

Análise de sentimentos rápida e precisa com um classificador Naive Bayes aprimorado

The background features a dark blue grid. A white line graph with circular markers is plotted across the middle of the image, showing a fluctuating trend. The line starts at a low point, rises to a peak, falls to a trough, rises again to a higher peak, falls to another trough, and then rises to a final peak before ending at a slightly lower point.

Denildo Veloso

Igor Neres Trindade

Lucas Ferraz

Ramon Neres

Um pouco sobre o artigo

Título: Fast and Accurate Sentiment Classification Using an Enhanced Naive Bayes Model

14th International Conference IDEAL 2013
(Intelligent Data Engineering and Automated Learning)

193 citações

Fast and Accurate Sentiment Classification Using an Enhanced Naive Bayes Model

Vivek Narayanan, Ishan Arora, and Arjun Bhatia

Department of Electronics Engineering,
Indian Institute of Technology (BHU), Varanasi, India
{vivek.narayanan.ece09, ishan.arora.ece09,
arjun.bhatia.ece09}@iitbhu.ac.in

Abstract. We have explored different methods of improving the accuracy of a Naive Bayes classifier for sentiment analysis. We observed that a combination of methods like effective negation handling, word n-grams and feature selection by mutual information results in a significant improvement in accuracy. This implies that a highly accurate and fast sentiment classifier can be built using a simple Naive Bayes model that has linear training and testing time complexities. We achieved an accuracy of 88.80% on the popular IMDB movie reviews dataset. The proposed method can be generalized to a number of text categorization problems for improving speed and accuracy.

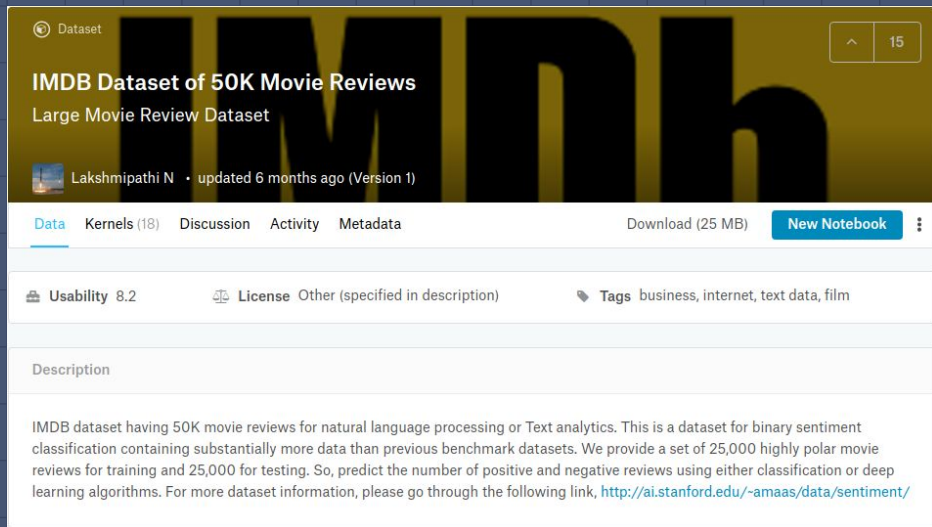
Keywords: Sentiment classification, Negation Handling, Mutual Information, Feature Selection, n-grams.

1 Introduction

Among the most researched topics of natural language processing is sentiment analysis. Sentiment analysis involves extraction of subjective information from documents like online reviews to determine the polarity with respect to certain objects. It is useful for identifying trends of public opinion in the social media, for the purpose of marketing and consumer research. It has its uses in getting customer feedback about new product launches, political campaigns and even in financial markets [14]. It aims to determine the attitude of a speaker or a writer with respect to some topic or simply the contextual polarity of a document. Early work in this area was done by Turney and Pang ([2], [7]) who applied different methods for detecting the polarity of product and movie reviews.

Sentiment analysis is a complicated problem but experiments have been done using Naive Bayes, maximum entropy classifiers and support vector machines. Pang et al. found the SVM to be the most accurate classifier in [2]. In this paper we present a supervised sentiment classification model based on the Naive Bayes algorithm.

Naive Bayes is a very simple probabilistic model that tends to work well on text classifications and usually takes orders of magnitude less time to train when compared to models like support vector machines. We will show in this paper that a high degree of accuracy can be obtained using Naive Bayes model, which is comparable to the current state of the art models in sentiment classification.



Dados utilizados

Andrew Maas

50.000 comentários, **2** conjuntos

- 25.000 avaliações para treino
- 25.000 avaliações para teste

<http://ai.stanford.edu/~amaas/data/sentiment/>

Kaggle:

<https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>

Mesma proporção pos/neg

Nenhum filme possui mais de 30 avaliações

Nenhum filme está em ambos os conjuntos

Conjunto de treino não apresenta avaliações com notas neutras.

Pipeline do artigo

- Tokenização

Sinais de pontuação importam!

"['- 'a -zA-ZÀ-Öø-öø-ÿ]+ | [!?:;,.,]"

- Utilização de n-gramas
- Manipulação da negação
- Seleção de Features

Maiores frequências

PMI

Naives Bayes Classifier vs. Bernoulli Naive Bayes Classifier

Pipeline do artigo

- Tokenização

Sinais de pontuação importam!

"['- 'a -zA-ZÀ-Öø-öø-ÿ]+ | [!?:,.,]"

- Utilização de n-gramas
- Manipulação da negação
- Seleção de Features

Maiores frequências

PMI

$$O(n + V \log V)$$

Naives Bayes Classifier vs. Bernoulli Naive Bayes Classifier

Mas o que há de inédito nesse trabalho?

O que não tratamos em sala?

Na implementação do artigo:

- Intel Core 2 Duo processor at 2.1 GHz
- 1 minute e 30 segundos
- 700 megabytes de memória

Na implementação do artigo:

- Intel Core 2 Duo processor at 2.1 GHz
- 1 minute e 30 segundos
- 700 megabytes de memória



Sobre o processamento

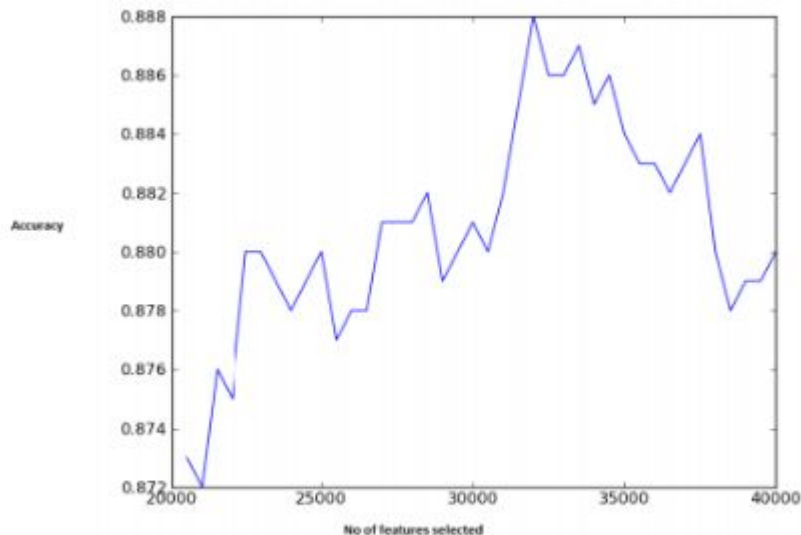


Fig. 1. Plot of Accuracy v/s No of features selected on Validation set

Não somente acurácia.

Tempo também é importante!

Table 1. Results Timeline

Feature Added	Accuracy on test set
Original Naive Bayes algorithm with Laplacian Smoothing	73.77%
Handling negations	82.80%
Bernoulli Naive Bayes	83.66%
Bigrams and trigrams	85.20%
Feature Selection	88.80%

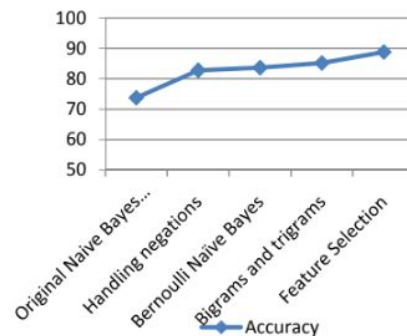


Fig. 2. Evolution of classification accuracy

Será que conseguimos?



Será que conseguimos?

Thursday
Aug 2019 22

CPU **43%**



chrome 19.02%
python 12.85%
pulseaudio 3.34%
chrome 2.57%

RAM **7.34GiB**

python 6.41GiB
chrome 135MiB
chrome 132MiB
chrome 106MiB

Swap **3.85GiB/ 1.15GiB**

Manjaro Linux 18.0.4 Illyria
igor@igor-manjaro
uptime: 14h 40m
kernel: 4.14.138-1-MANJARO

Nome	Status	35% CPU	96% Memória
>  Python 3.7		27,2%	11.383,1 MB
>  Google Chrome (23)		2,3%	764,8 MB

Será que conseguimos?

Nossa implementação

```

Digite o valor desejado para n-grams (Máximo 3):
3
Deseja utilizar feature selection? (S/N)
n
Deseja carregar as features dos arquivos? (S/N)
n
Início da leitura, tokenização, n-gramas e negation handling.
Fim da leitura dos arquivos.
--- 271.45 segundos ---
Textos de treino com avaliação positiva:
Total: 12500
Textos de treino com avaliação negativa:
Total: 12500
Textos de teste com avaliação positiva:
Total: 12500
Textos de teste com avaliação negativa:
Total: 12500
Início do treinamento.
Fim do treinamento.
--- 182.37 segundos ---
Início dos testes.
Acurácia da classe positiva: 86.86%
Acurácia da classe negativa: 89.58%
Acurácia: 88.22%
Fim dos testes.
--- 59.81 segundos ---
--- Tempo total: 522.35 segundos ---
Digite sua avaliação (0 para sair)

```

```

Digite o valor desejado para n-grams (Máximo 3):
3
Deseja utilizar feature selection? (S/N)
s
Deseja carregar as features dos arquivos? (S/N)
n
Início da leitura, tokenização, n-gramas e negation handling.
Fim da leitura dos arquivos.
--- 61.49 segundos ---
Textos de treino com avaliação positiva:
Total: 12500
Textos de treino com avaliação negativa:
Total: 12500
Textos de teste com avaliação positiva:
Total: 12500
Textos de teste com avaliação negativa:
Total: 12500
Início do treinamento.
Fim do treinamento.
--- 3.86 segundos ---
Início dos testes.
Acurácia da classe positiva: 63.78%
Acurácia da classe negativa: 78.03%
Acurácia: 70.91%
Fim dos testes.
--- 4.36 segundos ---
--- Tempo total: 73.35 segundos ---
Digite sua avaliação (0 para sair)

```

Nossa implementação

Mas... por que o **feature selection** influencia tanto?



Mas... por que o **feature selection** influencia tanto?



COM FEATURE SELECTION:
104.252

SEM FEATURE SELECTION:
5.176.322

PRÉ-PROCESSAMENTO TEXTUAL

- Tokenização
- Tratamento de negações
- Utilização de n-gramas (n=1, 2 e 3)
- Seleção de features

Complexidade $O(w)$

w -> total de palavras nos documentos de treino

```
def nGramAndNegate(document):
    features = Counter()
    punctuation = "?.,!;:"
    words = nltk.tokenize.regexp_tokenize(document.lower(), regex)
    inNegation = False
    oneBefore = None
    twoBefore = None
    for word in words:
        if any([word.find(p) != -1 for p in punctuation]):
            inNegation = False
        else:
            contador_vocabulario.add(word)
            if inNegation:
                actual = "not_" + word
            else:
                actual = word

            features[actual] += 1

            if oneBefore and n >= 2:
                bigram = oneBefore + " " + actual
                features[bigram] += 1

            if twoBefore and n >= 3:
                trigram = twoBefore + " " + bigram
                features[trigram] += 1

            twoBefore = oneBefore
            oneBefore = actual

        if any(neg in ["not", "n't", "no"] for neg in word):
            inNegation = not inNegation

    return features
```

```
negated := False
for each word in document:
    if negated = True:
        Transform word to "not_" + word.
    if word is "not" or "n't":
        negated := not negated
    if a punctuation mark is encountered
        negated := False.
```

PRÉ-PROCESSAMENTO TEXTUAL

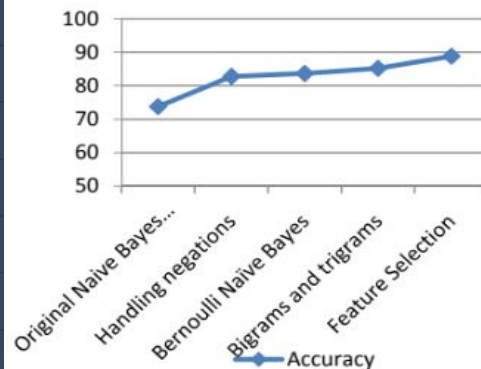
- Tokenização
- Tratamento de negações
- Utilização de n-gramas (n=1, 2 e 3)
- Seleção de features

Complexidade $O(w)$

w -> total de palavras nos documentos de treino

Table 1. Results Timeline

Feature Added	Accuracy on test set
Original Naive Bayes algorithm with Laplacian Smoothing	73.77%
Handling negations	82.80%
Bernoulli Naive Bayes	83.66%
Bigrams and trigrams	85.20%
Feature Selection	88.80%



PRÉ-PROCESSAMENTO TEXTUAL

- Seleção de features

Remoção das features que ocorreram apenas 1 vez:

- Maior velocidade
- Menor acurácia

Seleção das features mais relevantes com PMI

A decorative background graphic at the bottom of the slide. It features a white line chart with circular markers and a bar chart with blue bars of varying heights, all set against a dark blue grid background.

Relembrando um pouco de Naive Bayes...

$$P(c_i|d) = \frac{(\prod P(x_i|c_j)) * P(c_j)}{P(d)}$$

Relembrando Laplacian Smoothing...

$$P(x_i|c_j) = \frac{\text{Count}(x_i) + k}{(k + 1) * (\text{No of words in class } c_j)}$$

Complexidade do treinamento!

Complexidade $O(cf)$

- c é a quantidade de classes ($c = 2$)
- f é a quantidade de features ($f = \text{nº de n-gramas}$)

Complexidade do teste

Complexidade $O(w)$

- w é a quantidade de palavras nos testes



RESULTADOS

	Pré-processamento	Treinamento	Teste	Tempo total	Classe positiva	Classe negativa	Acurácia total
Sem feature selection	271,45	182,37	59,81	513,63	86,86%	89,58%	88,22%
Com feature selection	61,49	3,86	4,36	69,71	63,78%	78,03%	70,91%

Dúvidas

