Analise de Linguagem Natural no Twitter Voltado ao Sentimento

Amauri Melo Mendes Junior [1]

[1]Ciência da Computação, Aprendizado de Máquina – Universidade Federal do Tocantins (UFT) Palmas- Brazil

Resumo. *O trabalho consiste em aplicar técnicas de Aprendizado de Máquina, especificamente o algoritmo Naive Bayes, para identificar se um email é um spam ou não. O algoritmo Naive Bayes é uma abordagem estatística baseada no teorema de Bayes, que assume independência entre os recursos (palavras) de um documento. O conjunto de dados utilizado no trabalho é composto por emails previamente rotulados como spam ou não spam. O algoritmo Naive Bayes é treinado utilizando esses dados para aprender a probabilidade de um email ser spam ou não com base nas palavras contidas nele. Em seguida, o modelo treinado é utilizado para classificar novos emails como spam ou não spam com base nas probabilidades calculadas.*

1. Introdução

Este trabalho tem como objetivo a aplicação de técnicas de Aprendizado de Máquina na identificação de emails como spam ou não spam, utilizando o algoritmo Naive Bayes. Para isso, utilizamos um conjunto de dados disponível no site [Kaggle 2021], que consiste em emails previamente rotulados. O algoritmo Naive Bayes [sci 2007], baseado no teorema de Bayes e assumindo independência entre as palavras dos documentos, foi treinado com esse conjunto de dados para aprender a probabilidade de um email ser classificado como spam ou não spam. Com o modelo treinado, é possível classificar novos emails de forma eficiente e precisa, melhorando o processo de filtragem e reduzindo a presença de emails indesejados na caixa de entrada.

A utilização do conjunto de dados disponibilizado pelo Kaggle traz uma contribuição significativa para o desenvolvimento deste trabalho. A diversidade dos emails presentes no conjunto de dados permite que o modelo seja treinado com exemplos reais e representativos, abrangendo diferentes tipos de spam e não spam. Além disso, a disponibilidade desse conjunto de dados facilita o processo de treinamento e validação do algoritmo Naive Bayes, proporcionando resultados confiáveis e comparáveis com outros estudos na área. Com base nesses dados e com a aplicação do algoritmo Naive Bayes, esperamos obter uma solução eficaz na identificação de emails indesejados, contribuindo para uma melhor experiência de uso e segurança na comunicação por email.

2. Linguagem

- Python

2.1. Bibliotecas

- sklearn - Naive Bayes
- Pandas
- matplotlib

3. Algoritmo

O algoritmo começa importando as bibliotecas necessárias, como o Pandas para manipulação de dados, o Matplotlib para criação de gráficos e as classes e funções do Scikit-learn para o pré-processamento, treinamento e avaliação do modelo.

3.1. Leitura e Divisão dos dados

Em seguida, os dados de treinamento são carregados a partir de um arquivo CSV utilizando o Pandas. Esses dados consistem em mensagens de e-mail e seus respectivos rótulos (spam ou não spam).

Os dados são separados em duas partes: as mensagens de e-mail (features) e os rótulos correspondentes (spam ou não spam). Essa separação é necessária para o treinamento do modelo.

3.1.1. Treinamento

Os dados são divididos em um conjunto de treinamento e um conjunto de teste, utilizando a função traintestsplit do Scikit-learn. Essa divisão permite avaliar o desempenho do modelo em dados não vistos durante o treinamento.

Em seguida, é realizado o pré-processamento dos dados. É utilizado o objeto CountVectorizer do Scikit-learn para vetorizar as mensagens de e-mail, ou seja, transformar o texto em uma representação numérica que possa ser utilizada pelo modelo.

O modelo Naive Bayes Multinomial é criado utilizando a classe MultinomialNB do Scikit-learn. Esse modelo é adequado para tarefas de classificação de texto, como a detecção de spam.

O modelo é treinado utilizando as features vetorizadas e os rótulos do conjunto de treinamento. Ele aprende a relação entre as características das mensagens de e-mail e suas classes (spam ou não spam).

3.1.2. Pós-processamento

Após o treinamento, o modelo é avaliado utilizando o conjunto de teste. As mensagens de e-mail do conjunto de teste também são vetorizadas usando o mesmo objeto CountVectorizer utilizado no treinamento.

As previsões do modelo são comparadas com os rótulos verdadeiros do conjunto de teste, e a acurácia do modelo é calculada utilizando a função accuracyscore do Scikit-learn.

3.1.3. Dados e gráficos dos processos

Além disso, o algoritmo gera gráficos para acompanhar o desempenho do modelo durante o treinamento. Um gráfico de linha é criado com a acurácia do treinamento e do teste em cada época (epoch) do treinamento. Isso permite visualizar como o desempenho do modelo evolui ao longo das épocas.

Também é criada uma matriz de confusão para visualizar os resultados das previsões do modelo em relação aos rótulos verdadeiros do conjunto de teste. Essa matriz mostra quantas amostras foram classificadas corretamente e quantas foram classificadas erroneamente para cada classe (spam ou não spam).

3.1.4. Previsão

Por fim, o algoritmo realiza a previsão de uma mensagem de teste, que pode ser fornecida manualmente. A mensagem de teste é pré-processada e vetorizada da mesma forma que as mensagens de treinamento. Em seguida, o modelo é utilizado para prever a classe da mensagem e calcular as probabilidades de ser spam ou não spam.

3.2. Resultados

Durante cada época, o modelo ajusta seus pesos e parâmetros com base nos exemplos de treinamento, na tentativa de otimizar seu desempenho. [git 2023]

```
(env) PS C:\Users\amaur\OneDrive\Documentos\Apren_Maquina> python .\kkk.py
Epoch 1 - Acurácia de treinamento: 0.9869439071566731 Acurácia de teste: 0.9719806763285024
Epoch 2 - Acurácia de treinamento: 0.9886363636363636 Acurácia de teste: 0.9768115942028985
Epoch 3 - Acurácia de treinamento: 0.9898452611218569 Acurácia de teste: 0.9768115942028985
Epoch 4 - Acurácia de treinamento: 0.9905705996131529 Acurácia de teste: 0.9758454106280193
Epoch 5 - Acurácia de treinamento: 0.9905705996131529 Acurácia de teste: 0.9758454106280193
Epoch 6 - Acurácia de treinamento: 0.9908123791102514 Acurácia de teste: 0.9758454106280193
Epoch 7 - Acurácia de treinamento: 0.9915377176015474 Acurácia de teste: 0.9758454106280193
Epoch 8 - Acurácia de treinamento: 0.9915377176015474 Acurácia de teste: 0.9758454106280193
Epoch 9 - Acurácia de treinamento: 0.9915377176015474 Acurácia de teste: 0.9748792270531401
Epoch 10 - Acurácia de treinamento: 0.9915377176015474 Acurácia de teste: 0.9748792270531401
Previsão: não spam
Chances de ser spam: 0.29158607350096727
Chances de não ser spam: 0.708413926499032
```

Figura 1. Resultados no terminal - Épocas e Previsões.

Representação da taxa de acertos do modelo na classificação de mensagens de e-mail como spam ou não spam.

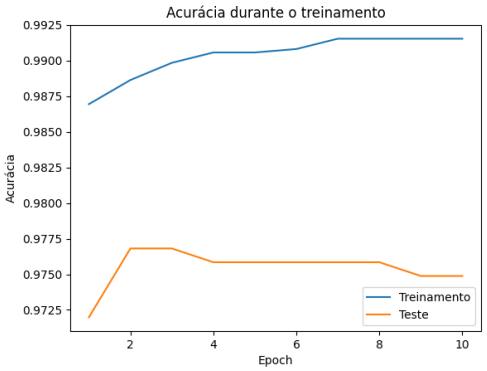

Figura 2. Acurácia

A matriz de confusão é uma tabela com duas linhas e duas colunas (para um problema de classificação binária). Cada linha representa uma classe verdadeira (spam ou não spam), e cada coluna representa a classe prevista pelo modelo. Os elementos da matriz

representam o número de exemplos classificados em cada combinação de classe verdadeira e classe prevista. No caso, podemos observar sobre a validação da nossa entrada, apontando como possível não spam.

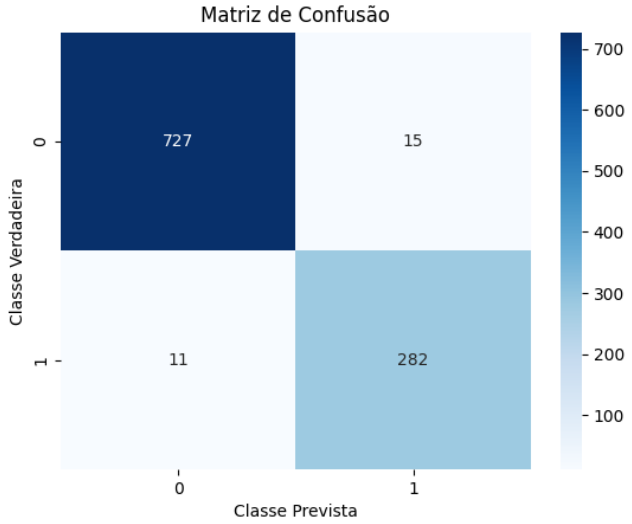

Figura 3. Matriz de Confusão

4. Conclusões

Em suma, a realização deste trabalho nos proporcionou um conhecimento aprofundado sobre o uso do algoritmo Naive Bayes para a classificação de emails como spam ou não spam. Ao explorar o conjunto de dados obtido no Kaggle, adquirimos experiência prática na construção e treinamento de modelos de Aprendizado de Máquina. Além disso, percebemos a relevância dessa aplicação na vida real, onde a identificação precisa de emails indesejados pode melhorar a eficiência, segurança e produtividade das comunicações.

A partir desse aprendizado, reconhecemos o potencial do algoritmo Naive Bayes e do Aprendizado de Máquina em diversos contextos. A classificação de emails é apenas uma das muitas aplicações possíveis. Essas técnicas podem ser utilizadas em áreas como filtragem de conteúdo em redes sociais, detecção de fraudes financeiras, análise de sentimentos e muito mais. Através desse trabalho, compreendemos o impacto positivo que a aplicação do Aprendizado de Máquina pode ter na resolução de problemas do mundo real e a importância contínua do desenvolvimento e aprimoramento dessas técnicas para enfrentar desafios futuros.

Referências

[sci 2007] (2007). scikit-learn.

[git 2023] (2023). Github, amaur1mmj.

[Kaggle 2021] Kaggle, V. G. (2021). Spam mails dataset.