

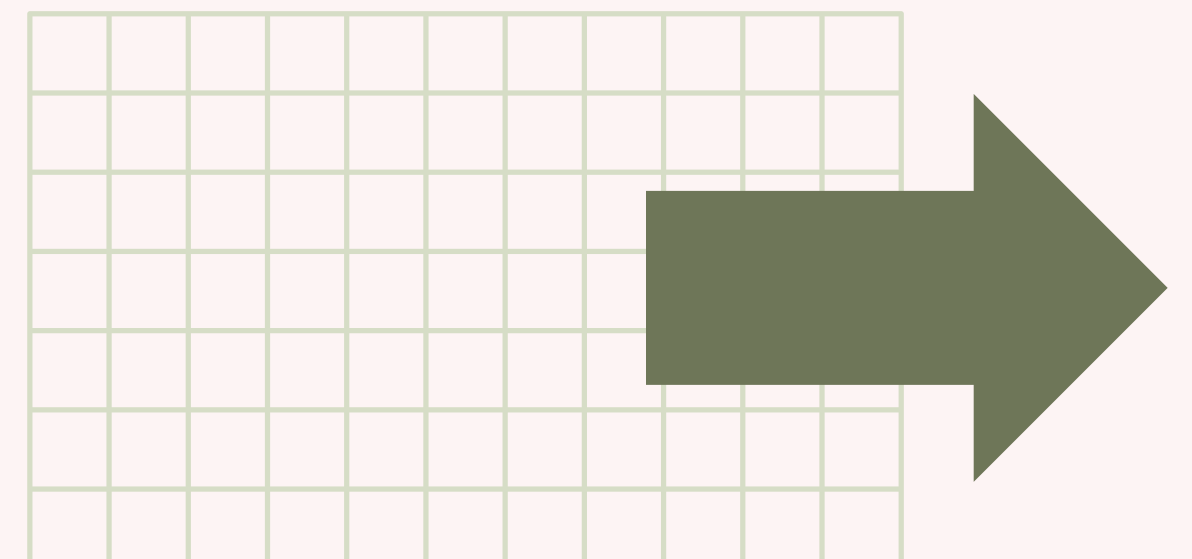


Classificador chamadas 112

Projeto MPEI

Dinis Cunha - 119316

Henrique Lopes - 119954



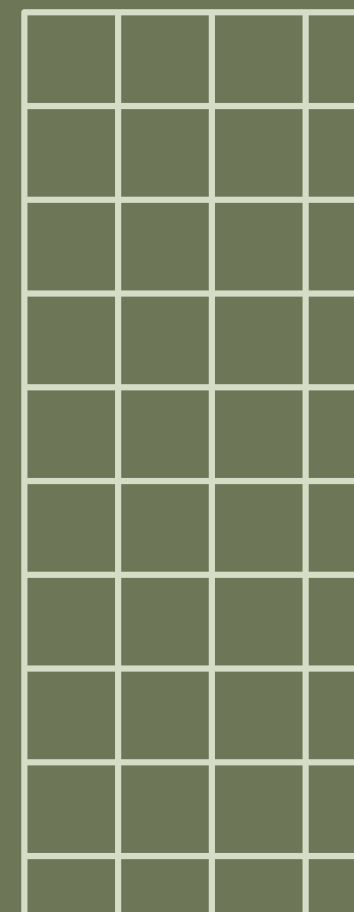
Introdução ao Tema

Problema: Classificar chamadas feitas ao número de emergência 112 para decidir se devem ser encaminhadas para:

- I: INEM (Serviços Médicos de Emergência)
- B: Bombeiros
- P: Polícia

Objetivo: Criar soluções eficientes para classificar e organizar as chamadas usando três métodos:

- **Naive Bayes** para classificação.
- **Bloom Filter** para legitimidade de chamadas.
- **Minhash** para encontrar similaridade entre chamadas.



Aplicação do Naive Bayes

Descrição:

- O Naive Bayes é um classificador probabilístico baseado no Teorema de Bayes.
- Ele assume independência entre as palavras em uma frase (chamada).

Aplicação ao Problema:

1. O modelo é treinado com um dataset de frases e as suas categorias: I, B, ou P e número de telemóvel.
2. Para uma nova frase, calcula-se:
 - $P(\text{categoria} \mid \text{frase}) = P(\text{categoria}) \prod P(\text{palavra} \mid \text{categoria})$
3. A frase é atribuída à categoria com maior probabilidade.

Exemplo:

- Frase: "Encontrada pessoa inconsciente no parque da Macaca".
- Classificação: INEM (I).


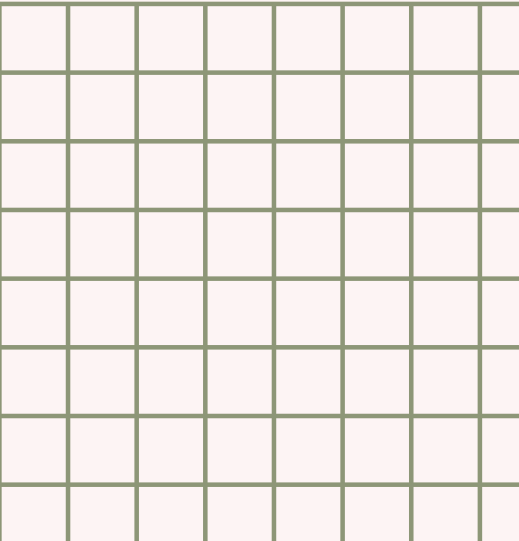




Etapas do processo de Naive Bayes



1. Processamento do Dataset

- Objetivo: Garantir que os dados estejam padronizados e prontos para o modelo.
 - Processos Realizados:
 - Leitura do Dataset: Carregamos as frases e suas categorias associadas (I, B, P), ignorando números de telefone (serão usados depois para o Bloom Filter).
 - Remoção das Linhas Duplicadas: Eliminamos frases repetidas juntamente com as suas categorias.
 - Processamento das frases:
 - Convertemos para minúsculas
 - Removemos pontuações, como pontos finais e vírgulas
 - Removemos Stop Words (como “the”, “is”, “a”, “and”, ...)
 - Divisão do Dataset para treino e teste:
 - 60% para treino: Para criar o modelo.
 - 40% para teste: Para avaliar o desempenho.
- 
- 

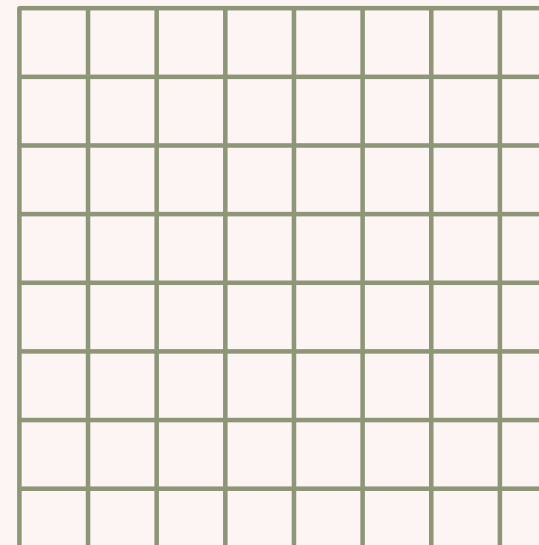


Etapas do processo de Naive Bayes



2. Criação da Matriz de Ocorrências

- Objetivo: Representar cada frase como um vetor de palavras no vocabulário.
- Processos Realizados:
 - Criação do Vocabulário (lista de palavras únicas):
 - Extraímos as palavras únicas presentes no conjunto de treino.
 - Eliminamos strings vazias e palavras irrelevantes.
 - Construção da Matriz Bag-of-Words:
 - Cada linha representa uma frase.
 - Cada coluna representa uma palavra no vocabulário.
 - Os valores indicam o número de vezes que cada palavra aparece na frase.



Etapas do processo de Naive Bayes

3. Cálculo das Probabilidades

- Objetivo: Usar as frequências das palavras para calcular probabilidades:

- Probabilidade de Categoria → $P(\text{categoria})$:

- Proporção de frases em cada categoria no conjunto de treino.

- Exemplo: $P(B) = \frac{\text{Número de frases Bombeiros}}{\text{Total de frases no treino}}$

- Probabilidade Condicional → $P(\text{palavra} \mid \text{categoria})$:

- Frequência de cada palavra em frases de uma categoria específica.
- Utilizamos suavização de Laplace para evitar probabilidades zero:

- $P(\text{palavra} \mid \text{categoria}) = \frac{\text{Contagem da palavra na categoria} + 1}{\text{Total de palavras na categoria} + \text{Número de palavras no vocabulário}}$

Etapas do processo de Naive Bayes

4. Classificação

- Objetivo: Determinar a categoria de uma nova frase (para onde deve ser reencaminhada).
- Processos Realizados:
 - Para cada categoria, calculamos a probabilidade posterior:
 - $P(\text{categoria} \mid \text{frase}) = P(\text{categoria}) \prod P(\text{palavra} \mid \text{categoria})$
 - Para evitar problemas de valores pequenos (underflow), usamos o logaritmo:
 - $\log P(\text{categoria} \mid \text{frase}) = \log P(\text{categoria}) + \sum \log P(\text{palavra} \mid \text{categoria})$
 - A frase é atribuída à categoria com maior probabilidade.

Aplicação do Bloom Filter

Descrição:

- Bloom Filter é uma estrutura de dados probabilística que é extremamente eficiente em termos de espaço e tempo, usada para verificar rapidamente se um item está ou não presente em um conjunto.
- A principal vantagem é que o Bloom Filter pode indicar com alta eficiência se um item já foi registrado antes, mas pode gerar falsos positivos (onde indica que um item está no conjunto, mesmo que não esteja).

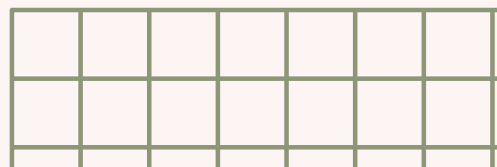
Aplicação ao problema:

- Detectar e Filtrar Chamadas Falsas: Quando uma chamada é registrada como falsa, usamos o Bloom Filter para garantir que chamadas subsequentes do mesmo número de telefone não precisem ser processadas novamente.
- Isso evita que o modelo Naive Bayes seja acionado para chamadas já identificadas como falsas, economizando tempo e recursos.

Etapas do processo do Bloom Filter

1. Processos envolvidos:

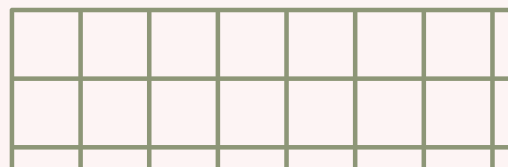
- Armazenamento de Chamadas Falsas no Bloom Filter:
 - Quando uma chamada (frase) é identificada como falsa (com base em padrões específicos, como frases sem sentido ou chamadas de brincadeira), a assinatura do número de telefone é registrada no Bloom Filter.
 - A assinatura do número de telefone é gerada usando uma função hash que cria uma representação compacta e única do número.
- Verificação Rápida de Chamadas:
 - Para cada nova chamada, antes de aplicar o Naive Bayes, o número de telefone da chamada é verificado no Bloom Filter.
 - Se o número já estiver presente no filtro, isso significa que a chamada foi previamente registrada como falsa.
 - Nesse caso, a chamada não será processada pelo Naive Bayes e poderá ser descartada ou tratada como de baixa prioridade.
 - Se o número não estiver no filtro, a chamada será encaminhada para o Naive Bayes para classificação.
- Eliminação de Chamadas Falsas:
 - Se a chamada for identificada como falsa, o seu número é adicionado ao Bloom Filter para futuras verificações.



Etapas do processo do Bloom Filter

1. Implementação :

- Preparação do Dataset:
 - Vamos garantir que os números de telefone são limpos e armazenados adequadamente para serem usados no Bloom Filter.
- Criação do Bloom Filter:
 - Usaremos a estrutura de dados Bloom Filter que pode armazenar os números de telefone de chamadas falsas.
 - A estrutura será inicializada com um tamanho adequado para o número esperado de entradas e uma quantidade razoável de funções de hash para garantir baixa taxa de falsos positivos.
- Implementação da Função de Hash:
 - Para garantir que o número de telefone seja representado unicamente, usaremos uma função de hash eficiente que converterá o número de telefone em um valor compacto para ser armazenado no Bloom Filter.
- Verificação:
 - Sempre que uma nova chamada chega, o número de telefone é passado pela função de hash.
 - O Bloom Filter verifica rapidamente se esse número já foi registrado como falso.
 - Se o número não estiver no filtro, a chamada é processada normalmente. Se estiver, é descartada ou tratada como não legítima.



Aplicação do MinHash

Como funciona:

- Cada chamada é transformada em um conjunto de palavras ou shingles (sequências de palavras ou caracteres).
 - Exemplo:
 - Chamada 1: "Fogo na floresta".
 - Chamada 2: "Incêndio na floresta".
 - Representação de conjuntos: $A = \{\text{"fogo"}, \text{"floresta"}\}$, $B = \{\text{"incêndio"}, \text{"floresta"}\}$

Geração de Assinaturas (Minhash);

- Para cada conjunto, geramos uma assinatura Minhash que representa compactamente o conjunto de palavras.
- O Minhash utiliza uma função de hash para gerar valores únicos para cada conjunto, de modo que conjuntos semelhantes terão assinaturas semelhantes.

Estimativa de similaridade:

- Assinaturas Minhash de dois conjuntos são comparadas para estimar a similaridade.
- Quanto mais elementos compartilhados em suas assinaturas, mais semelhantes são os conjuntos (ou chamadas).
- Isso permite que agrupemos chamadas semelhantes de forma eficiente.

Etapas do processo do MinHash

1. Aplicação ao problema:

- Agrupamento de Chamadas Similares:
 - Ao invés de tratar cada chamada como única, Minhash pode ser usado para agrupar chamadas que tratam do mesmo tipo de emergência ou que têm frases semelhantes.
 - Chamadas como:
 - "Incêndio na floresta".
 - "Fogo na floresta".
 - Serão agrupadas em um único grupo e não processadas separadamente.
- Eficiência:
 - Minhash é mais eficiente que comparar diretamente todas as palavras de cada frase, porque gera uma representação compacta (assinatura).
 - Com isso, economiza tempo computacional ao comparar chamadas semelhantes rapidamente.
- Integração com Outros Métodos:
 - Bloom Filter: Pode ser usado para verificar se uma chamada foi registrada como falsa antes, para evitar processamento.
 - Naive Bayes: Pode ser usado para classificar chamadas, enquanto Minhash agrupa chamadas semelhantes antes de serem passadas para o Naive Bayes.
 - Minhash ajuda a reduzir a quantidade de chamadas repetidas que passam pelo classificador.

