Relatório Projeto MPEI

Dinis Cunha 119316
Henrique Lopes 119954

Dezembro, 2024

1 Introdução

No presente projeto, implementado em MATLAB, desenvolvemos uma aplicação que combina diversas técnicas probabilísticas para análise e categorização de texto, bem como identificação de padrões de similaridade. O objetivo principal foi processar mensagens de texto, associá-las a categorias específicas e recomendar ações baseadas nos resultados, utilizando algoritmos como:

- **Bloom Filter**: usado para otimizar a deteção de mensagens de spam e filtragem eficiente de conteúdo indesejado.
- **Classificador Naive Bayes**: usado para classificar mensagens em categorias pré-definidas, como "Inem" (I), "Bombeiros" (B), e "Polícia" (P).
- **MinHash**: usado para estimar a similaridade entre frases e identificar mensagens semelhantes com base na Similaridade de Jaccard e atribuir recomendações.

2 Vantagens e Limitações das Técnicas Implementadas

Nesta seção, apresentamos as vantagens e limitações das técnicas utilizadas no projeto: *Bloom Filter*, *Naive Bayes*, e *MinHash*, considerando o seu contexto de aplicação no tratamento de SPAM, classificação de frases e análise de similaridade.

Table 1: Vantagens dos Modelos Implementados

Modelo	**Vantagens**
Bloom Filter	Detecção rápida e eficiente de SPAM; Reduz o espaço de armazenamento; Inserção e procura rápidas; Probabilidade de falsos positivos ajustável através da configuração dos parâmetros.
Naive Bayes	Classificação eficiente em categorias (**I** - INEM, **B** - Bombeiros, **P** - Polícia); Simples de implementar e com baixo custo computacional; Funciona bem mesmo em datasets pequenos; Lida com grandes volumes de texto após pré-processamento.
MinHash	Identificação eficiente de frases similares com estimativa de Jaccard; Reduz o tempo de cálculo em comparação com métodos exatos; Agrupamento eficiente de frases por similaridade; Escalável para grandes conjuntos de dados.

Table 2: Limitações dos Modelos Implementados

Modelo	**Limitações**
Bloom Filter	Possibilidade de falsos positivos; Não suporta remoção exata de elementos; Configuração dependente dos parâmetros (k, n); Não elimina falsos negativos completamente.
Naive Bayes	Sensível ao vocabulário de treino (palavras ausentes prejudicam o desempenho); Supõe independência entre palavras, o que nem sempre é realista; Afetado por desequilíbrio nas categorias do treino; Frases curtas limitam a robustez da classificação.
MinHash	Resultados aproximados que podem introduzir erro; Depende dos parâmetros (número de hashes e tamanho dos *shingles*); Frases curtas dificultam a geração eficaz de *shingles*; Maior número de funções hash aumenta o custo computacional.

2.1 Filtro Bloom: Deteção de SPAM

O projeto inicia-se com a implementação do Filtro Bloom para identificar chamadas ou mensagens classificadas como SPAM. Este filtro é utilizado como uma estrutura eficiente para verificar rapidamente se uma frase ou número de telefone pertence à lista negra (conjunto de SPAM).

Dataset Histórico: Antes de haver a implementação do Filtro Bloom, é verificado se o número d chamda tem historial de SPAM (Lista Negra).O sistema utiliza um conjunto de chamadas antigas consideradas como SPAM. Este dataset é processado, e as frases associadas são inseridas no Filtro Bloom para compor a estrutura inicial.

Verificação de SPAM: Durante o funcionamento, o Filtro Bloom compara as novas entradas (frases e números) com os dados armazenados. Caso haja um match, a frase é marcada como SPAM.

Evitar Processamento Desnecessário: Chamadas classificadas como SPAM pelo Filtro Bloom são ignoradas pelo classificador Naive Bayes, evitando o processamento desnecessário. Isso reduz o tempo de execução e otimiza os recursos do sistema.

Atualização Contínua da Lista Negra: Novos números e frases identificados como SPAM durante os testes são adicionados a uma lista negra, que é armazenada e utilizada para futuras verificações.

2.1.1 Testes e Resultados

Os testes focaram-se em medir a eficiência do Filtro Bloom, nomeadamente a taxa de falsos positivos (FP). A implementação envolveu os seguintes parâmetros:

Tamanho do Filtro (N): Determinado com base no número de entradas (frases e números de SPAM) e na probabilidade teórica de falsos positivos. **Número de Funções Hash (k):** Calculado para minimizar a taxa de falsos positivos. **Probabilidade Teórica de Falsos Positivos (P):**Avaliada em função dos parâmetros definidos.

A tabela abaixo resume as métricas obtidas durante os testes:

Table 3: Métricas do Bloom Filter

Métrica	**Valor Obtido**
Tamanho do Filtro (N)	2962
Número de Funções Hash (k)	10
Probabilidade Teórica (P)	0.001
Contagem de FP Prática	1
Taxa Prática de FP (%)	0.001418

Análise dos Resultados: Os testes confirmaram que o Filtro Bloom é altamente eficiente na identificação de SPAM, apresentando uma baixa taxa prática de falsos positivos próxima à teórica. Contudo, é importante destacar que o Filtro Bloom introduz a possibilidade de falsos positivos, ou seja, mensagens válidas podem ser classificadas incorretamente como SPAM.

2.2 Classificação com Naive Bayes

Após o pré-processamento inicial com o Filtro Bloom, que filtra chamadas consideradas SPAM, o classificador Naive Bayes recebe apenas as chamadas válidas. O objetivo é determinar a categoria apropriada para cada chamada, encaminhando-a para a Polícia (P), INEM (I) ou Bombeiros (B).

Fizemos um pré processamento do dados, as frases são tokenizadas, convertidas para minúsculas, e são removidas as stopwords e carateres desnecessários, garantindo um texto limpo e estruturado.

De seguida, houve uma divisão de dados, o dataste é dividido em treino (60%) e teste (40%) para treinar e validar o modelo.

O Naive Bayes calcula a probabilidade de cada frase pertencer a uma das categorias (**I**, **B** ou **P**) com base nas palavras presentes. Cada frase é automaticamente associada à categoria com maior probabilidade posterior, permitindo o encaminhamento rápido e eficiente das chamadas.

2.2.1 Testes Realizados

Foram implementados dois testes principais para validar e demonstrar o funcionamento do modelo Naive Bayes:

1. **Teste de Eficiência:** Neste teste, avaliamos a eficiência do modelo ao calcular a probabilidade de cada palavra na frase pertencer a uma das categorias (**I**, **B** ou **P**). As probabilidades individuais foram combinadas para determinar a categoria final da frase como um todo. A precisão global foi validada através de métricas como *accuracy* e *F1-score*, permitindo verificar o desempenho e robustez do modelo. Para isso, criamos o script Teste_NaiveBayes, que nos testa as funções.

```
>> Teste_NaiveBayes
Métricas por Classe - Naive Bayes:
    Categoria    Precisao    Recall     F1Score
    _________    ________    _______    _______

       "P"       0.96053     0.92797    0.94397
       "B"       0.89513     0.93359    0.91396
       "I"       0.90476     0.89202    0.89835

Precisão Global: 91.91%
```

Figure 1: Percentagem de eficiência do modelo Naive Bayes.

2. **Classificação com Input do Utilizador:** Implementámos um sistema onde o utilizador pode fornecer diretamente uma frase (simulando uma chamada para o 112). O modelo Naive Bayes processa a frase, calcula as probabilidades e determina automaticamente a categoria mais provável (**I**, **B** ou **P**). Este teste simula uma aplicação em tempo real, onde o modelo classifica novas chamadas e encaminha rapidamente para a entidade correta, implementado no script main.

```
>> main
Deseja usar frases customizadas e números de telefone? (s/n): s
Digite uma frase (ou pressione Enter para terminar): someone stole my phone
Digite o número de telefone associado: 963856234
Digite uma frase (ou pressione Enter para terminar):
Frase: "someone stole phone" -> Categoria prevista: P
```

Figure 2: Teste com Input do utilizador.

2.3 Resultados de Desempenho do Naive Bayes

Para avaliar o desempenho do modelo Naive Bayes, geramos duas representações principais: um gráfico com as métricas **Precisão** (relação entre os verdadeiros positivos e os valores previstos como positivos.), **Recall** (relação entre os verdadeiros positivos e os valores reais.) e **F1-score** (Combina precisão e recall em uma média harmônica..) e uma **Matriz de Confusão**, detalha as classificações corretas e incorretas do modelo, mostrando as correspondências entre classes reais (verdadeiras) e classes previstas.

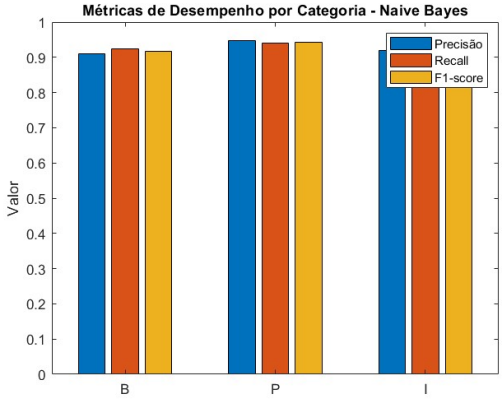

Figure 3: Gráfico de Precisão, Recall e F1-Score para cada categoria.

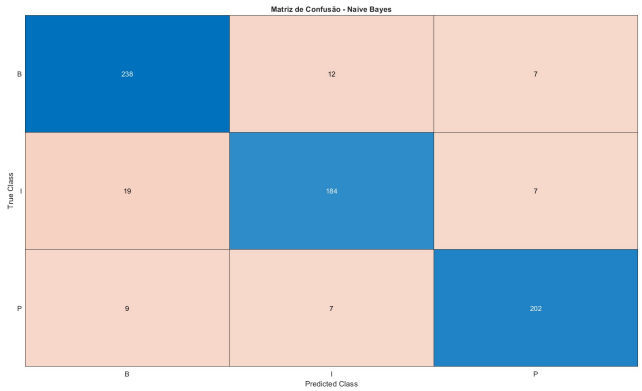

Figure 4: Matriz de Confusão do modelo Naive Bayes.

O gráfico à esquerda (Figure 3) apresenta as métricas de desempenho para cada categoria: **B** (Bombeiros), **P** (Polícia) e **I** (INEM). As três métricas (Precisão, Recall e F1-Score) demonstram que o modelo possui desempenho equilibrado em todas as classes.

Já a matriz de confusão (Figure 4) mostra que a maioria das classificações são corretas (representadas pela diagonal principal). As confusões fora da diagonal são mínimas, indicando que o modelo raramente erra a categorização das frases.

2.4 Identificação de Similaridade com MinHash

O MinHash foi implementado para identificar a similaridade entre frases após a classificação realizada pelo Naive Bayes. O objetivo principal é agrupar frases semelhantes e fornecer recomendações específicas dependendo da categoria (**INEM (I)**, **Bombeiros (B)**, ou **Polícia (P)**).

Geração de Shingles: Cada frase é processada para gerar *shingles* de tamanho 4 (sequências de 4 caracteres consecutivos), que servem como base para a comparação.

Assinaturas MinHash: As assinaturas MinHash são geradas a partir dos *shingles* utilizando múltiplas funções de hash. Isso permite uma estimativa eficiente da similaridade de Jaccard entre frases.

Cálculo de Similaridade: A similaridade entre as frases de teste e treino é calculada comparando as assinaturas MinHash geradas. Um limiar de similaridade (*threshold*) é definido para considerar frases como semelhantes.

2.4.1 Testes Realizados

O MinHash foi aplicado às frases classificadas pelo Naive Bayes, com dois objetivos principais, identificação de similaridade entre frases e recomendações baseadas na categoria.

Os testes realizados no script testeMinHash permitem calcular a similaridade média e a mediana da similaridade, assim como um gráfico que nos mostra a distribuição das similaridades.

```
>> testeMinHash
Similaridade Média: 0.28
Mediana da Similaridade: 0.25
```

Figure 5: Cálculo da média e mediana das similaridades.

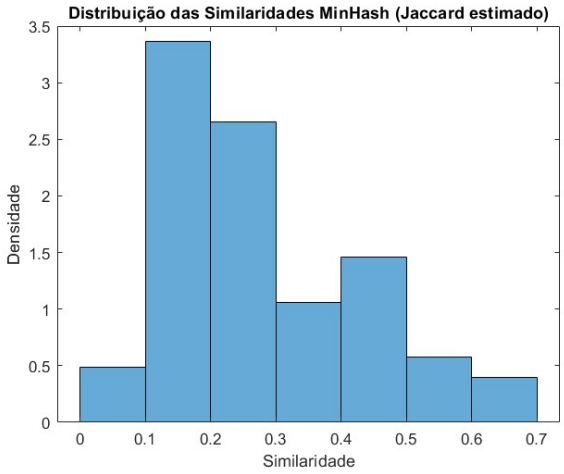

Figure 6: Distribuição das similaridades MinHash.

3 Implementação Conjunta

Para integrar as soluções desenvolvidas — Filtro Bloom, Naive Bayes e MinHash — implementámos uma aplicação conjunta que permite processar, classificar e recomendar ações com base nas frases fornecidas pelo utilizador.

A aplicação foi testada no script `main`, onde desenvolvemos um menu interativo que permite ao utilizador inserir as frases ou chamadas simuladas. O fluxo principal da aplicação é o seguinte:

1. Filtragem Inicial com Filtro Bloom: O sistema verifica se a frase ou número inserido pertence à lista negra de SPAM. Caso pertença, é descartado.

2. Classificação com Naive Bayes: As frases não identificadas como SPAM são classificadas em três categorias: **INEM (I)**, **Bombeiros (B)** e **Polícia (P)**.

3. Identificação de Similaridade com MinHash: Após a classificação, o sistema calcula a similaridade entre as frases fornecidas e as frases de treino, agrupando mensagens semelhantes.

4. Recomendações Automáticas: Dependendo da categoria e da similaridade, são apresentadas ao utilizador recomendações específicas associadas à categoria prevista.

O menu principal permite ao utilizador visualizar os resultados passo a passo, facilitando o teste e a demonstração do sistema implementado.

```
>> main
Deseja usar frases customizadas e números de telefone? (s/n): s
Digite uma frase (ou pressione Enter para terminar): someone stole my phone
Digite o número de telefone associado: 963487294
Digite uma frase (ou pressione Enter para terminar):
Frase: "someone stole phone" -> Categoria prevista: P
Digite o valor do threshold de similaridade (ex: 0.60): 0.0
Threshold de similaridade definido como: 0.00
Deseja ver todas as frases de teste ou apenas as similares? (0/1): 1
Frase de teste: "someone stole phone"
Frase semelhante no treino: "someone stole phone shopping "
Similaridade estimada de Jaccard: 0.63

Número de frases similares: 1
Percentagem de frases similares: 100.00%
```

Figure 7: Testes da aplicação conjunta.

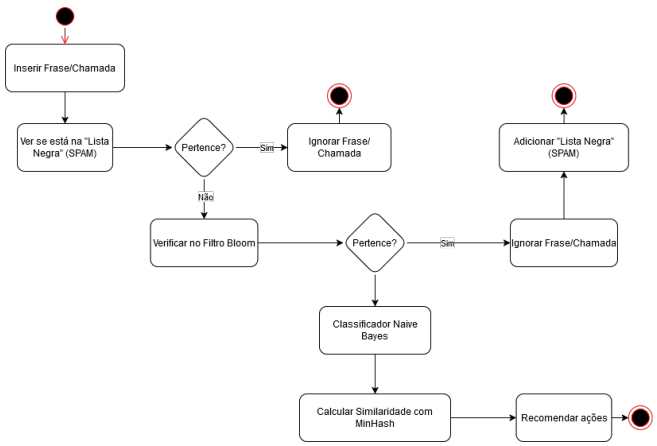

Figure 8: Diagrama fluxo conjunto.