Criptografia de César e Substituição

Andrei Miguel Cristeli
Yann Eduardo de Souza SIlva

Resumo— O programa oferece uma ferramenta básica para a quebra da cifra de César e avaliação da qualidade da descriptografia. Isto é, o programa consegue ler um arquivo cifrado em formato binário, e descriptografar o texto para todas as 26 possíveis chaves da cifra de César. E avalia a qualidade da descriptografia usando um conjunto de quadgrams predefinidos, calculando a pontuação de cada.

I. INTRODUÇÃO

A Cifra de César é um dos exemplos clássicos da criptografia. Essa cifra é realizada através de um deslocamento do alfabeto. Cada letra tem uma correspondente fixa, e a chave é um deslocamento circular das letras do alfabeto original. Desse modo, o trabalho apresentado consiste em um código em linguagem C que tem como objetivo descriptografar mensagens cifras através da cifra de César. A importância desse trabalho reside no contexto da segurança da informação, onde a capacidade de descriptografar mensagens cifras é crucial para entender e obter informações de mensagens cifradas é crucial para entender e obter informações de mensagens confidenciais. Além disso, a implementação envolve o uso de pontuação de quadgrams para determinar a frequência de letras e palavras na língua inglesa. Quanto maior a pontuação da sequência, maior a probabilidade de que seja uma mensagem válida em inglês.

O programa recebe como entrada um texto cifrado representado em formato binário, onde cada letra é codificada em um byte de 8 bits e segue o alfabeto inglês maiúsculo. O programa descriptografar o texto para todas as 26 possíveis chaves da cifra César e avalia a qualidade da descriptografia usando um conjunto de quadgrams. O programa também inclui o cálculo da pontuação de cada descriptografia e imprime o resultado na saída padrão

II. REFERENCIAL TEÓRICO

A técnica de cifra de César é uma forma de cifragem por substituição, na qual cada letra do texto original é deslocada por um número fixo de posições no alfabeto. Essa técnica simples, atribuída ao famoso general romano Júlio César, é um exemplo clássico de segurança por obscuridade. No entanto, a cifra de César é vulnerável a ataques de força bruta devido à sua previsibilidade. Por isso, ela é raramente usada na criptografia moderna, mas continua sendo um excelente ponto de partida para entender os princípios fundamentais da criptografia.

Além disso, a eficácia da cifra de César pode ser aprimorada usando métodos estatísticos e análise de frequência. O código utiliza um arquivo contendo quadgrams e suas pontuações para determinar a probabilidade de uma sequência de letra está correta. Isso se baseia na frequência de ocorrência de quadgrams em texto em inglês. Quanto maior a

pontuação da sequência, maior a probabilidade de que seja uma mensagem válida. Essa abordagem combina técnicas históricas de criptografia com a análise estatística moderna para criar um sistema mais robusto.

Por outro lado, O “Books Ngram Viewer” nasceu em 2004, quando os pesquisadores da Universidade de Harvard, Jean-Baptiste Michel e Lieberman Aiden, começaram uma pesquisa sobre verbos irregulares no inglês. Os dois buscavam determinar quando formas verbais específicas deixaram de ser usadas em detrimento de outras, mais modernas. Na época, esse tipo de pesquisa era possível apenas manualmente: página por página, livro por livros. O processo todo lhes custou longos 18 meses. Pouco mais de um ano depois, Jean-Baptiste e Lieberman souberam dos planos do Google para digitalizar todos os livros do mundo. Os dois, então, entraram em contato com Peter Norvig, diretor de pesquisa do Google. Novig logo percebeu a importância daquela ideia para a ciência e deu carta branca para os desenvolvedores. O Books Ngram Viewer é a versão mais acabada desta ideia e utiliza 4% do banco de dados do Google Books. O Google Books Ngram Viewer utiliza um método de modelagem chamado N-gram, que possibilita buscas em sequências de linguagem natural.

III. METODOLOGIA

O código é projetado para descriptografar mensagens cifradas usando a cifra de César e uma técnica estatística baseada em quadgrams. Vamos explicar o funcionamento de cada função no código e o passo a passo do processo.

1. Função numero_de_quad(): Esta função é responsável por contar o número de quadgrams presentes em um arquivo de referência.

Figura 1 - Print do código

```c
int numero_de_quad(){//funcao responsavel por contar o numero de quadgramas presente no arquivo
//declaração das variaveis
    int num=0;
    int a;
    char q[5];
    FILE *arquivo=fopen("arquivo_de_ngrams.txt", "r");//abrindo o arquivo

    if (arquivo == NULL) {
        perror("Erro ao abrir arquivo");
        exit(EXIT_FAILURE);
    }

    while (fscanf(arquivo, "%5s %d", q, &a) != EOF) {
        num++;
    }
    fclose(arquivo);
    return num;
}
```

Autoria própria

- Abrir o arquivo de quadgrams.
- Inicializa um contador para o número de quadgrams
- Ler cada linha do arquivo de quadgrams e incrementar o contador para cada quadgram encontrado.
- Fechar o arquivo e retornar o número de quadgrams

2. Função coloca_quad(): Esta função lê os quadgrams e suas pontuações do arquivo de referência e os armazena em uma estrutura de de dados.

Figura 2-Print do código

```c
void coloca_quad(int n, Quadgrams *quad) {
    FILE *arq = fopen("arquivo_de_ngrams.txt", "r");

    if (arq == NULL) {
        perror("Erro ao abrir arquivo");
        exit(EXIT_FAILURE);
    }

    for (int i = 0; i < n; i++) {
        if (fscanf(arq, "%5s %d", quad[i].quadgrams, &quad[i].pontuacao) != 2) {
            fprintf(stderr, "Erro ao ler quadgramas do arquivo.\n");
            exit(EXIT_FAILURE);
        }
    }

    fclose(arq);
}
```

Autoria própria

- Abrir o arquivo de quadgrams.
- Inicializar um array de estruturas quadgrams.
- Ler cada linha do arquivo de quadgrams e armazenar o quadgram e sua pontuação na estrutura de dados.
- Fechar o arquivo.

3. Função Pontuar (): Esta função calcula a pontuação de uma sequência de quadgrams com base no arquivo de referência de quadgrams. A pontuação reflete a probabilidade de a sequência ser válida em inglês.

Figura 3- Print do código

```c
float pontuar(char *quad, Quadgrams *quadgrams, int n) {//funcao responsavel por calcular a
probabilidade das metricas
    float soma = 0.0;

    for(int i=0; i<n; i++){
        if (!strcmp(quadgrams[i].quadgrams, quad)) {
            soma = log((quadgrams[i].pontuacao * 1.0) / TOTAL);
            return soma;
        }
    }

    return 0.0;
}
```

Autoria própria

- Recebe como entrada a sequência de quadgrams a ser pontuada, o array de estruturas Quadgrams e o número de quadgrams.
- Inicialize uma variável soma para armazenar a pontuação.

- Iterar sobre o array de quadgrams, ou seja, percorrer cada elemento do array de quadgrams para compará-lo com a sequência de quadgrams de entrada e calcular a pontuação.
- Se encontrar um quadgram que corresponda à sequência de entrada, calcula a pontuação com base na frequência relativa no arquivo de referências.
- Retornar a pontuação.

Resumindo, na função pontuar (), a iteração sobre o array de quadgrams envolve um loop que percorre cada quadgram presente no array quadgrams. O objetivo é encontrar um quadgram que corresponda à sequência de quadgrams de entrada. A cada iteração, o código compara o quadgrams presente na variável quadgrams[i].quadgrams com o quadgrams de entrada (quad). Quando um quadgram correspondente é encontrado, a pontuação é calculada com base nas estatísticas de frequência relativa desse quadgram e é retornada como resultado. Se nenhum quadgram correspondente for encontrado, a função retorna 0.0.

4. Função decryption (): Esta função realiza a descriptografia da mensagem criada e determina a pontuação de cada descriptografia possível, selecionando a descriptografia com a pontuação mais alta.

Figura 4 - Print do código

```
void decryption(char texto_cifrado[], char cif_aux[], int n,double menor, Quadgrams *quadgrams) {
    int i, chave;
    char cifrado[500],pega_os_quad[5];//a variavel pega_os_quad, ele pega os quadgramas que estão no arquivo
    double pontuacao, soma = 0.0;
    int tam = strlen(texto_cifrado);


    for (chave = 1; chave < 26; chave++) {//loop responsavel em percorrer todo o alfabeto
        // Fazer uma cópia do texto cifrado em cada iteração
        strcpy(cifrado, texto_cifrado); // copiando a string na variavel cifrado

        for (i = 0; i < tam; i++) { //loop responsavel em percorrer a string inteira
            if (cifrado[i] >= 'A' && cifrado[i] <= 'Z') { //verifica se a string está entre 65 a 90, se estiver entra
            dentro do if
                cifrado[i] = cifrado[i] - chave;//dar um shift para trás, do x vai para o w
                if (cifrado[i] < 'A' && cifrado[i] != ' ') { // se for menor que 65 e diferente de 32, soma 26, para entrar
                dentro alfabeto
                    cifrado[i] = cifrado[i] + 26;
                }
            }
        }

        for (int x = 0; x + 4 <= tam; x++) {//loop para extrair sequencias de quadgrams da mensagem descriptografada
            for (int k = 0; k < 4; k++) {
                pega_os_quad[k] = cifrado[x + k];//Extrai a sequência de 4 letras
            }
            pega_os_quad[4] = '\0';//adiciona um terminador nulo para indicar o fim do quadgram
            float valor_quad = pontuar(pega_os_quad, quadgrams, n);//calcula a pontuação do quadgram
            soma += valor_quad;//acumula as pontuações dos quadgrams para a mensagem descriptografada
        }
```

Autoria própria

- Recebe como entrada a mensagem cifrada, um array auxiliar para armazenar a mensagem descriptografada, o número de quadgrams, o valor inicial da menor pontuação e o array de quadgrams.
- Iterar sobre todas as possíveis chaves de deslocamento na cifra de César (1 a 25).
- Para cada chave, realizar a descriptografia da mensagem cifrada.
- Calcular a pontuação da sequência de quadgrams na mensagem descriptografada usando a função pontuar().
- Comparar a pontuação atual com a menor pontuação registrada até agora.

- Se a pontuação atual for maior do que a menor pontuação, atualizará a menor pontuação e armazenará a mensagem descriptografada correspondente.
- Após iterar por todas as chaves, imprimir a mensagem descriptografada com a pontuação mais alta.

5. Função main(): A função principal do programa, onde a execução começa.

- Ler uma mensagem cifrada em binário a partir de um arquivo.
- Converter o binário em caracteres alfabéticos (A-Z).
- Realizar a descriptografia com a função decryption().
- Alocar a memória para o array de quadgrams e preenchê-lo com os dados do arquivo de referência.
- Exibir a mensagem descriptografada com a maior pontuação.

IV. RESULTADO

+Os resultados obtidos com o código incluem a descriptografia da mensagem cifrada e a pontuação associada a cada tentativa. O código seleciona a descriptografia com a pontuação mais alta e a exibe como resultado final juntamente com a pontuação.O resultado final,foi a frase descriptografada: “IT HAS BEEN SAID THAT ASTRONOMY IS A HUMBLING AND CHARACTER BUILDING EXPERIENCE THERE IS PERHAPS NO BETTER DEMONSTRATION OF THE FOLLY OF HUMAN CONCEITS THAN THIS DISTANT IMAGE OF OUR TINY WORLD TO ME IT UNDERSCORES OUR RESPONSIBILITY TO DEAL MORE KINDLY WITH ONE ANOTHER AND TO PRESERVE AND CHERISH THE PALE BLUE DOT THE ONLY HOME WE HAVE EVER KNOWN”. Com a pontuação de: -2877.860475.

V. CONCLUSÕES

O objetivo deste trabalho foi desenvolver um código capaz de descriptografar mensagens cifradas usando a cifra de César, uma técnica de substituição simples, e a análise estatística baseada em quadgrams. O principal propósito era demonstrar como a probabilidade de um texto ser corretamente descriptografado pode ser calculada com base na frequência de ocorrência de quadgrams em inglês.

Na metodologia, as funções do código desempenharam papéis fundamentais. A função numero_de_quad() contou o número de quadgrams no arquivo de referência. A função pontuar() foi responsável por calcular a probabilidade das sequências de quadgrams na mensagem descriptografada. A função coloca_quad() leu e armazenou os quadgrams e suas pontuações a partir do arquivo de referência. A função decryption() descriptografou a mensagem cifrada com todas as chaves possíveis da cifra de César e calculou as pontuações para selecionar a descriptografia com a maior pontuação.

Os resultados obtidos com a execução do código demonstram claramente a utilidade da abordagem estatística baseada em quadgrams na descriptografia de mensagens cifradas. A função decryption() com sucesso revela a mensagem original a partir da cifra de César, identificando a chave que produz a maior pontuação, e a provável chave de descriptografia. No entanto, é importante notar que a eficácia da descriptografia depende em grande parte da

qualidade do arquivo de referência de quadgrams. Portanto, em cenários onde a qualidade do arquivo de referência é questionável ou onde a mensagem cifrada é muito curta, a precisão da descriptografia pode ser comprometida. Além disso, a complexidade de força bruta ao testar todas as 25 chaves possíveis também pode ser uma limitação em termos de tempo de execução, especialmente para mensagens mais longas. Portanto, considerações de eficiência e a qualidade do arquivo de referência são pontos importantes a serem considerados ao usar esta abordagem de descriptografia.

Analisando de forma mais abrangente o contexto dos resultados obtidos, é possível destacar a eficácia da abordagem utilizada no código. A função decryption() demonstrou a capacidade de descriptografar mensagens cifradas pela cifra de César, identificando a chave correta e revelando o conteúdo original.

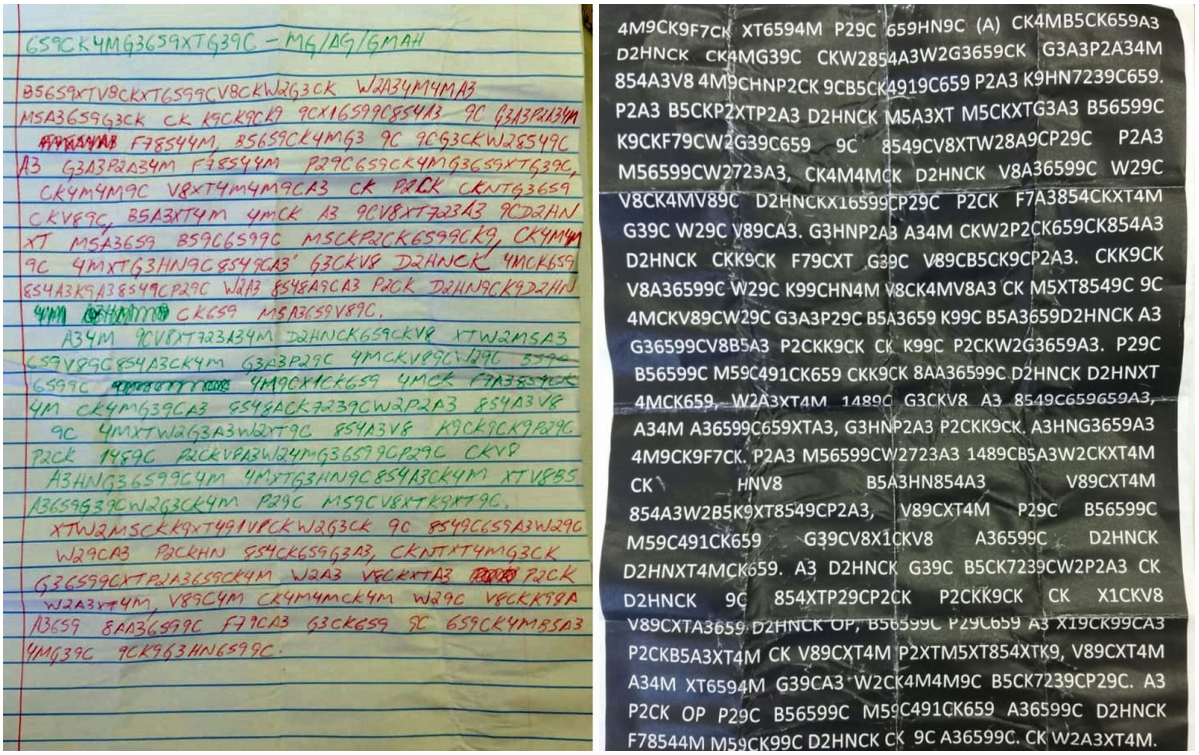
I. INTRODUÇÃO - SUBSTITUIÇÃO

Em criptografia, uma cifra de substituição é um método de criptografia que opera de acordo com um sistema pré-definido de substituição. Para criptografar uma mensagem, unidades do texto - que podem ser letras isoladas, pares ou outros grupos de letras - são substituídas para formar a cifra. As cifras de substituição são decifradas pela substituição inversa. Todavia, se a unidade de substituição estiver ao nível de palavras inteiras ou frases, como PORTA-AVIÕES ou ATAQUE ÀS 06H20M, o sistema é habitualmente dito ser um código, não uma cifra.
Uma cifra de substituição contrasta com uma cifra de transposição. Nestas últimas, as unidades do texto a cifrar são rearranjadas numa ordem diferente e habitualmente complexa, mas não modificadas. Por contraste, numa cifra de substituição, as unidades do texto são mantidas na mesma ordem, mas elas próprias são alteradas.
Existem diversos tipos de cifras de substituição. Se a cifra opera com letras isoladas, é denominada cifra de substituição simples. Se opera com grupos de letras chama-se cifra de substituição poligráfica. Uma cifra monoalfabética usa uma só substituição fixa na mensagem inteira, enquanto uma cifra polialfabética usa mais que uma. Uma cifra pode ainda recorrer a homófonos quando uma unidade de texto pode mapeada em mais que uma possibilidade distinta.
A Cifra de César é um dos exemplos clássicos da criptografia. Essa cifra é realizada através de um deslocamento do alfabeto. Cada letra tem uma correspondente fixa, e a chave é um deslocamento circular das letras do alfabeto original. Desse modo, o trabalho apresentado consiste em um código em língua.
(https://pt.wikipedia.org/wiki/Cifra_de_substituição)

II. REFERENCIAL TEÓRICO

A técnica de cifra de substituição é uma criptografia usada antigamente para esconder mensagens que não podiam ser vistas por pessoas que não eram bem vindas a um certo grupo. Aqui está um exemplo de cifra de substituição:

Figura 5 - Carta do pcc, criptografada



No geral, a criptografia de substituição é muito similar a cifra de césar, contendo apenas algumas diferenças. Algumas delas são a falta de previsibilidade, algo que é comum na cifra de césar.

III. METODOLOGIA

O código é projetado para descriptografar mensagens cifradas pela cifra de Substituição, este método é uma técnica estatística baseada em quadgrams.

O código em si é o mesmo usado anteriormente, porém com alterações em algumas partes, sendo elas bem importantes para que o código funcione corretamente.

Irei explicar um pequeno trecho:

Figura 5 - Trecho do código

```
for (int k = 1; k < 150; k++) {//loop responsavel em percorrer todo o alfabeto
    // Fazer uma cópia do texto cifrado em cada iteração
    strcpy(cifrado, texto_cifrado); // copiando a string na variavel cifrado
    for(int y = 0; y < 26; y++){
        chave = rand() % 26 + 65;
        chave2 = rand() % 26 + 65;
        for (i = 0; i < tam; i++) { //loop responsavel em percorrer a string inteira
            if (cifrado[i] >= 'A' && cifrado[i] <= 'Z') { //verifica se a string está entre 65 a 90, se estiver entra dentro do if
                if(cifrado[i] == chave)
                    cifrado[i] = chave2;
                else if (cifrado[i] == chave2) {
                    cifrado[i] = chave;
                }
            }
        }
    }
```

Aqui temos o trecho em que as chaves são trocadas aleatoriamente, essa parte troca a ordem alfabética 26 vezes e executa as tentativas 150 vezes, exibindo a mais próxima da frase certa. Frases essas que possuem o maior score entre elas.

IV. RESULTADO

Os resultados obtidos com o código incluem a descriptografia da mensagem cifrada e a pontuação associada a cada tentativa. O código seleciona a descriptografia com a pontuação mais alta e a exibe como resultado final:
“VDQXFHSQTQFKQVDQQCJNRMFVGRYREVDQTRKURKGKVDQXFHASQVDMQ
FVQYVDQTRYVGYOGYIREROMKJQTGQKGYVDFVWNQFZSRMNXFMUKWQFM
GYIMQKROMTQDOYIMHJQRJNQFYXYFVGRYKSRONXWQJMRYQVRFTVRYVD
QGMNRSTRYVMFTVQXJMQLOXGTQKFYXSRONXDFBQKQQYVDQNFKVIFKJRE
DOUFYQYNGIDVQYUQYVOYVGNVDQMGKQREFBGKGRYFMHYQSTONVOMQ
VDFVRYTQFIFGYQUWMFTQKVDQTRKUGTJQMKJQTVGBQFJQMKJQTVGBQGY
SDGTDSQFMQRYQEGVVGYIYQGVDQMFWRBQYRMWQNRSWOVSGVDGY”
Com a Pontuação: -4237.042178

V. CONCLUSÕES

O objetivo deste trabalho foi desenvolver um código capaz de descriptografar mensagens cifradas, obtemos alguns bons resultados, porém, como o algoritmo que fizemos é aleatório, tem chances de ser certo, como também pode não dar. Este código implementado em C# é muito raro de se acertar. Temos consciência de que há alguns métodos melhores para realizar essa tarefa, os quais tentamos algumas vezes, mas não conseguimos implementar corretamente na linguagem usada.

Este código retorna traduções com pontuações entre -2600 e -5500, planejávamos aumentar um pouco esse score e aproximar mais da real tradução.