An Extensive Empirical Evaluation of Preprocessing Techniques and Supervised One Class Learning Algorithms for Text Classification

Marcos P. S. Gôlo, Ricardo M. Marcacini and Rafael G. Rossi

[1]Federal University of Mato Grosso do Sul (UFMS)
Campus Três Lagoas
Av. Ranulpho Marques Leal, 3484, CEP 79613-000, Fone: 67 35093750

marcosgolo@outlook.com, {ricardo.marcacini,rafael.g.rossi@ufms}.br

Abstract. *The texts automatic classification (TAC) has become interesting for academic and business purposes due to the massive volume of texts being produced. TAC is usually performed through multi-class learning, in which a user must provide labeled texts for all classes of an application domain. However, in scenarios in which the intent is to verify if a text belongs to a class of interest, the one-class learning (OCL) is adequate. OCL requires only labeled texts from the class of interest to generate a classification model. Despite the applicability, studies about this theme disregard the use of different algorithms, text pre-processing techniques, and text collections from different domains and characteristics. Therefore, there is no guide about what algorithms and text preprocessing techniques to use in practical applications. Thus, this article aims to address this gap. The results demonstrated that the k-Means-based OCL algorithm obtained the best classification performances for most experiments. Also, the use of dimensionality reduction techniques, which is usually carried out in the literature, did not demonstrate to increase classification performance.*

Resumo. *Dado o volume massivo de textos sendo produzido nos dias atuais, a classificação automática de textos tem se tornado interessante tanto para fins acadêmicos quanto empresariais. Tradicionalmente, a classificação automática de textos é realizada por meio de aprendizado de máquina multi-classe, o qual requer que o usuário apresente textos rotulados de todas as classes de um domínio. Entretanto, em cenários onde deseja-se classificar apenas se um documento pertence ou não a uma classe de interesse, é mais adequada a utilização do aprendizado baseado em uma única classe (AMUC), o qual requer apenas textos rotulados da classe de interesse para se gerar um modelo de classificação. Apesar da aplicabilidade do AMUC, não há trabalhos na literatura que considerem avaliações experimentais envolvendo algoritmos de diferentes categorias, diferentes técnicas de pré-processamento de textos, e diferentes coleções de textos, de forma a indicar quais são as técnicas e algoritmos a serem utilizados em determinadas situações. Com isso, esse artigo visa sanar essa lacuna. Nos resultados é demonstrado que o algoritmo de AMUC baseado no k-Means obteve as melhores performances de classificação para a maioria dos experimentos realizados. Além disso, o uso de técnicas de redução de dimensionalidade, as quais são comumente empregadas na literatura, não proveram aumento na performance de classificação.*

1. Introdução

Atualmente há uma quantidade massiva dados textuais sendo produzida, como artigos, relatórios, publicações em redes sociais, notícias, e *e-mails*, uma vez que textos são a forma mais tradicional e simples de se transmitir informações [Aggarwal 2018]. Os textos contêm informações valiosas tanto na área acadêmica quanto na área empresarial [Biemann and Mehler 2014]. Porém, em muitas situações, é humanamente impossível organizar, gerenciar e extrair conhecimento manualmente dos dados textuais devido ao grande volume de textos a ser analisado. Para automatizar as atividades mencionadas, pode-se fazer uso da classificação automática textos [Aggarwal 2018, Rossi 2016].

A maneira mais viável de se realizar a classificação automática de textos é por meio da utilização de técnicas de aprendizado de máquina [Aggarwal 2018]. O objetivo dos algoritmos de aprendizado de máquina é aprender, generalizar, ou extrair padrões dos textos com base no conteúdo dos mesmos e seus respectivos rótulos (identificadores de classe). A partir do aprendizado de máquina, é possível construir um modelo de classificação o qual será capaz atribuir rótulos automaticamente a textos não rotulados.

Normalmente utiliza-se o aprendizado de máquina multi-classe (AMMC) [Tan et al. 2013]. Neste tipo de aprendizado, o usuário deve conhecer e rotular exemplos para todas as classes de uma aplicação. Porém, a rotulação de exemplos de várias classes pode inibir a utilização de uma aplicação envolvendo aprendizado de máquina, principalmente se o usuário tiver apenas interesse em uma classe específica, como ocorrem em sistemas de recomendação [Pan et al. 2008] ou sensoriamento *web* [Marcacini et al. 2017].

Uma maneira mais prática de gerar um modelo de classificação neste tipo de cenário é por meio do aprendizado baseado em uma única classe (AMUC) [Kemmler et al. 2013, Khan and Madden 2009, Shin et al. 2005, Tax 2001]. Nesse tipo de aprendizado, apenas exemplos da classe de interesse são utilizados no treinamento do algoritmo e consequente geração do modelo de classificação, diminuindo o esforço de rotulação do usuário e tornando o uso da classificação automática de textos mais atrativa.

Apesar da aplicabilidade, não há trabalhos na literatura que considerem avaliações experimentais envolvendo algoritmos de diferentes categorias do AMUC. Além disso, como textos são inerentemente não estruturados e precisam de uma etapa de pré-processamento para gerar uma representação estruturada, não há também a demonstração do impacto de diferentes técnicas de pré-processamento no AMUC. Por fim, a maioria dos trabalhos utilizam poucas coleções de textos, muitas vezes de um mesmo domínio, ou ainda um subconjunto de uma única coleção. Portanto, não há indicativos na literatura sobre quais técnicas de pré-processamento de textos e algoritmos são mais efetivos. Com isso, esse artigo visa sanar essa lacuna por meio de uma avaliação experimental extensa e padronizada considerando: (i) 22 coleções de diferentes domínios; (ii) 6 algoritmos de AMUC de três diferentes paradigmas: baseados em similaridade, probabilísticos e estatísticos; e (iii) 6 diferentes estratégias de representações de textos, sendo 3 esquemas de pesos dos termos e 3 técnicas de redução de dimensionalidade.

Nos resultados apresentados neste artigo demonstra-se que o algoritmo de AMUC *k-Means* obteve as melhores performances de classificação na maioria dos experimentos realizados. Além disso, algoritmos que realizam o aprendizado considerando medidas de proximidade obtiveram melhores performances em comparação algoritmos proba-

bilísticos ou estatísticos. Além disso, o uso de técnicas de redução de dimensionalidade, as quais são comumente empregadas na literatura, apenas proveram aumento na performance de classificação para um algoritmo específico, porém, mesmo assim não sendo capaz de superar performances de classificação obtidas sem o uso de tais técnicas.

O restante deste artigo está dividido da seguinte forma. Na Seção 2 são detalhados os trabalhos relacionados, algoritmos de AMUC e técnicas de pré-processamento de textos utilizados tanto na literatura quanto neste trabalho. Na Seção 3 são apresentadas as especificações da proposta deste artigo. Já na Seção 4 são apresentados e discutidos os resultados. Por fim, na Seção 5 são apresentadas as conclusões e os trabalhos futuros.

2. Conceitos e Trabalhos Relacionados

Nas próximas seções são apresentados os detalhes das técnicas de pré-processamento e algoritmos de AMUC da literatura e utilizados neste trabalho. Também serão apresentadas as características dos trabalhos relacionados, nos quais serão destacadas as técnicas de pré-processamento, algoritmos, coleções e esquemas de avaliação.

2.1. Representação estruturada das coleções de textos

É necessário gerar uma representação estruturada da coleção de textos para que esta seja interpretável pelos algoritmos de aprendizado de máquina. A maioria dos algoritmos de AMUC considera como entrada uma representação no modelo espaço-vetorial [Aggarwal 2018]. Neste tipo de representação, cada texto é representado por um vetor e cada dimensão representa uma característica do texto. Normalmente as dimensões representam termos simples. Neste caso, a representação é denominada *bag-of-words* [Rossi 2016].

Na Tabela 1 é apresentada uma ilustração de uma representação do modelo espaço-vetorial no formato matricial, matriz atributo-valor, para uma coleção de textos com m documentos e n atributos [Tan et al. 2013]. Aqui denotaremos o conjunto de documentos por $\mathcal{D} = \{d_1, d_2, \ldots, d_m\}$ e o conjunto de atributos por $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$.

Tabela 1. Representação no espaço vetorial para *m* documentos e *n* atributos.

	$\mathbf{t}_1$	$\mathbf{t}_2$	$\mathbf{t}_3$	...	$\mathbf{t}_n$
$\mathbf{d}_1$	w_{d_1,t_1}	w_{d_1,t_2}	w_{d_1,t_3}	...	w_{d_1,t_n}
$\mathbf{d}_2$	w_{d_2,t_1}	w_{d_2,t_2}	w_{d_2,t_3}	...	w_{d_2,t_n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\mathbf{d}_m$	w_{d_m,t_1}	w_{d_m,t_2}	w_{d_m,t_3}	...	w_{d_m,t_n}

As células da matriz, ou dimensões do vetor, quantificam a ocorrência de uma característica no documento. Quando considerados termos como característica, os respectivos valores podem corresponder a: (i) tf, i.e., a frequência de um termo em um documento; (ii) *tf-idf*, a qual corresponde à frequência de um termo no documento ponderada pelo inverso da frequência de documentos; e (iii) *binary*, indicando apenas a ocorrência ou não do termo no documento [Rossi 2016]. Normalmente é realizado um procedimento de "limpeza" dos textos para diminuir o número de termos e aumentar a qualidade das representações no modelo espaço vetorial como: (i) padronização das caixas das palavras

e remoção de acentos; (ii) remoção de sinais de pontuação e caracteres alfanuméricos; (iii) remoção de *stopwords*; e (iv) radicalização das palavras [Aggarwal 2018].

Uma característica das representações que utilizam termos como atributos é que o número de dimensões tende a ser alto, e termos semanticamente relacionados, mas com grafias distintas, são representados com atributos distintos, o que pode diminuir a performance de classificação. Para diminuir a dimensionalidade da representação, e consequentemente o custo computacional, e fazer com que termos semanticamente relacionados sejam representados pelo mesmo atributo, pode-se utilizar técnicas de redução de dimensionalidade [Aggarwal 2018]. Geralmente as novas dimensões correspondem a tópicos (conjunto de termos semanticamente relacionados), e o valor da dimensão corresponde ao grau de pertinência de um documento a um tópico.

Qualquer técnica de redução de dimensionalidade pode ser aplicada a coleções de textos, como em [Kumar and Ravi 2017b] em que é aplicado o *Principal Component Analyzis* (*PCA*). Entretanto, existem técnicas de redução de dimensionalidade mais adequadas para o domínio textual, como *Probabilistic Semantic Analysis (PLSA)* e *Latent Dirichlet Allocation (LDA)* [Aggarwal 2018], as quais consideram modelos de distribuição de probabilidade mais específicos para dados textuais. Além das duas últimas, nesse trabalho também foi considerada uma técnica de redução de dimensionalidade baseada em agrupamento de dados, na qual geram-se grupos (tópicos), e define-se a pertinência de um documento a um tópico de acordo com a similaridade desse documento ao centroide do grupo [Kim et al. 2005]. Neste caso, foi utilizado o algoritmo *Bisecting k-Means* (*BkM*) para realizar o agrupamento dos dados [Tan et al. 2013].

2.2. Aprendizado de Máquina Supervisionado baseado em uma Única Classe

Diferentes algoritmos de AMUC podem gerar diferentes superfícies de decisão para discriminar os documentos da classe de interesse para as demais classes, podendo levar um algoritmo obter melhor performance de classificação em determinados domínios de aplicação em relação a outros [Tax 2001]. Porém, mesmo com características diferentes, a forma de definir se um documento pertence à classe de interesse é comum.

Primeiramente o algoritmo baseado em uma única classe gera um *score* para um novo documento a ser classificado, ou seja, um valor que indica o quão propenso esse documento está de pertencer à classe de interesse. O *score* gerado para um documento $\mathbf{d}_i$ será denotado neste artigo por $f(\mathbf{d}_i)$. A partir daí, se o valor de $f(\mathbf{d}_i)$ estiver acima de um limiar (*threshold*), este será definido como sendo da classe de interesse, ou será definido como não sendo da classe de interesse caso contrário (Equação 1).

$$classe = \begin{cases} f(\mathbf{d}_i) \geq threshold \rightarrow \textit{interesse} \\ f(\mathbf{d}_i) < threshold \rightarrow \textit{não interesse} \end{cases} \quad (1)$$

Nas próximas seções são apresentados os algoritmos utilizados neste trabalho divididos em três categorias: probabilísticos, baseados em similaridade e estatísticos.

2.2.1. Algoritmos Baseados em Similaridade

Algoritmos baseados em similaridade consideram a similaridade de um novo documento com os grupo de pontos (ou representantes dos grupos de pontos) para gerar um *score* para

um novo documento. Nesta categoria enquadram-se os algoritmos baseados em vizinhos mais próximos ou em agrupamento de dados [Tan et al. 2013].

No caso dos vizinhos mais próximos, há duas variantes: O *k-Nearest Neighbor Density-based* (*k-NND*) e o *k-Nearest Neighbor Relative Density-based* (*k-NNRD*). No *k-NND*, $f(\mathbf{d}_i)$ é dado por:

$$f(\mathbf{d}_i) = densidade(\mathbf{d}_i, k) = \sum_{\mathbf{d}_j \in \mathcal{N}_{(\mathbf{d}_i,k)}} cos(\mathbf{d}_i, \mathbf{d}_j)/|\mathcal{N}_{(\mathbf{d}_i,k)}|, \tag{2}$$

na qual $\mathcal{N}_{(\mathbf{d}_i,k)}$ representa o conjunto dos k vizinhos mais próximos de d_i, e $cos(\mathbf{d}_i, \mathbf{d}_j)$ retorna similaridade cosseno entre os vetores dos documentos d_i e d_j. Já o algoritmo *k-NNRD* também leva em consideração a densidade dos vizinhos mais próximos, de forma a verificar se a densidade do exemplo a ser classificado é similar à densidade dos exemplos vizinhos. Com isso, $f(\mathbf{d}_i)$ é dado por:

$$f(\mathbf{d}_i) = densidade\_relativa(\mathbf{d}_i, k) = \frac{densidade(\mathbf{d}_i, k)}{\sum_{\mathbf{d}_j \in N(\mathbf{d}_i,k)} densidade(\mathbf{d}_j, k)/|\mathcal{N}(\mathbf{d}_i, k)|} \tag{3}$$

Já as abordagens baseadas em agrupamento, utilizam o conceito de agrupamentos particionais baseados em centroides, na qual a coleção de textos é subdividida em k grupos, i.e, $\mathcal{D} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \cdots \cup \mathcal{G}_k$, em que cada $\mathcal{G}_i \subset \mathcal{D}$ é representado por um vetor $\mathbf{g}_i$ que corresponde à média dos vetores dos pontos em $\mathcal{G}_i$, i.e., centroide do grupo $\mathcal{G}_i$. O valor de $f(\mathbf{d}_i)$ é dado considerando a similaridade com o centroide do grupo mais próximo, i.e.,

$$f(\mathbf{d}_i) = \max_{\mathcal{G}_j \subset \mathcal{D}} cos(\mathbf{d}_i, \mathbf{g}_j). \tag{4}$$

Portanto, o objetivo deste algoritmo é verificar se um novo ponto pertence a um grupo de pontos da classe de interesse. Neste trabalho foi utilizado o algoritmo *k-Means* para a abordagem baseada em agrupamento [Tan et al. 2013]. Vale ressaltar que apesar da abordagem baseada em agrupamento ser utilizada com frequência na literatura, não foi encontrado o seu uso para a classificação de textos. Vale ressaltar também que quando $k = 1$, essa abordagem é equivalente ao algoritmo Rocchio [Aggarwal 2018] para o AMUC.

2.2.2. Algoritmos Probabilísticos

O objetivo dos algoritmos probabilísticos é gerar um modelo de distribuição de probabilidade dos exemplos da classe de interesse e posteriormente verificar a probabilidade de um novo documento pertencer à classe de interesse. Observou-se na literatura um uso frequente do algoritmo *Multinomial Naïve Bayes* (*MNB*). O MNB faz uso das probabilidades dos termos ocorrerem na classe de interesse. A probabilidade de um termo t_i e o *score* para um novo documento são dados respectivamente pelas Equações 5 e 6.

$$p(t_i) = \frac{1 + \sum_{d_j \in \mathcal{D}} w_{d_j,t_i}}{|\mathcal{T}| + \sum_{d_j \in \mathcal{D}} \sum_{t_k \in \mathcal{T}} w_{d_j,t_k}} \tag{5}$$

$$f(\mathbf{d}_i) = \frac{\prod_{t_j \in \mathcal{T}} p(t_j) * w_{d_i,t_j}}{\prod_{t_j \in \mathcal{T}} w_{d_i,t_j}} \tag{6}$$

Um algoritmo probabilístico bastante utilizado na literatura de AMUC mas não utilizado para a classificação de textos é o *Gaussian Mixture Models (GMM)* [Tax 2001]. Em sua formulação original, há o cálculo da distância de Mahalanobis, a qual faz uso

inverso de uma matriz de covariância (Σ^{-1}). Uma vez que a matriz de covariância tem dimensões $|\mathcal{T}| \times |\mathcal{T}|$, sua inversão é custosa devido ao alto número de dimensões dos dados textuais. Devido a esse fator, nesse artigo foi utilizada uma adaptação da função Gaussiana utilizando a medida cosseno [Rossi 2016]. Com isso, cada componente do *GMM* e o *score* de classificação para um documento d_i são dados respectivamente por:

$$p_N(\mathbf{d}_i, \boldsymbol{\mu}_j, \sigma_j) = \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{1-cos(\mathbf{d}_i-\boldsymbol{\mu}_j)^2}{\sigma_j}\right)}, \quad (7) \qquad f(\mathbf{d}_i) = p(\mathbf{d}_i) = \frac{1}{k}\sum_{j=1}^{k} \alpha_j p_N(\mathbf{d}_i, \boldsymbol{\mu}_j, \sigma_j), \quad (8)$$

na qual k é o número de componentes do modelo, α_j, $\boldsymbol{\mu}_j$ e σ_j são respectivamente o peso, o ponto médio e a variância do j-ésimo modelo. Os coeficientes de cada modelo (α, $\boldsymbol{\mu}$ e α) são inferidos por meio da técnica *Expectation Maximization* [Rossi 2016].

2.2.3. Algoritmos Estatísticos

O algoritmo *One Class Support Vector Machine (OCSVM)* [Shin et al. 2005] é baseado no tradicional algoritmo de aprendizado de máquina *Support Vector Machine* [Tan et al. 2013]. O *SVM* visa gerar um hiperplano de margem de separação máxima entre pares de classes. Sua adaptação para o AMUC consiste em gerar exemplos fictícios próximos à origem do plano que correspondem a contraexemplos da classe de interesse. Com isso, a otimização do hiperplano do *OCSVM* e a classificação de um novo documento são dados respectivamente por:

$$\min \frac{1}{2} \parallel \mathbf{h} \parallel^2 + \frac{1}{\nu \cdot |\mathcal{D}|} \sum_{d_j \in \mathcal{D}} \varepsilon_{d_j} - \rho, \qquad (9) \qquad\qquad f(\mathbf{d}_i) = sgn(\mathbf{h} \cdot \Phi(\mathbf{d}_i) - \rho) \qquad (10)$$

na qual $\mathbf{h}$ corresponde aos coeficientes do hiperplano de separação, ε_{d_j} é o erro de classificação de um documento de treinamento d_j, que corresponde a sua distância até o hiperplano de separação, ρ é o limiar de erro de classificação, ν é proporção de exemplos não pertencentes à classe de interesse, $\Phi(\mathbf{d}_i)$ é uma função a qual mapeia o espaço original para outro espaço de forma a tornar os exemplos de classes diferentes linearmente separáveis, e a função $sgn()$ retorna 1 se $\mathbf{h} \cdot \Phi(\mathbf{d}_i) - \rho \geq 0$ e 0 caso contrário.

2.3. Trabalhos Relacionados

Em [Manevitz and Yousef 2001], os autores utilizaram somente uma base a qual consistia das 10 classes mais frequentes da coleção Reuters-21578. Foram utilizados os algoritmos *OCSVM*, e as versões baseadas em uma única classe do *MNB*, *k-NN*, Rocchio e uma Rede Neural do tipo *autoencoder*. Como técnicas de pré-processamento, foram utilizadas representações *bag-of-words* com diferentes esquemas de peso dos termos: *tf*, *tf-idf*, e *binary*. Porém, vale ressaltar que nem todos os tipos de esquemas de pesos foram aplicados a todos os algoritmos. Além disso, o número de dimensões das representações foi reduzido a 10 e 20 considerando os termos com maior frequência de documento, i.e., número de ocorrências nos documentos da coleção. As avaliações foram feitas considerando um conjunto fixo de exemplos de treino e teste para cada classe, e a medida F_1 foi utilizada para medir a performance de classificação.

Em [Kumar and Ravi 2017a], são utilizadas duas bases referentes à *phishing* em e-mail e páginas *web*, e uma base referente à *feedbacks* de consumidores (*Imperial Bank*). Todas as coleções são binárias, i.e., classe positiva e negativa. Foram consideradas para treinamento somente textos das classes negativas. Foi utilizado o algoritmo *OCSVM* para extrair os documento dos vetores de suporte, i.e., documentos mais relevantes. Em seguida, foi aplicada a técnica *Latent Semantic Indexing* (*LSI*) nesses documentos para a geração de um novo espaço semântico. A partir daí, um documento de teste é projetado no novo espaço e é calculada a sua similaridade com os demais documentos relevantes. Se a similaridade média estiver abaixo de 0.5, o texto é considerado como positivo.

No artigo de [Kumar and Ravi 2017b], foram utilizadas 4 coleções textuais: *20 Newsgroup*, *SyskillWebert*, *Imperial Bank*, e uma coleção sobre chamadas de sistemas referentes à *Malwares*. Todas as coleções são multi-classe, porém, foram transformadas em coleções com duas classes, seja considerando apenas as duas classes mais frequentes, ou por meio da união de textos de classes diferentes em uma única classe. Como medida de avaliação foi utilizada a Precisão. As classes consideradas negativas foram utilizadas para treino e a positivas para teste. Foi utilizada a técnica *PCA* para redução de dimensionalidade, porém, o número de dimensões foi variado para cada coleção. Foi utilizado apenas o *OCSVM* como algoritmo de AMUC.

Em [Manevitz and Yousef 2007], foram considerados os algoritmos *MNB*, *k-NN*, Rocchio, *OCSVM* e Redes Neurais (RNs) do tipo *autoencoder*. Apenas as 10 categorias mais frequentes da Reuters-21578 foram consideradas na avaliação. Foram utilizados 25% dos exemplos de cada categoria iterativamente como exemplos positivos e o restante para teste. A medida F_1 foi utilizada para medir a performance de classificação. Além disso, foi utilizado *tf-idf* como esquema de peso dos termos e foram mantidas na representação *bag-of-words* apenas os 20 termos com maior frequência de documento.

Por fim, em [Pan et al. 2008] foram utilizadas duas coleções: *Yahoo News Dataset* e *Delicious*. O esquema de validação utilizado consistiu em 20 repetições separando em cada repetição 80% dos exemplos positivos para treino e 20% para teste. As medidas utilizadas foram *Average Precision* e *Half-Life Utility*, ambas da área de Recuperação de Informação. Foram utilizados algoritmos tradicionais de filtragem colaborativa, como o *Singular Value Decomposition* (fatoração de matrizes) e a abordagem baseada em vizinhos (equivalente ao k-*NN*), além do o *OCSVM*.

Conforme apresentado nesta seção, a maioria dos trabalhos utilizam poucas coleções de textos e algoritmos de AMUC. Quando são utilizadas mais de uma forma de representação da coleção de textos, estas não são consideradas em todos os algoritmos. Além disso, há falta de padronização nas avaliações, uma vez que são considerados diferentes esquemas e medidas de avaliação. Portanto, não há indicativos na literatura sobre a efetividade de técnicas de pré-processamento de textos e algoritmos de AMUC para a classificação de textos. Com isso, na próxima seção é apresentada a proposta desse artigo visando sanar essa lacuna.

3. Proposta

Nas próximas subseções são apresentados os detalhes da proposta de artigo referentes às coleções de textos e técnicas de pré-processamento utilizadas, algoritmos e seus respectivos parâmetros, e o procedimento adotado para avaliar a performance de classificação.

3.1. Coleções de Textos e Técnicas de Pré-Processamentos Utilizadas

Foram utilizadas 22 coleções de textos neste artigo [Rossi et al. 2013][1]. As coleções de textos para cada domínio e o respectivo número de classes por coleção, apresentado entre parênteses, são[2]:

- **Análise de sentimentos**: MultiDomainSentiment (2) e ReviewPolarity (2).
- **Documentos científicos**: Classic4 (4) e CSTR (4).
- **Documentos médicos**: Oh0 (10), Oh5 (10), Oh10 (10) e Oh15 (10).
- **Notícias**: Fbis (17), Re0 (13), Re1 (25), Re8 (8), e Reviews (5).
- **Páginas *web***: SyskillWebert (4) e WAP (20).
- **Recuperação de informação**: Tr11 (9), Tr12 (8), Tr21 (6), Tr23 (6), Tr31 (7), Tr41 (10) e Tr45 (10).

Também foram geradas representações das coleções utilizando as técnicas de redução de dimensionalidade *PLSA*, *LDA* e *BkM* com diferentes números de tópicos. O número de tópicos para todas as técnicas foi 5, 10, 15, 20, 25, 100, 150, 200 e equivalente ao número de classes da coleção. No caso do LDA foi utilizada a biblioteca Mallet[3] juntamente com a funcionalidade de otimização dos parâmetros α e β.

3.2. Algoritmos e Parâmetros

Os algoritmos de aprendizado e seus respectivos parâmetros utilizados são:

- ***k-Means*, k-NND, k-NNRD e GMM**: $k \in$ [1,30].
- ***One-Class Support Vector Machines (OCSVM):*** *kernels* Linear, Polinomial, e RBF, $\gamma = 1 * 10^p, p \in [-7..1]$, e $\upsilon = 0.05 * q, q \in [1..20]$.
- ***MNB***: não possui parâmetros.

3.3. Avaliação da Performance de Classificação

Para fazer uso de coleções multi-classe na avaliação baseada em uma única classe, foi utilizado uma adaptação do procedimento de validação cruzada em pastas (*x-Fold Cross-Validation*) [Tan et al. 2013]. Nessa estratégia, iterativamente cada classe da coleção é considerada como classe de interesse. Para cada classe de interesse foi aplicado o *10-Fold Cross-Validation*, na qual em cada execução deste procedimento, 9 pastas da classe de interesse são utilizadas para treino, e a pasta restante bem como os documentos das demais classes (não interesse) são utilizados para teste. Portanto, são realizadas $|\mathcal{C}| * 10$ execuções para cada coleção de forma a se obter a performance de classificação.

Como medida de performance foi utilizada a F_1, a qual corresponde à uma média harmônica das medidas Precisão e Revocação. $F1$, Precisão e Revocação são dadas respectivamente por:

$$F_1 = \frac{2 \cdot Prec \cdot Rev}{Prec + Rev}, \quad (11) \qquad Prec = \frac{VP}{VP + FP}, \quad (12) \qquad Rev = \frac{VP}{VP + FN}, \quad (13)$$

na qual VP (Verdadeiros Positivos) refere-se ao número de documentos da classe de

[1]Vale ressaltar que devido à baixa disponibilidade de coleções de textos específicas para o aprendizado baseado em uma única classe, optou-se por utilizar coleções de texto multi-classe. Porém, para simular um cenário de aprendizado baseado em uma única classe, são feitas iterações nas quais em cada iteração uma determinada classe da coleção é considerada como classe de interesse.

[2]As características das coleções utilizadas neste artigo são apresentadas em [Rossi et al. 2013].

[3]Biblioteta Mallet: `http://mallet.cs.umass.edu/`

interesse que o algoritmo classificou corretamente, FP (Falsos Positivos) refere-se ao número de documentos que não são da classe de interesse mas foram classificados como sendo da classe de interesse, e FN (Falsos Negativos) referem-se ao número documentos da classe de interesse que o algoritmo classificou como não sendo da classe de interesse.

Quanto aos limiares para definir de um novo texto pertence à classe de interesse, foram utilizados:

- **Manualmente definidos**: sendo $threshold = 0.05 \times z$, na qual $z \in \mathbb{N} : 1 \leq z \leq 19$ para os algoritmos *k-NND*, *k-NNRD*, *k-Means*, e *GMM*. Para o MNB foi utilizado $threshold = r * 10^{-s}$, tal que $r \in [1..9]$ e $s \in [1..50]$[4]. Vale ressaltar os valores de $f(\mathbf{d}_i)$ estão no intervalo $[0, 1]$ para todos os algoritmos.
- **Automaticamente definidos**: utilizando a estratégia 6σ para definição dos limiares [Muir 2005]. Neste caso, geram-se os $scores$ $f(\mathbf{d}_i)$ para todo documento $\mathbf{d}_i$ utilizado no treinamento da classe de interesse, calcula-se a média (μ) e o desvio padrão (σ) de todos os *scores*, e define-se os limiares como $threshold \in \{\mu - 3\sigma, \mu - 2\sigma, \mu - 1\sigma, \mu, \mu + 1\sigma, \mu + 2\sigma, \mu + 3\sigma\}$.

Os resultados que serão reportados na próxima seção correspondem à uma média de todas as execuções do procedimento de avaliação para uma coleção de textos. Além disso, será apenas reportado o melhor resultado de um algoritmo considerando todos os parâmetros e limiares utilizados, i.e., a análise realizada aqui é a análise do melhor caso. Porém, todos os resultados obtidos por todos os algoritmos, parâmetros e representações, resultados de outras medidas de performance de classificação, resultados de testes de significância estatística, bem como as implementações utilizadas para gerar os resultados, estão disponíveis em: `http://gepic.ufms.br/one-class/eniac_2019/`.

4. Resultados

Na Tabela 2 são apresentados os maiores valores da medida F_1 e média das F_1s obtidas para cada algoritmo e para cada técnica de pré-processamento. Também são apresentad o *ranking* médio dos algoritmos[5].

No geral, o melhor desempenho médio foi obtido pelo algoritmo *k-Means* com esquema de pesos *tf-idf*. Além disso, observa-se que o algoritmo *k-Means* obteve o melhor resultado para a maioria das coleções com as diferentes técnicas de pré-processamento utilizadas, bem como a melhor performance média e melhor *ranking* médio no geral. A exceção se dá para o algoritmo *k-NND*, que obteve o melhor desempenho médio e melhor *ranking* médio em conjunto com a técnica *LDA*, e obteve o melhor *ranking* médio em conjunto com a técnica *BkM*.

Em relação aos esquemas de pesos na *bag-of-words*, pode-se observar que o melhor esquema de pesos dos termos foi o *tf-idf*, enquanto que o esquema de pesos binário obteve a pior performance média. Em relação ao uso das técnicas de redução de dimensionalidade, nota-se impacto positivo para os algoritmos *OCSVM* e *MNB*. Entretanto, para o último, mesmo com o aumento em sua performance, este apresentou um desempenho

[4]Vale O *OCSVM* não faz uso de limiares devido a função *sgn* retornar o valor 1 se o exemplo pertence à classe de interesse e 0 caso contrário.

[5]Para cada algoritmo é atribuído o seu *ranking*, sendo que o algoritmo que obteve a melhor performance irá receber o valor 1, a segunda melhor performance o valor 2 e assim por diante. Por fim, é calculada a média dos *rankings* de cada algoritmo.

Tabela 2. Resultados da medida F_1 para cada coleção, média das F_1s por algoritmo, e *ranking* médio considerando os diferentes algoritmos de AUMC para as coleções de textos com diferentes técnicas de pré-processamento.

TF-IDF

Bases	**KNND**	**KNNRD**	**KME**	**OCSVM**	**MNB**	**GMM**
Classic4	0,6940	0,6712	**0,7057**	0,5667	0,0983	0,5667
CSTR	**0,6471**	0,6312	0,5830	0,4532	0,2696	0,5187
Fbis	0,3943	0,3856	**0,4256**	0,2180	0,0199	0,4046
MultiDomSent	0,1860	0,1874	**0,2041**	0,1787	0,1675	0,2012
Oh0	0,6155	0,6087	**0,6458**	0,4113	0,0369	0,5917
Oh10	0,4202	0,4395	0,4416	0,3582	0,0500	**0,4641**
Oh15	0,5225	**0,5364**	0,5256	0,3134	0,0326	0,4954
Oh5	0,5693	0,5488	**0,5835**	0,4142	0,0148	0,5409
Review_Polarity	**0,2708**	0,2641	0,2499	0,1923	0,1352	0,2144
Re0	**0,4761**	0,4657	0,4710	0,1605	0,0205	0,4127
Re1	0,4658	0,4708	**0,5231**	0,2585	0,0106	0,5182
Re8	0,5687	0,5649	**0,5958**	0,3349	0,0285	0,5822
Reviews	0,5362	0,5386	**0,5390**	0,3465	0,0521	0,5279
SyskillWebert	0,6908	**0,7221**	0,6386	0,3303	0,1111	0,6029
Tr11	0,2362	0,2528	0,3472	0,1724	0,0105	**0,5030**
Tr12	0,5976	0,5807	**0,6407**	0,2913	0,0576	0,5714
Tr21	0,4227	0,4171	**0,4999**	0,1954	0,1417	0,4617
Tr23	0,4871	**0,4883**	0,4381	0,2435	0,0586	0,4729
Tr31	0,5068	0,4504	**0,5778**	0,1822	0,0752	0,4765
Tr41	0,7079	0,7011	**0,7462**	0,2404	0,0049	0,6664
Tr45	0,5828	0,5698	**0,6218**	0,1622	0,0190	0,6055
Wap	0,3323	0,3257	**0,3882**	0,2690	0,0029	0,3683
Média das F_1s	0,4969	0,4919	**0,5178**	0,2860	0,0645	0,4894
***Ranking* Médio**	2,5455	2,8636	**1,5909**	4,9773	6,0000	3,0227

TF

KNND	**KNNRD**	**KME**	**OCSVM**	**MNB**	**GMM**
0,6002	0,5800	0,6121	0,4648	0,0789	**0,6150**
0,5978	0,5491	0,5216	0,4018	0,0891	0,5200
0,3672	0,3587	**0,3790**	0,2622	0,0137	0,3026
0,1801	0,1797	0,1814	0,1750	0,1833	**0,2744**
0,3073	0,3154	**0,4080**	0,1859	0,0265	0,3731
0,3539	0,3428	**0,3835**	0,2180	0,0281	0,3477
0,2915	0,2967	0,3203	0,1318	0,0229	**0,3624**
0,3802	0,3767	**0,4242**	0,1927	0,0234	0,3814
0,2212	**0,2212**	0,1698	0,1715	0,1667	0,1822
0,3102	0,2984	0,2870	0,1959	0,0201	**0,3191**
0,3960	**0,4002**	0,3499	0,1714	0,0090	0,3810
0,5022	0,5059	**0,5639**	0,2328	0,0383	0,5561
0,3090	0,3607	**0,4761**	0,2845	0,0517	0,4441
0,7483	0,7219	**0,7662**	0,4417	0,0710	0,7625
0,4216	0,4200	**0,4459**	0,3284	0,0268	0,4052
0,4331	0,3986	**0,4718**	0,3835	0,0307	0,4525
0,3103	0,3103	**0,4254**	0,2183	0,0771	0,3547
0,3567	0,3555	**0,3791**	0,2657	0,0462	0,3324
0,4021	0,4000	**0,4498**	0,3130	0,0412	0,4282
0,5828	0,5659	0,5176	0,2880	0,0237	0,5789
0,4711	0,4312	**0,4972**	0,2651	0,0241	0,4298
0,2892	**0,2901**	0,2788	0,1722	0,0111	0,2814
0,4015	0,3945	**0,4231**	0,2620	0,0502	0,4129
2,6364	3,0909	**2,0000**	5,0000	5,8182	2,4545

Binary

Bases	**KNND**	**KNNRD**	**KME**	**OCSVM**	**MNB**	**GMM**
Classic4	0,6059	0,5495	**0,6168**	0,3931	0,0739	0,3931
CSTR	**0,6250**	0,5725	0,5534	0,3335	0,1622	0,3335
Fbis	0,1891	0,2002	**0,2965**	0,1284	0,0133	0,1284
MultiDomSent	0,2236	0,2203	0,2049	0,1821	0,1931	**0,3669**
Oh0	**0,4886**	0,4866	0,4618	0,2902	0,0269	0,4678
Oh10	0,3911	**0,4032**	0,3956	0,2671	0,0267	0,3984
Oh15	0,3962	**0,4182**	0,3699	0,1875	0,0228	0,3633
Oh5	**0,4564**	**0,4564**	0,4400	0,2370	0,0233	0,4374
Review_Polarity	0,2445	**0,2452**	0,1808	0,1701	0,1604	0,1783
Re0	0,3688	0,3460	**0,3693**	0,1585	0,0201	0,2766
Re1	**0,3897**	0,3758	0,3745	0,1202	0,0090	0,3178
Re8	**0,5604**	0,5232	0,5484	0,2783	0,0383	0,4909
Reviews	0,4940	0,4943	**0,4966**	0,2444	0,0517	0,3745
SyskillWebert	0,6061	0,5965	**0,6523**	0,3528	0,0663	0,5281
Tr11	0,2362	0,2528	0,3472	0,1724	0,0106	**0,3580**
Tr12	0,3890	0,3734	**0,4621**	0,1325	0,0295	0,4278
Tr21	0,2046	0,2069	**0,2895**	0,1920	0,0695	0,2868
Tr23	0,3537	0,2647	0,2862	0,1900	0,0554	**0,3678**
Tr31	0,3633	0,3549	**0,5257**	0,1396	0,0369	0,4728
Tr41	0,5122	0,5190	**0,5810**	0,1586	0,0237	0,5603
Tr45	0,3601	0,3552	**0,4711**	0,1589	0,0235	0,4524
Wap	**0,2476**	0,2466	0,2324	0,1605	0,0110	0,2075
Média das F_1s	0,3957	0,3846	**0,4162**	0,2113	0,0522	0,3722
***Ranking* Médio**	2,3409	2,5682	**2,0909**	4.9773	5.9545	3.0682

LDA

KNND	**KNNRD**	**KME**	**OCSVM**	**MNB**	**GMM**
0,5840	0,4971	**0,6542**	0,6512	0,2509	0,5316
0,7333	0,7053	0,6749	0,5944	0,3918	0,5307
0,0281	0,0273	0,0240	0,0253	0,0217	0,0188
0,1668	0,1671	0,1669	0,1669	0,2254	**0,2622**
0,6291	0,6037	0,6079	0,5752	0,1684	0,5856
0,4914	0,4977	0,4943	**0,5026**	0,1829	0,4452
0,5178	**0,5364**	0,4988	0,4823	0,1539	0,4895
0,6109	0,6022	0,5965	0,5713	0,1717	0,5869
0,1678	0,1682	0,1681	0,1698	**0,2256**	0,2196
0,4414	0,3958	**0,4831**	0,3982	0,0344	0,3040
0,4439	0,4409	0,4417	0,4003	0,0638	0,2965
0,6169	0,5909	0,6036	0,5950	0,1367	0,5353
0,0579	0,0525	0,0581	0,0710	**0,0975**	0,0854
0,5710	0,5518	**0,5759**	0,5697	0,2673	0,4940
0,6069	**0,6177**	0,5995	0,5757	0,1918	0,4249
0,6824	0,6407	0,6623	0,6263	0,2166	0,5531
0,5976	**0,6132**	0,5900	0,4431	0,1606	0,4970
0,5568	0,5380	**0,5939**	0,3816	0,1618	0,3737
0,6877	0,6777	0,6308	**0,6923**	0,2207	0,5703
0,6665	0,6591	**0,6740**	0,6545	0,1205	0,4506
0,6865	0,6825	0,6755	0,6831	0,1779	0,4887
0,3786	0,3514	0,3302	0,3579	0,0114	0,1145
0,4965	0,4826	0,4911	0,4631	0,1661	0,4026
2,2273	2,9545	2,7500	3,3864	5,3182	4,3636

PLSA

Bases	**KNND**	**KNNRD**	**KME**	**OCSVM**	**MNB**	**GMM**
Classic4	0,7223	0,6172	**0,8155**	0,7222	0,2105	0,6950
CSTR	0,4514	0,4413	**0,4816**	0,3458	0,3285	0,3972
Fbis	**0,5036**	0,5012	0,4673	0,3424	0,0206	0,4642
MultiDomSent	0,1800	0,1785	0,2133	0,1861	0,2429	**0,3423**
Oh0	0,4578	0,4543	**0,4827**	0,3768	0,1369	0,4490
Oh10	**0,4738**	0,4601	0,4632	0,3571	0,1786	0,3871
Oh15	**0,3999**	0,3687	0,3788	0,3432	0,0589	0,3339
Oh5	**0,4606**	0,4381	0,4570	0,3638	0,1117	0,3938
Review_Polarity	0,2232	0,2175	0,2313	0,1983	0,2474	**0,2925**
Re0	0,4394	0,4252	**0,4144**	0,2597	0,0586	0,3294
Re1	**0,3268**	0,3186	0,3050	0,2012	0,0132	0,2902
Re8	0,5136	0,4571	0,5276	0,3782	0,0865	**0,5534**
Reviews	0,5080	0,4486	**0,6056**	0,4896	0,0978	0,5444
SyskillWebert	0,5083	**0,5104**	0,4982	0,4822	0,1965	0,4063
Tr11	0,3659	0,4470	**0,4883**	0,4206	0,0945	0,3766
Tr12	**0,5685**	0,4890	0,4612	0,3394	0,1093	0,5238
Tr21	0,4347	0,4034	**0,4937**	0,2567	0,1640	0,4719
Tr23	**0,4972**	0,4529	0,4881	0,2792	0,2711	0,3872
Tr31	**0,6138**	0,6090	0,5810	0,3907	0,2253	0,5858
Tr41	**0,6159**	0,6157	0,5850	0,4208	0,1737	0,5461
Tr45	0,5303	0,5363	**0,6092**	0,4865	0,1910	0,5265
Wap	0,2526	0,2391	**0,2959**	0,2954	0,0118	0,2709
Média das F_1s	0,4567	0,4377	**0,4702**	0,3607	0,1468	0,4349
***Ranking* Médio**	2,1818	3,1818	**2,1364**	4,5455	5,6364	3,3182

BKM

KNND	**KNNRD**	**KME**	**OCSVM**	**MNB**	**GMM**
0,5840	0,5057	**0,8155**	0,6512	0,3339	0,0686
0,6353	0,5320	0,5363	0,4641	0,3852	0,0700
0,0281	0,0139	**0,4688**	0,0253	0,0734	0,0269
0,2240	0,1808	0,1969	0,1787	0,1820	0,1732
0,3589	0,3106	0,3713	**0,4212**	0,0750	0,0223
0,4366	0,3882	**0,4504**	0,4295	0,1874	0,0220
0,4335	0,2954	0,2954	0,3654	0,1209	0,0235
0,6109	0,5699	0,4570	0,5713	0,1207	0,0219
0,1755	0,1691	0,1758	0,1765	**0,2014**	0,1742
0,2401	0,2265	0,2461	**0,3385**	0,1467	0,0199
0,4439	0,4409	0,3050	0,4003	0,0820	0,0089
0,2240	0,3851	0,4358	**0,6107**	0,1523	0,0381
0,6408	0,5366	0,5915	0,5011	0,0532	0,0518
0,9062	0,8858	0,9046	0,8428	0,3690	0,0661
0,6069	**0,6177**	0,4883	0,5757	0,1502	0,0456
0,5520	0,5025	0,5508	0,5040	0,2133	0,0316
0,3792	0,3666	**0,5900**	0,3228	0,2241	0,0702
0,3192	0,3040	0,3061	0,3114	0,0473	0,0716
0,5595	0,5274	0,5343	0,5374	0,2298	0,0525
0,6665	0,6591	0,6292	0,6545	0,1023	0,1067
0,6865	0,6825	0,6092	0,6831	0,1728	0,0253
0,3786	0,3514	0,2959	0,3579	0,0344	0,0150
0,4586	0,4296	**0,4661**	0,4511	0,1662	0,0548
1.7727	3.4773	2.5682	2.7273	4.6818	5.7727

muito inferior aos demais algoritmos. Já para o *OCSVM*, este teve seu desempenho significativamente aumentado ao ser empregado em conjunto com técnicas de redução de dimensionalidade. Isso explica o seu uso em conjunto com tais técnicas na literatura. Entretanto, mesmo assim o *OCSVM* teve performance média inferior aos algoritmo *k-Means*, *k-NND* e *k-NNRD* para todas as técnicas de pré-processamento, exceto para o *BkM*.

Por fim, quanto aos domínios das coleções de textos, o algoritmo *k-Means*, em geral, apresentou melhores desempenhos para as coleções da TREC, notícias e documentos científicos. Para os demais domínios de coleções, não houve um algoritmo que obteve a melhor performance para a grande maioria das coleções.

5. Conclusões e Trabalhos Futuros

Dado o massivo volume de dados textuais produzidos, dificultando o processo de gerenciamento, organização e extração de conhecimento de forma manual, a classificação automática de textos se mostra importante nos dias atuais. Porém, para tornar o uso da classificação automática de textos mais prática em determinadas aplicações, áreas de aprendizado de máquina como o AMUC têm ganhado destaque nos últimos anos.

Este artigo visou sanar lacunas da literatura que até então não apresentou uma análise empírica extensa sobre o impacto de diferentes algoritmos de AMUC e técnicas de pré-processamento de textos na performance de classificação. Os resultados apresentados neste artigo demonstraram que técnicas frequentemente utilizadas na literatura não necessariamente obtém as melhores performances de classificação e nem que o emprego de técnicas de redução de dimensionalidade é essencial para melhorar a performance de classificação. De acordo com os resultados obtidos, o algoritmo *k-Means* em conjunto com uma representação *bag-of-words* e *tf-idf* como esquema de pesos dos termos obteve as melhores performances de classificação para a maioria das avaliações realizadas.

Como trabalhos futuros pretende-se (i) considerar outras técnicas de agrupamento de textos, uma vez que demonstraram ser promissoras; e (ii) considerar os resultados obtidos nesse trabalho na proposta de algoritmos de aprendizado semissupervisionado baseado em uma única classe (*Positive and Unlabeled Learning*) [Zhang and Zuo 2008].

Agradecimentos

Os autores agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), processos #433082/2018-6, #426663/2018-7 e #116072/2018-0, pelo suporte financeiro.

Referências

Aggarwal, C. C. (2018). *Machine Learning for Text*. Springer Publishing Company, Incorporated, 1st edition.

Biemann, C. and Mehler, A. (2014). *Text Mining: From Ontology Learning to Automated Text Processing Applications*. Springer.

Kemmler, M., Rodner, E., Wacker, E.-S., and Denzler, J. (2013). One-class classification with gaussian processes. *Pattern Recognition*, 46(12):3507–3518.

Khan, S. S. and Madden, M. G. (2009). A survey of recent trends in one class classification. In *Irish Conf. Artifici. Intelligence and Cognitive Science*, pages 188–197.

Kim, H., Howland, P., and Park, H. (2005). Dimension reduction in text classification with support vector machines. *Journal of Machine Learning Research*, 6:37–53.

Kumar, B. S. and Ravi, V. (2017a). One-class text document classification with OCSVM and LSI. In *Art. Intel. & Evolutionary Computations in Eng. Systems*, pages 597–606.

Kumar, B. S. and Ravi, V. (2017b). Text document classification with PCA and one-class SVM. In *Proc. Int. Conf. Frontiers in Intel. Computing: Theory and Applications*, pages 107–115.

Manevitz, L. and Yousef, M. (2007). One-class document classification via neural networks. *Neurocomput.*, 70(7-9):1466–1481.

Manevitz, L. M. and Yousef, M. (2001). One-class SVMs for document classification. *Journal of machine Learning research*, 2(Dec):139–154.

Marcacini, R. M., Rossi, R. G., Nogueira, B. M., Martins, L. V., Cherman, E. A., and Rezende, S. O. (2017). Websensors analytics: Learning to sense the real world using web news events. In *Simp. Brasileiro de Sistemas Multimídia e Web*, pages 169–173.

Muir, A. (2005). *Lean Six Sigma Statistics: Calculating Process Efficiencies in Transactional Project*. McGraw Hill professional – Six sigma operational methods series.

Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., and Yang, Q. (2008). One-class collaborative filtering. In *Proc Int. Conf. Data Mining*, pages 502–511.

Rossi, R. G. (2016). *Classificação automática de textos por meio de aprendizado de máquina baseado em redes*. PhD thesis, Universidade de São Paulo.

Rossi, R. G., Marcacini, R. M., and Rezende, S. O. (2013). Benchmarking text collections for classification and clustering tasks. *Institute of Mathematics and Computer Sciences, University of São Paulo*.

Shin, H. J., Eom, D.-H., and Kim, S.-S. (2005). One-class support vector machines—an application in machine fault detection and classification. *Computers & Industrial Engineering*, 48(2):395–408.

Tan, P., Steinbach, M., and Kumar, V. (2013). *Introduction to Data Mining: Pearson New International Edition*. Pearson Education Limited.

Tax, D. M. J. (2001). *One-class classification: Concept learning in the absence of counter-examples*. PhD thesis, Technische Universiteit Delft.

Zhang, B. and Zuo, W. (2008). Learning from positive and unlabeled examples: A survey. In *2008 International Symposiums on Information Processing*, pages 650–654. IEEE.