Universidade Federal de Pernambuco

Centro de Informática

Curso de Engenharia da Computação

Avaliação de Parâmetros do Método *Combined Dissimilarity Spaces*

Letícia Virgínia Netto Lapenda

lvnl@cin.ufpe.br

Trabalho de Graduação

Recife, Dezembro de 2017

Universidade Federal de Pernambuco

Centro de Informática

Curso de Engenharia da Computação

Letícia Virgínia Netto Lapenda

lvnl@cin.ufpe.br

Avaliação de Parâmetros do Método *Combined Dissimilarity Spaces*

Trabalho de Graduação

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Professor Orientador:
George Darmiton da Cunha Cavalcanti

Recife, Dezembro de 2017

É impossível progredir sem mudança, e aqueles que não mudam suas mentes não podem mudar nada.

George Bernard Shaw

Agradecimentos

Agradeço àqueles que são a expressão de Deus em minha vida, meus pais. Agradeço por todo o apoio e suporte que recebi desde os meus primeiros passos de formação educacional até a conclusão da minha graduação. A vitória é nossa.

Agradeço ao prof. George Cavalcanti pela orientação e pelo acompanhamento. Seu suporte foi fundamental para a conclusão deste trabalho.

Agradeço aos meus colegas de turma, que compartilharam as lutas e vitórias durante todos estes anos. Em especial a Anderson Urbano, Gustavo Pimentel, Gabriel Albuquerque, Larissa Lages, Eduardo Cintra, Luiz Antônio, Luiz de Lima, Gabriel Medeiros e Felipe Andrade.

Por fim, agradeço à UFPE, ao CIn, e a todos aqueles que contribuíram para a minha formação.

RESUMO

A categorização de textos pode ser aplicada em diversas áreas, tais como: *Business Intelligence*, análise de sentimento e monitoramento de redes sociais, segurança. Haja vista a extensa aplicabilidade das técnicas de análise textual, torna-se fundamental o aprimoramento destas técnicas a fim de obter resultados mais precisos e confiáveis. O modelo CoDiS (*Combined Dissimilarity Spaces*) implementa um sistema para classificação de textos no qual múltiplos classificadores são treinados com base em diferentes representações de dissimilaridade, de tal forma a contornar limitações dos modelos mais tradicionais. Este trabalho visa avaliar o desempenho do CoDiS quando o classificador usado varia. Além disso, no artigo original foi utilizado TF (*term frequency*) como formato de entrada dos dados. Outro formato muito utilizado é o TFIDF (*term frequency-inverse document frequency*), logo deseja-se comparar o uso de TF com TFIDF.

Para implementação do modelo CoDiS, foi utilizada a linguagem de programação Python, e para os testes estatísticos aplicados aos resultados, o *software* Matlab. Os experimentos foram realizados com 25 bases de dados, utilizando *10-fold stratified cross-validation*, e foram avaliadas 8 configurações paramétricas para o CoDiS, variando entre o classificador (Árvore de Decisão e SVM), o formato de entrada dos dados (TF e TFIDF) e a métrica de dissimilaridade (Distância Euclidiana e Cosseno). Ao final, foram aplicadas as métricas macro F1 e micro F1 para avaliar o desempenho das diferentes configurações testadas.

Os resultados mostram que existe uma diferença significativa entre (i) os classificadores AD e SVM, (ii) os formatos de entrada TF e TFIDF, (iii) as métricas de dissimilaridade DE e DC. Além disso, foi possível avaliar as 8 configurações ordenadas em relação ao desempenho, considerando a diferença crítica, tanto para macro F1, como para micro F1. Com base nos resultados dos testes estatísticos é possível afirmar que, no geral, (i) o classificador SVM superou a AD, (ii) o formato TFIDF é melhor do que o TF, (iii) a distância cosseno supera a distância euclidiana. Além disso, a configuração **SVM-TFIDF-DC** mostrou-se significativamente melhor do que pelo menos 6 outras: AD-TF-DE, AD-TF-DC, AD-TFIDF-DE, AD-TFIDF-DC, SVM-TF-DE e SVM-TFIDF-DE, comparando-se apenas à configuração SVM-TF-DC.

ABSTRACT

Text categorization can be applied in many areas, such as: Business Intelligence, sentiment analysis and social network monitoring, security. Considering textual analysis techniques' wide applications, it is important to improve these techniques in order to obtain more precise and reliable results. CoDiS (Combined Dissimilarity Spaces) model implements a system for text categorization in which multiple classifiers are trained based on different dissimilarity spaces, aiming to overcome more traditional models' limitations. This study evaluates CoDiS performance when the classifier model varies. Besides, in the original study TF (term frequency) was used as data input format. Another very common format is TFIDF (term frequency-inverse document frequency), therefore TF is compared to TFIDF.

In order to implement CoDiS, Python programming language was used, and for the statistical tests applied to results, the software Matlab. Experiments were run in 25 databases, using 10-fold stratified cross-validation, and 8 different parametric configurations for CoDiS, varying between the classifier (Decision Tree and SVM), the data input format (TF and TFIDF) and the dissimilarity measure (Distance Euclidean and Cosine). At the end, metrics macro F1 and micro F1 were applied to evaluate performance of all different configurations tested on CoDiS.

Results show there is a significant difference among (i) the classifiers Decision Tree (DT) and SVM, (ii) the data input formats TF and TFIDF, (iii) the dissimilarity measures Distance Euclidean (DE) and Distance Cosine (DC). Additionally, the 8 parametric configurations tested were evaluated regarding their performance, considering the critical difference, for macro F1 and micro F1. Based on the statistical tests' results, it is possible to conclude that in general, (i) SVM classifier overcomes the Decision Tree, (ii) TFIDF format is better than TF, (iii) Distance Cosine outperforms Distance Euclidean. Furthermore, **SVM-TFIDF-DC** configuration seemed to be significantly better than at least 6 others: DT-TF-DE, DT-TF-DC, DT-TFIDF-DE, DT-TFIDF-DC, SVM-TF-DE e SVM-TFIDF-DE, only to be compared to SVM-TF-DC.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

BoW - *Bag of Words*

CoDiS - *Combined Dissimilarity Spaces*

TF - *Term Frequency*

TFIDF - *Term Frequency - Inverse Document Frequency*

SVM - *Support Vector Machine*

AD - Árvore de Decisão

DE - Distância Euclidiana

DC - Distância Cosseno

1 Introdução

No problema de categorização de texto é muito comum a representação de documentos como um vetor de atributos utilizando uma técnica chamada *Bag of Words* (BoW). Isto é, um ou vários documentos são representados por um vocabulário, conjunto composto pelas palavras que os formam. Este conjunto é então utilizado para gerar um vetor de atributos por documento de tal forma que cada posição do vetor corresponda a um dos termos do conjunto e possua como valor correspondente um peso, que pode ser, por exemplo, a frequência em que o termo aparece no documento em questão (Sparck Jones, 1972). O funcionamento da técnica BoW está ilustrado na Figura 1.

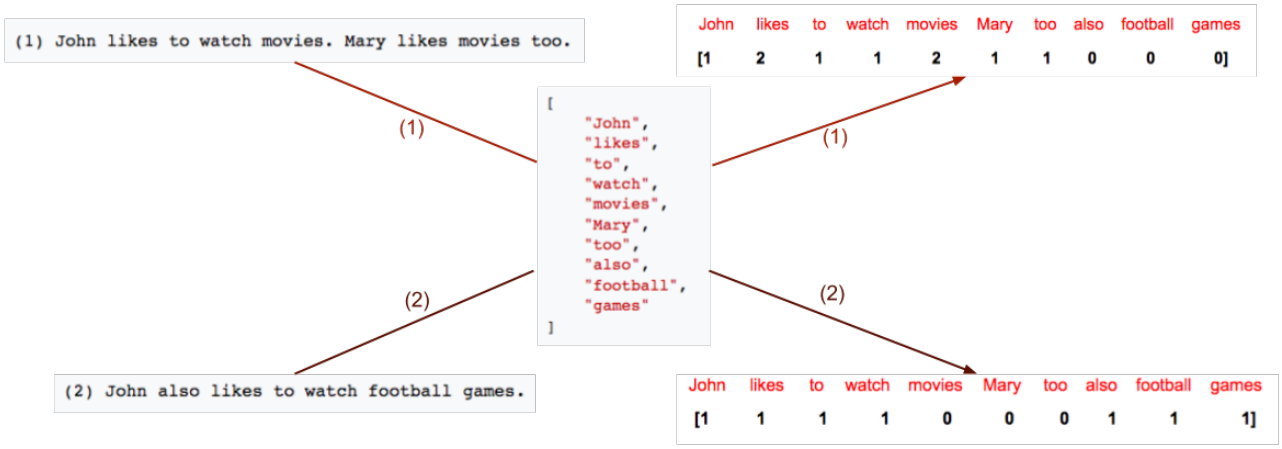

Figura 1: Exemplo de representação utilizando BoW

Esta técnica, no entanto, apresenta alguns problemas. Primeiramente, a alta dimensionalidade, uma vez que são considerados como atributos todos os termos presentes em qualquer dos documentos do *Corpus* (Lewis, 1992). Além disso, há uma desproporcionalidade em relação à quantidade de atributos, que é muito elevada, quando comparada à quantidade de instâncias (documentos) (Gangeh, Kamel and Duin, 2010). Por fim, a matriz de dados é esparsa, pois, para cada documento, haverá uma grande quantidade de termos que não estão presentes, sendo o problema especialmente evidente para uma grande base de dados (Su, Sayyad-Shirabad and Matwin, 2011).

Para lidar com as limitações apresentadas, foi proposto um sistema para classificação de texto no qual múltiplos classificadores são treinados com base em diferentes representações de dissimilaridade (Pinheiro, Cavalcanti and Tsang, 2017). O objetivo da transformação é gerar, a partir de vetores com alta dimensionalidade, novos vetores com baixa dimensionalidade. Cada documento passa, então, a ser representado por um vetor de dissimilaridade no qual os elementos correspondem à distância para todos os documentos presentes em um conjunto de Representação. O conjunto de Representação, por sua vez, é um subconjunto da base de documentos para treinamento, cujos elementos são selecionados de forma aleatória. Além disso, o sistema proposto, denominado de CoDiS (*Combined Dissimilarity Spaces*), não utiliza métodos de seleção de atributos, utilizando todos os atributos disponíveis no cálculo da distância entre documentos. A seleção de atributos pode ser nociva,

uma vez que descarta atributos, podendo levar à perda de informação relevante.

Os dados de entrada no sistema CoDiS são representados por uma matriz TF (*term frequency*), na qual os termos que aparecem em um documento são ponderados em função de sua frequência. Outra forma de representação leva em consideração não apenas a frequência do termo em um documento, mas também a sua frequência em todo o conjunto de dados, atribuindo maior peso (IDF) aos termos mais "raros", isto é, àqueles que aparecem menos vezes no conjunto de documentos. Além disso, no CoDiS, o modelo de classificador utilizado foi o SVM com *kernel* linear. Outros classificadores podem ser utilizados a fim de avaliar a robustez do sistema proposto. Dentre estes classificadores pode-se citar árvores de decisão.

Neste trabalho, são avaliadas diferentes configurações entre classificadores, formatos de entrada dos dados e métricas de dissimilaridade. Os resultados apontam que existem significativas diferenças de desempenho do modelo CoDiS dentre as combinações testadas. Os estudos sugerem que aplicar ao CoDiS o classificador SVM, combinado ao formato TFIDF e à distância cosseno, resulta em melhor desempenho quando comparado a outras configurações.

1.1 Motivação e Objetivos

Extrair conhecimento de bases de dados textuais tornou-se uma tecnologia altamente valiosa e aplicada em áreas diversas, tais como, *Business Intelligence*, análise de sentimento, monitoramento de redes sociais, segurança, entre outras. Ao ser comparado com outros modelos para mineração de texto na literatura, CoDiS apresentou um melhor micro-F1 e macro-F1 para quarenta e sete dos conjuntos de dados analisados. Após a aplicação de testes estatísticos, observou-se que o CoDiS foi melhor em aproximadamente 60% dos conjuntos de dados.

Diante da performance promissora do modelo CoDiS, torna-se importante avaliar seu desempenho considerando diferentes aspectos. Este trabalho tem como objetivo investigar três aspectos sobre o modelo CoDiS: (a) qual melhor formato de entrada para os dados entre TF e TFIDF?; (b) qual melhor métrica para geração dos conjuntos de Representação entre distância euclidiana e distância cosseno?; (c) qual melhor classificador a ser utilizado entre SVM e Árvore de Decisão?

1.2 Estrutura

Este trabalho é dividido da seguinte forma: a próxima seção faz uma revisão da literatura, apontando trabalhos relacionados, que lidam com problemas como a grande quantidade de características e os dados esparsos, além de combinarem classificadores buscando melhores resultados. A Seção 3 apresenta o método CoDiS, explicando seu funcionamento. Em seguida, na Seção 4, é explicado como os experimentos foram realizados, as ferramentas, as bases de dados utilizadas e os resultados. Por fim, a conclusão, na Seção 5.

2 Trabalhos Relacionados

Esta seção visa discutir alguns trabalhos relacionados à categorização de textos na literatura, com foco em três aspectos: i) a redução de características, dada através de métodos de transformação de atributos, ii) o tratamento de dados esparsos e iii) a utilização de métodos *ensemble*, dada através da combinação de classificadores.

Özgür e Güngör (Özgür and Güngör, 2010) propuseram um método que objetiva reduzir a dimensionalidade do vetor de atributos, tipicamente grande no *bag of words* (BoW), de uma forma orientada aos dados. Um parâmetro chamado de *Pruning Level* é utilizado para determinar qual deve ser a quantidade final de atributos.

Outros trabalhos, além de oferecer um método de tratamento para a alta dimensionalidade dos dados, alinham técnicas para tratar a escassez que ocorre na matriz de características. Altinel et al. (Altinel, Diri and Ganiz, 2015) introduziu um *kernel* semântico para o SVM através da mudança na ortogonalidade do vetor de atributos e da ponderação dos termos, feita com base nas classes. Jun et al. (Jun, Park and Jang, 2014) propõe uma técnica que se utiliza de *singular value decomposition* e *principal component analysis*. Uma técnica semelhante é proposta por Pinheiro et al. (Pinheiro, Cavalcanti and Ren, 2015), na qual a similaridade cosseno é combinada com métodos de seleção de protótipos para a reduzir a dimensionalidade e escassez no conjunto de dados.

Há também técnicas que abordam o problema da alta dimensionalidade utilizando métodos *ensemble*, isto é, combinando classificadores para potencializar os resultados da categorização de textos. Prabowo e Thelwall (Prabowo and Thelwall, 2009) usam um conjunto com poucos classificadores combinados, onde existem modelos baseado em regras e SVM. Destes classificadores, apenas um (aquele baseado em estatística), realiza seleção de características. Wang et al. (Wang et al., 2014) analisou três formas de geração de *ensembles* (*Bagging*, *Boosting* e *Random Subspace*) utilizando cinco tipos de classificadores base: *Naive Bayes*, *Maximum Entropy*, *Decision Tree*, *k Nearest Neighbors* e SVM. Dentre as configurações avaliadas, aquela que mais se destacou foi *Random Subspace* com SVM. Esta configuração lida com o problema da alta dimensionalidade dos dados já que cada classificador é treinado utilizando um subconjunto aleatório de características. Wu et al. (Wu et al., 2014) propôs um método chamado *ForestTexter* que combina os conceitos de SVM e *Random Forest* abordando problemas de conjuntos de dados textuais com classes desbalanceadas, onde cada nó da árvore é um classificador SVM. Já Onan et al. (Onan, Korukoğlu and Bulut, 2016) usa extração de palavras-chave para reduzir a quantidade de características.

3 Metodologia

O modelo CoDiS (Pinheiro, Cavalcanti and Tsang, 2017), utilizado neste trabalho, é baseado em múltiplos classificadores treinados com base em diferentes Espaços de Dissimilaridade, mais detalhes sobre as Representações de Dissimilaridade podem ser encontrados em Pekalska e Duin (Pekalska e Duin, 2002). Para entender o modelo CoDiS (Seção 3.2), é importante revisar o conceito de Representação de Dissimilaridade (Seção 3.1).

3.1 Representação de Dissimilaridade

A Representação de Dissimilaridade é um mapeamento definido calculando a dissimilaridade de cada documento em uma base de dados **X** para todos os documentos de uma outra base de dados **R**, chamada de conjunto de representação. Este conjunto é composto por Q elementos chamados protótipos, selecionados a partir de **X**, através de algum método de seleção de protótipos. A figura abaixo mostra um exemplo hipotético da geração de **R** a partir de **X**.

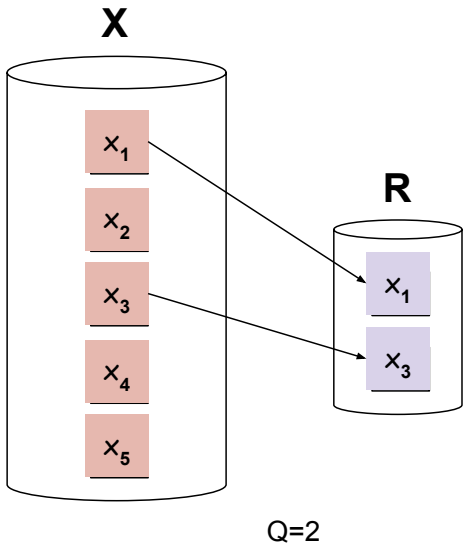

Figura 2: Geração de R a partir de X, considerando Q=2

Considerando o conjunto de representação $R = \{p_1, ..., p_Q\}$ com Q elementos, todos extraídos da base **X**, que, por sua vez, é composta por N documentos representados por BoW como $w^{x_i} = [w_1^{x_i}, w_2^{x_i}, ..., w_V^{x_i}]$, onde $w_h^{x_i}$ é a h-ésima característica de x_i, e V é o tamanho do vocabulário. A Representação de Dissimilaridade de um documento é dado por $D(x_i, R) = [d(x_i, p_1), d(x_i, p_2), ..., d(x_i, p_Q)]$, onde cada $d(\cdot, \cdot)$ é calculada utilizando uma métrica de dissimilaridade escolhida. Neste trabalho, utilizam-se as métricas distância cosseno e distância euclidiana, que são definidas a seguir, onde $w_h^{x_a}$ e $w_h^{x_b}$ são as h-ésimas características dos documentos x_a (pertencente à base **X**) e x_b (pertencente à base **R**) respectivamente.

Para a distância euclidiana:

$$d_{euclidiana}(x_a, x_b) = \sqrt{\sum_{h=1}^{V} (w_h^{x_a} - w_h^{x_b})^2} \tag{1}$$

Para a distância cosseno:

$$d_{cosseno}(x_a, x_b) = 1 - \frac{\sum_{h=1}^{V} w_h^{x_a} w_h^{x_b}}{\sqrt{\sum_{h=1}^{V} (w_h^{x_a})^2} \sqrt{\sum_{h=1}^{V} (w_h^{x_b})^2}} = 1 - similaridade_{cosseno} \quad (2)$$

Quando consideradas as Representações de Dissimilaridade de cada documento em conjunto, é possível montar uma matriz de dissimilaridade semelhante à da figura abaixo.

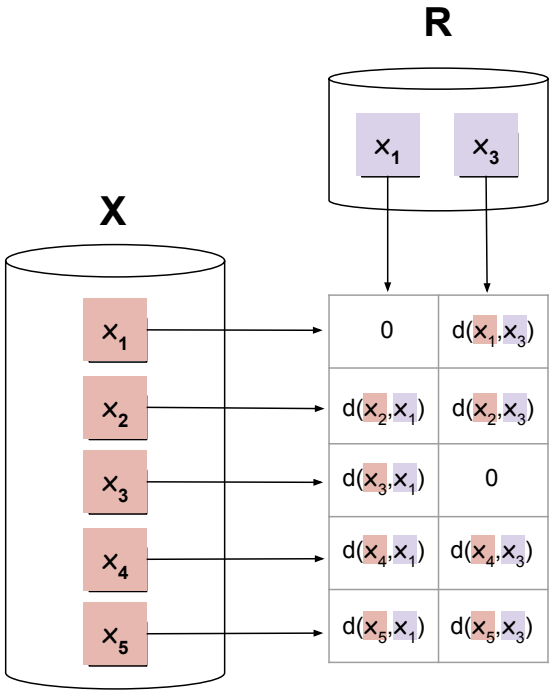

Figura 3: Exemplo de geração da matriz de Dissimilaridade a partir das bases X e R

A matriz possui N linhas, referente à quantidade de documentos na base $\mathbf{X}$, e Q colunas, referente à quantidade de documentos na base $\mathbf{R}$. A matriz de dissimilaridade pode então ser usada como entrada para os modelos classificadores, onde cada linha representa uma instância (documento) e cada coluna uma característica.

A representação dos dados por matriz de dissimilaridade reduz os problemas apontados anteriormente. Em relação à alta dimensionalidade do vetor de atributos, esta é reduzida uma vez que, ao invés de utilizar todas os termos presentes no *corpus*, a Representação por Dissimilaridade utiliza apenas Q característica (o tamanho do conjunto de representação). Além disso, a redução da quantidade de características torna a quantidade de atributos mais proporcional em relação à quantidade de documentos do *corpus*. Por fim, a matriz de dados diminui sua escassez. Isso ocorre porque utilizando a matriz de dissimilaridade o valor de um atributo será zero apenas se os vetores forem iguais, o que é raro.

3.2 CoDiS

É fornecida uma base de dados de documentos, *corpus*, onde cada palavra é representada em termos de sua frequência de aparição naquele documento, utilizando a técnica *Bag of Words*.

Em seguida, será explicada a aplicação do método CoDiS nas fases de treinamento e teste.

3.2.1 Treinamento

A fase de treinamento possui 4 módulos: *Bootstrapping*, *Representation Sets Generation*, *Representation Transformation* e *Classifier Training*. É importante ressaltar que o conjunto de dados **X** é composto de N documentos $x_i \in X = \{x_1, x_2, ..., x_N\}$, onde cada x_i é representado por um par formado por um vetor de características e uma classe associada àquela instância.

1. *Bootstrapping*

   Esta fase recebe como entrada a quantidade L de classificadores e o conjunto de dados **X**. São então gerados L diferentes subconjuntos do conjunto **X**, com reposição como em *Bagging*. Todos os subconjuntos tem o mesmo tamanho N do conjunto **X** original, com alguns documentos aparecendo mais de uma vez. A Figura 4 ilustra a geração dos conjuntos X_{ls}.

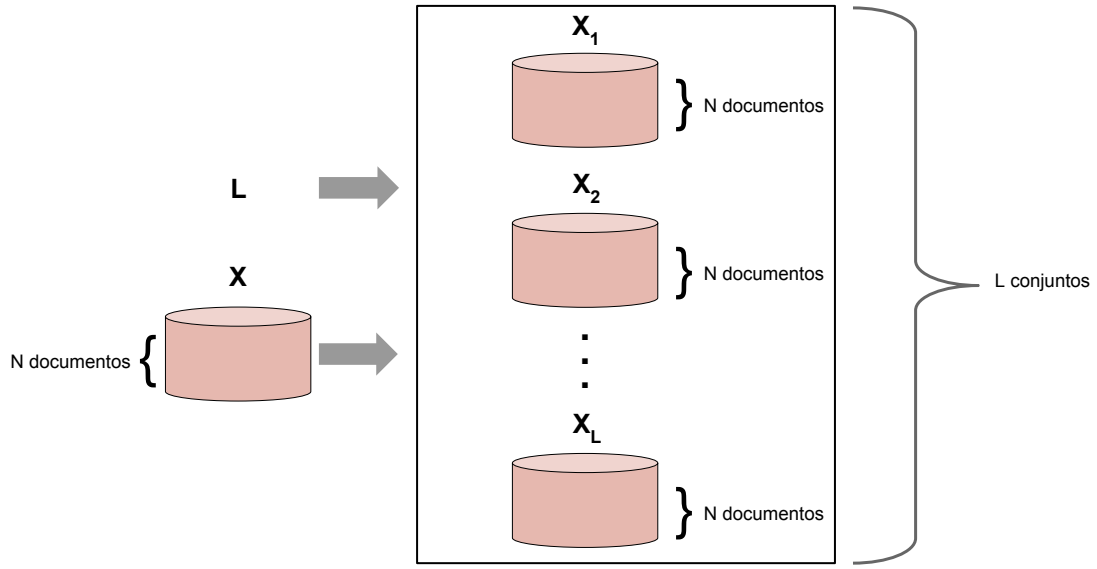

Figura 4: Módulo Bootstrapping gera L conjuntos a partir de X

2. *Representation Sets Generation*

   Este módulo gera os L conjuntos de Representação. A seleção dos protótipos para os conjuntos de Representação é feita a partir do conjunto de treinamento **X**, de forma aleatória. Cada conjunto de Representação possui Q documentos, como mostrado na figura a seguir.

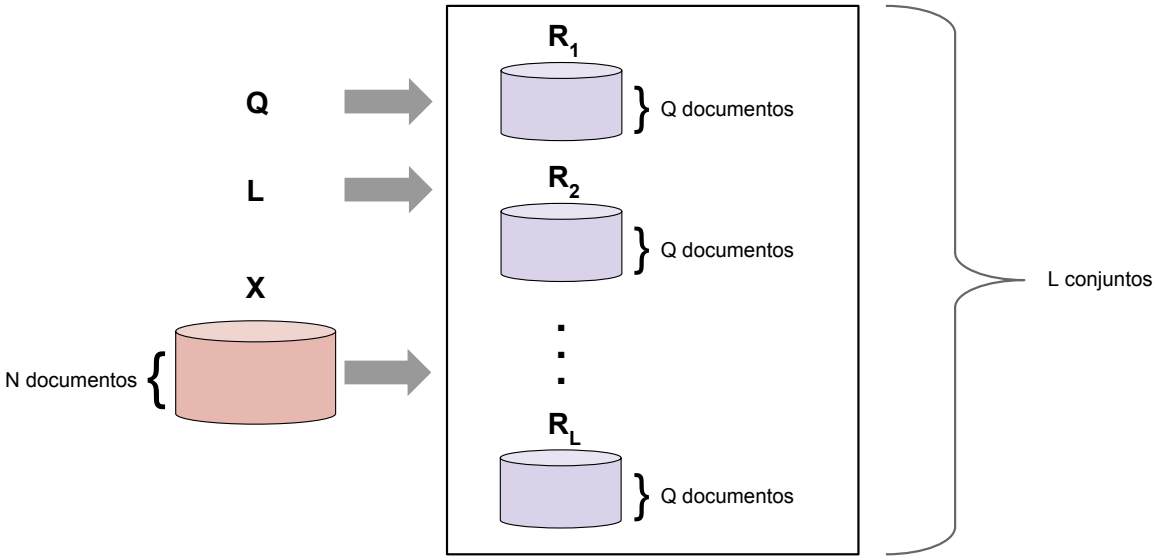

Figura 5: Módulo de geração dos conjuntos de Representação

Ao final desses módulos, o CoDiS gerou $2 \times L$ subconjuntos: L a partir do módulo *Bootstrapping* $[X_1, X_2, ..., X_L]$, onde cada X_l possui N documentos, e L do módulo de *Representation Sets Generation* $[R_1, R_2, ..., R_L]$, onde cada R_l possui Q documentos.

3. *Representation Transformation*

O módulo de *Representation Transformation* recebe como entrada dois conjuntos: um X_l, proveniente do módulo de *Bootstrapping*, e um R_l, proveniente do módulo de geração dos conjuntos de Representação. Esses vetores serão utilizados para gerar a matriz de dissimilaridade D_l correspondente, como mostrado na Figura 6.

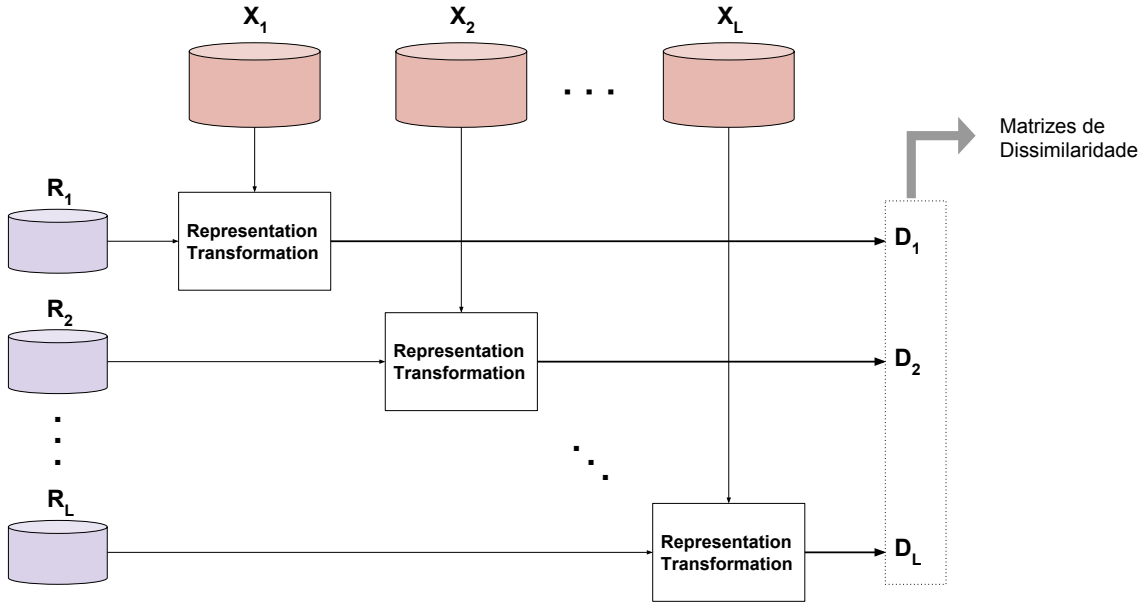

Figura 6: Módulo de geração das matrizes de Dissimilaridade

A matriz de Dissimilaridade assume a forma:

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

sendo cada $d(x_n, p_q)$, a q-ésima característica do documento x_n. Cada x_n é um documento de uma base X_l e cada p_q é um documento de uma base R_l.

4. *Classifier Training*

Este módulo consiste no treinamento dos L modelos classificadores, cada um que recebe como entrada os dados provenientes da matriz de dissimilaridade D_l correspondente, como na Figura 7.

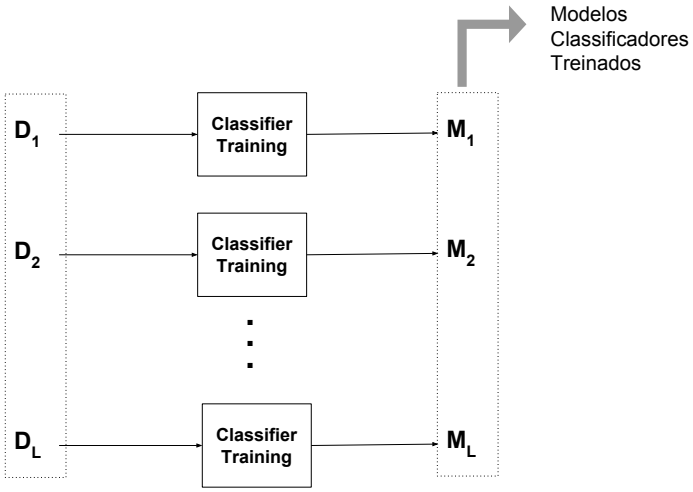

Figura 7: Módulo de Treinamento dos Classificadores

O processo completo de treinamento é mostrado na figura abaixo.

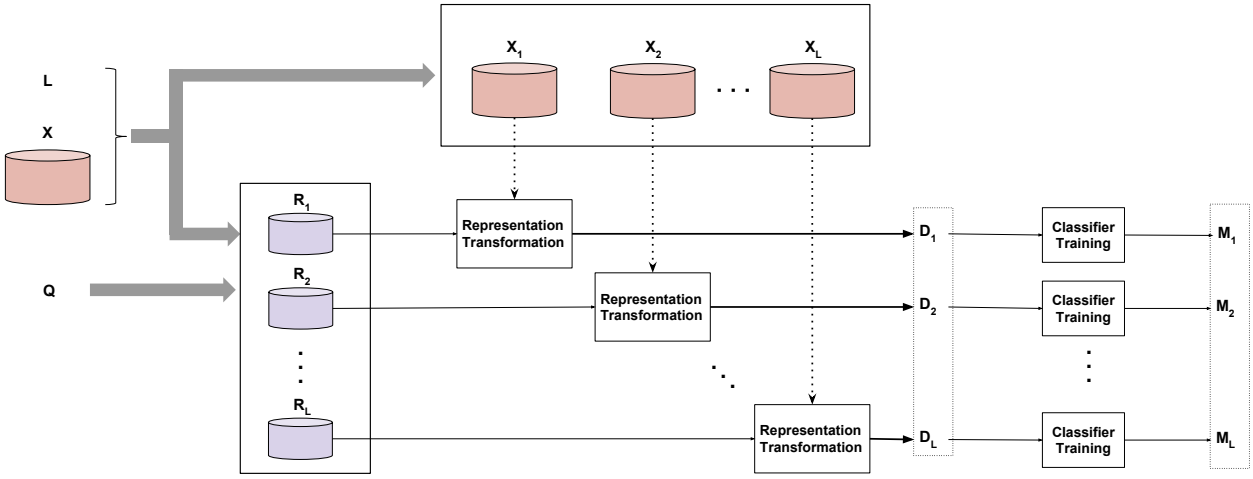

Figura 8: Fase de Treinamento do CoDiS e seus módulos

3.2.2 Teste

A fase de teste possui 3 módulos: *Representation Transformation*, *Classifier* e *Fuser*.

1. *Representation Transformation*

   Neste módulo, são gerados L vetores de dissimilaridade, D_{ls}, que representam o documento de entrada y modificado. O vetores são Representações de Dissimilaridade entre o documento y e os R_{ls} conjuntos correspondentes. Os conjuntos R_{ls} utilizados na fase de teste são os mesmos gerados na fase de treinamento. Os elementos do vetor - referente ao documento modificado - representam a distância entre o documento original y e um documento p_q do conjunto R_l correspondente, como a seguir:

   $$D(y, R_l) = [d(y, p_1), d(y, p_2), ..., d(y, p_Q)]$$

   onde existem Q características.

2. *Classifier*

   Neste módulo, os L vetores de dissimilaridade, gerados anteriormente, são apresentados aos L modelos classificadores treinados. Cada classificador produz uma saída correspondente à possível classe do documento y fornecido como entrada.

3. *Fuser*

   As classes previstas pelos L classificadores passam por esse módulo e através de voto majoritário (Kuncheva, 2004), é escolhida a classe "vencedora", prevista para o documento y de entrada.

A Figura 9 mostra o funcionamento da fase de teste.

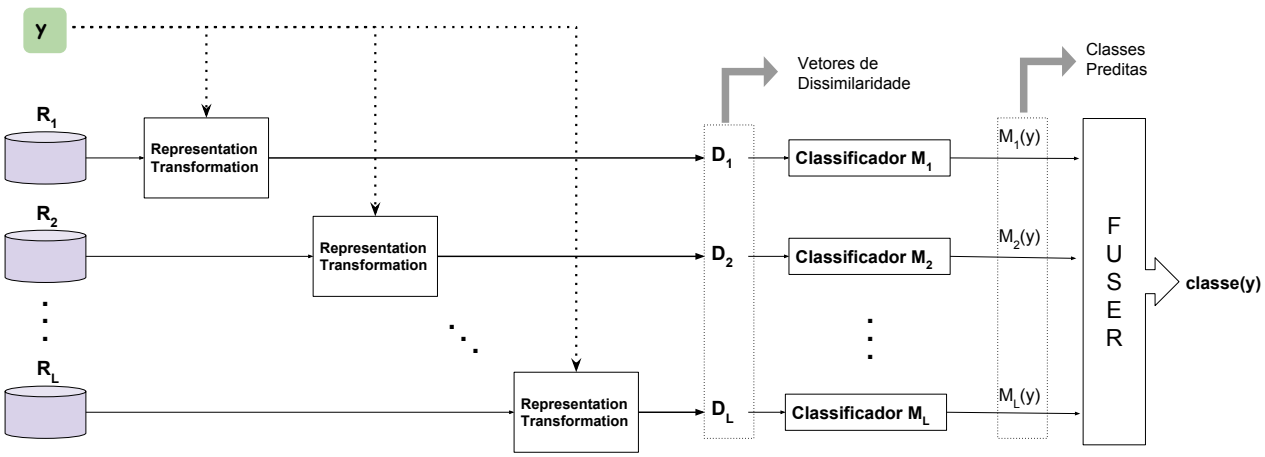

Figura 9: Fase de Teste do CoDiS e seus módulos

4 Experimentos

Nesta seção, são apresentados os experimentos realizados com 25 bases de dados utilizando *10-fold stratified cross-validation* e os classificadores SVM e Árvore de Decisão. Para cada base de dados, foi escolhida uma quantidade L de classificadores. A escolha do melhor L por base foi feita considerando o artigo *Combining dissimilarity spaces for text categorization* (Pinheiro, Cavalcanti and Tsang, 2017). Neste artigo, os autores apresentam resultados de experimentos realizados com 47 bases, e para cada uma delas, diferentes valores de L foram testados. Foi então escolhido o melhor valor por base considerando o desempenho final do modelo SVM. Nesta seção, serão abordadas as ferramentas utilizadas para implementação do CoDiS, bem como as medidas utilizadas para avaliação do desempenho do modelo, a descrição das bases utilizadas, algumas considerações sobre o experimento e os resultados.

4.1 Ferramentas Utilizadas

Para implementação do modelo, foi utilizada a linguagem de programação Python, uma linguagem de alto nível, escolhida neste trabalho por conta da facilidade que ela oferece para manipulação dos dados através de bibliotecas *open-source*, como a scikit-learn (Pedregosa et al., 2011), a NumPy e a SciPy (Jones, Oliphant and Peterson, 2001). A scikit-learn é uma biblioteca que oferece suporte para utilização de algoritmos de aprendizagem de máquina e é projetada para interagir com outras bibliotecas numéricas e científicas como a NumPy e a SciPy, que, por sua vez, funcionam muito bem de maneira integrada. A biblioteca NumPy oferece recursos para manipulação de vetores e matrizes e, quando combinada à SciPy, torna-se possível fazer uso de várias rotinas facilitadoras. Além disso, foi utilizado o *software* Matlab para os testes estatísticos aplicados aos resultados dos experimentos.

4.2 Medidas usadas na avaliação de desempenho

O desempenho dos modelos foi avaliada em termos das métricas micro-F1 e macro-F1 (Sebastiani, 2002), F1 é calculada usando:

$$F1 = 2 \times Recall \times \frac{Precision}{(Recall + Precision)} \tag{3}$$

As definições de *Recall* e *Precision* para micro F1 e macro F1 são dadas nas fórmulas abaixo, sendo C o número de classes, TP_k é a quantidade de documentos classificados corretamente na classe c_k, FN_k é o número de documentos incorretamente classificados que foram atribuídos a outras classes e não à c_k, e FP_k é o número de documentos classificados incorretamente e foram atribuídos à classe c_k.

Temos então para micro F1:

$$Precision_{micro} = \frac{\sum_{k=1}^{C} TP_k}{(\sum_{k=1}^{C} TP_k + \sum_{k=1}^{C} FN_k)} \tag{4}$$

$$Recall_{micro} = \frac{\sum_{k=1}^{C} TP_k}{(\sum_{k=1}^{C} TP_k + \sum_{k=1}^{C} FP_k)} \tag{5}$$

E para macro F1:

$$Precision_{macro} = \frac{\sum_{k=1}^{C} \frac{TP_k}{(TP_k + FN_k)}}{C} \tag{6}$$

$$Precision_{macro} = \frac{\sum_{k=1}^{C} \frac{TP_k}{(TP_k + FP_k)}}{C} \tag{7}$$

4.3 Descrição das Bases

Foram utilizadas 25 bases de dados neste trabalho. A Tabela 1 mostra algumas características das bases, tais como quantidade de documentos e atributos para cada uma delas, e o valor L utilizado no modelo CoDiS. As bases foram obtidas em `http://sites.labic.icmc.usp.br/text_collections/`.

Tabela 1: Descrição das bases

Base	Documentos	Atributos	Categorias	L
Classic3	3891	7749	3	20
CSTR	299	1726	4	90
FBIS	2463	2001	17	60
Hitech	2301	12942	6	80
Irish Sentiment (Irish)	1660	8659	3	60
La1s	3204	13196	6	100
La2s	3075	12433	6	80
Oh0	1003	3183	10	100
Oh5	918	3013	10	80
Oh10	1050	3239	10	70
Oh15	913	54142	10	40
Re0	1504	2887	13	50
Re1	1657	3759	25	90
Review Olarity (Polarity)	2000	15697	2	80
Reviews	4069	22927	5	90
SyskillWebert (SsW)	334	4340	4	20
Tr11	414	6430	9	50
Tr12	313	5805	8	60
Tr21	335	7903	6	70
Tr23	204	5833	6	70
Tr31	927	10129	7	40
Tr41	878	7455	10	100

Tr45	690	8262	10	60
WAP	1560	8461	20	70
WebACE	3900	8880	21	100

4.4 Considerações sobre o experimento

Primeiramente analisaremos algumas considerações feitas pelo artigo proposto por Pinheiro et al. para então abordarmos os aspectos experimentais deste trabalho.

4.4.1 Considerações Preliminares

No artigo proposto por Pinheiro et al., algumas questões consideradas foram: (i) qual deveria ser o algoritmo de seleção de protótipos para o módulo de geração dos conjunto de representação (*Representation Sets Generation*)?; (ii) qual seria a melhor métrica para este módulo (considerando a distância euclidiana e a similaridade cosseno) ?; (iii) qual o melhor tamanho Q para o conjunto de representação no CoDiS?

Para resolver a primeira questão, os autores testaram três diferentes categorias determinísticas de algoritmos para seleção de protótipos: condensação, edição e híbrido. Além destes, também foram considerados métodos randômicos para seleção de protótipos. Os modelos determinísticos foram combinados e testados contra a combinação dos modelos randômicos sobre algumas bases de dados, considerando as métricas de distância euclidiana e distância cosseno. Os resultados mostraram que a performance de métodos determinísticos e métodos não determinísticos é similar. Sendo assim, e tendo em vista que os métodos determinísticos são mais lentos do que métodos não determinísticos, os autores concluíram que modelos randômicos são mais satisfatórios.

A segunda questão abordada, tinha como objetivo identificar qual métrica deveria ser utilizada no módulo de *Representation Transformation*. Foram analisadas a distância euclidiana e a similaridade cosseno. Os autores variaram o tamanho do conjunto de representação, utilizando o modelo randômico para seleção dos protótipos. É de conhecimento comum que a similaridade cosseno é, geralmente, melhor que a distância euclidiana em problemas de classificação de documentos. No entanto, os autores declaram que para o CoDiS, a distância euclidiana mostrou resultados melhores.

Para a terceira questão, relativa ao melhor tamanho Q para o conjunto de Representação no CoDiS, os experimentos preliminares mostraram que este conjunto deveria conter 20% das amostras da base de treinamento. O valor $Q = 0,2 \times N$, mostrou-se o melhor equilíbrio entre diversidade e performance, sendo N o tamanho da base de treinamento.

4.4.2 Aspectos Experimentais

Considerando as questões pontuadas anteriormente, este trabalho as aborda da seguinte maneira: (i) segue a conclusão dos autores, selecionando os protótipos para os conjuntos de Representação de forma aleatória; (ii) aprofunda essa questão, realizando uma nova análise comparativa, desta vez entre a distância euclidiana e a distância cosseno, aplicando novas configurações ao modelo CoDiS; (iii) segue as conclusões do artigo de Pinheiro et al., utilizando a mesma definição para o valor de Q.

Este trabalho tem como objetivo investigar três aspectos sobre o modelo CoDiS: (a) qual melhor formato de entrada para os dados entre TF e TFIDF?; (b) qual melhor métrica para geração dos conjuntos de Representação

entre distância euclidiana e distância cosseno?; (c) qual melhor classificador a ser utilizado entre SVM e Árvore de Decisão?

Para realizar (a) são testados dois formatos de entrada para os dados: o TF (*term frequency*), utilizado também no artigo citado, e o TFIDF (*term frequency - inverse document frequency*), que se caracteriza como uma nova abordagem em relação ao artigo.

O valor TFIDF indica a importância de uma palavra em um documento considerando uma coleção de documentos. Ele é calculado utilizando a regra a seguir:

$$TFIDF = TF \times IDF \tag{8}$$

para uma palavra p e um documento d, tem-se que:

$$TF(p, d) = quant.\ de\ vezes\ em\ que\ p\ aparece\ em\ d \tag{9}$$

e, seja

$$IDF(p) = \log\Big(\frac{1 + quant.\ total\ de\ documentos\ no\ corpus}{1 + quant.\ de\ documentos\ que\ contém\ p}\Big) + 1 \tag{10}$$

A ideia por trás da representação TFIDF, é que palavras mais raras, recebam um peso maior e consequentemente um maior valor TFIDF, contribuindo de forma mais significativa para a representação do documento neste formato.

O aspecto (b) visa comparar as distâncias euclidiana e cosseno, caracterizando-se como um aprofundamento da questão (ii) citada anteriormente, uma vez que, tipicamente, a similaridade cosseno apresenta melhores resultados em problemas de categorização de textos.

Já em relação ao aspecto (c) serão comparadas as performances dos classificadores SVM e Árvore de Decisão (AD). O objetivo, neste caso, é verificar se a performance da AD é tão boa quanto ou melhor que a do SVM, tendo em vista que a AD é um modelo mais legível e rápido para ser treinado, a utilização deste classificador poderia ser útil para bases de dados com muitos documentos. Além disso, sistemas com múltiplos classificadores tendem a se beneficiar de modelos classificadores mais "fracos", como é o caso da Árvore de Decisão.

Sendo assim, a tabela abaixo sumariza as configurações a serem testadas neste trabalho.

Tabela 2: Combinações para Teste

Cenário	Classificador	Formato de Entrada	Métrica de Dissimilaridade
I	SVM	TF	Distância Euclidiana
II	SVM	TF	Distância Cosseno
III	SVM	TFIDF	Distância Euclidiana
IV	SVM	TFIDF	Distância Cosseno
V	AD	TF	Distância Euclidiana
VI	AD	TF	Distância Cosseno
VII	AD	TFIDF	Distância Euclidiana
VIII	AD	TFIDF	Distância Cosseno

4.5 Resultados

Inicialmente, cada uma das bases de dados analisadas, foi dividida em 10 *folds*. Os experimentos com o CoDiS foram realizados para as 10 subdivisões e os respectivos valores macro F1 e micro F1 foram anotados. Ao final dos experimentos por base, obteve-se a média para cada métrica. A tabela abaixo mostra um exemplo dos resultados para a base *Classic3* utilizando o classificador SVM, o formato TF e a distância euclidiana (DE).

Tabela 3: Resultados para a base *Classic3* utilizando SVM—TF—DE

Fold	macro F1	micro F1
1	0,9949273772	0,9948717949
2	0,9797480444	0,9794871795
3	0,9839531408	0,9846153846
4	0,981381984	0,9820051414
5	0,9790898801	0,9794344473
6	0,9949116669	0,9948586118
7	0,9870073384	0,9871465296
8	0,9976676004	0,9974293059
9	0,9869012028	0,9871134021
10	0,9972363273	0,9974226804
Média	0,9882824562	0,9884384477

Na tabela, são mostrados apenas os resultados para uma das 8 configurações. É importante ressaltar, no entanto, que os resultados por base foram gerados para as todas as configurações mostradas na Tabela 2. E ao final dos experimentos, os resultados foram sumarizados como na tabela a seguir, na qual cada linha representa uma das bases de dados testada, e cada coluna uma das 8 configurações testada. Os valores nas células correspondem aos valores médios calculados anteriormente para cada configuração. Os melhores valores por linha aparecem em negrito. Na tabela, **maF1** representa macro F1, enquanto **miF1** representa micro F1.

Tabela 4: Resultados Médios por Base

	SVM-TF-DE		SVM-TF-DC		SVM-TFIDF-DE		SVM-TFIDF-DC		AD-TF-DE		AD-TF-DC		AD-TFIDF-DE		AD-TFIDF-DC	
	maF1	miF1	maF1	miF1	maF1	miF1	maF1	miF1	maF1	miF1	maF1	miF1	maF1	miF1	maF1	miF1
Classic3	0,988	0,988	0,990	0,991	0,991	0,991	**0,993**	**0,993**	0,950	0,951	0,977	0,978	0,983	0,984	0,982	0,982
CSTR	0,816	0,806	0,824	0,823	0,757	0,792	**0,832**	**0,826**	0,644	0,659	0,818	0,800	0,803	0,799	0,810	0,806
FBIS	0,657	0,795	0,780	0,856	0,773	0,852	**0,798**	**0,863**	0,642	0,767	0,716	0,822	0,770	0,841	0,763	0,839
Hitech	0,701	0,754	**0,717**	**0,765**	0,681	0,751	0,702	0,759	0,548	0,650	0,657	0,724	0,650	0,727	0,653	0,728
Irish	0,646	0,674	0,660	0,685	0,650	0,673	**0,673**	**0,693**	0,549	0,577	0,614	0,638	0,606	0,627	0,604	0,625
La1s	0,822	0,864	**0,862**	**0,887**	0,823	0,868	0,850	0,882	0,734	0,788	0,814	0,847	0,814	0,856	0,818	0,858
La2s	0,846	0,880	**0,884**	**0,908**	0,852	0,890	0,871	0,902	0,763	0,813	0,825	0,861	0,835	0,872	0,837	0,871
Oh0	0,840	0,873	0,852	0,878	0,855	0,884	**0,880**	**0,898**	0,642	0,700	0,785	0,824	0,824	0,859	0,814	0,852
Oh5	0,875	0,883	0,881	0,890	0,896	0,900	**0,901**	**0,906**	0,675	0,706	0,828	0,843	0,852	0,864	0,841	0,854
Oh10	0,753	0,809	0,767	0,810	0,749	0,810	**0,781**	**0,828**	0,598	0,657	0,710	0,761	0,741	0,789	0,743	0,790
Oh15	0,787	0,809	0,802	0,823	0,815	0,833	**0,835**	**0,849**	0,565	0,624	0,717	0,751	0,749	0,777	0,756	0,785
Re0	0,680	0,845	0,768	0,871	0,794	0,868	**0,821**	**0,874**	0,540	0,773	0,664	0,819	0,746	0,838	0,749	0,839
Re1	0,644	0,848	0,718	0,880	0,697	0,861	**0,747**	**0,883**	0,426	0,715	0,639	0,832	0,668	0,848	0,672	0,847
Polarity	0,808	0,809	**0,823**	**0,824**	0,784	0,785	0,806	0,807	0,633	0,633	0,687	0,687	0,737	0,738	0,733	0,733
Reviews	0,914	0,950	**0,933**	**0,958**	0,920	0,955	**0,933**	**0,958**	0,839	0,907	0,904	0,942	0,908	0,945	0,907	0,946
SsW	0,946	0,949	**0,952**	**0,955**	0,947	0,948	0,950	0,951	0,876	0,885	0,925	0,933	0,924	0,933	0,934	0,942
Tr11	0,517	0,757	**0,712**	**0,872**	0,627	0,846	0,689	0,866	0,612	0,806	0,700	0,854	0,695	0,854	0,682	0,854
Tr12	0,650	0,766	0,785	**0,850**	0,757	0,842	0,763	0,847	0,665	0,741	**0,802**	0,837	0,771	0,824	0,748	0,831
Tr21	0,263	0,708	0,581	0,896	0,508	0,872	0,523	0,875	0,458	0,828	**0,671**	0,902	0,635	**0,905**	0,635	0,902
Tr23	0,424	0,680	**0,834**	0,897	0,788	0,887	**0,834**	**0,906**	0,594	0,787	0,757	0,860	0,764	0,886	0,800	0,889
Tr31	0,871	0,951	**0,953**	**0,987**	0,951	0,986	0,945	0,983	0,810	0,897	0,933	0,975	0,925	0,969	0,924	0,968
Tr41	0,840	0,942	0,871	0,958	0,884	0,957	**0,886**	**0,959**	0,721	0,866	0,828	0,933	0,840	0,943	0,839	0,943
Tr45	0,685	0,828	0,861	**0,936**	0,830	0,926	0,853	0,935	0,731	0,838	0,836	0,913	0,877	0,935	**0,878**	0,933
WAP	0,614	0,822	**0,641**	**0,843**	0,558	0,799	0,634	0,832	0,510	0,726	0,557	0,786	0,572	0,798	0,572	0,798
WebACE	0,583	0,873	**0,632**	**0,891**	0,552	0,863	0,629	0,888	0,500	0,805	0,553	0,852	0,568	0,860	0,572	0,861
Média	0,727	0,834	0,803	0,877	0,778	0,866	**0,805**	**0,879**	0,649	0,764	0,757	0,839	0,770	0,851	0,771	0,851

Os resultados em negrito na Tabela 4 mostram o melhor macro F1 e micro F1 por base. É possível notar que a maioria dos valores ressaltados corresponde às configurações SVM-TF-DC e SVM-TFIDF-DC. Além disso, a última linha da tabela mostra os valores médios para cada configuração.

As Figuras 10, 11 e 12 utilizam os valores médios em gráficos com objetivo de comparar o desempenho das diferentes configurações, dando um ideia geral das possíveis melhores combinações.

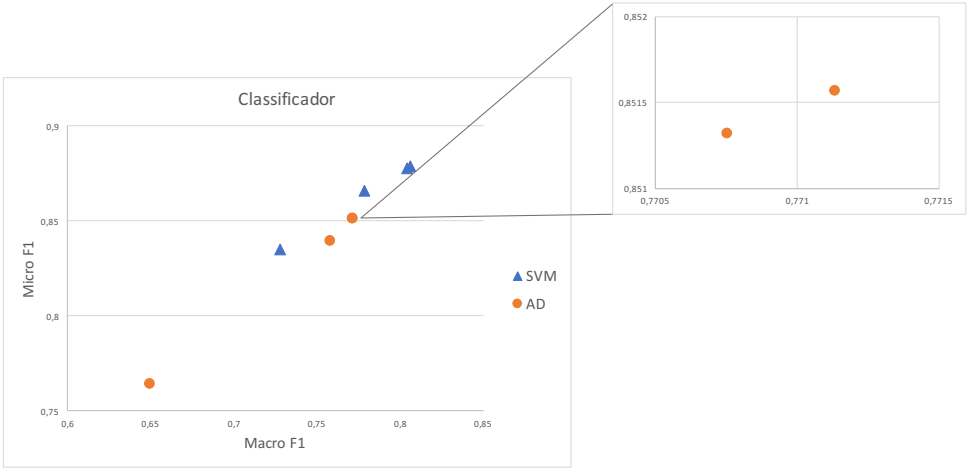

Figura 10: Gráfico dos valores médios por configuração SVMxAD

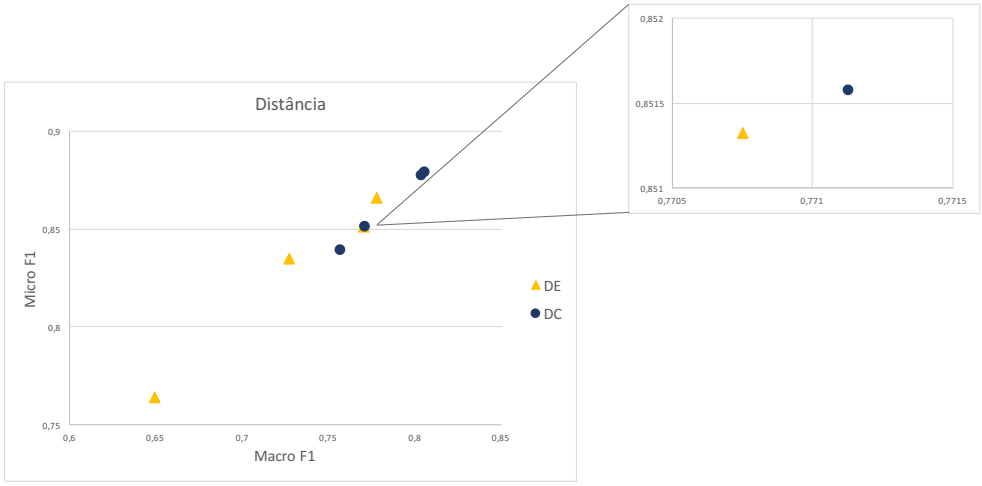

Figura 11: Gráfico dos valores médios por configuração DExDC

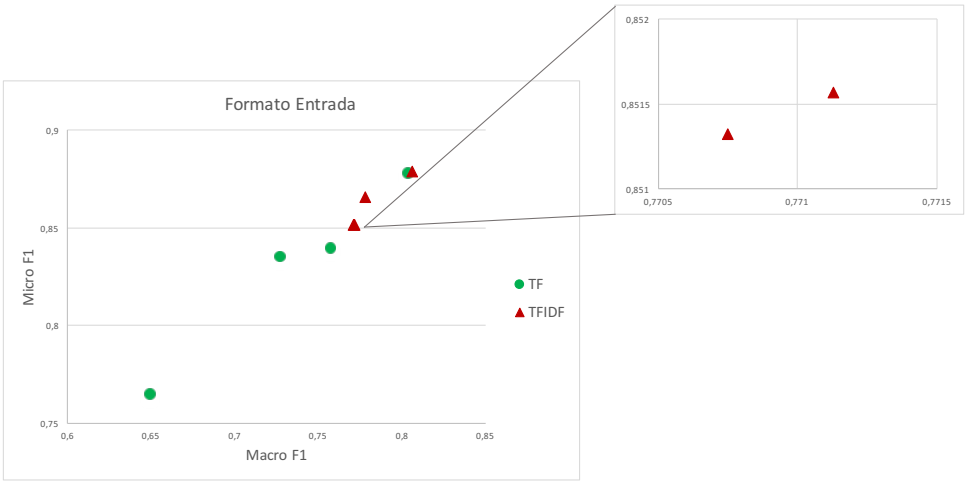

Figura 12: Gráfico dos valores médios por configuração TFxTFIDF

O gráfico ampliado no canto superior de cada imagem acima mostra dois pontos distintos que, por estarem muito próximos, aparecem sobrepostos no gráfico principal.

Em seguida, foram aplicados testes estatísticos para verificar a hipótese sobre quais configurações seriam responsáveis pelo melhor desempenho do modelo CoDiS. Dois testes estatísticos foram empregados: o teste de Wilcoxon e o teste de Friedman.

4.5.1 Teste de Wilcoxon

O objetivo neste caso foi avaliar se havia diferença significativa entre os quesitos (i) classificadores: AD e SVM, (ii) formato de entrada dos dados: TF e TFIDF, (iii) métrica de dissimilaridade utilizada: a distância cosseno e a euclidiana. O teste de Wilcoxon foi aplicado para as métricas macro F1 e micro F1 separadamente, como é mostrado na tabela a seguir.

Tabela 5: Resultados para teste de Wilcoxon p-values

	macro F1	micro F1
Classificador	$2,87 \times 10^{-58}$	$4,41 \times 10^{-84}$
Formato	$1,33 \times 10^{-56}$	$2,32 \times 10^{-66}$
Distância	$1,39 \times 10^{-90}$	$7,06 \times 10^{-88}$

Ao analisar a tabela, percebe-se que os valores p são muito pequenos para todos os aspectos analisado, o que indica, portanto, que existe diferença significativa (i) entre os classificadores SVM e AD, (ii) entre os formatos de entrada TF e TFIDF, (iii) entre as métricas distância cosseno e distância euclidiana.

4.5.2 Teste de Friedman

O teste de Friedman, realizado com um intervalo de confiança de 95%, foi empregado para comparar as 8 configurações mostradas na Tabela 2. Os *ranks* médios indicam que as configurações são diferentes, como mostrado abaixo.

Tabela 6: Resultados para teste de Friedman *ranks* médios

	SVM TF DE	SVM TF DC	SVM TFIDF DE	SVM TFIDF DC	AD TF DE	AD TF DC	AD TFIDF DE	AD TFIDF DC
macro F1	3,94	1,682	2,822	1,38	6,594	4,256	3,616	3,71
micro F1	3,808	1,472	2,536	1,28	6,702	4,606	3,762	3,834

A partir da Tabela 6, é possível obter as configurações em ordem de desempenho para Macro F1 e Micro F1:

Tabela 7: Configurações em ordem de desempenho

Ordem	Macro F1	Micro F1
1	SVM - TFIDF - DC	SVM - TFIDF - DC
2	SVM - TF - DC	SVM - TF - DC
3	SVM - TFIDF - DE	SVM - TFIDF - DE
4	AD - TFIDF - DE	AD - TFIDF - DE
5	AD - TFIDF - DC	SVM - TF - DE
6	SVM - TF - DE	AD - TFIDF - DC
7	AD - TF - DC	AD - TF - DC
8	AD - TF - DE	AD - TF - DE

Sendo assim, torna-se importante a análise dos diagramas com as diferenças críticas entre as configurações, para que seja possível notar quais configurações de fato se destacam em comparação a quais outras.

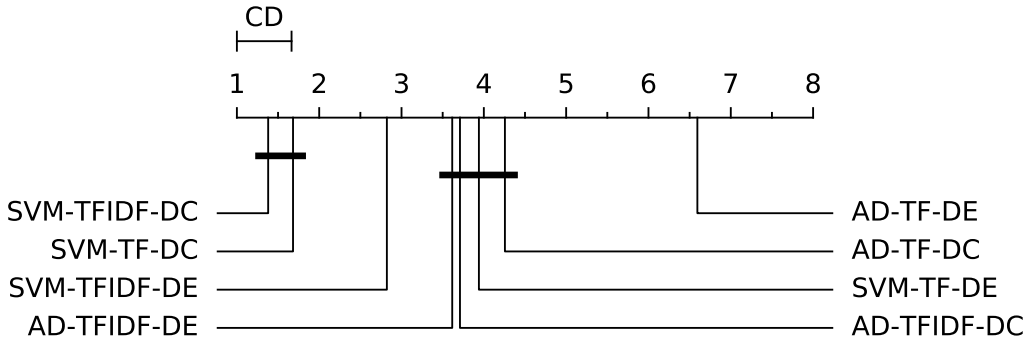

Figura 13: Diagrama com a diferença crítica (CD) para macro F1, CD=0,664

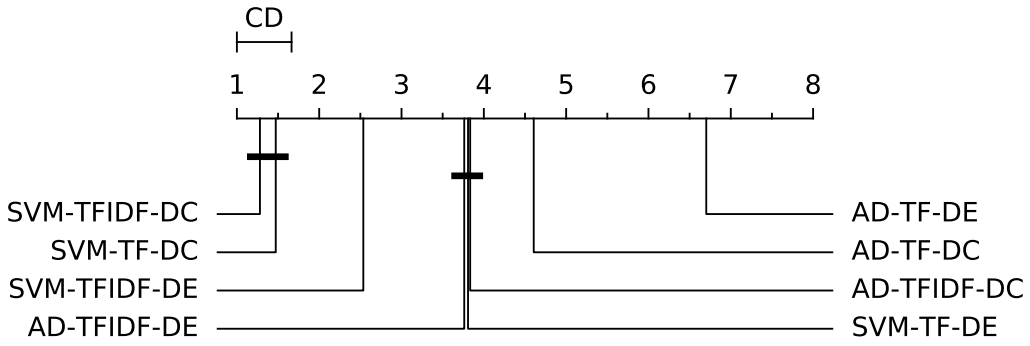

Figura 14: Diagrama com a diferença crítica (CD) para micro F1, CD=0,664

Ao observar as figuras acima, nota-se que a configuração **SVM-TFIDF-DC** é significativamente melhor do que pelo menos 6 outras: AD-TF-DE, AD-TF-DC, AD-TFIDF-DE, AD-TFIDF-DC, SVM-TF-DE e SVM-TFIDF-DE.

Este resultado mostra que o CoDiS, mesmo sendo um sistema de múltiplos classificadores, se beneficiou mais com a utilização de um modelo mais forte como o SVM, em comparação à AD. Além disso, o formato TFIDF representa melhor os dados de entrada, o que era esperado, uma vez que este formato leva em consideração não só a frequência do termo no documento, mas também a quantidade de documentos que contém o termo em questão, o que valoriza palavras mais raras. Por fim, em problemas de classificação de texto, a similaridade cosseno é tipicamente melhor do que a distância euclidiana. A distância cosseno, uma métrica derivada da similaridade cosseno, corrobora este conhecimento, mostrando-se melhor que a distância euclidiana.

4.6 Análise do Tempo Computacional

O custo computacional de algoritmos é um importante aspecto a ser observado em muitas aplicações práticas. Sendo assim, foi realizada uma breve análise do tempo de computação para o modelo CoDiS, considerando uma das bases de dados chamada *Tr23*. O tempo de execução das fases de treinamento e teste foram coletados, separadamente, para cada uma das 8 configurações.

Considerando T_{tr} como o menor tempo de execução observado na fase de treinamento, é possível escrever os resultados como na tabela a seguir, em que o tempo representa o período médio para cada execução do CoDiS, considerando cada *fold*.

Tabela 8: Tempo de uma execução para a base *Tr23* - Fase de Treinamento

Configurações	Tempo
AD - TF - DE	T_{tr}
AD - TFIDF - DE	$1,057 \times T_{tr}$
AD - TF - DC	$1,315 \times T_{tr}$
AD - TFIDF - DC	$1,333 \times T_{tr}$
SVM - TFIDF - DE	$1,846 \times T_{tr}$
SVM - TFIDF - DC	$2,127 \times T_{tr}$
SVM - TF - DC	$2,243 \times T_{tr}$
SVM - TF - DE	$2,245 \times T_{tr}$

Em que $T_{tr} = 9,98$ *segundos*. Da tabela acima, é possível notar que, no geral, a Árvore de Decisão é mais rápida para ser treinada do que o SVM.

Da mesma forma, também foram obtidos os tempos para a fase de teste, como na Tabela 9, na qual T_{te} representa o menor intervalo de execução observado e é igual a $7,18$ *segundos*.

Tabela 9: Tempo de uma execução para a base *Tr23* - Fase de Teste

Configurações	Tempo
AD - TF - DE	T_{te}
AD - TF - DC	$1,008 \times T_{te}$
AD - TFIDF - DE	$1,047 \times T_{te}$
SVM - TF - DE	$1,115 \times T_{te}$
SVM - TFIDF - DE	$1,179 \times T_{te}$
AD - TFIDF - DC	$1,281 \times T_{te}$
SVM - TF - DC	$1,289 \times T_{te}$
SVM - TFIDF - DC	$1,332 \times T_{te}$

Os valores apresentados acima foram obtidos em uma máquina com as seguintes especificações:

- Sistama Operacional: macOS Sierra
- Processador: 2,2 GHz Intel Core i7 - 4 *cores*
- Memória: 8 GB

5 Conclusão

Na literatura a técnica *bag of words* para classificação de documentos é bastante conhecida. No entanto, algumas limitações associadas à esta técnica são: a alta dimensionalidade do vetor de atributos, a grande quantidade de características se comparada à quantidade de instâncias e uma matriz de dados esparsa. Visando reduzir essas limitações foi proposto um sistema para classificação de documentos, chamado CoDiS, no qual múltiplos classificadores são treinados com base em diferentes Representações de Dissimilaridade.

Durante a fase experimental, os autores avaliaram a distância euclidiana e a similaridade cosseno como possíveis métricas de similaridade para o sistema CoDiS. A distância euclidiana superou a similaridade cosseno nos resultados. No entanto, é conhecido na literatura que, para problemas de classificação de texto, a similaridade cosseno produz melhores resultados. Sendo assim, este trabalho aprofundou essa questão realizando uma nova comparação, desta vez entre as distâncias euclidiana e cosseno. Além disso, foram também comparados dois formatos de entrada dos dados: o TF e o TFIDF, e dois classificadores: o SVM e a AD. Foram então aplicadas 8 combinações ao modelo CoDiS para avaliar seu desempenho.

Os resultados dos experimentos mostram que existe uma diferença significativa entre (i) os classificadores AD e SVM, (ii) os formatos de entrada TF e TFIDF, (iii) as métricas de dissimilaridade DE e DC. Além disso, foi possível avaliar as 8 configurações ordenadas em relação ao desempenho, considerando a diferença crítica, tanto para macro F1, como para micro F1. Com base nos resultados dos testes estatísticos é possível afirmar que, no geral, (i) o classificador SVM superou a AD, (ii) o formato TFIDF é melhor do que o TF, (iii) a distância cosseno supera a distância euclidiana.

O artigo de Pinheiro et al. utiliza o classificador SVM, o formato de entrada TF e a distância euclidiana como métrica de dissimilaridade. Sendo assim, as conclusões deste trabalho, levam a crer que o classificador SVM deve continuar sendo o modelo utilizado, porém o formato TF poderia ser substituído pelo formato TFIDF e a distância euclidiana poderia ser substituída pela distância cosseno para obter melhores macro F1 e micro F1.

Além disso, a configuração **SVM-TFIDF-DC** mostrou-se significativamente melhor do que pelo menos 6 outras: AD-TF-DE, AD-TF-DC, AD-TFIDF-DE, AD-TFIDF-DC, SVM-TF-DE e SVM-TFIDF-DE, comparando-se apenas à configuração SVM-TF-DC.

5.1 Trabalhos Futuros

Quando aplicado ao CoDiS, o modelo classificador SVM foi melhor do que a Árvore de Decisão. A AD é um modelo classificador mais fraco que o SVM. Para trabalhos futuros, seria interessante analisar modelos classificadores fortes como o *Extreme Gradient Boosting (XGBoost)*, o *Multi-Layer Perceptron (MLP)* e *Random Forest*, e compará-los ao SVM.

6 Referências

Altinel, B., Diri, B. and Ganiz, M. (2015). A novel semantic smoothing kernel for text classification with class-based weighting. Knowledge-Based Systems, 89, pp.265-277.

Gangeh, M., Kamel, M. and Duin, R. (2010). Random Subspace Method in Text Categorization. International Conference on Pattern Recognition, pp.2049-2052.

Jones, E., Oliphant, E. and Peterson, P. (2001). SciPy: Open Source Scientific Tools for Python. [online] Disponível em: http://www.scipy.org/ [Acessado em 28 Out. 2017].

Jun, S., Park, S. and Jang, D. (2014). Document clustering method using dimension reduction and support vector clustering to overcome sparseness. Expert Systems with Applications, 41(7), pp.3204-3212.

Kuncheva, L.I. (2004). Combining pattern classifiers: methods and algorithms. John Wiley & Sons.

Lewis, D. (1992). Feature Selection and Feature Extraction for Text Categorization. Workshop on Speech and Natural Language, pp.212-217.

Onan, A., Korukoğlu, S. and Bulut, H. (2016). Ensemble of keyword extraction methods and classifiers in text classification. Expert Systems with Applications, 57, pp.232-247.

Özgür, L. and Güngör, T. (2010). Text classification with the support of pruned dependency patterns. Pattern Recognition Letters, 31(12), pp.1598-1607.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, pp.2825-2830.

Pekalska, E. and Duin, R. (2002). Dissimilarity representations allow for building good classifiers. Pattern Recognition Letters, 23(8), pp.943-956.

Pinheiro, R., Cavalcanti, G. and Ren, T. (2015). Text Categorization Based on Dissimilarity Representation and Prototype Selection. 2015 Brazilian Conference on Intelligent Systems (BRACIS), pp. 163-168.

Pinheiro, R., Cavalcanti, G. and Tsang, I. (2017). Combining dissimilarity spaces for text categorization. Information Sciences, 406-407, pp.87-101.

Prabowo, R. and Thelwall, M. (2009). Sentiment analysis: A combined approach. Journal of Informetrics, 3(2), pp.143-157.

Sebastiani, F. (2002). Machine learning in automated text categorization. ACM Computing Surveys, 34(1), pp.1-47.

Sparck Jones, K. (1972). A Statistical Interpretation Of Term Specificity and its Application in Retrieval. Journal of Documentation, 28(1), pp.11-21.

Su, J., Sayyad-Shirabad, J. and Matwin, S. (2011). Large scale text classification using semi-supervised multinomial naive bayes. International Conference on Machine Learning, pp.97-104.

Wang, G., Sun, J., Ma, J., Xu, K. and Gu, J. (2014). Sentiment classification: The contribution of ensemble learning. Decision Support Systems, 57, pp.77-93.

Wu, Q., Ye, Y., Zhang, H., Ng, M. and Ho, S. (2014). ForesTexter: An efficient random forest algorithm for imbalanced text categorization. Knowledge-Based Systems, 67, pp.105-116.