Universidade Federal do Agreste de Pernambuco – UFAPE
Bacharelado em Ciência da Computação

ALLYSSON GUIMARÃES DOS SANTOS SILVA
HERMYSON CASSIANO DE MEDEIROS OLIVEIRA
JOANNE GABRIELA DOS SANTOS SILVA
MANOEL GUSTAVO GALINDO DE SALES

Artigo

Uma proposta para identificar duplicidade em um *dataset* de informações sobre músicas, envolvendo técnicas de blocagem, distância de Levenshtein e clusterização com K-means

Projeto de Banco de Dados 2022.1

Profº.: PRISCILLA KELLY MACHADO VIEIRA AZEVEDO

Garanhuns, 2022.1

RESUMO

O presente artigo apresenta uma técnica para lidar com a alta taxa de duplicidade em uma base de dados de informações sobre músicas. O processo é composto por três etapas principais: a primeira é a blocagem dos dados, que consiste na divisão dos registros em grupos de acordo com alguma característica em comum. Em seguida, é feita a comparação entre os registros dentro de cada bloco utilizando a distância de Levenshtein para calcular a similaridade entre eles. Por fim, os registros similares são agrupados em clusters, definidos através do algoritmo K-means, que apresentam as faixas mais prováveis de serem duplicatas em um mesmo conjunto. Os resultados mostraram que a técnica proposta foi capaz de os registros duplicados na base de dados. A técnica pode ser aplicada em outras bases de dados de informações sobre músicas que requerem a deduplicação de dados.

1. INTRODUÇÃO

A identificação de duplicatas de dados é um processo essencial para garantir a qualidade das informações armazenadas em bancos de dados. No contexto de informações sobre músicas, a presença de registros duplicados pode prejudicar a precisão de análises e recomendações feitas a partir desses dados. Uma técnica bastante utilizada para identificar registros duplicados é a blocagem, que agrupa registros similares em blocos para facilitar a comparação. Neste artigo, apresentaremos uma abordagem que combina blocagem, clusterização com K-means e a distância de Levenshtein para encontrar dados repetidos numa base de dados de informações sobre músicas. A técnica proposta é capaz de identificar registros com diferenças sutis de grafia ou formatação, o que a torna especialmente útil em casos onde a qualidade dos dados pode variar significativamente. Ao final do artigo, serão apresentados resultados experimentais que demonstram a eficácia da abordagem para o banco de dados escolhido para esse contexto.

2. TRABALHOS RELACIONADOS

A identificação de duplicatas de dados é um problema bem conhecido na área de gerenciamento de dados, e várias abordagens têm sido propostas para resolver esse problema. A técnica de blocagem é uma das mais populares, e tem sido aplicada em diferentes domínios, incluindo dados sobre músicas. Por exemplo, em [1], os autores propuseram uma abordagem de deduplicação baseada em blocagem e distância de Jaccard.

Outra técnica frequentemente utilizada em deduplicação de dados é a clusterização, que agrupa registros similares em clusters e facilita a identificação de registros duplicados. Em [2], o autor discorre sobre diferentes tipos de clusterização hierárquica, que poderiam vir a ser utilizados em processos de clusterização.

Por fim, a distância de Levenshtein é uma medida comumente utilizada em deduplicação de dados para comparar a similaridade entre strings. Em [3], o autor propôs uma abordagem de deduplicação de dados baseada em blocagem e distância de Levenshtein. Essa abordagem obteve resultados satisfatórios em termos de precisão, mas apresentou dificuldades em lidar com grandes volumes de dados.

Neste artigo, propomos uma abordagem que combina blocagem, clusterização e distância de Levenshtein para identificar dados duplicados em uma base de dados de informações sobre músicas. Acreditamos que essa abordagem supera as limitações das técnicas anteriores e apresenta uma solução eficaz para o problema de reconhecimento de informações repetidas em dados de músicas.

3. A BASE DE DADOS

A base de dados cuja proposta de trabalho foi construída é denominada *MusicBrainz*. Essa base é altamente estruturada e contém informações detalhadas sobre os artistas, além de informações sobre os álbuns, títulos, datas de lançamento, gravadoras, durações e números de catálogo.

A *MusicBrainz* inclui informações sobre as relações entre os artistas, álbuns e faixas, como as colaborações entre artistas e as diferentes versões de uma mesma música. As informações na base de dados são mantidas e atualizadas por uma comunidade de voluntários, garantindo que as informações estejam atualizadas e precisas.

A base é uma escolha popular para testar técnicas de processamento de dados de música, como a deduplicação de dados, o que a torna uma base de dados valiosa para pesquisadores e desenvolvedores que pesquisam sobre essa área. Por ser uma fonte aberta e gratuita de informações de música, a *MusicBrainz* tem potencial para ser amplamente utilizada em diversas aplicações, desde plataformas de streaming até sistemas de recomendação de música personalizados.

3. FUNDAMENTAÇÃO TEÓRICA

Como comentado em seções anteriores, esse trabalho tem como objetivo apresentar uma forma de encontrar e agrupar dados duplicados, desenvolvida através de três principais pilares: blocagem, cálculo de distância entre strings com Levenshtein e clusterização.

Cada um desses passos possui uma função dentro do contexto do algoritmo desenvolvido e desempenha um papel importante na identificação e eliminação de registros duplicados; a blocagem é uma técnica que consiste em dividir o conjunto de dados em blocos ou grupos com base em uma ou mais características em comum [3]. O objetivo da blocagem é reduzir o número de comparações necessárias para identificar registros duplicados, já que os registros dentro de cada bloco são mais propensos a serem duplicados entre si.

Já a distância de Levenshtein é uma medida que calcula a diferença entre duas strings. Essa medida é baseada no número mínimo de operações (inserções, exclusões ou substituições) necessárias para transformar uma string em outra [4]. O cálculo da distância de Levenshtein é utilizado em algoritmos de deduplicação para comparar as strings e identificar registros que são semelhantes, mas não idênticos.

Por fim, a clusterização é uma técnica que agrupa registros semelhantes em clusters (grupos). Essa técnica é usada em algoritmos de deduplicação para identificar conjuntos de registros que são provavelmente duplicados. Para diferentes problemas, há diferentes técnicas de abordagem [5]. Uma das mais utilizadas na literatura para clusterização de dados é o

K-means, método de agrupamento não supervisionado em que o objetivo é dividir um conjunto de dados em K grupos (clusters) distintos. A ideia básica do algoritmo é encontrar K centróides que representam os grupos e, em seguida, associar cada objeto do conjunto de dados ao centróide mais próximo.

Juntas, essas técnicas se mostram essenciais para garantir que um conjunto de dados esteja limpo e consistente, reduzindo erros e redundâncias nos registros.

4. TRABALHO PROPOSTO

Desenvolvido em Python, a proposta apresentada para a amenização do problema de duplicatas na base apresentada anteriormente se trata de um algoritmo, cujos passos de execução serão apresentados a seguir.

Primeiro, após a importação da base de dados e instalação das bibliotecas e dependências necessárias, os dados puxados precisaram ser pré-processados. O pré-processamento de dados é uma etapa crítica na identificação de dados repetidos. Ele ajuda a garantir que os dados estejam prontos para análise, permitindo que os algoritmos de deduplicação sejam mais eficazes na identificação de registros duplicados, pois a padronização pode ajudar a garantir que os dados sejam comparáveis e compatíveis com outros dados que já existem, bem como o processo de remoção de valores nulos ou inválidos e correção de erros de digitação. Assim, cada linha da tabela teve que passar por um processo de verificação e limpeza de dados nulos e caracteres fora do conjunto alfanumérico.

Houve um total de três tabelas escolhidas como relevantes para a aplicação: a que definia o título da música em questão, outra com o nome do artista e uma com o ano de lançamento da faixa. As colunas consideradas como não relevantes para o processamento foram descartadas. Os dados descartados se referiam a identificadores de cada faixa, tanto internos quanto externos, o tamanho da faixa e seu idioma.

Funções utilitárias para o cálculo entre a distância (com utilização de uma biblioteca que trás a implementação do cálculo da distância de Levenshtein) de strings foram desenvolvidas, bem como funções que comparavam outros atributos dentro do contexto da base, como o ano de lançamento de duas faixas e o número da música (dentro do contexto de um álbum). Essas funções serviram como base para uma função mais complexa, ***distance***, cuja entrada consiste em um item (linha da tabela) e em um conjunto de outros itens cujo primeiro deve ser comparado individualmente. Serão confrontados os atributos de título da canção, álbum a qual pertence, artista, ano de lançamento e número. Sabendo que cada um desses atributos teve um peso pré-definido, como resultado dessas comparações, tem-se um único valor, resultante de uma média ponderada dos valores resultantes nas comparações individuais. A saída é um *array* contendo todas as distâncias entre esse item em foco e os outros.

```
title_w = 1.2
album_w = 1
artist_w = 1
number_track_w = .8
year_w = .5
```

Imagem 1: Pesos pré-definidos dos atributos de uma faixa

Partindo para a definição da blocagem dos dados, é necessário um critério de agrupamento dos itens. No caso do algoritmo desenvolvido e discutido neste artigo, o critério foi o valor retornado pela função *phonex* da biblioteca de mesmo nome, cuja principal função é gerar uma representação codificada de uma string, sendo essa codificação resultante da sua pronúncia. Dessa forma, todos os itens têm o *phonex* de seu título calculado e, sendo este um valor que varia de 0 a 1, é feita a ordenação da base através desses valores calculados. O resultado disso é que as músicas que possuem o título de fonética igual ou parecida estão próximas e a base é então dividida em blocos de tamanho iguais. Dependendo do tamanho escolhido para os blocos, é possível que haja um bloco que possua uma quantidade menor de itens (quando a divisão do tamanho de itens total da base pelo tamanho dos blocos não seja exata).

Por fim, há a clusterização dos dados, etapa na qual as duplicatas são, de fato, descobertas. Tendo como base o algoritmo de clusterização K-Means e seguindo seus princípios, foi preciso definir para cada bloco formado na etapa anterior um centróide. O processo de escolha foi o mais simples possível: escolheu-se o primeiro elemento de cada bloco como centróide e, a partir disso foram feitos cálculos de distância (utilizando a função ***distance***) entre esse item central e os demais inseridos no mesmo grupo.

Haviam dois possíveis resultados advindos desses cálculos: ou aquele elemento era semelhante o suficiente ao centróide atual e, portanto, pertencia ao seu cluster, ou eles eram considerados muito distantes entre si e se transformaria ele também em um cluster. Caso o cálculo resultasse no primeiro caso, o elemento era inserido no cluster cujo centróide acabara de ter a distância medida e era a vez do próximo item da lista; ou a distância do próximo item da lista seria calculada tanto com o primeiro cluster definido quanto com esse novo, definido pela última iteração.

Depois de testes e avaliações, foi definido que o valor de *threshold* que obtinha um melhor resultado de clusterização era 0.4. Isso significa que a distância entre um centróide e outro elemento deveria ser menor que esse valor para que eles pudessem ser considerados como pertencentes a um mesmo grupo, nesse contexto em particular.

5. AVALIAÇÃO

No contexto de identificação de duplicatas de uma base de dados, a precisão é uma métrica importante para avaliar a qualidade do algoritmo. A precisão é uma medida de acurácia que indica a proporção de pares de registros que foram identificados corretamente

como duplicatas em relação ao total de pares de registros classificados como duplicatas pelo algoritmo.

Em outras palavras, a precisão mede a capacidade do algoritmo em identificar corretamente as duplicatas e minimizar os falsos positivos. Um falso positivo ocorre quando o algoritmo identifica erroneamente dois registros como duplicatas, sendo que eles são registros distintos. A precisão é calculada pela seguinte fórmula:

Precisão = Verdadeiros Positivos no cluster / Quantidade de duplicatas do item na base toda

Onde:

- Verdadeiros Positivos: número de pares de registros que foram classificados corretamente como duplicatas.

Uma precisão alta indica que o algoritmo de apontamento de duplicatas é eficaz na identificação de duplicatas, enquanto uma precisão baixa indica que o algoritmo está gerando muitos falsos positivos e identificando erroneamente registros como duplicatas.

Assim sendo, foram realizados testes de precisão do algoritmo desenvolvido aplicado a diferentes contextos para que uma visão clara de sua precisão fosse criada. Em cada testes, foram acompanhadas variáveis específicas, sendo elas: o tempo de execução total (em segundos), precisão por bloco em relação ao bloco (porcentagem), precisão total em relação ao bloco (porcentagem), precisão por bloco em relação a base toda (porcentagem), precisão total em relação a base toda (porcentagem).

Em cada teste foram variados a quantidade de itens por bloco, bem como a quantidade total de itens na base de dados. Os resultados serão listados a seguir:

	Base contendo 20k itens			
	Tamanho do bloco			
Métricas	**100 itens**	**250 itens**	**500 itens**	**1000 itens**
Tempo de execução total	143.3s (2:23min)	281.7s (4:40min)	511.4s (8:30min)	959.3s (16m)
Precisão total em relação ao bloco	98.173565% (Precisões por bloco	98.042602% (Precisões por bloco)	97.869863% (Precisões por bloco)	97.534444% (Precisões por bloco)
Precisão total em relação a base toda	73.620573%	74.075636%	74.479269%	74.979498%

	Base contendo 200k itens			
	Tamanho do bloco			
Métricas	**100 itens**	**250 itens**	**500 itens**	**1000 itens**
Tempo de execução total	1572s (26:12min)	2843.7s (47min)	5294s (1h:27m:29s)	9611s (2h:40m:11s)
Precisão total em relação ao bloco	98.023461% (Precisões por bloco)	97.978519 % (Precisões por bloco)	97.750481% (Precisões por bloco)	97.616183% (Precisões por bloco)
Precisão total em relação a base toda	73.225637%	73.439803 %	73.948131%	74.215413%

Com base nos dados obtidos, é possível observar que o algoritmo apresentou uma boa precisão em todos os blocos, alcançando uma média de precisão total em relação ao bloco de aproximadamente 98%. No entanto, a precisão total em relação a base toda ficou em torno de 74%. Além disso, é possível perceber que o tempo de execução aumentou consideravelmente à medida que o tamanho dos blocos aumentou. Enquanto os blocos de 100 levaram cerca de 26 minutos, os blocos de 1000 levaram mais de 2 horas e meia. Isso sugere que o aumento do tamanho dos blocos pode levar a um tempo de execução mais longo. Em geral, pode-se concluir que o algoritmo apresentou uma boa precisão em todos os blocos testados, mas ainda há espaço para melhorias em relação à precisão total em relação à base toda.

6. CONCLUSÃO

Considerando os resultados apresentados na seção anterior, pode-se observar que a abordagem proposta para resolver o problema de encontrar duplicatas de dados na base *MusicBrainz* foi satisfatória e houve um bom balanço de *tradeoff* de eficiência e tempo, levando em conta o expressivo aumento na quantidade de dados entre os testes (um sendo feito em uma base com total de 20k itens e outro em uma de 200k, constatando um aumento de 1000%).

A precisão total obtida nas identificações por bloco e em relação à base como um todo foi elevada, variando entre 97.53% e 98.17% e 73.2% e 74.9%, respectivamente. Os tempos de execução totais também foram adequados, variando entre 143.3 segundos (para a menor base e bloco) e 2 horas e 40 minutos (para a maior base e bloco). Assim, os resultados sugerem que a abordagem proposta pode ser aplicada com sucesso em diferentes tamanhos de

bases de dados, fornecendo uma solução eficiente e precisa para o problema de identificação de duplicatas de dados na base do *MusicBrainz*.

7. REFERÊNCIAS

[1] PIMENTEL, Luan Félix; VICENTE, Igor Lemos; BIANCO, Guilherme Dal. **Redblock: Uma ferramenta para a deduplicação de grandes bases de dados em tempo real**. *In*: ESCOLA REGIONAL DE BANCO DE DADOS (ERBD), 13. , 2017, Passo Fundo. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2017 . ISSN 2595-413X.

[2] DONI, Marcelo Viana. **Análise de Cluster: Métodos Hierárquicos e de particionamento.** Trabalho de Graduação em Ciência da Computação, Universidade Presbiteriana Mackenzie, 2004. Disponível em: http://meusite.mackenzie.com.br/rogerio/tgi/2004Cluster.PDF.

[3] BÖHM, Luiz Fernando. **Elaboração de uma Estratégia de Deduplicação de Dados Utilizando Técnicas de Blocagem em um Cadastro Hospitalar de Pacientes**. Trabalho de Conclusão de Curso (Graduação), Universidade Federal do Rio Grande do Sul, 2010. Disponível em: https://www.lume.ufrgs.br/bitstream/handle/10183/26350/000757805.pdf?sequence=1.

[4] STEORTS, R.C., Ventura, S.L., Sadinle, M., Fienberg, S.E. (2014). **A Comparison of Blocking Methods for Record Linkage**. In: Domingo-Ferrer, J. (eds) Privacy in Statistical Databases. PSD 2014. Lecture Notes in Computer Science, vol 8744. Springer, Cham. https://doi.org/10.1007/978-3-319-11257-2_20.

[5] L. Yujian and L. Bo, **A Normalized Levenshtein Distance Metric**. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1091-1095, June 2007, doi: 10.1109/TPAMI.2007.1078.

[6] WEI, Dong, et al. **Tradeoffs in Scalable Data Routing for Deduplication Clusters.** In Proceedings of the International Conference on Data Engineering (pp. 100-116). IEEE.