Processamento de Linguagem Natural

Railson Martins da Mata [1], Lucas dos Anjos[1], Gabriel Catarino[1], Laerte Mateus Rodrigues[2]

[(1)]Instituto Federal de Minas Gerais (IFMG) - Campus Bambuí

RESUMO

O presente trabalho busca por meio de 2 algoritmos um sendo o SVM do inglês (support vector machine) e também pelo decision tree conhecida em português como árvore de decisão, fazer a classificação por meio PLN – Processamento de Linguagem Natural textos coletados do twitter.

Palavras-Chave: Processamento de Linguagem Natural, SVM, Decision Tree, Machine Learning

1. INTRODUÇÃO

O aprendizado de máquina (em inglês, machine learning) é um método de análise de dados que automatiza a construção de modelos analíticos. É um ramo da inteligência artificial baseado na ideia de que sistemas podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana (Sas, 2018).

As Support Vector Machines (SVM) são uma das técnicas mais utilizadas em Data Mining quando é necessário construir um classificador para dados constituídos maioritariamente por variáveis numéricas continuas/discretas(Aprendis, 2018)

Árvores de decisão são modelos estatísticos que utilizam um treinamento supervisionado para a classificação e previsão de dados. Em outras palavras, em sua construção é utilizado um conjunto de treinamento formado por entradas e saídas. Estas últimas são as classes.(Maxwell, 2018)

2. DESENVOLVIMENTO

Foi se dado uma base de dados em csv com textos referentes ao twitter, o mesmo possuía 3 colunas sendo a primeira o sentimento que aquele texto proporcionava a segunda o nome da empresa e a terceira o respectivo texto.
Para o processo de predição foi necessário fazer a remoção das stop words figura 1 e também o stemming figura 2 a seguir temos os códigos dos métodos. Além dessas foram usadas outras funções para remoção de emojis etc.

```
def removeStopWords(texto):
    tokenizer = RegexpTokenizer(r'\w+')
    x = tokenizer.tokenize(texto)

    stopWords = set(stopwords.words('english'))

    wordsFiltered = []
    for w in x:
        if w not in stopWords:
            #print(w)
            wordsFiltered.append(w)

    listaAux = removerAcentos(wordsFiltered)
    #wordsFiltered = removerAcentos(wordsFiltered)

    return listaAux
```

Figura 1 – Remover Stop words

```
def stemming(texto):
    from nltk.stem import PorterStemmer
    x = []
    p = PorterStemmer()
    for i in texto:
        if len(i) <= 3:
            del i
        else:
            x.append(p.stem(i))
    return x
```

Figura 2 - Stemming

Outra função importante foi usada pra fazer a leitura do csv e conversão das palavras para que ambos os algoritmos pudessem interpretar mostraremos ela a seguir figura 3:

```
def lendoArquivoCsv():
    x = []
    global iteracao
    new_file = csv.reader(open('tweets.csv', encoding="latin-1"))
    for linha in new_file:
        x = removeUrl(linha[2])
        x = removeStopWords(x)
        x = removeEmoji(x)
        x = removeNumeros(x)
        #print(x)
        x = stemming(x)

        if len(x) == 1:
            del x
        else:
            #escrevendo no arquivo editado
            with open('tweets_filtrados.csv', 'a', newline='') as saida:
                escrever = csv.writer(saida)

                if iteracao == 0:
                    a = []
                    a = ["Sentimento", "Texto"]
                    escrever.writerow(a)
                    iteracao = iteracao + 1
                else:
                    documents = ' '.join(x)
                    sentimento = ''.join(linha[0])
                    lista = [sentimento, documents]
                    print(lista)
                    escrever.writerow(lista)

    saida.close()
```

Figura 3 – Leitura, modificação e gravação arquivo csv

Após todas essas conversões foi posto em prática ambos os algoritmos, primeiro iremos falar sobre o SVM.
Conseguimos classificar os tweets e as classes do arquivo, também conseguimos treinar o mesmo, só que na hora de conseguir determinar a acurácia não conseguimos que a função da mesma retorna-se um valor desejado. Na linha 31 do arquivo do SVM vemos o problema. No algoritmo Decison Tree houve o mesmo problema. A seguir temos os testes do svm figura 4.

```
11
12  dataset = pd.read_csv('tweets_filtrados.csv')
13  dataset.count()
14
15  #separando os tweets das classes
16  y = dataset['Sentimento'].values
17  x = dataset['Texto'].values
18
19  #treinando usando svm
20  X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size = 0.20)
21  vectorizer = CountVectorizer(analyzer="word")
22  freq_tweets = vectorizer.fit_transform(x)
23  modelo = svm
24  modelo.SVC(freq_tweets,y)
25
26  #Fazendo classificação
27  freq_testes = vectorizer.transform(X_test)
28  #modelo.fit(freq_tweets, y)
29
30  #resultados = cross_val_predict(modelo, freq_tweets, y, cv=10)
31  print(accuracy_score(Y_test, X_test))
32
33
34
```

Figura 4 - SVM

3. CONCLUSÃO

Como exposto acima, vimos que nossos resultados não foram como o esperado, onde na hora de usarmos a função para calcular a acurácia o retorno da mesma não foi como desejado. Entretanto na parte de remoção de stopwords e stemming o algoritmo se portou de forma esperada e retornou uma saída plausível que pode ser verificada junto ao arquivo mandado com esse relatório.

4. REFERÊNCIAS

Machine Learning, qual é a sua importância? Link disponível em: < https://guiadamonografia.com.br/citacao-de-site-e-artigo-da-internet/> acesso em: 16 de dezembro de 2018.
Support Vector Machines – Link disponível em: < http://aprendis.gim.med.up.pt/index.php/Support_Vector_Machines> acesso em: 16 de dezembro de 2018.
Arvore de decisão – Link disponível em: < https://www.maxwell.vrac.puc-rio.br/7587/7587_4.PDF> acesso em: 16 de dezembro de 2018.