Artigo

January 18, 2019

1 Análise de Sentimento

Neste artigo será utilizado o algoritmo "Naive Bayes" é um classificador probabilístico baseado no "Teorema de Bayes", a caracteristica mais proeminente deste algoritmo é o fato de que ele ignora a ligação de cada variável com outras, ou seja, em uma frase "Eu sou Leonardo" a frase pode ser separada da seguinte forma "Eu", "sou", "Leonardo", e cada trecho da frase se torna independente da outra.

Há outras maneiras de se dividir a frase também utilizando o conceito de n-grama, por exemplo se fosse utlizado na frase acima o bi-grama, a frase seria dividida da seguinte maneira: "Eu sou", "sou Leonardo". E essas duas partes continuam idependentes uma da outra. O fato de algoritmo não levar em conta as relações é o motivo pelo nome do algoritmo "naive" que significa ingênuo.

1.1 Imports

```
In [5]: from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.naive_bayes import MultinomialNB
        from sklearn import metrics
        from sklearn.model_selection import cross_val_predict
```

1.2 Leitura dos dados

```
In [11]: arq = open('amazon_cells_labelled.txt', 'r')
         amazon = []
         amazonC = []
         for linha in arq:
             frase = linha.replace('\t', ' ').replace('\n', '')
             amazonC.append(frase[len(frase)-1])
             frase = frase[0:len(frase)-2]
             amazon.append(frase)
         arq.close()
         arq = open('imdb_labelled.txt', 'r')
         imdb = []
         imdbC = []
         for linha in arq:
             frase = linha.replace('\t', ' ').replace('\n', '')
             imdbC.append(frase[len(frase)-1])
```

```
        frase = frase[0:len(frase) - 2]
        imdb.append(frase)
    arq.close()
    arq = open('yelp_labelled.txt', 'r')
    yelp = []
    yelpC = []
    for linha in arq:
        frase = linha.replace('\t', ' ').replace('\n', '')
        yelpC.append(frase[len(frase)-1])
        frase = frase[0:len(frase) - 2]
        yelp.append(frase)
    arq.close()
    frases = amazon + imdb + yelp
    classes = amazonC + imdbC + yelpC
```

1.3 Bag Of Words

O modelo bag-of-words é uma representação simplificada usada no processamento de linguagem natural e recuperação de informação (IR). Neste modelo, um texto (como uma frase ou um documento) é representado como o saco (multiset) de suas palavras, desconsiderando a gramática e até a ordem das palavras, mas mantendo a multiplicidade

1.3.1 Utilizando CountVectorizers

A primeira linha cria um objeto que irá vetorizar as frases as separando por palavra. A segunda linha utiliza o objeto criado contar as frequencias das palavras do IMDB.

```
In [13]: vectorizer = CountVectorizer(analyzer="word")
         freq_imdb = vectorizer.fit_transform(imdb)
```

1.4 Criando o Modelo

A primeira linha cria o modelo com base no algoritmo Naive Bayes, a segunda linha utiliza as frequencias guardadas na variável *freq_imdb* e o sentimento de cada frase captado na leituras dos documentos e aplica o sentimento para cada palavra e calcula a probabilidade de aquela variavel ter conotação positiva ou negativa.

```
In [15]: modelo = MultinomialNB()
         modelo.fit(freq_imdb,imdbC)

Out[15]: MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

1.5 Aplicando o Modelo

A primeira linha transforma o conteúdo da variável *frases* em um bag of words, já na segunda linha o modelo tenta predizer o sentimento de cada frase, na última linha é impresso a porcentagem de acerto.

```
In [20]: freqFrases = vectorizer.transform(frases)
         resultados = modelo.predict(freqFrases)
         print('Grau de Acerto: {}%'.format(round(metrics.accuracy_score(classes, resultados)*
```

```
Grau de Acerto: 81.17%
```

1.6 Melhorando o Modelo

Para fazer isso iremos utilizar a técnica de Cross-Validation (Validação Cruzada), consiste em dividir todo o dado em K partes, chamadas de folds. Dessas partes uma será separada para teste e as outras restantes serão usadas para treinar o modelo. No próximo exemplo o corpus será dividido em 100. E podemos ver uma leve melhora e quantos mais partes dividirmos melhor será o resultado, porém será mais demorado.

```
In [30]: resultados = cross_val_predict(
             modelo, freqFrases, classes, cv=100)
         print('Grau de Acerto: {}%'.format(round(metrics.accuracy_score(classes, resultados)*
```

```
Grau de Acerto: 82.8%
```

2 Conclusão

Apesar do tamanho reduzido do corpus e a simplicidade do algoritmo, este retornou um bom resultado.

3 Bibliografia

SANTANA, Rodrigo. **Análise de Sentimentos**: Aprenda de uma vez por todas como funciona utilizando dados do Twitter. 2017. Disponível em: http://minerandodados.com.br/index.php/2017/03/15/analise-de-sentimentos-twitter-como-fazer/. Acesso em: 18 jan. 2019.

CANDIAGO, Lorenzo. **Algoritmo de Classificação Naive Bayes.** 2017. Disponível em: https://www.organicadigital.com/seeds/algoritmo-de-classificacao-naive-bayes/. Acesso em: 18 jan. 2019.

NOVELLO, Rafael. **Um pouco de Machine Learning com Python.** 2012. Disponível em: https://imasters.com.br/back-end/um-pouco-de-machine-learning-com-python. Acesso em: 18 jan. 2019.


```
In [ ]:
```