Universidade Federal de Pernambuco
Centro de Informática
Graduação em Ciência da Computação

Classificação e Extração de Eventos de Trânsitos no Twitter

Trabalho de Graduação

Marcio Mendes Cavalcanti Junior

Recife
Junho de 2018

Universidade Federal de Pernambuco
Centro de Informática
Graduação em Ciência da Computação

Classificação e Extração de Eventos de Trânsitos no Twitter

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

Aluno: Marcio Mendes Cavalcanti Junior
Orientador: Prof. Luciano de Andrade Barbosa

Recife
Junho de 2018

“Vive-se de fé, não de sonhos.”
São Padre Pio de Pietrelcina

Agradecimentos

Inicialmente, agradeço a Deus por tudo o que vivenciei até aqui, por todo o cuidado, carinho e proteção. Agradeço a nossa Senhora das Graças e São Padre Pio, dos quais sou devoto, pela intercessão e condução em cada momento dessa caminhada em minha vida.

Agradeço ao professor Luciano de Andrade Barbosa e ao seu aluno de Mestrado Johny Moreira da Silva, por toda a orientação durante este trabalho, pela preocupação e acompanhamento, estando sempre dispostos a tirar dúvidas e a ajudarem, sendo fundamentais para a realização e finalização do mesmo.

Agradeço ao Centro de Informática e a todos que fazem parte deste centro incrivel, que nos proporciona uma construção não só para o mercado de trabalho, mas para toda a vida. Foram dias incríveis neste ambiente de aprendizado que é sem igual, obrigado. .

Aos meus amigos, agradeço o convívio e a busca por novos conhecimentos, ajudando sempre uns aos outros e criando um laço especial que vai perdurar pela vida.

Por fim, agradeço a meus pais, minha base, por cada esforço realizado para que eu pudesse realizar este sonho de estudar e buscar algo melhor para todos nós. Pelos conselhos e apoios dados sempre que precisei e por terem me educado para que eu fosse quem sou hoje. Obrigado por tudo, por tudo. Sem vocês eu também não estaria aqui.

Resumo

O volume de dados criado por redes sociais tem sido cada vez maior devido ao crescimento do número dessas redes criadas nos últimos anos e ao número cada crescente de pessoas conectadas a elas. Diversos tópicos são encontrados nesse meio (entretenimento, moda, ciência, tecnologia, entre outros). Um tópico particularmente interessante é o de trânsito. Usuários de mídias sociais muitas vezes postam informações de trânsito em seus perfis como acidentes, engarrafamento etc. Neste trabalho, iremos utilizar o Twitter para analisar postagens relacionadas a trânsito em grandes cidades brasileiras. Mais especificamente, pretendemos classificar postagens como eventos de trânsito ou não, e extrair delas informações como: onde, quando e porque o evento aconteceu. A partir disso, poderemos, como um trabalho futuro, construir uma aplicação para monitoramento em tempo real da condição de trânsito nessas cidades.

Palavras-chave: Classificação de Texto, Classificação de Twitter, Trânsito, Extração de Texto, Extração de Twitter, Eventos de Trânsito

Abstract

The volume of data created by social networks has been increasing due to the growth in the number of networks created in recent years and the increasing number of people connected to them. Several topics are found in this medium (entertainment, style, science, technology, among others). A particularly interesting topic is about traffic. Social media users often post traffic information on their profiles like accidents, traffic jam, etc. In this work, we will use Twitter to analyze traffic-related posts in large Brazilian cities. More specifically, we intend to classify postings as traffic events or not, and extract information such as: where, when, and why the event happened. From this, we can, as a future work, build an application for real-time monitoring of the traffic condition in these cities.

Keywords: Text Classification, Twitter Classification, Traffic, Text Extraction, Twitter Extraction, Traffic Events

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O crescimento no número de pessoas conectadas em *tempo real* a redes sociais ou aplicações dentro da internet tem aumentado o volume de dados gerados e disponibilizados a todo momento em todas as partes do mundo. Esse crescimento tem possibilitado às pessoas divulgarem notícias, suas opiniões e informações sobre os mais variados assuntos [1] no momento exato do dia e de maneira rápida.

Um interessante exemplo de informações fornecidas através das pessoas, por meio dessas redes, são as informações de como está o trânsito ou de eventos que estão acontecendo no trânsito.

Com esse crescente número de pessoas conectadas, muitas ferramentas também têm avançado e possibilitado cada vez mais facilidades no acesso a essas informações ou as suas fontes, permitindo que vários estudos ou aplicações sejam criadas a partir desses dados.

Uma dessas ferramentas que tem crescido e disponibilizado grandes condições para a extração de dados provenientes dela é o *Twitter*[1]. O Twitter é um site de rede social que permite a troca de mensagens curtas (tweets) entre seus usuários de maneira rápida e facilitada através de suas inúmeras formas de acesso (portal web, aplicativos para dispositivos móveis, dentre outros) [2]. Com tanto volume de *Tweets* ou dados gerados em tempo real, é de extrema dificuldade a análise de cada dado recebido e a classificação sobre se esse dado é ou não relevante para a nossa aplicação ou estudo.

O objetivo deste trabalho é desenvolver um coletor e classificador de mensagens do Twitter(Tweets) relacionados a eventos no trânsito de grandes cidades brasileiras, classificando o tweet coletado sobre se é ou não um evento de trânsito e extraindo dele, caso classificado como um evento, informações de **onde, quando e o motivo** do evento relacionado a ele.

Este documento está estruturado em capítulos com o conteúdo distribuído da forma explicada a seguir.

No capítulo 2, serão abordados os fundamentos básicos utilizados durante todo o processo, assim como as técnicas utilizadas no decorrer do mesmo e alguns trabalhos relacionados ao nosso projeto. No capítulo 3, abordaremos a solução proposta neste trabalho, demonstrando o fluxograma do processo e as ferramentas utilizadas em cada fase do fluxo realizado. No capítulo 4, traremos experimentos e resultados obtidos durante a construção, trazendo os algoritmos aplicados e os

[1] https://twitter.com/

resultados a partir deles. Por fim, o capítulo 5 expõe a conclusão do trabalho e possíveis trabalhos futuros.

Capítulo 2

Fundamentos

Neste capítulo apresentaremos importantes fundamentos para o entendimento do trabalho como um todo, introduzindo a área e os conceitos utilizados em nosso projeto.

2.1 Aprendizagem de Máquina

Aprendizagem de máquina é um subconjunto de inteligência artificial no campo da ciência da computação que usa diferentes técnicas tentando dar aos computadores a capacidade de "aprender" com dados, sem ser explicitamente programado[2].

A aprendizagem de máquina é usada em diversas aplicações computacionais, como aplicações para Web-Search, filtros de Spam, Sistemas de Recomendação, detecção de fraudes, entre outras [3].

Existem diversos cenários comuns no meio da aprendizagem de máquina. Esses cenários diferem nos tipos de dados de treinamento disponíveis, na ordem e métodos pelo qual os dados de treinamento são recebidos e na forma em que os dados de testes são utilizados para avaliar o algoritmo de aprendizagem [4]. Um importante cenário é o de Aprendizagem Supervisionada, cenário utilizado neste projeto, do qual falaremos mais um pouco, abaixo.

2.2 Aprendizagem Supervisionada

Aprendizagem supervisionada é a busca por algoritmos que raciocinam a partir de instâncias fornecidas externamente para produzir hipóteses, que então fazem previsões sobre instâncias futuras[5].

Essas instâncias externas são fornecidas como conjuntos de treinamento constituídos por um par formado por um grupo de características (*features*), escolhidas para representar o objeto e uma variável de saída, que é a classificação referente a esse objeto.

Dependendo do tipo da variável de saída, existem duas categorias de aprendizagem supervisionada, onde quando a saída é nominal ou categórica, temos a categoria de aprendizagem supervisionada de *classificação* ou de *reconhecimento de padrões*[6], e quando a variável de saída é um valor real escalar, temos a categoria de aprendizagem supervisionada de *regressão*[6].

No cenário de aprendizagem supervisionada, entender os resultados gerados pelo conjunto de treinamento é fundamental, já que diversos fatores(ruídos,

[2] https://en.wikipedia.org/wiki/Machine_learning

quantidade de features,.etc) podem melhorar ou piorar o aprendizado do nosso modelo. Para auxiliar no entendimento desses resultados, temos em aprendizagem de máquina dois conceitos diferentes, o *Underfitting* e o *Overfitting.*

O *underfitting*[3] acontece quando seu modelo apresenta resultados ruins nos dados de treinamento, ocorrendo quando o modelo não consegue relacionar os dados de entrada e o valor de saída(variável de saída), sendo aconselhável que se adicione mais features ao objeto. Já o *overfitting*[3]*,* acontece quando seu modelo apresenta um bom desempenho nos dados de treinamento mas não apresenta um bom desempenho nos dados de avaliação, ocorrendo porque o modelo memorizou os dados que treinou e se tornou incapaz de generalizar para eventos que ainda não viu.

Neste capítulo apresentaremos a categoria de aprendizagem supervisionada de classificação (Seção 2.2.1), abordando seus tipos, alguns algoritmos utilizados no processo de classificação e as métricas mais utilizadas para realizar a avaliação de desempenho de um modelo/algoritmo para determinado conjunto de dados e a Classificação de Texto (Seção 2.2.2), abordando brevemente suas características e fases básicas. .

2.2.1 Classificação

Neste sub capítulo, introduziremos os conceitos básicos sobre aprendizagem supervisionada de classificação, explanando seus tipos, alguns algoritmos que utilizaremos no trabalho e a forma de análise de desempenho de cada modelo gerado, verificando a eficiência de cada um através de diferentes métricas.

2.2.1.1 Tipos de Classificação

Como citado acima, a categoria de aprendizagem supervisionada de classificação tem uma saída nominal ou categórica (Seção 2.2), podendo se dividir em subcategorias e variando de acordo com a quantidade de classes referente a classificação. Podemos dividir as categorias em três diferentes tipos:

- **Binária:** No problema de classificação do tipo binária, existem apenas duas classes possíveis ao algoritmo, a classe positiva e a classe negativa (1 ou 0). Como a saída real de vários algoritmos deste tipo de classificação é uma pontuação de previsão, indicando se determinada observação pertence a classe positiva, é necessário escolher um limite de classificação (corte) para essa pontuação, separando as observações em positivas ou negativas de acordo com esse limite [11].

[3] https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

Apesar de simples, o tipo de classificação binária é bastante importante, já que problemas mais complexos podem ser reduzidos a eles, facilitando no resultado e previsão desses problemas.

- **Multiclasse:** Neste tipo de classificação, o algoritmo recebe um conjunto de dados que possui três ou mais classes possíveis para a previsão. Desta forma, não é necessária a escolha de um limite de classificação (corte), como no tipo binária, já que a previsão retorna uma classe ou um rótulo como saída para a classificação. Nos casos em que se esperam altas precisões sobre determinadas previsões para os dados de entrada, pode-se escolher um limite nas pontuações previstas com base no qual aceitará ou não a resposta retornada [11]. Neste trabalho, abordaremos este tipo de classificação, tendo em nosso conjunto, diferentes classes dos quais falaremos nas próximas seções.

- **Multi-label:** Neste tipo de categoria, os elementos do conjunto de dados podem pertencer a mais de uma classe ao mesmo tempo, o que os torna em sua maioria, problemas de classificação mais complexos. Como o conjunto de classes possíveis para esta categoria podem ser maiores que duas ao mesmo tempo, podemos vê-los como uma generalização do tipo de classificação multiclasse [12].

2.2.1.2 Algoritmos de Classificação

Diversas abordagens e algoritmos são utilizados na classificação supervisionada. Nesta sub seção, falaremos brevemente sobre alguns algoritmos que usaremos em nosso trabalho, trazendo de maneira geral, fundamentos usados por eles, para um maior entendimento relacionado aos resultados (seção 4.2).

- **Naive Bayes Multinomial Classification:** O algoritmo de Naive Bayes usa a regra de Bayes para calcular a probabilidade de cada classe dada a instância, assumindo que os atributos são condicionalmente independentes, dado o rótulo [14]. É considerado um algoritmo rápido que apresenta bom desempenho na previsão de categorias, mesmo com poucos dados de treinamento. Para dados discretos, já que alguns parâmetros precisam ser estimados, as estimativas tendem a estabilizar rapidamente e mais dados não mudam muito o modelar. Com atributos contínuos, a discretização pode formar mais intervalos quanto mais dados está disponível, aumentando assim o poder de representação. Entretanto, mesmo com dados contínuos, a discretização é global e não pode levar em conta as interações de atributos [14].

O classificador Naive Bayes multinomial é adequado para classificação com características discretas (por exemplo, contagens de palavras para classificação de texto). A distribuição multinomial normalmente requer contagens de recursos inteiros. No entanto, na prática, contagens fracionais como tf-idf também podem funcionar.

- **Logistic Regression:** A regressão logística é um algoritmo de aprendizagem de máquina usado para classificação, apesar do nome. Ele é usado para estimar valores discretos, baseado em um grupo de variáveis independentes, ou seja, a partir de uma função logística, ele ajusta os dados buscando prever a probabilidade de ocorrência de um evento. A regressão logística é uma poderosa forma estatística de modelar um resultado binomial com uma ou mais variáveis explicativas.

- **Decision Tree:** Como um método de aprendizagem de máquina não paramétrico, o algoritmo de classificação por árvore de decisão é um algoritmo usado para classificação e regressão. Através de dados de entrada, as árvores de decisão aprendem com os dados para aproximar uma curva senoidal com um conjunto de regras de decisão if-then-else. Quanto mais profunda a árvore, mais complexa é a decisão e mais apto o modelo. Uma das desvantagens do uso de árvore de decisão, é que existe a possibilidade de *overfitting* ("memorizando os casos de treinamento"), que pode reduzir potencialmente a precisão de um modelo para casos ainda não vistos [21].

- **XG Boost:** o Extreme Gradient Boosting (XG Boost), é baseado em árvore de decisão, tem sido amplamente utilizado em competição de aprendizagem de máquina e mineração de dados [20]. Além de ter uma flexibilidade muito grande em relação aos problemas, o XG Boost também é reconhecido por sua velocidade e alta performance. Através de uma biblioteca *open-source,* o XG Boost é composto por diversos algoritmos de aprendizagem de máquina, o que tem proporcionado resultados tão significativos em competições e projetos de aprendizagem de máquina e mineração de dados.

  Inicialmente criando uma árvore simples de baixo desempenho, o XG Boost constrói, a cada iteração, uma outra árvore com a capacidade de prever o que a anterior não conseguiu, realizando esse processo até que uma condição de parada seja atingida.

- **SVM:** o support vector machine (*SVM*), é uma técnica de aprendizagem de máquina para problemas de reconhecimento de padrão. Sendo utilizado em grandes problemas de classificação de texto. O *SVM* é uma abordagem geométrica para um problema de decisão. Com boa capacidade de generalização, robustez em grandes dimensões, convexidade da função

objetivo e com uma teoria bem definida, o *SVM* se destaca por sua boa capacidade de generalização [23]. Como o principal objetivo do *SVM* é encontrar um hiperplano entre dados de duas classes distintas, o *SVM* representa os exemplos como pontos no espaço e busca maximizar a distância entre os campos mais próximos em relação a cada uma das classes.

Com o conceito das funções de *Kernel* [23], os *SVM* podem lidar com classes de problemas linearmente não separáveis, realizando uma transformação dos dados de entrada para um novo espaço.

Neste projeto, utilizaremos o algoritmo de classificação baseado em *SVM* conhecido como *LinearSVC*[4], Semelhante ao SVC com o parâmetro kernel = 'linear', mas implementado em termos de liblinear em vez de libsvm. Ele tem mais flexibilidade na escolha de penalidades e funções de perda e deve ser dimensionado melhor para um grande número de amostras.

2.2.1.3 Métricas de Desempenho

Na análise de desempenho de um modelo de classificação, diferentes métricas são utilizadas para avaliar a sua eficiência. A porcentagem de acerto ou *acurácia* do modelo é uma importante medida de desempenho, porém, ela por si só, não nos fornece uma boa análise, sendo necessário uma junção de outras medidas para se obter um bom referencial de desempenho do mesmo.

Métricas como *Recall, Precision* e *F-measure* nos trazem uma precisão maior sobre como nosso modelo de classificação se comporta, desta forma, falaremos mais um pouco sobre elas, abaixo:

- *Acurácia (Accuracy)*: Métrica para avaliar modelos de classificação. Informalmente, a acurácia é a fração de previsões que nosso modelo acertou.

  Para calculá-la, realizamos a divisão da quantidade de todos os acertos (verdadeiros positivos e verdadeiros negativos), pela número de exemplos classificados em todas as possibilidades.

$$A = (VP + VN) / (VP + VN + FP + FN)$$

- *Precisão (Precision):* Métrica para uma determinada entrada de dados serem classificados na classe correta em relação às entradas que foram classificadas nesta classe (corretas ou não). Para calculá-la, realizamos a divisão do número de acertos (verdadeiros positivos), pelo número de exemplos classificados como positivos (verdadeiros positivos + falsos positivos), ou seja:

[4] http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

$$P = VP / (VP + FP)$$

- *Cobertura (Recall):* Métrica que avalia a proporção de resultados corretos em relação ao total de dados de entrada que deveriam ser classificados em uma classe específica. Para calculá-la, realizamos a divisão do número de acertos (verdadeiros positivos), dividido pela soma de verdadeiros positivos e mais falsos negativos, ou seja:

$$R = VP / (VP + FN)$$

- *F-measure:* Métrica que funciona como uma medida única para a comparação de diferentes classificadores. Ela é baseada na média harmônica ponderada da precisão e o recall, sendo importante, já que combina as duas medidas. Para calculá-la, utilizamos a combinação das duas outras métricas da seguinte forma:

$$F1 = 2 (P * R) / (P + R)$$

2.2.2 Classificação de Texto

Devido à maior disponibilidade de documentos digitais e à maior necessidade de organizá-los, o uso da classificação de texto com aprendizagem de máquina tem crescido nos últimos anos. A partir de um conjunto de documentos pré-classificados, processos indutivos buscam gerar modelos para previsões automáticas de classes, com o intuito de diminuir consideravelmente o esforço de trabalho especializado, além de criar uma portabilidade direta para diferentes domínios[15].

Apesar de trazer uma diminuição significativa no esforço especializado, como citado acima, a classificação de texto ainda necessita de um trabalho manual importante, já que precisamos rotular nossa base de treinamento e teste, para que através de algoritmos de aprendizagem de máquina (Seção 2.2.1.2), modelos sejam criados.

Nesta seção, falaremos um pouco de técnicas importantes para a classificação de textos, abordando fases importantes do processo, os quais usaremos em nosso trabalho.

2.2.2.1 Representação dos Elementos

Como diversos algoritmos de classificação podem ser usados, não podemos dispor como entrada para esses algoritmos, dados sem um padrão estruturado. Desta

forma, devemos representar nossos dados de entradas de diferentes maneiras, assim, algumas abordagens são usadas no meio de aprendizagem de máquina. Uma abordagem simples, mas bastante usada é a representação por *bag-of-words.*

A representação por *bag-of-words,* é um modelo de representação que transforma os dados em uma coleção de palavras, mantendo a multiplicidade, mas desconsiderando a estrutura gramatical e a ordem delas dentro do conjunto de dados[16].

Outras sub-formas são utilizadas junto a abordagem de *bag-of-words*, para uma melhor representação dos dados e para um maior entendimento do algoritmo de classificação que será usado no modelo. Dentre elas, a *Frequência dos termos* (Term Frequency -TF) que representa os dados por um vetor, onde cada dimensão do vetor contém um inteiro correspondente a quantidade de vezes que a palavra ocorre no documento, e a *Frequência do termo - frequência inversa dos documentos* (Term frequency - inverse document frequency - TF-IDF) onde essa representação não considera apenas a frequência do termo no documento, citado acima, mas também a quantidade de documentos que determinada palavra aparece. Assim, quanto mais documentos uma palavra aparecer, menos significativa para classificação ela é, pois não é uma característica discriminante de uma classe.

2.2.2.2 Pré-Processamento

A fase de pré-processamento, apesar de em alguns casos ser de alto custo computacional, é fundamental para a classificação. Como na classificação de texto, a maioria dos conjuntos de dados não são bem estruturados, abordagens precisam ser realizadas para ajustar os dados e a performance dos modelos. Por isso, técnicas como o de *Tokenização, Stopwords, Stemming,* entre outras, são comumente utilizadas.

No nosso trabalho, utilizaremos técnicas de *Tokenização,* que tem como objetivo segmentar as sentenças dos dados(Tweets, por exemplo) em unidades menores(*Tokens*), devolvendo um vetor de tokens que seguirão para o próximo grau de classificação na maneira correta.

2.3 Extração de Informações

Diversos dados estão dispostos hoje em dia de diferentes maneiras. Porém, grande parte desses dados estão dispostos de maneiras não estruturadas, o que dificulta o uso deles para diversos estudos e aplicações. Como uma área de processamento de linguagem natural que lida com a descoberta de informação em texto livre (não estruturado) [17], a extração de informação tenta através de diferentes técnicas, auxiliar na interpretação de fontes que não podem ser interpretadas por máquinas.

A tarefa de Extração de Informações (EI) é identificar um conjunto predefinido de conceitos em um domínio específico, ignorando outras informações irrelevantes. O processo de extração dessas informações estruturadas envolve a identificação de certas estruturas de pequena escala, como substantivos que denotam pessoa ou grupo de pessoas, referências geográficas e expressões numéricas, e encontrar relações semânticas entre eles[17].

No processo de interpretação de informações desses dados, algumas abordagens são utilizadas, entre elas, o *Named Entity Recognition,* que introduziremos abaixo e utilizaremos em nosso trabalho.

2.3.1 Named Entity Recognition

Named Entity Recognition (*NER*) é o processo de extrair entidades de um texto [18]. O NER é uma subtarefa de extração de informações que aborda um problema de detecção e classificação de tipos predefinidos de entidades nomeadas, buscando através desta classificação, auxiliar na extração de informações dentro de texto, por exemplo.

Organizações, nomes de lugares, expressões, nome de pessoas, são exemplos de entidades detectadas através do NER, que ainda pode incluir a extração de informações descritivas do texto sobre as entidades detectadas através do preenchimento de uma pequena escala modelo [17].

*Eu quero uma <Produto= **passagem**> de <Local= **Fortaleza**> para <Local= **Crato**>, saindo no próximo <Data= **domingo, às 22:00**>.*

Figura 2.1 Exemplo de detecção de entidades com *NER*

Fonte: https://pt.linkedin.com/pulse/reconhecimento-de-entidades-nomeadas-ner-o-que-%C3%A9-quais-lu%C3%ADs-fred

Para auxiliar no processo de extração de entidades, alguns modelos probabilísticos podem ser usados para modelar dados sequenciais, como rótulos de palavras em uma frase. Um modelo que vem sendo bastante usado atualmente é o Conditional Random Fields (*CRFs*), do qual falaremos mais na próxima seção.

2.3.2 Conditional Random Fields (CRFs)

Como uma estrutura que auxilia na construção de modelos probabilísticos para segmentar e rotular dados de sequência, o *CRF* tem sido bastante usado na extração de informação.

De acordo com Lafferty, McCallum e Pereira [19], o CRF oferece uma combinação única de propriedades, como modelos discriminatoriamente treinados

para seqüência de segmentação e rotulagem, treinamento eficiente e decodificação baseado em programação dinâmica; e estimativa de parâmetros garantida para encontrar o ótimo global. Para o uso do CRF como modelo discriminativo para uma tarefa de aprendizado sequencial, é necessária um conjunto de treinamento, para que a partir dele, um conjunto de parâmetros (conhecidos como pesos), que correspondem a importância dos recursos utilizados na tarefa, possam ser usados para gerar um modelo. Uma importante característica do *CRF* é que enquanto um classificador discreto prevê um rótulo para uma única amostra sem considerar amostras "vizinhas", um *CRF* pode levar em conta todo o contexto do conjunto de palavras, podendo ter uma maior eficácia no aprendizado.

2.4 Trabalhos Relacionados

Com o crescimento do número de redes sociais criadas e da quantidade de dados gerados por elas, diversos trabalhos estão sendo feitos, buscando extrair de diferentes maneiras, informações relevantes aos mais variados meios de pesquisas.

Nesta seção abordaremos alguns trabalhos relacionados ao nosso projeto e nossa área de atuação.

No trabalho [7], o autores buscam descobrir e informar a um usuário a melhor rota entre um local de origem e um local de destino, utilizando de informações de trânsito capturadas em *tempo real,* através da análise e classificação de tweets.

Os autores dividiram sua aplicação em duas etapas. Primeiro, os autores fizeram uso de técnicas de mineração de texto e linguagem natural, classificando os tweets que são relacionados a trânsito ou não. Por fim, os autores fizeram uso dos tweets classificados positivamente como evento de trânsito, para atualizar as melhores rotas entre uma origem e um destino, mostrando essas rotas através de uma aplicação android.

Em [8], os autores fazem uso da captura e classificação de tweets, buscando informações sobre incidentes de trânsito no Reino Unido, buscando substituir o uso de Técnicas de Detecção Automática de Incidentes (AID), que segundo eles tem um alto custo por se tratarem de monitoramento com câmeras e sensores, assim como dependência de um maciço volume de carros disponibilizando informações de seus Sistemas de Posicionamento Global(GPS). No trabalho proposto, foram apresentadas metodologias para recuperar, processar e classificar tweets públicos, combinando Processamento de Linguagem Natural (PLN) com um algoritmo Support Vector Machine (SVM) para classificação de texto, obtendo-se uma precisão de 88,27%.

No trabalho [9], os autores buscaram implementar uma ferramenta semelhante ao [7], utilizando da mesma forma, a classificação dos tweets relacionados a informações de trânsito para informar através de um serviço de *web*

service, de um serviço de *banco de dados* e de uma aplicação android, melhores rotas para um determinado local escolhido pelos usuários da aplicação.

Por fim, no trabalho [10], temos um sistema construído em SOA (Arquitetura orientada a serviços), buscando um bom modelo para detecção de eventos de trânsito em várias áreas da rede rodoviária Italiana. Através da exploração de técnicas de análise de texto e classificação de padrões citadas pelos autores, o trabalho obteve um ótimo resultado na classificação dos tweets, alcançando uma precisão de 95,75% com o SVM para o problema de 2 classes, e de 88,89% para o problema das 3 classes, considerando neste último, eventos externos.

Capítulo 3

Solução

Neste trabalho pretendemos desenvolver um classificador e extrator de textos relacionados à tweets no domínio de trânsito. Dividiremos o nosso processo em três fases: Captura, Classificação e extração de informação dos tweets. Diversas ferramentas foram utilizadas durante todo o processo, como APIs, Bibliotecas e diferentes técnicas de aprendizagem de máquina.

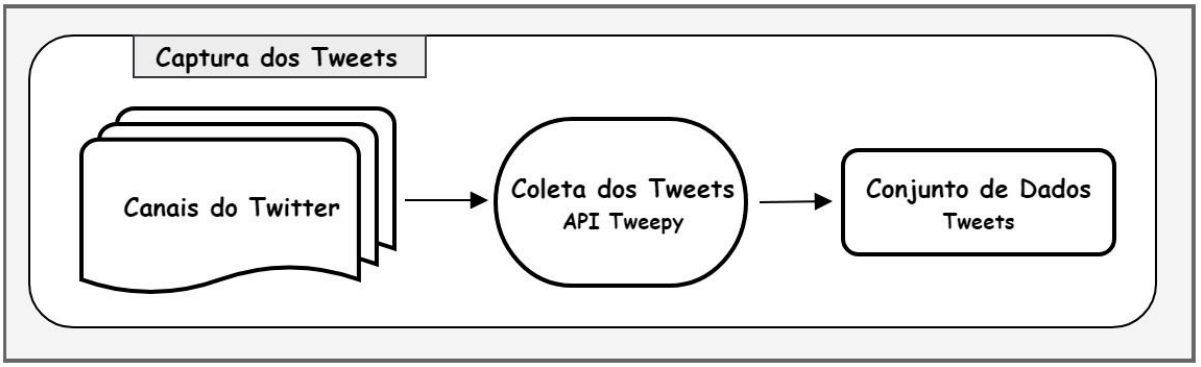

Figura 3.1 Fluxograma da Solução Proposta (Captura dos dados)

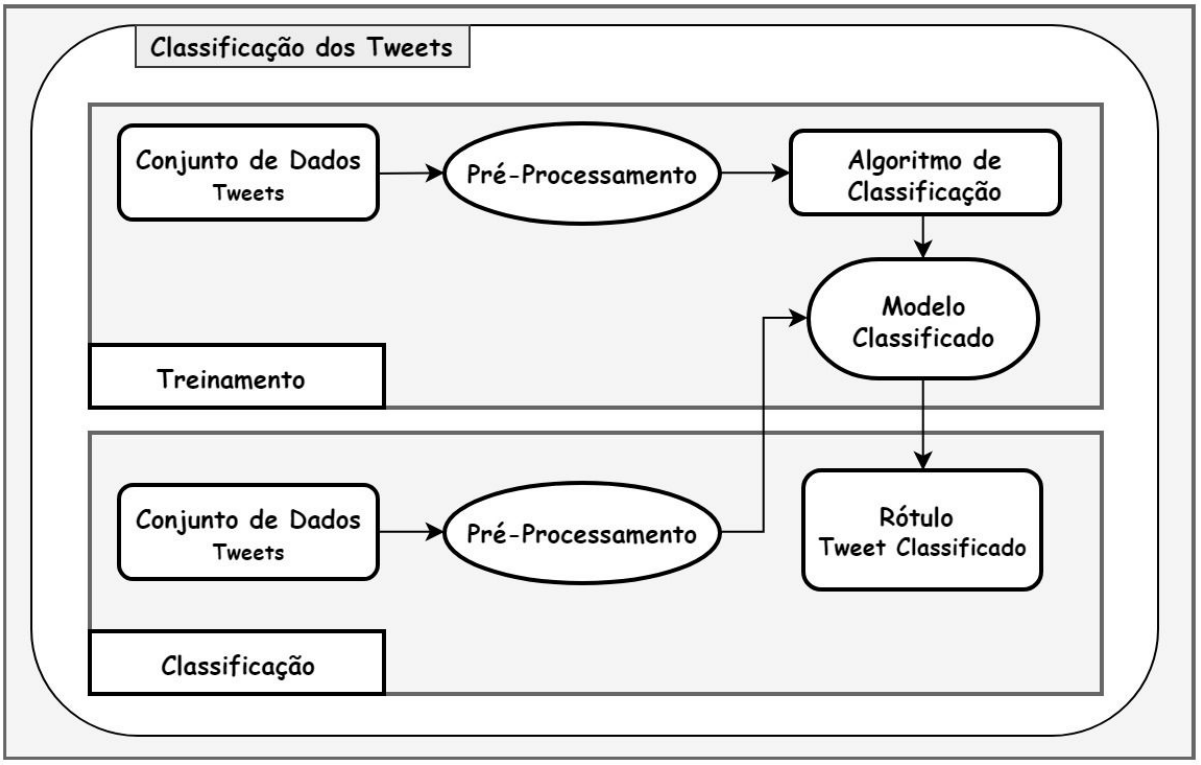

Figura 3.2 Fluxograma da Solução Proposta (Classificação dos dados)

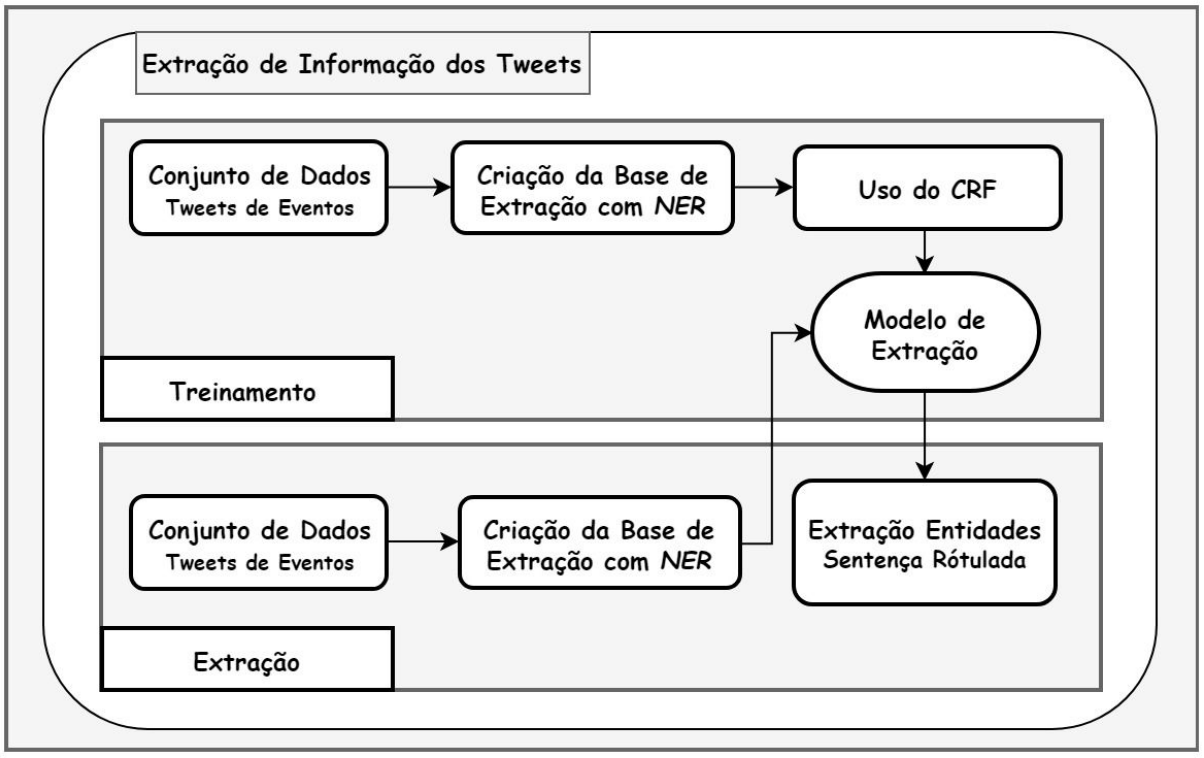

Figura 3.3 Fluxograma da Solução Proposta (Extração de informação dos dados)

Nas figuras 3.1, 3.2 e 3.3, temos o fluxograma de todo o processo da solução desenvolvida.

A solução proposta, como citado acima, se dividiu em três etapas: a captura dos dados para a realização do nosso trabalho, a classificação dos dados coletados e a extração de informação nos dados coletados.

Na fase de captura, canais do Twitter foram escolhidos e a partir deles e de um coletor, diversos tweets foram capturados e armazenados como nosso conjunto de dados. Na fase de classificação, a partir do conjunto de dados construído, realizamos um pré-processamento para substituir os metadados como *links, emoticons,* entre outros, e para criar vetores de tokens (*Tokenização*). A partir dos elementos pré-processados utilizamos cinco algoritmos de aprendizagem de máquina (ver Seção 2.2.1.2) para construir um modelo de classificação para nosso domínio.

Por fim, através da rotulagem manual de um conjunto de tweets relacionados a eventos de trânsito, criamos uma base estruturada para o uso da extração de informação com o *NER,* onde a partir dessa base e do uso do *CRF,* criamos um modelo de extração de elementos sequenciais, para extrair informações do nosso dado.

Neste capítulo, apresentaremos detalhes sobre a implementação de cada etapa e informações sobre cada fase necessária para a construção da solução.

3.1 Captura dos Dados

Para a construção da nossa solução, é necessário que tenhamos um conjunto de dados previamente capturados e rotulados para utilizarmos no treinamento do modelo da nossa classificação supervisionada.

Como o trabalho proposto busca utilizar dados fornecidos por usuários do Twitter (*Tweets*) relacionados a trânsito, consideramos relevantes somente páginas referentes a canais ligados ao nosso domínio e canais que atualizam suas postagens com uma frequência relevante, diariamente ou a cada hora pelo menos.

Para coletar os dados desses canais, utilizamos uma biblioteca de python chamada *Tweepy*[5] que faz uso da *API do Twitter*[6] e nos fornece uma rápida captura das postagens das páginas selecionadas.

Através destes canais escolhidos como relevantes e do coletor construído utilizando a biblioteca citada acima, realizamos a captura dos dados, e passamos para a próxima etapa da nossa solução.

3.2 Classificação dos Dados

Nesta etapa, já com os dados coletados, decidimos dividir nosso classificador em um classificador multiclasse (Seção 2.2.1.1) de três classes:

- **Eventos:** Classe de Tweets que fornecem informações sobre *manifestação, acidentes, carros quebrados, queda de árvores*, ou seja, eventos que estão acontecendo no trânsito.

  Ex: *"11h16 - Acidente entre carro e moto na R. Nelson Zang próx. à R. Padre Todesco, bairro Partenon. EPTC e SAMU no local."*

- **Informações:** Classe de Tweets que fornecem dados sobre a situação atual do trânsito em alguma rua, ou avenida.

  Ex: *"14h08 AV. RAJA GABAGLIA | Trânsito intenso entre Gentios e Contorno, no sentido centro."*

- **Outros:** Classe de Tweets relacionados a outras coisas, como informações sobre temperatura, propagandas, dicas de trânsito, entre outros.

[5] http://tweepy.readthedocs.io/en/v3.5.0/index.html
[6] https://developer.twitter.com/content/developer-twitter/en.html

Ex: *“22h50 Boa noite! Final das operações do Twitter BHTRANS.”*

A partir destas três classes, separamos manualmente todo o conjunto de dados para treinamento e teste, criando um subconjunto para classe *Eventos*, para a classe *Informações* e para a classe *Outros.*

Como ilustrado na figura 3.2, algumas abordagens foram necessárias para uma maior eficácia do nosso classificador, como a fase de pré-processamento, que abordaremos abaixo.

3.2.1 Pré-Processamento dos dados

Antes de encaminharmos o nosso conjunto de dados para o treinamento com o algoritmo de aprendizagem de máquina (Figura 3.2), analisamos o nosso conjunto, e diversas características foram observadas.

Uma característica presente em diversos tweets da nossa base, é o uso de *metadados,* como urls, emoticons, hashtags, entre outros. Como acreditamos que esses metadados não seriam relevantes em seu valor real para nosso classificador, realizamos um processo de substituição do valor real do metadado, por um valor único, que representa um token de string no lugar dele.

Para esse pré-processamento, utilizamos o *PreProcessor*[7]*,* biblioteca de python que transforma metadados em tokens exemplificados na imagem abaixo, escolhendo para o nosso contexto, somente a substituição dos tokens: *Url, Mention, Hashtag, Reserved e Emoji.*

Nome	**Código**
URL	URL
Mention	$MENTION$
Hashtag	$HASHTAG$
Reserved Words	$RESERVED$
Emoji	$EMOJI$
Smiley	$SMILEY$
Number	$NUMBER$

[7] https://github.com/s/preprocessor

Tabela 3.1 Exemplo de Opções de Tokens do PreProcessor

A partir deste pré-processamento, realizamos o processo de Tokenização (Seção 2.2.2.2), e posteriormente, criamos a representação *bag-of-words,* seguindo com os dados pré-processados a fase de criação do nosso modelo de classificação.

3.3.2 Técnica de Classificação

Com o conjunto de dados já pré-processados, diversos experimentos foram realizados, buscando descobrir a melhor técnica de classificação para o nosso domínio (Ver capítulo 4).

Para auxiliar no processo de classificação, utilizamos durante toda essa fase do nosso projeto, o *Scikit-learn*[8]*,* biblioteca para aprendizagem de máquina de código aberto para linguagem de programação *Python.* Essa biblioteca, já inclui diversos algoritmos de aprendizagem de máquina, incluindo os algoritmos utilizados neste trabalho (Seção 2.2.1.2), sendo projetada para interagir com as bibliotecas Python numéricas e científicas NumPy e SciPy. No Capítulo 4, daremos detalhes técnicos sobre os dados e os algoritmos utilizados em nosso estudo, além da métrica de desempenho referente a cada algoritmo.

3.3 Extração de Informação dos Dados

Na última etapa da nossa solução, abordaremos a fase de extração de informação dos nossos tweets. Nesta etapa, utilizamos o *NER e o CRF,* dos quais falamos nas seções 2.3.1 e 2.3.2.

Como da mesma forma que a etapa de classificação, a etapa de extração utiliza dados previamente treinados, como ilustra a figura 3.3, utilizamos o subconjunto de tweets da classe *Eventos* (Seção 3.2), para, através de um processo manual de nomeação de entidades, criarmos a base de treinamento para a extração das informações e criação do nosso modelo.

Buscando extrair as entidades de *Quando*, *Onde* e o *Motivo* do evento de trânsito, realizamos o processo manual de nomeação de entidades utilizando de rótulos pré-definidos:

- B-TIME: Inicio da informação da hora
- I-TIME: Continuação da informação da hora
- B-PLACE: Inicio da informação do local
- I-PLACE: Continuação da informação de local
- B-MOTIVATION: Inicio da informação do motivo do evento
- I-MOTIVATION: Continuação da informação do motivo do evento

[8] http://scikit-learn.org/stable/index.html

Através da base de entidades e do módulo da biblioteca do Scikit-learn, o *Sciki-learn.crfsuite*[9]*,* utilizamos o *CRF,* para a criação do modelo de extração de informação em nosso conjunto de dados.

No capítulo 4, traremos os resultados obtidos pelo nosso modelo, referentes a essa base criada e ao uso do CRF, citados acima, finalizando todo o fluxo ilustrado nas imagens citadas.

[9] https://sklearn-crfsuite.readthedocs.io/en/latest/index.html

Capítulo 4

Experimentos e Resultados

Buscando definir o melhor modelo e as melhores métricas de desempenho para nosso processo de classificação e extração de informação sobre trânsito, um conjunto de experimentos foi realizado em nosso trabalho.

Neste capítulo, daremos informações mais detalhadas sobre o conjunto de dados utilizado para nossos experimentos, os algoritmos de aprendizagem de máquina utilizados e os resultados obtidos em nossa aplicação, referentes a classificação e a extração de informação.

4.1 Experimentos

O conjunto de dados utilizado neste trabalho pertence a informações geradas por diferentes canais do twitter.

Para formar esse conjunto de dados usado, escolhemos *22 canais do Twitter,* distribuídos em grandes cidades do Brasil. Como mostra a tabela abaixo:

Cidades	**Canais**
Manaus	@TransitoManaus, @Manaustrans
Porto Alegre	@TransitoPOARS, @EPTC_POA
Belo Horizonte	@transitobhzmg, @Transito98FM, @OficialBHTRANS
São Paulo	@ecopistas, @radiotransitofm, @CETSP_, @_ecovias
Rio de Janeiro	@TransitoRioRJ, @OperacoesRio
Recife	@CTTU_Recife, @cbntransito, @jctransito
Curitiba	@TransitoSetran, @ecovia
Vitória	@cbnvitoria
Natal	@156Natal, @viacertanatal
Salvador	@Transalvador1

Tabela 4.1 Cidades e canais do Twitter

Através desses canais e do coletor apresentado na seção 3.1, capturamos cerca de *3000 tweets* para cada canal, rotulando manualmente após a captura, cerca de *300*

tweets para cada um deles, divididos entre *100 Tweets* para classe *Eventos, 100 Tweets* para a classe *Informações* e *100 Tweets* para a classe *Outros*, totalizando uma base com *6.678 Tweets*.

Como esse conjunto de dados foi utilizado para a classificação e para a extração de informação, detalharemos abaixo, o uso dos dados através de cada um:

- **Classificação**: Para a classificação, duas abordagens foram realizadas:
    - *1º abordagem:* Na primeira abordagem, dividimos o conjunto de dados entre treinamento e teste de forma manual, separando 16 canais para treinamento das cidades: Manaus, Belo Horizonte, Porto Alegre, Rio de Janeiro, Recife, Curitiba e Natal e 6 canais para teste das cidades: Salvador, São Paulo e Vitória, obtendo os seguintes subconjuntos de dados, ilustrados na tabela 4.2.

Classe	**Treinamento**	**Teste**
Eventos	1560	611
Informação	1747	629
Outros	1549	582

Tabela 4.2 Quantidade de Tweets em cada classe - abordagem 1 - Classificação

    - *2º abordagem:* Nesta segunda abordagem, a partir de todo o conjunto de dados pré-processado (6678 tweets), utilizamos a função do scikit-learn, o *train_test_split,* que divide a base em um conjunto de treinamento e outro de teste, separando o conjunto em 70% para treinamento e 30% para teste, realizando a classificação e análise das métricas de desempenho da mesma forma que na abordagem um.

	70%	**30%**
Conjunto de dados	4674	2004

Tabela 4.3 Quantidade de Tweets em cada classe - abordagem 2 - Classificação

- **Extração:** Para a extração, utilizamos os 2171 tweets de Eventos, realizando da mesma forma, duas abordagens para análise e criação do nosso modelo.

- *1º abordagem:* Na primeira abordagem, utilizamos o *Cross-validation*[10]*,* onde se particiona os dados em subconjuntos disjuntos entre si e os separa entre conjuntos de treinamento e teste em diversas iterações, passando como base 746 Tweets rotulados manualmente. Com o conjunto de dados montado, observamos a presença de *12663 entidades (tokens),* que foram utilizados nas duas abordagens e distribuídos da seguinte forma:

Rótulos	**Quantidade**
B-TIME	220
I-TIME	11
B-PLACE	585
I-PLACE	1590
B-MOTIVATION	599
I-MOTIVATION	1084
O	8574

Tabela 4.4 Quantidade de entidade por rótulo - Extração

- *2º abordagem:* Nesta abordagem, dividimos o conjunto de dados em treinamento e teste, como ilustra a tabela 4.5, criando-se o modelo a partir do conjunto de treinamento e posteriormente verificando o desempenho do modelo através do conjunto de teste. Neste caso, foram utilizados *600 Tweets* para treinamento e *146 tweets* para teste.

	Treinamento	**Teste**
Conjunto de Dados	600	146

Tabela 4.5 Quantidade de Tweets para treinamento/teste, caso 2 - Extração

A partir desses conjuntos de dados dispostos, na próxima seção, abordaremos os resultados obtidos para essas abordagens e para as diferentes técnicas e algoritmos já citados em seções anteriores.

[10] https://en.wikipedia.org/wiki/Cross-validation_(statistics)

4.2 Resultados

Nesta seção, falaremos dos resultados obtidos na criação dos nossos modelos para classificação e extração de informação, utilizando, como métrica de desempenho, a *média das classes (Eventos, Informações e Outros)* para a precisão, cobertura, f-measure e a acurácia (Seção 2.2.1.3).

4.2.1 Classificação

Como falado na seção anterior, duas abordagens foram realizadas para avaliação do modelo referente a cada algoritmo de classificação escolhido. Além dessas abordagens, foram realizados experimentos considerando apenas o modelo de representação dos nossos elementos pelo modelo de *bag-of-words,* e considerando a representação dos nossos elementos por uma matriz de representação por *Frequência do termo - inversa dos documentos* (TF-IDF), abordados na seção 2.2.2.1.

Na tabela 4.6, estão os resultados obtidos com o uso da primeira abordagem, citada na seção anterior, levando em conta as duas representações escolhidas e a média das classes (Eventos, Informações e Outros), para cada algoritmo utilizado.

Versão	**TF-IDF**	**Precisão**	**Cobertura**	**F-Measure**	**Acurácia**
MultinomialNB	Não	0.77	0.77	0.77	0.7744
SVM	Não	0.79	0.79	0.79	0.7925
Logistic Regression	Não	0.83	0.83	0.83	0.8298
XG Boost	Não	**0.87**	**0.86**	**0.86**	**0.8644**
Decision Tree	Não	0.75	0.75	0.75	0.7513
MultinomialNB	Sim	0.77	0.76	0.76	0.7601
SVM	Sim	**0.80**	**0.79**	**0.79**	**0.7881**
Logistic Regression	Sim	0.78	0.77	0.77	0.7683
XG Boost	Sim	0.67	0.58	0.54	0.5823
Decision Tree	Sim	0.68	0.68	0.68	0.6811

Tabela 4.6 Métricas de desempenho da classificação utilizando a média das classes para cada algoritmo e representações - Abordagem 1

Na segunda representação, utilizando a frequência dos termos, obtivemos uma diminuição da acurácia em todos os algoritmos, tendo o *SVM* como o algoritmo de melhor desempenho, com cerca de *79% de acurácia* e uma *precisão de 80%.* Porém, como esperado, para a primeira representação, a melhor abordagem encontrada foi com o uso do XG Boost, com cerca de *86% de acurácia* e com uma *F-measure de também 86%,* sendo o melhor algoritmo para ambas as representações da primeira abordagem (Seção 4.1).

Versão	**TF-IDF**	**Precisão**	**Cobertura**	**F-Measure**	**Acurácia**
MultinomialNB	Não	0.85	0.84	0.84	0.8433
SVM	Não	0.88	0.88	0.88	0.8842
Logistic Regression	Não	**0.89**	**0.89**	**0.89**	**0.8932**
XG Boost	Não	0.86	0.86	0.86	0.8577
Decision Tree	Não	0.84	0.84	0.84	0.8388
MultinomialNB	Sim	0.85	0.85	0.85	0.8463
SVM	Sim	**0.87**	**0.86**	**0.86**	**0.8552**
Logistic Regression	Sim	0.84	0.83	0.83	0.8298
XG Boost	Sim	0.80	0.77	0.77	0.7734
Decision Tree	Sim	0.70	0.70	0.70	0.6981

Tabela 4.7 Métricas de desempenho da classificação utilizando a média das classes para cada algoritmo e representações - Abordagem 2

Na segunda abordagem, utilizando o método de divisão do sckit-learn, o *train_test_split,* obtivemos resultados bem melhores do que o da primeira abordagem e em ambas as representações, como ilustrado na tabela 4.7.

Na primeira representação, sem o uso do TF-IDF, obtivemos um desempenho acima de 80% em todas as métricas de cada algoritmos treinado, obtendo um melhor resultado com o algoritmo de *Regressão Logística* com aproximadamente *89% de acurácia.*

Como ilustrado no trabalho [22], para ambas as abordagens do nosso trabalho, com o uso da representação de frequência de termo, obtivemos os melhores resultados para o algoritmo de SVM.

Desta forma, após analisar as métricas de desempenho em cada algoritmo proposto, verificamos que o algoritmo de *Regressão Logística* utilizado na segunda abordagem sem a representação por TF-IDF, teve o melhor desempenho entre todos os experimentos realizados.

Como esperado, o XG Boost obteve um desempenho muito bom para a primeira abordagem, com *86% de acurácia*, obtendo um desempenho bem próximo ao da regressão.

4.2.2 Extração de Informação

Através do *CRF,* utilizando diferentes *features* para o treinamento e as duas abordagens citadas na seção anterior para extração de informação, realizamos vários experimentos importantes, obtendo resultados interessantes para nossas métricas de desempenho, como mostram as tabelas 4.10 e 4.11.

Rótulo	**Precisão**	**Cobertura**	**F-measure**
B-TIME	0.944	0.927	0.936
I-TIME	0.000	0.000	0.000
B-PLACE	0.780	0.619	0.690
I-PLACE	0.694	0.584	0.634
B-MOTIVATION	0.769	0.546	0.639
I-MOTIVATION	0.719	0.609	0.659
O	0.839	0.910	0.873
Média	**0.806**	**0.812**	**0.806**

Tabela 4.8 Métricas de desempenho - Extração - Abordagem 1 - Cross-Validation

Na primeira abordagem, utilizamos o Cross-validation para treinar e validar o modelo de extração. Como podemos ver na tabela, o rótulo *I-TIME* obteve um valor zerado em todas as métricas, isso se deu pelo fato deste rótulo não ter elementos suficientes para que o modelo aprendesse com os dados de entrada, já que o rótulo só possui 11 entidades rotuladas, como ilustra a tabela 4.4.

Apesar de obtermos bons resultados na precisão desta abordagem, a cobertura na maioria dos rótulos foi ruim, isso se deve à baixa frequência com que o modelo encontra exemplos dos rótulos no conjunto de dados, sendo necessário

uma quantidade maior de dados de treinamento para ajustar e melhorar essa métrica e o aprendizado.

Rótulo	**Precisão**	**Cobertura**	**F-measure**
B-TIME	0.983	0.967	0.975
I-TIME	0.000	0.000	0.00
B-PLACE	0.808	0.669	0.732
I-PLACE	0.751	0.616	0.677
B-MOTIVATION	0.755	0.561	0.643
I-MOTIVATION	0.733	0.663	0.696
O	0.873	0.928	0.900
Média	**0.843**	**0.848**	**0.843**

Tabela 4.9 Métricas de desempenho - Extração - Abordagem 2

Com a segunda abordagem, obtivemos uma média um pouco maior quando comparada a da primeira, porém, os dados se mantiveram muitos próximos e apesar de serem bons resultados, como uma média de aproximadamente *80% de F-measure* na primeira abordagem e *84%* para a segunda abordagem. Talvez seja necessário para a melhoria do modelo, um conjunto de dados de treinamento um pouco maior e algumas novas features para auxiliar na classificação do *CRF.*

Capítulo 5

Conclusão

Neste trabalho apresentamos uma solução para classificação e extração de eventos de trânsito a partir de dados gerados pelo Twitter. Para tal, abordamos as fases da captura, classificação e extração dos dados, utilizando diferentes técnicas de aprendizagem de máquina e por fim, demonstrando diferentes métricas de desempenho para nossos modelos de classificação e extração de entidades.

O classificador apresentado como solução de melhor desempenho, foi o modelo gerado utilizando *Regressão Logística,* com uma acurácia de aproximadamente *88%.* Na extração de informação, obtivemos resultados interessantes, com um *F-measure* de *80%* na primeira abordagem e cerca de *84%* na segunda, tendo porém, valores de cobertura não tão significativos.

Buscando melhorar o classificador e o extrator, como trabalho futuro, é interessante novos experimentos utilizando diferentes parâmetros para os algoritmos de classificação, e uma base maior de entidades rotuladas para o modelo de extração, além do uso de novas features para o *CRF,* buscando resolver o problema de cobertura e melhorar o desempenho do modelo.

Referências

[1] David Jäderberg. “Sentiment and topic classification of messages on Twitter and using the results to interact with Twitter users”. Uppsala University, 2016

[2] Vinícius Pazzini , Tiago Schenkel , Mikael Poetsch e Ricardo Matsumura Araujo. “Classificação de Prioridade de Tweets utilizando Máquinas de Vetor de Suporte”. Centro de Desenvolvimento Tecnológico UFPel, 2013

[3] Pedro Domingos, “A Few Useful Things to Know about Machine Learning”. University of Washington, 2012

[4] M. Mohri, A. Rostamizadeh, and A. Talwalkar, Foundations of Machine Learning. The MIT Press, 2012

[5] J E T Akinsola, “Supervised Machine Learning Algorithms: Classification and Comparison”. Babcock University, 2017

[6] S. Idowu, “Machine Learning in Pervasive Computing”. Lulea University of Technology, 2013

[7] Sandeep G Panchal , Prof. R. S. Apare, “Real Time Traffic Detection using Twitter Tweets Analysis”. College of Engineering Pune&Savitribai Phule Pune University, 2017

[8] Salas, Angelica, Georgakis, Panagiotis; Petalas, Yannis, “Incident Detection Using Data from Social Media”. University of Wolverhampton. 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC): Workshop

[9] Mrs. Kavita Sawant , Miss. Shital Pawar, Miss. Poonam Jadhav, Mrs. Sayali Vidhate, Mrs. Nirasha Bule, Mrs. Snehal Patil, “Traffic Detection from Real Time Twitter Stream Analysis and Navigation System”. 2017 IJESC

[10] Eleonora D’Andrea, Pietro Ducange, Beatrice Lazzerini, Francesco Marcelloni,”Real-Time Detection of Traffic From Twitter Stream Analysis”. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 16, NO. 4, AUGUST 2015

[11] Guia do Desenvolvedor, “Amazon Machine Learning” https://docs.aws.amazon.com/pt_br/machine-learning/latest/dg/machinelearning-dg.pdf#binary-classification

[12] Multi-Label Classification,
https://en.wikipedia.org/wiki/Multi-label_classification

[13] Yianly Liu, Yourong Wang, Jian Zhang, “New Machine Learning Algorithm: Random Forest”. Basic Teaching Department, Tangshan College, Tangshan, Hebei, China, 2012

[14] Ron Kohavi, “Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid”. Data Mining and Visualization Silicon Graphics, Inc. 2011

[15] Fabrizio Sebastiani, “Machine learning in automated text categorization”. Consiglio Nazionale delle Ricerche, Pisa, Italy, 2002

[16] Wallach, Hanna M. "Topic modeling: beyond bag-of-words." Proceedings of the 23rd international conference on Machine learning. ACM, 2006.

[17] Jakub Piskorski and Roman Yangarber, “Information Extraction: Past, Present and Future”. Multilingual Information Extraction and Summarization 11, Theory and Applications of Natural Language Processing, Springer-Verlag Berlin Heidelberg 2013

[18] Lívia Pimentel, “INFOMOVIE: SISTEMA DE EXTRAÇÃO DE INFORMAÇÃO COM INTERFACE PARA LINGUAGEM NATURAL”. UFRJ ABRIL de 2013

[19] John Lafferty, Andrew McCallum, and Fernando C.N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data", . June 2001.

[20] Tianqi Chen and Carlos Guestrin, “XGBoost: A Scalable Tree Boosting System”. University of Washington, 2016

[21] Stephan Dreiseitl and Lucila Ohno-Machado, “Logistic regression and artificial neural network classification models: a methodology review”. 2003

[22] Weimin Xue, Weitong Huang, and Yuchang Lu. Application of svm in web page categorization. In Grandular Computing, 2006 IEEE International Conference on, pages 469- 472.IEEE, May 2006

[23] Ana C. Lorena, André C. P. L. F. de Carvaho, “Introdução às Máquinas de Vetores Suporte (Support Vector Machines)”. São Carlos, 2003