Text Classification for Twitter Sentiment

1st Igor Mourão Ribeiro
Computer Engineering Departament
Instituto Tecnológico de Aeronáutica
São José dos Campos, Brazil
igormr98mr@gmail.com

2nd Isabelle Ferreira de Oliveira
Computer Engineering Departament
Instituto Tecnológico de Aeronáutica
São José dos Campos, Brazil
isabelle.ferreira3000@gmail.com

3rd José Luciano de Morais Neto
Computer Engineering Departament
Instituto Tecnológico de Aeronáutica
São José dos Campos, Brazil
zluciano.t19@gmail.com

***Abstract*—Esse relatório documenta a implementação de algoritmos de Processamento de Linguagem Natural, aplicando diferentes técnicas de Machine Learning para classificar Tweets entre sentimentos positivos e negativos. Os algoritmos implementados foram Naive Bayes e Support Vector Machine, utilizando diferentes features produzidas a partir de um dataset do Kaggle, e comparando-os pelas métricas de acurácia e coeficiente Kappa.**

***Index Terms*—Processamento de Linguagem Natural, Naive Bayes, Aprendizagem Supervisionada, Support Vector Machine**

I. Introdução

Um dos aspectos relevantes da interação entre as pessoas na atualidade é a expressão de sentimentos por meio de textos nas mídias sociais. Nesse contexto, o monitoramento das redes sociais pode ser explorado como forma de extrair a aceitação e/ou aprovação de produtos e também obter conhecimento dos usuários. A análise de sentimentos surge da necessidade de tratar e interpretar textos, opiniões e comentários realizados pelos usuários em redes sociais. Por meio das informações subjetivas extraídas textos em linguagem natural, pode ser gerado conhecimento estruturado, auxiliando a tomada de decisão.

A expansão da Internet e a utilização das redes sociais definiram um ecossistema de interação, no qual os usuários deixaram de ser receptores passivos e se tornaram produtores, compartilhadores e avaliadores de conteúdo. Em um cenário onde as reputações de empresas e a aceitabilidade de produtos no mercado são diretamente afetadas pela repercussão de opiniões de seus clientes na web, tanto quanto pelas campanhas de publicidade, a análise de sentimentos surge como um diferencial para rastreamento do conteúdo emocional daquilo que se escreve e compartilha nas redes sociais. Nesse sentido, a análise de sentimentos alia-se à publicidade promovendo subsídios para definição de estratégias e garantia da vantagem competitiva.

A análise de sentimentos ser aplicada na gestão de informação por exemplo, fornecendo feedback do cliente a partir do conteúdo dos diversos canais de comunicação e entregando informações úteis para tomada de decisão e definição de estratégias para satisfação dos clientes.

O Twitter [2] é uma rede social e servidor para microblogging muito utilizada, que será utilizada para essa análise de sentimentos. Atualmente, o limite máximo de um *tweet* (mensagem postada no blog) é de 280 caracteres e tem-se um total de 6000 *tweets* por segundo o que implica em 200 bilhões por ano. Exemplos de *tweets* com emoções podem ser vistos nas Figuras 1 e 2.

Fig. 1. *Tweet* com mensagem positiva

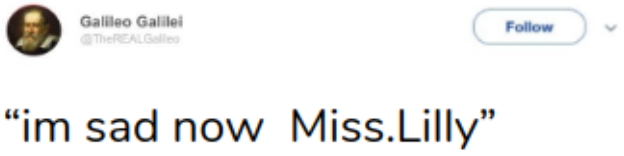

Fig. 2. *Tweet* com mensagem negativa

Neste sentido, este trabalho tem como objetivo apresentar a análise de sentimentos aplicada a textos em linguagem natural de uma rede social, usando diversos modelos e entradas para posterior comparação.

II. Fundamentos Teóricos

A. Definições

Neste projeto, são utilizadas as notações m_i, para a i-ésima mensagem em uma amostra do *dataset* com $1 \leq i \leq M$ (sendo M o número de mensagens na amostra) e correspondendo a um vetor frequência de *features* $f(x_j, m_i)$, onde x_j representa a j-ésima *feature* da mensagem m_i. Cada vetor m_i de frequência de *features* tem a mesma dimensão F, correspondendo à quantidade de *features* consideradas, i. e., $1 \leq j \leq F$.

As *features* são a menor unidade de informação considerada em uma mensagem. Foram consideradas quatro tipos de *features*: *Count Vectors* (CV), Palavra à palavra, Caracter

à caracter e N-Grams. No caso das três últimas *features* referidas, foi considerado o modelo *Term-Frequency Inverse Document Frequency*, enquanto a *feature* CV seguiu o modelo *Term Frequency*, ambos apresentados na seção D.

B. Naive-Bayes

Considerando as classes C_k a serem classificadas as mensagens m (no caso deste projeto C_1 = positiva e C_2 = negativa), o modelo *Naive-Bayes* realiza a classificação por meio da maximização da probabilidade $P(C_k|m)$, ou seja, busca a classe k que maximiza tal probabilidade. Para tanto, o *teorema de Bayes*, apresentado na **Equação** (1), é utilizado

$$P(C_k|m) = \frac{P(C_k)P(m|C_k)}{P(m)} \tag{1}$$

a suposição do modelo (e que dá nome ao mesmo) é que as *features* x_j^m em uma mensagem m são independentes. Assim, a probabilidade condicional $P(m|C_k)$ toma a forma apresentada na **Equação** (2),

$$P(m|C_k) = \prod_{j=1}^{F} P(x_j^m|C_k)^{f(x_j,m)} \tag{2}$$

na qual $f(x_j, m)$ é a frequência de ocorrência da *feature* x_j na mensagem m (definida na seção D), e de onde podemos expressar a probabilidade $P(C_k|m)$ como na **Equação** (3),

$$P(C_k|m) = \alpha P(C_k) \prod_{j=1}^{F} P(x_j^m|C_k)^{f(x_j,m)} \tag{3}$$

onde a constante de normalização α independe das classes C_k [3]. Enfim, o modelo realiza a classificação por meio da estimativa $\hat{C}_m$, dada pela **Equação** (4),

$$\hat{C}_m = \underset{k}{argmax} \{P(C_k|m)\} \tag{4}$$

As probabilidades $P(x_j^m|C_k)$ são estimadas em conjunto com a *suavização de Laplace*, por meio da **Equação** (5),

$$\hat{P}(x_j^m|C_k) = \frac{1 + \sum\limits_{m_p \in C_k} f(x_j, m_p)}{F + \sum\limits_{q|m_r \in C_k} f(x_q, m_r)} \tag{5}$$

onde a soma no numerador se dá sobre todas as mensagems m_p da amostra do *dataset* que pertencem à classe C_k e a soma no denominador se dá sobre todas as *features* x_q de todas as mensagems m_r da amostra do *dataset* que pertencem à classe C_k. As probabilidades $P(C_k)$ são estimadas pela simples frequência das classes na amostra do *dataset*, ou seja,

$$\hat{P}(C_k) = \frac{\sum\limits_{p} \mathbb{1}[m_p \in C_k]}{M} \tag{6}$$

onde o somatório no numerador conta a quantidade de mensagens classificadas como pertencentes à classe C_k.

C. Support Vector Machine - (SVM)

A ideia principal de uma SVM [6] é a de dividir o dataset através de um hiperplano que separe os dados nas duas classificações previstas no seu rótulo de acordo com o apredizado supervisionado.

Além disso, uma margem é definida a partir desse hiperplano que separa as classificações e as distancia uma da outra, como pode ser visot na ilustrção a seguir:

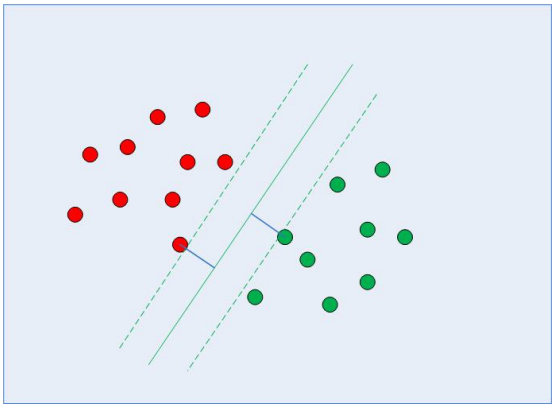

Fig. 3. Exemplo ilustrativo de uma divisão feita por um SVM.

A escolha do hiperplano e da margem, i.e. os parâmetros escolhidos para se definir a separação do *dataset*, é feita de tal forma que o erro associado às más classificações de dados presentes no *dataset* de treinamento seja minimizado.

A escolha é feita dessa forma para que o risco associado seja mínimo. Entende-se por risco, a probabilidade de se errar uma futura classificação de um dado desconhecido.

D. Term Frequency - Inverse Document Frequency

A princípio, as frequências $f(x_j, m_p)$ da seção B poderiam ser calculadas como a quantidade de vezes que a *feature* x_j está presente nas mensagens m_p. Esta é a chamada *Term Frequency* (TF), apresentada nas **Equações** (7) **e** (8),

$$f_{TF}(x_j, m_p) = \sum_{w \in m_p} \mathbb{1}[w = x_j] \tag{7}$$

$$f_{nTF}(x_j, m_p) = \frac{f_{TF}(x_j, m_p)}{\sum\limits_{q} f_{TF}(x_q, m_p)} \tag{8}$$

onde o somatório na **Equação** (7) se dá sobre todos os termos presentes na mensagem m_p e a **Equação** (8) representa a *Term Frequency* normalizada pela quantidade de termos na mensagem m_p. Porém, uma outra representação, chamada *Term Frequency - Inverse Document Frequency* (TF-IDF), se mostrou mais eficaz na classificação de texto em documentos (no caso deste projeto, mensagems). A f_{TFIDF} é dada pelas **Equações** (9) e (10),

$$f_{TFIDF}(x_j, m_p) = f_{nTF}(x_j, m_p) \cdot f_{IDF}(x_j) \tag{9}$$

$$f_{IDF}(x_j) = log\left(\frac{M}{\sum\limits_{q} \mathbb{1}\left[f_{TF}(x_j, m_q) \neq 0\right]}\right) \quad (10)$$

onde $f_{IDF}(x_j)$ é o *Inverse Document Frequency*, o logaritmo do inverso da fração de mensagens da amostra onde a *feature* x_j ocorre.

E. Coeficiente Kappa de Cohen

Dependendo dos dados escolhidos, mesmo que ao acaso, para o treinamento das diferentes redes neurais, vários enviesamentos podem acontecer e, por esse motivo, existe uma maneira de comparar duas formas de classificação através do Coeficiente Kappa de Cohen [5].

O objetivo deste coeficiente é de comparar, entre dois tipos de classificação, o grau de acordo a partir da seguinte equação:

$$\kappa = \frac{P_o - P_e}{1 - P_e} \quad (11)$$

Onde P_o representa a proporção de casos em que ambos classificaram na mesma categoria, enquanto que P_e representa a probabilidade da classificação na mesma categoria ocorrer por mera coincidência.

Quão mais próximo de P_e for P_o, temos que a proporção de concordância se aproxima da probabilidade de acerto por coincidência, ou seja, a concordância se aproxima do aleatório e o valor de κ se aproxima de 0. Enquanto que quanto mais próximo de 1 for P_o, temos que a concordância se aproxima da concordância absoluta e o valor de κ se aproxima de 1.

Assim, à medida que κ varia de 0 a 1, a concordância entre as classificações fica cada vez maior.

III. Proposta

Neste trabalho, propomos a utilização de classificadores baseados em *Naive-Bayes* e SVMs para a classificação da positividade de *tweets*. Foram experimentadas diversos tipos de *features* a fim de se realizar uma comparação da performance dos classificadores: para o *Naive-Bayes*, foram testados os modelos de *Count Vector* (TF), Palavra à palavra (TF-IDF), N-Gram (de palavras com TF-IDF) e Caracter à caracter (também com TF-IDF); para a SVM, foi testado o uso de N-Gram (de palavras com TF-IDF).

IV. Metodologia

O projeto foi implementado na linguagem *Python 3*, fazendo uso da biblioteca *scikit-learn*, feita para aplicações de *Machine Learning*.

Foi utilizado um *dataset* obtido no *website* Kaggle com 1.6 milhões de *tweets*, todos em inglês, em que, para cada *tweet*, há um número de 0 a 4 expressando a positividade do mesmo (4 mais positivo e 0 mais negativo). O *dataset* completo pode ser encontrado em [1].

Inicialmente, foi feito um pré-processamento do *dataset*, de forma a extrair os dados necessários para o treinamento e para retirar caracteres no texto que não são úteis para a classificação.

Foram removidos do *dataset* os campos de data que o *tweet* foi feito e o ID único de cada *tweet*. Com isso, obtiveram-se apenas a classificação da positividade e o texto para cada *tweet*.

Em seguida, foi feito o pré-processamento de texto, em que, inicialmente, foi colocado todo em minúsculo. Além disso, por meio de uma análise qualitativa de *tweets*, foram notados diversos nomes de pessoas marcadas no texto, representadas quando a palavra começa com um '@'. Esses nomes não ajudam na classificação do texto, assim como pontuações e números, de forma que foram eliminadas a presença de quaisquer caracteres não alfabéticos do texto, enquanto eliminado o nome de pessoas referenciados no texto.

Por último, foram randomizadas as entradas do *dataset*, de forma a diminuir o *bias* no treinamento. Depois o *dataset* foi separado em 75% para o conjunto de treinamento e 25% para o conjunto de testes. Finalmente, por motivos do treinamento do *dataset* completo não poder ser realizado devido ao limite de memória (8GB de RAM), foram utilizadas apenas 1 milhão de entradas para o treinamento com o modelo *Naive Bayes* e 100 mil entradas para o SVM. O número menor de entradas para o SVM é justificado em seu grande custo computacional no treinamento.

V. Experimentos, Resultados e Análise

A seguir tem-se duas tabelas que ilustram os resultados obtidos com os experimentos. Na primeira podem ser observados a acurácia dos resultados e dos testes para cada método utilizado de treinamento:

TABLE I
PORCENTAGENS DE ACERTO DOS MODELOS NOS CONJUNTOS DE TESTE E TREINAMENTO

Modelo	***Naive Bayes***				**SVM**
Features	***CV***[a]	***Word***[b]	***N-Gram***[b]	***Char***[b]	***N-Gram***[b]
Treino	81.71%	77.28%	77.52%	74.18%	70.49%
Testes	78.17%	77.03%	77.41%	74.13%	68.63%

[a] segundo o modelo TF.
[b] segundo o modelo TF-IDF.

O cálculo para o valor de Kappa para cada modelo foi feito considerando uma comparação com os próprios rótulos dos dados [7], ou seja, a comparação foi feita com um classificador perfeito que tem 100% de acurácia com o *dataset*. Esses valores podem ser vistos na tabela a seguir:

TABLE II
VALORES DO COEFICIENTE KAPPA PARA CADA MODELO

Modelo	***Naive Bayes***				**SVM**[a]
Features	***CV***[a]	***Word***[b]	***N-Gram***[b]	***Char***[b]	***N-Gram***[b]
Kappa	0.563	0.541	0.584	0.482	0.361

[a] segundo o modelo TF.
[b] segundo o modelo TF-IDF.

Quando um classificador randômico tenta classificar os dados, é esperado que ainda se obtenha uma acurácia de 50% pois estaria acertando de maneira aleatória. O que importa mesmo é a acurácia da classificação entre 50% e 100%, pois

é a partir daí que se ressalta o diferencial perante um palpite randômico.

Como era de se esperar, o valor de Kappa reflete exatamente isso. Seu valor variando de 0 a 1 representa a acurácia de cada método variando de 50% a 100%. Como podemos ver a seguir:

$$\kappa = \frac{P_1 - P_2}{1 - P_2} \tag{12}$$

P_1 representa a concordância do método com o valor do rótulo, ou seja, a acurácia dos testes. P_2 representa a probabilidade de acertar o valor de maneira aleatória que, por se tratar de 2 valores possíveis de rótulos e considerando que o *dataset* está bem distribuído, vale em torno de 50%.

Isso significa que, embora o entedimento estatístico do problema permita fazer essa observação sobre os valores de acurácia que realmente importam, o valor de Kappa dispensa esse conhecimento e dá um valor significativo sobre o quão bom é cada classificador.

VI. Conclusões

Pode-se concluir através dos resultados que o SVM, mesmo exigindo um processamento muito grande e trabalhando com 10% do tamanho padronizado de dados, foi o que teve o pior resultado, sendo assim um classificador ruim se comparado aos demais.

Sendo assim, o método de *Naive-Bayes* foi consideravelmente se baseando nos valores de Kappa obtidos. Sendo o método em *Word-Level* ligeiramente melhor que os demais, embora todos, tenham tido um desempenho similar, com exceção do método em *Char-Level* que teve um desempenho um pouco menor entre os que foram feitos baseados em *Bayes*.

References

[1] Dataset de Sentimentos, https://www.kaggle.com/kazanova/sentiment140.

[2] Twitter, https://twitter.com

[3] Kibriya, Ashraf & Frank, E. & Pfahringer, Bernhard & Holmes, Geoffrey. (2004). Multinomial naive Bayes for text categorization revisited. Advances in Artificial Intelligence. 488-499.

[4] SciKit-Learn, https://scikit-learn.org/stable/

[5] Cohen, Jacob (1960). "A coefficient of agreement for nominal scales". Educational and Psychological Measurement. 20 (1): 37–46. doi:10.1177/001316446002000104

[6] Implementing SVM and Kernel SVM with Python's Scikit-Learn https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/

[7] sklearn.metrics.cohen_kappa_score https://scikit-learn.org/stable/modules/generated/sklearn.metrics.cohen_kappa_score.html