UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

TRABALHO DE CONCLUSÃO DE CURSO

DESCOBERTA DE CONHECIMENTO EM REDES SOCIAIS E BASES DE DADOS PÚBLICAS

Autora:
Fernanda DE CAMARGO MAGANO

Supervisora:
Kelly Rosa BRAGHETTO

10 de dezembro de 2016

Resumo

O presente trabalho de conclusão de curso teve por objetivo aplicar as etapas de descoberta de conhecimento em bases de dados no domínio do cinema, cruzando informações coletadas da rede social Twitter com a base de dados *Internet Movie Database* (IMDb). Assim, foram realizadas atividades de coleta, pré-processamento (para extrair as informações úteis dentre as presentes em um grande volume de dados) e mineração de dados, utilizando-se como classificadores: *Naive Bayes* e SVM (Support Vector Machine). Foi realizada a mineração de opiniões para obtenção do sentimento expresso na frase analisada, fazendo a divisão nas categorias ruído e não ruído, o qual se subdivide em muito positivo, positivo, neutro, negativo e muito negativo. Além disso, os dados foram validados com métricas clássicas de recuperação de informação, como precisão, revocação, medida f e acurácia para que pudesse ser analisada a qualidade da classificação. Por fim, os dados mais relevantes foram armazenados em um banco de dados orientado a grafos, permitindo também a visualização dos resultados e encontrar padrões que geram conhecimento.

Palavras-chave: descoberta de conhecimento, mineração de dados, análise de sentimento, aprendizado de máquina, classificação, banco de dados, grafos.

Sumário

Parte I

Parte Objetiva

1. Introdução

1.1 CONTEXTUALIZAÇÃO

As mídias sociais representam uma forma de comunicação muito utilizada nos dias atuais. Cada vez mais pessoas, de diferentes faixa etárias e opiniões, têm acesso a essas mídias e apresentam interesse em usá-las. Para citar algumas: *Twitter*, *Facebook*, *Whatsapp*, *LinkedIn*, entre outras, ocupam um espaço importante na sociedade.

De acordo com as informações do Pew Research Center [1], 74% dos adultos que usavam internet em janeiro de 2014 acessavam sites de redes sociais. Além disso, na última década houve um grande salto no número de jovens de 18-29 anos que utilizam redes sociais. Em setembro de 2013, 90% dos jovens nessa faixa etária já utilizavam redes sociais. Eles eram apenas 9% em fevereiro de 2005.

Outras estatísticas mais recentes, coletadas do Smartinghts [2], comprovam a importância das redes sociais na atualidade: em janeiro de 2016, a população mundial contava com 7,395 bilhões de pessoas, sendo 3,149 bi usuários da internet e 2,307 bi usuários ativos de mídias sociais.

Dessa forma, mais de um terço da população mundial participa ativamente de redes sociais, expondo e compartilhando ideias e opiniões, fazendo comentários positivos ou negativos sobre as mais diversas experiências de suas vidas.

Diferentes tipos de informações relevantes podem ser automaticamente extraídas a partir do enorme volume de dados gerados diariamente nas redes sociais, por meio de técnicas como as de mineração de dados e opiniões.

1.2 OBJETIVOS

Com o impacto das redes sociais na sociedade em que estamos inseridos e o avanço de técnicas de aprendizado de máquina, a opinião das pessoas acerca de determinados assuntos pode ser obtida para melhorar a qualidade de serviços existentes. O presente trabalho estuda como diferentes tipos de técnicas de mineração de dados podem ser combinadas para a descoberta de conhecimento a partir do cruzamento de informações coletadas de mídias sociais e de bases de dados públicas.

Para isso, foi escolhido um domínio com bastante disponibilidade de dados abertos online: o domínio do cinema. Foi utilizada a base de dados do *Internet Movie Database* (IMDb [3]), que disponibiliza informações sobre os filmes, e dados extraídos de *posts* na rede social *Twitter*.

O Twitter [4] é uma das redes sociais mais usadas na atualidade, contando com 317 milhões de usuários ativos no terceiro trimestre de 2016, de acordo com Statista [5].

O *Twitter* caracteriza-se por apresentar mensagens com limite de 140 caracteres chamadas de *tweets*, os quais possuem *hashtags* que são palavras-chave para a informação contida no *tweet*. Esses dados dos *tweets* são públicos e podem ser coletados gratuitamente em tempo real, isto é, são fornecidos dados do momento da coleta originados de diferentes países do mundo. Se o desejado é fazer a análise desses dados por um período de tempo, cabe ao desenvolvedor o armazenamento desse volume de informações.

No presente trabalho foram aplicadas técnicas de aprendizagem de máquina para identificar a polaridade dos *tweets*, ou seja, para classificar o conteúdo dos *posts* como sendo uma opinião positiva ou negativa. A partir da detecção da polaridade foi possível estudar as preferências dos usuários com respeito aos filmes e fazer estudo por regiões, comparando o número de *tweets* por estados, por exemplo, e estudar interesses e influências das mídias e indústria cinematográfica nesses locais.

Foi utilizado um banco de dados orientado a grafos para armazenar e relacionar esses conhecimentos descobertos e facilitar a visualização das informações, visto que padrões ficam mais visíveis pelo uso de cores em grafos, em complemento aos dados de tabelas explicativas.

O domínio do cinema é apenas uma das possíveis áreas de aplicação para as técnicas estudadas neste trabalho. É possível citar outros exemplos de domínios: epidemias, mobilidade urbana, política e economia. A escolha do domínio se deve principalmente à disponibilidade de dados, como o acesso à base pública do IMDb. Se o domínio de epidemias fosse escolhido, por exemplo, dados de pacientes são confidenciais e as informações teriam que ser extraídas essencialmente pelos *tweets*. Não haveria uma base pública para cruzar as informações.

1.3 ESTRUTURAÇÃO DA MONOGRAFIA

A seção 2 tratará da fundamentação teórica, explicando conceitos importantes da área de aprendizagem de máquina, desde termos frequentemente usados, até uma descrição de alguns classificadores, como *Naive Bayes* e *SVM* (Support Vector Machine).

A seção 3 é mais aplicada, explicando detalhadamente atividades do procedimento de descoberta de conhecimento usando grandes volumes de dados. Para citar algumas: coleta dos dados, construção do *Corpus*, validação cruzada, classificação dos dados e a construção do *workflow*.

A seção 4 descreve o banco de dados orientado a grafos desenvolvido, ilustrando alguns resultados obtidos.

Por fim, a seção 5 apresenta as considerações finais e contém indicações de alguns possíveis trabalhos futuros.

2. Fundamentação Teórica

2.1 MINERAÇÃO DE DADOS E DESCOBERTA DE CONHECIMENTO EM BANCOS DE DADOS

Com a vasta quantidade de informações produzidas diariamente na internet, extrair informações relevantes, isto é, obter conhecimento a partir de um volume massivo de dados, de forma automatizada, torna-se fundamental. Se faz necessária a existência de métodos e ferramentas que permitam extrair esse conhecimento e isso é estudado no campo de descoberta de conhecimento em bases de dados (KDD - Knowledge Discovery in Databases [6]).

O principal desafio dessa área é transformar a enorme quantidade de dados em resultados resumidos e visuais, seja através de artigos, seja construindo modelos preditivos para os dados. O KDD se preocupa com o processo como um todo para obtenção de conhecimento, desde a coleta, processamento, armazenamento dos dados e questões de eficiência, até como os dados serão interpretados e visualizados. É um campo bastante interdisciplinar, abrangendo conhecimentos de aprendizado de máquina, reconhecimento de padrões, inteligência artificial, estatística, entre outros.

São aplicados métodos de mineração de dados, com o intuito de analisar os dados e utilizar algoritmos para enumerar padrões. A mineração é uma parte fundamental do processo KDD, permitindo correlacionar campos presentes em bases de dados imensas. Por trabalhar com quantidade massiva de informações, a redução do volume a ser minerado é imprescindível e pode ser realizada por intermédio de alguns métodos e técnicas, como a seleção de características e redução de dimensões.

A mineração de dados tem sido utilizada por grandes empresas para analisar hábitos de consumo de seus clientes, por exemplo, entendendo quais produtos são mais consumidos, permitindo desenvolver sistemas de recomendação baseando-se no histórico de produtos que os usuários costumam comprar.

2.2 MINERAÇÃO DE OPINIÃO

Mineração de opinião ou análise de sentimento (*sentiment analysis*) é definida como o estudo computacional que envolve opiniões, sentimentos e emoções expressas em um conjunto de dados [7]. Consiste na identificação da opinião expressa num conjunto textual e a classificação em categorias de acordo com a sua polaridade, isto é, baseado no sentimento positivo ou negativo (remete à ideia de polos, opostos) no texto analisado ou de acordo com alguma escala (com variação na intensidade de positivo e negativo).

É um desafio de processamento de linguagem natural, com enorme valor em aplicações práticas, o que explica o crescente aumento de estudo nessa área.

Opiniões são subjetivas, variam de acordo com experiências vivenciadas pelos indivíduos e refletem na sua forma de pensar e sentir. Mesmo sendo algo tão parcial, a importância da análise desse tipo de informação é real. Grandes empresas e organizações têm interesse em entender as necessidades de seus clientes e motivos de principais insatisfações. Também desejam saber o que devem fazer para melhorar seu produto ou serviço. Isso se aplica aos mais diversos ramos, para citar alguns: indústria cinematográfica, política, esportes, economia, entre outros.

Alguns conceitos importantes em análise de sentimentos são: alvo e opinião. O objeto ou alvo pode ser uma pessoa, produto, organização ou assunto sobre o qual é expressa a opinião. Esta, por sua vez, indica uma visão que pode ser positiva ou negativa. Assim, a opinião possui uma orientação, também chamada de polaridade da opinião ou orientação semântica.

2.3 CLASSIFICAÇÃO DE TEXTO

Sejam $C = \{c_1, c_2, \ldots, c_j\}$ um conjunto de classes ou categorias e X o espaço de documentos. Uma parte dos documentos pode ser usada como conjunto *T* de treinamento, utilizado para o aprendizado do classificador, e o restante como de teste, sobre o qual o classificador já treinado fará a divisão dentro das classes desejadas. O conjunto *T* é composto por exemplos rotulados no formato (x_i, y_i), onde x_i é o dado e y_i é seu rótulo associado, veja [8] para mais detalhes.

O objetivo é produzir um classificador f - dentro do universo de classificadores F possíveis de serem gerados por um algoritmo de aprendizado de máquina - capaz de predizer o rótulo de novos dados. Para isso, é levado em consideração o desempenho obtido por f com os dados de *T*, de modo a se obter bom desempenho com novos dados do mesmo domínio.

Esse tipo de aprendizado é chamado de supervisionado, pois há intervenção humana na parte de rotulação manual de um conjunto de treinamento que será utilizado como entrada do algoritmo [9]. Esse conjunto servirá para que padrões sejam aprendidos pelo método supervisionado Γ, produzindo como saída a função de classificação γ aprendida: $\Gamma(T) = \gamma$. Uma vez obtida a função γ, podemos aplicá-la ao conjunto de teste para obter resultados.

A meta é alcançar alta acurácia na classificação do conjunto de teste. Um cuidado que deve-se tomar é não testar o classificador com os mesmos dados de treinamento, já que haveria 100% de acurácia por simples memorização dos rótulos. Isso não serviria como validação do classificador. Também deve-se evitar sobreajuste (*overfitting*), isto é, um modelo que se ajusta bem a um conjunto específico de treinamento e de teste, apresentando alta precisão quando testado sobre esse conjunto, mas com diminuição significativa da precisão ao ser testado com outros dados.

Alguns termos bastante comuns no contexto de classificação de texto e aprendizado de máquina serão definidos a seguir.

2.3.1 Acurácia, precisão e cobertura

Para sabermos se o classificador é robusto ou não em suas predições, são utilizadas algumas métricas, como a acurácia, cuja fórmula está abaixo:

$$Acurácia = \frac{V_p + V_n}{V_p + F_p + V_n + F_n}$$

Onde:

V_p : verdadeiros positivos
V_n : verdadeiros negativos
F_p : falsos positivos
F_n : falsos negativos

Contudo, apenas a acurácia não traz informação suficiente, não prova a robustez. Com isso, são usadas as medidas de precisão, cobertura e medida *F* para dar maior embasamento.

Uma forma clara de representar os resultados de um classificador com duas ou mais classes é utilizar uma tabela de contingência, também denominada por matriz de confusão, como mostrada na tabela 2.1. Para uma boa classificação, deseja-se que os valores da diagonal principal da matriz sejam os maiores possíveis, enquanto que para os demais elementos sejam os menores possíveis, pois indicam predições erradas.

TABELA 2.1: Matriz de confusão

	Positivo(predição)	**Negativo(predição)**
Positivo(real)	V_p	F_n
Negativo(real)	F_p	V_n

Precisão (P) é uma medida de relevância dos resultados e refere-se a uma baixa taxa de falsos positivos:

$$P = \frac{V_p}{V_p + F_p}$$

Recall (R) - traduzido como cobertura ou revocação - é uma medida de quantos resultados relevantes foram retornados, dentre os possíveis, referindo-se à baixa taxa de falsos negativos:

$$R = \frac{V_p}{V_p + F_n}$$

O que se busca é alta precisão e alta revocação para que sejam retornados muitos resultados rotulados da maneira correta. A média harmônica da precisão e do recall é dada pela medida F_1:

$$F_1 = \frac{2PR}{P + R}$$

2.3.2 Stopwords

Stopwords são palavras muito comuns em textos, como artigos, preposições, pronomes e que, justamente por serem tão comuns, não agregam significado à frase ou texto analisado. Assim, elas podem ser removidas na fase de limpeza dos dados.

2.3.3 Tf-idf

A Tf-idf *Term frequency-inverse document frequency* trata-se de uma medida utilizada em mineração de textos para dar peso às palavras presentes nos documentos de uma coleção. Quanto mais frequente for uma palavra em um documento, maior sua importância; contudo, se for muito frequente na coleção como um todo, aquela palavra não é tão relevante assim. Portanto, essa medida destaca a raridade do termo (idf) na coleção e a frequência dele (tf) no documento. Assim, ela pode detectar *stopwords*.

Dado um termo t, a frequência do termo é calculada por:

$$Tf(t) = \frac{\text{Número de ocorrências do termo } t \text{ no documento}}{\text{Quantidade de termos no documento}}$$

Essa divisão pelo número de termos no documento se deve ao fato de ser mais provável um número maior de ocorrências de um termo em um texto grande. Assim, é feita essa normalização, calculando-se a frequência relativa.

O idf é uma medida da importância do termo na coleção. Termos muito frequentes têm seu peso diminuído, enquanto que o inverso ocorre para termos raros. O idf é dado por:

$$idf(t) = log_e(\frac{\text{Número total de documentos}}{\text{Número de documentos que possuem o termo } t})$$

Para exemplificar, considere o documento abaixo que é um *tweet* coletado sobre o filme Alice Através do Espelho:

"I really really really want to see Alice Through the Looking Glass"

O tf da palavra *really* é $\frac{3}{12} = \frac{1}{4} = 0,25$ sendo 12 a quantidade de termos do documento, se cada *tweet* for considerado como um documento. Agora considere, hipoteticamente, que há um milhão de documentos e que a palavra *really* aparece em 100 mil deles.

Assim, o idf é $log_e(\frac{1*10^6}{1*10^5}) = log_e 10 \approx 2,30$. O tf-idf é dado pela multiplicação do $tf * idf = 0,25 * 2,30 = 0,575$.

2.3.4 Capitalização

Uma outra estratégia usada em processamento de texto é converter todas as palavras para letras minúsculas para que palavras como *Movie*, *movie*, *MOVIE* e outras variações sejam consideradas iguais.

2.4 ALGORITMOS DE CLASSIFICAÇÃO

Para realizar a mineração de opinião se faz necessário o uso de classificadores que permitam dividir em categorias os dados coletados. Há uma vasta gama de classificadores, para citar alguns: *Naive Bayes*, Máquina de Vetores de Suporte (SVM de Support Vector Machine), Máxima Entropia, K-Vizinhos mais próximos (KNN de K-Nearest Neighbors), entre outros.

Lunardi et al. [10] fizeram uma compilação e comparação dos principais algoritmos de classificação supervisionada na área de análise de sentimentos. Na avaliação feita pelos autores, os algoritmos que obtiveram melhores resultados dentre os trabalhos analisados foram o SVM e o *Naive Bayes*.

No mesmo artigo, mostra-se também o domínio em que foram realizados os estudos. No contexto de revisão de filmes e de dados coletados do *Twitter* foram utilizados os algoritmos *Naive Bayes* e SVM, obtendo-se boas acurácias (no geral acima de 80%).

Com isso, conclui-se que ambos os classificadores são boas escolhas para o domínio de aplicação abordado neste trabalho de conclusão.

2.5 EXTRAÇÃO DE CARACTERÍSTICAS

Dado um conjunto de documentos textuais é necessário realizar um processo de transformação sobre esses dados brutos de modo a permitir que o classificador possa utilizá-los. Assim, esses dados são transformados em um vetor numérico de *features* (características) - que correspondem às frequências de ocorrência de cada *token* – e esse processo de transformação é chamado de vetorização.

Um *token* é uma instância de uma palavra em algum documento e trata-se de uma unidade semântica útil na etapa de processamento [9].

Seja x_i um documento e y_i a respectiva classe ou categoria a qual pertence. Dado um conjunto n de documentos, cada um deles é composto por m características, isto é, $x_i = (x_{i1}, x_{i2}, \ldots x_{im})$, como pode ser ilustrado pela matriz abaixo:

$$
\begin{array}{c}
 \\
x_1 \\
x_2 \\
\vdots \\
x_n
\end{array}
\begin{array}{c}
\begin{array}{cccc}
\multicolumn{4}{c}{atributos}
\end{array} \\
\left(\begin{array}{cccc|c}
x_{11} & x_{12} & \ldots & x_{1m} & y_1 \\
x_{21} & x_{22} & \ldots & x_{2m} & y_2 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
x_{11} & x_{12} & \ldots & x_{nm} & y_n
\end{array}\right)
\end{array}
\begin{array}{c}
 \\
 \\
 \\
 \\
 \\
classes
\end{array}
$$

O que costuma ser feito para armazenar a matriz na memória e acelerar cálculos é utilizar representações de matrizes esparsas. Isso porque normalmente o número de características presentes em um documento é bem menor do que o número total presente na coleção e, assim, a matriz possui vários zeros. Dessa foma, basta armazenar a informação não nula.

Foi utilizado o modelo de *"Bag of Words"* que consiste das etapas de **tokenização**, **contagem** de ocorrências dos *tokens* em cada documento e **normalização**, atribuindo pesos diferentes dependendo da frequência da palavra num documento. Contudo, a posição relativa entre as palavras não é considerada.

2.6 SVM

O algoritmo de Máquina de Vetores de Suporte (*Support Vector Machine*) trata-se de uma técnica de aprendizado supervisionado. Os documentos a serem analisados são representados por pontos ou vetores no espaço. Trata-se de um classificador efetivo com espaços vetoriais de várias dimensões.

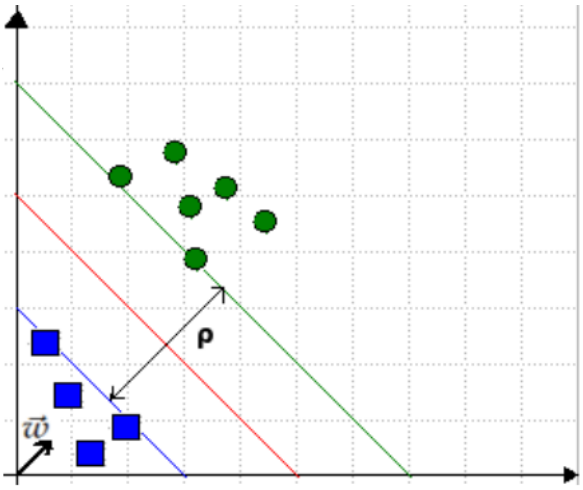
Seja f o classificador, define-se como risco esperado ([8]) o erro médio cometido por f na classificação dos dados de teste. A relação entre os dados $(x_1, x_2 \dots x_n)$ do domínio e seus rótulos $(y_1, y_2, \dots y_n)$ é dada por uma distribuição conjunta de probabilidades P(x, y). Nenhuma suposição é feita sobre P e sua distribuição é desconhecida no momento do aprendizado. Com um número suficientemente grande de exemplares de treinamento, é possível estimar P.

Considerando o caso com duas classes (bidimensional), tenta-se obter uma superfície de decisão, um hiperplano - o qual é uma reta no caso de duas dimensões - de modo que cada um dos lados representa uma classe diferente. Para obter essa divisão, procura-se a região mais afastada de qualquer ponto dos dados analisados para formar a margem do classificador. Assim, um subconjunto de pontos determina a posição desse divisor e são chamados de vetores de suporte.

O objetivo é tentar maximizar a margem, uma vez que a região próxima ao hiperplano separador está mais suscetível a erros de classificação. Então, com uma margem maior obtemos mais segurança, já que pequenas alterações nos documentos analisados não levarão a uma classificação errônea.

Na figura 2.1 o hiperplano está em vermelho e os vetores de suporte estão em verde e azul. As duas classes são de quadrados e de círculos.

FIGURA 2.1: SVM linear



Baseada na figura 15.3 de [9]

2.7 NAIVE BAYES

Naive Bayes (NB) é um método probabilístico de aprendizado supervisionado, derivado da regra de *Bayes*:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)} \tag{2.1}$$

Sendo *c* a categoria, *d* o documento, $P(c)$ e $P(d)$ probabilidades a priori e $P(c|d)$ a posteriori. A probabilidade a priori não tem informações sobre outros eventos, enquanto que a posteriori é uma probabilidade condicional e considera a ocorrência de um evento.

A probabilidade de um documento *d* (como um *tweet*) estar em determinada categoria *c* é dada pela fórmula:

$$P(c|d) \propto P(c) \prod_{k=1}^{n_d} P(t_k|c) \tag{2.2}$$

onde $P(c)$ é a probabilidade a priori de um documento d estar na categoria c; $P(t_k|c)$ é a probabilidade do termo k do documento ocorrer num documento de classe c.

Naive Bayes assume a hipótese de independência condicional, isto é, os atributos são considerados não correlacionados. Por isso que da fórmula 2.1 para 2.2 pode-se trocar $P(d|c)$ por $\prod_{k=1}^{n_d} P(t_k|c)$

Os termos $< t_1, t_2, \ldots, t_{n_d} >$ são *tokens* presentes no vocabulário usado no processo de classificação, sendo n_d a quantidade de *tokens* do vocabulário presentes no documento *d*, já com a remoção de *stopwords*.

Por exemplo:

"Mogli is such a beautiful movie, totally worth it"

Os *tokens* do documento acima são: $< Mogli, such, beautiful, movie, totally, worth >$ e, assim, $n_d = 6$.

Com o *Naive Bayes* pretendemos identificar a melhor classe para cada documento analisado, isto é, classificar da melhor forma possível. A fórmula usada pela maioria das implementações de NB ([9]) é:

$$c_{map} = argmax[log\widehat{P}(c) + \sum_{k=1}^{n_d} log\widehat{P}(t_k|c)]$$

O símbolo ˆ é utilizado em $P(c)$ e $P(t_k|c)$ porque são estimativas feitas a partir do conjunto de treinamento. A maximização é feita, pois deseja-se obter a melhor classe para o documento. O logaritmo da soma é usado para evitar *underflow*, visto que as probabilidades se tornam números cada vez menores ao se realizar multiplicações sucessivas. O logaritmo pode ser usado, porque a hierarquia das probabilidades é mantida, isto é, a classe com maior probabilidade ao aplicar o *log* também é a classe mais provável para o documento que está sendo classificado se não fosse aplicado o logaritmo.

Para calcular a probabilidade a priori é utilizada frequência relativa. Sejam N o total de documentos e N_c o número de documentos na classe c. Então, $\widehat{P}(c)$ é dado por:

$$\widehat{P}(c) = \frac{N_c}{N}$$

Para a probabilidade condicional de um termo ser de determinada classe, é feito:

$$\widehat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)}$$

Sendo V o vocabulário e T_{ct} é a quantidade de vezes que o termo t ocorre na classe c no conjunto de treinamento.

O 1 somado no numerador e denominador é o "*Laplace Smoothing*" ([9]), usado para evitar probabilidades nulas, ou seja, caso o termo t não tenha ocorrido na classe c. Como em 2.2 é usado um produtório, se uma das probabilidades fosse nula, a probabilidade a posteriori seria nula.

Apesar de ser um método simples e apresentar a hipótese de independência condicional - o que não é semelhante ao que ocorre na vida real - a classificação textual feita por *Naive Bayes* costuma ser boa, até melhor que alguns métodos mais sofisticados.

2.8 PROCESSO KDD

O processo *Knowledge Discovery in Databases* (KDD) abrange várias etapas descritas em *From Data Mining to Knowledge Discovery in Databases* [11] que estão resumidas a seguir e ilustradas na figura 2.2:

- Primeiro é importante entender o domínio de aplicação (no presente trabalho é o domínio dos filmes) e quais são os objetivos pretendidos na descoberta de conhecimento (concluir se um filme obteve sucesso de público, por exemplo).

- Depois é necessário selecionar os dados que serão utilizados, já que há um grande volume de dados e não necessariamente será utilizado tudo.

- Outra etapa é realizar a limpeza e pré-processamento dos dados, separando ruídos, fazendo processo de tokenização e determinando como lidar com atributos faltantes em partes dos dados. Exemplo: nem todos os *tweets* têm time_zone, o campo é *null*. Contudo, esse atributo nem será relevante para o estudo, então pode ser desconsiderado.

- Extrair características relevantes dos dados, realizando transformações sobre eles, como a obtenção da matriz de características, processo explicado mais à frente.

- Escolher métodos de mineração de dados para extração do conhecimento. Essa escolha deve se preocupar em alcançar os objetivos propostos inicialmente. Foram utilizados algoritmos de classificação para extrair conhecimentos e permitir análise de sentimentos.

- Realizar a classificação e ajustar os parâmetros do classificador de forma a trazer melhores resultados.

- Interpretar e analisar as informações obtidas e projetar formas de visualização do conhecimento extraído.

- Chegar a conclusões, aproveitar esse conhecimento obtido e documentar os resultados ou incorporá-los em outros sistemas, como foi feito com o banco de dados orientado a grafos.

FIGURA 2.2: As etapas do KDD

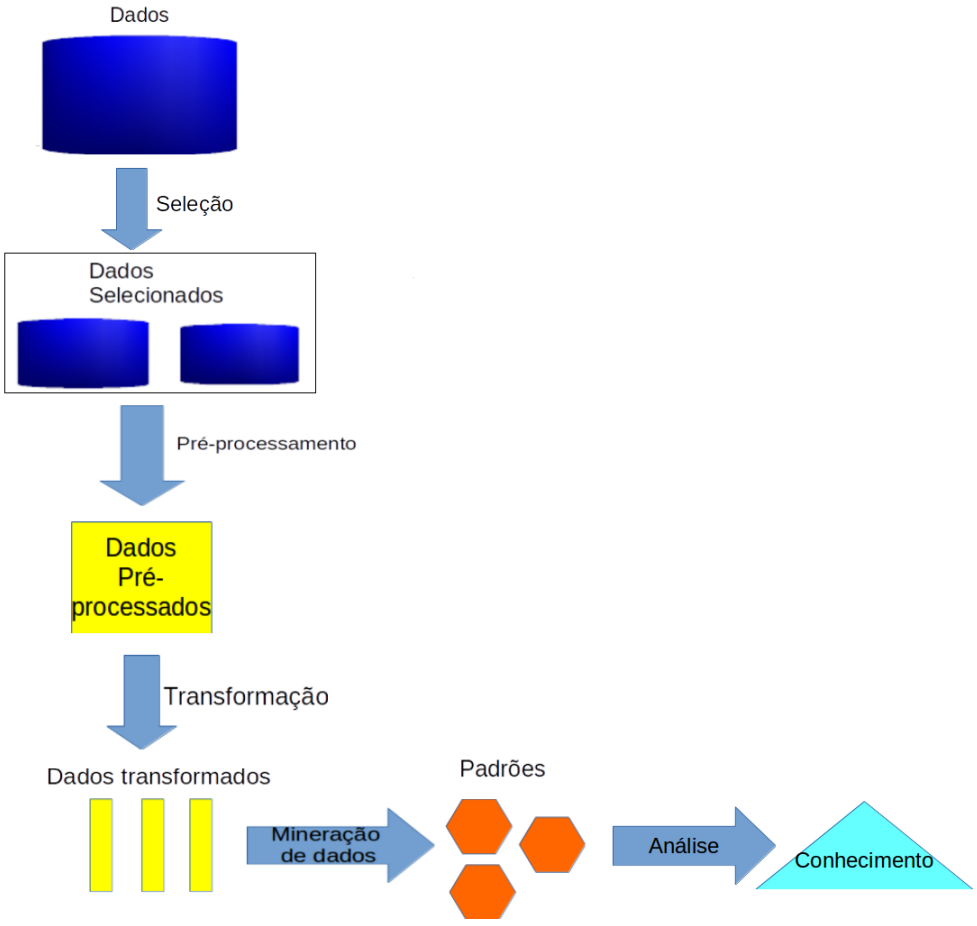


Figura baseada em [11]

3. Coleta, processamento e classificação de dados

3.1 COLETA DE DADOS

Os dados analisados foram coletados utilizando a API do *Twitter* chamada Streaming API [12]. Com ela, os dados são coletados em tempo real, o que é apropriado para conseguir de forma rápida vários resultados sobre algum evento ou assunto que está bastante em voga no momento da pesquisa. Dessa forma, são feitos *requests* com a consulta desejada. Para isso, precisa-se obter uma chave de acesso e um *token* que permitam essa conexão com o fluxo de dados.

Para poder usar o *Streaming API* é criada uma aplicação associada a uma conta do *Twitter*. É permitido que cada conta tenha poucas conexões simultâneas por aplicação, mesmo que sejam utilizados IPs diferentes. Além disso, o número de conexões simultâneas por IP também é limitado, não importando qual aplicação que está fazendo a solicitação.

Existem alguns parâmetros que podem ser especificados para fazer os *requests*. Um deles é o *track* que funciona da seguinte forma: se separarmos uma lista de termos por espaços é similar a fazer um AND lógico, enquanto a separação por vírgulas equivale a um OR. Por exemplo: *[track = café leite]* corresponde a 'café AND leite' e *[track = café, leite]* equivale a 'café OR leite'. Assim, no primeiro caso, ambos os termos devem estar presentes no *tweet* para que ele seja coletado. Já na segunda forma, basta haver apenas um dos termos.

Para serem obtidos "casamentos" (*matches*) entre as palavras listadas em *track* e o *tweet* coletado são levados em consideração não somente o texto do *tweet*, mas também alguns atributos como: *expanded_url*, *display_url* e *screen_name*.

3.1.1 Palavras-chave utilizadas na coleta

Na obtenção do fluxo de *tweets* para cada filme são utilizadas palavras-chave. É interessante notar que por ter sido utilizada a API de *Streaming*, isto é, os dados são coletados em tempo real, os filmes que estão em cartaz são bastante discutidos e comentados no *Twitter*. Assim, para o filme da "Alice Através do Espelho", por exemplo, a busca usando a palavra "Alice" trouxe muitos resultados relacionados ao filme. Bem como *"Deadpool"* também permitiu coletar muitos *tweets* do filme de mesmo nome, sem muitos ruídos associados.

Em contrapartida, outros filmes não trouxeram tantos resultados relevantes ou foram coletados muitos ruídos juntamente com informações relevantes, mesmo utilizando palavras-chave mais complexas. Um exemplo

disso é o filme "Como eu era antes de você" (no original *"Me Before You"*). Como o título traz palavras bastante comuns, tanto o original quanto o traduzido, muitos *tweets* que apresentavam todos os termos do título não eram necessariamente relacionados ao filme em questão.

A reflexão acima ilustra alguns dos fatores que influenciaram na busca: o **momento** em que foi realizada, isto é, se o filme está em cartaz e, portanto, bastante em voga, bem como os termos presentes no **título** (se é composto por *stopwords* ou então por palavras que se diferenciam das demais do idioma).

Outra possibilidade para palavras-chave, além de termos do título, poderia ser o nome dos protagonistas dos respectivos filmes. Contudo, isso aumentaria ainda mais o ruído, já que os *tweets* poderiam se referir a outros trabalhos desses atores ou expressar opiniões sobre o ator em si e não sobre o filme.

Como o intuito era obter *tweets* que expressassem a opinião sobre alguns filmes para, em seguida, fazer análise de sentimentos sobre o conteúdo desses dados, a busca utilizando palavras-chave presentes em partes do título ou em sua totalidade já se mostrou suficiente.

3.2 DADOS COLETADOS DO IMDB

Para coletar os dados do IMDb sobre os filmes buscados no *Twitter* foi utilizada a OMDb API [13]. Trata-se de um serviço web livre, mantido por usuários. Foi necessária a elaboração de *scripts* para coletar e processar os dados obtidos.

As requisições são feitas para *"http://www.omdbapi.com/?"* através de *get* e podem ser usados alguns parâmetros. Os principais utilizados foram: i e s. O s vem de *search* e é utilizado para pesquisar por título de filme. Com isso, consegue-se obter algumas informações, como o ano em que foi produzido e o identificador no IMDb. Com este último dado, pode-se pesquisar com o parâmetro i (imdbID) e obter mais informações: gênero, diretor, alguns atores, país, *imdbRating*, entre outros.

A seguir está um exemplo de como é realizada a coleta e alguns resultados obtidos. Ao fazer um get para *http://www.omdbapi.com/?s=Deadpool*, aparecem alguns resultados mas, dentre eles, o que se deseja coletar é o de id = "tt1431045" e a requisição é feita para

http://www.omdbapi.com/?i=tt1431045. O resultado obtido é:

```
1 {"Title":"Deadpool","Year":"2016","Rated":"R","
   Released":"12 Feb 2016","Runtime":"108 min","Genre"
   :"Action, Adventure, Comedy","Director":"Tim Miller
   ","Writer":"Rhett Reese, Paul Wernick, Fabian
   Nicieza (character), Rob Liefeld (character)","
   Actors":"Ryan Reynolds, Karan Soni, Ed Skrein,
   Michael Benyaer","Plot":"A former Special Forces
   operative turned mercenary is subjected to a rogue
   experiment that leaves him with accelerated healing
    powers, adopting the alter ego Deadpool.","
   Language":"English","Country":"USA","Awards":"2
   wins & 6 nominations.","Poster":"http://ia.media-
   imdb.com/images/M/MV5BMjQyODg5Njc4N15BMl5
   BanBnXkFtZTgwMzExMjE3NzE@._V1_SX300.jpg","Metascore
   ":"65","imdbRating":"8.2","imdbVotes":"433,391","
   imdbID":"tt1431045","Type":"movie","Response":"True
   "}
```


O resultado encontra-se em formato JSON e nem todas as informações dele foram utilizadas. Assim, foi criado um *script* para selecionar apenas o conteúdo relevante que foi, posteriormente, inserido no banco de dados.

3.3 DIFICULDADES NA MINERAÇÃO SOBRE OS DADOS COLETADOS

O uso de *tweets* como fontes de dados para realizar a mineração de opinião traz algumas dificuldades, como restrição do tamanho do texto (limite de 140 caracteres), qualidade do texto, uso de palavras abreviadas ou escritas de maneira incorreta, entre outros problemas que estão mais detalhados a seguir.

A qualidade dos dados é heterogênea, já que existem vários tipos de pessoas e organizações que utilizam mídias sociais e, assim, nem sempre a fonte de informação é confiável, ou seja, nem tudo é verídico.

Sobre isso, há também os *bots*, mensagens automáticas, como propagandas ou outras formas de influenciar a opinião das pessoas. Essas mensagens são geradas por *scripts* e máquinas com o intuito de tendenciar para alguma opinião e deixá-la mais em voga. Como esses *bots* podem ser distribuídos numa rede por diferentes máquinas enviando em tempos randômicos, o *Twitter* não tem controle sobre esse tipo de mensagem para poder fazer o bloqueio. Portanto, embora os dados coletados usando a *streaming* API do *Twitter* sejam provenientes de *tweets* de contas de usuários, é possível que algumas mensagens sejam automáticas e que estejam camufladas, de forma que não consegue-se filtrar previamente para fazer a mineração de opiniões sobre os dados 'limpos'.

Outra dificuldade são *tweets* utilizados para fazer propagandas de algum produto, como exemplo a maquiagem associada ao elenco de um filme X. Essas mensagens não agregam significado para a busca, uma vez que não expressam uma opinião do filme em questão.

Coletar opiniões trata-se de obter dados subjetivos, então depende de uma série de fatores, como o meio em que o indivíduo está inserido, suas

crenças pessoais e experiências de vida que refletem na sua forma de pensar.

Trabalhar com grandes volumes de dados gerados continuamente (*Big Data*) é complexo das perspectivas da coleta, armazenamento e análise. O volume de dados gerado por uma rede social em um único ano atinge facilmente *Terabytes* ou até mesmo *Petabytes*.

Na análise dos dados para a mineração de opiniões, o processamento de linguagem natural é particularmente bem trabalhoso, já que postagens possuem abreviações, gírias, sarcasmo (o que é difícil de ser detectado por uma máquina), *emoticons*, além do conteúdo que pode estar em várias línguas. Desse modo, o tratamento dos dados tem que levar todos esses aspectos em consideração.

3.4 CONSTRUÇÃO DO CORPUS

Com o objetivo de se obter a polaridade das palavras e frases, há algumas possíveis abordagens utilizadas na ciência: baseada em dicionário, aprendizado de máquina, estatísticas e abordagem semântica.

A utilizada no presente trabalho é a de aprendizado de máquina supervisionado com técnicas de classificação de dados que pode ser dividida em duas etapas: aprender o modelo de classificação mediante um *corpus* usado como treinamento com dados previamente rotulados e depois utilizar esse modelo para prever a polaridade sobre novos dados.

Assim, percebe-se que no contexto de análise de sentimentos na abordagem escolhida, a obtenção de uma boa classificação está extremamente relacionada ao *corpus* utilizado como conjunto de treinamento.

O *corpus* é uma coleção textual escolhida de forma a ser representativa dentro do domínio. Isso significa que seu conteúdo pode ser generalizado e representa bem os dados que serão analisados, de forma que se esse *corpus* for utilizado para treinamento, irá gerar um bom classificador para o conjunto de teste.

Percebe-se que uma das limitações da abordagem escolhida para este trabalho é o volume de dados de treinamento e a qualidade dos mesmos interferindo diretamente nos resultados obtidos pelo classificador. Uma forma de contornar esse desafio de rotular manualmente uma grande quantidade de dados seria a geração automática de dados de treinamento através de métodos baseados em regras sobre a estrutura sintática dos dados, mas isso não se ajusta bem para dados de *streaming*, como descrito no artigo de Becker e Tumitan [14]: *"além de grandes volumes de dados que devem ser analisados com baixa latência, existe uma volatilidade enorme nos tópicos e termos utilizados"* nesse contexto de *streaming*.

Então, como contribuição adicional deste trabalho de conclusão de curso, foi contruído um *corpus* com dados coletados do *Twitter* sobre o domínio de filmes lançados no primeiro semestre de 2016.

Os classificadores são bastante sensíveis ao domínio dos dados de treinamento, já que o vocabulário e as frases podem ser específicos de um determinado domínio. Além disso, uma mesma palavra pode ter orientação semântica - polaridade - diferente quando em contextos distintos. Um exemplo dessa situação segue abaixo:

"Sorry but the cast of warcraft is so cute im dying"

Esse é um *tweet* sobre o filme *Warcraft: The Beginning*. Na frase acima o sentido da palavra "morrendo" é hiperbólico e a conotação do *tweet* é positiva. No geral, expressões como "louco para assistir", "morrendo de tanto rir", entre outras, expressam positividade e são comuns no domínio do cinema. Contudo, no domínio médico, por exemplo, "morrendo" seria provavelmente classificado como negativo.

3.5 PROCESSAMENTO E AUTOMATIZAÇÃO

3.5.1 Redução do tamanho e seleção do idioma

Ao coletar dados do *Twitter* no formato JSON, nem todas as informações seriam utilizadas. Além disso, o tamanho do arquivo em formato JSON com todos os atributos é de outra ordem de grandeza - de *gigabytes* cai para *megabytes* - quando comparado com arquivo contendo apenas a parte textual do *tweet*. Um exemplo é o do filme da "Alice através do Espelho", cujo JSON possuía 2,2GB e o conteúdo textual (o *tweet* em si, sem os demais atributos) foi de 68,9 MB, correspondendo a 503.745 *tweets*. Embora o original fosse mantido para obter outras informações relevantes, como a língua e geolocalização do *tweet*, quando disponível, a manipulação dos dados se tornou viável devido a essa redução realizada.

Para que a classificação pudesse alcançar maior qualidade, escolheu-se utilizar um único idioma no lugar de uma mistura deles, uma vez que *stopwords* e palavras em geral são diferentes e o número de palavras usadas como *features* aumentaria muito. O idioma escolhido foi o inglês devido ao maior volume de *tweets* nesse idioma, identificado pelo atributo *lang* presente no JSON do *tweet*.

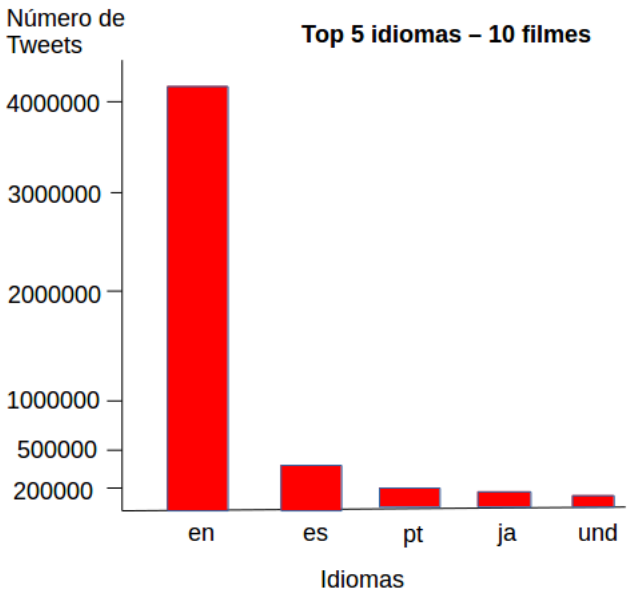
O gráfico 3.1 ilustra a predominância do inglês nos *tweets* de dez filmes, com 4.175.576 *tweets*, mostrando como é significativa essa diferença em relação aos demais idiomas (o segundo lugar, o espanhol, contém 404.946). Assim, a escolha da língua inglesa já é suficiente para a análise de sentimento.

Uma possibilidade para aumentar o número de *tweets* seria utilizar ferramentas de tradução das outras línguas para o inglês. Contudo, não haveria um ganho muito maior em termos de volume de dados e teria que se levar em consideração outros fatores, como a qualidade da tradução. Como a quantidade de dados em inglês já é satisfatória para a tarefa de análise pretendida no presente trabalho, a tradução seria algo adicional e que fugiria do escopo.

3.5.2 Validação cruzada

No processo de aprendizagem supervisionada, parte dos dados rotulados manualmente são utilizados como conjunto de treinamento e outra parte como conjunto de teste. Para evitar que os parâmetros sejam super ajustados ao conjunto de treinamento, causando sobreajuste (*overfitting*), costuma-se ocorrer a divisão em três conjuntos: treinamento, teste e validação. Assim, o classificador é validado usando o conjunto de validação e, somente depois de parecer bem sucedido, é aplicado no conjunto de teste. Contudo, um problema advindo dessa situação é a redução do conjunto de treinamento, já que parte será usada para validar. Dessa forma, os resultados

FIGURA 3.1: Cinco idiomas mais presentes



poderiam depender da partição feita a cada momento, isto é, depender de como foi feita a escolha aleatória dos conjuntos.

Visando a resolver isso é usado o processo de validação cruzada, um procedimento em que o conjunto de treinamento é dividido em k conjuntos menores (por isso a abordagem chama-se *k-fold*). Para cada k, o k-ésimo conjunto é utilizado como teste e o restante dos dados como conjunto de treinamento. O desempenho é medido usando a média desses k processos que, embora possa ser custoso em termos de recursos computacionais, não precisa de muitas amostras dos dados.

A classificação de dados implementada neste trabalho usou a biblioteca de mineração de dados *scikit-learn* [15] do python, a qual apresenta alguns métodos para validação cruzada. Para citar alguns: train_test_split(), *StratifiedKFold()* e *Stratified_Shuffle_Split()*. Foram testados os três para analisar qual traria melhores funcionalidades e resultados.

O train_test_split() faz divisão aleatória do conjunto de treinamento e de teste, permitindo a definição do tamanho desses conjuntos, isto é, se for atribuído o número 0.4 ao parâmetro *test_size* significa que 40% dos dados serão utilizados como teste e o restante como treinamento.

O *StratifiedKFold()* também gera índices que permitem a divisão dos dados em conjunto de teste e de treinamento. Ele utiliza a estratégia de *k-folds* estratificados, isto é, a porcentagem das amostras para cada classe é preservada. Isso é útil quando as classes são desbalanceadas entre si, que é o caso do conjunto de validação usado no presente trabalho.

Por fim, o *Stratified_Shuffle_Split()* permite a divisão dos conjuntos de teste e treinamento na proporção especificada nos parâmetros (foi utilizado 40% dos dados para teste) e retorna *folds* aleatórios e estratificados. A quebra aleatória em *folds* não garante que todos sejam diferentes entre si, mas é bem possível que sejam, visto que são apenas 10 *folds* para centenas de dados utilizados.

Os propósitos do presente trabalho eram usar estratificação por causa do desbalanceamento das classes e escolher a proporção dos dados que seriam usados para treinamento e, assim, o método que mais se adequava era *Stratified_Shuffle_Split()*, sendo o escolhido para fazer a análise da qualidade da classificação.

3.6 WORKFLOW CONSTRUÍDO

Um *workflow* científico define as dependências e relacionamentos das atividades de um processo científico e permite que cientistas de diferentes áreas, seja dentro ou fora da computação, possam se comunicar. Facilita a compreensão e realização dos objetivos a serem alcançados no processo.

O modelo do *workflow* define as atividades e sua ordem de execução. Elas podem ser manuais, automáticas ou semi-automáticas, dependendo do grau de participação humana em cada atividade. Mais detalhes podem ser vistos em [16].

Os dados coletados passaram por uma sequência de ações que resultam em sua classificação ao final do processo. Assim, foi construído um *workflow*, ilustrado na figura 3.2 cujo modelo é detalhado a seguir, explicando-se os passos que foram realizados.

Foram coletados dados brutos em formato JSON com a *Streaming* API, usando palavras-chave de filmes em lançamento no momento da busca. Como esses dados estavam em diversos idiomas, foi implementado um filtro para que fosse possível trabalhar apenas com os que tivessem o atributo *lang* definido como inglês.

Uma amostra desses dados filtrados foi rotulada manualmente, resultando em um *corpus* de 2401 *tweets* (considerados como documentos) rotulados ao todo. Esse processo manual é essencial e está presente em outros trabalhos dessa área como em [17].

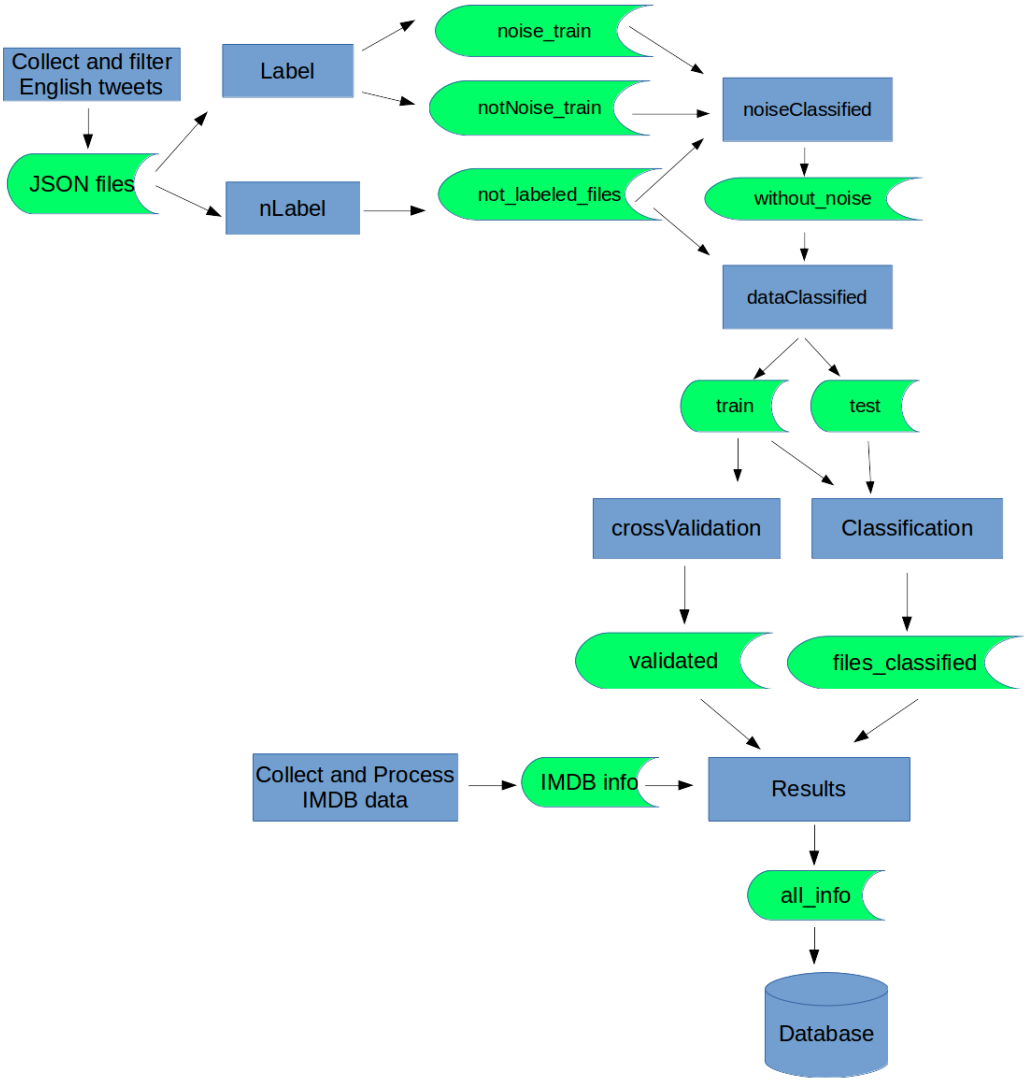
Os arquivos rotulados ficam em *'Label'* – são utilizados como conjunto de treinamento - e os demais ficam em *'nLabel'* – e formam o conjunto de teste. Dentro dos rotulados, um programa divide os documentos de acordo com seus rótulos em duas categorias: *'noise_train'* e *'notNoise_train'* que são ruídos e não ruídos, respectivamente.

Os dados com ruído são aqueles que não estão relacionados aos filmes, podendo ser propagandas ou assuntos completamente diferentes. Por exemplo, no filme do "Mogli: O Menino Lobo", a palavra-chave "Mogli" trouxe como resultado *tweets* que falavam de "Yogli Mogli" (que vende iogurte congelado) e que, portanto, não tinham relação alguma com o filme em questão. Exemplo de *tweet*:

"Need Yogli Mogli rn."

Assim, a eliminação de ruídos se faz crucial para a obtenção de dados de melhor qualidade. Embora as palavras-chave utilizadas pudessem ser mais específicas, isso reduziria o número de *tweets* obtidos e, além disso, foi interessante ter a necessidade de limpar os dados para entender melhor como funciona esse processo e permitir a inclusão do ruído como uma nova classe para o classificador.

FIGURA 3.2: Estrutura do workflow



O resultado da classificação fica armazenado em *"noiseClassified"*, onde é feito também o processo de validação cruzada para analisar a qualidade do classificador na separação dos ruídos.

Em *"dataClassified"* um *script* separa os dados manualmente rotulados em duas classes (positivo ou negativo) e outro *script* separa em cinco (muito negativo, negativo, neutro, positivo e muito positivo). Esses dados são usados como conjunto de treinamento para duas e cinco classes, respectivamente.

Os dados de teste podem ser tanto dados já tratados pelo *"noiseClassified"*, isto é, sem os ruídos, quanto aqueles com ruído se é desejado incluir o ruído como uma classe a mais presente nos conjuntos de treinamento e de teste. Isso permite verificar a diferença de treinar e classificar com duas ou três classes (positivo, negativo e ruído). Por isso que o *"dataClassified"* pode receber tanto dados brutos, quanto aqueles em que a limpeza já foi realizada.

Antes de fazer a classificação dos dados em duas, três e cinco categorias

ocorre, de maneira similar ao que é feito no *noiseClassified*, um processo de validação sobre os dados do conjunto de treinamento para avaliar a qualidade da predição realizada pelo classificador.

A tarefa de classificação em categorias pode ser feita para cada filme para analisar a recepção do mesmo pelo público ou com a junção de todas informações de todos os filmes para proporcionar uma visão geral sobre o sentimento expresso nos *tweets* na totalidade, independentemente do filme buscado.

Os resultados da classificação ficam armazenados em *Results*, juntamente com os dados coletados e selecionados do IMDb, em formato JSON, contendo título, gênero, ano de lançamento, entre outras informações sobre os filmes pesquisados.

O *workflow* utilizado foi o de fluxo de dados, em que dados brutos passam por uma série de atividades, como limpeza, transformação, classificação e análise. A estrutura dessas atividades está intimamente ligada ao tipo de dados utilizado por elas, estabelecendo-se uma relação de dependência. Trata-se de um fluxo, pois dados produzidos por algumas atividades são consumidos por outras que as sucedem.

3.7 REPRESENTAÇÃO DOS DADOS DO WORKFLOW

Os dados do *workflow* - coletados do *Twitter*, incluindo os processados e classificados - foram armazenadas em arquivos. Não foram mantidos em sua totalidade em banco de dados, embora fosse possível, mas há alguns motivos para isso.

Por ser um *workflow* de fluxo de dados em que programas em *python* processavam alguns dados que seriam utilizados por outros programas, nem toda a informação seria, de fato, importante. Por exemplo, cada bloco em JSON não contém apenas o texto do *tweet*, seu conteúdo é mais amplo, possuindo data de criação, idioma, se é *retweet* ou não, cor de fundo do perfil do usuário e alguns outros campos. Haveria *gigabytes* de informações que não seriam fundamentais para o presente trabalho e não seriam necessárias de se manter em um banco. Além disso, como seriam usados todos os textos dos **tweets** do idioma inglês, parte na rotulação, parte no conjunto de teste, se os dados estivessem em um banco de dados precisaria ser feita uma consulta no formato:

```
SELECT *
FROM textTweets
WHERE lang="en"
```

Sendo *"textTweets"* uma possível tabela onde ficariam os *tweets*. Isso significa que a consulta teria que retornar todos os *tweets* para que pudessem ser processados. A maneira mais prática seria manter esses dados em arquivos para leitura e que já estariam na máquina local, evitando fazer requisições a um SGBD a cada novo *script* de processamento, validação e classificação.

Embora nem todos os dados precisem ficar no banco, este é importante para possibilitar não somente o armazenamento de informações, mas também a realização de consultas sofisticadas e o retorno de resultados. Portanto, os resultados considerados relevantes foram disponibilizados no

Neo4J que possibilita fazer consultas, além de consistir em boa forma de visualização.

3.8 CLASSIFICAÇÃO

3.8.1 Classificadores do python

O *python* 3 [18] apresenta algumas bibliotecas que realizam a implementação de classificadores. Merece destaque o *scikit-learn* por ser uma biblioteca de aprendizagem de máquina, bastante completa e com código aberto.

O *scikit-learn* apresenta ferramentas para mineração, análise e validação de dados, cobrindo uma vasta gama de métodos estatísticos e de aprendizado de máquina. Apresenta diversos classificadores, para citar alguns: *Naives Bayes*, k-vizinhos mais próximos, máquinas de vetores de suporte (SVM), árvores de decisão, gradiente descendente estocástico, entre outros.

Para o *Naive Bayes* foi escolhida a variante *Multinomial* para realizar a classificação dos dados de teste. O parâmetro utilizado foi o *alpha* que utiliza um suavizador de Laplace (citado anteriormente na seção de Naive Bayes) para os valores. Se o valor de *alpha* for zero, não há suavização.

Para o SVM podem ser usados o SVC e o LinearSVC do *scikit-learn*, ambos podendo ser aplicados para classificação binária e multiclasses. Associados a esses classificadores há os parâmetros que podem ser ajustados de modo a melhorar a acurácia dos resultados obtidos. Para o SVC os parâmetros chave são: *C*, *kernel*, *degree* e *gamma*. Por sua vez, os do LinearSVC são: *C*, *loss* e *penalty*.

O parâmetro *C* presente em ambos está relacionado ao quanto o algoritmo de classificação deve se ajustar aos dados de treinamento. Valores pequenos de *C* indicam que o SVM tem que se adaptar menos aos pontos, havendo chance de *underfitting*, enquanto que para valores grandes o algoritmo deve se ajustar mais aos pontos do treinamento, aumentando o risco de *overfitting*. Portanto, o ideal é usar um valor intermediário para esse parâmetro.

Depois disso, são necessários ajustes nos demais parâmetros. O *kernel* indica se o modelo do SVM irá usar uma curva ou uma linha no momento de predizer a classe – o *kernel* é linear para o SVCLinear.

Para que o SVM permita a classificação em várias categorias, a sua aplicação não é direta, precisam ser feitas reduções para vários problemas binários. O parâmetro *multi_class* do LinearSVC é, por padrão, *'ovr'* que usa a estratégia um-contra-todos em que os classificadores binários construídos distinguem uma classe em relação às demais. Em contraste, o SVC usa a estratégia "um-contra-um" para múltiplas classes. Assim, se n for o número de classes, são construídos $\frac{n*(n-1)}{2}$ classificadores e cada um deles treina dados de duas classes.

Antes de chamar o classificador é criado um vetorizador que passa posteriormente por um *fit_transform*. O vetorizador usado é *tf-idf* com a finalidade de transformar os textos dos *tweets* em uma matriz de características, levando-se em consideração a frequência dos termos em um documento e a frequência inversa deles nos documentos da coleção. Nesse processo podem ser removidas as *stopwords* e ser estabelecido um limiar para ignorar termos que aparecem em muitos documentos.

3.8.2 Quantidade de dados de treinamento

Com o intuito de mostrar que o volume de dados de treinamento influencia na qualidade do classificador, foram feitos experimentos que ilustram esse fato utilizando-se, para isso, métricas comumente usadas em estudos de mineração e classificação de dados: acurácia, precisão, cobertura e medida f.

Como dados de treinamento para esse experimento foi utilizado o *corpus 'movie_reviews'* do NLTK (Natural Language ToolKit) do *python* que contém resenhas de filmes divididas em duas classes: positivo e negativo. Esse *corpus* possui o total de 2000 documentos.

Nas tabelas são ilustrados os resultados, comparando-se quando foi utilizado $\frac{1}{4}$ dos documentos como conjunto de treinamento em contraste com $\frac{3}{4}$. Conforme aumenta esse conjunto, o classificador "aprende" melhor e apresenta resultados mais satisfatórios. Foi utilizado o mesmo conjunto de teste para ambos os casos, composto por $\frac{1}{4}$ dos documentos, de modo que esse conjunto fosse separado dos dados de treinamento.

É interessante notar que, embora o *Naive Bayes* seja um bom classificador, não é um bom estimador, como descrito em [19] e, assim, não houve uma diferença significativa para o *Naive Bayes* ao aumentar o volume do conjunto de treinamento se comparado ao SVM.

TABELA 3.1: Comparação usando métricas: acurácia e F-measure

	Acurácia		**F-measure**	
Classificadores	**1/4**	**3/4**	**1/4**	**3/4**
Linear SVM penalty 'l2'	0,834	0,874	0,834	0,873
Linear SVM penalty 'l1'	0,778	0,858	0,794	0,864
Linear SVM with L1-based feature selection	0,802	0,846	0,814	0,848
Multinomial Naive Bayes	0,770	0,764	0,763	0,765
Bernoulli Naive Bayes	0,764	0,764	0,760	0,758

TABELA 3.2: Comparação usando métricas: precisão e cobertura

	Precisão		**Recall**	
Classificadores	**1/4**	**3/4**	**1/4**	**3/4**
Linear SVM penalty 'l2'	0,833	0,878	0,836	0,868
Linear SVM penalty 'l1'	0,740	0,830	0,856	0,900
Linear SVM with L1-based feature selection	0,769	0,839	0,864	0,856
Multinomial Naive Bayes	0,787	0,762	0,740	0,768
Bernoulli Naive Bayes	0,773	0,777	0,748	0,740

3.8.3 NLTK como corpus para treinamento

O NLTK apresenta alguns *corpus* que são formados por dados já rotulados em suas respectivas categorias. Dentre os diferentes *corpus* existentes, o que mais se assemelha ao domínio deste trabalho é o *movie_reviews* que possui

dois rótulos, positivo e negativo, para um conjunto de 2000 documentos de resenhas de filmes. Isso totaliza 7.786MB de conjunto de treinamento.

Foram feitos alguns experimentos utilizando-se como conjunto de teste 773 documentos, correspondendo a 0.084MB. Esses dados foram coletados do Twitter sobre cinco filmes e os *tweets* foram rotulados manualmente, nas categorias positivo e negativo.

Os resultados podem ser vistos na tabela abaixo, usando como classificadores o *Naive Bayes* e SVM:

TABELA 3.3: NLTK como treinamento e *tweets* como teste com desbalanceamento de classes

Classificadores	**Acurácia**	**Precisão**	**Cobertura**	**Medida f**
Linear SVM penalty 'l2'	0,533	0,871	0,477	0,616
Linear SVM penalty 'l1'	0,431	0,768	0,396	0,523
Linear SVM with L1-based feature selection	0,444	0,791	0,398	0,53
Multinomial Naive Bayes	0,567	0,821	0,574	0,676

Sabe-se que o *Naive Bayes*, embora seja um bom classificador, não é tão bom estimador. Então, os resultados serão explicados observando-se principalmente as probabilidades do SVM. Como pode ser visto, a acurácia dos classificadores (desconsiderando-se o de Bernoulli) ficou em torno de 50% o que significa que não é melhor do que a aleatoriedade. As medidas de cobertura e medida f também não foram satisfatórias.

Contudo, percebe-se que a precisão foi boa, mais de 80% para alguns classificadores. Para se obter uma precisão boa é necessário haver baixa taxa de falsos positivos, enquanto que para se ter uma boa cobertura, o número de falsos negativos deve ser baixo.

No conjunto de teste havia 608 *tweets* positivos e 165 negativos o que justifica um número bem menor de falsos positivos, já que boa parte do conjunto era de polaridade positiva, em contraste com o número maior de falsos negativos. Isso explica a precisão obtida.

Pode-se notar que esse *corpus* não é adequado com o intuito de treinar os classificadores. Isso pode ser explicado, principalmente, pela estrutura de uma resenha quando comparada com um *tweet*. O primeiro é maior do que o segundo, visto que no geral o *tweet* tem o tamanho de uma linha (não passa de 140 caracteres). Assim, na resenha pode ter um conjunto de sentimentos, mas um ser mais acentuado e, por isso, aquela resenha pertence à categoria positivo ou negativo.

Logo, as características extraídas para formar a matriz de características não é a mais adequada para esse conjunto e, assim, foi decidido rotular manualmente alguns dados dos filmes com o objetivo de melhorar os resultados obtidos pelos classificadores.

3.8.4 Dados manualmente rotulados

Foram coletados *tweets* de vinte e cinco filmes que estavam em cartaz no primeiro semestre de 2016. Contudo, pelo escopo do trabalho, foram rotuladas manualmente amostras de dez filmes (cada uma delas continha, no geral, 200 *tweets*), chegando-se ao total de 2401 para o conjunto de treinamento. Este, por sua vez, encontra-se distribuído nas classes: ruído (909

tweets), muito positivo (291), positivo (590), neutro (341), negativo (220) e muito negativo (50).

Essa distribuição ilustra em si o desbalanceamento inerente aos dados, isto é, não sabia-se de antemão quantos dados da amostra pertenceriam à cada classe e essas diferentes proporções ilustram que houve vários elogios aos filmes coletados, provavelmente porque se tratavam de lançamentos bastante aguardados pelo público.

A tarefa de classificar uma frase como positiva ou muito positiva é bastante subjetiva, visto que os seres humanos apresentam diferentes visões acerca do mundo que os envolve, baseando-se em suas experiências de vida.

Sendo assim, alguns critérios foram levados em consideração durante a atividade de rotulação manual. Entre eles, pode-se citar: a presença de advérbios e outras palavras modificadoras em *tweets* foi utilizada para fazer a distinção entre positivo e muito positivo, bem como negativo e muito negativo; a pontuação também denota a emoção do indivíduo; uso de palavras no superlativo; o tipo de adjetivo usado, por exemplo, o filme ser bom é menos intenso do que o filme ser maravilhoso; entre outros.

Um critério utilizado para distinguir uma afirmação positiva de neutra, por exemplo, foi: se a frase escrita fosse similar a *"Alguém irá assistir ao filme"*, a emoção é neutra, não conota um sentimento positivo ou negativo; por outro lado, estar ansioso para assisti-lo, expressa positividade. Assim, pequenas variações em uma frase, como a presença de um adjetivo, pode mudar a classe a que ela pertence. Isso dificulta a tarefa do classificador, visto que as *features* serão quase as mesmas para duas categorias distintas.

Uma outra classe analisada foi a dos ruídos. Dados ruidosos não trazem informações inerentes ao conteúdo buscado - no presente trabalho o conteúdo seria o sentimento de um telespectador ao assistir um filme em lançamento - e prejudica a qualidade dos dados. Assim, para remover boa parte dos *tweets* com ruídos, foi montado um conjunto de treinamento que apresentasse alguns exemplos representativos de ruídos para que o classificador pudesse aprender a separar as informações ruidosas das relevantes. Desse modo, essa separação pode ser feita de maneira automatizada - exceto pela parte da rotulação, a qual é manual.

Para exemplificar o que foi considerado como ruído, seguem dois deles que foram rotulados dessa maneira por motivos diferentes. O primeiro exemplo, retirado do filme *"The Revenant"*, não tratava do filme buscado, mas sim de um assunto completamente diferente (se referia a um concerto que ocorreu na cidade de *Dallas*). No exemplo 2, embora estivesse relacionado ao filme, não expressava alguma opinião sobre o mesmo, mas sim a respeito de um jogo associado.

Exemplo 1: *"Embark on a Musical Adventure with The Revenant Composer at DSO..."*

Exemplo 2: *"#gameinsight #gaming E3 2016 World of Warcraft EXCLUSIVE Game Card Code With 4 ..."*

Esses dados não acrescentam informações relevantes, já que não são opiniões ou sentimentos despertados ao assistir ao filme.

Abaixo estão mais dois exemplos para ilustrar a rotulação, mas agora diferenciando um *tweet* considerado como positivo de um muito positivo.

Positivo: *"HAVE to see the new Alice this weekend"*

Muito positivo: *"saw the new Alice movie today and it was amazing. Shoutout to Johnny Depp, I love him being a Mad hatter"*

Para ver mais exemplos de dados rotulados manualmente, acesse a página do gitHub: https://github.com/nandacamargo/TCC/ .

3.8.5 Seleção de tweets com localização

Os *tweets* coletados apresentam algumas informações além da parte textual. Dentre elas, pode-se citar: data de criação, se é *retweet* ou não, identificador do usuário, pode conter o seu número de amigos e seguidores, idioma, entre outros.

Quando a geolocalização está habilitada, pode-se obter como informação as coordenadas geográficas que indicam a posição em que a pessoa estava quando escreveu o *tweet*.

Foi utilizado o pacote *geopy* do python 3 para fazer a conversão de coordenadas para endereço, o qual pode ser o Estado, país ou continente associado às coordenadas. O parâmetro *timeout* do *geopy* foi ajustado para permitir tempos maiores de esperas pela resposta do servidor, antes de lançar a exceção: *"geopy.exc.GeocoderTimedOut"*. Esse ajuste foi importante para o processo de automatização para permitir a conversão de milhares de coordenadas para locais.

Assim, além da parte textual do *tweet*, foram incluídas algumas informações, como coordenadas geográficas, endereço (localização) e *id* do usuário. Com isso, foi escolhido um formato que pudesse representar melhor esses dados: o GeoJSON. Trata-se de um formato apropriado para estruturas de dados geográficos. Segue um exemplo para o filme *Warcraft: The Beginning* de um *tweet* já convertido para o formato GeoJSON, contendo apenas as informações mais relevantes para a análise de dados.

```
{
    "type": "Feature",
    "geometry": {
        "type": "Point",
        "coordinates": [
            -43.12408918,
            -22.89734014
        ]
    },
    "properties": {
        "user_location": "Araruama",
        "user_name": "Robson Rodrigues",
        "tweet": "I know nothing about the game. But
           the movie was awesome!\n#warcraftmovie #
           warcraft #cinemark @\u2026 https://t.co/
           oJmyNuFyq4",
        "address": "RJ, Brasil",
```

```
15          "user_id": 69360766
16      }
17  },
```

Dentre os tipos geométricos existentes, *Point* era o mais adequado. Objetos geométricos podem apresentar propriedades, nas quais podem ser colocadas mais informações relacionadas ao ponto. Por apresentar propriedades trata-se de um objeto geométrico do tipo *Feature*.

As propriedades *user_id*, *user_name* e *user_location* foram coletadas junto ao *tweet*, enquanto que o *address* foi obtido através da conversão das coordenadas geográficas.

O *user_location* nem sempre é verídico, pois é indicado pelo usuário e, não necessariamente, indica a posição física do indivíduo. Em um dos *tweets* do filme citado anteriormente continha como *user_location* a frase *"Somewhere Over the Rainbow"* e, assim, depende da criatividade do usuário, não consistindo em uma fonte confiável. Com isso, percebe-se a importância de se incluir o campo *address*.

3.9 VALIDAÇÃO

3.9.1 Validação do classificador de ruídos

A tarefa de eliminação dos ruídos se faz fundamental para a obtenção de melhores resultados, uma vez que dados ruidosos, como descrito previamente, não agregam informação relevante, ou por não serem fontes de dados opinativas, ou por se tratarem de assuntos diferentes do que se busca, ou até mesmo por serem propagandas.

Esse procedimento de limpeza dos dados pode ser aplicado para filmes individuais, como também para o conjunto total de filmes, isto é, há o interesse em apresentar algumas medidas calculadas sobre os dados separados por filme para análises individuais, assim como obter um panorama geral do filmes coletados e rotulados.

A validação cruzada foi aplicada no conjunto de dados rotulados a mão e mostram a qualidade da classificação e, consequentemente, o grau da limpeza dos dados coletados.

Na tabela 3.4 estão a acurácia e o desvio padrão comparados para os classificadores SVM e *Naive Bayes*. O desvio é +-, pois pode apresentar variação para mais ou para menos.

Como é possível verificar no gráfico 3.3, a acurácia e desvio padrão para os filmes mostrou-se similar para ambos os classificadores, isto é, não houve diferença significativa para a classificação nas classes ruído e não ruído, embora o classificador SVM tenha apresentado resultados melhores. Desse modo, para o conjunto de dez filmes manualmente rotulados serão feitas análises baseadas no classificador SVM.

Na tabela 3.5 e na matriz 3.4 estão as medidas calculadas para duas classes, contando com conjunto de treinamento composto por 1680 *tweets*, sendo 636 rotulados como ruído e 1044 como não sendo ruído. O restante dos dados, isto é, 721 *tweets* para completar o total de 2401 de *tweets* rotulados, foram utilizados como conjunto de teste.

Para os dez filmes foi realizado também o experimento de colocar o ruído como sendo uma das três classes, isto é, as categorias poderiam ser:

TABELA 3.4: Acurácia e desvio padrão para as classes ruído e não ruído, comparando SVM e NB

	SVM		NB	
Filmes	**Acurácia**	**Desvio(+-)**	**Acurácia**	**Desvio(+-)**
Alice Through the Looking Glass	0,91	0,11	0,85	0,12
Captain America: Civil War	0,88	0,14	0,56	0,13
Deadpool	0,87	0,13	0,83	0,15
Finding Dory	0,84	0,11	0,70	0,19
Ratchet & Clank	0,88	0,13	0,74	0,20
The Jungle Book	0,83	0,23	0,81	0,20
The Huntsman: Winter's War	0,82	0,14	0,79	0,17
The Revenant	0,88	0,12	0,87	0,09
Warcraft: The Beginning	0,78	0,19	0,74	0,21
Zootopia	0,84	0,10	0,77	0,26
Média	0,853	0,14	0,766	0,172

FIGURA 3.3: Acurácia e desvio padrão por filme para as classes ruído e não ruído - SVM e NB

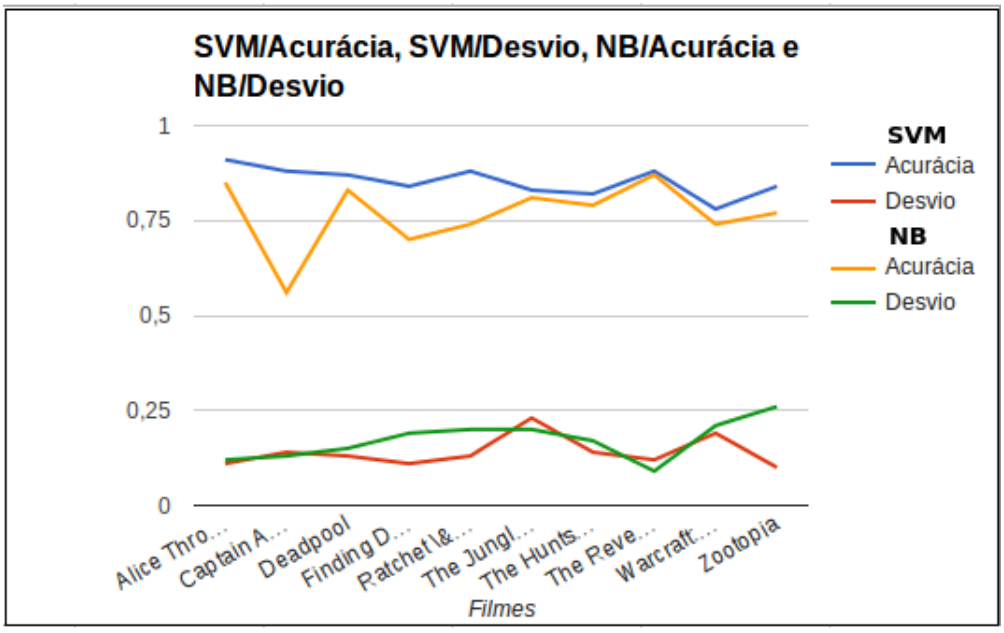


TABELA 3.5: Medidas para duas classes: não ruído e ruído

CLASSES	precisão	cobertura	medida f	suporte
não ruído	0,84	0,96	0,90	448
ruído	0,92	0,71	0,80	273
média / total	0,87	0,87	0,86	721

positivo, negativo ou ruído. Os dados das classes positivo e muito positivo

FIGURA 3.4: Matriz de contingência para duas classes com SVM

$$\begin{array}{c} \\ notNoise \\ noise \end{array} \begin{array}{c} \begin{array}{cc} notNoise & noise \end{array} \\ \begin{pmatrix} 430 & 18 \\ 79 & 194 \end{pmatrix} \end{array}$$

foram utilizados como treinamento do conjunto positivo para o atual experimento - e o mesmo foi feito para as categorias negativo e muito negativo - com o intuito de aumentar o volume de treinamento.

Então, de um total de 909 *tweets* de ruído, 881 positivos e 270 negativos, uma parte foi utilizada para treinamento (70% deles) e o restante para teste (30%). O conjunto de treinamento possui tamanho 1442, nas proporções 617 positivo, 189 negativo e 636 ruído. A validação foi realizada sobre um conjunto de teste com 618 *tweets*. Os *tweets* da categoria neutro (correspondendo a 341) não foram considerados para o atual experimento. Os resultados estão ilustrados na tabela 3.6 e na matriz de contingência 3.5.

TABELA 3.6: Medidas de três classes com classificador SVM

CLASSES	precisão	cobertura	medida f	suporte
neg	0,72	0,42	0,53	81
pos	0,75	0,88	0,81	264
ruído	0,87	0,83	0,85	273
média/total	0,80	0,80	0,79	618

FIGURA 3.5: Matriz de contingência para 3 classes com SVM

$$\begin{array}{c} \\ neg \\ pos \\ noise \end{array} \begin{array}{c} \begin{array}{ccc} neg & pos & noise \end{array} \\ \begin{pmatrix} 34 & 38 & 9 \\ 7 & 232 & 25 \\ 6 & 40 & 227 \end{pmatrix} \end{array}$$

3.9.2 Validação do classificador de filmes

Como mencionado em algoritmos de classificação, foram escolhidos os classificadores SVM e *Naive Bayes* para categorizar os dados por apresentarem bom desempenho para o contexto de análise de sentimento.

Seguem alguns resultados da validação cruzada realizando-se estratificação, isto é, considerando-se o desbalanceamento do tamanho das classes. Foram usadas as métricas: precisão, cobertura, medida f e acurácia. A matriz de contingência foi utilizada por ilustrar visualmente a relação entre os rótulos reais e os previstos pelos classificadores.

Para duas classes, positivo e negativo, utilizando-se 60% dos dados rotulados como conjunto de treinamento e o classificador SVM com *kernel* linear foi obtida a tabela 3.7 .

TABELA 3.7: Positivo e negativo com 60% de treinamento - SVM

CLASSES	precisão	cobertura	medida f	suporte
neg	0,84	0,41	0,55	88
pos	0,81	0,97	0,89	236
média/total	0,82	0,82	0,79	324

Como o volume de treinamento para a classe positiva era bem maior que a da classe negativa, mais que o dobro, é razoável que o classificador tenha aprendido a classificar melhor *tweets* positivos. Assim, consegue-se uma precisão boa para a classe negativa devido à baixa taxa de cobertura, isto é, possui uma taxa grande de falsos negativos. Prefere-se rotular algo como positivo - evento de maior probabilidade - mesmo quando for negativo para, assim, ter uma precisão maior. Se houvesse um volume maior de *tweets* negativos no treinamento, os resultados poderiam ser melhores.

A matriz de confusão 3.6 mostra justamente como a maioria dos *tweets* positivos foram classificados corretamente.

FIGURA 3.6: Matriz de confusão (pos e neg) com 60% de treinamento

$$\begin{array}{c} \\ neg \\ pos \end{array} \begin{array}{c} \begin{array}{cc} neg & pos \end{array} \\ \begin{pmatrix} 36 & 52 \\ 7 & 229 \end{pmatrix} \end{array}$$

A acurácia foi de 0,80, isto é, 80% com desvio padrão de +/- 0,07.

Na tabela 3.8 estão as medidas para duas classes, positivo e negativo, utilizando-se 70% dos dados rotulados como conjunto de treinamento para o SVM linear.

TABELA 3.8: Positivo e negativo com 70% de treinamento - SVM

CLASSES	precisão	cobertura	medida f	suporte
neg	0,86	0,47	0,61	66
pos	0,83	0,97	0,90	177
média/total	0,84	0,84	0,82	243

É notável como todas as medidas melhoraram utilizando-se um conjunto um pouco maior de treinamento (de 486 para 567), mostrando a sensibilidade dos resultados do classificador ao número de exemplos dados para treinar. Contudo, precisa-se mencionar que pelo fato do conjunto de teste ter aumentado (recebeu 10% dos dados que eram de treinamento), o conjunto de teste é um pouco diferente nos dois experimentos, o que poderia ser um motivo adicional para a alteração nos resultados analisados.

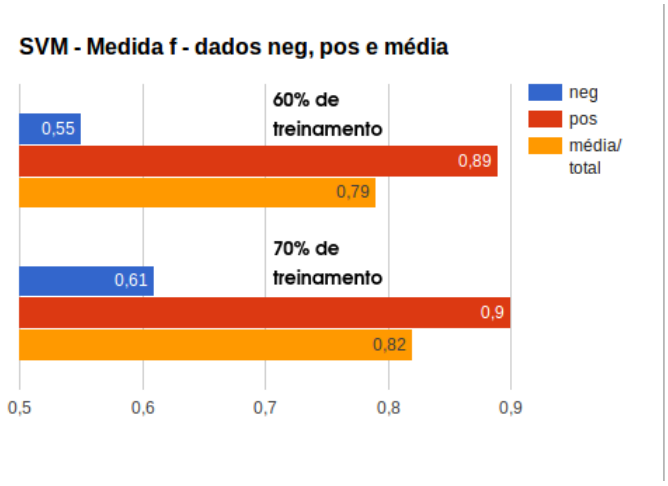
A matriz 3.7 retrata o experimento realizado com 70% de dados de treinamento e, consequentemente, menor volume de teste em relação ao experimento que utiliza 60%, já que os dados de teste são aqueles não utilizados no treinamento.

FIGURA 3.7: Matriz de confusão (pos e neg) com 70% de treinamento

$$\begin{array}{c} \\ neg \\ pos \end{array} \begin{array}{c} \begin{array}{cc} neg & pos \end{array} \\ \begin{pmatrix} 31 & 35 \\ 5 & 172 \end{pmatrix} \end{array}$$

O gráfico 3.8 ilustra uma comparação da medida f calculada para o classificador SVM com 60% e 70% dos dados rotulados sendo usados para treinamento, de modo a tornar mais visível a diferença nos resultados.

FIGURA 3.8: Medida f para o classificador SVM com volumes diferentes de treinamento



Para fins de comparação, o mesmo experimento feito para duas classes e 70% do conjunto de treinamento foi realizado com o classificador *Naive Bayes* e os resultados estão na tabela 3.9 e na matriz 3.9.

Como os resultados para duas classes foi bastante similar para os dois classificadores, foi escolhido o SVM (conhecido por estimar melhor que o *Naive Bayes* - veja seção de classificação) para a análise dos resultados por filme. Embora tenham sido rotulados dez filmes, a análise individual foi feita para sete, pois os filmes: *"Captain America: Civil War"*, *"Deadpool"*, *"The Huntsman: Winter's War"* e *"Ratchet & Clank"* apresentavam poucos *tweets* negativos, o que levava à baixa precisão das medidas para se fazer

TABELA 3.9: Métricas da classificação com duas classes - Naive Bayes

CLASSES	precisão	cobertura	medida f	suporte
negativo	0,89	0,50	0,64	66
positivo	0,84	0,98	0,90	177
média/total	0,85	0,85	0,83	243

FIGURA 3.9: Matriz de contingência para duas classes (positivo e negativo) com *Naive Bayes*

$$\begin{array}{c} \\ neg \\ pos \end{array} \begin{array}{c} \begin{array}{cc} neg & pos \end{array} \\ \begin{pmatrix} 33 & 33 \\ 4 & 173 \end{pmatrix} \end{array}$$

com k=10 *folds* na validação cruzada. Contudo, o resultado geral dado anteriormente foi calculado sobre os dez filmes.

TABELA 3.10: Tabela comparativa de duas classes (positivo e negativo) por filme

Filmes	Classe	Precisão	Cobertura	Medida f	Suporte
Alice Through the Looking Glass	neg	1,00	0,54	0,70	24
	pos	0,86	1,00	0,93	70
Finding Dory	neg	0,75	0,27	0,40	11
	pos	0,68	0,94	0,79	18
The Jungle Book	neg	1,00	0,20	0,33	5
	pos	0,81	1,00	0,89	17
The Revenant	neg	1,00	0,18	0,31	11
	pos	0,79	1,00	0,88	33
Warcraft: The Beginning	neg	0,67	0,29	0,40	7
	pos	0,81	0,96	0,88	23
Zootopia	neg	1,00	0,14	0,25	7
	pos	0,86	1,00	0,92	37

Com o intuito de melhorar a visualização da precisão e da cobertura por filme, foram construídos os gráficos que ilustram esses resultados para os seis filmes acima. A precisão e a cobertura utilizadas se referem à média das medidas obtidas para duas classes (positivo e negativo).

O filme com melhor precisão e cobertura foi o *"Alice Through the Looking Glass"* que apresenta maior quantidade de dados rotulados em relação aos demais e, consequentemente, maior número de retas suporte. O *"Finding Dory"* foi o que apresentou menor precisão e cobertura. A grande diferença desse filme em relação aos demais foi que a proporção de retas suporte entre as classes positivo e negativo era mais equilibrada (11 e 18 retas, respectivamente), enquanto que nos outros filmes analisados a quantidade para a classe positiva foi quase o triplo da negativa ou maior que isso.

FIGURA 3.10: Precisão obtida na validação dos seis filmes

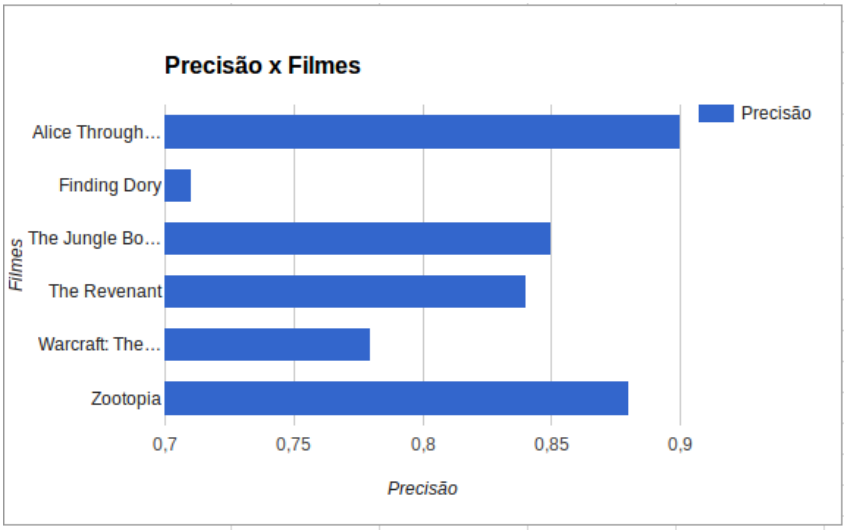
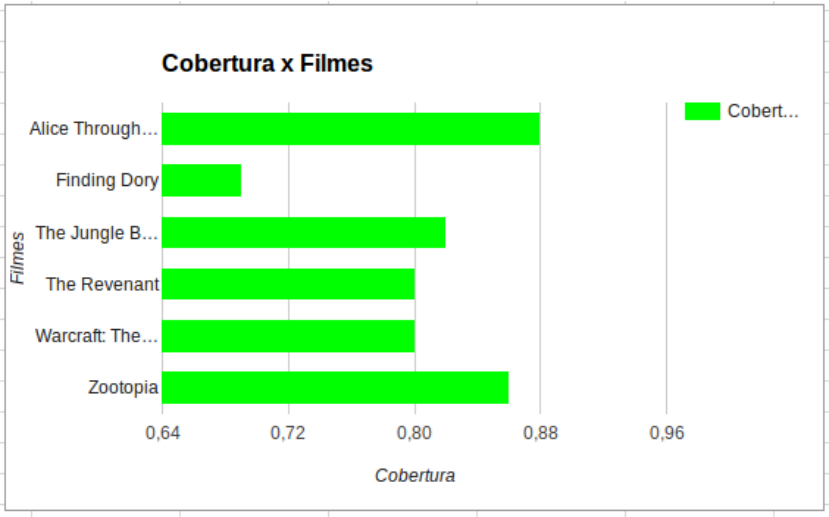


FIGURA 3.11: Cobertura obtida na validação dos seis filmes



Desta forma, o classificador apresentou maiores dificuldades na classificação, pois não houve grandes contrastes entre as duas classes observadas, como ocorreu nos demais exemplos. Isso é mais perceptível principalmente pelo fato do volume dos dados utilizado na validação não ser grande.

Para cinco classes (muito positivo, positivo, neutro, negativo e muito negativo), os resultados seguem na tabela 3.11 e em 3.12.

Na tabela 3.12 e na matriz 3.13 está o mesmo experimento aplicado ao classificador *Naive Bayes* Multinomial.

É possível constatar que os melhores resultados de medida *f* foi para as classes positivo, muito positivo e neutro, as quais são justamente aquelas que possuem maior número de exemplares no conjunto de treinamento.

Como as classes positivo e muito positivo apresentam *features* em comum - o que varia é a intensidade de um elogio, por exemplo - é normal que o classificador apresente um pouco de confusão na divisão nessas duas classes, como pôde ser verificado na matriz de contingência.

O mesmo se aplica às categorias positivo e neutro para o conjunto de

TABELA 3.11: Métricas da classificação com cinco classes - SVM

CLASSES	precisão	cobertura	medida f	suporte
muito neg	0,59	0,31	0,40	88
negativo	0,80	0,20	0,32	20
neutro	0,56	0,67	0,61	136
positivo	0,63	0,74	0,68	236
muito pos	0,70	0,63	0,66	116
média/total	0,63	0,62	0,61	596

FIGURA 3.12: Matriz de confusão para cinco classes com SVM

$$\begin{array}{c} \\ muitoneg \\ neg \\ neutro \\ pos \\ muitopos \end{array} \begin{array}{c} \begin{array}{ccccc} muitoneg & neg & neutro & pos & muitopos \end{array} \\ \begin{pmatrix} 27 & 0 & 23 & 36 & 2 \\ 5 & 4 & 3 & 7 & 1 \\ 7 & 0 & 91 & 33 & 5 \\ 5 & 0 & 32 & 175 & 24 \\ 2 & 1 & 13 & 27 & 73 \end{pmatrix} \end{array}$$

TABELA 3.12: Métricas da classificação com cinco classes - Naive Bayes

CLASSES	precisão	cobertura	medida f	suporte
muito neg	0,53	0,34	0,41	88
negativo	1,00	0,10	0,18	20
neutro	0,59	0,50	0,54	136
positivo	0,59	0,76	0,66	236
muito pos	0,60	0,61	0,61	116
média/total	0,60	0,59	0,57	596

FIGURA 3.13: Matriz de confusão para cinco classes com NB

$$\begin{array}{c} \\ muitoneg \\ neg \\ neutro \\ pos \\ muitopos \end{array} \begin{array}{c} \begin{array}{ccccc} muitoneg & neg & neutro & pos & muitopos \end{array} \\ \begin{pmatrix} 30 & 0 & 16 & 33 & 9 \\ 6 & 2 & 0 & 8 & 4 \\ 11 & 0 & 68 & 47 & 10 \\ 10 & 0 & 23 & 179 & 24 \\ 0 & 0 & 9 & 36 & 71 \end{pmatrix} \end{array}$$

dados rotulados no domínio de filmes. Isso ocorreu porque frases como: *"Watching movie X"* e *"Finally watching movie X"* são bem parecidas, mas

a primeira é uma afirmação, não demonstra polaridade no sentimento, enquanto que a segunda expressa ansiedade, porque finalmente chegou o momento esperado de assistir ao filme. Assim, embora as frases sejam bem semelhantes - o que muda é a presença da *feature "Finally"* na segunda - elas receberam rótulos diferentes, se encaixando em categorias distintas. Isso ilustra coerência do classificador.

4. Banco de dados orientado a grafos

4.1 SGBD

Para a organização dos dados foi necessário um Sistema Gerenciador de Banco de Dados orientado a grafos, de modo a facilitar o entendimento e permitir a visualização das informações obtidas. Dentre os vários SGDBs existentes com esse intuito, foi escolhido o Neo4j [20] que é altamente escalável, suportando centenas de milhares de transações ACID por segundo, além de possuir armazenamento e processamento nativo para grafos.

4.2 NEO4J - ESTRUTURA BÁSICA

O Neo4J é um banco de dados orientado a grafos. A estrutura básica de um grafo é composta por nós e suas ligações. Os nós são entidades, podem ser pessoas, objetos ou lugares e se relacionam por meio de arestas que consistem nos relacionamentos.

Os nós podem ser associados a rótulos que os agrupam. Por exemplo, pode-se criar um rótulo chamado filme. Um nó que tenha título de um filme, ano de lançamento e diretor pode possuir esse rótulo. É importante salientar que um nó pode ter zero ou mais rótulos e nós semelhantes podem apresentar propriedades diferentes.

Relacionamentos conectam nós e sempre apresentam uma direção, isto é, são arestas dirigidas de um grafo. Além disso, relacionamentos também podem possuir propriedades.

4.3 VANTAGENS

O modelo de banco de dados orientado a grafos é bastante expressivo se comparado com bancos relacionais e outros BDs NoSQL. Os relacionamentos são tomados como prioridade, são explícitos, diferente dos relacionais em que são utilizadas chaves estrangeiras para relacionar uma tabela com outra.

Assim, foi escolhido esse tipo de banco com o objetivo de se obter uma representação mais fiel possível do domínio a ser representado (domínio do cinema) e melhorar a visibilidade dos resultados no lugar de usar tabelas, como nos bancos de dados relacionais.

Outras vantagens são performance e flexibilidade. Em bancos de dados tradicionais o desempenho vai diminuindo conforme o número e a profundidade dos relacionamentos aumentam. No quesito de flexibilidade é possível adaptar o grafo do banco atual a novas mudanças no cenário em que está sendo implementado. Não é necessário prever essas modificações de antemão e pensar em detalhes do domínio de aplicação.

4.4 CYPHER

A linguagem de consulta usada no Neo4j é chamada *Cypher*. Ela é uma linguagem declarativa, isto é, indica o que fazer e não como, diferentemente de linguagens imperativas como C. Permite fazer tarefas como seleção, inserção, atualização e deleção dos dados. O *Cypher* descreve padrões em grafos, fazendo uso de ASCII-Art para visualização do dados. Os nós são representados entre parênteses e os relacionamentos são indicados por setas entre dois nós. A partir de nós e relacionamentos, padrões mais complexos podem ser construídos. Algumas das palavras-chave utilizadas no *Cypher* tiveram por base a SQL (*Structured Query Language*), como exemplo da cláusula `WHERE` usada para especificar um filtro em uma consulta.

4.5 APLICAÇÃO

As informações mais relevantes obtidas com a mineração de dados e opiniões foram organizadas em um banco de dados orientado a grafos. Assim, dados do IMDb, o conjunto de treinamento (composto por dados manualmente rotulados) e *tweets* classificados em positivo ou negativo pelo *Naive Bayes* e SVM que estavam no idioma inglês e possuíam geolocalização ativa, foram escolhidas como sendo relevantes para serem armazenadas no banco.

Dentre as possíveis informações que podem ser extraídas, estão: mostrar os relacionamentos entre as pessoas e suas publicações, se a opinião foi positiva ou negativa, qual o gênero do filme mais comentado, se já recebeu premiações de acordo com o IMDb e a procedência dos *tweets* (se coincidem ou não com o local de produção do filme).

4.6 INSERÇÃO DE DADOS

Os dados coletados do IMDb e do Twitter estão em formato JSON. É importante salientar que não existe uma forma direta de inserir os dados em formato JSON no banco de dados.

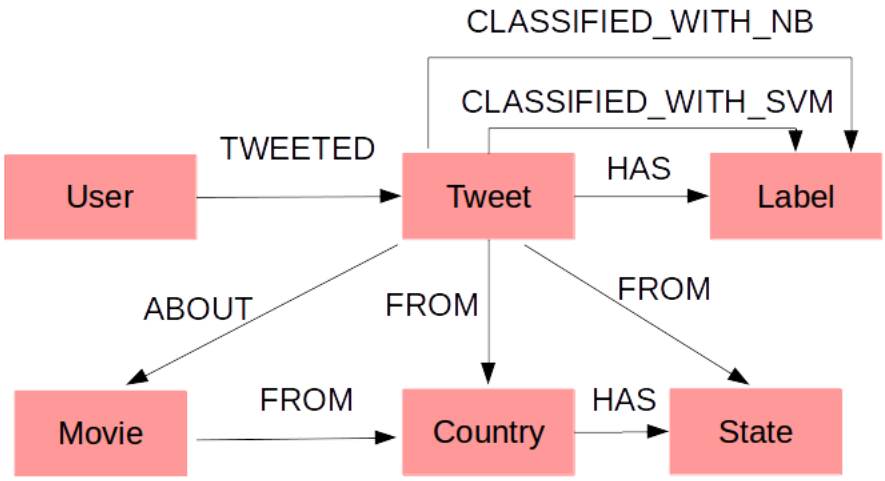
Para isso, como a linguagem utilizada no presente trabalho é python3, há uma forma de inserir esses dados com o encapsulamento dessa linguagem. Existe a biblioteca *py2neo* que permite trabalhar com o Neo4J de dentro de aplicações feitas em python. Possui métodos para autenticação de um usuário no banco de dados, além de possibilitar fazer consultas e inserções.

Merecem destaque as cláusulas `UNWIND e MERGE` que foram utilizadas no processo de inserção e modificação de dados no banco. A primeira faz a expansão de uma lista em uma sequência de linhas que podem ser utilizadas como dados a serem inseridos. A segunda cláusula, `MERGE`, 'casa' (*match*) com nós existentes e faz associação entre eles. Caso não existam, esses nós são criados por essa cláusula. O `MERGE` permite modificar propriedades de nós ou relacionamentos já existentes no banco, ao ser combinado com `ON MATCH`.

4.7 ESTRUTURA DO BANCO DE DADOS

A figura 4.1 ilustra a organização do banco.

FIGURA 4.1: Estrutura do banco de dados



É importante salientar que nem todos os relacionamentos existem para todos os nós, isto é, pode haver relacionamentos que se aplicam apenas para alguns deles. Um exemplo que esclarece esse contexto é o caso dos *tweets*, que podem apresentar geolocalização, caso em que os relacionamentos

```
Tweet FROM Country
```

e

```
Tweet FROM State
```

se aplicam, mas não existem quando considera-se os *tweets* sem geolocalização habilitada. Assim como o relacionamento pode não existir em determinados contextos, em outros podem existir mais de um relacionamento entre dois nós, como ocorre com *tweets* classificados com os algoritmos SVM e *Naive Bayes* se o rótulo dado por ambos for igual, por exemplo uma classificação positiva. Temos, assim, duas arestas ligando os mesmos dois nós (filme ao respectivo rótulo).

4.8 PADRÕES OBSERVADOS NO BANCO DE DADOS

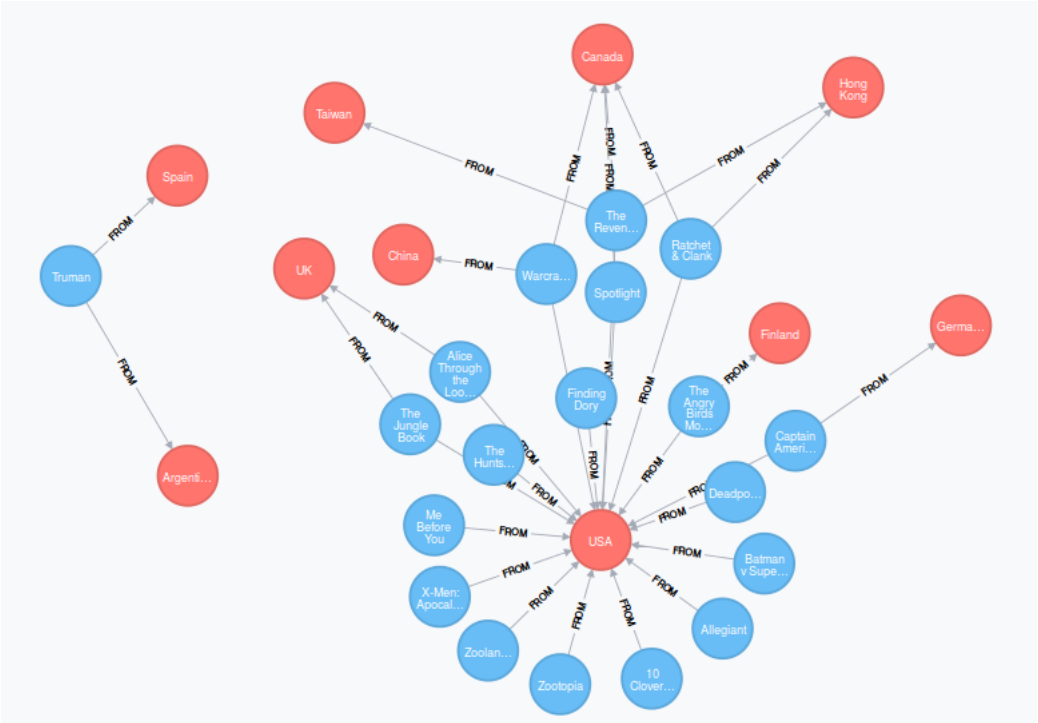
Com as informações armazenadas e organizadas no banco, foi possível identificar alguns padrões que resultam na geração de conhecimento.

Como exemplo, com o relacionamento dos filmes com os países em que foram produzidos, pode-se perceber que dos filmes buscados, quase todos foram produzidos nos Estados Unidos, como mostra a figura 4.2.

Na figura estão ilustrados 19 países, embora a rotulação tenha sido realizada sobre 10 deles. Os outros 9 são filmes que foram coletados com a *Streaming API*, mas que não foram rotulados nem utilizados no processo de classificação. Foram armazenados no banco para mostrar que é possível colocar informações sobre mais filmes.

Os países envolvidos na produção de mais de um filme, além dos Estados Unidos, foram: Canadá e Hong Kong, além de países do Reino Unido (produção dos filmes da Alice e Mogli).

FIGURA 4.2: Filmes e os países em que foram produzidos



Utilizando informações de *tweets* com geolocalização habilitada foi possível extrair as coordenadas geográficas e convertê-las para países e estados. Esses dados foram mantidos pelo relacionamento `[:HAS]` e o panorama geral de estados e países que produziram *tweets* poderia ser obtido pela consulta 4.3.

FIGURA 4.3: Consulta dos estados de todos os países

```
MATCH (c:Country)-[rel:HAS]->(s:State)
RETURN rel
```

O atributo *lang* dos *tweets* coletados pela *Streaming API* é obtido através de algoritmos do Twitter para detecção de linguagem. É importante notar que embora tenham sido escolhidos *tweets* com o atributo *lang* = *'en'*, alguns deles são de estados brasileiros e, de fato, foram publicados com o idioma inglês.

A consulta 4.4 devolve como resultado os estados brasileiros que tiveram *tweets* publicados e com geolocalização habilitada, como pode ser visto em 4.5.

Como os *tweets* foram coletados em tempo real sobre determinados filmes, sem considerar qual usuário que está publicando, não é tão comum a obtenção de mais de um *tweet* do mesmo usuário, embora isso possa acontecer. Então, a consulta 4.6 permite buscar por esse tipo de evento.

Ao fazer essa consulta, o Neo4J devolve os nós que representam os usuários que possuem mais de um *tweet* publicado e armazenado no banco de dados. Interagindo com o nó do grafo que corresponde ao usuário com

FIGURA 4.4: Consulta dos estados brasileiros

```
MATCH (c:Country)-[rel:HAS]->(s:State)
WHERE c.name ='Brasil'
RETURN rel
```

FIGURA 4.5: Estados de origem dos tweets no Brasil

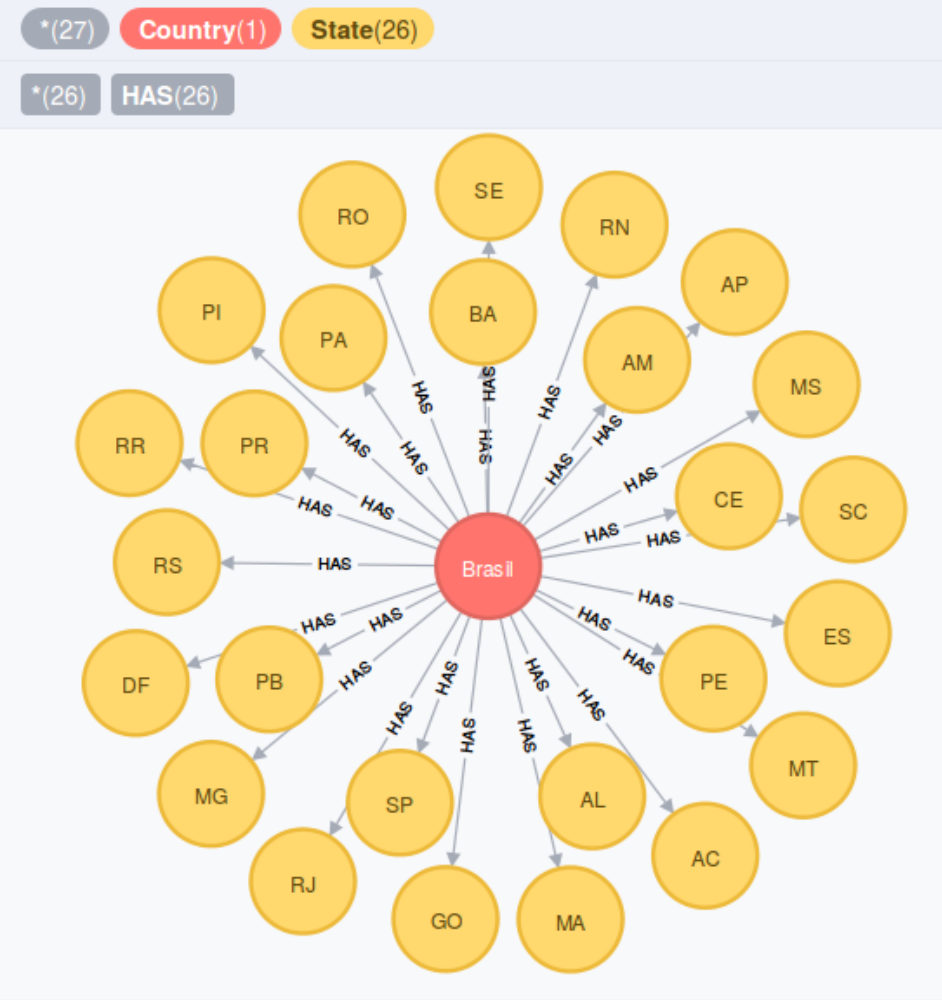
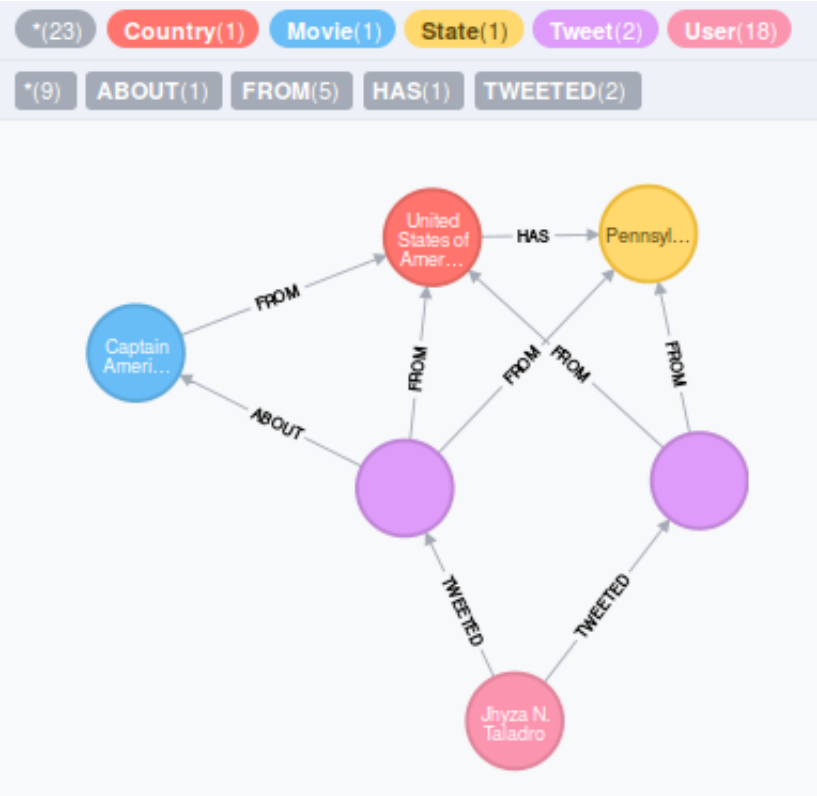


FIGURA 4.6: Consulta usuários com mais de um tweet

```
MATCH (u:User)-[rel:TWEETED]->(t:Tweet)
WITH u, count(rel) AS count
WHERE count > 1
RETURN u, count
```

user_name chamado *Jhyza N. Taladro*, por exemplo, é possível visualizar os relacionamentos com outros nós, como *tweet*, estado e país e o resultado pode ser visto na figura 4.7.

FIGURA 4.7: Usuário que publicou mais de um tweet na base de dados



Um padrão interessante que pode ser visto na figura 4.8 é que nem sempre o rótulo dado pelo *Naive Bayes* é o mesmo que o do SVM, já que são classificadores que empregam técnicas diferentes, sendo o primeiro um classificador probabilístico e o segundo utiliza vetores de suporte e técnicas de otimização para tentar maximizar a margem. A quantidade de resultados que satisfazem esse caso particular de dois rótulos diferentes para o mesmo *tweet* totaliza 24 entre um total de 9924 *tweets* classificados e adicionados no banco, ou seja, eles convergem significativamente na classificação. A consulta que permite obter a informação de divergência na rotulação é 4.9.

Outras informações que podem ser obtidas se referem aos gêneros de filmes mais comentados no *Twitter*. Considerando-se *tweets* em inglês com geolocalização habilitada, pode-se obter a tabela 4.10 a partir da consulta 4.11.

Pode-se perceber que os dois filmes mais comentados dentre os *tweets* analisados foram *"Finding Dory"* e *"Captain America: Civil War"* que possuem três gêneros cada, mas o que é comum a eles é aventura. Além disso, o terceiro filme mais comentado, *"Deadpool"*, também contém aventura, bem como comédia (gênero em comum com *"Finding Dory"*) e ação (em comum com *"Captain America"*), mostrando que filmes de aventura, comédia e ação são comentados em redes sociais.

4.9 LOCALIZAÇÃO DOS TWEETS

Com os dados de localização foi possível produzir um mapa do mundo que indica por pontos pretos os locais dos *tweets*, permitindo a obtenção de

FIGURA 4.8: Classificação diferente para o mesmo *tweet*

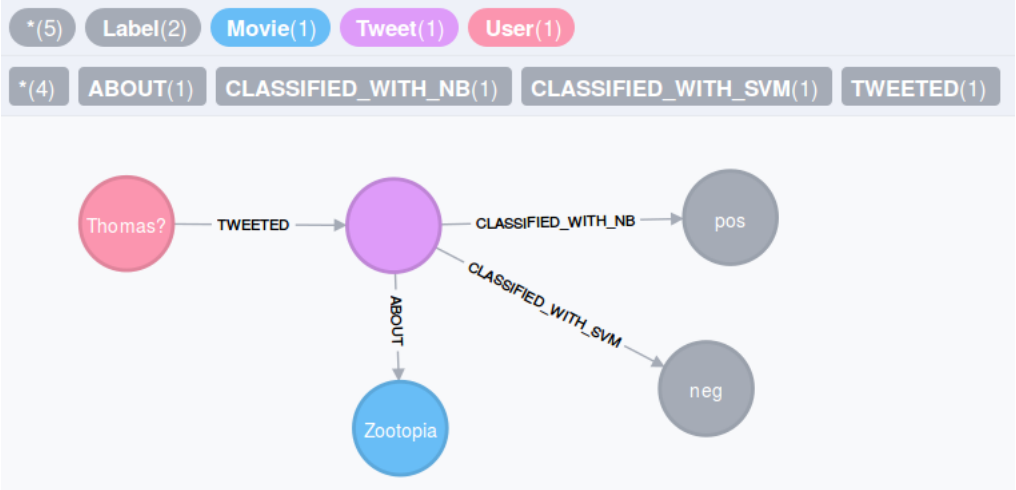


FIGURA 4.9: Consulta que retorna *tweets* classificados com diferentes rótulos

```
MATCH (t:Tweet)-[r1:CLASSIFIED_WITH_SVM]->(l1:Label),
      (t:Tweet)-[:CLASSIFIED_WITH_NB]->(l2:Label)
WHERE l1.name <> l2.name
RETURN count(t) as number_tweets
```

FIGURA 4.10: Tabela com os filmes, gêneros e contagem de *tweets*

m.title	m.genre	num_tweets
The Jungle Book	Adventure, Drama, Family	98
The Huntsman: Winter's War	Action, Adventure, Drama	142
Ratchet & Clank	Animation, Action, Adventure	151
The Revenant	Adventure, Drama, Thriller	242
Alice Through the Looking Glass	Adventure, Family, Fantasy	372
Warcraft: The Beginning	Action, Adventure, Fantasy	729
Zootopia	Animation, Action, Adventure	928
Deadpool	Action, Adventure, Comedy	1047
Finding Dory	Animation, Adventure, Comedy	4091
Captain America: Civil War	Action, Adventure, Sci-Fi	4746

FIGURA 4.11: Retorna os filmes, respectivos gêneros e quantidade de *tweets* por filme

```
MATCH (t:Tweet)-[r:ABOUT]->(m:Movie)
RETURN m.title, m.genre, COUNT(t) AS num_tweets
ORDER BY num_tweets
```

um panorama geral da distribuição das postagens pelos países e identificar qual contribui mais que os outros para os dados obtidos e classificados.

Na figura 4.12 temos o globo terrestre, enquanto que em 4.13 está uma imagem ampliada para melhor legibilidade dos pontos dos Estados Unidos, mostrando a enorme quantidade de *tweets* advindos de lá.

O mapa-múndi foi construído utilizando-se a biblioteca *matplotlib*[21] do *python*, escolhendo-se a projeção de *Robinson* para desenhar os pontos pretos. Cada um deles representa um *tweet* com geolocalização no idioma inglês.

FIGURA 4.12: Localização dos tweets no mapa-múndi

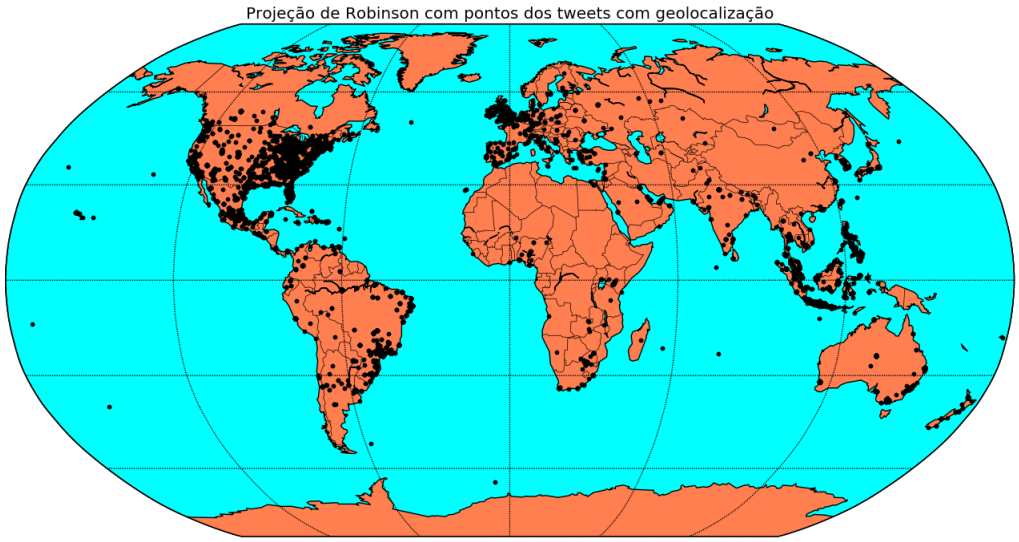
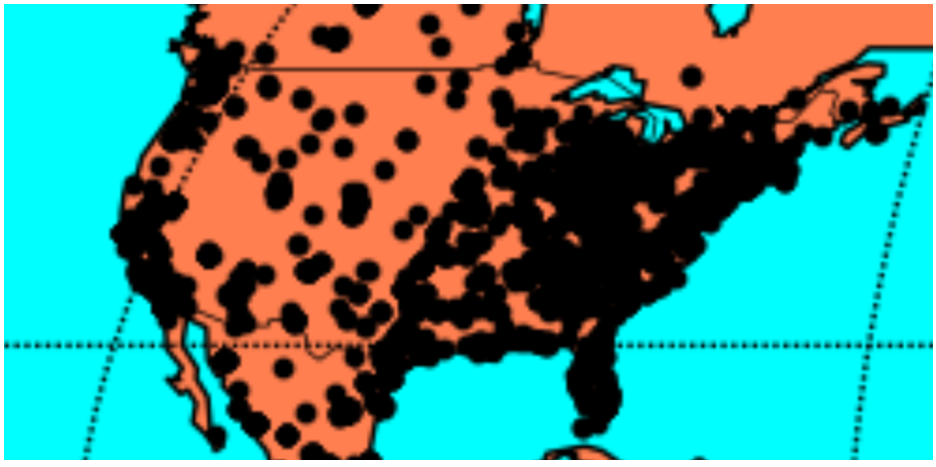


FIGURA 4.13: Mapa ampliado com os tweets dos Estados Unidos



Como pode ser visto, tanto os Estados Unidos, quanto México e a parte da Europa associada ao Reino Unido apresentam um grande volume de *tweets* sobre os filmes buscados. Como boa parte desses filmes foram produzidos nos Estados Unidos, é esperado que a população de lá assista aos filmes nacionais. Estando geograficamente próxima, a população mexicana também assistiu e comentou acerca dos filmes. E, no Reino Unido, como

FIGURA 4.14: Deleção do relacionamento TWEETED

```
MATCH p=()-[t:TWEETED]->()
DELETE t
```

a língua oficial é o inglês, também fez parte do público-alvo que assistiu e manisfestou sua opinião nas redes sociais nesse idioma.

É importante salientar que países que não possuem pontos no mapa não necessariamente não comentaram sobre os filmes no Twitter. Isso porque seus *tweets* podem estar sem a geolocalização habilitada ou em outro idioma diferente do inglês.

4.10 DELEÇÃO DOS DADOS

Embora a inserção dos dados no banco seja uma tarefa fundamental por viabilizar o armazenamento dos dados, também é necessário, em alguns casos, deletar a base da dados - ou parte dela - devido a alguns motivos: foi inserido algo que não era preciso, deseja-se esvaziar o banco para voltar ao estado original, entre outros. Tanto que as operações básicas em um banco de dados podem ser identificadas pelas iniciais *CRUD (create, read, update, delete)*, isto é, **criar** e colocar novos dados no banco (isso é feito pelos *scripts* do *workflow* para inserção de dados), **ler** as informações por intermédio de consultas, **atualizar** os dados (isso também é feito pelos *scripts* de inserção, com a operação *MERGE* que apenas cria o nó ou relacionamento se ele não existir) e **deleção**.

Para saber qual é o conteúdo a ser removido, é realizada uma consulta com *MATCH* que permite recuperar os dados desejados, seguido por um *DELETE* para concretizar a deleção.

O programa para a deleção dos dados poderia ser construído de duas formas: uma consulta de deleção poderia remover todos ou quase todos os nós e relacionamentos de uma só vez ou várias requisições menores poderiam realizar essa tarefa por partes.

Foi adotada a segunda opção por causa do *heap* do Java. O Neo4J foi implementado em Java e embora o tamanho do *heap* seja, por padrão, calculado dinamicamente com base nos recursos disponíveis no sistema (informação obtida na documentação do [20]), foi testada a operação *MATCH* para recuperar parte significativa dos dados do banco (por exemplo, relacionamentos HAS, ABOUT, TWEETED e nó Tweet) para fazer, em seguida, a deleção. Contudo, a operação fica lenta, pois o volume de dados é grande e, além disso, acaba com o limite de recursos disponíveis para o *heap*.

Um exemplo de deleção está em 4.14 que remove o relacionamento `TWEETED`.

É importante salientar que para remover os nós é preciso primeiro apagar os relacionamentos nos quais o nó está envolvido.

Assim, o programa de deleção se encontra no *workflow* construído e realiza consultas menores, como recuperar um relacionamento após o outro, deletá-los e fazer o mesmo procedimento para os nós, de forma a caber nos limites do *heap* e permitindo uma deleção mais rápida e eficiente.

5. Conclusão e Trabalhos Futuros

Este trabalho aborda a descoberta de conhecimento a partir do cruzamento de dados das publicações da rede social *Twitter* e da base de dados de filmes IMDb, a qual é pública. O trabalho aplicou técnicas de processamento, para limpar os dados, e classificação para identificar a opinião expressa nas postagens coletadas e processadas no domínio de filmes.

Os dados coletados passaram por uma etapa de pré-processamento, necessária para a eliminação de ruídos, filtragem do idioma - foram escolhidos *tweets* da língua inglesa por serem bem mais numerosos se comparados com aqueles dos demais idiomas - e escolha dos *tweets* com geolocalização habilitada - informação relevante para possibilitar a visualização da distribuição dos *tweets* pelo mundo.

Foram utilizados os classificadores *Naive Bayes* e SVM que obtiveram resultados similares para as métricas analisadas, embora o desempenho do SVM tenha sido um pouco melhor nos testes realizados sobre os dados durante o processo de validação.

O classificador consegue categorizar melhor as classes que são distintas como positivo e negativo, ruído e não ruído, do que classes próximas que apresentam *features* em comum, como as classes positivo e muito positivo.

Com as informações de cada *tweet* armazenadas no banco de dados foi possível encontrar padrões que geram conhecimento, como exemplo, identificar usuários realizando duas postagens sobre um filme. Outro padrão observado foi a presença de *tweets* em inglês no Brasil, isto é, alguns usuários brasileiros publicaram em inglês. Por ser um banco de dados orientado a grafos, a visualização fica mais intuitiva e mais fiel à realidade das redes sociais e representação dos dados.

A área de classificação de dados é bastante ampla e complexa, possibilitando sempre melhorias, seja no refinamento do texto analisado, no caso *tweets*, utilizando técnicas de processamento de linguagem natural, estudando ironia, *hashtags* e análise semântica.

Por ser um trabalho bem abrangente, algumas sutilezas foram percebidas aos poucos, conforme foram utilizados os algoritmos de aprendizagem na classificação e durante as etapas de validação e análise de dados. Assim, há possíveis extensões e trabalhos futuros que poderiam complementar o presente trabalho e são descritos abaixo.

Percebeu-se que uma das dificuldades do classificador foi a percepção da ocorrência de ironia, levando a uma classificação incorreta em alguns casos. O que foi considerado como tratamento de ironia foi a rotulação manual de alguns *tweets* irônicos, colocando-se a classificação que era considerada mais adequada, o que no caso seria o sentido oposto do que está formalmente escrito na frase. Um possível trabalho futuro seria analisar as *hashtags* separadamente do restante do *tweet* e tentar combinar ambas as

partes para identificar contradições resultantes de sarcasmo e ironia presentes no texto.

Outro estudo que poderia ser feito é com relação à proximidade de palavras negativas com as palavras-chave buscadas, porque nem sempre a crítica está relacionada ao filme analisado. Exemplos disso: uma comparação de um filme ruim, muito criticado, com o aquele que foi buscado na consulta que é considerado bom; alguém pode estar aborrecido de não ter assistido ao filme tão aguardado, o que faz a frase no geral ser negativa, embora não se trate de um comentário depreciativo do filme.

Utilizar análise semântica seria interessante para gerar relações entre nós por intermédio de inferências resultantes de dados já presentes no banco. Desse modo, no relacionamento

```
Movie-[:FROM]->Country
```

poderia-se inferir que o país em questão produziu o filme, isto é,

```
Country-[:PRODUCED]->Movie
```

A semântica poderia ampliar o conhecimento obtido com os dados.

E, por fim, aumentar o tamanho do *corpus* sempre se faz essencial, uma vez que o classificador "aprende" melhor conforme aumenta o número de exemplos rotulados que recebe como treinamento.

Parte II

Parte Subjetiva

6. Agradecimentos e disciplinas importantes

6.1 AGRADECIMENTOS

Gostaria muito de agradecer à minha família, especialmente aos meus pais e avós, por estarem sempre presentes me apoiando e incentivando durante toda minha vida, em todos os momentos e por todo amor e carinho que sempre me deram.

Também sou grata a todos os meus amigos, principalmente os ótimos amigos que fiz no IME, por todas as boas conversas que tivemos, pela ajuda e apoio porque, de fato, são necessários união, amizade e compartilhamento de experiências, não somente na faculdade, mas na vida.

Agradeço também aos professores que tive nesses anos de graduação pelos ótimos ensinamentos que foram fundamentais na minha formação. Um agradecimento especial à professora Kelly, minha orientadora, pela dedicação e atenção, me ajudando a concretizar o presente trabalho.

6.2 DISCIPLINAS IMPORTANTES PARA O TCC

Diversas disciplinas de graduação foram importantes para o TCC, mas algumas merecem destaque por estarem bastante relacionadas ao projeto:

- **Introdução à Probabilidade e Estatística I (MAE0121)**

  Apresentados conceitos básicos e fundamentais de estatística que foram necessários para entender como funcionam os classificadores e as métricas de validação.

- **Introdução à Computação (MAC0110), Princípios de Desenvolvimento de Algoritmos (MAC0122) e Estruturas de Dados (MAC0323)**

  Matérias importantes por formarem a base da lógica de programação, bem como estruturas necessárias para implementação do código.

- **Sistemas de Bancos de Dados (MAC0426) e Laboratório de Banco de Dados (MAC0439)**

  Para o entendimento dos conceitos de banco de dados e como organizar as informações de forma a facilitar as consultas realizadas para recuperar informações do banco.

- **Armazenamento e Recuperação de Informação (MAC0333)**

Disciplina importante para a escolha do tema do TCC, introduziu métricas clássicas de recuperação de informação como precisão, cobertura, medida *f*, além de introduzir uma ideia acerca de classificadores.

- **Sistemas Baseados em Conhecimento (MAC0444)**

  Por tratar de alguns tópicos de raciocínio lógico e processamento de linguagem natural.

Bibliografia

[1] Pew Research Center. http://www.pewinternet.org/fact-sheets/social-networking-fact-sheet/. [Acessado 12/11/2016].

[2] Smartinsights. http://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/. [Acessado 12/11/2016].

[3] IMDB website. http://www.imdb.com/. [Acessado 11/11/2016].

[4] Twitter. http://twitter.com/. [Acessado 11/11/2016].

[5] Statista. https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/. [Acessado 12/11/2016].

[6] The KDD process for extracting useful knowledge from volumes of data. Communications of the ACM 39.11, 1996.

[7] Bing. Liu. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2:627–666, 2010.

[8] Ana Carolina Lorena e André CPLF de Carvalho. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, 14:43–67, 2007.

[9] Schütze H. Manning C. Raghavan P. *An Introduction to Information Retrieval*. Cambridge University Press, 2009.

[10] Bernardini F. C. Lunardi A. C. Viterbo J. Um Levantamento do Uso de Algoritmos de Aprendizado Supervisionado em Mineração de Opiniões. *ENIAC*, 2015.

[11] Usama Fayyad, Gregory Piatetsky-Shapiro e Padhraic Smyth. From data mining to knowledge discovery in databases. Em *AI Magazine*, volume 17, 1996.

[12] Streaming API. https://dev.twitter.com/streaming/.

[13] OMDb API. http://www.omdbapi.com/. [Acessado 11/11/2016].

[14] Karin Becker e Diego Tumitan. Introdução à Mineração de Opiniões: Conceitos, Aplicações e Desafios. *Simpósio Brasileiro de Banco de Dados*, 2013.

[15] scikit-learn. http://scikit-learn.org/stable/. [Acessado 11/11/2016].

[16] K. Braghetto e D. Cordeiro. Workflows científicos. *XXXIII Jornadas de Atualização em Informática (JAI)*, 2014.

[17] Alexander Pak e Patrick Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. Em *LREc*, volume 10, páginas 1320–1326, 2010.

[18] Python 3 Programming Language. https://www.python.org/download/releases/3.0/. [Acessado 12/11/2016].

[19] Harry Zhang. The optimality of naive bayes. *AA*, 1:3, 2004.

[20] Neo4J Database. http://neo4j.com/. [Acessado 11/11/2016].

[21] J. D. Hunter. Matplotlib: a 2D graphics environment. *Computing In Science & Engineering*, 9:90–95, 2007.

[22] Mohammed J Zaki e Wagner Meira Jr. *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.