wales, Bangor*, Citeseer, 2003.

[158] MIKAWA, K.; ISHIDA, T.; GOTO, M. A proposal of extended cosine measure for distance metric learning in text classification. In: IEEE. *International Conference on Systems, Man, and Cybernetics*. [S.l.], 2011. p. 1741–1746.

[159] GOTO, M. et al. Asymptotic evaluation of distance measure on high dimensional vector spaces in text mining. In: IEEE. *International Symposium on Information Theory and Its Applications*. [S.l.], 2008. p. 1–6.

[160] GOTO, M.; ISHIDA, T.; HIRASAWA, S. Statistical evaluation of measure and distance on document classification problems in text mining. In: IEEE. *International Conference on Computer and Information Technology*. [S.l.], 2007. p. 674–673.

[161] WILCOXON, F. Individual comparisons by ranking methods. *Biometrics Bulletin*, JSTOR, v. 1, n. 6, p. 80–83, 1945.

[162] DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, v. 7, n. Jan, p. 1–30, 2006.

[163] HOLM, S. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, JSTOR, p. 65–70, 1979.

[164] KITTLER, J. et al. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 20, n. 3, p. 226–239, 1998.

[165] PANG, G.; JIANG, S. A generalized cluster centroid based classifier for text categorization. *Information Processing & Management*, Elsevier, v. 49, n. 2, p. 576–586, 2013.

[166] ROSSI, R. G. et al. Inductive model generation for text classification using a bipartite heterogeneous network. *Journal of Computer Science and Technology*, Springer, v. 29, n. 3, p. 361–375, 2014.

[167] YE, Y. et al. Stratified sampling for feature subspace selection in random forests for high dimensional data. *Pattern Recognition*, Elsevier, v. 46, n. 3, p. 769–787, 2013.

[168] CAMPOS, Y.; MORELL, C.; FERRI, F. J. A local complexity based combination method for decision forests trained with high-dimensional data. In: IEEE. *International Conference on Intelligent Systems Design and Applications*. [S.l.], 2012. p. 194–199.

Apêndice A - Lista de *Stopwords*

lemma, proof, theorem, sigir, international, symposium, figure, show, conference, proceedings, section, describe, brazil, keywords, doi, citation, republish, redistribute, publication, permission, make, digital, personal, classroom, fee, copyright, copies, acm, ieee, a, able, about, above, according, accordingly, across, actually, after, afterwards, again, against, all, allow, allows, almost, alone, along, already, also, although, always, am, among, amongst, an, and, another, any, anybody, anyhow, anyone, anything, anyway, anyways, anywhere, apart, appear, appreciate, appropriate, are, around, as, aside, ask, asking, associated, at, available, away, awfully, b, be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, believe, below, beside, besides, best, better, between, beyond, both, brief, but, by, c, came, can, cannot, cant, cause, causes, certain, certainly, changes, clearly, co, com, come, comes, concerning, consequently, consider, considering, contain, containing, contains, corresponding, could, course, currently, d, definitely, described, despite, did, different, do, does, doing, done, down, downwards, during, e, each, edu, eg, eight, either, else, elsewhere, enough, entirely, especially, et, etc, even, ever, every, everybody, everyone, everything, everywhere, ex, exactly, example, except, f, far, few, fifth, first, five, followed, following, follows, for, former, formerly, forth, four, from, further, furthermore, g, get, gets, getting, given, gives, go, goes, going, gone, got, gotten, greetings, h, had, happens, hardly, has, have, having, he, hello, help, hence, her, here, hereafter, hereby, herein, hereupon, hers, herself, hi, him, himself, his, hither, hopefully, how, howbeit, however, i, ie, if, ignored, immediate, in, inasmuch, inc, indeed, indicate, indicated, indicates, inner, insofar, instead, into, inward, is, it, its, itself, j, just, k, keep, keeps, kept, know, knows, known, l, last, lately, later, latter, latterly, least, less, lest, let, like, liked, likely, little, look, looking, looks, ltd, m, mainly, many, may, maybe, me, mean, meanwhile, merely, might, more, moreover, most, mostly, much, must, my, myself, n, name, namely, nd, near, nearly, necessary, need, needs, neither, never, nevertheless, new, next, nine, no, nobody, non, none, noone, nor, normally, not, nothing, novel, now, nowhere, o, obviously, of, off, often, oh, ok, okay, old, on, once, one, ones, only, onto, or, other, others, otherwise, ought, our, ours, ourselves, out, outside, over, overall, own, p, particular, particularly, per, perhaps, placed, please, plus, possible, presumably, probably, provides, q, que, quite, qv, r, rather, rd, re, really, reasonably, regarding, regardless, regards, relatively, respectively, right, s, said, same, saw, say, saying, says, second, secondly, see, seeing, seem, seemed, seeming, seems, seen, self, selves, sensible, sent, serious, seriously, seven, several, shall, she, should, since, six, so, some, somebody, somehow, someone, something, sometime, sometimes, somewhat, somewhere, soon, sorry, specified, specify, specifying, still, sub, such, sup, sure, t, take, takes, taken, tell, tends, th, than, thank, thanks, thanx, that, thats, the, their, theirs, them, themselves, then, thence, there, thereafter, thereby, therefore, therein, theres, thereupon, these, they, think, third, this, thorough, thoroughly, those, though, three, through, throughout, thru, thus, to, together, too, took, toward, towards, tried, tries, truly, try, trying, twice, two, u, un,

under, unfortunately, unless, unlikely, until, unto, up, upon, us, use, used, useful, uses, using, usually, uucp, v, value, various, very, via, viz, vs, w, want, wants, was, way, we, welcome, well, went, were, what, whatever, when, whence, whenever, where, whereafter, whereas, whereby, wherein, whereupon, wherever, whether, which, while, whither, who, whoever, whole, whom, whose, why, will, willing, with, wish, within, without, wonder, would, x, y, yes, yet, you, your, yours, yourself, yourselves, z, zero, about, all, am, an, and, are, as, at, be, been, but, by, can, cannot, did, do, does, doing, done, for, from, had, has, have, having, if, in, is, it, its, of, on, that, the, they, these, this, those, to, too, want, wants, was, what, which, will, with, would.