

INSTITUTO DE MATEMÁTICA E
ESTATÍSTICA

TRABALHO DE FORMATURA SUPERVISIONADO

Análise de Sentimentos Aplicada à Política

Lucas Romão Silva

Orientado por
Prof. Dr. Roberto Hirata Jr.

3 de fevereiro de 2018

À minha mãe, Claudete Romão Silva

Agradecimentos

Ao meu orientador, Roberto Hirata Jr. pois sem seu auxílio e orientação este trabalho não seria possível.

À minha família que sempre me apoiou ao longo desta graduação em especial à minha irmã Sarah que foi um grande apoio para mim e com certeza sem ela não teria conseguido ir até o fim.

Aos meus amigos que me apoiaram e me ajudaram ao longo de toda a jornada. Em especial à família do paninho, Antonio Abello, Alice Dias, Caroline Gonçalves, Cesar Cano, Cristiane Augusta Gabriel Nascimento e João Luciano por ficarem ao meu lado e me ajudar em diversos momentos importantes ao longo deste ano, sejam eles diretamente ligados à academia ou não.

Resumo

Com a grande expansão da internet, o uso das redes sociais tem se tornado cada vez mais ativo e, com o passar do tempo, tornou-se um lugar onde os usuários relatam não só informações acerca do cotidiano, mas também para opinar sobre temas como política, esportes, etc. O uso das redes sociais também causa maior mobilização dos usuários no cenário político através de petições online ou organização de manifestações.

Este trabalho tem como objetivo explorar a rede de usuários do Twitter para analisar os sentimentos dos seus usuários acerca do cenário político brasileiro atual utilizando técnicas de processamento de linguagem natural em conjunto de técnicas de aprendizado de máquina.

Para isso coletou-se tuítes sobre grandes decisões atuais como: (1) a Reforma da Previdência (PEC287); (2) a proposta de Lei da Terceirização; (3) a proposta de emenda constitucional sobre o Teto dos Gastos Públicos (PEC55); além de tuítes sobre políticos em grande visibilidade no cenário político atual.

Os tuítes coletados foram classificados manualmente através de uma ferramenta desenvolvida neste TCC (um site na Internet) onde era possível realizar a tarefa de classificação de forma menos árdua e permitir a ajuda de outras pessoas.

A preparação dos dados foi feita através de técnicas de processamento de linguagem natural para extrair informações subjetivas dos tuítes e classificou-se as opiniões associadas a eles utilizando técnicas de aprendizado de máquina.

Induzidos os classificadores, analisou-se métricas de desempenho de cada um como Acurácia, Precisão e Revocação para diferentes abordagens de extrair as informações subjetivas do texto. Como resultado, obteve-se uma acurácia média de 71%.

*A utopia está lá no horizonte.
Me aproximo dois passos, ela se
afasta dois passos. Caminho dez
passos e o horizonte corre dez
passos. Por mais que eu
caminhe, jamais alcançarei.
Para que serve a utopia? Serve
para isso: para que eu não deixe
de caminhar.*

Fernando Birri

Sumário

1	Introdução	1
2	Revisão Bibliográfica	3
3	<i>Machine Learning</i> e Análise de Sentimentos	5
3.1	Contextualização	5
3.2	O problema da classificação	5
3.3	Logistic Regression	6
3.3.1	Método do Gradiente	7
3.3.2	Método de Newton-Raphson	9
3.3.3	Extensão para o caso de várias classes	10
3.3.4	Evitando <i>overfitting</i>	12
3.4	Support Vector Machine	12
3.4.1	Extensão ao caso de multiclassess	16
3.5	Análise de Sentimentos	17
3.5.1	Análise de tuítes de política	18
3.5.2	Obtendo um vetor de <i>features</i>	18
3.5.3	Classificação manual da opinião	21
4	Experimentos	27
4.1	Dataset	27
4.1.1	Distribuição das classes	27
4.1.2	Características dos tuítes de cada classe	28
4.2	Descrição dos experimentos	30
4.2.1	Parâmetros avaliados	32
4.2.2	Métricas avaliadas	33
4.3	Primeira rodada de experimentos	33
4.3.1	Implementações próprias	34
4.3.2	Implementações do <code>scikit</code>	37
4.4	Segunda rodada de experimentos	41
4.4.1	Implementações próprias	42

4.4.2	Implementações do <code>scitkit</code>	44
4.5	Comparação entre experimentos	46
4.5.1	Implementações Próprias	47
4.5.2	Implementações do <code>scikit</code>	47
5	Considerações finais	49
A	Configurando e instalando o CLAM	51
A.1	Breve Introdução ao Heroku	51
A.2	Breve Introdução ao Django	51
A.3	Preparando o ambiente local	52
A.3.1	Clonando o repositório e instalando dependências	52
A.3.2	Cadastro no Heroku	53
A.3.3	Criando banco de dados e usuário local	53
A.4	Rodando o CLAM localmente	54
A.5	Fazendo deploy do CLAM no Heroku	54
	Referências Bibliográficas	55

Capítulo 1

Introdução

A expansão da internet impulsionou o uso cada vez maior das redes sociais por cada vez mais usuários. Segundo estudo realizado pelo portal Statista[1], existem 2,46 bilhões de usuários nas redes sociais do mundo todo no ano de 2017, o que contabiliza quase metade da população mundial.

As redes sociais tornaram-se um local para a manifestação de opinião dos usuários acerca de diversos assuntos. Dentre as principais redes sociais, o Twitter possui na comunidade brasileira a maior quantidade de usuários fora dos Estados Unidos com 27,7 milhões de usuários. Estudos recentes do Twitter Brasil mostram que o país foi o terceiro em maior crescimento em número de usuários no ano de 2016, avançando em 18% o número de usuários que utilizam a rede social ao menos uma vez por mês em relação ao mesmo estudo realizado em 2015.[2]

O uso do Twitter como plataforma de expressão de opiniões acerca de diversos temas fez com que muitas empresas utilizassem os dados da rede social para análise de informações relevantes. O IBM Watson possui uma integração com o Twitter onde a ferramenta é capaz de coletar e analisar os dados e revelar insights sobre os mesmos como localização, gênero dos usuários, polaridade dos tuítes e palavras indicativas de cada polaridade. [3]

No mercado financeiro a rede social também é utilizada para captar informações relevantes como exemplo da Bloomberg que disponibiliza uma ferramenta que monitora tuítes de empresas, executivos, blogs financeiros e economistas e é possível gerar alertas para quando determinado tópico ou empresa recebe muita atenção, uma vez que isso pode ser um sinal de movimentação no mercado de ações. [4]

Motivado pela popularidade do Twitter em uma parcela da população brasileira, aliada à facilidade de obter-se dados postados na rede social decidiu-se analisar as opiniões dos usuários do Twitter acerca do cenário político brasileiro atual construindo classificadores de tuítes onde cada tuíte era classi-

ficado como opinião positiva, negativa ou neutra e estudar como cada classificador e forma de representação dos dados contribui para a construção de um bom classificador.

Para conhecer o estado da arte na análise de sentimentos usando tuítes, fez-se uma revisão bibliográfica dos artigos mais citados e mais relacionados ao trabalho, em especial aqueles que tratavam do problema da análise de tuítes de política. Com os artigos pesquisados, teve-se uma noção de possíveis problemas que poderiam ser encontrados no trabalho como a presença de ironia que atrapalha a classificação e partes do texto que poderiam trazer mais informações relevantes como as *hashtags* e emojis. Ademais os artigos forneceram uma base de como pré-processar o texto para então treinar os modelos.

Para isso, coletou-se tuítes de julho de 2016 a julho de 2017 sobre as reformas da previdência, lei da terceirização, a proposta de emenda constitucional do teto dos gastos e sobre políticos com notoriedade no cenário brasileiro atual como os ex-presidentes Luis Inácio Lula da Silva, Dilma Rousseff e o senador Aécio Neves.

Ao longo deste trabalho foi desenvolvido uma ferramenta (CLAM) para auxiliar na tarefa da classificação manual dos dados e que estivesse disponível para ser utilizada por outras pessoas da comunidade se assim quisessem. Disponibilizou-se também neste trabalho um tutorial de como subir uma instância do CLAM no Heroku [A](#).

Além do CLAM, implementou-se dois classificadores, um utilizando o método de regressão logística e outro utilizando Support Vector Machines (SVM) tanto para a versão binária da classificação quanto para três classes (opinião positiva, negativa e neutra) e comparou o desempenho destas implementações com as já disponibilizadas pelo `scikit` do Python a fim de verificar se ambas apresentavam mesma acurácia e precisão sobre os dados de teste.

Após esta Introdução, no capítulo 2 é apresentada uma revisão bibliográfica dos artigos lidos que foram mais relevantes para o trabalho, no capítulo 3 é definido os modelos de classificação que serão usados bem como explica-se análise de sentimentos. No capítulo 4 são comentados os resultados dos experimentos realizados e analisa-se o conjunto de dados. Por último no capítulo 5 são discutidos os resultados e perspectivas futuras para o trabalho.

Capítulo 2

Revisão Bibliográfica

Diversos estudos recentes têm sido realizados na área de análise de sentimentos. Medhat (2014)[5], sumariza diversos estudos feitos dividindo-os em abordagens (usando técnicas de aprendizado de máquina, dicionário léxico ou híbrida) e em objetivos (classificação de sentimentos, detecção de emoções etc).

Em Pak (2010)[6] é analisado o uso de um corpus composto de tuítes para a realização de tarefas de análise de sentimentos e mineração de opinião. Nesse estudo coletou-se 300000 tuítes divididos igualmente entre opiniões positivas, negativas e neutras, para isso é descrito um método de como coletar textos com opiniões positivas e negativas procurando por textos que contivessem emoticons comumente associados a opiniões positivas e negativas e para textos de sem nenhuma polaridade, coletou-se textos diretamente de portais de notícias.

Além da descrição de como extrair tuítes de cada categoria, o trabalho realiza uma análise linguísticas de quais palavras são mais presentes em textos objetivos e em subjetivos, ao exemplo de superlativos em textos positivos e verbos conjugados na terceira pessoa ou no passado serem predominantes em textos objetivos. Para a classificação dos textos é usado o método Naive Bayes e o trabalho descreve algumas formas de melhorar o desempenho do algoritmo como o uso de bigramas, e filtrar os n-gramas na hora de treinar o modelo além de analisar como o aumento do tamanho do conjunto de dados contribui para a melhora da acurácia.

Outros estudos, assim como este trabalho, têm como foco o uso de análise de sentimentos aplicados à política utilizando tuítes como o corpus. Em Tumasjan et. al (2010)[7] é estudado se é possível utilizar o Twitter para obter uma previsão para os resultados de uma eleição tendo como base as eleições do parlamento alemão realizadas em 2009.

Para isso, usou-se mais de 100 mil tuítes contendo referências a políticos

ou aos respectivos partidos e extraiu-se os sentimentos associados aos textos usando a ferramenta LIWC (Linguistic Inquiry and Word Count), ferramenta usada para avaliar a intensidade do texto sob uma certa perspectiva (ou dimensão como é chamado no texto) como emoções positivas e negativas, raiva, ansiedade, raiva entre outros, e a partir da análise dessa extração foi estudado se:

1. A atenção dada a cada partido / candidato no Twitter reflete os resultados das eleições, onde utilizando o erro médio absoluto obteve-se uma taxa de erro de apenas 1,65%, valor próximo aos obtidos em outros meios de pesquisa aceitos.
2. Verificar se a rede social pode dar insights sobre possíveis alianças partidárias pós-eleições e quais ideologias em comum existem entre os partidos.

Risquandl e Petković (2013)[8] realizam experimentos tendo como cenário as eleições presidenciais estado-unidenses de 2012 para classificar sentimentos dos usuários do Twitter, porém diferente dos outros trabalhos anteriormente mencionados, esse estudo utiliza técnicas de *part-of-speech tagging* para realizar extração de aspecto, isto é, a quem o tuíte se refere. A técnica em questão extrai subjetivos que podem ser consideradas como tópicos de campanha e mede-se a correlação entre o termo e cada candidato usando uma medida de associação.

Por último, utilizou-se um léxico de sentimentos para classificar a opinião relativa a cada tópico de campanha de cada candidato e, a partir disso, elaborou-se um sumário das opiniões acerca de cada tópico para melhor visualização das opiniões dos usuários acerca de cada tópico relevante para cada candidato.

Bakliwal et. al (2013)[9] realiza um estudo semelhante ao deste trabalho a classificação de opinião tendo como base três classes (positivo, negativo ou neutro) acerca de tuítes coletados sobre as eleições gerais da Irlanda em fevereiro de 2011. A diferença entre os trabalhos é que Bakliwal divide as classificações das opiniões entre cada partido.

Outra diferença entre este trabalho e os demais é que neste os tuítes são analisados não só por meio de um método de aprendizado de máquina, mas também utilizando um léxico de sentimentos. Importante comentar que neste artigo é discutido o método como a classificação manual foi feita: utilizando mais de um avaliador e, caso ouvesse discordância, a opinião de um terceiro seria pedida além de especificar mais classes para as quais os tuítes poderiam ser classificados (como tuítes que não estão em inglês ou que não falam sobre o tópico além de tuítes de opinião mista).

Capítulo 3

Machine Learning e Análise de Sentimentos

3.1 Contextualização

Os problemas tratados por *Machine Learning* classificam-se de forma geral em três tipos:

- Aprendizado supervisionado: nesse caso tem-se os elementos de entrada e para cada um desses elementos, tem-se associado um rótulo. Nesse caso o modelo deve ser treinado com base nos elementos dados para que se possa prever o rótulo de uma nova entrada;
- Aprendizado não-supervisionado: nesse caso tem-se apenas os elementos de entrada. O objetivo deste tipo de problema é tentar modelar uma distribuição ou estrutura comum entre os dados para que se possa entendê-los melhor;
- Aprendizado semi-supervisionado: nesse último caso alguns elementos possuem um rótulo associado. Problemas desse tipo aplicam técnicas tanto de aprendizado supervisionado como de não-supervisionado.

3.2 O problema da classificação

Neste trabalho será tratado um problema de aprendizado supervisionado que é o da classificação. No problema de classificação, tem-se K classes $\{1, \dots, K\}$ com $K \geq 2$ e o objetivo é treinar um modelo que, para uma dada entrada $x \in \mathbb{R}^m$, determine qual classe ela pertence.

Um exemplo de problema classificação é o caso do *iris dataset* onde se classifica uma flor iris entre três tipos *Iris setosa*, *Iris virginica* e *Iris versicolor* baseado nos tamanhos das pétalas e sépalas.

Para a etapa de treinamento, utiliza-se um conjunto $X = (x_1, x_2, \dots, x_n)$ e para cada $x_i \in \mathbb{R}^m$ está associado um $t_i \in \{1, \dots, K\}$. Será usado n para o tamanho do conjunto de entrada e m para o tamanho de cada vetor. Os modelos de classificação que serão discutidos neste trabalho realizam a etapa de treinamento resolvendo um problema de otimização que tem como fim direta ou indiretamente diminuir o erro associado a classificação de um elemento, além disso é necessário que o modelo final tenha boa generalização, isto é, apresente baixo erro para a classificação de novas amostras além das presentes no conjunto de entrada.

Qual função objetivo a ser otimizada e como fazer isso varia para cada método uma vez que um método não necessariamente se aplica a demais.

Como será discutido mais a frente no capítulo, cada modelo possui diferentes abordagens para o caso binário e o caso multiclass, porém os fundamentos são os mesmos portanto será explicado tanto para a regressão logística quanto para o SVM o caso binário e depois irá estender para o problema a ser resolvido neste trabalho.

3.3 Logistic Regression

Na regressão logística o modelo de classificação é probabilístico o que significa que, a partir de um discriminante linear atribui-se a probabilidade do elemento x pertencer à classe C^1 , denotada por $P(C^1|x)$. O discriminante linear é dado pela equação:

$$f(x) = w^T x + w_0, \quad (3.1)$$

o vetor $\tilde{w} = (w, w_0)$ é chamado de vetor de pesos e o termo w_0 é chamado de viés. Para efeitos de contas será usado w para se referir ao vetor \tilde{w} e o vetor x será dado por $x = (x_{original}, 1)$.

Utilizando 3.1 obtém-se a probabilidade $P(C^1|x)$ utilizando a função sigmóide sobre $f(x)$ que é dado por:

$$P(C^1|x) = y(x) = \sigma(f(x)) = \frac{1}{1 + e^{-f(x)}} \quad (3.2)$$

e consequentemente $P(C^2|x) = 1 - P(C^1|x)$. A classe ao qual um elemento x pertence é aquela que possui maior probabilidade. Utiliza-se a sigmoide, pois a função possui a propriedade de mapear o conjunto dos números reais no intervalo $[0, 1]$.

No caso da classificação binária, o rótulo de um elemento t_i corresponde a 1 se o elemento x_i pertence à classe C^1 e 0 se pertence a C^2 .

O objetivo do processo de treinamento da regressão logística é obter um vetor w que minimize o erro da classificação. Para obter-se a função de erro, primeiro calcula-se o estimador de máxima verossimilhança para $P(C^1|x)$:

$$p(t|w) = \prod_{i=1}^n y_i^{t_i} (1 - y_i)^{1-t_i}, \quad (3.3)$$

com $t = (t_1, \dots, t_n)$ e $y_i = P(C^1|x_i)$.

Uma vez que 3.3 é uma função difícil de se otimizar, uma vez que possui um produtório de funções exponenciais, toma-se o negativo do logaritmo de 3.3, que será a função de erro. Assim obtém-se:

$$\begin{aligned} E(w) &= -\log(p(t|w)) = -\log\left(\prod_{i=1}^n y_i^{t_i} (1 - y_i)^{1-t_i}\right) \\ &= -\sum_{i=1}^n \{t_i \log(y_i) + (1 - t_i) \log(1 - y_i)\} \end{aligned} \quad (3.4)$$

Dois métodos são comumente usados para minimizar 3.4: método do gradiente e método de Newton-Raphson. Esses métodos são utilizados tanto para o caso da classificação binária quanto o caso da classificação com $k > 2$. A diferença entre eles será explicada nas subseções que seguem.

Uma dúvida natural que surge ao ter que resolver um problema de otimização é não obter um minimizador global, entretanto, tem-se função 3.4 é convexa, isto é, $E(\lambda w + (1 - \lambda)w') \leq \lambda E(w) + (1 - \lambda)E(w') \forall w, w' \in \mathbb{R}^{m+1}, \lambda \in [0, 1]$, e portanto tem-se que existe um único minimizador.

3.3.1 Método do Gradiente

O método do gradiente, como o nome sugere, minimiza a função objetivo, no caso 3.4 utilizando o gradiente da função $\nabla E(w)$ e uma constante $\alpha \in \mathbb{R}$ chamada de passo. O valor do passo deve ser escolhido cautelosamente, pois

8CAPÍTULO 3. MACHINE LEARNING E ANÁLISE DE SENTIMENTOS

influencia diretamente na quantidade de iterações necessárias para convergência. Valores baixos de α acarretam em muitas iterações ao passo que valores muito altos fariam que convergisse para um valor distante do ótimo.

Por convergência entende-se como um critério de parada do algoritmo que no caso é caso o número de iterações passe de um limite pré-estabelecido (200 iterações neste caso) ou que a diferença do valor de $E(w)$ entre uma iteração e outra seja inferior a um ϵ também pré-estabelecido (no caso 10^{-4}).

O valor de $\nabla E(w)$ é calculado utilizando a derivada de 3.2 que é dada por:

$$\begin{aligned}\frac{d\sigma}{da} &= \frac{\left(d\frac{1}{1+e^{-a}}\right)}{da} \\ &= \frac{e^{-a}}{(1 - e^{-a})^2} = \frac{1}{1 - e^{-a}} \frac{e^{-a}}{1 - e^{-a}} \\ &= \sigma(a) * (1 - \sigma(a))\end{aligned}\tag{3.5}$$

usando 3.5 obtém-se a fórmula do gradiente de $E(w)$:

$$\nabla E(w) = \sum_{i=1}^n (y_i - t_i) x_i \text{ ou } = X^T (y - t) \text{ em notação vetorial,}\tag{3.6}$$

com $y = (y_1, \dots, y_n)$ e $t = (t_1, \dots, t_n)$ onde $y_n = P(C^1|x_n) = \sigma(w^T x)$ e t_n tal qual assumido no começo da seção.

Utilizando o valor obtido em 3.6 atualiza-se w usando a fórmula:

$$w^{(novo)} = w^{(antigo)} + \alpha \nabla E(w),\tag{3.7}$$

As fórmulas 3.6 e 3.7 constituem o procedimento realizado a cada iteração do método do gradiente, descrita no algoritmo abaixo.

Algorithm 1 Logistic Regression usando método do gradiente**Input:** Matriz $X \in \mathbb{R}^{n \times m}$, vetor de rótulos $t \in \{0, 1\}^n$ **Output:** Vetor de pesos $w \in \mathbb{R}^{m+1}$

```

1:  $iteracao \leftarrow 0$ 
2:  $w \leftarrow 0$ 
3: while  $|E(w)^{(iteracao)} - E(w)^{(iteracao-1)}| \geq \epsilon$  and  $iteracao < maxIteracoes$ 
   do  $\triangleright$ critérios de convergência
4:    $y \leftarrow (\sigma(w^T x_1), \sigma(w^T x_2), \dots, \sigma(w^T x_n))^T$   $\triangleright$ aplica a função
      $\triangleright$ discriminante sobre cada entrada de  $X$ 
5:    $\nabla E(w) \leftarrow X^T(y - t)$   $\triangleright$ computa o gradiente da função
6:    $w \leftarrow w - \alpha \nabla E(w)$   $\triangleright$ atualização do modelo
7:    $E(w)^{(iteracao)} \leftarrow -\sum_{i=1}^n \{t_i \log(y_i) + (1 - t_i) \log(1 - y_i)\}$   $\triangleright$ recalcula o
      $\triangleright$ erro
8:    $iteracao \leftarrow iteracao + 1$ 
9: end while

```

3.3.2 Método de Newton-Raphson

Em 3.3.1 viu-se o método do gradiente, que apesar da simplicidade de implementação, pode demorar para convergir e possui uma constante que deve ser cuidadosamente escolhida.

O método de Newton-Raphson, em contrapartida, não utiliza constantes a serem definidas e converge mais rápido, ao custo de um custo computacional maior comparado ao método do gradiente.

A atualização agora utiliza o hessiano da função erro que é dado por:

$$\begin{aligned}
 H &= \nabla \nabla E(w) = \frac{\partial \nabla E(w)}{\partial w} \\
 &= \frac{\partial X^T(y - t)}{\partial w} = X^T \frac{\partial y}{\partial w} \quad (\text{t não depende de } w) \\
 &= X^T R X
 \end{aligned} \tag{3.8}$$

com R sendo uma matriz diagonal onde $R_{ii} = y_i(1 - y_i)$.

Juntando 3.6 e 3.8, a nova atualização é dada pela fórmula:

$$\begin{aligned}
 w^{(novo)} &= w^{(antigo)} - H^{-1} \nabla E(w) \\
 &= (X^T R X)^{-1} [(X^T R X) w^{(antigo)} - X^T(y - t)] \\
 &\quad (\text{substituindo os valores de 3.6 e 3.8})
 \end{aligned} \tag{3.9}$$

O restante do procedimento da iteração do algoritmo de treino é semelhante ao método do gradiente, portanto o pseudocódigo será omitido nesta parte.

3.3.3 Extensão para o caso de várias classes

Para o caso multiclasse utiliza-se K discriminantes lineares a_i , $i = \{1, \dots, K\}$ e, conseqüentemente, tem-se que o vetor de pesos agora é dado por $W = (w_1, \dots, w_K)$ com $w_i \in \mathbb{R}^m$ e, conseqüentemente cada a_i é definido por $a_i(x) = w_i^T x$.

Os rótulos também precisam de uma codificação diferente da usada no caso binário. Usa-se a codificação dada por Bishop (2006)[10] de $1 - K$ onde cada rótulo $t_i \in \{0, 1\}^K$ com $t_{ic} = 1$ se o i -ésimo elemento pertencer à classe c e 0 caso contrário e o vetor de rótulos agora é uma matriz $T \in \{0, 1\}^{n \times K}$ com cada linha T_i correspondendo ao t_i definido no começo do parágrafo.

Quanto a função de probabilidade que deseja-se estimar, utiliza-se a função *softmax* que é dada pela equação:

$$P(C^i|x_n) = y_{ni} = \frac{\exp(a_i(x_n))}{\sum_j \exp(a_j(x_n))} \quad (3.10)$$

Assim como no caso binário, a classe de um elemento x_i é dada pela que possuir maior probabilidade. Para derivar a função erro, novamente utiliza-se a máxima verossimilhança de 3.10

$$P(T|W) = \prod_{i=1}^n \prod_{j=1}^K P(C^j|x_i)^{T_{ij}} = \prod_{i=1}^n \prod_{j=1}^K y_{ij}^{T_{ij}}, \quad (3.11)$$

a partir da verossimilhança, obtém-se a função de erro,

$$E(w) = -\log(P(T|W)) = -\sum_{i=1}^n \sum_{j=1}^K T_{ij} \log y_{ij} \quad (3.12)$$

Os métodos para encontrar W que minimize 3.12 são os mesmos discutidos em 3.3.1 e 3.3.2, porém agora tem-se que as fórmulas para o gradiente e hessiano são diferentes das discutidas previamente. Deriva-se o gradiente usando o fato de que a derivada de 3.10 com respeito a cada discriminante a_j é dada por:

$$\begin{aligned}
\frac{\partial y_i}{\partial a_j} &= \frac{\partial \frac{e^{a_i}}{\sum_l e^{a_l}}}{\partial a_j} \\
&= \frac{e^{a_i} e^{a_j}}{(\sum_l e^{a_l})^2} = \frac{e^{a_i}}{\sum_l e^{a_l}} \frac{e^{a_j}}{\sum_l e^{a_l}} = y_i y_j \text{ assumindo } j \neq i \text{ ou,} \\
&= \frac{e^{a_i} (\sum_l e^{a_l}) - (e^{a_i})^2}{(\sum_l e^{a_l})^2} \text{ assumindo } j = i \\
&= \frac{e^{a_i} (\sum_{l \neq i} e^{a_l})}{(\sum_l e^{a_l})^2} = y_i (1 - y_i)
\end{aligned} \tag{3.13}$$

Aplicando o valor de 3.13 para cada w_i , obtém-se a fórmula para o gradiente

$$\nabla_{w_j} E(W) = \sum_{i=1}^n (y_{ij} - T_{ij}) x_i \text{ ou } = X^T (Y_j - T_j) \text{ em notação matricial,} \tag{3.14}$$

Com Y_j e T_j correspondendo, respectivamente, às j -ésimas colunas de Y e T .

Com o gradiente em mãos tem-se o que é necessário para o método do gradiente e a atualização seria feita da forma $W^{(novo)} = W^{(antigo)} - \alpha \nabla E(W)$.

Para aplicar o método de Newton-Raphson, seria necessário computar o Hessiano que nesse caso seria uma matriz $m * k \times m * k$ com cada bloco (j, i) contendo uma matriz $m \times m$ calculada pela equação:

$$\nabla_{w_i} \nabla_{w_j} E(W) = - \sum_{l=1}^n y_{li} (\mathbb{1}_{i=j} - y_{lj}) X_l^T X_i \tag{3.15}$$

onde X_i é a i -ésima linha de X e $\mathbb{1}_{i=j}$ é uma função indicadora que vale 1 se i for igual a j e 0 caso contrário. Com essas equações em mãos a atualização de W seria feita usando a fórmula:

$$W^{(novo)} = W^{(antigo)} - H^{-1} \nabla E(W) \tag{3.16}$$

3.3.4 Evitando *overfitting*

Tanto para o caso de classificação binária quanto para o de várias classes, corre-se o risco de que o modelo obtido após o treinamento performe muito bem no conjunto de dados fornecido a ele, porém possui uma generalização ruim. Esse efeito é chamado de *overfitting*. Para evitar que ocorra *overfitting*, adiciona-se uma penalização $\lambda||W||^2$ à função de erro. O valor escolhido para λ é importante para obter-se um bom modelo. Um λ pequeno corresponde a baixa penalização e consequentemente risco de *overfitting* ao passo que valores muito altos iriam acarretar em valores muito baixos para cada entrada de W .

Utilizando a penalização, tem-se as seguintes funções de erro, gradiente e hessiano

$$E(W) = - \sum_{i=1}^n \sum_{j=1}^K t_{ij} \ln(y_{ij}) + \frac{\lambda}{2} ||W||^2 \quad (3.17a)$$

$$\nabla E(W) = X^T(Y_j - T_j) + \lambda ||W|| \quad (3.17b)$$

$$\nabla_{w_i} \nabla_{w_j} E(W) = - \sum_{l=1}^n y_{li} (\mathbb{1}_{i=j} - y_{lj}) X_i^T X_l + \lambda * I_{mK} \quad (3.17c)$$

onde I_{mK} é a matriz identidade com mK linhas e mK colunas.

3.4 Support Vector Machine

Assim como feito com a regressão logística, será dada a definição para o caso binário e depois se estenderá para o caso multiclases. Nesse caso as classes serão $t_i \in \{-1, 1\}$ onde $t_i = 1$ se x_i pertence à classe C^1 e $t_i = -1$ se pertence à classe C^2 .

No algoritmo SVM a classificação é feita a partir de um discriminante linear da forma

$$y(x) = w^T x + b \quad (3.18)$$

Tal $y(x)$ é chamado de superfície de decisão e a classificação é baseada no sinal de $y(x)$. Se $y(x) > 0$, x é atribuído à classe C^1 , caso contrário é atribuído à classe C^2 .

Porém ao invés de procurar um w que separe perfeitamente todas as classes (que não necessariamente existe), o objetivo é maximizar a margem do discriminante linear, isto é, a menor distância de um ponto à superfície. A distância de um ponto x_i à superfície é dado pela fórmula

$$\frac{[t_i(w^T x_i + b)]}{||w||} \quad (3.19)$$

Na descrição inicial do problema será tratado o caso em que o conjunto X linearmente separável, o que indica que é possível obter uma superfície que separe sem erro todas as classes para, em seguida, tratar o caso real que é o do conjunto que não é linearmente separável. O fato de o conjunto ser linearmente separável garante que $t_i y(x_i) > 0, \forall i$. Como o problema em questão é maximizar a margem é equivalente a querer maximizar a distância do ponto mais próximo à superfície de decisão, como dado pela fórmula abaixo:

$$\operatorname{argmax}_{x,b} \left\{ \frac{1}{||w||} \min_i [t_i(w^T x_i + b)] \right\} \quad (3.20)$$

Pode-se ajustar w e b de forma a ter que $t_i(w^T x_i + b) = 1$ para o ponto mais próximo da margem e $t_i(w^T x_i + b) \geq 1, \forall i$. Com esse reajuste tem-se que o problema de encontrar um vetor de pesos de margem maximizada seria de maximizar $\frac{1}{||w||}$ que é equivalente ao problema de otimização quadrática:

$$\begin{aligned} & \operatorname{argmin}_{w,b} ||w||^2 \\ & \text{sujeito a } t_i(w^T x_i + b) \geq 1, \quad i = 1, \dots, n \end{aligned} \quad (3.21)$$

Agora supõe-se que a entrada não é linearmente separável, isto é, não existe uma superfície de decisão que separe perfeitamente as duas classes. Assim é permitido que alguns valores estejam classificados incorretamente, para isso será necessário suavizar a margem penalizando cada uma das entradas incorretamente classificadas. Cortes e Vapnik (1995)[11] descrevem a penalização através da introdução de variáveis de folga $\xi_i \geq 0$ para cada elemento de X . Um elemento corretamente classificado terá $\xi_i = 0$, os demais pontos têm $\xi_i = |t_i - y(x_i)|$.

Com isso a restrição de $t_i y(x_i) \geq 1$ é modificada e tem-se $t_i y(x_i) \geq 1 - \xi_i \forall i$.

O valor de ξ_i assim nos indica três possíveis casos:

- Se $\xi_i = 0$, x_i está corretamente classificado e se encontra ou na margem ou do lado correto dela.

- Se $0 < \xi_i \leq 1$, x_i está corretamente classificado e se encontra entre a margem e a superfície.
- Se $\xi_i > 1$, x_i não está classificado corretamente.

A função objetivo agora precisa conter os valores de ξ e para isso coloca-se uma constante $C > 0$ que define a compensação entre a penalização das variáveis de folga e a margem. Com isso tem-se o seguinte problema de otimização quadrática:

$$\begin{aligned} \underset{w, b, \xi}{\operatorname{argmin}} \quad & C \sum_{i=1}^n \xi_i + \frac{1}{2} \|w\|^2 \\ \text{sujeito a} \quad & t_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, n \end{aligned} \quad (3.22)$$

Para implementar o SVM basta então resolver o problema de otimização quadrática acima. Isto é feito seguindo os seguintes passos

1. Introduz-se multiplicadores de Lagrange α_i e μ_i e obtém-se o Lagrangiano com as restrições[12]

$$L(w, b, \xi) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i \{t_i(w^T x_i + b) - 1 + \xi_i\} - \sum_{i=1}^n \mu_i \xi_i \quad (3.23a)$$

$$\alpha_i \geq 0 \quad (3.23b)$$

$$t_i y(x_i) - 1 + \xi_i \geq 0 \quad (3.23c)$$

$$\alpha_i \{t_i(w^T x_i + b) - 1 + \xi_i\} = 0 \quad (3.23d)$$

$$\mu_i \geq 0 \quad (3.23e)$$

$$\mu_i \xi_i = 0 \quad (3.23f)$$

2. Deriva-se o lagrangiano com respeito a w , b e ξ e iguala-se a 0 para obter os valores ótimos para essas variáveis e, assim obtemos os seguintes valores:

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^n \alpha_i t_i x_i \quad (3.24)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^n \alpha_i t_i = 0 \quad (3.25)$$

$$\frac{\partial L}{\partial \xi} = 0 \Rightarrow \alpha_i = C - \mu_i \quad (3.26)$$

3. Com isso em mãos, resolve-se não o problema primal e sim o dual (utiliza-se aqui o fato de que se as condições mencionadas no item anterior são satisfeitas, tem-se que vale a dualidade forte e o valor ótimo de ambas as funções coincide). O dual é dado pelo problema

$$\begin{aligned} \operatorname{argmax}_{\alpha} \quad & \tilde{L}(\alpha) = \sum_{i=1}^n \alpha_i - 1/2 \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j t_i t_j k(x_i, x_j) \\ \text{sujeito a} \quad & 0 \leq \alpha_i \leq C, i = 1, \dots, n \\ & \sum_{i=1}^n \alpha_i t_i = 0 \end{aligned} \quad (3.27)$$

Ao resolver o dual do problema utiliza-se uma técnica que é chamado de truque do Kernel, que se consiste no uso de uma função chamada kernel que transforma os elementos do conjunto de entrada para elementos de outro espaço vetorial, usualmente de dimensionalidade maior. A grande vantagem do uso desse truque é que o mapeamento é feito indiretamente, isto é, não se computa o valor de uma transformação linear elemento por elemento pois o que é computado é apenas o produto interno de dois elementos desse novo espaço vetorial valendo que $k(x, x') = \phi(x)^T \phi(x')$ com ϕ sendo alguma transformação linear.

4. Acha-se o valor de b usando a fórmula:

$$b = \frac{1}{N_{\mathcal{M}}} \sum_{i \in \mathcal{M}} \left(t_i - \sum_{j \in \mathcal{S}} \alpha_j t_j k(x_i, x_j) \right) \quad (3.28)$$

Com \mathcal{M} sendo o conjunto de pontos que satisfazem $0 < \alpha_n < C$ e \mathcal{S} o conjunto de vetores de suporte (pontos que possuem $\alpha_n > 0$ e, consequentemente, contribuem para a classificação do modelo).

Uma vez resolvido o problema dual e encontrado valor de b , pode-se classificar um novo x usando o sinal do discriminante $y(x)$ dado por

$$y(x) = \sum_{i=1}^n \alpha_i t_i k(x, x_i) + b = \sum_{i \in \mathcal{S}} \alpha_i t_i k(x, x_i) + b \quad (3.29)$$

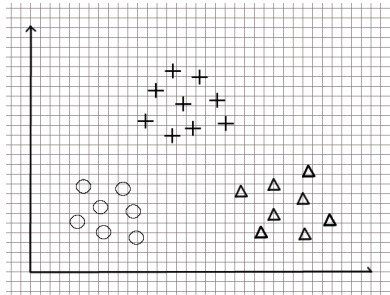
3.4.1 Extensão ao caso de multiclass

A abordagem usada neste trabalho para o caso multiclass do SVM foi o método intuitivo chamado *One-versus-all* (OVA). Nesse método são construídos K classificadores y_i cada um definindo uma superfície de decisão que separa a classe i das demais (por isso o nome *One-versus-all*). Um novo x tem sua classe dada pela que o classificador y_i tem maior valor, isto é

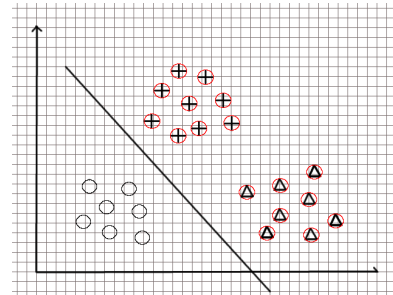
$$y(x) = \underset{i}{\operatorname{argmax}} y_i(x) \quad (3.30)$$

Portanto a classificação multiclass se utiliza de todos os recursos já apresentados no caso binário o que torna simples a construção do classificador.

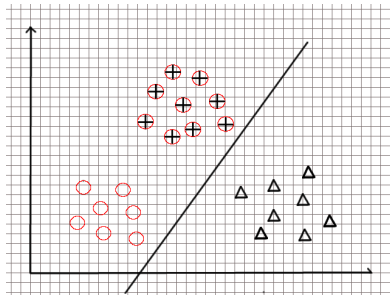
Na imagem abaixo é mostrado a ideia por trás do método OVA nele a classe C^1 é dada pelos círculos, C^2 pelos triângulos e C^3 pelas cruzes. As figuras em vermelho em cada classificador são os pontos onde $t_i = -1$.



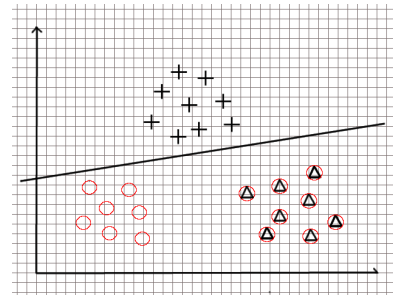
(a) Classes originais



(b) Classificador para a classe 1



(c) Classificador para a classe 2



(d) Classificador para a classe 3

3.5 Análise de Sentimentos

O campo de análise de sentimentos, também conhecido como mineração de opinião é uma área da ciência da computação que analisa as opiniões, sentimentos e atitudes das pessoas em relação a uma entidade (Bing Liu, 2012)[13]. Uma entidade pode ser definida como o sujeito ao qual está se observando as opiniões, seja ela uma pessoa, instituição ou produto.

A área possui uma diversa gama de aplicações, dentre elas, a área de classificação de sentimentos que através do uso de informações subjetivas contidas num texto avalia qual a opinião relacionada ao mesmo.

Classificação de sentimentos teve um crescimento devido à expansão da Web 2.0 que fez com que as pessoas manifestassem suas opiniões e sentimentos através de blogs, fóruns, redes sociais e páginas de reviews de produtos que, conseqüentemente fez com que empresas e pessoas de interesse tivessem um acesso mais aberto e fácil a essas opiniões. O fácil acesso às opiniões dos usuários junto com o grande volume delas tornou a análise manual um processo muito custoso, tornando necessário recorrer a métodos automatizados para a análise e sumarização desses dados que acabou por fim impulsionando estudos na área (Petković e Ringsquandl, 2013)[8].

Bing Liu (2012)[13] divide as formas de classificar sentimentos em um documento em quatro formas:

- Entidade: um produto, pessoa sobre o qual o texto se refere, seja direta ou indiretamente. Nessa tarefa procura-se analisar se a opinião do texto em torno de uma entidade é positiva ou não.
- Aspecto: características sobre uma entidade. Por exemplo, se tem-se uma entidade que é um produto, aspectos de um produto poderiam ser material ou preço. Nessa forma procura-se avaliar a opinião acerca de cada um dos aspectos.
- Sentença: esse tipo de análise trabalha com o sentimento associado a cada sentença de um documento.
- Documento: nesse caso analisa-se o sentimento associado a um documento como um todo, tratando de uma forma mais genérica em relação aos demais.

Comum a todas as formas de se analisar as opiniões de um documento é o uso dos métodos para obter a opinião. Há duas abordagens para esse problema de classificação: a de aprendizado de máquina (que iremos utilizar nesse trabalho) que consiste no uso de algoritmos de classificação em conjunto com técnicas de processamento de texto (que será explicado nas próximas

seções) e a abordagem com um dicionário léxico onde a análise é feita baseada na pontuação entre palavras positivas e negativas contidas no documento (Medhat et al., 2014) [5].

No caso desse trabalho, foi escolhido a análise em torno do documento / sentença (devido à estrutura dos tuítes tem-se que os documentos são, em sua maioria, compostos por apenas uma sentença).

3.5.1 Análise de tuítes de política

Dentre os domínios de aplicação da classificação de sentimentos, escolheu-se como domínio o escopo político. Nesse caso, tem-se que as entidades nos documentos são constituídas por políticos e projetos de lei. E neste trabalho serão analisados tuítes.

O Twitter desde as eleições presidenciais estadunidenses de 2008 mostrou influência na decisão das eleições evidenciado pela campanha online realizada pela equipe do candidato Barack Obama (Petković e Ringsquandl, 2013)[8]. Desde então, a rede social tem sido usada não só para a campanha de políticos, mas também para a avaliação da opinião em relação às medidas tomadas por eles. A escolha foi feita tendo em vista o uso da rede social para expressar opiniões sobre diversos assuntos, incluindo política.

3.5.2 Obtendo um vetor de *features*

Para realizar a extração das informações do texto, existe uma etapa de pré-processamento na qual cada texto é transformado num vetor numérico para então ser processado por algum algoritmo de classificação, as mais famosas são o modelo *bag of words* (BOW) e o n-grama.

Antes de obter um vetor de *features*, é feito um pré-processamento no documento removendo *stop-words* (palavras muito usadas na língua portuguesa como artigos e preposições), caracteres especiais, adicionando espaços antes e depois de pontuações e transformando em um vetor de *tokens* com cada token representando as palavras do texto. Uma vez que foi obtido o vetor de *tokens*, é removido todo elemento que seja apenas espaço e pontuação.

Na abordagem n-grama os termos não são apenas as palavras, mas um conjunto de n palavras juntas, o que traz mais informação pelo fato de juntar substantivo e verbo, substantivo e adjetivo, por exemplo, entretanto acaba gerando um vetor de *features* ainda maior que, consequentemente, faz com que os algoritmos demorem mais para convergir [14]. A figura 3.2 exemplifica o funcionamento do n-grama.

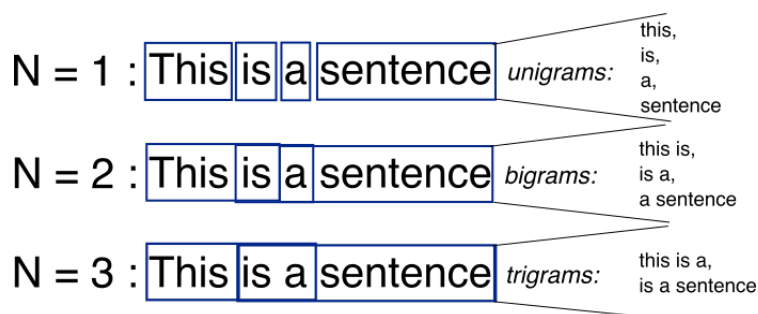


Figura 3.2: Funcionamento do n-grama para $n = 1, 2$ e 3

Na abordagem usando o BOW, constrói-se uma matriz de entrada onde cada linha i representa um documento (texto sobre o qual quer extrair a opinião) e as colunas representam a frequência de um termo, o conjunto de todos os textos é chamado de *corpus* [15]. O funcionamento do BOW é ilustrado em 3.3.

the dog is on the table



Figura 3.3: Funcionamento do *Bag-of-Words*

A escolha pelo BOW foi tida baseada no compromisso entre desempenho e performance do algoritmo.

Após tokenizar os textos, monta-se uma nova *string* juntando todas as *tokens* usando espaços para então montar um vetor de frequência a partir do novo *corpus* higienizado.

A medida de frequência de um termo pode ser uma medida simples como a quantidade de vezes em que um termo j aparece no documento i ou se o termo j aparece no documento i (nesse caso cada vetor de entrada seria um vetor binário) bem como pode ser uma medida mais complexa como TF-IDF (*term frequency - inverse document frequency*). A relação entre os métodos de computar a frequência e o desempenho dos algoritmos de classificação serão exibidas no capítulo seguinte.

O método TF-IDF baseia-se na frequência de cada termo ponderada pela frequência inversa no documento, que é usada para mensurar o quão importante um termo é em um documento (por exemplo a palavra "um" em um

conjunto de textos em português aparece várias vezes, apesar de ter uma baixa importância) [16]. Ele pode ser calculado pelas expressões:

$$TF(t) = \frac{\# \text{ de aparições de } t \text{ no documento}}{\text{total de termos no documento}} \quad (3.31a)$$

$$IDF(t) = \log \left(\frac{\text{Total de documentos}}{\# \text{ de documentos que contenham } t} \right) \quad (3.31b)$$

$$TF - IDF(t) = TF(t) * IDF(t) \quad (3.31c)$$

Abaixo será dado um exemplo de como funciona o processo de vetorização por frequências e por TF-IDF.

O corpus em questão é constituído pelos seguintes textos:

1. Eu amo cachorros
2. Eu odeio cachorros e costurar
3. Costurar é meu hobby e meu amor

Construindo o vocabulário e realizando a vetorização, obtém-se os vetores de frequência descritos na tabela abaixo, o cabeçalho da tabela representa o vocabulário.

	eu	amo	cachorros	odeio	e	costurar	é	meu	hobby	amor
Vetor 1	1	1	1	0	0	0	0	0	0	0
Vetor 2	1	0	1	1	1	1	0	0	0	0
Vetor 3	0	0	0	0	1	1	1	2	1	1

Já usando calculando o valor tf-idf para cada um, obtém-se os seguintes vetores:

	eu	amo	cachorros	odeio	e	costurar	é	meu	hobby	amor
Vetor 1	0.18	0.48	0.18	0	0	0	0	0	0	0
Vetor 2	0.18	0	0.18	0.48	0.18	0.18	0	0	0	0
Vetor 3	0	0	0	0	0.18	0.18	0.48	0.95	0.48	0.48

O procedimento da obtenção do vocabulário utilizando o BOW e vetorização dos textos, independente da medida de frequência, foi feito utilizando o módulo `feature_extraction` da biblioteca `scikit-learn`. Um exemplo de código onde é realizado todo o processo de vetorização descrito nesta subseção pode ser encontrado em <https://github.com/romaolucas/bow>.

3.5.3 Classificação manual da opinião

Com o conjunto de dados obtido em 3.5.2, podemos aplicar os algoritmos descritos no começo do capítulo. Entretanto como também mencionado no começo deste capítulo, problemas de aprendizado supervisionado exigem que o conjunto de dados seja composto não só dos dados, mas também do valor esperado para cada entrada. No caso da classificação da opinião de textos, tal opinião deve ser primeiro atribuída manualmente para que, com posse dessas opiniões, seja possível treinar os algoritmos.

Para facilitar a etapa de classificação manual, foi desenvolvido um sistema online onde cada usuário informa a quantidade de tuítes que deseja classificar e consegue classificá-los de forma mais simples. Deu-se a essa ferramenta o nome de CLAM (de Classificador Manual).

O desenvolvimento do CLAM foi motivado pela ausência de ferramentas no Brasil que permitem a colaboração nessa etapa de classificação manual (uma ferramenta comumente usada é o *Amazon Mechanical Turk*, porém na época do desenvolvimento do CLAM não estava disponível no Brasil) em conjunto com a falta de ferramentas gratuitas de uma forma geral, uma vez que a maioria das ferramentas existentes são pagas (vide o próprio *Amazon Mechanical Turk*). Por isso priorizou-se construir um código simples para que fosse fácil a modificação da base do sistema por colaboradores externos, que fosse de fácil uso para o usuário final e também de fácil hospedagem do sistema em alguma plataforma como Heroku e Firebase.

O código é de domínio público e está disponível no link: <https://github.com/romaolucas/manual-classifier-helper>


O projeto foi desenvolvido usando a linguagem Python em conjunto do *framework* web Django por já possuir diversas ferramentas integradas não só de gerenciamento do sistema, bem como modelagem do banco de dados e sistema de migração para que possa modificar os modelos de dados já existentes sem precisar realizar um acesso direto à base de dados além de possuir um conjunto de bibliotecas que facilitam o processo de importação de dados em csv e vasta quantidade de tutoriais de como desenvolver uma aplicação usando Django.

Para a modelagem de dados construiu-se dois modelos: um para tuítes e outro para as opiniões. Na modelagem, considerou-se que cada usuário irá classificar apenas textos que não possuem uma classificação, não só para evitar ter que lidar com opiniões divergentes em um texto, mas também para garantir um maior número de dados para o treinamento.

O CLAM possui o seguinte fluxo para a classificação:

1. Informa a quantidade de tuítes que se deseja classificar.

22CAPÍTULO 3. MACHINE LEARNING E ANÁLISE DE SENTIMENTOS

 Classificador Manual de Tweets (Clam) Classificar Tweets Tweets Classificados

Quantos tweets você deseja classificar?

Desenvolvido por Lucas Romão Silva.
Código fonte: <https://github.com/romaolucas/manual-classifier-helper>
Ícone feito por Freepik de www.flaticon.com licenciado por CC 3.0 BY

2. O usuário é levado a uma página com os n tuítes para classificar. Enquanto o campo da opinião é obrigatório, existe um campo optativo para informar se o tuíte é irônico ou não, uma vez que textos com ironia atrapalham o treinamento por conter palavras positivas sendo usadas de maneira negativa e vice-versa.

 Classificador Manual de Tweets (Clam) Classificar Tweets Tweets Classificados

Tweets para classificar

Texto :

de repente é porque as pessoas não estão a favor da reforma da previdência e da terceirização , coisas que o mbi defende

☐ Positivo ☐ Neutro ☐ Negativo É irônico? ☐

Texto :

Pec55, reforma da previdência , terceirização geral, reforma do EM.. tudo aprovada ou sendo aprovado e vcs falando de BBB. VÃO TOMAR NO CÚ!

☐ Positivo ☐ Neutro ☐ Negativo É irônico? ☐

Texto :

Abaixo a reforma trabalhista, reforma dá previdência , a terceirização | pic.twitter.com/uW92izT7g8

Desenvolvido por Lucas Romão Silva.
Código fonte: <https://github.com/romaolucas/manual-classifier-helper>
Ícone feito por Freepik de www.flaticon.com licenciado por CC 3.0 BY

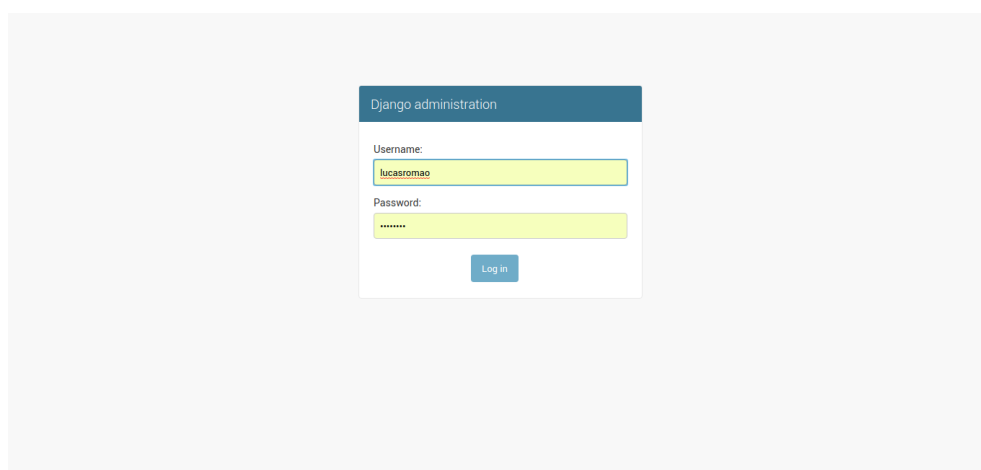
3. Uma vez classificados, o usuário é levado a uma página onde é possível ver todos os tuítes já classificados e também gerar um arquivo csv contendo todos aqueles que não foram marcados como irônicos.

Classificador Manual de Tweets (Clam)	
Exportar CSV com os tweets já classificados	
Username	Texto
BastidoresMidia	URGENTE. O BRASIL VAI PARAR CONTRA A TERCEIRIZAÇÃO , REFORMA DA PREVIDÊNCIA , REFORMA TRABALHISTA E MALDADES DO... http://fb.me/1Kt5Smg07
josemazfilho	Vigarista lulopetista -anunciou saída do PT enojado com a corrupção, mas continua lá- Paim é contra terceirização , reforma da previdência ...
VigaRodrigo	Terceirização , reforma da previdência , reforma trabalhista. Quando empresários e governo vão ajudar a dividir a conta ???
apys	Só de contratos da Odebrecht, Lula teria sido o destino de mais de R\$ 200 milhões em valores indevidos https://buff.ly/2x1Ltwj
androlett	Transposição do rio São Francisco, reforma do ensino médio/ da previdência , operação carne fraca, terceirização ... Enem vai bombar esse ano
cristina_diniz9	@CamaraDeputados Este maldito governo acaba em 18 meses. E quem votar a favor da terceirização e reforma da previdência não volta
alexmunds	Você ainda tem coragem de falar, Ameba Neves???
OsvladPereira	Candidato rejeitado por mais de 50% do eleitor tem tanta chance de vencer quanto Dilma Rousseff de fabricar uma frase com começo, meio e fim pic.twitter.com/hSw9qgtDn
karolfarias_	o povo tá nas ruas lutando contra a reforma da previdência e a terceirização . o que temer faz? >> sanciona a... http://fb.me/1U9eabRXD
josesaoleopoldo	Paralisação geral dia 31 de março!! Contra a lei da terceirização , contra a reforma da ... http://fb.me/6bMgimlgG
JosLuizBezerra1	Dilma passadena vai te colocar na cadeia. Vc tem muito mas dinheiro escondido que Gedel....
Desenvolvido por Lucas Romão Silva. Código fonte: https://github.com/romao/lucas/manual-classifier-helper Ícone feito por Freepik de www.flaticon.com licenciado por CC 3.0 BY	

Para um usuário que deseja hospedar a plataforma e usá-la para si, existe também o admin onde é possível importar tuítes para serem classificados.

Abaixo é descrito o fluxo para a importação de novos dados no admin.

1. Realiza o login no sistema utilizando seu usuário e senha de administrador.



2. Seleciona a parte de Tweets na tela principal.

24CAPÍTULO 3. MACHINE LEARNING E ANÁLISE DE SENTIMENTOS

Django administration

WELCOME: **LUCASROMAO** VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups	Add	Change
Users	Add	Change

CLASSIFICATION

Reviews	Add	Change
Tweets	Add	Change

Recent actions

My actions

None available

- Uma vez na parte de visualizar todos os tuítes já armazenados na base de dados, seleciona a opção de importar no canto superior direito.

Django administration

WELCOME: **LUCASROMAO** VIEW SITE / CHANGE PASSWORD / LOG OUT

Home / Classification / Tweets

Select tweet to change

IMPORT EXPORT ADD TWEET +

Action: Go 0 of 100 selected

TWEET

☐ VIVA A TERCEIRIZAÇÃO ! VIVA A REFORMA DA PREVIDÊNCIA ! SE A DILMA E O LULA APROVAM ENTÃO É PORQUE É BOM! <http://fb.me/7DaUPpckg>

☐ O seu Lula -lixo está é com o fiofó na mão! kkkkkkkkkk <https://noticias.uol.com.br/politica/ultim-as-noticias/2017/09/21/lula-ficou-preocupado-com-fala-sobre-intervencao-militar-dizem-petistas.htm> ...

☐ Delator do caso do Instituto Lula nega autoria de bilhete <https://folha.com/no1920325>

☐ Retweeted Jose de Abreu (@zehdeabreu): "Se a Justiça não tirar Lula do páreo nós tiraremos." Sargento Garcia... <http://fb.me/8KBv7VDwB>

☐ Falou do Aécio finalmente essa semana! Pensei que tinha esquecido do Aécio Neves . pic.twitter.com/NP3AmonuW1

☐ Hehehe que enterrou Dilma <https://twitter.com/otempo/status/910848222851211264> ...

☐ Impactos da terceirização , reforma trabalhista e da previdência foram debatidos em audiência pública. <http://casadeleis.blogspot.com.br/2017/04/impact-os-da-terceirizacao-reforma.html> ... pic.twitter.com/AdiIMeyhlp

☐ Cada pais tem a Dilma que merece <https://twitter.com/FoxNews/status/908672110679121920> ...

☐ LULA E DILMA NÃO SERÃO PRESOS. SE PRENDEREM GILMAR MENDES MANDA SOLTAR.

☐ Paralisação geral dia 31 de março!!! Contra a lei da terceirização , contra a reforma da ... <http://fb.me/6S30aROCX>

☐ País tem protestos contra reforma da Previdência e terceirização | Brasil 24/7 <http://www.brasil247.com/pt/247/brasil/288053/Pa%C3%ADs-tem-protestos-contra-reforma-da-Previd%C3%Aancia-e-terceiriza%C3%A7%C3%A3o.htm> ... via -- @brasil247

- Na página que segue, basta informar um arquivo .csv, .xls (formato do excel) ou .json contendo nessa ordem: o id do tuíte (fornecido pelo Twitter), usuário que escreveu, texto, espaço vazio para representar o id que será preenchido automaticamente na hora de importar no banco de dados.

Django administration

WELCOME LUCASROMAO | [VIEW SITE](#) | [CHANGE PASSWORD](#) | [LOG OUT](#)

[Home](#) > [Classification](#) > [Tweets](#) > [Import](#)

Import

This importer will import the following fields: tweet_id, tweet_username, tweet_text, id

File to import: [Escolher arquivo](#) | Nenhum arquivo selecionado

Format: ---

[SUBMIT](#)

5. Caso todos os dados sejam fornecidos corretamente, será passado a uma nova página para confirmar se deseja importar e, por fim, será redirecionado à página que contém todos os tuítes.

No apêndice [A](#) é descrito em detalhes como instalar e realizar as configurações iniciais do CLAM.

Capítulo 4

Experimentos

Uma vez implementado os modelos de classificação, realizou-se experimentos sobre os tuítes classificados manualmente a fim de encontrar o melhor classificador. Além disso, decidiu-se comparar os resultados dos classificadores implementados neste trabalho com os já disponíveis na biblioteca `scikit` do Python para verificar qual obteria melhores resultados ou se não haveria diferença.

Para todos os experimentos, variou-se diversos parâmetros (que serão descritos em 4.2.1) a fim de encontrar a melhor combinação de parâmetros para cada classificador. Essa variação foi feita utilizando a função `logspace` da biblioteca `numpy` que retorna valores igualmente espaçados (numa escala logarítmica) dentro do intervalo especificado.

4.1 Dataset

4.1.1 Distribuição das classes

Antes de mais nada, será a composição do conjunto de dados e como se dá a distribuição de classes. Na figura 4.1 tem-se um histograma das classes associadas a cada tuíte.

Classe	Total
Positivo	115
Negativo	1248
Neutro	1223

Tabela 4.1: Quantidade de tuítes pertencentes a cada classe

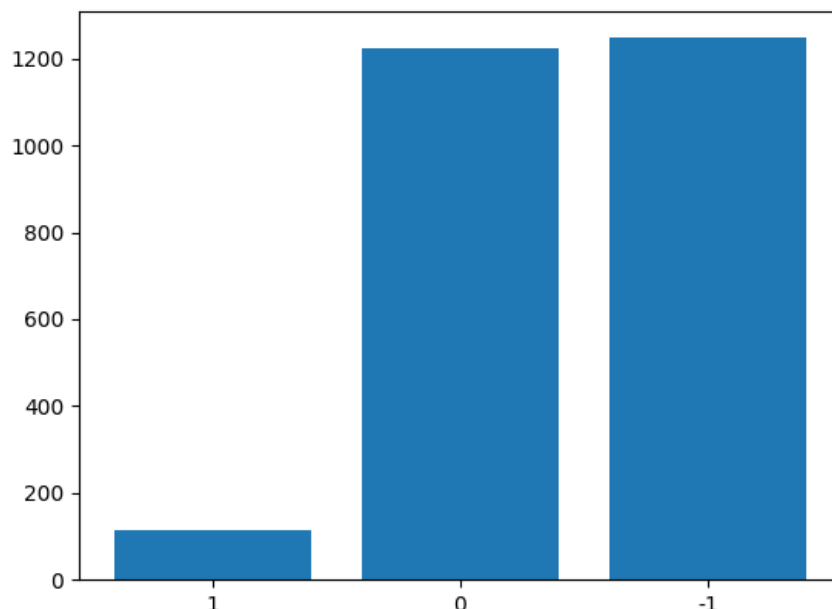


Figura 4.1: Histograma da frequência de tuítes

Pode-se observar nas figura 4.1 e tabela 4.1 que , ao passo em que as classes negativa e neutra possuem distribuição próxima (48,26% e 47,29% respectivamente), a classe positiva corresponde a apenas 4,45% de todos os tuítes. No final da seção 4.3 será visto que esse desbalanceamento entre as classes acarretará um desempenho ruim dos classificadores, sejam os desenvolvidos para este trabalho quanto os do `scipy`.

4.1.2 Características dos tuítes de cada classe

A fim de entender como os classificadores realizariam o treinamento, analisou-se o conjunto de dados a procura de padrões de características para cada classe.

Para os tuítes avaliados como neutro, notou-se o padrão que são tuítes derivados de portais de notícias ou são indagações, porém sem nenhuma crítica feita nelas. A tabela abaixo mostra como exemplo alguns tuítes avaliados como neutro.

usuário	tuíte
Tica_Fernandes	Cidades têm protestos contra reforma da Previdência e terceirização http://g1.globo.com/politica/noticia/cidades-tem-protestos-contr-reforma-da-previdencia-e-terceirizacao.ghml
andrcolett	Transposição do rio São Francisco, reforma do ensino médio/da previdência , operação carne fraca, terceirização ... Enem vai bombar esse ano
VictoorAugustoo	Via @estadao : Manifestantes protestam em capitais do País contra reforma da Previdência e terceirização - http://ln.is/estadão.com.brb29w1
EdmilsonPequeno	O meu deputado @luizcoutopt votou contra a terceirização e é contra a reforma da previdência

Já para os tuítes negativos, nota-se a presença de palavrões e xingamentos junto de *hashtags* de teor negativo como por exemplo #ForaTemer, além disso é comum a presença de tuítes onde as pessoas são convocadas para manifestações. Outros termos comuns são escravidão e retrocesso, referindo-se diretamente às reformas da previdência e terceirização.

usuário	tuíte
MgracaGalvao	Dória é o maior Fake de todos os tempos. Se f... Palhaço
adamastaquio	Acorda POVO trabalhador. Previdencia + Terceirização + Reforma da CLT é P*** NO RABO DO POVO! #Reforma-TrabalhistaNÃO
TheMairaBastos	Reforma da previdência , desmonte da CLT, terceirização ,congelamento de gastos com a saúde e educação #ForaTemerLadrao #TemerGolpista
neyeverest	O PT esteve no poder por 14 anos o bandido do Lula e a burra da Dilma e o país está um caos pela instituição a roubalheira deles também !!
carinasotero	GREVE GERAL DIA 28/04 CONTRA RETROCESSOS DE TEMER • Reforma da Previdência • Reforma Trabalhista • Terceirização irrestrita

Assim como para os tuítes classificados como negativo, os tuítes positivos também são caracterizados pela presença de *hashtags*, sejam elas declarando apoio a um candidato (como #Aecio2018 ou #Lula2018) ou simplesmente manifestando alguma opinião positiva.

usuário	tuíte
Verinha_Lu	É Aécio pelo Brasil !!! #AécioPresidenteDoBrasil2018 #EstamosComAécio #DeusÉmaior e #VitóriaVem #FÉ #SouAécio
Haddad_Fernando	O salário mínimo aumentou 71% durante os governos Lula e Dilma #BrasilQueOPovoQuer
rovisacro	A favor da terceirização e tmb da reforma da previdência ! Mas óbvio, de forma justa e igualitária para todos, principalmente na previdência

4.2 Descrição dos experimentos

Realizou-se duas etapas de experimentos, uma primeira etapa utilizando todas as classes onde se observou que a baixa quantidade de elementos da classe positiva acabava por atrapalhar o desempenho geral dos algoritmos, em especial os que utilizavam a abordagem OVA para o caso de multiclases. Com isso, decidiu-se descartar esses elementos e rodar uma nova sequência de experimentos considerando apenas as classes negativa e neutra (que para efeito dos algoritmos será a classe positiva) e, com isso, verificar a melhoria dos algoritmos nas métricas.

Para ambas as etapas, procurou-se não só medir o desempenho dos algoritmos, mas também encontrar uma escolha certa de parâmetros para os mesmos que obtivesse a melhor performance (como a escolha do Kernel no caso do SVM ou a constante de regularização usada tanto no SVM quanto na regressão logística).

Tais testes de escolha de parâmetro, caso feitos usando apenas a divisão do conjunto de dados entre conjunto de treino e conjunto de testes acabaria por dar uma escolha enviesada de parâmetros, uma vez que uma escolha de parâmetros que tenha bons números em determinado conjunto de testes não indica que ele possui uma boa generalização, isto é, apresentará uma boa precisão para novos dados.

A fim de garantir maior generalização, utiliza-se um método de validação chamado de validação cruzada, em particular a forma k-fold foi usada.

Nessa forma de validação cruzada, o conjunto de dados é dividido em conjunto de treino e de teste. O conjunto de treino, por sua vez, é dividido em k partições de igual tamanho sendo $k - 1$ delas usadas efetivamente para treinar o modelo e a partição restante é usada para validá-lo. Esse procedimento de dividir o conjunto de treino em k partições e separar entre partição de treino e validação é realizado k vezes e, por último, a escolha de

modelo que apresentou melhor desempenho é escolhida. A figura 4.2 mostra o funcionamento do método[17].

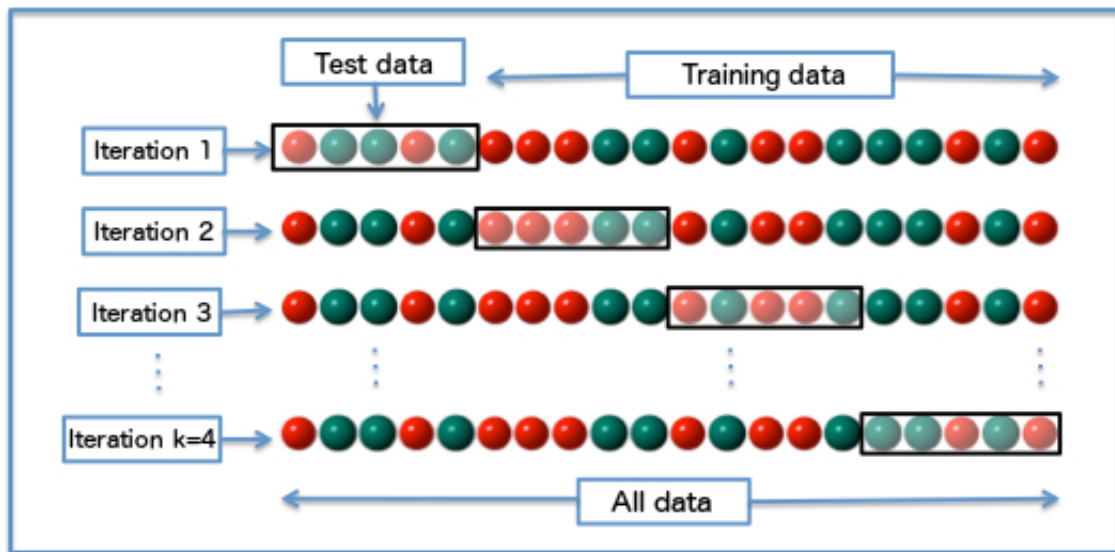


Figura 4.2: Funcionamento do método k-fold da validação cruzada

fonte: [https://en.wikipedia.org/wiki/Cross-validation_\(statistics\)#/media/File:K-fold_cross_validation_EN.jpg](https://en.wikipedia.org/wiki/Cross-validation_(statistics)#/media/File:K-fold_cross_validation_EN.jpg)

Outro procedimento que foi analisado nos experimentos é se as diferentes escolhas de vetorização afetam as métricas (ver 4.2.2), todavia esse procedimento de vetorização é feito antes do treinamento dos modelos e portanto não foi variado no procedimento de Cross-Validation. Em 4.3 e 4.4 os resultados são separados por vetorização e, ao fim de cada rodada, explica-se os resultados observados.

O fluxo da realização dos experimentos é descrita no diagrama abaixo:

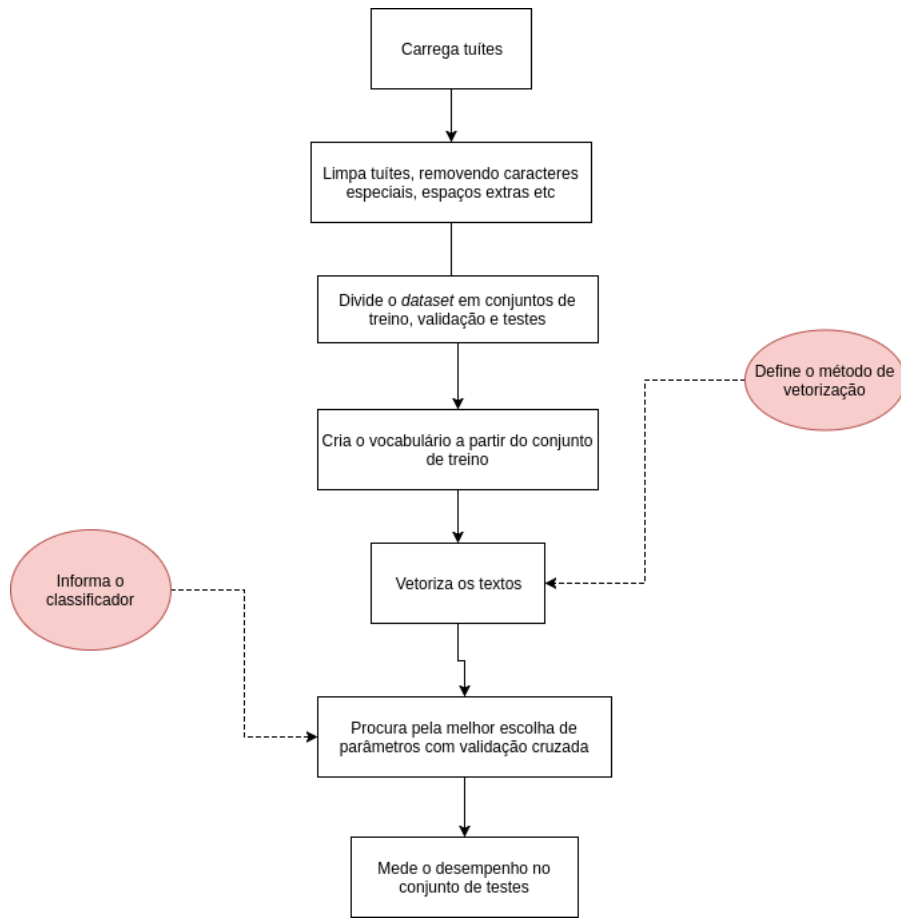


Figura 4.3: Passos dos experimentos

4.2.1 Parâmetros avaliados

SVM

Para o SVM, testou-se o desempenho do algoritmo com diferentes kernels e com diferentes parâmetros para cada um. Os seguintes kernels foram utilizados:

- Kernel Linear: $K(x, y) = x^T y$
- Kernel RBF: $K(x, y) = \exp(-\gamma ||x - y||^2)$
- Kernel Polinomial: $K(x, y) = (x^T y + c)^d$

Em comum a todos os kernels, variou-se o parâmetro C do SVM utilizado para penalizar os valores mal classificados. Já para o RBF, varia-se o

parâmetro γ . Por último para o polinomial, varia-se tanto o parâmetro d que determina o grau do polinômio (aqui variou-se entre 3, 4 e 5) e o coeficiente.

Regressão Logística

Para a regressão logística, variou-se o parâmetro de penalização λ responsável por evitar *overfitting*. Mais detalhes sobre o funcionamento da regressão logística com penalização pode ser encontrado em 3.3.4.

4.2.2 Métricas avaliadas

Para a realização dos experimentos, levou-se em consideração quatro métricas para avaliar a eficiência dos classificadores.

- Acurácia: taxa calculada pela quantidade de dados corretamente classificados sobre o total de dados. Quanto mais próximo de 1, melhor.
- Precisão: taxa calculada pela expressão $\frac{tp}{tp+fp}$ onde tp corresponde aos verdadeiro positivos, isto é, elementos corretamente classificados como da classe C e fp são elementos erroneamente classificados na classe C . Uma classe ter precisão alta significa que possui um baixo número de amostras classificadas erroneamente nela.
- Revocação: taxa calculada pela expressão $\frac{tp}{tp+fn}$ onde tp corresponde aos verdadeiro positivos, como na precisão e fn corresponde aos falso negativos, isto é, os elementos incorretamente classificados como não pertencentes a uma classe C quando na verdade pertencem. Uma alta revocação indica que muitos elementos de determinada classe foram corretamente classificados como pertencentes a ela.
- Pontuação F1: pode ser interpretado como uma média harmônica entre precisão e revocação é dado pela expressão: $2 * \frac{precis*revocao}{precis+revocao}$. Assim como as demais medidas, quanto mais próximo de 1, melhor o valor.

4.3 Primeira rodada de experimentos

Nesse primeiro experimento, avaliou-se o desempenho dos classificadores desenvolvidos quanto dos já disponíveis pela biblioteca `scikit` a fim de comparar o desempenho das implementações. Além disso, testou-se diferentes escolhas de parâmetro para cada abordagem de vetorizar o corpus que no caso foram vetor binário, vetor de frequência e vetor com os valores TF-IDF conforme descrito em 3.5.2. Note que neste primeiro experimento, todos os dados foram tratados *in-natura*, sem nenhum método de seleção de características ou qualquer abordagem semelhante a fim de melhorar o desempenho.

4.3.1 Implementações próprias

Dos algoritmos desenvolvidos neste trabalho, testou-se o SVM variando o valor de C entre valores do intervalo $[10^{-2}, 10^5]$, kernels polinomial, RBF e linear. Para o kernel polinomial foi utilizado polinômios de grau 3, 4 e 5 e coeficientes 1, 10 e 100. Para o kernel RBF testou valores de γ dentro do intervalo $[10^{-9}, 10^3]$. Para cada escolha de vetorização, será colocado apenas os melhores resultados para cada kernel.

Frequência

Tabela 4.2: Resultados para o SVM

Kernel	C	parâmetros	Acurácia média
Linear	0,1		0,678
Linear	1		0,651
RBF	1	$\gamma = 0,1$	0,683
RBF	10^5	$\gamma = 4,64 * 10^{-7}$	0,678

A escolha de parâmetros que obteve maior classificação foi o kernel RBF com $C = 1$ e $\gamma = 0,1$ com acurácia média de 68,3%. Usando o classificador com estes parâmetros no conjunto de testes obteve-se o seguinte resultado:

classe	precisão	revocação	f1score	support
-1	0.65	0.66	0.66	306
0	0.64	0.69	0.67	311
1	1.00	0.03	0.06	30
média / total	0.67	0.65	0.64	647

Vetor binário

Tabela 4.3: Resultados para o SVM

Kernel	C	parâmetros	Acurácia média
Linear	0,146		0,675
Linear	0,563		0,658
RBF	2,154	$\gamma = 0,1$	0,677
RBF	464,16	$\gamma = 4,64 * 10^{-7}$	0,676
Polinomial	0,01	$c = 1, d = 3$	0,641
Polinomial	0,01	$c = 1, d = 3$	0,642

Como melhor escolha de parâmetros, tivemos o kernel RBF com $C =$

464,16 e $\gamma = 0,1$ com 67,7% de acurácia. Utilizando os parâmetros no conjunto de testes, obteve-se os seguintes resultados:

classe	precisão	revocação	f1score	support
-1	0.69	0.72	0.70	307
0	0.70	0.72	0.71	315
1	1.00	0.08	0.15	25
média / total	0.71	0.70	0.69	647

TF-IDF

Tabela 4.4: Resultados para o SVM

Kernel	C	parâmetros	Acurácia média
Linear	0,146		0,651
Linear	0,563		0,664
RBF	1778,279	$\gamma = 2,15 * 10^{-4}$	0,667
RBF	26101,572	$\gamma = 10^{-5}$	0,666
Polinomial	0,038	$c = 1, d = 4$	0,660
Polinomial	0,147	$c = 1, d = 3$	0,663

Assim como as demais escolhas de vetorização, a melhor escolha de kernel foi o RBF. Com valor de $C = 1778,279$ e $\gamma = 2,15 * 10^{-4}$ conseguiu-se acurácia média de 66,7%. Utilizando esse classificador no conjunto de testes, obtiveram-se os seguintes resultados:

classe	precisão	revocação	f1score	support
-1	0.68	0.73	0.70	322
0	0.67	0.69	0.68	296
1	1.00	0.03	0.07	29
média / total	0.69	0.68	0.66	647

Pelos resultados dos experimentos, viu-se que a melhor escolha de classificador e vetorização foi o kernel RBF com vetorização por frequência com 68,3% de acurácia. Ao observar as outras métricas, percebe-se que os valores médios para precisão, revocação e pontuação f1 estão razoáveis, entretanto ao olhar para os valores de revocação e pontuação f1 da classe positiva, vê-se que estes são quase 0 ao passo que a precisão é 1. Isso indica que apenas poucos elementos foram colocados nesta classe corretamente e nenhum foi colocado incorretamente, porém a maioria dos elementos desta classe foram marcados nas demais classes.

Tais valores para classe positiva são devidos à baixa quantidade de dados da mesma, o que acaba prejudicando o desempenho do classificador que separa esta classe das demais.

Utilizou-se o melhor classificador para verificar alguns tuítes mal classificados e entender melhor o funcionamento do classificador.

Tuíte	Previsto	Real
O cara tá falando da terceirização anta. Eu sou contra essa reforma da previdência deste jeito, ela deve ser revista.A trabalhista eu apoio	-1	1
Ops @MichelTemer não tem um papel com os trabalhadores e a lei da terceirização assim fala os especialistas imagina essa REFORMA PREVIDÊNCIA http://pic.twitter.com/KjLa26TCwy	-1	0
A agenda do Lula para os próximos meses está lotada... de depoimentos em processos em que ele é réu. http://owl.li/F3Ij30flUaY	0	-1
Contra reforma da Previdência e terceirização ... http://fb.me/18guG6FGR	0	-1
Não podemos discutir a Reforma da Previdência sem discutir a Terceirização e outros elementos desse desmonte de direitos", Ruy (APLB)"	-1	0
Muso inspirador da presidenta Dilma !	0	1

Para os tuítes negativos que foram erroneamente classificados, percebe-se que o classificador os colocou na classe neutra pela maior presença de palavras encontradas em tuítes de notícias, no segundo caso, há uma ambiguidade no tuíte e, enquanto foi classificado com negativo, poderia ter sido também classificado como neutro por outra pessoa.

Na classe neutra, tem-se que em um há uma ambiguidade no tuíte e não possui informação suficiente para avaliar sua polaridade, porém no segundo caso pode-se dizer que o tuíte foi classificado corretamente pois o classificador levou em conta que chamou a reforma de previdência e a terceirização de desmonte de direitos, apesar de o tuíte em si apenas relatar o que foi dito.

Para a classe positiva notou-se dois casos: em um a presença de um xingamento colocou o tuíte como opinião negativa e no outro caso o classificador interpretou o tuíte como uma simples constatação de fatos.

4.3.2 Implementações do `scikit`

Assim como para as implementações próprias, será separado cada teste por forma de vetorização para verificar qual escolha de parâmetros é melhor em cada. Ao fim será comentado os resultados gerais deste experimento.

Frequência

Tabela 4.5: Resultados para o SVM

Kernel	C	parâmetros	Acurácia média
Linear	0,1		0,666
Linear	1		0,637
RBF	100	$\gamma = 10^{-3}$	0,663
RBF	100	$\gamma = \frac{1}{\#amostras}$	0,653
Polinomial	0,01	$c = 10, d = 5$	0,666
Polinomial	100	$c = 1, d = 4$	0,663

Importante ressaltar que em casos de empate, o primeiro melhor será selecionado, no caso o melhor foi o kernel linear com $C = 0.1$, utilizando esses valores no conjunto de testes obtiveram-se os seguintes resultados:

classe	precisão	revocação	f1score	support
-1	0.70	0.73	0.71	324
0	0.67	0.69	0.68	298
1	1.00	0.04	0.08	25
média / total	0.70	0.68	0.67	647

Para a regressão logística, variou-se o valor de λ entre 10^{-4} a 1 e, além disso variou-se a abordagem para o caso multiclases, uma vez que o `scikit` disponibiliza uma implementação usando o método OVA e outro usando o método multinomial (assim como implementou-se).

Tabela 4.6: Resultados da regressão logística

Método	λ	Acurácia média
OVA	0.1	0.654
OVA	1	0.651
Multinomial	0.1	0.654
Multinomial	1	0.649

Assim como para o SVM, escolheu-se o primeiro melhor que no caso é a abordagem OVA com $\lambda = 0.1$. Deu os seguintes resultados no conjunto de testes:

classe	precisão	revocação	f1score	support
-1	0.70	0.68	0.69	320
0	0.68	0.73	0.70	307
1	0.50	0.15	0.23	20
média / total	0.68	0.69	0.68	647

Vetor binário

Tabela 4.7: Resultados para o SVM

Kernel	C	parâmetros	Acurácia média
Linear	0,1		0,658
Linear	1		0,643
RBF	100	$\gamma = 10^{-3}$	0,649
RBF	100	$\gamma = \frac{1}{\#amostras}$	0,646
Polinomial	100	$c = 1, d = 4$	0,656
Polinomial	1	$c = 5, d = 4$	0,658

Assim como para o vetor de frequências, a melhor escolha de parâmetros encontrada na validação cruzada foi $C = 0, 1$ e kernel linear. Executando no conjunto de testes obtiveram-se os seguintes resultados:

classe	precisão	revocação	f1score	support
-1	0.66	0.76	0.71	310
0	0.69	0.66	0.67	302
1	1.00	0.06	0.11	35
média / total	0.69	0.68	0.66	647

Tabela 4.8: Resultados da regressão logística

Método	λ	Acurácia média
OVA	0.1	0.631
OVA	1	0.634
Multinomial	0.1	0.631
Multinomial	1	0.630

Tal qual com o vetor de frequências, escolheu-se a abordagem OVA e $\lambda = 1$ obtendo os seguintes resultados no conjunto de testes:

classe	precisão	revocação	f1score	support
-1	0.68	0.73	0.71	305
0	0.70	0.69	0.69	317
1	0.33	0.04	0.07	25
média / total	0.67	0.69	0.68	647

TF-IDF

Tabela 4.9: Resultados para o SVM

Kernel	C	parâmetros	Acurácia média
Linear	1		0,658
Linear	10		0,627
RBF	100	$\gamma = 10^{-3}$	0,651
RBF	1	$\gamma = 10^{-4}$	0,483
Polinomial	1	$c = 5, d = 5$	0,678
Polinomial	10	$c = 10, d = 3$	0,677

Com essa vetorização, diferente das demais, os melhores parâmetros escolhidos foram kernel polinomial com coeficiente e grau 5. Utilizando esses valores, obtiveram-se o seguinte resultado no conjunto de testes:

classe	precisão	revocação	f1score	support
-1	0.64	0.79	0.71	311
0	0.69	0.60	0.64	301
1	0.00	0.00	0.00	35
média / total	0.63	0.66	0.64	647

Tabela 4.10: Métricas para o melhor classificador do `scikit` no caso de três classes

Tabela 4.11: Resultados da regressão logística

Método	λ	Acurácia média
OVA	0.1	0.659
OVA	1	0.665
Multinomial	0.1	0.660
Multinomial	1	0.662

Neste teste, assim como os demais usando a regressão logística, escolheu-se a abordagem OVA com $\lambda = 1$ obtendo seguintes resultados no conjunto de testes:

classe	precisão	revocação	f1score	support
-1	0.66	0.75	0.70	312
0	0.69	0.66	0.68	310
1	0.00	0.00	0.00	25
média / total	0.65	0.68	0.66	647

Observa-se que, para ambos os algoritmos tem-se que nenhum valor foi classificado como positivo quando usou-se a vetorização através do valor TF-IDF de cada termo, apesar de ainda apresentar desempenho médio aceitável.

Isso se deve ao fato de vetorização por TF-IDF leva-se em consideração a relevância das palavras na sentença como um todo e, possivelmente algumas palavras associadas a documentos positivos estão também associadas a documentos neutros e, conseqüentemente, não possuem um valor TF-IDF alto, aliado ao fato de o vocabulário ser montado levando em conta apenas as 5000 palavras com melhores pontuações ou presença, portanto algumas palavras explicitamente associadas a documentos positivos não compuseram o vetor de *features*.

Por outro lado se olharmos os desempenhos utilizando frequências simples ou um vetor binário obtemos resultados semelhantes para ambos os algoritmos, todavia ainda existe um melhor desempenho com a vetorização por frequências tal como o SVM apresenta melhores resultados em relação à regressão logística, mesmo que com pouca diferença.

Assim como o outro classificador, analisou-se os tuítes classificados incorretamente a fim de entender o funcionamento do classificador:

Tuíte	Previsto	Real
Lula ainda lidera as pesquisas oficiais em todo o Brasil	-1	0
Sem querer ser chata, mas foi a Dilma quem começou com os Projetos de Reforma da Previdência e de Terceirização	-1	0
simmm, e irei, quero a reforma da previdência , a terceirização , reforma da CLT, tudo isso p o Brasil voltar a crescer :)	-1	1
O cara tá falando da terceirização anta. Eu sou contra essa reforma da previdência deste jeito, ela deve ser revista.A trabalhista eu apoio	-1	1
Temer não é santo,mas não entender o conceito que deu uma freada na beira do abismo que Lula Dilma estavam nos empurrando é DESINFORMAÇÃOhttps://twitter.com/Acppprovesi/status	0	-1
Texto: Os alunos da FDCE se posicionam CONTRA a reforma da previdência , trabalhista e a terceirização ! Quem é de luta e... http://fb.me/1YA1jro5z	0	-1

Observa-se na tabela, assim como para as implementações próprias, houve a classificação errada de tuítes positivos como negativos devida à presença de xingamentos no tuíte. Além disso tuítes negativos costumam ser classificados erroneamente como neutros pelo texto se assemelhar com uma constatação de um fato, mesmo que possua a presença de termos negativos.

Comum a todos os experimentos foram resultados ruins associados à classe positiva. No caso do SVM obtiveram-se alta precisão em alguns casos, porém as taxas de revocação e f1-score foram baixas. Valores baixos para revocação e f1-score podem ser interpretados como um sinal de que os classificadores são ruins para detectar opiniões positivas. A exemplo do melhor classificador, tem-se uma taxa de revocação de 0,08 que indica que 92% das amostras positivas não são detectadas pelo algoritmo.

Motivado pelo baixo desempenho visto na classe positiva em todos os experimentos aliado à baixa quantidade de tuítes nessa classe, rodou-se mais experimentos, porém dessa vez usando apenas as outras classes.

4.4 Segunda rodada de experimentos

Nesta segunda rodada de experimentos, removeu-se os elementos pertencentes à classe positiva. Assim como na primeira rodada, testou-se diferentes escolhas de parâmetros para cada um dos algoritmos e testou-se com os dados *in natura*. Como viu-se na primeira rodada de experimentos que havia pouca

diferença entre a vetorização por frequência e presença do termo, optou-se por testar apenas usando frequência e TF-IDF.

4.4.1 Implementações próprias

Para as implementações próprias do SVM, testou-se os kernels polinomial, linear e RBF alternando diversos parâmetros destes. Assim como na rodada anterior, só será mostrado aqui os melhores resultados obtidos.

Frequência

Tabela 4.12: Resultados para o SVM

Kernel	C	parâmetros	Acurácia média
Linear	0,1		0,679
Linear	10^{10}		0,667
RBF	1	$\gamma = 0,1$	0,698
RBF	10	$\gamma = 0,1$	0,697
Polinomial	0,1	$c = 10^{-4}, d = 3$	0,679
Polinomial	100	$c = 10^{-4}, d = 5$	0,679

Nesta rodada, escolheu-se usar o kernel RBF com $\gamma = 0,1$ e $C = 1$ que obteve 69,8% de acurácia média. No conjunto de testes teve os seguintes resultados:

classe	precisão	revocação	f1-score	support
-1	0.73	0.62	0.67	343
0	0.60	0.71	0.65	275
média / total	0.67	0.66	0.66	618

Tabela 4.13: Resultados da regressão logística

λ	Acurácia média
10^{-5}	0,637
0,316	0,684
10^4	0,544

Utilizando então $\lambda = 0,316$ foi obtido 68,4% de acurácia média e, ao utilizar o classificador no conjunto de testes teve os seguintes resultados:

classe	precisão	revocação	f1-score	support
-1	0.68	0.69	0.68	301
0	0.70	0.69	0.69	317
média / total	0.69	0.69	0.69	618

TF-IDF

Tabela 4.14: Resultados para o SVM

Kernel	C	parâmetros	Acurácia média
Linear	0,1		0,688
Linear	1		0,697
RBF	1	$\gamma = 1$	0,704
RBF	10^9	$\gamma = 1$	0,700
Polinomial	1	$c = 10^{-4}, d = 3$	0,697
Polinomial	1	$c = 10^{-4}, d = 5$	0,697

Os melhores parâmetros para esta rodada de testes foi o kernel RBF com γ e C valendo 1 que obteve acurácia média de 70,4%. Utilizando estes parâmetros no conjunto de testes obtiveram-se os seguintes valores:

classe	precisão	revocação	f1-score	support
-1	0.69	0.79	0.74	305
0	0.76	0.65	0.71	313
média / total	0.73	0.72	0.72	618

Tabela 4.15: Resultados da regressão logística

λ	Acurácia média
10^{-5}	0,662
0,316	0,697
10^4	0,585

Assim como na rodada anterior, o melhor parâmetro foi $\lambda = 0,316$, porém desta vez obteve 69,7% de acurácia. Utilizando este parâmetro de regularização no conjunto de testes obtiveram-se os seguintes resultados:

classe	precisão	revocação	f1-score	support
-1	0.70	0.68	0.69	323
0	0.66	0.67	0.67	295
média / total	0.68	0.68	0.68	618

Por fim da rodada de testes com as implementações próprias dos classificadores, chegou a conclusão de que o algoritmo SVM em conjunto com a vetorização por TF-IDF trouxe um bom compromisso entre acurácia, precisão e revocação tendo todas as suas médias acima de 70% para todas as métricas sendo assim um classificador com alta probabilidade de trazer resultados relevantes para novos documentos. Além disso o procedimento de validação cruzada garante que o classificador possui capacidade de generalização, uma vez que realiza o treinamento e validação com conjuntos diferentes mais de uma vez.

Interessante comentar que ao passo que a regressão logística não chegou a ter acurácia melhor em nenhuma vetorização, porém ao olharmos para os valores médios de precisão, revocação e pontuação f1 vemos que, com exceção da melhor versão do SVM, foi superior indicando que este algoritmo possui robustez ao trazer valores relevantes.

4.4.2 Implementações do scikit

Para as implementações do `scikit`, realizou-se os experimentos da mesma forma que para o caso com três classes com a única diferença sendo os valores para γ ao qual testou o kernel RBF.

Frequência

Tabela 4.16: Resultados para o SVM

Kernel	C	parâmetros	Acurácia média
Linear	0,1		0,699
Linear	1		0,680
RBF	1	$\gamma = 0,158$	0,713
RBF	10	$\gamma = 0,158$	0,701
Polinomial	1	$c = 5, d = 4$	0,698
Polinomial	100	$c = 1, d = 4$	0,697

Como melhor resultado teve-se o kernel RBF com $\gamma = 0,158$ e $C = 1$ com acurácia média de 71,3%. Utilizando essa escolha de parâmetros no conjunto de testes, obtiveram-se os seguintes resultados:

classe	precisão	revocação	f1-score	support
-1	0.69	0.63	0.66	327
0	0.62	0.68	0.65	291
média / total	0.66	0.66	0.66	618

Tabela 4.17: Resultados da regressão logística

λ	Acurácia média
0,1	0,693
1	0,696
1000	0,676

Como resultado, obtivemos que o melhor valor para λ era 1 que nos deu uma acurácia média de 69,6% e para o conjunto de testes deu os seguintes resultados:

classe	precisão	revocação	f1-score	support
-1	0.65	0.68	0.66	300
0	0.68	0.65	0.67	318
média / total	0.66	0.66	0.66	618

TF-IDF

Tabela 4.18: Resultados para o SVM

Kernel	C	parâmetros	Acurácia média
Linear	1		0,702
Linear	10		0,676
RBF	1	$\gamma = 0,158$	0,710
RBF	100	$\gamma = 0,00398$	0,702
Polinomial	0,1	$c = 10, d = 5$	0,701
Polinomial	10	$c = 10, d = 4$	0,702

Como melhor escolha de parâmetros tivemos o kernel RBF com $\gamma = 0,158$ e $C = 1$ com 71% de acurácia média. Testando no conjunto de testes obtiveram-se os seguintes resultados:

classe	precisão	revocação	f1-score	support
-1	0.68	0.75	0.71	326
0	0.68	0.61	0.65	292
média / total	0.68	0.68	0.68	618

Tabela 4.19: Resultados da regressão logística

λ	Acurácia média
0,1	0,686
1	0,688
10	0,671

Com isso temos que a melhor escolha para λ é 1 que obteve 68,8% de acurácia média. Utilizando este parâmetro para classificar o conjunto de testes, obtiveram-se os seguintes resultados:

classe	precisão	revocação	f1-score	support
-1	0.71	0.77	0.74	313
0	0.74	0.68	0.71	305
média / total	0.73	0.72	0.72	618

As implementações do `scikit` apresentaram uma acurácia maior em relação às desenvolvidas, apesar de a diferença não ser expressiva. Interessante notar que, ao contrário das implementações próprias, o melhor classificador, no caso o SVM com kernel RBF e vetorização por frequência não obtiveram os melhores valores para todas as métricas avaliadas, uma vez que o melhor classificador em termos de precisão, revocação e pontuação f1 foi a regressão logística usando a vetorização com TF-IDF, que por sua vez apresentou resultados acima de 70%.

Percebemos também que a vetorização por TF-IDF acaba sendo melhor para as métricas de precisão, revocação e pontuação f1, pois todos os classificadores performaram melhor ao escolher essa vetorização. Isso se deve ao fato de o TF-IDF de um termo dar maior ênfase aos termos característicos de um documento, facilitando a identificação de uma entrada que corresponda a um texto neutro ou a um negativo.

4.5 Comparação entre experimentos

Ao fim da última rodada dos experimentos, resolveu comparar os resultados vistos em 4.3 e 4.4 para saber ver as mudanças efetivas nos resultados com e sem a classe positiva no *dataset*.

Importante ressaltar que nessa comparação serão usados apenas os melhores resultados de cada rodada, separando apenas as comparações entre as implementações próprias e as implementações do `scikit`.

4.5.1 Implementações Próprias

Nas implementações próprias, comparou-se as duas versões do SVM com kernel RBF, uma com $\gamma = 0.1$ e $C = 1$ (caso multiclases) e outra com C e γ iguais a 1 (caso binário).

	Multiclases	Binário
Acurácia Média	68,3%	70,4%
Precisão Média	0.67	0.73
Revocação Média	0.65	0.72
F1-Score Média	0.64	0.72

Nas implementações próprias, nota-se melhora significativa ao remover a classe positiva, ao passo em que todos os valores médios aumentaram em pelo menos 2% em comparação com o caso multiclases. Isso se deve ao fato de que, no caso multiclases, muitos elementos da classe positiva eram classificados erroneamente, portanto abaixavam os valores de precisão, revocação e F1-Score médios.

4.5.2 Implementações do `scikit`

Já para as versões do `scikit`, comparou-se também duas versões do SVM, uma com kernel polinomial, $C = 1$ e grau 5 com coeficiente 5 (caso multiclases) com outra utilizando o kernel RBF, $C = 1$ e $\gamma = 0.158$ (caso binário).

	Multiclases	Binário
Acurácia Média	67,8%	71,3%
Precisão Média	0.63	0.66
Revocação Média	0.66	0.66
F1-Score Média	0.64	0.66

Para as implementações do `scikit`, nota-se que ao passo em que a acurácia média aumentou em quase 4%, as demais métricas não obtiveram aumento significativo em relação ao caso de três classes, porém ao olhar para [4.10](#) observa-se que para a classe positiva não teve nenhum elemento atribuído a ela, portanto as médias foram realizadas levando em consideração apenas as duas classes predominantes, justificando a semelhança de valores com a apresentada no caso binário.

Capítulo 5

Considerações finais

Neste trabalho, estudou-se a classificação de tuítes de política e, para isso, estudou-se a literatura a respeito do problema análise de sentimentos e modelos de aprendizado de máquina que poderiam ser usados para realizar a tarefa.

Ao fim deste trabalho, implementou-se versões do SVM e da regressão logística tanto para o caso binário quanto o de várias classes e, ao fim dos experimentos, teve-se que os modelos apresentaram uma acurácia de 68,3% para o caso de três classes e 70,4% no caso binário, valores próximos aos obtidos pelas implementações da biblioteca `scikit-learn`, além de apresentar valores médios próximos para as outras métricas avaliadas.

Além da implementação dos classificadores e dos experimentos realizados, desenvolveu-se o CLAM, ferramenta *open source* criada com o intuito de facilitar a etapa de classificação manual das opiniões de textos, disponível em <https://github.com/romaolucas/manual-classifier-helper>. Em [A](#) foi escrito um tutorial de como subir uma instância do CLAM no Heroku.

Ao fim das rodadas de experimentos, obteve-se um bom classificador para as classes mais presentes do nosso conjunto de dados que conciliava não só uma boa acurácia, mas também com um bom compromisso entre precisão e revocação. Futuras melhorias a serem feitas neste classificador seria utilizar seleção de características antes de treinar o modelo para assim mantermos apenas as *features* mais relevantes para cada classe e não só melhorar a acurácia do estimador, mas também o tempo de convergência de cada modelo.

Quanto ao problema com três classes é possível tentar melhorar a revocação e pontuação *f1* para a classe positiva introduzindo mais dados dessa classe ao nosso conjunto de dados e dar um peso maior a elementos que pertençam a esta classe para assim garantir uma diminuição da taxa de falso negativos, entretanto é importante ressaltar que conseqüentemente diminuiria a precisão do algoritmo pois com mais elementos classificados como positivos, maior

a chance de falso positivos.

Para o CLAM uma possível melhoria a ser implementada seria a possibilidade de mais usuários avaliarem um mesmo conjunto de textos para garantir um consenso na hora de associar uma opinião a um texto. Importante ressaltar que o sistema já foi desenvolvido preparado para suportar essa funcionalidade, bastaria modificar a listagem de textos a serem classificados para incluir aqueles que já possuem uma avaliação e aqueles que não possuem ainda.

Percebeu-se ao longo deste trabalho que mais importante do que um bom entendimento do funcionamento de cada classificador é analisar melhor os dados e achar elementos mais relevantes para cada classe para assim conseguir melhorar a representação dos dados como um vetor de *features* e, conseqüentemente, garantir melhores resultados.

Outra melhoria importante a ser realizada é ter mais tuítes classificados, para assim conseguir obter resultados mais expressivos e ter um conjunto de testes maior.

Do ponto de onde o trabalho é finalizado há espaço para não só melhorias quanto as discutidas nos parágrafos anteriores, mas também é possível aplicar técnicas de *part-of-speech tagging* [18] para descobrir não só a polaridade da opinião, mas também o que falaram acerca de cada entidade facilitando o uso da ferramenta para obter melhor entendimento acerca das discussões dos usuários. Além disso, classificar mais dados manualmente para garantir maior eficácia nos classificadores junto com permitir que alguns tuítes já classificados tenham uma segunda opinião a fim de diminuir a ambiguidade presente em alguns documentos também são melhorias a serem feitas.

Apêndice A

Configurando e instalando o CLAM

Neste apêndice, será ensinado como configurar e rodar o CLAM não só na sua máquina local, mas também disponibilizá-lo na internet utilizando o heroku para hospedagem. É interessante possuir uma versão do CLAM rodando localmente não só para verificar se as configurações atendem à suas necessidades, mas também para realizar testes locais de qualquer modificação que se deseja fazer.

A.1 Breve Introdução ao Heroku

Heroku é um serviço de nuvem *platform as a service* (PaaS) - uma categoria de serviços de nuvem que permite que usuários desenvolvam, administrem e rodem aplicações web sem a complexidade da construção da infraestrutura associada com o deploy da aplicação - que dá suporte a diversas linguagens de programação, dentre elas, Python.

Com o Heroku torna-se fácil escolher quais aplicações e tarefas devem ser executadas em produção e provem uma vasta documentação para auxiliar a construir uma aplicação nele, portanto sendo ideal para este tutorial.

A.2 Breve Introdução ao Django

Django é um framework web open source para Python que segue o padrão arquitetural model-view-template (MVT) que tem como objetivo facilitar a criação de aplicações complexas para a web. Para isso, é pregado fortemente os princípios de reusabilidade e "plugabilidade" de componentes, desenvolvimento rápido e o princípio DRY de engenharia de software.

O conjunto principal do Django traz consigo um *object-relational mapper* (ORM) para mediar entre os modelos de dados (representados como classes em Python) e uma base de dados relacional (Model), um sistema para processar requisições HTTP com um sistema de templates web (View) e um despachador de URLs que utiliza expressões regulares (Controller). Além disso possui integrado um admin onde é possível realizar a criação, procura, modificação e deleção de dados e este mesmo admin pode ter suas funcionalidades estendidas, caso necessário.

A.3 Preparando o ambiente local

Antes de mais nada, será preciso instalar os seguintes programas:

- PostgreSQL - banco de dados usado para a aplicação. Para instalar no Linux use o comando: `sudo apt-get install postgresql postgresql-contrib`
- Python-Pip - ferramenta para realizar o download de bibliotecas em python. Pode ser instalado usando o comando `sudo apt-get install python-pip`.
- Virtualenv - ferramenta utilizada para criar um ambiente isolado para o funcionamento de nossa aplicação, assim as instalações de pacotes e bibliotecas não irão afetar o funcionamento das demais bibliotecas do computador. Pode ser instalado usando o comando `sudo pip install virtualenv`.
- Heroku CLI - ferramenta necessária para utilizar as funcionalidades do heroku como fazer o deploy da aplicação, executar comando no servidor etc. Pode ser instalado pelo link: <https://devcenter.heroku.com/articles/getting-started-with-python#set-up>

A.3.1 Clonando o repositório e instalando dependências

Uma vez instalado todos os programas necessários, é necessário clonar o repositório do CLAM. Para isso execute o comando:

```
git clone https://github.com/romaolucas/manual-classifier-helper.
```

Com o repositório clonado, vamos criar um ambiente para instalarmos todas as bibliotecas necessárias para subir o servidor. Um ambiente é criado usando o comando `virtualenv <nome_do_ambiente>`, uma pasta com o nome fornecido será criado e, para ativar o ambiente, basta usar o comando `source <nome_do_ambiente>/bin/activate` e desativando usando o comando `deactivate`.

Agora acesse o diretório que contém o CLAM, caso não tenha renomeado, se chama `manual-classifier-helper`. No diretório iremos instalar as dependências utilizando o comando `pip install -r requirements.txt`.

A.3.2 Cadastro no Heroku

Outro passo importante a se fazer é cadastrar-se no Heroku e criar uma aplicação em python. O cadastro pode ser feito no link <https://signup.heroku.com/>. Depois de se cadastrar e confirmar o e-mail, você será levado ao dashboard do heroku onde é possível criar um novo app. Dê um nome para a sua aplicação e siga os passos seguintes para criar sua aplicação.

Uma vez com a aplicação criada, iremos fazer o login na heroku CLI. Execute o comando `heroku login` para fazer o login na sua máquina local.

Após o login acessa o diretório onde o clam se encontra que, caso não tenha renomeado, se chama `manual-classifier-helper`, no diretório iremos indicar ao git do heroku qual é o repositório remoto onde nossa aplicação se encontra usando o comando: `heroku git:remote -a <nome_do_app>`.

A.3.3 Criando banco de dados e usuário local

Nesta parte, iremos ensinar como criar um usuário para o django se conectar ao Postgres.

1. Acesse o Postgres com o comando: `sudo -U postgres psql`.
2. Crie o usuário `'mc_user'` com o comando: `CREATE USER mc_user WITH CREATEDB CREATEUSER PASSWORD 'cl4ss1f1c4r_d4d0$_eh_muito_chato'`. Note que aqui definimos nome e senha de usuário conforme definidos no arquivo `mchelper/settings.py`. Caso queira criar outro usuário ou outra senha para utilizar o banco de dados, basta mudar as linhas 82 e 83 do arquivo `settings.py`.
3. Crie a base de dados `manual_classifier` com o comando: `CREATE DATABASE manual_classifier`. Assim como para usuário e senha, caso queira usar outro nome de base de dados, basta modificar a linha 81 do arquivo `settings.py`.

Criados usuário e base de dados, iremos agora pedir ao Django para que ele crie as tabelas e aplique as migrações feitas às tabelas. Para isso, rode o comando `python manage.py migrate`. Feito isso todas as tabelas estarão criadas e a nossa disposição para usarmos.

A.4 Rodando o CLAM localmente

Seguindo passo a passo a seção anterior, você já consegue rodar perfeitamente o clam na sua máquina local. Para inicializar o servidor, basta fazer: `heroku local web`. Feito isso você conseguirá acessar o clam localmente no endereço `localhost:5000`.

Para utilizar o admin, será necessário criar um superusuário, isso pode ser feito com o comando `python manage.py createsuperuser` onde você será instruído a selecionar um nome de usuário e senha. Feito isso você conseguirá acessar todas as funcionalidades do CLAM.

A.5 Fazendo deploy do CLAM no Heroku

Antes de realizar o deploy da aplicação, será necessário indicar para o Django que o host '`sua_aplicação.herokuapp.com`' tem a autorização para ser um host HTTP da nossa aplicação. Isso é feito modificando a linha 28 do arquivo `mchelper/settings.py` trocando o `meu-app-teste.herokuapp.com` pelo nome correto de sua aplicação.

Feito isso, basta rodar os comandos:

```
git add .
git commit -m"<mensagem_de_commit>"
git push heroku master #fara o deploy da aplicação
```

Pronto! Agora sua aplicação já está deployada. Note que agora precisaremos criar as tabelas e executar as migrações assim como fizemos localmente além de criar um superusuário rodando os mesmos comandos usados na subseção [A.3.3](#). Com isso, sua aplicação já estará funcionando corretamente.

Referências Bibliográficas

- [1] Social Media Statistics & Facts. <https://www.statista.com/topics/1164/social-networks/>, 2017. [Online; acessado 25 de novembro de 2017.].
- [2] Filipe Oliveira. Brasil tem o 3º maior crescimento do Twitter em número de usuários. <http://www1.folha.uol.com.br/tec/2017/02/1861175-numero-de-usuarios-do-twitter-no-brasil-cresce-18-em-2016.shtml>, 2017. [Online; acessado 25 de novembro de 2017.].
- [3] Forsyth Alexander. How to use Twitter activity to measure the effectiveness of your marketing. <https://www.ibm.com/blogs/business-analytics/how-to-use-twitter-activity-to-measure-the-effectiveness-of-your-marketing/>, 2016. [Online; acessado 21 de janeiro de 2018.].
- [4] William Alden. Twitter Arrives on Wall Street, Via Bloomberg. <https://dealbook.nytimes.com/2013/04/04/twitter-arrives-on-wall-street-via-bloomberg/>, 2013. [Online; acessado 21 de janeiro de 2018.].
- [5] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, pages 1093–1113, 2014.
- [6] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. *Proceedings of the International Conference on Language Resources and Evaluation*, pages 1320–1326, 2010.
- [7] Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner, and Isabella M. Welp. Predicting elections with twitter: What 140 characters reveal about political sentiment. *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 178–185, 2010.

- [8] Martin Ringsquandl and Dušan Petković. Analyzing political sentiment on twitter. *AAAI Spring Symposium*, 2013.
- [9] Akshat Bakliwal, Jennifer Foster, Jennifer van der Puil, Ron O’ Brien, Lamia Tounsi, and Mark Hughes. Sentiment analysis of political tweets: Towards an accurate classifier. *Proceedings of the Workshop on Language in Social Media*, pages 49–58, 2013.
- [10] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1 edition, 2006.
- [11] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, pages 273–297, 1995.
- [12] Paul Dawkins. Lagrange multipliers. <http://tutorial.math.lamar.edu/Classes/CalcIII/LagrangeMultipliers.aspx>, 2003. [Online; acessado 1 de fevereiro de 2018.].
- [13] Bing Liu. *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers, 1 edition, 2012.
- [14] Kevin Sookocheff. Modeling natural language with n-gram models. <https://sookocheff.com/post/nlp/n-gram-modeling/>, 2015. [Online; acessado 1 de fevereiro de 2018.].
- [15] Jason Brownlee. A gentle introduction to the bag-of-words model. <https://machinelearningmastery.com/gentle-introduction-bag-words-model/>, 2017. [Online; acessado 1 de fevereiro de 2018.].
- [16] Julia Silge e David Robinson. Term frequency and inverse document frequency (tf-idf) using tidy data principles. https://cran.r-project.org/web/packages/tidytext/vignettes/tf_idf.html, 2018. [Online; acessado 1 de fevereiro de 2018.].
- [17] Scikit-Learn Developers. Cross-validation: evaluating estimator performance. http://scikit-learn.org/stable/modules/cross_validation.html#k-fold, 2017. [Online; acessado 1 de fevereiro de 2018.].
- [18] Daniel Jurafsky & James H. Martin. *Speech and Language Processing*. Prentice Hall, 3 edition, 2017.