Análise e Comparação de Algoritmos de Similaridade e Distância entre *strings* Adaptados ao Português Brasileiro

Diogo Luis Von Grafen Ruberto[1], **Rodrigo Luiz Antoniazzi**[2]

[1]Programa de Pós Graduação em Informática (PPGI) – Universidade Federal de Santa Maria (UFSM) – Santa Maria – RS – Brasil

[2]Centro de Ciências Humanas e Sociais (CCHS)
Universidade de Cruz Alta (UNICRUZ) – Cruz Alta – RS – Brasil

diogo.rubert@gmail.com, rodrigoantoniazzi@yahoo.com.br

Abstract. *The use of databases in business is fundamental to decision talking, but the information extraction in DBMS could use techniques, to make a smart search. Searches that use relational operators are limited when there are typos or when the database is inconsistent. To correct this, some systems have functions that allow you to search based on the similarity of strings, for example, searches based on phonetic algorithms such as Soundex and Metaphone, but both methods are unusual in languages other than English and therefore need of an adaptation. The algorithm calculating the distance between strings, based on calculating the Levenshtein distance, another alternative is to find similarities between two strings. In this context, it is necessary to identify which algorithm is more efficient both in performance and in the accuracy of the data returned. Furthermore, it should be considered that the efficiency varies with the database, and if hybrid methods are the best alternative.*

Resumo. *A utilização bancos de dados nas empresas é fundamental para tomada de decisões, porém a recuperação de informações nos SGBD poderia utilizar técnicas para tornar as buscas mais inteligentes. As buscas que utilizam operadores relacionais são limitadas quando ocorrem erros de digitação ou quando a base de dados está inconsistente. Para suprir esta deficiência, alguns sistemas possuem funções que permitem fazer buscas baseadas na similaridade das strings, por exemplo, as buscas baseadas em algoritmos fonéticos como o Soundex e o Metaphone, porém ambos os métodos não são usuais em idiomas diferentes do inglês e precisam, portanto, de uma adaptação. O algoritmo do cálculo da distância entre strings, baseado no cálculo da distância de Levenshtein, é outra alternativa para encontrar similaridades entre duas cadeias de caracteres. Neste contexto, é necessário identificar qual algoritmo é o mais eficiente, tanto na performance quanto na precisão dos dados retornados. Além disso, deve ser analisado se a eficiência varia de acordo com a base de dados, e se os métodos híbridos são a melhor alternativa.*

1. Introdução

O armazenamento e relacionamento de informações em um Sistema Gerenciador de Banco de Dados (SGBD) são essenciais para uma organização, uma vez que grandes

bancos de dados podem conter informações importantes, as quais permitem as empresas tomar decisões e gerar conhecimento [Sudarshan et al. 2006].

A recuperação das informações é tão importante quanto o seu armazenamento e, por isso, deve ocorrer de maneira simples e precisa [Frantz 2009]. No momento em que um grande volume de dados é armazenado, a Lógica *Fuzzy* auxilia no reconhecimento de padrões para que estes dados se tornem informações úteis aos usuários [Chen et al. 2016]. Entretanto, a recuperação de informações pode ser trabalhosa por meio dos métodos tradicionais de comparação de *strings* dos SGBD atuais.

Apesar da eficiência do operador *like*, do operador de igualdade e demais operadores lógicos em consultas SQL, eles são limitados em bases de dados onde ocorreram erros de digitação ou em buscas fonéticas. Mesmo que o banco de dados esteja consistente e as informações cadastradas corretamente, a falha humana pode ocorrer no momento em que o usuário digita a informação que deseja buscar [Frantz 2009].

O erro humano não deve impedir que um sistema funcione corretamente e, por este motivo, as técnicas que serão abordadas têm inúmeras aplicabilidades. Por exemplo, em um hospital o paciente não poder deixar de ser atendido porque o usuário digitou "Amiuton" ao invés de "Hamilton". Em um sistema de vendas *on-line* o produto não pode deixar de ser enviado para a cidade correta porque o comprador digitou "Pajuçara" quando o correto seria "Pejuçara". Ao mesmo tempo, os métodos podem ser utilizados em corretores ortográficos [Piltcher et al. 2005] ou buscadores que sugerem a palavra correta ao usuário que cometer um erro.

Algumas linguagens de programação e Sistemas Gerenciadores de Banco de Dados (SGBD) disponibilizam funções nativas ou extensões que possibilitam a busca de informações com base na similaridade dos dados, como por exemplo, o cálculo da distância *Levenshtein* e os algoritmos fonéticos *Soundex* e *Metaphone*. Entretanto, conforme [PostgreSQL 2016] as funções fonéticas são pouco usuais em idiomas diferentes do inglês e, portanto, existe a necessidade de adaptar estes algoritmos para outros idiomas.

Além de adaptar as funções fonéticas para o português brasileiro, no presente estudo são aplicados métodos a fim de facilitar a recuperação de dados em grandes bases, mesmo que estes estejam inconsistentes, que tenham ocorrido erros ao informar o dado que se deseja encontrar ou até mesmo quando não se sabe exatamente a informação que deseja localizar. Após, é realizada uma comparação dos resultados obtidos com os métodos aplicados com objetivo de identificar qual deles foi mais eficiente na recuperação de informações. Também será comparado o desempenho para execução das funções, com a intenção de identificar qual o melhor método.

1.1. Organização do trabalho

Este trabalho está organizado em cinco sessões. A Sessão 2 trata da revisão bibliográfica sob a qual este artigo foi fundamentado. A Sessão 3, apresenta um estudo detalhado sobre algoritmos de similaridade e distância entre *strings*, em especial os algoritmos *Soundex*, *Metaphone* e *Levenshtein*.

A Sessão 4 trata da implementação das funções adaptada ao português brasileiro e do ambiente de testes que foi utilizado. Por fim, a Sessão 5 apresenta as considerações finais.

2. Revisão Bibliográfica

A seguir são apresentadas algumas pesquisas sobre similaridade e distância entre *strings*, fonética da língua portuguesa e problemas em consultas a bases de dados, que serviram como base para a elaboração desta pesquisa.

[Frantz 2009] descreveu as dificuldades encontradas na recuperação de informações, tais como: a redundância, inconsistência e ambiguidade dos dados, e destacou como a qualidade de uma aplicação pode ser comprometida com estes problemas. Analisou que as soluções disponíveis para solução destas questões são limitadas quando tratam de textos escritos em línguas diferentes do inglês. Fez um estudo dos fonemas da língua portuguesa e de algoritmos fonéticos, onde adaptou os métodos para o português brasileiro e criou uma função para recuperação de informações em bancos de dados. Por fim, elaborou um estudo comparativo entre o protótipo desenvolvido e uma ferramenta existente, e por meio de um questionário avaliou a eficácia do algoritmo. O estudo dos fonemas e a adaptação dos algoritmos fonéticos para o português brasileiro são de suma importância para este estudo, pois apresentam os resultados estatísticos e as dificuldades encontradas na adaptação destes métodos.

[Jardini 2012] propôs um ambiente *data cleaning* a fim de melhorar a qualidade dos dados armazenados em bancos de dados inconsistentes. No trabalho são discutidos os problemas de inconsistência e duplicidade de dados, além de expor diversas técnicas para identificação de similaridades, baseadas em caracteres, *token* e fonética. Entre as técnicas discutidas, registrou o uso dos algoritmos de *Levenshtein*, *Soundex* e *Metaphone*, e explicou o conceito de cada um. Para detecção de duplicidades e inconsistências, usou o algoritmo de *Levenshtein* e para permitir que a ferramenta identificasse duplicidade, independente do idioma, implementou uma detecção fonética multi-idioma. Também aplicou testes do ambiente e comprovou que a ferramenta cobriu aproximadamente 90% das inconsistências. As técnicas que serão analisadas na presente pesquisa foram conceituadas e testadas no trabalho correlato supracitado, por isso a sua importância.

[Borges 2008] apresentou um mecanismo de deduplicação de metadados e rastreio da proveniência. O sistema foi aplicado em bibliotecas digitais onde ocorrem problemas como variações de grafia e omissão de palavras. Para identificação da similaridade entre os títulos dos objetos digitais foram aplicadas técnicas baseadas no algoritmo de *Levenshtein*. Explicou que a técnica foi escolhida pois preserva a ordem em que as palavras aparecem em uma *string*, porém pode ter alguns problemas com caracteres acentuados. Foram definidas métricas de avaliação e aplicados experimentos para medir a precisão de cada algoritmo aplicado. A aplicação do algoritmo de *Levenshtein* e as métricas utilizadas devem auxiliar nas conclusões sobre a eficiência do método.

3. Algoritmos Fonéticos e de Similaridade

Nesta sessão serão abordados alguns algoritmos que fazem buscas inteligentes. Apesar das linguagens de programação e SGBD disponibilizarem funções fonéticas que retornam representações das palavras em forma de códigos, não existem funções específicas para língua portuguesa.

Os algoritmos fonéticos escolhidos para ser estudados e adaptados, foram o *Soundex* e o *Metaphone*, pois os mesmos são a base para os diversos outros existentes

[Croft et al. 2016]. O algoritmo de *Levenshtein* foi escolhido pois, mesmo que o usuário saiba escrever corretamente a palavra ou nome que deseja buscar, poderia ocorrer um simples erro de digitação que, mesmo que mude sua pronúncia, é muito próximo da palavra correta. O cálculo da distancia entre *strings* deve resolver este problema.

A intenção destes métodos é ir além da busca exata, aquela que utiliza operadores relacionais [Snae 2007]. Quando utiliza-se o operador "=" (igual), é necessário transcrever a *string* exatamente como ela está armazenada. Mesmo que se utilizasse a função *"upper"* para que a consulta não seja *case sensitive*, a necessidade de digitar a sequência de caracteres exatamente como ela está armazenada no banco de dados é uma premissa básica. Mesmo que fosse utilizado o operador *like* em um comando SQL para procurar um determinado padrão, chega-se a uma proximidade matemática, exata e previsível. Para encontrar o nome "Vilson" quando não se sabe se o correto é "Vilson", "Wilson", "Vilsom" ou "Wilsom" pode ser bastante trabalhoso.

3.1. *Soundex*

O código *Soundex* foi criado por Robert C. Russell e Margaret K. Odell em 1918. Inicialmente foi usado no censo americano como uma forma de indexar os nomes das pessoas. A ideia de Russell foi ordenar o nome das pessoas não por ordem alfabética, mas sim pela forma como era pronunciado. O código *Soundex* de uma palavra é representado pela letra inicial mais um código de três números que é obtido a partir de uma tabela [Binstock and Rex 1995].

A tabela de códigos *Soundex* criada por Russell foi baseada na classificação dos fonemas da língua inglesa [Reyes-Barragán et al. 2009]. Para adaptar o *Soundex* para o português brasileiro, a proposta do presente trabalho é mudar o valor da tabela de códigos baseado na classificação fonética língua portuguesa, mais precisamente nos pontos de articulação utilizados para pronunciar as consoantes, já que as vogais são ignoradas neste método. A tabela 1 apresenta o proposta para adaptação.

Tabela 1. Códigos *Soundex* para Português Brasileiro

Letra(s)	Valor	Pontos de Articulação
A, E, I, O, U, H, W, Y	0	-
P, B, M	1	Bilabiais
F, V	2	Labiodentais
T, D, N	3	Linguodentais
L, R	4	Línguo-Alveolares
S, Z	5	Línguo-Alveolares Convexas
J, DI, GI, TI, CH, LH, NH	6	Línguo-Palatais
K, C, G, Q	7	Velares
X	8	-

Como no português existem dígrafos, ou seja, duas consoantes que juntas formam um só fonema como observa-se nas palavras que possuem as consoantes "CH", "LH" e "NH", o algoritmo deverá ser adaptado para verificar estes encontros consonantais e tratar de forma adequada. As letras "W" e "Y" foram mantidas como letras a serem descartadas, pois são utilizadas apenas para representar palavras estrangeiras que ainda

não foram aportuguesas. Também por produzirem os sons de "U" e "I", respectivamente, as quais também são descartadas no *Soundex* assim como as demais vogais.

Entretanto, a letra "X", pode representar quatro fonemas na língua portuguesa [Chbane 1994]. É o caso das palavras exame, máximo, complexo e xícara. Percebe-se que em cada uma das palavras o "X" é pronunciado de forma diferente. Por esse motivo, ele será tratado como um código independente.

3.2. *Metaphone*

Assim como o *Soundex*, o algoritmo *Metaphone* também é considerado um algoritmo fonético. Ele foi escrito por Lawrence Philips em 1990 com o objetivo de suprir as deficiências do *Soundex*. Além desse método, o autor também desenvolveu os métodos *Double Metaphone* e *Metaphone* 3, que são melhoramentos do algoritmo original [Lisbach and Meyer 2013].

Como uma mesma letra pode representar mais de um som, a ideia do *Metaphone* é identificar a posição onde a letra está inserida, para assim definir a sua melhor representação. Outra definição deste método, é que aplicam-se regras para transformar um som em outro. Diferente do *Soundex*, não são consideradas apenas consoantes para definir uma representação fonética. As vogais também são importantes para identificar o som que um conjunto de caracteres pode representar [Binstock and Rex 1995]. Assim como o *Soundex*, o algoritmo *Metaphone* usa regras para fazer as transformações. Obviamente, estas regras foram baseadas na língua inglesa e não são usuais no português. Com estas regras, as palavras são transcritas para uma representação fonética, de maneira que palavras que soam de maneira semelhante serão representadas da mesma forma.

Baseado nas regras deste algoritmo fonético, [Jordão and Rosa 2012] escreveram um artigo sobre a importância da fonética na busca e correção de informações textuais. Neste artigo, apresentaram uma proposta de adaptação para o português brasileiro, denominado *Metaphone-pt_BR*. Os autores explicam que obtiveram resultados satisfatórios com as novas regras, entretanto, mesmo com a adaptação, o algoritmo é eficiente com palavras encontradas no dicionário, mas em nomes e sobrenome em que a pronuncia depende do local de origem, deveriam ser usadas regras específicas.

3.3. *Levenshtein*

O conceito da distância de *Levenshtein* foi escrito em 1965 pelo matemático Vladimir I. Levenshtein e baseado na distância de *Hamming*, porém, a diferença é que pode ser usado para comparar palavras com tamanhos diferentes. O princípio de *Levenshtein* é definir a distância entre duas palavras com base no número de operações necessárias para torná-las iguais. Cada operação tem um determinado custo que é acumulado de acordo com as edições realizadas: inserção, exclusão ou substituição [Lisbach and Meyer 2013].

[Wagner and Fischer 1974] observaram que para calcular a distância de *Levenshtein* seria necessário no mínimo $m * n^2$ comparações e, então, publicaram um algoritmo capaz de reduzir esta complexidade para $m * n$, conhecido como *edit-distance*. Apesar da sua utilidade, este algoritmo pode ser lento para comparar *strings* muito longas, pois a matriz que precisa ser criada é diretamente proporcional ao tamanho de cada *string*.

4. Ambiente Experimental

Após conceituar os algoritmos de *Levenshtein*, *Soundex* e *Metaphone*, e definir as adaptações necessárias, foram implementadas em *PL/pgSQL* as funções abaixo. Esta linguagem foi escolhida por tratar-se de uma extensão do padrão SQL que permite a implementação de funções robustas no *PostgreSQL*. Este SGBD, por sua vez, foi escolhido por ser *opensource*, robusto e com uma boa documentação.

- *br_levenshtein*: função para calcular a distância de *Levenshtein* baseada no algoritmo de Wagner e Fisher (1974);
- *br_soundex*: função para calcular o código *Soundex* adaptado para o português brasileiro, conforme proposto neste trabalho;
- *br_metaphone*: função para calcular o código *Metaphone* adaptado para o português brasileiro, conforme proposto por [Jordão and Rosa 2012].

4.1. Métricas Utilizadas

Conforme [Sudarshan et al. 2006], para medir a eficácia de funções que recuperam informações, podem ser aplicadas medidas de precisão e de revocação. Assim, os algoritmos foram comparados entre si, e efetuadas estas medidas a partir da geração de dados estatísticos e gráficos. Também foi analisado se a hipótese de que soluções híbridas tem um resultado melhor, é verdadeira.

Os testes realizados a seguir utilizaram as seguintes métricas para avaliação dos resultados.

- Precisão: quando executada uma consulta, deve medir a taxa de acerto da função, isto é, quantos registros foram retornados corretamente em relação a todos os registros retornados;
- Revocação: quando executada uma consulta, deve medir a taxa de registros relevantes retornados, ou seja, quantos registros foram retornados corretamente em relação ao total de registros que a função deveria de fato retornar;
- Medida F Balanceada: é a média harmônica ponderada da precisão e da revocação. Será utilizada para medir a relação entre as duas métricas utilizadas;

4.2. Testes e Resultados

Os testes das funções foram executados em bases que simulam várias situações de buscas. Um banco de dados com 5.500 registros com nomes de cidades. Outro, com 310.000 registros com palavras da língua portuguesa. E, por fim, uma base com 10.000 registros com nomes de pessoas e empresas.

A análise e comparação foi efetuada a partir da coleta de algumas amostras de cada base de dados, as quais determinaram quais dados deveriam ser retornados. A partir disso, foi calculado o percentual médio de todas as consultas. Na base com 5.500 registros foram realizadas 25 consultas para cada um dos tópicos. Na base com 10.000 registros, 50 consultas. Já na base com 310.000 registros, foram realizadas 100 consultas.

4.2.1. Base de dados com nomes de cidades

O primeiro teste utilizou uma base de dados com o nome de cidades, cujo objetivo foi analisar a eficiência das funções em situações onde os dados não sofreram erros de digitação,

e que os operadores lógicos também encontrariam. Mas também procurou analisar a situação em que o dado inicial foi digitado incorretamente, porém o erro não alterou a forma como a palavra é pronunciada. Para exemplificar, poderia-se tentar buscar a cidade de "Chiapeta" onde o correto é "Chiapetta". Da mesma forma, fez-se a verificação que utiliza dados inconsistentes, isto é, caracteres faltando e caracteres que alteram, em partes, forma como uma palavra é pronunciada.

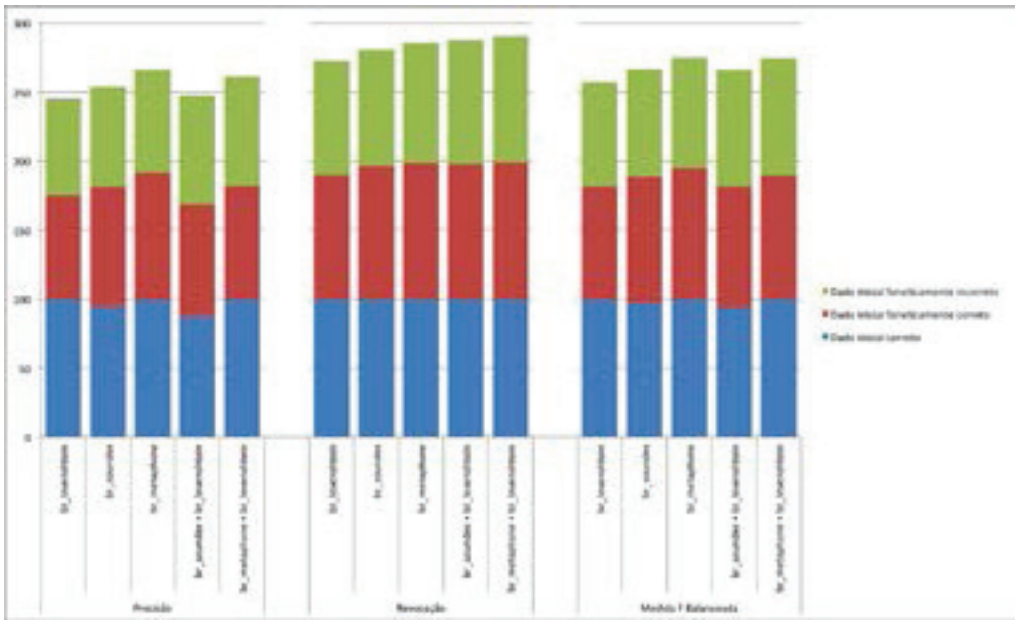

Figura 1. Gráfico da base de dados com nomes de cidades

Nesta base, foi observado que nenhuma função deixou de mostrar uma informação relevante quando o dado inicial estava correto, pois atingiram 100% de revocação. Por outro lado, os métodos *br_soudex*, *br_metaphone* e o híbrido *br_soundex* + *br_levenshtein* retornaram alguns dados a mais, o que reduziu a precisão. De qualquer forma, o resultado da média harmônica ponderada foi satisfatório, e ficou acima dos 90%.

Pode-se observar na Figura 1 que as funções fonéticas superam o método da distância entre strings quando dado inicial sofre alterações que não alteram a sua pronúncia. Observa-se ainda que a função *br_metaphone* obteve a melhor média ao relacionar precisão e revocação.

4.2.2. Base de dados com nomes de pessoas e empresas

Este teste utilizou uma base de dados com nomes de pessoas e empresas, onde tem-se praticamente o dobro de registros da base utilizada no teste anterior. O objetivo foi verificar a eficácia e performance em uma base com maior volume de registros e que nomes estrangeiros podem não representar necessariamente um fonema da língua portuguesa. A intenção do teste é simular situações onde o usuário não sabe escrever corretamente o sobrenome de uma pessoa, por exemplo. Da mesma forma que o teste anterior, este experimento fez consultas para simular a inconsistência em bases que armazenam nomes. São situações em que um nome está armazenado incorretamente ou quando houve uma falha na digitação. Os resultados são demonstrados na Figura 2.

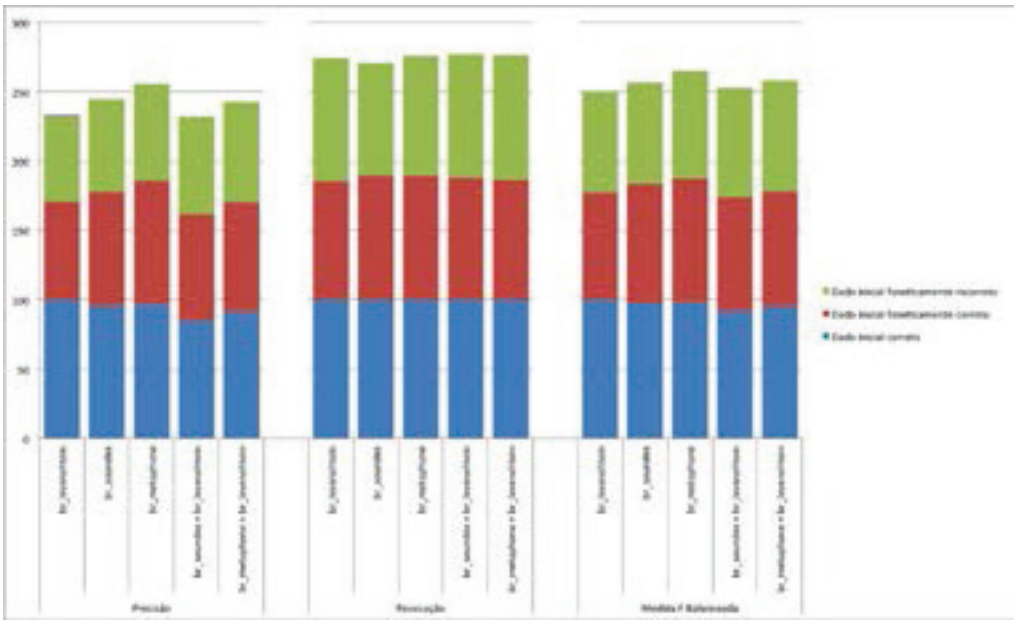

Figura 2. Gráfico da base de dados com nomes de pessoas e empresas

No caso dos nomes de empresas e pessoas, as funções fonéticas ainda são superiores, porém verificou-se uma pequena queda nos percentuais de precisão e revocação, consequentemente, na medida F balanceada. Isto explica-se pois alguns sobrenomes estrangeiros não são pronunciados exatamente da forma que foram escritos. Cabe lembrar que os métodos foram adaptados apenas para o português brasileiro.

4.2.3. Base de dados com palavras do dicionário

A terceira e última análise, utiliza uma base com 310.000 registros que, na verdade, trata-se de um dicionário da língua portuguesa. O objetivo do teste foi determinar efetividade das funções com grandes volumes de dados em um ambiente que contém a maioria das palavras utilizadas habitualmente na língua portuguesa. Este ambiente de testes representa a situação de um corretor ortográfico, que precisa analisar o dado digitado e retornar sugestões ao usuário.

Ao avaliar os métodos neste experimento, verificou-se que a média para a função *br_metaphone* mantem-se constante e é superior as demais, e que as soluções híbridas tiveram bons resultados como pode ser verificado na Figura 3. Apesar da precisão ser pequena, pois retornou uma série de informações desnecessárias, a revocação teve um bom percentual, afinal os dados que precisavam ser mostrados foram apresentados corretamente.

No quesito revocação, pode-se afirmar que todas as funções são totalmente eficientes, pois nenhum registro deixou de ser listado. Em relação à precisão, apenas a função *br_levenshtein* foi totalmente precisa. As demais retornaram alguns registros a mais, porém chegaram muito próximo dos 100%.

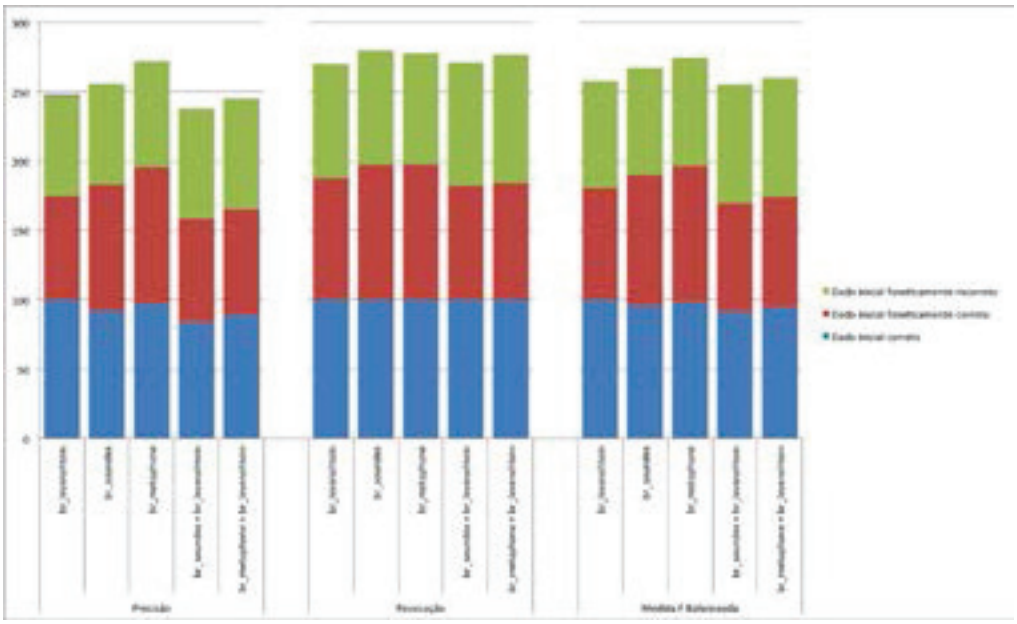

Figura 3. Gráfico da base de dados com palavras do dicionário

5. Considerações Finais

Neste estudo procurou-se comprovar que as adaptações dos métodos fonéticos são possíveis para a língua portuguesa e responder qual o melhor método para recuperação de informações. A compreensão da fonética, fonologia e fonemas da língua portuguesa foi fundamental para a adaptação dos métodos *Soundex* e *Metaphone*, pois ainda existem poucos estudos para o português brasileiro.

Quando o dado inicial está correto, todos os métodos atingiram 100% no quesito revocação, isto é, nenhum registro que deveria ser listado deixou de ser apresentado. Entretanto, o objetivo deste trabalho é ir além das buscas exatas utilizando operadores lógicos. Por este motivo, os testes com o dado inicial incorreto devem ter maior relevância na análise dos resultados.

No quesito precisão, as funções fonéticas demonstraram ser bastante eficientes quando ocorrem erros de digitação. Destaca-se que a função *br_metaphone* se sobressai em relação as demais. Os dados retornados por esta função foram bastante precisos e chegaram à 99,1% de precisão com o dado inicial foneticamente correto. Outro fator observado, é que o uso da função *br_metaphone* em uma solução híbrida traz resultados superiores a mesma solução híbrida utilizando *br_soundex*. Já a função *br_levenshtein*, foi precisa apenas quando o dado inicial estava correto. Nos demais casos, foi pouco precisa com médias entre 61,4% e 75,0%.

As funções de similaridade mostraram ser uma alternativa interessante para suprir as limitações dos operadores lógicos. O uso destas técnicas podem ter aplicabilidades em inúmeros tipos de sistema, pois possibilitam uma busca alternativa ao usuário e permite a sugestão de informações, situação esta que vimos em buscadores modernos e corretores ortográficos. Entretanto, os métodos estudados são eficientes apenas com palavras do dicionário, e perdem bastante eficiência na busca de nomes, sobrenomes e palavras estrangeiras.

Como sugestão de trabalhos futuros, pode-se comparar a utilização de outras técnicas de detecção de similaridade entre *strings*, sejam elas baseadas em caracteres, *token* ou fonética. A adaptação destas funções para ambientes multi-idioma também poderia ser pesquisada. Estes métodos também podem ser usados para criação ou melhoria de sistemas de reconhecimento da fala.

Referências

Binstock, A. and Rex, J. (1995). *Practical algorithms for programmers*. Addison-Wesley Longman Publishing Co., Inc.

Borges, E. N. (2008). Md-prom: um mecanismo de deduplicação de metadados e rastreio da proveniência. Master's thesis, Universidade Federal do Rio Grande do Sul.

Chbane, D. T. (1994). *Desenvolvimento de sistema para conversão de textos em fonemas no idioma português*. PhD thesis, Universidade de São Paulo.

Chen, S.-M., Cheng, S.-H., and Lan, T.-C. (2016). A novel similarity measure between intuitionistic fuzzy sets based on the centroid points of transformed fuzzy numbers with applications to pattern recognition. *Information Sciences*, 343:15–40.

Croft, D., Brown, S., and Coupland, S. (2016). An effective named entity similarity metric for comparing data from multiple sources with varying syntax. *Digital Scholarship in the Humanities*, page fqw035.

Frantz, R. R. R. (2009). Recuperação de informações por similaridade de fonemas adaptada à lingua portuguesa. *Centro Universitário Ritter dos Reis*.

Jardini, T. (2012). Ambiente data cleaning: suporte extensível, semântico e automático para análise e transformação de dados. Master's thesis, Universidade Estadual Paulista (UNESP).

Jordão, C. C. and Rosa, J. L. G. (2012). Metaphone-pt_br: the phonetic importance on search and correction of textual information. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 297–305. Springer.

Lisbach, B. and Meyer, V. (2013). *Linguistic identity matching*. Springer.

Piltcher, G., Borges, T., Loh, S., Lichtnow, D., and Simoes, G. (2005). Correçao de palavras em chats: Avaliaçao de bases para dicionários de referência. *XXV Congresso da Sociedade Brasileira de Computação, São Leopoldo: UNISINOS*.

PostgreSQL (2016). Postgresql 9.6.1 documentation. *Diponível em https://www.postgresql.org/docs/9.6/static/index.html, Acesso em: 09 nov. 2016*.

Reyes-Barragán, M. A., Pineda, L. V., and Montes-y Gómez, M. (2009). Inaoe at qast 2009: Evaluating the usefulness of a phonetic codification of transcriptions. In *CLEF (Working Notes)*.

Snae, C. (2007). A comparison and analysis of name matching algorithms. *International Journal of Applied Science. Engineering and Technology*, 4(1):252–257.

Sudarshan, S., Silberschatz, A., and Korth, F. H. (2006). Sistemas de banco de dados. 5ª. edição.

Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.