Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo Experimental de Aprendizado de Máquina para Desenvolvimento de um Classificador de Texto de Incidentes de Grandes Eventos

André Accioly Lima
Renato Carlos Pinto

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientadora
Prof.ª Dr.ª Célia Ghedini Ralha

Brasília
2016

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Rodrigo Bonifácio de Almeida

Banca examinadora composta por:

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora) — CIC/UnB
Prof. Dr. Edison Ishikawa — CIC/UnB
Prof. Dr. Jan Mendonça Corrêa — CIC/UnB

CIP — Catalogação Internacional na Publicação

Lima, André Accioly.

Estudo Experimental de Aprendizado de Máquina para Desenvolvimento de um Classificador de Texto de Incidentes de Grandes Eventos / André Accioly Lima, Renato Carlos Pinto. Brasília : UnB, 2016.
76 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2016.

1. inteligência artificial, 2. mineração de dados, 3. processamento de texto, 4. comando e controle, 5. grandes eventos

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo Experimental de Aprendizado de Máquina para Desenvolvimento de um Classificador de Texto de Incidentes de Grandes Eventos

André Accioly Lima
Renato Carlos Pinto

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora)
CIC/UnB

Prof. Dr. Edison Ishikawa
CIC/UnB

Prof. Dr. Jan Mendonça Corrêa
CIC/UnB

Prof. Dr. Rodrigo Bonifácio de Almeida
Coordenador do Bacharelado em Ciência da Computação

Brasília, 29 de Novembro de 2016

Dedicatória

Dedico este trabalho à minha família, pelo apoio incondicional em todas as etapas da minha vida e por sempre me incentivarem em meus estudos.

- Renato

Dedico este trabalho aos meus amigos, professores e familiares que sempre me deram suporte, principalmente a meus pais e irmã. Dedico também às pessoas que futuramente leiam este trabalho para obter algum tipo de conhecimento.

- André

Agradecimentos

Agradeço aos meus pais e ao meu irmão pelo suporte e apoio ao longo de toda minha graduação, e também aos inúmeros amigos que tornaram mais agradável estes tempos de faculdade. Em especial, agradeço ao André, pelo trabalho conjunto neste projeto de conclusão de curso.

Agradeço à professora Célia Ghedini que nos guiou neste trabalho de graduação do começo ao fim, nos incentivando e nos dando confiança para seguir em frente. Agradeço também ao Leonardo Henrique Moreira, Oficial do Exército Brasileiro, que tornou este trabalho possível.

Agradeço aos professores Edison Ishikawa e Jan Mendonça Correa por participarem de nossa banca avaliadora.

Por fim, agradeço ao Departamento de Ciência da Computação da UnB, que conta com professores excelentes, tanto tecnicamente quanto como pessoas. Em particular, agradeço à professora Alba Melo pelo apoio nesta etapa final de minha graduação.

- Renato

Agradeço aos meus pais, amigos e minha irmã pelo apoio durante essa jornada. Agradeço também à professora Célia Ghedini, por nos orientar e nos ensinar durante este ano, e ao Leonardo Henrique Moreira, Oficial do Exército Brasileiro, por nos dar dicas e ideias, além de nos ajudar com sua experiência e fornecer material para o projeto.

Agradeço aos professores que tive na UnB e ao Departamento de Ciência da Computação, pois eles foram responsáveis por formar meu conhecimento para que fosse possível realizar este trabalho, assim como meus colegas de graduação que sempre me ajudaram com os estudos.

Agradeço ao companheiro Renato, que fez este trabalho comigo, por ter sempre se esforçado e me ajudado quando não compreendia algo.

- André

Resumo

Em sistemas de apoio ao gerenciamento de incidentes, em especial de grandes eventos, é importante que o operador do sistema tenha à sua disposição o máximo possível de informações que o auxiliem no processo de tomada de decisão. Nesse contexto, este trabalho utiliza métodos de aprendizado de máquina para desenvolver um classificador automático de texto de incidentes a partir de incidentes reais da Copa das Confederações, evento realizado no Brasil em 2013, com o objetivo de auxiliar a tomada de decisão dos operadores que utilizam o sistema Pacificador do Exército Brasileiro. Foram estudados alguns métodos de aprendizado de máquina, juntamente com suas configurações, com a finalidade de criar um modelo de classificação. O *Naive Bayes*, a Máquina de Vetor de Suporte (SVM) e a Árvore de Decisão foram aplicados. Esse modelo é então disponibilizado por meio de uma interface Java, que reúne também informações estatísticas sobre cada categoria de incidentes.

Palavras-chave: inteligência artificial, mineração de dados, processamento de texto, comando e controle, grandes eventos

Abstract

Considering incident management systems, in particular major events incidents, it is important that the system operator has at his disposal the maximum possible information to assist the decision process. In this context, this project applies machine learning methods to develop an automatic incident text classifier. The text set used is from the last Confederations Cup incidents, event held in Brazil in 2013, in order to assist the operator of the Brazilian Army's *Pacificador* system. Some machine learning methods, together with their settings, were investigated in order to create a classifier model. The Naive Bayes, the Support Vector Machine (SVM) and the Decision Tree were applied. This model is available through a Java interface, which also includes statistical information on each category of incidents.

Keywords: artificial intelligence, data mining, text processing, command and control, major events

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Recentemente, o Brasil assumiu o compromisso de sediar diversos grandes eventos, o que ocasionou uma elevada frequência dos mesmos no país. Esses eventos, como a Copa das Confederações de 2013 e a Copa do Mundo de 2014, exigem uma extraordinária capacidade operacional e de planejamento, pois reúnem diversos países e envolvem uma grande quantidade de pessoas, além de diversas localizações em que os eventos são realizados. É, sem dúvida, uma grande responsabilidade para o anfitrião desses acontecimentos.

Nesse sentido, um dos aspectos chave em grandes eventos é a segurança dos envolvidos. Esse é um esforço que vai muito além do tradicional emprego de forças de segurança em campo. Nesse cenário, a análise de informações estratégicas e a consciência situacional do ambiente assumem posição de destaque, ao possibilitar a utilização eficiente de recursos materiais e humanos a partir da investigação e interpretação dos dados de eventos passados. O processo de análise dessas informações e gerenciamento desses recursos também envolve a atividade especializada de Comando e Controle (C2) [1], que trata do funcionamento de uma cadeia de comando envolvendo: (1) a figura de uma autoridade que recebe informações constantes para a tomada de decisão; (2) um processo decisório que estabelece esse fluxo de informações; e (3) a estrutura física e de pessoal necessária para essa atividade. Com essa atividade especializada, o comandante de uma operação tem melhores condições de planejar, controlar e coordenar seus recursos com o objetivo de cumprir uma missão.

No Brasil, o Exército Brasileiro emprega essa atividade especializada de C2, tendo um importante papel no gerenciamento de grandes eventos. Utilizando Sistemas de Tecnologia da Informação e Comunicações (TIC), como o sistema Pacificador [2], as informações são coletadas, monitoradas, armazenadas, processadas, fundidas, disseminadas, apresentadas e protegidas [1].

Dada a magnitude do contexto de grandes eventos no escopo de segurança pública, o emprego de técnicas de Inteligência Artificial (IA) para o tratamento de dados, informação e conhecimento no domínio de incidentes em grandes eventos é de clara relevância. Nesse sentido, este trabalho aplica técnicas de IA para criação de um classificador automático de incidentes para o sistema Pacificador a partir da análise de incidentes reais da Copa das Confederações de 2013.

1.1 Problema

No gerenciamento de eventos e operações, o Exército Brasileiro utiliza um sistema de C2 chamado Pacificador [2]. Com esse sistema, agentes em campo podem entrar com informações em tempo real sobre incidentes, incluindo imagens, localização e descrição textual. Essas informações aparecem em um mapa interativo. Com esse mapa e com essas informações, aumenta-se a consciência situacional sobre o evento gerenciado.

Atualmente, porém, não há uma categorização padrão para os incidentes reportados, dificultando o tratamento sistematizado dos mesmos, o que influi nas análises estatísticas geradas para os eventos. Claramente, este cenário incipente em sistematização pode trazer problemas para o processo de tomada de decisão gerencial em vários níveis organizacionais.

Nesse sentido, a questão básica de pesquisa que este trabalho aborda foi definida como: é possível a criação de um módulo de suporte à decisão para o sistema Pacificador, contendo um classificador automático de incidentes, que produza estatísticas das categorias geradas dos dados de grandes eventos ocorridos no país?

1.2 Objetivos

O objetivo geral é o desenvolvimento do módulo de suporte à decisão, o qual irá conter um classificador automático de incidentes com respectivas estatísticas das categorias geradas.

Como objetivos específicos do trabalho cita-se:

1. Estudar métodos de aprendizado de máquina e técnicas de mineração, para aplicação no tratamento do problema de categorização automática de incidentes;
2. Aplicar métodos e técnicas de aprendizado de máquina na base de incidentes do Pacificador e analisar os resultados;
3. Avaliar qual técnica de aprendizado de máquina é mais eficiente para a classificação de textos de incidentes;
4. Desenvolver uma interface para o módulo de categorização automática de incidentes e obtenção de estatísticas de categorias de eventos passados.

1.3 Hipóteses

Como hipóteses desse trabalho, é possível elencar:

1. É possível fazer uma categorização automática de incidentes, de forma a padronizar a base de dados do Pacificador, auxiliando o levantamento de informações estatísticas dessa base de dados;
2. A utilização de métodos de aprendizado de máquina pode auxiliar no processo de categorização dos incidentes de grandes eventos;
3. A categorização automática de incidentes auxilia no processo de padronização da base do Pacificador;

4. A padronização da base de dados do Pacificador auxilia no processo de tomada de decisão pelo operadores do Pacificador;
5. É possível a criação de uma interface que reúna o classificador automático de incidentes e estatísticas associadas à cada categoria de incidentes.

1.4 Metodologia

A metodologia utilizada neste trabalho seguem etapas de desenvolvimento definidas conforme segue:

- Percepção do problema: reuniões de discussão sobre atuação do Exército Brasileiro na realização de grandes eventos realizados no Brasil. Investigação sobre o sistema Pacificador, como é sua base de dados e o procedimento de inserção dos incidentes na base;
- Estudo da teoria relacionada: estudo de métodos de aprendizado de máquina, técnicas e ferramentas de mineração de dados e de processamento de texto. Estudo do modelo processual CRISP-DM para utilizar como base para o desenvolvimento do projeto.
- Modelagem da solução: foi feito o pre-processamento dos dados, foram testados algoritmos de clusterização e algoritmos de aprendizagem supervisionada a fim de verificar a viabilidade e performance desses métodos de aprendizagem para a classificação;
- Avaliação e análise dos resultados: os resultados da aplicação das técnicas de aprendizagem supervisionada foram analisados com abordagem comparativa, onde os que obtiveram porcentagem de acerto maiores foram considerados melhores. Dessa forma, é possível escolher qual a técnica de aprendizagem será utilizada;
- Desenvolvimento tecnológico: foi desenvolvido um módulo de interface gráfica contendo o modelo de predição obtido a partir da análise dos resultados dos experimentos e estatísticas de cada categoria de incidentes.

1.5 Estrutura do Documento

O restante dessa monografia inclui no Capítulo 2 uma apresentação sucinta dos fundamentos teóricos envolvidos no trabalho, incluindo métodos de aprendizado de máquina, técnicas de mineração de dados, processamento automático de texto com ferramentas e conceitos de C2.

No Capítulo 3 está apresentada a proposta da solução do problema, apresentando o fluxo de trabalho seguido para a realização dos experimentos e como foi idealizada a implementação da interface. O Capítulo 4 é reservado para a análise dos experimentos e resultados obtidos.

Finalmente, no Capítulo 5, são apresentadas as conclusões e trabalhos futuros.

Capítulo 2

Fundamentação

Este capítulo apresenta os conceitos teóricos necessários para o entendimento do trabalho desenvolvido. Na Seção 2.1 são apresentados conceitos gerais de aprendizado de máquina e diferentes métodos de aprendizagem supervisionada que serão utilizados nos experimentos. Na Seção 2.2 são apresentados conceitos relacionados à mineração de dados. Por fim, na Seção 2.3, a terminologia do contexto de C2 e do sistema Pacificador, de onde foram retirados os incidentes utilizados nos experimentos, é explicada.

2.1 Aprendizado de Máquina[1]

Quando o projetista de uma aplicação pensa na solução que será utilizada no sistema, é improvável que, por mais tempo que ele invista, ele consiga considerar todos os cenários possíveis de entradas para o problema em questão. Além disso, os cenários podem sofrer inúmeros tipos de mutação, o que torna ainda mais complicado criar uma solução que consiga cobrir todos esses cenários. Assim sendo, as técnicas de aprendizado de máquina (ou aprendizagem automática) se tornam essenciais para a criação de sistemas inteligentes e adaptáveis, pois estes métodos têm como objetivo generalizar uma solução para que seja adequada para o problema e que tenha a habilidade de se adaptar a novas entradas. De forma resumida, pode-se dizer que aprendizado de máquina é a área de estudo que se preocupa em como construir soluções computacionais que automaticamente se aperfeiçoam com suas experiências [4].

Existem diversos tipos de aplicações para aprendizado de máquina. Alguns exemplos dessas aplicações são: detecção de defeitos em software [5], previsão de comportamentos anômalos a partir de uma base de dados [6], categorização de *tweets* em categorias de Segurança Pública [7], enfim, qualquer situação onde seja adequado a criação de uma função de predição para novas entradas.

Os algoritmos de aprendizado de máquina podem seguir diferentes abordagens. Eles dependem da aplicação em questão e do conhecimento a *priori* que se tem do domínio ao qual a aplicação faz parte. Na literatura, encontramos uma divisão tradicional das abordagens em: aprendizagem por reforço, aprendizagem supervisionada, aprendizagem não-supervisionada e aprendizagem semi-supervisionada.

[1] Como fonte básica do estudo de aprendizado de máquina foi utilizado um livro padrão internacional [3].

2.1.1 Aprendizagem por Reforço

Na aprendizagem por reforço, trabalha-se com a ideia de condicionamento (saber se uma ação é correta ou não) a partir de recompensas ou punições. Esses condicionamentos são modelados a partir da análise do problema, onde ao término de uma ação e a partir dos resultados desta, o módulo inteligente da aplicação terá que descobrir quais ações *a priori* causaram a recompensa ou punição. Dessa forma, a partir do *feedback* recebido, a aplicação pode priorizar determinadas ações em detrimento de outras.

Um exemplo que ilustra essa abordagem é o de um restaurante, onde o garçom pode ou não receber uma gorjeta, dependendo de seu serviço. Se serviu bem os clientes, pode receber uma recompensa (no caso a gorjeta), então naturalmente vai tentar tratar os clientes seguintes da mesma forma que tratou aquele que lhe deu a gorjeta.

2.1.2 Aprendizagem Supervisionada

Na abordagem de aprendizagem supervisionada, há um conjunto rotulado de dados de treinamento com uma série de pares entrada-saída. Estes dados serão processados pelo módulo inteligente, cuja finalidade é tentar generalizar este mapeamento de pares e obter uma função que consiga classificar novas entradas.

Uma analogia para esse caso seria um manual de instruções: quando uma situação estranha é encontrada, o manual (que seria a função de generalização) é consultado para avaliar que ações devem ser realizadas naquela situação.

Métodos de Aprendizagem Supervisionada

Algumas técnicas de aprendizagem supervisionada serão apresentadas conforme citadas na literatura.

Árvores de Decisão

Árvore de Decisão é uma das formas mais populares de aprendizagem supervisionada. Nessa técnica, o problema é representado em uma árvore, onde em cada nó os dados de entrada são analisados e divididos, realizando-se um teste de atributo. Dependendo do resultado de cada teste, o caminho dos dados da raiz até as folhas da árvore é formado. Dessa forma, a árvore é percorrida de cima para baixo até encontrar uma folha, que retorna a classificação para a entrada fornecida.

Nesta abordagem, o objetivo é obter, a partir do conjunto de treinamento, a menor árvore consistente possível (boa acurácia, boa taxa de acerto), dessa forma, os testes que ocorrem em cada nó devem ser organizados de forma que dividam os dados da melhor forma possível [8]. No entanto, não existe forma eficiente de procurar por todas as árvores possíveis. Por isso, utiliza-se uma heurística para a obtenção dessa árvore. Essa heurística utiliza a abordagem de dividir para conquistar, em um procedimento que a cada etapa tenta identificar qual atributo melhor divide os dados de treinamento. Um algoritmo conhecido para construção de Árvores de Decisão é o algoritmo C4.5 do autor Ross Quinlan. O algoritmo J48 é implementação Java em código aberto do algoritmo C4.5 e está disponível na ferramenta de mineração WEKA [9].

A Figura 2.1 mostra um exemplo de uma árvore de decisão. O problema explorado nessa árvore é a verificação de um restaurante, para saber se uma pessoa irá jantar nele ou não. Um conjunto de treinamento foi aplicado à técnica, resultando nessa árvore, com as folhas (saídas do algoritmo) podendo ser sim ou não, dependendo dos nós que verificam e dividem o conjunto de atributos inseridos. Para utilização da árvore, uma sequência de atributos são inseridos e testados (cada atributo é testado por um nó). No exemplo em questão, os atributos são: pessoas no restaurante, tempo de espera, necessidade de reserva e assim por diante.

Independente da forma de obtenção da árvore, deve-se sempre tomar cuidado com um dos problemas clássicos em aprendizado de máquina: *overfitting*. Este ocorre quando a hipótese obtida (função de generalização) se adapta demais aos dados de treinamento e dessa forma não consegue classificar corretamente novos dados fornecidos, divergindo, assim, da finalidade de utilização de métodos de aprendizagem. Em particular, para evitar este problema em árvores de aprendizagem, utiliza-se a poda na árvore, cortando exemplos que não são relevantes. Claramente, uma árvore podada também tem a vantagem de ser menor e portanto de mais fácil compreensão.

Para testar a acurácia da árvore obtida, deve-se utilizar um conjunto de teste e, começando com apenas um dado do conjunto, ir aumentando o tamanho do conjunto e testando a acurácia deste. Este gráfico (tamanho do conjunto de teste *versus* proporção de acerto) chama-se *curva de aprendizagem*.

Uma das propriedades mais importantes desta abordagem de aprendizagem supervisionada é o fato de ser mais fácil para um ser humano entender a razão de classificação de uma entrada.

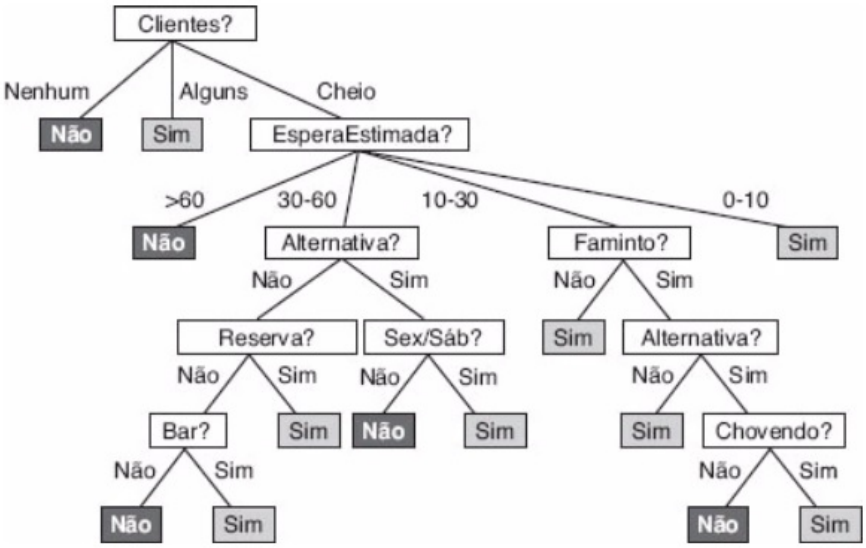

Figura 2.1: Ilustração do método Árvore de Decisão (traduzida de [3]).

Redes Neurais Artificiais

Utilizando um modelo matemático baseado na ideia de neurônios biológicos, uma rede neural artificial (RNA) é ativada quando uma combinação linear de suas entradas atinge

um certo valor determinado (*threshold*).

As redes neurais são compostas de nodos interligados. Um nodo pode ativar outro se este atingir determinado *threshold*. Dessa forma, cada entrada e cada ligação têm um peso associado, e esse peso é utilizado para calcular a combinação linear das entradas de um nodo.

Existem várias formas diferentes de se construir uma RNA. Deve-se decidir, por exemplo, se a rede terá conexões só em uma direção ou se será retro-alimentada com suas próprias saídas.

Uma RNA pode ter uma ou mais camada. Se tiver só uma camada, cada unidade (nodo) conectará as entradas diretamentas às saídas. No caso de multicamadas, existem as camadas escondidas que não são conectadas diretamente às saídas.

Como qualquer método de aprendizagem, é necessário uma forma de aprender, a partir do conjunto de treinamento, uma função de generalização (hipótese). No caso das redes neurais, deve-se raciocinar a partir de duas questões: qual peso cada entrada terá, e qual será a estrutura da rede. Para obtermos os pesos associados à cada entrada, testamos as diferentes possibilidades analisando sob a ótica de uma função de perda. Esta função é definida como o quanto se perde em prever errado uma saída para uma entrada específica. Mais particularmente, para definir os pesos associados às camadas escondidas (que não estão conectadas diretamente às saídas), utiliza-se um algoritmo de *back-propagation* onde erros associados às saídas são propagados de volta para as funções de cálculo de peso das camadas escondidas.

Para determinar a estrutura da rede (quantos nodos, quantas camadas, totalmente conectada ou parcialmente conectada) testa-se diferentes estruturas e observa-se o resultado em termos de uma função de perda para então escolher a melhor estrutura.

Máquina de Vetor de Suporte

Atualmente um dos métodos mais populares em aprendizagem supervisionada é a Máquina de Vetor de Suporte, do inglês *Support Vector Machine* (SVM). Esse método é recomendado para quando não se tem um conhecimento especializado, *a priori*, sobre o domínio da aplicação.

Este método funciona criando um separador que seja o mais distante possível dos exemplos do conjunto de treinamento. Os pontos mais próximos desse separador são chamados vetores de suporte (ver Figura 2.2 onde os vetores de suporte estão envoltos em círculos maiores e o separador máximo está entre as linhas tracejadas).

O SVM é um método não-paramétrico pois precisa guardar exemplos de treinamento para o funcionamento do classificador. Porém, na prática, não precisa armazenar todos os dados de treinamento, como ocorre em outros métodos. O que esta abordagem faz é criar um separador linear entre os dados das diferentes categorias do conjunto de treinamento. Para classificar um novo dado, então, será observado em qual lado do separador este foi alocado.

Há duas abordagens possíveis ao utilizar esse método. Empregando a ideia de "margem rígida", um dado que aparece de um lado do separador é classificado como daquela categoria. Com um classificador de "margem suave", será analisado à que distância do separador o dado ficou, associando uma penalidade para distância maiores, criando uma tolerância maior para conjuntos de dados ruidosos.

Porém, nem todos os conjuntos serão linearmente separáveis. Nestas situações, utiliza-se uma técnica chamada *kernel trick*. Com essa técnica, os dados são mapeados em uma dimensão maior em que seja possível dividi-los linearmente. Essa característica torna o SVM um método poderoso por ser capaz de representar funções de diferentes complexidades.

Treinar uma SVM envolve um problema de otimização quadrática. O algoritmo *Sequential Minimal Optimization* (SMO) treina uma SVM de forma mais rápida dividindo esse problema em problemas menores e utilizando uma abordagem númerica ao invés de uma abordagem analítica [10].

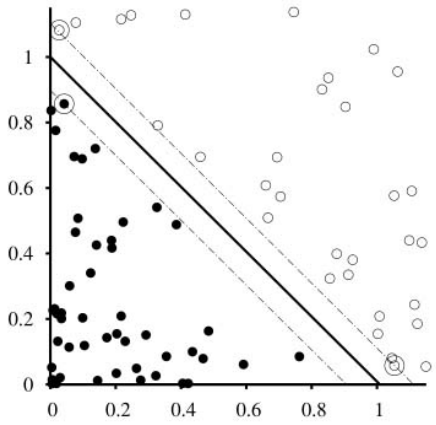

Figura 2.2: Ilustração do método SVM [3].

K-Vizinhos mais Próximos

Um dos métodos mais simples de classificação é o método dos K-Vizinhos mais Próximos. Esta técnica se baseia em, dado um ponto (entrada), encontrar a maioria dos k vizinhos mais próximos (utilizando distância de *Manhattan*, por exemplo) e rotulá-lo na classe destes vizinhos. Logo, é uma técnica não paramétrica, ou seja, sempre utiliza todo o conjunto de treinamento ao realizar a classificação de uma nova entrada.

Apesar de simples e eficiente, deve-se projetar com cautela a solução que utilizará este método para que esta não fique excessivamente lenta. Analisar todos os vizinhos do conjunto de treinamento tornaria a classificação proporcional à quantidade de dados de treinamento. Logo, pode-se pensar em formas diferentes de armanezar os dados de treinamento (*K-Dimensional Trees*, por exemplo, que armazena os valores em uma árvore binária, onde cada nó da árvore está em uma dimensão k e divide o espaço que a árvore representa em dois subespaços). Deve-se proteger a solução contra ruídos também, escolhendo-se metodicamente os dados do conjunto de treinamento [9].

Naive Bayes

O *Naive Bayes* é um método de aprendizagem supervisionada baseado na teoria das probabilidades. Mais particularmente, este método opera com a ideia de independência

de atributos. Ou seja, desconsidera a relação que os atributos têm entre si, e estuda a probabilidade individual de cada um deles.

Para entender esta abordagem supervisionada, é necessário relembrar alguns conceitos da teoria de probabilidade. Na teoria da probabilidade, um espaço amostral é o conjunto dos resultados possíveis de um experimento aleatório. Dentro desse conjunto universo, pode-se definir subconjuntos e chamá-los de eventos. Pode-se, então, associar valores de probabilidade à esses eventos:

$$P(A) = \frac{n(A)}{n},$$

onde $n(A)$ é a quantidade de elementos contidos no subconjunto (evento) A, e n é a quantidade de elementos do espaço amostral.

Definimos, então, a probabilidade de um evento A ocorrer condicionado a algum subconjunto B do espaço amostral:

$$P(A|B) = \frac{n(A \cap B)}{n(B)} = \frac{P(A \cap B)}{P(B)},$$

logo,

$$P(A \cap B) = P(A|B) \times P(B),$$

mas,

$$P(A \cap B) = P(B \cap A) = P(B|A) \times P(A).$$

Obtemos então, o Teorema de *Bayes*:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}.$$

E, estendendo para mais eventos:

$$P(A|E_1, ..., E_n) = \frac{P(E_1, ..., E_n|A) \times P(A)}{P(E_1, ..., E_n)}.$$

Se considerarmos a hipótese da independência, ou seja, os eventos E são independentes entre si, podemos usar a seguinte propriedade:

$$P(E_1, ..., E_n|A) = P(E_1|A) \times P(E_2|A) \times ... \times P(E_n|A).$$

Por estarmos assumindo independência entre os eventos, obtemos a versão *naive* do Teorema de *Bayes*:

$$P(A|E_1, ..., E_n) = \frac{P(E_1|A) \times P(E_2|A) \times ... \times P(E_n|A) \times P(A)}{P(E_1, ..., E_n)}$$

A partir dessa versão do Teorema de *Bayes*, podemos classificar documentos considerando diferentes classes e *features*.

Primeiro, pode-se desconsiderar o denominador da fórmula (chamado de 'evidência') pois este será uma constante em todas as atribuições de probabilidade. Então, considerando que temos uma classe C com diferentes *features* F, obtemos:

$$P(C|F_1, ..., F_n) = P(C) \times \prod_{i=1}^{n} P(F_n|C).$$

Com essa equação obtemos a probabilidade de um documento ser da classe C. Se tivermos diferentes classes, consideramos que o documento pertence àquela a qual possui maior valor de probabilidade. Ou seja:

$$classificar(f_1, ..., f_n) = argmax P(C = c) \times \prod_{i=1}^{n} P(F_i = f_i|C = c)$$

O *Naive Bayes* é um algoritmo com aprendizagem pois aprende a partir de dados. Especificamente, é um algoritmo de aprendizagem supervisionada pois os dados do *training set* estarão separados em diferentes classes rotuladas.

A partir do *training set*, analisa-se as *features* do documento, e depois analisa-se o texto alvo para enfim classificar o documento utilizando a versão *naive* do teorema de *Bayes*.

Há diferentes formas possíveis de se considerar as *features* do documento à ser classificado. No caso de categorização de um documento em classes diferentes, pode-se analisar a quantidade de vezes que uma determinada palavra aparece no documento, por exemplo, ou outra forma possível seria considerar em quantas mensagens a palavra aparece. Decidir como essas *features* serão tratadas é importante para a definição do procedimento do algoritmo que irá extrair as probabilidades *a priori*.

Outras definições importantes são como serão armazenadas as probabilidades *a priori*, e como serão comparadas as palavras do texto alvo com essas informações obtidas do *training set*. A forma de armazenamento escolhido pode afetar consideravelmente a eficiência do algoritmo.

E por último, há uma melhoria importante que é o *smoothing*. Está técnica deve ser utilizada pois com o Teorema de *Bayes* seguindo a hipótese de independência, caso uma probabilidade *a priori* de uma palavra seja zero, a probabilidade do texto alvo pertencer àquela classe seria zerada por inteiro. Então, pode-se utilizar diferentes métodos estatísticos de *smoothing* para evitar este problema.

2.1.3 Aprendizagem Não-supervisionada

Na abordagem não-supervisionada, os dados de entrada não são rotulados e nenhum *feedback* é esperado pelo módulo de aprendizagem. Ou seja, não há dados de treino ou dados de exemplo para o treinamento desse módulo. A ideia é achar semelhança entre os dados não-rotulados.

Nesse tipo de aprendizagem, a tarefa mais comum abordada é a de clusterização. A clusterização é uma forma de aprendizagem não-supervisionada de divisão de dados, que tem por objetivo criar agrupamentos (ou clusteres) de dados semanticamente parecidos.

Essa organização se dá por semelhança ou padrões existentes entre os dados, geralmente representados na forma de vetores de atributos, pontos em um espaço multidimensional ou espaço de atributos.

Nesta abordagem não-supervisionada, uma série de técnicas pode ser utilizada para a identificação de dados semelhantes. No caso de dados textuais, normalmente, os dados são lidos e transformados para um valor numérico (é possível manter valores nominais, porém valores numéricos são mais fáceis de lidar no processamento dos dados) e então são utilizadas fórmulas que calculam as distâncias entre os valores numéricos desses dados, para assim descobrir quais pertencem à mesma categoria.

A transformação dos dados de treinamento em valores, que podem ser comparados, pode ser feita de diversas formas, impactando a configuração final dos clusteres encontrados. Além disso, a escolha inicial dos dados que serão comparados também pode afetar consideralvemente a formação dos agrupamentos.

Ao final de uma rotina de clusterização, serão evidenciados os agrupamentos encontrados pelo algoritmo de aprendizagem. Este resultado deve ser validado por especialistas no domínio da aplicação para saber se a clusterização foi eficaz na separação dos dados.

Cálculos de Distância

Apesar das diferenças entre os algoritmos utilizados em rotinas de clusterização, uma atividade comum é analisar a proximidade entre dois pontos (dados). Dessa forma, objetos que são encontrados como mais próximos são rotulados como pertencendo a um mesmo cluster. Esse cálculo de proximidade entre pontos também é utilizado para saber em qual dos clusteres novos dados serão rotulados.

Dentre as distâncias mais utilizados, estão as distâncias de *Manhattan* e de *Euclides*. Na Figura 2.3, é possível entender como funciona cada uma destas distâncias.

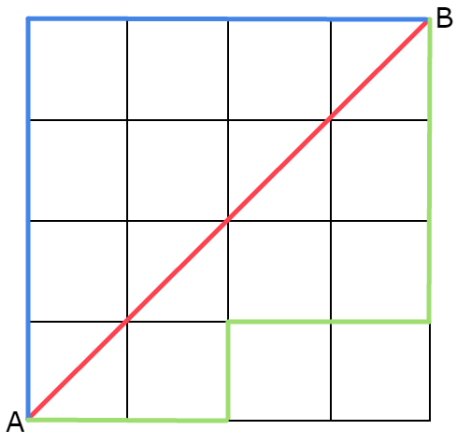

Figura 2.3: Cálculo de Distâncias de *Manhattan* e *Euclides*.

Entre os pontos A e B, estão ilustrados três caminhos diferentes. Em vermelho, é possível observar a ideia da distância euclidiana entre dois pontos e, em azul, como é a ideia da distância de *Manhattan*. O caminho em verde é equivalente ao caminho em azul,

ou seja, também utiliza a distância de *Manhattan*. Vale lembrar que existem outras configurações que também representam essa distância de *Manhattan*. A distância *Euclidiana* é o comprimento do segmento de reta que liga os dois pontos. A distância de *Manhattan* é simplesmente a soma das componentes horizontal e vertical entre os dois pontos.

Métodos de Aprendizagem Não-Supervisionada

Existem na literatura vários métodos de aprendizagem não-supervisionada. Serão apresentados os mais citados.

K-means

Um dos métodos mais simples e efetivos de clusterização é o método *K-Means*. Este método funciona escolhendo de antemão a quantidade k de agrupamentos que se está procurando. Então, k pontos são escolhidos como centros de agrupamentos. A partir daí, os dados restantes são atribuídos a um dos clusters utilizando distâncias entre pontos. Então, começa a fase iterativa do método. A partir desses agrupamentos obtidos inicialmente, calcula-se o centroide de cada *cluster* utilizando os dados associados àquele agrupamento. Então, repete-se todo o processo de atribuição de pontos aos clusteres e obtenção de centroides até que estes centros se tornem estáveis.

Nota-se que este método é bastante dependente da escolha inicial dos k pontos que servirão de centros de *clusteres*. Por isso é comum, assim como em outros métodos de clusterização, executar o algoritmo e avaliar os resultados com diferentes pontos iniciais [9].

Método Hierárquico

Existem diversas abordagens que os algoritmos de clusterização podem seguir. O algoritmo explorado anteriormente, *k-means*, é caracterizado como uma clusterização *particional*, pois já começa inicialmente com um conjunto de k clusteres e coloca os dados em cada um desses clusteres à medida que a rotina é executada.

Uma outra abordagem é a linha de clusterização hierárquica. O objetivo nesta forma de clusterização, é considerar, inicialmente, cada dado como representando um agrupamento. Então, à medida que acontecem as iterações do algoritmo, estes agrupamentos vão se juntando com os agrupamentos que forem mais similares. Até que, em um último passo, haverá apenas um agrupamento. Cabe ao usuário decidir quando o agrupamento deve parar. Pode-se adotar o critério de distância máxima para que não se agrupem mais os dados. Ou também uma análise de quantidade mínima de clusteres. Essa é a forma dita *aglomerativa*. Pode-se também seguir a forma *divisiva*, que começa com um cluster contendo todos os dados, e a cada iteração divide-se mais os dados [11].

2.1.4 Aprendizagem Semi-supervisionada

Na aprendizagem semi-supervisionada, assim como na supervisionada, há um conjunto de dados rotulados. A diferença é que, além dos dados rotulados, temos dados não-rotulados também. O sistema deve analisar esses conjuntos e fazer o melhor possível com o de dados não rotulados. Este tipo de abordagem é utilizado quando o conjunto de treinamento contém registros insuficientes para obter um resultado com boa precisão.

Essa é uma situação de interesse para a área de IA, pois rotular dados pode ser um processo muito custoso. Assim, a ideia é rotular, com uma certa margem de segurança, alguns dos exemplos do conjunto de exemplos não rotulados, os quais são posteriormente utilizados durante a fase de treinamento do classificador [12]. Em tarefas de *clustering*, os exemplos rotulados seriam utilizados como um conhecimento *a priori* para criação dos clusteres.

Podemos estar interessados, por exemplo, em criar um sistema que detecta a idade de pessoas a partir de suas fotos. Poderíamos, facilmente, encontrar uma base de dados com grande quantidade de registros de fotos de rostos de pessoas, mas sem a respectiva idade. Poderíamos encontrar, também, alguns registros de rostos de pessoas com a idade associada. Nesse caso, usaríamos esses dados de rostos com a idade para tentar rotular alguns dos exemplos do conjunto não-rotulado, e então melhorar nosso classificador.

2.2 Aspectos de Mineração[2]

Estamos na era da informação e a cada dia que passa geramos mais dados. O conceito de *Big Data* é utilizado para descrever esses grandes volumes de dados. Esse conceito cresce em relevância, pois a sociedade se depara com o crescimento sem precedentes na quantidade de informações geradas no dia a dia, onde apenas uma pequena fração dessas informações é analisada e interpretada de forma que se torne informação útil [14]. Bases de Dados estão se tornando essenciais para todo tipo de negócio e aplicações computacionais. É nesse contexto que fica explícita a necessidade de explorar esses dados para diminuirmos o *gap* existente entre esses e a informação organizada. Apesar de todo o avanço das técnicas computacionais e estatísticas, muitos gestores ainda têm dificuldades de lidar com a avalanche de dados que se encontra em suas bases eletrônicas.

Desde 1960 as bases de dados evoluíram de simples processadores primitivos de dados para ambientes sofisticados de geração de informação e de suporte à decisão. Segundo Fayyad et al. [15], existe uma urgente necessidade de geração de teorias e ferramentas computacionais que nos ajudem a extrair informação útil dos dados digitais, que crescem rapidamente. Nesse sentido, o crescente campo da Mineração de Dados ganha força e importância. Pode-se definir Mineração de Dados como a descoberta de padrões em dados de forma automática ou semi-automática. Ao buscar estes padrões, damos sentido aos dados e criamos condições de predizer situações futuras ao encontrar indícios de situações já conhecidas. Assim, a Mineração de Dados é uma evolução natural da tecnologia da informação e dá força a essas bases de dados dando a capacidade de explorá-las em busca de informações valiosas.

Neste campo, há uma divergência de nomenclatura. Por um lado, há quem considere que a Mineração de Dados é apenas uma das etapas do processo de descoberta de conhecimento *Knowledge-Discovery in Databases - KDD*. A etapa de Mineração de Dados seria a etapa de aplicação de algoritmos específicos para extração de padrões dos dados, sem abranger as outras fases do processo de KDD, tais como a preparação e limpeza dos dados [15]. Por outro lado, Mineração de Dados está se tornando, tambem, um sinônimo para todo o processo mais amplo de descoberta de conhecimento em grande quantidade de dados.

[2] O conteúdo desta seção foi baseado nas referências [9] e [13].

No processo de descoberta de conhecimento ilustrado na Figura 2.4, existem as seguintes etapas:

- limpeza dos dados: remoção de dados inconsistentes.
- integração de dados: combinação de diferentes fontes de dados.
- seleção de dados: recuperação de dados relevantes da base dados.
- transformação de dados: agregação, sumarização e qualquer tipo de transformação necessária para tornar os dados aptos para a mineração.
- mineração dos dados: aplicação de métodos inteligentes para extração de padrões nos dados.
- avaliação de padrões: identificação dos padrões realmente interessantes para o contexto.
- apresentação do conhecimento: apresentar o conhecimento obtido aos usuários.

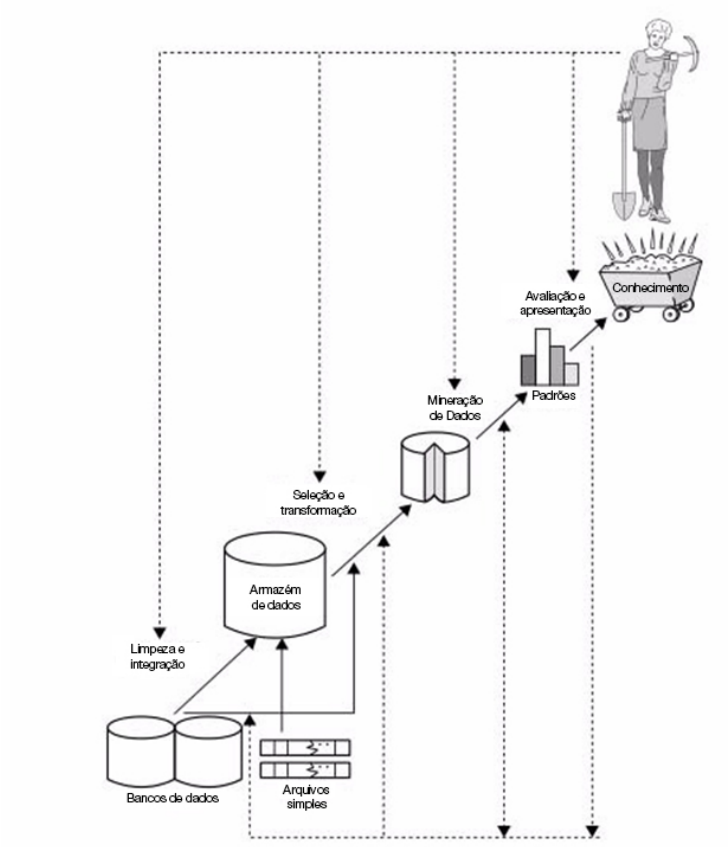

Figura 2.4: Processo de Descoberta de Conhecimento (traduzido de [13]).

2.2.1 Tarefas de Mineração de Dados

Considerando uma visão mais ampla, Mineração de Dados pode ser visto como o processo de descoberta de padrões e conhecimento a partir de quantidades massivas de

dados [13]. Mas, é possível refinar mais esse conceito, dividindo a Mineração de Dados em *Tarefas de Mineração*, para assim clarear mais as possibilidades desta área de pesquisa ao explícitar os diferentes objetivos que podem ser definidos ao utilizar técnicas de Mineração.

Desta forma, pode-se dividir o processo de mineração de dados nas seguintes tarefas [16]:

- análise de dados exploratória: exploração dos dados sem uma ideia clara do que se está buscando. Normalmente, são técnicas interativas e visuais, e podem ajudar em uma compreensão inicial dos dados.
- modelagem descritiva: nesta tarefa o objetivo é descrever os dados. Podem ser utilizados modelos para descobrir a distribuição de probabilidade dos dados, particionar o espaço de dados em grupos (clusterização e segmentação), e também podem ser empregados modelos para a descoberta de relações entre variváveis.
- modelagem preditiva: a partir de valores conhecidos de outras variáveis, o objetivo nesta tarefa é predizer o valor de uma variável. Podem ser utilizadas técnicas de classificação (quando a tarefa envolve predizer em qual *categoria* a variável pertence) ou técnicas de regressão (que preveem valores futuros baseando-se em valores anteriores).
- descoberta de padrões e regras: com a utilização de *regras de associação* o objetivo nesta tarefa é a detecção de comportamentos diferentes dos usuais em um conjunto de dados.
- recuperação por conteúdo: nesta tarefa o usuário já possui padrões de interesse e deseja encontrar padrões similares no conjunto de dados. Por exemplo, a tarefa pode ser encontrar documentos relevantes dado um conjunto de palavras-chave.

2.2.2 CRISP-DM

Ao longo da década de 90, observou-se que cada vez mais empresas utilizavam técnicas de mineração de dados para lidar com a quantidade massiva de dados que eram gerados em seus processos de negócio. Entretanto, esta ainda era uma indústria nova e imatura. E então, com a intenção de padronizar o uso desta técnica, um consórcio de empresas veteranas no assunto resolveu criar um modelo de processo industrial e independente de aplicação para a mineração de dados. Para evitar a criação de um padrão puramente teórico e desconexo da realidade empresarial, foi realizado um workshop onde foram convidados profissionais de mineração de dados para propor e discutir ideias para o modelo. A partir desse e de outros workshops e de mais de dois anos de discussão, refinamento e validação consolidou-se o modelo processual CRISP-DM (*CRoss-Industry Standard Process for Data Mining*) [17]. Existe também outro modelo de processo que chama-se *SEMMA* (acrônimo para *Sample*, *Explore*, *Modify*, *Model*). Foi desenvolvido pelo Instituto SAS [18]. Apesar de também ser um modelo genérico para o desenvolvimento de aplicações de Mineração de Dados, o *SEMMA* é normalmente mais utilizado em aplicações desenvolvidas com o software *SAS Enterprise Miner*.

O modelo processual CRISP-DM é dividida de forma hierárquica, utilizando abordagem *top-down*, do mais geral para o mais específico: fases, tarefas genéricas, tarefas

especializadas e instâncias de processos. O objetivo é que os dois níveis iniciais de abstração sejam estáveis e genéricos, enquanto os dois níveis finais sejam mais específicos à situação do projeto. Por essa razão, deve-se ocorrer um mapeamento entre modelo e processo entre esses dois conjuntos de níveis de abstração.

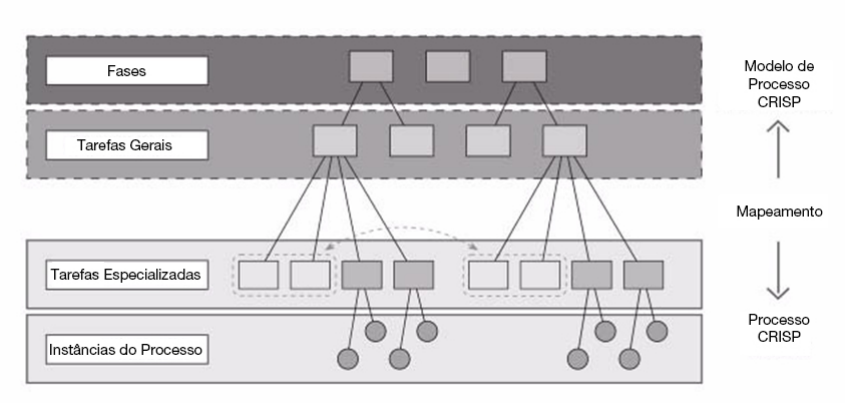

Figura 2.5: Níveis de Abstração no CRISP-DM (traduzida de [17]).

O modelo processual CRISP-DM se divide em seis fases não rígidas, ou seja, é possível e muitas vezes necessário ir e voltar entre as fases. Na Figura 2.6 estão ilustradas as seis fases, e é possível observar que há um círculo externo às fases. Este círculo representa a melhoria contínua do processo de mineração. Com os novos *insights* obtidos com o projeto, obtém-se uma noção melhor do contexto do negócio, e pode-se realizar projetos melhores de mineração e assim por diante.

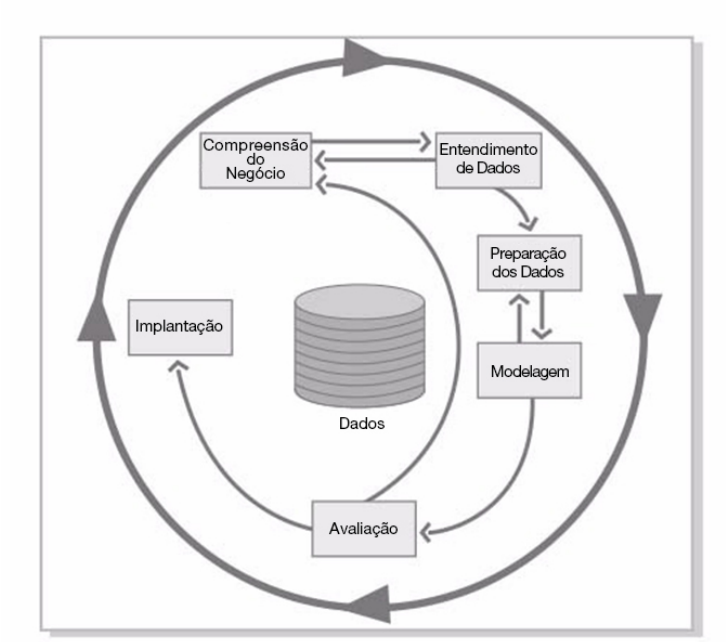

Figura 2.6: Fases do CRISP-DM (traduzida de [17]).

Segue uma breve descrição sobre cada fase do CRISP-DM:

- entendimento do negócio: nesta fase, são identificados os objetivos e requisitos do projeto sob a perspectiva do negócio para então converter estas metas em um projeto de mineração de dados. Deve-se entender o *background* do negócio, critérios de sucesso do projeto, orçamento e ferramentas disponíveis, terminologia do contexto, riscos e então delinear um plano preliminar para atingir os objetivos identificados.
- entendimento dos dados: aqui é realizada a coleta e familiarização com os dados. Com isso, é possível identificar a formatação destes e também seus possíveis problemas de qualidade (*missing values* por exemplo). Já nesta fase, pode-se obter alguns *insights* sobre o problema e eventualmente criar hipóteses sobre sub-conjuntos relevantes para o contexto.
- preparação dos dados: esta fase inclui todas as atividades necessárias para a construção do conjunto final de dados que será utilizado na fase seguinte de modelagem. Comumente, estas atividades incluirão seleção de tabelas, registros e atributos; transformar dados para que fiquem no formato de entrada das ferramentas que serão utilizadas; transformações sintáticas; derivação de novos atributos; reordenação de atributos.
- modelagem: nesta etapa, as técnicas de modelagem são de fato aplicadas. Podem ser utilizada várias abordagens diferentes e talvez seja necessário voltar à fase de preparação de dados para adequá-los às exigências das técnicas empregadas. Além de escolher os modelos em si (redes neurais, árvores de decisão), deve-se delinear os conjuntos de teste e de treinamento e ajustar parâmetros.
- avaliação: nesta fase, deve-se verificar que o modelo obtido de fato atende às necessidades do negócio. Deve-se também avaliar se é necessário repetir alguma atividade.
- implantação: utilização do modelo obtido, seja para geração de relatórios que deêm suporte á tomada de decisões, seja na forma de um programa que torne a mineração um processo repetível. É importante também delinear planos de monitoramento e manutenção do modelo.

2.2.3 Avaliação dos Modelos de Classificação

Após a utilização de um conjunto de treinamento para a obtenção de um modelo de classificador, deve-se proceder à fase de avaliação do modelo. Existem várias métricas que podem ser utilizadas para avaliar um classificador [13]. Mas antes de analisá-las, deve-se estar confortável com os seguintes termos:

- verdadeiro positivo (*true positive* - TP): instâncias classificadas corretamente como pertencentes à classe de interesse.
- falso positivo (*false positive* - FP): instâncias classificadas como pertencentes à classe de interesse mas que não são da classe de interesse.
- verdadeiro negativo (*true negative* - TN): instâncias classificadas corretamente como não pertencentes à classe de interesse.

- falso negativo (*false negative* - FN): instâncias classificadas como não pertencentes à classe de interesse mas que são, de fato, da classe de interesse.

Os valores TP, FP, TN, FN aparecem nas chamadas *matrizes de confusão*. Programas de mineração de dados normalmente exibem estas matrizes para avaliação do classificador.

A partir destas noções primárias, podemos proceder às métricas comumente utilizadas para avaliação de classificadores:

$$Precisão = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

O valor de precisão mede quantas, dentre as instâncias classificadas como positivas (TP e FP), são realmente instâncias pertencentes à classe de interesse (exatidão).

Por outro lado, o valor de *recall* mede quantas instâncias que pertencem à classe de interesse são realmente classificadas como tal (completude).

Por fim, uma medida que combina estas previamente citadas é a Medida F (*F-Measure*), que é uma media harmônica entre estas medidas:

$$F = \frac{2 * Precisão * Recall}{Precisão + Recall}$$

As medidas de precisão, *recall* e Medida F são baseadas em noções de acurácia de classificação. Mas existem outras formas de avaliar estes modelos, tais como velocidade de geração e utilização do classificador, capacidade do classificador de lidar com ruídos e dados faltantes, escalabilidade do classificador, grau de entendimento do modelo obtido e Área sob a Curva ROC.

Curva ROC

Utilizada pela primeira vez na Segunda Guerra Mundial, a Curva ROC (*Receiver Operating Characteristic*) foi desenvolvida para mostrar a relação entre um sinal e um ruído. Um sinal é um exemplo de um verdadeiro positivo (TP), e o ruído seria um alarme falso, ou seja, um falso positivo (FP).

A Curva ROC é um gráfico de *sensibilidade* versus *1-especificidade*. A sensibilidade é o valor da taxa de verdadeiros positivos (TP), também conhecida como taxa de *recall*. A especificidade é a taxa de verdadeiros negativos (TN).

A Figura 2.7 ilustra uma curva ROC (a curva mais escura irregular).

O que essa curva expõe é o *trade-off* entre capacidade de detectar casos que devem ser detectados e falsos alarmes. Quanto mais brando for o critério do classificador para a detecção de um dado como pertencente à uma categoria, maior a chance de aparecer um falso alarme.

Uma forma, então, de analisar o desempenho de um classificador é analisar a Área sob a Curva ROC pois esse valor leva em consideração os diferentes pontos de *trade-off* da curva. Quanto maior esse valor, mais o gráfico consegue atingir uma taxa alta de verdadeiros positivos sem aumentar tanto a taxa de falsos positivos. Nesse sentido, um

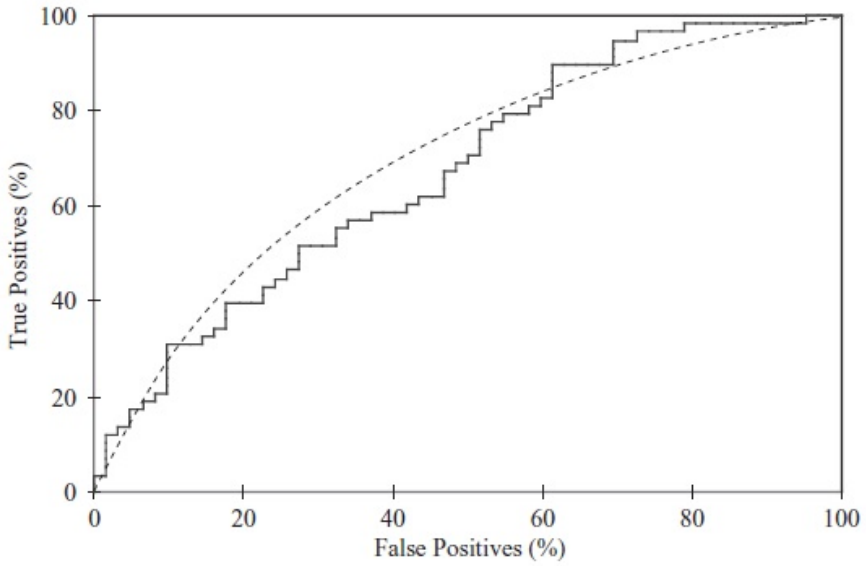

Figura 2.7: Exemplo de uma Curva ROC [9].

classificador que obtivesse uma Área sob a Curva de valor 0,5 não estaria melhor do que a sorte, pois estaria detectando a mesma proporção de verdadeiros positivos e falsos positivos. Um classificador que obtivesse uma Área sob a Curva de valor 1,0 seria um classificador excelente. Por fim, classificadores com valores abaixo de 0,5 estariam piores do que o acaso.

2.2.4 Processamento de Texto

Processamento de dados textuais em linguagem natural é uma área de pesquisa que se encontra na interseção entre a área de IA e Linguística Computacional. É uma área com grande demanda devido à crescente necessidade de tratamento de informações digitais expressas em texto. Como a informação é produzida rapidamente e em larga quantidade no mundo moderno, técnicas inteligentes são necessárias para um tratamento rápido e eficiente desses dados. Dentre as aplicações dessa área de pesquisa pode-se citar: extração e recuperação de informações textuais, sumarização, tradução de texto, agentes inteligentes de conversação e diálogo [19].

Análise Linguística

Historicamente, a análise linguística se dividiu em etapas e o processamento de linguagem natural segue também essa divisão [20]. As etapas são: análise sintática (ordem e estrutura do texto), análise semântica (significado) e análise pragmática (análise do discurso). Mesmo que seja uma divisão eminentemente pedagógica (devido à dificuldade de se analisar um texto em etapas tão claramente distintas), essa divisão traz benefícios para o gerenciamento mais adequado do processamento.

Porém, essa divisão foi ainda mais refinada ao longo do tempo e a granularização do modelo foi refinada, produzindo uma decomposição mais detalhada para atender o tratamento de dados em linguagem real [21]. Não obstante a variação em divisão de fases, ainda há diversas formas de dar sequência às diferentes etapas, seguindo de forma linear ou não-linear, com retro-alimentações ou não. Mas, de forma didática, pode-se pensar na decomposição linguística como tendo a estrutura representada na Figura 2.8.

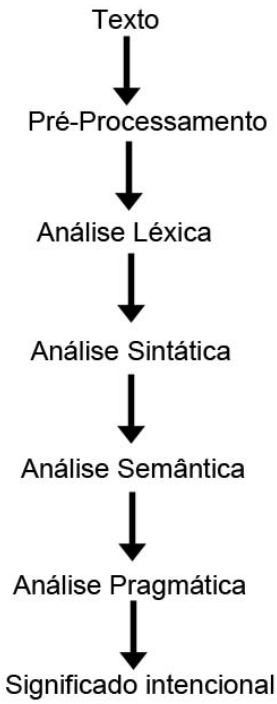

Figura 2.8: Estágios do processamento de Linguagem Natural.

Cada etapa tem suas peculiaridades. Segue uma breve explicação sobre cada uma delas:

- pré-processamento textual: pode ser dividido em duas etapas. Primeiro, é feita a conversão do arquivo digital em um documento textual bem formatado. Com a conclusão desse estágio, obtemos um *corpus* bem definido. A segunda etapa do pré-processamento consiste na segmentação do texto em unidades lexicais e sentenças. Para isso, é importante definir os delimitadores textuais, como por exemplo espaços em branco e vírgulas. A divisão em unidades lexicais terá como resultado a obtenção dos *tokens* do texto. Depois de reconhecer as unidades lexicais, prossegue-se para reconhecer as orações presentes no texto.
- análise léxica: nessa estapa, ocorre a decomposição das unidades lexicais identificadas no pré-processamento. Com essa decomposição, é possível obter regras de formação morfológica das palavras e assim obter uma economia de espaço de armazenamento, aumentar a velocidade do processamento e além disso preparar a aplicação para caso encontre palavras que ainda não tinha se deparado antes. Para esta etapa, normalmente são utilizados autômatos de estado finito.

- análise sintática: nessa fase, é realizada uma análise estrutural de uma sequência de unidades lexicais, identificadas em etapas anteriores. O objetivo é obter como saída uma estrutura hierárquica sintaticamente decomposta que permita uma análise semântica. Para isso, deve ser definida uma gramática formal livre de contexto.
- análise semântica: divide-se, tradicionalmente, em análise semântica léxica e composicional. Na análise semântica léxica, o objetivo é estabelecer o significado das unidades léxicais. Na segunda, o objetivo é compreender as inifinitas possibilidades de combinações de unidades lexicais em frases que respeitem a gramática formal estabelecida.
- análise pragmática: nesta etapa, o discurso e o contexto são analisados. Entre outros, deve-se tentar obter relações de coerência e coesão textual.

Classificação Automática de Textos

A classificação, ou categorização, de textos, é o processo de organização dos textos em categorias. Essas categorias podem ser utilizadas para extrair o conteúdo principal do texto, realizar previsões sobre novos fragmentos de texto ou obter estatísticas sobre esses. Por exemplo: a partir de uma série de textos, quantos deles são relacionados à política, quantos são relacionados à esportes e quantos são relacionados à resenhas de filmes.

A partir de categorias dentro de um contexto de estudo, pode-se aplicar técnicas de aprendizagem supervisionada para criar um classificador textual automático. Para essa tarefa, é necessário um conjunto de treinamento, ou seja, amostras relativas à cada categoria definida. Com isso, técnicas diferentes devem ser testadas, e avaliar dentro do contexto, e com os dados disponíveis, qual método apresenta a melhor eficácia em prever a categoria de novos textos.

Antes da obtenção das amostras de treinamento, é necessário definir as categorias do texto. Geralmente essa definição é feita por algum técnico ou especialista da área. Também é necessário preparar os dados textuais para que possam ser utilizados nas ferramentas ou rotinas de aprendizagem (fase de pré-processamento textual).

Utilizando como exemplo o contexto de Segurança Pública, como uma nova entrada no classificador um incidente com a descrição "Há manifestantes na rua", o classificador provavelmente iria classificar esse incidente como "Manifestação". De forma similar, outros incidentes com descrições semelhantes às descrições da amostra de treinamento, como por exemplo "Tumulto em protesto contra a Copa", seriam classificados como "Manifestação".

Observa-se, então, que essa é uma *tarefa de mineração* do tipo de modelagem preditiva (Seção 2.2.2). Ou seja, estamos a partir de conjuntos rotulados de dados (no caso, fragmentos de texto) criando um modelo que consegue prever uma característica (no caso, categoria) do dado de entrada.

Também se evidencia aqui as etapas do modelo CRISP-DM (vide Seção 2.2.1) para realizar esta tarefa de mineração: entendimento do négocio e dos dados (no projeto desenvolvido nesta monografia essa etapa seria o entendimento do contexto de C2 e funcionamento do sistema Pacificador de onde serão retirados os dados de incidentes), preparação dos dados (técnicas de pré-processamento), modelagem (aplicação das técnicas de aprendizagem supervisionada), avaliação e implantação (disponibilização do classificador para uso).

Pré-Processamento Textual

Com a finalidade de criar um classificador automático de incidentes de Segurança Pública em grandes eventos, a fase de *pré-processamento textual* será de fundamental importância. Deve-se dedicar uma atenção especial à essa etapa pois representa o primeiro passo da fase de modelagem e escolhas feitas aqui irão afetar significativamente a eficácia do classificador.

Nesta etapa de pré-processamento, há uma diversidade de ações que são aplicadas para melhorar a eficàcia do modelo. Segue algumas dessas ações:

- remoção de *stopwords*: *stopwords* são as palavras mais comuns de um idioma (artigos, alguns pronomes, etc) e que por isso trazem pouca informação sobre o significado de uma frase [22]. Nessa etapa, normalmente há uma lista de stopwords e então o texto é percorrido removendo essas palavras.
- *lower case*: na maior parte dos contextos, não é interessante que uma palavra seja considerada diferente da outra só por que difere em uma letra maiúscula por exemplo. Nesse sentido, deve-se converter todas as palavras em minúsculo.
- *stemming*: há uma série de palavras com o mesmo radical (*stem*) e que refletem a mesma semântica, mas que diferem em seus sufixos. Por exemplo, para um classificador, é interessante que as palavras "manifestantes" e "manifestando" sejam tratadas como o mesmo indicativo de categoria. Poderíamos então, manter apenas o radical "manifest".

Pode-se observar nessas sub-tarefas de pré-processamento que a linguagem do contexto é importante. A lista de *stopwords* varia de uma língua para outra. A análise de radicais e semelhança de palavras varia de uma linguagem para outra.

Outra etapa essencial ainda dentro do pré-processamento é a transformação do texto em uma estrutura de dados que pode ser utilizada eficientemente por algoritmos de aprendizagem. Isso se dá transformando a *string textual* em um vetor de palavras. O valor contido em cada posição do vetor é um fator de ajuste. Pode-se utilizar os seguintes valores ou pesos para cada palavra:

- presença binária: a posição do vetor correspondente à uma palavra simplesmente indica a presença ou não daquela palavra no fragmento de texto.
- frequência no fragmento (*term frequency* - TF): a posição do vetor correspondente à uma palavra indica a quantidade de vezes (frequência) que a palavra aparece no fragmento.
- frequência no *Corpus*: a posição do vetor correspondente à uma palavra indica a quantidade de vezes que a palavra aparece no conjunto total de documentos.
- inverso da frequência no *Corpus* (*inverse document frequency* - IDF): a posição do vetor correspondente à uma palavra indica o inverso da quantidade de vezes que a palavra aparece no conjunto total de documentos. É de grande valia para descobrir a raridade de uma palavra. Quanto menos ela aparece no conjunto total dos textos, maior sua raridade e valor.
- misturado: junção de mais de uma dessas medidas acimas. Por exemplo, considerando a frequência no fragmento e o inverso da frequência no *Corpus* - TF-IDF.

2.2.5 Ferramentas de Mineração de Dados

Existe uma grande diversidade de ferramentas que podem ser utilizadas para tarefas de mineração de dados. Entre as ferramentas de código proprietário pode-se citar a Microsoft Azure [23] e a Amazon Machine Learning [24]. Dentre as ferramentas *open source* pode-se citar WEKA [25], RapidMiner [26], Orange [27] e *R* [28]. Dentre essas ferramentas, o WEKA foi escolhido por ser *open source*, posuir uma interface de mais fácil entendimento, possuir portabilidade boa entre sistemas operacionais e pela disponibilidade de ampla documentação.

WEKA

Escrito em Java e desenvolvido na Universidade de Waikato (Nova Zelândia) o WEKA é um software popular de Mineração de Dados. Fornece uma interface intuitiva e disponibiliza uma série de algoritmos para processar os dados de entrada e tentar generalizar uma solução para o problema estudado.

O software pode ser utilizado para uma gama variada de aplicações, tais como criação de classificadores automáticos, processamento de linguagem natural, tarefas de clusterização e obtenção de regras de associação. Essas ferramentas são suficientes e adequadas para o desenvolvimento do projeto aqui desenvolvido, e nessa Seção será mostrado um pouco de como classificar textos utilizando esse programa.

Em sua interface inicial, o WEKA oferece ao usuário diferentes possibilidades quanto à utilização do software [29]:

- *simple CLI*: neste modo de operação, o usuário poderá entrar com comandos em um terminal. Algumas tarefas só são possíveis de serem realizadas desta forma, como por exemplo a transformação de diretórios contendo arquivos de texto em um arquivo ARFF que possa ser trabalhado pelo WEKA.
- *explorer*: interface gráfica onde o usuário pode conduzir experimentos em dados nos mais diferentes formatos, inclusive dados textuais. Neste modo de operação o usuário pode testar diferentes técnicas de classificação, clusterização e obter regras de associação para o conjunto de dados em estudo.
- *experimenter*: interface gráfica onde o usuário pode conduzir experimentos e variações desses com manipulações estatísticas.
- *knowledge flow*: similar ao modo *experimenter*, mas com funcionalidades de *drag and drop*.

É importante ressaltar que os modos de operação citados acima não são excludentes. É possível e necessário para algumas tarefas, utilizar inicialmente um modo e posteriormente outro. No trabalho desenvolvido nesta monografia, o modo *simple CLI* é utilizado para transformar os dados de incidentes em dados que possam ser utilizados pelo WEKA. O modo *explorer* é utilizado para testar os diferentes classificadores e o modo *knowledge flow* é utilizado para obter as Curvas ROC.

A Figura 2.9 ilustra o WEKA operando no modo *explorer*.

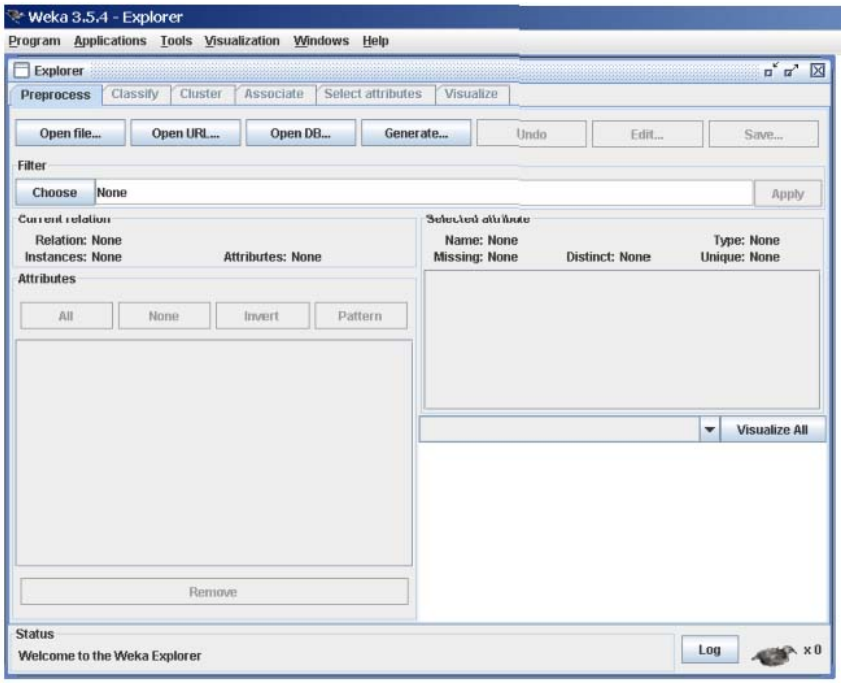

Figura 2.9: Interface do modo *explorer* do WEKA [29].

No modo *explorer*, o WEKA apresenta as seguintes seções [29]:

- *pre-process*: nesta seção, os dados sob análise são carregados no programa, podendo ser a partir de banco de dados, URL ou do arquivo padrão do Weka (arquivo ARFF, que será explicado a seguir), pré-processados (aplicação de filtros, em destaque filtros que convertem *strings* de texto para vetores de atributos) e seus atributos são analisados. É possível observar informações sobre os dados carregados, como a ocorrência dos mesmos no arquivo de origem. É nesta aba onde são aplicados filtros nos dados de origem para torná-los aptos ao processo de mineração.
- *classify*: aqui pode-se testar a criação de diferentes modelos de classificadores para o conjunto de treinamento fornecido e analisar e comparar resultados, sendo possível escolher dentre diferentes métodos de aprendizagem para fazer a classificaçao. Nesta aba também há opções para decidir quem será o conjunto de teste.
- *cluster*: semelhante à seção de *Classify*, mas nesta seção podem ser utilizadas técnicas de clusterização para obtenção de agrupamentos a partir de dados não rotulados.
- *associate*: semelhante à seção de *Classify*, mas nesta seção podem ser utilizadas técnicas de descoberta de associações nos dados. Regras de associação podem ser utilizadas para encontrar comportamentos anômalos em dados.
- *select attributes*: permite buscar dentre todas as possibilidades de atributos quais subconjuntos destes é mais adequado para a utilização em modelos de predição.
- *visualize*: diversas formas de visualização para os resultados obtidos nas seções anteriores.

Pré-Processamento dos Dados

Conforme apresentado, a aba de *Pre-process* permite a aplicação de filtros nos dados carregados. Em especial, para tarefas de mineração de texto, o filtro *StringToWordVector* é de fundamental importância. Com esse filtro, ilustrado na Figura 2.10, é possível preparar os dados para a etapa seguinte de mineração.

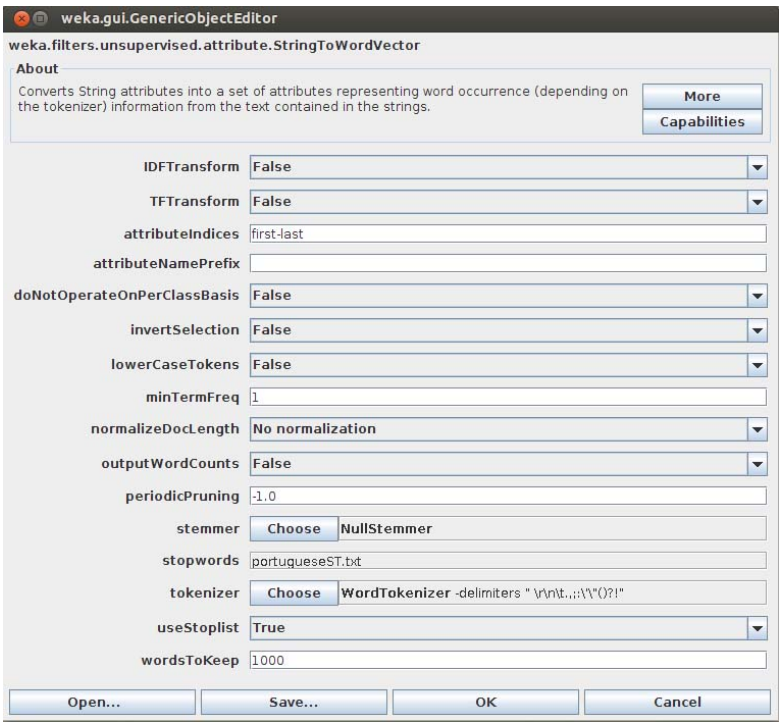

Figura 2.10: Edição do Filtro *StringToWordVector* do WEKA.

Pode-se observar que esse filtro oferece ao usuário a capacidade de ponderar as palavras, utilizando a frequência das palavras (*Term Frequency Transform* e *Inverse Document Frequency Transform*). Oferece também a possibilidade de ajustar qual a frequência mínima de ocorrências que um termo deve ter para que seja considerado. É nesta janela também que pode-se inserir, através de um arquivo, uma lista de palavras de *stopwords* para serem desconsideradas pelo WEKA. Também pode-se entrar com a quantidade máxima de palavras distintas que o WEKA deve armazenar. Esse valor é importante para fins de eficiência computacional e para tornar possível a análise de alguns contextos onde a quantidade de palavras é excessiva. E finalmente, o WEKA oferece várias possibilidades de *Stemmers*, ou seja, métodos de redução de palavras à seus radicais, para uma melhor generalização da solução.

Opções de Teste de Modelos no WEKA

Na tarefa de obtenção de modelos de predição no WEKA, utilizando diferentes algoritmos como *Naive Bayes* ou Árvore de Decisão, a acurácia destes modelos dependerá também da forma de teste empregada.

O WEKA permite, em tarefas de classificação, as seguintes formas de teste, ilustradas na Figura 2.11, canto superior esquerdo da interface de *Classify* do WEKA:

- *use training set*: o classificador será testado com as instâncias que foram utilizadas para criação do modelo em si.
- *supplied test set*: o classificador será testado com instância fornecidas através de um arquivo.
- *cross-validation*: o conjunto é dividido em N partições (*folds*) aproximadamente iguais e uma destas partições é utilizada para teste, enquanto $N - 1$ são utilizadas para treinamento. Ou seja, o procedimento é executado N vezes, e ao final tira-se uma média para obtenção do resultado final. Dividir um conjunto de dados textuais em diferentes partições permite avaliar um classificador em conjuntos com diferentes vocabulários.
- *percentage split*: escolhe-se uma porcentagem dos dados que será utilizada para teste do modelo.

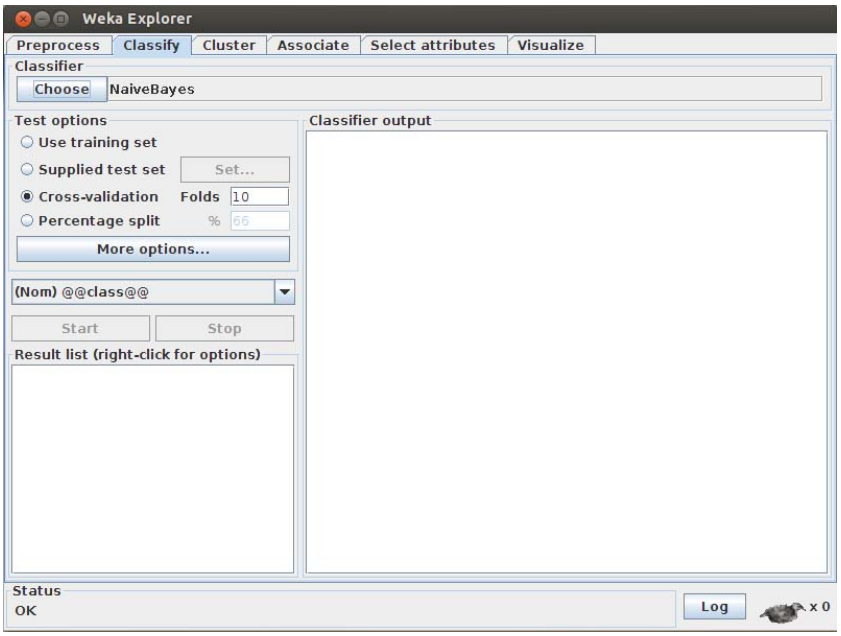

Figura 2.11: Aba *Classify* do WEKA.

Arquivo ARFF

O arquivo padrão do WEKA é chamado de *ARFF* (*Attribute-Relation File Format*). É um arquivo texto que descreve uma lista de instâncias e um conjunto de atributos. Ele possui duas seções: uma de *header* e outra de *data*. A parte do header contém o nome da relação (definida pela expressão @RELATION), uma lista de atributos (@ATTRIBUTE) e para cada atributo, seu tipo (NUMERIC, STRING, etc). O campo data se inicia pela expressão @DATA, onde em cada linha tem-se um vetor (representando um determinado

dado) com os valores dos atributos separados por vírgulas. Um exemplo de arquivo arff está representado na Figura 2.12.

```
@RELATION relationName

@ATTRIBUTE attribute1 NUMERIC
@ATTRIBUTE attribute2 NUMERIC
@ATTRIBUTE attribute3 STRING

@DATA
5.1,3.5,"string"
4.9,3.0,"string"
```

Figura 2.12: Exemplo da estrutura de um arquivo ARFF [9].

Em tarefas de mineração de texto, deve-se dispor de um método de conversão de arquivos texto para o formato de arquivo ARFF. Para isso, utiliza-se o comando *TextDirectoryLoader*. Este comando está disponível para utilização no modo *SimpleCLI* que é uma das formas de utilização do WEKA (junto com *Explorer*, *KnowledgeFlow* e *Experimenter*). Entra-se com esse comando no terminal fornecido pelo modo *SimpleCLI* fornecendo um caminho de diretório contendo uma quantidade arbitrária de pastas. Cada pasta deve ser nomeada e representará uma determinada categoria. Dentro da pasta, devem estar arquivos contendo cada uma da descrições textuais da categoria representada nessa pasta. Por exemplo, podemos criar uma pasta com o nome de "Manifestação" contendo vários arquivos de texto de incidentes relativos à manifestações. O comando se encarregará de transformar estas descrições em um arquivo ARFF que pode ser trabalhado pelo WEKA.

WEKA API para Java

Além da utilização do WEKA em interface gráfica, nos modos *Explorer* ou *SimpleCLI* por exemplo, é possível também utilizar o WEKA de forma embutida em programas Java. Essa ferramenta possui uma série de funções bem especificadas definidas através de uma *Application Programming Interface* (API).

A partir desta API pode-se utilizar o WEKA embutido de formas diferentes dependendo na necessidade do programador. Uma forma possível é utilizar as funções fornecidas para todas as tarefas envolvidas na mineração. Ou seja, carregar dados, criar classificador, e usar o classificador. Um outro formato possível e de grande valia é utilizar primeiro a interface gráfica para experimentar modelos e testar classificadores e depois utilizar a API apenas para tornar o modelo re-utilizável.

Esta última forma citada é interessante por que permite ao programador criar algum tipo de aplicativo que torne fácil e intuitivo o uso de seu modelo criado com a interface gráfica do modo *Explorer*, por exemplo.

A Figura 2.13 ilustra a utilização de algumas funções da API. No exemplo ilustrado, cria-se uma Árvore de Decisão, com o algoritmo *J48*, testa-se o modelo obtido com um conjunto próprio de dados para teste e então são utilizadas funções para que as estatísticas resultantes do testes sejam exibidas para o usuário.

```
import weka.core.Instances;
import weka.classifiers.Evaluation;
import weka.classifiers.trees.J48;
...
Instances train = ...   // from somewhere
Instances test = ...    // from somewhere
// train classifier
Classifier cls = new J48();
cls.buildClassifier(train);
// evaluate classifier and print some statistics
Evaluation eval = new Evaluation(train);
eval.evaluateModel(cls, test);
System.out.println(eval.toSummaryString("\nResults\n\n", false));
```

Figura 2.13: Exemplo de utilização da API do WEKA [29].

Com a ferramenta WEKA, é possível utilizar os métodos de aprendizado de máquina para o processamento e mineração de dados de incidentes de Segurança Pública em grandes eventos.

2.3 Grandes Eventos

Grandes eventos reúnem uma imensa quantidade de pessoas, diferentes países, e atividades espalhadas por diversas localidades. É o caso, por exemplo, da Copa do Mundo de 2014 e da Copa das Confederações de 2013, sediadas no Brasil. Essas características demandam do anfitrião uma capacidade de gerenciamento extraordinária. Em particular, o cuidado com a segurança do evento é um dos aspectos mais essenciais. Essa segurança envolve o planejamento de onde alocar recursos, e também o gerenciamento de incidentes que ocorram durante o evento.

Devido à magnitude do contexto de Segurança Pública, o gerenciamento da segurança de grandes eventos é um trabalho interagências. Envolve diferentes instituições, como Marinha, Polícias Civil, Militar e Federal, Agência Brasileira de Inteligência (ABIN), e em particular, o Exército Brasileiro desempenha um papel essencial.

O Exército Brasileiro atua em diferentes frentes. As operações militares podem ser ofensivas, defensivas, de pacificação ou de apoio a orgãos governamentais. E para o sucesso dessas operações, a atividade especializada de C2 é essencial [1].

2.3.1 Comando e Controle

Operações militares envolvem uma cadeia de comando com um fluxo de informações constante. E com essas informações, decisões são tomadas para o gerenciamento eficiente de recursos humanos e materiais. Para zelar pelo funcionamento dessa cadeia de comando, emprega-se a atividade especializada de *C2*. A atividade de C2, envolve três aspectos interconectados:

- Autoridade: figura de um comandante que possui legitimidade para exercer o comando, tomando decisões e avaliando informações que chegam atráve de um fluxo constante de dados.
- Processo Decisório: estabelecimento de um fluxo de informações que permite a formulação de ordens.
- Estrutura: envolve todos os recursos, materiais ou humanos para o exercício eficiente da atividade de C2. Envolve pessoal especializado, infra-estrutura física e tecnológica.

Então, pode-se definir um Sistema de C2, como um sistema que abrange essas três características citadas.

Os Sistemas de TIC são parte essencial da *Estrutura* de um Sistema de C2. Esses sistemas dão suporte à tomada de decisão ao permitir uma *consciência situacional* mais adequada.

A *consciência situacional* é a percepção acurada do ambiente. Envolve a percepção de como e onde estão alocados os recursos disponíveis e uma visão geral dos incidentes e quais necessitam de mais atenção. Essa percepção do ambiente permitirá um planejamento e consequente decisão de ações à serem tomadas [30].

O modelo de Observar, Orientar, Decidir e Agir (OODA), desenvolvido por John Boyd, é um ciclo que ajuda a compreender o processo de tomada de decisão e a atividade de C2. A fase de *Observar* envolve a captação do maior número possível de estímulos que afetam o ambiente operacional. A fase de *Orientar* envolve a análise e interpretação dos dados coletados na etapa anterior, além da identificação de ameaças reais ou riscos. Nessa fase também são formuladas as linhas de ação que são passadas ao comandante. Na fase de *Decidir*, o comandante toma suas decisões a partir das linhas de ação apresentadas e emite ordens. E, por fim, na fase de *Agir*, ações específicas são realizadas, atuando sobre o ambiente operacional. Nessa fase também é exigido uma atualização do ambiente, e dessa forma, reinicia-se o ciclo. Então, essas fases são executadas continuamente e quanto mais rápido for a execução de cada fase, mais ágil será o processo decisório. A Figura 2.14 ilustra o Ciclo OODA.

Nesse sentido, fica claro a importância de uma *consciência situacional* adequada. Uma percepção clara do ambiente operacional é a primeira etapa desse ciclo, e se as percepções do ambiente forem falsas ou incompletas, as ações tomadas não irão afetar o ambiente de forma efetiva. Fica também evidente a importância dos Sistemas de TIC. Esses sistemas melhoram a percepção do ambiente, e tornam essa percepção mais ágil, tornando todo o processo de tomada de decisão mais ágil também.

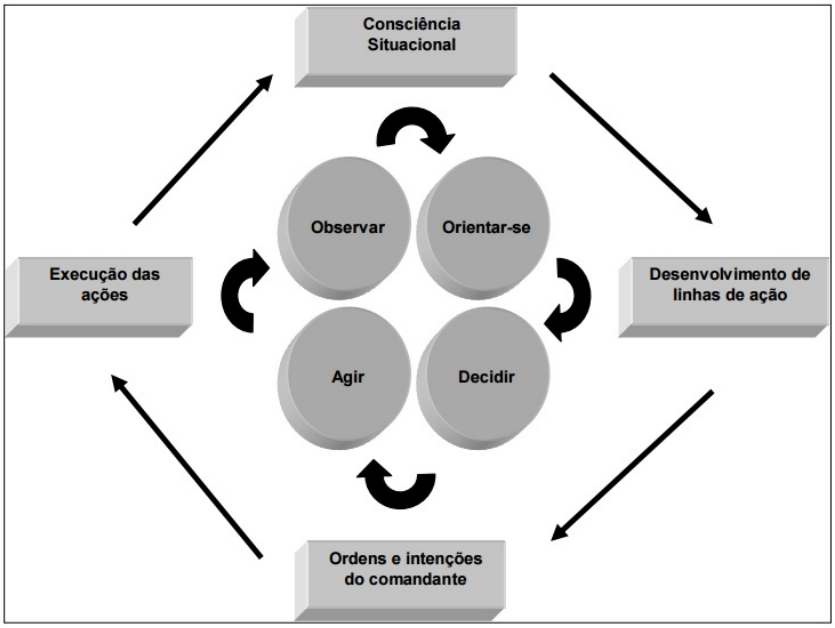

Figura 2.14: Ciclo OODA [1].

2.3.2 Pacificador[3]

O Pacificador é um sistema desenvolvido pelo Centro de Desenvolvimento de Sistemas (CDS) do Exército Brasileiro. É utilizado como suporte em operações de grande porte auxiliando na tomada de decisão. O Pacificador reúne dados e informações relativas a situações que estão acontecendo em tempo real em diversos locais. Essas informações são expostas para o Comando e operadores do sistema que com isso, obtêm uma *consciência situcional* do ambiente de um evento. Essa percepção aumenta a qualidade da ação tomada.

Esse sistema de C2 é utilizado para o suporte à operações em grandes eventos, e foi utilizado por exemplo na Copa das Confederações em 2013 e na Copa do Mundo FIFA em 2014.

O Pacificador funciona com mapas interativos de onde estão ocorrendo os eventos. Nesses mapas, podem ser adicionados marcadores que indicam situações que exigem atenção ou relatos de atualização. Essas informações são adicionadas por agentes móveis (em campo através de um *smartphone*), ou pelos operadores do sistema. Dessa forma, os responsáveis pela operação tem um quadro melhor do que está ocorrendo, possibilitando decisões como mudança de localização de uma equipe, alteração de rota, deslocamento emergencial de alguma unidade, etc.

Entre os elementos principais do Pacificador, serão utilizados no experimentos desse trabalho os incidentes. Incidente é "qualquer evento não planejado que possa comprometer a segurança do evento, requerendo tratamento adequado, e pode ser classificado como aberto (se ainda estiver ocorrendo), tratado (se ações já foram tomadas porém ainda não foi resolvido) ou fechado (caso já tenha sido resolvido)" [2].

A Figura 2.15 ilustra a marcação de um incidente no mapa da cidade de Brasília.

[3]Esta seção foi escrita baseando-se no manual do sistema Pacificador [2].

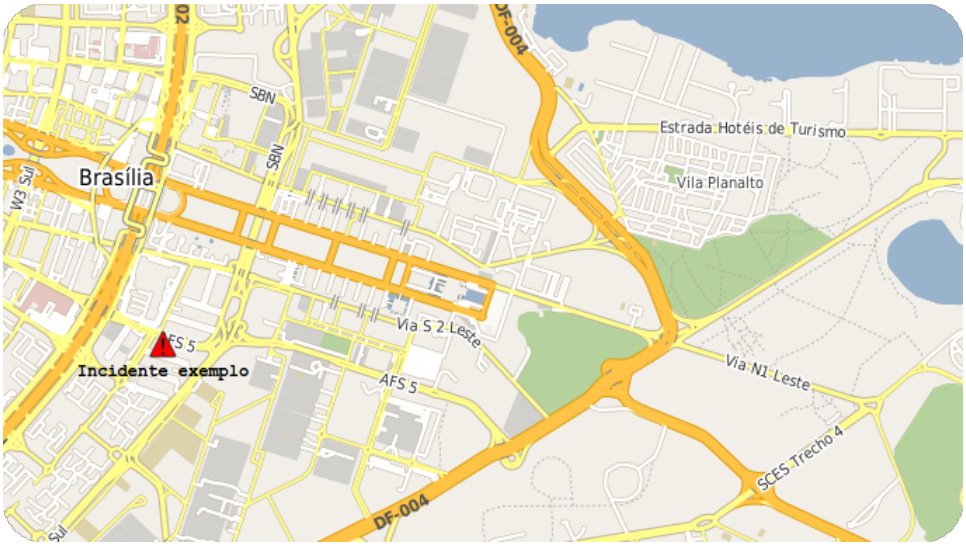

Figura 2.15: Cenário do Pacificador com um Incidente [2].

Para um agente adicionar um incidente, deverá ser fornecido um nome, data e hora, e uma descrição do ocorrido. Na Figura 2.16, que ilustra a interface do Pacificador, é possível observar um exemplo da janela de inserção de um novo incidente.

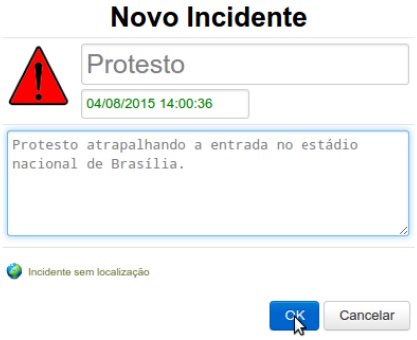

Figura 2.16: Exemplo de um Incidente do Pacificador [2].

Os incidentes relacionados a um evento são armazenados e podem ser consultados posteriormente. A análise de incidentes de eventos passados permite aumentar a compreensão sobre o quadro de problemas enfrentados em uma operação. Desta forma, é possível gerenciar melhor os recursos em um próximo grande evento.

2.4 Trabalhos Correlatos

Uma série de trabalhos com características semelhantes já foram desenvolvidos. As técnicas de mineração de dados são cada vez mais populares para encontrar padrões e criar modelos de predição. E com o aumento de dados em linguagem natural, a mineração de texto é uma área que cresce cada vez mais. Alguns pontos que diferenciam esses trabalhos

são o domínio de aplicação, programação manual ou utilização de software de mineração, aprendizagem supervisionada ou não supervisionada, formato dos dados.

O trabalho de mestrado desenvolvido na Universidade Federal de Minas Gerais (UFMG) elaborado por Larissa dos Santos sob a orientação do professor Marcos Prates e professora Erica Rodrigues [7] utiliza mineração de texto para estudar a ocorrência de crimes, por localização geográfica, data e hora a partir de dados de *tweets* (tipo de mensagens utilizado no twitter) postados. Este trabalho utiliza técnicas similares ao trabalho apresentado nesta monografia. Diferenças são o tipo de texto empregado: *tweets* geralmente utilizam expressões coloquiais, enquanto no domínio militar existe um vocabulário mais formal e próprio das forças armadas. Além disso, a forma de criação do conjunto de treinamento no trabalho [7] foi feito através de uma classificação manual, enquanto nesse trabalho foi definido um dicionário de termos relativo as categorias de incidentes, sendo comparado com a base de dados rotulando cada incidente conforme a categoria do dicionário.

Outro trabalho com objetivos similares de Segurança é o trabalho desenvolvido no mestrado de Albert II Pak na Universidade de Brasília (UnB) [31]. O autor utiliza técnicas de mineração de texto para categorizar eventos de incidentes reportados por e-mail ao Centro de Tratamento de Incidentes de Segurança em Redes de Computadores da Administração Pública Federal - CTIR Gov. O CTIR Gov é subordinado ao Departamento de Segurança da Informação e Comunicações (DSIC) do Gabinete de Segurança Institucional da Presidência da República (GSI/PR) e tem a finalidade de atender os incidentes em redes de computadores da Administração Pública Federal. Nesse trabalho, também são comparadas as técnicas de *Naive Bayes*, Árvore de Decisão J48 e Máquina de Vetor de Suporte. Uma diferença, porém, é que para escolha de parâmetros de pré-processamento, foi utilizada uma ferramenta auxiliar chamada *Pre-Text*, desenvolvida no Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo - ICMC/USP [32].

Os trabalhos [7], [31] e o trabalho apresentado nesta monografia comparam as mesmas técnicas de aprendizado de máquina: *Naive Bayes*, SVM e Árvore de Decisão. Outro ponto de semelhança também é que nos projetos [7], [31] e no projeto desenvolvido nesta monografia foram consideradas as medidas de precisão, *recall* e Medida F para avaliação de classificadores. O tamanho das bases de dados, porém, variou entre os projetos. No trabalho [7], trabalhou-se com bancos de dados com 5.000 registros, e em alguns cenários, com um banco de dados de 10.000 registros. No trabalho [31], apesar de possuir, inicialmente, uma base dados de 102.260 *e-mails*, após balanceamentos na amostra e limitações computacionais, como tempo de processamento no *Pre-Text* e limitação de memória para trabalhar com o WEKA, utilizou-se conjuntos de dados com 2.400 mensagens em média. No trabalho desenvolvido nesta monografia, utilizou-se os incidentes da Copa das Confederações 2013, totalizando 811 registros.

No trabalho [7], foi utilizada a linguagem *R* para realização dos experimentos, enquanto no trabalho [31] e no trabalho desenvolvido nesta monografia foi utilizada a ferramenta de mineração WEKA. O trabalho [7] também utiliza a linguagem *R* para criação de uma interface e obtenção de estatísticas sobre a dinâmica de ocorrência de crimes. No projeto desenvolvido nesta monografia, a interface foi feita utilizando a linguaguem *Java* e as estatísticas obtidas com a criação de um programa auxiliar utilizando a linguagem *C++*. No trabalho [7], a interface criada disponibiliza mais gráficos do que a interface do projeto desenvolvido, inclusive uma aba da interface foi criada a partir de informações de georreferenciamento de *tweets* postados.

Dessa forma, foi apresentado toda a referência teórica para o entendimento do projeto desenvolvido. Após o estudo dos conceitos de aprendizagem automática, processamento e mineração de texto, classificação de texto, ferramenta WEKA e sistema Pacificador, é possível prosseguir para a modelagem do problema estudado neste trabalho.

Capítulo 3

Proposta de Solução

Neste Capítulo é apresentada a proposta de elaboração do módulo de classificação automática de incidentes para o sistema Pacificador, conforme os passos metodológicos expostos na Seção 1.5. Na Seção 3.1, é relembrado o contexto do trabalho. Na Seção 3.2 é descrito o modelo conceitual do projeto, com a descrição das etapas seguidas na realização dos experimentos e resultados esperados. Na Seção 3.3 é descrita a metodologia de trabalho. Finalmente, na Seção 3.4 a interface desenvolvida é apresentada, detalhando as tecnologias utilizadas e apresentando telas de utilização.

3.1 Contextualização

Primeiramente, é importante relembrar o objetivo geral do trabalho: desenvolver um módulo de classificação automática de incidentes em grandes eventos, que sirva de apoio a decisão para o sistema Pacificador. Quando um agente móvel, registra um incidente no Pacificador, esse é marcado no mapa interativo do sistema. Esse dado é analisado pelos comandantes da operação, para decidir como proceder, ou seja, quais ações tomar. O módulo tem uma interface que permite a classificação automática do incidente registrado em uma categoria pré-estabelecida, além de conter informações estatísticas sobre a dinâmica daquele tipo de incidente. Essas informações ajudam no gerenciamento de incidentes em tempo real. Além disso, contribuem para o planejamento de próximas operações.

Para criar esse módulo, registros referente a Copa das Confederações (2013) do banco de dados do Pacificador foram obtidos. Esses registros possuem dados referentes a relatos e incidentes, que foram registrados durante as operações que monitoravam a segurança dos jogos. Esses dados obtidos possuem informações sobre a data e hora do incidente ou relato, assim como sua descrição textual, coordenadas de onde ocorreu e qual a central que o monitora. Dessa forma, é possível obter estatísticas tais como: em qual cidade incidentes de determinado tipo têm maior ocorrência, em quais horários e dias da semana.

Inicialmente, foi realizado um experimento com técnicas de clusterização (aprendizagem não-supervisionada) para tentar detectar automaticamente agrupamentos (categorias) nos dados desses incidentes. Porém, os resultados obtidos não foram satisfatórios (ver Seção 4.2). Dessa forma, os experimentos seguintes adotaram a abordagem supervisionada e a proposta de solução reflete essa escolha.

Para utilizar as técnicas de aprendizagem supervisionada, deve-se definir de antemão as categorias do contexto. Originalmente, o exército possui uma divisão em 76 possíveis

categorias. Porém, essa divisão é muito abrangente e sub-utilizada. Definiu-se, junto ao especialista, categorias mais abrangentes a partir dessas 76: credenciamento, criminalidade, infra-estrutura, logística, manifestação, saúde, terrorismo e trânsito. Como a amostra contém apenas 811 incidentes, este projeto utilizou inicialmente duas categorias ("Manifestação" e "Trânsito"), pois são as categorias de maior expressividade da base de dados (verificado após uma análise manual dos registros). Havendo mais amostras de incidentes, seria possível melhorar o classificador adicionando novas categorias, seguindo passos semelhantes aos expostos neste Capítulo.

3.2 Modelo Conceitual

A Figura 3.1 apresenta as etapas da Proposta de Solução do trabalho, as quais serão detalhadas nas Seções 3.2.1 a 3.2.5.

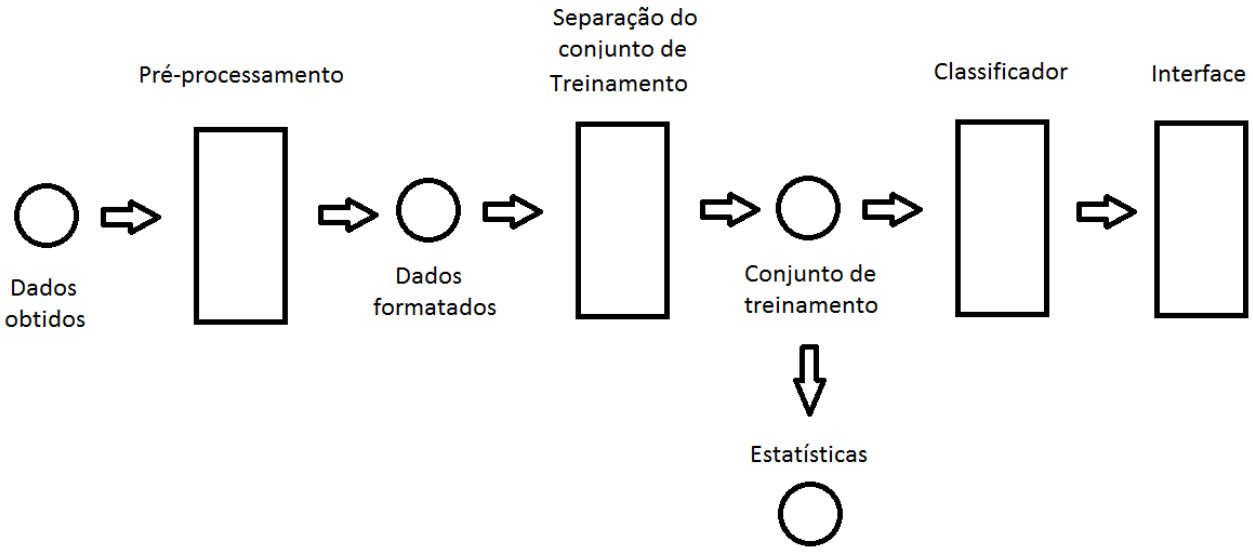

Figura 3.1: Etapas da Proposta de Solução.

3.2.1 Preparação dos Dados

Inicialmente, tem-se os dados dos incidentes em arquivos individuais, porém estes possuem muita informação acessórias à classificação baseada em descrições de incidentes. Portanto, antes de classificá-los, esses arquivos foram tratados, de forma que a parte acessória fosse removida (informações como identificadores, agente que submeteu o incidente e fotos, por exemplo, não são interessantes para a criação do conjunto de treinamento). Para isso, criou-se um programa utilizando a linguagem *C++* que percorre os dados de incidentes coletando apenas as informações do campo "descrição". Dessa forma, tem-se arquivos somente com a informação desejada para a criação do conjunto de treinamento. Para a obtenção de estatísticas, o mesmo programa busca as informações de "horário" e "cidade" e as adiciona no arquivo com a descrição.

Com os dados tratados, segue-se para a seguinte etapa de criação dos dados de treinamento, que é um grupo de dados já previamente dispostos nas categorias pré-definidas ("Manifestação" e "Trânsito").

3.2.2 Criação de Conjunto de Treinamento

Para criar o classificador utilizando técnicas de aprendizagem supervisionada, é necessário que exista um conjunto de treinamento, ou seja, uma amostra da base de dados já classificada para que o algoritmo consiga criar um modelo de generalização.

Para a obtenção desse conjunto, pode-se classificar os dados manualmente ou automaticamente. Classificar manualmente seria desnecessário e demorado, pois os dados seriam verificados por pessoas e rotulados (de acordo com sua categoria) de um em um. Então, a classificação da base de treino foi feita utilizando dicionários de palavras (um para cada categoria) para automatizar a criação da amostra. Primeiro, foi criado um dicionário contendo os termos mais característicos de cada categoria. Um programa desenvolvido em *C++* varre a base de incidentes à procura daqueles que possuam as palavras chave definidas nesse dicionário, separando os incidentes na categoria referente ao mesmo, criando uma amostra de treinamento.

Após essa separação automática, o conjunto resultante deve ser verificado manualmente, para averiguar se os dados foram divididos da forma esperada de acordo com sua categoria.

A partir da criação de uma amostra de treinamento, percorre-se o resto dos dados (que não foram coletados pelo programa com o primeiro dicionário) selecionando o dicionário da próxima categoria, ou seja, cada categoria é criada utilizando-se os dados complementares as categorias previamente selecionadas. Dessa forma, foram criadas as amostras das categorias de "Manifestação" e "Trânsito". Para a categoria de "Manifestação", a partir de um dicionário contendo 5 termos tais como "manifest", "marcha" e "protest" obteve-se uma amostra com 187 incidentes. Com os 624 incidentes restantes, e utilizando um dicionário com termos característicos tais como "comboio", "trafego" e "veiculo", obteve-se a amostra para "Trânsito" contendo 81 incidentes.

3.2.3 Verificação de Algoritmos de Aprendizagem

Após a criação do conjunto de treinamento, foi utilizado a ferramenta de mineração WEKA (vide Seção 2.2.5) para testar os métodos de aprendizagem Naive Bayes, SVM e Árvore de Decisão. Esses serão comparados através de medidas que indicam a capacidade do método de predizer novas entradas. Os algoritmos serão comparados e os que obtiverem melhor resultado serão utilizados no classificador final desse projeto. Os parâmetros para a comparação são a Medida F e as curvas ROC de cada método (vide Seção 2.2.3). Esses experimentos estão detalhados no Capítulo 4.

3.2.4 Obtenção de Estatísticas

Com as categorias e o modelo de classificador definidos, foram obtidas estatísticas sobre cada categoria, para fornecer as informações desejadas para o auxílio na tomada de decisão.

Para melhor compreensão das dinâmicas dos incidentes, pensou-se em obter as estatísticas referentes ao horário de notificação da ocorrência e localização. Dessa forma, o mesmo programa citado na Seção 3.2.1 utilizado para separar a descrição de cada incidente é utilizado, agora, para obter somente dados sobre horário de notificação e localização (região) de cada ocorrência de incidente. Dessa forma, é possível saber em quais cidades ocorreram mais incidentes e em qual horário do dia os incidentes foram mais frequentes

3.2.5 Interface de Utilização

Após todos os testes e experimentos com os diferentes modelos de classificadores obtidos, foi utilizada a API do WEKA para a criação de uma interface de utilização do modelo obtido. Esta API permite ao programador utilizar as mesmas funções da interface de modo a embutir suas funcionalidades em qualquer aplicação conforme exposto na Seção 2.2.5. Dessa forma, o modelo obtido foi inserido dentro de uma interface, que possibilita o operador do Pacificador obter imediatamente uma classificação automática para o novo incidente inserido. A partir da detecção da categoria, estatísticas relevantes sobre a dinâmica do incidente são recuperadas. Estas informações são exibidas na tela em formato de gráficos para que o entendimento daquele tipo de incidente seja rápido e intuitivo.

3.3 Metodologia do Trabalho

Com o objetivo de desenvolver o classificador automático de incidentes, foram seguidas as seguintes etapas, ilustradas na Figura 3.2:

- Etapa 1: Discussão com especialistas da área militar para delinear categorias de interesse e dicionários de termos para essas categorias. A criação de dicionários foi necessário para criar um conjunto de treinamento que foi utilizado nos experimentos com métodos de aprendizagem supervisionados. Assim, os arquivos da base de dados do sistema Pacificador referentes à Copa das Confederações 2013 contendo descrições textuais de incidentes foi percorrida comparando com os dicionários definidos. Essa forma de definir o conjunto de treinamento não deixa de ser uma classificação automática, porém limitada, pois é feita apenas uma comparação de *strings* específicas.
- Etapa 2: Essa fase envolve a transformação do conjunto de dados de trabalho em um arquivo (ARFF) que possa ser manipulado pelo WEKA. Após carregar o arquivo ARFF no WEKA, envolve a aplicação do filtro *StringToWordVector* para transformar o vocabulário de termos contidos nas descrições de incidentes em um vetor de palavras cujo valor em cada índice é um fator de comparação (ver Seção 2.2.4 para relembrar ponderações possíveis para esse vetor de palavras).
- Etapa 3: Aplicação das técnicas de aprendizagem e posterior comparação dessas técnicas avaliando as medidas de precisão, recall e Curvas ROC.
- Etapa 4: Desenvolvimento da interface utilizando as funções fornecidas pela API do WEKA. Com essa API, são seguidas as mesmas etapas seguidas no experimentos.

Ou seja, transforma-se os arquivos de texto em um arquivo ARFF, aplica-se o filtro *StringToWordVector* e por fim, são disponibilizados os modelos de classificadores com as melhores configurações avaliadas nos experimentos.

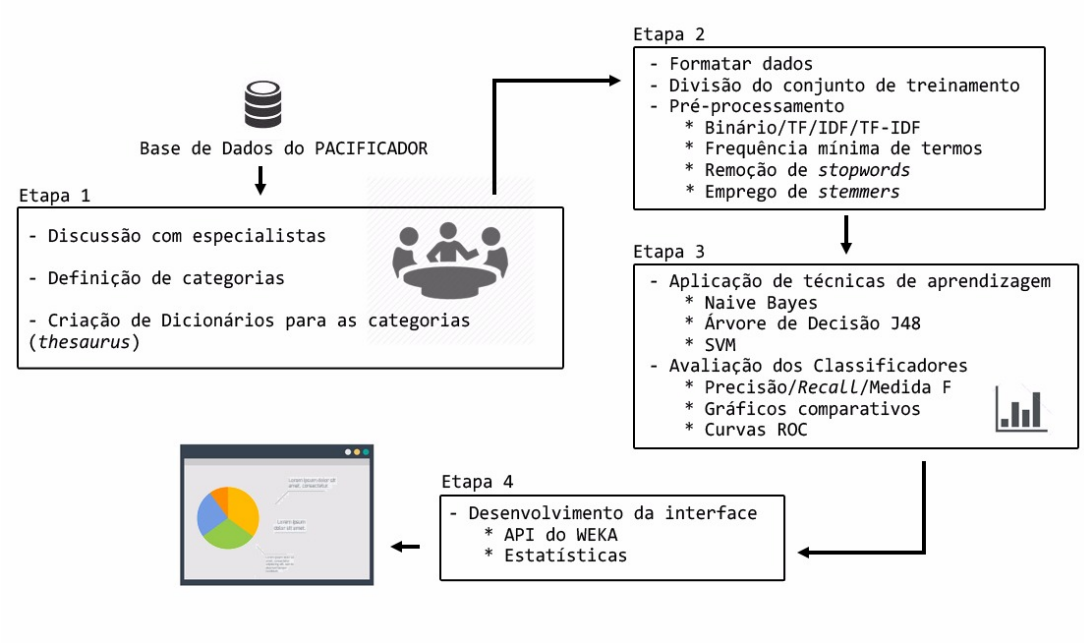

Figura 3.2: Metodologia do trabalho.

3.4 Interface

Nesta Seção será detalhado como funciona a interface desenvolvida, que reúne o classificador automático de incidentes e estatísticas associadas. Na Seção 3.3.1 serão expostas as tecnologias utilizadas para o desenvolvimento da interface, incluindo alguns pontos sobre a API do WEKA. Na Seção 3.3.2, é ilustrada a utilização da interface desenvolvida com algumas telas.

3.4.1 Tecnologias Utilizadas

A Interface foi desenvolvida utilizando a IDE Eclipse Luna (v4.4.1) com o pacote do WEKA versão 3.8.0, para a criação do modelo de classificador dos dados. Foi utilizado também a biblioteca gráfica *Swing*, para criação da janela da interface, captura de *inputs* e apresentação de resultados. Essa biblioteca foi escolhida por ser de fácil utilização, já que o Eclipse possui ferramentas que facilitam a criação de uma interface utilizando *Swing*. Finalmente, foi utilizado o pacote *Snowball Stemmer* [33] para obter os radicais das palavras na fase de pré-processamento.

Após a análise dos resultados obtidos nos experimentos detalhados no Capítulo 4, foi possível identificar quais são as melhores configurações para o modelo de classificador de incidentes (melhor método de aprendizagem, frequência mínima de termos e ponderação

no vetor de palavras). Para a categoria de "Manifestação", a melhor configuração foi SVM com ponderação TF, TF-IDF ou Binário com frequência mínima de ocorrência de termos 5. Para a categoria de "Trânsito", a melhor configuração foi SVM com ponderação TF-IDF e frequência mínima 3 (esses resultados serão explicados na seção 4). Para tornar o modelo re-utilizável de tal forma que o usuário não precise conhecer o WEKA e configurá-lo, foi necessário utilizar a API do WEKA para embutir o classificador na interface. Esta API fornece funções que podem ser utilizadas em código Java, possibilitando ao programador contar com todas as funcionalidades presentes na interface gráfica do WEKA.

Com a API do WEKA, foram seguidos todos os passos para criação do modelo da forma como estava sendo criado no modo *Explorer*. Ou seja, primeiro carregou-se os arquivos ARFF, depois aplicou-se o filtro de *StringToWordVector* com as configurações de pré-processamento relativas aos melhores resultados obtidos nos experimentos. Então, criou-se o modelo de classificação. Para testar um novo incidente (texto fornecido pelo usuário), foi necessário criar uma nova instância cujo valor é o texto inserido pelo usuário.

Para criar a interface com o usuário, a biblioteca gráfica *Swing* foi utilizada, com auxílio de uma ferramenta do Eclipse chamada *Window Builder* (possibilita utilização de funcionalidades visuais - clica e arrasta - para ajudar a criação da interface). Com essa biblioteca, é possível criar janelas com botões, imagens, e também é possível receber *inputs* de texto do usuário. Esses inputs são recebidos no formato de descrição textual de um incidente digitado pelo usuário na interface.

O texto inserido na interface serve de entrada ao modelo de classificador, e a interface mostra a previsão de categoria retornada pelo modelo.

Para cada categoria, exibe-se também imagens contendo estatísticas de horário de notificação e região de ocorrência dos incidentes.

3.4.2 Visualização

As Figuras 3.3, 3.4 e 3.5 ilustram como a interface funciona, cada uma para um tipo possível de classificação. A primeira mostra o resultado de um texto classificado em "Manifestação". A segunda, para um texto classificado como sendo da categoria de "Trânsito". A terceira imagem mostra um texto cujo classificador não detectou como sendo de uma das duas categorias definidas. Para cada categoria definida, a interface traz estatísticas sobre a o horário de notificação e sobre a região de ocorrência do incidente.

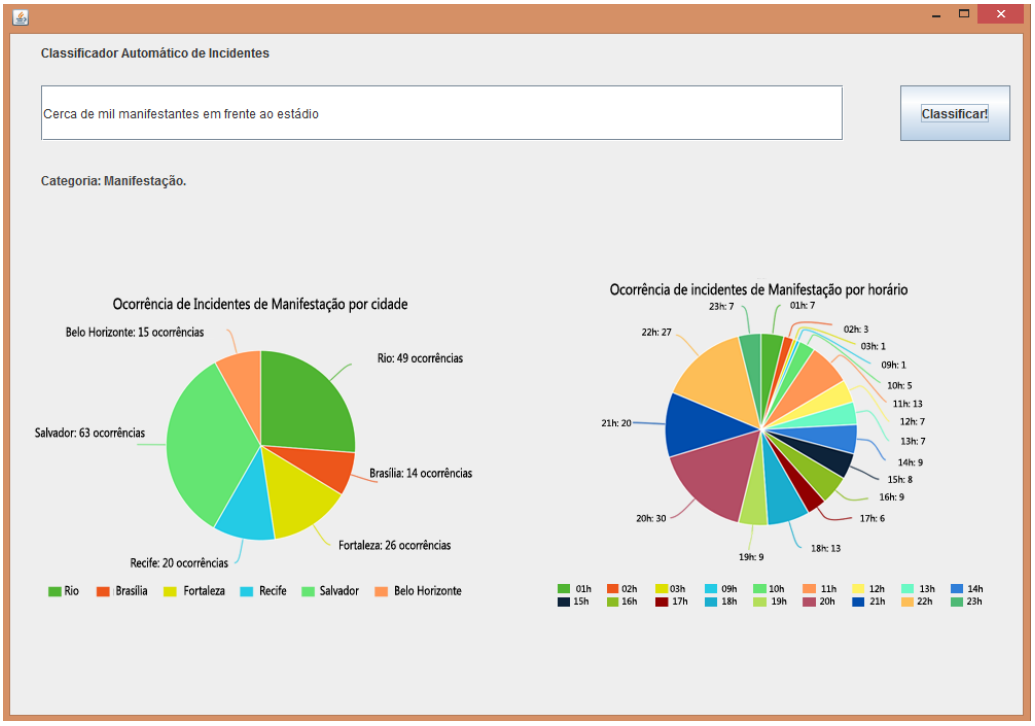

Figura 3.3: Imagem da execução do programa para um incidente classificado como "Manifestação".

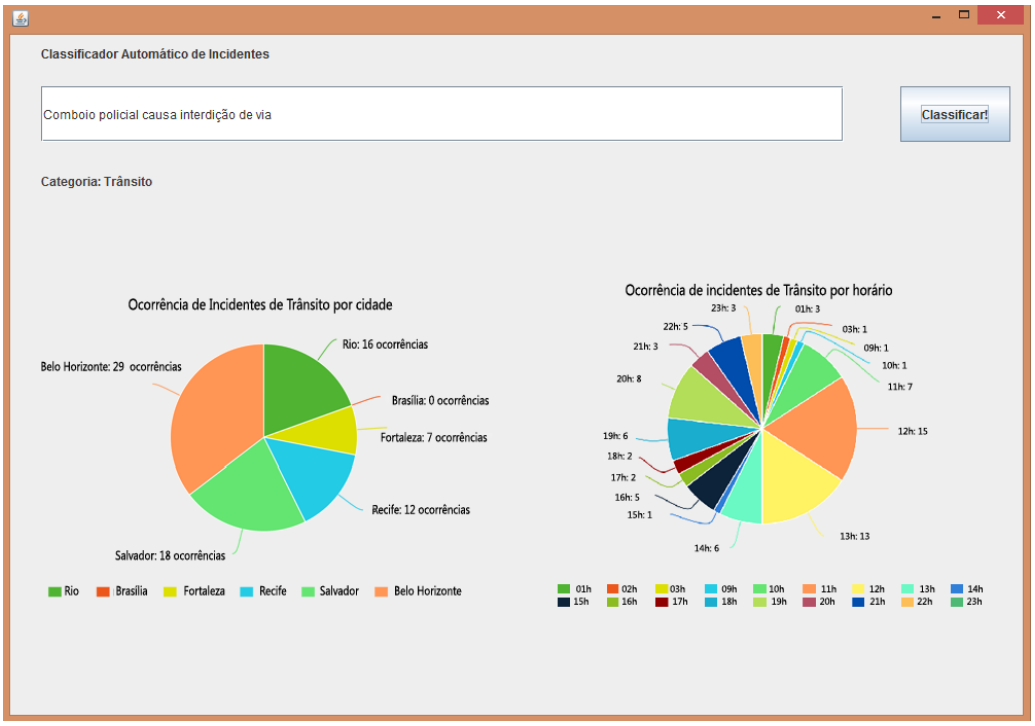

Figura 3.4: Imagem da execução do programa para um incidente classificado como "Trânsito".

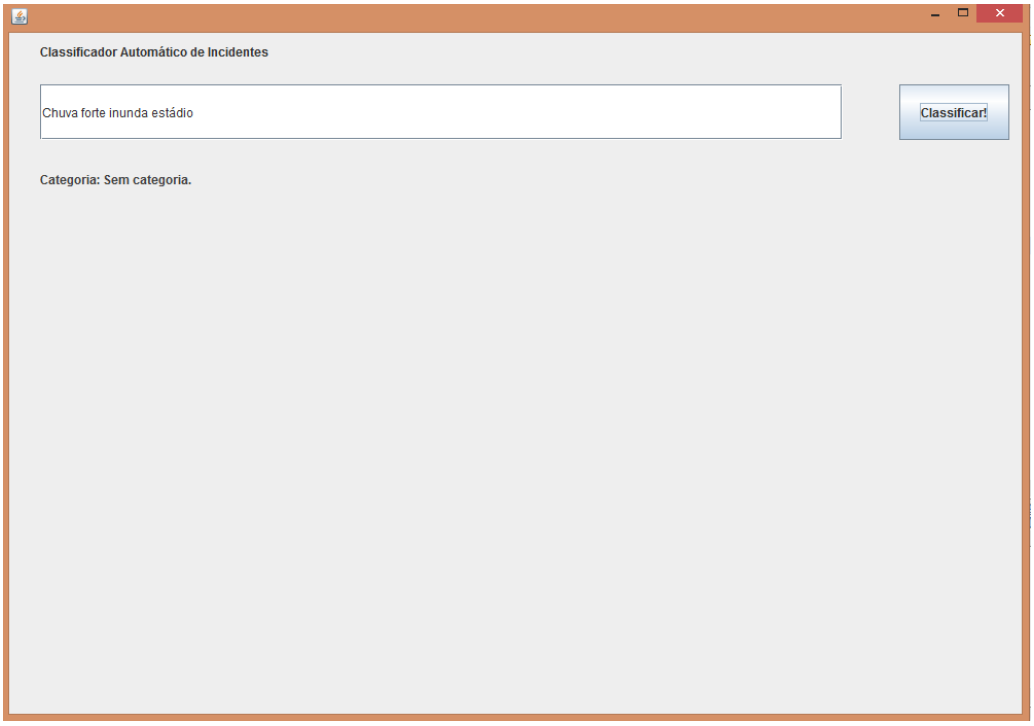

Figura 3.5: Imagem da execução do programa para um incidente classificado como “Sem Categoria”.

Capítulo 4

Experimentos

Neste capítulo são detalhados os experimentos realizados e resultados obtidos, seguindo a proposta de solução apresentada no Capítulo 3 e a respectiva metodologia do trabalho apresentada na Seção 1.5.

4.1 Ambiente de Teste

O ambiente de teste utilizado para validar a proposta apresentada no Capítulo 3 inclui os seguintes itens:

- Configurações de **Hardware**:
  - Sistema Operacional Ubuntu 13.04
  - 2GB de Memória RAM
  - CPU AMD E-450 1.65GHz
- Configurações do **Software**:
  - WEKA Versão 3.8.0
  - *Snowball Stemmer Portuguese*
- Configurações dos **Arquivos de Teste**:
  - Evento: Copa das Confederações 2013
  - Total de 811 Incidentes

Pela configuração padrão da ferramenta WEKA, o limite de memória RAM alocado para a utilização da interface é limitado. Essa limitação pode ser um problema dependendo da quantidade de registros disponíveis. Mas, com a quantidade de incidentes desse projeto (811 incidentes) não foi necessário alterar essa configuração.

4.2 Experimento 0: Clusterização

Com o objetivo de tentar detectar automaticamente categorias de incidentes no grupo total de 811 arquivos, inicialmente testou-se a técnica de clusterização. Utilizando essa técnica, espera-se que os dados sejam divididos em grupos com características semelhantes. Ou seja, o objetivo era detectar classes de incidentes semelhantes. O WEKA também foi utilizado para realizar esse experimento.

O método utilizado foi o *K-Means* e esse método requer que seja definido de antemão a quantidade de grupos que serão criados. Como inicialmente pensou-se em trabalhar com mais categorias, escolheu-se de forma empírica o valor de 3 categorias. Pelo tamanho da amostra, um número maior seria improvável, e essa parecia uma quantidade razoável de classes de incidentes. Segue uma descrição mais detalhada do experimento:

- Clusterização com método K-Means com 3 Categorias ($K = 3$).
- Peso TF-IDF para ponderação de palavras.
- Frequência mínima de ocorrência de termo = 1.
- *Snowball Stemmer Portuguese*

Após a aplicação dos filtros, havia um total de 1623 termos (palavras) diferentes para os 811 incidentes. Para testar a clusterização, utilizou-se incialmente o próprio conjunto de treinamento como conjunto de teste e, em um segundo cenário, utilizou-se uma divisão do conjunto de dados sendo 60% dedicado ao conjunto de treinamento e 40% para o conjunto de teste (vide Seção 2.2.5 para as formas de teste no WEKA).

A Tabela 4.1 apresenta os resultados desse experimento inicial.

Tabela 4.1: Resultado da Clusterização.

	Atribuição de Cluster		
Opção de Teste	Cluster 0	Cluster 1	Cluster 2
Training Set	0.97	0.03	0.00
Percentage Split 60%	0.01	0.99	0.00

O resultado que aparece na tabela não é bom, pois no primeiro cenário 97% dos dados foram atribuídos a apenas um *cluster*, e 3% para o segundo *cluster*, e nenhum dado atribuído ao terceiro agrupamento. O segundo cenário demonstra que a maior parte dos incidentes (99%) é atribuída à apenas um agrupamento e 1% ao primeiro agrupamento, não sendo atribuído nenhum dado ao terceiro *cluster*. Esses resultados demonstram que o algoritmo não detectou claramente categorias no conjunto de dados. Dessa forma, decidiu-se continuar os próximos experimentos utilizando a abordagem dos métodos supervisionados.

4.3 Experimento 1: Categoria Manifestação

Após uma análise inicial manual dos dados, escolheu-se a categoria de "Manifestação" para ser a primeira categoria de experimentos. Essa escolha se deu pelo fato de que "Manifestação" é uma categoria com mais expressividade nos registros possuindo um conjunto

de treinamento maior e com termos mais característicos. Assim, é provável que se consiga generalizar uma boa solução de predição.

Conforme exposto na Proposta de Solução do Capítulo 3, trabalhou-se com categorias binárias. Nesse experimento, temos então, duas categorias: "Manifestação" e "Não Manifestação".

O conjunto de treinamento da categoria "Manifestação" foi obtido, através do dicionário da Tabela 4.2, da forma descrita no Capítulo 3, sendo obtidos um total de 187 arquivos para a categoria de "Manifestação", restando 624 arquivos para a categoria de "Não Manifestação". O total desses arquivos da amostra possuiam 1622 termos distintos.

Tabela 4.2: Dicionário para a categoria Manifestação.

manifest	Manifest	MANIFEST
marcha	Marcha	MARCHA
passeata	Passeata	PASSEATA
protest	Protest	PROTEST
turb	Turb	TURB

Com as amostras definidas, utilizou-se o WEKA para testar os algoritmos de aprendizagem. Com objetivo de avaliar os métodos de aprendizagem supervisionada *Naive Bayes*, Máquina de Vetores de Suporte e Árvore de Decisão, serão comparadas suas Medidas F ponderadas pela quantidade de instâncias de cada classe. Conforme exposto na Seção 2.2.3, a Medida F avalia em conjunto a precisão e o revocação (*recall*) do método. Os valores em Medida F registrados estão em formato decimal variando entre 0 e 1, e, dessa forma, um valor de 0.91, por exemplo, representa uma porcentagem de 91% para a Medida F. Essa medida só atinge um valor alto se o valor de precisão e de revocação forem altos.

Os métodos testados foram o *Naive Bayes*, Árvore de Decisão (J48) e SVM com otimização SMO (ver Seção 2.1.2). Conforme explicado na Seção 2.2.5, para cada método foram aplicadas as ponderações TF, IDF, TF-IDF e Binário (vide Seção 2.2.4 para o significado dessas ponderações), e a frequência mínima (quantidade mínima de ocorrência dos termos no conjunto total para que seja considerada) das palavras foi testada com os valores de 1, 3 e 5. Não foram testados os valores de frequência mínima 2 e 4 pois preferiu-se adotar um intervalor maior entre esses valores. Valores maiores que 5 também não foram testados pois restringiriam demais o vocabulário. A forma de teste utilizada foi *Cross Validation*, com divisão de 10 partições dos dados, conforme explicado na Seção 2.2.5.

As Tabelas 4.3, 4.4 e 4.5 apresentam os resultados obtidos com o Experimento 1 da Categoria "Manifestação".

A Tabela 4.3 apresenta os valores da Medida F do método *Naive Bayes* com ponderações TF, IDF, TF-IDF e Binário para a categoria "Manifestação". A Figura 4.1 apresenta o gráfico comparativo elaborado a partir desses valores. É possível perceber que a ponderação Binário teve o melhor desempenho dentre as ponderações testadas para todas as variações de frequência mínima, e teve seu maior valor da Medida F para as frequências mínimas 3 e 5 atingindo um valor de 0.910. Porém, para o peso IDF, a Medida F atinge seu maior valor com frequência mínima 1, e para TF e TF-IDF, com frequência mínima 3. Os resultados registrados são bons pois independente de restrição de vocabulário (variação de frequência mínima) e ponderação, os valores da Medida F ficaram acima de

90%, indicando que um classificador com estes parâmetros tem uma boa capacidade de predição da categoria de um novo incidente.

Tabela 4.3: Medida F com Naive Bayes para Manifestação.

		Frequência Mínima		
		1	3	5
PESO	TF	0.907	0.908	0.905
	IDF	0.907	0.903	0.904
	TF-IDF	0.907	0.908	0.905
	Binário	0.909	0.910	0.910

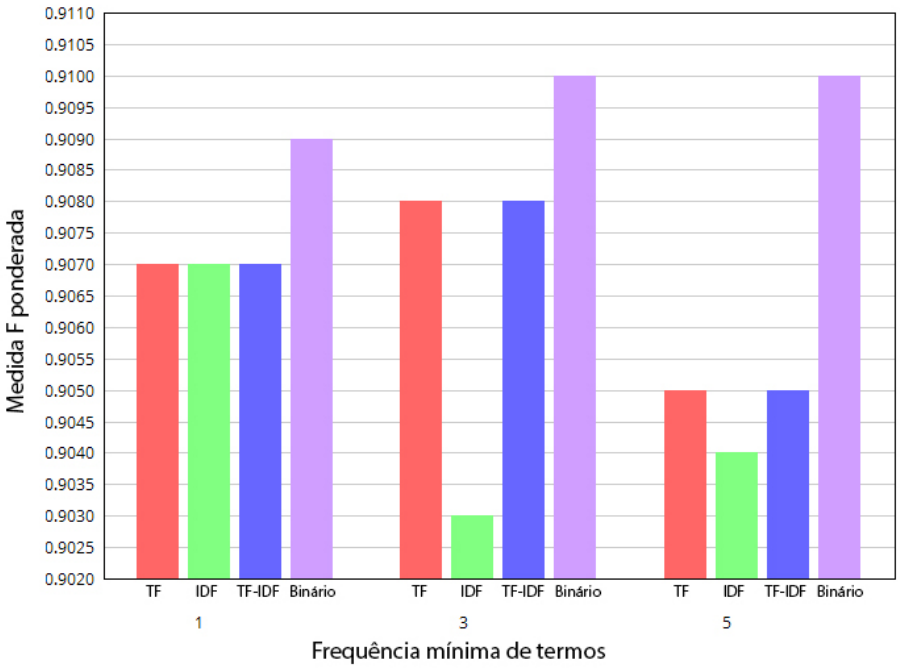

Figura 4.1: Comparação de valores da Medida F para as diferentes combinações de frequências mínimas e ponderações com método *Naive Bayes* para Manifestação.

A Tabela 4.4 apresenta os valores da Medida F do método de Árvore de Decisão para a categoria "Manifestação" com ponderações TF, IDF, TF-IDF e Binária. A Figura 4.2 apresenta o gráfico comparativo elaborado a partir desses valores. Note que em todos os testes foram obtidos os mesmos resultados. O resultado obtido com Árvore de Decisão (0.958) independentemente de ponderação ou frequência mínima de termo é maior que o obtido com a melhor combinação de parâmetros do método *Naive Bayes*, que alcançou para as frequências mínimas de termos 3 e 5 com ponderação Binário o valor de 0.91.

Tabela 4.4: Medida F com Árvore J48 para Manifestação.

		Frequência Mínima		
		1	3	5
PESO	TF	0.958	0.958	0.958
	IDF	0.958	0.958	0.958
	TF-IDF	0.958	0.958	0.958
	Binário	0.958	0.958	0.958

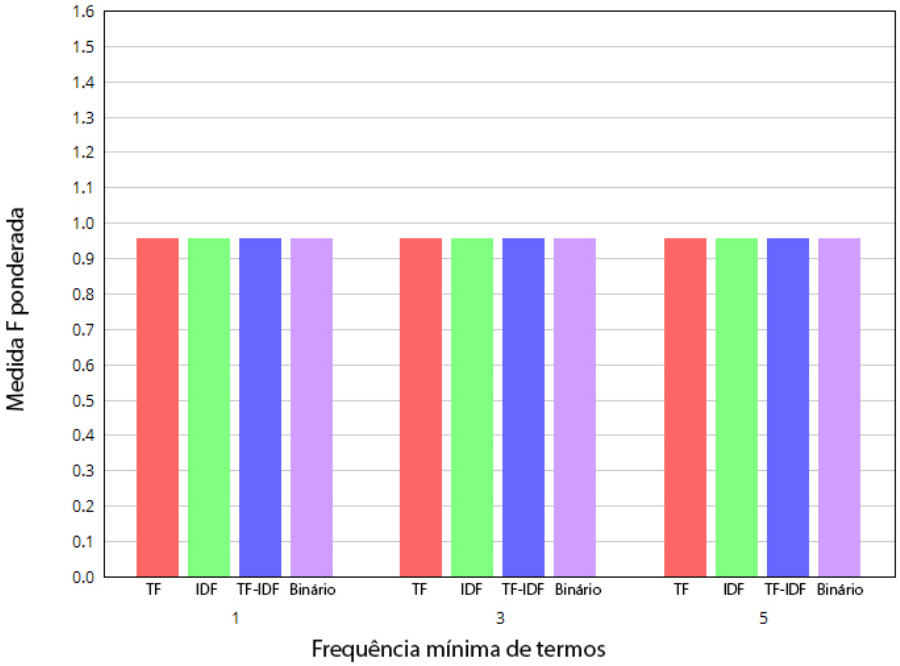

Figura 4.2: Comparação de valores da Medida F para as diferentes combinações de frequências mínimas e ponderações com método de Árvore de Decisão J48 para Manifestação.

A Tabela 4.5 apresenta os valores da Medida F do método SVM com ponderações TF, IDF, TF-IDF e Binário para a categoria "Manifestação". A Figura 4.3 apresenta o gráfico comparativo elaborado a partir desses valores. Note que os valores obtidos em todas as combinações de parâmetros foram maiores que os obtidos com o método *Naive Bayes*. E, à exceção da combinação de frequência mínima 1 e ponderação IDF que resultou em uma medida de 0.952, as outras combinações de parâmetros também resultaram em valores maiores que as do método Árvore de Decisão, que tinha como valor 0.958 em todas suas combinações de parâmetros. Pode-se observar que, com o aumento da frequência mínima, ou seja, restringindo mais o vocabulário, a Medida F do método SVM também aumenta. O maior valor (0.995) foi atingido com os parâmetros de frequência mínima 5, com os pesos TF, TF-IDF e Binário. Esse alto valor para a Medida F indica que um classificador com estes parâmetros tem uma ótima capacidade de predição de categorias de novos incidentes.

Tabela 4.5: Medida F com SVM para Manifestação.

		Frequência		
		1	3	5
PESO	TF	0.983	0.990	0.995
	IDF	0.952	0.978	0.988
	TF-IDF	0.983	0.990	0.995
	Binário	0.989	0.989	0.995

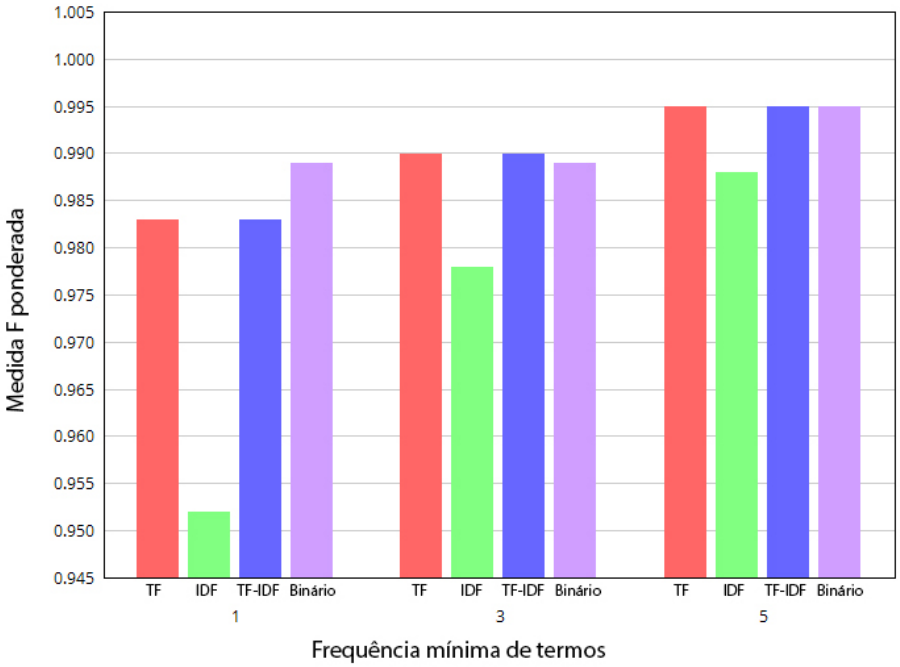

Figura 4.3: Comparação de valores da Medida F para as diferentes combinações de frequências mínimas e ponderações com método SVM para Manifestação.

4.4 Experimento 2: Categoria Trânsito

A partir da categoria de "Não Manifestação" (624 arquivos), partiu-se para a definição da segunda categoria. De forma similar, escolheu-se a categoria de "Trânsito" após uma análise manual dos dados. A amostra obtida representa 12% do conjunto de 624 arquivos, e apresenta termos característicos que facilitam a detecção de um incidente como sendo da categoria.

O conjunto de treinamento da categoria "Trânsito" foi obtido, através do dicionário da Tabela 4.6, da forma descrita no Capítulo 3, sendo obtidos um total de 81 arquivos para a categoria, restando 543 arquivos para a categoria de "Não Trânsito". O total desses arquivos possuiam 1233 termos distintos.

Tabela 4.6: Dicionário para a categoria Trânsito.

acidente	Acidente	ACIDENTE
engarrafa	Engarrafa	ENGARRAFA
engaveta	Engaveta	ENGAVETA
escolt	Escolt	ESCOLT
coli	Coli	COLI
comboio	Comboio	COMBOIO
comitiva	Comitiva	COMITIVA
congestiona	Congestiona	CONGESTIONA
desloca	Desloca	DESLOCA
interdit	Interdit	INTERDIT
trafego	Trafego	TRAFEGO
tráfego	Tráfego	TRÁFEGO
transito	Transito	TRANSITO
trânsito	Trânsito	TRÂNSITO
veiculo	Veiculo	VEICULO
veículo	Veículo	VEÍCULO
via	Via	VIA

De forma similar ao Experimento 1, utilizou-se o WEKA para testar os algoritmos de aprendizagem registrando as Medidas F ponderadas pela quantidade de instâncias de cada classe. Os valores em Medida F registrados estão em formato decimal variando entre 0 e 1, e, dessa forma, um valor de 0.91, por exemplo, representa uma porcentagem de 91% para a Medida F. Essa medida só atinge um valor alto se o valor de precisão e de revocação forem altos (vide Seção 2.2.3).

Os métodos testados foram o *Naive Bayes*, Árvore de Decisão J48 e SVM com otimização SMO (ver Seção 2.1.2). Para cada método, foram aplicadas as ponderações TF, IDF, TF-IDF e Binário (vide Seção 2.2.4 para o significado dessas ponderações), e a frequência mínima (quantidade mínima de ocorrência dos termos no conjunto total para que seja considerada) das palavras foi testada com os valores de 1, 3 e 5. A forma de teste utilizada foi *Cross Validation*, com divisão de 10 partições dos dados.

As Tabelas 4.7, 4.8 e 4.9 apresentam os resultados obtidos com o Experimento 2 - Categoria "Trânsito".

A Tabela 4.7 apresenta os valores da Medida F do método *Naive Bayes* com ponderações TF, IDF, TF-IDF e Binário para a categoria "Trânsito". A Figura 4.4 apresenta o gráfico comparativo elaborado a partir desses valores. Com esse gráfico, é possível notar que os valores da Medida F obtidos com a ponderação Binário foram melhores (0.918, 0.919 e 0.922) que as outras ponderações em todas as variações de frequência mínima, e os mesmos crescem conforme o aumento da frequência mínima de termos, ou seja, os valores crescem conforme se restringe mais o vocabulário. Interessante notar que os valores das Medidas F das ponderações TF e TF-IDF coincidem em todas as variações de frequência mínima. Os resultados registrados são bons pois independente de restrição de vocabulário (variação de frequência mínima) e ponderação, os valores da Medida F ficaram acima de 90%, indicando que um classificador com estes parâmetros tem uma boa capacidade de predição da categoria de um novo incidente.

Tabela 4.7: Medida F com Naive Bayes para Trânsito.

		Frequência Mínima		
		1	3	5
PESO	TF	0.902	0.904	0.905
	IDF	0.909	0.908	0.913
	TF-IDF	0.902	0.904	0.905
	Binário	0.918	0.919	0.922

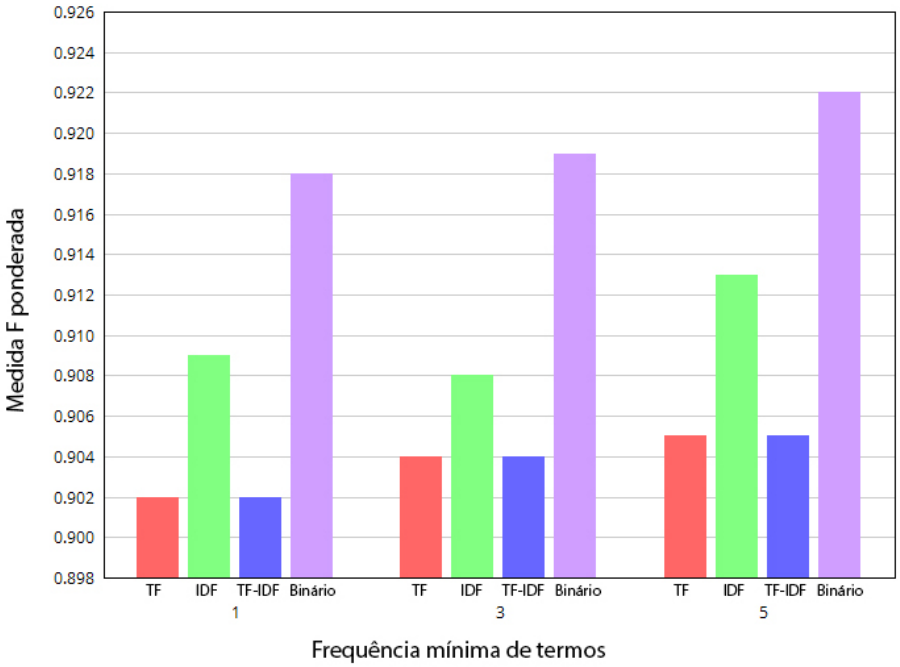

Figura 4.4: Comparação de valores da Medida F para as diferentes combinações de frequências mínimas e ponderações com método Naive Bayes para Trânsito.

A Tabela 4.8 apresenta os valores da Medida F do método de Árvore de Decisão J48 para a categoria "Trânsito" com ponderações TF, IDF, TF-IDF e Binário. A Figura 4.5 apresenta o gráfico comparativo elaborado a partir desses valores. Nesse caso, similar aos resultados da categoria "Manifestação", parte dos resultados obtidos foram iguais independentemente da variação de frequência mínima de termos. Para as frequências mínimas de termos 1 e 3, independentemente de variação de ponderação, o resultado obtido foi de 0.957, e para a frequência mínima 5, o resultado foi de 0.961. Logo, restringir o vocabulário aumentou o valor da Medida F para o método de Árvore de Decisão. Pode-se notar também que o menor valor obtido (0.957) é maior que o melhor valor obtido com o método *Naive Bayes*, que era de 0.922.

Tabela 4.8: Medida F com Árvore J48 para Trânsito.

		Frequência Mínima		
		1	3	5
P	TF	0.957	0.957	0.961
E	IDF	0.957	0.957	0.961
S	TF-IDF	0.957	0.957	0.961
O	Binário	0.957	0.957	0.961

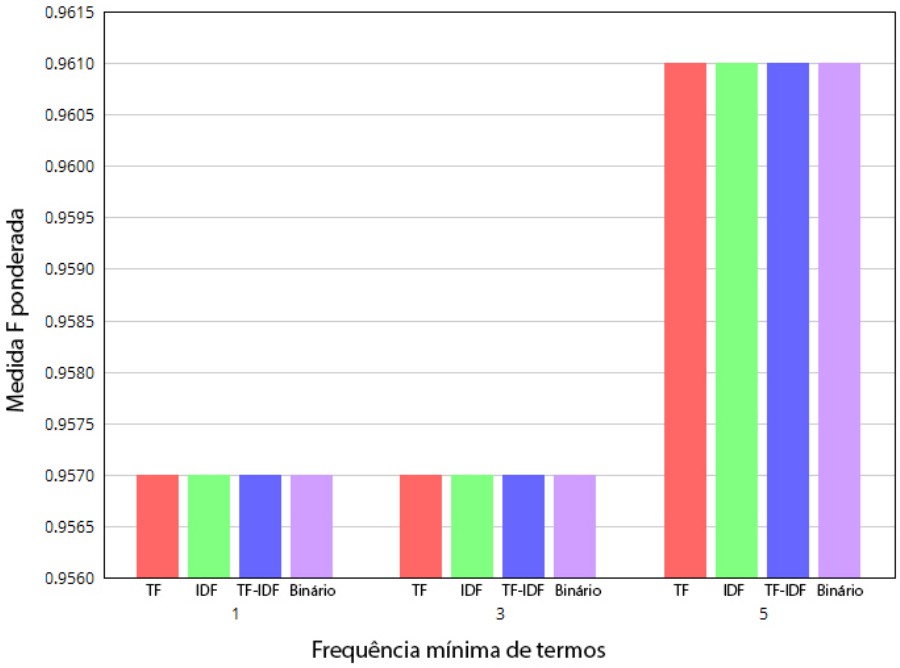

Figura 4.5: Comparação de valores da Medida F para as diferentes combinações de frequências mínimas e ponderações com método de Árvore de Decisão J48 para Trânsito.

A Tabela 4.9 apresenta os valores da Medida F do método SVM (com otimização SMO) para a categoria "Trânsito" com ponderações TF, IDF, TF-IDF e Binário. A Figura 4.6 apresenta o gráfico comparativo elaborado a partir desses valores. Assim como na categoria "Manifestação", o SVM teve os valores da Medida F maiores que os métodos *Naive Bayes* e a Árvore de Decisão J48. O menor valor obtido de 0.966 é maior que melhor valor da Árvore de Decisão que foi de 0.961. Diferente do experimento de SVM para a categoria de "Manifestação" que tinha os melhores resultados com a frequência mínima 5, nesse experimento o valor de frequência mínima 3 foi o que resultou em melhores Medidas F. Em particular, com frequência mínima 3 e utilizando o peso TF-IDF alcançou-se o maior valor dessa medida atingindo 0.987. Esse alto valor para a Medida F significa que o classificador com estes parâmetros possui um valor alto tanto de precisão quanto de revocação, indicando uma ótima capacidade de prever a categoria de um novo incidente.

Tabela 4.9: Medida F com SVM para Trânsito.

		Frequência Mínima		
		1	3	5
PESO	TF	0.975	0.985	0.966
	IDF	0.977	0.984	0.966
	TF-IDF	0.975	0.987	0.966
	Binária	0.975	0.982	0.968

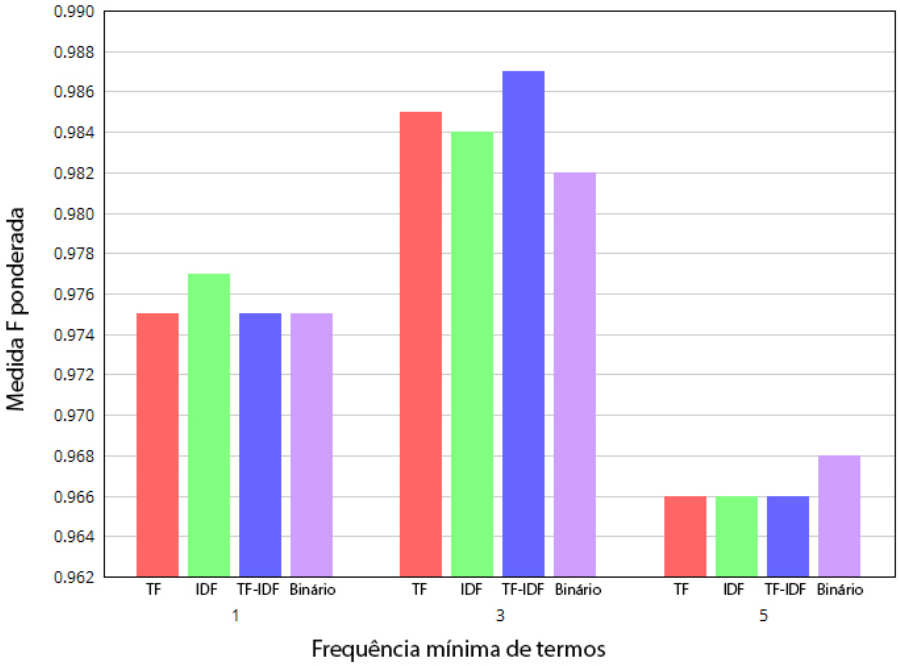

Figura 4.6: Comparação de valores da Medida F para as diferentes combinações de frequências mínimas e ponderações com método SVM para Trânsito.

4.5 Análise dos Resultados

Após obter os dados referentes aos resultados de cada método de aprendizado de máquina para cada categoria, é possível perceber qual o melhor algoritmo para cada caso sob a perspectiva da Medida F. A Figura 4.7 apresenta o gráfico com os resultados para a categoria "Manifestação", com os maiores valores da Medida F de cada método.

A Figura 4.8 compara o desempenho dos classificadores de Manifestação com a pior e a melhor Medida F. O gráfico à esquerda mostra a curva ROC para um classificador *Naive Bayes* com ponderação IDF e frequência mínima 3. O gráfico à direita mostra a curva ROC para um classificador utilizando o método SVM com ponderação TF-IDF e frequência mínima 5. Interessante notar que mesmo o classificador de menor Medida F, ainda possui um valor alto dessa medida, e um desempenho considerado bom sob a perspectiva da Área sob a Curva ROC. O classificador de melhor Medida F, em particular,

atinge um desempenho excelente sob a perspectiva da Área sob a Curva ROC. Um valor alto da medida Área sob a Curva ROC, conforme exposto na Seção 2.2.3, indica que o classificador possui um valor elevado de verdadeiros positivos, com uma taxa baixa de falsos positivos.

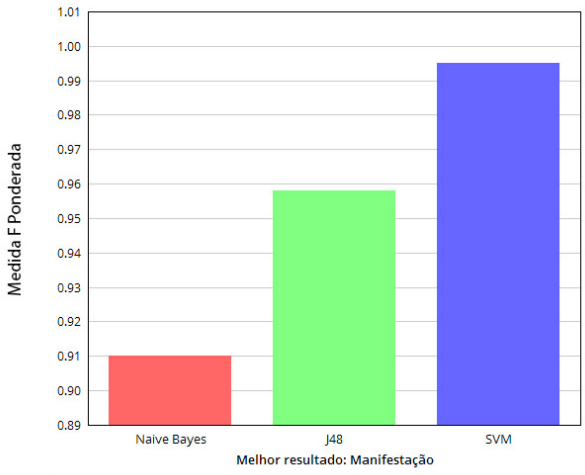

Figura 4.7: Comparação dos Métodos de Aprendizagem para a categoria de Manifestação.

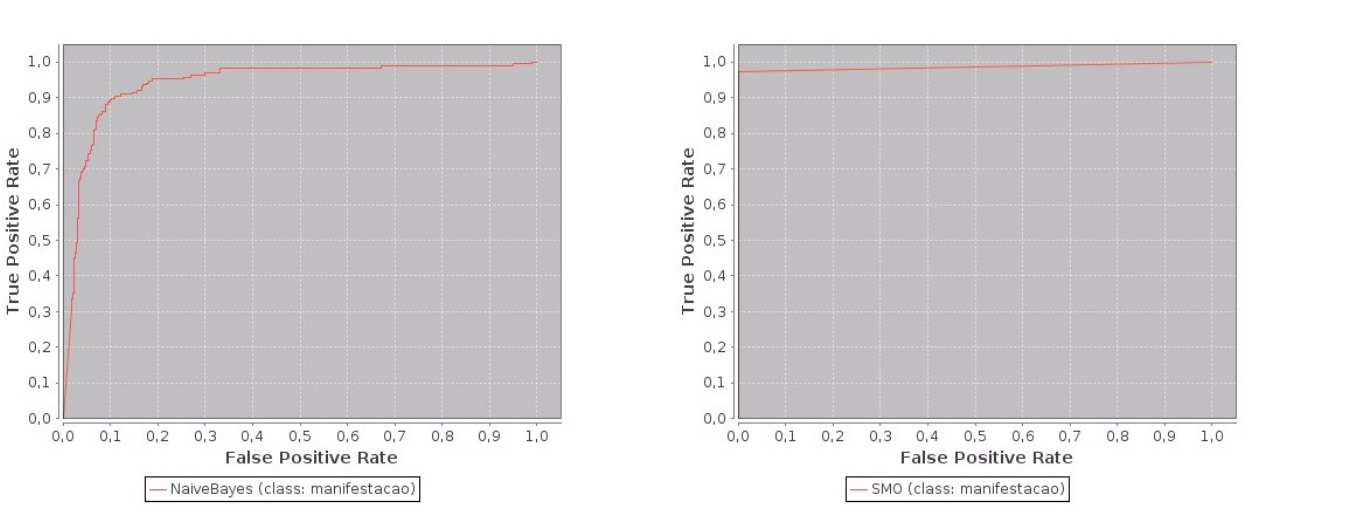

Figura 4.8: Comparação das Curvas ROC para a Categoria de Manifestação (Esquerda: Curva ROC para Configuração de menor Medida F. Direita: Curva ROC para Configuração de maior Medida F).

A Tabela 4.10 apresenta resultados mais detalhados relativos às configurações de melhor Medida F para a categoria Manifestação. Ou seja, as configurações de Medida F 0.995 com as configurações de ponderações TF-IDF, TF e Binária com frequência mínima de ocorrência de termo 5, utilizando o método SVM.

Tabela 4.10: Resultados dos Classificadores de maior Medida F para a Categoria de Manifestação.

	Expansão SVM para Manifestação		
	Precisão	Recall	Medida F
Manifestação	1.000	0.979	0.989
Não-Manifestação	0.994	1.000	0.997
Média Ponderada	0.995	0.995	0.995

A Figura 4.9 apresenta o gráfico comparativo da categoria "Trânsito". Da mesma forma, é possível notar que o método SVM teve melhores resultados, enquanto o método Naive Bayes, apresentou a menor Medida F.

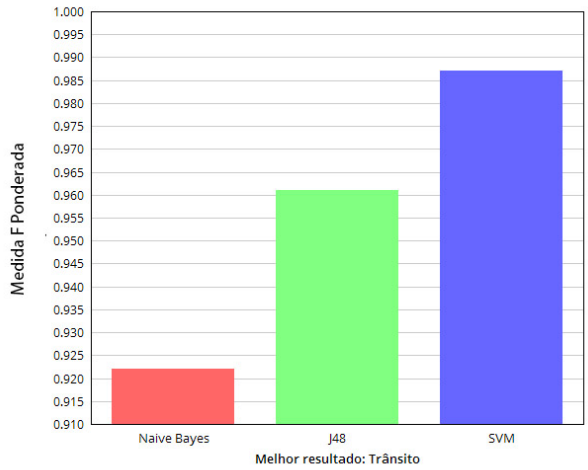

Figura 4.9: Comparação dos Métodos de Aprendizagem para a categoria de Trânsito.

A Figura 4.10 compara o desempenho do classificadores de Trânsito com pior/melhor Medida F. O gráfico à esquerda mostra a curva ROC para um classificador *Naive Bayes* com ponderação TF e frequência mínima 1. O gráfico à direita mostra a curva ROC para um classificador SVM com ponderação TF-IDF e frequência mínima 3. Interessante notar que mesmo o classificador de menor Medida F, ainda possui um valor alto dessa medida, e um desempenho considerado bom sob a perspectiva da Área sob a Curva ROC. O classificador de melhor Medida F, em particular, atinge um desempenho excelente sob a perspectiva da Área sob a Curva ROC. Um valor alto da medida Área sob a Curva ROC, conforme exposto na Seção 2.2.3, indica que o classificador possui um valor elevado de verdadeiros positivos, com uma taxa baixa de falsos positivos.

A Tabela 4.11 apresenta resultados mais detalhados relativos à configuração de melhor Medida F para a categoria "Trânsito", ou seja, com frequência mínima de ocorrência de termos 3, e ponderação TF-IDF, utilizando o método SVM.

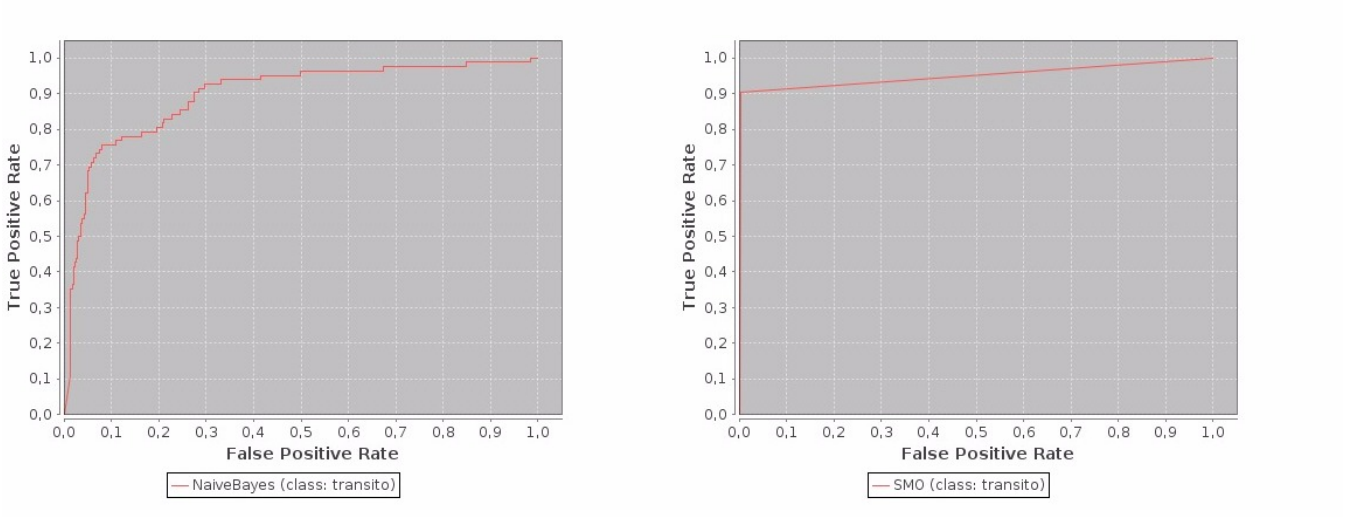

Figura 4.10: Comparação das Curvas ROC para a Categoria de Trânsito (Esquerda: Curva ROC para Configuração de menor Medida F. Direita: Curva ROC para Configuração de maior Medida F).

Tabela 4.11: Resultados do Classificador de maior Medida F para a Categoria de Trânsito.

	Expansão SVM para Trânsito		
	Precisão	Recall	Medida F
Trânsito	0.987	0.915	0.949
Não-Trânsito	0.987	0.998	0.993
Média Ponderada	0.987	0.987	0.987

De modo geral, os resultados obtidos com a categoria de "Manifestação" foram melhores que os resultados da categoria "Trânsito". Isso se deve ao fato de a primeira categoria possuir uma maior base de dados para a aprendizagem do que a segunda. A categoria de "Manifestação", representava 23% do total de seu conjunto (811 arquivos), enquanto a categoria "Trânsito", representava 12% do total de seu conjunto (624 arquivos). Além disso, a variação dentro da própria base de dados da categoria "Trânsito" é maior com um dicionário mais amplo, com 17 termos característicos, enquanto o dicionário da categoria de "Manifestação" tinha apenas 5 termos. Essa variação dificulta mais a criação de um modelo de predição.

Porém, tanto para a categoria de "Manifestação" quanto para a categoria de "Trânsito", as Medidas F apresentaram, em geral, ótimos valores. Ambas categorias apresentam termos muito característicos, facilitando para os métodos de aprendizagem a detecção de uma ou outra classe.

Interessante também observar que para ambas as categorias, o método SVM foi o método de melhor medida, variando, entretanto, os parâmetros de configuração. Para a categoria "Manifestação", o método SVM atingiu seu maior valor com a frequência mínima 5 e ponderações TF, TF-IDF ou Binário. Na Categoria "Trânsito", o SVM atingiu o melhor valor com frequência mínima 3 e ponderação TF-IDF.

Capítulo 5

Conclusões e Trabalhos Futuros

O Exército Brasileiro desenvolveu um sistema chamado Pacificador, para apoiar operações na área de Segurança Pública. Com esse sistema, amplia-se a *consciência situacional* dos eventos, sendo possível visualizar em um mapa interativo informações trazidas por agentes em campo. Dentre as informações que podem ser relatadas, destacam-se os incidentes, que são eventos não planejados que podem comprometer a segurança do evento. Esses incidentes podem trazer componentes textuais, gráficos e geográficos. Porém, não há nesse sistema uma divisão dos incidentes em categorias, o que traz prejuízos para uma análise estatística dos problemas encontrados em grandes eventos. Nesse sentido, neste projeto foram realizados experimentos para analisar a viabilidade de criação de uma solução de classificador automático de textos de incidentes, a partir do estudo de técnicas de aprendizado de máquina e mineração de texto.

Assim, neste trabalho, utilizou-se a ferramenta WEKA para estudar a aplicação de métodos de aprendizagem em textos de incidentes da Copa das Confederações de 2013, obtidos a partir da base de dados do sistema Pacificador. Nos experimentos realizados, definiu-se duas categorias (Manifestação e Trânsito), criou-se amostras de treinamento a partir de dicionários com termos característicos de cada categoria e comparou-se métodos de aprendizagem supervisionada para avaliar a possibilidade de criação de um classificador automático de texto de incidentes. Foram avaliados três métodos: *Naive Bayes*, SVM e Árvore de Decisão.

Esses métodos foram comparados a partir dos valores das Medidas F obtidas, sendo que para cada método foram testados os parâmetros de frequência mínima de ocorrência de termos e a ponderação dada para cada termo no vetor de palavras. As frequências mínimas de termos podiam assumir os valores 1, 3 ou 5. E as ponderações avaliadas foram TF, IDF, TF-IDF e Binário.

Para a categoria de Manifestação, as melhores configurações foram às de ponderação TF-IDF, TF e Binário com frequência mínima de ocorrência 5 com o método de SVM. Para a categoria de Trânsito, a melhor configuração foi com ponderação TF-IDF, frequência mínima de ocorrência de termos 3 também com o método de SVM. Para essas melhores configurações, também foi analisada a Curva ROC gerada, confirmando um ótimo desempenho.

Com a análise de qual melhor configuração para cada categoria, criou-se uma interface utilizando a linguagem de programação JAVA para disponibilizar o modelo de classificador escolhido a partir de experimentos no modo *Explorer* do WEKA. Para cada categoria,

foram obtidas estatísticas de horário de notificação e região de ocorrência dos incidentes para fornecer informações da dinâmica de cada classe definida ao usuário operador do sistema.

Concluiu-se, assim, que é possível a criação de um módulo de suporte ao operador do Pacificador que contenha um classificador automático de texto de incidentes de grandes eventos com estatísticas de eventos passados. Apesar das enorme variação de textos com termos militares específicos e muitas descrições breves, os métodos empregados, com auxílio de ferramentas como *stemmers* e *remoção de stopwords*, conseguem obter um resultado muito satisfatório e criar classificadores muito acurados para as categorias definidas.

Há uma grande gama de possíveis trabalhos futuros. Pode-se citar:

- Definir Ações Operacionais Padrão para cada categoria de forma à auxiliar os comandantes de operações na tomada de decisão. Essas ações seriam retornadas após a categorização do incidente e seriam definidas com o auxílio de especialistas na área de Segurança.
- Definição de novas categorias. As categorias definidas nesse trabalho foram categorias de grande expressividade, com termos específicos e com uma porcentagem relevante dentro da base de dados total, aumentando as chances de boas medidas para o classificador. É interessante analisar categorias com termos menos específicos e comparar os resultados. Por exemplo, pode-se realizar experimentos para analisar os resultados da categoria de *saúde* ou *terrorismo*.
- Criação de ontologias para melhorar o tratamento semântico e representação de cada categoria de incidentes.
- Obter novas estatísticas para cada categoria de incidentes. Quanto mais estatísticas sobre a dinâmica de cada categoria, maior será o auxílio à tomada de decisão.
- Integrar o módulo de classificação automática ao sistema Pacificador, inclusive avaliando a possibilidade de alterar o símbolo dos incidentes reportados nos mapas interativos do sistema de acordo com suas classificações.
- Testar com outros grandes eventos, como a Copa do Mundo de 2014, e avaliar os resultados.
- Testar a utilização de algoritmos de associação nos textos de incidentes para a obtenção de regras de associação, possibilitando a detecção de eventos anômalos.

Referências

[1] Ministério da Defesa. Doutrina para o sistema militar de comando e controle, 3a edição. http://www.defesa.gov.br/arquivos/legislacao/emcfa/publicacoes/comando_controle/md31_m_03_dout_sismc_3_ed_2015.pdf, 2014. Acessado em: 2016-05-26. x, 1, 28, 30

[2] Centro de Desenvolvimento de Sistemas do Exército Brasileiro. *Manual do Sistema Pacificador*. x, 1, 2, 30, 31

[3] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010. x, 4, 6, 8

[4] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997. 4

[5] Saiqa Aleen, Luiz Fernando Caprez, and Faheen Ahmed. Benchmarking machine learning techniques for software defect detection. *International Journal of Software Engineering & Applications (IJSEA)*, (3):11–23, May 2015. 4

[6] Sakima Omar, Asru Ngadi, and Hamid H. Jebur. Machine learning techniques for anomaly detection: An overview. *International Journal of Computer Applications (0975 - 8887*, 79(2):33–41, Outubro 2013. 4

[7] Larissa Sayuri Futino Castro dos Santos. *Estudo Online da Dinâmica Espaço-temporal de Crimes através de Dados da Rede Social Twitter*. dissertation, Departamento de Estatística. Universidade Federal de Minas Gerais, 2015. 4, 32

[8] Eduardo Corrêa Gonçalves. Extração de Árvores de decisão com a ferramenta de data mining weka. http://www.devmedia.com.br/extracao-de-arvores-de-decisao-com-a-ferramenta-de-data-mining-weka/3388, 2007. Acessado em: 2016-06-25. 5

[9] Mark Hall, Ian Witten, and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, 2011. x, 5, 8, 12, 13, 19, 27

[10] John C. Platt. A fast algorithm for training support vector machines. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-98-14.pdf. Acessado em: 2016-10-25. 8

[11] Oded Maimon and Lion Rokach. *Data Mining and Knowledge Discovery Handbook Second Edition*. Springer, 2010. 12

[12] Marcelo Kaminski Sanches. *Aprendizado de máquina semi-supervisionado: proposta de um algoritmo para rotular exemplos a partir de poucos exemplos rotulados*. PhD thesis, USP - São Carlos, 2003. 13

[13] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining - Concepts and Techniques Third Edition*. Morgan Kaufmann, 2012. x, 13, 14, 15, 17

[14] Pedro Henrique Tessarolo and William Barbosa Margalhães. A era do big data no conteúdo digital: os dados estruturados e não estruturados. *XVII SEINPAR - Semana de Informática de Paranavaí*, 2015. 13

[15] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37–54, 1996. 13

[16] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. The MIT Press, 2001. 15

[17] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rüdiger Wirth. CRISP-DM 1.0 step-by-step data mining guide. https://www.the-modeling-agency.com/crisp-dm.pdf. Acessado em: 2016-05-04. x, 15, 16

[18] SAS Institute. http://www.sas.com/. Acessado em: 2016-11-05. 15

[19] Daniel Jurafsky and James H. Martin. *Speech and Language Processing, second edition*. Pearson Education, Upper Sandle River, New Jersey (USA) 07458, 2009. 19

[20] Robert Dale. *Classical approaches to natural language processing*. Handbook of natural language processing 2a edição, 2010. 19

[21] Auto Tavares da Camara Junior. *Processamento de Linguagem Natural para Indexação Automática Semântica-Ontológica*. PhD thesis, Universidade de Brasília, 2013. 20

[22] Xiaojin Zhu. Basic text process. http://pages.cs.wisc.edu/~jerryzhu/cs769/text_preprocessing.pdf, 2010. Acessado em: 2016-06-25. 22

[23] Microsoft Azure. https://azure.microsoft.com/. Acessado em: 2016-11-05. 23

[24] Amazon Machine Learning. https://aws.amazon.com/pt/machine-learning/. Acessado em: 2016-11-05. 23

[25] The University of Waikato. WEKA: Data Mining Software in Java. http://www.cs.waikato.ac.nz/ml/weka/. Acessado em: 2016-11-05. 23

[26] Rapidminer. https://rapidminer.com/. Acessado em: 2016-11-05. 23

[27] Orange Data Mining. http://orange.biolab.si/. Acessado em: 2016-11-05. 23

[28] The R Project for Statistical Computing. https://www.r-project.org/. Acessado em: 2016-11-05. 23

[29] Eibe Frank, Mark Hall, et al. *WEKA Manual for Version 3-9-0*. University of Waikato, Abril 2016. x, 23, 24, 28

[30] Ministério da Defesa. Glossário das forças armadas. http://www.defesa.gov.br/arquivos/File/legislacao/emcfa/publicacoes/md35_g_01_glossario_fa_4aed2007.pdf, 2007. Acessado em: 2016-06-04. 29

[31] Albert Frederico de Menezes II Pak. *Aplicação de Técnicas de Mineração de Texto para Categorização de Eventos de Segurança no CTIR Gov*. dissertation, Departamento de Ciência da Computação. Universidade de Brasília, 2010. 32

[32] Edson Takashi Matsubara, Claudia Aparecida Martins, and Maria Carolina Monard. Pretext: uma ferramenta para pre-processamento de textos utilizando a abordagem bag-of-words. http://conteudo.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_RT_209.pdf, 2003. Acessado em: 2016-11-03. 32

[33] Snowball Stemmer. https://snowballstem.org/. Acessado em: 2016-10-25. 38

Apêndice A

Código Java do Classificador

Código Front-end

```
package monografia;

import java.awt.EventQueue;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.Color;
import java.awt.Image;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.JLabel;

public class Interface {

    //declaracao das labels (botao, caixa de texto, imagens, titulo)
    private JFrame frame;
    private JTextField entrada;
    private JLabel lblResultado;
    static String resultado;
    static Image img;
    static Classificador classificador;

    public static void main(String[] args) throws Exception{
        //instancia o classificador
        classificador = new Classificador();
        classificador.create_model();
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    //instancia a interface
                    Interface window = new Interface();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
```

```
            }
        }
    });
}

public Interface() {
    initialize();
}

private void initialize() {
    //definicao da janela do aplicativo
    frame = new JFrame();
    frame.setBounds(100, 100, 1000, 700);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(null);

    //criacao da area para a segunda imagem
    JLabel lblEstatistica = new JLabel("");
    lblEstatistica.setBounds(10, 120, 532, 562);
    frame.getContentPane().add(lblEstatistica);

    //criacao da area para a segunda imagem
    JLabel lblEstatistica2 = new JLabel("");
    lblEstatistica2.setBounds(500, 120, 532, 562);
    frame.getContentPane().add(lblEstatistica2);

    //Instancia um botao
    JButton btnNewButton = new JButton("Classificar!");
    //define a acao que sera realizada quando o botao for clicado.
    btnNewButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            try{
                //recebe o resultado (classificacao) do classificador
                resultado = classificador.evaluate_text(entrada.getText
                    ());
            }catch(Exception ex){
                System.out.println("Excecao: classificacao do dado");
            }
            if(resultado.equals("manifestacao")){
                //se o resultado for manifestacao, carrega as imagens
                    referentes a manifestacao e escreve o resultado na
                    tela
                img = new ImageIcon(this.getClass().getResource("/
                    manifest-cidade.png")).getImage();
                lblEstatistica.setIcon(new ImageIcon(img));
                img = new ImageIcon(this.getClass().getResource("/
                    manifestacao-hora.png")).getImage();
                lblEstatistica2.setIcon(new ImageIcon(img));
                resultado = "Manifestacao.";
            }else if(resultado.equals("transito")){
                //se o resultado for transito, carrega as imagens
                    referentes a transito e escreve o resultado na tela
                img = new ImageIcon(this.getClass().getResource("/
                    transito-cidade.png")).getImage();
                lblEstatistica.setIcon(new ImageIcon(img));
```

```
                    img = new ImageIcon(this.getClass().getResource("/
                        transito-hora.png")).getImage();
                    lblEstatistica2.setIcon(new ImageIcon(img));
                    resultado = "Transito";
                }else{
                    //se nao encontrar nenhum dos resultados anteriores,
                        escreve que nao tem categoria
                    img = new ImageIcon(this.getClass().getResource("")).
                        getImage();
                    lblEstatistica.setIcon(new ImageIcon(img));
                    img = new ImageIcon(this.getClass().getResource("")).
                        getImage();
                    lblEstatistica2.setIcon(new ImageIcon(img));
                    resultado = "Sem categoria.";
                }
                lblResultado.setText("Categoria: " + resultado);
            }
        });
        //Definicao das dimensoes do botao
        btnNewButton.setBounds(865,50, 107, 54);
        frame.getContentPane().add(btnNewButton);

        //criacao do campo de texto em que se escreve o incidente
        entrada = new JTextField();
        entrada.setBounds(30, 50, 780, 54);
        frame.getContentPane().add(entrada);
        entrada.setColumns(10);

        //Criacao da area onde aparece o resultado
        lblResultado = new JLabel("Classificacao:");
        lblResultado.setBackground(Color.GREEN);
        lblResultado.setForeground(Color.DARK_GRAY);
        lblResultado.setBounds(30, 115, 424, 54);
        frame.getContentPane().add(lblResultado);

        //Definicao da area que aparece o titulo
        JLabel lblTitulo = new JLabel("Classificador Automatico de
            Incidentes");
        lblTitulo.setBounds(30,11,413,14);
        frame.getContentPane().add(lblTitulo);
    }
}
```

Código Back-end

```
package monografia;

import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.Random;

import weka.classifiers.Evaluation;
import weka.classifiers.functions.SMO;
```

```
import weka.classifiers.meta.FilteredClassifier;
import weka.core.DenseInstance;
import weka.core.Instance;
import weka.core.Instances;
import weka.filters.unsupervised.attribute.StringToWordVector;
import weka.core.Attribute;

public class Classificador {
    private static FilteredClassifier cls_manifest;
    private static FilteredClassifier cls_transito;
    public static void create_model() throws Exception{

        BufferedReader reader_manifest = null;
        reader_manifest = new BufferedReader(new FileReader("src//
           monografia//arquivos1.arff"));

        BufferedReader reader_transito = null;
        reader_transito = new BufferedReader(new FileReader("src//
           monografia//arquivos2.arff"));

        Instances train_manifest = new Instances (reader_manifest);
        //seta atributo 1 como a classe
        train_manifest.setClassIndex(1);

        Instances train_transito = new Instances (reader_transito);
        //seta atributo 1 como a classe
        train_transito.setClassIndex(1);

        //converte strings para vetores de palavras
        StringToWordVector stwv_manifest = new StringToWordVector(2000);
        stwv_manifest.setInputFormat(train_manifest);

        //converte strings para vetores de palavras
        StringToWordVector stwv_transito = new StringToWordVector(2000);
        stwv_transito.setInputFormat(train_transito);

        //aplica filtros
        stwv_manifest.setOptions(weka.core.Utils.splitOptions(" -R first-
           last -W 15000 -prune-rate -1.0 -C -T -I -N 0 -L -stemmer \"weka.
           core.stemmers.SnowballStemmer -S portuguese \" -stopwords-
           handler \"weka.core.stopwords.WordsFromFile -stopwords
           inserir_path/portuguese.txt\" -M 5 -tokenizer \"weka.core.
           tokenizers.WordTokenizer -delimiters \\\" \\\\r\\\\n\\\\t
           .,;:\\\\\\\'\\\\\\\"()?!\\\"\"\r\n\r\n   "));
        stwv_transito.setOptions(weka.core.Utils.splitOptions(" -R first-
           last -W 15000 -prune-rate -1.0 -C -T -I -N 0 -L -stemmer \"weka.
           core.stemmers.SnowballStemmer -S portuguese\" -stopwords-handler
            \"weka.core.stopwords.WordsFromFile -stopwords inserir_path/
           portuguese.txt\" -M 3 -tokenizer \"weka.core.tokenizers.
           WordTokenizer -delimiters \\\" \\\\r\\\\n\\\\t
           .,;:\\\\\\\'\\\\\\\"()?!\\\"\"   "));

        cls_manifest = new FilteredClassifier();
        //carrega os dados que foram filtrados
        cls_manifest.setFilter(stwv_manifest);
        //seleciona o metodo de classificacao
```

```
        cls_manifest.setClassifier(new SMO());
        //cria o classificador
        cls_manifest.buildClassifier(train_manifest);

        cls_transito = new FilteredClassifier();
        //carrega os dados que foram filtrados
        cls_transito.setFilter(stwv_transito);
        //seleciona o metodo de classificacao
        cls_transito.setClassifier(new SMO());
        //cria o classificador
        cls_transito.buildClassifier(train_transito);

    }

    //classificacao de texto inserido pelo usuario
    public static String evaluate_text(String text)throws Exception{
        ArrayList<Attribute> attributeList_manifest = new ArrayList<
            Attribute>(2);
        ArrayList<Attribute> attributeList_transito = new ArrayList<
            Attribute>(2);

        Attribute descricao = new Attribute("text", (ArrayList<String>)
            null);

        ArrayList<String> classVal_manifest = new ArrayList<String>();
        ArrayList<String> classVal_transito = new ArrayList<String>();
        classVal_manifest.add("resto");
        classVal_manifest.add("manifestacao");

        classVal_transito.add("transito");
        classVal_transito.add("restoTransito");

        attributeList_manifest.add(descricao);
        attributeList_manifest.add(new Attribute("@@class@@",
            classVal_manifest));

        attributeList_transito.add(descricao);
        attributeList_transito.add(new Attribute("@@class@@",
            classVal_transito));

        Instances data = new Instances("TestInstances",
            attributeList_manifest,0);

        // cria nova instancia para receber o texto inserido na interface
        Instance nova;
        nova = new DenseInstance(data.numAttributes());
        data.add(nova);
        data.setClassIndex(1);

        nova.setValue(descricao, text);
        nova.setDataset(data);

        // testa com o classificador de Manifestacao, depois com o
            classificador de Transito
        double predicted = cls_manifest.classifyInstance(nova);
```

```
        System.out.print("\nClasse (Cls de Manifestacao):" + nova.
            classAttribute().value((int) predicted));

        String predicao = new String(nova.classAttribute().value((int)
            predicted));
        if(predicao.equals(new String("manifestacao"))){
            return (new String("manifestacao"));
        } else {
             Instances data_transito = new Instances("TestInstances",
                attributeList_transito,0);

             Instance nova_transito;
             nova_transito = new DenseInstance(data_transito.numAttributes
                 ());
             data_transito.add(nova_transito);
             data_transito.setClassIndex(1);

             nova_transito.setValue(descricao, text);
             nova_transito.setDataset(data_transito);

             predicted = cls_transito.classifyInstance(nova_transito);
             System.out.print("\nClasse (Cls de Transito):" + nova_transito
                .classAttribute().value((int) predicted));
             String predicao_transito = new String(nova_transito.
                classAttribute().value((int) predicted));
             if(predicao_transito.equals(new String("transito"))){
                 return (new String("transito"));
             } else {
                 return (new String("Nao detectada"));
             }
        }
    }
}
```