UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

JOHNNY MARCOS SILVA SOARES

INDEXAÇÃO DE IMPRESSÕES DIGITAIS UTILIZANDO UMA ABORDAGEM TEXTUAL

QUIXADÁ

2021

JOHNNY MARCOS SILVA SOARES

INDEXAÇÃO DE IMPRESSÕES DIGITAIS UTILIZANDO UMA ABORDAGEM TEXTUAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Regis Pires Magalhães
Coorientador: Prof. Dr. Luciano de Andrade Barbosa

QUIXADÁ

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S654i Soares, Johnny Marcos Silva.
Indexação de impressões digitais utilizando uma abordagem textual / Johnny Marcos Silva Soares. – 2021.
52 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Computação, Quixadá, 2021.
Orientação: Prof. Dr. Regis Pires Magalhães.
Coorientação: Prof. Dr. Luciano de Andrade Barbosa.

1. Recuperação da informação. 2. Impressão digital (Computação). 3. Biometria. I. Título.
CDD 621.39

JOHNNY MARCOS SILVA SOARES

INDEXAÇÃO DE IMPRESSÕES DIGITAIS UTILIZANDO UMA ABORDAGEM TEXTUAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: __/__/____.

BANCA EXAMINADORA

Prof. Dr. Regis Pires Magalhães (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Luciano de Andrade Barbosa (Coorientador)
Universidade Federal de Pernambuco (UFPE)

Prof. Dr. Paulo Antonio Leal Rego
Universidade Federal do Ceará (UFC)

Prof. Dr. José Gilvan Rodrigues Maia
Universidade Federal do Ceará (UFC)

Aos meus pais.

AGRADECIMENTOS

Agradeço imensamente à minha mãe, Maria Nezian, e ao meu pai, Marcos Antônio, por todos os sacrifícios realizados, por todo apoio, carinho, dedicação, força e confiança. Obrigado por terem acreditado em mim. Nada disso seria possível sem vocês. Amo vocês com minha vida.

Agradeço à minha namorada Letícia Saraiva, por todo carinho, amor, companheirismo, dedicação e apoio em todos os momentos nesses anos. Sua presença, carinho e ajuda foram essenciais nesse período. Te amo.

A esta universidade, seu corpo docente, direção e administração que oportunizaram todo o aprendizado, oportunidades, ótimas vivências, como também péssimas, mas que me trouxeram grande aprendizado profissional e pessoal.

Ao professor Paulo Rego, por me dar a primeira grande oportunidade como pesquisador, sou muito grato por todas as portas que foram abertas. Meus sinceros agradecimentos ao professor Luciano Barbosa, pelas oportunidades, conselhos, ensinamentos, conversas e orientações. Ao professor Regis Pires, pela orientação, apoio e confiança nesses anos. Agradeço também ao professor Gilvan Maia, pela disponibilidade, conversas e apoio.

Agradeço aos meus amigos que conquistei nesta jornada, sou grato ao Robertty, Carlos Alberto, Gabriel Uchôa, Camila, Marcelo, Beatriz, Rafaella, Júlio, Andson, Deyvisson e João Mateus, pela amizade e por todos os momentos juntos. Principalmente ao Robertty por ser minha dupla nesses anos de estudo e um grande amigo.

Agradeço aos amigos de longa data da equipe 9HR, William, Raul, Leonardo, Humberto e Jardel. Agradeço muito ao meu amigo de mais longa data, Jardel por toda nossa amizade. E agradeço também ao meu amigo Caio César, mais conhecido como Chapa César, por todos os anos de vivência juntos nesse período de graduação e por nossa amizade.

“Faça o teu melhor, na condição que você tem, enquanto você não tem condições melhores, para fazer melhor ainda!”

(Mario Sergio Cortella)

RESUMO

O uso de credenciais como senhas e códigos podem ser facilmente perdidas e esquecidas, além de ser mais suscetível à fraudes. Por esse motivo, o uso de informações biométricas na autenticação de pessoas cresceu nos últimos anos. Além disso, as informações biométricas possuem uma grande quantidade de domínios de aplicações. As impressões digitais são as informações mais utilizadas em aplicações que utilizam sistemas biométricos, sendo aplicadas em dispositivos móveis, sistemas bancários e sistemas de identificação de pessoas em órgãos governamentais, entre outras. Com o aumento na quantidade de pessoas, mais informações biométricas são armazenadas em sistemas de identificação, tornando necessário o uso de métodos mais robustos de indexação e busca de dados de impressões digitais. Além disso, as informações textuais são geradas em maior velocidade e volume, tornando ainda mais necessário o uso de técnicas mais robustas de processamento paralelo e distribuído de dados. Por esse motivo, este trabalho propõe o uso de abordagens de processamento textual aplicação no problema de indexação e busca de impressões digitais. São implementadas duas abordagens utilizando dados textuais: o Método 1, utiliza tabelas *hash* e *locality sensitive hashing*; o Método 2, utiliza índice invertido do Elasticsearch e *locality sensitive hashing*. Ambos os métodos obtiveram bons resultados no conjunto de dados da FVC 2002 DB1a, com 0,85% e 0,42% de *Erro Rate* com 10% de *Penetration Rate*, respectivamente. Porém, no experimento com o conjunto de dados NIST SD14 os métodos não escalaram com o aumento de impressões digitais.

Palavras-chave: Recuperação da informação. Impressões digitais. Biometria.

ABSTRACT

The use of credentials such as passwords and codes can easily be lost and forgotten. For this reason, the use of biometric information in authenticating people has grown in recent years. Also, biometric information has a large number of application domains. Fingerprints are the most used information in applications that use biometric systems, being applied in mobile devices, banking systems, people identification systems in government agencies, and among others. With the increase in the number of people, more biometric information is stored in identification systems, making it necessary to use more robust indexing methods and searching for fingerprint data. Furthermore, textual information is generated faster, making it even more essential to use more robust parallel and distributed data processing techniques. For this reason, this work proposes the use of textual processing approaches applied to the problem of indexing and searching for fingerprints. Two techniques are implemented using textual data: Method 1 uses tables hash and locality sensitive hashing; Method 2 uses inverted index from Elasticsearch and locality sensitive hashing. Both methods obtained good results in the FVC 2002 DB1a data set, with 0.85% and 0.42% Error Rate on 10% Penetration Rate, respectively. However, in the experiment with the NIST SD14 dataset, the methods did not scale with the increase in fingerprints.

Keywords: Information retrieval. Fingerprint. Biometry.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AFIS	*Automated Fingerprint Identification System*
NBIS	*NIST Biometric Image Software*
NIST	*National Institute of Standards and Technology*
FBI	*Federal Bureau of Investigation*
DHS	*Department of Homeland Security*
WSQ	*Wavelet Scalar Quantization*
MCC	*Minutia Cylinder-Code*
LSH	*Locality Sensitive Hashing*
HBFT	*Hierarchical Bloom Filter Tree*
API	*Application Programming Interface*
FVC	*Fingerprint Verification Competition*
NIST SD4	*NIST Special Database 4*
NIST SD14	*NIST Special Database 14*

SUMÁRIO

1 INTRODUÇÃO

As impressões digitais são as informações biométricas mais estudadas e amplamente utilizadas em aplicações comerciais, governamentais e forenses para identificar indivíduos (MALTONI *et al.*, 2009). As aplicações são de vários domínios diferentes, como: investigação criminal, identificação de pessoas desaparecidas, identificação nacional, identificação em telefones celulares e transações bancárias, entre outras.

Maltoni *et al.* (2009) definem um sistema biométrico com impressões digitais como um sistema de verificação e identificação. A verificação possui o objetivo de analisar se um sujeito é realmente quem ele diz ser, comparando com informações previamente recolhidas das impressões digitais. Já a identificação é o processo de busca de uma ou mais impressões digitais em um conjunto de impressões digitais.

Em 1977, foi criado o primeiro *Automated Fingerprint Identification System* (AFIS) (em português, Sistema Automático de Identificação de Impressões Digitais) para realizar as operações que vão desde a coleta das impressões digitais até o carregamento de informações em bancos de dados, como também, a realização das buscas de impressões digitais em conjuntos de dados armazenados em uma ou mais AFIS (KOMARINSKI, 2005).

Para realizar a comparação entre impressões digitais são utilizados algoritmos de *matching* que relacionam informações entre impressões digitais, com o objetivo de verificar a similaridade entre elas. Porém, com o aumento na quantidade de dados de impressões digitais, a comparação de uma impressão digital em uma base completa é computacionalmente complexo.

Por esse motivo, nos AFIS são utilizadas técnicas de indexação de impressões digitais com o objetivo de diminuir o espaço de busca, processamento e tempo das consultas de impressões digitais. O processo de indexação é baseado na criação de estruturas que facilitam a localização de informação, utilizando técnicas para comparação rápida de similaridade entre dados e retornando um *ranking* de elementos mais semelhantes (MELUCCI; BAEZA-YATES, 2011).

Existem vários métodos de indexação de impressões digitais que utilizam informações extraídas das impressões digitais (MALTONI *et al.*, 2009). Em alguns deles são aplicadas técnicas que utilizam informações de características chamadas minúcias. As minúcias são informações extraídas em algumas regiões das impressões digitais e são utilizadas em métodos de *matching* e indexação. Cappelli *et al.* (2010a) desenvolveram uma representação binária para as minúcias e utilizaram essa informação para indexar os dados. Já Khodadoust e Khodadoust

(2017) criaram triângulos a partir da posição das minúcias e indexaram as informações dos triângulos.

Com a mineração de texto é possível extrair informações úteis a partir de documentos textuais, usando técnicas de mineração de dados, recuperação de informação e processamento de linguagem natural (FELDMAN *et al.*, 2007). Lin e Dyer (2010) informam que devido ao aumento na quantidade de dados (*Big Data*), tornou-se necessário o aprimoramento de técnicas cada vez mais robustas de processamento paralelo e distribuído em mineração de texto.

O trabalho de Soares *et al.* (2020) utiliza o Elasticsearch[1] na indexação e buscas textuais de impressões digitais. Uma das abordagens propostas neste trabalho utiliza processamento textual e tabelas *hash* para realizar a indexação e busca de impressões digitais. A outra abordagem, semelhante a de Soares *et al.* (2020), propõe o uso da escalabilidade da ferramenta Elasticsearch. O Elasticsearch é um mecanismo de busca que utiliza índice invertido para a realização de indexação e busca, podendo chegar a *petabytes* de dados indexados (GORMLEY; TONG, 2015).

Além disso, ele possui parâmetros para configurar um índice de maneira simples, como por exemplo, o parâmetro *shards* particiona o índice através dos nós no *cluster* do Elasticsearch. Outro parâmetro importante são as *réplicas* utilizadas no índice. Elas são cópias dos dados dos *shards* salvas em nós diferentes, garantindo mais disponibilidade aos dados.

O Elasticsearch suporta consultas de documentos completos, denominada *full-text search*. Esse tipo de consulta utiliza modelos de similaridade para associar os documentos indexados com o documento buscado. O Elasticsearch possui disponíveis vários modelos tradicionais de Recuperação de Informação (BAEZA-YATES *et al.*, 1999), como por exemplo, BM25 (ROBERTSON *et al.*, 2004) e similaridade cosseno.

Por esse motivo, o trabalho proposto aborda o uso de processamento textual aplicado na indexação e busca de impressões digitais. A principal contribuição deste trabalho é a abertura para o uso de técnicas mais robustas de processamento de textos no problema de indexação de impressões digitais.

1.1 Objetivo Geral

Este trabalho possui o objetivo de desenvolver uma técnica de indexação de impressões digitais baseado em minúcias para facilitar a busca em um conjunto de impressões digitais.

[1] *www.elastic.co*

E realizar a implementação de uma abordagem similar utilizando o Elasticsearch.

1.2 Objetivos Específicos

a) Implementar o método de indexação de impressão digital baseado no processamento de documentos;
b) Implementar a mesma abordagem utilizando os princípios de índice invertido com o Elasticsearch;
c) Realizar a mudança dos parâmetros do método proposto e analisar o impacto no tempo de resposta e na qualidade do *ranking* das consultas;
d) Analisar a escalabilidade do método aumentando a quantidade de impressões digitais indexadas;
e) Comparar os resultados com métodos no estado da arte na área de indexação de impressões digitais.

1.3 Organização

Este trabalho está dividido na seguinte forma: no Capítulo 2, são apresentados conceitos necessários para o entendimento do trabalho; no Capítulo 3, são retratados alguns trabalhos relacionados com o método proposto; o Capítulo 4 apresenta os procedimentos metodológicos que foram seguidos neste trabalho; no Capítulo 5, são mostradas as técnicas implementadas, como também os resultados de qualidade do *ranking* e tempo de busca; por fim, o Capítulo 6 conclui este trabalho, resumindo as contribuições e trabalhos futuros.

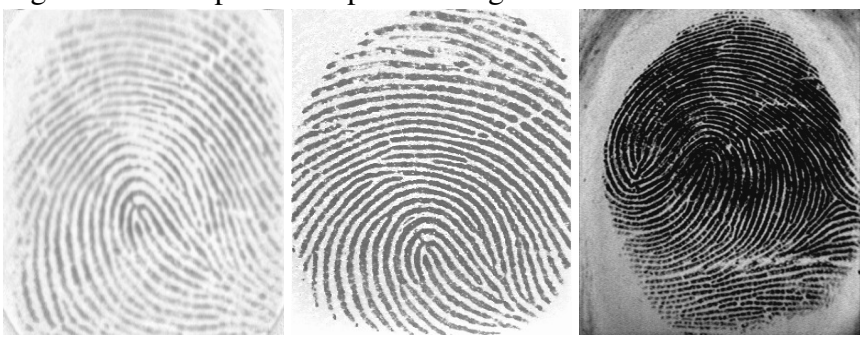
2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados os conceitos fundamentais para o entendimento e desenvolvimento deste trabalho.

2.1 Impressão Digital

As impressões digitais ou datilogramas são formadas por estruturas na pele chamadas de papilas, sendo estas elevações na pele nas pontas dos dedos. Essas formações variam entre as pessoas e geralmente não se modificam com o tempo. Por esse motivo, os datilogramas são usados desde a antiguidade para identificar pessoas e autenticar documentos (JUNIOR, 1991). Exemplos de impressões digitais podem ser vistos na Figura 1.

Figura 1 – Exemplos de impressões digitais



Fonte: Maio *et al.* (2004)

2.1.1 Sistemas Biométricos

Maltoni *et al.* (2009) definem sete características que os sistemas biométricos possuem, sendo elas:

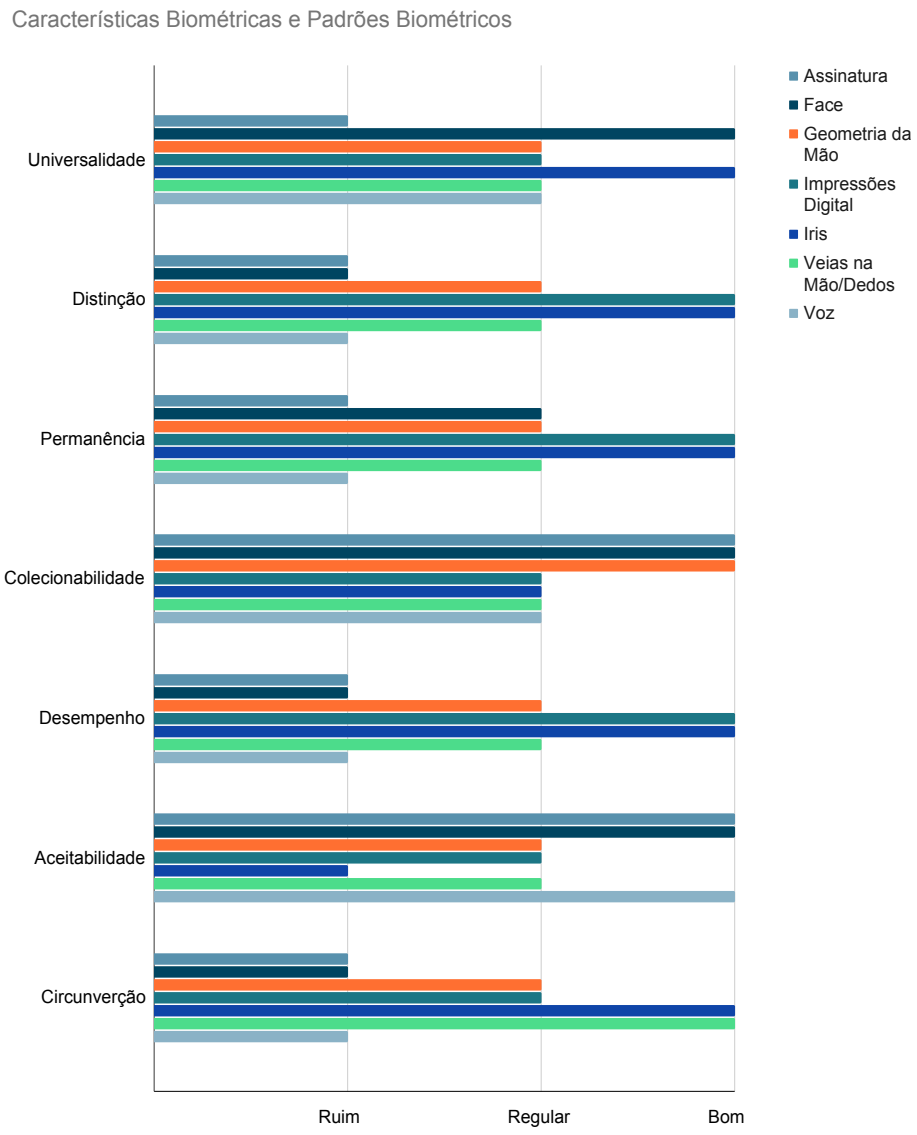
- Universalidade: todas as pessoas no mundo possui o padrão biométrico.
- Distinção: quaisquer duas pessoas possuem características biométricos suficiente para distingui-las.
- Permanência: com o passar dos anos, as características biométricas tendem a não ser alteradas.
- Colecionabilidade: os padrões biométricos podem ser mensuráveis quantitativamente.
- Desempenho: pode ser definido como velocidade de processamento e acurácia das técnicas.
- Aceitabilidade: uma medida que informa a disposição dos usuários para aquele padrão

biométrico.

- Circunvenção: a facilidade que o sistema biométrico pode ser contornado por técnicas fraudulentas.

Para complementar a análise das técnicas biométricas, Maltoni *et al.* (2009) classificam alguns dos principais padrões biométricos em todas as características biométricas, utilizando Ruim, Regular e Bom para realizar uma comparação entre as técnicas. Nesta classificação, as impressões digitais obtiveram o melhor resultado, considerando a não ocorrência de Ruim e a quantidade de ocorrência de Bom. No Figura 2 sete dos principais métodos biométricos são avaliados nas sete características. Quanto maior a barra horizontal do padrão biométrico, melhor o seu resultado na característica biométrica.

Figura 2 – Comparativo das técnicas biométricas



Fonte: Elaborado pelo autor

2.1.2 Features das Impressões Digitais

As imagens de impressões digitais possuem uma grande quantidade de informações que podem ser extraídas por algoritmos. As características extraídas são utilizadas no reconhecimento dos usuários e são agrupadas em três grupos: *Features Level* 1, *Level* 2 e *Level* 3.

2.1.2.1 Features Level 1 (L1)

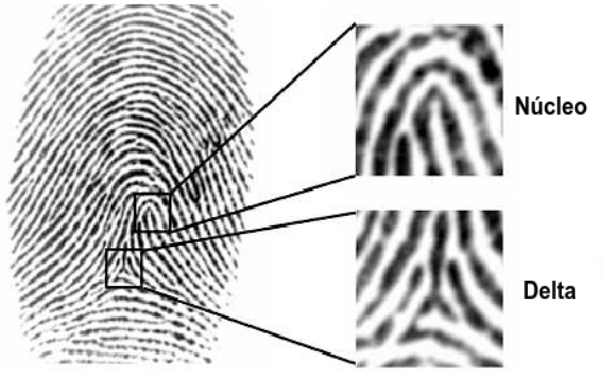
As *features* de *level* 1 são informações globais das impressões digitais, como por exemplo, os pontos singulares são estruturas conhecidas como núcleo e delta. O núcleo é uma formação localizada na região central das impressões digitais e podem possuir uma direção. O delta é uma angulação expressiva de algumas linhas ou uma formação em formato de triângulo que pode ocorrer algumas vezes em uma impressão digital (LEVI; SIROVICH, 1972). Na Figura 3 são apresentados exemplos de um núcleo e um delta em uma impressão digital. A organização e quantidade de núcleos e deltas são utilizadas para classificar as impressões digitais em classes. O sistema de classificação de Henry (HENRY, 1905) descrito na Figura 4, divide as impressões digitais em 5 classes:

- *Arch*: não possui deltas e núcleo, além disso, apresenta linhas que atravessam lateralmente a impressão digital.
- *Left loop*: apresenta núcleo e delta, e possui formação com angulação voltada para a esquerda.
- *Right loop*: apresenta núcleo e delta, além de possuir uma formação com angulação voltada para a direita.
- *Tented arch*: existe núcleo e delta e possui uma formação de tenda.
- *Whorl*: o núcleo tem formato de espiral no centro da impressão digital e existem dois deltas nas laterais.

2.1.2.2 Features Level 2 (L2)

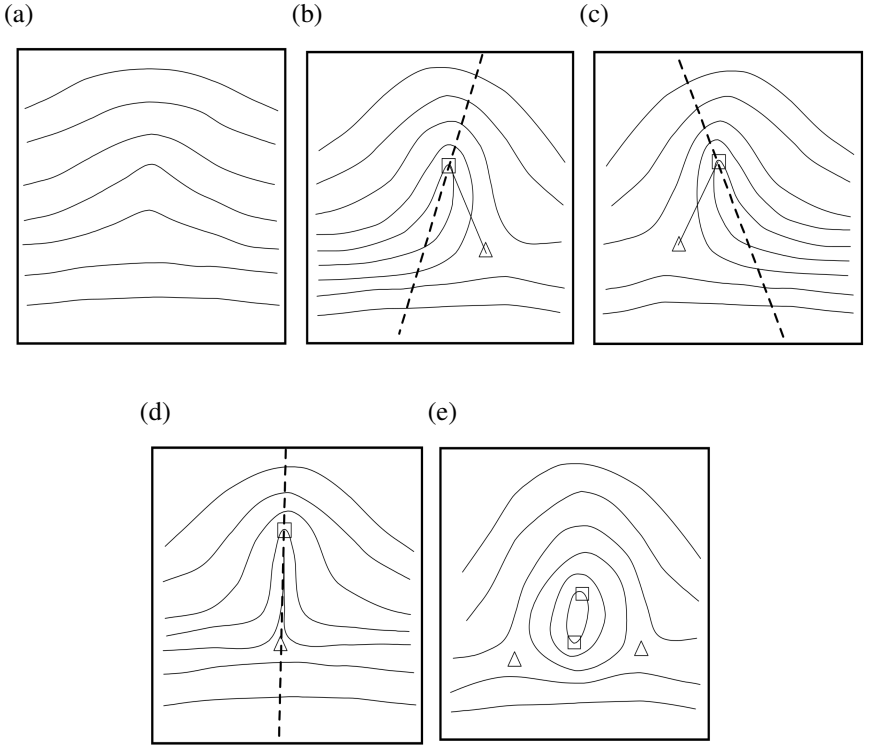
Também conhecidas como *features* locais, as *features* L2 são constituídas por informações encontradas nas estruturas denominadas cristas (MALTONI *et al.*, 2009). As características encontradas nas cristas são chamadas de minúcias e possuem quatro tipos, porém, os finais de linha e bifurcações são as *features* mais recorrentes. Os finais de linha são linhas

Figura 3 – Pontos singulares em uma impressão digital



Fonte: Adaptado de Chikkerur e Ratha (2005)

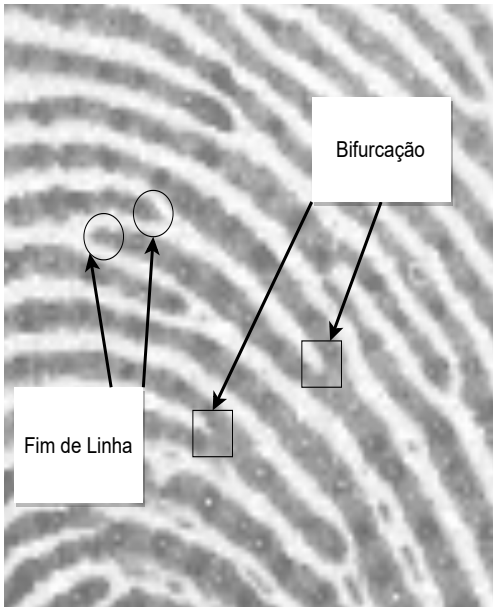
Figura 4 – Padrão de classificação das impressões digitais em (a) *Arch*, (b) *Left loop*, (c) *Right loop*, (d) *Tented arch* e (e) *Whorl*. Os deltas são representados pelos triângulos e os núcleos pelos quadrados



Fonte: Adaptado de Maltoni *et al.* (2009)

finalizadas abruptamente e bifurcações são linhas que dividem-se em outras duas linhas. As minúcias são as características das impressões digitais mais utilizadas em algoritmos de *matching* e de indexação, devido a rápida velocidade de processamento e estabilidade na qualidade de resultados (HOLDER *et al.*, 2011). Na Figura 5 são mostrados os dois tipos de minúcias em uma impressão digital.

Figura 5 – Minúcias em uma impressão digital



Fonte: Elaborado pelo autor

2.1.2.3 Features Level 3 (L3)

São as características mais específicas dentre as características descritas. As *Features* L3 levam em consideração os detalhes das cristas, por exemplo, espessura da linha, formato, contorno, curvatura e furos nas linhas. Porém, é preciso imagens de alta qualidade para encontrar esse tipo de característica, por esse motivo, esse tipo de *feature* não é muito comum (MALTONI *et al.*, 2009).

2.2 *NIST Biometric Image Software* (NBIS)

O *NIST Biometric Image Software* (NBIS) é um pacote de *softwares* de domínio público para extração e uso de informações biométricas. O NBIS foi desenvolvido pelo *National Institute of Standards and Technology* (NIST) para o *Federal Bureau of Investigation* (FBI) e o *Department of Homeland Security* (DHS) para ser utilizado no processamento de imagens de impressões digitais (KO, 2007). O NBIS é dividido em 7 pacotes de *softwares* utilitários, sendo eles:

- AN2K7: um pacote de referencia de implementação no padrão ANSI/NIST-ITL.
- BOZORTH3: ferramenta para inferir a similaridade entre arquivos de minúcias gerados pelo MINDTCT.
- IMGTOOLS: um conjunto de ferramentas para imagens, por exemplo, conversores entre padrões de compressão de imagem JPEG e *Wavelet Scalar Quantization* (WSQ).

- MINDTCT: detector de minúcias de impressões digitais. É gerado um arquivo de minúcias com a posição na imagem, ângulo e qualidade das minúcias.
- NFIQ: algoritmo de análise de qualidade das imagens de impressões digitais.
- NFSEG: ferramenta de segmentação da região contendo uma impressão digital na imagem.
- PCASYS: classifica as impressões digitais em grupos semelhante ao sistema de classificação de Henry.

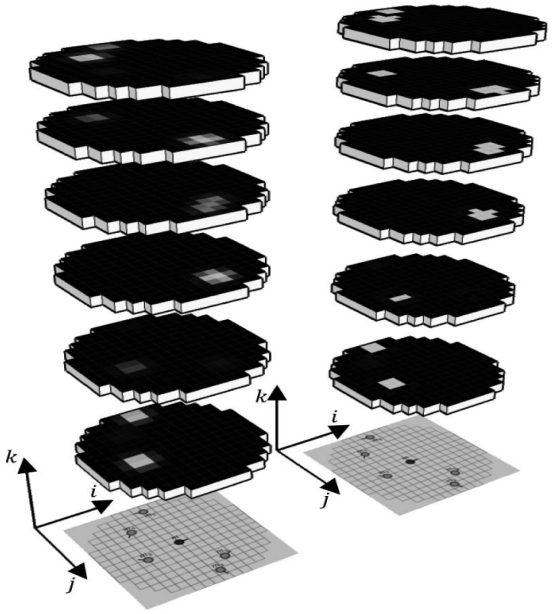
Todas as ferramentas do pacote NBIS possuem compatibilidade com as imagens de impressões digitais no padrão WSQ, que é um formato de compressão de imagens desenvolvido pelo FBI na década de 90. Esse método foi criado especificamente para tratar imagens de impressões digitais, devido à necessidade da padronização das imagens que seriam utilizadas no AFIS desenvolvido pelo FBI (BRISLAWN *et al.*, 1996). Este trabalho utiliza apenas a ferramenta MINDTCT para extrair as minúcias na etapa de processamento das impressões digitais.

2.3 *Minutia Cylinder-Code* (MCC)

Cappelli *et al.* (2010b) propõem o uso de uma representação associada às minúcias das impressões digitais. O *Minutia Cylinder-Code* (MCC) utiliza as minúcias e informações das suas vizinhanças para criar uma representação em forma de cilindro para cada minúcia. Cada cilindro é centralizado em uma minúcia m e dividido em seções. Cada seção recebe a contribuição espacial e direcional das minúcias da vizinhança que possuem uma certa diferença angular, comparando com o ângulo da minúcia m. As contribuições das minúcias da vizinhança são dadas por uma função f e são acumuladas nas seções. Em seguida, as seções são binarizadas por um limiar δ e são organizadas no formato de vetor binário. Na Figura 6 são mostradas as seções dos cilindros com algumas regiões brancas com contribuições das minúcias da vizinhança da minúcia m.

O MCC é invariante à translação e rotação da impressão digital, como também à distorção de pele e pequenos erros de extração (CAPPELLI *et al.*, 2010b). Além disso, o MCC também pode ser utilizado em frações das impressões digitais e pode ser relacionado com frações semelhantes de outras impressões digitais. Esta representação faz a discretização de uma imagem de impressão digital para um conjunto de vetores binários que podem ser utilizadas em técnicas de *matching* e indexação. A comparação simples de *matching* é realizada, fazendo a similaridade entre cada posição dos vetores binários para encontrar a quantidade de posições similares, utilizando por exemplo, distância de Hamming (WEGNER, 1960).

Figura 6 – Seções de dois cilindros do MCC



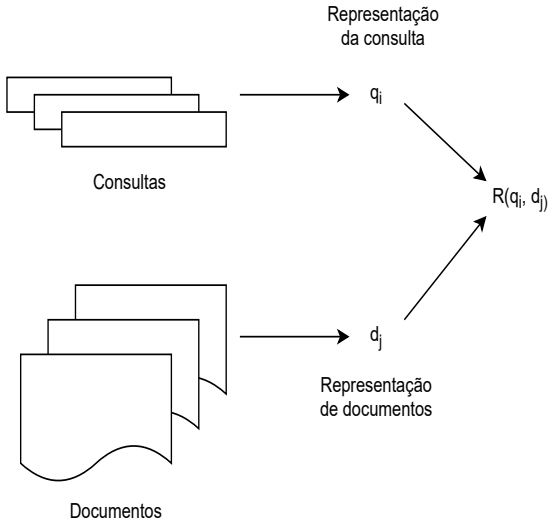
Fonte: Cappelli *et al.* (2010b)

2.4 Recuperação de Informação

A recuperação de informação aborda o armazenamento e recuperação de documentos, páginas *Web*, arquivos multimídia, dados estruturados e semiestruturados (SCHÜTZE *et al.*, 2008). Muitas pessoas utilizam a recuperação de informação no cotidiano quando realizam buscas por páginas *Web*, buscas por *email* ou busca de arquivos por título. Antes de realizar a busca dos dados é preciso indexar as informações em estruturas que facilitem a realização da busca. Um modelo de recuperação de informação é baseado na realização de uma consulta que será representada como q_i e um conjunto de documentos que são representados por d_j. Por fim, uma função de *ranking* associa uma consulta q_i com um documento d_j para criar o *rank* dos documentos mais semelhantes (RICARDO; BERTHIER, 2011). A Figura 7 mostra a associação dos documentos com as consultas em um modelo de recuperação de informação, que utiliza representação da *query* e do documento e gera o *rank* a partir de uma função de *ranking* $R(q_i, d_j)$.

A indexação consiste no processo de inserir elementos em estruturas que evitam a comparação com todos os elementos do conjunto de dados ou realiza comparações aproximadas no momento da realização da busca (SCHÜTZE *et al.*, 2008). A etapa de indexação possui maior custo na inserção, porém, reduz a complexidade da busca.

Figura 7 – Modelo de recuperação de informação

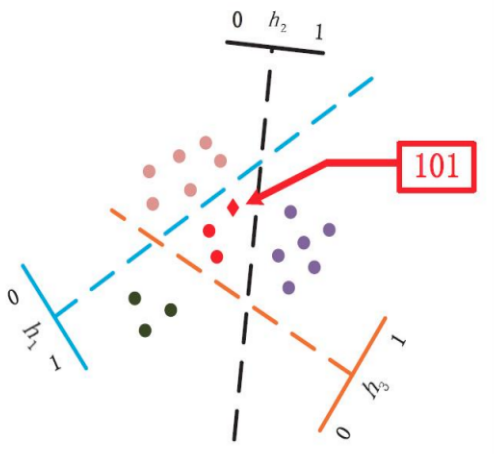


Fonte: Adaptado de Ricardo e Berthier (2011)

2.4.1 Locality Sensitive Hashing (LSH)

A indexação pode ser realizada em forma de agrupamento dos dados a partir de características semelhantes, como por exemplo, com o uso de *Locality Sensitive Hashing* (LSH) (em português, Hash Sensível à Localidade), que agrupa elementos semelhantes em grupos denominados *buckets* (GIONIS *et al.*, 1999), (MIMAROGLU; SIMOVICI, 2008). Então, a busca por algum elemento será realizada apenas no *bucket* onde o elemento buscado mais se assemelha. A Figura 8 mostra a abordagem de LSH com três posições binárias. Cada posição binária pode se posicionar de um lado das funções h_1, h_2 e h_3. O grupo que um elemento será inserido ou buscado é dado pela posição dos 3 *bits*.

Figura 8 – Abordagem de LSH para selecionar o grupo mais semelhante

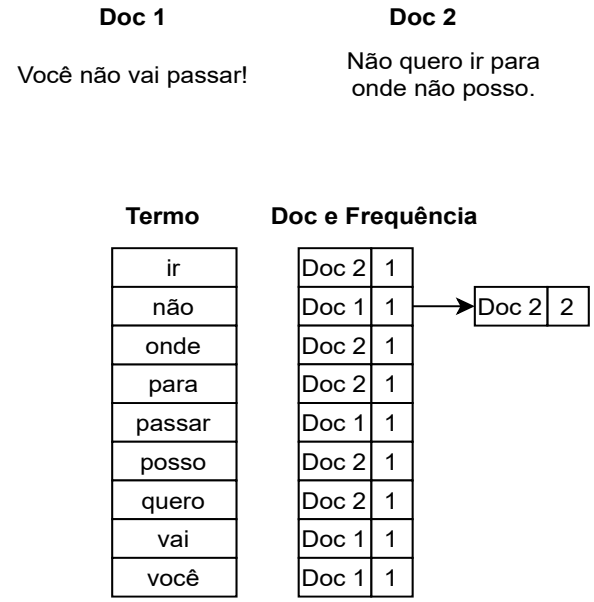


Fonte: Li *et al.* (2017)

2.4.2 Índice Invertido

Outra forma de indexação é a utilização de índice invertido, onde cada termo dos documentos textuais são contabilizados e são armazenados com base na frequência de cada termo em cada documento (SCHÜTZE *et al.*, 2008). O índice invertido é criado a partir da inversão da forma de armazenar os documentos. Consistindo no armazenamento do termo, uma lista de documentos que possuem o termo, juntamente com a quantidade de ocorrência em cada documento. A Figura 9 mostra um exemplo de criação de índice invertido utilizando dois documentos de textos.

Figura 9 – Criação de um índice invertido



Fonte: Elaborado pelo autor

O índice invertido possui maior custo computacional na etapa de indexação, já na busca, a utilização do índice invertido permite buscas rápidas de textos completos (CUTTING; PEDERSEN, 1989). O Elasticsearch utiliza o índice invertido para indexar os documentos e realizar buscas de textos completos. Por esse motivo, será utilizado o Elasticsearch em uma das abordagens deste trabalho.

2.5 Métrica de Avaliação

A avaliação dos métodos de indexação de impressões digitais é realizada com a relação entre *Penetration Rate* e *Error Rate* (MANSFIELD; WAYMAN, 2002). O *Penetration Rate* é a porcentagem da base que é pesquisada e que pode ter o resultado correto. O *Error*

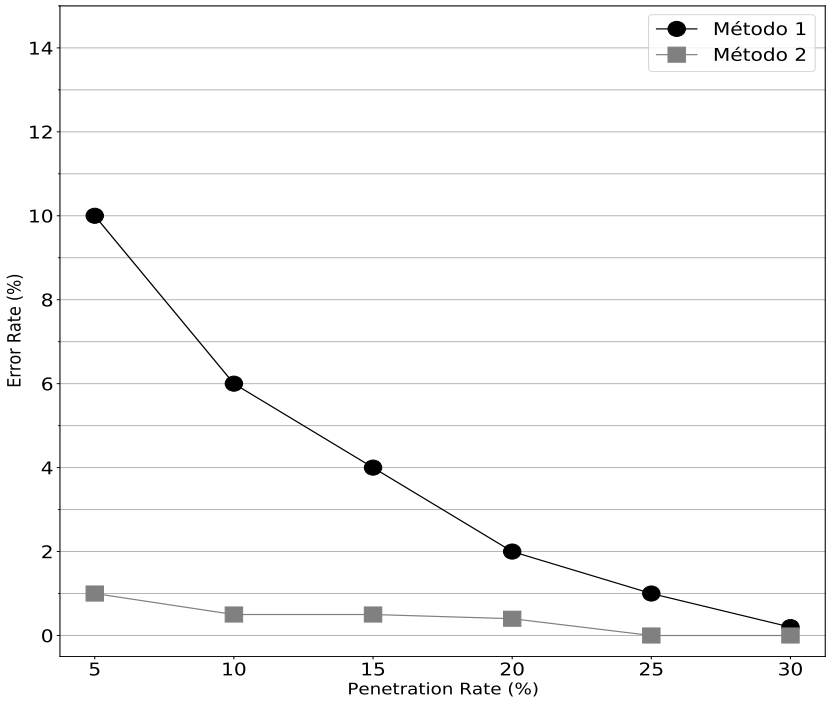
Rate é a porcentagem de elementos buscados que não foram encontrados com um certo valor de *Penetration Rate*. Usualmente nos métodos de indexação de impressões digitais é avaliado o *Error Rate* com 10% de *Penetration Rate* (CAPPELLI *et al.*, 2010a). Por exemplo, para uma busca realizada em uma base de 100 impressões digitais, caso a impressão digital correta não esteja entre a primeira e a decima posição no *ranking*, esta consulta será considerada errada e aumentará o valor do *Error Rate* com 10% de *Penetration Rate*.

Para calcular o *Error Rate* é preciso selecionar o valor de *Penetration Rate*. Para em seguida, verificar a quantidade de buscas que encontraram os documentos corretos dentro do limiar definido no *Penetration Rate*. A Equação 2.1 mostra como é calculado o *Error Rate* a partir da quantidade de buscas encontradas e o total de buscas realizadas.

$$Error\ Rate = 1 - \frac{Buscas\ Encontradas}{Total\ de\ Buscas} \tag{2.1}$$

Na Figura 10 é mostrada a forma gráfica da métrica de *Error Rate* X *Penetration Rate* com dados de dois métodos. São mostrados dois métodos e seus valores de *Error Rate* para valores de *Penetration Rate* de 5, 10, 15, 20, 25, 30. Além disso, é possível verificar que o Método 1 conseguiu um melhor resultado, pois seu valor de *Error Rate* com 10% de *Penetration Rate* é inferior ao *Error Rate* obtido no Método 2.

Figura 10 – Gráfico de *Error Rate* X *Penetration Rate*



Fonte: Elaborado pelo autor

3 TRABALHOS RELACIONADOS

Nesta seção, é apresentado o trabalho de Cappelli *et al.* (2010a), que propõe o uso do MCC e de LSH, no qual esse trabalho é baseado. É apresentado também, o trabalho de Paulino *et al.* (2013) que utiliza uma abordagem de junção de vários métodos de indexação para gerar um *score* de similaridade. Já o trabalho de Moia e Henriques (2018) utiliza uma abordagem com o MCC e uma estrutura de dados chamada *Hierarchical Bloom Filter Tree* (HBFT). Por fim, Soares *et al.* (2020) utiliza o Elasticsearch e LSH na indexação e busca.

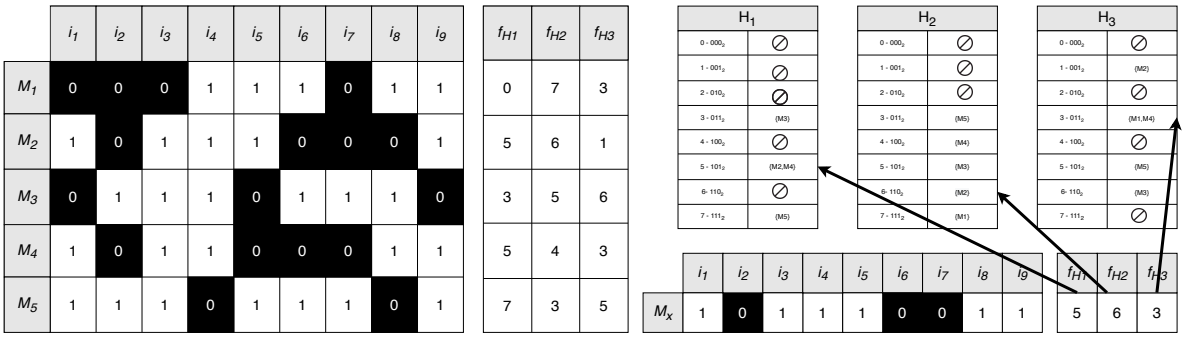
3.1 *Fingerprint Indexing Based on Minutia Cylinder-Code*

No trabalho proposto em Cappelli *et al.* (2010a) é utilizado o MCC para criar a representação binária das minúcias de cada impressão digital. Em seguida, é utilizado LSH para agrupar segmentos dos vetores binários de cada minúcia m_i e armazenar a informação i em *buckets* em tabelas *hash* a partir do valor inteiro gerado pelo segmento do vetor. São utilizadas l tabelas *hash* e l funções de *hash*, no qual cada função de *hash* f_{H_k} no intervalo de $f_{H_1}, ..., f_{H_l}$ recebe um subvetor de tamanho h bits e retorna a posição do *bucket* referente ao número decimal gerado pelos h bits. Em seguida, o identificador da minúcia é inserido no *bucket* informado. Para realizar a busca de um vetor binário de uma minúcia m_x, são computados os valores decimais das l funções de *hash*. Para cada valor decimal encontrado, é acessada uma posição na tabela *hash* para recuperar uma lista de minúcias. Em seguida, as minúcias com valor máximo de ocorrência são consideradas semelhantes. Depois que todas as minúcias forem consultadas, é criado um ranqueamento das impressões digitais mais semelhantes utilizando uma função de similaridade baseada na distância *Hamming*, descrita em Gionis *et al.* (1999).

Na Figura 11 é descrito um exemplo com $n = 9$, $l = 3$ e $h = 3$, no qual existem 5 vetores binários, gerados por 5 minúcias, que serão indexados e uma minúcia M_x que será buscada. Depois da busca da minúcia M_x é contabilizada a quantidade de ocorrências de cada minúcia acessada nas tabelas *hash* e são retornadas as minúcias $\{M2, M4\}$, pois possuem maior ocorrência. Após isso, é computado o *score* da distância *Hamming* de cada impressão digital para criar o ranqueamento.

O trabalho proposto é baseado na técnica de Cappelli *et al.* (2010a) no uso de LSH, *features* L2 juntamente com vetores do MCC. Porém, este trabalho, utiliza informações textuais armazenadas em cada tabela *hash* e o trabalho de Cappelli *et al.* (2010a) armazena apenas os

Figura 11 – Exemplo do método de indexação com LSH proposto em Cappelli *et al.* (2010a)



Fonte: Elaborado pelo autor

valores numéricos nas tabelas. Além disso, é utilizada a distância Hamming como métrica de similidade na busca em Cappelli *et al.* (2010a), já neste trabalho, é utilizado contagem simples de termos.

3.2 *Latent Fingerprint Indexing: Fusion of Level 1 and Level 2 Features*

O método descrito em Paulino *et al.* (2013) propõe o uso de *features* de *Level 1* e *Level 2* na indexação de impressões digitais. Tal método utiliza uma combinação de minúcias, com o uso da técnica proposta em Cappelli *et al.* (2010a), combinando com pontos singulares, campo de orientação e outras informações gerais das impressões digitais. Um *score* de similaridade é gerado por cada um dos métodos de indexação, a partir da comparação de uma impressão digital buscada e cada uma das impressões digitais indexadas. Em seguida, os valores dos *scores* são combinados para gerar um *score* final para ranquear as impressões digitais mais semelhantes.

O trabalho de Paulino *et al.* (2013) utiliza várias *features*, incluindo *features* L2, como no trabalho proposto. Em Paulino *et al.* (2013) é utilizado várias métricas de similaridade, diferente do trabalho proposto.

3.3 *A New Similarity Digest Search Strategy Applied to Minutia Cylinder-Codes for Fingerprint Identification*

Em Moia e Henriques (2018) é proposta uma estratégia aproximativa de identificação de impressões digitais denominado MCC-HBFT. Este método também utiliza a representação MCC e uma estrutura de dados chamada *Hierarchical Bloom Filter Tree* (HBFT). O HBTF é uma estrutura baseada em localidade e probabilística usada para armazenar uma grande quantidade de dados utilizando funções *hash* e juntamente com um tipo especial de árvore. Cada função

hash recebe um subvetor e o indexa na estrutura da árvore HBTF.

Para realizar as buscas, cada um dos vetores binários do MCC são separados e utilizados em cada função *hash*. Em seguida, são feitas as buscas em cada uma das HBFT para retornar os elementos com maior similaridade. A abordagem de similaridade do trabalho de Moia e Henriques (2018) é baseada na contagem de de *hashs* que obtiveram *match* na busca e em algumas características armazenadas nas árvores. Moia e Henriques (2018) utilizam *features* L2, como os outros trabalhos citados anteriormente. Além disso, são utilizadas partes de vários conjuntos de dados para avaliar o método, diferente do método do Cappelli *et al.* (2010a) que utiliza cada conjunto de dados separadamente.

3.4 Indexando Impressões Digitais Utilizando Índice Invertido: Uma Investigação Inicial

Soares *et al.* (2020) cria uma representação em forma de documentos textuais que generaliza todas as minúcias de uma impressão digital. Inicialmente, é utilizado MCC para gerar vetores binários de cada impressão digital. Em seguida, cada conjunto de vetores é agrupado para gerar um valor inteiro que é inserido em um documento, como mostrado na Figura 12.

Figura 12 – Criação do documento de termos a partir dos vetores binários

(a) Vetores binários do *Minutia Cylinder-Code*

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9
M_1	0	0	0	1	1	1	0	1	1
M_2	1	0	1	1	1	0	0	0	1
M_3	0	1	1	1	0	1	1	1	0
M_4	1	0	1	1	0	0	0	1	1
M_5	1	1	1	0	1	1	1	0	1

(b) Resultado da função f_H termos gerados

f_H	f_H	f_H
0	7	3
5	6	1
3	5	6
5	4	3
7	3	5

(c) Documento de

2_7 3_3
1_5 2_6 3_1
1_3 2_5 3_6
1_5 2_4 3_3
1_7 2_3 3_5

Fonte: Soares *et al.* (2020)

As etapas de indexação e busca são realizadas utilizando requisições ao Elasticsearch. A indexação é realizada utilizando API *Bulk* para indexar um conjunto de documentos em um índice do Elasticsearch. Na etapa de busca é utilizado o *tf-idf* (JONES, 1972) para indicar a importância de cada palavra no documento, para em seguida, realizar a similaridade entre cada termo entre o documento buscado com cada uma dos documentos indexados. Por fim, o Elasticsearch cria um *ranking* com os documentos mais semelhantes. Portanto, esse método

utiliza *features* L2 em suas etapas e realiza a avaliação com um conjunto de dados privado e outro público.

3.5 Comparativo entre os Trabalhos

No Quadro 1 é apresentada a comparação dos trabalhos citados anteriormente com a técnica proposta neste trabalho. O método utilizado informa qual abordagem foi utilizada na indexação e busca. A abordagem de similaridade é referente a técnica que é utilizada para dar os *scores* para cada impressão digital. Já o tipo de *feature* de impressão digital informa qual tipo de características foram utilizadas na indexação e busca. Por fim, o conjunto de dados utilizado descreve se o conjunto de dados é público e pode ser solicitado ou é privado e não foi publicado.

Quadro 1 – Comparação dos trabalhos relacionados com o trabalho proposto

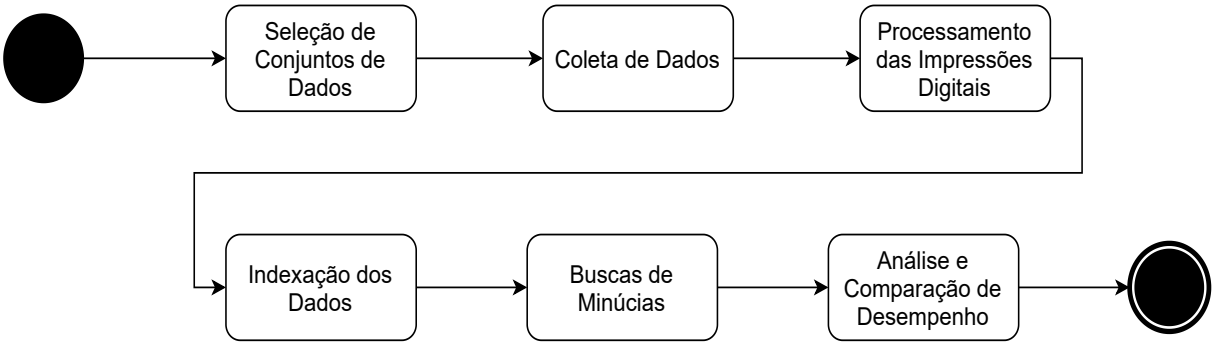
Trabalho	Método Utilizado	Abordagem de Similaridade	Tipo de Features de Impressão Digital	Conjunto de Dados Utilizado
(CAPPELLI *et al.*, 2010a)	Uso do MCC com LSH	Similaridade Hamming	Feature Level 2	Público
(PAULINO *et al.*, 2013)	Combinação de vários métodos	Um método de similaridade por técnica	Feature Level 1 e Level 2	Parcialmente Privada
(MAIO *et al.*, 2004)	Uso do MCC com HBFT	Contagem do número de funções hash com similaridade	Feature Level 2	Público
(SOARES *et al.*, 2020)	Uso do MCC com LSH e Elasticsearch	Utiliza a similaridade *tf-idf* do Elasticsearch	Feature Level 2	Um conjunto de dados público e outro privado
Este Trabalho	Uso do MCC com LSH	Contagem de termos semelhantes	Feature Level 2	Público

Fonte: Elaborado pelo autor

4 PROCEDIMENTOS METODOLÓGICOS

Neste Capítulo são descritas as etapas de execução para atingir os objetivos deste trabalho. A Figura 13 mostra os passos executados neste trabalho que são divididos em: seleção de conjuntos de dados, coleta de dados, processamento das impressões digitais, indexação dos dados, buscas de minúcias e análise e comparação de desempenho.

Figura 13 – Etapa de execução do trabalho



Fonte: Elaborado pelo autor

4.1 Seleção de Conjuntos de Dados

A primeira etapa consiste na seleção de conjuntos de dados de imagens de impressões digitais em conjuntos de dados públicos. Para a realização dos experimentos, são selecionados 2 conjuntos de dados públicos, sendo um deles com uma quantidade menor de impressões digitais e o outro com uma maior quantidade.

4.2 Coleta de Dados

Os conjuntos de dados públicos estão disponíveis em *sites* próprios de cada *dataset* ou é necessário entrar em contato com os distribuidores dos conjuntos de dados. As imagens das impressões digitais que não estão no formato WSQ são convertidas, pois é necessário que todas as imagens estejam neste formato na etapa de processamento das impressões digitais. Por fim, as imagens são separadas em conjunto de indexação, para serem utilizadas como conjunto de impressões digitais já processadas, que são indexadas em estruturas que facilitem o processo de busca. E um conjunto de busca, que consiste nas imagens que são utilizadas na etapa de busca para realizar a avaliação do método.

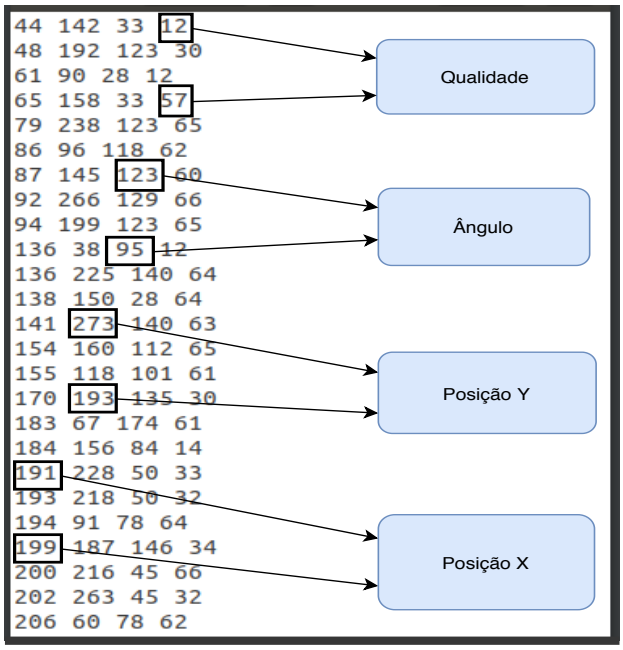
4.3 Processamento das Impressões Digitais

Nesta seção são apresentadas as etapas de processamento das impressões digitais coletadas, sendo divididas em extração de minúcias e criação dos vetores MCC. O processamento é necessário para criar informações úteis que serão utilizadas na indexação e busca das informações.

4.3.1 Extração das Minúcias

Nesta etapa é utilizada a ferramenta MINDTCT do pacote NBIS para gerar um arquivo com as informações cada minúcia. O MINDTCT recebe uma imagem como entrada e salva um arquivo contendo as informações de posição X e Y, ângulo e qualidade das minúcias, como mostrado na Figura 14. A posição X e Y pode assumir valores de acordo com a largura e altura da imagem, já o ângulo pode ter valor entre 0 e 180 e a qualidade tem valores entre 1 e 100.

Figura 14 – Arquivo com informações das minúcias



Fonte: Elaborado pelo autor

Os valores dos ângulos são corrigidos para a etapa seguinte, visto que, a ferramenta MINDTCT considera a escala de 0 a 180°, já o método para criar vetores binários do MCC

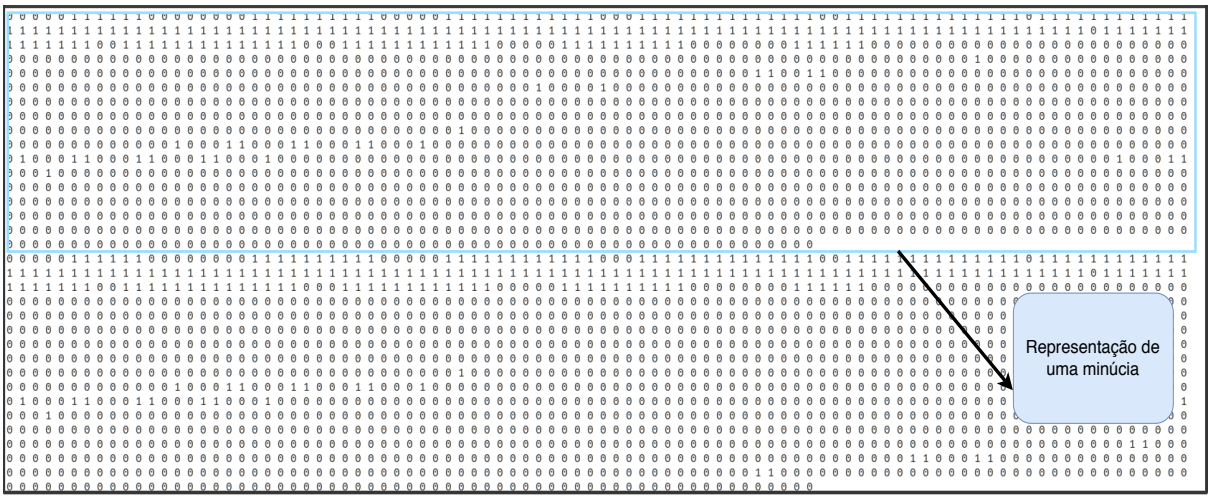
utiliza a escala de radianos. Por esse motivo, é utilizada a Equação 4.1 para realizar a conversão.

$$\hat{A}ngulo\ MCC = \frac{2 * \hat{A}ngulo\ MINDTCT * \pi}{180} \tag{4.1}$$

4.3.2 Criação dos Vetores MCC

Após a criação dos arquivos com as informações das minúcias de cada impressão digital, é utilizado o MCC para gerar uma representação das minúcias no formato de vetor binário de tamanho *n*. Na Figura 15 é possível ver vetores binários que representam minúcias de uma impressão digital.

Figura 15 – Exemplo de vetor binário gerado pelo MCC com tamanho 1328



Fonte: Elaborado pelo autor

Nesta etapa também é utilizado um limite na quantidade de minúcias utilizadas antes de gerar os vetores binários, selecionando as minúcias com maior qualidade. Pois muitas minúcias possuem pouca qualidade e em alguns casos não são minúcias verdadeiras. O uso deste limite também diminui a quantidade de dados que precisam ser processados, gerando menos vetores binários.

4.4 Indexação dos Dados

Na etapa de indexação, os dados gerados na etapa de processamento são transformados em informações textuais para serem indexados. A indexação dos Método 1 e 2 são baseadas no trabalho de Cappelli *et al.* (2010a), dado que é uma abordagem conceituada e que serve como base para vários outros trabalhos de indexação e busca de impressões digitais. Todos os

dados são indexados em estruturas que facilitem a busca para tentar diminuir o tempo de busca. Além disso, as minúcias são indexadas de forma individual, dado que o trabalho de Soares *et al.* (2020) não obteve um resultado de *Penetration Rate* X *Error Rate* semelhante aos trabalhos consolidados, pois as informações textuais foram indexadas generalizando todas as minúcias em um mesmo documento.

4.5 Buscas de Minúcias

Após a indexação, são realizadas buscas com impressões digitais dos mesmos dedos de impressões digitais que foram utilizadas na etapa de indexação. Esta etapa tem o objetivo de descobrir quais impressões digitais são retornadas nas buscas. Cada minúcia é buscada individualmente para ser possível encontrar as minúcias mais semelhantes a cada uma das minúcias buscadas. Por fim, é gerado um ranqueamento das impressões digitais com maiores similaridades com cada uma das impressões digitais buscadas.

4.6 Análise e Comparação de Desempenho

Cappelli *et al.* (2010a), Paulino *et al.* (2013), Moia e Henriques (2018) e Soares *et al.* (2020) utilizam um gráfico com a métrica de *Penetration Rate X Error Rate* para avaliar a quantidade de erros por porcentagem da base. Esta métrica é útil para avaliar a qualidade do resultado das buscas, além de ser possível comparar com outros trabalhos. Neste trabalho também é avaliado o tempo de cada consulta das impressões digitais.

5 RESULTADOS

Neste capítulo são apresentados os resultados obtidos a partir da execução dos procedimentos metodológicos. Inicialmente, são descritos os processos de seleção, coleta e processamento das impressões digitais. Por fim, são descritas as etapas de indexação e buscas das abordagens, juntamente com a análise dos resultados.

Foram desenvolvidas duas abordagens de indexação e busca neste trabalho. O Método 1 utiliza tabelas *hash* para indexar os dados e realizar as buscas. Já o Método 2 utiliza o Elasticsearch como sistema de busca, por meio da sua *Application Programming Interface* (API).

5.1 Seleção de Conjuntos de Dados

Os trabalhos de indexação de impressões digitais que utilizam bases de dados públicas para avaliar o método, geralmente utilizam um ou mais conjuntos de dados e os principais são descritos a seguir:

- *Fingerprint Verification Competition* (FVC): A FVC disponibilizou conjuntos de dados em alguns anos e dividiu cada conjunto de dados em DB1 (*Database 1*), DB2 (*Database 2*), DB3 (*Database 3*) e DB4 (*Database 4*). A FVC 2000 (MAIO *et al.*, 2002a), FVC 2002 (MAIO *et al.*, 2002b), FVC 2004 (MAIO *et al.*, 2004) e FVC 2006 (CAPPELLI *et al.*, 2007) são conjuntos de dados muito utilizados para analisar o desempenho de métodos de *maching* e indexação, porém, são conjuntos de dados pequenos, com 110 dedos em cada DB e 8 imagens por dedo. Totalizando 880 imagens de impressões digitais disponibilizadas em cada ano.
- *NIST Special Database 4* (NIST SD4): O NIST SD4 possui 2 imagens de cada dedo, em um conjunto de 2000 dedos, totalizando 4000 imagens de impressões digitais distribuídas igualmente nas 5 classes descristas em Henry (1905).
- *NIST Special Database 14* (NIST SD14): O NIST SD14 possui 27 mil dedos e 2 imagens de impressão digital por dedo, totalizando 54 mil imagens de impressões digitais. Este conjunto de dados é utilizado para realizar experimentos mais robustos, devido ao maior número de dados.

Os conjuntos de dados FVC 2002 DB1 e NIST SD14 foram selecionados para a realização dos experimentos deste trabalho, pois o primeiro conjunto de dados é bastante utilizado em métodos de indexação e busca de impressões digitais e o segundo possui uma quantidade

maior de imagens.

5.2 Coleta de Dados

O conjunto de dados FVC 2002 DB1 possui 110 dedos com 8 impressões digitais para cada dedo. Porém, em Cappelli *et al.* (2010a) são utilizadas as imagens referentes aos 100 primeiros dedos nos experimentos. Existem 100 impressões digitais consideradas principais que foram indexadas e 700 que foram utilizadas nas buscas. Cappelli *et al.* (2010a) também utiliza o conjunto de dados do NIST SD14 para avaliar o método. Utilizando 27 mil impressões digitais na indexação e as ultimas 2700 na etapa de busca. Portanto, o presente trabalho utiliza as mesmas configurações dos experimentos descritos em Cappelli *et al.* (2010a).

5.3 Processamento das Impressões Digitais

Após isso, foram gerados todos os arquivos contendo as informações das minúcias de cada impressão digital. Após a geração das minúcias é utilizado o MCC para gerar vetores binários. Foram utilizados os valores de 10, 25, 50 e 100 como limites máximos de minúcias utilizadas para realizar um teste exaustivo de parâmetros, juntamente com a variação do tamanho do vetor binário de 1.152, 1.536 e 1.944. O tamanho padrão do vetor binário é 1.536 *bits*, pois obteve melhores resultados de acordo com Cappelli *et al.* (2010b). Portanto, foram selecionados outros dois valores semelhantes ao valor padrão.

5.4 Método 1 - Tabelas *hash*

Este método se assemelha ao trabalho de Cappelli *et al.* (2010a), pois também utiliza tabelas *hash* para armazenar os dados gerados por cada uma das minúcias. Os dados são transformados em textos e indexados nas tabelas *hash* com o uso de uma função de *hash* que associa um segmento do vetor binário com uma posição (*bucket*) de uma tabela *hash*.

5.4.1 Indexação dos Dados

Inicialmente, cada vetor binário j do MCC é dividido em grupos de *h bits*, ou seja, cada sequência de *h bits* em cada vetor binário de *n bits* é agrupada separadamente. Cada *h bits* possui uma posição no vetor binário, informando a localização dentro do vetor, por exemplo, os

h primeiros *bits* estão na posição 1 e assim sucessivamente até a posição *i* no vetor binário.

É utilizada uma função f_H que recebe cada conjunto de *h bits* e retorna um valor inteiro não negativo referente aos *h* bits. Existem também tabelas *hash* que vão de $H_1, H_2, ..., H_i$, com o mesmo valor *i* do número de posições no vetor binário. Cada tabela *hash* está relacionada a uma posição no vetor binário e possui 2^h *buckets*, referente ao valor máximo gerado pela função f_H. Portanto, quando um conjunto de *h* bits que está na posição 1 do vetor binário passa pela função f_H, será retornado um valor inteiro referente ao índice do *bucket* no qual, será inserido o texto "Mj"na tabela *hash* H_1, indicando a inserção dos primeiros *h* bits do vetor j.

As informações inseridas nas tabelas *hash* são no formato textual. Por esse motivo, para cada informação inserida em um *bucket* é adicionado um espaço em branco no texto, pois, são inseridos outros termos referentes as outras minúcias processadas. Além disso, seguindo o método de Cappelli *et al.* (2010a), os conjuntos de bits que possuírem valor zero como retorno da função f_H não são inseridos nas tabelas *hash*, dado que, os vetores binários do MCC possuem muitas posições com valor zero e essas informações não contribuem para a melhoria do resultado. Na Figura 16 é mostrado um exemplo simples do método, com 5 vetores binários de tamanho 9 e os *bits* são agrupados em grupos de 3 bits, portanto, são gerados 3 inteiros não negativos na função f_H para cada vetor binário. Em seguida, é inserida a informação Mj referente a cada minúcia nos *buckets* de cada tabela *hash*.

Em casos reais existe uma grande quantidade de impressões digitais de indivíduos diferentes, sendo preciso identificar um indivíduo utilizando um identificador único (ID). Porém, nos exemplos já citados é inserida apenas a informação de qual minúcia foi processada, sendo preciso inserir também a informação do ID do indivíduo, juntamente com a numeração da minúcia, por exemplo, o texto "15_1"é referente a minúcia 1 do indivíduo com identificador 15. Não sendo preciso inserir o caractere "M", como nos exemplos mostrados anteriormente. O Algoritmo 1 descreve o processo de indexação em casos reais.

Por fim, as tabelas *hash* são avaliadas para encontrar tabelas que possuem textos irrelevantes para à busca. O objetivo da análise é verificar à quantidade de *buckets* utilizados em cada tabela *hash*. Pois, podem existir tabelas sem dados indexados ou com apenas um *bucket* sendo utilizado. Portanto, nestes casos, as tabelas não são utilizadas.

Figura 16 – Exemplo da indexação do textos nas tabelas *hash*

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	f_H	f_H	f_H
M_1	0	0	0	1	1	1	0	1	1	0	7	3
M_2	1	0	1	1	1	0	0	0	1	5	6	1
M_3	0	1	1	1	0	1	1	1	0	3	5	6
M_4	1	0	1	1	0	0	0	1	1	5	4	3
M_5	1	1	1	0	1	1	1	0	1	7	3	5

H_1	
0 - 000_2	" "
1 - 001_2	" "
2 - 010_2	" "
3 - 011_2	"M3"
4 - 100_2	" "
5 - 101_2	"M2 M4 "
6- 110_2	" "
7 - 111_2	" M5"

H_2	
0 - 000_2	" "
1 - 001_2	" "
2 - 010_2	" "
3 - 011_2	"M5 "
4 - 100_2	"M4 "
5 - 101_2	"M3 "
6- 110_2	"M2 "
7 - 111_2	"M1 "

H_3	
0 - 000_2	" "
1 - 001_2	"M2 "
2 - 010_2	" "
3 - 011_2	"M1 M4 "
4 - 100_2	" "
5 - 101_2	"M5 "
6- 110_2	"M3 "
7 - 111_2	" "

Fonte: Elaborado pelo autor

Algoritmo 1: Abordagem de Indexação do Método 1

Entrada:
Um conjunto de imagens de impressões digitais, $DB = \{F_1, F_2, ...\}$
Tamanho do vetor binário n
Tamanho do grupo de bits h
Threshold de qualidade T_q
Saída:
Tabelas *hash* $H_1, H_2, ..., H_i$
início
  $i = n/h$
  Cria i tabelas *hash* com 2^h *buckets* cada
  para *cada imagem de impressão digital* F_k *em DB* **faça**
    Gere o arquivo de minúcias da imagem F_k utilizando MINDTCT
    Use o MCC para criar os vetores binários V_k do arquivo de minúcias de F_k removendo minúcias com qualidade menor que T_q
    para *cada vetor* v_j *em* V_k **faça**
      para *cada subvetor* p_k *de h bits no vetor* v_j **faça**
        $b = f_H(p_k)$
        if $b > 0$ **then**
          Insere o texto "k_j"no bucket b da tabela *hash* k;
        end
      fim
    fim
  fim
  Retorna as tabelas *hash*
fim

5.4.2 Busca das Impressões Digitais

Depois da indexação e análise das tabelas *hash*, são realizadas as buscas para encontrar os dados mais semelhantes com os dados das impressões digitais buscadas. A indexação

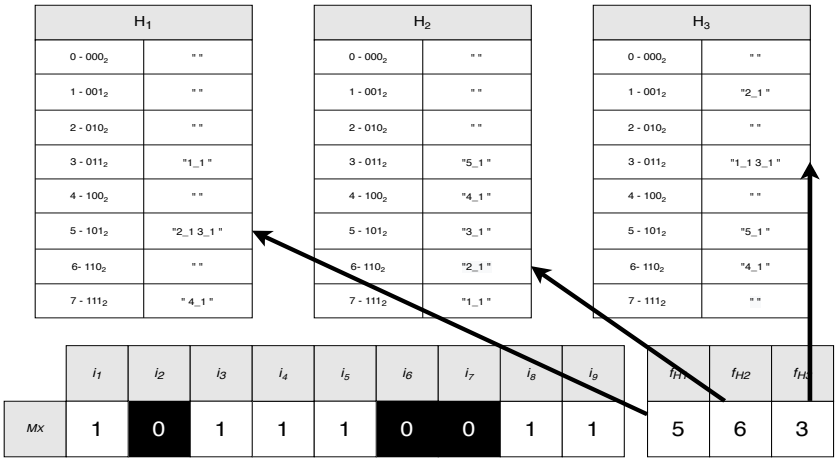
pode ser considerada como baseada em minúcia, pois os dados inseridos nas tabelas *hash* são referentes a cada minúcia das impressões digitais. Por isso, as buscas também serão realizadas com informações extraídas das minúcias.

Depois da criação dos arquivos de minúcias e dos vetores binários do MCC no conjunto de dados de busca, cada vetor é dividido em grupos de h bits e é utilizada a mesma função f_H para retornar o valor inteiro b de cada um dos h bits. Depois, o *bucket* na posição b da tabela *hash* referente ao grupo é acessado e o texto armazenado é retornado. Como exitem w grupos, podem ser retornados w textos para cada minúcia. Em seguida, os termos dos w textos são separados pelo espaço, gerando uma lista de termos, no qual as ocorrências serão contadas. Os termos que possuem valor máximo de contagem são considerados os mais semelhantes.

Os termos são compostos por 2 valores numéricos separados por "_", onde o primeiro é referente ao identificador da impressão digital e o segundo referente a minúcia. Então, para identificar a impressão digital mais semelhante é utilizado apenas o primeiro termo. Portanto, apenas o primeiro valor numérico de cada termo com maior ocorrência será retornado.

Na Figura 17 é mostrado um exemplo da busca em 3 tabelas *hash* de apenas um vetor binário referente a uma minúcia M_x. No exemplo, o vetor binário possui 9 *bits* que são divididos em 3 grupos de 3 bits. Depois do uso da função f_H são gerados 3 inteiros que são utilizados como índices dos *buckets* de cada tabela *hash*. Portanto, os textos são acessados, os termos são separados e contados, para em seguida, o primeiro valor numérico de cada termos com maior ocorrência serem retornados. Neste exemplo, será retornado "3 2 ", referentes aos identificadores das impressões digitais que possuem alguma minúcia muito semelhante com a minúcia buscada.

Figura 17 – Exemplo da busca de uma minúcia nas tabelas *hash*



Fonte: Elaborado pelo autor

Essa busca é realizada para vetores binários de cada minúcia, gerando vários textos

com os identificadores das impressões digitais que possuem minúcias semelhantes. Portanto, é necessário realizar outra contagem com os identificadores retornados para ranquear as impressões digitais mais semelhantes baseadas nas ocorrências dos termos.

A abordagem de indexação utiliza princípios de LSH para agrupar subvetores semelhantes de *h* bits nos mesmos *buckets*. Portanto, as buscas por minúcias nas tabelas *hash* só irão acessar um conjunto de dados reduzidos, pois é acessado apenas *buckets* específicos de cada tabela *hash*.

5.5 Método 2 - Elasticsearch

O segundo método utiliza o índice invertido do Elasticsearch para indexar os dados textuais das minúcias. Na etapa de busca, é utilizado scripts de similaridade para realizar exatamente o mesmo processamento do método 1, porém utilizando funções já implementadas do Elasticsearch.

5.5.1 Indexação dos Dados

A indexação do Método 2 possui grandes semelhanças com o Método 1, dado que consideramos a modelagem da mesma abordagem utilizando um engenho de busca textual. Porém, o Elasticsearch utiliza uma abordagem orientada a documentos. Por esse motivo, é realizada a modelagem para indexar as informações de cada impressão digital em documentos diferentes. Neste método, é indexado no documento a informação da posição *i* do subvetor no vetor binário, juntamente com o valor *b* retornado pela função f_H. Portanto, as informações textuais serão neste formato *i_b*, considerando *i* como o mesmo valor que o índice da tabela *hash* e *b* o *bucket* no Método 1. Além disso, o Método 1 utiliza a indexação de minúcias de forma individual, por esse motivo, a indexação no Elasticsearch também é realizada considerando cada minúcia de forma individual.

Para isolar cada minúcia, foram criados subdocumentos dentro do documento que representa uma impressão digital. A Figura 18 mostra o padrão do documento usado no Elasticsearch nesta abordagem. Consistindo de um documento chamado *fingerprint* com as informações do identificador da impressão digital, juntamente com uma lista de minúcias representada pelo tipo *nested*. Cada minúcia possui o texto que será indexado e o identificador da minúcia.

Figura 18 – Formato do documento usado no Elasticsearch

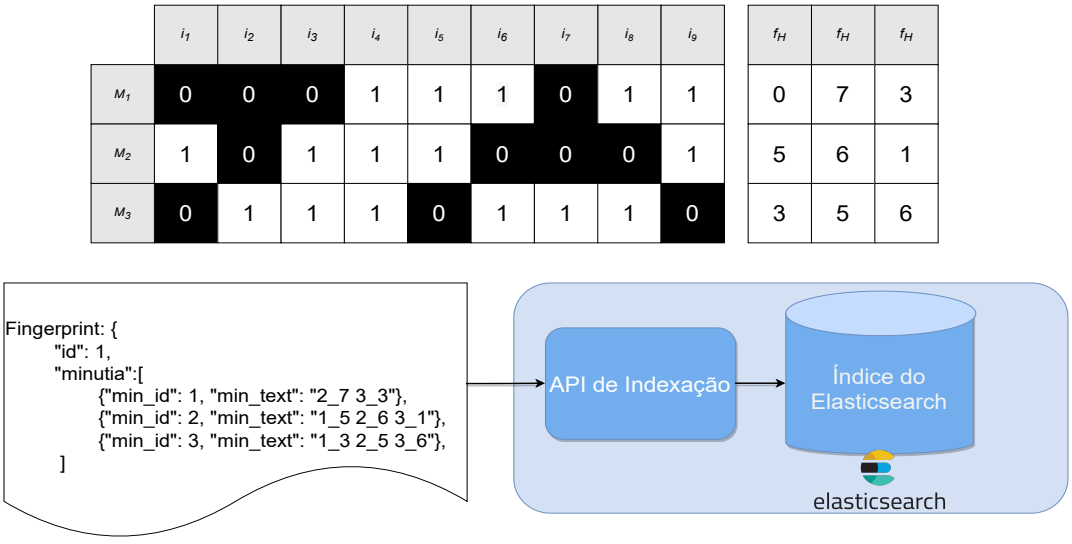
```
{
    "fingerprint":{
        "id" :{"type": "text"},
        "minutiae":{
            "type": "nested",
            "properties": {
                "min_text": {
                    "type": "text"
                },
                "min_id": {
                    "type": "text"
                }
            }
        }
    }
}
```

Fonte: Elaborado pelo autor

O Elasticsearch utiliza um sistema de busca chamado Lucene[1] por baixo das suas camadas de aplicação e neste caso, cada subdocumento com os textos das minúcias são considerados um documento isolado no índice do Lucene. Por esse motivo, todos as minúcias são isoladas uma das outras dentro do índice.

Para realizar a indexação é utilizada uma API já implementada do Elasticsearch para indexar os documentos em um índice. A Figura 19 mostra 3 vetores binários, com o valor de h igual a 3. Em seguida, é utilizada uma função f_H para gerar os valores inteiros que são utilizados para criar o documento com os dados gerados de cada minúcia. Por fim, a API de indexação é chamada para armazenar o documento em um índice do Elasticsearch.

Figura 19 – Exemplo da indexação dos textos no Elasticsearch



Fonte: Elaborado pelo autor

[1] *https://lucene.apache.org/*

5.5.2 *Busca das Impressões Digitais*

Na etapa de buscas, são utilizados os vetores binários das impressões digitais que são buscadas. Todos os vetores são separados em grupos de bits, com a utilização da mesma função f_H para retornar o valor inteiro do conjunto de bits. Depois, é elaborado uma requisição no Elasticsearch para cada minúcia de cada impressão digital.

É utilizado o tipo de busca *multi search* do Elasticsearch. Esse tipo de busca realiza várias requisições em paralelo para o Elasticsearch e retorna todos os *rankings* na mesma requisição. Cada minúcia buscada é comparada com cada subdocumento dentro dos documentos indexados no índice. Por esse motivo, é retornado apenas o score do subdocumento com os termos mais semelhantes dentro de um documento. A Figura 20 mostra a *query* para uma minúcia, que consiste em uma busca do tipo *nested* sobre a lista de subdocumentos das minúcias e realiza a busca do texto de uma minúcia e retorna o *score* máximo de cada um dos documentos.

Figura 20 – *Query* para uma minúcia no Elasticsearch

```
{ "query": {
    "nested": {
        "path": "minutia",
        "query": {
            "bool": {
                "should": [
                    {"match": {"minutia.text": "texto da minúcia"}},

                ]
            }
        },
        "score_mode": "max"
    }
```

Fonte: Elaborado pelo autor

Para cada termo no texto de cada uma das minúcias é executado o *script* da Figura 21. O método de similaridade consiste na frequência que o termo buscado aparece em cada um dos subdocumentos, multiplicado por um *fator* baseado na quantidade de documentos que possui o termo. Esse método de similaridade faz exatamente a abordagem de contagens de termos do Método 1 e faz a filtragem de termos que estão contidos em todas as minúcias, semelhante à remoção de tabelas *hash* com termos no mesmo *bucket*.

O Elasticsearch cria um *ranking* baseado nos *scores* dos documentos para cada

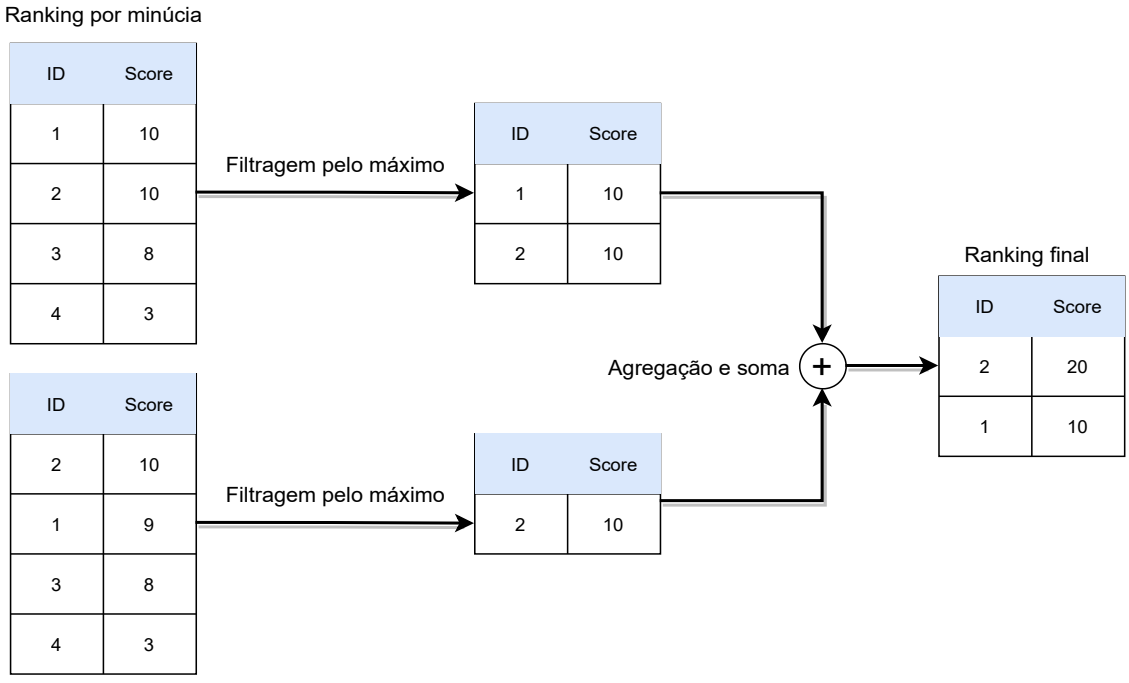
Figura 21 – *Script* da busca utilizada no Elasticsearch

```
{   "similarity":{
        "script_tf_idf":{
            "type": "scripted",
            "script": {
                "source": "
                    double idf = Math.log((field.docCount+1.0)/
                        (term.docFreq+1.0));
                    double t_idf = Math.ceil(idf);
                    return doc.freq*t_idf;
                    "
            }
        }
    }
}
```

Fonte: Elaborado pelo autor

minúcia buscada. Por fim, os documentos retornados são filtrados pelo valor máximo de cada *score*, para em seguida, serem agrupados pelo identificador do usuário e ordenados pela soma dos *scores*. Portanto, é criado um *ranking* único a partir de várias buscas de minúcias. A Figura 22 mostra um exemplo de criação do *ranking* final, no qual é passado os *rankings* de duas minúcias e é criado o *ranking* final a partir da filtragem de máximo, agregação e soma dos *scores*.

Figura 22 – Exemplo da criação do *ranking* final no Método 2



Fonte: Elaborado pelo autor

5.6 Análise e Comparação de Desempenho

Os experimentos de indexação e busca foram realizados em uma máquina Ubuntu 20 com Intel i5 8 CPUs virtuais e contendo 16 GB de memória RAM, . Foram realizados experimentos com os datasets FVC 2002 DB1a e NIST SD14.

5.6.1 Conjuntos de dados FVC 2002 DB1a

Cada vetor binário criado foi dividido em grupos de *h bits*, onde o valor de *h* foi modificado para os valores 8, 12 e 16. Portanto, cada termo criado a partir de cada subvetor foi inserido em um *bucket* em sua respectiva tabela *hash*. A quantidade de tabelas *hash* criadas para cada caso, depende do tamanho do vetor binário e da quantidade de subvetores gerados.

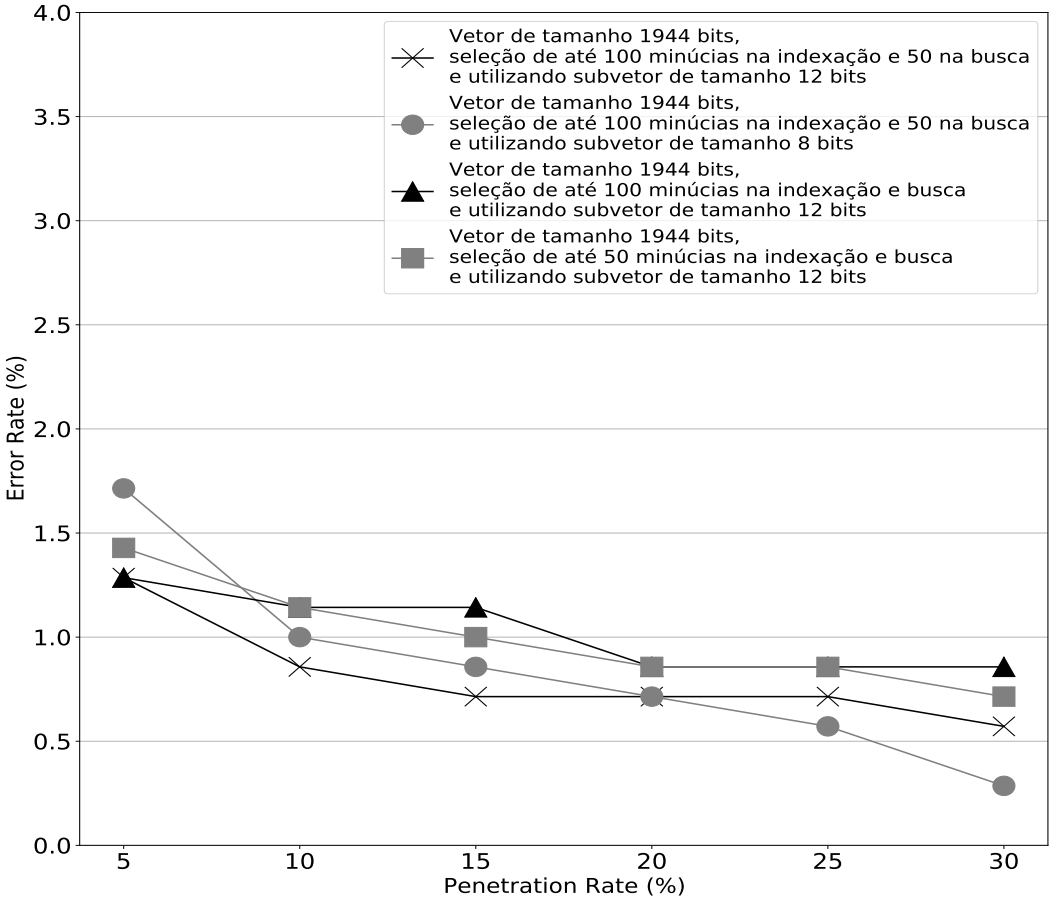
Considerando todos os casos do teste exaustivo, foram analisados todos os *buckets* de todos os casos. Foi constatado o valor de mediana de 11 tabelas *hash* sem texto para cada caso. Além disso, 20 tabelas *hash* tiveram todos os textos no mesmo *bucket*. Esse tipo de situação não colabora para a busca, pois não existem informações para serem recolhidas das tabelas *hash* ou toda a informação está alocada no mesmo *bucket* e será realizado mais processamento para encontrar os casos mais similares, aumentando o tempo de busca.

Foram realizadas 700 buscas de impressões digitais da FVC 2002 DB1 por configuração. Para cada combinação dos parâmetros foi gerado um arquivo com dados de cada impressão digital buscada e a posição do *ranking* que se localiza a impressão digital correta, juntamente com o tempo de busca. Portanto, foram indexadas informações referentes a 100 impressões digitais da FVC 2002 DB1 em cada variação dos parâmetros. Os textos inseridos nas tabelas *hash* possuem identificadores das impressões digitais com valores de 1 a 100, juntamente com os identificadores de cada minúcia. Foi feita uma análise nos arquivos gerados, utilizando o cálculo do *Error Rate* para 10% de *Penetration Rate* para selecionar a melhor combinação de parâmetros com menor *Error Rate*.

A Figura 23 mostra as combinações de parâmetros com melhores valores de *Error Rate* com 10% de *Penetration Rate* no Método 1. Além disso, também são mostrados os valores de erro até 30% de *Penetration Rate*. Dentre os parâmetros combinados, a combinação com o melhor resultado em 10% de *Penetration Rate* obteve *Error Rate* de 0,85% com o seguintes parâmetros: Vetor MCC de tamanho 1.944 bits, com seleção das 100 minúcias com maior qualidade na indexação e 50 minúcias para a busca. Utilizando agrupamento de 12 bits em cada

subvetor.

Figura 23 – Gráfico dos resultados dos melhores parâmetros do Método 1 na FVC 2002 DB1



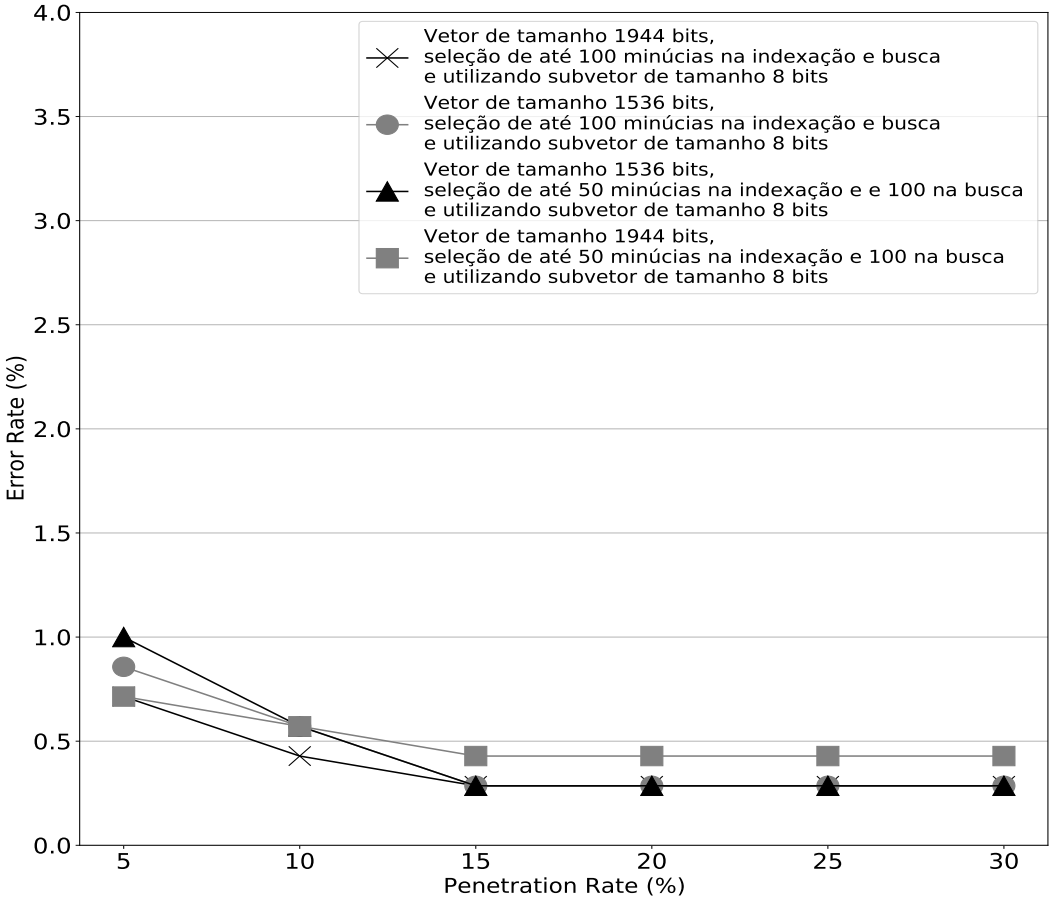
Fonte: Elaborado pelo autor

O Método 2 obteve resultados superiores neste mesmo conjunto de dados. Por exemplo, o melhor resultado do Método 2, obteve 0,42% de *Error Rate* também utilizando 10% de *Penetration Rate*, selecionando as 100 minúcias com maior qualidade na indexação e busca, agrupando os vetores binários de tamanho 1944 em 8 bits por subvetor. A Figura 24 apresenta os melhores resultados do Método 2.

Os casos em que foram indexados apenas 10 minúcias por impressão digital obtiveram os piores resultados de *Error Rate*. O pior resultado dos dois métodos obteve o *Error Rate* de 40,7% utilizando o tamanho do vetor de 1.944 bits, 16 bits do tamanho de cada subvetor e seleção de minúcias 10 minúcias na indexação e 100 para a busca.

O tempo por consulta também é um fator muito importante que precisa ser analisado em métodos de busca. A Figura 25 mostra um *boxplot* com as informações de tempo por consulta para cada quantidade de bits utilizadas para agrupar os vetores. Todas as consultas realizadas

Figura 24 – Gráfico dos resultados dos melhores parâmetros do Método 2 na FVC 2002 DB1

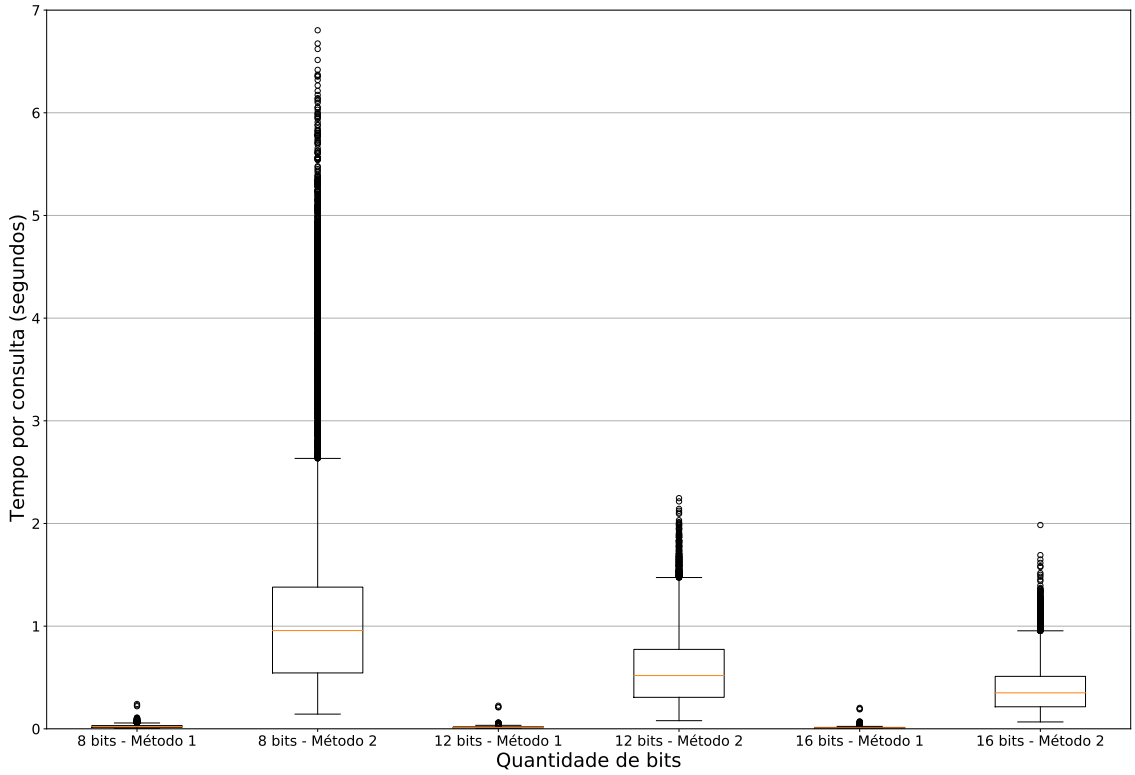


Fonte: Elaborado pelo autor

com cada quantidade de bits para todas as outras combinações dos parâmetros foram agregadas para contabilizar os tempos necessários para a realização de cada consulta. Pela Figura é possível concluir que na base FVC 2002 DB1 houve uma diminuição do tempo por consulta quando a quantidade de bits para agrupar os vetores foram aumentadas. O Método 2, que utiliza o Elasticsearch necessitou de mais tempo para realizar as consultas em todos os casos, comparando com o tempo gasto no Método 1. Dado que, o processamento de poucos dados no Elasticsearch pode ser mais lento com a distribuição dos *shards*.

O trabalho de Moia e Henriques (2018) utiliza parcialmente o conjunto de dados utilizado neste trabalho e o trabalho de Paulino *et al.* (2013) não utiliza o mesmo conjunto de dados. Além disso, não foi possível executar estes métodos, pois os códigos-fonte não foram divulgados. Portanto, na Figura 26 foi realizada a comparação do trabalho proposto com os trabalhos de Kavati *et al.* (2017), Khodadoust e Khodadoust (2017), Cappelli *et al.* (2010a), Feng e Cai (2006) e Soares *et al.* (2020) utilizando a relação de *Error Rate* com *Penetration Rate*. A abordagem proposta no Método 2 obteve um resultado superior aos trabalhos comparados. O

Figura 25 – Gráfico de tempo por consulta para cada tamanho de subvetor



Fonte: Elaborado pelo autor

Método 1 também obteve um bom resultado na métrica.

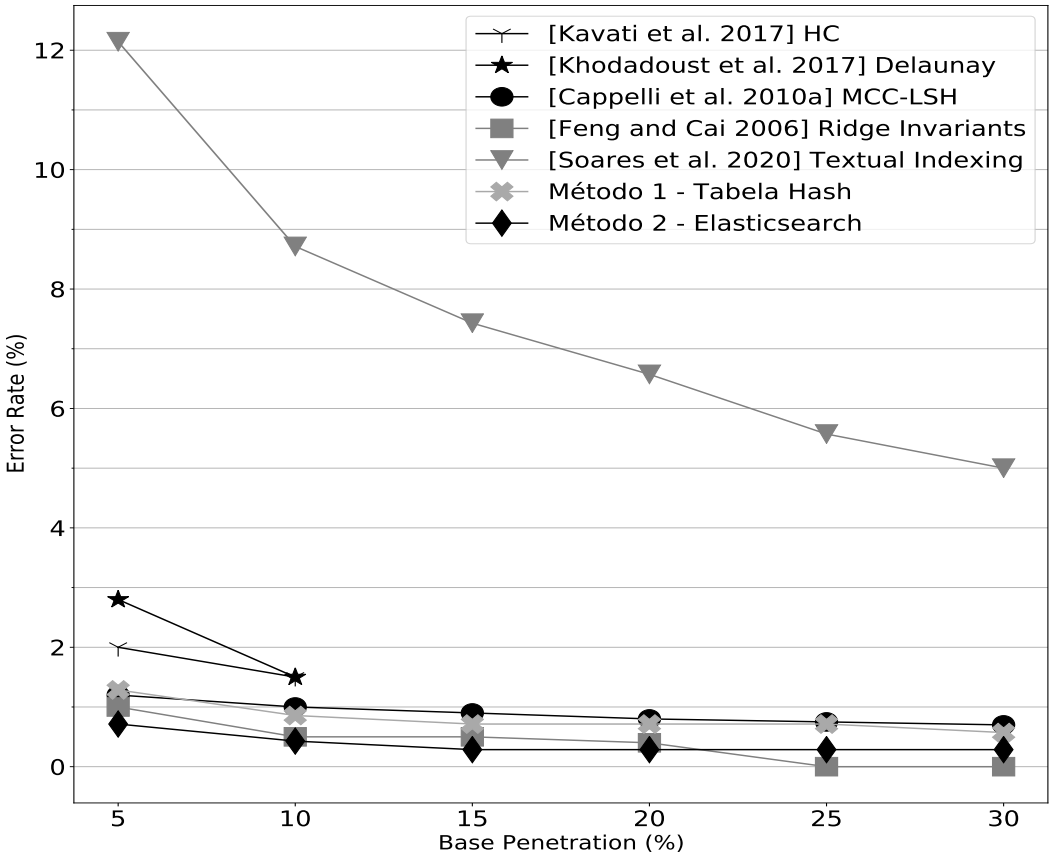
A maioria dos métodos comparados possui bons resultados de tempo de busca. Por exemplo, Cappelli *et al.* (2010a) realiza buscas em cerca de 2 milissegundos na FVC 2002 DB1a, já Soares *et al.* (2020) obteve 0,2 segundos em um conjunto de impressões digitais maior que o utilizado nessa avaliação. O Método 1 obteve um resultado de tempo de busca semelhante, porém, o resultado do Método 2 foi muito inferior ao tempo de Cappelli *et al.* (2010a).

5.6.2 Conjuntos de dados NIST SD14

Neste experimento, os vetores binários também foram divididos em grupos de 8, 12 e 16 bits e foi configurado todos os parâmetros para realizar o teste exaustivo. Seriam indexadas 27.000 impressões digitais e seria buscadas 2.700 outras impressões digitais.

Porém, no Método 1, com o aumento na quantidade de textos inseridos nas tabelas *hash*, houve um grande aumento no tempo de indexação, demorando horas para realizar uma inserção. Além disso, o tempo médio de busca foi de 1 hora e 45 minutos em uma parcela de 17.000 impressões digitais do conjunto de dados NIST SD14. Tornando a utilização do Método 1 inviável para conjunto de dados maiores.

Figura 26 – Gráfico de comparação de desempenho no FVC 2002 DB1



Fonte: Elaborado pelo autor

A Abordagem 2 conseguiu utilizar a facilidade de indexação do Elasticsearch para indexar as 27.000 impressões digitais do conjunto de dados, consumindo apenas 2GB de memória RAM para armazenar toda a informação. Porém, o tempo de busca em um grande conjunto de dados também foi extremamente alto, gastando o tempo médio de 2 horas e 5 minutos para realizar uma busca.

Portanto, os métodos propostos obtiveram resultados satisfatórios em um pequeno conjunto de dados, porém, com o aumento dos dados, as abordagens não escalaram para realizar as buscas em tempo hábil.

6 CONCLUSÕES E TRABALHOS FUTUROS

A indexação e busca de características biométricas são fundamentais para o reconhecimento e identificação de indivíduos. Neste trabalho, foram propostos dois métodos de indexação e busca de impressões digitais utilizando documentos de textos gerados a partir de imagens de impressões digitais. Uma das abordagens é baseada no trabalho de Cappelli *et al.* (2010a) e utiliza LSH para agrupar os textos em tabelas *hash*. A outra abordagem proposta utiliza o índice invertido do Elasticsearch para indexar os documentos.

Este trabalho contribui com a utilização de técnicas aplicadas em buscas textuais no domínio de impressões digitais. São apresentadas várias etapas de processamento para a criação de dados textuais. Além de aplicar a mudança sugerida em Soares *et al.* (2020), pois a indexação e busca de minúcias são realizadas de forma individual, ou seja, cada minúcia é buscada separadamente neste trabalho.

Na etapa de avaliação, os dois métodos propostos obtiveram resultados satisfatórios considerando a métrica de *Penetration Rate* X *Error Rate* no conjunto de dados da FVC 2002 DB1a, com resultados de 0,85% de *Error Rate* no Método 1 e 0,42% do Método 2. Porém, com o experimento no conjunto do NIST SD14, foi constatado que as abordagens propostas sofreram problemas de escalabilidade e não funcionaram para conjuntos de dados maiores. Portanto, a grande desvantagem das abordagens são os tempos de busca que crescem de maneira não escalável com o aumento da quantidade de dados indexados.

Para trabalhos futuros, pretende-se explorar codificações de texto que diminuam a quantidade de dados que precisem ser processadas, além de utilizar estruturas de dados de mais baixo nível para diminuir o tempo de processamento. Por outro lado, pode ser utilizadas abordagens com Spark[1] e a linguagem de programação Scala para paralelizar e distribuir alguma abordagem semelhante ao método utilizado. A realização de uma busca por minúcia gerou um aumento significativo no tempo de busca. Para diminuir esse tempo, também planeja-se investigar o uso de *Deep Learning* (SCHUCH *et al.*, 2018) para diminuir a quantidade de textos gerados. Pois a partir das imagens de impressões digitais ou dos vetores do MCC, podem ser criados vetores denominados *embeddings* (SONG *et al.*, 2019) com uma quantidade menor de dimensões. Esses *embeddings* podem ser utilizados para criar uma informação textual em menor quantidade para serem indexados no Elasticsearch ou na estrutura de tabela *hash* para serem recuperados com apenas uma única busca.

[1] *https://spark.apache.org/*

REFERÊNCIAS

BAEZA-YATES, R.; RIBEIRO-NETO, B. *et al.* **Modern information retrieval**. [*S. l.*]: ACM press New York, 1999. v. 463.

BRISLAWN, C. M.; BRADLEY, J. N.; ONYSHCZAK, R. J.; HOPPER, T. Fbi compression standard for digitized fingerprint images. *In*: APPLICATIONS OF DIGITAL IMAGE PROCESSING, 9., 1996, Denver. **Applications of Digital Image Processing XIX**. Denver: International Society for Optics and Photonics, 1996. v. 2847, p. 344–355.

CAPPELLI, R.; FERRARA, M.; FRANCO, A.; MALTONI, D. Fingerprint verification competition 2006. **Biometric Technology Today**, Elsevier, v. 15, n. 7-8, p. 7–9, 2007.

CAPPELLI, R.; FERRARA, M.; MALTONI, D. Fingerprint indexing based on minutia cylinder-code. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 33, n. 5, p. 1051–1057, 2010.

CAPPELLI, R.; FERRARA, M.; MALTONI, D. Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 32, n. 12, p. 2128–2141, 2010.

CHIKKERUR, S.; RATHA, N. Impact of singular point detection on fingerprint matching performance. *In*: IEEE WORKSHOP ON AUTOMATIC IDENTIFICATION ADVANCED TECHNOLOGIES, 4., 2005, Buffalo. **Fourth IEEE Workshop on Automatic Identification Advanced Technologies (AutoID'05)**. Buffalo: IEEE, 2005. p. 207–212.

CUTTING, D.; PEDERSEN, J. Optimization for dynamic inverted index maintenance. *In*: ACM SIGIR INTERNATIONAL CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 13., 1989, Brussels. **Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval**. New York, NY, USA: Association for Computing Machinery, 1989. p. 405–411.

FELDMAN, R.; SANGER, J. *et al.* **The text mining handbook**: advanced approaches in analyzing unstructured data. [*S. l.*]: Cambridge university press, 2007.

FENG, J.; CAI, A. Fingerprint indexing using ridge invariants. *In*: INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 18., 2006, Hong Kong. **18th International Conference on Pattern Recognition (ICPR'06)**. Hong Kong: IEEE, 2006. v. 4, p. 433–436.

GIONIS, A.; INDYK, P.; MOTWANI, R. *et al.* Similarity search in high dimensions via hashing. *In*: INTERNATIONAL CONFERENCE VERY LARGE DATA BASES, 25., 1999, Edinburgh. **Proceedings of the 25th International Conference Very Large Data Bases**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. v. 99, n. 6, p. 518–529.

GORMLEY, C.; TONG, Z. **Elasticsearch**: the definitive guide. [*S. l.*]: O'Reilly Media, Inc., 2015.

HENRY, E. R. **Classification and uses of finger prints**. [*S. l.*]: HM Stationery Office, 1905.

HOLDER, E. H.; ROBINSON, L. O.; LAUB, J. H. **The Fingerprint Sourcebook**. [*S. l.*]: US Department. of Justice, Office of Justice Programs, National Institute of Justice, 2011.

JONES, K. S. A statistical interpretation of term specificity and its application in retrieval. **Journal of documentation**, MCB UP Ltd, 1972.

JUNIOR, G. **A papiloscopia nos locais de crime**: dactiloscopia, quiroscopia, podoscopia. [*S. l.*]: São Paulo: Editora Ícone, 1991.

KAVATI, I.; PRASAD, M. V.; BHAGVATI, C. Hierarchical decomposition of extended triangulation for fingerprint indexing. *In*: **Efficient Biometric Indexing and Retrieval Techniques for Large-Scale Systems**. [*S. l.*]: Springer, 2017. p. 21–40.

KHODADOUST, J.; KHODADOUST, A. M. Fingerprint indexing based on expanded delaunay triangulation. **Expert Systems with Applications**, Elsevier, v. 81, p. 251–267, 2017.

KO, K. **User's guide to NIST biometric image software (NBIS)**. [*S. l.*], 2007.

KOMARINSKI, P. **Automated fingerprint identification systems (AFIS)**. [*S. l.*]: Elsevier, 2005.

LEVI, G.; SIROVICH, F. Structural descriptions of fingerprint images. **Information Sciences**, Elsevier, v. 4, n. 3-4, p. 327–355, 1972.

LI, H.; ZHAO, T.; LI, N.; CAI, Q.; DU, J. Feature matching of multi-view 3d models based on hash binary encoding. **Neural Network World**, Czech Technical University in Prague, Faculty of Transportation Sciences, v. 27, n. 1, p. 95, 2017.

LIN, J.; DYER, C. Data-intensive text processing with mapreduce. **Synthesis Lectures on Human Language Technologies**, Morgan & Claypool Publishers, v. 3, n. 1, p. 1–177, 2010.

MAIO, D.; MALTONI, D.; CAPPELLI, R.; WAYMAN, J. L.; JAIN, A. K. Fvc2000: Fingerprint verification competition. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 24, n. 3, p. 402–412, 2002.

MAIO, D.; MALTONI, D.; CAPPELLI, R.; WAYMAN, J. L.; JAIN, A. K. Fvc2002: Second fingerprint verification competition. *In*: IEEE. **Object recognition supported by user interaction for service robots**. [*S. l.*], 2002. v. 3, p. 811–814.

MAIO, D.; MALTONI, D.; CAPPELLI, R.; WAYMAN, J. L.; JAIN, A. K. Fvc2004: Third fingerprint verification competition. *In*: INTERNATIONAL CONFERENCE ON BIOMETRIC AUTHENTICATION, 1., 2004, Hong Kong. **Proceedings of the first international conference on biometric authentication**. Hong Kong: Springer, 2004. p. 1–7.

MALTONI, D.; MAIO, D.; JAIN, A. K.; PRABHAKAR, S. **Handbook of fingerprint recognition**. [*S. l.*]: Springer Science & Business Media, 2009.

MANSFIELD, A. J.; WAYMAN, J. L. **Best practices in testing and reporting performance of biometric devices**. [*S. l.*], 2002.

MELUCCI, M.; BAEZA-YATES, R. **Advanced topics in information retrieval**. [*S. l.*]: Springer Science & Business Media, 2011. v. 33.

MIMAROGLU, S.; SIMOVICI, D. A. Approximate computation of object distances by locality-sensitive hashing. *In*: INTERNATIONAL CONFERENCE ON DATA MINING, 4., 2008, Las Vegas. **Proceedings of the 2008 International Conference Data Mining**. Las Vegas: CSREA Press, 2008. p. 714–718.

MOIA, V. H. G.; HENRIQUES, M. A. A. A new similarity digest search strategy applied to minutia cylinder-codes for fingerprint identification. *In*: SIMPÓSIO BRASILEIRO EM SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS, 18., 2018, Natal. **Anais Principais do XVIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais**. Natal: SBC, 2018. p. 99–112.

PAULINO, A. A.; LIU, E.; CAO, K.; JAIN, A. K. Latent fingerprint indexing: Fusion of level 1 and level 2 features. *In*: INTERNATIONAL CONFERENCE ON BIOMETRICS: THEORY, APPLICATIONS AND SYSTEMS (BTAS), 6., 2013, Arlington. **2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)**. Arlington: IEEE, 2013. p. 1–8.

RICARDO, B.-Y.; BERTHIER, R.-N. **Modern information retrieval**: the concepts and technology behind search. [*S. l.*]: Addison-Wesley Professional, Pearson Education, 2011.

ROBERTSON, S.; ZARAGOZA, H.; TAYLOR, M. Simple bm25 extension to multiple weighted fields. *In*: ACM INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 13., 2004, Washington. **Proceedings of the thirteenth ACM international conference on Information and knowledge management**. New York, NY, USA: Association for Computing Machinery, 2004. p. 42–49.

SCHUCH, P.; MAY, J. M.; BUSCH, C. Learning neighbourhoods for fingerprint indexing. *In*: INTERNATIONAL IEEE CONFERENCE ON SIGNAL-IMAGE TECHNOLOGIES AND INTERNET-BASED SYSTEM, 14., Las Palmas de Gran Canaria. **2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)**. Las Palmas de Gran Canaria: IEEE, 2018. p. 48–55.

SCHÜTZE, H.; MANNING, C. D.; RAGHAVAN, P. **Introduction to information retrieval**. [*S. l.*]: Cambridge University Press Cambridge, 2008. v. 39.

SOARES, J. M. S.; BARBOSA, L.; REGO, P. A. L.; MAGALHÃES, R. P.; MACÊDO, J. A. F. de. Indexando impressões digitais utilizando índice invertido: uma investigação inicial. *In*: SIMPÓSIO BRASILEIRO DE BANCOS DE DADOS, 35., 2020, Rio de Janeiro. **Anais do XXXV Simpósio Brasileiro de Bancos de Dados**. Rio de Janeiro: SBC, 2020. p. 181–186.

SONG, D.; TANG, Y.; FENG, J. Aggregating minutia-centred deep convolutional features for fingerprint indexing. **Pattern Recognition**, Elsevier, v. 88, p. 397–408, 2019.

WEGNER, P. A technique for counting ones in a binary computer. **Communications of the ACM**, ACM New York, NY, USA, v. 3, n. 5, p. 322, 1960.