Marciele de Menezes Bittencourt

ML-MDLText: um método de classificação de textos multirrótulo de aprendizado incremental

Sorocaba, SP

27 de Março de 2020

Marciele de Menezes Bittencourt

ML-MDLText: um método de classificação de textos multirrótulo de aprendizado incremental

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCC-So) da Universidade Federal de São Carlos como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação. Linha de pesquisa: Computação Científica e Inteligência Computacional.

Universidade Federal de São Carlos – UFSCar

Centro de Ciências em Gestão e Tecnologia – CCGT

Programa de Pós-Graduação em Ciência da Computação – PPGCC-So

Orientador: Prof. Dr. Tiago Agostinho de Almeida

Coorientador: Dr. Renato Moraes Silva

Sorocaba, SP

27 de Março de 2020

Bittencourt, Marciele de Menezes

ML-MDLText: um método de classificação de textos multirrótulo de aprendizado incremental / Marciele de Menezes Bittencourt. -- 2020.
132 f. : 30 cm.

Dissertação (mestrado)-Universidade Federal de São Carlos, campus Sorocaba, Sorocaba
Orientador: Prof. Dr. Tiago A. Almeida
Banca examinadora: Profa. Dra. Katti Faceli, Profa. Dra. Solange Oliveira Rezende
Bibliografia

1. Classificação Multirrótulo. 2. Princípio da Descrição mais Simples. 3. Aprendizado Incremental. I. Orientador. II. Universidade Federal de São Carlos. III. Título.

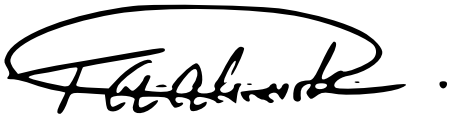
Ficha catalográfica elaborada pelo Programa de Geração Automática da Secretaria Geral de Informática (SIn).
DADOS FORNECIDOS PELO(A) AUTOR(A)
Bibliotecário(a) Responsável: Maria Aparecida de Lourdes Mariano – CRB/8 6979

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências em Gestão e Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado da candidata Marciele de Menezes Bittencourt, realizada em 27/03/2020:

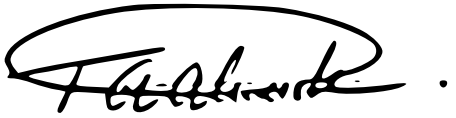


Prof. Dr. Tiago Agostinho de Almeida
UFSCar

Profa. Dra. Solange Oliveira Rezende
USP

Profa. Dra. Katti Faceli
UFSCar

Certifico que a defesa realizou-se com a participação à distância do(s) membro(s) Tiago Agostinho de Almeida Solange Oliveira Rezende, Katti Faceli e, depois das arguições e deliberações realizadas, o(s) participante(s) à distância está(ao) de acordo com o conteúdo do parecer da banca examinadora redigido neste relatório de defesa.



Prof. Dr. Tiago Agostinho de Almeida

Dedico essa dissertação aos meus pais, Geraldo e Dirce, à minha irmã, Made, e ao meu companheiro, Fernando.

Agradecimentos

Agradeço,

à Deus pela minha vida e por ter me dado força para superar as dificuldades.

ao meu orientador Tiago e ao meu coorientador Renato, pela oportunidade de tê-los como parceiros de trabalho, pela paciência, orientações e conselhos que foram indispensáveis para a realização dessa pesquisa.

aos meus pais Geraldo e Dirce, por todo amor, carinho, dedicação e por terem me ensinado a caminhar. Agradeço a Made, por ser minha irmã e amiga. Agradeço aos meus avós, Maria e Expedito (*in memorian*), que juntos com os meus pais, foram referência e me ensinaram os valores e as lições essenciais para vida. Estendo ainda meus agradecimentos à todos os demais membros de minha família, por entenderem minha ausência durante esses últimos dois anos.

ao meu companheiro, Fernando, pelas leituras, sugestões e correções em meus textos. Agradeço também por todo apoio, incentivo, paciência, compreensão e por estar ao meu lado em todos os momentos.

à minha amiga Aline, pelas revisões dos artigos em inglês e pela disposição em me ouvir.

à todos os meus professores, especialmente aos da UFSCar. Agradeço também à UFSCar pelos recursos concedidos para o desenvolvimento dessa pesquisa.

à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), pelo apoio financeiro concedido para a realização desse trabalho (Código de Financiamento 001).

a todos que de alguma forma contribuíram para a realização desse trabalho.

"Eu não posso imaginar como esse mecanismo do relógio do universo poderia existir sem um relojoeiro"

(Voltaire)

Resumo

A classificação de textos tem sido estudada extensivamente nas últimas décadas e grande parte dos trabalhos relacionados ao tema são direcionados à classificação de rótulo único e ao aprendizado *offline*. Neste tipo de aprendizado, os documentos de texto são associados a apenas um rótulo e devem estar disponíveis com antecedência para o treinamento. Problemas reais de classificação de textos, no entanto, frequentemente envolvem instâncias multirrotuladas, que se tornam disponíveis continuamente e com padrões que mudam ao longo do tempo. Para manipular esses problemas, os classificadores idealmente deveriam ser capazes de predizer múltiplos rótulos para cada documento de texto e de atualizar o seu modelo preditivo de forma eficiente, para ser escalável mesmo com recursos de memória e tempo limitados, e ser rapidamente adaptável às mudanças nos padrões dos dados. Por isso, o aprendizado *online* e a classificação multirrótulo tem atraído grande interesse de pesquisa, uma vez que existem poucos métodos capazes de abordar os dois problemas simultaneamente e frequentemente é necessário retreinar todo o modelo ou recorrer a técnicas de transformação de problemas. Nesta dissertação, é apresentado um método de classificação de textos baseado no princípio da descrição mais simples, que pode ser empregado em problemas de classificação multirrótulo sem a necessidade de transformá-los em problemas de rótulo único. Ele também apresenta a vantagem de considerar a existência de dependência entre os rótulos e de suportar o treinamento incremental naturalmente. O desempenho desse método foi avaliado empregando-o na tarefa de classificação em 15 aplicações de diferentes domínios e o resultado obtido foi comparado com os resultados de outros classificadores referência na literatura, considerando cenários de aprendizado *offline* e *online*. Os resultados obtidos pelo método proposto são muito competitivos com os resultados obtidos pelos métodos estado-da-arte avaliados.

Palavras-chaves: Classificação multirrótulo, Princípio da Descrição mais simples, Aprendizado *online*, Categorização de textos, Aprendizado de máquina.

Abstract

Single-label text classification has been extensively studied in the last decades and usually more attention has been given to offline learning scenarios, where all of the training data is available in advance. However, real-world text classification problems often involve multilabel instances and have dynamic textual patterns that can change frequently. In this context, ideally, the methods should be able ideally to predict a subset of target labels rather than a single one, and to update their model incrementally to be scalable and adaptable to changes in data patterns using limited time and memory. Therefore, online and multilabel learning have attracted great research interest, since there are few methods capable of addressing both problems simultaneously. In this study, we present a text classification method based on the minimum description length principle. It can be applied to multilabel classification without requiring the transformation of the classification problem. It also takes advantage of dependency information among labels and naturally supports online learning. We evaluated its performance using fifteen datasets from different application domains and compared it with traditional benchmarks classifiers, considering offline and online learning scenarios. The results obtained by the proposed method were very competitive with the ones of existing state-of-the-art methods.

Key-words: Multilabel classification, Minimum description length, Online learning, Text categorization, Machine learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AdaBoost	*Adaptive Boosting*
ANN	*Artificial Neural Network*
B.NB	Bernoulli *Naïve* Bayes
BOMC	*Bayesian Online Multilabel Classification Framework*
BoosTexter	*Adaptive Boosting*
BP-MLL	*Backpropagation for Multilabel Learning*
BR	*Binary Relevance*
BR+	*Binary Relevance Plus*
BRKNN-a	*Binary Relevance k-Nearest Neighbor a*
BRKNN-b	*Binary Relevance k-Nearest Neighbor b*
C4.5-ML	Método de classificação C4.5 multirrótulo
CART	*Classification and Regression Trees*
CC	*Classifiers Chain*
CF	*Confidence Factors*
DFS	*Distinguishing Feature Selector*
DMLkNN	*Dependent Multilabel k-Nearest Neighbor*
DT	*Decision Tree*
ECC	*Ensemble Classifiers Chain*
EM	*Expectation-Maximization*
EPS	*Ensemble Pruned Sets*
GMDL	*Gaussian Mixture Descriptor Learner*
HT	Hoeffding *Tree*
KNN	*k-Nearest Neighbor*

L.SVM	*Linear Support Vector Machines*
LP	*Label Powerset*
M.NB	*Multinomial Naïve* Bayes
MAP	*Maximum a Posteriori*
MDL	*Minimum Description Length*
MDL-CF	Método de classificação de spam baseado no princípio MDL e na função CF
MDL-FS	*Minimum Description Length for Feature Selection*
MDLC	*Minimum Description Length Criterion*
MDLClass	Método de classificação com dados contínuos baseado no princípio MDL
MDLText	Método de classificação de textos baseado no princípio MDL
MHT	*Multilabel* Hoeffding *Tree*
ML-KNN	*Multilabel k-Nearest Neighbor*
ML-RBF	*Multilabel Radial Basis Function*
MLNB	*Multilabel Naïve* Bayes
MLP	*Multi-layer* Perceptron
MMP	*Multiclass Multilabel* Percetron
NB	*Naïve* Bayes
PA	*Passive-Aggressive*
PS	*Pruned Sets*
PW	*Pairwise*
RAkEL	*RAndom k-labELsets*
Rank-SVM	*Ranking Support Vector Machines*
RF	*Random Forest*
SGD	*Stochastic Gradient Descent*
SVM	*Support Vector Machines*
TF	*Term Frequency*
TF-IDF	*Term Frequency-Inverse Document Frequency*

Lista de símbolos

$\mathcal{D}$	conjunto de amostras, exemplos, observações ou documentos
d_i ou d_r	amostra de índice i ou r
m	índice da última amostra ou tamanho do conjunto de amostras
$\mathcal{Q}$	conjunto de classes ou rótulos
c_j ou c_z	classe ou rótulo de índice j ou z
q ou $\|\mathcal{Q}\|$	índice da última classe ou tamanho do conjunto de classes
$\mathcal{T}$	conjunto de associações entre amostras e classes ou entre amostras e conjuntos de classes
$\|\mathcal{T}\|$	índice da última associação ou tamanho do conjunto de associações
Y_i	conjunto de rótulos associado a amostra i (*labelset* da amostra i)
$\in$	pertence
$\subseteq$	subconjunto
H	função hipótese ou modelo
H_{meta}	função hipótese ou modelo construído a partir de metadados
$\mathcal{V}$	conjunto de atributos ou termos, ou vocabulário
t_i	atributo ou termo de índice i
n	índice do último atributo ou tamanho do conjunto de atributos
$\mathcal{Q}^*$	conjunto de todos os distintos *labelsets* de $\mathcal{T}$
Y_j	*labelset* de índice j
q^* ou $\|\mathcal{Q}^*\|$	índice do último *labelset* ou tamanho do conjunto de distintos *labelsets*
$\boldsymbol{y_i}$	vetor binário que representa Y_i
$y_{i,j}$	valor atribuído à posição j do vetor binário $\boldsymbol{y_i}$
p	número de classificadores criados no EPS ou ECC
b	primeiras posições de um *ranking* de *labelsets*
k	número de vizinhos mais próximos
s	média do tamanho dos *labelsets* dos k-vizinhos mais próximos
TP_{c_j}	número de verdadeiros positivos da classe c_j
FP_{c_j}	número de falsos positivos da classe c_j
TN_{c_j}	número de verdadeiros negativos da classe c_j
FN_{c_j}	número de falsos negativos da classe c_j
$\mathcal{D}_{Teste}$	conjunto de associações separadas para teste
$H(d_i)$	conjunto de rótulos preditos para d_i pelo classificador H

$\|H(d_i)\|$	tamanho do conjunto de rótulos preditos para d_i pelo classificador H
I	função que define o valor 1 para condição verdadeira e 0 para condição falsa
Δ	diferença simétrica entre dois conjuntos
$\cup$	união de dois conjuntos
$\cap$	intersecção de dois conjuntos
$\mathcal{X}$ ou $\mathcal{B}$	conjunto de símbolos ou alfabeto de símbolos
$\|\mathcal{X}\|$	tamanho do alfabeto $\mathcal{X}$
$\mathcal{B}^*$	conjunto de todas as sequências possíveis contendo os símbolos de $\mathcal{B}$
x_i ou x_z	símbolo de índice i ou z do conjunto $\mathcal{X}$
C	função ou código que mapeia cada símbolo de um alfabeto para os símbolos de outro alfabeto
$C(x_i)$	sequência codificada de x_i
$L_C(x_i)$	tamanho da sequência gerada por $C(x_i)$
P	distribuição de probabilidade
$P(x)$	probabilidade de x
$\mathcal{M}$	conjunto de modelos
m_j	modelo ou código de índice j
$L(m_j)$	tamanho de descrição do modelo ou código m_j
$L(x\|m_j)$	tamanho de descrição de x quando codificado com o modelo ou código m_j
$P(x\|m_j)$	probabilidade condicional de x dado o modelo ou código m_j
$\bar{L}(x\|m_j)$	complexidade estocástica de x dado o modelo m_j
$L(d\|c_j)$	tamanho de descrição de d quando codificado com o modelo da classe c_j
n_{t_i,c_j}	somatório dos pesos dos termos de t_i de $\mathcal{V}$ para a classe c_j
$\hat{n}_{c_j}$	somatório dos pesos de n (somatório dos pesos de todos os termos t_i) para a classe c_j
$\hat{\mathcal{T}}_{c_j}$	quantidade de documentos de treinamento pertencentes à classe c_j
$\|d\|$	quantidade de termos no documento d
$L(t_i\|c_j)$	tamanho da descrição do termo t_i em relação à classe c_j
$K(t_i)$	penalidade baseada na relevância dos termos aplicada ao termo t_i
$\hat{S}(d, c_j)$	penalidade baseada na similaridade de cosseno para a classe c_j
$\|\Omega\|$	porção do tamanho de descrição para termos que nunca apareceram em $\mathcal{T}$
μ	constante para evitar indeterminação
$F(t_i)$	pontuação de contribuição do termo t_i

$S(d, \overline{c}_j)$	similaridade de cosseno
$\overline{c}_j$	vetor protótipo da classe c_j
$\|\|\overline{c}_j\|\|_2$	norma do vetor $\overline{c}_j$
$\hat{w}(t_i, d\|c_j)$	peso normalizado do termo t_i nos documentos da classe c_j
$\hat{w}(:, d)$	todos os pesos normalizados dos termos do documento d
$\hat{w}(t_i, d)$	peso normalizado do termo t_i do documento d
$\mathcal{Q}'$	conjunto dos distintos tamanhos dos *labelsets* de $\mathcal{Q}^*$
$\mathcal{Q}'$	índice do último elemento ou tamanho do conjunto $\mathcal{Q}'$
a_j	classe de índice j pertencente a $\mathcal{Q}'$
q'	índice do tamanho do maior *labelset* de $\mathcal{Q}^*$
n_d	classe de $\mathcal{Q}'$ predita pelo metamodelo
cm	número de classes relevantes a ser considerada
τ	índice da combinação mais frequente
ϕ_{Y_i,t_i}	número de documentos da combinação Y_i que contém o termo t_i
$\lambda_1, \lambda_2, \lambda_3$	constantes para o cálculo de fatores de confidência
σ_{n_d}	desvio padrão da classe n_d para o cálculo da função gaussiana
$G(\sigma_{n_d}, n_d, a_j)$	valor da função gaussiana G da classe a_j de acordo com σ e n_d
$\mathcal{R}$	classes com os menores tamanhos de descrição
$\mathcal{S}$	subconjunto de $\mathcal{Q}^*$
γ_1	peso aplicado a F-medida para o cálculo de σ
γ_2	peso da penalidade máxima aplicada ao tamanho de descrição
$\bar{n}$	número médio de termos por documento de $\mathcal{T}$

Sumário

Introdução

A categorização de textos é a tarefa de associar rótulos ou classes aos documentos escritos em linguagem natural para informar brevemente sobre seu conteúdo (SEBASTIANI, 2002). Com os avanços da tecnologia no armazenamento e no compartilhamento de informações em texto, um grande volume de documentos de textos é gerado a todo momento. Assim, a categorização de textos automática tem se tornado a solução mais efetiva para explorar o conteúdo desses documentos de maneira rápida e eficiente.

Em aplicações reais, o conteúdo dos documentos de textos pode estar relacionado a vários assuntos (rótulos ou categorias) ao mesmo tempo. Esse tipo de problema de classificação é conhecido como *multirrótulo* e contém várias características desafiantes. Por exemplo, o número de rótulos que são associados em cada documento é uma nova incógnita neste tipo de problema e deve ser definido pelo método de classificação. Quanto maior o número de rótulos associados em cada amostra, mais relacionamentos entre rótulos e documentos devem ser aprendidos e, consequentemente, é exigido um maior custo computacional para a criação do modelo de predição do que no aprendizado de rótulo único (ALVARES-CHERMAN; METZ; MONARD, 2012).

Muitos métodos de aprendizado de máquina clássicos na literatura não são capazes de manipular os problemas de classificação multirrótulo naturalmente e necessitam de truques de transformação. Alguns deles consideram que o problema de classificação multirrótulo é um conjunto de problemas de rótulo único e usam técnicas que os transformam em problemas de classificação binária ou multiclasse (TSOUMAKAS; KATAKIS; VLAHAVAS, 2010; GIBAJA; VENTURA, 2015). Contudo, essas transformações não são viáveis quando o número de rótulos ou de distintas combinações de rótulos (*labelsets*) é alto. Além disso, algumas características específicas do problema de classificação multirrótulo são ignoradas quando os dados são manipulados dessa forma. No aprendizado multirrótulo, por exemplo, os rótulos podem ser dependentes ou estar correlacionados e eles podem fornecer informações que ajudam a identificar a presença de outros rótulos. Algumas técnicas de transformação, como a binária, ignoram essa característica e desconsideram que exista qualquer tipo de relacionamentos entre os rótulos.

Nos problemas de classificação de textos no contexto atual, as técnicas empregadas para automatizar essa tarefa devem ser capazes de manipular um grande número crescente de amostras, de extrair aprendizado usando recursos de memória limitados e de se adaptar a ambientes dinâmicos, onde novos textos são gerados a todo momento. No entanto, muitos classificadores multirrótulo conhecidos são construídos com o aprendizado *offline*. Neste tipo de aprendizado, todos os documentos rotulados devem ser apresentados de uma só vez

para o treinamento, o que torna esses classificadores inaplicáveis em cenários dinâmicos e de larga escala. Métodos para problemas reais idealmente deveriam suportar o aprendizado *online* (incremental), pois assim, o processo de treinamento poderia ser executado com recursos limitados de memória e poderia ser atualizado, incorporando rapidamente os padrões de novos textos (READ et al., 2012a; SILVA; ALMEIDA; YAMAKAMI, 2017).

O aprendizado *online* tem sido extensivamente explorado em problemas de classificação de rótulo único nos últimos anos (GAMA, 2010; ALMEIDA; YAMAKAMI; ALMEIDA, 2010; GEPPERTH; HAMMER, 2016; READ et al., 2012b; SILVA; ALMEIDA; YAMAKAMI, 2017). Alguns autores, como Almeida, Yamakami e Almeida (2010), Almeida e Yamakami (2012), Silva, Almeida e Yamakami (2017), Freitas, Silva e Almeida (2020), criaram métodos de classificação totalmente incrementais baseados na teoria do Princípio da Descrição mais Simples (*Minimum Description Length – MDL*) (RISSANEN, 1978). O princípio MDL foi proposto por Rissanen (1978) e pode ser um grande aliado para resolver problemas de classificação, visto que sua estratégia de seleção de modelos evita o problema de sobreajustamento (*overfitting*). No entanto, os trabalhos que englobam esse princípio e o aprendizado incremental são direcionados aos problemas binários ou multiclasse.

Para preencher as lacunas relacionadas à classificação de textos multirrótulo, esse trabalho visa apresentar um método capaz de lidar com os principais desafios associados à classificação de textos e ao aprendizado multirrótulo simultaneamente. Tendo em vista as características e os trabalhos relacionados aos dois assuntos, a definição desse método foi fundamentada na seguinte pergunta de pesquisa: seria possível criar um classificador multirrótulo confiável através da adaptação de um método baseado no princípio MDL que mantenha a sua característica de ser naturalmente incremental e que não necessite de técnicas de transformação?

Para responder essa pergunta, é apresentado nesse estudo o ML-MDLText, uma adaptação do MDLText (SILVA; ALMEIDA; YAMAKAMI, 2017) capaz de realizar a categorização multirrótulo em cenários de aprendizado *online*. O MDLText é um método de categorização de textos multiclasse, que seleciona modelos baseados no princípio MDL e apresenta vantagens como a eficiência na classificação de textos, robustez contra sobreajustamento, escabilidade e aplicabilidade em cenários de aprendizado *online*. Dado o seu alto desempenho apresentado recentemente por Silva, Almeida e Yamakami (2017), é levantada a hipótese de que uma abordagem multirrótulo adaptada desse método pode manter suas principais vantagens e, portanto, levar a um alto desempenho na categorização, mesmo sem a necessidade de transformar o problema de classificação multirrótulo em problemas binários ou multiclasse.

Objetivos e contribuições

O principal objetivo desta dissertação é apresentar uma solução para os problemas de classificação de textos multirrótulo e *online* utilizando o princípio MDL, que seja eficiente em termos de desempenho de classificação e em custo computacional.

Em resumo, as principais contribuições desse trabalho são:

- Apresentação de um estudo sobre o problema de classificação de textos, sobre as abordagens direcionadas ao problema de classificação multirrótulo e sobre o princípio MDL, que serviu como base para a construção do método proposto;
- Definição e avaliação de um novo método de classificação de textos, baseado no princípio da descrição mais simples, que é capaz de lidar com os desafios da classificação de textos multirrótulo e *online*.
- Analise do desempenho do método proposto usando estratégias de avaliação de classificadores multirrótulo em diferentes cenários de aprendizado *offline* e *online*;

Organização

Essa dissertação está estruturada da seguinte maneira:

1. No Capítulo 1, são apresentados os principais conceitos relacionados à classificação de textos, bem como os principais tipos de problemas e técnicas de aprendizado de máquina empregadas para essa tarefa;
2. No Capítulo 2, são apresentadas as principais características da classificação multirrótulo e uma revisão da literatura explorando as principais abordagens empregadas na resolução desse problema. Também, são discutidas as medidas utilizadas para avaliar os classificadores multirrótulo;
3. No Capítulo 3, é descrito brevemente o princípio MDL, juntamente com suas principais aplicações na literatura. Adicionalmente, é apresentado o método MDLText, que serviu de base para a implementação do método proposto.
4. No Capítulo 4, o método proposto, o ML-MDLText, é detalhado.
5. No Capítulo 5, são reportados todos os procedimentos experimentais executados para avaliar o desempenho do método e os resultados obtidos.
6. Finalmente, são apresentadas as conclusões e os direcionamentos para possíveis trabalhos futuros.

1 Categorização de textos

Existem diversas formas de se registrar informações e, uma delas, é através da escrita. Até pouco tempo atrás, os textos eram escritos e armazenados somente em papéis e o acesso e a recuperação das informações contidas neles eram processos difíceis de serem executados, principalmente quando havia uma grande quantidade de documentos para serem analisados, pois dependiam da leitura e da organização humana. Com o avanço dos computadores, os textos puderam ser armazenados em formatos digitais, o que trouxe grandes benefícios em relação ao espaço ocupado no armazenamento e facilidade no acesso, na recuperação e na disseminação da informação.

Devido às facilidades geradas por esta nova forma de armazenar e publicar textos, um enorme volume de documentos são gerados a todo momento em meios eletrônicos. Livros, jornais e revistas, que até pouco tempo eram disponibilizados somente em papel, atualmente também contam com versões publicadas na *Internet* (SILVA, 2017). Nas empresas e até mesmo nas casas, o uso de serviços informatizados para as tarefas do dia-a-dia, como o *e-mail* e as mensagens de texto, fazem com que muitas das informações estejam armazenadas como textos. Além disso, com a popularização do uso de *smartphones* e da *Internet*, muitas pessoas passaram a compartilhar comentários e opiniões em sites e em redes sociais, o que não era uma atividade comum até algum tempo atrás, mas que se tornou grande aliada para serviços de publicidade e vendas.

Essa explosão na quantidade de documentos de texto trouxe novos desafios para a busca e a análise de seus conteúdos. Os seres humanos não são capazes de processar muitos documentos rapidamente através da leitura, o que torna cada vez mais necessário o emprego de técnicas automatizadas e eficientes para a recuperação de informação. Uma importante técnica para auxiliar nesse objetivo é a *categorização de textos*, que pode ser descrita como a tarefa de associar categorias ou rótulos aos documentos textuais, escritos em linguagem natural (SEBASTIANI, 2002). Através da categorização de textos, é possível selecionar um conjunto de documentos que compartilhem características similares e que possam conter informações úteis verificando apenas as categorias associadas, sem precisar consultar e ler todos os documentos existentes.

Neste capítulo, são abordados os principais conceitos relacionados a categorização de textos, bem como os principais tipos de problemas e a aplicação de técnicas de aprendizado de máquina na tarefa de categorização.

1.1 A categorização de textos e o aprendizado de máquina

A tarefa de classificar textos pode ser executada manualmente pelo ser humano, uma vez que através da leitura é possível interpretar o conteúdo do texto e escolher a categoria que o descreve melhor. No entanto, a categorização manual se torna um processo custoso quando existe um grande volume de documentos, o que exige a aplicação de estratégias automatizadas.

Técnicas de aprendizado de máquina são frequentemente empregadas para automatizar a tarefa de categorização de textos (SEBASTIANI, 2002). Através da observação de documentos rotulados previamente, os algoritmos de aprendizado de máquina podem aprender diferentes padrões e associações entre segmentos de textos e rótulos. Esses padrões são então utilizados para categorizar documentos não rotulados, de acordo com com os segmentos encontrados em seus conteúdos.

Formalmente, sendo $\mathcal{D} = \{d_1, d_2, ..., d_m\}$, um conjunto de m amostras ou observações, e $\mathcal{Q} = \{c_1, c_2, ..., c_q\}$, um conjunto finito com as possíveis classes ou rótulos do problema, é apresentado ao algoritmo um conjunto de documentos rotulados que pode ser expresso como $\mathcal{T} = \{(d_1, Y_1), (d_2, Y_2), ..., (d_m, Y_m)\}$, onde $d_m \in \mathcal{D}$ e $Y_m \subseteq \mathcal{Q}$. O objetivo no aprendizado supervisionado é encontrar uma função H (conhecida também como modelo, hipótese ou classificador) que mapeia as associações presentes em $\mathcal{T}$ para predizer os rótulos ou as classes de documentos não rotulados, como ilustra a Figura 1 (SCHAPIRE; SINGER, 2000; ZHANG; ZHOU, 2007; GIBAJA; VENTURA, 2015). A etapa da construção do modelo H é chamada de *treinamento*, enquanto que a etapa de predição dos rótulos de documentos não rotulados é chamada de etapa de *teste* ou de *classificação*.

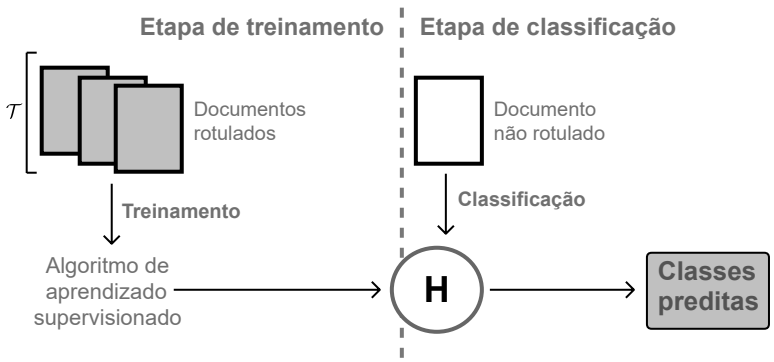


Figura 1 – Etapas de treinamento e de classificação dos métodos de aprendizado de máquina.

Nas subseções a seguir, serão descritos os conceitos básicos e fundamentais para a execução dessas etapas em aprendizado de máquina e da classificação de textos.

1.1.1 Representação computacional dos documentos de texto

Os textos são escritos de forma que as ideias e informações contidas neles possam ser facilmente compreendidas por seres humanos. No entanto, os computadores não são capazes de interpretar o conteúdo dos textos escritos em seu formato original, pois o texto é um tipo de dado *não estruturado* (WEISS; INDURKHYA; ZHANG, 2015). Tipicamente, os algoritmos de aprendizado de máquina manipulam informações estruturadas, o que obriga que os textos escritos em linguagem natural passem por processos que os transformam em uma representação mais adequada, antes de serem utilizados pelos métodos de classificação.

Existem diversas formas de representar o documento de texto computacionalmente. Uma maneira simples e frequentemente utilizada é através de um vetor de pesos de termos, também conhecido como *modelo espaço-vetorial* (VSM - *Vector Space Model*). Nesse modelo de representação, cada documento é representado por um vetor, onde cada elemento corresponde a um atributo ponderado por um peso (SALTON; WONG; YANG, 1975; SEBASTIANI, 2002; WEISS; INDURKHYA; ZHANG, 2015). O conjunto com todos os atributos usados para a representação é chamado de *vocabulário* ou de *dicionário*. A Figura 2 ilustra um exemplo dessa representação. Apesar de simples, esse modelo obtém bons resultados na maioria das aplicações (SILVA, 2017; SEBASTIANI, 2002).

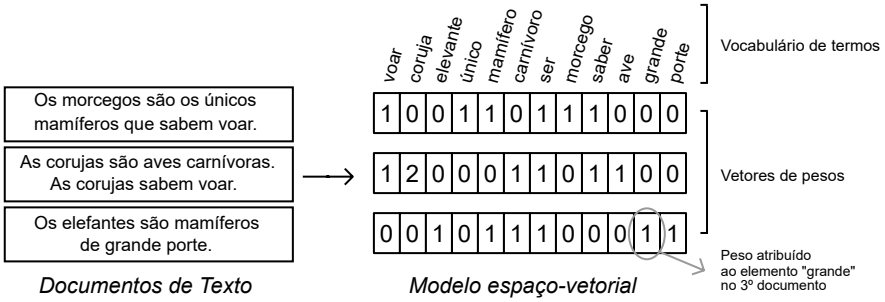


Figura 2 – Transformação dos documentos de texto para o modelo espaço-vetorial.

Para converter os documentos textuais para o modelo espaço-vetorial, inicialmente é preciso definir quais são os atributos relevantes que irão compor o vocabulário. Esse vocabulário pode ser um dicionário de palavras de um idioma, de um domínio específico ou até mesmo de palavras ou expressões presentes em um conjunto de documentos.

Uma etapa imprescindível para a definição dos atributos relevantes dos textos é chamada de pré-processamento (WEISS; INDURKHYA; ZHANG, 2015). No pré-processamento, inicialmente os textos são segmentados em *tokens* ou termos, que são as unidades mínimas do texto que podem prover informação, como símbolos, palavras individuais ou conjunto de palavras. Uma típica abordagem para realizar esta segmentação é considerar como termos qualquer sequência de caracteres separada por espaços em brancos, pontuações ou caracteres delimitadores (SILVA, 2017; WEISS; INDURKHYA; ZHANG, 2015). Termos que são considerados irrelevantes para determinar o conteúdo do

texto, como pontuação, *tags*, números, *stopwords* ou termos raros podem ser removidos dos textos. *Stopwords* são termos com alta frequência que não agregam informação ao texto, como preposições, conjunções ou artigos (SILVA, 2017; WEISS; INDURKHYA; ZHANG, 2015). Termos raros também dificilmente contribuem na identificação da categoria devido a sua baixa frequência nos documentos textuais (SILVA, 2017; WEISS; INDURKHYA; ZHANG, 2015). Processos de *estemização* e *lematização* também são aplicados para reduzir as palavras aos seus radicais ou a sua forma canônica, eliminando a possibilidade de que palavras com mesma raiz ou lema sejam consideradas diferentes devido às variações morfológicas ou de flexão. Os termos devem também ser padronizados para a mesma grafia para evitar que termos idênticos sejam tratados como diferentes apenas por diferença na caixa das letras (SILVA, 2017; UYSAL; GUNAL, 2014).

Após todas as etapas de pré-processamento, os termos que restaram são usados para criar o modelo de representação. Todos esses termos, que também compõe o vocabulário, são indexados e passam a representar cada posição do vetor, que guardam um peso relacionado a sua importância para o documento de texto, como ilustra a Figura 2.

As estratégias mais conhecidas para a atribuição de peso são o esquema *binário*, *frequência de termos (TF - Term Frequency)* e *frequência do termo–inverso da frequência nos documentos (TF-IDF - Term Frequency-Inverse Document Frequency)*(SALTON; BUCKLEY, 1988). No esquema de peso binário, os termos presentes no documento recebem o valor 1 e caso estejam ausentes, recebem o valor 0. Apesar de ser muito popular, o peso binário não faz distinção entre os termos que ocorreram apenas uma vez e termos que se repetem em um documento. Diferente do peso binário, o peso TF atribui um número maior que zero, de acordo com a contagem das ocorrências do termo no documento. Já no esquema de peso TF-IDF, a frequência de um termo no documento é ponderada pela sua ocorrência em outros textos do conjunto. O valor desse peso é alto quando o termo ocorre com alta frequência no documento em questão e com baixa frequência nos outros documentos do conjunto (SILVA, 2017; WILBUR; KIM, 2009; RENNIE et al., 2003).

A escolha do esquema de peso mais adequado depende do problema e do método de classificação que será empregado, pois cada um deles pode exigir um esquema de peso diferente.

1.1.2 Algoritmos de aprendizado supervisionado para classificação

Uma variedade de métodos tem sido empregadas em tarefas de categorização de textos, que se diferenciam pela estratégia utilizada para se obter a hipótese H. As principais estratégias utilizadas pelos métodos preditivos de aprendizado de máquina são descritas a seguir (SILVA, 2017; SEBASTIANI, 2002):

- *Métodos baseados em distâncias*: consideram a proximidade entre os documentos

para realizar as predições (FACELI et al., 2011). O método dos k-vizinhos mais próximos (KNN - *k-Nearest Neighbor*) (COVER; HART, 1967) e o k-vizinhos mais próximos multirrótulo (ML-KNN - *Multilabel k-Nearest Neighbor*) (ZHANG; ZHOU, 2007) são exemplos de métodos que utilizam essa estratégia.

- *Métodos probabilísticos*: se baseiam na probabilidade do documento pertencer a cada uma das classes possíveis do problema, de acordo com seus atributos. O Bayes ingênuo (NB - *Naïve* Bayes) (MCCALLUM; NIGAM, 1998) e o Bayes ingênuo multirrótulo (MLNB - *Multilabel Naïve* Bayes) (ZHANG; PEÑA; ROBLES, 2009) são exemplos de métodos probabilísticos.

- *Métodos baseados em árvores de decisão (DT - decision tree)*: são métodos que dividem um problema complexo em subproblemas mais simples, sob uma estrutura de árvore. As árvores de classificação e regressão (CART - *Classification and Regression Trees*) (BREIMAN et al., 1984), o C4.5 (QUINLAN, 1993) e o C4.5-multirrótulo (C4.5-ML - C4.5-*Multilabel*) (CLARE; KING, 2001) são métodos baseados em árvore de decisão.

- *Métodos baseados em otimização*: a hipótese é encontrada através da otimização de alguma função que avalia a capacidade de predição (FACELI et al., 2011). As máquinas de vetores de suporte (SVM - *Support Vector Machines*) (CORTES; VAPNIK, 1995) e as redes neurais artificias (ANN - *Artificial Neural Network*) (HAYKIN, 1998) são duas técnicas típicas que se baseiam em otimização.

- *Métodos ensemble*: treinam diferentes classificadores para a mesma tarefa de classificação e combinam os julgamentos individuais desses classificadores para gerar a predição final (SEBASTIANI, 2002). São exemplos de métodos *ensemble*, a floresta aleatória (RF - *Random Forest*) (BREIMAN, 2001) e o reforço adaptativo (AdaBoost - *Adaptive Boosting*) (FREUND; SCHAPIRE, 1996).

1.1.3 Aprendizado *offline* e *online*

Além da estratégia de classificação empregada, a técnica mais apropriada para um problema de categorização também pode ser definida de acordo com a forma com que a etapa de treinamento deve ser realizada. Diversos métodos requerem que todos os documentos rotulados sejam apresentados de uma só vez, em um processo único de treinamento conhecido como treinamento *offline* ou em *lote* (SEBASTIANI, 2002; GAMA, 2010; SILVA, 2017). Esses métodos não são capazes de incorporar novas informações no modelo de predição após a etapa de treinamento, o que exige que o processo seja refeito desde o ponto inicial quando novos documentos de texto rotulados são gerados (SILVA, 2017).

A classificação de textos costuma enfrentar problemas com muitos documentos e com altas dimensões no espaço de atributos, e assim, carregar todos os documentos em memória para o treinamento em lote nem sempre é possível. Ainda, em outros problemas reais, os documentos textuais são gerados continuamente e os conceitos das classes e os padrões nos textos podem mudar com o tempo. Métodos de classificação que suportam o aprendizado *incremental* ou *online* são mais aconselhados para essas situações (GAMA, 2010; SILVA, 2017).

No aprendizado *online*, o modelo é construído através de um conjunto de documentos de treinamento, que pode ser atualizado incrementalmente conforme os novos documentos apareçam (SEBASTIANI, 2002). Dessa forma, o aprendizado obtido nos processos anteriores de treinamento não é perdido a cada iteração. O algoritmo Perceptron (ROSENBLATT, 1958) é um método muito conhecido que permite o aprendizado incremental. Outros métodos de aprendizado *online* famosos são o Bayes ingênuo, o gradiente descendente estocástico (SGD – *Stochastic Gradient Descent*) (ZHANG, 2004) e o passivo-agressivo (PA - *Passive-Aggressive*) (CRAMMER et al., 2006). Apesar de métodos de aprendizado *online* terem vantagens em problemas dinâmicos ou de larga escala, métodos de aprendizado *offline* costumam gerar melhores resultados quando podem ser empregados (SILVA, 2017; CRAMMER; DREDZE; PEREIRA, 2012).

1.2 Classificação monorrótulo e multirrótulo

Da mesma forma que as tarefas de classificação no geral, uma das formas de distinguir as tarefas de categorização de textos é através da quantidade de categorias que estão associadas aos documentos. Em algumas aplicações, a tarefa de classificação é denominada *monorrótulo* ou classificação *de rótulo único*, pois apenas uma categoria do conjunto $\mathcal{Q}$ pode ser atribuída à cada documento do problema (CARVALHO; FREITAS, 2009). Em outras situações, os documentos de textos podem estar vinculados a um subconjunto de rótulos de $\mathcal{Q}$, ou seja, múltiplos rótulos podem ser atribuídos ao mesmo documento de texto. Nestes casos, a tarefa de classificação é denominada *multirrótulo* (CARVALHO; FREITAS, 2009; TSOUMAKAS; KATAKIS; VLAHAVAS, 2010). Para visualizar as diferenças entre a classificação monorrótulo e multirrótulo, a Figura 3 ilustra uma notícia categorizada seguindo esses dois contextos. Um texto pode abordar simultaneamente diversos assuntos e a categorização multirrótulo permite extrair mais conceitos e informações, o que é útil em uma infinidade de aplicações.

1.2.1 Classificação monorrótulo

A classificação monorrótulo é a abordagem padrão no aprendizado supervisionado. Quando o problema de classificação monorrótulo tem apenas duas classes ($|\mathcal{Q}| = 2$), ele é

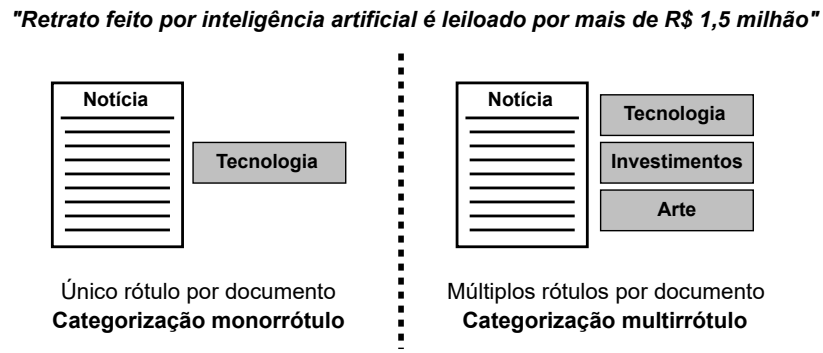


Figura 3 – Tipos de problemas de categorização de textos.

chamado de classificação *binária* (TSOUMAKAS; KATAKIS, 2007). O filtro de *spam* é um exemplo desse tipo de problema, já que existem duas possibilidades de rótulos para as mensagens de e-mail, *spam* e não spam.

Em casos onde o problema de classificação monorrótulo contém mais que duas classes ($|\mathcal{Q}| > 2$), ele recebe o nome de classificação *multiclasse* (TSOUMAKAS; KATAKIS, 2007). Um exemplo de classificação multiclasse é na categorização dos documentos de uma empresa de acordo com o setor responsável por ele.

1.2.2 Classificação multirrótulo

Muitos problemas de categorização reais dependem de uma rotulação múltipla. Uma mensagem de e-mail, por exemplo, pode ser categorizada de acordo com os diversos assuntos contidos em seu conteúdo. No diagnóstico médico, o paciente pode apresentar diversas doenças simultaneamente. Um filme pode ser rotulado com um ou mais gêneros diferentes.

Quando as classes do problema não apresentam nenhum tipo uma estrutura de níveis ou relacionamento de super e sub classe, a tarefa de classificação é denominada classificação *plana*. A rotulação de filmes por gêneros, por exemplo, pode ser considerada como um problema de classificação multirrótulo plana. Existe também um subconjunto de problemas de classificação multirrótulo, denominado *hierárquico*, na qual as classes do problema se encontram organizadas em uma estrutura de hierarquia (GIBAJA; VENTURA, 2015). Um exemplo clássico é a organização das notícias de jornal, cujos tópicos podem estar divididos em subcategorias.

1.3 Considerações finais

Este capítulo introduziu os principais conceitos relacionados à categorização de textos sob a perspectiva do aprendizado de máquina. Sistemas automáticos de categorização

de textos frequentemente usam tecnologias relacionadas ao aprendizado de máquina, pois, elas são capazes de executar essa tarefa de forma eficiente e rápida. Na discussão sobre o uso das técnicas de aprendizado, também foi apresentado o modelo de representação espaço-vetorial, que é muito utilizado para representar os textos computacionalmente e que foi utilizado nos experimentos dessa dissertação. Adicionalmente, as principais estratégias para a categorização e formas de aprendizado empregadas pelos métodos foram descritas, com o intuito de orientar o leitor nas próximas seções.

Por fim, foram distinguidos os problemas de categorização de textos monorrótulo e multirrótulo. A abordagem padrão de classificação em aprendizado de máquina considera que as classes são mutuamente exclusivas, ou seja, diferentes classes não podem ocorrer simultaneamente para um mesmo documento. Entretanto, uma grande gama de problemas em categorização de textos ignora esta restrição e dependem de abordagens específicas, capazes de prever não apenas um, mas múltiplos rótulos para um mesmo documento.

2 Classificação multirrótulo

É natural para o ser humano abordar diferentes assuntos ou informações em um mesmo texto e, por esta razão, também é comum encontrar problemas de classificação multirrótulo no processamento de textos. Várias pesquisas relacionadas ao aprendizado multirrótulo estão associadas a categorização de textos (SCHAPIRE; SINGER, 2000; CRAMMER; SINGER, 2003; MENCÍA; FÜRNKRANZ, 2008b; MENCÍA; FÜRNKRANZ, 2008a). Contudo, existem também diversas outras aplicações de aprendizado multirrótulo que abrangem as mais variadas áreas de conhecimento, como a identificação das funções de proteínas (CLARE; KING, 2001; ELISSEEFF; WESTON, 2002) e a classificação de cenas (BOUTELL et al., 2004) e vídeos (WORRING et al., 2007).

Com o propósito de apresentar uma visão geral sobre a classificação multirrótulo, neste capítulo serão abordadas algumas características do problema, os métodos existentes na literatura e as principais medidas utilizadas para avaliar o desempenho dos classificadores multirrótulo.

2.1 Definição e características do aprendizado multirrótulo

No aprendizado supervisionado multirrótulo, apresenta-se ao algoritmo um conjunto de associações de exemplos-rótulos. Sendo $\mathcal{D}$ um conjunto de exemplos ou amostras e $\mathcal{Q}$ um conjunto finito de rótulos ou classes, o conjunto de associações multirrótulo $\mathcal{T}$ é ilustrado pelo primeiro quadro da Figura 4.

$\mathcal{Q} = \{A, B, C, D\}$

Base de dados multirrótulo ($\mathcal{T}$)	
$\mathcal{D}$	Y
d_1	$\{A, C\}$
d_2	$\{A, C, D\}$
d_3	$\{B, D\}$
...	...
d_m	$\{B\}$

$\mathcal{Q} = \{A, B, C, D\}$

Base de dados multirrótulo ($\mathcal{T}$)				
$\mathcal{D}$	A	B	C	D
d_1	1	0	1	0
d_2	1	0	1	1
d_3	0	1	0	1
...	...	...	...	...
d_m	0	1	0	0

Figura 4 – Diferentes representações de um conjunto multirrótulo.

O objetivo na tarefa de classificação multirrótulo é definir uma função $H : \mathcal{D} \rightarrow 2^{\mathcal{Q}}$ que seja capaz de descrever as associações em $\mathcal{T}$ para predizer um subconjunto de rótulos Y para um exemplo não rotulado e desconhecido. A classificação binária e a multiclasse podem ser vistas como casos particulares de classificação multirrótulo, com $|\mathcal{Q}| = 2$ e $H : \mathcal{D} \rightarrow \mathcal{Q}$, respectivamente, e $|Y| = 1$ para todos os documentos (SCHAPIRE; SINGER, 2000; ZHANG; ZHOU, 2007; GIBAJA; VENTURA, 2015).

O conjunto de rótulos Y é também comumente chamado de *labelset* (GIBAJA; VENTURA, 2015) ou de combinação de rótulos. Nesta dissertação, $\mathcal{Q}^* = \{Y_1, Y_2, ..., Y_{q^*}\}$ é o conjunto que contém todos os q^* distintos *labelsets* Y presentes nas amostras de treinamento. Segundo Tsoumakas, Katakis e Vlahavas (2010), o conjunto de rótulos Y ou a saída do classificador pode ser interpretada também como uma bipartição do conjunto $\mathcal{Q}$ que distingue os rótulos relevantes dos irrelevantes. Neste caso, o conjunto Y é representado por um vetor binário $\mathbf{y} = (y_1, y_2, \ldots, y_j) = \{0, 1\}^q$, onde $y_j = 1$ se o rótulo c_j é relevante e faz parte de Y e onde $y_j = 0$, caso contrário. O segundo quadro da Figura 4 apresenta esta forma de representação.

Um problema de classificação multirrótulo pode ser visto como um conjunto de subproblemas de classificação monorrótulo, o que não atrai a atenção para o estudo de técnicas específicas para o problema. No entanto, existem características da classificação multirrótulo que as distinguem da classificação monorrótulo, o que faz com que seja necessário aplicar técnicas específicas para o tratamento adequado do problema.

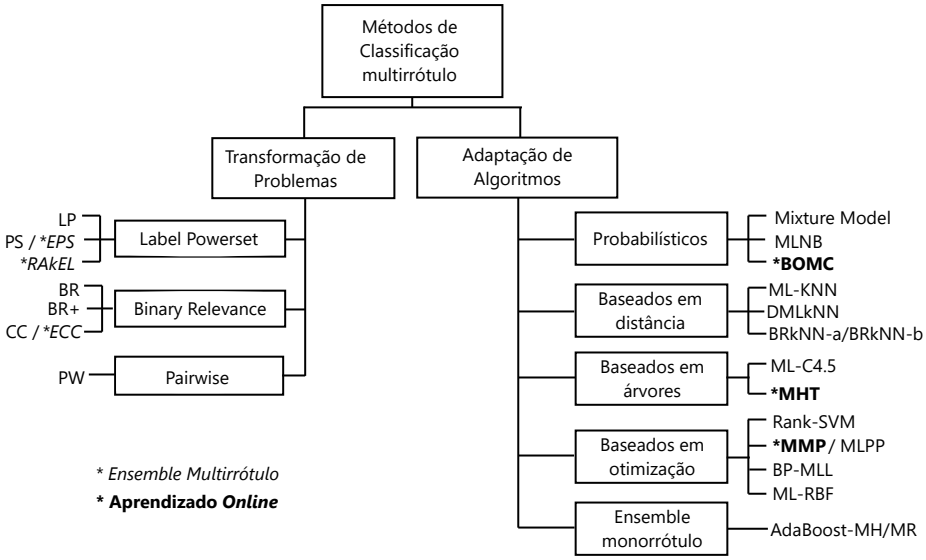
Um dos principais desafios desse tipo de classificação é que quanto maior o número de rótulos vinculados às amostras, mais relações existem entre elas e, consequentemente, maior é o custo computacional exigido para a descoberta dessas relações e para a construção dos modelos de predição. Em função disso, a complexidade de algumas abordagens utilizadas na classificação aumenta rapidamente e impede sua aplicação em problemas de larga escala (ALVARES-CHERMAN; METZ; MONARD, 2012).

Os rótulos podem ainda ser dependentes ou estar correlacionados entre si. Um texto de uma notícia rotulado como *política*, por exemplo, tem mais chances de abordar algum assunto que seja de interesse dos leitores da seção *economia* do que dos leitores da seção de *entretenimento*. Uma imagem rotulada como *mar*, tem grandes chances de pertencer a *Bahia*, dada a quantidade de praias presente nesse estado, mas as chances são nulas quando é pensado no contexto *Minas Gerais*, já que é impossível uma foto sobre *mar* ocorrer no estado de Minas Gerais. Geralmente, é assumido que estes relacionamentos entre os rótulos são conhecimentos adicionais que podem ser explorados pelos métodos de classificação para melhorar a capacidade de predição do modelo (GIBAJA; VENTURA, 2015).

2.2 Métodos de classificação multirrótulo

Segundo Tsoumakas, Katakis e Vlahavas (2010), os métodos de classificação multirrótulo podem ser divididos em dois grupos: *métodos de transformação de problemas* e *métodos de adaptação de algoritmos*. O primeiro grupo envolve os métodos que transformam o problema multirrótulo em um ou mais problemas monorrótulo e aplicam algum método tradicional para a classificação. Métodos desse grupo tem a vantagem de não

serem dependentes de um algoritmo base, ou seja, qualquer método binário ou multiclasse pode ser empregado para a classificação. Já o segundo grupo é composto por métodos de classificação monorrótulo que foram adaptados para manipular diretamente os dados multirrotulados, sem qualquer necessidade de aplicar transformações nos dados originais. Esses métodos têm como vantagem a capacidade de trabalhar com as características intrínsecas do problema multirrótulo (GIBAJA; VENTURA, 2015; TSOUMAKAS; KATAKIS; VLAHAVAS, 2010; CARVALHO; FREITAS, 2009). A Figura 5 mostra um diagrama que descreve alguns dos métodos de classificação multirrótulo mais populares que serão descritos nas próximas seções, organizando-os de acordo com a estratégia que utilizam nos problemas de classificação.



Fonte: adaptada de Gibaja e Ventura (2015, p. 52:9)

.

Figura 5 – Métodos de classificação multirrótulo.

2.2.1 Métodos de transformação de problemas

A seguir, são descritos os principais métodos de transformação de problema existentes na literatura.

2.2.1.1 Métodos baseados em relevância binária (BR)

Abordagens baseadas na transformação relevância binária enxergam o problema de classificação multirrótulo como um conjunto de problemas de classificação binária (BOUTELL et al., 2004). Seguindo essa visão, o método BR transforma o problema de classificação multirrótulo em $|\mathcal{Q}|$ subproblemas de classificação binária, um para cada rótulo.

A Figura 6 apresenta esta transformação na base de dados da Figura 4. Um classificador binário é empregado em cada um desses subproblemas, que se torna especialista na predição do rótulo presente nos dados. Dado um exemplo, cada classificador define se o seu rótulo é relevante ou não para o exemplo e, assim, a predição final é obtida através da combinação das predições individuais desses classificadores (TSOUMAKAS; KATAKIS; VLAHAVAS, 2010; GIBAJA; VENTURA, 2015). O método BR é um dos métodos mais utilizados na literatura, porém também é muito criticado por assumir que os rótulos são independentes, já que eles são manipulados de forma individual (GIBAJA; VENTURA, 2015). Por esta razão, diversas variações de métodos BR, como o cadeia de classificadores (CC - *Classifiers Chain*) (READ et al., 2011) e o BR+ (BR Plus) (ALVARES-CHERMAN; METZ; MONARD, 2012), foram propostos para agregar a informação de dependência que pode existir entre os rótulos.

Base de Dados A

$\mathcal{T}$	Y
d_1	A
d_2	A
d_3	$\sim A$
$\ldots$	$\ldots$
d_m	$\sim A$

Base de Dados B

$\mathcal{T}$	Y
d_1	$\sim B$
d_2	$\sim B$
d_3	B
$\ldots$	$\ldots$
d_m	B

Base de Dados C

$\mathcal{T}$	Y
d_1	C
d_2	C
d_3	$\sim C$
$\ldots$	$\ldots$
d_m	$\sim C$

Base de Dados D

$\mathcal{T}$	Y
d_1	$\sim D$
d_2	D
d_3	D
$\ldots$	$\ldots$
d_m	$\sim D$

Figura 6 – Transformação BR aplicada no conjunto de dados da Figura 4.

O método CC é uma variação do método BR, na qual os $|\mathcal{Q}|$ classificadores binários são ligados em cadeia de acordo com uma ordenação aleatória. O espaço de atributos de cada classificador é estendido com os rótulos dos classificadores que se encontram em posições anteriores nessa cadeia. Na predição, cada classificador recebe o exemplo a ser rotulado seguindo a ordem definida aleatoriamente, de modo que os resultados dos classificadores anteriores são usados como atributos para os próximos (READ et al., 2011; ALVARES-CHERMAN, 2014). Um problema desse método é que a ordem de montagem da cadeia pode influenciar no resultado final. Por esta razão, os autores propuseram uma outra variação do método, o *ensemble* de cadeia de classificadores (ECC - *Ensemble Classifiers Chain*). No método ECC, p classificadores CC são treinados considerando diferentes ordenações aleatórias. Os rótulos mais populares (selecionados através de um limiar) nos classificadores CC são escolhidos para a predição (READ et al., 2011).

O método BR+ tem como objetivo passar a responsabilidade de detectar as correlações entre rótulos para os próprios algoritmos de aprendizado (ALVARES-CHERMAN,

2014). Para isso, o método BR+ expande o espaço dos atributos dos subproblemas binários com as informações dos rótulos, ignorando o próprio rótulo a ser predito. Na classificação, o método gera uma estimativa inicial dos rótulos através de outro classificador multirrótulo e agrega estas estimativas como atributos na amostra a ser classificada. Dessa forma, além dos atributos originais da amostra, os classificadores binários conseguem usar também as informações das possíveis previsões dos demais rótulos. O método BR+ exige maior tempo de execução que o BR, mas é capaz de oferecer melhor qualidade de predição (ALVARES-CHERMAN, 2014).

2.2.1.2 Métodos baseados em *Label Powerset* (LP)

Métodos baseados em transformação *Label Powerset* (LP) consideram cada subconjunto de rótulos das amostras de treinamento como se fosse uma classe independente. Para trabalhar dessa forma, a técnica LP transforma os diferentes *labelsets* das amostras de treinamento em classes únicas. A Figura 7 mostra esta transformação no conjunto da Figura 4. Depois dessa transformação, um classificador multiclasse é treinado com este conjunto e é utilizado para predizer o *labelset* mais provável. Como os rótulos são analisados em conjunto com as ocorrências dos outros, é possível considerar a presença de correlação entre eles. Em situações nas quais existem muitos *labelsets* distintos e um alto desbalanceamento entre eles, o desempenho do método LP é prejudicado (TSOUMAKAS; KATAKIS; VLAHAVAS, 2010; GIBAJA; VENTURA, 2015). Outros métodos, como RAkEL (RAndom k-labELsets) (TSOUMAKAS; VLAHAVAS, 2007) e o de poda de conjuntos (PS - *Pruned Sets*) (READ; PFAHRINGER; HOLMES, 2008), foram propostos para amenizar esta limitação.

$\mathcal{Q} = \{AC, ACD, BD, ..., B\}$

Base de dados Multiclasse	
$\mathcal{T}$	y
d_1	AC
d_2	ACD
d_3	BD
$\dots$	$\dots$
d_m	B

Figura 7 – Transformação LP aplicada no conjunto de dados da Figura 4.

O método PS é uma variação da transformação LP que poda ou descarta os *labelsets* que estão presentes em um baixo número de amostras no treinamento. Para que as amostras dos *labelsets* raros não sejam perdidas, elas são reintroduzidos nos subconjuntos de rótulos que estão presentes nesses *labelsets*. Os autores utilizaram duas estratégias diferentes para realizar o procedimento de inserção: a primeira organiza os subconjuntos em um *ranking*, de acordo com o número de rótulos e a frequência nos dados de treinamento, e replica a amostra para todos os subconjuntos que ocupam as primeiras b posições. Já a segunda replica a amostra para os subconjuntos que contém mais que b rótulos. Apesar do PS ser

capaz de explorar a dependência entre os rótulos, ele tem a limitação de não considerar subconjuntos que foram podados e que não estão nos dados de treinamento. Por esta razão, os autores propuseram o uso de *ensemble* de classificadores de poda de conjuntos (EPS - *Ensemble Pruned Sets*), no qual p classificadores PS são criados. Na predição, os rótulos mais populares (definidos através de um limiar) obtidos pelos classificadores PS são selecionados (READ; PFAHRINGER; HOLMES, 2008).

O RAkEL é um método *ensemble* que constrói um conjunto de classificadores LP treinados com subconjuntos aleatórios de k rótulos do problema. Para uma amostra não conhecida, cada classificador gera uma decisão binária para cada um dos k rótulos investigados. Os rótulos de predição são definidos através da aplicação de um limiar de corte sobre a decisão média[1] de cada classe, calculada com base nos resultados dos classificadores. RAkEL resolve o problema de esparsidade gerada pela transformação LP e, ao mesmo tempo, aproveita o benefício de conseguir captar as relações entre os rótulos, gerando resultados superiores em comparação com os métodos BR e LP (TSOUMAKAS; VLAHAVAS, 2007; TSOUMAKAS; KATAKIS; VLAHAVAS, 2011).

2.2.1.3 Métodos baseados em transformação emparelhada

Na transformação emparelhada (PW - *Pairwise*) (MENCÍA; FÜRNKRANZ, 2008b; TSOUMAKAS; KATAKIS; VLAHAVAS, 2010), o problema de classificação multirrótulo é dividido em $\frac{q(q-1)}{2}$ subproblemas binários, um para cada par de rótulos. Cada um desses subproblemas contém as amostras que são rotuladas com um dos dois rótulos e ignora aquelas que são rotuladas com as duas classes simultaneamente, como ilustra a Figura 8. Da mesma forma que nos métodos de transformação BR, um classificador binário é empregado para resolver cada um desses subproblemas binários. Na classificação, os rótulos que mais apareceram nas saídas dos classificadores binários (selecionados através da aplicação de um limiar na frequência de ocorrências) são escolhidos para a predição.

2.2.2 Métodos de adaptação de algoritmo

Diversos métodos de classificação monorrótulo foram adaptados para trabalhar com o problema de classificação multirrótulo. Algumas das adaptações mais conhecidas são descritas a seguir.

2.2.2.1 Métodos probabilísticos

McCallum (1999) foi um dos pioneiros a propor um algoritmo específico para lidar com problemas de classificação multirrótulo. Em seu algoritmo, é selecionado o conjunto de

[1] A decisão média descrita pelos autores refere-se ao cálculo que relaciona o número de classificadores que apontaram o rótulo na predição e a quantidade de vezes que o rótulo apareceu nos subconjuntos aleatórios.

Base de Dados A vs B

$\mathcal{T}$	Y
d_1	A
d_2	A
d_3	B
...	...
d_m	B

Base de Dados B vs C

$\mathcal{T}$	Y
d_1	C
d_2	C
d_3	B
...	...
d_m	B

Base de Dados A vs D

$\mathcal{T}$	Y
d_1	A
d_3	D
...	...

Base de Dados B vs D

$\mathcal{T}$	Y
d_2	D
...	...
d_m	B

Base de Dados C vs D

$\mathcal{T}$	Y
d_1	C
d_3	D
...	...

Figura 8 – Transformação PW aplicada no conjunto de dados da Figura 4.

classes com maior probabilidade de predição através do princípio do máximo *a posteriori* (MAP), que é calculado analisando a mistura dos pesos e a mistura de distribuição das palavras das classes. O autor utilizou também o algoritmo de maximização de expectativa (EM - *Expectation-Maximization*) para estimar a classe responsável pela geração de cada palavra do documento de treinamento. Esse método obteve desempenho superior ao método BR na rotulação de classes que continham uma quantidade pequena ou moderada de amostras de treinamento (MCCALLUM, 1999).

Ainda, seguindo o conceito de probabilidade, Zhang, Graepel e Herbrich (2010) propuseram um *framework* de classificação multirrótulo Bayesiano de aprendizado *online* (BOMC - *Bayesian Online Multilabel Classification Framework*). Esse modelo usa um discriminante linear probabilístico para cada rótulo do problema. A verossimilhança é modelada através de um grafo de fator criado pelos autores e a probabilidade *a posteriori* pode ser estimada marginalizando os fatores desse grafo. A inferência é baseada na filtragem da densidade Gaussiana e a propagação de expectativas é aplicada em cada amostra de treinamento (ZHANG; GRAEPEL; HERBRICH, 2010). BOMC obteve resultados semelhantes ao SVM, porém, foi mais lento na etapa de treinamento. Os autores também o compararam com o LaSVM em um cenário de aprendizado incremental. Segundo eles, o BOMC apresentou resultados inferiores conforme o número de amostras do conjunto de treinamento aumentava em problemas com poucas classes.

2.2.2.2 Métodos baseados em distância

Vários métodos de categorização multirrótulo baseados no algoritmo KNN foram propostos. Uma das adaptações mais populares é o ML-KNN, proposta por Zhang e Zhou (2007). Dada uma amostra, o método identifica os k vizinhos mais próximos de acordo com a distância entre essa amostra e os exemplos de treinamento. O conjunto de rótulos de predição final é definido através do princípio MAP. Younes et al. (2011) estendeu o algoritmo ML-KNN para levar em conta a dependência entre os rótulos e propôs o kNN

multirrótulo dependente (DMLkNN - *Dependent Multilabel k-Nearest Neighbor*). Nesse método, foi adicionada a informação da presença dos outros rótulos dos vizinhos no cálculo do MAP de cada rótulo. Os resultados do DMLkNN foram superiores ao ML-KNN em todas as medidas avaliadas, porém, ele foi mais lento para executar, uma vez que existe uma complexidade extra para agregar a dependência. Devido à complexidade e ao alto uso de memória na etapa de classificação, o ML-KNN e o DMLkNN demostraram não ser viáveis para problemas com dados em larga escala (GONZALEZ-LOPEZ; VENTURA; CANO, 2018).

Spyromitros, Tsoumakas e Vlahavas (2008) propuseram outras variações do algoritmo KNN, denominadas BRKNN-a e BRKNN-b (*Binary Relevance k-Nearest Neighbor*), que são extensões da transformação BR. Essas abordagens buscam pelos k vizinhos mais próximos uma única vez para todos os rótulos, o que permite que as suas execuções sejam muito mais rápidas do que o emprego da abordagem de transformação de problema. Além disso, a possibilidade de retornar como saída um conjunto de rótulos vazio é ignorada no método BRKNN-a e, por esta razão, os autores acreditam que melhores resultados podem ser oferecidos. Na extensão BRKNN-b, os s rótulos mais populares nos k vizinhos mais próximos são selecionados para a predição, onde s é a média do tamanho dos *labelsets* desses vizinhos. Nos experimentos, as adaptações BRKNN foram melhores do que a transformação BR e o ML-KNN em diversos conjuntos de dados, enquanto que o BRKNN-b se destacou nos conjuntos com alta cardinalidade (SPYROMITROS; TSOUMAKAS; VLAHAVAS, 2008).

2.2.2.3 Métodos baseados em árvores

Problemas de classificação multirrótulo já foram alvo de estudo também na área de biologia. Nos trabalhos de Clare e King (2001), o método C4.5 (QUINLAN, 1993), baseado em árvore de decisão, foi adaptado para a classificação de gene de acordo com suas várias funções genéticas. Para levar em consideração todo o conjunto multirrótulo, a fórmula da entropia utilizada para definir o ganho de informação dos atributos na criação da árvore de decisão foi modificada, agregando informação adicional sobre a não ocorrência do rótulo nos dados de treinamento. Além disso, as folhas das árvores, que no algoritmo original se referiam às classes do problema, possuem agora informações dos subconjuntos de rótulos (CLARE; KING, 2001). As árvores criadas por Clare e King (2001) levaram à descobertas de novas regras de classificação das classes funcionais dos genes que eram desconhecidas até então.

Utilizando a mesma versão da fórmula da entropia de Clare e King (2001) para definir o ganho de informação dos atributos, Read et al. (2011) aplicaram árvores de Hoeffding (HT - Hoeffding *Tree*) para desenvolver um novo método totalmente incremental. A árvore de Hoeffding difere da clássica árvore de decisão porque é considerado apenas

um pequeno número de amostras na construção de cada nó da árvore, número este definido por um limite de Hoeffding. Na proposta dos autores, denominada árvores Hoeffding multirrótulo (MHT - *Multilabel* Hoeffding *Trees*), classificadores PS também foram empregados nas folhas das árvores para melhorar o poder de predição (READ et al., 2011).

2.2.2.4 Métodos baseados em otimização

Com o intuito de melhorar o desempenho da execução com a transformação BR com o SVM como algoritmo base, Elisseeff e Weston (2002) propuseram uma adaptação do algoritmo SVM, chamada de Rank-SVM (*Ranking Support Vector Machines*). Os autores adaptaram o algoritmo padrão para gerar um modelo linear que minimiza a perda de *ranking* e que maximiza a margem de separação através de uma equação que leva em consideração tanto a relevância quanto a irrelevância dos rótulos nos exemplos de treinamento. Segundo o autor, a habilidade de incorporar conhecimento *a priori* no *kernel* e o controle da complexidade do método são vantagens sobre outros métodos avaliados no aprendizado com conjuntos de treinamento com poucos exemplos (ELISSEEFF; WESTON, 2002).

Inúmeros trabalhos também agregaram adaptações de métodos baseados em redes neurais artificiais na literatura da classificação multirrótulo. Crammer e Singer (2003) propuseram um método que aplica o algoritmo Perceptron multiclasse e multirrótulo (MMP - *Multiclass Multilabel* Percetron) para gerar os protótipos de predição, como uma extensão à transformação BR, para a aplicação em problemas de *ranking* de tópicos. Quando é encontrado um erro na comparação entre a posição na qual o rótulo ocupa no *ranking* e o conjunto de rótulos conhecido, este exemplo é reavaliado no treinamento do protótipo da classe que gerou o erro, de modo que seus pesos são reajustados através de uma função de atualização. Os autores executaram vários experimentos com várias funções de atualização com o intuito de encontrar aquela que obtém um melhor desempenho. Diferente do MMP, Mencía e Fürnkranz (2008a), Mencía e Fürnkranz (2008b) propuseram a combinação do algoritmo Perceptron com a transformação PW. Segundo os autores, o método PW é superior ao método de relevância binária, pois, os subproblemas do primeiro são menores, característica que faz aumentar a chance de encontrar um bom hiperplano de separação. Nas variações propostas por eles, a forma de atualização dos pesos nos protótipos faz com que sejam consideradas as correlações entre os rótulos. Além disso, os classificadores construídos por estas adaptações são preparados para manipular problemas com grande quantidade de rótulos e podem ser treinados de forma incremental (MENCÍA; FÜRNKRANZ, 2008b).

Zhang e Zhou (2006) e Zhang (2009) também propuseram versões de classificadores baseados em redes neurais artificiais. A primeira versão é conhecida como BP-MLL

(*Backpropagation for Multilabel Learning*) (ZHANG; ZHOU, 2006) e agregou uma função de erro no *Backpropagation* adaptada para manipular diretamente o conjunto de dados multirrótulo. Já a segunda é uma adaptação de uma rede neural de função de base radial gaussiana para o aprendizado multirrótulo, denominada ML-RBF (*Multilabel Radial Basis Function*) (ZHANG, 2009). Nesse último, os protótipos para cada rótulo foram definidos através das centróides obtidas por análise de agrupamento (método *k-médias*) e os pesos para ativação da camada de saída foram otimizados pela minimização da soma dos erros quadráticos. O desempenho obtido pelo BP-MLL foi considerado equivalente ao de diversos métodos conhecidos para a categorização de textos, mas foi inferior ao ML-RBF, cujos resultados apontam diferença no desempenho com relação a precisão e ao tempo de treinamento (ZHANG, 2009).

2.2.2.5 *Ensembles*

A partir da técnica de *boosting*, Schapire e Singer (2000) criaram o BoosTexter, um famoso conjunto de ferramentas específicas para a categorização de textos. O BoosTexter é composto de dois métodos nomeados como Adaboost-MH e Adaboost-MR, ambos adaptados do algoritmo Adaboost (FREUND; SCHAPIRE, 1997). A ideia básica dos métodos é encontrar um modelo de classificação com alta precisão através da combinação de hipóteses fracas e da definição de pesos para as amostras de acordo com a dificuldade de predição. O que diferencia os dois métodos é o algoritmo fraco, a abordagem utilizada na distribuição dos pesos entre os exemplos e a forma como os rótulos de predição são retornados pelo classificador. Os autores observaram que costuma ocorrer sobreajustamento quando o conjunto de treinamento é pequeno, aconselhando o uso desses métodos em problemas complexos com um grande número de exemplos (SCHAPIRE; SINGER, 2000).

2.2.2.6 Métodos específicos para o aprendizado *online*

Das diversas abordagens que foram propostas para manipular os problemas de classificação multirrótulo, apenas algumas delas podem ser aplicadas na resolução de problemas de classificação em um cenário de aprendizado *online*. Árvores de Hoeffding, BOMC, e MMP são exemplos de métodos listados neste capítulo, que foram avaliados em um cenário incremental pelos seus autores.

2.3 Medidas de avaliação

O processo de avaliação dos classificadores multirrótulo exige medidas diferentes daquelas utilizadas na avaliação monorrótulo. Isso porque, como múltiplos rótulos podem ser atribuídos à um único documento, uma predição pode estar parcialmente correta, totalmente correta ou totalmente errada (GIBAJA; VENTURA, 2015), o que torna

a avaliação dos classificadores multirrótulo uma tarefa mais complicada. Dessa forma, medidas específicas foram criadas para avaliar o classificador multirrótulo sob diferentes pontos de vista. Segundo Gibaja e Ventura (2015) e Tsoumakas, Katakis e Vlahavas (2010), as medidas que avaliam os classificadores multirrótulo podem ser divididas em dois grupos: *medidas baseadas em rótulos* e *medidas baseadas em exemplos*.

2.3.1 Medidas baseadas em rótulos

O primeiro grupo é composto por medidas que avaliam a predição de cada rótulo individualmente através de alguma métrica binária e calculam uma média sobre todos os rótulos para indicar uma estimativa geral. As métricas binárias que podem ser utilizadas são aquelas baseadas na tabelas de contingência, que levam em conta a presença e a ausência de um rótulo no conjunto verdadeiro e no conjunto predito, como a precisão, revocação, F-medida e acurácia (mais informações sobre as métricas binárias podem ser conferidas no trabalho de Sokolova e Lapalme (2009)). Já a média pode ser obtida através de duas técnicas populares: a *média macro* e a *média micro*, descritas pelas equações abaixo (TSOUMAKAS; KATAKIS; VLAHAVAS, 2010; GIBAJA; VENTURA, 2015):

$$Medida_{Macro}(H) = \frac{1}{|\mathcal{Q}|}\sum_{j=1}^{|\mathcal{Q}|} Medida(TP_{c_j}, FP_{c_j}, TN_{c_j}, FN_{c_j}) \tag{2.1}$$

$$Medida_{Micro}(H) = Medida\left(\sum_{j=1}^{|\mathcal{Q}|} TP_{c_j}, \sum_{j=1}^{|\mathcal{Q}|} FP_{c_j}, \sum_{j=1}^{|\mathcal{Q}|} TN_{c_j}, \sum_{j=1}^{|\mathcal{Q}|} FN_{c_j}\right), \tag{2.2}$$

onde TP_{c_j}, FP_{c_j}, TN_{c_j} e FN_{c_j} é o número de verdadeiros e falsos positivos e de verdadeiros e falsos negativos relacionadas as predições do rótulo c_j da tabela de contingência. $Medida$ refere-se a métrica binária empregada. A média *macro* considera pesos iguais para cada rótulo e é influenciada pelo desempenho em classes raras. Já a média *micro* tende a ser dominada pelo desempenho em classes mais frequentes, pois ela considera pesos iguais para cada amostra (GIBAJA; VENTURA, 2015). Não existe um consentimento sobre a melhor medida a ser utilizada na avaliação, porém, o comportamento da média macro pode ser desejado para avaliar a classificação em problemas com rótulos desbalanceados (ALVARES-CHERMAN, 2014). Em ambos os casos, os melhores desempenhos apontam valores próximos de 1.

2.3.2 Medidas baseadas em exemplos

As medidas do segundo grupo avaliam a predição em cada amostra e geram uma avaliação média sobre todas as amostras de teste. A acurácia de subconjunto e perda de Hamming são duas medidas desse grupo muito utilizadas na literatura. Existem

também outras medidas, como a precisão, revocação, acurácia e F-medida, que são medidas originalmente empregadas na classificação monorrótulo, mas que foram adaptadas para a avaliação multirrótulo.

Para as medidas apresentadas a seguir, considerar que $\mathcal{D}_{Teste} = \{d_1, d_2, ..., d_m\}$ é o conjunto de m amostras ou exemplos selecionados para teste, $H(d_i)$ é o conjunto de rótulos preditos pelo classificador H para a amostra i e Y_i é o conjunto de rótulos verdadeiros da amostra i.

2.3.2.1 Acurácia de subconjunto (*Subset Accuracy*)

A *acurácia de subconjunto* calcula a porcentagem das vezes que um classificador prediz todo o conjunto de rótulos relevantes corretamente. Esta medida é muito rigorosa, pois não leva em consideração predições parcialmente corretas. Os melhores desempenhos apontam valores próximos de 1 (ZHU et al., 2005; GHAMRAWI; MCCALLUM, 2005; GIBAJA; VENTURA, 2015).

$$AcuraciaSubconjunto(H) = \frac{1}{m}\sum_{i=1}^{m} I(H(d_i) = Y_i), \tag{2.3}$$

onde $I(true) = 1$ e $I(false) = 0$;

2.3.2.2 Perda de Hamming (*Hamming Loss*)

A *perda de Hamming*, por sua vez, considera predições parcialmente corretas em seu cálculo. Esta medida analisa a quantidade de vezes que, em média, um par de rótulos é classificado incorretamente. Esta medida leva em consideração tanto os erros de predição quanto as omissões de predição, sendo que valores próximos de zero caracterizam um melhor poder de predição (SCHAPIRE; SINGER, 2000; GIBAJA; VENTURA, 2015).

$$PerdaHamming(H) = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{|\mathcal{Q}|}|H(d_i)\Delta Y_i|, \tag{2.4}$$

onde Δ é a diferença simétrica entre dois conjuntos;

Dembczyński et al. (2012) realizou um trabalho de comparação entre a perda de Hamming e a perda de subconjunto ($1 - AcuraciaSubconjunto(H)$). Para ele, um único algoritmo de classificação não é capaz de obter um desempenho bom em ambas medidas. Os métodos de transformação BR minimizam a perda de Hamming, enquanto que os métodos de transformação LP maximizam a acurácia de subconjunto.

2.3.2.3 Precisão, revocação, acurácia e F-medida

Outras métricas bem conhecidas para avaliar classificadores monorrótulo foram adaptadas para os problemas de classificação multirrótulo. A *revocação* (Equação 2.5) é

a porção de rótulos classificados corretamente dentre os rótulos verdadeiros da amostra e a *precisão* (Equação 2.6) é a porção de rótulos corretamente classificados dentre os rótulos positivos preditos. Já a *acurácia* (Equação 2.7) é a porção de rótulos classificados corretamente dentre todos os rótulos verdadeiros e os preditos da amostra, enquanto que a *F-medida* (Equação 2.8) é a média harmônica entre a precisão e a revocação (GODBOLE; SARAWAGI, 2004; GIBAJA; VENTURA, 2015).

$$Revocacao(H) = \frac{1}{m}\sum_{i=1}^{m}\frac{|H(d_i) \cap Y_i|}{|Y_i|} \tag{2.5}$$

$$Precisao(H) = \frac{1}{m}\sum_{i=1}^{m}\frac{|H(d_i) \cap Y_i|}{|H(d_i)|} \tag{2.6}$$

$$Acuracia(H) = \frac{1}{m}\sum_{i=1}^{m}\frac{|H(d_i) \cap Y_i|}{|H(d_i) \cup Y_i|} \tag{2.7}$$

$$F\text{-}medida(H) = \frac{1}{m}\sum_{i=1}^{m}\frac{2\,|H(d_i) \cap Y_i|}{|H(d_i)| + |Y_i|} \tag{2.8}$$

2.4 Considerações finais

Neste capítulo, foram apresentados os principais conceitos relacionados ao aprendizado multirrótulo e as principais características desse contexto de classificação. Essas características implicam em novos desafios para a tarefa de classificação, desde a criação de novos métodos de categorização até o processo de avaliação.

Algumas das principais abordagens utilizadas na categorização multirrótulo também foram introduzidas neste capítulo. Foi mostrado que uma característica que deve ser considerada na escolha da melhor abordagem para um problema de classificação é a escalabilidade. Muitas técnicas transformam a tarefa multirrótulo em uma ou várias tarefas binárias ou multiclasses e essas transformações não são apropriadas quando o número de rótulos ou o número de diferentes *labelsets* é alto. De maneira similar, a complexidade de alguns métodos adaptados para manipular diretamente os dados multirrotulados pode aumentar rapidamente, pois, como existem mais rótulos associados a cada amostra, mais recursos computacionais são necessários para identificar os padrões existentes entre eles. Apesar dessas limitações relacionadas à capacidade dos métodos em lidar com problemas de larga escala, o aprendizado multirrótulo incremental foi pouco explorado na literatura.

Por fim, dois grupos de medidas de avaliações relacionadas a bipartições também foram apresentadas. Entre elas, uma medida baseada em rótulos, a média macro com a F-medida, e duas medidas baseadas em exemplos, a perda de Hamming e a acurácia de subconjunto, foram empregadas para avaliar os classificadores nos experimentos presentes nesta dissertação.

3 O princípio da descrição mais simples e o MDLText

Em aprendizado de máquina, os métodos buscam por padrões ou regularidades em conjuntos de dados conhecidos e utilizam estas informações para criar um modelo que explica o comportamento de dados desconhecidos. De maneira geral, esses métodos também podem ser vistos como seletores de modelos (SILVA, 2017; BOUCHARD; CELEUX, 2006). Dados vários modelos ou hipóteses que descrevem um conjunto de dados, os métodos usam critérios de seleção para escolher aquele que é o mais apropriado para o conjunto de dados em questão.

Um método para a seleção de modelos que vem recebendo grande destaque é o princípio da descrição mais simples (MDL - *minimum description length*) (RISSANEN, 1978). O MDL é uma formalização da navalha de Occam (*Occam's razor*), um princípio filosófico aplicado informalmente em várias ciências e que afirma que o melhor modelo para explicar uma observação é o modelo mais simples. Diversos métodos baseados no princípio MDL foram propostos, como o MDLText (SILVA; ALMEIDA; YAMAKAMI, 2017), e seus resultados comprovam a eficiência do MDL para a seleção de modelos.

As seções a seguir descrevem brevemente o princípio MDL, juntamente com suas principais aplicações na literatura. Adicionalmente, é apresentado o método MDLText na Seção 3.3, que serviu de base para o desenvolvimento do método proposto e reportado nesta dissertação.

3.1 O princípio da descrição mais simples

O princípio MDL é um método de seleção de modelos originalmente criado por Rissanen (1978). Em um problema de seleção de modelos, se existem muitas hipóteses que descrevem uma observação, a melhor hipótese a ser escolhida, segundo o princípio MDL, é aquela que produz a descrição mais compacta (com o menor tamanho de descrição), porque é a que consegue capturar mais regularidade nos dados (SILVA; ALMEIDA; YAMAKAMI, 2017; RISSANEN, 1978; RISSANEN, 1983; BARRON; RISSANEN; YU, 1998; GRÜNWALD, 2007). O princípio MDL se baseia no conceito de compressão de dados e também na complexidade de *Kolmogorov* (KOLMOGOROV, 1965).

O conceito de aprendizado por trás da compressão é que qualquer regularidade encontrada em um dado pode ser utilizada para comprimí-lo, ou seja, qualquer regularidade pode ser usada para gerar uma codificação ou uma descrição com menos símbolos do que a quantidade necessária para listar o dado literalmente (GRÜNWALD, 2007). A descrição

de um dado pode ser, por exemplo, um programa de computador qualquer que apenas imprime ou exibe o dado e então finaliza. A Figura 9 apresenta duas sequências binárias com 10.000 elementos. A primeira sequência aparenta repetir de forma regular o trecho 0001 e ela pode ser listada usando o código da coluna ao lado, que é bem mais compacto do que a sequência original. Já a segunda sequência parece ter sido gerada de forma aleatória e como não há nenhuma regularidade observada, o código do programa não consegue descrever a sequência sem listar todos os seus elementos. Isso quer dizer que, quanto mais uma sequência pode ser comprimida (pode ser descrita de uma forma mais compacta), mais regularidades ela contém e maior é o aprendizado obtido sobre ela (GRÜNWALD, 2007).

Sequência	Forma compacta de descrição
000100010001 ... 000100010001	*for i = 1 to 2.500 print (0001)*; exit;
010000010001 ... 000101001110	*print (010000010001 ... 000101001110)*; exit;

Fonte: adaptado de Grünwald (2007, p. 4)

Figura 9 – Sequências binárias e as possíveis formas de descrição.

Kolmogorov (1965) examinou o relacionamento entre a teoria matemática da aleatoriedade e sua aplicação prática, avaliando o tamanho das descrições de sequências aleatórias e regulares quando eram exibidas por programas de computador. A complexidade de *Kolmogorov* de uma sequência foi definida como o tamanho do menor programa que a imprime e então finaliza. Dessa forma, quanto menor a complexidade de *Kolmogorov*, mais regular é a sequência. Em teoria, como o menor programa é também o que consegue capturar mais regularidade da sequência, esse conceito pode ser usado também para definir a inferência indutiva em geral. No entanto, a complexidade de *Kolmogorov* não é computável e depende de uma constante entre os tamanhos de descrição dos programas utilizados, o que a impede de ser usada para inferência indutiva com dados reais (HANSEN; YU, 2001; COVER; HART, 1967).

Diferente da complexidade de *Kolmogorov*, ao invés de avaliar os programas de computador, o princípio MDL considera que os modelos são distribuições de probabilidade e os tamanhos de descrição obtidos por eles na codificação dos dados são usados para selecionar o melhor modelo.

A codificação é a tarefa de transformar uma sequência de símbolos de um conjunto finito $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ em uma nova sequência com símbolos de outro conjunto $\mathcal{B}$ (GRÜNWALD, 2007). Os conjuntos $\mathcal{X}$ e $\mathcal{B}$ são chamados de *alfabeto*, enquanto que $\mathcal{B}^*$ é o conjunto com todas as sequências possíveis com os símbolos de $\mathcal{B}$. Um código C é uma função que mapeia cada símbolo do alfabeto $\mathcal{X}$ para uma sequência binária finita e única dos símbolos de $\mathcal{B}$, ou seja $C : \mathcal{X} \to \mathcal{B}^*$. Dessa forma, $C(x_1)$ indica o código ou a sequência utilizada para representar x_1 com o alfabeto $\mathcal{B}$ e é conhecido como *palavra-código*. A partir da palavra código $C(x_1)$, é possível decodificar para o símbolo original x_1, já que

cada símbolo de $\mathcal{X}$ tem sua própria palavra-código. O tamanho da palavra-código $C(x_1)$, ou seja, o número de bits necessários para descrever x_1 quando a codificação é gerada por C, é denotado por uma função L_C (GRÜNWALD, 2007). Para definir a relação de compressão associada à teoria do princípio MDL, são considerados códigos binários para as codificações, ou seja, $\mathcal{B} = \{1, 0\}$ e, consequentemente, cada palavra-código gerada por um código C é uma sequência de zeros e uns.

O princípio MDL utiliza códigos *livres de prefixo* para a compressão dos dados (GRÜNWALD, 2007; RODRIGUES, 2012). Segundo Grünwald (2007), existe uma relação entre os códigos livres de prefixos e os tamanhos das palavras-código. Códigos com alfabeto binário que são livres de prefixos devem satisfazer a desigualdade de Kraft abaixo (COVER; THOMAS, 2006; SILVA, 2017):

$$\sum_{z=0}^{|\mathcal{X}|} 2^{-L_C(x_z)} \leq 1 \tag{3.1}$$

De forma análoga, dado um conjunto de tamanhos de palavras-código que satisfazem esta desigualdade, existe também um código prefixo que pode ser construído com estes tamanhos de palavras-código (GRÜNWALD, 2007; COVER; THOMAS, 2006).

Os tamanhos de palavras-código de um código C que satisfazem a equação de Kraft definem uma distribuição de probabilidade P (algumas vezes imperfeita, quando $\sum_{z=0}^{|\mathcal{X}|} 2^{-L_C(x_z)} < 1$) sobre $\mathcal{X}$ (GRÜNWALD, 2007). Sendo assim, o tamanho de código $L_C(x)$, gerado por um código prefixo C ao codificar x, corresponde à função probabilidade $P(x)$, expressa por:

$$x \in \mathcal{X} : P(x) \equiv 2^{-L_C(x)}, \tag{3.2}$$

e, consequentemente, dada uma distribuição de probabilidade P sobre $\mathcal{X}$ e dado que os tamanhos das palavras-código são números naturais, é possível também encontrar um código prefixo C, que é constituído por tamanhos de palavra-códigos que podem ser expressos por:

$$x \in \mathcal{X} : L_C(x) \equiv \lceil -\log_2 P(x) \rceil \tag{3.3}$$

Assim, é possível identificar uma correspondência entre o tamanho da palavra-código e as distribuições de probabilidade de $\mathcal{X}$, já que quanto maior a probabilidade de x, menor será o tamanho da palavra-código que representa x na codificação C e vice-versa (GRÜNWALD, 2007).

Como é possível calcular o tamanho da codificação de uma sequência sem ser necessário gerar a codificação literalmente (HANSEN; YU, 2001), apenas os tamanhos

das codificações são considerados para realizar a seleção de modelos no princípio MDL. Dessa forma, são calculados os tamanhos das codificações ou das descrições por diferentes codificadores ou modelos e esses tamanhos são usados como uma métrica para comparar e selecionar os modelos avaliados.

Maiores detalhes sobre a teoria por trás do princípio MDL podem ser consultados nos livros Cover e Thomas (2006) e Grünwald (2007), que são excelentes referências para a base teórica. Os artigos de Barron, Rissanen e Yu (1998) e Hansen e Yu (2001) fornecem uma visão geral sobre esse princípio.

3.2 O princípio MDL na prática

Existem várias versões do princípio MDL que já foram usadas na prática. A mais antiga e também a mais simples é conhecida como *crude* MDL ou MDL em duas partes (GRÜNWALD, 2007). Nessa versão, dado um conjunto de modelos $\mathcal{M} = \{m_1, m_2, ..., m_t\}$ e uma amostra x, o modelo que deve ser escolhido para representar x é aquele que minimiza a seguinte equação:

$$M_{MDL} = \arg\min_{m_j \in \mathcal{M}} [L(m_j) + L(x|m_j)], \tag{3.4}$$

onde $L(m_j)$ é o tamanho da descrição do modelo m_j e $L(x|m_j)$ é interpretado como o tamanho da descrição (tamanho do código) de x quando codificado pelo modelo m_j (código ou distribuição de probabilidade). Essa última expressão pode ser definida pela Equação 3.3, relacionada com a probabilidade condicional de x dado m_j, ou seja, $L(x|m_j) = \lceil -\log_2 P(x|m_j) \rceil$, onde $P(x|m_j)$ é a massa de probabilidade condicional de x quando codificado com m_j.

Já a definição do valor de $L(m_j)$ da Equação 3.3 é um problema para o *crude* MDL (GRÜNWALD, 2007). Idealmente, espera-se obter o valor de $L(m_j)$ mínimo para o modelo que possui o menor tamanho em todas as funções de tamanho de descrição. No entanto, o tamanho da codificação pode ser muito grande para um código e muito pequeno para outro, o que pode tornar $L(m_j)$ uma medida aleatória. Nas primeiras publicações do MDL em duas partes, Rissanen (1978) e Rissanen (1983) propuseram a escolha de algum código *minimax* que minimizasse o menor tamanho no pior caso, onde o pior caso é obter o menor tamanho sobre todas as possíveis sequências de dados. No entanto, esse código exige uma pesada discretização do espaço de modelos, o que nem sempre é viável na prática (GRÜNWALD, 2007).

Para sanar os problemas relacionados ao princípio do MDL de duas partes, foi proposta uma versão refinada que usa modelos universais, chamada de princípio MDL *refinado* (GRÜNWALD, 2007). Nessa versão, o tamanho de descrição é projetado em

apenas uma parte do código de tamanho $\bar{L}(x|m_j)$. Assim, o modelo que deve ser escolhido para representar o dado x é aquele que minimiza a equação abaixo:

$$M_{MDL} = \arg\min_{m_j \in \mathcal{M}} \bar{L}(x|m_j), \tag{3.5}$$

onde $\bar{L}(x|m_j)$ é a complexidade estocástica de x, dado o modelo m_j (GRÜNWALD, 2007).

Uma particularidade importante que se destaca nos métodos baseados no princípio MDL é que o modelo selecionado procura um equilíbrio entre a capacidade de ajuste aos dados de treinamento e a complexidade do modelo, ou seja, o modelo escolhido é o que consegue capturar mais características nos dados e, ao mesmo, é o modelo menos complexo (SILVA; ALMEIDA; YAMAKAMI, 2017; RODRIGUES, 2012). Em aprendizado de máquina, essa é uma qualidade desejável para evitar o problema de *overfitting* (SILVA; ALMEIDA; YAMAKAMI, 2017).

3.2.1 Aplicações

O princípio MDL vem sendo empregado em inúmeras aplicações em aprendizado de máquina. Na área de pesquisa relacionada a seleção de atributos, um dos mais antigos trabalhos foi realizado por Sheinvald, Dom e Niblack (1990), que criaram um critério baseado no princípio MDL, denominado MDLC (*Minimum Description Length Criterion*), para detectar os atributos irrelevantes ou redundantes. O método separa os atributos em subgrupos e busca exaustivamente o subgrupo de atributos relevantes que satisfaz o critério MDLC. Neste mesmo tema, Drugan e Wiering (2010) propuseram uma função de pontuação, denominada MDL-FS (*Minimum Description Length for Feature Selection*), para a escolha de boas estruturas de redes bayesianas e para a remoção de atributos redundantes.

Diversos trabalhos relacionados a tarefa de categorização com árvores de decisão também fizeram uso do princípio MDL. Quinlan e Rivest (1989) o aplicaram para auxiliar na construção de nós para escolher a melhor árvore de decisão para inferir sobre um conjunto de dados. Já Mehta, Rissanen e Agrawal (1995) empregaram esse princípio em um algoritmo de poda nas árvores para evitar o problema de *overfitting*. A técnica proposta é rápida e as árvores obtidas após a poda são compactas e possuem boa acurácia.

Existem ainda inúmeros trabalhos relacionados ao processamento de linguagem natural e a categorização de textos. Grünwald (1996) usou o princípio MDL para identificar a gramática dos textos. Nesse trabalho, o algoritmo recebe um fragmento escrito em linguagem natural e deve inferir sobre o conjunto de regras gramaticais empregadas. Almeida, Yamakami e Almeida (2010) e Almeida e Yamakami (2012) empregaram o uso do princípio de Rissanen para filtragem de *spams* e obtiveram resultados superiores a

métodos renomados nessa tarefa. No método proposto, chamado de MDL-CF, os autores agregaram a função fatores de confidência (CF - *Confidence Factors*) (ASSIS et al., 2006) para dar pesos maiores para termos mais relevantes para a categorização de *spam*. Silva, Almeida e Yamakami (2017) estenderam o método MDL-CF para a categorização de textos multiclasse, chamando-o de MDLText. Nessa proposta, os autores incorporaram uma penalidade ao tamanho de descrição baseada na similaridade de cosseno, para aumentar a margem de separação entre as classes. No contexto avaliado, o MDLText foi considerado superior a diversos métodos da literatura e equivalente ao SVM. Esses estudos indicam que o princípio MDL oferece um bom equilíbro entre o poder preditivo e a robustez contra *overfitting* e ainda, pode ser empregado em cenários de aprendizado *online* (SILVA; ALMEIDA; YAMAKAMI, 2017).

Inspirados pelos bons resultados obtidos pelo MDLText, Silva, Almeida e Yamakami (2016) e Freitas, Silva e Almeida (2020) também propuseram versões de métodos de classificação multiclasse genéricos (podem ser aplicados em outros problemas de classificação, além de categorização de texto) baseados no princípio MDL. Silva, Almeida e Yamakami (2016) propuseram o MDLClass, um método de classificação que permite manipular atributos contínuos ou categóricos através de discretizações. Apesar de ter vantagens sobre o MDLText na aplicação em problemas não-textuais, a necessidade da discretização o impede de ser aplicado em cenários de aprendizado incremental. Já no método proposto por Freitas, Silva e Almeida (2020), chamado de Aprendiz de Descritores de Mistura Gaussiana (GMDL - *Gaussian Mixture Descriptor Learner*), o tamanho da descrição é computado de forma que o processo de discretização se torna desnecessário. Nesse método, o estimador oKDE (KRISTAN; LEONARDIS; SKOČAJ, 2011) foi empregado para gerar estimativas de densidade de distribuições para computar o tamanho de descrição do princípio MDL. Os resultados do GMDL apontam benefícios ao incorporar esse princípio na classificação e, por ter a característica de ser naturalmente incremental, tem a vantagem de ser flexível e aplicável tanto em cenários de classificação *offline*, quanto *online* (FREITAS; SILVA; ALMEIDA, 2020).

Apesar da eficiência comprovada e da quantidade de aplicações, os trabalhos que envolvem o princípio MDL no aprendizado multirrótulo são raros na literatura. Laghmari, Marsala e Ramdani (2018), por exemplo, usaram o princípio MDL como uma estratégia de pré-corte para evitar o efeito de *overfitting* em árvores de decisão binárias, aplicadas na classificação multirrótulo. Contudo, não se tem conhecimento sobre métodos que usam o princípio MDL como principal estratégia para a classificação multirrótulo.

Pelos benefícios apresentados nas diversas aplicações do princípio MDL, esta dissertação avalia a hipótese de que é possível empregá-lo também na tarefa de classificação multirrótulo. Com base no notável desempenho apresentado pelo MDLText e suas variantes, nesta dissertação é proposta uma extensão do MDLText, chamada ML-MDLText, para

lidar com problemas de categorização de textos multirrótulo. A seguir, é apresentada a descrição do método MDLText para servir de referência para o entendimento da adaptação proposta.

3.3 MDLText

O método MDLText (SILVA; ALMEIDA; YAMAKAMI, 2017) emprega o princípio MDL para a seleção de modelos na categorização de textos. Neste método, as classes possíveis do problema são vistas como os modelos do princípio MDL. Dado um documento d não rotulado, é calculado o tamanho da descrição L, que corresponde à codificação de d por cada modelo (classe) c_j do problema. A classe de predição é aquela cujo modelo obtiver o menor tamanho de descrição, formalmente descrita pela equação:

$$y = \arg\min_{c_j \in \mathcal{Q}} L(d|c_j), \tag{3.6}$$

onde $\mathcal{Q} = \{c_1, c_2, ..., c_q\}$ é o conjunto de classes do problema, que são avaliadas como os modelos do princípio MDL. A Equação 3.6 é uma aplicação direta da Equação 3.4 do princípio MDL, que leva em conta a expressão relativa ao tamanho de descrição do documento d quando codificado com cada modelo c_j (SILVA, 2017).

A Figura 10 ilustra as etapas de treinamento e classificação do MDLText. De maneira geral, a etapa de treinamento do MDLText é responsável por construir os modelos das classes do problema, enquanto que a etapa de classificação é responsável por obter o tamanho de descrição $L(d|c_j)$ de cada modelo e determinar a classe mais adequada para d. Elas são detalhadas nas seções subsequentes.

3.3.1 Etapa de treinamento

A etapa de treinamento do MDLText é responsável por obter as informações listadas abaixo, relacionadas à frequência dos termos nos documentos de treinamento pertencentes à cada uma das classes, que compõem os modelos de predição:

- $\mathcal{V}$: vocabulário ou lista dos termos que aparecem nos documentos de treinamento;
- $n_{c_j,\forall t_i} = \sum_{\forall d_r \in \mathcal{T}} \hat{w}(t_i, d_r|c_j)$: somatório dos pesos TF-IDF de cada termo t_i presente nos documentos de treinamento da classe c_j;
- $\hat{n}_{c_j} = \sum_{\forall d_r \in \mathcal{T}} \hat{w}(:, d_r|c_j)$: somatório de todos os termos que estão presentes na classe c_j;
- $|\hat{\mathcal{T}}_{c_j}|$: quantidade de documentos de treinamento da classe c_j;

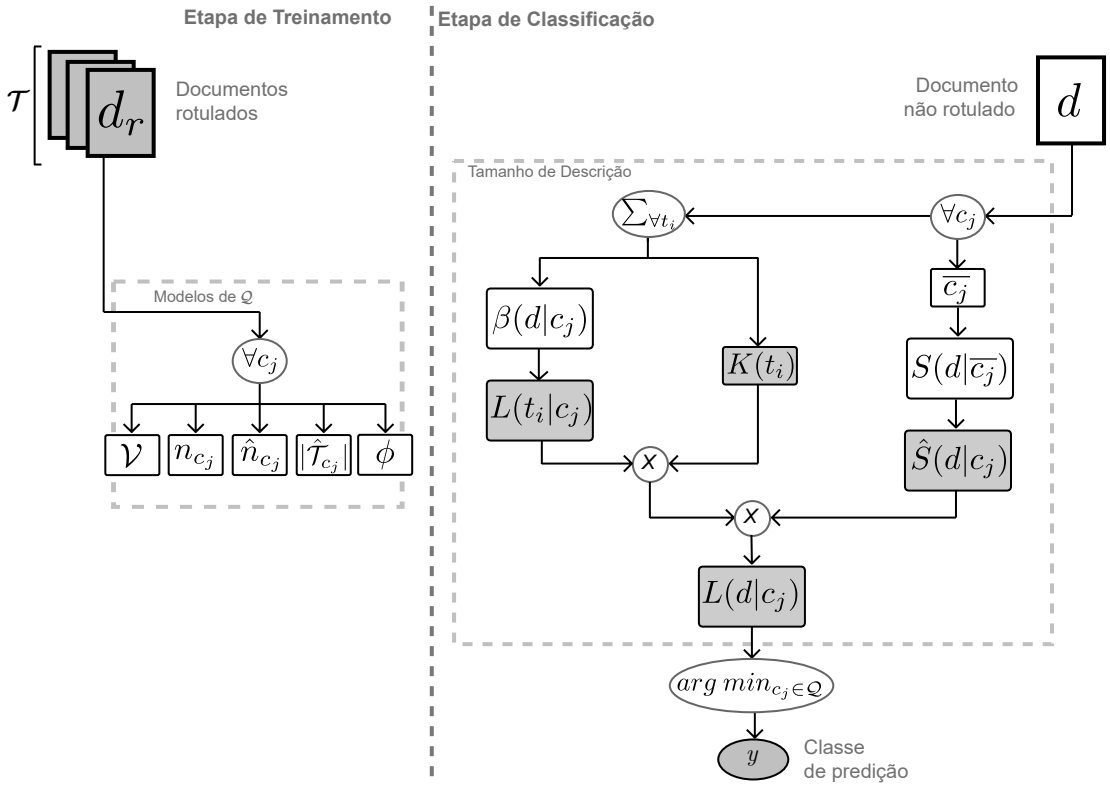


Figura 10 – Etapas de treinamento e classificação do MDLText.

- ϕ: quantidade de documentos de cada classe em que cada um dos termos do vocabulário ocorre.

Como os modelos são compostos de informações estatísticas relacionadas aos termos e aos documentos das classes, a etapa de treinamento desse método pode ser realizada de forma incremental, o que permite que ele possa ser empregado para a classificação em cenários de aprendizado *online* (SILVA; ALMEIDA; YAMAKAMI, 2017).

3.3.2 Etapa de classificação

Na Equação 3.6, o tamanho de descrição do documento d ao ser codificado com c_j é obtido através da seguinte equação:

$$L(d|c_j) = \left[\sum_{i=1}^{|d|} L(t_i|c_j) \times K(t_i)\right] \times \hat{S}(d, c_j) \tag{3.7}$$

Como mostra a Equação 3.7 e a Figura 10, o cálculo do tamanho da descrição $L(d|c_j)$ é constituído de três partes:

- $L(t_i|c_j)$: tamanho da descrição de cada termo quando este é representado pelo modelo c_j;

- $K(t_i)$: peso atribuído à importância do termo t_i para a definição das classes;
- $\hat{S}(d, c_j)$: penalidade baseada na similaridade de cosseno entre o documento e o vetor protótipo do modelo.

A seguir, estas três partes são descritas de forma detalhada, de acordo com a definição de Silva, Almeida e Yamakami (2017).

3.3.2.1 O tamanho da descrição dos termos

O tamanho da descrição de cada termo $L(t_i|c_j)$ foi inspirado na codificação Shannon-Fano (SHANNON, 1948) e pode ser obtida pela Equação 3.8:

$$L(t_i|c_j) = \lceil -log_2 \beta(t_i, c_j) \rceil \tag{3.8}$$

$$\beta(t_i, c_j) = \frac{n_{c_j,t_i} + \frac{1}{|\Omega|}}{\hat{n}_{c_j} + 1}, \tag{3.9}$$

onde n_{c_j,t_i} representa a soma dos pesos TF-IDF normalizados do termo t_i nos documentos de treinamento pertencentes à classe c_j, a expressão $\hat{n}_{c_j}$ representa a soma dos pesos de todos os termos dos documentos do modelo c_j e $|\Omega|$ é um parâmetro utilizado para definir um tamanho de descrição para os termos que nunca apareceram nos documentos de treinamento do modelo.

O valor de β varia entre zero e um. Como no problema de categorização de textos, grande parte dos termos aparecem em poucos documentos de treinamento, os valores de β normalmente ficam mais próximos de zero do que de um. Já $L(t_i|c_j)$ varia dentre os números naturais. Análoga a Equação 3.3, existe uma correspondência entre o valor de β e o tamanho $L(t_i|c_j)$: valores baixos de β geram altos tamanhos de descrição para o termo t_i e vice-versa.

Como abordado por Silva, Almeida e Yamakami (2017), o parâmetro $|\Omega|$ regula como os termos que nunca apareceram nos documentos de treinamento da classe c_j contribuem no tamanho de descrição. Um valor alto desse parâmetro proporciona altos valores de descrição para termos que não estão presentes nos documentos ($n_{c_j,t_i} = 0$) das classes que contém um alto peso $\hat{n}_{c_j}$. Os autores recomendam realizar uma busca em grade para definir o valor desse parâmetro de forma apropriada.

3.3.2.2 A relevância dos termos

O propósito da expressão $K(t_i)$ é atribuir pesos diferenciados para os termos de acordo com suas importâncias para determinar as classes do problema. Seja t_i um termo que aparece em muitos documentos da classe c_j e em poucos documentos pertencentes as demais classes. Como a presença do termo t_i é importante para determinar a classe c_j,

o valor de $K(t_i)$ será alto na equação do tamanho de descrição de todos os modelos. No entanto, como t_i aparece em muitos documentos da classe c_j, o valor de $L(t_i|c_j) \times K(t_i)$ será muito menor do que o valor de $L(t_i|c_z) \times K(t_i)$ das demais classes $c_z \in \mathcal{Q}$. Logo, o termo $K(t_i)$ é usado para melhorar a habilidade de detecção das classes pelo classificador (SILVA, 2017). Ele é formulado pela equação:

$$K(t_i) = \frac{1}{(1+\mu) - F(t_i)}, \tag{3.10}$$

onde μ é uma constante para evitar problemas de indeterminação no cálculo da divisão e $F(t_i)$ corresponde a uma função de pontuação de relevância do termo t_i, que leva em consideração a frequência dos termos nos documentos de treinamento.

Silva, Almeida e Yamakami (2017) empregaram fatores de confidência (CF – *Confidence Factors*) (ASSIS et al., 2006) como função de pontuação de relevância de termos. No entanto, qualquer outra função de pontuação pode ser utilizada, desde variem no intervalo de zero a um. Segundo a Equação 3.10, o valor de $K(t_i)$ varia conforme o valor de $F(t_i)$. Valores de $F(t_i)$ próximos de um geram valores reais positivos maiores para $K(t_i)$, limitados a $1/\mu$.

3.3.2.3 A penalidade baseada na similaridade de cosseno

O termo $\hat{S}(d, c_j)$ da Equação 3.7 é uma penalização aplicada para aumentar a margem de separação entre os modelos e é baseada na similaridade de cosseno entre o documento e o vetor protótipo do modelo c_j. Quanto menos similar for o documento e o modelo c_j, maior será a penalização e, consequentemente, maior será o tamanho da descrição necessário para representar o documento. O termo $\hat{S}(d, c_j)$ pode ser obtido através da equação:

$$\hat{S}(d, c_j) = -log_2\left(\frac{1}{2} \times S(d, \overline{c}_j)\right) \tag{3.11}$$

A expressão $S(d, \overline{c}_j)$ corresponde a similaridade de cosseno entre o documento d e o vetor protótipo da classe c_j:

$$S(d, \overline{c}_j) = \frac{\sum_{i=1}^{|d|} \hat{w}(t_i, d|c_j) \times \overline{c}_j(t_i)}{||\hat{w}(:, d)||_2 \times ||\overline{c}_j||_2}, \tag{3.12}$$

onde $\hat{w}(t_i, d|c_j)$ representa o peso TF-IDF do termo t_i nos documentos de treinamento do modelo c_j. O MDLText calcula um vetor protótipo para cada uma das classes, formado pela média dos pesos dos termos que apareceram nos documentos de treinamento da classe. Na Equação 3.12, $c_j(t_i)$ é o valor do vetor protótipo da classe c_j correspondente ao termo t_i e $||\overline{c}_j||_2$ é a norma do vetor protótipo da classe c_j. O valor de $c_j(t_i)$ pode ser obtido por:

$$\overline{c}_j(t_i) = \frac{n_{c_j, t_i}}{|\hat{\mathcal{T}}_{c_j}|}, \tag{3.13}$$

onde $|\hat{\mathcal{T}}_{c_j}|$ é o número de documentos de treinamento pertencentes à classe c_j. Já a norma do vetor protótipo da classe c_j pode ser obtida por:

$$||\overline{c}_j||_2 = \sqrt{\sum_{\forall t \in \mathcal{V}} \overline{c_j}(t)^2}, \tag{3.14}$$

onde $\mathcal{V}$ é o vocabulário de termos de $\mathcal{T}$.

O valor de $S(d, \overline{c}_j)$ varia entre zero e um e valores próximos de um indicam uma grande similaridade entre o documento d e o vetor protótipo da classe. As classes, cujos protótipos são mais similares ao documento avaliado, geram um valor baixo de $\hat{S}(d, c_j)$ e, consequentemente, são menos penalizadas no tamanho de descrição. O valor de $\hat{S}(d, c_j)$ varia entre um e $+\infty$.

3.3.3 Análise assintótica

Com relação a complexidade computacional do método MDLText, dado um conjunto de treinamento $\mathcal{T}$ com uma média de $\bar{n}$ termos por documento, a etapa de treinamento possui complexidade de ordem linear $\mathcal{O}(|\mathcal{T}| \times \bar{n})$. Já na etapa de classificação, a complexidade varia também conforme o número de classes, porém ela ainda é linear, na ordem de $\mathcal{O}(|\mathcal{Q}| \times n)$, com n sendo o número de termos do documento a ser classificado (SILVA, 2017).

3.3.4 Exemplo

Para exemplificar e facilitar o entendimento do MDLText na prática, considere um conjunto de treinamento $\mathcal{T} = \{(d_1, y_1), (d_2, y_2), ..., (d_6, y_6)\}$ contendo seis mensagens rotuladas com uma das classes do conjunto binário $\mathcal{Q} = \{spam, ham\}$ e uma mensagem d sem classe definida, como ilustra a Figura 11. Nessa figura, as mensagens aparecem convertidas para a representação espaço-vetorial com peso TF-IDF normalizado e a expressão $\hat{w}(t_i, d_r)$ representa o valor do peso TF-IDF do termo t_i no documento d_r.

O objetivo neste exemplo é escolher uma das classes presentes em $\mathcal{Q}$ para rotular d. Segundo o MDLText, a classe mais apropriada é aquela que obtiver o menor tamanho de descrição ao codificar d, conforme a equação abaixo:

$$y = \arg\min \left[L(d|spam), L(d|ham)\right] \qquad \text{(Equação 3.4)}$$

A seguir, são descritos os passos executados para calcular os tamanhos de descrição $L(d|spam)$ e $L(d|ham)$, de acordo com a etapa de treinamento e classificação do MDLText.

Etapa de Treinamento: nesta etapa, são coletadas as informações de frequência dos termos nas mensagens do conjunto $\mathcal{T}$ que compõe os modelos das classes (criação dos modelos usados para a predição):

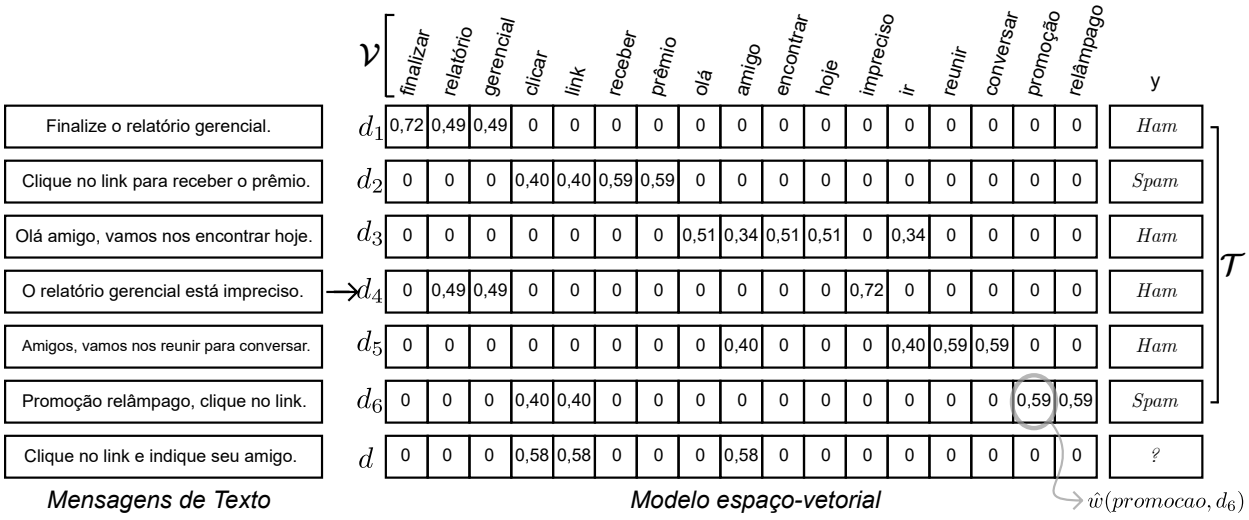


Figura 11 – Base de dados usada no exemplo de aplicação do MDLText.

- Computar $n_{c_j,t_i} = \sum_{\forall d_r \in \mathcal{T}} \hat{w}(t_i, d_r|c_j)$: soma dos pesos $\hat{w}$ dos termos *clicar*, *link* e *amigo* que aparecem nos documentos de treinamento das classes *spam* e *ham*:

t_i	n_{spam,t_i}	n_{ham,t_i}
clicar	$0,40 + 0,40 = 0,80$	0
link	$0,40 + 0,40 = 0,80$	0
amigo	0	$0,40 + 0,34 = 0,74$

- Computar $\hat{n}_{c_j} = \sum_{\forall d \in \mathcal{T}} \hat{w}(:, d|c_j)$: soma dos pesos $\hat{w}$ de todos os termos que aparecem nos documentos de treinamento das classes *spam* e *ham*:

$$\hat{n}_{spam} = 3,96 \qquad \hat{n}_{ham} = 7,59$$

- Computar $|\hat{\mathcal{T}}_{c_j}|$: quantidade de documentos de treinamento pertencentes as classes *spam* e *ham*:

$$|\hat{\mathcal{T}}_{spam}| = 2 \qquad |\hat{\mathcal{T}}_{ham}| = 4$$

Etapa de classificação: nesta etapa, é calculado o tamanho da descrição de d quando codificado por ambos os modelos (*spam* e *ham*) e o tamanho de descrição de d quando codificado com a combinações mais apropriadas. O modelo da classe que obtiver o menor tamanho é escolhido como a classe da mensagem:

- Computar $L(t_i|c_j) = \lceil -log_2 \beta(t_i|c_j) \rceil$ (Equação 3.8): tamanho de descrição de cada termo da mensagem d quando codificada com cada uma das classes *spam* e *ham* (considerar $|\Omega| = 2^{10}$):

$$\beta(t_i, c_j) = \frac{n_{c_j,t_i} + \frac{1}{|\Omega|}}{\hat{n}_{c_j} + 1} \text{ (Equação 3.9)}$$

t_i	$\beta(t_i, spam)$	$\beta(t_i, ham)$
clicar	$\frac{0,80+\frac{1}{2^{10}}}{3,96+1} = 0,16$	$\frac{0+\frac{1}{2^{10}}}{7,59+1} = 1,14 \times 10^{-4}$
link	$\frac{0,80+\frac{1}{1,97^{10}}}{3,96+1} = 0,16$	$\frac{0+\frac{1}{2^{10}}}{7,59+1} = 1,14 \times 10^{-4}$
amigo	$\frac{0+\frac{1}{2^{10}}}{3,96+1} = 1,97 \times 10^{-4}$	$\frac{0,74+\frac{1}{2^{10}}}{7,59+1} = 0,09$

$$\boldsymbol{L(t_i|c_j)}$$

t_i	$L(t_i\|spam)$	$L(t_i\|ham)$
clicar	$\lceil -log_2(0,16) \rceil = 3$	$\lceil -log_2(1,14 \times 10^{-4}) \rceil = 14$
link	$\lceil -log_2(0,16) \rceil = 3$	$\lceil -log_2(1,14 \times 10^{-4}) \rceil = 14$
amigo	$\lceil -log_2(1,97 \times 10^{-4} \rceil = 13$	$\lceil -log_2(0,09) \rceil = 4$

- Computar $K(t_i) = \frac{1}{(1 + 10^{-3}) - F(t_i)}$ (Equação 3.10): relevância de cada termo t_i da mensagem (considerar $\mu = 10^{-3}$ e fatores de confidência (CF) como função de relevância para o cálculo de $F(t_i)$):

t_i	$K(t_i)$
clicar	$\frac{1}{(1+10^{-3})-F(clicar)} = \frac{1}{(1+10^{-3})-0,15} = 1,18$
link	$\frac{1}{(1+10^{-3})-F(link)} = \frac{1}{(1+10^{-3})-0,15} = 1,18$
amigo	$\frac{1}{(1+10^{-3})-F(amigo)} = \frac{1}{(1+10^{-3})-0,15} = 1,18$

- Computar $\hat{S}(d, c_j) = -log_2\left(\frac{1}{2} \times S(d, \overline{c}_j)\right)$ (Equação 3.11): penalidade baseada na similaridade de cosseno de cada uma das classes *spam* e *ham*:

$$\overline{c}_j(t_i) = \frac{n_{c_j,t_i}}{|\hat{\mathcal{T}}_{c_j}|} \text{ (Equação 3.13)}$$

t_i	$\overline{spam}(t_i)$	$\overline{ham}(t_i)$
clicar	$\frac{n_{clicar,spam}}{\lvert\hat{\mathcal{T}}_{spam}\rvert} = \frac{0,80}{2} = 0,40$	$\frac{n_{clicar,ham}}{\lvert\hat{\mathcal{T}}_{ham}\rvert} = \frac{0}{4} = 0$
link	$\frac{n_{link,spam}}{\lvert\hat{\mathcal{T}}_{spam}\rvert} = \frac{0,80}{2} = 0,40$	$\frac{n_{link,ham}}{\lvert\hat{\mathcal{T}}_{ham}\rvert} = \frac{0}{4} = 0$
amigo	$\frac{n_{amigo,spam}}{\lvert\hat{\mathcal{T}}_{spam}\rvert} = \frac{0}{2} = 0$	$\frac{n_{amigo,ham}}{\lvert\hat{\mathcal{T}}_{ham}\rvert} = \frac{0,74}{4} = 0,18$

$$||\overline{c_j}||_2 = \sqrt{\sum_{\forall t \in \mathcal{V}} \overline{c_j}(t)^2} \text{ (Equação 3.14)}$$

$$||\overline{spam}||_2 = \sqrt{\overline{spam}(finalizar)^2 + ... + \overline{spam}(relampago)^2} = 0,81$$

$$||\overline{ham}||_2 = \sqrt{\overline{ham}(finalizar)^2 + ... + \overline{ham}(relampago)^2} = 0,59$$

$$S(d, \overline{c}_j) = \frac{\sum_{i=1}^{|d|} \hat{w}(t_i, d|c_j) \times \overline{c}_j(t_i)}{||\hat{w}(:,d)||_2 \times ||\overline{c}_j||_2} \text{ (Equação 3.12)}$$

$$S(d, \overline{spam}) = \frac{0,58 \times 0,40 + 0,58 \times 0,40}{1 \times 0,81} = 0,57 \qquad S(d, \overline{ham}) = \frac{0,58 \times 0,18}{1 \times 0,59} = 0,18$$

$$\hat{\boldsymbol{S}}(\boldsymbol{d}, \boldsymbol{c_j})$$

$$\hat{S}(d, spam) = -log_2\left(\tfrac{1}{2} \times 0,57\right) = 1,84 \qquad \hat{S}(d, ham) = -log_2\left(\tfrac{1}{2} \times 0,18\right) = 3,47$$

- Computar $L(d|c_j) = \left[\sum_{i=1}^{|d|} L(t_i|c_j) \times K(t_i)\right] \times \hat{S}(d, c_j)$ (Equação 3.7): por fim, é calculado o tamanho de descrição $L(d|c_j)$ de cada uma das classes *spam* e *ham*:

$$L(d|spam) = \left[\sum_{i=1}^{|d|} L(t_i|spam) \times K(t_i)\right] \times \hat{S}(d, spam) = [3 \times 1,18 + 3 \times 1,18 + 16 \times 1,18] \times 1,84 = 47,77$$

$$L(d|ham) = \left[\sum_{i=1}^{|d|} L(t_i|ham) \times K(t_i)\right] \times \hat{S}(d, ham) = [14 \times 1,18 + 14 \times 1,18 + 4 \times 1,18] \times 3,47 = 131,03$$

- $y = \arg\min_{c_j \in \mathcal{Q}} L(d|c_j)$ (Equação 3.4): escolha da classe com menor tamanho de descrição:

$$y = \arg\min\left[L(d|spam), L(d|ham)\right] = spam$$

3.4 Considerações finais

Os principais conceitos da compressão de dados relacionados ao princípio MDL foram introduzidos neste capítulo. Conforme foi discutido, as regularidades encontradas em uma sequência podem ser usadas para descrevê-la de forma mais compacta e, portanto,

a capacidade de compressão de uma sequência pode ser associada à capacidade de aprendizado. Dessa forma, os conceitos relacionados à compressão de dados também podem ser utilizados para solucionar os problemas de inferência indutiva.

O princípio MDL já foi aplicado na resolução de problemas de diversas áreas. Porém, apesar de mostrar vantagens da sua aplicação em tarefas de categorização de rótulo único, não se tem conhecimento sobre métodos que empregam o princípio MDL na classificação multirrótulo.

Este capítulo também teve como objetivo identificar e discutir as principais motivações que levaram a extensão do método MDLText. O MDLText é um método de seleção de modelos baseado no princípio MDL que apresentou um bom desempenho em diversos problemas de categorização de textos monorrótulo. Adicionalmente, a etapa de treinamento pode ser executada de forma incremental, o que permite sua aplicação em problemas de larga escala, característica importante e pouco explorada na classificação multirrótulo. A base matemática desse método e um exemplo de aplicação também foram apresentados neste capítulo com o intuito de auxiliar o entendimento do cálculo do tamanho de descrição empregado na adaptação proposta nesta dissertação.

4 O método ML-MDLText

Conforme detalhado na Seção 3.3 e de acordo com Silva, Almeida e Yamakami (2017), no método MDLText, as possíveis classes do problema de classificação são tratadas como modelos independentes no princípio MDL. Assim, a classe de um documento d não rotulado é definida como aquela que é capaz de descrever o documento de forma mais compacta, ou seja, é a que obtém o menor tamanho de descrição ao codificar d. O tamanho da descrição de cada modelo na codificação de um documento é definido por:

$$L(d|c_j) = \left[\sum_{i=1}^{|d|} L(t_i|c_j) \times K(t_i)\right] \times \hat{S}(d, c_j), \tag{4.1}$$

sendo que:

1. $L(t_i|c_j)$ é um peso condicional de cada termo presente no documento d relativo a todos os outros termos que já apareceram em documentos da classe c_j;

2. $K(t_i)$ é um peso diferenciado por termo do documento baseado em quanto cada termo pode ajudar na identificação das classes;

3. $\hat{S}(d|c_j)$ é uma penalidade baseada na similaridade de cosseno entre o documento d que está sendo classificado e o protótipo da classe c_j.

Essa formulação, no entanto, não pode ser empregada diretamente para manipular problemas multirrotulados, já que ela implica na escolha de uma única classe como saída. Para utilizar o MDLText na categorização multirrótulo, seria necessário empregá-lo em conjunto com técnicas de transformação. Como discutido no Capítulo 2, BR e LP são duas técnicas muito utilizadas com métodos de classificação monorrótulo para oferecer uma saída multirrotulada. A primeira, transforma o problema multirrótulo em diversos problemas binários, e a segunda, transforma o problema multirrótulo em um problema multiclasse, assumindo que os diferentes *labelsets* são as classes do problema. Apesar de serem populares, a técnica BR é constantemente criticada por não considerar a dependência existente entre os rótulos, enquanto que a técnica LP costuma gerar problemas multiclasse com alto desbalanceamento.

Neste capítulo, é apresentada uma proposta de adaptação da formulação do método MDLText, denominada ML-MDLText, para manipular os problemas de categorização de textos multirrótulo diretamente. A principal ideia dessa proposta é processar os documentos de cada rótulo, como é feito no método MDLText e BR, e, ao mesmo tempo, usar informações relacionadas às ocorrências simultâneas de diferentes rótulos nos documentos

de treinamento, como no método LP, para explorar a correlação entre eles. Dessa forma, para gerar a predição multirrótulo, o método usa as informações das classes e da dependência que pode existir entre elas, sem empregar truques de transformação.

Neste capítulo, são apresentadas as adaptações realizadas no MDLText para torná-lo capaz de realizar a classificação multirrótulo. A princípio, é apresentada uma síntese do ML-MDLText. Em seguida, todos os passos, os algoritmos das etapas de treinamento e classificação e um exemplo de aplicação são detalhados para um melhor entendimento.

4.1 Visão geral

A etapa de treinamento do ML-MDLText é responsável por extrair informações dos dados de treinamento, como os pesos e frequência de cada termo para cada classe e para cada *labelset*, e por gerar os modelos respectivos. Na etapa de classificação, essas informações são usadas para predizer o *labelset* de um documento desconhecido. Dado um documento d não rotulado, o ML-MDLText prediz o seu conjunto de rótulos baseado no tamanho de descrição de cada classe possível e de cada *labelset* candidato. Os *labelsets* candidatos são aqueles que contêm as classes mais relevantes para o documento d e cuja quantidade de rótulos presentes no *labelset* satisfaz uma estimativa definida por um metamodelo e por funções gaussianas. Para facilitar o entendimento do método, os procedimentos executados nas etapas de treinamento e de classificação do ML-MDLText foram agrupados nos seguintes quatro processos, ilustrados na Figura 12 e que são descritos a seguir.

Processo 1 (detalhes são fornecidos na Seção 4.2):

> ***Etapa de treinamento***: são criados os modelos das classes da mesma forma que no MDLText, de acordo com os documentos pertencentes a cada classe de $\mathcal{Q}$ individualmente. Além disso, é definido o número cm de classes com menor tamanho de descrição que devem ser consideradas como relevantes para definir o *labelset* de saída.
>
> ***Etapa de classificação***: a relevância de cada classe é definida de acordo com o tamanho de descrição do documento d quando codificado com o modelo de cada classe $c_j \in \mathcal{Q}$, descrita como $L(d|c_j)$. Quanto menor o tamanho de descrição, mais relevante a classe c_j é para d. As cm classes mais relevantes, compõem um conjunto $\mathcal{R}$ de classes, que são selecionadas para definir os *labelsets* candidatos.

Processo 2 (detalhes são fornecidos na Seção 4.3):

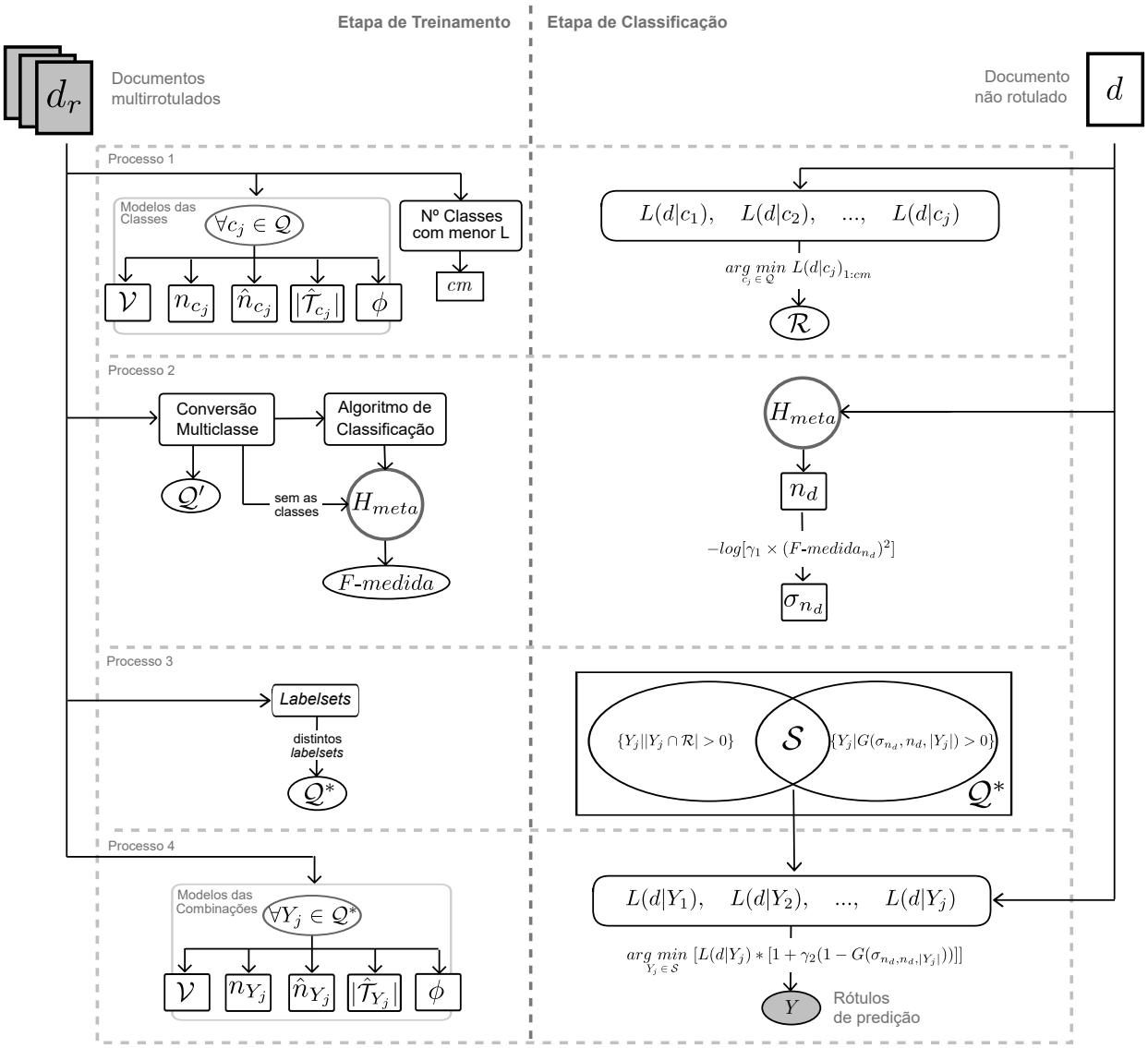


Figura 12 – Processos executados no ML-MDLText.

Etapa de treinamento: é criado um conjunto de classes $\mathcal{Q}'$, cujas classes correspondem aos diferentes tamanhos de *labelsets* que estão atribuídos às amostras de treinamento. O conjunto de dados multirrótulo é então transformado em um metaconjunto multiclasse, rotulado com as classes pertencentes a $\mathcal{Q}'$, de acordo com tamanho do *labelset* associado a cada amostra. Esse novo metaconjunto é utilizado para treinar um metamodelo e para avaliá-lo na classificação. O metamodelo é descrito como H_{meta} na Figura 12. De acordo com os resultados do metamodelo, obtidos na classificação das amostras de treinamento, é calculado a *F-medida* para cada classe de $\mathcal{Q}'$.

Etapa de classificação: o tamanho do *labelset* de d é estimado pelo metamodelo. Essa estimativa é descrita como n_d na Figura 12. Como o metamodelo pode não

ser preciso na predição do tamanho do *labelset*, uma função gaussiana G é definida de acordo com a *F-medida* da classe (tamanho do *labelset* estimado) e ela é usada para determinar a confiança nas predições do metamodelo. Através dessa função, a influência do metamodelo é definida de forma proporcional a confiança das suas predições.

Processo 3 (detalhes são fornecidos na Seção 4.4):

Etapa de treinamento: um conjunto de classes $\mathcal{Q}^*$ é criado, contendo os distintos *labelsets* existentes nos dados de treinamento.

Etapa de classificação: são selecionados os *labelsets* candidatos de $\mathcal{Q}^*$, que contêm pelo menos uma das classes relevantes definidas no *Processo 1* e cujo tamanho do *labelset* é igual ao definido pelo metamodelo ou que está no intervalo definido pela função gaussiana G do *Processo 2*. Esses *labelsets* compõem o conjunto S.

Processo 4 (detalhes são fornecidos na Seção 4.5):

Etapa de treinamento: a partir dos *labelsets* de $\mathcal{Q}^*$, os modelos de *labelsets* são construídos, baseado na ocorrência desses *labelsets* dentre as amostras de treinamento.

Etapa de classificação: a relevância de cada *labelset* é definida pelo tamanho de descrição do documento d quando codificado com cada modelo dos *labelsets* $Y_j \in \mathcal{S}$, descrita como $L(d|Y_j)$. O modelo criado para cada *labelset* detecta os padrões existentes entre as amostras de treinamento e as diversas combinações de múltiplos rótulos, e por isso, esse processo permite explorar a dependência ou a correlação existente entre os rótulos na predição das classes. O *labelset* com menor tamanho de descrição, é selecionado como o *labelset* de d, descrito como Y.

Cada um desses processos são detalhados nas seções a seguir. Para facilitar a compreensão, os algoritmos e os códigos-fonte para os estágios de treinamento e classificação do ML-MDLText também são apresentados.

4.2 Processo 1: Classes com menor tamanho de descrição

Inicialmente, o tamanho da descrição é usado para medir a relevância de c_j em relação a d. Para isso, as classes são usadas como modelos na equação do tamanho de descrição. O tamanho de descrição corresponde ao tamanho da codificação gerado por cada classe $c_j \in \mathcal{Q}$ ao codificar d e foi definido por Silva, Almeida e Yamakami (2017) de acordo com a Equação 4.1.

No método proposto, uma versão modificada da função CF (ASSIS et al., 2006) foi empregada para calcular K usando dados multirrotulados. As modificações foram

propostas para obter pontuações adequadas para termos que melhor definem os distintos *labelsets* presentes nos exemplos de treinamento. A equação CF modificada é a seguinte:

$$F(t_i) = \frac{1}{(|\mathcal{Q}^*|-1)} \times \sum_{\forall_{Y_j \in \mathcal{Q}^* \mid Y_j \neq Y_\tau}} \frac{\left[\frac{\left(\phi_{Y_\tau,t_i} - \phi_{Y_j,t_i}\right)^2 + \phi_{Y_\tau,t_i} \times \phi_{Y_j,t_i} - \frac{\lambda_1}{\phi_{\tau,t_i} + \phi_{Y_j,t_i}}}{\left(\phi_{Y_\tau,t_i} + \phi_{Y_j,t_i}\right)^2}\right]^{\lambda_2}}{1 + \left(\frac{\lambda_3}{\phi_{Y_\tau,t_i} + \phi_{Y_j,t_i}}\right)}, \quad (4.2)$$

onde τ é o índice do *labelset* mais frequente, j representa os índices dos *labelsets* presentes nas amostras de treinamento, ϕ_{Y_τ,t_i} é o número de documentos que contém o termo t_i e que pertence a Y_τ e ϕ_{Y_j,t_i} é o número de documentos que contém o termo t_i e que pertencem a Y_j. Nesta equação, λ_1, λ_2 e λ_3 são constantes que ajustam a velocidade de decaimento do fator de confidência. Neste trabalho, foram utilizados os mesmos valores propostos originalmente por Assis et al. (2006), que são $\lambda_1 = 0,25$, $\lambda_2 = 10$ e $\lambda_3 = 8$.

Como mencionado, o tamanho de descrição é usado para estimar a relevância das classes. Baseado nessa medida, o ML-MDLText seleciona as primeiras *cm* classes (em ordem ascendente de tamanho de descrição) como candidatas a compor o *labelset* de d. O valor de *cm* é definido como o menor número de posições de *ranking* na qual pelo menos uma classe faça parte do *labelset* verdadeiro da amostra, considerando uma porcentagem de $p_{treino}\%$ de amostras de treinamento. Por exemplo, assumindo que as classes são ordenadas por ordem ascendente de tamanho de descrição em cada amostra de treinamento, se a primeira classe é parte do *labelset* de no mínimo $p_{treino}\%$ das amostras de treinamento, então $cm = 1$. Caso contrário, se pelo menos uma entre as classes que ocupam as duas primeiras posições do *ranking* faça parte do *labelset* de no mínimo $p_{treino}\%$ amostras de treinamento, então $cm = 2$. Senão, o mesmo procedimento é executado considerando as classes que ocupam as três primeiras posições do *ranking* e assim por diante. Nesse trabalho, empiricamente foi empregado $p_{treino} = 90\%$ das amostras de treinamento para a definição de *cm*. Posteriormente, com *cm* definido na etapa de treinamento, as classes relevantes para o documento não rotulado d podem ser formalmente definidas por:

$$\mathcal{R} = \arg\min_{c_j \in \mathcal{Q}} [L(d|c_j)]_{1:cm}, \quad (4.3)$$

Os seguintes passos detalham a execução desse processo:

1. Na etapa de treinamento, são criados os modelos das classes que são constituídos das informações estatísticas abaixo, relacionadas à presença dos termos nos documentos de treinamento de cada uma das classes:

   - $\mathcal{V}$: vocabulário ou lista dos termos que aparecem nos documentos de treinamento;

- $n_{c_j} = \sum_{\forall_{d_r \in \mathcal{T}}} \hat{w}(t_i, d_r|c_j)$: somatório dos pesos TF-IDF de cada termo t_i presente nos documentos de treinamento da classe c_j;
- $\hat{n}_{c_j} = \sum_{\forall_{d_r \in \mathcal{T}}} \hat{w}(:, d_r|c_j)$: somatório de todos os termos que estão presentes na classe c_j;
- $|\hat{\mathcal{T}}_{c_j}|$: quantidade de documentos de treinamento da classe c_j.
- ϕ (para o cálculo de K): quantidade de documentos de cada *labelset* em que cada um dos termos do vocabulário ocorre.

2. Ainda na etapa de treinamento, é calculado o tamanho de descrição $L(d_i|c_j)$ de cada amostra de treinamento quando codificado com cada classe. Iniciando pela classe que obteve o menor tamanho de descrição em todas as amostras, é avaliado se ela faz parte do conjunto de rótulos verdadeiro em, pelo menos, $p_{treino}\%$ das amostras. Se sim, a quantidade mínima *cm* é definida como 1. Senão, é observado se pelo menos uma entre as duas classes que obtiveram o menor tamanho de descrição faz parte conjunto de rótulos verdadeiro em $p_{treino}\%$ das amostras. Se fizer, a quantidade mínima *cm* é definida como 2, senão, o mesmo processo é executado considerando as três classes com menores tamanhos de descrição e assim por diante, até que a regra seja satisfeita.

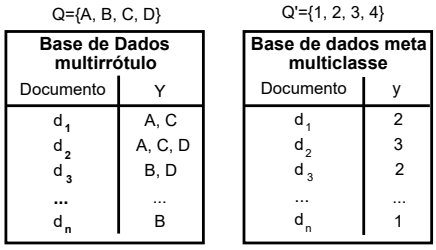
3. Na etapa de classificação, são escolhidas as *cm* classes que obtiveram o menor tamanho de descrição ao codificar d, para selecionar os *labelsets* mais prováveis.

4.3 Processo 2: Definição do tamanho do conjunto de classes

Para identificar o conjunto de classes de um documento não rotulado, pode-se considerar apenas possível atribuir um *labelset* que contém um número de rótulos pré-determinado ou mais provável. Contudo, umas das incógnitas na classificação multirrótulo é justamente o tamanho do conjunto de predição, que é variável de acordo com o problema e com o documento que está sendo classificado.

Para estimar o tamanho provável do *labelset* de uma amostra, foi empregado um metamodelo, como o proposto no trabalho de Tang, Rajan e Narayanan (2009). Para a construção desse metamodelo, os dados originais multirrótulo são convertidos para um conjunto de dados multiclasse. Nesta conversão, os rótulos originais são descartados e a nova classe de cada documento passa a ser o número de rótulos associados a ele no problema multirrótulo original, como mostra a Figura 13. Nessa figura, o conjunto de classes original $\mathcal{Q}$ é substituído pelo novo conjunto de classes $\mathcal{Q}' = \{a_1, a_2, ..., a_{q'}\}$, onde q' é o índice do tamanho do maior *labelset* encontrado nas amostras de treinamento.

Baseada nessa nova base de dados, é construída uma função hipótese que mapeia os atributos de cada documento para suas novas classes. Uma escolha natural seria empregar



Fonte: Tang, Rajan e Narayanan (2009, p. 213).

Figura 13 – Construção do metaconjunto multiclasse com as informações da quantidade de rótulos de cada exemplo.

um método de regressão. Contudo, segundo Tang, Rajan e Narayanan (2009), determinar o tamanho do conjunto de rótulos com a regressão pode gerar um impasse, já que ela retorna um número real e o tamanho de um *labelset* qualquer é sempre um número inteiro. Por esta razão, Tang, Rajan e Narayanan (2009) optaram pela classificação, no qual cada número inteiro representa uma categoria ou rótulo do metaconjunto criado. No Apêndice A, são descritos os experimentos que foram realizados para determinar a melhor técnica para estimar o número de rótulos de predição, utilizada nos experimentos reportados nesta dissertação.

Após sua construção, o metamodelo pode ser utilizado para predizer o tamanho provável do *labelset* do documento d através da escolha de uma das classes de $\mathcal{Q}'$. No entanto, como ele foi construído através de técnicas de classificação, seus resultados nem sempre são precisos e, portanto, se o algoritmo confiar apenas em suas predições, ele poderá ignorar *labelsets* que se aproximam mais do conjunto de rótulos mais adequado para o documento. Com o intuito de considerar outros tamanhos de *labelsets*, foram criadas curvas gaussianas, baseadas na Equação 4.4, para cada classe de $\mathcal{Q}'$, de acordo com a capacidade preditiva do metamodelo. Essa capacidade preditiva é avaliada de acordo com a F-medida obtida na predição das classes nas amostras de treinamento:

$$G(\sigma_{n_d}, n_d, a_j) = e^{\frac{-(a_j - n_d)^2}{2\sigma_{n_d}^2}}, \tag{4.4}$$

$$\sigma_{n_d} = -log_2\Big[\gamma_1 \times (F\text{-}medida_{n_d})^2\Big], \tag{4.5}$$

onde n_d é a classe estimada pelo metamodelo (*i.e.*, tamanho do *labelset* da amostra), a_j é a classe de índice j pertencente a $\mathcal{Q}'$ e $F\text{-}medida_{n_d}$ é a F-medida obtida pelo metamodelo quando avaliado na classificação das amostras de treinamento pertencentes à classe n_d. Nessa equação, o termo σ_{n_d} define a abertura da curva gaussiana baseada na F-medida. Quanto menor for o valor da $F\text{-}medida_{n_d}$, maior será o valor de σ_{n_d} e maior será a abertura da curva gaussiana. Adicionalmente, γ_1 também é usado para expandir essa

abertura. Nesse trabalho, foi empregado empiricamente $\gamma_1 = 0{,}95$ para indicar que outros tamanhos de *labelsets* devem ser considerados quando o metamodelo predizer classes com $F\text{-}medida \leq 0{,}95$.

A Figura 14 ilustra o comportamento das curvas gaussianas geradas por cada classe de $\mathcal{Q}'$ de uma base de dados qualquer. O eixo x representa a diferença entre as predições do metamodelo e as classes verdadeiras das amostras de treinamento, correspondendo ao termo $(a_j - n_d)$ na Equação 4.4. O eixo y corresponde ao valor da função G (Equação 4.4) de acordo com a F-medida obtida por cada classe.

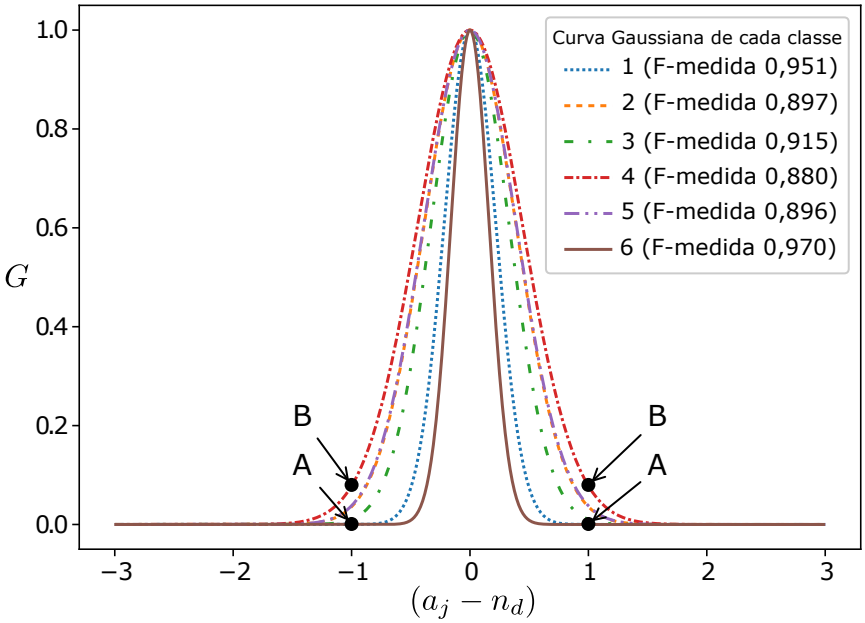


Figura 14 – Dispersão da saída do metamodelo com relação a saída verdadeira e a curva gaussiana gerada para cada classe da base de dados.

Na Figura 14, a F-medida obtida pelo metamodelo quando classificou as amostras da classe 6 foi alta ($F\text{-}medida = 0{,}970$) e, portanto, a função gaussiana do segundo gráfico chega a zero antes de alcançar algum número inteiro, como indicam os pontos A. Nesta situação, se o metamodelo apontar a classe 6 ao classificar um documento desconhecido, por exemplo, é definido que a predição final irá conter apenas um rótulo. Por outro lado, o metamodelo não obteve a mesma capacidade preditiva ao estimar o tamanho do *labelset* da classe 4 ($F\text{-}medida = 0{,}880$). Assim, sua função gaussiana retorna valores maiores que zero para classes com diferença de um da estimativa do metamodelo, conforme indicam os pontos B. Isso significa que uma margem de segurança será usada para que o tamanho do *labelset* não esteja totalmente definido pela predição do metamodelo. Nesse caso, a função aponta a possibilidade da predição correta conter de 3 a 5 rótulos ($4-1$, 4 e $4+1$, respectivamente), ou seja, se o metamodelo apontar a classe 4 ao classificar um documento desconhecido, a predição final poderá conter 3, 4 ou 5 rótulos.

Os seguintes passos resumem a execução desse processo:

1. Na etapa de treinamento, é construído um metaconjunto de dados multiclasse com os mesmos dados de observação, porém com novas classes que referem-se ao número de rótulos associados na base de dados original. Um metamodelo é treinado com esse novo conjunto para predizer essas novas classes em documentos não rotulados.

2. Ainda na etapa de treinamento, o metamodelo é avaliado ao predizer a classe das amostras de treinamento do metaconjunto de dados através da F-medida.

3. Na etapa de classificação, o metamodelo prediz o tamanho provável do *labelset* de d e são definidos os parâmetros para gerar uma função gaussiana G. A abertura dessa curva gaussiana indica o grau de confiança da predição desse tamanho de conjunto e a necessidade de considerar outros tamanhos de *labelsets*.

4.4 Processo 3: Seleção do grupo de *labelsets*

Nesse processo, é criado um conjunto $\mathcal{S}$ dos *labelsets* Y_j que contêm pelo menos uma das classes de $\mathcal{R}$ (como definido no Processo *Processo 1*) e, de acordo com a quantidade de rótulos presentes no *labelset*, que devem satisfazer a desigualdade $G(\sigma_{n_d}, n_d, |Y_j|) > 0$ (com os parâmetros da curva definidos no Processo *Processo 2*).

Os seguintes passos descrevem a execução desse processo:

1. Na etapa de treinamento, são extraídos os distintos *labelsets* que estão associados às amostras de treinamento para compor um conjunto $\mathcal{Q}^*$.

2. Na etapa de classificação, são selecionados os *labelsets* $Y_j \in \mathcal{Q}^*$ que satisfazem a desigualdade $G(\sigma_{n_d}, n_d, |Y_j|) > 0$ e que contêm pelo menos uma das classes de $\mathcal{R}$. Esses *labelsets* formam o conjunto $\mathcal{S}$.

4.5 Processo 4: Identificação do conjunto de rótulos de predição

Para identificar o conjunto de rótulos de predição do documento, o tamanho de descrição é calculado novamente, dessa vez de acordo com os *labelsets* presentes no grupo $\mathcal{S}$, ou seja, os *labelsets* são usados como modelos e $L(d|Y_j)$ é calculado para cada $Y_j \in \mathcal{S}$. Além disso, *labelsets* com tamanho muito diferente do predito pelo metamodelo são penalizados de acordo com o valor da função gaussiana. Essa penalização visa evitar *labelsets* de tamanhos muito distantes daquele que foi predito pelo metamodelo.

Os rótulos que estão presentes no *labelset* cujo modelo obteve o menor tamanho de descrição são escolhidos para a predição, definida pela seguinte equação:

$$Y = \underset{\forall_{Y_j \in S}}{\arg\min} \left[L(d|Y_j) \times [1 + \gamma_2(1 - G(\sigma_{n_d}, n_d, |Y_j|))] \right], \tag{4.6}$$

onde j é o índice do *labelsets* presente em $\mathcal{S}$ e γ_2 é o peso da penalidade máxima aplicada ao tamanho de descrição. Nessa dissertação, empiricamente foi definido $\gamma_2 = 0{,}2$, ou seja, 20% do tamanho de descrição é aplicada como a penalidade máxima para *labelsets* com tamanho diferente do valor que foi predito pelo metamodelo.

Os seguintes passos detalham a execução desse processo:

1. Na etapa de treinamento, são criados os modelos dos *labelsets* que são constituídos das informações estatísticas abaixo, relacionadas à presença dos termos nos documentos de treinamento de cada um dos *labelsets*:

   - $\mathcal{V}$: vocabulário ou lista dos termos que aparecem nos documentos de treinamento;
   - $n_{Y_j,\forall t_i} = \sum_{\forall_{d_r \in \mathcal{T}}} \hat{w}(t_i, d_r|Y_j)$: somatório dos pesos TF-IDF de cada termo t_i presente nos documentos de treinamento do *labelset* Y_j;
   - $\hat{n}_{Y_j} = \sum_{\forall_{d_r \in \mathcal{T}}} \hat{w}(:, d_r|Y_j)$: somatório de todos os termos que estão presentes nos documentos do *labelset* Y_j;
   - $|\hat{\mathcal{T}}_{Y_j}|$: quantidade de documentos de treinamento do *labelset* Y_j.
   - ϕ (para o cálculo de K): quantidade de documentos de cada *labelset* em que cada um dos termos do vocabulário ocorre.

2. Na etapa de classificação, é calculado o tamanho de descrição $L(d_i|Y_j)$ do documento d para cada modelo de *labelset* $Y_j \in \mathcal{S}$ e o valor da função G de acordo com o tamanho de Y_j. O conjunto de rótulos de predição Y do documento d é o *labelset* que satisfaz a Equação 4.6.

4.6 Algoritmos e análise assintótica

Os processos executados nas etapas de treinamento e classificação do ML-MDLText estão sumarizados nos Algoritmos 1 e 2. Para facilitar a reprodução dos experimentos, o código fonte do ML-MDLText está disponível publicamente no `GitHub`[1].

Nos dois algoritmos, o modelo de classificação H é composto pelas variáveis $\mathcal{Q}$ (conjunto de classes), $\mathcal{Q}^*$ (distintos *labelsets*), $\mathcal{Q}'$ (conjunto de classes do metamodelo,

[1] *Código fonte do ML-MDLText* na linguagem de programação Python. Disponível em <http://github.com/m-bittencourt/ML-MDLText> (accessed on 12/09/2019).

ou seja, o conjunto dos tamanhos dos *labelsets*), H_{meta} (metamodelo) e pelos valores de F-medida obtidos na classificação das amostras de treinamento. O modelo de classificação também contém as frequências de termos e documentos para cada classe e para cada *labelset*. Essas variáveis, necessárias para calcular o comprimento da descrição das classes e dos *labelsets*, são entradas opcionais na etapa de treinamento. Dessa forma, o modelo pode ser atualizado com as informações de novos documentos de treinamento. Como as informações relacionadas aos termos, às classes e aos *labelsets* são extraídas de forma independente, o ML-MDLText também permite que novos termos e novas classes sejam apresentados ao longo do tempo. Essa é uma característica marcante e importante para cenários de aprendizado *online* em aplicações reais.

O pior cenário possível para o ML-MDLText executar é aquele composto por uma base de treinamento que contém todos os *labelsets* possíveis (ou seja, $|\mathcal{Q}^*| = 2^{|\mathcal{Q}|}$ e consequentemente, $|\mathcal{Q}'| = |\mathcal{Q}|$), por um valor de cm definido bem próximo de $|\mathcal{Q}|$ e por um metamodelo que tem baixo poder de predição na definição do tamanho dos *labelsets*. Essa configuração obriga avaliar todos os $2^{|\mathcal{Q}|}$ *labelsets* possíveis através do tamanho de descrição do último processo do ML-MDLText. Na prática, este cenário deve ocorrer muito raramente e, portanto, a complexidade média do ML-MDLText é bem melhor do que a apresentada no pior caso.

Na etapa de treinamento do ML-MDLText, os modelos das classes e dos *labelsets* e o metamodelo são construídos. Para gerar os modelos de predição de cada classe e de cada *labelset*, o algoritmo apresenta a complexidade $\mathcal{O}(|\mathcal{T}| \times \overline{n} \times |\mathcal{Q}|)$, onde $\overline{n}$ é o número médio de termos nos documentos de treinamento. A construção do metamodelo e a definição dos parâmetros da curva gaussiana apresentam complexidade computacional $\mathcal{O}(f_{treinamento}(|\mathcal{T}|, \overline{n}, |\mathcal{Q}'|) + f_{classificacao}(|\mathcal{T}|, \overline{n}, |\mathcal{Q}'|))$, onde $f_{treinamento}$ e $f_{classificacao}$ são as complexidades apresentadas pelo método empregado como metamodelo para treinar e classificar um conjunto com $|\mathcal{T}|$ amostras, com uma média de $\overline{n}$ atributos por amostra e $|\mathcal{Q}'|$ classes. O método apresenta ainda a complexidade $\mathcal{O}(|\mathcal{Q}^*| \times |\mathcal{T}| \times |\overline{n}| + |\mathcal{T}| \times |\mathcal{Q}| \log |\mathcal{Q}|)$ para o cálculo dos tamanhos de descrição de cada amostra de treinamento presente em $|\mathcal{T}|$ e na ordenação das classes de acordo com os tamanhos obtidos, para definir o valor de cm. De maneira geral, a complexidade do ML-MDLText na etapa de treinamento, no pior caso de execução, é $\mathcal{O}((2^{|\mathcal{Q}|} \times |\mathcal{T}| \times \overline{n}) + f_{treinamento}(|\mathcal{T}|, \overline{n}, |\mathcal{Q}|) + f_{classificacao}(|\mathcal{T}|, \overline{n}, |\mathcal{Q}|))$, que varia significativamente conforme o número de classes e de *labelsets* aumenta e de acordo com a complexidade do método empregado para gerar o metamodelo.

Na etapa de classificação, o cálculo do tamanho de descrição do documento d e a seleção das cm classes com menor tamanho tem a complexidade computacional $\mathcal{O}(|\mathcal{Q}^*| \times |d|)$. Para a definição de n_d e das curvas gaussianas, é exigida complexidade $\mathcal{O}((f_{classificacao}(1, n, |\mathcal{Q}'|))$. Por fim, a complexidade para a criação do conjunto de *labelsets* $\mathcal{S}$ e para o cálculo do tamanho de descrição de cada modelo é $\mathcal{O}(|\mathcal{Q}^*| \times |d|)$. Portanto,

a complexidade do método na etapa de classificação no pior caso é $\mathcal{O}((2^{|\mathcal{Q}|} \times |d|) + f_{classificacao}(1, n, |\mathcal{Q}|))$, que também varia conforme o número de classes e de acordo com o método empregado para gerar o metamodelo.

É importante destacar que como na grande maioria dos problemas de classificação, o número de classes é bem menor do que o número de amostras e o número médio de termos, pode-se dizer que a etapa de treinamento do ML-MDLText, nesses casos, possui complexidade linear em relação ao número médio de termos e amostras, se o método de classificação usado para gerar o metamodelo também tiver complexidade linear. De maneira similar, a complexidade da etapa de classificação é linear em relação a quantidade de termos presente no documento, se o metamodelo também tiver complexidade linear nessa etapa.

4.7 Exemplo

Para exemplificar e facilitar o entendimento do ML-MDLText na prática, considere um conjunto de treinamento $\mathcal{T} = \{(d_1, Y_1), (d_2, Y_2), ..., (d_6, Y_6)\}$ contendo seis mensagens rotuladas com uma ou mais classes do conjunto $\mathcal{Q} = \{spam, propaganda, corrente\}$ e uma mensagem d sem classe definida, como ilustra a Figura 11. Nessa figura, as mensagens aparecem convertidas para a representação espaço-vetorial com peso TF-IDF normalizado e a expressão $\hat{w}(t_i, d_r)$ representa o valor do peso TF-IDF do termo t_i no documento d_r.

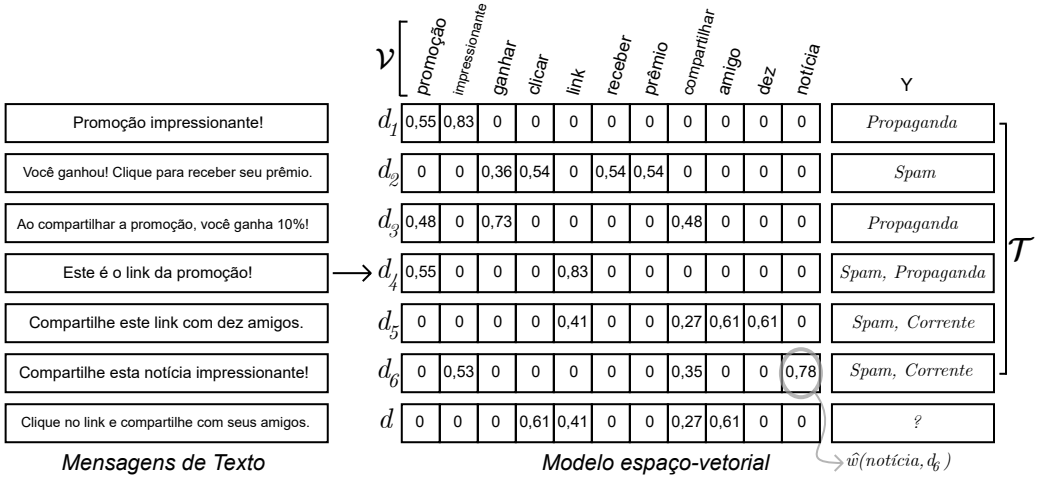


Figura 15 – Base de dados usada no exemplo de aplicação do ML-MDLText.

O objetivo do ML-MDLText é escolher um conjunto de classes presentes em $\mathcal{Q}$ para rotular d. A seguir, são descritos os passos executados.

Algoritmo 1 Etapa de treinamento do ML-MDLText

Entradas: $\mathcal{T}$ (conjunto de treinamento), $\mathcal{Q}$ (conjunto de classes), H (modelo de predição - opcional)
Saídas: H (modelo de predição atualizado)

Início
    INSTANCIAMODELO($\mathcal{Q}$) ▷ Inicialize os parâmetros do método
    $\mathcal{D}, Y \longleftarrow \mathcal{T}$ ▷ Separe o conjunto de treinamento em um conjunto de amostras e outro de rótulos
    ▷ Processo 1 (Seção 4.2)
    ATUALIZAESTATISTICA($\mathcal{D}$, Y, $\mathcal{Q}$) ▷ Construa os modelos das classes
    $H.cm \longleftarrow$ número de classes relevantes ▷ Use os procedimentos da Seção 4.2 Item 2 para definir cm
    ▷ Processo 2 (Seção 4.3)
    $\boldsymbol{y} \longleftarrow |Y_i|_{\forall_{Y_i \in Y}}$ ▷ Atribua o tamanho do conjunto de rótulos a cada amostra
    $H.\mathcal{Q}' \longleftarrow H.\mathcal{Q}' \cup \{\text{elementos únicos de } \boldsymbol{y}\}$ ▷ Defina o conjunto de classes possíveis
    $H.H_{meta}.treina(\mathcal{D}, \boldsymbol{y})$ ▷ Treine o metamodelo
    ATUALIZAF-MEDIDA($\boldsymbol{y}$, $H.H_{meta}.classifica(\mathcal{D})$) ▷ Classifique as amostras de $\mathcal{T}$ e calcule a F-medida de cada classe
    ▷ Processo 3 (Seção 4.4)
    $H.\mathcal{Q}^* \longleftarrow H.\mathcal{Q}^* \cup \{\text{elementos únicos de } Y\}$ ▷ Defina os distintos *labelsets*
    ▷ Processo 4 (Seção 4.5)
    ATUALIZAESTATISTICA($\mathcal{D}$, Y, $H.\mathcal{Q}^*$) ▷ Construa os modelos dos *labelsets*
    devolve H
Fim

Algoritmo 2 Etapa de classificação do ML-MDLText

Entradas: d (documento não rotulado), $\mathcal{Q}$ (conjunto de classes), H (modelo de predição)
Saídas: Y (rótulos preditos)

Início
    ▷ Processo 1 (Seção 4.2)
    $\mathcal{R} \longleftarrow \underset{c_j \in \mathcal{Q}}{\arg\min}\ L(d|c_j)_{1:H.cm}$ ▷ Use a Eq. 4.3 para definir as cm classes com menor L
    ▷ Processo 2 (Seção 4.3)
    $n_d \longleftarrow H.H_{meta}.classifica(d)$ ▷ Obtenha o número de classes com a estimativa do metamodelo
    $\sigma_{n_d} \longleftarrow -log_2(\gamma_1 \times H.F\text{-}medida_{n_d})^2$ ▷ Use a Eq. 4.5 para definir o σ da função G
    ▷ Processo 3 (Seção 4.4)
    $\mathcal{S} \longleftarrow \{Y_i \in H.\mathcal{Q}^* \mid |Y_i \cap \mathcal{R}| > 0 \wedge G(\sigma_{n_d}, n_d, |Y_i|) > 0\ \}$ ▷ Use as classes de $\mathcal{R}$ e a Eq. 4.4 para definir o conjunto $\mathcal{S}$
    ▷ Processo 4 (Seção 4.5)
    se $\mathcal{S} \neq \emptyset$ **então**
        $Y \longleftarrow \underset{Y_j \in \mathcal{S}}{\arg\min}\ L(d, Y_j) \times [1 + \gamma_2(1 - G(\sigma_{n_d}, n_d, |Y_j|))]$ ▷ Se houver conjuntos para serem avaliados, use a Eq. 4.6 para escolher o melhor conjunto de classes para a predição
    senão
        $Y \longleftarrow \underset{c_j \in \mathcal{Q}}{\arg\min}\ L(d|c_j)_{1:n_d}$ ▷ Se não houver conjuntos para avaliar, as classes preditas são as n_d classes com menor L
    fim se
    devolve Y
Fim

Algoritmo 3 Funções Auxiliares

função INSTANCIAMODELO($\mathcal{Q}$ (conjunto de classes))
  se $H = \emptyset$ **então**
    $\mathcal{Q}' \longleftarrow \emptyset$ ▷ Instancie o conjunto de tamanho de *labelsets*
    $\mathcal{Q}^* \longleftarrow \emptyset$ ▷ Instancie o conjunto de distintos *labelsets*
    $cm \longleftarrow 1$ ▷ Inicie com 1 o número mínimo de classes
    $F\text{-}medida \longleftarrow \emptyset$ ▷ Inicie com 0 o valor da F-medida de cada classe
    $H_{meta} \longleftarrow$ novo classificador ▷ Instancie um classificador multiclasse
    $\mathcal{V} \longleftarrow \emptyset$ ▷ Instancie o vocabulário de termos
    $|\hat{\mathcal{D}}| \longleftarrow \emptyset$ ▷ Instancie o contador de documentos
    $n \longleftarrow \emptyset$ ▷ Instancie o acumulador de peso de termos
    $\hat{n} \longleftarrow \emptyset$ ▷ Instancie o acumulador de peso de termos
    $\phi \longleftarrow \emptyset$ ▷ Instancie o acumulador de frequência de termos
    $H \longleftarrow \mathcal{Q}'$, $\mathcal{Q}^*$, cm, $F\text{-}medida$, H_{meta}, $\mathcal{V}$, $|\hat{\mathcal{D}}|$, n, $\hat{n}$, ϕ
  fim se
fim função

função ATUALIZAESTATISTICA($\mathcal{D}$ (conjunto de documentos), $\mathcal{Y}$ (conjunto de rótulos associados), $\mathcal{C}$ (conjunto de classes ou *labelsets*))
  para $r \leftarrow 0$ **até** $|\mathcal{D}|$ **faça**
    para cada classe/*labelset* $c_j \in \mathcal{C}$ em Y_r **faça**
      $H.|\hat{\mathcal{D}}_{c_j}| \longleftarrow H.|\hat{\mathcal{D}}_{c_j}| + 1$ ▷ Acumule 1 no contador de documentos da classe c_j
      para cada termo t_i em d_r **faça**
        se $t_i \notin H.\mathcal{V}$ **então**
          $H.\mathcal{V} \longleftarrow H.\mathcal{V} \cup t_i$ ▷ Adicione o termo no vocabulário
          $H.n_{\forall_{c \in \mathcal{C}}, t_i} \longleftarrow 0$ ▷ Inicialize a soma dos pesos de t_i para cada classe de $\mathcal{C}$
          $H.\hat{n}_{\forall_{c \in \mathcal{C}}, t_i} \longleftarrow 0$ ▷ Inicialize a soma dos pesos para cada classe de $\mathcal{C}$
          $H.\phi_{\forall_{c \in \mathcal{C}}, t_i} \longleftarrow 0$ ▷ Inicialize a frequência de t_i para cada classe de $\mathcal{C}$
        fim se
        $H.n_{c_j,t_i} \longleftarrow H.n_{c_j,t_i} + \hat{w}(t_i, d_r)$ ▷ Acumule o peso de t_i para a classe c_j
        $H.\hat{n}_{c_j} \longleftarrow H.\hat{n}_{c_j} + \hat{w}(t_i, d_r)$ ▷ Acumule o peso de t_i em para a classe c_j
        $H.\phi_{c_j,t_i} \longleftarrow H.\phi_{c_j,t_i} + 1$ ▷ Acumule 1 na frequência do termo t_i para a classe c_j
      fim para
    fim para
  fim para
fim função

função L(d (documento não rotulado), c_j (classe ou *labelset*))
  $L(d|c_j) \longleftarrow 0$
  para cada termo t_i em d **faça**
    $\overline{c_j}(t_v) \longleftarrow \frac{H.n_{c_j,t_i}}{H.|\hat{D}_{c_j}|}$ ▷ Use a Eq. 3.13 para obter o protótipo da classe
    $\beta(t_i, c_j) \longleftarrow \frac{H.n_{c_j,t_i} + \frac{1}{|\Omega|}}{H.\hat{n}_{c_j,t_i} + 1}$ ▷ Use a Eq. 3.9 para obter o valor de β
    $L(t_i|c_j) \longleftarrow \lceil -log_2 \beta(t_i, c_j) \rceil$ ▷ Use a Eq. 3.8 para obter o tamanho de descrição do termo t_i
    $K(t_i) \longleftarrow \frac{1}{(1+\mu) - F(t_i)}$ ▷ Use a Eq. 4.2 para obter a penalidade baseada na relevância de t_i
    $L(d|c_j) \longleftarrow L(d|c_j) + L(t_i|c_j) \times K(t_v)$ ▷ Use a Eq. 3.7 para obter o trecho do cálculo
  fim para
  $S(d, \overline{c}_j) \longleftarrow \frac{\sum_{v=1}^{|d|} \hat{w}(t_v, d|c_j) \times \overline{c}_j(t_v)}{||\hat{w}(:, d)||_2 \times ||\overline{c}_j||_2}$ ▷ Use a Eq. 3.12 para calcular a similaridade de cosseno
  $\hat{S}(d, c_j) \longleftarrow -\log_2\left(\frac{1}{2} \times S(d, \overline{c}_j)\right)$ ▷ Use a Eq. 3.11 para obter a penalidade baseada na similaridade de cosseno
  $L(d|c_j) \longleftarrow \hat{S}(d, c_j) \times L(d|c_j)$ ▷ Use a Eq. 3.7 para obter o tamanho de descrição
  devolve $L(d|c_j)$
fim função

Etapa de Treinamento: nesta etapa, são coletadas as informações de frequência dos termos nas mensagens do conjunto $\mathcal{T}$ que compõe os modelos das classes e dos distintos *labelsets*, bem como é definido o número de classes relevantes e a construção do metamodelo, conforme descrito nos tópicos a seguir:

- Computar n_{c_j,t_i}, $\hat{n}_{c_j}$ e $|\hat{\mathcal{T}}_{c_j}|$ para cada classe $c_j \in \mathcal{Q} = \{spam, propaganda, corrente\}$ (construção dos modelos das classes, conforme descrito na Seção 3.3.1).

$$n_{c_j,t_i} = \sum_{\forall_{d_r \in \mathcal{T}}} \hat{w}(t_i, d_r|c_j)$$

t_i	n_{spam,t_i}	$n_{propaganda,t_i}$	$n_{corrente,t_i}$
clicar	0,54	0	0
link	1,24	0,83	0,41
compartilhar	0,62	0,48	0,62
amigo	0, 61	0	0, 61

$$\hat{n}_{c_j} = \sum_{\forall_{d \in \mathcal{T}}} \hat{w}(:, d|c_j)$$

$n_{spam} = 6{,}92$ $\quad n_{propaganda} = 4{,}45$ $\quad n_{corrente} = 3{,}56$

$$|\hat{\mathcal{T}}_{c_j}|$$

$|\hat{\mathcal{T}}_{spam}| = 4$ $\quad |\hat{\mathcal{T}}_{propaganda}| = 3$ $\quad |\hat{\mathcal{T}}_{corrente}| = 2$

- Verificar qual o número de classes com menor tamanho de descrição que, em $p_{treino} = 90\%$ das amostras de treinamento, pelo menos uma delas faz parte do conjunto de rótulos verdadeiro. Para isso, é calculado o tamanho de descrição de cada amostra quando codificado por cada modelo da classe (usam-se os valores de n_{c_j,t_i}, $\hat{n}_{c_j}$ e $|\hat{\mathcal{T}}_{c_j}|$ calculados anteriormente). Abaixo, o menor tamanho de descrição de cada documento é destacado em negrito e os tamanhos de descrição que pertencem as classes atribuídas ao documento aparecem sublinhados:

t_i	$L(d_r\|spam)$	$L(d_r\|propaganda)$	$L(d_r\|corrente)$
d_1	21,68	**7,62**	50,59
d_2	**36,56**	169,36	572,93
d_3	33,36	**14,41**	101,34
d_4	12,73	**7,62**	56,83
d_5	24,98	106,38	**19,14**
d_6	25,04	50,02	**14,82**

É possível notar que a classe com o menor tamanho de descrição sempre está no conjunto de rótulos verdadeiro das amostras de treinamento (*i.e.*, todos os valores em negrito estão sublinhados em 100% dos exemplos), logo $cm = 1$. No entanto, caso essa classe não estivesse presente em pelo menos 90% das amostras, seria necessário observar se essa condição seria satisfeita observando as duas classes com os menores tamanhos ($cm = 2$), e assim por diante, até que fosse encontrado um número de classes com tamanho mínimo de descrição (*i.e.*, um valor de cm) que atendesse essa condição.

- Converter o conjunto de dados multirrótulo para multiclasse e, depois, treinar e avaliar o metamodelo (nesse exemplo, o método SGD foi empregado para gerar o metamodelo):

  Na transformação, os atributos das amostras de treinamento são mantidos, porém as novas classes atribuídas a cada amostra apontam o tamanho do conjunto multirrótulo original, como mostra a Figura 16. O novo conjunto de classes possíveis é definido como $\mathcal{Q}' = \{1, 2\}$. Logo após, estes dados são apontados para o treinamento do metamodelo SGD e estes mesmos dados são utilizados para avaliá-lo na classificação. Os resultados da classificação pelo metamodelo aparecem à direita da Figura 16.

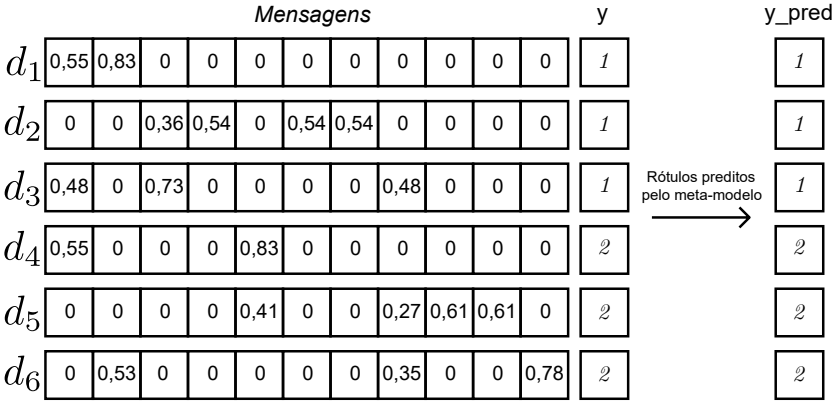


Figura 16 – Metaconjunto com as classes relativas ao tamanho do conjunto de rótulos original. À direita são apresentados os rótulos preditos pelo metamodelo.

- Computar a *F-medida* de cada classe $a_j \in \mathcal{Q}'$:

$$F\text{-}medida_{a_j} = \frac{2 \times TP_{a_j}}{2TP_{a_j} + FN_{a_j} + FP_{a_j}}$$

$a_j = 1$ $\qquad$ $a_j = 2$

$F\text{-}medida_1 = \frac{2 \times 3}{2 \times 3 + 0 + 0} = 1$ $\qquad$ $F\text{-}medida_2 = \frac{2 \times 3}{2 \times 3 + 0 + 0} = 1$

- Obter o conjunto dos distintos *labelsets* $\mathcal{Q}^*$ que estão presentes em $\mathcal{T}$ e computar n_{Y_j,t_i}, $\hat{n}_{Y_j}$ e $|\hat{\mathcal{T}}_{Y_j}|$ para cada *labelset* $Y_j \in \mathcal{Q}^*$ (*i.e.*, construir os modelos dos *labelsets*, conforme descrito na Seção 3.3.1).

  $\mathcal{Q}^* = \{propaganda, spam, spam\text{-}corrente, spam\text{-}propaganda\}$.

$$n_{Y_j,t_i} = \sum_{\forall_{d_r \in \mathcal{T}}} \hat{w}(t_i, d_r|Y_j)$$

t_i	$n_{propaganda,t_i}$	n_{spam,t_i}	$n_{spam-corrente,t_i}$	$n_{spam-propaganda,t_i}$
clicar	0	0,54	0	0
link	0	0	0,41	0,83
compartilhar	0,48	0	0,62	0
amigo	0	0	0,61	0

$$\hat{n}_{Y_j} = \sum_{\forall_{d \in \mathcal{T}}} \hat{w}(:, d|Y_j)$$

$n_{propaganda} = 3{,}07$ $n_{spam} = 1{,}98$ $n_{spam-corrente} = 3{,}56$ $n_{spam-propaganda} = 1{,}38$

$$|\hat{\mathcal{T}}_{Y_j}|$$

$|\hat{\mathcal{T}}_{propaganda}| = 2$ $|\hat{\mathcal{T}}_{spam}| = 1$ $|\hat{\mathcal{T}}_{spam\text{-}corrente}| = 2$ $|\hat{\mathcal{T}}_{spam\text{-}propaganda}| = 1$

  Etapa de classificação: nesta etapa, é calculado o tamanho da descrição de d quando codificado pelos modelos das classes $\mathcal{Q}$ e dos *labelsets* mais prováveis pertencentes a $\mathcal{Q}^*$. O modelo do *labelset* que obtiver o menor tamanho é escolhido como o conjunto de classes da mensagem.

- Computar $L(d|c_j)$ (Equação 4.1) para os modelos das classes, compostos por n_{c_j,t_i}, $\hat{n}_{c_j}$ e $|\hat{\mathcal{T}}_{c_j}|$:

$L(d|spam) = 25{,}71$ $L(d|propaganda) = 106{,}38$ $L(d|corrente) = 47{,}75$

  Como $cm = 1$, apenas a primeira classe com menor tamanho de descrição, *i.e.*, *spam*, foi escolhida para ajudar a obter os *labelsets* candidatos.

- Predizer o número de classes n_d com o metamodelo e definir o valor de σ_{n_d} através da função $\sigma_{n_d} = \log_2(\gamma_1 \times F\text{-}medida_{a_j})^2$ (Equação 4.5):

  $n_d = 2$ (classe predita pelo metamodelo)

  $\sigma_2 = \log_2(0{,}95 \times 1)^2 = 0{,}15$

- Selecionar os *labelsets* $Y_j \in \mathcal{Q}^*$ que contém a classe *spam* e que $G(\sigma_{n_d}, n_d, a_j) > 0$ (Equação 4.4) para compor o conjunto S:

$G(\sigma_2, 2, |spam|) = e^{\frac{(1-2)^2}{2\times 0{,}15^2}} = 0$

$G(\sigma_2, 2, |spam\text{-}corrente|) = e^{\frac{(2-2)^2}{2\times 0{,}15^2}} = 1$

$G(\sigma_2, 2, |spam\text{-}propaganda|) = e^{\frac{(2-2)^2}{2\times 0{,}15^2}} = 1$

$S = \{spam\text{-}corrente, spam\text{-}propaganda\}$.

- Computar $L(d|Y_j) \times [1 + \gamma_2(1 - G(\sigma_{n_d}, n_d, |Y_j|))]$ (Equação 4.6) com os modelos dos *labelsets* $Y_i \in S$, que são compostos por n_{Y_j,t_i}, $\hat{n}_{Y_j}$ e $|\hat{\mathcal{T}}_{Y_j}|$ (considerar $\gamma_2 = 0{,}2$):

  $L(d|spam\text{-}corrente) = 47{,}75$

  $L(d|spam\text{-}propaganda) = 99{,}55$

  $L(d|spam\text{-}corrente) \times [1 + \gamma_2(1 - G(\sigma_2, 2, G(\sigma_2, 2, |spam\text{-}propaganda|))]$

  $= 47{,}75 \times [1 + 0{,}2 \times (1 - 1)] = 47{,}75$

  $L(d|spam\text{-}propaganda) \times [1 + \gamma_2(1 - G(\sigma_2, 2, |G(\sigma_2, 2, |spam\text{-}propaganda|))]$

  $= 99{,}55 \times [1 + 0{,}2 \times (1 - 1)] = 99{,}55$

- Computar $y = \arg\min_{Y_j \in \mathcal{S}} L(d|Y_j) \times [1 + \gamma_2(1 - G(\sigma_{n_d}, n_d, |Y_j|))]$ (Equação 4.6) para determinar o *labelset* com menor tamanho de descrição:

  O *labelset*, cujo modelo obteve o menor tamanho de descrição é *spam-corrente*. Portanto, o conjunto predito de classes de d é $\{spam, corrente\}$.

4.8 Considerações finais

Neste capítulo, foram descritas e detalhadas todas as etapas e processos realizados pelo ML-MDLText, método proposto nesta dissertação. O ML-MDLText é um método incremental e voltado especialmente para problemas de categorização de textos multirrótulo. Como o ML-MDLText utiliza tanto informações da ocorrência de cada classe quanto da ocorrência simultânea de classes nos documentos, é possível explorar a dependência que pode existir entre elas na classificação.

O método realiza quatro processos principais para determinar o conjunto de classes de um documento não rotulado. Inicialmente, são determinadas as classes com os menores tamanhos de descrição e o possível tamanho do conjunto de rótulos verdadeiro. Em seguida, são escolhidos os distintos *labelsets* de rótulos que ocorreram na base de dados, que contém alguma das classes com os menores tamanhos de descrição e que satisfazem o tamanho do conjunto definido anteriormente. Finalmente, esses *labelsets* são avaliados e as classes pertencentes ao *labelset* cujo modelo obteve o menor tamanho de descrição, são definidas como os rótulos de predição.

5 Avaliação experimental

Esta seção apresenta todos os procedimentos experimentais executados para avaliar o desempenho do método proposto, as bases de dados utilizadas, as medidas de desempenho empregadas e toda a metodologia experimental executada. Nessa seção, os resultados dos experimentos também são apresentados e discutidos.

5.1 Base de dados e pré-processamento

Para explorar a capacidade do método ao lidar com problemas de classificação multirrótulo, foram selecionados diversos conjuntos textuais com diferentes características que podem influenciar no desempenho dos algoritmos e que são amplamente empregados nos trabalhos relacionados (GIBAJA; VENTURA, 2015; ZHANG; GRAEPEL; HERBRICH, 2010; WU et al., 2016; SCHAPIRE; SINGER, 2000; MCCALLUM, 1999; MENCÍA; FÜRNKRANZ, 2008b; CRAMMER; SINGER, 2003; ZHANG; ZHOU, 2006; TANG; RAJAN; NARAYANAN, 2009; MADJAROV et al., 2012; GONZALEZ-LOPEZ; VENTURA; CANO, 2018; TSOUMAKAS; KATAKIS; VLAHAVAS, 2008). A Tabela 2 lista os 15 conjuntos de dados usados nos experimentos.

As colunas m, n, q e ls nessa tabela apontam a quantidade de exemplos de treinamento, a dimensão do espaço de atributos, número de classes e número de distintos *labelsets* do problema, respectivamente. Essas colunas conseguem dar uma visão da dificuldade do problema em relação a escolha do método de classificação devido a sua complexidade. Já as colunas *min.*, *med.* e *max.* da Classe apontam, respectivamente, a quantidade mínima, média e máxima de exemplos pertencentes a cada classe e as colunas *min.*, *med.* e *max.* do *Labelset* apontam, respectivamente, a quantidade mínima, média e máxima de exemplos pertencentes a cada distinto *labelset*; juntas, essas colunas fornecem uma visualização do desbalanceamento da base de dados. Por fim, o grau da rotulação dos conjuntos de dados pode ser observado através da cardinalidade e densidade, apontadas nas colunas *card.* e *dens.*. A cardinalidade é o número médio de rótulos por amostra, enquanto que a densidade é a cardinalidade dividida pelo número de rótulos, usada para comparar bases com diferentes números de rótulos. Essas duas medidas são definidas pela Equação 5.1 e pela Equação 5.2 e avaliam o quanto o conjunto de dados é multirrotulado (GIBAJA; VENTURA, 2015).

$$card. = \frac{1}{m}\sum_{i=1}^{m}|Y_i|, \tag{5.1}$$

$$dens. = \frac{card.}{q}, \tag{5.2}$$

onde Y_i é o conjunto de rótulos da amostra i.

Tabela 2 – Bases de dados utilizadas nos experimentos.

					Classe			*Labelset*				
Base de Dados	m	n	q	ls	min.	med.	max.	min.	med.	max.	card.	dens.
Reuters-ORGs	881	6.983	32	60	1	31	349	1	15	308	1,119	0,035
Reuters-places	18.798	34.169	147	889	1	155	12.541	1	21	10.879	1,212	0,008
RCV1-nivel1	27.974	64.286	4	15	4.150	8.169	12.752	16	1.865	9.943	1,168	0,292
RCV1-nivel2	27.047	61.439	54	1.239	7	714	4.753	1	22	4.133	1,425	0,026
RCV2-IT-nivel1	28.400	31.780	4	15	5.268	8.938	11.731	3	1.893	7.907	1,259	0,315
RCV2-IT-nivel2	27.954	31.176	43	356	1	824	4.477	1	79	3.499	1,267	0,029
RCV2-PT-nivel1	8.837	14.681	4	14	502	3.054	5.276	1	631	2.975	1,382	0,346
RCV2-PT-nivel2	8.773	14.582	37	218	1	383	2.937	1	40	1.603	1,613	0,044
RCV2-SP-nivel1	18.651	26.038	4	15	2.455	5.669	12.888	2	1.243	10.049	1,216	0,304
RCV2-SP-nivel2	18.463	25.551	44	284	2	569	6.475	1	65	5.109	1,357	0,031
Bibtex	7.395	1.836	159	2.856	51	112	1.042	1	3	471	2,402	0,015
Delicious	16.091	500	983	15.805	21	312	6.495	1	1	19	19,037	0,019
Enron	1.702	1.001	53	753	1	108	913	1	2	163	3,378	0,064
Medical	978	1.449	45	94	1	27	266	1	10	155	1,245	0,028
TMC2007	28.596	49.060	22	1.341	441	2.805	16.173	1	21	2.486	2,158	0,098

Os dois primeiros conjuntos foram extraídos da coleção *Reuters21578*[1]. Essa coleção é constituída de notícias do ano de 1987 e é muito utilizada em experimentos de classificação. O primeiro conjunto de dados contém documentos rotulados com as categorias de *ORGs* e o segundo contém documentos relacionados as categorias de *Places*.

Os seguintes oito conjuntos foram extraídos das coleções *RCV1* e *RCV2*[2], que também são compostas por notícias da agência *Reuters* e são consideradas referência na classificação de textos. De RCV1 foram extraídos dois conjuntos de documentos rotulados com as categorias de *Topics*: um do primeiro nível hierárquico e o outro do segundo. Além disso, foram selecionados apenas os documentos publicados entre 20 de agosto de 1996 e 4 de setembro de 1996. Já de RCV2, foram extraídos os documentos escritos nos idiomas italiano (IT), português (PT) e espanhol (SP), explorando também o primeiro e o segundo nível hierárquico.

Os documentos das bases de dados *Reuters21578*, *RCV1* e *RCV2* estavam em arquivos de formatos XML ou SGM. Para obter as informações relevantes ao experimento, as *tags* das categorias e do texto das notícias foram extraídas através da biblioteca

[1] A coleção *Reuters21578* original pode ser acessada em <http://www.daviddlewis.com/resources/testcollections/reuters21578/> (accessada em 13/02/2020).

[2] As coleções *RCV1* e *RCV2* originais podem ser acessadas em <https://trec.nist.gov/data/reuters/reuters.html> (acessada em 13/02/2020).

`Beautiful Soup`[3]. Todas as palavras dos textos foram convertidas para grafia em letras minúsculas e os caracteres de pontuação como ponto de interrogação, ponto final, dois-pontos, vírgula, dentre outros, foram substituídos por espaços. Qualquer conjunto de letras separadas por espaços (as palavras do texto) foram tratadas como os termos ou atributos do documento. As *stopwords* foram removidas das frases, pois são frequentes na maioria dos documentos do conjunto de dados e oferecem pouca informação para a distinção das classes. Foi realizado também o processo de estemização para obter apenas os radicais das palavras e unir os termos similares. Tanto para o processo de remoção de *stopwords* quanto para o de estemização, foi utilizada a biblioteca `NLTK`[4], de acordo com o idioma do texto. Todos os textos dessas bases de dados foram convertidos para a representação espaço-vetorial.

Os demais conjuntos de dados textuais (*Bibtex*, *Delicious*, *Enron*, *Medical* e *TMC2007*) foram adquiridos através do *framework* `Mulan`[5]. Eles foram utilizados nos experimentos de vários trabalhos de categorização multirrótulo e se encontram pré-processados em arquivos de treino e teste no formato ARFF. Cada termo é representado pelo seu peso binário.

5.2 Medidas e estratégias de avaliação

O método proposto foi avaliado em cenários de aprendizado *offline* e *online*, com o intuito de analisar seu desempenho nas mais diferentes situações. Os resultados obtidos na etapa de classificação foram comparados com os resultados de outros métodos aplicados para a categorização nessas mesmas condições.

Em experimentos com o aprendizado multirrótulo, é essencial utilizar diversas medidas para avaliar e fornecer uma visão de desempenho do método sob diferentes perspectivas (MADJAROV et al., 2012). As três medidas escolhidas para a comparação foram a macro F-medida, perda de Hamming e acurácia de subconjunto, que são largamente empregadas para a avaliação dos classificadores (TSOUMAKAS; KATAKIS; VLAHAVAS, 2010; GIBAJA; VENTURA, 2015).

Para verificar se existem diferenças estatisticamente significativas entre os resultados obtidos, foi aplicado também o teste estatístico não-paramétrico de Friedman. Esse teste verifica se a hipótese nula de que todos os métodos são equivalentes, pode ser rejeitada.

[3] `Beautiful Soup` é uma biblioteca para a linguagem Python que contém diversas funcionalidades, uma delas é a extração de trechos relevantes em documentos de texto. Biblioteca disponível em <https://www.crummy.com/software/BeautifulSoup/> (acessada em 13/02/2020)

[4] `NLTK` é uma biblioteca que oferece diversos recursos para auxiliar no processamento de linguagem natural na linguagem Python. Ela está disponível em <http://nltk.org/> (acessada em 04/03/2019)

[5] `Mulan` (TSOUMAKAS et al., 2011) é uma biblioteca de código aberto escrita em Java que inclui diversos algoritmos estado-da-arte, conjuntos de dados e métricas de avaliação para o aprendizado multirrótulo. As bases de dados estão disponíveis em: <http://mulan.sourceforge.net/Basededadoss-mlc.htlm> (acessado em 04/03/2019)

Para isso, ao invés de considerar o valor real da medida de avaliação, esse teste utiliza a média entre as posições de *ranking* ocupadas pelos métodos de acordo com o desempenho obtido em cada conjunto de dados (quanto melhor a medida, menor a posição de *ranking* ocupada), para verificar se existem diferenças estatísticas entre eles (DEMŠAR, 2006). A hipótese nula pode ser rejeitada se o valor crítico definido para um dado intervalo de confiança α, com um certo número de métodos avaliados k e um número de conjunto de dados observados N, for menor do que o valor obtido pela seguinte equação:

$$X_F^2 = \frac{12N}{k(k+1)} \left[\sum_{j=0}^{k} R_j^2 - \frac{k(k+1)^2}{4} \right], \tag{5.3}$$

onde R_j é a média de *ranking* obtida pelo *j-ésimo* método avaliado.

Se a hipótese nula for rejeitada pelo teste de Friedman, é executada uma comparação par-a-par usando o teste *post-hoc* de Bonferroni-Dunn para identificar se existem evidências estatísticas suficientes para apontar diferenças de desempenho entre o método proposto e os outros métodos em alguma das medidas avaliadas. Nesta análise, o desempenho de dois métodos se diferem significativamente se os seus *rankings* médios se diferem por, pelo menos, uma diferença crítica (DEMŠAR, 2006).

Foi considerado $\alpha = 0.05$ para a execução dos testes estatísticos. Mais detalhes sobre esses testes podem ser conferidos nos trabalhos de Demšar (2006).

5.3 Experimentos com cenário de aprendizado *offline*

Nos experimentos executados neste cenário, um único lote de amostras rotuladas é apresentado ao método na etapa de treinamento para a construção do modelo de predição. O modelo gerado é então utilizado para categorizar um conjunto de amostras não rotuladas e o desempenho do método é avaliado de acordo com essa rotulação.

Os resultados foram obtidos por meio de validação cruzada 5-*fold* usando a abordagem de estratificação proposta por Sechidis, Tsoumakas e Vlahavas (2011) para problemas de classificação multirrótulo. Nesse tipo de execução, o conjunto de dados é dividido em cinco partes de forma que a proporção de exemplos de cada classe seja mantida. Então são realizadas cinco iterações de treinamento e teste e, em cada uma dessas iterações, uma partição (*fold*) é utilizada para teste e as demais formam o conjunto de treinamento. A abordagem 5-*fold* foi escolhida ao invés da tradicional 10-*fold* (WEISS; INDURKHYA; ZHANG, 2015), pois essa última se torna muito custosa em experimentos com bases de dados que possuem um grande número de documentos ou classes (e.g, *Bibtex*, *TMC2007* e *RCV1*) e com métodos que possuem grande custo computacional e requerem busca em grade (e.g, SVM).

Tabela 3 – Métodos aplicados em cenário de aprendizado *offline*.

Transformação de Problema		
Método	Classificador base	Configuração de parâmetros
Relevância Binária (BR)[6] Boutell et al. (2004) Label Powerset (LP)[6]	Bayes ingênuo Multinomial (M.NB)[8] McCallum e Nigam (1998)	
	Árvores de decisão (DT)[8] Breiman et al. (1984)	
	Regressão logística (RL)[8] Cox (1958)	`penalty` : "l2" `solver` : "liblinear" `C` : $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}\}$
	Floresta aleatória (RF)[8] Breiman (2001)	`criterion` : "gini" `n_estimators` : $\{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$
	SVM Linear (L.SVM)[8] Fan et al. (2008)	`penalty` : "l2" `C` : $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$

Adaptação de Algoritmos	
Método	Configuração de parâmetros
ML-KNN[6] Zhang e Zhou (2007)	`k` : $\{1, 6, 11, 16, 21\}$ `s` : $\{5 \times 10^{-1}, 7 \times 10^{-1}, 10 \times 10^{-1}\}$
BRkNNa[6] Spyromitros, Tsoumakas e Vlahavas (2008)	`k` : $\{1, 6, 11, 16, 21\}$
BP-MLL[6,7,5] Zhang e Zhou (2006)	`hiddenLayers` : $(30,)$ `epochs` : 100 `learningRate` : 0,01
ML-MDLText	$\Omega_{c_j} : 2^{10}$ $\Omega_{Y_j} : 2^{10}$ $\gamma_1 : 0{,}95$ $\gamma_2 : 0{,}20$ H_{meta} : L.SVM

Ensemble		
Método	Classificador base	Configuração de parâmetros
RAkELd[6] Tsoumakas e Vlahavas (2007)	SVM Linear (L.SVM)[8] Fan et al. (2008)	`labelset_size` : $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ `criterion` : "*gini*" `C` : $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$

Para analisar o desempenho do método proposto, seus resultados foram comparados com os resultados de diversos métodos de transformação de problemas e de adaptação de algoritmos. A Tabela 3 descreve os métodos comparados nos experimentos. BR e LP foram as técnicas de transformação de problemas selecionadas para comparação, pois são duas abordagens muito utilizadas e tratam a dependência entre os rótulos de forma diferente: LP explora a dependência condicional enquanto que BR ignora a existência de dependência entre os rótulos. Todas as implementações foram retiradas da biblioteca `scikit-multilearn`[6], que disponibiliza diversos métodos de classificação multirrótulo e fornece uma interface para linguagem Python dos métodos implementados no `MEKA`[7] e no `Mulan`, dois grandes *frameworks* para o aprendizado multirrótulo.

Diversos métodos amplamente utilizados como *baselines* em trabalhos de classificação monorrótulo (JOACHIMS, 1998; ALMEIDA; YAMAKAMI; ALMEIDA, 2011; WU et al., 2014; ALMEIDA et al., 2016; SILVA, 2017) foram selecionados para gerar o

[6] `scikit-multilearn` (SZYMAŃSKI; KAJDANOWICZ, 2017) é uma biblioteca que contém diversos métodos de aprendizado multirrótulo implementados em Python nativo e segue os princípios de implementação da biblioteca `scikit-learn`. Biblioteca disponível em <http://scikit.ml/> (acessada em 13/02/2020)

[7] `MEKA` (READ et al., 2016) é um projeto que provê a implementação de diversos métodos de aprendizado multirrótulo da literatura cientifica, métricas para avaliação e uma interface para o uso dos recursos do `Mulan`, escrita na linguagem Java. Biblioteca Java para o aprendizado multirrótulo, disponível em <http://waikato.github.io/meka/> (acessada em 13/02/2020)

classificador base das técnicas de transformação. Todas as implementações de métodos de classificação monorrótulo foram obtidas através da biblioteca `scikit-learn`[8]. Já os métodos de adaptações de algoritmo selecionados se baseiam nos k-vizinhos mais próximos e em redes neurais artificiais, que também são amplamente encontrados nos trabalhos relacionados (GIBAJA; VENTURA, 2015). As implementações desses métodos foram obtidos através da biblioteca `scikit-multilearn`, `MEKA` e `Mulan`.

Alguns métodos, como o L.SVM, RF, ML-KNN, BRkNN-a e RAkELd, geram modelos diferentes de acordo com a variação de seus parâmetros e, por esta razão, foi realizada uma busca em grade usando a validação cruzada 5-*fold* estratificada sobre a partição de treinamento, para selecionar a configuração ideal. A melhor configuração é definida com base na média das macro F-medidas apresentadas na execução em cada *fold*. Os parâmetros empregados nesta busca estão descritos na coluna *Configuração de parâmetros* da Tabela 3. Embora o método BP-MLL seja sensível á variação da taxa de aprendizado, do peso de regularização, do número de iterações, do número de unidades e de camadas ocultas, a busca em grade não foi executada a fim de deixar a execução do método viável computacionalmente. Tanto o BP-MLL como os demais métodos foram executados com a configuração padrão da biblioteca.

Para o método ML-MDLText proposto, foi escolhido o método SVM Linear para gerar o metamodelo, já que foi o método que obteve as melhores avaliações dentre os resultados descritos no Apêndice A.

Nos experimentos com métodos baseados em *naïve* Bayes, foi utilizado o modelo de representação espaço-vetorial com pesos binários em todos os conjuntos de dados, já que ele foi projetado para trabalhar com atributos discretos (PEDREGOSA et al., 2011). Nos demais métodos, foi empregado o modelo de representação espaço-vetorial com pesos TF-IDF.

Nos conjuntos de dados listados na Tabela 2, as classes com menos de 25 exemplos foram removidas. Isso foi realizado com o objetivo de evitar problemas no cálculo das medidas de avaliação com classes raras, garantindo a presença de amostras de cada classe nas partições de teste e treino na execução da validação cruzada e na busca em grade. Os métodos que não foram capazes de finalizar a execução desse procedimento experimental em 48h foram reportados nos resultados com um traço (-), que indica a pior medida possível na análise.

[8] `scikit-learn` (PEDREGOSA et al., 2011) é uma biblioteca para a linguagem Python, projetada para interagir com as principais bibliotecas da linguagem e que contém diversos algoritmos de aprendizado supervisionado e não-supervisionado da literatura. Biblioteca disponível em <https://scikit-learn.org/> (acessada em 04/03/2019)

5.3.1 Resultados

A Tabela 4 apresenta as médias da macro F-medida, da perda de Hamming e da acurácia de subconjunto obtidas nas execuções de cada método e em cada base de dados. Para facilitar a comparação dos resultados, as medidas são apresentadas em um mapa de calor em tons de cinza, em que quanto melhor a medida, mais escuro é o tom utilizado. Além disso, a melhor medida obtida em cada base de dados está destacada em negrito.

Mesmo sem executar a busca em grade, o BP-MLL não foi capaz de terminar a execução nos conjuntos obtidos da coleção *RCV1*. Não obstante, a decisão de utilizar os parâmetros configurados na implementação do BP-MLL para permitir sua execução gerou resultados de classificação pobres em todos os conjuntos avaliados.

De forma geral e de acordo com o padrão da coloração observado nas colunas, os métodos de transformação que empregaram o método L.SVM e RL como algoritmo base conquistaram as melhores medidas em diversos conjunto de dados. Embora existam algumas diferenças entre o algoritmo L.SVM e da RL, como ambos empregaram *l2* como penalidade e *liblinear* como método de otimização, o desempenho dos dois foram muito similares na prática.

Todos os métodos, inclusive o ML-MDLText, obtiveram valores de macro F-medida abaixo de 0,4 nos conjuntos de dados *Bibtex*, *Delicious* e *Enron*. No Apêndice V, as curvas de aprendizado do ML-MDLText são apresentadas e é possível observar o comportamento na classificação. Nestas bases, o método proposto tem um ótimo desempenho na categorização da base de treinamento e um desempenho bem inferior na categorização dos documentos de teste. Essa característica pode estar associada ao grande número de *labelsets* distintos existentes nestas bases e ao alto desbalanceamento entre eles. Muitos desses *labelsets* não aparecem nas amostras de treinamento e, por isso, a classificação se torna complicada para métodos que os consideram para formular a predição. Apesar disso, o ML-MDLText ainda obteve o melhor desempenho em macro F-medida em duas dessas bases.

Ainda, de acordo com a F-medida, RAkELd e L.SVM (LP) apresentaram as melhores avaliações em mais conjuntos de dados: foram os melhores em cinco bases. ML-MDLText e RL (BR) obtiveram destaque nessa medida em dois conjuntos, enquanto que os métodos M.NB, DT (BR) e ML-KNN, foram os melhores em apenas um conjunto cada. Já observando a perda de Hamming, os resultados do RAkELd, RL (LP) e L.SVM (LP) dominaram as melhores avaliações e foram muito parecidos em diversos conjuntos de dados. A diferença entre a perda de Hamming obtida pelo método proposto e a melhor medida de cada base é muito pequena, chegando, no máximo, no valor de 0,023 de diferença. Por fim, as melhores acurácia de subconjunto foram obtidas pelos métodos com transformação LP, em sua maioria quando empregado o RL ou L.SVM como classificador base.

Tabela 4 – Resultados obtidos por cada método e em cada base de dados nas três medidas de avaliação.

(Continua)

	Transformação BR					Transformação LP					RAkELd				
Base de dados	M.NB	DT	RF	RL	L.SVM	M.NB	DT	RF	RL	L.SVM	ML-KNN	BRkNNa	BP-MLL	L.SVM	**ML-MDLText**
							Macro F-medida								
Reuters-ORGs	0,791	0,892	0,727	0,892	0,904	0,782	0,868	0,887	0,908	0,910	0,819	0,830	0,095	***0,914***	0,902
Reuters-places	0,133	0,812	0,305	0,793	***0,838***	0,092	0,493	0,380	0,779	0,793	0,692	0,684	0,016	0,834	0,769
RCV1-nivel1	0,844	0,823	0,863	0,909	0,912	0,840	0,806	0,844	0,912	***0,914***	0,882	0,874	-	***0,914***	0,878
RCV1-nivel2	0,342	0,551	0,386	0,662	0,672	0,263	0,445	0,534	0,673	***0,678***	0,600	0,600	-	0,668	0,655
RCV2-IT-nivel1	0,844	0,816	0,880	0,900	0,901	0,864	0,813	0,865	0,903	***0,904***	0,890	0,890	0,219	***0,904***	0,866
RCV2-IT-nivel2	0,393	0,544	0,385	0,649	0,666	0,357	0,462	0,560	0,690	***0,701***	0,617	0,617	0,013	0,654	0,667
RCV2-PT-nivel1	0,832	0,826	0,845	0,902	0,904	0,848	0,833	0,832	0,905	0,906	0,899	0,891	0,300	***0,907***	0,855
RCV2-PT-nivel2	0,457	0,583	0,460	0,643	0,661	0,427	0,540	0,618	***0,705***	0,700	0,630	0,624	0,032	0,674	0,668
RCV2-SP-nivel1	0,859	0,829	0,887	0,913	0,914	0,884	0,825	0,882	***0,919***	0,918	0,905	0,903	0,272	***0,919***	0,880
RCV2-SP-nivel2	0,421	0,550	0,374	0,657	0,689	0,393	0,490	0,599	0,727	***0,728***	0,658	0,662	0,026	0,694	0,692
Bibtex	0,211	0,280	0,080	0,303	0,324	0,145	0,197	0,243	0,293	0,301	0,323	0,323	0,017	0,321	***0,363***
Delicious	0,104	0,161	0,149	0,139	0,118	0,085	0,132	0,144	0,166	0,160	0,163	0,166	0,009	0,118	***0,167***
Enron	***0,328***	0,304	0,265	0,322	0,315	0,277	0,266	0,295	0,310	0,315	0,259	0,252	0,216	0,316	0,317
Medical	0,568	***0,869***	0,565	0,766	0,810	0,528	0,835	0,792	0,814	0,813	0,716	0,608	0,109	0,832	0,802
TMC2007	0,426	0,483	0,305	0,566	0,568	0,241	0,375	0,429	0,526	0,539	***0,583***	0,489	0,061	0,568	0,571
							Perda de Hamming								
Reuters-ORGs	0,037	0,026	0,037	0,022	***0,020***	0,042	0,026	0,021	***0,020***	***0,020***	0,041	0,040	0,159	***0,020***	0,026
Reuters-places	0,009	0,005	0,007	***0,003***	***0,003***	0,011	0,008	0,008	0,004	0,004	0,006	0,007	0,018	***0,003***	0,006
RCV1-nivel1	0,083	0,091	0,057	0,044	0,043	0,070	0,099	0,063	0,042	***0,041***	0,059	0,064	-	***0,041***	0,063
RCV1-nivel2	0,024	0,019	0,016	***0,012***	***0,012***	0,021	0,023	0,015	***0,012***	***0,012***	0,017	0,017	-	***0,012***	0,014
RCV2-IT-nivel1	0,099	0,109	0,067	0,060	0,060	0,081	0,111	0,074	***0,057***	***0,057***	0,065	0,066	0,388	0,058	0,080
RCV2-IT-nivel2	0,023	0,019	0,016	0,013	0,014	0,018	0,021	0,014	0,012	***0,011***	0,017	0,017	0,058	0,013	0,015
RCV2-PT-nivel1	0,071	0,073	0,044	0,041	0,039	0,059	0,072	0,053	***0,038***	***0,038***	0,041	0,041	0,308	0,039	0,060
RCV2-PT-nivel2	0,040	0,026	0,020	0,017	0,017	0,025	0,027	0,018	***0,015***	0,016	0,020	0,024	0,103	0,016	0,021
RCV2-SP-nivel1	0,071	0,080	0,050	0,044	0,043	0,058	0,080	0,053	***0,041***	***0,041***	0,048	0,049	0,236	***0,041***	0,059

Tabela 4 – Resultados obtidos por cada método e em cada base de dados nas três medidas de avaliação.

(Conclusão)

Base de dados	Transformação BR					Transformação LP									
	M.NB	DT	RF	RL	L.SVM	M.NB	DT	RF	RL	L.SVM	ML-KNN	BRkNNa	BP-MLL	RAkELd	**ML-MDLText**
Perda de Hamming															
RCV2-SP-nivel2	0,024	0,019	0,015	0,013	0,014	0,017	0,019	0,013	0,012	***0,011***	0,015	0,016	0,056	0,013	0,015
Bibtex	0,075	0,019	***0,013***	***0,013***	0,015	0,016	0,021	0,016	0,015	0,016	0,019	0,019	0,026	0,015	0,015
Delicious	0,054	0,024	***0,018***	0,020	0,020	0,027	0,029	0,029	0,027	0,028	0,027	0,028	0,029	0,020	0,028
Enron	0,188	0,109	***0,081***	0,095	0,103	0,100	0,118	0,097	0,100	0,099	0,114	0,125	0,098	0,100	0,104
Medical	0,075	***0,024***	0,043	0,035	0,032	0,058	0,028	0,028	0,034	0,035	0,049	0,074	0,126	0,031	0,038
TMC2007	0,068	0,085	0,073	0,064	0,066	0,071	0,098	0,071	0,065	0,064	***0,056***	0,059	0,117	0,066	0,066
Acurácia de subconjunto															
Reuters-ORGs	0,743	0,811	0,731	0,841	0,850	0,801	0,862	0,876	***0,879***	0,878	0,753	0,768	0,368	0,873	0,863
Reuters-places	0,667	0,815	0,742	0,872	0,871	0,691	0,788	0,765	***0,874***	***0,874***	0,805	0,802	0,438	0,869	0,813
RCV1-nivel1	0,734	0,716	0,829	0,859	0,862	0,820	0,759	0,832	0,874	***0,878***	0,828	0,824	-	0,873	0,834
RCV1-nivel2	0,393	0,465	0,473	0,634	0,633	0,526	0,522	0,624	0,676	***0,679***	0,586	0,586	-	0,630	0,619
RCV2-IT-nivel1	0,688	0,658	0,787	0,803	0,803	0,770	0,715	0,799	***0,830***	***0,830***	0,798	0,799	0,196	0,821	0,777
RCV2-IT-nivel2	0,486	0,532	0,534	0,657	0,643	0,633	0,595	0,706	0,726	***0,744***	0,647	0,647	0,108	0,663	0,671
RCV2-PT-nivel1	0,788	0,773	0,867	0,875	0,880	0,845	0,818	0,871	0,894	***0,895***	0,880	0,882	0,268	0,885	0,851
RCV2-PT-nivel2	0,567	0,622	0,662	0,739	0,743	0,710	0,697	0,783	***0,805***	0,798	0,733	0,721	0,074	0,767	0,737
RCV2-SP-nivel1	0,778	0,752	0,843	0,858	0,859	0,835	0,795	0,859	***0,876***	***0,876***	0,854	0,851	0,479	***0,876***	0,839
RCV2-SP-nivel2	0,593	0,649	0,694	0,748	0,737	0,751	0,724	0,802	0,816	***0,823***	0,759	0,752	0,217	0,771	0,755
Bibtex	0,063	0,086	0,097	0,173	0,161	0,197	0,147	0,228	***0,247***	0,240	0,196	0,196	0,009	0,161	0,241
Delicious	0,000	0,012	0,012	0,002	0,001	0,010	0,013	0,013	***0,014***	0,013	0,011	0,013	0,000	0,001	0,012
Enron	0,014	0,086	0,123	0,113	0,093	***0,170***	0,116	0,169	0,155	0,162	0,059	0,078	0,073	0,113	0,133
Medical	0,442	0,757	0,549	0,658	0,687	0,585	***0,798***	0,781	0,728	0,726	0,559	0,500	0,208	0,721	0,716
TMC2007	0,254	0,146	0,192	0,259	0,251	0,264	0,170	0,282	0,305	0,317	***0,327***	0,305	0,030	0,251	0,277

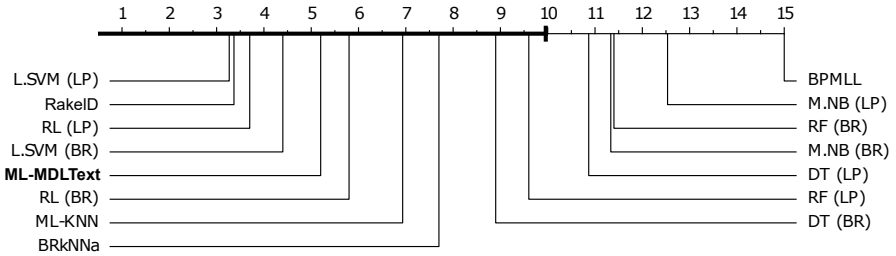
Métodos que empregaram o L.SVM como classificador base obtiveram melhores pontuações do que o ML-MDLText em todas as medidas avaliadas. Esse comportamento já era esperado, visto que o SVM apresentou desempenho semelhante em comparação com o MDLText nos experimentos em aprendizado monorrótulo apresentado por Silva, Almeida e Yamakami (2017) e métodos de aprendizado *offline* geralmente obtém melhores resultados que os métodos de aprendizado *online* quando eles podem ser aplicados (SILVA, 2017; CRAMMER; DREDZE; PEREIRA, 2012). Apesar do ML-MDLText não ter sido o melhor avaliado, seu desempenho sempre se manteve em posição intermediária e não foi o pior em nenhum conjunto de dados considerando essas medidas.

Para verificar se existem diferenças estatísticas de resultados entre os métodos, foi executada uma análise estatística usando o teste não-paramétrico de Friedman. Para um intervalo de confiança de $\alpha = 0.05$, o teste de Friedman descartou a hipótese nula de que todos os métodos são equivalentes e indicou que existem diferenças estatísticas significativas entre os resultados obtidos em todas as medidas avaliadas. Dessa forma, foi executada uma comparação par-a-par usando o teste *post-hoc* com o método de Bonferroni-Dunn. Nas Figuras 17a, 17b e 17c, os métodos são posicionados de acordo com seu *ranking* médio. A linha horizontal mais escura liga os métodos que não apresentaram diferenças estatísticas significativas em relação ao desempenho do ML-MDLText. Desconectados dessa linha, os métodos posicionados à direita ou à esquerda apresentaram resultados estatisticamente superiores e inferiores, respectivamente, com diferença estatística significativa.

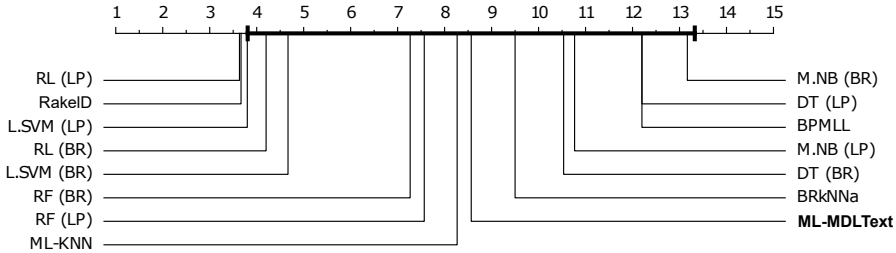
Na Figura 17a, o método ML-MDLText ocupa o quinto lugar no *ranking* e apresentou diferenças estatísticas significativas em relação aos métodos de transformação BR com M.NB e RF como algoritmo base, aos métodos de transformação LP com M.NB e DT como algoritmo base e ao BP-MLL. Além disso, segundo o teste, não há evidências estatísticas significativas que comprove diferença de desempenho entre o ML-MDLText e os métodos de transformação estado-da-arte como o RAkELd, LP e BR que utilizaram L.SVM ou RL como algoritmo base.

No caso da acurácia de subconjunto, apresentada na Figura 17c, as cinco melhores posições de *ranking* médio foram ocupadas por métodos que consideram a ocorrência simultânea de classes. Métodos de transformação LP, por exemplo, maximizam a acurácia de subconjunto e, por esta razão, costumam se sair melhor nesta medida (DEMBCZYŃSKI et al., 2012). O ML-MDLText obteve um melhor *ranking* médio do que o L.SVM (BR), o RL (BR) e o clássico ML-KNN, apesar de seus resultados não apresentarem diferenças estatisticamente significativas.

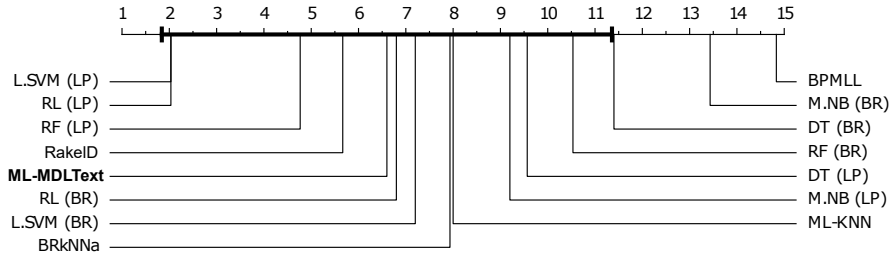
Por fim, em perda de Hamming, existem evidências estatísticas para afirmar que o ML-MDLText teve desempenho inferior aos métodos RL (LP) e RAkELd com L.SVM como algoritmo base. Apesar desse resultado, o método proposto tem como vantagem a capacidade de manipular diretamente os dados multirrotulados sem a necessidade de



(a) Macro F-medida.



(b) Perda de Hamming.



(c) Acurácia de subconjunto.

Figura 17 – *Ranking* médio e diferença crítica calculada usando o teste post-hoc Bonferroni-Dunn para as três medidas de avaliação. Os métodos estão posicionados de acordo com seu *ranking* médio e a linha horizontal mais escura liga os métodos que não apresentaram diferenças estatísticas significativas em relação ao desempenho do ML-MDLText.

aplicar abordagens de transformação.

5.4 Experimentos com cenário de aprendizado *online*

Para simular cenários de aprendizado *online*, um lote de amostras rotuladas (20% do conjunto de dados) é apresentado ao método para realizar um treinamento inicial. O modelo gerado é então utilizado para categorizar um conjunto de amostras não rotuladas, usando a abordagem *prequential* (também conhecida como teste-então-treina intercalados) (GAMA; SEBASTIÃO; RODRIGUES, 2013). Neste processo, um documento por vez

é apresentado ao classificador e o resultado da predição é armazenado para avaliar o desempenho do método. Caso a predição do classificador não esteja totalmente correta, ou seja, se pelo menos um rótulo predito não confere com o conjunto de classes verdadeiro, o método pode receber o documento com os rótulos corretos para atualizar o modelo de predição. Esse esquema é ilustrado pela Figura 18.

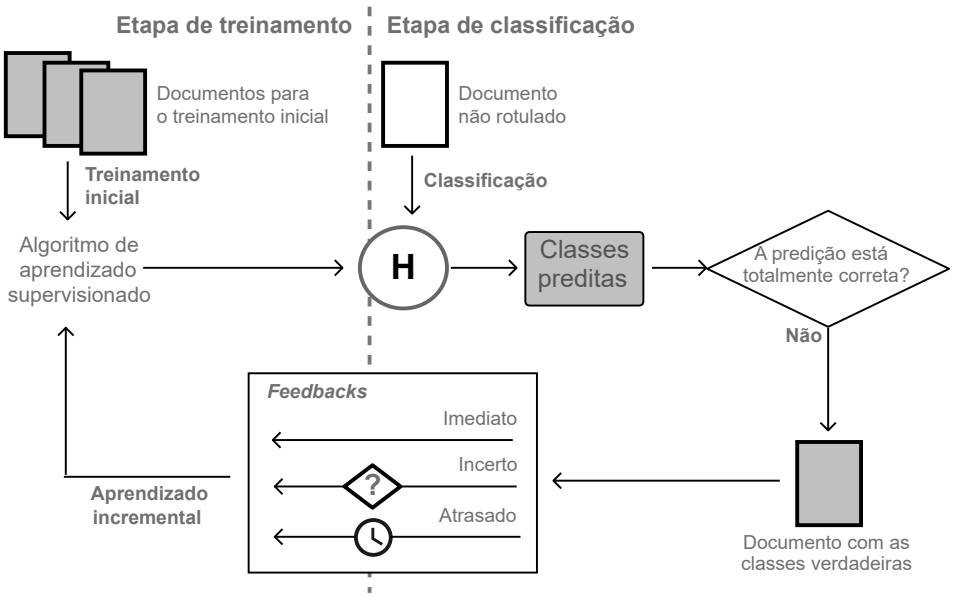


Figura 18 – Diagrama da simulação do cenário de aprendizado *online* com os diferentes *feedbacks*.

Para simular cenários reais, foram realizados experimentos variando a forma com que o documento é apresentado ao classificador para o treinamento incremental. Os três tipos de *feedbacks* empregados neste trabalho são descritos a seguir:

- ***Feedback* imediato:** após cometer um erro, o método de classificação recebe imediatamente o documento com as classes verdadeiras e atualiza seu modelo preditivo. Esse experimento simula um cenário ideal, onde os usuários avaliam a classificação instantaneamente.
- ***Feedback* incerto:** apesar de ter gerado uma predição incorreta, nem sempre o classificador recebe o *feedback*. Esse experimento simula usuários que não confirmam todas as predições efetuadas.
- ***Feedback* atrasado:** após obter uma predição incorreta, o *feedback* é apresentado ao método de classificação com atraso. Esse experimento simula usuários que levam um tempo para avaliar os resultados de predição.

Esquema semelhante também já foi empregado em estudos anteriores para avaliar as abordagens de aprendizado *online*, como nos trabalhos de Cormack e Lynam (2006) e Almeida e Yamakami (2012).

Nos experimentos com o *feedback* incerto, a decisão sobre enviar ou não enviar o documento é definida de forma aleatória. O tempo de espera no *feedback* atrasado também é definido de forma aleatória em um intervalo de 0 a 20, ou seja, caso ocorra um erro na predição, o documento com os rótulos corretos será apresentado ao método após, no máximo, 20 outras rodadas de teste. As partições de treino e teste são determinadas utilizando a abordagem de estratificação para dados multirrotulados proposta por Sechidis, Tsoumakas e Vlahavas (2011). A média de cinco execuções, variando-se a partição de treino e teste, foram utilizadas para a avaliação do desempenho do método.

Os resultados obtidos com a execução do método proposto foram comparados com diversos métodos que permitem o aprendizado incremental, empregados com a abordagem de transformação de problemas BR e CC. Como diversos métodos não permitem que a dimensão do espaço de rótulos seja estendida (*i.e.*, que sejam adicionadas novas classes ao classificador conforme novas amostras se tornam disponíveis), a transformação de problemas LP não pôde ser aplicada nos experimentos com o cenário de aprendizado *online*. O ML-MDLText foi comparado também com os resultados do método MLP, que suporta amostras multirrotuladas sem aplicar transformação de problema.

Todos os métodos utilizados na comparação são listados na Tabela 5. As implementações dos métodos empregados foram obtidas da biblioteca `scikit-learn` e foram aplicadas com a configuração de parâmetros padrão da biblioteca, já que o treinamento inicial com 20% de documentos não é suficiente para uma execução de busca em grade eficaz. A abordagem de transformação de problema BR também foi retirada da biblioteca `scikit-learn`, enquanto que a abordagem de transformação CC foi retirada da biblioteca `scikit-multiflow`[9].

Para a execução do método ML-MDLText, foi escolhido o método SGD para gerar o metamodelo, já que este foi o método que obteve os melhores resultados dentre os que permitem o aprendizado *online*, nos experimentos descritos no Apêndice A.

Nos experimentos com *naïve* Bayes, foi utilizado o modelo de representação com pesos binários em todos os conjuntos de dados, já que esse método foi projetado para trabalhar com atributos discretos (PEDREGOSA et al., 2011). Nos demais métodos, foi empregado o modelo de representação com pesos TF-IDF. Como foram avaliados cenários de aprendizado *online* e o peso TF-IDF depende de informações sobre os dados de treinamento, o processo de conversão na representação com peso TF-IDF também foi aplicado incrementalmente.

Nos conjuntos de dados listados na Tabela 2, as classes com menos de dez exemplos foram removidas, para garantir a presença de amostras representando cada classe nas

[9] `scikit-multiflow` é um *framework* de código aberto de aprendizado de máquina que apresenta opções para experimentos com dados em fluxo e o aprendizado multirrótulo, escrito na linguagem Python. Biblioteca disponível em <https://scikit-multiflow.github.io/> (acessada em 04/03/2019)

Tabela 5 – Métodos aplicados em cenário de aprendizado *online*.

Transformação de Problema		
Método	Classificador Base	Configuração de parâmetros
Relevância Binária (BR)[6] Boutell et al. (2004) Classifier Chain (CC)[9] (READ et al., 2011)	Bayes ingênuo Multinomial (M.NB)[8] McCallum e Nigam (1998)	
	Bayes ingênuo Bernoulli (B.NB)[8] McCallum e Nigam (1998)	
	Gradiente descente estocástico (SGD)[8] Zhang (2004)	`penalty` : "l2" `loss` : "hinge" `alpha` : 0,0001
	Passivo-agressivo (PA)[8] Crammer et al. (2006)	`C` : 1,0
	Perceptron[8] Rosenblatt (1958)	`penalty` : None `alpha` : 0,0001

Adaptação de Algoritmo	
Método	Configuração de parâmetros
MLP[8] Breiman (2001)	`hidden_layer_sizes` : (100,) `activation` : "relu" `learning_rate` : constant `alpha` : 0,0001
ML-MDLText	$\Omega_{c_j} : 2^{10}$ $\Omega_{Y_j} : 2^{10}$ $\gamma_1 : 0{,}95$ $\gamma_2 : 0{,}20$ H_{meta} : L.SVM

partições de treino e teste e evitar problemas no cálculo das medidas de avaliação. Os métodos que não foram capazes de finalizar a execução desse procedimento experimental em até 48h foram reportados nos resultados com um traço (-), que indica a pior medida possível na análise.

5.4.1 Resultados com *feedback* imediato

A Tabela 6 apresenta as médias da macro F-medida, da perda de Hamming e da acurácia de subconjunto obtidas nas execuções de cada método e em cada base de dados. Para facilitar a comparação dos resultados, as medidas são apresentadas em um mapa de calor em tons de cinza, em que quanto melhor a medida, mais escuro é o tom utilizado. Além disso, a melhor medida obtida em cada base de dados está destacada em negrito.

De maneira geral, o ML-MDLText não obteve a pior medida em nenhuma das medidas avaliadas, alcançando pontuações médias e altas em todos os conjunto de dados e se destacou em macro F-medida e acurácia de subconjunto.

A exploração da dependência entre os rótulos através da transformação CC não trouxe vantagens no aprendizado *online*, visto que os classificadores CC obtiveram resultados inferiores aos resultados obtidos pelos métodos de transformação BR, que ignoram qualquer tipo dependência, principalmente quando foi empregado o M.NB como classificador base.

Similar aos resultados apresentados na Seção 4, todos os métodos obtiveram baixa macro F-medida com as bases de dados *Bibtex*, *Delicious* e *Enron*, provavelmente devido à quantidade de rótulos e à cardinalidade dessas bases, que acentuam ainda mais o desafio

Tabela 6 – Resultados obtidos por cada método e em cada base de dados no cenário de aprendizado imediato.

	Transformação BR					Transformação CC						
Base de dados	M.NB	B.NB	SGD	PA	Perceptron	M.NB	B.NB	SGD	PA	Perceptron	MLP	ML-MDLText
Macro F-medida												
Reuters-ORGs	0.671	0.453	0.868	0.857	0.800	0.171	0.456	0.763	0.780	0.682	0.670	***0.873***
Reuters-places	0.123	0.059	0.634	***0.731***	0.683	0.018	0.059	0.545	0.657	0.506	0.537	0.674
RCV1-nivel1	0.851	0.815	***0.901***	0.895	0.870	0.761	0.816	0.879	0.860	0.844	0.888	0.875
RCV1-nivel2	0.337	0.258	0.488	***0.629***	0.587	0.087	0.260	0.507	0.567	0.501	0.562	0.624
RCV2-IT-nivel1	0.848	0.846	***0.892***	0.881	0.859	0.851	0.847	0.875	0.854	0.841	0.877	0.860
RCV2-IT-nivel2	0.392	0.282	0.444	0.610	0.582	0.153	0.284	0.483	0.565	0.495	0.516	***0.648***
RCV2-PT-nivel1	0.828	0.809	0.891	***0.893***	0.868	0.786	0.809	0.857	0.853	0.835	0.887	0.868
RCV2-PT-nivel2	0.405	0.316	0.534	0.585	0.581	0.231	0.317	0.524	0.551	0.491	0.541	***0.641***
RCV2-SP-nivel1	0.868	0.850	***0.901***	0.896	0.871	0.833	0.851	0.882	0.868	0.852	0.890	0.877
RCV2-SP-nivel2	0.376	0.250	0.473	0.597	0.575	0.155	0.251	0.484	0.546	0.471	0.499	***0.644***
Bibtex	0.207	0.173	0.187	0.259	0.275	0.066	0.142	0.197	0.238	0.220	0.185	***0.328***
Delicious	0.098	0.078	0.060	0.093	0.108	0.046	0.041	0.071	0.092	0.089	0.059	***0.150***
Enron	***0.235***	0.181	0.215	0.207	0.220	0.127	0.182	0.175	0.180	0.172	0.150	0.230
Medical	0.490	0.236	0.679	***0.688***	0.618	0.116	0.235	0.599	0.622	0.536	0.172	0.670
TMC2007	0.482	0.432	0.497	***0.583***	0.530	0.126	0.435	0.512	0.522	0.491	0.577	0.559
Perda de Hamming												
Reuters-ORGs	0.037	0.060	0.024	***0.022***	0.035	0.079	0.060	0.042	0.038	0.053	0.038	0.026
Reuters-places	0.007	0.009	***0.003***	***0.003***	0.004	0.008	0.009	0.004	0.004	0.006	***0.003***	0.005
RCV1-nivel1	0.078	0.098	***0.049***	0.052	0.066	0.084	0.097	0.060	0.071	0.079	0.057	0.064
RCV1-nivel2	0.026	0.027	0.012	***0.011***	0.016	0.023	0.027	0.013	0.015	0.018	0.012	0.014
RCV2-IT-nivel1	0.096	0.096	***0.065***	0.072	0.085	0.084	0.095	0.075	0.089	0.096	0.074	0.083
RCV2-IT-nivel2	0.024	0.023	0.013	***0.012***	0.016	0.022	0.023	0.014	0.016	0.019	0.013	0.014
RCV2-PT-nivel1	0.072	0.080	***0.046***	0.047	0.058	0.060	0.080	0.059	0.063	0.070	0.049	0.056
RCV2-PT-nivel2	0.035	0.037	***0.016***	***0.016***	0.022	0.030	0.037	0.020	0.021	0.026	0.017	0.020
RCV2-SP-nivel1	0.066	0.072	***0.050***	0.053	0.064	0.071	0.072	0.060	0.066	0.073	0.056	0.062
RCV2-SP-nivel2	0.021	0.022	***0.012***	***0.012***	0.015	0.018	0.022	0.013	0.015	0.017	***0.012***	0.014
Bibtex	0.060	0.065	***0.012***	***0.012***	0.021	0.014	0.102	0.014	0.015	0.021	0.013	0.017
Delicious	0.052	0.169	***0.018***	0.020	0.030	0.073	0.679	0.021	0.027	0.030	***0.018***	0.028
Enron	0.131	0.228	0.066	0.063	0.087	0.076	0.223	0.081	0.081	0.089	***0.060***	0.081
Medical	0.043	0.048	0.030	***0.026***	0.039	0.051	0.048	0.039	0.035	0.044	0.047	0.035
TMC2007	0.070	0.077	***0.058***	0.061	0.076	0.074	0.078	0.062	0.074	0.080	0.062	0.068
Acurácia de Subconjunto												
Reuters-ORGs	0.673	0.479	0.791	0.808	0.718	0.300	0.481	0.718	0.742	0.658	0.685	***0.814***
Reuters-places	0.627	0.570	0.840	***0.852***	0.779	0.570	0.573	0.803	0.805	0.743	0.809	0.805
RCV1-nivel1	0.742	0.692	***0.838***	0.828	0.790	0.743	0.696	0.814	0.784	0.764	0.816	0.821
RCV1-nivel2	0.323	0.305	0.573	***0.612***	0.525	0.198	0.307	0.579	0.552	0.493	0.585	0.604
RCV2-IT-nivel1	0.690	0.692	***0.784***	0.763	0.726	0.735	0.696	0.769	0.732	0.716	0.758	0.761
RCV2-IT-nivel2	0.416	0.413	0.599	0.640	0.571	0.314	0.420	0.630	0.611	0.563	0.621	***0.666***
RCV2-PT-nivel1	0.773	0.750	***0.852***	***0.852***	0.815	0.814	0.753	0.819	0.810	0.791	0.847	0.847
RCV2-PT-nivel2	0.516	0.472	0.698	0.713	0.643	0.421	0.477	0.673	0.664	0.618	0.699	***0.724***
RCV2-SP-nivel1	0.786	0.772	***0.834***	0.826	0.796	0.781	0.774	0.811	0.793	0.778	0.819	0.825
RCV2-SP-nivel2	0.551	0.523	0.715	0.728	0.667	0.542	0.523	0.708	0.693	0.651	0.709	***0.744***
Bibtex	0.056	0.060	0.143	0.161	0.087	0.083	0.061	0.135	0.137	0.101	0.128	***0.194***
Delicious	0.000	0.000	0.001	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.001	***0.010***
Enron	0.014	0.002	0.092	0.101	0.044	0.060	0.003	0.087	0.085	0.069	***0.106***	0.103
Medical	0.416	0.289	0.555	***0.593***	0.463	0.164	0.297	0.505	0.548	0.454	0.236	0.577
TMC2007	0.238	0.214	***0.298***	0.274	0.198	0.167	0.215	0.281	0.219	0.195	0.270	0.255

da classificação com esse tipo de aprendizado. O ML-MDLText obteve as melhores macro F-medida em seis bases de dados, inclusive na *Bibtex* e *Delicious*, que apresentam alto

grau de dificuldade na classificação. PA obteve destaque em cinco base de dados, SGD em três e M.NB em uma.

Considerando a perda de Hamming, as medidas obtidas pelo ML-MDLText foram superiores aos métodos NB em todos os experimentos executados, mas inferiores aos resultados de SGD e PA. Esses resultados eram esperados, já que a transformação BR minimiza a perda de Hamming (DEMBCZYŃSKI et al., 2012).

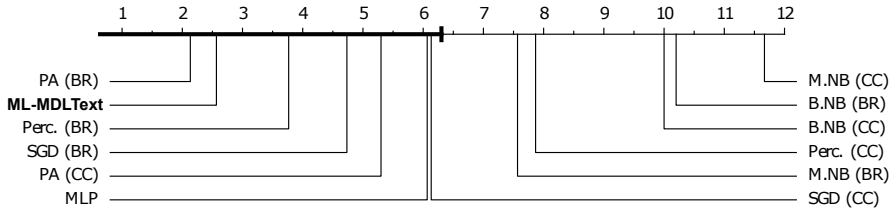
Por fim, de acordo com a acurácia de subconjunto, o método ML-MDLText obteve a melhor avaliação em seis conjuntos em um contexto geral. Embora o valor de acurácia de subconjunto de todos os métodos tenha sido relativamente baixo em *Bibtex* e *Delicious*, o ML-MDLText também obteve a melhor avaliação dentre os métodos nessas bases, que contém um número excessivamente grande de classes e de distintos *labelsets*. SGD, PA e MLP também obtiveram destaque em cinco, quatro e em uma base, respectivamente. Embora o SGD tenha obtido bons resultados em perda de Hamming, em geral, seus resultados de macro F-medida e acurácia de subconjunto foram inferiores aos resultados do ML-MDLText em muitos conjuntos de dados.

Para verificar se existem evidências estatísticas que comprovam a superioridade de algum método sobre os demais, foi executada uma análise estatística usando o teste não-paramétrico de Friedman. Com um intervalo de confiança $\alpha = 0.05$, o teste descartou a hipótese nula e indicou que existem diferenças estatísticas significativas entre os resultados obtidos em todas as medidas avaliadas. Dessa forma, foi executada uma comparação par-a-par usando o teste *post-hoc* de Bonferroni-Dunn para identificar se existem evidências estatísticas que indicam a superioridade do ML-MDLText sobre os métodos em alguma das medidas de desempenho. As Figuras 19a, 19b e 19c ilustram os resultados obtidos.

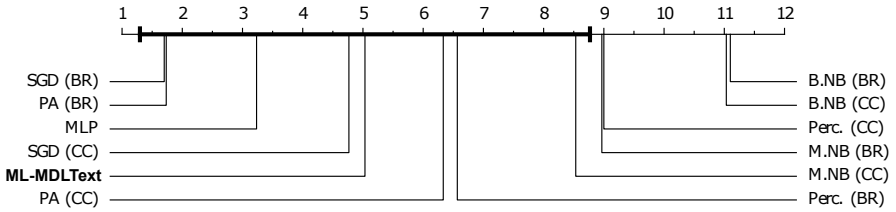
Nas Figuras 19a, 19b e 19c é possível observar que, sem utilizar técnicas de transformação, o ML-MDLText alcançou o segundo melhor *ranking* médio em macro F-medida e em acurácia de subconjunto, logo atrás do PA. Em todas as medidas de desempenho, o teste estatístico indica que existem evidências estatísticas para afirmar que o ML-MDLText obteve desempenho superior ao método B.NB, foi melhor que o M.NB em termos de macro F-medida e que o M.NB e Perceptron em termos de acurácia de subconjunto. Em relação à perda de Hamming, embora SGD, PA e MLP tenham obtido melhor *ranking* médio, não há diferenças estatísticas para afirmar que esses métodos foram superiores ao ML-MDLText.

5.4.2 Resultados com *feedback* incerto

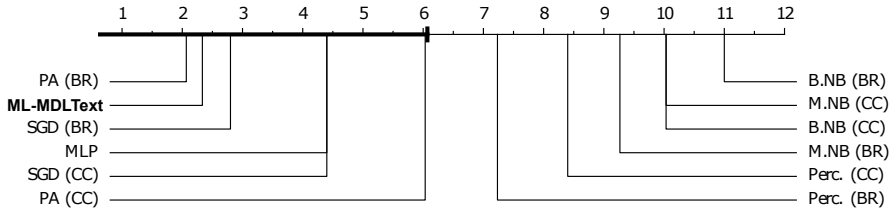
A Tabela 7 apresenta as médias da macro F-medida, da perda de Hamming e da acurácia de subconjunto obtidas nas execuções de cada método e em cada base de dados. Para facilitar a comparação dos resultados, as medidas são apresentadas em um mapa de



(a) Macro F-medida.



(b) Perda de Hamming.



(c) Acurácia de subconjunto.

Figura 19 – *Ranking* médio e diferença crítica calculada usando o teste post-hoc Bonferroni-Dunn para as três medidas de avaliação considerando o *feedback* imediato. Os métodos são posicionados de acordo com seu *ranking* médio e a linha horizontal mais escura liga os métodos que não apresentaram diferenças estatísticas significativas em relação ao desempenho do ML-MDLText.

calor em tons de cinza, em que quanto melhor a medida, mais escuro é o tom utilizado. Além disso, a melhor medida obtida em cada base de dados está destacada em negrito.

De maneira geral, os resultados obtidos foram ligeramente inferiores, mas foram condizentes em relação aos resultados obtidos com o *feedback* imediato: o método ML-MDLText dominou as melhores macro F-medidas enquanto que SGD e o PA obtiveram os melhores valores de perda de Hamming. Esses três métodos também obtiveram os melhores resultados em relação à acurácia de subconjunto.

O ML-MDLText, novamente, não foi o pior avaliado em nenhum conjunto e em nenhuma medida considerada nessa avaliação. De acordo com a macro F-medida e em comparação com o *feedback* imediato, o método PA foi o mais prejudicado nesse cenário onde nem sempre ocorre a retroalimentação para realizar o treinamento incremental. No

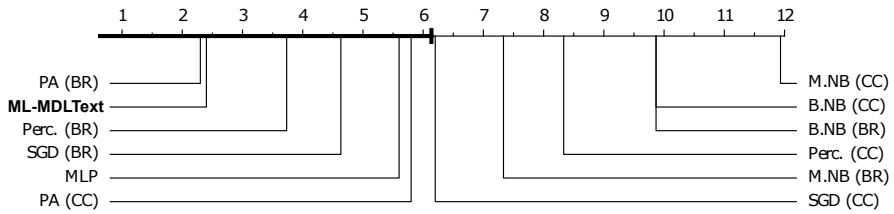
Tabela 7 – Resultados obtidos por cada método e em cada base de dados no cenário de aprendizado incerto.

	Transformação BR					Transformação CC						
Base de dados	M.NB	B.NB	SGD	PA	Perceptron	M.NB	B.NB	SGD	PA	Perceptron	MLP	ML-MDLText
Macro F-medida												
Reuters-ORGs	0.586	0.346	0.847	0.826	0.779	0.080	0.269	0.326	0.307	0.251	0.600	***0.863***
Reuters-places	0.084	0.043	0.595	***0.691***	0.654	0.013	0.043	0.501	0.604	0.457	0.502	0.665
RCV1-nivel1	0.848	0.796	***0.898***	0.892	0.866	0.708	0.797	0.876	0.854	0.837	0.882	0.872
RCV1-nivel2	0.291	0.203	0.483	0.605	0.571	0.058	0.204	0.492	0.539	0.475	0.545	***0.610***
RCV2-IT-nivel1	0.846	0.844	***0.890***	0.879	0.857	0.838	0.845	0.873	0.850	0.836	0.871	0.858
RCV2-IT-nivel2	0.350	0.242	0.438	0.572	0.561	0.113	0.243	0.468	0.529	0.462	0.484	***0.630***
RCV2-PT-nivel1	0.834	0.805	0.889	***0.892***	0.862	0.749	0.805	0.850	0.849	0.825	0.883	0.867
RCV2-PT-nivel2	0.381	0.280	0.520	0.556	0.564	0.212	0.281	0.506	0.521	0.456	0.526	***0.626***
RCV2-SP-nivel1	0.864	0.838	***0.898***	0.893	0.866	0.793	0.838	0.878	0.863	0.846	0.884	0.871
RCV2-SP-nivel2	0.337	0.210	0.449	0.555	0.554	0.118	0.212	0.457	0.512	0.430	0.460	***0.630***
Bibtex	0.198	0.156	0.177	0.228	0.259	0.053	0.129	0.184	0.211	0.201	0.174	***0.312***
Delicious	0.091	0.075	0.062	0.092	0.104	0.041	0.042	0.072	0.089	0.086	0.056	***0.139***
Enron	***0.225***	0.181	0.211	0.199	0.213	0.116	0.181	0.173	0.176	0.172	0.144	0.217
Medical	0.414	0.171	***0.657***	***0.657***	0.597	0.077	0.172	0.575	0.596	0.489	0.138	0.656
TMC2007	0.432	0.375	0.495	0.571	0.525	0.087	0.380	0.504	0.509	0.479	***0.573***	0.550
Perda de Hamming												
Reuters-ORGs	0.041	0.067	0.026	***0.025***	0.039	0.082	0.077	0.122	0.114	0.135	0.043	0.027
Reuters-places	0.007	0.009	***0.003***	***0.003***	0.004	0.008	0.009	0.004	0.004	0.006	0.004	0.005
RCV1-nivel1	0.079	0.106	***0.050***	0.054	0.068	0.094	0.105	0.062	0.074	0.083	0.060	0.066
RCV1-nivel2	0.024	0.025	***0.012***	***0.012***	0.016	0.024	0.025	0.013	0.015	0.019	0.013	0.014
RCV2-IT-nivel1	0.096	0.097	***0.066***	0.074	0.087	0.088	0.096	0.077	0.091	0.099	0.078	0.084
RCV2-IT-nivel2	0.021	0.022	***0.013***	***0.013***	0.017	0.024	0.021	0.014	0.016	0.019	0.014	0.015
RCV2-PT-nivel1	0.069	0.079	***0.047***	***0.047***	0.060	0.064	0.078	0.062	0.064	0.074	0.050	0.056
RCV2-PT-nivel2	0.032	0.036	***0.017***	***0.017***	0.023	0.032	0.036	0.021	0.022	0.027	0.018	0.020
RCV2-SP-nivel1	0.068	0.077	***0.051***	0.054	0.066	0.081	0.077	0.061	0.069	0.075	0.058	0.064
RCV2-SP-nivel2	0.019	0.021	***0.012***	***0.012***	0.016	0.020	0.021	0.014	0.015	0.018	0.013	0.014
Bibtex	0.049	0.052	***0.012***	0.013	0.021	0.014	0.082	0.014	0.015	0.021	0.013	0.017
Delicious	0.049	0.149	***0.018***	0.020	0.031	0.069	0.658	0.021	0.027	0.030	***0.018***	0.028
Enron	0.121	0.203	0.068	0.065	0.090	0.076	0.199	0.083	0.084	0.090	***0.061***	0.083
Medical	0.043	0.048	0.031	***0.027***	0.041	0.053	0.048	0.041	0.036	0.047	0.049	0.036
TMC2007	0.066	0.073	***0.059***	0.062	0.077	0.077	0.073	0.063	0.075	0.081	0.063	0.068
Acurácia de Subconjunto												
Reuters-ORGs	0.641	0.427	0.780	0.783	0.688	0.274	0.419	0.369	0.420	0.314	0.641	***0.804***
Reuters-places	0.631	0.587	0.832	***0.843***	0.765	0.571	0.588	0.791	0.793	0.721	0.796	0.796
RCV1-nivel1	0.744	0.678	***0.837***	0.824	0.783	0.717	0.682	0.812	0.776	0.754	0.806	0.819
RCV1-nivel2	0.361	0.311	0.571	***0.605***	0.512	0.152	0.311	0.572	0.541	0.473	0.570	0.599
RCV2-IT-nivel1	0.691	0.692	***0.783***	0.761	0.722	0.724	0.696	0.766	0.725	0.706	0.747	0.760
RCV2-IT-nivel2	0.463	0.432	0.597	0.630	0.562	0.247	0.439	0.624	0.602	0.547	0.607	***0.663***
RCV2-PT-nivel1	0.786	0.760	0.852	***0.853***	0.810	0.802	0.763	0.816	0.806	0.780	0.843	0.848
RCV2-PT-nivel2	0.545	0.485	0.692	0.706	0.633	0.388	0.492	0.663	0.655	0.601	0.691	***0.725***
RCV2-SP-nivel1	0.783	0.760	***0.832***	0.823	0.790	0.758	0.761	0.808	0.787	0.771	0.811	0.822
RCV2-SP-nivel2	0.582	0.536	0.711	0.722	0.657	0.504	0.536	0.701	0.685	0.634	0.700	***0.739***
Bibtex	0.066	0.067	0.141	0.153	0.083	0.083	0.069	0.126	0.129	0.095	0.124	***0.190***
Delicious	0.000	0.000	0.001	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.001	***0.008***
Enron	0.018	0.004	0.084	0.093	0.041	0.047	0.004	0.077	0.077	0.063	***0.101***	0.099
Medical	0.403	0.252	0.547	***0.585***	0.448	0.126	0.259	0.477	0.527	0.427	0.208	0.570
TMC2007	0.254	0.219	***0.294***	0.270	0.196	0.150	0.220	0.273	0.215	0.190	0.260	0.257

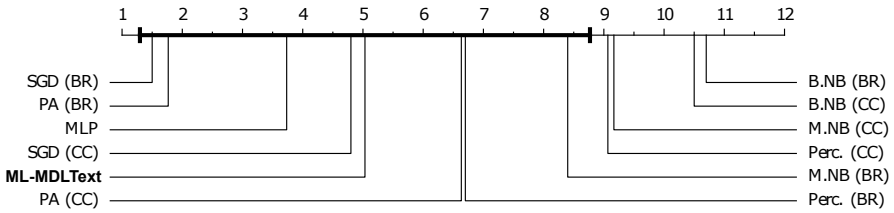
feedback imediato, PA obteve a melhor macro F-medida nos experimentos com *Reuters-places*, *RCV1-nivel2*, *RCV2-PT-nivel1*, *Medical* e *TMC2007*. Contudo, no cenário com

feedback incerto, PA obteve a melhor pontuação apenas em *Reuters-places*, *RCV2-PT-nivel1* e *Medical*. ML-MDLText se mostrou mais estável e manteve suas melhores avaliações nos dois cenários avaliados.

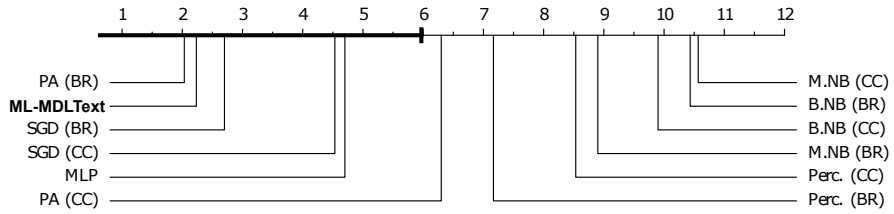
Para verificar se existem evidências estatísticas que comprovam a superioridade de algum método sobre os demais, foi executada uma análise estatística usando o teste não-paramétrico de Friedman. Com um intervalo de confiança $\alpha = 0.05$, o teste descartou a hipótese nula e indicou que existem diferenças estatísticas significativas entre os resultados obtidos em todas as medidas avaliadas. Dessa forma, foi executada uma comparação par-a-par usando o teste *post-hoc* de Bonferroni-Dunn para identificar se existem diferenças estatísticas significativas entre o ML-MDLText e os outros métodos em alguma das medidas de desempenho, como mostram as Figuras 20a, 20b e 20c.



(a) Macro F-medida.



(b) Perda de Hamming.



(c) Acurácia de subconjunto.

Figura 20 – *Ranking* médio e diferença crítica calculada usando o teste post-hoc Bonferroni-Dunn para as três medidas de avaliação considerando o *feedback* incerto. Os métodos são posicionados de acordo com seu *ranking* médio e a linha horizontal mais escura liga os métodos que não apresentaram diferenças estatísticas significativas em relação ao desempenho do ML-MDLText.

Novamente, mesmo sem exigir o emprego de técnicas de transformação de problemas, o ML-MDLText alcançou o segundo melhor *ranking* médio em macro F-medida e em acurácia de subconjunto. As figuras mostram ainda que existem evidências estatísticas para afirmar que o ML-MDLText teve desempenho superior aos métodos B.NB e Perceptron (CC) em todas as medidas avaliadas. Em acurácia de subconjunto, a análise estatística indicou que o desempenho obtido pelo ML-MDLText foi significativamente melhor do que M.NB, Perceptron e PA (CC). No caso da macro F-medida, a análise estatística indicou que o desempenho do ML-MDLText foi significativamente melhor do que os métodos NB e Perceptron (CC). Em termos de perda de Hamming, o teste mostra que não existem diferenças estatísticas para afirmar que o desempenho do SGD e PA, que empregam técnicas de transformação de problemas, foi superior aos resultados do ML-MDLText.

5.4.3 Resultados com *feedback* atrasado

A Tabela 8 apresenta as médias da macro F-medida, da perda de Hamming e da acurácia de subconjunto obtidas nas execuções de cada método e em cada base de dados. Para facilitar a comparação dos resultados, as medidas são apresentadas em um mapa de calor em tons de cinza, em que quanto melhor a medida, mais escuro é o tom utilizado. Além disso, a melhor medida obtida em cada base de dados está destacada em negrito.

De maneira geral, os resultados obtidos neste experimento também foram condizentes e muito próximos aos resultados obtidos com o *feedback* imediato: o ML-MDLText dominou as melhores macro F-medidas enquanto que SGD e o PA dominaram as melhores perda de Hamming. Os três métodos também obtiveram os melhores valores de acurácia de subconjunto. O ML-MDLText, novamente, não foi o pior avaliado em nenhum conjunto e em nenhuma medida considerada nessa avaliação, alcançando sempre pontuações médias e altas.

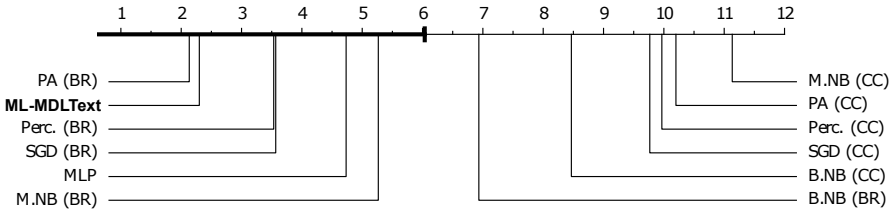
Nesse cenário, é possivel verificar uma piora no desempenho dos métodos Perceptron, SGD e PA com transformação CC. Provavelmente, o atraso na correção dos classificadores fez com que fosse necessário usar estimações incorretas da cadeia de classificadores para determinar a presença ou a ausência de um determinado rótulo, o que trouxe prejuízos para a execução dos métodos de transformação CC.

Para verificar se existem evidências estatísticas que comprovam a superioridade de algum método sobre os demais, foi executada uma análise estatística usando o teste não-paramétrico de Friedman. Usando um intervalo de confiança $\alpha = 0.05$, o teste descartou a hipótese nula e indicou que existem evidências estatísticas significativas entre os resultados obtidos em todas as medidas avaliadas. Dessa forma, foi executada uma comparação par-a-par usando o teste *post-hoc* de Bonferroni-Dunn para identificar se existem diferenças estatísticas que indicam a superioridade do ML-MDLText sobre os métodos em alguma das medidas de desempenho.

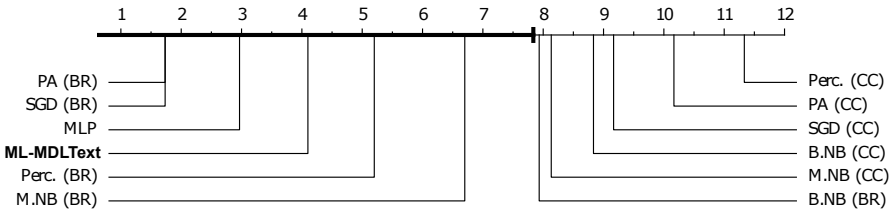
Tabela 8 – Resultados obtidos por cada método e em cada base de dados no cenário de aprendizado atrasado.

	Transformação BR					Transformação CC						
Base de dados	M.NB	B.NB	SGD	PA	Perceptron	M.NB	B.NB	SGD	PA	Perceptron	MLP	ML-MDLText
Macro F-medida												
Reuters-ORGs	0.670	0.448	0.860	0.852	0.787	0.080	0.269	0.326	0.307	0.251	0.665	***0.871***
Reuters-places	0.123	0.059	0.634	***0.732***	0.682	0.010	0.027	0.071	0.088	0.128	0.537	0.673
RCV1-nivel1	0.851	0.815	***0.901***	0.895	0.869	0.600	0.747	0.587	0.441	0.422	0.888	0.875
RCV1-nivel2	0.337	0.258	0.490	***0.629***	0.586	0.013	0.156	0.114	0.099	0.099	0.562	0.623
RCV2-IT-nivel1	0.848	0.846	***0.892***	0.881	0.859	0.691	0.836	0.523	0.438	0.418	0.877	0.860
RCV2-IT-nivel2	0.391	0.282	0.444	0.609	0.582	0.026	0.204	0.097	0.097	0.094	0.516	***0.647***
RCV2-PT-nivel1	0.829	0.808	0.891	***0.892***	0.867	0.656	0.760	0.472	0.466	0.445	0.887	0.870
RCV2-PT-nivel2	0.404	0.316	0.535	0.588	0.580	0.127	0.254	0.144	0.138	0.139	0.544	***0.641***
RCV2-SP-nivel1	0.868	0.850	***0.901***	0.896	0.871	0.480	0.788	0.350	0.422	0.436	0.890	0.877
RCV2-SP-nivel2	0.376	0.250	0.473	0.596	0.574	0.048	0.158	0.085	0.091	0.097	0.500	***0.646***
Bibtex	0.207	0.173	0.188	0.258	0.274	0.025	0.092	0.036	0.035	0.052	0.185	***0.328***
Delicious	0.098	0.078	0.060	0.093	0.108	0.023	0.039	0.019	0.027	0.032	0.059	***0.151***
Enron	***0.235***	0.181	0.218	0.204	0.219	0.099	0.160	0.106	0.103	0.109	0.153	0.227
Medical	0.485	0.235	0.667	***0.687***	0.619	0.014	0.131	0.205	0.200	0.172	0.181	0.667
TMC2007	0.482	0.432	0.497	***0.583***	0.531	0.061	0.329	0.152	0.139	0.155	0.578	0.560
Perda de Hamming												
Reuters-ORGs	0.037	0.061	0.025	***0.023***	0.037	0.082	0.077	0.122	0.114	0.135	0.039	0.025
Reuters-places	0.007	0.009	***0.003***	***0.003***	0.004	0.010	0.011	0.010	0.011	0.013	***0.003***	0.005
RCV1-nivel1	0.078	0.098	***0.049***	0.052	0.066	0.122	0.125	0.170	0.288	0.306	0.056	0.064
RCV1-nivel2	0.026	0.027	0.012	***0.011***	0.016	0.026	0.031	0.035	0.042	0.046	0.012	0.014
RCV2-IT-nivel1	0.096	0.096	***0.065***	0.072	0.085	0.128	0.100	0.244	0.327	0.346	0.075	0.083
RCV2-IT-nivel2	0.024	0.023	0.013	***0.012***	0.016	0.030	0.023	0.041	0.050	0.054	0.013	0.014
RCV2-PT-nivel1	0.071	0.080	***0.046***	0.047	0.059	0.095	0.088	0.255	0.277	0.308	0.048	0.056
RCV2-PT-nivel2	0.035	0.037	***0.016***	***0.016***	0.022	0.044	0.039	0.071	0.080	0.089	0.017	0.020
RCV2-SP-nivel1	0.066	0.072	***0.050***	0.053	0.064	0.153	0.094	0.192	0.200	0.226	0.056	0.062
RCV2-SP-nivel2	0.021	0.022	***0.012***	***0.012***	0.015	0.026	0.025	0.034	0.045	0.052	***0.012***	0.014
Bibtex	0.060	0.064	***0.012***	***0.012***	0.021	0.014	0.105	0.019	0.023	0.028	0.013	0.017
Delicious	0.052	0.169	***0.018***	0.020	0.030	0.070	0.496	0.023	0.031	0.034	***0.018***	0.028
Enron	0.131	0.227	0.066	0.063	0.087	0.080	0.201	0.103	0.106	0.108	***0.060***	0.081
Medical	0.043	0.048	0.032	***0.026***	0.040	0.059	0.056	0.084	0.081	0.090	0.046	0.036
TMC2007	0.070	0.077	***0.058***	0.061	0.076	0.083	0.076	0.101	0.133	0.137	0.062	0.068
Acurácia de Subconjunto												
Reuters-ORGs	0.672	0.478	0.787	0.800	0.697	0.274	0.419	0.369	0.420	0.314	0.679	***0.816***
Reuters-places	0.627	0.570	0.840	***0.852***	0.778	0.581	0.572	0.597	0.572	0.504	0.810	0.805
RCV1-nivel1	0.742	0.692	***0.838***	0.828	0.787	0.633	0.647	0.631	0.400	0.359	0.816	0.821
RCV1-nivel2	0.323	0.306	0.574	***0.611***	0.523	0.034	0.272	0.218	0.139	0.102	0.585	0.604
RCV2-IT-nivel1	0.690	0.692	***0.784***	0.763	0.725	0.613	0.689	0.482	0.337	0.302	0.758	0.761
RCV2-IT-nivel2	0.416	0.413	0.598	0.639	0.568	0.071	0.416	0.253	0.151	0.110	0.620	***0.666***
RCV2-PT-nivel1	0.775	0.749	***0.852***	0.850	0.813	0.721	0.752	0.427	0.386	0.321	0.847	0.847
RCV2-PT-nivel2	0.517	0.472	0.700	0.713	0.639	0.220	0.500	0.257	0.206	0.150	0.700	***0.724***
RCV2-SP-nivel1	0.786	0.772	***0.833***	0.826	0.794	0.625	0.726	0.570	0.560	0.502	0.819	0.825
RCV2-SP-nivel2	0.551	0.524	0.714	0.727	0.664	0.375	0.491	0.399	0.262	0.184	0.709	***0.743***
Bibtex	0.056	0.059	0.143	0.161	0.087	0.063	0.061	0.021	0.016	0.013	0.129	***0.196***
Delicious	0.000	0.000	0.001	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.001	***0.010***
Enron	0.014	0.002	0.094	0.103	0.046	0.017	0.002	0.033	0.036	0.029	***0.107***	0.103
Medical	0.415	0.289	0.538	***0.593***	0.451	0.023	0.188	0.164	0.172	0.145	0.247	0.565
TMC2007	0.238	0.214	***0.299***	0.273	0.198	0.124	0.206	0.104	0.047	0.042	0.271	0.254

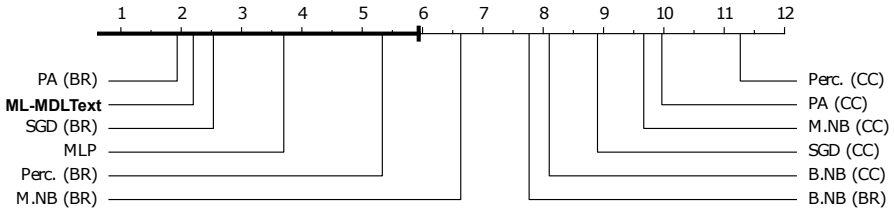
As Figuras 21a, 21b e 21c mostram que existem evidências estatísticas para afirmar que o ML-MDLText teve desempenho superior aos métodos de transformação CC e que o



(a) Macro F-medida.



(b) Perda de Hamming.



(c) Acurácia de subconjunto.

Figura 21 – *Ranking* médio e diferença crítica calculada usando o teste post-hoc Bonferroni-Dunn para as três medidas de avaliação considerando o *feedback* atrasado. Os métodos são posicionados de acordo com seu *ranking* médio e a linha horizontal mais escura liga os métodos que não apresentaram diferenças estatísticas significativas em relação ao desempenho do ML-MDLText.

B.NB em todas as medidas avaliadas e foi superior ao M.NB em termos de acurácia de subconjunto. Já em termos de perda de Hamming, o teste afirma que não existem evidências estatísticas para afirmar que o desempenho do ML-MDLText foi diferente do desempenho do MLP e de SGD e PA, que empregam técnicas de transformação de problemas. É possível também observar que, mesmo sem exigir técnicas de transformação de problemas, o ML-MDLText alcançou o segundo melhor *ranking* médio em macro F-medida e em acurácia de subconjunto, logo atrás do PA.

5.5 Considerações finais

Neste capítulo, foram apresentados todos os experimentos executados para a avaliação do método proposto. Os experimentos foram realizados utilizando 15 conjuntos de dados que são frequentemente utilizados no aprendizado multirrótulo e que apresentam diferentes desafios para a classificação. O desempenho do ML-MDLText obtidos na tarefa de classificação com essas bases foram comparados com os resultados de outros métodos conhecidos para a classificação multirrótulo, em diferentes cenários de aprendizado.

Inicialmente, experimentos com o cenário tradicional de aprendizado foram executados. Neste cenário, conhecido por aprendizado *offline* ou aprendizado por lote, um conjunto único amostras rotuladas são apresentadas ao método de classificação de uma só vez, em uma etapa de treinamento única. Para considerar situações reais encontradas nos problemas classificação de textos, também foram realizados experimentos simulando cenários em que seria necessário aplicar etapas de treinamento adicionais para corrigir previsões erradas do classificador, através de processos de aprendizado *online* ou incremental. Neste tipo de aprendizado, o método recebe um *feedback* contendo a classificação correta, para incrementar o treinamento já efetuado e atualizar o modelo de predição. Foram realizados testes com três tipos distintos de *feedbacks* para avaliar a robustez dos métodos. Por fim, a avaliação do método nestes diferentes cenários de aprendizado foi efetuada de acordo com seu desempenho ao classificar um conjunto de amostras selecionadas.

Em todos os cenários, o método proposto obteve uma boa macro F-Medida em conjuntos de dados com grande quantidade de rótulos e com alta cardinalidade, como em *Bibtex* e *Delicious*. Além disso, o ML-MDLText não obteve a pior avaliação em nenhum conjunto de dados ou medida considerada e se manteve dentre as melhores avaliações segundo a macro F-medida e acurácia de subconjunto e entre as intermediárias, segundo a perda de Hamming.

Os métodos que utilizaram o L.SVM e a RL como algoritmo base no cenário de aprendizado *offline*, obtiveram desempenho muito similares e conquistaram melhores posições de *ranking* médio do que o método proposto. No entanto, diferente desses métodos, é importante notar que o ML-MDLText é mais flexível do que L.SVM e RL, pois não depende de transformação na base de dados e é naturalmente incremental.

Já nos resultados em cenário de aprendizado *online*, o método proposto foi capaz de manter seus resultados em situações nos quais a correção era efetuada com atraso ou quando nem sempre era efetuada. Além disso, para todas as medidas avaliadas, o ML-MDLText obteve desempenho competitivo com o de métodos tradicionais de aprendizado incremental, como SGD, PA, Perceptron e MLP, que exigem transformação de problemas para serem aplicados na classificação multirrótulo.

Conclusões

Esta dissertação apresentou o ML-MDLText, um novo método de classificação de textos multirrótulo. Esse método foi projetado tendo como base as principais características de um método ideal para manipular os problemas de classificação de textos no contexto atual: deve ter a habilidade de manipular um grande volume de documentos de texto e se adequar as mudanças nos padrões encontrados nos textos, conforme novos documentos são criados. Ainda, o ML-MDLText é capaz de extrair mais conceitos do texto e oferecer uma predição com múltiplos rótulos, o que é útil em uma infinidade de aplicações.

Dado um documento não rotulado, o método proposto prediz o seu conjunto de rótulos considerando (*i*) o tamanho de descrição do documento de acordo com cada classe possível, e (*ii*) o tamanho de descrição do documento de acordo com cada possível *labelset* conhecido. Além disso, o número de rótulos do conjunto de rótulos a ser predito é estimado por um metamodelo, treinado com metadados extraídos dos dados de treinamento multirrótulo, e através de funções gaussianas, que definem o grau de confiança da saída do metamodelo. De acordo com a definição do método, a correlação entre os rótulos é explorada de forma condicional, pois a relevância do conjunto de rótulos mais apropriado é identificada levando em consideração os atributos do documento a ser rotulado.

Foi conduzida uma avaliação abrangente usando 15 bases de dados que são referências na literatura multirrótulo. O desempenho do ML-MDLText foi avaliado no tradicional cenário de aprendizado *offline* e também em simulações de cenários de aprendizado *online* reais, na qual o treinamento incremental deve ser executado para corrigir predições incorretas do classificador. Foram exploradas também diferentes possibilidades de *feedbacks* para avaliar a robustez do ML-MDLText nas mais diferentes situações. Os resultados obtidos pelo ML-MDLText foram comparados com o desempenho de outros métodos adaptados para a classificação multirrótulo, através de três medidas de avaliação específicas para o problema (macro F-medida, perda de Hamming e acurácia de subconjunto) e de testes estatísticos.

De acordo com os resultados experimentais, o ML-MDLText sempre esteve entre os dois melhores métodos em todos os cenários de aprendizado *online* e entre os cinco melhores em cenário de aprendizado *offline*, quando a macro F-medida e a acurácia de subconjunto foram usadas como medidas de desempenho. Todos os métodos de transformação de problemas que empregaram o SVM Linear como algoritmo base obtiveram melhores posições de *ranking* médio do que o método proposto no cenário de aprendizado *offline*, apesar do teste estatístico não ter apontado diferenças estatisticamente significativas entre eles. Considerando a perda de Hamming, o ML-MDLText esteve entre os cinco melhores

métodos nos cenários de aprendizado *online* e entre os nove melhores no aprendizado *offline*, isso porque, muitos dos métodos empregados na comparação minimizam essa medida. Alguns métodos, como o SGD e o MLP, obtiveram uma ótima posição de *ranking* médio em perda de Hamming, no entanto ocuparam posições inferiores ao ML-MDLText em macro F-medida e acurácia de subconjunto em cenários de aprendizado *online*. O ML-MDLText obteve ainda destaque na macro F-Medida em conjuntos de dados mais desafiadores, que contém grande quantidade de rótulos e alta cardinalidade.

Em resposta a principal pergunta de pesquisa deste trabalho, é notório que o ML-MDLText é um método de classificação multirrótulo robusto e eficiente. Além de ter obtido um desempenho notável nos experimentos, o ML-MDLText é capaz de manipular naturalmente documentos multirrotulados, sem exigir que o problema multirrótulo seja transformado em problemas de rótulo único. Portanto, ele possui um custo computacional muito menor que o SVM Linear, regressão logística, PA, SGD e Perceptron empregados com transformação de problema, principalmente em tarefas com grande número de classes ou quando o classificador base é custoso computacionalmente. Ainda, o ML-MDLText mostrou ser estável e apresentou resultados competitivos com o de outros métodos avaliados em diferentes cenários e domínios de problemas de classificação de textos. Isso faz com que ele seja uma opção mais vantajosa em situações com um número elevado de amostras ou com fluxo contínuo de dados.

Trabalhos futuros

À medida que as pesquisas foram avançando, foram observadas diversas possibilidades de melhorias e de complemento da proposta apresentada, mas que não puderam ser exploradas no escopo desse trabalho. A seguir, algumas sugestões de trabalhos futuros são listadas:

- **Desenvolver novas opções de definição do tamanho do *labelset* de predição.** O poder de predição do método é, de certa forma, influenciado pela capacidade preditiva do metamodelo. Se o metamodelo apontar um tamanho de *labelset* incorreto, o ML-MDLText pode ignorar *labelsets* apropriados que contém um tamanho diferente do que foi apontado. Para amenizar essa limitação, foi proposto o uso de curvas gaussianas para estipular a confiabilidade das predições do metamodelo. Apesar dos bons resultados obtidos por essa proposta, uma sugestão seria definir e avaliar outras técnicas para determinar o tamanho do *labelset* sem o uso de classificadores auxiliares, com o intuito de eliminar essa influência do metamodelo na qualidade preditiva do ML-MDLText.
- **Empregar novas funções para pontuar termos.** Na formulação do cálculo do tamanho de descrição, é empregada uma versão adaptada da função CF para pontuar

termos. Para trabalhos futuros, seria interessante avaliar outras técnicas de atribuição de pontuação para termos específicas para a seleção de atributos em problemas de classificação multirrótulo.

- **Avaliar o método proposto em problemas nos quais a dimensão do espaço de termos e de classes é variável.** Como explicado no Capítulo 4, o ML-MDLText é capaz de atualizar seu modelo preditivo conforme novas instâncias, com novos termos e novas classes, são apresentadas ao longo do tempo. Dessa forma, outro rumo de pesquisa seria explorar essa propriedade excepcional, que raramente é encontrada em métodos de classificação e é muito útil em problemas de classificação reais, dinâmicos e de larga escala.

- **Aplicar o método proposto na resolução de problemas de domínios específicos.** Outro complemento para o trabalho seria explorar o desempenho do ML-MDLText na classificação multirrótulo em algum problema específico de classificação de textos multirrótulo, como na categorização de e-mails, na classificação de documentos de acordo com categorias de emoções, na sugestão de *tag*, no diagnóstico médico ou na rotulação de documentos e páginas da *web*, por exemplo.

- **Adaptar o método proposto para tarefas de classificação hierárquicas.** Uma vez que a proposta desta dissertação foi direcionada a aplicação em problemas de classificação multirrótulo plana, uma outra alternativa para trabalhos futuros seria adaptá-la também para tarefas hierárquicas multirrótulo, nas quais um documento pode ter múltiplas classes organizadas hierarquicamente.

Publicações

Durante o período de elaboração desse estudo, os seguintes trabalhos foram submetidos e publicados com os resultados reportados nessa dissertação:

- BITTENCOURT, M. M.; SILVA, R. M.; ALMEIDA, T. A. ML-MDLText: A multilabel text categorization technique with incremental learning. In: *2019 8th Brazilian Conference on Intelligent Systems (BRACIS'19)*. Salvador, BA, Brasil: IEEE, 2019. p. 580–585. ISSN 2643-6256. Doi: 10.1109/BRACIS.2019.00107.

- BITTENCOURT, M. M.; SILVA, R. M.; ALMEIDA, T. A. ML-MDLText: an efficient and lightweight multilabel text classifier with incremental learning. *Applied Soft Computing*, Elsevier. *** Artigo convidado do BRACIS'19. Em revisão desde janeiro de 2020.**

Referências

ALMEIDA, T. A. et al. Text normalization and semantic indexing to enhance instant messaging and SMS spam filtering. *Knowledge-Based Systems*, Elsevier, v. 108, p. 25–32, maio 2016. Citado na página 91.

ALMEIDA, T. A.; YAMAKAMI, A. Facing the spammers: A very effective approach to avoid junk e-mails. *Expert Systems with Applications*, Elsevier, v. 39, n. 7, p. 6557–6561, jun. 2012. Citado 3 vezes nas páginas 28, 57 e 98.

ALMEIDA, T. A.; YAMAKAMI, A.; ALMEIDA, J. Filtering spams using the minimum description length principle. In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2010. (SAC '10), p. 1854–1858. Citado 2 vezes nas páginas 28 e 57.

ALMEIDA, T. A.; YAMAKAMI, A.; ALMEIDA, J. Spam filtering: how the dimensionality reduction affects the accuracy of naive Bayes classifiers. *Journal of Internet Services and Applications*, Springer-Verlag, v. 1, n. 3, p. 183–200, fev. 2011. Citado na página 91.

ALVARES-CHERMAN, E. *Aprendizado de máquina multirrótulo: explorando a dependência de rótulos e o aprendizado ativo*. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, BR, 2014. Citado 3 vezes nas páginas 42, 43 e 49.

ALVARES-CHERMAN, E.; METZ, J.; MONARD, M. C. Incorporating label dependency into the binary relevance framework for multi-label classification. *Expert Systems with Applications*, Elsevier, v. 39, n. 2, p. 1647–1655, fev. 2012. Citado 3 vezes nas páginas 27, 40 e 42.

ASSIS, F. et al. Exponential differential document count – a feature selection factor for improving Bayesian filters accuracy. In: *Proceedings of the 2006 MIT Spam Conference (SP'06)*. Cambridge, MA, USA: [s.n.], 2006. v. 535, p. 1–6. Citado 4 vezes nas páginas 58, 62, 72 e 73.

BARRON, A.; RISSANEN, J.; YU, B. The minimum description length principle in coding and modeling. *IEEE Transaction on Information Theory*, v. 44, n. 6, p. 2743–2760, out. 1998. Citado 2 vezes nas páginas 53 e 56.

BOUCHARD, G.; CELEUX, G. Selection of generative models in classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 28, n. 4, p. 544–554, abr. 2006. Citado na página 53.

BOUTELL, M. R. et al. Learning multi-label scene classification. *Pattern Recognition*, Elsevier, v. 37, n. 9, p. 1757–1771, set. 2004. Citado 4 vezes nas páginas 39, 41, 91 e 100.

BREIMAN, L. Random forests. *Machine Learning*, v. 45, n. 1, p. 5–32, out. 2001. Citado 3 vezes nas páginas 35, 91 e 100.

BREIMAN, L. et al. *Classification and regression trees*. [S.l.]: Chapman and Hall/CRC press, 1984. Citado 2 vezes nas páginas 35 e 91.

CARVALHO, A. C. P. L. F. de; FREITAS, A. A. A tutorial on multi-label classification techniques. In: ABRAHAM, A.; HASSANIEN, A.-E.; SNáŠEL, V. (Ed.). *Foundations of Computational Intelligence Volume 5: Function Approximation and Classification*. [S.l.]: Springer Berlin Heidelberg, 2009. p. 177–195. ISBN 978-3-642-01536-6. Citado 2 vezes nas páginas 36 e 41.

CLARE, A.; KING, R. D. Knowledge discovery in multi-label phenotype data. In: RAEDT, L. D.; SIEBES, A. (Ed.). *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*. [S.l.]: Springer Berlin Heidelberg, 2001. p. 42–53. ISBN 978-3-540-44794-8. Citado 3 vezes nas páginas 35, 39 e 46.

CORMACK, G. V.; LYNAM, T. R. TREC 2006 spam track overview. In: *TREC-2006: Fifteenth Text REtrieval Conference*. Gaithersburg, Maryland, USA: NIST Special Publication, 2006. Citado na página 98.

CORTES, C.; VAPNIK, V. N. Support-vector networks. *Machine Learning*, v. 20, n. 3, p. 273–297, set. 1995. Citado na página 35.

COVER, T. M.; HART, P. E. Nearest neighbor pattern classification. *IEEE Transaction on Information Theory*, IEEE, v. 13, n. 1, p. 21–27, jan. 1967. ISSN 0018-9448. Citado 3 vezes nas páginas 35, 54 e 126.

COVER, T. M.; THOMAS, J. A. *Elements of Information Theory*. 2nd. ed. New Jersey, EUA: John Wiley & Sons, 2006. Citado 2 vezes nas páginas 55 e 56.

COX, D. R. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, Wiley Online Library, v. 20, n. 2, p. 215–232, 1958. Citado na página 91.

CRAMMER, K. et al. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, JMLR.org, v. 7, p. 551–585, mar. 2006. Citado 3 vezes nas páginas 36, 100 e 126.

CRAMMER, K.; DREDZE, M.; PEREIRA, F. Confidence-weighted linear classification for text categorization. *Journal of Machine Learning Research*, JMLR.org, v. 13, n. 60, p. 1891–1926, jun. 2012. Citado 2 vezes nas páginas 36 e 96.

CRAMMER, K.; SINGER, Y. A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, JMLR.org, v. 3, p. 1025–1058, fev. 2003. Citado 3 vezes nas páginas 39, 47 e 87.

DEMBCZYŃSKI, K. et al. On label dependence and loss minimization in multi-label classification. *Machine Learning*, v. 88, n. 1, p. 5–45, jul 2012. ISSN 1573-0565. Citado 3 vezes nas páginas 50, 96 e 102.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, JMLR.org, v. 7, p. 1–30, jan. 2006. Citado na página 90.

DRUGAN, M. M.; WIERING, M. A. Feature selection for bayesian network classifiers using the MDL-FS score. *International journal of approximate reasoning*, Elsevier, v. 51, n. 6, p. 695–717, 2010. Citado na página 57.

ELISSEEFF, A.; WESTON, J. A kernel method for multi-labelled classification. In: DIETTERICH, T. G.; BECKER, S.; GHAHRAMANI, Z. (Ed.). *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. Cambridge, MA, USA: The MIT Press, 2002, (NIPS'01, v. 1). p. 681–687. Citado 2 vezes nas páginas 39 e 47.

FACELI, K. et al. *Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina*. [S.l.]: LTC, 2011. ISBN 9788521618805. Citado na página 35.

FAN, R.-E. et al. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, JMLR.org, v. 9, p. 1871–1874, ago. 2008. ISSN 1532-4435. Citado 2 vezes nas páginas 91 e 126.

FREITAS, B. L. de; SILVA, R. M.; ALMEIDA, T. A. Gaussian mixture descriptors learner. *Knowledge-Based Systems*, Elsevier, v. 188, p. 1–9, 2020. ISSN 0950-7051. Citado 2 vezes nas páginas 28 e 58.

FREUND, Y.; SCHAPIRE, R. E. Experiments with a new boosting algorithm. In: *Proceedings of the Thirteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996. (ICML'96), p. 148–156. ISBN 1558604197. Citado na página 35.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, v. 55, n. 1, p. 119–139, ago. 1997. ISSN 0022-0000. Citado na página 48.

GAMA, J. *Knowledge discovery from data streams*. [S.l.]: Chapman and Hall/CRC, 2010. ISBN 1439826110. Citado 3 vezes nas páginas 28, 35 e 36.

GAMA, J.; SEBASTIÃO, R.; RODRIGUES, P. P. On evaluating stream learning algorithms. *Machine Learning*, v. 90, n. 3, p. 317–346, mar. 2013. ISSN 0885-6125. Citado na página 97.

GEPPERTH, A.; HAMMER, B. Incremental learning algorithms and applications. In: *European Symposium on Artificial Neural Networks (ESANN)*. Bruges, Belgium: d-side publications, 2016. Citado na página 28.

GHAMRAWI, N.; MCCALLUM, A. Collective multi-label classification. In: *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*. New York, NY, USA: Association for Computing Machinery, 2005. (CIKM '05), p. 195–200. ISBN 1595931406. Citado na página 50.

GIBAJA, E.; VENTURA, S. A tutorial on multilabel learning. *ACM Computing Surveys (CSUR)*, ACM, New York, NY, USA, v. 47, n. 3, p. 1–38, abr. 2015. ISSN 0360-0300. Citado 15 vezes nas páginas 27, 32, 37, 39, 40, 41, 42, 43, 48, 49, 50, 51, 87, 89 e 92.

GODBOLE, S.; SARAWAGI, S. Discriminative methods for multi-labeled classification. In: DAI, H.; SRIKANT, R.; ZHANG, C. (Ed.). *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2004)*. Sydney, Australia: Springer Berlin Heidelberg, 2004. p. 22–30. ISBN 978-3-540-24775-3. Citado na página 51.

GONZALEZ-LOPEZ, J.; VENTURA, S.; CANO, A. Distributed nearest neighbor classification for large-scale multi-label data on Spark. *Future Generation Computer Systems*, Elsevier, v. 87, p. 66–82, out 2018. ISSN 0167-739X. Citado 2 vezes nas páginas 46 e 87.

GRÜNWALD, P. A minimum description length approach to grammar inference. In: WERMTER, S.; RILOFF, E.; SCHELER, G. (Ed.). *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing (IJCAI 1995)*. Montreal, Canada: Springer Berlin Heidelberg, 1996. p. 203–216. ISBN 978-3-540-49738-7. Citado na página 57.

GRÜNWALD, P. D. *The Minimum Description Length Principle*. [S.l.]: The MIT Press, 2007. Citado 5 vezes nas páginas 53, 54, 55, 56 e 57.

HANSEN, M. H.; YU, B. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, v. 96, n. 454, p. 746–774, jun. 2001. Citado 3 vezes nas páginas 54, 55 e 56.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 2nd. ed. New York, NY, USA: Prentice Hall, 1998. Citado na página 35.

JOACHIMS, T. Text categorization with support vector machines: Learning with many relevant features. In: *Proceedings of the 10th European Conference on Machine Learning (ECML'98)*. Chemnitz, Germany: Springer, Berlin, Heidelberg, 1998. p. 137–142. ISBN 978-3-540-69781-7. Citado na página 91.

KOLMOGOROV, A. N. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, v. 1, p. 1–7, 1965. Citado 2 vezes nas páginas 53 e 54.

KRISTAN, M.; LEONARDIS, A.; SKOČAJ, D. Multivariate online kernel density estimation with gaussian kernels. *Pattern Recognition*, Elsevier, v. 44, n. 10, p. 2630–2642, 2011. ISSN 0031-3203. Semi-Supervised Learning for Visual Content Analysis and Understanding. Citado na página 58.

LAGHMARI, K.; MARSALA, C.; RAMDANI, M. An adapted incremental graded multi-label classification model for recommendation systems. *Progress in Artificial Intelligence*, v. 7, n. 1, p. 15–29, mar. 2018. ISSN 2192-6360. Citado na página 58.

MADJAROV, G. et al. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, Elsevier, v. 45, n. 9, p. 3084–3104, set. 2012. ISSN 0031-3203. Citado 2 vezes nas páginas 87 e 89.

MCCALLUM, A.; NIGAM, K. A comparison of event models for naive Bayes text classification. In: *Proceedings of the 15th AAAI Workshop on Learning for Text Categorization (AAAI'98)*. Madison, Wisconsin: AAAI Press, 1998. p. 41–48. Citado 4 vezes nas páginas 35, 91, 100 e 126.

MCCALLUM, A. K. Multi-label text classification with a mixture model trained by EM. In: *AAAI 99 Workshop on Text Learning*. [S.l.: s.n.], 1999. Citado 3 vezes nas páginas 44, 45 e 87.

MEHTA, M.; RISSANEN, J.; AGRAWAL, R. MDL-based decision tree pruning. In: *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*. Montréal, Québec, Canada: AAAI Press, 1995. (KDD'95, 2), p. 216–221. Citado na página 57.

MENCÍA, E. L.; FÜRNKRANZ, J. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In: DAELEMANS, W.; GOETHALS, B.; MORIK, K. (Ed.). *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Antwerp, Belgium: Springer Berlin Heidelberg, 2008. (ECML PKDD '08), p. 50–65. ISBN 978-3-540-87481-2. Citado 2 vezes nas páginas 39 e 47.

MENCÍA, E. L.; FÜRNKRANZ, J. Pairwise learning of multilabel classifications with perceptrons. In: IEEE. *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. Hong Kong, China, 2008. p. 2899–2906. ISSN 2161-4393. Citado 4 vezes nas páginas 39, 44, 47 e 87.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, n. 85, p. 2825–2830, out. 2011. Citado 2 vezes nas páginas 92 e 99.

QUINLAN, J. R. *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN 1558602380. Citado 2 vezes nas páginas 35 e 46.

QUINLAN, J. R.; RIVEST, R. L. Inferring decision trees using the minimum description length principle. *Information and Computation*, v. 80, n. 3, p. 227–248, mar. 1989. ISSN 0890-5401. Citado na página 57.

READ, J. et al. Scalable and efficient multi-label classification for evolving data streams. *Machine Learning*, v. 88, n. 1, p. 243–272, jul. 2012. ISSN 1573-0565. Citado na página 28.

READ, J. et al. Batch-incremental versus instance-incremental learning in dynamic and evolving data. In: HOLLMÉN, J.; KLAWONN, F.; TUCKER, A. (Ed.). *Advances in Intelligent Data Analysis XI*. Helsinki, Finland: Springer Berlin Heidelberg, 2012. p. 313–323. ISBN 978-3-642-34156-4. Citado na página 28.

READ, J.; PFAHRINGER, B.; HOLMES, G. Multi-label classification using ensembles of pruned sets. In: *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2008. (ICDM '08), p. 995–1000. ISBN 978-0-7695-3502-9. ISSN 1550-4786. Citado 2 vezes nas páginas 43 e 44.

READ, J. et al. Classifier chains for multi-label classification. *Machine Learning*, v. 85, n. 3, p. 333, jun. 2011. ISSN 1573-0565. Citado 4 vezes nas páginas 42, 46, 47 e 100.

READ, J. et al. MEKA: A multi-label/multi-target extension to Weka. *Journal of Machine Learning Research*, v. 17, n. 21, p. 1–5, 2016. Disponível em: <http://jmlr.org/papers/v17/12-164.html>. Citado na página 91.

RENNIE, J. D. et al. Tackling the poor assumptions of naive Bayes text classifiers. In: *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*. Washington, DC, USA: AAAI Press, 2003. v. 3, p. 616–623. Citado na página 34.

RISSANEN, J. Modeling by shortest data description. *Automatica*, v. 14, n. 5, p. 465–471, set. 1978. Citado 3 vezes nas páginas 28, 53 e 56.

RISSANEN, J. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, v. 11, n. 2, p. 416–431, jun. 1983. Citado 2 vezes nas páginas 53 e 56.

RODRIGUES, E. S. da C. *Teoria da informação e adaptatividade na modelagem de distribuição das espécies*. Tese (Doutorado) — Escola Politécnica, Universidade de São Paulo, São Paulo, SP, BR, 2012. Citado 2 vezes nas páginas 55 e 57.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, n. 6, p. 386–408, 1958. Citado 2 vezes nas páginas 36 e 100.

SALTON, G.; BUCKLEY, C. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, v. 24, n. 5, p. 513–523, ago. 1988. ISSN 0306-4573. Citado na página 34.

SALTON, G.; WONG, A.; YANG, C. S. A vector space model for automatic indexing. *Magazine Communications of the ACM*, ACM, New York, NY, USA, v. 18, n. 11, p. 613–620, nov. 1975. ISSN 0001-0782. Citado na página 33.

SCHAPIRE, R. E.; SINGER, Y. BoosTexter: A Boosting-based system for text categorization. *Machine Learning*, v. 39, n. 2-3, p. 135–168, maio 2000. ISSN 1573-0565. Citado 5 vezes nas páginas 32, 39, 48, 50 e 87.

SEBASTIANI, F. Machine learning in automated text categorization. *ACM Computing Surveys*, ACM, New York, NY, USA, v. 34, n. 1, p. 1–47, mar. 2002. Citado 7 vezes nas páginas 27, 31, 32, 33, 34, 35 e 36.

SECHIDIS, K.; TSOUMAKAS, G.; VLAHAVAS, I. On the stratification of multi-label data. In: *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD'11)*. Athens, Greece: Springer Berlin, Heidelberg, 2011. p. 145–158. ISBN 978-3-642-23807-9. Citado 2 vezes nas páginas 90 e 99.

SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, v. 27, p. 379–423, 623–656, jul. 1948. Citado na página 61.

SHEINVALD, J.; DOM, B.; NIBLACK, W. A modeling approach to feature selection. In: *Proceedings 10th International Conference on Pattern Recognition*. Atlantic City, NJ, USA: IEEE, 1990. v. 1, p. 535–539. Citado na página 57.

SILVA, R. M. *Da Navalha de Occam a um método de categorização de textos simples, eficiente e robusto*. Tese (Doutorado) — Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, SP, BR, mar. 2017. Citado 12 vezes nas páginas 31, 33, 34, 35, 36, 53, 55, 59, 62, 63, 91 e 96.

SILVA, R. M.; ALMEIDA, T. A.; YAMAKAMI, A. Towards web spam filtering using a classifier based on the minimum description length principle. In: *Proceedings of the 15th International Conference on Machine Learning and Applications (ICMLA'16)*. Anaheim, California, USA: IEEE, 2016. p. 470–475. Citado na página 58.

SILVA, R. M.; ALMEIDA, T. A.; YAMAKAMI, A. MDLText: An efficient and lightweight text classifier. *Knowledge-Based Systems*, Elsevier, v. 118, p. 152–164, fev. 2017. ISSN 0950-7051. Citado 12 vezes nas páginas 28, 53, 57, 58, 59, 60, 61, 62, 69, 72, 96 e 126.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, v. 45, n. 4, p. 427–437, jul. 2009. Citado na página 49.

SPYROMITROS, E.; TSOUMAKAS, G.; VLAHAVAS, I. An empirical study of lazy multilabel classification algorithms. In: *Proceedings of the 5th Hellenic Conference on Artificial Intelligence: Theories, Models and Applications*. Syros, Greece: Springer Berlin, Heidelberg, 2008. (SETN '08), p. 401–406. ISBN 978-3-540-87880-3. Citado 2 vezes nas páginas 46 e 91.

SZYMAŃSKI, P.; KAJDANOWICZ, T. A scikit-based python environment for performing multi-label classification. *ArXiv e-prints*, fev. 2017. Citado na página 91.

TANG, L.; RAJAN, S.; NARAYANAN, V. K. Large scale multi-label classification via metalabeler. In: *Proceedings of the 18th International Conference on World Wide Web*. New York, NY, USA: ACM, 2009. (WWW '09), p. 211–220. ISBN 978-1-60558-487-4. Citado 3 vezes nas páginas 74, 75 e 87.

TSOUMAKAS, G.; KATAKIS, I. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, Idea Group Publishing, v. 3, n. 3, p. 1–13, set. 2007. Citado na página 37.

TSOUMAKAS, G.; KATAKIS, I.; VLAHAVAS, I. Effective and efficient multilabel classification in domains with large number of labels. In: *Proceedings ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*. [S.l.: s.n.], 2008. v. 21, p. 53–59. Citado na página 87.

TSOUMAKAS, G.; KATAKIS, I.; VLAHAVAS, I. Mining multi-label data. In: MAIMON, O.; ROKACH, L. (Ed.). *Data Mining and Knowledge Discovery Handbook*. Boston, MA: Springer US, 2010. p. 667–685. ISBN 978-0-387-09823-4. Citado 9 vezes nas páginas 27, 36, 40, 41, 42, 43, 44, 49 e 89.

TSOUMAKAS, G.; KATAKIS, I.; VLAHAVAS, I. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, v. 23, n. 7, p. 1079–1089, jul. 2011. Citado na página 44.

TSOUMAKAS, G. et al. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, v. 12, p. 2411–2414, 2011. Disponível em: <mulan.sourceforge.net/>. Citado na página 89.

TSOUMAKAS, G.; VLAHAVAS, I. Random k-labelsets: An ensemble method for multilabel classification. In: KOK, J. N. et al. (Ed.). *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. v. 4701, p. 406–417. ISBN 978-3-540-74958-5. Citado 3 vezes nas páginas 43, 44 e 91.

UYSAL, A. K.; GUNAL, S. The impact of preprocessing on text classification. *Information Processing & Management*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 50, n. 1, p. 104–112, jan. 2014. ISSN 0306-4573. Citado na página 34.

WEISS, S. M.; INDURKHYA, N.; ZHANG, T. *Fundamentals of predictive text mining*. 2. ed. [S.l.]: Springer Verlag London, 2015. ISBN 978-1-4471-6749-5. Citado 3 vezes nas páginas 33, 34 e 90.

WILBUR, W. J.; KIM, W. The ineffectiveness of within-document term frequency in text classification. *Information Retrieval*, Kluwer Academic Publishers, Hingham, MA, USA, v. 12, n. 5, p. 509–525, out. 2009. Citado na página 34.

WORRING, M. et al. The mediamill semantic video search engine. In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP'07*. Honolulu, HI, USA: IEEE, 2007. v. 4, p. IV–1213–IV–1216. ISSN 1520-6149. Citado na página 39.

WU, Q. et al. ML-FOREST: A multi-label tree ensemble method for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, v. 28, n. 10, p. 2665–2680, out. 2016. ISSN 1041-4347. Citado na página 87.

WU, Q. et al. ForesTexter: An efficient random forest algorithm for imbalanced text categorization. *Knowledge-Based Systems*, Elsevier, Amsterdam, The Netherlands, The Netherlands, v. 67, p. 105–116, 2014. ISSN 0950-7051. Citado na página 91.

YOUNES, Z. et al. A dependent multilabel classification method derived from the k-nearest neighbor rule. *EURASIP Journal on Applied Signal Processing*, Springer International Publishing, v. 2011, n. 1, p. 1–14, mar. 2011. ISSN 1687-6180. Citado na página 45.

ZHANG, M.-L. ML-RBF: RBF neural networks for multi-label learning. *Neural Processing Letters*, Springer US, v. 29, n. 2, p. 61–74, abr. 2009. ISSN 1573-773X. Citado 2 vezes nas páginas 47 e 48.

ZHANG, M.-L.; PEÑA, J. M.; ROBLES, V. Feature selection for multi-label naïve Bayes classification. *Information Sciences*, v. 179, n. 19, p. 3218–3229, 2009. ISSN 0020-0255. Citado na página 35.

ZHANG, M.-L.; ZHOU, Z.-H. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 18, n. 10, p. 1338–1351, out. 2006. ISSN 1041-4347. Citado 4 vezes nas páginas 47, 48, 87 e 91.

ZHANG, M.-L.; ZHOU, Z.-H. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, Elsevier Science Inc., New York, NY, USA, v. 40, n. 7, p. 2038–2048, jul. 2007. ISSN 0031-3203. Citado 5 vezes nas páginas 32, 35, 39, 45 e 91.

ZHANG, T. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: *Proceedings of the 21th International Conference on Machine Learning (ICML'04)*. Banff, Alberta, Canada: ACM, 2004. p. 116–123. Citado 3 vezes nas páginas 36, 100 e 126.

ZHANG, X.; GRAEPEL, T.; HERBRICH, R. Bayesian online learning for multi-label and multi-variate performance measures. In: TEH, Y. W.; TITTERINGTON, M. (Ed.). *Proceedings of the 30th International Conference on Artificial Intelligence and Statistics*. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010. (Proceedings of Machine Learning Research, v. 9), p. 956–963. Citado 2 vezes nas páginas 45 e 87.

ZHU, S. et al. Multi-labelled classification using maximum entropy method. In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: Association for Computing Machinery, 2005. (SIGIR '05), p. 274–281. ISBN 1595930345. Citado na página 50.

APÊNDICE A – Experimentos para definir o método a ser empregado como metamodelo

Para identificar os melhores métodos para definir o número de rótulos presentes no conjunto de predição, o desempenho de diferentes métodos de classificação e regressão foram comparados, que podem tanto ser aplicados em um cenário de aprendizado *offline* e *online*. Com exceção do MDLText, as implementações dos métodos empregados neste experimento foram obtidas através da biblioteca `scikit-learn`[1]. Os métodos empregados são listados na Tabela 9.

Os resultados foram obtidos por meio de validação cruzada 5-*fold* estratificada. Para métodos como o L.SVM, L.SVM-Regressão e KNN, foi realizada uma busca em grade usando validação cruzada 5-*fold* estratificada para selecionar a melhor combinação de parâmetros. Os parâmetros são descritos na coluna *Configuração de parâmetros* da Tabela 9. A melhor combinação de parâmetros foi selecionada com base na macro F-medida para os métodos de classificação e o erro quadrático médio para os métodos de regressão. Para os métodos MDLText, SGD e PA, foram utilizados os parâmetros padrões recomendados pelos autores com o intuito de empregá-los em cenários de aprendizado *online*. Os demais métodos foram executados com a configuração padrão da biblioteca.

Os métodos de regressão retornam valores reais como saída e, para ficar de acordo com os rótulos que sempre são números inteiros, seus valores de predição foram arredondados.

A Tabela 22 apresenta a macro F-medida média de cada método e em cada base de dados. Para facilitar a comparação dos resultados, as medidas são apresentadas em um mapa de calor em tons de cinza, na qual, quanto melhor a medida, mais escuro é o tom utilizado. Além disso, a melhor medida obtida em cada base de dados está destacada em negrito.

Boa parte das melhores avaliações foram dominadas pelo método L.SVM. A execução da busca em grade pela melhor configuração do parâmetro C não trouxe melhoria significativa de resultados. Em diversas bases de dados, como em *Delicious*, *RCV1-nivel1*, *RCV1-nivel2*, *RCV2-SP-nivel2* e *Reuters-ORGs*, a versão com o parâmetro C fixo obteve resultados melhores do que a versão com a implementação de busca em grade, devido ao alto desbalanceamento de amostras entre as classes e a dificuldade de executar a validação cruzada nestas condições.

[1] Biblioteca disponível em <https://scikit-learn.org/> (acessada em 13/02/2020).

Tabela 9 – Métodos aplicados para a definição do metamodelo.

Classificação	
Método	Configuração de parâmetros
Bayes ingênuo Multinomial (M.NB) McCallum e Nigam (1998)	
Bayes ingênuo Bernoulli (B.NB) McCallum e Nigam (1998)	
ML-MDLText Silva, Almeida e Yamakami (2017)	
K-vizinhos mais próximos (KNN) Cover e Hart (1967)	`k` : $\{1, 6, 11, 16, 21\}$
SVM Linear (L.SVM)[8] Fan et al. (2008)	`penalty` : "l2" `C` : $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$
SVM Linear* (L.SVM*)	`penalty` : "l2" `C` : 1,0
Gradiente descente estocástico (SGD) [8] (ZHANG, 2004)	`penalty` : "l2" `loss` : "hinge" `alpha` : 0,0001
Passivo-agressivo (PA) [8] (CRAMMER et al., 2006)	`C` : 1,0

Regressão	
Método	Configuração de parâmetros
K-vizinhos mais próximos (KNN-Regressão) Cover e Hart (1967)	`k` : $\{1, 6, 11, 16, 21\}$
SVM Linear (L.SVM-Regressão)[8] Fan et al. (2008)	`penalty` : "l2" `C` : $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$
Gradiente descente estocástico (SGD-Regressão) [8] (ZHANG, 2004)	`penalty` : "l2" `loss` : "hinge" `alpha` : 0,0001
Passivo-agressivo (PA-Regressão) [8] (CRAMMER et al., 2006)	`C` : 1,0

Dentre as versões que podem ser aplicadas em cenários de aprendizado *online*, o PA, SGD e MDLText tiveram resultados semelhantes, sendo destaque em três e duas bases distintas, respectivamente.

Para verificar se existem evidências estatísticas que comprovam a superioridade de algum método sobre os demais, foi executada uma análise estatística usando o teste não-paramétrico de Friedman. Com 95% de confiança, o teste de Friedman descartou a hipótese nula de que todos os métodos são equivalentes e indicou que existem diferenças estatísticas significativas entre os resultados obtidos. Dessa forma, foi executada uma comparação par-a-par usando o teste *post-hoc* com o método de Nemenyi para identificar se existem evidências estatísticas suficientes para apontar diferenças de desempenho entre os métodos avaliados. Na Figura 23, os métodos estão posicionados de acordo com seu *ranking* médio e as linhas horizontais mais escuras ligam métodos que não apresentaram diferenças estatísticas significativas.

As versões do L.SVM foram estatisticamente superiores aos métodos de regressão SGD e PA e ao método de classificação B.NB. Além disso, apesar de ocuparem as melhores posições de *ranking* médio, não foram encontradas diferenças estatísticas significativas para afirmar superioridade das versões do L.SVM sobre os outros métodos. O SGD na classificação foi considerado superior a sua versão de regressão e ao B.NB. Devido às

Figura 22 – Macro F-medida obtida por cada método avaliado por meio de validação cruzada 5-*fold* estratificada.

Classificação								
Base de Dados	M.NB	B.NB	MDLText	SGD	PA	KNN	L.SVM	L.SVM*
Bibtex	0.104	0.100	***0.113***	0.104	0.096	0.109	0.102	0.102
Delicious	0.078	0.065	0.082	0.077	0.068	***0.096***	0.084	0.085
Enron	0.158	0.164	0.200	0.205	0.194	0.166	0.209	0.204
Medical	0.377	0.366	0.443	0.453	***0.500***	0.373	0.448	0.428
RCV1-nivel1	0.507	0.351	0.499	0.495	0.511	0.501	0.511	***0.519***
RCV1-nivel2	0.263	0.207	0.274	0.270	0.270	***0.290***	0.268	0.281
RCV2-IT-nivel1	0.449	0.394	0.518	0.488	0.575	0.468	***0.593***	***0.593***
RCV2-IT-nivel2	0.333	0.301	0.367	0.335	0.422	0.356	***0.441***	***0.441***
RCV2-PT-nivel1	0.622	0.620	0.628	***0.672***	0.663	0.668	0.671	0.671
RCV2-PT-nivel2	0.476	0.462	0.478	***0.577***	0.564	***0.577***	0.574	0.574
RCV2-SP-nivel1	0.494	0.461	0.474	0.528	0.529	0.531	***0.540***	***0.540***
RCV2-SP-nivel2	0.354	0.291	0.446	0.469	***0.505***	0.480	0.477	0.486
Reuters-ORGs	0.417	0.370	0.412	0.463	0.476	0.442	0.411	0.456
Reuters-places	0.120	0.096	0.161	0.138	***0.170***	0.153	0.169	0.154
TMC2007	0.145	0.149	***0.172***	0.157	0.138	0.138	0.145	0.139

Regressão				
Base de Dados	SGD	PA	KNN	L.SVM-Regressão
Bibtex	0.091	0.092	0.108	0.096
Delicious	0.066	0.062	0.064	0.063
Enron	0.208	0.197	0.176	***0.214***
Medical	0.380	0.494	0.377	0.425
RCV1-nivel1	0.275	0.475	0.462	0.470
RCV1-nivel2	0.193	0.263	0.274	0.275
RCV2-IT-nivel1	0.343	0.430	0.437	0.437
RCV2-IT-nivel2	0.262	0.312	0.366	0.302
RCV2-PT-nivel1	0.548	0.639	0.661	0.662
RCV2-PT-nivel2	0.354	0.446	0.521	0.498
RCV2-SP-nivel1	0.331	0.456	0.505	0.419
RCV2-SP-nivel2	0.218	0.342	0.340	0.335
Reuters-ORGs	0.345	***0.479***	0.393	0.471
Reuters-places	0.089	0.128	0.116	0.131
TMC2007	0.117	0.128	0.144	0.138

posições de *ranking* médio obtido pelos métodos de classificação L.SVM* e SGD, eles foram selecionados para a aplicação em cenário de aprendizado *offline* e *online*, respectivamente.

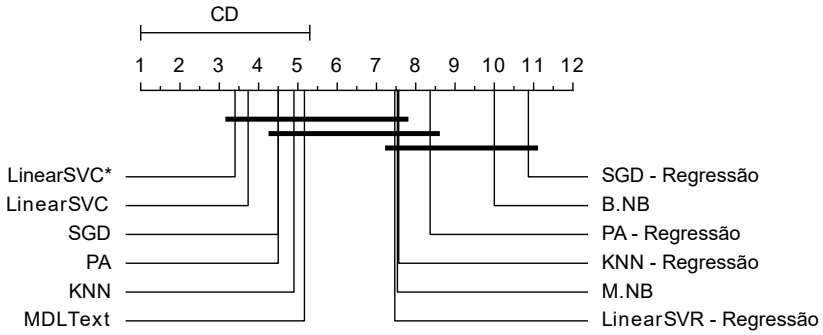
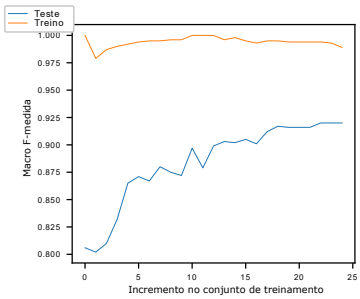


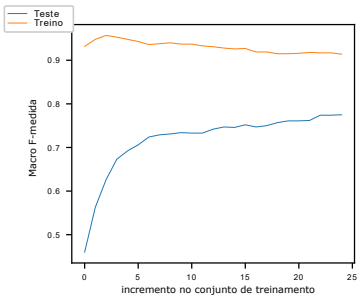
Figura 23 – *Ranking* médio e diferença crítica (CD) calculada usando o teste post-hoc Nemenyi para a macro F-medida na definição do metamodelo. Os métodos são posicionados de acordo com o seu *ranking* médio e as linhas horizontais mais escuras ligam métodos que não apresentaram diferenças estatísticas significativas.

APÊNDICE B – Curvas de Aprendizado do ML-MDLText

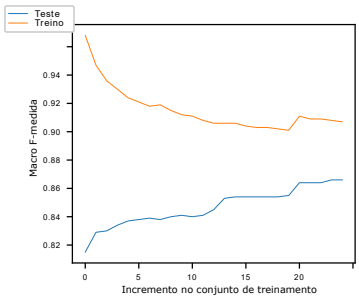
A Figura 24 ilustra as curvas de aprendizado obtidas pelo ML-MDLText na classificação. Nas bases de dados *Delicious*, *Bibtex* e *Enron*, o método proposto tem um ótimo desempenho na categorização das amostras da base de treinamento e um desempenho bem inferior na categorização dos documentos separados para teste, desconhecidos por ele. Essa característica pode estar associada ao grande número de *labelsets* distintos existentes nestas bases e ao baixo número de amostras pertencentes a eles. Muitos dos *labelsets* das amostras de teste não aparecem nas amostras de treinamento e, por isso, a classificação dessas amostras se torna bem mais complicada do que a classificação das amostras de treinamento.



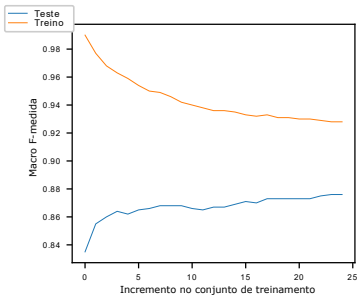
(a) Reuters-ORGs



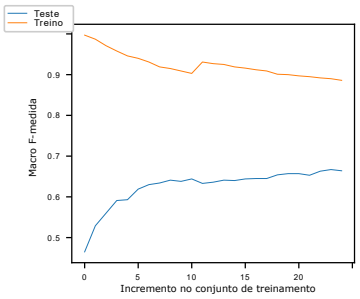
(b) Enron



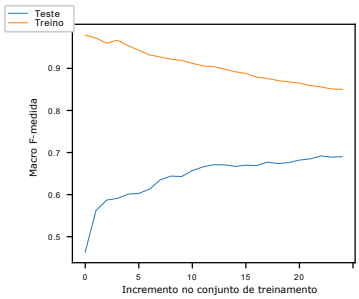
(c) RCV2-IT-nivel1



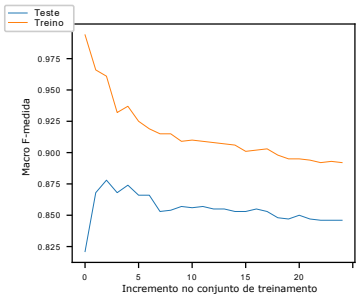
(d) RCV1-nivel1



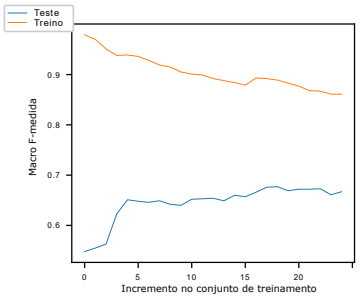
(e) RCV1-nivel2



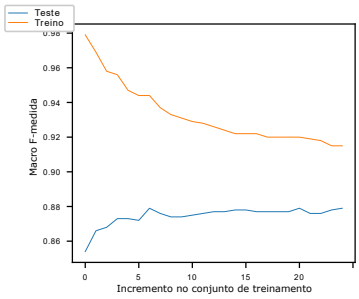
(f) RCV2-IT-nivel2



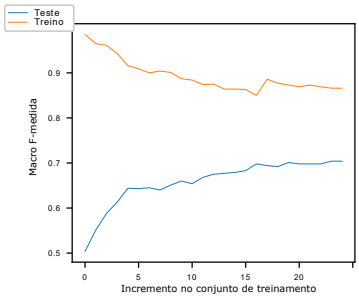
(g) RCV2-PT-nivel1



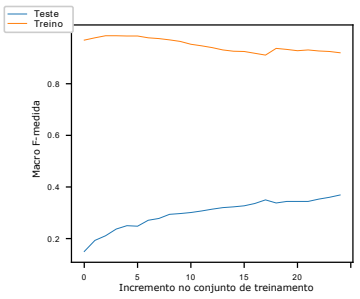
(h) RCV2-PT-nivel2



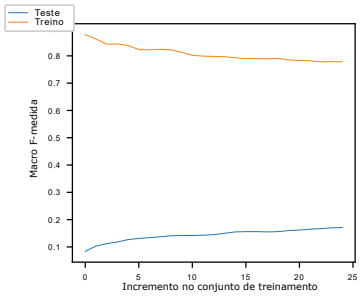
(i) RCV2-SP-nivel1



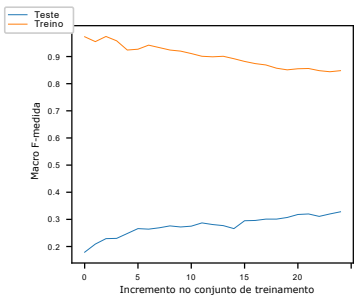
(j) RCV2-SP-nivel2



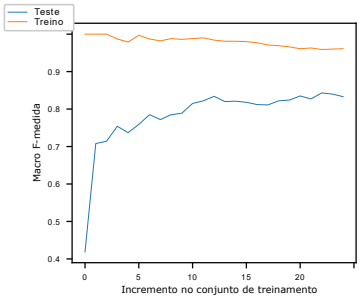
(k) Bibtex



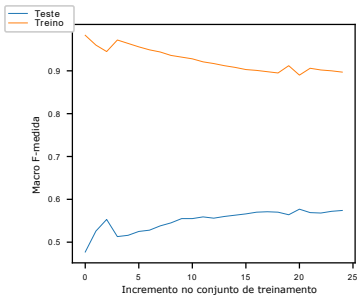
(l) Delicious



(m) Enron



(n) Medical



(o) TMC2007

Figura 24 – Curvas de aprendizado do método ML-MDLText.