CLASSIFICAÇÃO DE ORDENS DE SERVIÇO DE EQUIPAMENTOS DE GRANDE PORTE DE MINERAÇÃO

Diego da Silva Rodrigues

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: José Manoel de Seixas

Rio de Janeiro
Janeiro de 2013

CLASSIFICAÇÃO DE ORDENS DE SERVIÇO DE EQUIPAMENTOS DE GRANDE PORTE DE MINERAÇÃO

Diego da Silva Rodrigues

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. José Manoel de Seixas, D.Sc.

Prof. Luiz Pereira Calôba, Dr.Ing.

Prof. Vitor Hugo Ferreira, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JANEIRO DE 2013

Rodrigues, Diego da Silva

Classificação de Ordens de Serviço de Equipamentos de Grande Porte de Mineração/Diego da Silva Rodrigues. – Rio de Janeiro: UFRJ/COPPE, 2013.

XII, 46 p.: il.; 29, 7cm.

Orientador: José Manoel de Seixas

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2013.

Referências Bibliográficas: p. 44 – 46.

1. Categorização de Texto. 2. Mineração de Texto. 3. SVM. 4. Máquina de Comitê. I. de Seixas, José Manoel. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

À minha família, pelo amor e apoio incondicional ao longo da vida.

Agradecimentos

Agradeço a minha mãe Angela e a minha madrinha Dora pela minha educação, criação, e por todo amor recebido ao longo da vida, que me deram a força necessária para superar os desafios.

Agradeço ao meu orientador José Manoel de Seixas pela sabedoria, orientação e paciência ao me ensinar muito além necessário para desenvolver este trabalho e amadurecer como estudante e como profissional.

Agradeço a Mariana Teixeira por estar ao meu lado, pelo amor, carinho, incentivo e compreensão durante esta jornada.

Agradeço a Convergência Latina e seus colaboradores, por proporcionar o material para esta dissertação e pelo apoio profissional. Agradeço ao amigo Alexander Virtser pelo apoio e orientação durante o percurso na empresa e no desenvolvimento deste trabalho.

Agradeço aos meus amigos pelo incentivo e pela companhia ao longo de todos esses anos.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

CLASSIFICAÇÃO DE ORDENS DE SERVIÇO DE EQUIPAMENTOS DE GRANDE PORTE DE MINERAÇÃO

Diego da Silva Rodrigues

Janeiro/2013

Orientador: José Manoel de Seixas

Programa: Engenharia Elétrica

A informação relacionada aos reparos realizados em equipamentos de grande porte, em um setor específico da indústria, fica armazenada em ordens de serviço. A análise dos reparos executados permite identificar padrões e tendências de que tipo de problema está ocorrendo com os equipamentos ou com a frota. Porém, esta informação está registrada em texto desestruturado, escrito por inspetores e mecânicos. Para analisar este histórico agregado de ordens de serviço é necessário realizar a categorização individual das mesmas em termos da ação realizada e qual componente e qual o sintoma que se manifestou. O objetivo deste trabalho é propor uma metodologia de extração automática da informação, a partir do que é escrita e classificar as ordens de serviço. Para isto, foram utilizadas técnicas de mineração de texto para construir uma matriz documentos $\times$ termos e, em seguida, apresentar a um classificador multi-classe para as categorias de ação e sintoma. O classificador escolhido foi a máquina de vetor suporte, estendida para o modelo multi-classe utilizando votação de especialistas (máquina de comitê). Foram implementadas seis arquiteturas de máquina de comitê, combinando diferentes topologias e parametrizações para cada especialista. A taxa de acerto média, calculada com validação cruzada com dez partições, para a melhor arquitetura, foi de $98,3\%$ para categoria de ações e $94,5\%$. Além disso, foram discutidas quatro estratégias de parametrização das máquinas de comitê, para reduzir, significativamente, o tempo de trabalho de um analista experiente.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

LARGE MINING OFFROAD TRUCKS WORK ORDER CLASSIFICATION

Diego da Silva Rodrigues

January/2013

Advisor: José Manoel de Seixas

Department: Electrical Engineering

The information related to repairs executed on large equipments, from a given industry segment, is documented on service orders. The analysis of those repairs allows pattern recognition and trending of which type of problems is happening for each equipment or for the whole fleet. However, this information is registered in unstructured text, written down by inspectors and mechanics. In order to analyze this aggregated historical database of service orders is necessary to categorize individually those orders in terms: which action was taken, which component was repaired and which symptom manifested. The goal of this work is to stabilish a methodology for automatic information extraction from the written text and classify the service orders. To achieve this goal, text mining techniques were applied to construct a document $\times$ term matrix, followed by presenting those vectors to a multiclass classifier trained for action and symptom categories. Support Vector Machines were used as the classifier, combined to a multiclass approach using expert voting (Committee machine). Six different topologies were developed, combining different specialist parametrizations and multiclass approach. The mean accuracy rate for training, using k-fold cross validation, for the best topology, was 98.3% for actions and 94.5% for symptoms. Moreover, four strategies were discussed for customizing the classifier to reduce, significantly, the amount of working hours from an experient analyst.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Na manutenção de equipamentos de mineração de grande porte, uma das principais fontes de dados para monitoramento da saúde mecânica dos equipamentos é o conjunto de ordens de serviço, que são registros individuais de reparos executados nos mesmos. Centenas dessas ordens de serviço são executadas numa oficina, a cada mês. Estas ordens de serviço armazenam diversas informações referentes ao ciclo de vida de um problema no equipamento. Uma análise detalhada de cada ordem, individualmente, permite entender o atual cenário de problemas de um determinado equipamento, que podem estar distribuídos pelos seus componentes e peças.

A abertura de uma ordem de serviço pode ser realizada por: um inspetor, em uma inspeção de campo, realizada periodicamente, quando detecta algo a ser corrigido; um mecânico que esteja executando algum reparo e detecte um problema; uma análise de dados que detecte algum problema eminente - vibração, análise de óleo, análise de sensores eletrônicos embarcados. Durante a abertura, o responsável especifica o sintoma detectado, uma ação a ser executada para mitigação do problema, bem como determina qual parte do equipamento precisa ser reparada. Caso a ordem de serviço não seja imediatamente executada, ela é armazenada para posterior agendamento da execução, recebendo o nome de *backlog*. Caso o equipamento necessite ser reparado imediatamente, a ordem de serviço é caracterizada como um reparo corretivo.

Após a execução, o mecânico responsável atualiza o registro da ordem de serviço, escrevendo a ação executada, as peças utilizadas, quais foram as causas, problemas encontrados e o número de horas de execução. A ordem de serviço é inserida no sistema de dados e, por fim, encerrada.

1.1 Atributos das Ordens de Serviço

Se as ordens de serviço possuíssem classificação categórica dos principais campos responsáveis pela apropriação do evento detectado ou ocorrido, seria possível analisar

a oficina em diferentes dimensões. Estas possibilidades estão listadas abaixo:

- Com qual frequência um determinado determinado evento ocorre.
- Se há relação de causa-efeito entre diferentes eventos.
- Qual é o desempenho de diferentes indivíduos, participantes dos processos de detecção ou execução, relacionados aos eventos.

Cada uma dessas análises permitiria agregar valor diferenciado aos processos da oficina, seja no treinamento dos mecânicos, seja na estimativa da ocorrência de problemas dada análise de causa-efeito ou na inferência de custo estimado para o próximo período. Independente da contribuição, todas dependem de uma correta categorização dos campos registrados pelos inspetores e mecânicos nas ordens de serviço.

Nestas ordens de serviço, existem alguns campos que são campos escrito a mão livre. O registro eletrônico é feito por um indivíduo que não o responsável pela abertura ou execução da ordem. Portanto, todo tipo de erro de caligrafia, escrita, sintaxe pode ser encontrado nesses campos. Além disso, por haver uma interação com um terceiro participante do processo, o registro escrito precisa ser interpretado antes de ser apropriado em meio eletrônico. Além dos campos registrados a mão livre, outros campos são categóricos, mas também sujeitos a erro de registro. Em geral, os equipamentos possuem uma taxonomia profunda que define uma topologia de componentes, subcomponentes e partes. Essa topologia possui uma codificação, e o mecânico deve registrar o código corretamente na ordem. Além do erro do registro manual e interpretação, o mecânico pode não consultar o manual específico para encontrar o código referente a parte específica na qual o reparo foi efetuado e registrar um código errado. É comum, por exemplo, ser utilizado um código de um elemento em nível mais alto na hierarquia topológica (por exemplo, um reparo em uma parte específica do motor, ser categorizado como um reparo no motor).

1.2 Objetivo

Para viabilizar as análises supracitadas, o objetivo deste trabalho é extrair da ordem de serviço as informações necessárias para categorizá-la corretamente, no que tange qual atividade foi realizada em qual componente e o porquê. O trabalho pode ser dividido em duas partes. A primeira parte consiste na extração do conhecimento existente em cada ordem de serviço. Para isto, é necessário utilizar técnicas de mineração de texto para limpar e identificar termos relevantes, que sejam capazes de ajudar a determinar o que ocorreu. A base de dados de ordem de serviço pode,

então, ser utilizada para treinar um conjunto de classificadores com o objetivo de corretamente categorizá-la.

No capítulo 2 serão discutidas as técnicas de mineração de texto escolhidas para realizar a extração, limpeza e organização da informação. Além disso, será apresentado qual metodologia e algoritmo de classificação foi utilizado. O capítulo 3 apresentará como as técnicas foram utilizadas para formar as metodologias existentes neste trabalho. O capítulo 4 discute os resultados obtidos. O último capítulo disserta sobre as possíveis evoluções do trabalho.

Capítulo 2

Revisão Bibliográfica

A tarefa de agrupamento de documentos remete às grandes bibliotecas e às disciplinas de biblioteconomia, nos tempos onde não havia meios digitais para organização, taxonomia e armazenamento digital da informação. O principal objetivo era a capacidade de agrupar os documentos por categorias pré-estabelecidas, de forma a consolidar em grupos aquilo de conhecimento similar, auxiliando a busca e a organização. Num contexto mais atual, a categorização e agrupamento de documentos permite executar buscas por conteúdo na internet, encontrar padrões e tendências em textos jornalísticos, redes sociais, além de filtrar e-mails de spam.

No escopo deste trabalho, o agrupamento de documentos tem como viés permitir a análise de ocorrência de falhas corretivas em equipamentos de grande porte, extraindo a informação das ordens de serviço de manutenção. O problema será contextualizado de duas formas: pela categorização de documentos, através de extração de termos chaves e comparação com um dicionário, e pela categorização supervisionada de documentos.

Neste capítulo serão descritas as técnicas de extração da informação de campos textuais utilizadas neste trabalho. Também serão descritas as técnicas de aprendizado de máquina utilizadas para realizar a categorização das ordens de serviço.

2.1 Extração do Vetor de Termos

Técnicas de tratamento para detectar termos específicos nos textos não-estruturados das ordens de serviço são necessárias para identificar a porção relevante do que está escrito. Essas técnicas podem ser divididas em algumas categorias: segmentação (*tokenization*), redução de radicais e normalização. A descrição e detalhamento de cada uma das etapas pode ser encontrada em [6].

- **Segmentação:** Consiste em segmentar o texto em palavras (*tokens*). Esta é a primeira etapa do processamento de texto. Além disso, é nesta etapa que carac-

teres irrelevantes são descartados, ou termos de exclusão são removidos. Estes termos de exclusão (conhecidos como *stop words*) são palavras da língua específica que não agregam informação para a análise que será realizada e podem ser descartadas antes da execução das outras etapas de pré-processamento.

- **Normalização:** É a atividade de remover plurais, identificar sinônimos e encontrar radicais das palavras.

Para a etapa de segmentação é suficiente a utilização de bibliotecas de tratamento de *strings* encontradas em diversas linguagens de programação. Quanto à normalização, a redução dos radicais utiliza-se de um algoritmo específico para o português, conhecido como Orengo [1], que baseia-se no algoritmo de radicalização de Porter [7], o qual foi desenvolvido para lingua inglesa. Este algoritmo será explicado na Subseção de redução de radicais. Além disso, é necessário ter um dicionário técnico de termos e um algoritmo de comparação de *strings*, que será descrito na próxima seção.

2.1.1 Distância de Edição

O processo de normalização de texto depende de um dicionário técnico. Este dicionário contém os termos relevantes para o trabalho de categorização dos documentos. Além disso, possui a informação de relação entre sinônimos, para que múltiplas palavras sejam relacionadas e identificadas por um único termo. Após a divisão do texto não-estruturado em palavras, o processo de normalização depende de uma medida de distância para identificar qual elemento do dicionário está relacionado a cada uma das palavras encontradas numa dada ordem de serviço. Uma métrica utilizada para comparação de duas *strings* é a Distância de Edição ou Distância de *Levenshtein* [8]. A distância entre duas *strings* é definida pelo número mínimo de edições a ser realizado para que uma se iguale a outra. Edições podem ter três ações: inserção, remoção ou substituição de um único caractere da *string*. Por exemplo, a distância entre as *strings* "estrada" e "morada" é igual a 3, pois três alterações precisam ser realizadas (remoção da primeira letra da palavra "estrada", por exemplo, e substituição das letras "s" e "t" por "m" e "o" respectivamente). O método utilizado para calcular a distância é conhecido como algoritmo de Wagner-Fisher [9] e está definido abaixo:

$$\text{lev}_{a,b}(i,j) = \begin{cases} 0 & ,i = j = 0 \\ i & ,j = 0 \wedge i > 0 \\ j & ,i = 0 \wedge j > 0 \\ \min \begin{cases} \text{lev}_{a,b}(i-1,j) + 1 \\ \text{lev}_{a,b}(i,j-1) + 1 \\ \text{lev}_{a,b}(i-1,j-1) + [a_i \neq b_j] \end{cases} & , \text{ senão} \end{cases} \tag{2.1}$$

Uma forma de exemplificar a função é através do exemplo da figura 2.1. O cálculo da distância entre as duas *strings* inicia-se pela distância entre as duas primeiras letras. Por exemplo, na figura 2.1, a distância entre "m" e "e", no primeiro quadrante, é o menor valor das células adjacentes, ou seja 1 (segundo quadrante). Caso as letras fossem iguais, como está representado no terceiro quadrante, repete-se o elemento na diagonal (neste caso, 3, valor da célula $i-1$ e $j-1$). Desta forma, a matriz é preenchida e a distância é o valor encontrado na última célula, conforme descrito anteriormente. Utilizando a Distância de Edição é possível comparar cada uma das

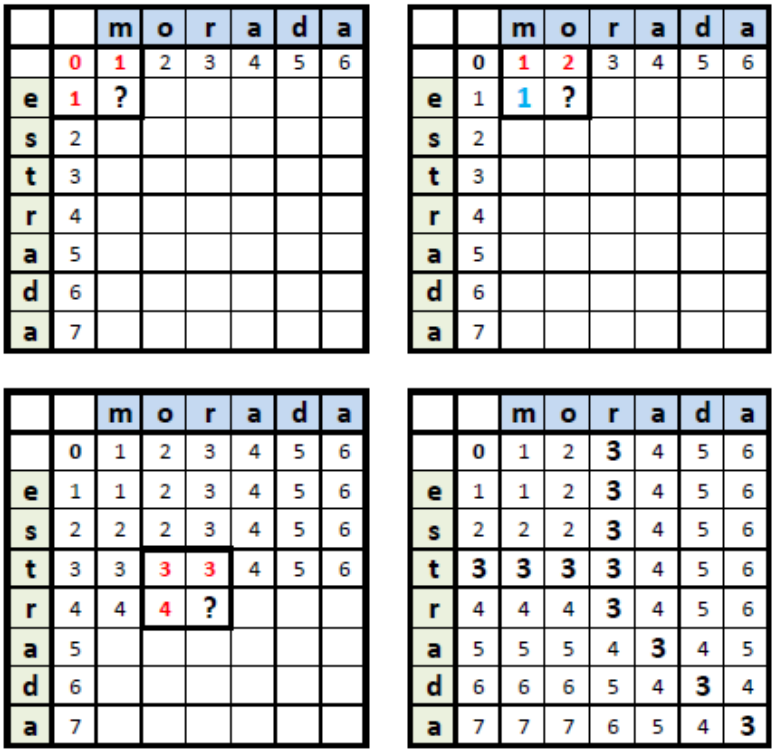

Figura 2.1: Exemplo do cálculo da distância de Levenshtein.

palavras obtidas de uma ordem de serviço e mapeá-las em termos relevantes de um dicionário técnico.

2.1.2 Redução de Radicais

De forma a reduzir o número de palavras encontradas no texto, um primeiro passo é a redução de radicais. Este passo visa remover plurais, diminutivos, femininos e flexões verbais. Um algoritmo conhecido para a língua inglesa é o Radicalizador de Porter [7]. Uma versão para o português foi desenvolvida em [1], chamado de RSLP (Removedor de Sufixos da Língua Portuguesa). Este algoritmo obedece a uma árvore de regras, na qual, em cada passo, um tipo de redução é aplicado. Cada uma das regras possui algumas exceções, que também são consideradas em cada passo do processamento do texto. O diagrama abaixo exibe a sequência em que as ordens de serviço são testadas.

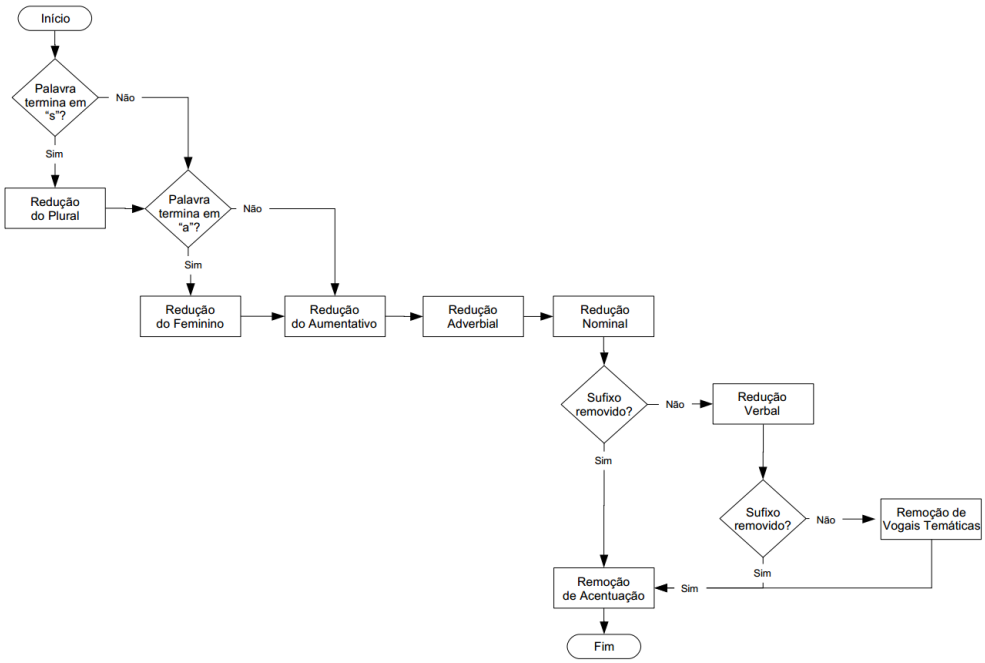

Figura 2.2: Diagrama especificando a árvore de decisões do algoritmo de radicalização desenvolvido em [1]. Figura adaptada de [2].

Em cada um dos passos, um tipo de redução é aplicado:

1. Redução de Plural: de acordo com [1], com raras exceções, o plural ocorre em palavras encerrando-se com um -s. Há exceções como Lápis, Português, que são tratadas como tal.

2. Redução de Feminino: Remoção do gênero. Por exemplo, Chinesa $\longrightarrow$ Chinês.

3. Redução de Advérbio: remoção de sufixo -mente do final dos advérbios (rapidamente, lentamente, etc).

4. Redução de Aumentativo/Diminutivo: Por exemplo, casinha $\longrightarrow$ Casa.

5. Redução de Sufixos de Substantivos: neste passo, são testados casos específicos de redução para alguns substantivos e adjetivos.

6. Redução de Verbos: neste passo, as regras servem para reduzir as flexões verbais encontradas.

7. Remoção de Vogal: Passo para tratar as palavras que não foram tratadas pelos passos de redução de substantivos ou verbos.

8. Remoção de Acentuação: Tratamento final de remoção de acentos.

Através de todos estes passos, o RSLP é capaz de aplicar a redução de radicais e eliminar todas as redundâncias e flexões que poderiam ser encontradas no texto.

Combinando as duas etapas descritas nesta seção, é possível extrair o vetor de termo de cada uma das ordens de serviço. A figura 2.3 exemplifica, para um texto de O.S. específico, como seria a divisão.

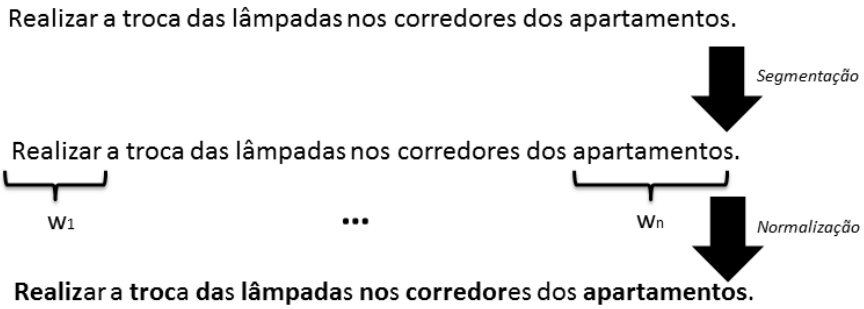

Figura 2.3: Diagrama representando a extração dos termos. Após a segmentação, estão ilustradas as palavras e após a normalização, estão ressaltados os radicais.

2.2 Categorização de Documentos

A tarefa de categorização (classificação) de elementos, dado um conjunto único de classes, no contexto do agrupamento de documentos, é conhecida como Categorização de Texto (*Text Categorization*, TC, em inglês) [6]. Dado um conjunto de categorias (tópicos, assuntos) e uma coleção de documentos, é o processo de encontrar a(s) categoria(s) correta(s) para os elementos da coleção.

O estudo da classicação automática de texto data dos anos 60, onde o principal objetivo era indexar literatura científica [10]. Atualmente, a mineração de texto (termo que dá nome ao grupo de metodologias de extração de informação, agrupamento, categorização de texto) é uma disciplina abrangente.

Há duas principais abordagens para executar categorização de texto. A primeira, conhecida como "engenharia de conhecimento" (*knowledge engineering*) tem como

objetivo o desenvolvimento de sistemas especialistas que categorizam os documentos, principalmente utilizando regras e especificação de termos e palavras-chaves. A outra abordagem, que será explorada neste trabalho, utiliza metódos estatísticos e de aprendizado de máquina para treinar classificadores que realizarão a tarefa de classificação dos documentos.

Entre as metodologias de aprendizado de máquina para categorização de documentos, uma bastante utilizada é o classificador conhecido como Naïve Bayes. Este classificador é baseado no teorema de Bayes e parte da premissa de que os termos que compõem uma determinada categoria de documentos (isto é, podem ser identificados e utilizados para a classificação) são independentes. Diversos trabalhos de categorização de texto desenvolveram suas metodologias utilizando este classificador [11], [12], [13] e [14]. Há uma extensão para a utilização do classificador bayesiano simples (*Naïve Bayes*) com treinamento semi-supervisionado, utilizando parte uma amostra de treino devidamente categorizada e uma amostra sem rótulos de classe. Esta é a contribuição de [15], [16] e [17]. Caso a premissa de independência não seja interessante ou suficiente para o problema em questão, é preciso supor um modelo paramétrico para distribuição multivariada dos termos que determinam uma categoria. O trabalho de [18] utiliza um modelo de regressão logística para treinar os classificadores.

Outras metodologias de treinamento também foram utilizadas na literatura: redes neurais [19]; um classificador conhecido como "Rocchio" [20], que parte de um conceito parecido com o do vizinhos mais próximos: o método de classificação envolve associar à observação o rótulo do membro do conjunto de treinamento de maior proximidade.

O classificador utilizado neste trabalho são as máquinas de vetor suporte, também utilizado em [21]. Este classificador será descrito a seguir.

2.2.1 Máquina de Vetor Suporte

A máquina de vetor-suporte (SVM), do inglês *Support Vector Machine* é um algoritmo que é capaz de elencar qual é o separador linear que separa dois conjuntos de pontos no plano, garantindo que a distância entre os conjuntos e o separador seja a maior possível [22]. De acordo com a teoria da dimensão V-C [22], tal separador linear é o que possui o maior poder de generalização. Isto quer dizer que um classificador ajustado utilizando essa metodologia é mais adequado para realizar a classificação de observações fora da amostra, isto é, outras observações retiradas da população mas que não possuiram exemplos no conjunto usado para treinar o classificador.

As SVMs foram propostas por Vapnik em 1995 [22], atuando na resolução de

problemas de classificação binários, tendo sido utilizadas com sucesso em aplicações de reconhecimento de padrões [23] [24].

O funcionamento de uma SVM pode ser simplificado da seguinte forma: para um conjunto de treinamento composto de duas classes, isto é, observações pertencentes a dois grupos de uma determinada população, o algoritmo de treinamento de uma SVM determina o hiperplano que separa os dois grupos, maximizando a distância entre os mesmos. Esta distância, representada pela distância entre as observações que se encontram na fronteira entre os dois grupos, é conhecida como margem de separação. Portanto, estes vetores que auxiliam a definir a margem, e portanto, "suportam" a construção de um hiperplano separador voltado para maximização da margem, são chamados de vetores de suporte [24]. Ver figura 2.4.

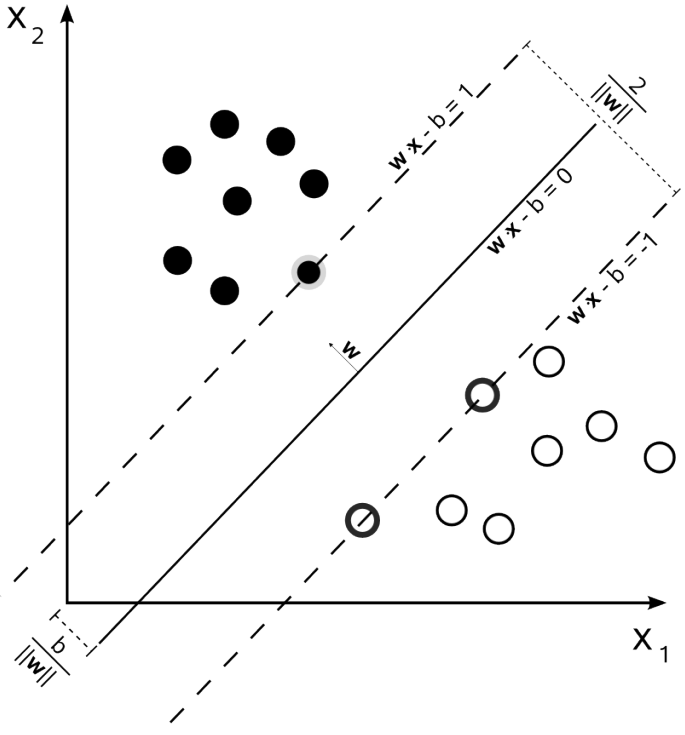

Figura 2.4: Hiperplano separador definido pela reta $\mathbf{w}^T\mathbf{x} - b$. Os vetores suporte do conjunto de observações em preto e branco estão ressaltadas outras observações do conjunto. Extraido de `http://en.wikipedia.org/wiki/Support_vector_machine`.

O treinamento de um SVM consiste em um problema de otimização quadrática. A formulação, que será vista adiante, possui a atrativa característica de assegurar a existência de um mínimo global que satisfaz a solução do problema [23].

A SVM é um classificador binário que pode ser adaptado para problemas multiclasse. A primeira formulação, que será vista na seção seguinte, refere-se a um problema binário com duas classes separáveis linearmente (existe um hiperplano que separa completamente as observações das duas classes). Em seguida, será descrito o relaxamento do método para atender a problemas com classes que não são separáveis

linearmente. Essas duas formulações costumam ser chamadas de SVM com margem rígida e margem suave. Por último, serão descritos alguns métodos de extensão de SVM para abordagem multiclasse.

2.2.2 SVM - Caso Linearmente Separável

O caso linearmente separável permite apresentar a formulação das máquinas de vetor suporte. Para este caso, a solução do treinamento de uma SVM consiste em encontrar o hiperplano que separa perfeitamente os pontos pertencentes às classes, maximizando a margem de separação. Esse hiperplano é chamado de hiperplano ótimo.

Dado o conjunto de observações de treinamento de tamanho $\mathbf{T}$, composto de n observações m-dimensionais $x_i = \{x_{i1}, x_{i2}, \ldots, x_{ij}\}$, onde i é o índice correspondente a observação e j é o atributo correspondente às m dimensões. Cada observação x_i possui um rótulo de classe associado y_i, sendo $y_i = 1$ quando x_i pertence a classe de interesse e $y_i = -1$ quando x_i pertence a classe complementar. Se os dados forem linearmente separáveis, é possível definir o hiperplano ótimo através da equação 2.2 [23]:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{2.2}$$

Onde $\mathbf{w}$ é o vetor normal ao hiperplano separador, $\mathbf{x}$ é o conjunto de observações e b determina o deslocamento do hiperplano em relação à origem. Assim, para $i = 1, \ldots, T$, existe a seguinte relação:

$$\mathbf{w}^T x_i + b \begin{cases} > 0, & y_i = 1 \\ < 0, & y_i = -1 \end{cases} \tag{2.3}$$

Entretanto essa formulação não garante que a margem seja a maior possível. Na figura 2.4, observa-se que é possível selecionar uma combinação dos parâmetros w e b que posicione as observações limítrofes de cada grupo, onde $g(\mathbf{x}) = 1$ para a classe representada pelos círculos pretos e $g(\mathbf{x}) = -1$ para classe representada pelos círculos brancos. Essa parametrização acaba garantindo as relações determinadas pela equação 2.3.

$$\begin{aligned} \mathbf{w}^T\mathbf{x} + b \geq 1, \quad \forall y_i = 1 \\ \mathbf{w}^T\mathbf{x} + b \leq 1, \quad \forall y_i = -1 \end{aligned} \tag{2.4}$$

Desta forma, o objetivo é escolher os parâmetros w e b que atendam estas condições. Isto é possível de ser feito ao minimizar a norma de w. Ao fazê-lo,

consequentemente, é obtido o separador de maior margem [23].

$$\begin{array}{ll} \text{minimizar} & J(\mathbf{w}, b) \equiv \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{sujeito à} & y_i\left(\mathbf{w}^T x_i + b\right) \geq 1 \quad i = 1, 2, \ldots, n \end{array} \tag{2.5}$$

Esse é um problema de otimização quadrática sujeito a um conjunto de restrições lineares de desigualdade. Aplicando as condições de Karush-Kuhn-Tucker (KKT) na formulação 2.5 [23]:

$$\begin{array}{ll} \text{maximizar} & J(\mathbf{w}, b, \lambda) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{T} \lambda_i \left[y_i\left(\mathbf{w}^T x_i + b\right) - 1\right] \\ \text{sujeito à} & \mathbf{w} = \sum_{i=1}^{T} \lambda_i y_i x_i \\ & \sum_{i=1}^{T} \lambda_i y_i = 0 \end{array} \tag{2.6}$$

Os multiplicadores de Lagrange λ_i podem ser iguais à zero ou positivos. Os vetores de suporte vão ser justamente aqueles associados a um multiplicador de Lagrange diferente de zero. Estas são as observações do conjunto de treino mais relevantes para a definição do separador: são os pontos que compõem a margem. Para encontrar os parâmetros, já existem diversas abordagens [23], sendo [25] uma abordagem bastante relevante. A implementação em MATLAB do algoritmo conhecido como Otimização Sequencial Mínima (*Sequential Minimal Optimization* - SMO), originário em [26], foi o método utilizado neste trabalho, como descreverá o Capítulo 3.

A próxima subseção descreve como esta formulação é flexibilizada para o caso em que os conjuntos não sejam linearmente separáveis.

2.2.3 SVM - Caso Não-linearmente Separável

Quando os conjuntos de classe não são linearmente separáveis, a formulação vista na seção anterior não é mais válida. Qualquer tentativa de definir um separador vai acabar com alguma das observações caindo em uma das três categorias:

- Observações localizadas fora da margem, que são corretamente classificadas. Estas observações continuarão obedecendo a restrição do modelo linear $y_i(\mathbf{w}^T\mathbf{x} + b) \geq 1$.

- Observações localizadas dentro da margem de separação, mas ainda sendo corretamente classificadas. Estas observações satisfazem a desigualdade $0 \leq\ y_i\left(\mathbf{w}^T\mathbf{x} + b\right) < 1$.

- Observações classificadas incorretamente. Estas obedecem a desigualdade $y_i\left(\mathbf{w}^T\mathbf{x} + b\right) < 0$.

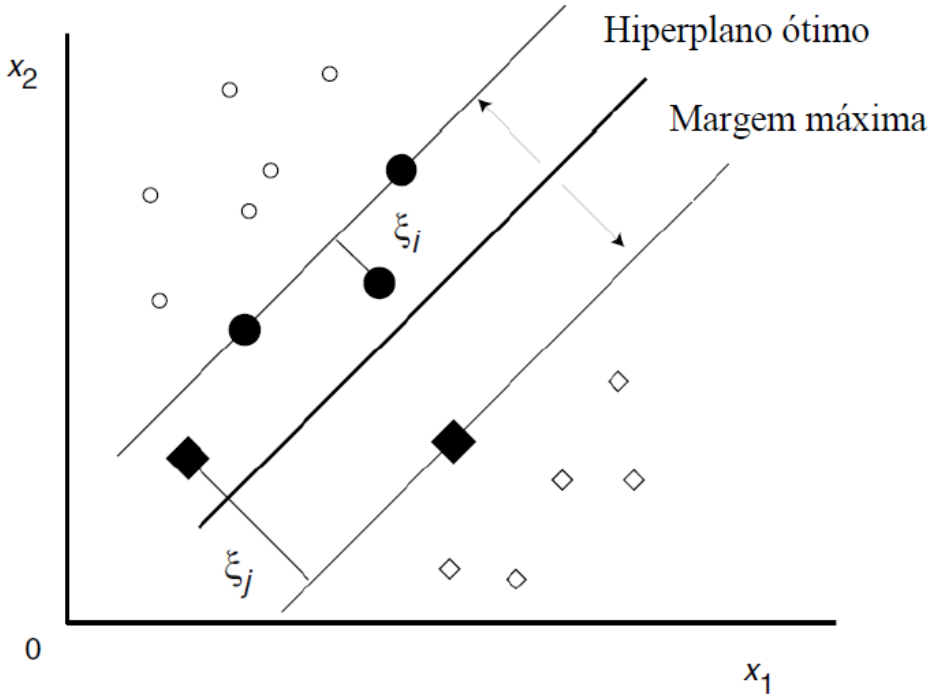

Figura 2.5: Exemplo de conjunto de dados não-linearmente separáveis. Algumas observações acabam residindo dentro da margem de separação. Figura retirada de [3].

Os três casos podem ser reunidos em um único conjunto de restrições, introduzindo uma nova variável ξ, definida na equação 2.7:

$$y_i[\mathbf{w}^T\mathbf{x} + b] \geq 1 - \xi_i \tag{2.7}$$

A primeira categoria das observações correspondem a $\xi_i = 0$, a segunda a $0 < \xi_i \leq 1$ e a última a $\xi_i > 0$. As variáveis ξ são conhecidas como *variáveis de folga*. Embora o problema de otimização seja alterado, o princípio continua o mesmo: encontrar o separador com a margem mais larga possível e ao mesmo tempo manter o menor número de observações com $\xi > 0$. Para a formulação do problema, isto é equivalente a alterar a função custo para 2.8:

$$J(\mathbf{w}, b, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{T} I(\xi_i) \tag{2.8}$$

onde ξ é o vetor de parâmetros ξ e:

$$I(\xi_i) = \begin{cases} 1 & \xi_i > 0 \\ 0 & \xi_i = 0 \end{cases} \tag{2.9}$$

O parâmetro C é uma constante de regularização que controla a influência dos dois termos. Entretanto, a equação 2.9 é difícil de minimizar [23], devido a existência da função não contínua $I(\cdot)$. Porém, é possível otimizar uma função relacionada que serve ao mesmo princípio, substituindo $I(\xi_i)$ por ξ_i. Assim, a função objetivo se torna:

$$\text{minimizar} \quad J(\mathbf{w}, b, \xi) = \tfrac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{T} \xi_i \tag{2.10}$$

$$\text{sujeito a} \quad y_i[\mathbf{w}^T \mathbf{x} + b] \geq 1 - \xi_i, \quad i = 1, 2, \ldots, T \tag{2.11}$$

$$\xi_i \geq 0, \quad i = 1, 2, \ldots, T \tag{2.12}$$

A derivação destas condições e o modo como o problema de minimização pode ser resolvido encontram-se em [23]. A formulação da SVM para o caso não linearmente separável introduz uma nova variável importante: a variável de regularização C. Esta variável determina a relevância das variáveis de folga ξ_i e deve ser definida durante o treinamento para determinar apropriadamente o separador.

2.2.4 Extensão SVM Multiclasse

A máquina de vetor-suporte, por natureza, é um classificador binário. Para utilização deste tipo de classificador em problemas multiclasse, é necessário realizar uma adaptação. A primeira formulação que surge, com naturalidade, é a conhecida como a *one-against-all*. Nesta adaptação, um classificador é treinado para cada uma das M classes do problema. Para uma observação fora da amostra, a função discriminante $g_j(\mathbf{x})$ é calculada para cada um dos j classificadores. A observação $\mathbf{x}$ será associada ao j-ésimo classificador que gerar a maior saída y_j. A interpretação para isto é: a classe escolhida é aquela para a qual a observação se situar mais distante do separador. Em [27] essa abordagem é utilizada. Uma limitação desta metodologia é que algumas regiões do hiperespaço acabam por não responder por nenhuma classe. Isto é, apresentam $g_j(\mathbf{x})$ negativo para todos os M classificadores. Nestas regiões de dúvida, o conjunto de classificadores simplesmente não tem como classificar a observação. A figura 2.6 representa este problema.

Uma segunda abordagem é conhecida como *one-against-one*. Neste caso, são treinados $M(M-1)/2$ classificadores, onde cada um deles realiza a comparação entre duas das M classes. A decisão pode ser encontrada através de uma votação. A desvantagem desta metodologia é a necessidade de treinar uma quantidade grande de classificadores.

O trabalho de [4] sugere a criação de grafos de decisão, similares a uma árvore de decisão, para realizar a discriminação multiclasse. Cada nó do grafo é um separador linear que vai particionando o hiperespaço a cada nível do grafo. Quando

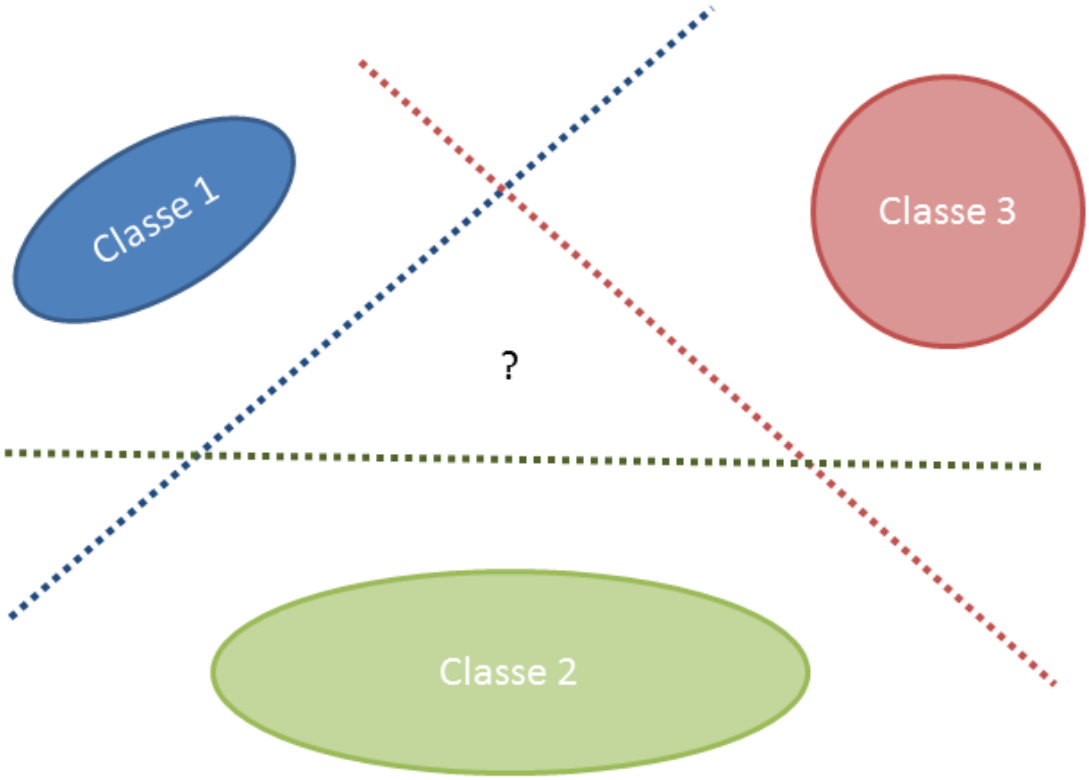

Figura 2.6: Exemplo da região de indeterminação causada pela abordagem *One-Against-All*. O triângulo no meio da figura não possui classe definida.

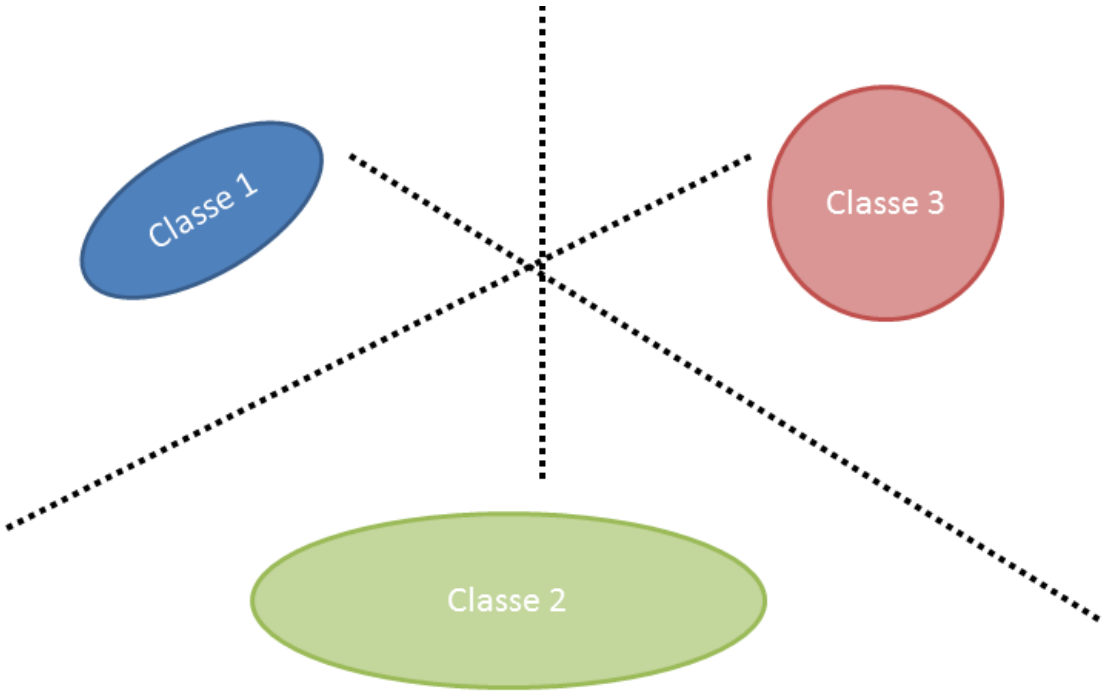

Figura 2.7: Exemplo de como a região de indeterminação causada pela abordagem *One-Against-All* é mitigada pela abordagem *One-Against-One*.

os separadores são SVM, o classificador é conhecido como *Directed Acyclic Graph - Support Vector Machine* (DAGSVM). Em [4], é demonstrado que a taxa de acerto para as DAGSVMs são maiores que a do *one-against-all* e do *one-against-one* e que o treinamento é mais rápido do que para o *one-against-one* para alguns conjuntos de dados selecionados. A figura 2.8 mostra como em um problema multiclasse as categorizações são realizadas, navegando na DAGSVM.

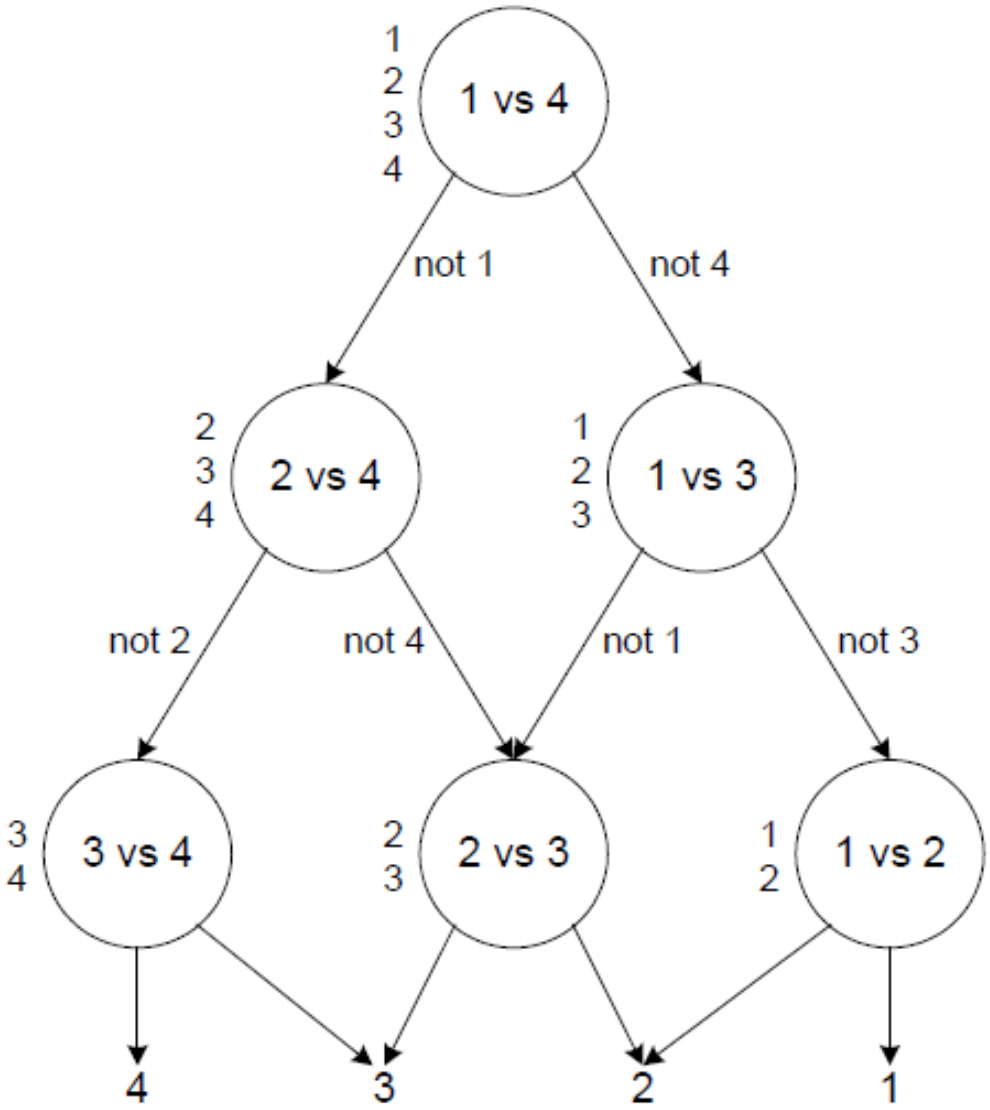

Figura 2.8: Exemplo de uma DAG. Extraído de [4].

O conjunto de classificadores binários pode ser interpretado como uma palavra binária de tamanho L. No caso do *one-against-all*, $L = M$, e no caso do *one-against-one*, $L = M(M-1)/2$. É possível determinar um L intermediário, onde uma determinada classe acione uma porção destes classificadores. Por exemplo, em um problema de M classes, escolhendo uma palavra $L = 5$, a resposta para uma das classes pode ser uma palavra binária $[+1 + 1 - 1 - 1 - 1]$. A comparação da palavra gerada pela saída dos classificadores e um dicionário que define as palavras

associadas a cada uma das classes, através da distância de Hamming é uma forma de estabelecer a classificação [28]. Por exemplo, no *One-Against-All*, a distância de Hamming máxima entre as palavras representantes de cada uma das classes é igual a 1. Apenas um erro (um BIT errado na palavra binária) faz com que o sistema multiclasse seja comprometido. Portanto, escolher uma palavra binária que permita uma flexibilidade maior quanto ao erro dos classificadores individuais permite estabelecer, de forma mais controlada, um compromisso entre a quantidade de classificadores treinados e as regiões de interseção entre os mesmos.

A extensão da formulação original de Vapnik [22] para o caso multiclasse é discutida em [29]. Neste trabalho, o problema multiclasse é abordado como uma única formulação, ao invés de segmentá-lo em diversas formulações binárias.

2.3 SVM - Utilização de *Kernel*

No caso do conjunto de dados não-linearmente separável, na Subseção 2.2.3 foi apresentada a formulação da SVM que é capaz de definir um separador "tolerante" a uma quantidade de observações localizadas dentro da margem ou do lado oposto. De qualquer maneira, o separador construído ainda é um hiperplano.

A função discriminante definida pelo treinamento de uma SVM apresenta o formato abaixo:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{i=1}^{N_s} \lambda_i y_i \mathbf{x}_i^T\mathbf{x} + b \tag{2.13}$$

Esta formulação é obtida da equação 2.10 através da representação dual do problema de otimização. As derivações podem ser encontradas em [23] e [24]. Na equação 2.13, N_s é o número de vetores suporte da SVM. É possível percebe que a função discriminante depende do vetor a ser classificado $\mathbf{x}$ e da operação de produto interno com os vetores suporte. Além disso, a formulação dual da equação 2.10 [23], é baseada no produto interno das observações do conjunto de treinamento. Isso possibilita utilizar uma técnica conhecida como *Kernel Trick*: através de funções com características específicas e que respeitem o Teorema de Mercer [23], é possível mapear os vetores da dimensão original para uma dimensão maior (até infinita), e construir a função discriminante nesta nova dimensão. As funções que possuem essas características são conhecidas como *Kernels*.

Seja a função $\mathbf{z} = \Phi(\mathbf{x})$ um mapeamento não-linear de $\mathbf{x} \in \mathcal{X} \longmapsto \mathbf{z} \in \mathcal{Z}$, o produto interno $\langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle \in \mathcal{Z}$ pode ser substituido pela função $K(\mathbf{x}_1, \mathbf{x}_2)$, que representa este produto interno no espaço $\mathcal{Z}$. Desta forma, a formulação dual para o problema proposto na equação 2.10 pode ser alterada para considerar o *kernel* $K(,)$.

A figura 2.9 apresenta um exemplo de como o problema não-linear pode ser

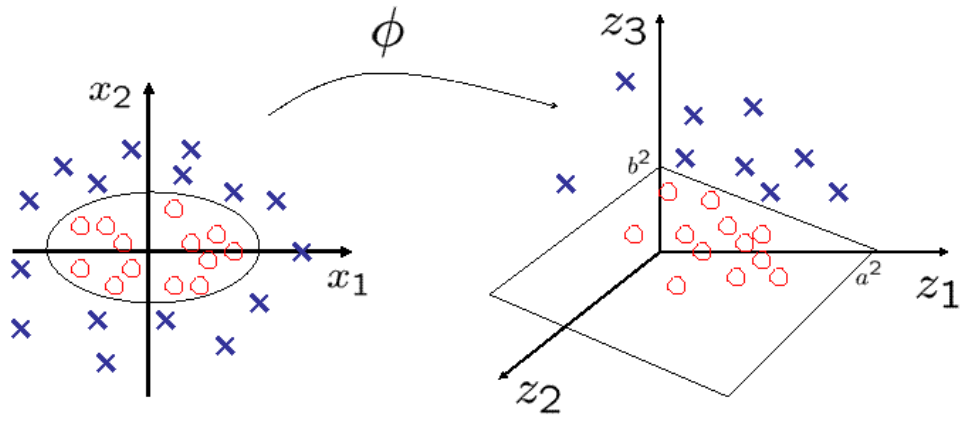

Figura 2.9: Exemplo de mapeamento dos dados de dimensão 2 para um vetor de atributos $\mathbf{x}$, para uma dimensão maior de tamanho 3 representada por $\mathbf{z}$. O hiperplano construído na dimensão maior, representa um discriminante não linear (uma elipse) no espaço original dos vetores. Figura adaptada de `http://omega.albany.edu:8008/machine-learning-dir/notes-dir/ker1/phiplot.gif`.

separado em outro espaço de dimensão elevada.

Dos diversos *kernels* existentes, foi selecionado para este trabalho o *kernel* RBF (*Radial Basis Function*), definido na equação 2.14.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right) \tag{2.14}$$

Este *kernel* possui o parâmetro σ. Este parâmetro σ representa o raio da "esfera de influência" de um determinado vetor $\mathbf{x}$ no espaço de dimensão elevada. Quando σ é pequeno, o vetor só influencia uma região muito próxima. Ao aumentá-lo, aumenta-se também sua região de influência. Portanto, ao utilizar o *kernel* RBF, é necessário escolher apropriadamente o parâmetro σ.

Na literatura, os autores de [30] propõem uma metodologia completa de treinamento para que os parâmetros específicos do *kernel* escolhido sejam otimizados em conjunto com a seleção dos vetores suporte. Já em [5], através da discussão sobre limites assintóticos para os parâmetros C e σ, é proposta uma metodologia de busca para estes parâmetros. Esta metodologia se baseia no compromisso entre "underfitting" e "overfitting" do classificador nos dados de treinamento. "Underfitting" é quando a parametrização do classificador torna-o muito simples para modelar a superfície de classificação mais adequada para o conjunto de treinamento. Esta adequação refere-se ao desempenho do classificador quando utilizado para classificar observações não pertencentes ao conjunto de treinamento.

No caso de "overfitting", a parametrização torna o classificador complexo. Desta

forma, o treinamento faz com que o percentual de acerto para o conjunto de treinamento seja alto, já que o classificador consegue modelar com muita complexidade a superfície de separação. Porém, ao ser apresentado a outras observações fora-da-amostra, o percentual de acertos cai, já que a superfície de separação tornou-se muito específica para o conjunto de treinamento e não garantiu generalização. A Figura 2.10 representa esta discussão.

O trabalho de [5] discute exatamente esta troca entre complexidade e generalização, para o treinamento de SVM utilizando *kernel* RBF. Esta troca se baseia principalmente na escolha do parâmetro C e no tamanho do *kernel* RBF σ. De acordo com os autores, o comportamento da SVM, baseado na seleção destes parâmetros, pode ser dividido nas quatro categorias abaixo:

- **"Underfitting" severo** - Neste caso, o hiperespaço dos atributos será associado à classe majoritária, que é aquela que apresenta a maior quantidade de observações na base de treino. Isto costuma acontecer nos seguintes casos: (1) σ é fixo e $C \to 0$, (2) $\sigma \to 0$ and C fixo em um valor pequeno e (3) $\sigma \to \infty$ e C fixo.

- **"Overfitting" severo** - Neste caso, pequenas regiões ao redor das observações do conjunto de treinamento associadas à classe minoritária são associadas a esta classe. O resto do hiperespaço é associado ã classe majoritária. Isto ocorre quando $\sigma \to \infty$ e C é fixo.

- Quando σ é fixo e $C \to \infty$, a SVM separa estritamente as observações de treinamento. Este também é um caso de "overfitting" quando há ruído.

- Se $\sigma \to \infty$ e $C = \tilde{C}\sigma$, onde $\tilde{C}$ é fixo, a SVM converge para a SVM linear com parâmetro de folga $\tilde{C}$.

Logo, a proposta de [5] é encontrar os parâmetros σ e C que garantam a generalização do classificador. Um método para fazer isso seria buscar combinações dos dois parâmetros que minimizassem uma estimativa de erro de generalização. Este é usualmente um processo custoso, já que depende de treinar o classificador diversas vezes, para diversas combinações dos parâmetros.

O método proposto por [5] divide a busca bidimensional em duas buscas unidimensionais. O primeiro passo do algoritmo é encontrar, para uma SVM linear treinada com o conjunto de treinamento, o parâmetro C que minimize uma estimativa de erro de generalização. Após isso, outra busca deve ser realizada, desta vez sobre a reta definida por $\log\ \sigma^2 = \log\ C - \log\ \tilde{C}$, onde $\tilde{C}$ é o valor de C encontrado na primeira busca, para uma SVM linear.

Para implementar a busca foi utilizado o algoritmo do MATLAB de busca direta conhecido como *patternsearch* [31]. O algoritmo funciona da seguinte forma:

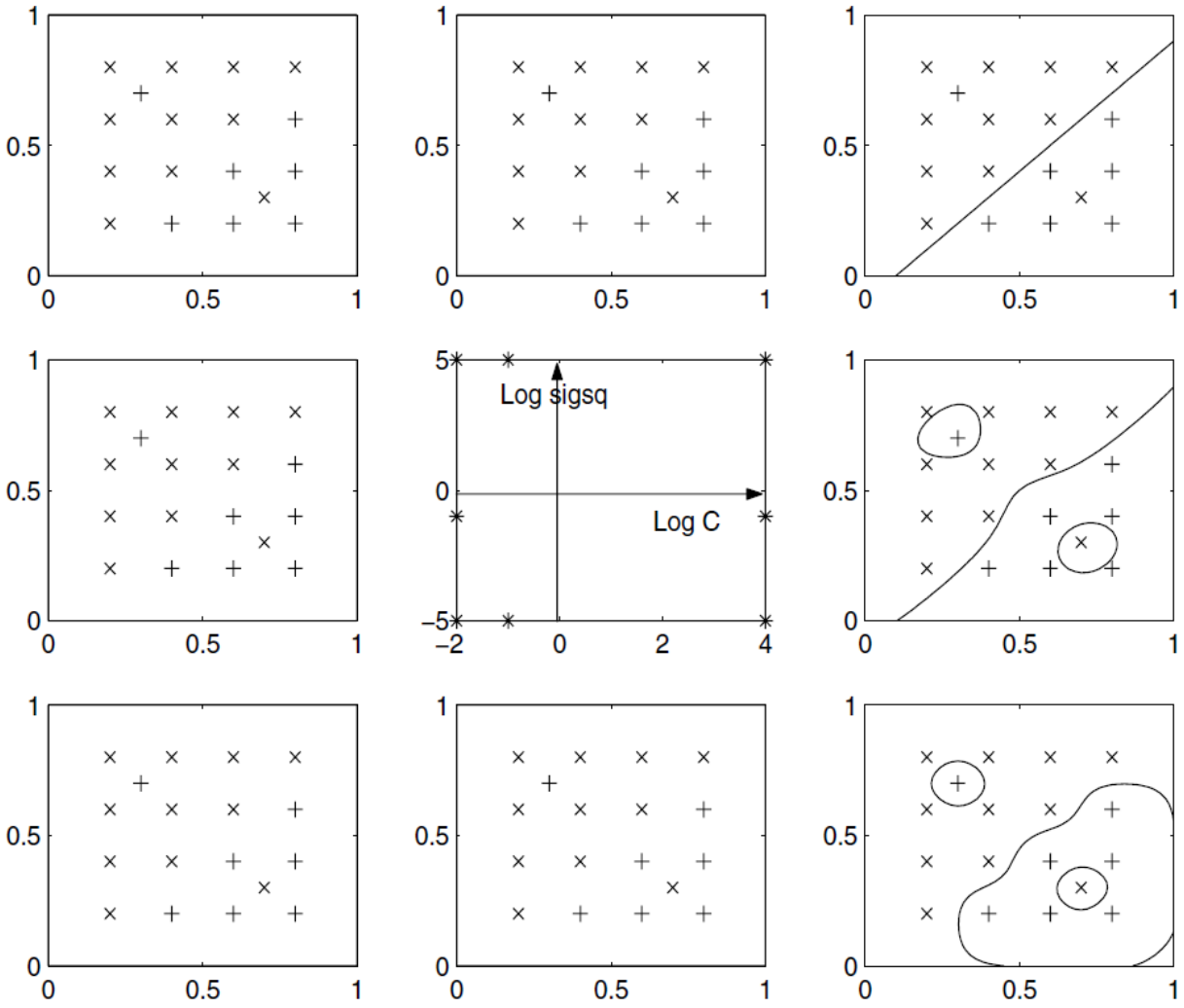

Figura 2.10: Exemplo extraído de [5]. As 8 figuras representam o separador entre a classe majoritária 'x' e a classe minoritária '+'. É possível notar quatro combinações de parâmetros que resultam em "underfitting" (nenhum separador é criado). Dois casos resultam em "overfitting", onde a superfície de separação cria uma pequena região em torno do elemento extremo de cada uma das classes. Apenas o separador representado pelo gráfico superior, a esquerda, apresenta um exemplo adequado de generalização.

1. A busca se inicia calculando o valor da função objetivo em um ponto inicial determinado pelo usuário.

2. Etapa de *pooling*: o algoritmo calcula o valor da função objetivo em algumas direções, a partir do ponto inicial. O conjunto de direções do método de *pool* conhecido como $Np1$ (N plus one), onde N é o número de dimensões do espaço de busca, para um problema bidimensional, é composto dos vetores $d_1 = [+1 \ +0]$, $d_2 = [+0 \ +1]$ e $d_1 = [-1 \ -1]$. Caso o valor da função objetivo seja mais favorável para algum destes pontos $x_0 + d_i$ (x_0 sendo o ponto inicial), este ponto se torna o novo ponto inicial para a execução de mais uma etapa de *pooling*.

3. Variação do *mesh*: O valor de *mesh* é um ponderador para a movimentação das $Np1$ direções, que se inicia com valor $m = 1$. Caso a etapa de *pooling* determine um novo ponto inicial de função objetivo mais favorável, este valor de *mesh* aumenta, por um fator definido por usuário. Caso nenhum ponto mais favorável seja encontrado, o valor de *mesh* é reduzido, para uma busca com maior acurácia ao redor do ponto atual. Desta forma, o valor de *mesh* costuma ser o critério de parada do algoritmo.

O algoritmo *patternsearch* foi utilizado nas duas buscas necessárias para se estabelecerem os parâmetros dos especialistas SVM com *kernel* RBF.

2.4 Categorização de Ordens de Serviço

Nenhum artigo foi encontrado, na literatura, para o problema específico de categorização de ordens de serviço de uma oficina de equipamentos de grande porte. No escopo da manutenção, foram encontrados dois artigos que tratam do mesmo problema: a categorização de ordens de serviço de sistemas de distribuição de energia elétrica, voltado para a correta priorização das ordens mais críticas e despacho das equipes para o campo [32], [33]. Dado um conjunto de ordens necessárias de execução em diversos lugares diferentes da malha, seja por demandas emergenciais ou comerciais, é necessário despachar uma equipe de campo para executar os serviços. Portanto, agrupar as execuções geograficamente com priorização de criticidade torna-se o objetivo do trabalho.

A próxima seção apresenta as metodologias que serão utilizadas neste trabalho.

Capítulo 3

Metodologia

Este capítulo descreve como o processo de classificação das ordens de serviço é realizado. O primeiro passo é a definição do que ocorre no processo de classificação: qual é a informação que deve ser rastreada nas ordens de serviço e sob que categorias uma ordem de serviço pode ser associada. Em seguida, é descrita a primeira etapa do processamento dos dados. Ela consiste em extrair do texto das ordens de serviço os termos relevantes para a metodologia de classificação. O que foi implementado está descrito na Seção 3.3. A segunda etapa da extração de característica consiste na utilização de um dicionário técnico para detecção de termos relevantes. Este dicionário está dividido por classes e contém uma relação entre sinônimos. A última seção descreve como as SVMs foram utilizadas de forma combinada produzindo um modelo de classificação multiclasse para atender ao problema de categorização das ordens de serviço.

3.1 Classes das Ordens de Serviço

Cada ordem de serviço possui um conjunto de campos categóricos e textuais que definem o que ocorreu. Para a base de ordens de serviço analisada, apenas os campos de texto desestruturado foram selecionados para prover os atributos do modelo. Isto ocorreu porque os campos de texto, embora desestruturados, representam um relato concreto, rico em informação, do que foi detectado e executado na ordem de serviço. Foi, inclusive por isso, que um modelo de mineração de texto foi selecionado para possibilitar o trabalho de classificação, já que os campos categóricos que, em teoria, proveriam informação discretizada "bem comportada", não puderam ser utilizados.

3.1.1 Definição das Classes

O objetivo do trabalho de classificação é identificar, nas ordens de serviço, qual atividade padrão ocorreu. Uma atividade padrão é a composição de um sintoma, um com-

ponente e uma ação executada. Por exemplo, uma determinada atividade padrão pode ser: "*Realizar a ação1 no componente1 devido ao sintoma1*". Neste contexto, a atividade padrão J é uma combinação de três classes categóricas $J \equiv \{A, C, S\}$, onde no exemplo temos $J = \{a = \text{ação1}, c = \text{componente1}, s = \text{sintoma1}\}$.

Para encontrar as atividades do tipo padrão, as funções discriminantes de classificação $g(\mathbf{x})$ podem ser definidas de algumas formas diferentes, a partir do vetor de entradas $x = x_1, x_2, \ldots, x_n$, onde n é o número de atributos utilizados. Idealmente, a função discriminante deveria ser construída para realizar a classificação da atividade padrão J a partir dos atributos $\mathbf{x}$ e chamada de $g_J(\mathbf{x})$. Entretanto, foi optado por realizar a construção de classificadores para categorizar, separadamente, ação, sintoma e componente, através de funções discriminantes $g_A(\mathbf{x})$, $g_C(\mathbf{x})$ e $g_S(\mathbf{x})$ respectivamente. Desta forma, a classificação da atividade padrão é a tupla definida como:

$$g_J(\mathbf{x}) \equiv \{g_A(\mathbf{x}), g_C(\mathbf{x}), g_S(\mathbf{x})\} \tag{3.1}$$

3.2 Coleta de Dados

O conjunto de ordens de serviço foi obtido da base de dados e classificado por um conjunto de engenheiros nas três categorias $\{A, C, S\}$. Posteriormente, elas foram avaliadas de forma a comparar a consistência da categorização dos diferentes indivíduos que participaram do processo. Um total de 5000 ordens de serviço foram homologadas para consolidação da amostra de treinamento. Como será visto adiante, foi construído um dicionário técnico para determinar quais termos encontrados nas ordens de serviço serão relevantes para cada um dos três discriminantes. Portanto, apenas um subconjunto da amostra de treinamento foi utilizado para os três diferentes problemas de classificação.

3.3 Extração de Características

A extração de características consiste em obter dos campos de texto desestruturados das ordens de serviço os termos necessários para realizar o processo de classificação. Esta etapa pode ser dividida em duas partes: o pré-processamento de texto para limpeza e exclusão de palavras irrelevantes e a detecção de *termos*. Os termos são expressões de uma ou mais palavras utilizadas como atributos de entrada dos classificadores. Ou seja, das ω palavras encontradas nos três campos de texto desestruturado das ordens de serviço, é extraído um único vetor de termos $\mathbf{x}$. Esta nomenclatura (palavras e termos) será utilizada ao longo do trabalho para separar o conteúdo bruto da ordem de serviço da informação relevante encontrada para o algoritmo de classificação. O diagrama da Figura 3.1 exemplifica o processo de ex-

tração de características utilizado. Este processo será descrito ao longo desta seção em maiores detalhes.

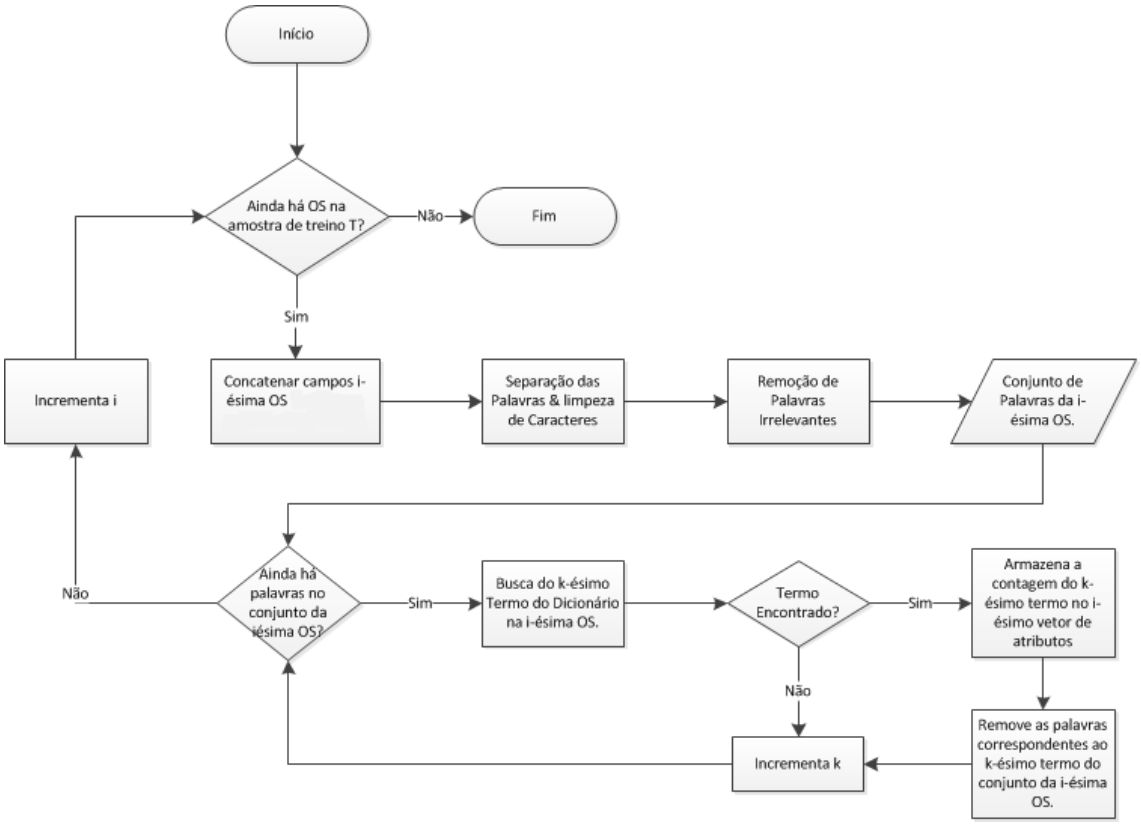

Figura 3.1: Fluxograma de extração de características das ordens de serviço para composição da amostra de treinamento.

A extração de características inicia-se pela limpeza do texto. Nesta etapa, são removidos caracteres não-alfanuméricos, acentuação e caracteres específicos da língua portuguesa. Em seguida, é realizada a exclusão de algumas palavras irrelevantes para o trabalho de categorização. É utilizado um dicionário de palavras irrelevantes (*Stop Words*) para determinar quais devem ser excluídas neste estágio do pré-processamento. Após esta etapa, inicia-se a etapa de normalização do texto e extração de termos, que será descrito nas próximas subseções.

3.3.1 Dicionário Técnico

Além do dicionário de palavras irrelevantes, outro dicionário foi construído para auxiliar a etapa de extração de características. Este segundo dicionário foi batizado de *Dicionário Técnico de Termos*. Este dicionário serve para dois propósitos:

- Identificar quais termos são sinônimos.
- Determinar quais termos são relevantes para cada uma das classes das três categorias.

Rótulo de Classe	**Termos**
COMPONENTE 1	termo1, termo 2, termo 3, termo 4
COMPONENTE 2	termo5, termo6
COMPONENTE 3	termo7, termo8

Tabela 3.1: Exemplo da estrutura do dicionário técnico de termos para três classes da categoria de componentes C.

A definição de quais termos são relevantes para cada uma das classes foi um trabalho de associação feito por um especialista técnico em manutenção dos equipamentos de mineração de grande porte. A tabela 3.1 exemplifica como é a estrutura do dicionário para a categoria de componentes. Alguns termos específicos são relevantes para um rótulo de classe específico da categoria.

Logo, o dicionário técnico é a parte principal da metodologia de seleção de atributos. Apenas os termos contidos no dicionário são buscados nas palavras encontradas nas ordens de serviço, por categoria. A próxima subseção explica como essa busca é realizada.

3.3.2 Normalização do Texto

Após a segmentação do texto das ordens de serviço em palavras, é realizado o processo de normalização do texto: a identificação de quais dessas palavras são termos expecíficos existentes no dicionário técnico. O algoritmo de busca realiza até três comparações, para cada uma das palavras existentes no dicionário, de forma a extrair os termos existentes em meio ao texto da ordem de serviço. A primeira comparação é exata. Caso o i-ésimo termo do dicionário exista no texto da ordem de serviço processada, este termo é armazenado como existente e é removido da cadeia de caracteres. Caso ele não exista, é realizada a busca utilizando o algoritmo de *Levenshtein* descrito com limiar "'MINIMUM_MSD"' igual a $0,8$. Por último, o radical do i-ésimo termo é procurado no texto da O.S. com limiar "'MINIMUM_MSD"' igual a $0,9$. Estes parâmetros foram selecionados utilizando validação manual sobre um conjunto de 500 ordens de serviço. Para cada experimento, as 500 ordens de serviço eram validadas, tolerando-se nenhum erro. Os parâmetros foram inicializados em $0,95$ e sofreram variação de $0,05$ a cada iteração da validação.

O pseudo-código do algoritmo utilizado encontra-se na figura 3.2.

Os termos são extraídos individualmente dos campos de texto de ação sugerida, ação executada e sintoma. Um exemplo deste processo encontra-se na tabela 3.2. Após a extração dos termos, eles são agrupados e contabilizados (Tabela 3.3). Cada i-ésimo elemento do vetor de atributos x_i é a soma de quantas vezes o termo i foi encontrado nos três campos. No exemplo da Tabela 3.3, os i-ésimos elementos correspondentes a {realizar, substituir, filtro, limalhas} apresentariam os valores

```
sort Term_Dictionary by number_of_terms desc;
while i <= Term_Dictionary.length() {
        while j <= OS.length() {
                if Term_Dictionary.Term(i) = OS.Term(j) {
                        match=1; break;
                }
                if MSD(Term_Dictionary.Term(i), OS.Term(j))>MINIMUM_MSD {
                        match=1; break;
                if MSD(Term_Dictionary.Term(i).stem(),
OS.Term(j).stem())>MINIMUM_MSD_STEM
                         match=1; break;
                j++;
        }
i++;
}
```

Figura 3.2: Pseudo-código do algoritmo de busca no dicionário técnico.

Campo	**Texto**	**Termos**
Campo Categórico 1	Realizar a substituição do componente devido a presença partículas	realizar, substituir, componente, partículas
Campo Categórico 2	Componente foi substituido	componente, substituir
Campo Categórico 3	Presença de partículas	partículas

Tabela 3.2: Exemplo genérico de extração da informação da ordem de serviço.

Termo	**Quantidade**
realizar	1
substituir	2
componente	2
partículas	2

Tabela 3.3: Exemplo de contagem de termos.

exibidos na tabela, enquanto todos os outros elementos, correspondentes aos outros termos do dicionário, apresentariam valor igual à zero.

O processo de normalização utilizando o dicionário técnico é o último passo necessário para formar a base de treino com os vetores de termos $\mathbf{x}$ encontrados nas ordens de serviço. A matriz de documentos por termos, de dimensão $j \times i$, onde j é o número de ordens da amostra categorizada pelos especialistas e i a quantidade de termos do dicionário forma por fim a base de treinamento para os classificadores.

3.4 Metodologia de Classificação

Conforme explicado na Subseção 3.1.1, para obter a atividade padrão realizada na ordem de serviço é necessário identificar o sintoma, a ação realizada e o componente no qual o serviço foi executado. A categorização de cada uma dessas características foi considerado um problema de classificação distinto. Portanto, para classificar a atividade padrão é necessário treinar um classificador multiclasse para ação, um para sintoma e um para componente. Após isso, combinando o resultado dos três classificadores, obtém-se a atividade que foi realizada.

Cada observação da amostra original de treinamento é um vetor de tamanho i, onde cada elemento é quantas vezes o i-ésimo termo foi encontrado na j-ésima ordem de serviço. Esta amostra completa poderia ser utilizada para treinar a função discriminante $g_J(\mathbf{x})$. Porém, como o problema de categorização foi dividido em três funções discriminantes referentes a cada uma das categorias, nem todos os i termos são relevantes para o treinamento dos três classificadores. Por exemplo, para o problema de classificação de ações, apenas um conjunto reduzido de termos, i_A foi considerado. Esses i_A termos são aqueles que *aparecem relacionados a pelo menos um rótulo de classe da categoria de ações.* A tabela 3.1 ilustra algumas entradas do dicionário técnico de componentes. Todos os termos listados na tabela fazem parte dos i_C termos considerados para o problema de classificação de componentes. Assim, o vetor de atributos original $\mathbf{x}$ deriva três definições distintas para cada um dos problemas de classificação: $\mathbf{x_A}$, $\mathbf{x_C}$ e $\mathbf{x_S}$, gerando três conjuntos de treinamento T_A, T_C e T_S de dimensionalidade $j \times\ i_A$,$j \times\ i_C$ e $j \times\ i_S$ respectivamente. Desta forma, três conjuntos de dados foram gerados, um para cada categoria.

3.4.1 Seleção do Classificador

O classificador escolhido foi uma máquina de comitê, que utiliza um conjunto de classificadores "especialistas". Cada um dos especialistas é uma máquina de vetor suporte. Dependendo da arquitetura selecionada, o resultado dos especialistas é combinado de forma diferente. Algumas arquiteturas foram experimentadas. O

resultado das implementações foi comparado utilizando validação cruzada *k-folds* [23] com dez partições. Estas arquiteturas serão descritas adiante. O resultado do treinamento será descrito no próximo capítulo.

A primeira arquitetura implementada foi o *one-against-all*, com especialistas sendo SVM lineares. Nesta implementação, cada especialista realiza a classificação de uma categoria específica contra todas as outras. Portanto, cada especialista representa um rótulo específico. A avaliação da máquina de comitê leva em consideração o resultado de todos os especialistas. O rótulo escolhido pela máquina de comitê é o rótulo do especialista que possuir o maior resultado numérico positivo. Caso o maior resultado seja menor que zero, significa que o vetor classificado se localiza na região de "dúvida". O parâmetro de margem C foi escolhido igual a 10^5. De acordo com [5] e [**?**], conforme $C \to \infty$, a SVM tende a reduzir a zero o tamanho da margem. Logo, a escolha deste valor representa uma hipótese inicial para o parâmetro, assegurando uma margem pequena, de forma a servir de comparação para as outras arquiteturas implementadas. O algoritmo de treinamento do *MATLAB* automaticamente normaliza o C de acordo com a quantidade de vetores representando o rótulo e o "não-rótulo" para cada especialista.

A segunda arquitetura implementada foi o *one-against-one*, com especialistas SVM lineares. A mesma consideração foi feita para o parâmetro de regularização. Neste problema, cada especialista responde por um rótulo contra outro. Portanto, o número de especialistas é igual ao total de combinação entre rótulos. O resultado da máquina de comitê é a votação dos especialistas. O rótulo que receber mais votos é o vencedor.

A terceira e a quarta arquitetura implementadas são extensões da primeira e da segunda, mas nestes dois casos foi implementado um algoritmo de busca para utilizar o *kernel* RBF. A terceira arquitetura usa este treinamento para os especialistas, os combinando com o *one-against-all* e a quarta arquitetura os combina com o *one-against-one*. A metodologia de treinamento dos especialistas é a seguinte:

1. É realizada a busca para encontrar o melhor C para uma SVM linear, utilizando o *patternsearch*, utilizando os parâmetros da Tabela 3.4.

2. Caso $C > 10^5$, o *kernel* linear é selecionado. Senão, é feita a busca utilizando *kernel* RBF. Este critério é baseado na conclusão de que se $C \to \infty$, a SVM tende a formulação linearmente separável ("hard margin").

3. E realizada a busca em $\log\ \sigma^2 = \log\ C - \log\ \tilde{C}$ para definir o tamanho do *kernel* e o parâmetro C correspondente.

O algoritmo usa os parâmetros descritos na tabela 3.4. Caso a etapa de *pool* encontre um ponto na vizinhança com função objetivo mais favorável, o tamanho do *mesh* é

Parâmetro	**Valor**
Critério de Parada: Tamanho do Mesh	0,9
Contração do Mesh	0,1
Expansão do Mesh	10
Tamanho Inicial do Mesh	100
Tamanho Máximo do Mesh	10^5
C_0	1

Tabela 3.4: Parâmetros utilizados no *patternsearch*.

multiplicado por 10. Caso não encontre um ponto favorável, o tamanho do *mesh* é multiplicado por $0,1$. Caso o tamanho do *mesh* fique menor do que 1, que também é o valor inicial do *mesh*, a busca se encerra. O valor inicial é $C_0 = 1$ para os dois algoritmos.

Duas estimativas de erro foram utilizadas para o erro de generalização: O número de vetores suportes, que é um limite superior para o erro *leave-one-out* [30]; o erro calculado sobre uma amostra de teste, correspondente a 20% da amostra de treino para cada especialista (*Hold-out*). O primeiro, embora seja um limite superior conservativo, é de fácil implementação e o mais rápido de ser calculado. Isto foi útil para reduzir o tempo total de treinamento das máquinas de comitê.

As arquiteturas foram comparadas utilizando a taxa de acerto. Esta taxa de acerto foi mensurada utilizando a média da taxa de acerto de dez partições do conjunto de treino (validação cruzada com dez partições). O próximo capítulo descreve os resultados obtidos pela metodologia.

Capítulo 4

Resultados e Discussões

Este capítulo apresenta os resultados alcançados para as máquinas de comitê treinadas para a categorias de ação e sintoma. A primeira seção apresenta o resultado para a categoria de ações, a segunda, os resultados para categoria de sintomas e a última seção discute os resultados e algumas estratégias para o uso das máquinas de comitê propostas.

4.1 Máquina de Comitê para Ações

A subseleção de vetores para categoria de ação, como explicado no capítulo anterior, possui apenas aqueles cujos termos estão no dicionário técnico relacionados à categoria. Isto representou um total de 3687 observações distintas para o treinamento, com 691 termos. Um total de 21 rótulos apareceram nestas observações e foram utilizados para o treinamento.

A tabela 4.1 apresenta uma nomenclatura para as arquiteturas. A coluna "Comitê" define a estrutura para uma determinada arquitetura (método de votação). A coluna especialistas representa as duas abordagens implementadas para indução: SVM Linear e *Pattern Search*, que pode usar *Kernel* RBF. A coluna "Est. Erro" define qual estimativa de erro foi utilizada como função objetivo do algoritmo de busca. "LOO" representa *Leave-One-Out*, que foi estimado a partir do número de vetores de suporte. "HO" representa "Hold-Out" e neste caso 20% do conjunto de treinamento foi usado para calcular uma estimativa do erro.

A tabela 4.2 apresenta o resultado, por rótulo, do treinamento para ações. Em negrito está o melhor resultado para cada um dos rótulos. Para categoria de ações, a arquitetura **AA2** foi a que apresentou o maior número de rótulos com maior média de acerto, totalizando 14. A tabela 4.3 exibe o resultado estimado da taxa de acertos da máquina de comitê, baseado na validação cruzada com dez partições. O resultado das arquiteturas **AA1**, **AA2** e **AA3L** são comparáveis. A não-linearidade inserida pelo *kernel* RBF não aprimorou significativamente o resultado, a troco de um maior

Código	Comitê	Especialistas	Est. Erro	Duração (s)
AA1	*One-against-all*	SVM Linear	-	96
AA2	*One-against-one*	SVM Linear	-	36
AA3L	*One-against-all*	*Pattern Search*	LOO	1780
AA3H	*One-against-all*	*Pattern Search*	HO	1800
AA4L	*One-against-one*	*Pattern Search*	LOO	840
AA4H	*One-against-one*	*Pattern Search*	HO	900

Tabela 4.1: Arquiteturas implementadas para categoria de ações.

custo computacional (ver as durações na Tabela 4.1).

Também baseado no resultado apresentado na Tabela 4.2, é possível imaginar que, combinando o resultado das arquiteturas **AA1** e **AA3L** (ambas utilizam o *one-against-all*), o resultado da máquina de comitê pode ser aprimorado. A tabela 4.4 apresenta o resultado das duas arquiteturas combinado: os especialistas que obtiveram maior taxa de acerto para uma das arquiteturas, foi treinado utilizando os parâmetros da melhor. É possível perceber que o resultado da máquina de comitê não melhorou com esta alteração.

A máquina de comitê suportará o analista, ao categorizar corretamente as ordens de serviço nos rótulos. Procurar melhorias ou alterações em algoritmo que façam com que a taxa de acerto alcance 100%, contudo, não é necessariamente a única forma de suportar o analista. Sem a máquina de comitê, o trabalho de avaliação das ordens de serviço envolve ler uma a uma. Portanto, é possível auxiliá-lo, garantindo 100% de acerto para um grupo grande de ordens de serviço, mesmo que a troco de não classificar automaticamente um certo número de ordens.

Ao variar a sensibilidade de cada uma das SVM, é possível criar uma *região de incerteza*, na qual a ordem de serviço precisará de avaliação do especialista. Esta sensibilidade é calculada em uma tolerância mínima para o valor do discriminante gerado para cada uma das ordens de serviço. Nas implementações que usam o *one-against-all*, a sensibilidade envolve considerar como incerta, qualquer ordem de serviço cuja saída da função discriminante seja menor que um valor de sensibilidade $g(x) < |\epsilon|$. Para avaliar o resultado da sensibilidade, devem ser avaliadas em conjunto as figuras 4.1 e 4.2. A primeira figura apresenta o histograma da saída do discriminante para o rótulo vencedor. É possível perceber que algumas ordens de serviço obtiveram como máximo um valor menor do que zero. Estes valores já são considerados como erros, no cálculo da taxa de acerto. A segunda figura exibe o comportamento da taxa de acerto e da quantidade de ordens de serviço categorizadas como rótulo incerto, contra sensibilidade. Para o valor $\epsilon = 0,1$, por exemplo, a taxa de acerto é igual a $99,67\%$. Isso corresponde a 7% das ordens de serviço, categorizadas como incertas. É interessante observar também o resultado da tabela 4.5. Treze dos vinte e um rótulos apresentam taxa de acerto igual a 100%.

Rótulo	#	**AA1%**	**AA2%**	**AA3L%**	**AA3H%**	**AA4L%**	**AA4H%**
Ação01	3	**66,7**	**66,7**	33,3	33,3	33,3	33,3
Ação02	55	94,5	**96,4**	90,9	85,5	83,6	89,1
Ação03	724	98,5	98,1	99,2	98,9	**99,4**	99,2
Ação04	27	92,6	**100,0**	92,6	85,2	88,9	88,9
Ação05	14	**100,0**	**100,0**	**100,0**	78,6	57,1	57,1
Ação06	148	94,6	95,9	96,6	93,9	89,9	**97,3**
Ação07	148	96,6	96,6	**97,3**	96,6	93,2	96,6
Ação08	4	**50,0**	**50,0**	**50,0**	**50,0**	25,0	25,0
Ação09	19	**94,7**	89,5	78,9	84,2	73,7	73,7
Ação10	9	**100,0**	**100,0**	**100,0**	88,9	88,9	77,8
Ação11	293	**99,0**	**99,0**	**99,0**	98,6	91,8	98,6
Ação12	186	96,2	96,2	96,8	94,6	91,4	**97,3**
Ação13	17	88,2	**94,1**	82,4	88,2	88,2	88,2
Ação14	257	**97,3**	94,2	96,5	96,1	91,4	96,5
Ação15	13	**100,0**	**100,0**	**100,0**	92,3	92,3	92,3
Ação16	1432	99,6	99,2	99,8	99,6	**100,0**	99,9
Ação17	4	75,0	**100,0**	75,0	75,0	**100,0**	**100,0**
Ação18	195	99,5	**100,0**	99,0	98,5	91,8	96,9
Ação19	40	95,0	**97,5**	85,0	85,0	82,5	87,5
Ação20	74	**100,0**	**100,0**	**100,0**	**100,0**	93,2	93,2
Ação21	25	96,0	**100,0**	88,0	96,0	92,0	92,0
Média		92,1	94,0	88,6	86,6	83,2	84,8
Média Ponderada		98,3	98,1	98,2	97,6	95,9	97,8

Tabela 4.2: Comparação do resultado da taxa de acerto percentual por rótulo, para cada uma das arquiteturas implementadas para categoria de ações. Melhor resultado em negrito.

ID	**AA1%**	**AA2%**	**AA3L%**	**AA3H%**	**AA4L%**	**AA4H%**
1	98,6	98,6	98,0	97,7	96,6	98,3
2	98,4	99,2	98,7	97,9	97,6	98,7
3	97,0	97,7	97,7	97,7	95,4	97,2
4	98,0	98,6	98,6	97,8	96,1	98,0
5	98,0	97,1	97,7	97,4	94,5	97,4
6	98,1	98,1	97,6	97,1	95,8	98,1
7	98,2	98,2	97,9	97,1	94,8	96,9
8	99,4	97,8	98,0	97,8	95,0	97,2
9	98,4	98,9	98,9	97,9	96,5	98,4
10	98,9	96,7	98,9	97,5	96,1	97,5
Média	98,3	98,1	98,2	97,6	95,8	97,8
Desvio	0,6	0,8	0,5	0,3	0,9	0,6

Tabela 4.3: Comparação do resultado da taxa de acerto por partição, para cada uma das arquiteturas implementadas, para categoria de ações.

ID	**AA1+AA3L**%
1	96,9
2	98,2
3	97,0
4	98,6
5	97,1
6	97,1
7	97,4
8	98,6
9	97,9
10	95,6
Média	97,4

Tabela 4.4: Comparação do resultado da taxa de acerto por partição, para a uma arquitetura mista, combinando **AA1** e **AA3L**.

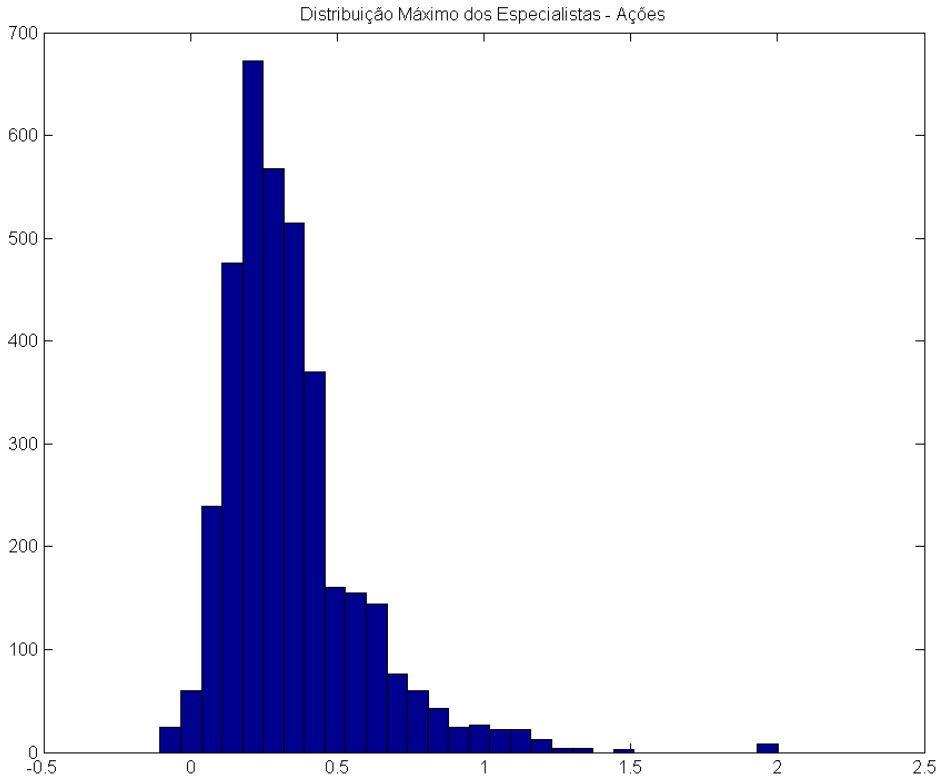

Figura 4.1: Histograma da saída do discriminante, arquitetura **AA1**, para as ordens de serviço da base de dados.

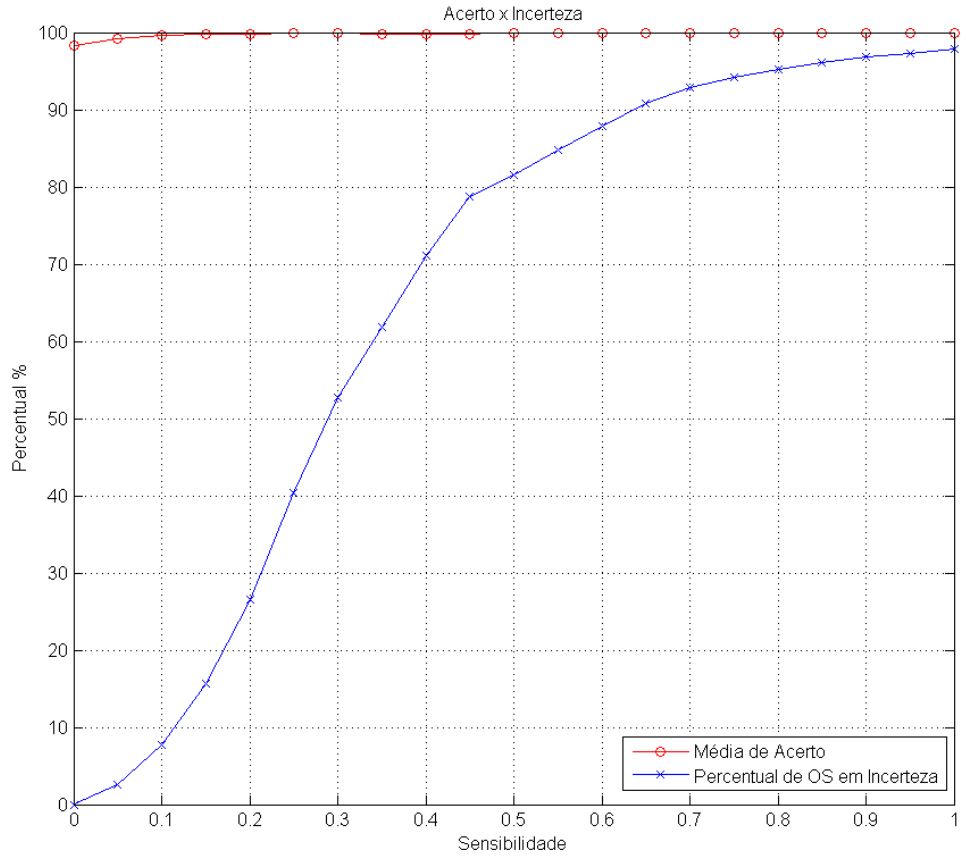

Figura 4.2: Taxa de acerto e quantidade de ordens de serviço categorizadas como "categoria incerta" contra a sensibilidade ϵ, categoria de ações.

Rótulo	**Acerto%** $\epsilon = 0,1$
Ação01	100,0
Ação02	98,0
Ação03	99,8
Ação04	96,0
Ação05	100,0
Ação06	100,0
Ação07	99,2
Ação08	100,0
Ação09	100,0
Ação10	100,0
Ação11	100,0
Ação12	99,4
Ação13	100,0
Ação14	98,7
Ação15	100,0
Ação16	99,9
Ação17	100,0
Ação18	99,5
Ação19	100,0
Ação20	100,0
Ação21	100,0

Tabela 4.5: Acerto por especialista, com $\epsilon = 0,1$.

Isso permitiria também ao analista focar a validação, além das incertas, em apenas algumas categorias. Uma discussão sobre estas estratégias para beneficiar o trabalho do analista será vista na subseção 4.2.1.

4.2 Máquina de Comitê para Sintomas

A subseleção de vetores para categoria de sintomas também possui apenas aqueles cujos termos estão no dicionário técnico relacionados à categoria. Isto representou um total de 2004 observações distintas para o treinamento, com 844 termos. Um total de 43 rótulos apareceram nestas observações e foram utilizados para o treinamento.

No caso desta categoria, a implementação **SA1** foi a que obteve a maior média de acerto por rótulo, conforme exibido na tabela 4.7. O resultado por partição se encontra na tabela 4.8. Similar ao realizado para categoria de ações, combinar o resultado das arquiteturas **SA1** e **SA3** não melhorou o resultado geral da máquina de comitê.

As figuras 4.3 e 4.4 exemplificam a avaliação da sensibilidade. No caso desta categoria, para $\epsilon = 0,1$, a incerteza é igual a 10%. O desvio padrão da estimativa

Código	Comitê	Especialistas	Est. Erro	Duração (s)
SA1	*One-against-all*	SVM Linear	-	80
SA2	*One-against-one*	SVM Linear	-	30
SA3L	*One-against-all*	*Pattern Search*	LOO	1484
SA3H	*One-against-all*	*Pattern Search*	HO	1502
SA4L	*One-against-one*	*Pattern Search*	LOO	700
SA4H	*One-against-one*	*Pattern Search*	HO	750

Tabela 4.6: Arquiteturas implementadas para categoria de sintomas.

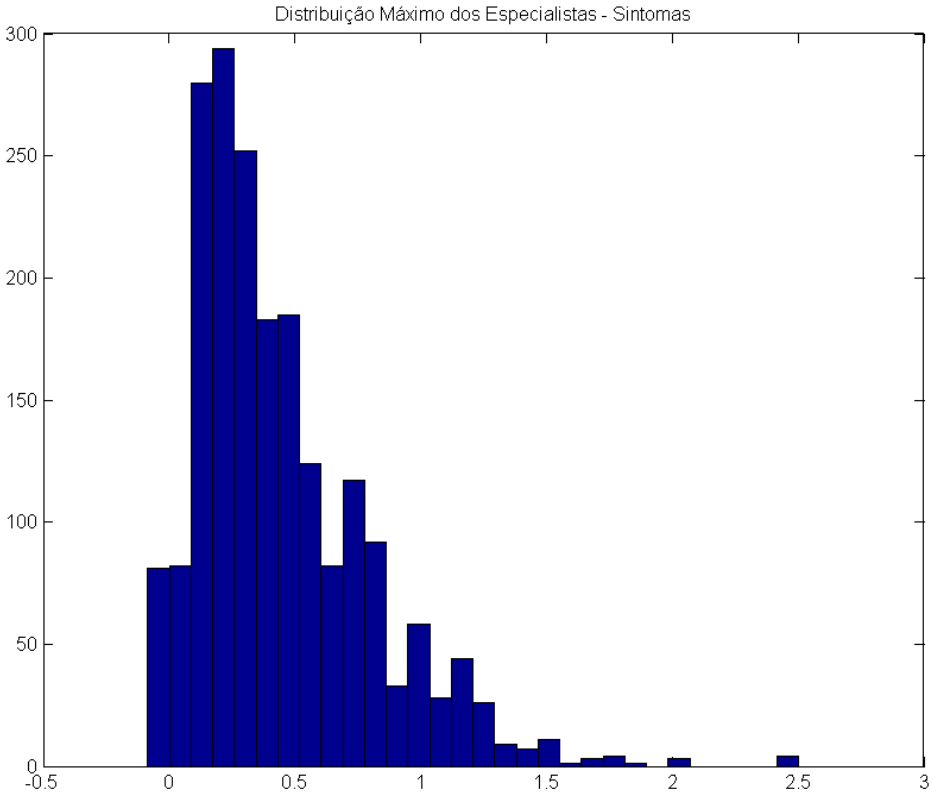

Figura 4.3: Histograma da saída do discriminante, arquitetura **SA1**, para as ordens de serviço da base de dados.

Rótulo	**#**	**SA1%**	**SA2%**	**SA3L%**	**SA3H%**	**SA4L%**	**SA4H%**
Sintoma01	43	**93,0**	**93,0**	93,0	90,7	72,1	86,0
Sintoma02	42	**95,2**	**95,2**	81,0	88,1	88,1	88,1
Sintoma03	101	88,1	**94,1**	77,2	87,1	84,2	89,1
Sintoma04	11	**100,0**	90,9	54,5	90,9	63,6	63,6
Sintoma05	6	**83,3**	50,0	50,0	16,7	50,0	50,0
Sintoma06	10	70,0	**80,0**	40,0	50,0	50,0	50,0
Sintoma07	3	**66,7**	**66,7**	33,3	33,3	33,3	33,3
Sintoma08	1	0,0	0,0	0,0	0,0	0,0	0,0
Sintoma09	1	0,0	0,0	0,0	0,0	0,0	0,0
Sintoma10	95	**97,9**	**97,9**	87,4	92,6	91,6	96,8
Sintoma11	28	82,1	78,6	**85,7**	**85,7**	82,1	78,6
Sintoma12	115	**99,1**	98,3	**99,1**	94,8	93,9	93,9
Sintoma13	21	**90,5**	**90,5**	71,4	76,2	66,7	71,4
Sintoma14	342	99,1	98,5	99,1	98,0	**99,7**	99,4
Sintoma15	18	**83,3**	**83,3**	55,6	72,2	61,1	61,1
Sintoma16	56	**92,9**	89,3	**92,9**	91,1	78,6	89,3
Sintoma17	15	**93,3**	**93,3**	73,3	86,7	73,3	73,3
Sintoma18	28	**82,1**	71,4	**82,1**	78,6	67,9	71,4
Sintoma19	40	**97,5**	**97,5**	90,0	95,0	92,5	92,5
Sintoma20	181	**98,9**	97,8	**98,9**	97,8	98,3	**98,9**
Sintoma21	181	98,3	96,7	**99,4**	96,7	92,3	95,0
Sintoma22	12	**91,7**	**91,7**	75,0	66,7	75,0	75,0
Sintoma23	84	94,0	92,9	**98,8**	96,4	95,2	97,6
Sintoma24	6	**83,3**	**83,3**	66,7	**83,3**	66,7	66,7
Sintoma25	108	86,1	82,4	88,0	**88,9**	83,3	84,3
Sintoma26	65	**95,4**	**95,4**	78,5	93,8	81,5	87,7
Sintoma27	13	**92,3**	84,6	84,6	69,2	76,9	76,9
Sintoma28	1	0,0	0,0	0,0	0,0	0,0	0,0
Sintoma29	7	**100,0**	**100,0**	85,7	85,7	85,7	85,7
Sintoma30	4	75,0	50,0	25,0	25,0	25,0	25,0
Sintoma31	9	**100,0**	**100,0**	88,9	88,9	88,9	88,9
Sintoma32	24	**95,8**	91,7	**95,8**	**95,8**	87,5	87,5
Sintoma33	4	50,0	**75,0**	**75,0**	**75,0**	**75,0**	**75,0**
Sintoma34	51	**92,2**	90,2	78,4	88,2	80,4	86,3
Sintoma35	63	**100,0**	**100,0**	92,1	98,4	93,7	95,2
Sintoma36	73	93,2	**97,3**	93,2	94,5	94,5	95,9
Sintoma37	6	83,3	83,3	83,3	83,3	**100,0**	**100,0**
Sintoma38	26	88,5	80,8	76,9	**92,3**	80,8	80,8
Sintoma39	57	**96,5**	93,0	**96,5**	93,0	89,5	91,2
Sintoma40	3	0,0	0,0	0,0	0,0	0,0	0,0
Sintoma41	36	**86,1**	75,0	83,3	83,3	80,6	83,3
Sintoma42	2	**100,0**	**100,0**	**100,0**	**100,0**	**100,0**	**100,0**
Sintoma43	12	**91,7**	83,3	58,3	83,3	58,3	66,7
Média		81,5	79,4	71,8	74,6	71,1	72,8
Média Pond.		94,5	93,3	90,4	92,0	88,8	90,9

Tabela 4.7: Comparação do resultado da taxa de acerto por rótulo, para cada uma das arquiteturas implementadas para categoria de sintomas. Melhor resultado em negrito.

ID	SA1%	SA2%	SA3L%	SA3H%	SA4L%	SA4H%
1	96,4	92,3	92,8	94,8	90,2	94,3
2	93,0	91,0	86,4	91,0	85,4	87,9
3	93,4	93,9	90,8	92,9	89,8	92,3
4	98,4	96,7	93,4	94,0	90,1	92,3
5	92,3	90,8	88,2	88,7	86,7	87,2
6	95,6	94,6	89,2	93,6	88,7	93,1
7	94,0	92,7	89,9	90,4	86,2	89,0
8	96,2	94,5	92,3	92,3	91,8	93,4
9	93,5	94,4	93,5	91,6	92,6	93,0
10	92,7	92,2	87,7	90,9	86,8	87,2
Média	94,5	93,3	90,4	92,0	88,8	91,0
Desvio	1,9	1,8	2,4	1,8	2,3	2,7

Tabela 4.8: Comparação do resultado da taxa de acerto por partição, para cada uma das arquiteturas implementadas para categoria de sintomas.

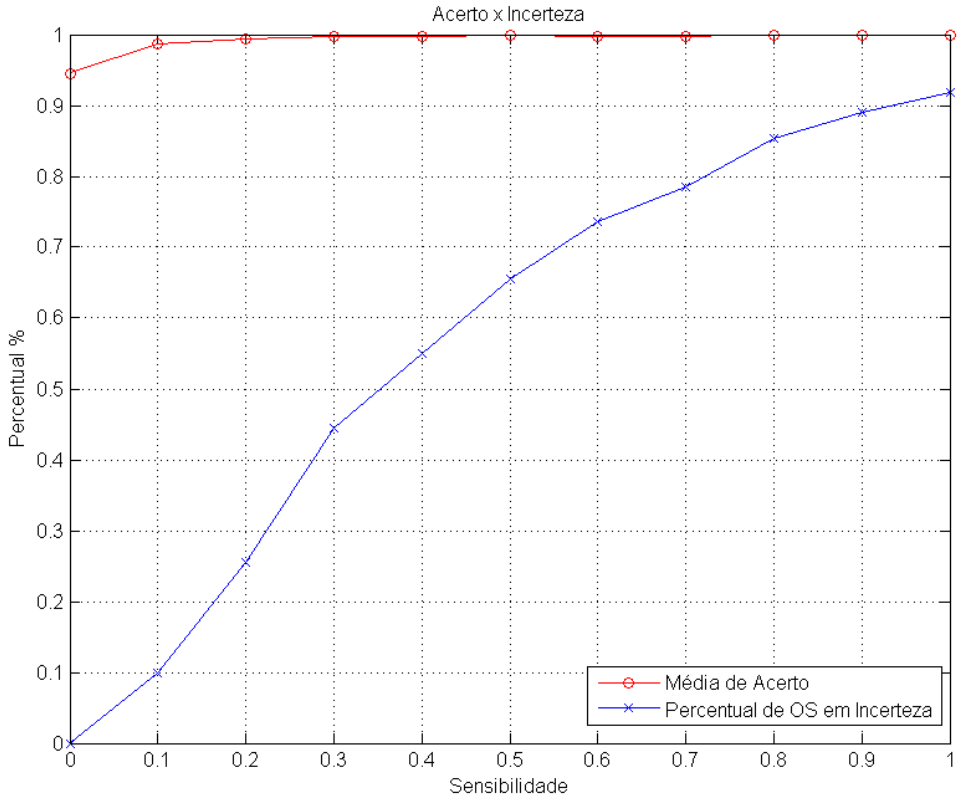

Figura 4.4: Taxa de acerto e quantidade de ordens de serviço categorizadas como "categoria incerta" contra a sensibilidade ϵ, categoria de sintomas.

de média da taxa de acerto, para as arquiteturas implementadas para categoria de componente, é maior do que o que foi visto para ações. Comportamento similar, em relação ao uso do *kernel* RBF, também foi averiguado. A inserção da não-linearidade não melhorou a taxa de acerto, vinculado a um aumento grande da complexidade computacional. Desta forma, as topologias mais simples, novamente, obtiveram melhor resultado. A próxima subseção discute as estratégias para a utilização das máquinas de comitê.

4.2.1 Discussão

A seleção da arquitetura adequada para as máquinas de comitê de ambas as categorias é baseada na troca entre confiabilidade da classificação e tempo alocado de especialista para avaliação dos resultados. Como não foi encontrado na literatura nenhuma metodologia de classificação de ordens de serviço desta natureza, a análise de desempenho fica restrita as implementações realizadas neste trabalho.

A quantidade de ordens de serviço inseridas no sistema, por mês, gira em torno de 75 por equipamento. Uma mina intermediária possui cerca de 40 equipamentos. Um analista experiente consegue realizar a categorização de uma ordem de serviço num intervalo de trinta a sessenta segundos, dependendo da complexidade da ordem de serviço. Logo, o trabalho total por mês gira em torno de 25 a 50 horas. Como o analista realiza a classificação dos três campos, assumindo que seja gasto o mesmo tempo por campo e arredondando o resultado, há um trabalho de 8 a 16 horas, por categoria, por mês. Ainda que experiente, ao realizar a classificação manual de diversas ordens, o analista está sujeito ao cansaço. Portanto, reduzir a quantidade de ordens para as quais é necessária a avaliação manual, já é um trabalho benéfico.

Algumas estratégias poderiam ser selecionadas para utilizar os resultados obtidos pelas arquiteturas. Estas estratégias estão resumidas na listagem abaixo:

1. **Confiabilidade Máxima** - Utilizando a arquitetura **AA1** para ações e **SA1** para sintomas, com o fator de sensibilidade, é possível obter 100% de acerto. No caso da categoria de ações, este valor está relacionado a 50% de incerteza, enquanto para sintomas, o valor está relacionado a 85% de incerteza. Isto representa uma economia que gira em torno de 5 a 10 horas/mês de um analista.

2. **Tolerância Parcial a Erros de Classificação** - Caso fosse possível tolerar uma quantidade de erros específica, seria possível selecionar outra faixa para a sensibilidade. Por exemplo, selecionando $\epsilon = 0,1$, a taxa de acertos é igual a $99,67\%$ para ações e igual a $98,62\%$ para sintomas, associado a 7% e 10% de incerteza, respectivamente. No pior caso, se os erros acontecessem em ordens

de serviço mutuamente exclusivas para as duas categorias, o total seria de aproximadamente 50 ordens de serviço classificadas erroneamente. Este erro estaria associado, também, a no máximo duas horas de validação sobre as ordens categorizadas como incertas pela máquina de comitê.

3. **Tolerância Total a Erros de Classificação** - Neste caso, bastaria escolhe a arquitetura com maior taxa de acerto, utilizando, ou não, algum critério para avaliar a dispersão. Levando em consideração apenas a média de acerto, as arquiteturas vencedoras seriam **AA1** e **SA1**.

4. **Tolerância Parcial por Especialista** - Para esta estratégia, seria necessário primeiro assumir a premissa de que a proporção entre os rótulos se mantivesse com o tempo. Os dados deste exemplo estão na Tabela 4.9. Para $\epsilon = 0,1$, utilizando a arquitetura **AA1**, as taxas de acerto estão ilustradas na tabela. Sendo tolerável um erro de $0,05\%$ por especialista, apenas 7 das 21 categorias, precisariam ser avaliadas. Caso as proporções se mantivessem, isso representaria 23% do tempo, mais o tempo gasto para avaliar as categorizadas como incertas (que no caso desta arquitetura é igual a 7%), totalizando uma economia de 70% de horas/mês do analista.

Neste capítulo foram discutidos os resultados relacionados as arquiteturas de máquina de comitê, construídas para classificar as categorias de ação e sintoma. Estratégias de utilização das máquinas foram propostas, de forma a tratar de forma concomitante, a confiabilidade do resultado da máquina e a redução do trabalho operacional do analista em horas/mês. Ficaria a critério do analista ou da gestão, definir qual das estratégias é mais adequada para realizar o processo de classificação das ordens de serviço.

Rótulo	**#**	**Proporção**	**Acerto**	**Prop. Risco**
Ação01	2	0,06	100,00	0,00
Ação02	51	1,50	98,04	1,50
Ação03	651	19,14	99,85	0,00
Ação04	25	0,74	96,00	0,74
Ação05	14	0,41	100,00	0,00
Ação06	128	3,76	100,00	0,00
Ação07	126	3,70	99,21	3,70
Ação08	2	0,06	100,00	0,00
Ação09	16	0,47	100,00	0,00
Ação10	9	0,26	100,00	0,00
Ação11	284	8,35	100,00	0,00
Ação12	169	4,97	99,41	4,97
Ação13	15	0,44	100,00	0,00
Ação14	231	6,79	98,70	6,79
Ação15	13	0,38	100,00	0,00
Ação16	1339	39,37	99,85	0,00
Ação17	3	0,09	100,00	0,00
Ação18	193	5,67	99,48	5,67
Ação19	34	1,00	100,00	0,00
Ação20	73	2,15	100,00	0,00
Ação21	23	0,68	100,00	0,00
			Risco	23,38

Tabela 4.9: A coluna proporção representa o percentual do rótulo em relação ao tamanho da amostra. A coluna acerto apresenta a taxa de acertos para o rótulo, utilizando a arquitetura **AA1** e $\epsilon = 0,1$. A coluna Prop. Risco é igual a zero, caso a taxa de acertos seja maior que $99,5\%$. A quantidade de tempo em risco, caso as proporções se mantenham, é igual a $23,38\%$.

Capítulo 5

Conclusões

O primeiro capítulo apresentou o objetivo do trabalho: extrair, dos campos estruturados e desestruturados das ordens de serviço, informações necessárias para realizar a classificação das mesmas. Esta classificação, dividida em três categorias principais: qual ação foi realizada em que componente e qual o sintoma associado. O segundo capítulo discutiu as técnicas utilizadas para implementar a metodologia. Técnicas de mineração de texto utilizadas para extrair, limpar e normalizar as palavras encontradas. Descrição de como funciona o treinamento das máquinas de vetor suporte, para o caso separável e não-separável e suas possíveis extensões multiclasse. Foi também discutido o uso de *kernels* para tornar o separador não-linear e foi apresentada uma heurística de otimização para encontrar o melhor parâmetro para o *kernel* RBF.

O capítulo de metodologia explicou como as técnicas de mineração de texto foram utilizadas para extrair as características das ordens de serviço, como o dicionário técnico foi usado para auxiliar no processo de normalização e como as bases de treinamento, por categoria, foram geradas. Explicou como as máquinas de comitê foram concebidas, através das diferentes arquiteturas implementadas. O capítulo anterior apresentou os resultados obtidos para os treinamentos e discutiu quatro estratégias de utilização das diferentes arquiteturas, para a redução da quantidade de horas/mês dedicadas, por um analista, para realizar a classificação manual das ordens de serviço.

É importante reforçar que a ferramenta desenvolvida neste trabalho permite agilizar significativamente a classificação, além de desalocar um recurso experiente, possibilitando que a revisão dos resultados seja realizado por um analistas de menor senioridade, consumindo menos tempo. Isso sem contar que a máquina de comitê não é sucetível a fadiga causada pelo procedimento repetitivo de classificação. Desta forma, os resultados obtidos por este trabalho permitem auxiliar os especialistas em suas atividades rotineiras de categorização dos problemas.

5.1 Trabalhos Futuros

Os próximos passos deste trabalho estão elencados abaixo.

Melhorias na captura da informação:

- Avaliação e acompanhamento contínuo da qualidade de dados, dos campos estruturados e desestruturados.
- Utilização dos campos categóricos para o treinamento dos classificadores.

Melhorias no dicionário técnico:

- Flexibilizar a relação termos $\times$ categorias para considerar palavras que não estejam no dicionário técnico e que possam contribuir com mais informação para o processo de classificação.
- Detecção automática da relação entre termos e categorias.
- Consideração do sequenciamento dos termos no problema de classificação.

Melhorias nos classificadores:

- Treinamento do classificador para a categoria de componentes, que possui cerca de 1800 rótulos.
- Treinamento de um classificador ou máquina de comitê que realize a classificação A, C, S, tupla das três categorias, de forma simultânea.
- Aprimoramento do treinamento das máquinas de comitê através de melhoria nos especialistas ou de melhorias na combinação dos resultados individuais, utilizando técnicas como *boosting*, *bagging*, ou implementando novas abordagens multiclasse.
- Categorização de mais ordens de serviço para expandir a quantidade de rótulos do dicionário técnico presentes nos dados e para enriquecer as distribuições já existentes.

Referências Bibliográficas

[1] ORENGO, V. M., BURIOL, L. S., COELHO, A. R. "A Study on the Use of Stemming for Monolingual Ad-Hoc Portuguese Information Retrieval", *Conference ad Labs of the Evaluation Forum*, pp. 91–98, 2007.

[2] COELHO, A. R. "Stemming para a língua portuguesa: estudo, análise e melhoria do algoritmo RSLP". Trabalho de Graduação, 2007.

[3] RODRIGUES, F. A. A., MACULAN, N. "Modelo de Análise de Risco de Crédito utilizando Máquina de Vetor Suporte". Dissertação de Mestrado, 2012.

[4] PLATT, J. C., CRISTIANINI, N., SHAWE-TAYLOR, J. "Large margin dags for multiclass classification". In: *Advances in Neural Information Processing Systems 12*, pp. 547–553, 2000.

[5] KEERTHI, S. S., LIN, C. J. "Asymptotic behaviors of support vector machines with Gaussian kernel", *Neural Computation*, v. 15, pp. 1667–1689, 7 2003.

[6] MANNING, D. C., RAGHAVAN, P., SCHUTZE, H. *Introduction to Information Retrieval*. 1 ed. London, Cambrige, 2010.

[7] PORTER, M. F. "An Algorithm for Suffix Stripping", *Readings in information retrieval*, pp. 313–316, 1997.

[8] LEVENSHTEIN, I. V. "Binary Codes Capable of Correcting Deletions, Insertions and Reversals", *Soviet Physics Doklady*, v. 10, n. 8, pp. 707–710, 1966.

[9] WAGNER, R. A., FISHER, M. J. "The String-to-String Correction Problem", *Journal of the Association for Computing Machinery*, v. 21, n. 1, pp. 168–173, jan. 1974.

[10] FELDMAN, R., SANGER, J. *Text Mining Handbook*. 1 ed. London, Cambrige University Press, 2007.

[11] EYHERAMENDY, S., LEWIS, D., MADIGAN, D. "On the Naive Bayes Model for Text Categorization", *Journal of Elasticity*, v. 1, n. 1, pp. 1, jan. 2000.

[12] MCCALLUM, A., NIGAM, K. "A comparison of event models for Naive Bayes text classification". In: *IN AAAI-98 WORKSHOP ON LEARNING FOR TEXT CATEGORIZATION*, pp. 41–48. AAAI Press, 1998.

[13] LEWIS, D. D. "Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval". pp. 4–15. Springer Verlag, 1998.

[14] JAAKKOLA, T., RENNIE, J. D. M., RENNIE, J. D. M. "Improving Multi-class Text Classification with Naive Bayes". 2001.

[15] LEE, C. "A Semi-Naive Bayesian Learning Method for Utilizing Unlabeled Data", 2001.

[16] CONG, G., LEE, W. S., WU, H., et al. "Semi-supervised Text Classification Using Partitioned EM". In: *11 th Int. Conference on Database Systems for Advanced Applications (DASFAA*, p. 482493, 2004.

[17] CONG, G., LEE, W., WU, H. "Semi-Supervised Text Classification Using Partitioned EM", 2000.

[18] GENKIN, A., LEWIS, D., MADIGAN, D. "Large-Scale Bayesian Logistic Regression for Text Categorization", *Technometrics*, pp. 291–304, 2007.

[19] RUIZ, M., SRINVASAN, P. "Hierarchical Text Categorization Using Neural Networks", *Information Retrieval*, v. 5, n. 1, pp. 87–118.

[20] JOACHIMS, T. "A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization", pp. 143–151, 1997.

[21] JOACHIMS, T. "Text Categorization with Support Vector Machines: Learning with many Relevant Features", *Lecture Notes in Computer Science*, v. 1398, pp. 137–142, 1998.

[22] VAPNIK, V. N. *The nature of statistical learning theory*. New York, Springer-Verlag, 1995.

[23] THEODORIDIS, S., KOUTROUMBAS, K. *Pattern Recognition*. 4 ed. 30 Corporate Drive, Suite 400, Burlington, MA 01803, USA, Elsevier, 2000.

[24] DUDA, R. O., HART, P. E. *Pattern Classification*. 2 ed. Corporate Headquarters, 111 River Street, John Wiley and Sons Inc, 2000.

[25] BAZARAA, M. S., SHETTY, C. M. *Nonlinear Programming*. Unknown, John Wiley and Sons, 1979.

[26] PLATT, J. C. "Advances in kernel methods". MIT Press, cap. Fast training of support vector machines using sequential minimal optimization, pp. 185–208, Cambridge, MA, USA, 1999. ISBN: 0-262-19416-3. Disponível em: <http://dl.acm.org/citation.cfm?id=299094.299105>.

[27] LIU, Y., ZHENG, Y. F. "One-Against-All Multi-Class SVM Classification Using Reliability Measure", .

[28] DIETTERICH, T. G., BAKIRI, G. "Solving multiclass learning problems via error-correcting output codes", *Journal of Artificial Intelligence Research*, v. 2, pp. 263–286, 1995.

[29] CRAMMER, K., SINGER, Y. *On the algorithmic implementation of multiclass kernel-based vector machines*. Relatório técnico, School of Computer Science and Engineering, 2001.

[30] CHAPELLE, O., CRISTIANINI, N. "Choosing multiple parameters for support vector machines". In: *Machine Learning*, p. 2002, 2002.

[31] AUDET, C., JUNIOR, J. E. D. "Analysis of Generalized Pattern Searches", *Siam Journal on Optimization*, 2003.

[32] MASCIA, A., RECK, W., GARCIA, V. "Algoritmo heurístico para agrupamento de ordens de serviço em concessionárias de distribuição de energia elétrica considerando priorização", 2000.

[33] GARCIA, V., BERNARDON, D., SPERANDIO, M. "Gestão Estratégica das Ordens de Serviço: uma abordagem para despacho centralizado". Congresso de Energia Elétrica, Cidel, 2010.