Relatório de MPEI - Sistema de Análise de URLs

Guilherme Matos [114252], Rafael Dias [114258]

15 de dezembro de 2024

Conteúdo

1 Introdução

Este relatório apresenta o desenvolvimento de um sistema de análise de URLs que combina técnicas classificação e deteção - Filtros de Bloom, *Naive Bayes* e *MinHash*. O dataset utilizado foi retirado do Kaggle e contém URLs classificadas como *safe* ou *unsafe*.

2 Descrição do Sistema

O sistema desenvolvido tem como objetivos principais:

- Verificar rapidamente, através do Filtro de Bloom, se o URL pertence à lista de URLs *unsafe* conhecida.
- Avaliar um URL, classificando-o como *safe* ou *unsafe*, a partir do modelo Naive Bayes, treinado com um dataset de URLs já classificados.
- Identificar URLs semelhantes a um dado URL, procurando variações fraudulentas de outro URL *unsafe* já conhecido/existente.

3 Implementação

3.1 *Naive Bayes*

Funcionamento

O módulo *Naive Bayes* utiliza as palavras extraídas dos URLs como características principais. Estas palavras são processadas para formar um conjunto único de tokens (i.e., `secure`, `login`), que são usados para treinar o classificador. Durante o treinamento, o modelo calcula a probabilidade de cada token aparecer em URLs associados às classes *safe* e *unsafe*. Para a classificação, as palavras de um URL são analisadas, e as probabilidade acumulada determina se o URL será classificado como *safe* ou *unsafe*.

Os testes podem ser feitos dando "Run"ao programa de teste - "NaiveBayesTest.m"

Resultados

Os testes individuais do módulo foram realizados selecionando 100 URLs aleatórios para teste, sendo os restantes utilizados para treinar o modelo *Naive Bayes*. A precisão do modelo esteve sempre por volta de 0.9, classificando corretamente a maioria dos URLs como *safe* ou *unsafe*. No entanto, nem sempre foi o caso. Por exemplo, um caso observado foi:

- **URL 86:** Classe Real = *unsafe*, Classe Predita = *safe*

Estas imprecisões podem ser atribuídas à sobreposição de tokens entre as classes e de certo, ao dataset utilizado.

Durante o teste onde está inserido o erro anterior, estas foram as métricas obtidas.

- **Precision (Precision):** 0.87
- **Recall (Recall):** 1.00

- **F1-Score:** 0.93

Mesmo com estes resultados a indicar que o modelo têm boa precisão, melhorias podem ser feitas para reduzir os erros, especialmente nos casos de URLs que compartilham características semelhantes entre as duas classes. Poderia também ser escolhido um método de divisão dos URLs diferente.

3.2 Filtro de Bloom

Configuração

O filtro foi configurado com os seguintes parâmetros:

- **Tamanho do filtro (n)**: 200.000 bits. Este valor foi escolhido considerando o tamanho do conjunto de URLs inseguros e a necessidade de minimizar a probabilidade de falsos positivos, calculado através de uma espressão nos slides das aulas.
- **Número de funções de hash (k)**: 8. O número de funções de hash foi determinado para balancear a eficiência e a probabilidade de falsos positivos, calculado através de uma espressão nos slides das aulas.

Funcionamento

A implementação do Filtro de Bloom está encapsulada numa classe chamada `BloomFilter`, que oferece três métodos principais:

- **Inserção (`insert`)**: Adiciona URLs inseguros ao filtro calculando múltiplos valores de hash e definindo as posições correspondentes no vetor booleano como `true`, Apenas os URLs classificados como *unsafe* foram inseridos no filtro..
- **Verificação (`check`)**: Determina se um URL está presente no filtro verificando todas as posições indicadas pelas funções de hash.
- **Hashing (`createHash`)**: Gera múltiplos valores de hash para um URL, utilizando variações da string original.

Resultados

Os testes foram realizados selecionando 50 URLs aleatórios do dataset e verificando as suas classificações no filtro de Bloom.

Resultados Obtidos Para testar, é só alterar o URL para o que quer testar. Após um teste em especifico, obtivemosos seguintes valores:

- Total de URLs testadas: 50
- URLs presentes no Bloom Filter: 28
  - Verdadeiros Positivos: 28
  - Falsos Positivos: 0
- URLs não presentes no Bloom Filter: 22

Análise dos Resultados

Vantagens

- A verificação é extremamente rápida devido à eficiência do Bloom Filter.
- Reduz a carga computacional em sistemas maiores, pois URLs conhecidos são processados rapidamente.

Limitações

- A existência de falsos positivos é inerente à natureza do filtro. URLs seguros podem ser classificadas como *unsafe*, o que pode causar alertas desnecessários.
- A precisão depende fortemente do tamanho do filtro e do número de funções de hash escolhidas.

3.3 *MinHash*

Configuração

O módulo de *MinHash* foi configurado com os seguintes parâmetros:

- **Número de funções hash (k)**: 100. Este número determina a dimensão da assinatura MinHash para cada URL. este valor foi escolhido para uma maior precisão.
- **Tamanho dos *shingles***: 15. Cada URL foi decomposta em substrings de 15 caracteres consecutivos para gerar os conjuntos de *shingles*, este valor foi escolhido para uma maior precisão.
- **Funções de hash**: Para cada função hash, foram gerados valores aleatórios a, b e p (um número primo). Os valores hash foram calculados como:

$$h(x) = (a \cdot x + b) \mod p$$

onde x é o valor hash da substring, esta foi a expressãp usada na aula prática.

Implementação

A implementação foi encapsulada numa classe chamada `MinHash`, com os seguintes métodos principais:

- `generateShingles`: Gera os *shingles* únicos para uma string de entrada.
- `createHash`: Calcula valores hash para um conjunto de *shingles* com base em parâmetros específicos (a, b, p).
- `computeMinHashMatrix`: Gera a matriz de assinaturas MinHash para um conjunto de URLs.
- `computeSimilarities`: Calcula a similaridade entre a assinatura MinHash de uma URL de entrada e as assinaturas do dataset.

Resultados

Resultados Obtidos

- **URL de Entrada**: `http://wpengine.com/2015/01/21/icymi-handling-peak-capacity-sp`
- **URL Mais Semelhante**: `http://wpengine.com/2015/01/21/icymi-handling-peak-capac`
- **Classe da URL Mais Semelhante**: Unsafe
- **Similaridade Calculada**: 92%

Análise dos Resultados

Para testar, é só alterar o URL para o que quer testar.

Vantagens

- O método é eficiente para grandes conjuntos de dados, pois a assinatura MinHash reduz significativamente a complexidade de comparação.
- É ideal para detecção de typosquats ou URLs maliciosas que sejam pequenas variações de URLs legítimas.
- A similaridade calculada reflete bem as características estruturais dos URLs.

Limitações

- URLs com diferenças estruturais muito grandes (e.g., URL completamente diferente) não são capturados, mesmo que possam ser maliciosos.
- A precisão depende da escolha de parâmetros como o tamanho dos *shingles* e o número de funções hash.

3.4 Sistema Integrado

Funcionamento

O sistema realiza a análise de um URL em três etapas:

1. **Verificação com *Bloom Filter***: URLs conhecidos como maliciosos são verificadas rapidamente no filtro de Bloom. Se identificados, são classificados como *unsafe*.
2. **Classificação com *Naive Bayes***: Caso o URL não esteja no filtro, é aplicada uma classificação probabilística baseada no conteúdo do URL.
3. **Similaridade com *MinHash***: URLs restantes são comparados ao dataset, identificando variações similares (*typosquats*) com base em assinaturas MinHash.

Conclusão

O sistema integrado mostrou-se eficaz ao combinar velocidade (via *Bloom Filter*) e precisão (via *Naive Bayes* e *MinHash*). Sua principal limitação é a dependência de dados previamente conhecidos para o *Bloom Filter* e a sensibilidade dos parâmetros nos outros módulos.