UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"

Pós-Graduação em Ciência da Computação

Tiago Luís de Andrade

Ambiente independente de idioma para suporte a identificação de tuplas duplicadas por meio da similaridade fonética e numérica: otimização de algoritmo baseado em multithreading

UNESP
2011

Tiago Luís de Andrade

Ambiente independente de idioma para suporte a identificação de tuplas duplicadas por meio da similaridade fonética e numérica: otimização de algoritmo baseado em multithreading

Orientador: Prof. Dr. Carlos Roberto Valêncio
Co-Orientador: Prof. Dr. Maurizio Babini

Dissertação apresentada para obtenção do título de Mestre em Ciência da Computação, Área de Concentração em Sistemas de Computação, junto ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

UNESP
2011

Tiago Luís de Andrade

Ambiente independente de idioma para suporte a identificação de tuplas duplicadas por meio da similaridade fonética e numérica: otimização de algoritmo baseado em multithreading

Dissertação apresentada para obtenção do título de Mestre em Ciência da Computação, Área de Concentração em Sistemas de Computação, junto ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

BANCA EXAMINADORA

Prof. Dr. Carlos Roberto Valêncio
IBILCE/UNESP – São José do Rio Preto
Orientador

Prof. Dr. Pedro Luiz Pizzigatti Corrêa
POLI/USP – São Paulo

Prof. Dr. José Márcio Machado
UNESP – São José do Rio Preto

São José do Rio Preto, 05 de agosto de 2011.

AGRADECIMENTOS

Primeiramente, a Deus e ao Nosso Senhor Jesus Cristo pela saúde e a força dada para encarar todos os desafios que a minha frente apareceram.

Em especial, aos meus pais, Célio e Maria, e irmãos pelo carinho, amor, e compreensão em todas as horas difíceis por mim enfrentadas nesses dois anos de estudo.

A Avó Geni, Tia esmeralda, Tio Toninho, Junior e Carol que, gentilmente, me hospedaram em sua casa, em São José do Rio Preto, sempre que precisei.

A Avó Rodete, Avô Martinho, Tio Índio, Tia Rita, Tia Jô, Tia Sandra, Tio Telei e a Edna que se fizeram presentes nos meus momentos de angústia e dificuldade.

Ao professor Valêncio, pela oportunidade de qualificação por meio deste mestrado. Suas orientações fizeram - me uma pessoa mais sábia e entendedora das circunstâncias da vida, não só no meio científico, mas, também, como ser humano.

Aos Profs. Drs. José Márcio Machado, Rogéria Cristina Gratão de Souza e Pedro Luiz Pizzigatti Corrêa, por terem aceitado o convite de avaliar este trabalho em suas diversas etapas e pelas contribuições dadas.

Aos professores Maurízio Babini e Fernando Ferrari pela participação na elaboração da pesquisa deste trabalho.

A Universidade do Estado de Mato Grosso – UNEMAT, na figura de seu reitor, pela dispensa do trabalho para este período de qualificação, dada a importância de um afastamento integral com remuneração para os estudos.

A Tássia Carvalho pelo apoio durante o mestrado.

Aos amigos de Cáceres, Nivaldi Calonego, Marcos Paulo, Roberto Tikao, Camillo Araújo, Sérgio Carvalho e Vitérico Maluf, que me incentivaram a fazer o mestrado e não mediram esforços para a concessão do meu afastamento da UNEMAT.

A Andréia Bordon que me ajudou com as correções dos textos escritos e pela companhia nesses últimos meses do meu mestrado.

Aos amigos de Fernandópolis e Meridiano, William, Bruno, Vinicius e Fernanda, pelos momentos de descontração que aliviavam as tensões.

Aos funcionários, professores e demais amigos de Rio Preto, Alexandre, Jonathan, Roberta, Juliano e membros do Grupo de Banco de Dados, presentes durante todo o tempo da minha pesquisa. Em especial, ao amigo Danilo, que nesses trinta meses, auxiliou-me e incentivou-me a buscar os melhores rumos no meu trabalho, fazendo-me continuar firme na caminhada dos estudos, mesmo nos momentos mais difíceis da minha vida.

Enfim, a todos aqueles que, de alguma forma, contribuíram para esta conquista, encarada, por mim, como o maior desafio da minha vida.

Dedico este trabalho ao meu pai Célio, minha mãe Maria, aos Irmãos Célio e Kiara, e as sobrinhas Maria Eduarda e Sophia.

“Nada na vida está realmente em nossas mãos... mas tudo está diante das nossas possibilidades!”

Autor Desconhecido

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

KDD	*Knowledge Discovery in Databases*
J2SE	*Java 2 Standard Edition*
SGBD	Sistema Gerenciador de Banco de Dados
IPA	*International Phonetic Alphabetic*
IP	*Internet Protocol*
SQL	*Structured Query Language*
PSI	*Phonetic Similarity Identification*
DIBP	*Data Identification Based on Phonetic*
DDR2	*Double Data Rate 2*
GB	*Gigabyte*
MB	*Megabyte*
SIVAT	Sistema de Vigilância de Acidentes de Trabalho

RESUMO

Com o objetivo de garantir maior confiabilidade e consistência dos dados armazenados em banco de dados, a etapa de limpeza de dados está situada no início do processo de Descoberta de Conhecimento em Base de Dados (*Knowledge Discovery in Database* - KDD). Essa etapa tem relevância significativa, pois elimina problemas que refletem fortemente na confiabilidade do conhecimento extraído, como valores ausentes, valores nulos, tuplas duplicadas e valores fora do domínio. Trata-se de uma etapa importante que visa a correção e o ajuste dos dados para as etapas posteriores. Dentro dessa perspectiva, são apresentadas técnicas que buscam solucionar os diversos problemas mencionados. Diante disso, este trabalho tem como metodologia a caracterização da detecção de tuplas duplicadas em banco de dados, apresentação dos principais algoritmos baseados em métricas de distância, algumas ferramentas destinadas para tal atividade e o desenvolvimento de um algoritmo para identificação de registros duplicados baseado em similaridade fonética e numérica independente de idioma, desenvolvido por meio da funcionalidade *multithreading* para melhorar o desempenho em relação ao tempo de execução do algoritmo. Os testes realizados demonstram que o algoritmo proposto obteve melhores resultados na identificação de registros duplicados em relação aos algoritmos fonéticos existentes, fato este que garante uma melhor limpeza da base de dados.

Palavras-chave: limpeza de dados, tuplas duplicadas, fonética, PSI, *multithreading*.

ABSTRACT

In order to ensure greater reliability and consistency of data stored in the database, the data cleaning stage is set early in the process of Knowledge Discovery in Database - KDD. This step has significant importance because it eliminates problems that strongly reflect the reliability of the knowledge extracted as missing values, null values, duplicate tuples and values outside the domain. It is an important step aimed at correction and adjustment for the subsequent stages. Within this perspective, techniques are presented that seek to address the various problems mentioned. Therefore, this work is the characterization method of detecting duplicate tuples in the database, presenting the main algorithms based on distance metrics, some tools designed for such activity and the development of an algorithm to identify duplicate records based on phonetic similarity numeric and language-independent, developed by multithreading functionality to improve performance over the runtime of the algorithm. Tests show that the proposed algorithm achieved better results in identifying duplicate records regarding phonetic algorithms exist, a fact that ensures better cleaning of the database.

Keywords: data cleaning, duplicate tuples, phonetics, PSI, *multithreading*.

Capítulo 1
Introdução

1.1. Considerações Iniciais

Diante das evoluções nos sistemas de banco de dados, ocorreram também avanços significativos nas próprias arquiteturas dos computadores, em que rapidamente os antigos computadores passaram a ceder espaço aos novos, com melhoramento das tecnologias de software e hardware. Conseqüentemente, o volume de dados que estes computadores passaram a lidar ficou maior, fator este que exigiu uma maior capacidade de processamento e de armazenamento desses dados. No entanto, tornou-se normal deparar-se com grandes volumes de dados oriundos de aplicações distintas, como a médica, engenharia, financeira, comercial, economia, entre outras (LUO, 2008). Surgiu então a necessidade de lidar não somente com o método de armazenamento e processamento de dados mais arrojados, mas, também, com algo que pudesse extrair dados de forma a beneficiar as aplicações envolvidas, com a finalidade de buscar muitas vezes conhecimentos úteis ou ainda desconhecidos, melhorar o processo de projeção dos dados e contribuir no suporte adequado às tomadas de decisões por meio desses dados armazenados. No decorrer dos trabalhos e análises mais detalhadas dos bancos de dados, verificou-se que uma significativa porção dos dados armazenados apresentou problemas de inconsistências ou relacionados ao esquema (RAHM; DO, 2000).

Para satisfazer a necessidade de realização da limpeza em banco de dados, diversas técnicas foram desenvolvidas, cada uma voltada a solução de determinado

problema. Na análise efetuada, constatou-se que as técnicas possuem um modelo de funcionamento específico com seu grau de complexidade e execução pertinente, ao fazer uso de um ou mais algoritmos com a finalidade de atingir o objetivo final que é a limpeza de dados. Sendo assim, diversos algoritmos passaram a ser utilizados para esta finalidade.

Mediante os estudos de vários algoritmos, observou-se a pertinência de desenvolvimento de um algoritmo que seja capaz de trabalhar baseado na similaridade fonética e numérica dos registros da base de dados e que opere com melhor desempenho em relação ao tempo de execução. Diante disso, surgiu a perspectiva de desenvolvimento de um algoritmo baseado na funcionalidade *multithreading*, que trabalhe com transcrições fonéticas em diferentes idiomas conforme o critério do usuário no ambiente desenvolvido, além de considerar os números como parte integrante dos registros do banco de dados, com a finalidade de identificar os registros duplicados em bases de dados. A importância do uso desta funcionalidade é dada pelo fato de que a tarefa de transcrever foneticamente cada letra do alfabeto é árdua e demorada e esta tecnologia pode melhorar o tempo de processamento. Como forma de medir o desempenho, são apresentados cálculos do tempo de processamento em segundos/milissegundos e em *bytes* por segundos e, por último, a exibição dos resultados obtidos.

1.2. Motivação e escopo

Com o grande volume de dados armazenados nas bases de dados das diversas aplicações existentes, surgiu a necessidade de tê-las íntegras e confiáveis a fim de garantir que o processo de extração de informações úteis possa auxiliar as futuras projeções e contribuir para tomadas de decisões. Com isso, surgiu no final da década de 80 o conceito denominado de Descoberta de Conhecimento em Bases de Dados (*Knowledge Discovery in Databases - KDD*), que tem como principal objetivo a transformação dos dados em conhecimento, um papel importante de superação do processamento humano feito manualmente (LUO, 2008).

Dentre as etapas integrantes do processo de *KDD*, cita-se a Limpeza de Dados, responsável pela eliminação de dados inadequados que possam comprometer a tomada de decisão e elaboração de estratégias de negócio.

Uma análise realizada por pesquisadores nos bancos de dados constatou que grande parte desses dados armazenados encontravam-se com algum problema: ou relacionado à inconsistência de dados ou ao esquema na confecção do próprio banco

(RAHM; DO, 2000). Tais distorções contribuíam para o armazenamento de dados errados, surgindo a necessidade da realização de limpeza de dados como forma de garantir a integridade e consistência dos dados para futura extração de conhecimento e projeção, aumentando a confiabilidade dos resultados.

Para a realização da limpeza, diversos algoritmos foram adotados de forma a auxiliar esse processo. Alguns destes algoritmos contribuíram significativamente para a identificação de problemas nos dados, outros nem tanto, visto que muitos deles não foram desenvolvidos especificamente para serem utilizados na tarefa de limpeza buscando a solução dos problemas.

Os algoritmos utilizados pelas diversas técnicas de limpeza de dados, em especial para a identificação de registros duplicados, produzem resultados satisfatórios, mas, ainda sim, há a necessidade de melhora. Se aplicados a uma base de dados com um grande número de registros, principalmente alfanuméricos, os resultados podem ser duvidosos, confusos ou inaproveitados, visto que a maioria dos algoritmos despreza os dados numéricos no momento da execução. Isso pode comprometer a confiabilidade da limpeza, pois dados integrantes do conteúdo das tuplas podem não ser considerados no processo de identificação de registros duplicados.

Diante do exposto, este trabalho contribui no processo de limpeza de dados com o desenvolvimento de um ambiente que opera sob vários idiomas para a busca de tuplas duplicadas. Para isso, este ambiente conta com a utilização de dois algoritmos difundidos na literatura e com um algoritmo desenvolvido para a identificação desses registros por meio da similaridade fonética e numérica, utilizando a funcionalidade *multithreading* para melhorar o tempo de execução. A busca pela similaridade numérica parte do pressuposto de que os algoritmos existentes desprezam a identificação de números, desconsiderando-os como parte integrante de um registro. A utilização da fonética parte do pressuposto de que uma determinada palavra pode assumir diferentes grafias, conforme as características de pronúncia ou a falta de conhecimento das regras ortográficas e gramáticas da língua em questão (BRADLOW; CLOPPER; SMILJANIC; WALTER, 2010) (SILVA, 2000).

Portanto, verifica-se a importância do desenvolvimento de um algoritmo voltado para a tarefa de identificação de registros duplicados baseado nas características de som e números de uma palavra, visto que muitos dados são digitados e, posteriormente, armazenados não obedecendo às regras ortográficas de escrita e características numéricas. Para a elaboração deste algoritmo, foi realizado um levantamento das principais referências sobre limpeza de dados e algoritmos voltados para este fim, embora haja poucos trabalhos

que focam este desenvolvimento especificamente com a finalidade de identificar registros duplicados, tornando essa tarefa um grande desafio.

1.3. Objetivos

Este trabalho tem como objetivo propor um algoritmo capaz de identificar tuplas duplicadas somado a funcionalidade *multithreading*, além de apresentar o conceito de limpeza de dados dentro do universo da Descoberta de Conhecimento em Bases de Dados - KDD, os principais problemas encontrados na realização dessa tarefa, as técnicas para efetuar a limpeza desses dados, exemplos de ferramentas e os principais algoritmos capazes de detectar tuplas duplicadas.

Como objetivo específico, é propósito deste trabalho o desenvolvimento de um algoritmo que identifica registros duplicados em bases de dados por meio da similaridade fonética e numérica, independente de idioma. Além disso, objetiva-se o desenvolvimento de um ambiente de suporte a identificação de tuplas duplicadas por meio da aplicação dos algoritmos *Soundex*, *Metaphone* e *PSI*. A tarefa de identificar registros duplicados por meio das transcrições fonéticas em diferentes idiomas permite ao usuário do ambiente desenvolvido escolher qual idioma pretende trabalhar dentro de um leque de possibilidades.

Portanto, a proposta é trabalhar com a identificação de registros duplicados que foram armazenados na base de dados conforme as suas características de som, além de considerar os registros numéricos como parte integrante da palavra. Para um melhor desempenho do algoritmo proposto, agrega-se a funcionalidade *multithreading* com a finalidade de melhorar o seu tempo de processamento, além de apresentar melhores resultados em sua busca.

Para esta proposta, foram elaboradas etapas de funcionamento do algoritmo, com as construções de tabelas contendo diferentes transcrições fonéticas das vogais, consoantes e dígrafos de dois idiomas, português e italiano, para favorecer o entendimento e funcionamento do algoritmo proposto, fato este que auxiliará a identificação de registros que possuem similaridade fonética e numérica. Para finalizar, visa-se a análise e comparação dos resultados e do tempo de processamento obtidos entre o algoritmo proposto e os dois principais difundidos na literatura: *Soundex* e *Metaphone*.

1.4. Metodologia

O presente trabalho foi iniciado com o estudo sobre as etapas do *KDD*, posteriormente a escolha da etapa de limpeza de dados e, como foco principal, a identificação de registros duplicados nas bases de dados. Dentre todas as técnicas voltadas para a identificação, a técnica baseada em distância foi a escolhida, visto que não trabalha com dados treinados para a execução dessa tarefa, conforme constatado pelo estudo e levantamento bibliográfico realizado dos principais algoritmos existentes utilizados para esta finalidade.

A partir das leituras, do estudo das regras de funcionamento e da apresentação dos resultados dos algoritmos difundidos na literatura, foi possível notar que não atendem eficientemente as características de identificação de registros duplicados por meio da similaridade fonética e numérica, pois seu funcionamento é superficial ao atribuir valores iguais para caracteres diferentes conforme sua metodologia de funcionamento baseada na comparação fonética. Este fato contribui para a apresentação de resultados muito abrangentes e motivou a elaboração do presente trabalho.

A fim de sanar tal limitação, primeiramente foi realizado um levantamento dos algoritmos que trabalhassem com a identificação por meio da similaridade fonética e numérica. Após esse levantamento, evidenciou-se a necessidade de construção de um algoritmo que abordasse especificamente as características numéricas e fonéticas em diferentes idiomas. Uma vez constatado essa necessidade, iniciaram-se as fases de projeto e implementação do algoritmo proposto. Nessas etapas, foram consideradas as características numéricas de cada registro e elaboradas as diversas transcrições fonéticas possíveis das consoantes e vogais de diferentes idiomas, aplicando-as a programação, processamento, realização de testes e coleta de resultados.

O algoritmo elaborado para o presente trabalho foi implementado em linguagem de programação *Java*, plataforma *J2SE*, versão 1.6. O ambiente de desenvolvimento integrado utilizado para a codificação e confecção da interface gráfica foi o *Eclipse Helios*. Dentre as razões para a escolha dessa linguagem de programação, destaca-se por ser uma linguagem livre de licença, portabilidade e independência de plataforma, e existência de boa documentação oficial e não-oficial, tais como fóruns de discussão, tutoriais etc.

Para a finalização desta dissertação, realizou-se um estudo comparativo entre os resultados adquiridos por meio da utilização dos diversos algoritmos citados e o proposto. Tais testes para a análise de resultados foram realizados com o desenvolvimento de um

ambiente para a identificação de registros duplicados, a fim de permitir a verificação do desempenho do algoritmo proposto na busca por resultados comparado aos algoritmos difundidos na literatura. Vale ressaltar que o ambiente desenvolvido é para teste experimental do funcionamento do algoritmo e tem como única finalidade a identificação de registros duplicados e não a de realizar a limpeza dos dados em si. Diante dos resultados apresentados, testes com a funcionalidade *multithreading* também foram realizados, a fim de melhorar o desempenho em relação ao tempo do algoritmo proposto.

1.5. Organização da dissertação

O presente trabalho está organizado em cinco capítulos brevemente descritos da seguinte forma:

- Capítulo 2 – Limpeza de Dados: apresentação dos conceitos de limpeza de dados, explanação dos principais problemas e técnicas para solução desses problemas levantados, exemplificação de ferramentas para esta finalidade, além da caracterização da detecção de tuplas duplicadas por meio de técnicas baseadas em distância, com a apresentação dos principais conceitos, algoritmos e da aplicação dos conceitos de *multithreading* para o melhoramento de desempenho;
- Capítulo 3 – Identificação de tuplas duplicadas por meio da similaridade fonética e numérica: visão geral da proposta de desenvolvimento de um algoritmo para a identificação de registros duplicados por meio da similaridade fonética e numérica, com a caracterização das etapas de funcionamento e do ambiente de aplicação desenvolvido. Apresentação das tabelas contendo as transcrições fonéticas do estudo de caso da língua portuguesa e italiana.
- Capítulo 4 – Testes e Resultados: apresentação dos testes realizados em dois idiomas por meio do ambiente desenvolvido. Os testes consistem na apresentação dos resultados obtidos por meio da aplicação dos algoritmos *Soundex*, *Metaphone* e *PSI*. Além disso, representações gráficas de desempenho na utilização do algoritmo *PSI* com a funcionalidade *multithreading* são apresentadas neste capítulo.
- Capítulo 5 – Conclusões: apresentação das conclusões obtidas com o desenvolvimento do trabalho e algumas sugestões de continuidade do trabalho.

Capítulo 2
Limpeza de Dados

2.1. Considerações iniciais

Para iniciar este capítulo que envolve os conceitos de Limpeza de Dados, faz-se necessária uma breve explicação do termo *KDD* e suas etapas. Após a explicação, são apresentados os principais conceitos de Limpeza de Dados, os problemas comumente encontrados nas bases de dados, diversas técnicas destinadas a tratar os problemas detectados e uma breve descrição das características de algumas ferramentas voltadas para a realização da etapa de limpeza. Além disso, o conceito de identificação de registros duplicados utilizando a técnica baseada em distância e os seus principais algoritmos são abordados.

2.2. Descoberta de Conhecimento em Banco de Dados - *KDD*

Com a grande quantidade de informações armazenadas, iniciou-se a busca por um melhor aproveitamento desses dados, a fim de se obter maiores vantagens e conseqüentemente sucesso das organizações que trabalham com as informações armazenadas, consideradas um dos bens de maior valor da sociedade. Com o surgimento do processo de Descoberta de Conhecimento em Banco de Dados – *KDD*, no final da década de 80, pesquisadores passaram a transformar os dados armazenados em conhecimento útil que auxiliam as empresas e organizações nas tomadas de decisões.

Cientificamente conhecido como processo de Extração de Conhecimento das Bases de Dados, *KDD* pode ser definido como processo não trivial de identificação de padrões válidos, desconhecidos, potencialmente úteis e compreensíveis a partir de um volume de dados, onde cada termo dessa definição pode ser melhor interpretada da seguinte maneira (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996):

- Dado - conjunto de fatos de um repositório. Por exemplo, valores dos campos de um laudo;
- Padrão - conjunto de modelos que representam alguma abstração de um subconjunto de dados;
- Processo - etapas da extração do conhecimento em base de dados, como preparação dos dados e avaliação do conhecimento;
- Válidos - padrões descobertos que devem possuir algum grau de certeza, que sejam aceitáveis;
- Novos - o padrão encontrado deve fornecer novas informações sobre os dados, medido por meio de comparações entre as mudanças ocorridas nos dados ou no conhecimento anterior;
- Úteis - padrões descobertos devem ser incorporados de forma a serem utilizados;
- Compreensíveis - os padrões podem ser entendidos pelos usuários a fim de permitir a compreensão mais profunda dos dados;
- Conhecimento - dependente do domínio que estão relacionados com medidas de utilidade, originalidade e compreensão.

Diante da idéia de encontrar padrões úteis em grandes volumes de dados, várias nomenclaturas foram dadas a essa técnica. Dentre os nomes estão *knowledge extraction*, *information discovery*, *data archeology*, *information harvesting*, *data pattern processing*, mas prevalecendo as nomenclaturas *data mining* e *knowledge discovery in databases*, que muitas vezes são utilizadas como sinônimos por estarem ligadas. O conceito de *KDD* é referenciado como todo o processo de descoberta de conhecimento útil de uma base de dados, enquanto que *data mining* é referenciado à aplicação de algoritmos para extração de padrões em uma base de dados, caracterizando como uma das etapas do processo de *KDD* (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996).

O conceito de *knowledge discovery in databases* surgiu no *KDD*-89 *Workshop*, considerado o primeiro *workshop* sobre descoberta de conhecimento em banco de dados, organizado por Gregory Piatetsky-Shapiro (PIATETSKY-SHAPIRO, 1991). Esse processo

é constituído de várias etapas para extração de conhecimento, mas que podem ser resumidas em três fases principais, conforme é apresentado na Figura 2.1:

- Pré-processamento de dados - responsável por limpar e integrar os dados que servirão de matéria prima para as próximas fases. Esta fase lida diretamente com a qualidade do resultado final, tendo um grande grau de importância, oferece maior integridade e consistência aos dados a serem analisados na etapa de *data mining*;
- *Data Mining* - fase em que ocorre a aplicação de algoritmos nos dados já pré-processados, ou seja, integrados, consistentes e consolidados com o objetivo de extrair padrões;
- Pós-processamento dos resultados - fase em que são realizadas as visualizações e interpretações dos padrões extraídos. Estes dados podem ser incompreensíveis, ilegíveis e muitas vezes necessitam da ajuda de especialistas para serem interpretados.

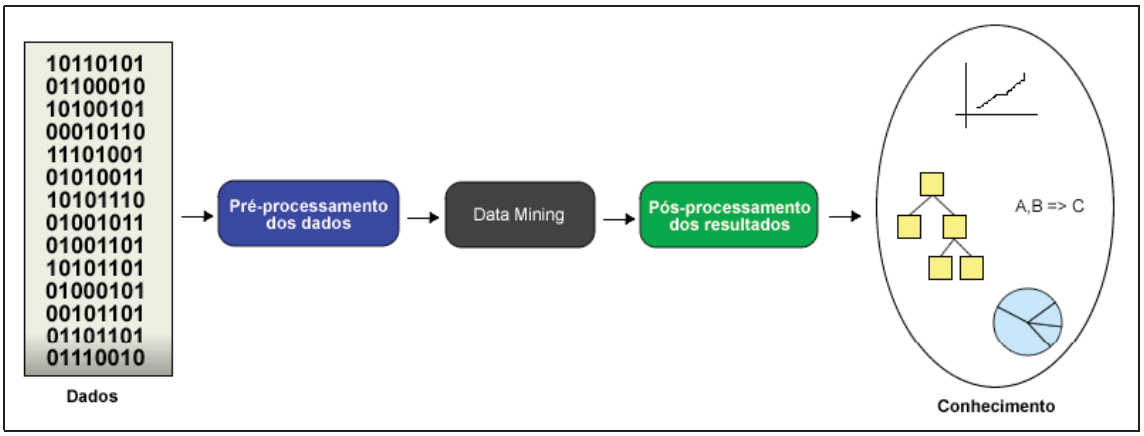


Figura 2.1 Fases do Processo de *KDD* (Baseado em FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996).

Alguns autores detalham cada fase apresentada na Figura 2.1 a fim de melhorar e caracterizar o processo de *KDD*, constituindo-o de diversos passos. Segue uma visão detalhada de cada passo, segundo a visão de (MITRA; ACHARYA, 2003):

- Compreensão do domínio da aplicação - etapa responsável em compreender o domínio da aplicação e do conhecimento prévio importante;
- Extração de um conjunto de dados alvo - etapa responsável em selecionar um conjunto de dados ou concentrar-se em um subconjunto de amostras de dados, na qual a extração de conhecimento será aplicada;

- Preparação dos dados - etapa responsável pelo fornecimento de maior qualidade dos dados para uma melhor prospecção, a fim de reduzir a quantidade de informação inútil. Divide-se em quatro subfases (HAM; KAMBER, 2001):

a) Limpeza de dados - conforme é apresentado na Figura 2.2, essa etapa é responsável pelo tratamento de valores nulos, a remoção de ruídos e o tratamento de dados inconsistentes, sendo muito importante na garantia da qualidade dos resultados futuros;

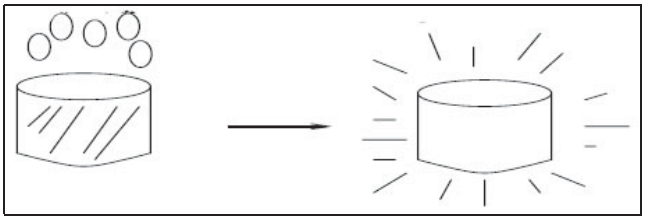


Figura 2.2 Limpeza de Dados (Extraído de HAM; KAMBER, 2001)

b) Integração dos dados - nessa etapa, as múltiplas fontes de dados são integradas em uma única fonte de dados, como um *data warehouse*, conforme é apresentado na Figura 2.3;

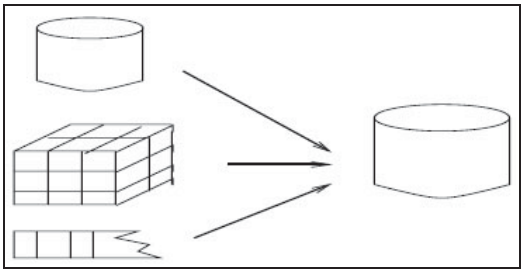


Figura 2.3 Integração de Dados (Extraído de HAM; KAMBER, 2001)

c) Transformação de dados - nessa etapa, os dados são transformados em formas apropriadas para a operação de extração e mineração, conforme é apresentado na Figura 2.4;

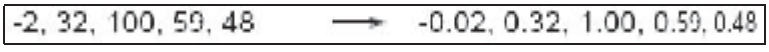


Figura 2.4 Transformação de dados (Extraído de HAM; KAMBER, 2001)

d) Redução de dados - conforme é apresentado na Figura 2.5, essa etapa é responsável pela redução do volume de dados utilizando técnicas como agregação, redução de características redundantes e agrupamento;

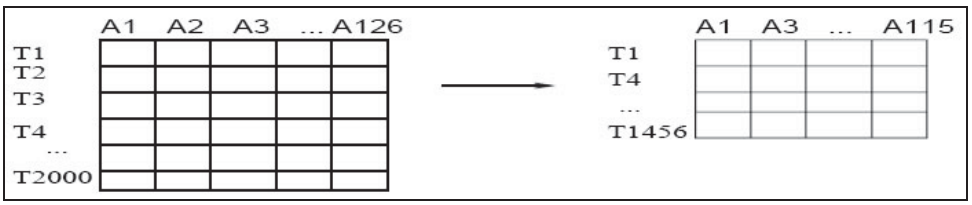


Figura 2.5 Redução de dados (Extraído de HAM; KAMBER, 2001)

- *Data Mining* - trata-se de uma etapa essencial do *KDD* no qual métodos específicos são aplicados para a extração de padrões de dados;
- Interpretação - essa etapa envolve a visualização e interpretação dos padrões descobertos, utilizando as técnicas de representação de conhecimento e visualização de dados para apresentar o conhecimento obtido, os resultados extraídos;
- Utilização do conhecimento descoberto - essa etapa envolve a utilização do conhecimento para tomada de decisões.

É importante analisar que o processo de *KDD* ilustrado na Figura 2.6 é constituído de passos iterativos e interativos, onde cada etapa do processo pode ser retomada ou avançada, conforme a visão do analista responsável pelo projeto.

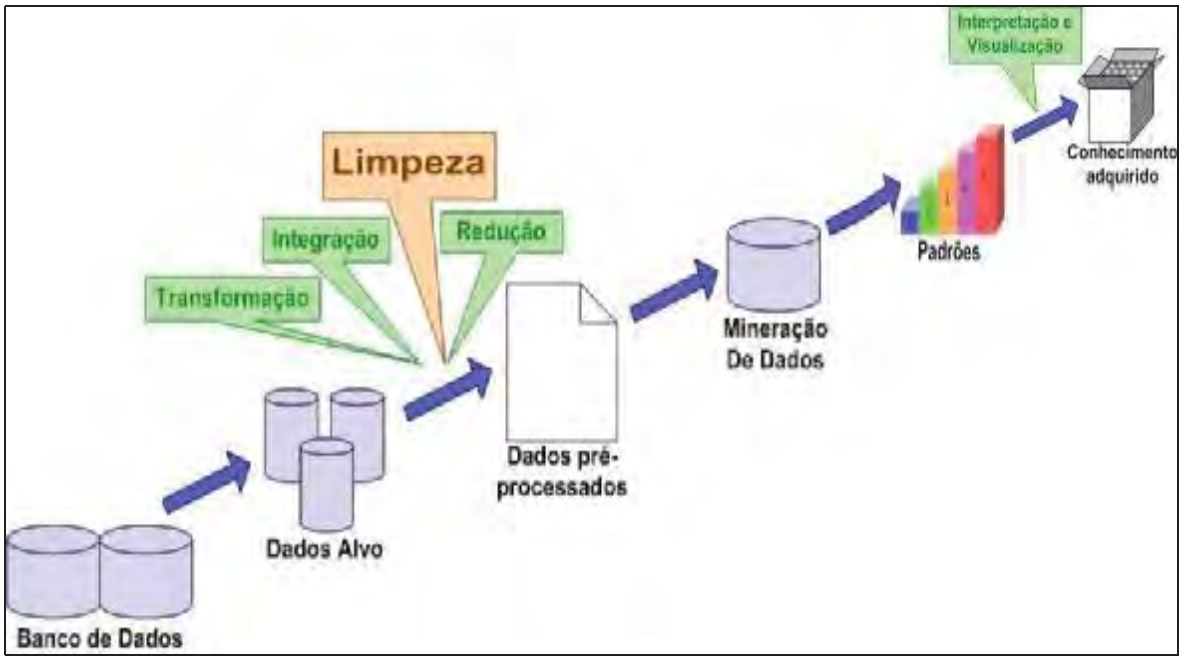


Figura 2.6 Etapas detalhadas do processo de *KDD* (Baseado em LUO, 2008).

É possível extrair o conhecimento útil necessário apenas com a fase de *data mining*, não sendo necessário as fases de pré-processamento e pós-processamento dos dados, embora sejam de grande importância para melhoramento e visualização dos dados. Daí o grande destaque para a fase de *data mining*, considerada a etapa central e a mais

importante do *KDD*, o que tem sido muitas vezes confundido com o conceito de descoberta de conhecimento. Embora isso seja possível, não é recomendável que somente a fase de *data mining* seja utilizada, pois sem a preparação dos dados poderão ser encontradas informações irrelevantes, informações sem um padrão aceitável e de difícil interpretação, o que acarretaria em informações desencontradas e sem reconhecimento do usuário.

2.3. Limpeza de Dados

Com o aumento significativo do tamanho das bases de dados, as organizações começaram a perceber o valor dos dados que tem a disposição e a considerá-los como um bem importante no aumento da produtividade, eficiência e competitividade. Por meio desses dados armazenados é possível executar algumas operações que auxiliam as estratégias e táticas de trabalho, como a análise de dados e exploração de dados. Essas operações requerem um elevado grau de qualidade dos dados, já que são utilizados na tomada de decisão (OLIVEIRA; RODRIGUES, 2004).

Ao partir para a análise dos dados nas bases de dados, verifica-se que grande parte desses dados encontra-se com algum erro, incompletos ou inconsistentes, correspondendo a valores de atributos faltantes, errados ou representados de formas diferentes dos mesmos dados. Esses dados podem influenciar negativamente nas tomadas de decisões de empresas e instituições. Estudos estatísticos revelam que as grandes empresas possuem entre 1% a 5% de dados armazenados com algum tipo de erro (FAN; GEERTS; JIA, 2008). Os erros nos dados são provocados, em grande parte, por dois motivos: preenchimento incorreto ou por falha do sistema utilizado. Então, sistemas que operam erradamente ou irregularmente favorecem o armazenamento de dados incompletos, inconsistentes e duplicados, estabelecendo uma má qualidade dos dados armazenados. Dessa forma, é necessário melhorar a qualidade dos dados, com a detecção e eliminação dos erros e inconsistências nos dados, por meio do processo chamado de limpeza de dados (YAN; DIAO, 2008).

A utilização de dados incorretos provoca uma série de problemas que prejudicam substancialmente a validade dos resultados e conclusões obtidas. A qualidade das informações armazenadas nos bancos de dados pode causar implicações significativas no custo de funcionamento dos sistemas que dependem das próprias informações armazenadas para funcionar e realizar estratégias de negócios (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007). Observa-se que os dados problemáticos devem ser identificados antes da fase de *data mining*, pois a persistência dos mesmos implica no aumento significativo

do tempo de prospecção, já que esses dados são processados desnecessariamente, de onde são extraídos padrões falsos que prejudicam as organizações que os interpretam e buscam transformá-los em conhecimento útil (RAHM; DO, 2000). Portanto, antes da aplicação da análise de dados para projeção, os dados devem ser identificados e corrigidos no intuito de remover e reparar quaisquer anomalias que possam existir, a fim de melhorar conseqüentemente a qualidade dos dados e auxiliar a uma precisa geração de conhecimento. Assim, a limpeza de dados tem sido objeto de estudos e de grande interesse nos últimos anos.

A fase de limpeza de dados está situada na etapa de pré-processamento dos dados. Conceitualmente, a limpeza de dados visa à detecção e remoção de erros e inconsistências de dados, e busca-se a melhora e o aumento de sua qualidade (RAHM; DO, 2000). Tecnicamente, o processo de limpeza de dados deve sempre ser acompanhado por um perito do domínio, pois a detecção e correção de anormalidades requerem conhecimento especializado do problema e do sistema usado. Portanto, considera-se este processo semi-automático, em virtude da interferência humana necessária e do processamento computacional nesses grandes volumes de dados (OLIVEIRA; RODRIGUES, 2004).

A abordagem de limpeza de dados envolve várias fases, independente da técnica utilizada para efetuar a limpeza em si. Todas as fases são definidas como etapas de preparação dos dados, para que posteriormente seja definida uma ou mais técnica de limpeza a serem empregadas. De maneira sucinta, a preparação envolve as fases de análise, transformação e padronização dos dados, onde se busca melhorar a qualidade dos dados, torná-los comparáveis e utilizáveis. Após a padronização, é necessária a identificação dos campos a serem utilizados para a comparação, a fim de trabalhar com áreas devidamente correspondentes, e não comparar dados não correspondentes, como por exemplo sobrenome com endereço de um indivíduo (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007). Alguns autores preferem detalhar cada fase de limpeza de dados ao especificar a funcionalidade de cada uma, conforme segue (RAHM; DO, 2000):

- Análise dos dados - deve ser feita minuciosamente a fim de detectar problemas de qualidade dos dados, onde encontra os erros e inconsistências de dados que devem ser removidos;
- Definição do fluxo de trabalho de transformação e regras de mapeamento - dependendo do número de fonte de dados e sua heterogeneidade, muitas etapas de limpeza deverão ser executadas. A primeira etapa de limpeza pode corrigir problemas de fonte única e preparar os dados para a integração. Uma etapa

posterior pode lidar com problemas de esquema / instância originados da integração de fontes, como dados duplicados. Dessa forma, cria-se um fluxo de trabalho necessário para a transformação dos dados;

- Verificação - a certeza e eficiência de um fluxo de trabalho de transformação devem ser testadas e avaliadas em uma amostra ou cópia de dados da origem, e melhorar a fase de transformação se necessário;
- Transformação - é a execução da etapa de transformação definida no fluxo de trabalho e posteriormente validada na etapa de verificação;
- Refluxo de dados limpos - depois que os erros são removidos, os “dados limpos” devem substituir os “dados sujos” nas fontes originais, a fim de melhorar a qualidade dos dados e evitar que seja necessário refazer o trabalho de limpeza nas futuras extrações de dados.

Na etapa de limpeza de dados, diversos são os problemas encontrados em bases de dados e que devem ser tratados. Os mais comuns são tuplas duplicadas, valores nulos, valores fora do domínio e valores fora do padrão (HAN; KAMBER, 2006). Mediante esses problemas comumente encontrados, pesquisas apontam que a abordagem de limpeza de dados deve satisfazer diversas exigências. Primeiro de tudo, deve detectar e remover todos os mais significativos erros e inconsistências, tanto em fontes de dados isolados susceptíveis com problemas de falha humana no processo de alimentação do banco e projeção do esquema, quanto na integração de múltiplas fontes.

Empresas e organizações governamentais passaram a medir o quanto custa em termos de tempo a execução da limpeza de dados em suas bases. Estudos indicam que limpeza de dados representa cerca de 30% do tempo de desenvolvimento em projetos que envolvam *data warehouses* (FAN; GEERTS; JIA, 2008). Em *data warehouses*, é alta a probabilidade de conter dados com algum tipo de problema devido à quantidade de dados oriundos de uma variedade de fontes de dados, o que exige um suporte importante de limpeza de dados, onde são utilizados para tomada de decisão, de modo que a exatidão dos seus dados é fundamental para evitar conclusões erradas. Então, é importante que o tratamento dos dados não seja feito de maneira isolada, mas sim juntamente com alterações aplicadas ao esquema do banco de dados, pois considera-se que alguns desses problemas ocorrem por causa de um projeto inadequado de banco de dados (RAHM; DO, 2000).

De certo modo, os problemas de limpeza de dados são também classificados como: heterogeneidade estrutural e heterogeneidade lexical. O primeiro refere-se aos

campos que tem a mesma finalidade de armazenamento, mas são estruturados diferentemente. Como por exemplo, os dados de endereço, onde em determinado banco de dados é armazenado no campo chamado ‘addr’, enquanto que em outro banco os dados são armazenados em vários campos, tais como rua, bairro e cidade. O segundo refere-se a estruturas idênticas dos campos de armazenamento, mas os mesmos dados são armazenados diferentemente, como *44 W. 4th St. versus 44 West Fourth Street* (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007).

Os problemas que são tratados pelo processo de limpeza de dados podem ser divididos em duas categorias: problemas de fonte única e problemas de múltiplas fontes (CISZAK, 2008) (RAHM; DO, 2000).

2.3.1. Problemas de fonte única

A qualidade dos dados de uma fonte depende muito de como está sendo controlada, por um esquema e por regras de integridade. Arquivos de texto possuem grande probabilidade de conter erros, já que não possuem esquema e nem definição de regras de integridade, o que ocasiona um controle manual, e conseqüentemente, sujeito a falhas. A existência de um esquema definido em algum modelo de dados específico não garante integridade e consistência de informações quando não utilizadas de forma adequada, ou seja, nem sempre a existência de um esquema e de regras de integridade garante a ausência de erros em uma coleção de dados. Além da possibilidade do modelo apresentar algumas falhas e limitações, a omissão de definição de algumas restrições no projeto e construção da base de dados pode conduzir a um projeto inadequado e com falhas (RAHM; DO, 2000). Destacam-se dois exemplos de problemas de fonte única: o primeiro em nível de esquema ilustrado na Tabela 2.1 e o segundo em nível de instância exibido na Tabela 2.2.

Tabela 2.1 Problemas em nível de esquema. (Adaptado de RAHM; DO, 2000).

Escopo/Problema		Dado Sujo	Razão/Advertência
Atributo	Valores ilegais	data_nasc=13/30/70	Valor fora do domínio
Registro	Dependência de atributo violada	idade=22, data_nasc=12/02/70	Idade=(data atual – data de nascimento)
Tipo de Registro	Violação de unicidade	emp1=(nome="John Smith", SSN="123456") emp2=(nome="Peter Miller", SSN="123456")	Violação de unicidade para SSN
Fonte	Violação de integridade referencial	Emp=(nome="John Smith", depto=127)	Departamento referenciado (127) não definido

Tabela 2.2 Problemas em nível de instância. (Adaptado de RAHM; DO, 2000).

Escopo/Problema		Dado Sujo	Razão/Advertência
Atributo	Valores ausentes	fone=9999-999999	Dados não disponíveis durante a entrada
	Erros de digitação	cidade="São José do Rio Petro"	Erros na digitação ou fonéticos
	Valores integrados	endereço="Rua A, 555, São Paulo"	Múltiplos valores em um mesmo atributo
	Valores desencontrados	cidade="Brasil"	
Registro	Violação de dependência de atributo	cidade="São Paulo", CEP=999999	Cidade e CEP devem corresponder
Tipo de Registro	Registros duplicados	emp1=(nome="John Smith"); emp2=(nome="John Smith")	Representação duplicada do mesmo empregado devido a algum erro de entrada de dados
Fonte	Referências erradas	emp=(nome="John Smith", depto=17)	Departamento referenciado (17) está definido, mas errado

2.3.2. Problemas de múltiplas fontes

Os problemas de fonte única podem ser agravados quando múltiplas fontes precisam ser integradas, aumentando significativamente a necessidade de limpeza de dados. Esta tarefa pode ser realizada para detectar e remover redundâncias originadas a partir da integração de dados (HAM; KAMBER, 2006). Cada fonte pode conter dados "sujos" ou dados representados diferentemente. Isso ocorre porque as fontes são desenvolvidas e mantidas de forma independente, atendendo as necessidades específicas de uma organização, gerando um alto nível de heterogeneidade entre os dados e suas estruturas. Como exemplo dessa heterogeneidade em relação ao esquema, cita-se quando um mesmo nome faz referência a atributos diferentes, por exemplo, o campo data referencia data de nascimento e data de cadastro, nomes diferentes referenciam atributos iguais, por exemplo Rua/Logradouro, representação diferente de um mesmo atributo, por exemplo nome completo/primeiro nome, último nome. Já a heterogeneidade em relação à instância, cita-se conflitos entre dados causados por interpretações diferentes para atributos específicos, como moeda (Dollar/Euro), comprimento (quilômetro/metro), tempo (hora/minuto) ou domínios de valores diferentes para representar um mesmo atributo (0 – Masculino, 1 - Feminino) (RAHM; DO, 2000). Alguns exemplos são ilustrados nas Tabelas 2.3 e 2.4.

Tabela 2.3 Consumidor (fonte 1) (Adaptado de RAHM; DO, 2000).

Código	Nome	Rua	Cidade	Sexo
11	Kristen Smith	2 Hurley PI	South Fork, MN 48503	1
24	Christian Smith	Hurley St 2	S Fork MN	0

Tabela 2.4 Cliente (fonte 2) (Adaptado de RAHM; DO, 2000).

Cód	Último nome	Primeiro Nome	Gênero	Endereço	Fone/Fax
24	Smith	Christoph	M	23 Harley St, Chicago IL, 60633-2394	333-222-6542 / 333-222-6599
493	Smith	Kristen L.	F	2 Hurley Place, South Fork MN, 48503-599	444-555-666

Na Tabela 2.5 é apresentado a junção das duas Tabelas 2.3 e 2.4, onde são eliminados todos os problemas de integração. Nota-se que nas Tabelas 2.3 e 2.4 encontram-se vários problemas de esquema como os conflitos de nomes e conflitos estruturais como representações diferentes para nomes e endereços. Já problema de instâncias, é possível verificar a utilização de domínios diferentes de valores para o atributo Sexo/Gênero, hora representado como 0/1 ou M/F (RAHM; DO, 2000).

Tabela 2.5 Consumidores (Integração e limpeza) (Adaptado de RAHM; DO, 2000).

Nº	Unome	Pnome	Gênero	Rua	Cidade	Estado	Cep	Fone	Fax	Código	Cod
1	Smith	Kristen L.	F	2 Hurley Place	South Fork	MN	48503-599	444-555-666		11	493
2	Smith	Christian	M	2 Hurley Place	South Fork	MN	48503-599			24	
3	Smith	Christoph	M	23 Harley Street	Chicago	IL	60633-2494	333-222-6542	333-222-6599		24

Observa-se que no processo de junção de fontes de dados, dados repetidos devem ser eliminados, enquanto os dados complementares devem ser agregados à tabela principal. Na Tabela 2.5 é apresentada a solução, onde todos os problemas relacionados ao esquema, necessariamente aos conflitos de nome e estrutura, foram resolvidos por meio de padronização. Já os problemas de instâncias também foram resolvidos por meio da padronização dos domínios de valores e eliminação de tuplas duplicadas, agregando dados complementares.

Atualmente, muitos destes recursos realizam a limpeza dos dados errados principalmente por meio da execução de um conjunto de regras pré-estabelecidos em sistemas. Dessa forma, verifica-se que essas operações de limpeza podem não satisfazer plenamente as exigências dos usuários, porque as regras de limpeza são pré-estabelecidas e não podem ser ampliadas ou modificadas de acordo com as necessidades práticas dos usuários. Assim, as atuais ferramentas voltadas a tarefa de limpeza de dados têm como característica à limitada capacidade de extensão, onde muitas vezes não atende os interesses do usuário, comprometendo a utilização desses sistemas (YAN; DIAO, 2008).

2.4. Técnicas de Limpeza de Dados

Como descrito anteriormente, os bancos de dados muitas vezes encontram-se com problemas na qualidade de seus dados, apresentando-se na maioria das vezes incompletos, fora de padrão e inconsistentes. Diante disso, são apresentadas algumas técnicas pesquisadas para efetuar a limpeza de dados a fim de solucionar os principais problemas em um banco de dados, como valores ausentes, valores fora de padrão e identificação de tuplas duplicadas.

2.4.1. Valores Ausentes

A principal característica é a existência em diversas tuplas de atributos que não possuem valores armazenados, os quais podem ser importantes no processo de mineração, implicando numa redução de qualidade. Dentre as principais técnicas apresentadas estão (HAM; KAMBER, 2006):

- Ignorar a tupla - utilizada quando o conteúdo da variável estiver ausente, assumindo que o processo de *data mining* envolverá a técnica de classificação. Não é uma técnica muito desejável, a menos que existam muitas tuplas com valores ausentes;
- Preencher o valor manualmente - consiste em preencher manualmente todas as tuplas que possuir valores ausentes. Consome-se muito tempo e pode não ser possível em grandes bases de dados que possuem muitos valores ausentes;
- Usar uma constante global para preencher os valores ausentes - técnica de preenchimento dos valores ausentes utilizando um único valor constante, como “desconhecido”. Embora seja simples, não é recomendado;
- Utilizar um atributo médio para preencher os valores ausentes - utilizado quando se trata de valores numéricos, calcula-se a média de todos os valores existentes de um atributo e atribui aos valores ausentes. Possui boa aceitação;
- Utilizar um atributo médio pertencente a mesma classe a qual a tupla pertença - utilizado também quando se trata de valores numéricos, calcula-se a média baseando-se em algum grupo em que a tupla pertence e atribui aos valores ausentes. Também possui uma boa aceitação;
- Utilizar o valor de maior probabilidade para preencher os valores ausentes - técnica de preenchimento de valores ausentes baseada em técnica de regressão, formalismo

bayseano ou indução de árvores de decisão, predizendo um provável valor para preencher os valores ausentes. Trabalhar com predições de valor talvez não seja uma técnica adequada para determinadas situações.

2.4.2. Valores fora de padrão

Valores fora do padrão podem ser encontrados na busca de variáveis, realçando uma variação acentuada na medição dessa variável. Dentre as principais técnicas para correção desses valores estão (HAM, KAMBER, 2006):

- *Binning* - atribuídos a valores numéricos, é um método que ordena os valores do atributo, e após a ordenação, os valores são distribuídos em grupos contendo a mesma quantidade de elementos. Em cada grupo, substitui os valores pelas medidas calculadas, podendo ser realizada por meio da média aritmética, mediana ou medida de valor limite, ajustando dessa forma os valores dos atributos;
- Agrupamento - método que detecta valores fora do padrão por meio de agrupamento de valores similares. Os valores que ficam fora dos grupos são considerados valores fora do padrão;
- Combinação de inspeção humana e computador - valores fora de padrão podem ser identificados por meio da combinação de inspeção humana e do uso do computador. Uma vez identificados e considerados lixo, podem ser excluídos;
- Regressão - essa técnica visa um melhor ajuste dos dados por meio das funções de ajustamento de dados, como funções de regressão.

Alguns algoritmos podem ser empregados a fim de solucionar os problemas encontrados em variáveis fora de padrão, a fim de obter uma padronização de valores. Dentre os algoritmos utilizados atualmente estão *Edit Distance*, *Soundex*, etc. Todos esses algoritmos trabalham na identificação de similaridade entre *strings*.

2.4.3. Detecção de tuplas duplicadas

Diversas são as técnicas utilizadas atualmente capazes de detectar tuplas duplicadas (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007). Entre elas, citam-se:

- Modelo probabilístico apropriado - posteriormente conhecida como *Inferência Bayseana*, verifica-se equivalência entre tuplas diferentes. Uma determinada tupla

designada *x* é obtida como entrada e submetida a um modelo a fim de decidir a qual classe essa tupla pertence, dentre as várias classes pré-existentes, caracterizando assim a equivalência de tuplas pertencentes a uma mesma classe;

- Aprendizado supervisionado e semi-supervisionado - essa técnica de aprendizagem conta com a existência de um conjunto de dados treinados sob a forma de pares de tuplas, pré-classificando os dados dessas tuplas como equivalentes ou não equivalentes. Assim, é possível classificar se qualquer par de tuplas são equivalentes ou não, estabelecendo um modelo de aprendizado supervisionado. Essa técnica exige dados de treinamento computacional ou esforço humano para criar modelos de dados, sendo necessário um grande número de formação de exemplos para posterior classificação e comparação de equivalências. Dentre os mais conhecidos algoritmos, cita-se o *CART*, capaz de separar os dados de acordo com a sua classe;
- Técnica baseada em conhecimento ativo - considerada uma evolução da técnica de aprendizado supervisionado, capaz de reconhecer casos ambíguos. A técnica de aprendizado supervisionado e semi-supervisionado mencionada acima apenas aponta se as tuplas são ou não equivalentes, não sendo capaz de identificar casos ambíguos, semelhantes. Essa técnica já é capaz de identificar os casos ambíguos por meio de um conjunto de dados treinados sob a forma de pares de tuplas, aprendendo rapidamente as peculiaridades desse conjunto de dados e detectando tuplas duplicadas, utilizando um pequeno número de conjunto de dados de treinamento. O algoritmo mais conhecido que emprega a utilização dessa técnica é chamado de *ALIAS*;
- Técnicas baseadas em distância - diferentemente das técnicas que trabalham baseando-se na formação de um conjunto de dados treinados, essa técnica opera sem a necessidade dessa formação de dados, tratando as tuplas como grandes *strings*. Após esse tratamento, utiliza-se de algoritmos capazes de identificar similaridade entre essas tuplas. Entre os principais algoritmos, estão o *Edit Distance* e *Soundex* (XIAO; WANG; LIN; YU, 2008);
- Técnicas baseadas em regras - baseada nas técnicas de distância, utiliza-se de regras desenvolvidas por especialistas que são capazes de identificar um conjunto de atributos e usá-los coletivamente, a fim de coletar dados semelhantes entre tuplas, agrupando registros múltiplos que representam à mesma entidade do mundo real. Para exemplificação, utiliza-se uma regra que busca em tuplas diferentes

pessoas com mesmo nome e o mesmo endereço, conseqüentemente infere-se que seja a mesma pessoa;

- Técnica de aprendizado não supervisionado - tradicionalmente conhecida como técnica de agrupamento, esta técnica recorre a existência de um conjunto de dados treinados para classificação de equivalência. Portanto, utiliza-se de algoritmos que agrupam tuplas equivalentes em um mesmo grupo, e tuplas não equivalentes em grupos diferentes.

Dentre as diversas técnicas apresentadas para detecção de tuplas duplicadas, é improvável que seja esclarecido em breve qual delas é a melhor a ser usada ou mais eficiente. O esforço na tarefa de detecção de tuplas duplicadas é altamente dependente da qualidade dos dados armazenados, sendo muitas vezes necessária a adoção de apenas uma das técnicas para a solução do problema de detecção, ou a adoção de várias técnicas conjuntamente (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007).

2.4.4. Demais técnicas de limpeza de dados

Embora as técnicas apresentadas sejam consideradas as principais na identificação de valores incompletos, fora de padrão ou inconsistentes, existem técnicas muito importantes que lidam diretamente com valores fora de domínio, ou seja, valores encontrados na base de dados que muitas vezes são discrepantes da realidade. Esses valores podem ser encontrados em atributos que armazenam principalmente dados numéricos, como é o caso do atributo idade. Fazendo uma busca de valores nesse atributo, podem ser encontrados valores que não correspondem a valores válidos dentro de um padrão, como é o caso de ser encontrado um valor como 300 (trezentos) para um atributo correspondente a idade de uma pessoa. Portanto, para o tratamento de valores fora de domínio pode ser usada a técnica de consultas diretamente em base de dados ou utilizando algoritmos atuais como *Squeezer* (HE; XU; DENG, 2005).

2.5. Detecção de tuplas duplicadas por meio de técnicas baseadas em distância

Com o advento da massificação da informação vem se cristalizando não apenas o interesse pelo amplo armazenamento dos dados, mas também pela garantia da qualidade

destes dados armazenados que muitas vezes encontram-se com algum tipo de problema. Dentre os problemas identificados há o de registros duplicados, que passou a ser atacado por diversos pesquisadores a fim de solucionar e melhorar a qualidade das informações armazenadas. Importante na melhoria da qualidade dos dados armazenados, o processo de detecção de dados duplicados garante uma melhor integridade e confiabilidade dos dados da base que serão trabalhados na aquisição de informações (CHRISTEN, 2009).

Pesquisas sobre registros duplicados receberam as mais diversas nomenclaturas, como detecção de duplicados, incertezas de identidade, registros encadeados, entre outros. Essa diversidade de nomes é reflexo de várias pesquisas com diferentes finalidades e das mais variadas áreas de banco de dados, como estatísticas e mineração de dados, entre outras (BILENKO; MOONEY; COHEN; RAVIKUMAR; FIENBERG, 2003).

Para um melhor aproveitamento das informações armazenadas e elaboração de estratégias e tomadas de decisões, é importante que os dados estejam armazenados corretamente, pois dados ruins podem influenciar negativamente no resultado de uma pesquisa. A má qualidade dos dados armazenados é um grande obstáculo a ser resolvido pela detecção de registros duplicados, pois dificulta a comparação dos registros devido às informações estarem incorretas ou faltantes. Assim, a limpeza de dados é um passo importante para a melhoria da qualidade dos dados armazenados, convertendo dados ruins em dados de formatos compatíveis e bem definidos, para que posteriormente ocorra a etapa de eliminação de registros duplicados (CHRISTEN, 2009).

Um dos itens mais estudados por pesquisadores é a detecção e eliminação de registros duplicados. Diante disso, essa tarefa é considerada um dos maiores problemas da etapa de limpeza de dados, visto que a detecção de registro duplicado é que identifica se um dado armazenado tem uma outra representação armazenada igual ou semelhante na base de dados, ou seja, visa detectar todos os casos em que um mesmo objeto do mundo real tem múltiplas representações em um banco de dados (WEIS; NAUMANN; JEHLE; LUFTER; SCHUSTER, 2008). A maioria dos estudos científicos vem se aprofundando em técnicas que trabalham com similaridade textual para determinar se existem registros duplicados nas bases, utilizando-se de algoritmos de identificação de similaridade, como *Edit Distance* (ARASU; KAUSHIK, 2009).

Preferencialmente adotada pela simplicidade de funcionamento, a técnica de detecção de tupla duplicada baseada em distância tem ganhado muito espaço por não depender de dados treinados. Técnicas baseadas em conhecimentos pré-definidos, como é o caso dos baseados em aprendizagem, geralmente necessitam de modelos de dados,

aprendendo as peculiaridades desses conjuntos de dados para posteriormente detectar tuplas duplicadas por meio dos dados treinados.

A detecção de tuplas duplicadas tem como característica um processamento muito custoso, trabalhoso, visto que é necessário comparar cada registro utilizando uma métrica de similaridade, mesmo que essa métrica seja funcionalmente simples, como é o caso da baseada em distância (WEIS; NAUMANN; JEHLE; LUFTER; SCHUSTER, 2008). Mesmo assim, tem sido objeto de grandes estudos e dedicação por parte dos pesquisadores do mundo inteiro.

A técnica baseada em distância não necessita de dados treinados para seu funcionamento. Essa técnica tem como característica a detecção de registros duplicados em banco de dados por meio da utilização de métricas de distância, ou seja, algoritmos que utilizam a comparação de caracteres como forma de detecção de registros iguais ou semelhantes. Para isso, é necessária a adoção das fases anteriormente citadas como etapas de preparação dos dados e definição de um ou mais algoritmos a serem utilizados. Além disso, é possível realizar a detecção em uma única base ou várias bases, independente de estarem interligadas ou não.

Portanto, a grande vantagem da detecção de tuplas duplicadas por meio da técnica baseada em distância é a capacidade de operar sem a formação de dados treinados pré-existentes, utilizando o método de comparação de caracteres (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007).

Uma importante observação para detecção de registros duplicados numa base de dados é que nem sempre os dados duplicados possuem suas representações textualmente semelhantes. Dois registros podem ter a mesma representação, mas serem textualmente desiguais, por exemplo M. Stonebraker *versus* Michael Stonebraker, da mesma forma que pode existir registro com muita semelhança textual, mas inversamente diferentes na sua representatividade, como por exemplo Laptop Thinkpad, T43, Centrino, PM, 750, 1.8Ghz, 512Mb *versus* Laptop Thinkpad, T43, Centrino, PM, 740, 1.7Ghz, 512Mb, exigindo maior versatilidade e eficiência dos algoritmos empregados para tal detecção (ARASU; KAUSHIK, 2009).

Para detecção de tuplas duplicadas, diversos são os algoritmos que podem ser utilizados. A seguir, serão apresentados os principais algoritmos baseados em métricas de similaridade, caracterizando seus conceitos relevantes e funcionalidade.

2.5.1. Algoritmo *Soundex*

Conhecido como algoritmo de distância baseado em métricas de similaridade fonética, *Soundex* foi desenvolvido por Odell/Russel em 1918 e é capaz de dizer se a pronúncia de duas *strings* são parecidas apenas utilizando as suas consoantes, ou melhor, realizando a codificação por meio de suas consoantes. Esse algoritmo adota o seguinte procedimento de codificação (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007):

1. Mantém a primeira letra da *string* como prefixo e ignora as letras H e W
2. Assume os seguintes códigos para as letras restantes:
   - B, F, P, V = 1;
   - C, G, J, K, Q, S, X, Z = 2;
   - D, T = 3;
   - L = 4;
   - M, N = 5;
   - R = 6;
3. A, E, I, O, U e Y não são codificados, mas utilizados como separadores;
4. Elimine os códigos duplicados;
5. Elimine os separadores (vogais + Y);
6. Mantenha a letra prefixo (definida no passo 1) e os três primeiros códigos seguintes formando um código *soundex* com quatro posições. Caso exista menos do que três códigos após a letra prefixo, complete com 0's, formando o código *soundex* de tamanho quatro, atendendo o formato <letras><dígito><dígito><dígito>.

Portanto, uma palavra como *Hobbs* assumiria uma codificação igual a "H120". Realizando o mesmo procedimento com a palavra *Hubbs*, a codificação também seria "H120". Neste caso acusaria similaridade entre as duas palavras, o que não deixa de ser verdade, sendo diferenciada apenas por uma vogal que é desprezada pelo processo de codificação *soundex*.

Aprimoramento da codificação *soundex* já foi realizado pela dificuldade desse algoritmo codificar palavras principalmente de origem asiática, judia e germânica. Em 1985, Gary Mokotoff e Randy Daitch desenvolveram o algoritmo *Daitch-Mokotoff Soundex* buscando solucionar este problema, realizando uma modificação no algoritmo, transformando uma determinada palavra em um código de seis caracteres. Posteriormente a esta data, em 1988 um novo algoritmo chamada *Phonix* foi desenvolvido, aprimorando

também a codificação *soundex*. *Phonix* apresenta um método de funcionamento mais complexo do que *Soundex*, embora seja baseado também na substituição de caracteres por códigos numéricos (GÁLVEZ, 2006).

2.5.2. Algoritmo *NYSIIS*

Diferente da codificação *Soundex* que utiliza apenas as consoantes, a codificação *NYSIIS* (*New York State Identification and Intelligence System*) utiliza todos os caracteres de uma palavra, transformando-os em códigos alfa numéricos. Essa técnica de codificação foi proposta por Robert L. Taft em 1970. Seu método de funcionamento realiza a codificação de uma palavra em até seis caracteres, substituindo as consoantes por outras letras foneticamente parecidas.

Segundo estudos científicos realizados pelo autor da codificação, *NYSIIS* tem uma eficiência maior em busca de resultados em comparação com *Soundex*, girando em torno de 3%. Atualmente, tem sido utilizado para serviços da Divisão da Justiça Criminal do Estado de Nova Iorque (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007). As regras utilizadas pelo algoritmo são as seguintes (GÁLVEZ, 2006):

1. O primeiro caracter corresponde ao primeiro caracter do código;
2. Transforma os primeiros caracteres do nome em:
    - MAC = MCC;
    - PH = FF;
    - KN = NN;
    - K = C;
    - SCH = SSS;
3. Transforma os últimos caracteres do nome em:
    - EE = Y;
    - IE = Y;
    - DT, RT, RD, NT, ND = D;
    - Se o ultimo caracter é S, eliminá-lo;
    - Se o último caracter é A, eliminá-lo;
    - Se os últimos caracteres são AY, substituir por Y.

Portanto, aplicando o algoritmo *NYSIIS* na palavra *Hobbs*, a codificação originada seria “HAB”. Realizando o mesmo procedimento com a palavra *Hubbs*, a codificação também seria “HAB”, acusando similaridade entre as duas palavras.

2.5.3. Algoritmo *Metaphone*

Desenvolvida por Lawrence Philips em 1990, utiliza apenas 16 consoantes foneticamente parecidas, codificando-as para possíveis comparações em busca de nomes semelhantes. Esse algoritmo remove todas as vogais das palavras, ao menos que a vogal seja a primeira letra, restando apenas as consoantes (GÁLVEZ, 2006), formando um código de até quatro caracteres. Uma palavra como *Hobbs* teria sua codificação *Metaphone* como *HBS*.

Posteriormente melhorada, a *Double Metaphone* permite múltiplas codificações de nomes que podem ter várias pronúncias possíveis, tentando as várias possibilidades de codificação e recuperando os nomes semelhantes (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007). O algoritmo *Metaphone* funciona da seguinte forma:

- B → B, exceto em final de uma palavra;
- C → X, se aparecer como -cia-, -ch-;
  S, se aparecer como -ci-, -ce-, -cy-;
  K para o resto dos casos;
- D → J se aparecer como -dge-, -dgy-, -dgi-;
  T para o resto dos casos;
- F → F;
- G → silêncio, se aparecer como -gh-;
  J se estiver a frente de -i-, -e-, -y-;
  K para o resto dos casos;
- H → silêncio, se aparecer depois de vogal e não seguida por vogal;
  H para o resto dos casos;
- J → J;
- K → silêncio se aparecer depois de -c-;
  K para o resto dos casos;
- L → L;
- M → M;
- N → N;

- P → F se aparecer antes de -h-;
  P para o resto dos casos;
- Q → K;
- R → R;
- S → X se aparecer antes de -h- ou como -sio-, -sai-;
  S para o resto dos casos;
- T → X se aparecer como -tia-, -tio-;
  O se aparecer antes de -h-;
  T para o resto dos casos;
- V → F;
- W → silêncio, se não está seguida por vogal;
  W se está seguida por vogal;
- X → KS;
- Y → silêncio, se não está seguida por vogal
  Y se está seguida por vogal;
- Z → S.

2.5.4. Algoritmo *Edit Distance*

Também conhecido como *Distância de Levenshtein*, o algoritmo *Edit Distance* tem a finalidade de calcular a menor distância entre duas *strings*, calculada pela menor quantidade de operações de edição de caracteres para transformar determinada palavra em outra (palavra 1 em palavra 2) e vice-versa (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007). Para a transformação, cita-se três tipos de adições que podem ser realizadas:

- Inserção de caracter em uma palavra, transformando-a em outra - FCA em FACA.
- Remoção de um caracter em uma palavra, transformando-a em outra - GARFOS em GARFO.
- Substituição de um caracter por outro diferente - TRAS por TRAZ.

Observa-se que este algoritmo não trabalha com codificação de caracteres, substituindo um caracter por um código alfanumérico. No entanto, é muito utilizado por pesquisadores de banco de dados não só para identificação de similaridade de caracteres textuais, mas também por pesquisadores biólogos que utilizam para identificação de

seqüências biológicas (BILENKO; MOONEY; COHEN; RAVIKUMAR; FIENBERG, 2003).

2.5.5. Algoritmo ONCA

O algoritmo *ONCA* (*Oxford Name Compression Algorithm*) trabalha com duas técnicas diferentes para detecção de tuplas duplicadas. Primeiramente, este algoritmo faz uso do algoritmo *NYSIIS*, comprimindo o tamanho da palavra original. Em seguida, utiliza-se do algoritmo *Soundex* para codificação (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007).

2.5.6. Demais exemplos de algoritmos

Existem muitos outros algoritmos que merecem ser citados neste trabalho, mas que são menos relevantes por se basearem em técnicas já apresentadas, por tratar problemas muito específicos, ou pela limitada característica de funcionamento e operacionalidade. Entre os algoritmos que trabalham com medições de similaridade, pode-se citar o *Affine Gap Distance*, *Jaccard similarity*, *Smith Waterman Distance* (muito utilizado na Bioinformática), *Jaro Distance Metric* (destinado principalmente para pequenas palavras, como nomes e sobrenomes) *Cosine Similarity* e *Q-Gram Distance* (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007) (BILENKO; MOONEY; COHEN; RAVIKUMAR; FIENBERG, 2003). Todos os citados possuem suas particularidades de funcionamento e características diversificadas de desenvolvimento, embora sejam voltados para identificação de tuplas duplicadas por meio da similaridade em *strings*, especificamente para tratar erros de entrada de dados e armazenamento de dados digitados erradamente.

2.6. Ferramentas de Limpeza de Dados

Existem muitas ferramentas voltadas para a limpeza de dados. Grande parte delas estão disponíveis gratuitamente ou por preços bem acessíveis, mas com funcionalidade restrita, limitados a lidar com diferentes tipos de dados. Há ferramentas mais robustas, mas devido ao alto custo são acessíveis apenas às grandes organizações comerciais (CHRISTEN, 2009). Várias dessas ferramentas ainda dependem de uma intensiva interferência humana para seu correto funcionamento, principalmente quando se trata de

detectar registros duplicados, visto que uma detecção e conseqüentemente a eliminação completa totalmente automática via sistema sem intervenção humana pode resultar em perda de informações precisas e valiosas do banco de dados (HASSANZADEH; MILLER, 2009).

Muitas ferramentas voltadas a limpeza de dados estão relacionadas na página *web Kdnuggets* (KDNUGGETS, 1997), que discute sobre *KDD* e de todas as suas etapas. Diante disso serão abordadas a seguir algumas dessas ferramentas:

- *WinPure*: lida diretamente com a padronização de valores e detecção de tuplas duplicadas;
- *DQ Global*: realiza a detecção de tuplas duplicadas;
- *GritBot*: trabalha na padronização de valores;
- *DQ Now*: responsável pela detecção de tuplas duplicadas;
- *ProMiss*: especializada na detecção de valores ausentes;
- *Data Ladder*: especializada na detecção de tuplas duplicadas.

Outras ferramentas têm sido destacadas em artigos científicos e relatadas por meio da utilização dos algoritmos utilizados para a construção (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007) (WEIS; NAUMANN; JEHLE; LUFTER; SCHUSTER, 2008). Dentre as estudadas com a finalidade de detectar tuplas duplicadas, destacam-se:

- *Febrl*: utiliza os algoritmos *Soundex*, *Metaphone*, *Edit Distance*, *Jaro Distance Metric* e *Jaccard Similarity*;
- *BigMatch*: utiliza os algoritmos *Soundex* e *Metaphone;*
- *WordMatcher*: utiliza os algoritmos *Soundex*, *Metaphone*, *Edit Distance* e *Jaro Distance Metric*;
- *Whirl4*: utiliza os algoritmos *Edit Distance* e *Jaccard Similarity;*
- *DogMatiX*: utiliza basicamente o algoritmo SNM.

É possível observar que nem todas as ferramentas fazem uso exclusivamente de algoritmos fonéticos para a identificação de tuplas duplicadas, como os mencionados *Soundex* e *Metaphone,* mas que tem essa tarefa desempenhada com eficiência. Neste sentido, destaca-se a ferramenta proposta neste trabalho denominada DIBP (*Data Identification Base don Phonetic*) que utiliza somente os algoritmos fonéticos *Soundex* e *Metaphone,* além do algoritmo proposto neste trabalho denominado *PSI*. Detalhes são abordados no capítulo 3.

2.7. *Multithreading*

Com os avanços significativos das arquiteturas de computadores, o volume de informações armazenadas passou a ser maior, exigindo um grande poder de processamento por parte dos computadores das informações guardadas. Consequentemente, todos os processos envolvidos no tratamento das informações passaram a ser objetos de estudos, sendo que a limpeza de dados tornou-se um dos focos desses processos.

Com a importância da limpeza de dados, os algoritmos desenvolvidos para esta finalidade passaram a ser objeto de grandes estudos. Vários foram desenvolvidos, mas descartados pelo elevado tempo de execução e resultados inadequados. Diante disso, uma das principais dificuldades encontradas ao se tratar de limpeza de dados envolve o tempo de execução dos algoritmos nas bases de dados, pois bases que contém milhares de tuplas exigem maiores processamentos e, devido ao grande número de registros, o tempo de execução torna-se maior, aumentando proporcionalmente conforme o número de registros. Portanto, o processamento em *multithreading* vem ganhando força nos aplicativos desenvolvidos, a fim de dividir as tarefas a serem executadas e melhorar o desempenho de aplicação, diminuindo o tempo gasto para executá-lo (Kyueun; Gaudiot, 2010).

O conceito de *multithreading* pode ser dividido em três categorias (SILVA, 2011):

- Granulosidade fina – apenas uma instrução de cada *thread* é executada em cada instante de tempo;
- Granulosidade grossa – as instruções de uma *thread* são executadas sucessivamente até que um evento de grande latência ocorra;
- *Multithreading* simultâneo – as instruções são enviadas simultaneamente para as unidades funcionais de um processador superescalar e consequentemente são executadas ao mesmo tempo.

Atualmente, os aplicativos estão sendo desenvolvidos para fazer uso desse tipo de funcionalidade, pois a maioria dos computadores já dá suporte a esse tipo de instrução, a fim de melhorar o tempo de execução. Nos computadores recentes, este tipo de aplicabilidade evita o grande desperdício do poder computacional existente.

Diante disso, o conceito de *multithreading*, neste trabalho, será adotado para divisão do processamento das tarefas de execução entre os núcleos de processadores existentes na aplicação do algoritmo proposto, a fim de mensurar o tempo de execução e desempenho na tarefa de identificar registros duplicados na base de dados.

2.8. Considerações finais

Neste capítulo foram abordados os principais aspectos sobre Limpeza de Dados, situando-a dentro do universo de Descoberta de Conhecimento em Banco de Dados, e apresentadas as diferentes técnicas para a eliminação de incidências em Banco de Dados de instâncias que comprometam a integridade dos dados e que fogem do domínio do SGBD e bons projetos de Banco de Dados. Além disso, foram contextualizados os principais conceitos relativos a detecção de tuplas duplicadas, principalmente a técnica baseada em distância, a fim de enfatizar os aspectos positivos da escolha desta técnica e caracterizar o seu funcionamento. Também foram apresentados os principais algoritmos utilizados pela técnica baseada em distância para detecção de tuplas duplicadas, suas características de funcionamento e as principais ferramentas existentes que utilizam tal técnica, entre outras. Para finalizar, foram apresentados os principais conceitos de *multithreading* caracterizando-o dentro do universo deste trabalho.

Capítulo 3
Identificação de tuplas duplicadas por meio da similaridade fonética e numérica

3.1. Considerações iniciais

A identificação de registros duplicados em bases de dados passou a ser objeto de estudo de diversos pesquisadores, com o objetivo de melhorar a qualidade das informações armazenadas que posteriormente são utilizadas para tomada de decisões e definições de regras de negócio. Para garantir uma melhora na qualidade dos dados armazenados, um dos processos que vem ao encontro a esta finalidade é a detecção de registros duplicados na base de dados, a fim de garantir uma melhor integridade e confiabilidade dos dados que serão utilizados na aquisição de informações (CHRISTEN, 2009).

Neste capítulo é apresentada a proposta de trabalho para a identificação de tuplas duplicadas por meio de técnica baseada em distância, a fim de enfatizar o desenvolvimento de um algoritmo de identificação de similaridade de *string* baseada na fonética. Também é abordada a identificação de registros duplicados por meio da similaridade numérica, bem como conceitos relevantes para o desenvolvimento do trabalho. Para finalizar, é apresentado o ambiente independente de idioma desenvolvido para dar suporte a identificação de registros duplicados.

3.2. Definição do problema

Os algoritmos apresentados neste trabalho, como *Soundex*, *NYSIIS* e *Metaphone*, foram desenvolvidos mediante as técnicas baseadas em distância, que são capazes de operar sem a necessidade de dados treinados para a identificação de tuplas duplicadas. Detalhadamente, esses algoritmos utilizam método de transformação de caracteres de uma *string* de entrada em códigos, com exceção do algoritmo *Edit Distance*, para posteriormente efetuar uma comparação com registros armazenados nos bancos de dados. Resumidamente, esses algoritmos trabalham baseados em métricas de similaridade, onde o caracter de uma *string* é substituído por um código de determinado algoritmo empregado, formando uma seqüência de códigos representativos de uma ou várias palavras conjuntamente. Apenas o algoritmo *Edit Distance* não trabalha com codificação, pois realiza a detecção de similares por meio de operações de cálculo de distância.

Mediante os estudos sobre os algoritmos mencionados, identifica-se à necessidade de se trabalhar com registros alfanuméricos, abrangendo letras e números de um alfabeto, como forma de melhorar a identificação de similaridade entre palavras por meio da técnica baseada em distância. Os algoritmos estudados que trabalham com codificação de caracteres como *Soundex*, *NYSIIS* e *Metaphone* utilizam métodos próprios de transformação de *string* em códigos, onde cada um obedece as suas particularidades. Diante disso, verifica-se que estes algoritmos trabalham apenas com a codificação de letras, e ignora aqueles que possuem como parte de seu registro caracteres numéricos.

Para exemplificar, quando se busca codificar por meio do Algoritmo *Soundex* as palavras “7 de Maio”, “8 de Maio”, “9 de Maio”, e assim sucessivamente, a codificação originada é “d500”. Observa-se que este algoritmo de codificação despreza completamente o caracter numérico, e parte diretamente para o próximo caracter da frase. O mesmo acontece com o Algoritmo *Metaphone*, que ignora a ocorrência de caracteres numéricos. Então, considere o seguinte caso: uma cidade que tem como característica o nome de suas ruas com caracteres numéricos para identificação, como “rua 1, rua 2, rua 3”, e assim sucessivamente. Se for realizada uma busca pelo endereço “rua 1” a fim de identificar registros duplicados no banco de dados, não serão expostos resultados exatos, e sim todas as tuplas que contém como conteúdo a palavra codificada “rua”. Caso o banco de dados tenha 1000 registros com esta palavra, certamente serão apresentados todos como resultados, ou seja, os 1000 registros, o que na verdade não se tratam de registros similares, pois são dados diferentes.

É importante destacar que isto não acontece somente com *Soundex*, mas com as suas extensões e variações, como os mencionados *Daitch-Mokotoff Soundex*, *NYSIIS*, *Metaphone*, entre outros. Por meio desse estudo, verifica-se que as palavras que contém em seus registros números não são codificadas na sua totalidade, onde é retornado muitas vezes resultados imprecisos na consulta diretamente no banco.

Diante desse exposto, verifica-se a necessidade de trabalhar com a codificação também dos caracteres numéricos, abrangendo todas as letras e números de um alfabeto alfanumérico, visto que hoje nas grandes bases de dados é cada vez mais comum encontrar registros de atributos alfanuméricos, a citar nome de rua, Cadastro de Pessoa Física - CPF, número da residência, telefone e nomes de pessoa no Brasil, e seguro social nos Estados Unidos. Utilizando um algoritmo capaz de codificar todos esses caracteres, evidencia-se a facilidade de se identificar registros duplicados armazenados em uma base de dados, independente do domínio do atributo.

Portanto, fica evidente a necessidade de desenvolvimento de um algoritmo que aborde o tratamento numérico na limpeza de dados e que seja capaz de codificar caracteres alfanuméricos, buscando-se resultados mais precisos em uma base de dados.

Uma outra necessidade faz-se presente quando é considerada a influência na ortografia dos aspectos fonéticos. Diante disso, surge a idéia de incluir neste algoritmo a função de identificar dados duplicados por meio da similaridade fonética. A fonética será utilizada na identificação de registros duplicados numa base de dados textual partindo da idéia de que uma determinada palavra pode ser escrita de diferentes formas, podendo obedecer as características de pronúncia, dialetos regionalizados do país, ou a falta de conhecimento das regras ortográficas e gramáticas de um idioma (BRADLOW; CLOPPER; SMILJANIC; WALTER, 2010) (SILVA, 2000).

Portanto, a fonética será utilizada para transcrever de diversas formas possíveis uma determinada palavra, a fim de identificar as diversas maneiras que uma palavra seria escrita e conseqüentemente facilitar a busca na base de dados. Essas variações das letras do alfabeto só são possíveis identificar por meio das transcrições das letras do alfabeto em questão.

Após as análises realizadas, surge a proposta de desenvolvimento de um algoritmo que realiza buscas por meio da similaridade fonética e que aborde o tratamento numérico, a fim de identificar registros duplicados numa base de dados textual. Para isso, é necessária a descrição do que é a fonética e suas principais características.

3.3. Visão geral da proposta

A proposta de desenvolvimento do algoritmo que trabalha na identificação de registros duplicados por meio da similaridade fonética e numérica surge diante das limitações em encontrar registros duplicados na base de dados por meio dos algoritmos existentes e mencionados. Tais algoritmos trabalham por meio da técnica baseada em distância, sem definições de modelos de dados treinados, mas diretamente na transcrição de *strings*. Portanto, a idéia de desenvolvimento de um algoritmo que aborde tais características vem ao encontro a necessidade de trabalhar com a detecção de registros duplicados, mas com um foco diferente, baseado na identificação da similaridade numérica e fonética de diferentes idiomas e utilizando a funcionalidade *multithreading*, com o intuito de medir o tempo de processamento.

3.3.1. Fonética

A utilização da fonética neste trabalho parte do pressuposto que uma determinada palavra pode ser escrita de diferentes formas, obedecendo as características regionalizadas do país, as diferentes formas de pronuncia ou até mesmo a falta de conhecimento das regras ortográficas e gramáticas da língua materna. Por isso, ao fazer uma análise da escrita de uma palavra, dentre os principais fatores que podem influenciar nesta escrita é a região geográfica, faixa etária, gênero (masculino, feminino), estilo (formal, não-formal), grau de instrução e classe social (BRADLOW; CLOPPER; SMILJANIC; WALTER, 2010) (SILVA, 2000).

Diante dessa posição, a fonética pode ser definida como a ciência que utiliza métodos para a descrição, classificação e transcrição dos sons da fala, principalmente aqueles sons utilizados pela linguagem humana. Busca-se caracterizar uma palavra por meio do seu som individual, para que então seja possível estimar semelhanças entre diferentes palavras por meio das transcrições fonéticas (KONDRAK, 2001). Portanto, a fonética estuda os sons da fala produzidos pelo aparelho fonador, que é dividido por três grupos de órgãos do corpo humano, responsáveis pela produção da fala. São eles: o sistema respiratório, composto pelos pulmões, músculos pulmonares, brônquios e traquéia; o sistema fonatório, composto pela laringe; e por último o sistema articulatório, composto pela faringe, língua, nariz, palato, dentes e os lábios (HENRIQUES, 2007) (SILVA, 2000).

A fonética é basicamente descritiva, baseando-se no processo de percepção e de produção dos sons. Pode ser estudada sem levar em consideração a sintaxe e a semântica da língua (LOWE, 1996). No apêndice A deste documento tem-se as diversas definições de fonética difundida entre diferentes autores.

Atualmente, a fonética tem sido objeto de muitos estudos, o que caracteriza a sua importância para o reconhecimento de palavras, independente se for de forma escrita ou falada. Então, diante dessas duas formas, alguns trabalhos de diversas áreas, como a médica, têm trabalhado com a caracterização do som, aliando a semântica e a fonética para a identificação da fala e construção de documentos textos, como relatórios e projetos (LIST, 2010) (PETRIK; DREXEL; FESSLER; JANCSARY; KLEIN; KUBIN; MATIASEK; PERNKOPF; TROST, 2011).

Existem trabalhos que tratam exclusivamente a parte textual. Tais trabalhos utilizam como objetos de estudos diferentes idiomas e pretende atingir o mesmo objetivo de encontrar semelhanças fonéticas em diferentes palavras. Um dos trabalhos que realizam esta tarefa é a de Nakov, Paskaleva e Nakov (NAKOV; PASKALEVA; NAKOV, 2009), que trabalham com os idiomas búlgaro e russo. Mesmo se tratando de dois idiomas diferentes, os autores relatam que estes idiomas possuem características semelhantes. Dentre algumas características, cita-se o de possuir letras semelhantes, mas com significados diferentes, o que torna o objetivo do trabalho ainda mais custoso, pois busca também a identificação de cognatos, ou seja, palavras que possuem uma grafia igual ou muito semelhante foneticamente e que possuem os mesmos significados. Na sua forma de trabalhar, os autores relatam que primeiramente busca-se a similaridade ortográfica por meio da fonética entre duas palavras para que posteriormente seja possível estimar a medida de similaridade entre elas, usando algoritmos que trabalham com técnicas baseadas em distância, como o *Edit Distance* (NAKOV; PASKALEVA; NAKOV, 2009).

Um estudo de caso foi efetuado com a finalidade de abordar a identificação de registros duplicados por meio da similaridade fonética da língua portuguesa e italiana em uma base de dados textual. Portanto, para este trabalho, é utilizada a língua portuguesa e italiana como forma de exemplificar o ambiente desenvolvido para a limpeza de dados, embora este ambiente permita ao usuário escolher dentro de um leque de idiomas qual quer trabalhar, para que, posteriormente, ocorra a transcrição fonética desta palavra conforme o idioma escolhido.

A língua italiana é um objeto de estudo deste trabalho por ser uma língua européia que apresenta grande disponibilidade de materiais que auxiliam a elaboração das diversas

transcrições fonéticas, favorecendo a construção de um ambiente para a realização dos testes nas bases de dados. Além disso, somam-se as orientações dadas pelo co-orientador deste trabalho, Professor Maurizio Babini (BABINI, 2002), que contribuiu significativamente para esta pesquisa.

A idéia de realizar um estudo de caso com a fonética da língua portuguesa é inovadora, não sendo encontrado nenhum material científico que caracteriza a identificação de registros duplicados por meio da sua utilização fonética. No entanto, aspectos da grafia da palavra foram utilizadas. Na língua portuguesa, a grafia é representada por letras, diacríticos (cedilha, til, acentos) e sinais de pontuação. Deve-se considerar que no alfabeto português há 26 letras, e que não são suficientes para representar unitariamente os fonemas da língua portuguesa. Para melhor exemplificar, é importante destacar que das 26 letras do alfabeto português, 5 delas são vogais e 21 consoantes (HENRIQUES, 2007). Dependendo do caso, uma letra pode ter várias transcrições, como é o caso da consoante “c”, que pode ser transcrito com os sons das consoantes “c”, “k” e “s”. Adiante é transcrito em diferentes tabelas cada letra do alfabeto da língua portuguesa, conforme explicações das obras de Maurizio Babini (BABINI, 2002) e Thaïs Cristófaro Silva (SILVA, 2000).

Diante das várias transcrições fonéticas existentes e aperfeiçoadas no mundo inteiro desde o final do século XIX, a Associação Fonética Internacional - IPA criou o Alfabeto Fonético Internacional, conforme apresentado na Tabela 3.1, utilizado em dicionário bilíngüe e muitos livros de lingüística destinados a ensinar como se deve pronunciar cada palavra (HENRIQUES, 2007).

Tabela 3.1 Alfabeto Internacional de Fonética (revisado em 1993, atualizado em 1996) (Extraído de SILVA, 2000).

	Bilabial	Lábio-dental	Dental	Alveolar	Pós-alveolar	Retroflexa	Palatal	Velar	Uvular	Faringal	Glotal
Oclusiva	p b		t d			ʈ ɖ	c ɟ	k g	q ɢ		ʔ
Nasal	m	ɱ	n			ɳ	ɲ	ŋ	ɴ		
Vibrante	ʙ		r						ʀ		
Tepe			ɾ			ɽ					
Fricativa	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Fricativa lateral			ɬ ɮ								

Aproximante		ʋ	ɹ	ɻ	j	ɰ			
Aprox. lateral			l	ɭ	ʎ	ʟ			

Na Tabela 3.1 é apresentado o Alfabeto Internacional de Fonética, adotado como base para a confecção de tabelas (vogais, consoantes, dígrafos, caracteres especiais e trígrafos, se houver) dos mais variados idiomas que o ambiente de identificação deste trabalho possa operar. Como forma de exemplificar, as tabelas são utilizadas para a construção do algoritmo denominado Algoritmo *PSI* (Identificação de Similaridade Fonética - *Phonetic Similarity Identification*), e tem como principal característica às diferentes transcrições fonéticas e em diferentes idiomas, diferentemente da tabela do Alfabeto Internacional de Fonética que obedece a uma regra mundial de transcrição. É importante ressaltar que o algoritmo proposto não leva em consideração o sentido das palavras cognatas, e sim apenas a grafia dessas palavras (KONDRAK, 2001).

Dentre as várias tabelas existentes no algoritmo, nesta seção são apresentadas as tabelas construídas baseadas na língua portuguesa, conforme o estudo de caso. As tabelas criadas na língua italiana são apresentadas no Apêndice B e obedecem as mesmas características de entendimento e estruturação das tabelas aqui apresentadas. É importante a apresentação das tabelas da língua portuguesa nesta seção para que haja um melhor entendimento das suas estruturas e formação da linha de raciocínio para a confecção do algoritmo proposto. Tais tabelas foram elaboradas e são contribuições do autor deste trabalho a fim de demonstrar as transcrições fonéticas das línguas em estudo.

Para a apresentação do estudo de caso da língua portuguesa, foram criadas quatro tabelas, sendo elas divididas em vogais, consoantes, dígrafos e caracteres especiais. A Tabela 3.2 refere-se as vogais, que possui como primeira coluna à sua descrição, a segunda coluna as diferentes transcrições fonéticas para cada vogal mencionada na primeira coluna, e na terceira coluna palavras utilizadas para exemplificar as diferentes transcrições dessas vogais.

Tabela 3.2 Transcrição fonética das vogais – língua portuguesa.

VOGAIS		
Caractere	**Trans. Fonética**	**Exemplos**
A	/a/	*lá, batata*
	/ã/	*lã, lâmpada*
E	/e/	*vê, lê*
	/ɛ/	*pé, reto*

	/ẽ/	*bem*
	/i/	*parte, emprestar*
I	/i/	*apito*
O	/o/	*oba, avô*
	/ɔ/	*pó, pote*
	/õ/	*bom*
	/u/	*dedo, comida*
U	/u/	*leitura*

Nota-se que na Tabela 3.2 são apresentadas as 5 vogais do alfabeto da língua portuguesa, e suas transcrições fonéticas, totalizando 12 vogais fonéticas (HENRIQUES, 2007).

As Tabelas 3.3 a 3.22 referem-se as tabelas das consoantes da língua portuguesa, caracterizada pela divisões de várias colunas. Em todos os casos, a primeira coluna refere-se a própria consoante, a segunda coluna refere-se as diferentes transcrições fonéticas de cada consoante mencionada na primeira coluna, a terceira coluna refere-se as junções das consoantes mencionadas na primeira coluna com cada vogal existente na língua portuguesa. A quarta coluna refere-se aos diferentes sons foneticamente adquiridos com a junção da consoante mencionada na primeira coluna com as transcrições fonéticas de cada vogal existente na língua portuguesa, no qual foi mencionada na Tabela 3.2. A quinta coluna é de palavras que exemplificam os sons característicos obtidos na quarta coluna.

A Tabela 3.3 refere-se ao estudo da consoante “b”, onde esta apresenta o som apenas da própria letra segundo o alfabeto da língua portuguesa.

Tabela 3.3 Transcrição fonética da consoante B – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
B	/b/	B+A	[ba]	*bala*
			[bã]	*bamba*
		B+E	[be]	*benefício*
			[bɛ]	*bela*
			[bẽ]	*bem*
			[bi]	*bebe*
		B+I	[bi]	*biscoito*
		B+O	[bo]	*bolacha*
			[bɔ]	*bola*
			[bõ]	*bom*
			[bu]	*bonito*
		B+U	[bu]	*burro*

Ao analisar a Tabela 3.3, é verificado por meio dos exemplos dados na coluna cinco que algumas palavras podem ser transcritas de diferentes formas foneticamente. As palavras “bala”, “benefício”, “biscoito”, “bolacha” e “burro” são exemplos de que as sílabas “ba”, “be”, “bi”, “bo” e “bu” respectivamente podem não possuir alterações de transcrição fonética.

A palavra “bamba” é um exemplo que pode sofrer alteração de transcrição fonética, pois a sua primeira sílaba possui o som “bã”, o que poderia resultar foneticamente na escrita desta palavra como “bãba”. O mesmo pode ser dito para as palavras “bem” e “bom”, que foneticamente podem ser escritas como “bẽ” e “bõ”, respectivamente.

Outros exemplos podem ser vistos e detalhados, como a palavra “bebe” que foneticamente a sílaba final “be” pode ser transcrita como “bi”, resultando numa palavra pronunciada e escrita “bebi”. A palavra “bonito” pode ser escrita como “bunito”, pois foneticamente a sílaba “bo” pode ser transcrita como “bu”, relatando assim as características de pronunciação conforme mencionada anteriormente de um determinada palavra de acordo com a região do país. A palavra “bela” possui a primeira sílaba “be” que pode ser pronunciada com se tivesse um acento agudo na vogal “e”, o que caracteriza foneticamente sendo a letra “ɛ”, podendo ser escrita também como “béla”. A palavra “bola” seria o mesmo caso da vogal “e”, o que caracteriza a primeira sílaba da palavra como se tivesse o acento agudo na vogal “o” quando pronunciada, caracterizando-o foneticamente com um caractere especial “ɔ”, onde esta palavra pode ser escrita como “bóla”.

Portanto, estes exemplos citados são apenas para mostrar que uma determinada palavra pode ser escrita de diferentes formas conforme antes explicado, o que reforça ainda mais a importância desse estudo para a identificação de registros duplicados em uma base de dados. Vale ressaltar que foi mencionada nestes exemplos anteriores apenas a transcrição da consoante “b” em junção com as vogais do alfabeto. Muitas outras maneiras poderiam ser transcritas algumas palavras utilizadas nestes exemplos, mas que fique evidente que neste caso só foi considerado até então as transcrições da consoante “b”. Os mesmos casos de transcrições são utilizados para as demais consoantes do alfabeto da língua portuguesa, conforme seguem as tabelas abaixo.

A Tabela 3.4 refere-se ao estudo da consoante “c”, onde são apresentadas todas as possíveis transcrições fonéticas desta letra, podendo ser “/c/”, “/k/” ou “/s/”, dependendo do caso.

Tabela 3.4 Transcrição fonética da consoante C – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
C	/c/	C+A	[ka]	*casa*
			[kã]	*canta*
		C+E	[se]	*cebola*
			[sɛ]	*cego*
			[sẽ]	*centrado*
			[si]	*anoitece*
		C+I	[si]	*cidade*
		C+O	[ko]	*coruja*
			[kɔ]	*cola*
			[kõ]	*com*
			[ku]	*comida*
		C+U	[ku]	*curau*
	/k/	C+A	[ka]	*casa*
			[kã]	*canta*
		C+O	[ko]	*coruja*
			[kɔ]	*cola*
			[kõ]	*com*
			[ku]	*comida*
		C+U	[ku]	*curau*
	/s/	C+E	[se]	*cebola*
			[sɛ]	*cego*
			[sẽ]	*centrado*
			[si]	*anoitece*
		C+I	[si]	*cidade*

É importante observar que, embora as transcrições fonéticas podem ser “/c/”, “/k/” ou “/s/”, seus sons hora são com as letras “k+vogais” ou “s+vogais”. Necessariamente, os sons de “k+vogais” (“ka”, “ko”, “ku”) e seus derivados (entenda-se “kã”, “kɔ”, “kõ” e “ku”), só podem ter as transcrições fonéticas para as letras “c” e “k”, casos estes de palavras que contenham “c+a” (exemplo casa), “c+o” (exemplo coruja) e “c+u” (curau).

Para ter os sons de “s+vogais” (“se” e “si”) e seus derivados (entenda-se “sɛ”, “sẽ” e “si”), só podem ser transcritos foneticamente para as letras “c” e “s”, casos estes para as palavras que contenham “c+e” (exemplo cebola) e “c+i” (cidade). As demais combinações de palavras obedecem às mesmas regras de exemplares mencionadas para a consoante “b”, a fim de formar possíveis combinações e as explicações devidas.

A Tabela 3.5 refere-se ao estudo da consoante “d”, onde esta possui apenas a sua própria transcrição fonética, conforme a língua portuguesa. As possíveis variações de transcrição fonética acontecem como exemplificado na consoante “b”.

Tabela 3.5 Transcrição fonética da consoante D – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
D	/d/	D+A	[da]	*dado*
			[dã]	*dando*
		D+E	[de]	*dedo*
			[dɛ]	*deve*
			[dẽ]	dente
			[di]	pretende
		D+I	[di]	disse
		D+O	[do]	*dores*
			[dɔ]	*dorme*
			[dõ]	*dom*
			[du]	*doblar*
		D+U	[du]	*duvida*

A Tabela 3.6 refere-se ao estudo da consoante “f”, onde esta possui apenas a sua própria transcrição fonética, conforme a língua portuguesa. Também as possíveis variações de transcrição fonética acontecem como exemplificado na consoante “b”.

Tabela 3.6 Transcrição fonética da consoante F – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
F	/f/	F+A	[fa]	*faca*
			[fã]	*fantasia*
		F+E	[fe]	*feira*
			[fɛ]	*fera*
			[fẽ]	*fenda*
		F+I	[fi]	*figo*
		F+O	[fo]	*fogo*
			[fɔ]	*foca*
			[fõ]	*fonte*
			[fu]	*focinho*
		F+U	[fu]	*furo*

A Tabela 3.7 refere-se ao estudo da consoante “g”, com as possíveis transcrições fonéticas como “/g/”, “/j/” e “/gu/”.

Tabela 3.7 Transcrição fonética da consoante G – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
G	/g/	G+A	[ga]	*gato*
			[gã]	*gancho*
		G+E	[ʒe]	*gelado*
			[ʒɛ]	*gera*
			[ʒẽ]	*gente*
			[ʒi]	*laringe*
		G+I	[ʒi]	gibi
		G+O	[go]	*goma*
			[gɔ]	*gola*
			[gõ]	*gôndola*
		G+U	[gu]	*gula*
	/j/	G+E	[je]	*geada*
		G+I	[ji]	*girafa*
	/gu/	GU+A	[gwa]	*água*
		GU+E	[gwe]	*guerrilha*
		GU+I	[gwi]	*guitarra*
		GU+O	[gwo]	*águo*

Os exemplos da Tabela 3.7 podem ser vistos e lidos como os exemplos da Tabela 3.4, que trata das transcrições fonéticas da consoante “c”. Neste caso, as possíveis transcrições fonéticas podem ter sons de “g+vogais” (“ga”, “go”, “gu”) e seus derivados (“gã”, “gɔ”, “gõ” ou “gu”), “j+vogais” (“je” e “ji”) e seus derivados (“jɛ”, “jẽ” e “ji”), e por último “gu+vogais” (“gua”, “gue”, “gui”, “guo”).

A Tabela 3.8 refere-se ao estudo da consoante “j”, com as possíveis transcrições fonéticas como “/j/” e “/g/”.

Tabela 3.8 Tabela de transcrição fonética da consoante J – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
J	/j/	J+A	[já]	*jarra*
			[já]	*janta*
		J+E	[je]	*jeito*
			[jɛ]	*jegue*
			[jẽ]	*jendiroba*
			[ji]	*alfanje*
		J+I	[ji]	*jipe*
		J+O	[jô]	*jogo*
			[jɔ]	*jovem*

			[jõ]	*jongo*
		J+U	[ju]	*juros*
	/g/	J+E	[ʒe]	*jeito*
		J+I	[ʒi]	*jiló*

Os exemplos da Tabela 3.8 que apresentam as transcrições fonéticas da consoante "j" para o "g" podem ser vistos como os exemplos da Tabela 3.7 que trata das transcrições fonéticas do "g" para a consoante "j". Uma característica desta tabela é que a consoante "j" terá seu próprio som nos casos "ja", "jo" e "ju" e seus derivados (jã, jɛ, jẽ e ji), como nas palavras "jarra", "jogo" e "juros". Para ter os sons da consoante "g", a consoante "j" só pode ser acompanhada das vogais "e" e "i", formando as sílabas "ge" e "gi", como nas palavras "jeito" e "jiló". Isto abre a possibilidade destas duas últimas palavras exemplificadas serem escritas foneticamente com a consoante "g" ou "j", pois possuem sons parecidos.

A Tabela 3.9 refere-se ao estudo da consoante "k", com as possíveis transcrições fonéticas como "/k/" e "/c/".

Tabela 3.9 Transcrição fonética da consoante K – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
K	/k/	K+A	[ka]	*kart*
			[kã]	*kanga*
		K+E	[ke]	*kendô*
			[kɛ]	*
			[kẽ]	*
			[ki]	*
		K+I	[ki]	*kilo*
		K+O	[ko]	*kochia*
			[kɔ]	*
			[kõ]	*
	/c/	C+A	[ka]	*kart*
			[ka]	*kanga*
		C+O	[ko]	*kochia*

Ao analisar a Tabela 3.9, é possível verificar que a consoante "k" só pode ser transcrita foneticamente como "k" nos casos que estiver acompanhada das vogais "a", "e", "i" ou "o", formando as sílabas "ka", "ke", "ki", "ko" e seus derivados, respectivamente, exemplificadas por meio das palavras "Kart", "Kendô", "Kilo" e "Kochia".

Também é possível transcrever foneticamente a consoante “k” em “c”, para os casos que a consoante “k” estiver acompanhada das vogais “a” e “o”, obtendo-se os sons “ka” e seu derivado (“kã”), e “ko”, exemplificadas por meio das palavras “Kart” e “Kochia”. A utilização do asterisco (*) em alguns casos na coluna de exemplos é devido a não existência de palavras na língua portuguesa que pudessem ser empregadas para a exemplificação dos sons em questão, mas que é importante a sua citação, visto que podem ser encontradas em nomes próprios.

A Tabela 3.10 refere-se ao estudo da consoante “l”, com as possíveis transcrições fonéticas como “/l/” e a vogal “/u/”.

Tabela 3.10 Transcrição fonética da consoante L – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
L	/l/	L+A	[la]	*lápis*
			[lã]	*lâmpada*
		L+E	[lê]	*leitura*
			[lɛ]	*leve*
			[lẽ]	*lente*
			[li]	*pele*
		L+I	[li]	*liso*
		L+O	[lo]	*louco*
			[lõ]	*longo*
			[lu]	*calo*
		L+U	[lu]	*lucro*
	/u/	A+L	[au]	*cal*
		E+L	[eu]	*abalável*
			[ɛu]	*papel*
		I+L	[iu]	*refil*
		O+L	[ou]	*gol*
			[ɔu]	*girassol*

Nota-se que na Tabela 3.10 é possível ver as duas transcrições fonéticas da consoante “l”, sendo uma para ela mesma e a outra para a vogal “u”. No caso da vogal, só é possível a transcrição fonética para “u” se a palavra tiver a letra “l” no final e precedida de uma vogal ou precedida de uma vogal e sucedida de uma consoante.

A transcrição fonética da consoante “l” para ela mesma obedece à regra de estar sucedida de uma vogal (“l+vogal”), podendo ser “l+a”, “l+e”, “l+i”, “l+o” e “l+u”,

formando as sílabas “la”, “le”, “li”, “lo”, “lu” e suas derivações, dando origem a algumas palavras que podem ser vistas nos exemplos.

A Tabela 3.11 refere-se ao estudo da consoante “m”, possuindo apenas a sua própria transcrição fonética. Exemplos de palavras formadas da junção da consoante com as vogais do alfabeto da língua portuguesa e as características de sons podem ser vistos a seguir.

Tabela 3.11 Transcrição fonética da consoante M – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
M	/m/	M+A	[ma]	*macaco*
			[mã]	*mancha*
		M+E	[me]	*melancia*
			[mɛ]	*mela*
			[mẽ]	*mente*
			[mi]	*enorme*
		M+I	[mi]	*milho*
		M+O	[mo]	*molho*
			[mɔ]	*amostra*
			[mõ]	*monge*
			[mu]	*moela*
		M+U	[mu]	*muro*

A Tabela 3.12 refere-se ao estudo da consoante “n” que, igualmente a Tabela 3.11 que trata da consoante “m”, possui apenas a sua própria transcrição fonética. Detalhes podem ser vistos na tabela a seguir.

Tabela 3.12 Transcrição fonética da consoante N – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
N	/n/	N+A	[na]	*navio*
			[na]	*alternante*
		N+E	[ne]	*neste*
			[nɛ]	*neve*
			[nẽ]	*pertinente*
			[ni]	*carne*
		N+I	[ni]	*nível*
		N+O	[no]	*novo*
			[nɔ]	*nosso*
			[nu]	*abandono*
		N+U	[nu]	*nuvem*

A Tabela 3.13 refere-se ao estudo da consoante “p”. Esta tabela mostra que esta consoante possui apenas uma transcrição fonética, ou seja, para o próprio “p”.

Tabela 3.13 Transcrição fonética da consoante P – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
P	/p/	P+A	[pa]	*pato*
			[pã]	*pântano*
		P+E	[pe]	*pedaço*
			[pɛ]	*espera*
			[pẽ]	*penta*
			[pi]	*jipe*
		P+I	[pi]	*pires*
		P+O	[po]	*posto*
			[pɔ]	*poste*
			[põ]	*ponte*
			[pu]	*campo*
		P+U	[pu]	*puro*

A Tabela 3.14 refere-se ao estudo da consoante “q”, com suas respectivas transcrições, sendo “/q/”, “/k/” e “/c/”.

Tabela 3.14 Transcrição fonética da consoante Q – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
Q	/q/	QU+A	[kwa]	*quase*
			[kwã]	*quanta*
		QU+E	[ke]	*queijo*
			[kɛ]	*queda*
			[kẽ]	*quente*
		QU+I	[ki]	*quiabo*
		QU+O	[kwo]	*quotidiano*
			[kwɔ]	*quota*
	/k/	QU+A	[kwa]	*quase*
			[kwã]	*quanta*
		QU+E	[ke]	*queijo*
			[kɛ]	*queda*
			[kẽ]	*quente*
		QU+I	[ki]	*quiabo*
		QU+O	[kwo]	*quotidiano*
			[kwɔ]	*quota*

	/c/	QU+A	[kwa]	*quase*
			[kwã]	*quanta*
		QU+O	[kwo]	*quotidiano*
			[kwɔ]	*quota*

É importante observar que as transcrições fonéticas da consoante “q” apresentadas na Tabela 3.14 podem ser “/q/”, “/k/” ou “/c/”. Em todos os casos a consoante “q” será acompanha da vogal “u”, formando a sílaba “qu”, além das demais vogais do alfabeto.

Na transcrição fonética da consoante “q” para o próprio “q”, os sons originados com a presença das vogais “a” e “o” são realçados com a presença da vogal “u” intermediária, e o próprio “q” será transcrito com “k”, pronunciando “k+u+a” (lê-se kua) e seu derivado (kuã), e “k+u+o” (lê-se kuo) e seu derivado (kuɔ), formando as palavras exemplificadas “quase” e “quotidiano”. Embora haja a presença da vogal “u”, esta é considerada uma semi-consoante para a pronunciação, por isso é representada pela letra “w” ao se tratar de som, pois em uma sílaba é possível a pronúncia de apenas uma vogal tônica (BABINI, 2002). Já os sons originados com a presença das vogais “e” e “i”, a vogal “u” não será pronunciada, mesmo esta vogal fazendo parte da formação da sílaba. Diante disso, as sílabas formadas com “qu+e” e “qu+i” obtem sons de “ke”, “ki” e seus derivados (kɛ, kẽ), formando respectivamente palavras exemplificadas como “queijo” e “quiabo”, onde se observa que a vogal “u” faz parte da palavra, mas não é pronunciada.

A transcrição fonética da consoante “q” para “k” obedece as mesmas regras e explicações da transcrição para a consoante “q”. Já a transcrição fonética para a consoante “c” só é possível com a presença da vogal intermediária “u”, dando origem aos sons “k+u+a”, “k+u+o” e seus derivados, formando as sílabas “qua” e “quo”, exemplificadas nas palavras “quase” e “quotidiano”, sendo a vogal “u” considerada uma semi-consoante “w” na caracterização do som. Portanto, isto mostra que, com estas duas palavras exemplificadas tanto nas transcrições fonéticas para a consoante “q” quanto para “c”, é possível escrever foneticamente estas palavras com as letras “q” ou “c”, ambas sucedidas da vogal “u” com caracterização de som evidenciado pela semi-consoante “w”, pois em uma sílaba é possível a pronúncia de apenas uma vogal tônica (BABINI, 2002).

A Tabela 3.15 refere-se ao estudo da consoante “r”, onde é verificado que esta consoante só possui a sua própria transcrição fonética. Obedece à mesma característica da consoante “b”.

Tabela 3.15 Transcrição fonética da consoante R – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
R	/r/	R+A	[ra]	*rato*
			[rã]	*rancho*
		R+E	[re]	*repolho*
			[rɛ]	*resto*
			[rẽ]	*rente*
			[ri]	*ampere*
		R+I	[ri]	*riso*
		R+O	[ro]	*rosto*
			[rɔ]	*roça*
			[rõ]	*rompimento*
			[ru]	*abatedouro*
		R+U	[ru]	*rumo*

A Tabela 3.16 refere-se ao estudo da consoante “s”, onde esta possui as seguintes transcrições fonéticas: “/s/”, “/c/”, “/sc/” e “/z/”.

Tabela 3.16 Transcrição fonética da consoante S – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
S	/s/	S+A	[sa]	*sapo*
			[sã]	*santa*
		S+E	[se]	*sebo*
			[sɛ]	*sela*
			[sẽ]	*senta*
			[si]	*fase*
		S+I	[si]	*sismo*
		S+O	[so]	*sobre*
			[sɔ]	*sol*
			[sõ]	*sombra*
			[su]	*soado*
		S+U	[su]	*sul*
	/c/	S+E	[se]	*separado*
		S+I	[si]	*sismo*
	/sc/	Vogal+SC+E	[se]	*nasce*
		Vogal+SC+I	[si]	*nascido*
	/z/	S+A	[za]	*casa*
			[zã]	*causante*
		S+E	[ze]	*braseiro*

			[zɛ]	*miséria*
			[zẽ]	*ausente*
			[zi]	*básico*
		S+I	[zi]	*visita*
		S+O	[zo]	*liso*
			[zɔ]	*casório*
			[zu]	*vaso*
		S+U	[zu]	*casual*

A transcrição fonética da consoante “s” para a consoante “s” forma sílabas compostas pelas letras “s+vogais”, com os sons “sa”, “se”, “si”, “so”, “su” e seus derivados (“sɛ”, “sẽ”, “sɔ”, “sõ”), onde é formada as palavras exemplificadas “sapo”, “sebo”, “sismo”, “sobre” e “sul”.

A transcrição fonética da consoante “s” para a consoante “c” só é possível para os casos de junção das letras “s+e” e “s+i”, que formará o som “se” e “si”, conseqüentemente originando palavras exemplificadas como “separado” e “nascido”.

Para as transcrições fonéticas da consoante “s” para “sc” ser possível, faz-se necessário que estas letras “sc” sejam precedidas de vogal e sucedidas com vogal “e” ou “i”, formando sons como “se” ou “si”, originando palavras exemplificadas como “nasce” e “nascido”.

A transcrição fonética da consoante “s” para a consoante “z” é semelhante à transcrição para a consoante “s” explicada, formando sílabas compostas pelas letras “s+vogais”, originando sons como “za”, “ze”, “zi”, “zo”, “zu” e seus derivados (“zɛ”, “zẽ”, “zɔ”), exemplificadas pelas palavras “casa”, “braseiro”, “visita”, “liso” e “casual”.

A Tabela 3.17 refere-se ao estudo da consoante “t”, que possui a sua própria transcrição fonética. Estudo semelhante a consoante “b” foi realizado, e o entendimento da tabela segue o mesmo raciocínio.

Tabela 3.17 Transcrição fonética da consoante T – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
T	/t/	T+A	[ta]	*tatu*
			[tã]	*tanto*
		T+E	[te]	*texto*
			[tɛ]	*teste*
			[tẽ]	*tente*

			[ti]	*teste*
		T+I	[ti]	*tiro*
		T+O	[to]	*touro*
			[tɔ]	*tora*
			[tõ]	*tombamento*
			[tu]	*conto*
		T+U	[tu]	*turma*

A Tabela 3.18 refere-se ao estudo da consoante “v”, que possui também uma única transcrição fonética, para ela mesma.

Tabela 3.18 Transcrição fonética da consoante V – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
V	/v/	V+A	[va]	*valor*
			[vã]	*vandalismo*
		V+E	[ve]	*velocidade*
			[vɛ]	*vela*
			[vẽ]	*vento*
			[vi]	*deve*
		V+I	[vi]	*vinho*
		V+O	[vo]	*você*
			[vɔ]	*voz*
			[võ]	*vontade*
			[vu]	*objetivo*
		V+U	[vu]	*vulto*

A Tabela 3.19 refere-se ao estudo da consoante “w”, onde esta possui a transcrição fonética para “/w/” e “/v/”.

Tabela 3.19 Transcrição fonética da consoante W – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
W	/w/	W+A	[wa]	*
			[wã]	*
		W+E	[we]	*
			[wɛ]	*
			[wẽ]	*
			[wi]	*
		W+I	[wi]	*
		W+O	[wo]	*

			[wɔ]	*
			[wõ]	*
			[wu]	*
		W+U	[wu]	*
	/v/	V+A	[wa]	*
			[wã]	*
		V+E	[we]	*
			[wɛ]	*
			[wẽ]	*
			[wi]	*
		V+I	[wi]	*
		V+O	[wo]	*
			[wɔ]	*
			[wõ]	*
			[wu]	*
		V+U	[wu]	*

As formações de sílabas que caracterizam as transcrições fonéticas são possíveis se somadas as vogais do alfabeto. Neste caso, a junção da consoante “w+vogais” formam sílabas que dão origens a sons que caracterizam a pronunciação das sílabas, como “wa”, “we”, “wi”, “wo”, “wu” e seus derivados (“wã”, “wɛ”, “wẽ”, “wɔ” e “wõ”). É possível verificar que a consoante “w” também pode ser transcrito foneticamente como “v”, pois várias palavras que possuem a consoante “w” da língua portuguesa podem ser pronunciadas com o som característico da consoante “v”, e que as características sonoras são as mesmas da consoante “w” mencionada na tabela.

Nota-se que na coluna de exemplos possui apenas asterisco (*), pois não foram encontrados no dicionário da língua portuguesa exemplos de palavras que utilizam tal letra, também observado na consoante “y”. Vale lembrar que o fato de não ter sido encontrada palavras de exemplo com estas letras não descarta a necessidade de suas transcrições fonéticas, visto que muitos nomes próprios ainda utilizam-as com freqüência.

A Tabela 3.20 refere-se ao estudo da consoante “x”, onde esta possui a transcrição fonética para “/x/”, “/ks/”, “/s/” e “/z/”.

Tabela 3.20 Transcrição fonética da consoante X – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
X	/x/	X+A	[xa]	*xadrez*
			[xã]	*xantato*

		X+E	[xe]	*xereta*
			[xẽ]	*xendengue*
			[xi]	*praxe*
		X+I	[xi]	*xícara*
		X+O	[xo]	*xodó*
			[xɔ]	*xote*
			[xu]	*abaixo*
		X+U	[xu]	*xucro*
	/ks/	X+A	[ksa]	*fixar*
		X+I	[ksi]	*táxi*
		X+O	[kso]	*fixo*
			[ksu]	*nexo*
	/s/	E+X+P	[esp]	*explorar*
		E+X+T	[est]	*extremo*
		E+X+C	[esse]	*exceção*
	/z/	X+A	[za]	*exagerado*
			[zã]	*exame*
		X+E	[zé]	*executar*
		X+I	[zi]	*exibir*
		X+O	[zo]	*exonerar*
		X+U	[zu]	*exuberante*

Nota-se que as transcrições fonéticas da consoante “x” são variadas. Esta consoante tem sua transcrição para o próprio “x” quando somadas a uma vogal, como “x+a”, “x+e”, “x+i”, “x+o” e “x+u”, dando origens aos sons “xa”, “xe”, “xi”, “xo”, “xu” e derivados, e originando palavras exemplificadas como “xadrez”, “xereta”, “xícara”, “xodó” e “xucro”.

A consoante “x” também pode ser transcrita foneticamente como “ks”, para os casos de formação de sílabas do “x+a”, “x+i” e “x+o”, dando origem aos sons “ksa” (lê-se “quissa”), “ksi” (lê-se “quissi”) e “kso” (esta podendo ter sua leitura variada entre “quisso” ou “quissu”) respectivamente, formando palavras exemplificadas como “fixar”, “táxi” e “fixo”.

A consoante “x” terá sua transcrição fonética para “s” nos casos em que a sílaba conter o “x” precedido da vogal “e” e sucedido de uma das consoantes “p”, “t” ou “c”, formando respectivamente os sons “esp”, “est” ou “esse”. Palavras exemplificadas para estes casos são “explorar”, “extremo” e “exceção”.

Por último, a consoante “x” também pode ser transcrita foneticamente como “z”, formando as sílabas “za”, “ze”, “zi”, “zo” e “zu”, mas com os sons de “za” (derivado zã),

“ze”, “zi”, “zo” e “zu”, originando palavras exemplificas como “exagerado”, “executar”, “exibir”, “exonerar” e “exuberante”.

A Tabela 3.21 refere-se ao estudo da consoante “y”, onde esta possui a transcrição fonética para “/i/”.

Tabela 3.21 Transcrição fonética da consoante Y – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
Y	/i/	Y	[i]	*

Assim como na tabela de transcrição fonética da consoante “w”, nota-se que na coluna de exemplos possui apenas asterisco (*), pois não foram encontrados no dicionário da língua portuguesa exemplos de palavras que utilizam tal letra, o que não descarta a necessidade de sua transcrição fonética, visto que muitos nomes próprios ainda utilizam-as com freqüência.

A Tabela 3.22 refere-se ao estudo da consoante “z”, onde esta possui a transcrição fonética para “/z/” e “/s/”.

Tabela 3.22 Transcrição fonética da consoante Z – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
Z	/z/	Z+A	[za]	*zagueiro*
			[zã]	*zangado*
		Z+E	[zé]	*zelo*
			[zɛ]	*zero*
			[zi]	*bronze*
		Z+I	[zi]	*ziar*
		Z+O	[zo]	*zombado*
			[zõ]	*zonzo*
		Z+U	[zu]	*zunir*
	/s/	A+Z	[as]	*paz*
		E+Z	[es]	*insensatez*
			[ɛs]	*dez*
		I+Z	[is]	*aprendiz*
		O+Z	[ɔs]	*veloz*
		U+Z	[us]	*luz*

A consoante “z” tem sua transcrição fonética para o próprio “z” quando somadas a uma vogal, ou seja, “z+vogal” (“z+a”, “z+e”, “z+i”, “z+o”, “z+u”), consequentemente

origina os sons “za”, “ze”, “zi”, “zo”, “zu” e derivados (“zã”, “zɛ”, “zõ”), e forma palavras como “zagueiro”, “zelo”, “ziar”, “zombado” e “zunir”.

A transcrição fonética da consoante “z” pode ser para “s”, quando esta vir precedida de uma vogal e nos finais da palavra, formando sílabas contendo “az”, “ez”, “iz”, “oz” e “uz”. Estas sílabas têm as respectivas sonoridades: “as”, “es” e derivadas, “is”, “os” e “us”, formando palavras exemplificadas como “paz”, “insensatez”, “aprendiz”, “veloz” e “luz”.

A consoante “h” (lê-se “aga”) não foi transcrita foneticamente, pois não há sonoridade para ela. Diante disso, as ocorrências da letra “h” no inicio de palavras serão ignoradas no algoritmo proposto, visto que o “h” foneticamente não tem som expressivo de forma a alterar a pronúncia de uma palavra.

As transcrições de “ss” e “rr” não foram consideradas, visto que um dos passos do algoritmo proposto elimina os caracteres repetidos seguidamente.

A Tabela 3.23 refere-se aos dígrafos. Nesta tabela, a primeira coluna refere-se aos caracteres, a segunda coluna refere-se as transcrições fonéticas dos caracteres da primeira coluna, a terceira coluna as junções dos dígrafos com vogais, a quarta coluna aos sons das junções do dígrafos com as vogais e a quinta coluna aos exemplos dos sons.

Tabela 3.23 Transcrição fonética dos dígrafos – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
CH	/ʃ/	CH+A	[ʃa]	*chave*
			[ʃã]	*chance*
		CH+E	[ʃe]	*cheiro*
			[ʃɛ]	*cheque*
			[ʃẽ]	*enchente*
			[ʃi]	*avalanche*
		CH+I	[ʃi]	*chimarrão*
		CH+O	[ʃo]	*chorar*
			[ʃɔ]	*chove*
		CH+U	[ʃu]	*chuva*
LH	/ʎ/	LH+A	[ʎa]	*palha*
			[ʎã]	*calhambeque*
		LH+E	[ʎe]	*palheiro*
			[ʎɛ]	*mulher*
			[ʎẽ]	*piolhenta*
			[ʎi]	*acolhe*
		LH+I	[ʎi]	*velhice*

		LH+O	[ʎo]	*milho*
			[ʎɔ]	*melhor*
		LH+U	[ʎu]	*orelhudo*
NH	/ɲ/	NH+A	[ɲa]	*manha*
			[ɲã]	*amanhã*
		NH+E	[ɲe]	*pinheiro*
			[ɲɛ]	*munheca*
			[ɲi]	*apanhe*
		NH+O	[ɲo]	*banho*
			[ɲɔ]	*minhoca*
		NH+U	[ɲu]	*nenhuma*

As transcrições fonéticas dos dígrafos “ch”, “lh” e “nh” obedecem às mesmas características das consoantes antes estudadas. Para todos os casos, verifica-se a formação de sílabas com a junção dos dígrafos e vogais, e caracterização de sons com ou sem derivados, além das exemplificações por meio das palavras.

A Tabela 3.24 refere-se ao estudo da transcrição fonética do caracter especial “ç”, pois trata-se de um caractere comum na formação de palavras. Este caractere tem sua transcrição fonética para a consoante “s”, exclusivamente nas junções da consoante “ç” com as vogais “a”, “o” e “u”. Nestes casos, os sons característicos serão respectivamente “sa” e derivado (“sã”), “so” e derivado (“su”), e “su”, onde forma-se palavras exemplificadas como “calçado”, “almoço” e “açucar”. Por não existir ocorrências de palavras com as sílabas “çe” e “çi” no estudo de caso da língua portuguesa, não serão relatas nesta tabela.

Tabela 3.24 Transcrição fonética do caracter especial Ç – língua portuguesa.

Caractere	Trans. Fonética	Letras	Sons	Exemplos
Ç	/s/	Ç+A	[as]	*calçado*
			[sã]	*almoçando*
		Ç+O	[so]	*almoço*
			[su]	*moço*
		Ç+U	[su]	*açucar*

Diante das explicações das transcrições fonéticas das vogais, consoantes, dígrafos e caractere especial, foi desenvolvido um algoritmo que trabalhe com a transcrição fonética de uma determinada palavra ou um conjunto delas, conforme idioma escolhido dentro de um leque de possibilidades. Portanto, por meio das tabelas confeccionadas e

exemplificadas no estudo de caso da língua portuguesa, será descrito a seguir a funcionalidade do algoritmo, bem como a sua metodologia de funcionamento e resultados de alguns testes realizados.

3.4. Algoritmo *PSI*

O algoritmo proposto foi desenvolvido mediante a técnica baseada em distância, capaz de operar sem a necessidade de dados treinados para a identificação de tuplas duplicadas.

Denominado como Algoritmo *PSI* (Identificação de Similaridade Fonética - *Phonetic Similarity Identification*) tem como principal característica a identificação de registros duplicados independente do domínio do banco de dados, abrangendo os caracteres alfanuméricos, como letras e números do alfabeto, e de trabalhar com variações fonéticas dos caracteres da palavra.

Como primeira observação do funcionamento do algoritmo, os caracteres numéricos não são ignorados, permanecendo-o da forma que foi registrado no banco de dados ou como valor de entrada. Portanto, caso seja dado como valor de entrada uma palavra contendo número, o número permanecerá como está, sendo considerado como um caractere integrante do registro. Para exemplificar, se o usuário entrar com o valor “2 de maio”, o caractere “2” permanecerá inalterado no termo de busca e também nos registros do banco de dados. Conseqüentemente, próximas etapas que caracterizam o funcionamento do algoritmo são adotadas para os caracteres restantes.

O Algoritmo *PSI* tem como característica a divisão em seis etapas para o seu funcionamento, seguindo uma série de regras e rotinas pré-estabelecidas. Estas etapas devem ser respeitadas, sendo que todas obedecem a uma seqüência lógica na execução. Para cada etapa de execução é apresentado um pseudocódigo (GUIMARÃES; LAGES, 1994).

3.4.1. Transformação dos caracteres em minúsculo

A primeira etapa do algoritmo é transformar todos os caracteres de entrada em caracteres minúsculos, o que favorece a uniformidade no tratamento dos dados. Isto implica na necessidade da busca por *string* ser exata, pois se tiver diferente, a busca falha.

Portanto, os caracteres de entrada são transformados em minúsculos, e o mesmo procedimento é adotado para os registros no banco de dados.

Para isso, será utilizada a classe de *String* da linguagem de programação *Java* chamada *toLowerCase*, conforme apresentado no pseudocódigo da Figura 3.1.

```
Algoritmo PSI: Transforma em Minúsculo
Entrada: termo_pesquisa
Saída: termo_pesquisa_minúsculo

1) letra: CARACTER;
2) termo_pesquisa: CARACTER;
3) termo_pesquisa_minusculo: CARACTER;

4) Leia (termo_pesquisa);

5) Para i de 1 até tamanho(termo_pesquisa) faça
6)     letra <- termo_pesquisa{i};
7)     letra <- minusculo(letra); {transforma cada letra do termo de pesquisa na sua
       forma minúscula}
8)     termo_pesquisa_minusculo <- termo_pesquisa_minusculo + letra; {concatena
       cada letra transformada à variável de saída}
9) Fim para
```

Figura 3.1 Pseudocódigo referente à etapa de transformação em minúsculo.

3.4.2. Eliminação de símbolos

A segunda etapa do algoritmo é a eliminação de símbolos, conforme é apresentado na Tabela 3.25. Esses símbolos não favorecem a exatidão da busca, e por não existirem transcrições fonéticas para eles, são desconsiderados neste algoritmo.

Tabela 3.25 Símbolos

(*Abre parênteses*	) *Fecha parênteses*	- *traço*	. *ponto*	&	_ *underline*	º	ª	@ *arroba*

Exemplo da estruturação da codificação deste passo pode ser visto no pseudocódigo apresentado na Figura 3.2.

```
Algoritmo PSI: Elimina Símbolos
Entrada: termo_pesquisa_minusculo
Saída: termo_pesquisa

termo_pesquisa_minusculo: CARACTER;
termo_pesquisa: CARACTER;

Leia (termo_pesquisa_minusculo);
termo_pesquisa <- termo_pesquisa_minusculo;

Para i de 0 até tamanho(termo_pesquisa) - 1 faça
        Se (termo_pesquisa[i]="(" ou termo_pesquisa[i]=")" ou termo_pesquisa[i]="-" ou
            termo_pesquisa[i]="." ou termo_pesquisa[i]="&" ou termo_pesquisa[i]="_" ou
            termo_pesquisa[i]="º" ou termo_pesquisa[i]="ª" ou termo_pesquisa[i]="@") então
                termo_pesquisa[i]=""; {retira os símbolos do termo de pesquisa}
                i+1;
        Fim se
Fim para
```

Figura 3.2 Pseudocódigo referente à etapa de eliminação de símbolos.

3.4.3. Eliminação de preposições e conjunção

A segunda etapa do algoritmo é a eliminação de ocorrências de algumas preposições, conforme é apresentado na Tabela 3.26. Conforme o estudo de caso da língua portuguesa, tais preposições são derivadas da preposição “de + vogais” (“de+a”, “de+e”, “de+i”, “de+o”, “de+u”, “de+as”, “de+os”). Também é adotada como procedimento a eliminação da conjunção “e” isolada no texto, separada entre duas ou mais palavras.

Esses procedimentos são necessários visto que muitos casos não são considerados como parte integrante de um nome, o que justifica o descarte ou não tendo o seu valor devido.

Para exemplificar, considere os registros de nome “Benedito dos Santos” e “Benedito Santos”. Embora o primeiro registro possua a preposição entre os dois nomes diferindo do segundo registro, pode-se tratar da mesma pessoa, o que justifica a eliminação do uso das preposições.

Tabela 3.26 Preposições e conjunção

da	de	di	do
du	das	dos	e

Portanto, ao utilizar ou não a preposição na busca de valores, os resultados retornados devem ser os mesmos quando se tratam do termo de busca ser similar ao termo

registrado. Exemplo da estruturação da codificação deste passo pode ser visto no pseudocódigo apresentado na Figura 3.3.

```
Algoritmo PSI: Eliminação de preposições e conjunção "e"
Entrada: termo_pesquisa
Saída: termo_pesquisa

termo_pesquisa_minusculo: CARACTER;
termo_pesquisa: CARACTER;

Leia (termo_pesquisa_minusculo);
termo_pesquisa <- termo_pesquisa_minusculo;

Para i de 0 até tamanho(termo_pesquisa) - 1 faça
        Se (termo_pesquisa[i-1]=" " e termo_pesquisa[i]="e" e termo_pesquisa[i+1]=" ") então
                termo_pesquisa[i]=""; {retira a conjunção "e" do termo de pesquisa}
                i+1;
        Fim se
        Se (termo_pesquisa[i-1]="" e termo_pesquisa[i]="d") então
                Se (termo_pesquisa[i+1]="a" ou termo_pesquisa[i+1]="e" ou termo_pesquisa[i+1]="i"
                 ou termo_pesquisa[i+1]="o" ou termo_pesquisa[i+1]="u") então
                        Se (termo_pesquisa[i+2]=" ") então {retira as preposições "da", "de", "di",
                        "do", "du" do termo de pesquisa}
                                termo_pesquisa[i]="";
                                termo_pesquisa[i+1]="";
                                i+2;
                        Senão
                                Se (termo_pesquisa[i+2]="s" e termo_pesquisa[i+3]=" ") então {retira
                                as preposições "das" e "dos" do termo de pesquisa}
                                        termo_pesquisa[i]="";
                                        termo_pesquisa[i+1]="";
                                        termo_pesquisa[i+2]="";
                                        i+3;
                                Fim se
                        Fim se
                Fim se
        Fim se
Fim para
```

Figura 3.3 Pseudocódigo referente à etapa de eliminação de preposições e conjunção

3.4.4. Eliminação dos acentos

A terceira etapa do Algoritmo *PSI* é eliminar os acentos das vogais acentuadas. Tal eliminação é importante para que não haja diferença na busca por registro acentuado ou não. Portanto, com este procedimento é possível identificar registros iguais armazenados com ou sem acentos em suas vogais, tanto na fase de entrada de dados como

na fase de busca nos registros de dados do banco. Na Tabela 3.27 é apresentado um exemplo da eliminação dos acentos das vogais.

Tabela 3.27 Vogais acentuadas

á, à, ã, â, ä = a	é, è, ẽ, ê, ë = e	í, ì, ï, î = i
ó, ò, ô, õ, ö = o	ú, ù, û, ü = u	

Portanto, se for realizada a busca pela palavra “Pássaro” ou “Passaro”, ambos os registros devem ser retornados caso existam na base de dados. Isto se deve ao fato de que o usuário pode esquecer a utilização do acento ou desconhecer as regras ortográficas de acentuação da língua em questão. Exemplo da estruturação da codificação pode ser visto no pseudocódigo apresentado na Figura 3.4.

```
Algoritmo PSI: Elimina Vogais Acentuadas
Entrada: termo_pesquisa
Saída: termo_pesquisa

letra: CARACTER;
termo_pesquisa: CARACTER;

Leia (termo_pesquisa);

Para i de 1 até tamanho(termo_pesquisa) faça
        letra <- termo_pesquisa{i};
        Se (letra="á" ou letra="à" ou letra="ã" ou letra="â" ou letra="ä") então
                termo_pesquisa[i]="a"; {Substitui a vogal "a" acentuada pelo "a" sem acento}
        Fim se
        Se (letra="é" ou letra="è" ou letra="ẽ" ou letra="ê" ou letra="ë") então
                termo_pesquisa[i]="e"; {Substitui a vogal "e" acentuada pelo "e" sem acento}
        Fim se
        Se (letra="í" ou letra="ì" ou letra="î" ou letra="ï") então
                termo_pesquisa[i]="i"; {Substitui a vogal "i" acentuada pelo "i" sem acento}
        Fim se
        Se (letra="ó" ou letra="ò" ou letra="õ" ou letra="ô" ou letra="ö") então
                termo_pesquisa[i]="o"; {Substitui a vogal "o" acentuada pelo "o" sem acento}
        Fim se
        Se (letra="ú" ou letra="ù" ou letra="û" ou letra="ü") então
                termo_pesquisa[i]="u"; {Substitui a vogal "u" acentuada pelo "u" sem acento}
        Fim se
Fim para
```

Figura 3.4 Pseudocódigo referente à etapa de eliminação de vogais acentuadas

3.4.5. Eliminação de caracteres repetidos

A quarta etapa do Algoritmo *PSI* é eliminar os caracteres repetidos seguidamente. Tal procedimento é adotado tanto nos termos de entrada na busca de dados quanto nos registros armazenados na base. Trata-se de um passo importante e que se justifica por tratar da fonética, onde o que importa é o som do caractere e da pronunciação da palavra.

Para exemplificar a utilização, considere a palavra “carro”, pois esta palavra assumirá o valor “caro”. Portanto, os caracteres “rr” são substituídos por um único “r”. Por se tratar da leitura, a pronunciação do som pode ser tanto dos dois “erres” quanto de um único “erre”. Este procedimento será tomado para a ocorrência de todas as letras do alfabeto que vier seguidamente repetida neste algoritmo, assumindo os seguintes valores conforme apresentados na Tabela 3.28:

Tabela 3.28 Caracteres repetidos

aa = a	hh = h	oo = o	vv = v
bb = b	ii = i	pp = p	ww = w
cc = c	jj = j	qq = q	xx = x
dd = d	kk = k	rr = r	yy = y
ee = e	ll = l	ss = s	zz = z
ff = f	mm = m	tt = t	
gg = g	nn = n	uu = u	

Exemplo da estruturação da codificação pode ser visto no pseudocódigo apresentado na Figura 3.5.

```
Algoritmo PSI: Elimina Caracteres Repetidos
Entrada: termo_pesquisa
Saída: termo_pesquisa

termo_pesquisa: CARACTER;

Leia (termo_pesquisa);

Para i de 1 até tamanho(termo_pesquisa) faça
        Se (termo_pesquisa[i] = termo_pesquisa[i+1]) então
                termo_pesquisa[i+1]=""; {Elimina o segundo caractere repetido seguidamente}
        Fim se
Fim para
```

Figura 3.5 Pseudocódigo referente à etapa de eliminação de caracteres repetidos

3.4.6. Transformação fonética

A quinta e última etapa do Algoritmo *PSI* é a parte principal do algoritmo, pois é responsável pela transcrição fonética do termo de busca para posteriores pesquisas na base de dados. Tais transcrições são possíveis por meio das transcrições fonéticas de cada letra do alfabeto, conforme foi explicado no capítulo 3.3, visão geral da proposta.

O funcionamento desta transformação ocorre com a transcrição individual de cada letra que compõe uma palavra a ser buscada. Desta palavra são extraídas as suas diferentes formas de escrita, o que caracterizam diferentes termos a ser buscado no banco de dados.

A estruturação de codificação deste passo pode ser visto no pseudocódigo apresentado na Figura 3.6.

```
Algoritmo PSI: Transformação Fonética
Entrada: termo_pesquisa
Saída: vetor_transcricoes

letra: CARACTER;
termo_pesquisa: CARACTER;
vetor_transcricoes: CARACTER;
vetor_aux: CARACTER;

Leia (termo_pesquisa);

Para i de 1 até tamanho(termo_pesquisa) faça
        letra <- termo_pesquisa{i};
        Se (vetor_transcricoes = vazio) então
                vetor_transcricoes = transcricao(letra); {atribui ao vetor as primeiras letras transcritas}
        Senão
                Para j de 1 até tamanho (vetor_transcicoes) faça
                        vetor_aux = vetor_transcricoes[j] + transcricao(letra); {atribui ao vetor auxiliar
                        as próximas letras transcritas}
                Fim para
                vetor_transcricoes = vetor_aux; {atribui ao vetor principal as transcrições do vetor
                auxiliar}
                vetor_aux = vazio; {limpa o vetor auxiliar}
        Fim se
Fim para
```

Figura 3.6 Pseudocódigo referente à etapa de transformação fonética

Para exemplificar, é utilizada a língua portuguesa, que pode ser escolhida dentro de um leque de idiomas que o ambiente disponibiliza. Para este estudo de caso, se a busca a ser realizada for da palavra “casa”, todas as possíveis transcrições fonéticas serão retornadas, obtendo os seguintes resultados: “kasa”, “kasã”, “kaza”, “kazã”, “kãsa”,

“kãsã”, “kãza”, “kãzã”, “casa”, “casã”, “caza”, “cazã”, “cãsa”, “cãsã”, “cãza” e “cãzã”. Conforme é apresentado na Figura 3.7, para cada letra foi realizada a sua transcrição, formando diversos resultados e combinações que serão buscados na base de dados. Essa operação de transcrição fonética é realizada independente do idioma escolhido, mas que para este caso está exemplificado pela língua portuguesa.

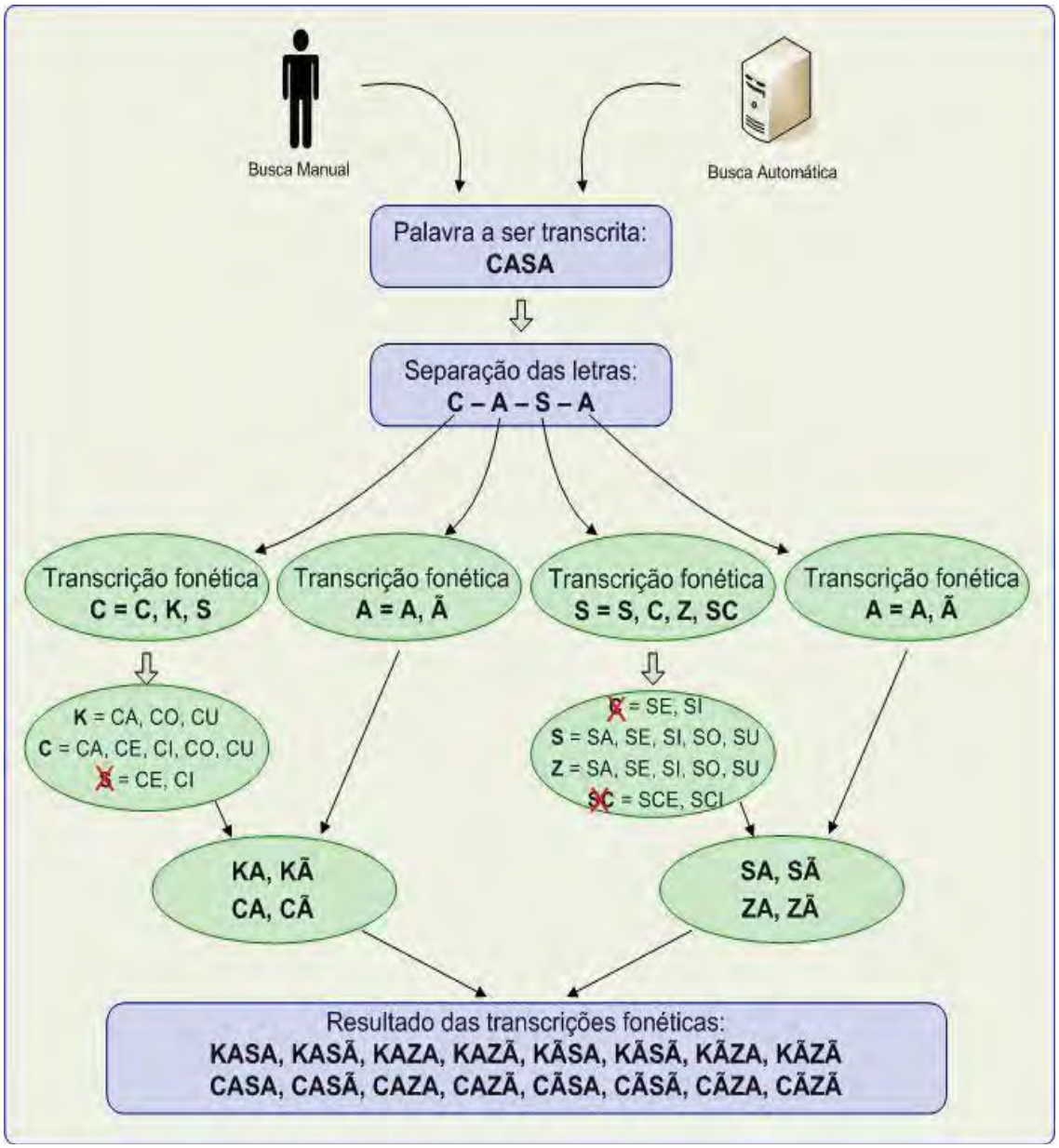


Figura 3.7 Transcrições fonéticas

Portanto, o algoritmo é capaz de identificar valores similarmente armazenados e duplicados nas bases de dados, principalmente com a realização das diferentes transcrições fonéticas que uma palavra pode passar. Portanto, o algoritmo utiliza técnicas de transformação de caracteres em seus respectivos valores similares foneticamente como forma de buscar registros duplicados existentes na base de dados.

3.5. Descrição do Ambiente *DIBP*

Para a realização de testes para obtenção dos resultados foi construído um ambiente de aplicação denominado DIBP (*Data Identification Based on Phonetic*), que permite a escolhe do Sistema Gerenciador de Banco de Dados *MySQL* ou *PostgreSQL*.

Na Figura 3.8 é apresentado um diagrama de blocos com divisões entre camadas que ajuda a entender a organização do ambiente construído para a execução dos testes (LARMAN, 2004).

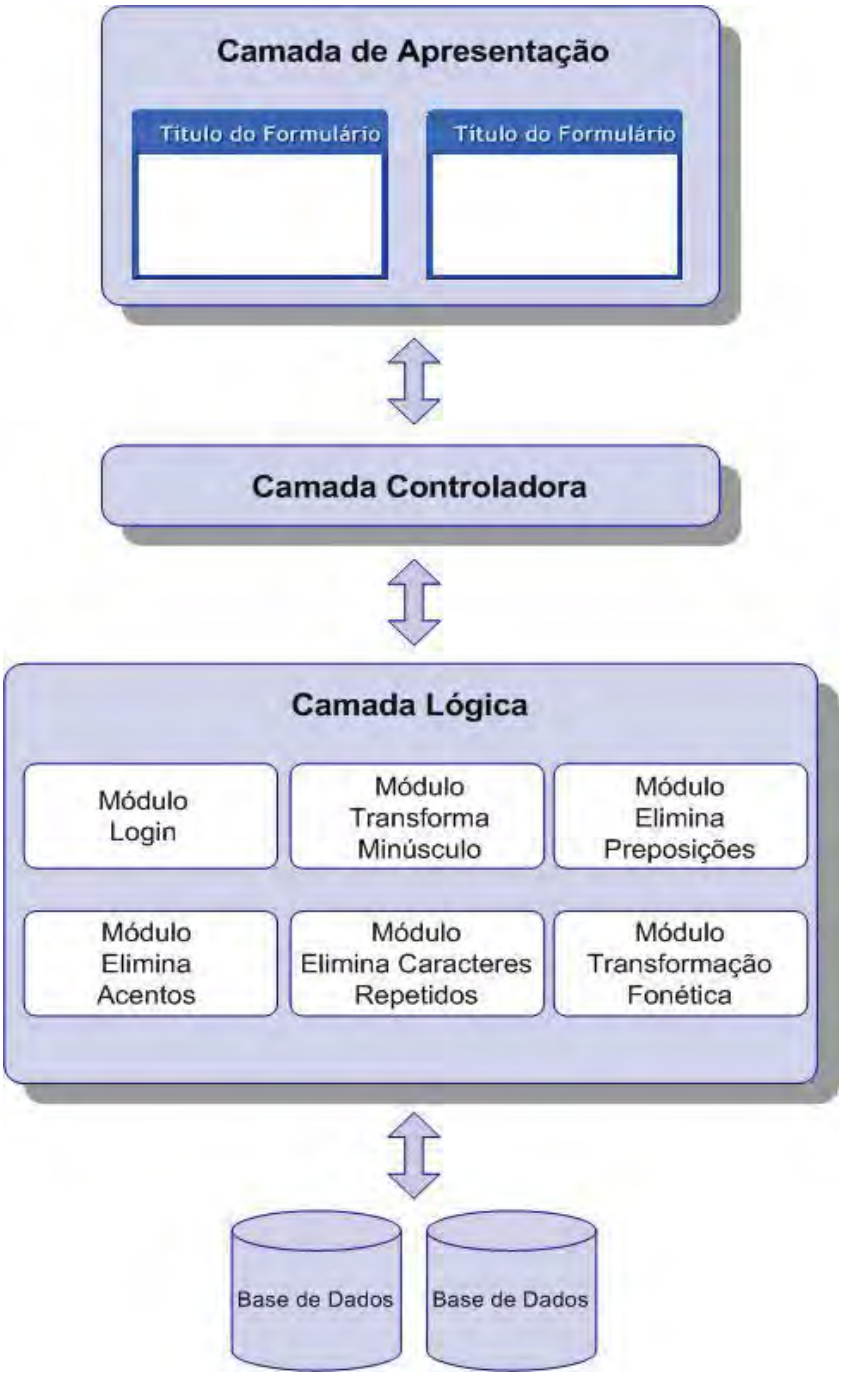


Figura 3.8 Diagrama de blocos do ambiente de aplicação DIBP (Baseado em LARMAN, 2004).

A primeira camada apresentada na Figura 3.8 do diagrama de blocos é a de Apresentação, responsável pela interface gráfica do ambiente. Esta camada é a que apresenta todas as funcionalidades do ambiente ao usuário, como botões caixas de textos e características de fonte. Dessa forma, visa-se a interação, sendo pertinente a preocupação com a usabilidade, viabilidade, interface amigável e padronização no desenvolvimento do ambiente nesta camada.

A segunda camada da Figura 3.8 é a Controladora, responsável pela interligação da camada de apresentação com a lógica. Após a entrada do usuário na camada de apresentação, esta camada direciona para a utilização dos módulos na camada lógica.

Já a última camada da Figura 3.8, a Camada Lógica, contém todos os módulos existentes no ambiente, e que representam as suas funcionalidades. Dentre os módulos existentes, cita-se: *Login, Transforma Minúsculo, Elimina Preposições, Elimina Acentos, Elimina Caracteres Repetidos* e *Transformação Fonética*. Esta última camada é responsável pelo tratamento dos dados e a conexão diretamente com a base de dados.

Cada algoritmo trabalha com um ou mais módulos descritos na Camada Lógica, a citar o Algoritmo *Soundex* e *Metaphone* que trabalham apenas com o *Módulo Transformação Fonética*, e o Algoritmo *PSI*, que trabalha com todos os módulos apresentados.

O módulo de *Login* é responsável pela conectividade com a base de dados, onde o usuário é determinado a fornecer algumas informações necessárias para estabelecer conexão com o banco de dados, como nome de usuário, senha, nome do banco, número do *IP* e o tipo de gerenciador da base (*MySQL* ou *PostgreSQL*). Após ser estabelecida a conectividade com a base de dados por meio da interface de *login*, o usuário será remetido a uma próxima tela que contem algumas informações necessárias para a realização da tarefa de identificação de registros duplicados. Dentre as informações, o usuário terá que escolher o tipo de variação (*Soundex, Metaphone* ou *PSI*) por meio da opção “Selecione a Variação”, a tabela do banco de dados na opção “Selecione a Tabela”, a coluna (atributo) na opção “Selecione a Coluna” e o idioma na opção “Selecione o Idioma” que pretende aplicar tal variação. A opção idioma abrange um leque de opções, como italiano, português e inglês. Além dessas informações, fica a critério do usuário escolher por meio da opção “Tipo de pesquisa” a forma que o algoritmo escolhido deve trabalhar, que varia entre automática ou manual.

O tipo de pesquisa automática não permitirá ao usuário a entrada de dados na opção “Termo de Pesquisa”. Portanto, a aplicação de sua funcionalidade ocorre em todos

os registros do banco de dados, conforme funcionamento do algoritmo escolhido, restringindo no caso da utilização do *Soundex* ao máximo de quatro caracteres, o *Metaphone* ao máximo de seis caracteres da(s) palavra(s) pesquisada(s), e no *PSI* a transcrição dos dois primeiros termos da palavra pesquisada.

O tipo de pesquisa manual permite o usuário que entre com um dado qualquer, para que seja aplicado a funcionalidade do algoritmo neste dado de entrada e posterior realização da consulta na base de dados, e obedece os mesmos critérios de codificação citados na funcionalidade do tipo de pesquisa automático. Detalhes da interface da tela de aplicação são apresentados na Figura 3.9.

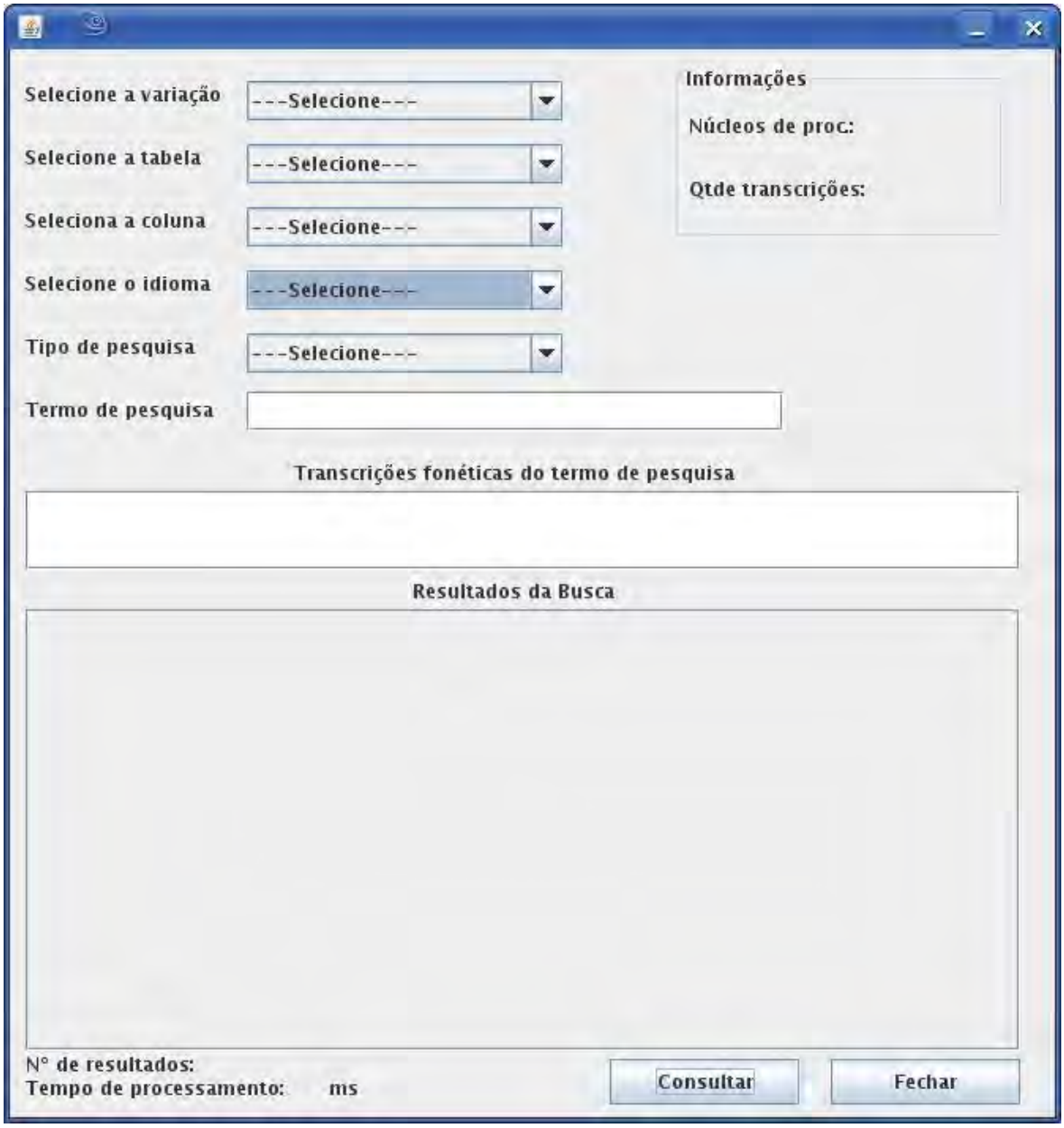


Figura 3.9 Interface do ambiente de aplicação

Nota-se no ambiente de aplicação que existe um espaço reservado a quantidade de transcrições e as transcrições fonéticas da palavra que for fornecida como termo de pesquisa no caso de utilização do tipo de pesquisa manual e da escolha da variação *PSI*. Além disso, informações sobre a quantidade de núcleos de processadores no computador que está sendo executado no ambiente também é exibida, aplicando-se a utilização de *multithreading*.

Após o fornecimento de todas as informações necessárias para a realização da busca, os resultados são apresentados no espaço nomeado como “Resultados da Busca”, destinado à apresentação de todos os resultados possíveis por meio da exposição dos dados de dois atributos da base de dados, sendo o principal o escolhido na opção “Selecione a coluna” e o subseqüente a ele. Também são exibidos o número total de resultados obtidos e o tempo de processamento em milissegundos do algoritmo escolhido.

3.6. Considerações Finais

Neste capítulo, foram apresentados os principais conceitos sobre a fonética a fim de melhor explicar o porquê de sua utilização, destacando-a dentro do universo do trabalho proposto. Para complementar, foi apresentado um estudo de caso da língua portuguesa, com exemplos das diferentes transcrições fonéticas deste idioma e apresentação da utilização de uma técnica para identificação de registros numéricos similares. Após o estudo de caso, foram apresentadas as principais etapas de funcionamento do algoritmo proposto, além de demonstrações de trechos de códigos para a codificação. Para finalizar, foi apresentada a interface principal do ambiente desenvolvido.

Capítulo 4
Testes e Resultados

4.1. Considerações iniciais

Neste capítulo são apresentados os testes efetuados com o algoritmo proposto, com o intuito de apresentar o desempenho obtido pela proposta em relação a outros algoritmos. O estudo realizado compreende a análise do algoritmo *PSI* comparando-o a algoritmos difundidos na literatura, como *Soundex* e *Metaphone*, a fim de verificar a contribuição do trabalho em termos práticos. Além disso, os testes visam à análise comparativa entre a aplicação executada em um núcleo de processador e dois núcleos de processadores, com o intuito de mensurar os benefícios trazidos pela utilização de *multithreading*.

4.2. Bases de dados e parâmetros de testes

As bases de dados utilizadas são bases que contém registros informados por profissionais de diferentes áreas e que fazem uso de um sistema computacional de cadastro. A Tabela 4.1 apresenta as informações das bases de dados utilizadas para realização de testes, tais como o nome das tabelas, nome das colunas e o número de registros armazenados.

Tabela 4.1 Bases de dados utilizadas para testes

Base de Dados	Tabela	Coluna	Nº de registros
SIVAT	Máquina Causadora	Nome	444
	Ramo Atividade	Descrição	260
	Ocupação	Descrição	1.330
	Médico	Nome	141.375
FUNCIONÁRIOS	Inscritos	Nome	100
		Endereço	100

A primeira base de dados, denominada SIVAT (*Sistema de Vigilância de Acidentes de Trabalho*), registra informações sobre acidentes de trabalho na cidade de São José do Rio Preto – SP. Atualmente, possui cerca de 50 mil fichas cadastradas de acidentes coletados pelo órgão vinculado a prefeitura do município e responsável pelo acompanhamento. Essa base de dados é gerenciada pelo Grupo de Banco de Dados do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista – UNESP.

A segunda base de dados, denominada FUNCIONÁRIOS, é uma base de dados construída localmente no computador que contém 486 registros de nomes e endereços de pessoas na língua italiana para a realização de testes do ambiente, a fim de comprovar a funcionalidade da aplicação nesse idioma.

A base SIVAT encontra-se armazenada em um SGBD *PostgreSQL*, enquanto que a base FUNCIONÁRIOS encontra-se armazenada em um SGBD *MySQL*. O ambiente desenvolvido dá suporte a utilização dos dois sistemas gerenciadores, não sendo necessária mudanças ou manipulação de dados para o seu correto funcionamento.

O equipamento utilizado para a execução do ambiente desenvolvido para testes contém a seguinte configuração:

1. *Notebook* com processador Intel Core 2 Duo (2.2 Ghz), 4 *gigabytes* de memória principal *DDR2* e um disco rígido de 500 *gigabytes*, com o sistema operacional *Windows 7 Home Edition*, Kit de Desenvolvimento *Java J2SE* 1.6 e Ambiente de Desenvolvimento *Eclipse Helios*.

Para a realização dos testes neste computador, o ambiente DIBP foi configurado para ser executado utilizando um núcleo e dois núcleos de processadores. Os testes consistiram na aplicação dos algoritmos em bases de dados, a fim de obter medidas que

descrevam o desempenho do algoritmo proposto em relação ao estado da arte, com intuito de medir o tempo de execução de cada algoritmo e comparar os resultados na busca de informações precisas e adequadas.

Vale antecipar que o desempenho do algoritmo *PSI* está diretamente relacionado ao idioma escolhido, pois, conforme o idioma será realizado um determinado número de transcrições fonéticas para cada letra. Para exemplificar, no estudo de caso realizado para a língua portuguesa, a consoante "c" possui um maior número de transcrição que na língua italiana, o que caracteriza um maior número de transcrições que o algoritmo necessita realizar.

4.3. Estudo Comparativo

Com o objetivo de comprovar a eficiência do algoritmo proposto, neste capítulo são apresentados os testes realizados com os algoritmos *Soundex*, *Metaphone* e *PSI* em diferentes bases de dados. O estudo consiste na comparação dos resultados obtidos e do tempo de execução desses três algoritmos implementados.

Para o algoritmo *PSI*, é implementada a divisão de processamento entre os núcleos de processadores do computador, denominado *multithreading*, a fim de aferir o tempo de execução em seu funcionamento. Para melhorar a amostragem dos dados coletados, tabelas e gráficos ilustram os diversos testes efetuados.

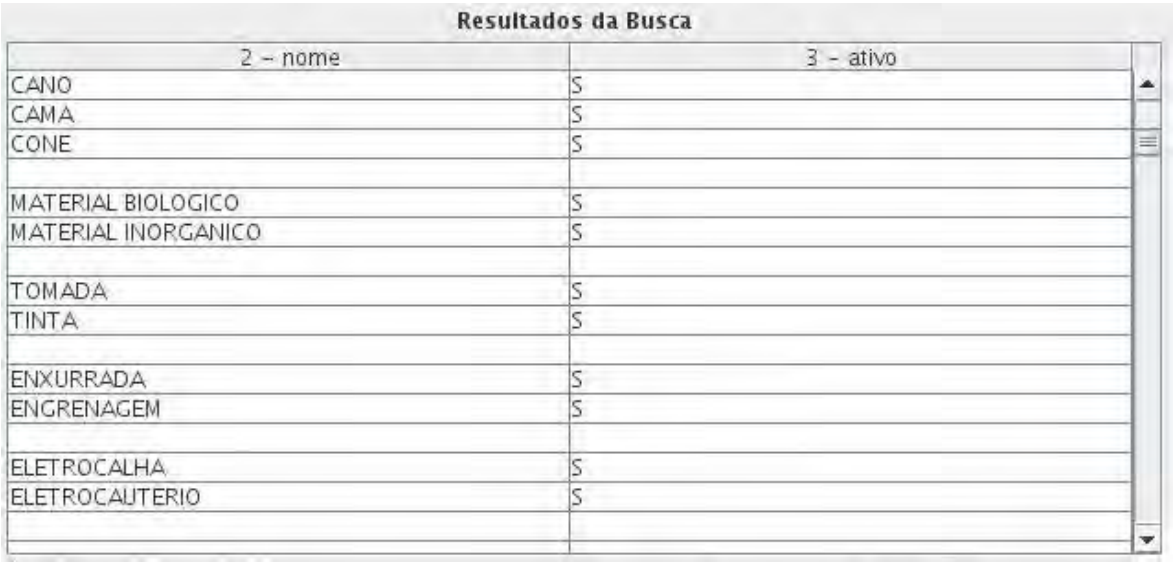
Os testes foram realizados nos idiomas Português e Italiano, a fim de comprovar a eficiência do ambiente e aplicação do algoritmo em mais de um idioma, bastando para tal que o usuário o escolha dentro as opções.

4.3.1. Estudo comparativo entre os resultados do *Soundex*, *Metaphone* e *PSI*

Nesta seção, serão apresentados por meio de figuras os resultados da aplicação entre os três algoritmos implementados, *Soundex*, *Metaphone* e *PSI*, bem como o tempo de execução exibido graficamente de cada um.

4.3.1.1. Experimento 1 – Base SIVAT

O primeiro teste foi realizado na base de dados "SIVAT", tabela "Máquina Causadora" e coluna "Nome", executados por meio do "tipo de pesquisa automático" e o idioma "Português". Os resultados são apresentados nas Figuras 4.1, 4.2 e 4.3.

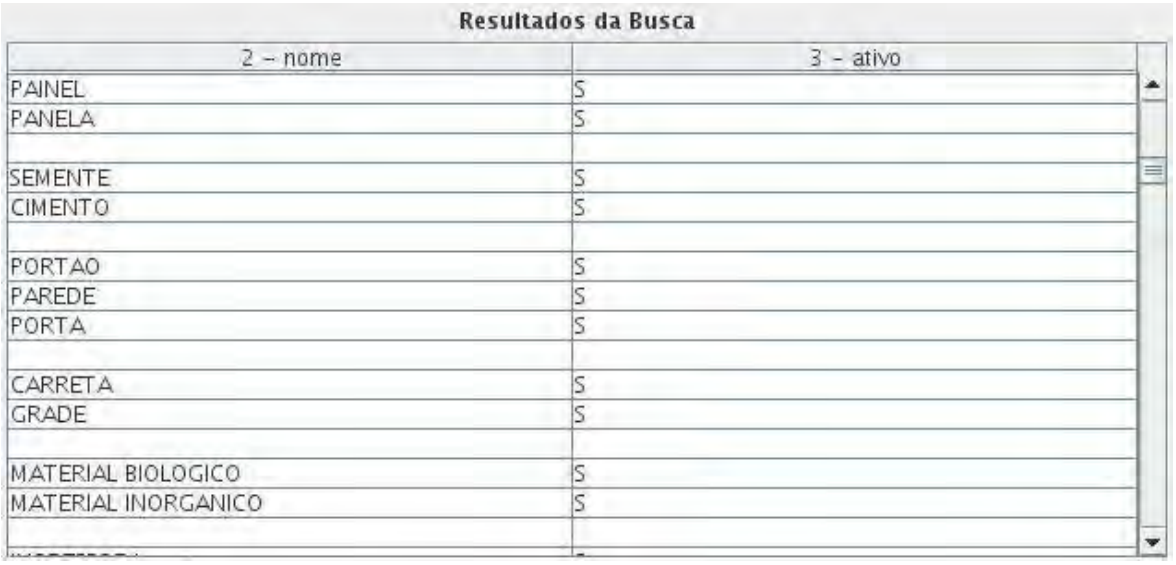


Resultados da Busca

2 - nome	3 - ativo
CANO	S
CAMA	S
CONE	S
MATERIAL BIOLOGICO	S
MATERIAL INORGANICO	S
TOMADA	S
TINTA	S
ENXURRADA	S
ENGRENAGEM	S
ELETROCALHA	S
ELETROCAUTERIO	S

Figura 4.1 Busca automática na tabela máquina causadora utilizando o algoritmo *Soundex*.

Por meio da Figura 4.1 apresentada, o Algoritmo *Soundex* mostra 1.178 resultados. Pode-se observar que, por este algoritmo não trabalhar com codificação das vogais, os resultados apresentados foram os mais variados possíveis foneticamente. Dentre eles, pode-se destacar a similaridade apresentada entre as palavras “Cano”, “Cama” e “Cone”, e “Tomada” e “Tinta”, entre outros. Nota-se que as palavras apresentadas como similares possuem sua codificação igual, mesmo diante das desigualdades de significado, o que resulta em uma apresentação de resultados imprecisos e representa um problema na busca pelos registros duplicados.

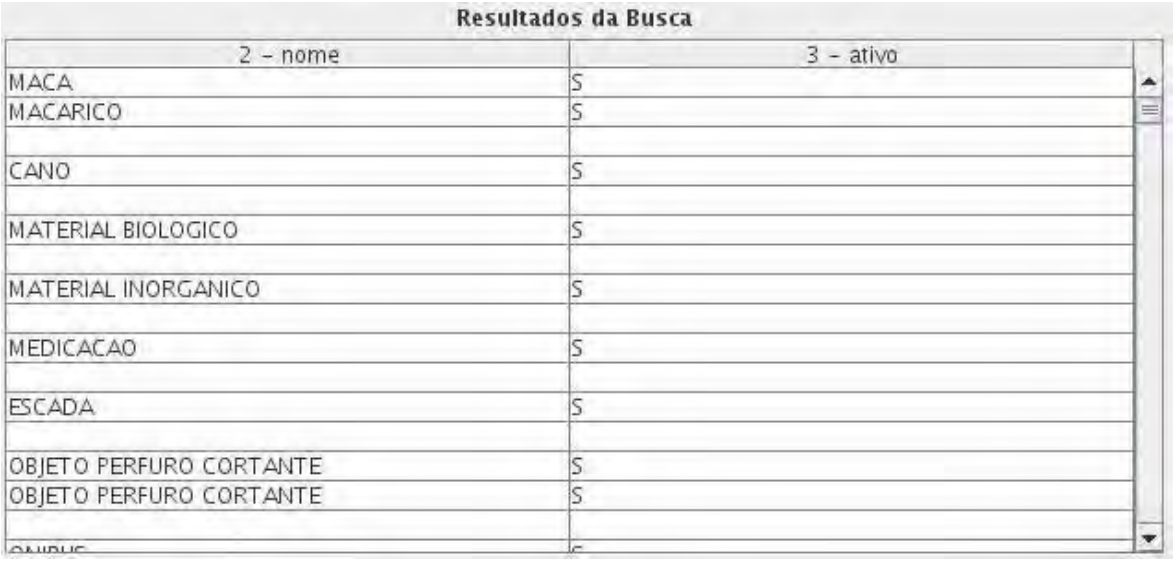


Resultados da Busca

2 - nome	3 - ativo
PAINEL	S
PANELA	S
SEMENTE	S
CIMENTO	S
PORTAO	S
PAREDE	S
PORTA	S
CARRETA	S
GRADE	S
MATERIAL BIOLOGICO	S
MATERIAL INORGANICO	S

Figura 4.2 Busca automática na tabela máquina causadora utilizando o algoritmo *Metaphone*.

Conforme é apresentado na Figura 4.2, o Algoritmo *Metaphone* mostra resultados similares e discrepantes foneticamente. Dos 1.068 resultados apresentados, os termos “Cimento” e “Semente” têm características fonéticas parecidas, o que demonstra uma precisão na busca por similaridade, mas resultados como “Portão”, “Parede” e “Porta”, e “Carreta” e “Grade” não possuem similaridade fonética alguma, o que caracteriza uma grande discrepância ao se tratar de similaridade fonética entre esses termos. Fato que é um problema na identificação de registros duplicados, pois resultados imprecisos favorecem uma busca imprópria para utilização.



Resultados da Busca

2 - nome	3 - ativo
MACA	S
MACARICO	S
CANO	S
MATERIAL BIOLOGICO	S
MATERIAL INORGANICO	S
MEDICACAO	S
ESCADA	S
OBJETO PERFURO CORTANTE	S
OBJETO PERFURO CORTANTE	S

Figura 4.3 Busca automática na tabela máquina causadora utilizando o algoritmo *PSI*.

Conforme é apresentado na Figura 4.3, com a utilização do Algoritmo *PSI* foram observados 1.271 resultados foneticamente similares. Termos que apresentaram similaridade fonética ao utilizar o Algoritmo *Soundex*, como “Cano”, “Cama” e “Cone”, não são apresentados neste algoritmo, sendo mostrados isoladamente, já que não caracteriza similaridade fonética entre eles. Diante disso, verifica-se uma busca mais exata dos termos duplicados. Uma observação importante na Figura 4.3 é a apresentação dos resultados “Maca” e “Macarico”, pois só foram relatados como similares devido a não utilização do caractere especial “ç”, o que, portanto, caracteriza uma similaridade entre esses dois termos. Resultados como “material biológico” e “material inorgânico”, também, não foram caracterizados como foneticamente similares, pois o algoritmo trabalha com a transcrição dos dois termos da busca, o que faz com eles sejam caracterizados diferentemente.

O desempenho da aplicação dos algoritmos *Soundex*, *Metaphone* e *PSI* é apresentado na Figura 4.4.

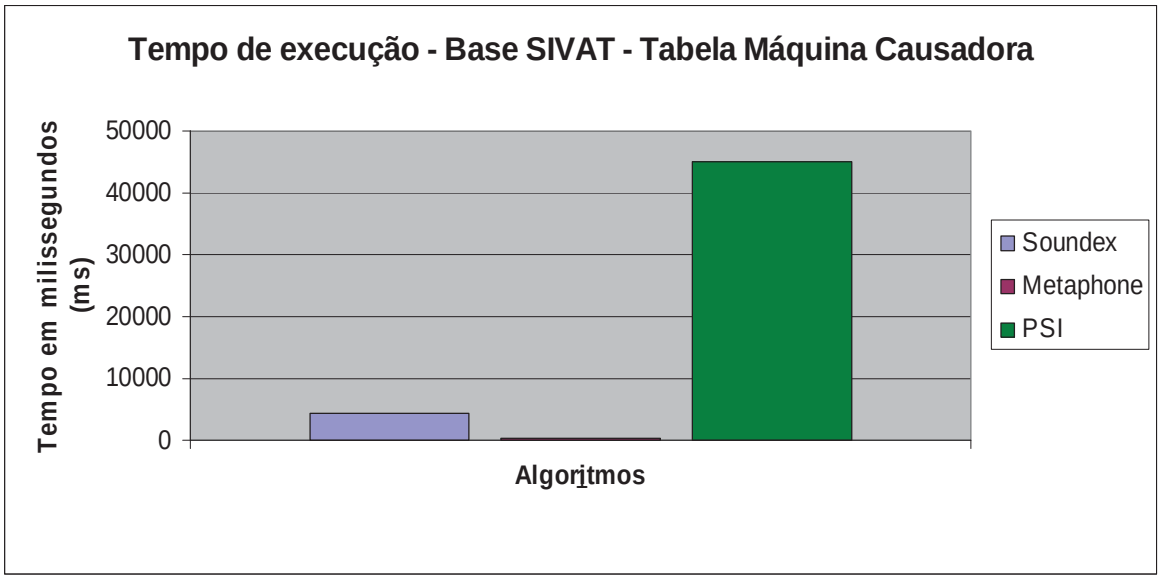
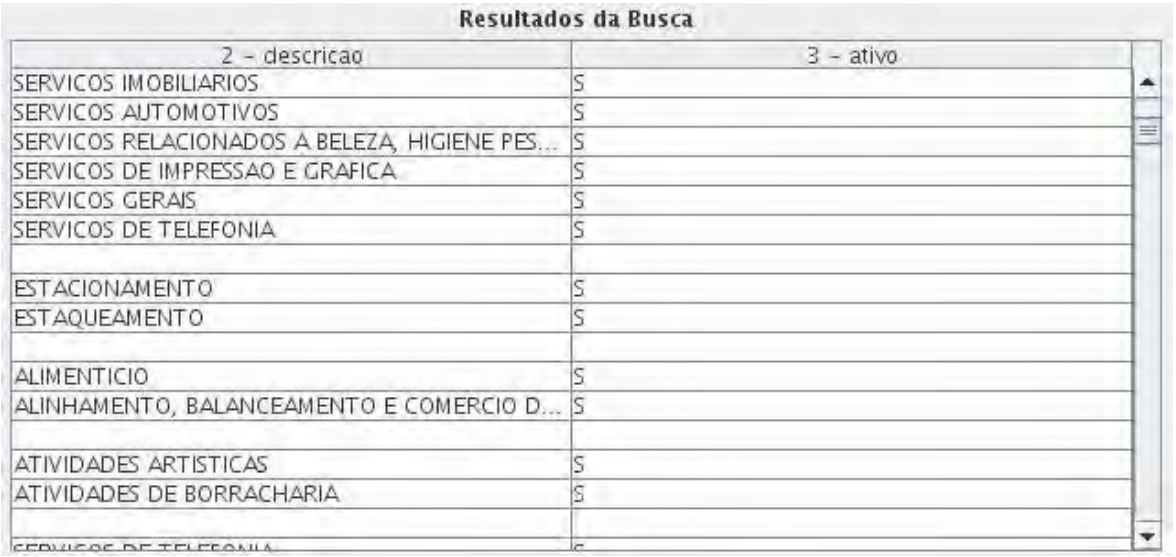


Figura 4.4 Gráfico do tempo de execução – Base SIVAT – Tabela máquina causadora.

Conforme é apresentado na Figura 4.4, o tempo de execução dos algoritmos foram diferenciados diante dos 444 registros armazenados nessa tabela do banco de dados. O Algoritmo *Soundex* processou as informações em 5 segundos, o Algoritmo *Metaphone* em 1 segundo, e o Algoritmo *PSI* em 45 segundos. Nota-se que o tempo de processamento do Algoritmo *PSI* é muito superior se comparado aos demais, isto porque sua metodologia de funcionamento é diferenciada e composta por várias etapas e os resultados atingidos são melhores na busca por registros duplicados. O Algoritmo *Soundex* trabalha com a codificação de consoantes até a formação de quatro caracteres codificados, o *Metaphone* com a formação de até seis caracteres codificados, enquanto que o *PSI* trabalha com a transcrição de todos os caracteres da palavra, além de analisar os casos de caracteres que possuem mais de uma transcrição, o que torna necessária a realização de mais de uma rotina de codificação, o que, consequentemente, torna o seu tempo de execução maior em relação aos demais algoritmos. Portanto, embora o tempo de processamento seja maior, a precisão da busca é compensatória se comparado aos demais, já que apresenta somente os registros que possuem grau de similaridade entre os termos buscados.

4.3.1.2. Experimento 2 – Base SIVAT

O segundo teste foi realizado na base de dados “SIVAT”, tabela “Ramo Atividade” e coluna “Descrição”, executados por meio do “tipo de pesquisa automático” e o idioma “Português”. Os resultados são apresentados nas Figuras 4.5, 4.6 e 4.7.



Resultados da Busca

2 - descricao	3 - ativo
SERVICOS IMOBILIARIOS	S
SERVICOS AUTOMOTIVOS	S
SERVICOS RELACIONADOS A BELEZA, HIGIENE PES...	S
SERVICOS DE IMPRESSAO E GRAFICA	S
SERVICOS GERAIS	S
SERVICOS DE TELEFONIA	S
ESTACIONAMENTO	S
ESTAQUEAMENTO	S
ALIMENTICIO	S
ALINHAMENTO, BALANCEAMENTO E COMERCIO D...	S
ATIVIDADES ARTISTICAS	S
ATIVIDADES DE BORRACHARIA	S
SERVICOS DE TELEFONIA	S

Figura 4.5 Busca automática na tabela ramo atividade utilizando o algoritmo *Soundex*.

Com a utilização do Algoritmo *Soundex*, foram mostrados 1.324 resultados conforme é apresentado na Figura 4.5. Termos como “estacionamento” e “estaqueamento” foram identificados como foneticamente similares. De acordo com as características de funcionamento deste algoritmo, termos como “Atividades artísticas” e “atividades de borracharia” são similares foneticamente, pois restringe sua codificação apenas a palavra “atividade” O mesmo acontece com as diferentes palavras que contém o termo “Serviço”. As palavras “alimentício” e “alinhamento”, também, atingem o mesmo código no processo de codificação, por isso são apontadas como similares. Diante dessas análises, é possível verificar uma discrepância dos resultados, o que torna a busca por registros duplicados imprecisa.

Resultados da Busca

2 – descricao	3 – ativo
CONFECCAO E COMERCIO DE ROUPAS E CALCADOS	S
CONFECCAO DE PAINEIS	S
CONFECCOES INTIMAS	S
INDUSTRIA E COMERCIO DE EQUIPAMENTOS ELET...	S
INDUSTRIA E COMERCIO DE MOVEIS	S
INDUSTRIA E COMERCIO DE ESTOFADOS	S
ATIVIDADES ARTISTICAS	S
ATIVIDADES DE BORRACHARIA	S
SERVICOS DE JARDINAGEM	S
SERVICOS PUBLICOS	S
SERVICO SOCIAL	N
SERVICOS DE LIMPEZA	S
SERVICOS DE ADVOCACIA	S

Figura 4.6 Busca automática na tabela ramo atividade utilizando o algoritmo *Metaphone*.

Conforme é apresentado na Figura 4.6, com a utilização do Algoritmo *Metaphone* foram vistos 1.278 resultados. Observa-se que as codificações dos termos restringiram-se apenas as primeiras palavras da busca, o que evidenciou a identificação de similaridade fonética entre termos, como “Atividades Artísticas” e “Atividades de Borracharia”, “Serviços de jardinagem” e os demais serviços, e “Indústria e Comércio de Estofados” e as demais indústrias.

Resultados da Busca

2 – descricao	3 – ativo
FARMACIA	S
TECNICO DE FARMACIA	S
SERVICOS IMOBILIARIOS	S
SERVICOS AUTOMOTIVOS	S
ESTACIONAMENTO	S
COMERCIO VAREJISTA DE ALIMENTOS	S
COMERCIO VAREJISTA DE CONFECCAO	S
AUXILIAR DE EXPEDICAO	S
VETERINARIA, PET SHOP E ATIVIDADES COM ANIM...	S

Figura 4.7 Busca automática na tabela ramo atividade utilizando o algoritmo *PSI*.

De acordo com o que é apresentado na Figura 4.7, com a utilização do Algoritmo *PSI* foram observados 492 resultados foneticamente similares. Resultados obtidos nos Algoritmos *Soundex* e *Metaphone* que identificaram similaridade entre os termos que continham as palavras “Serviços” e “Auxiliar” não foram apresentados na utilização deste

algoritmo. “Comercio Varejista de Alimentos” e “Comercio Varejista de Confecção” foram mostrados como foneticamente similares, já que a metodologia de funcionamento restringe a transcrição fonética nas duas primeiras palavras dos termos pesquisados. Nota-se que a utilização deste algoritmo torna a busca mais precisa na identificação de registros duplicados na base de dados.

O desempenho dos algoritmos mencionados é apresentado no gráfico da Figura 4.8.

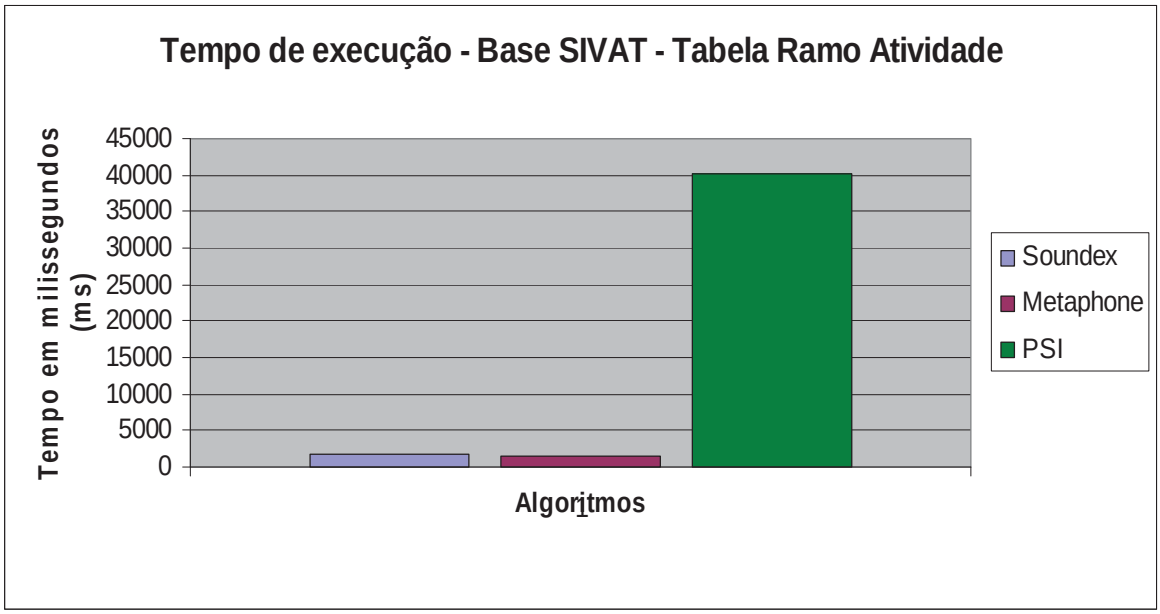
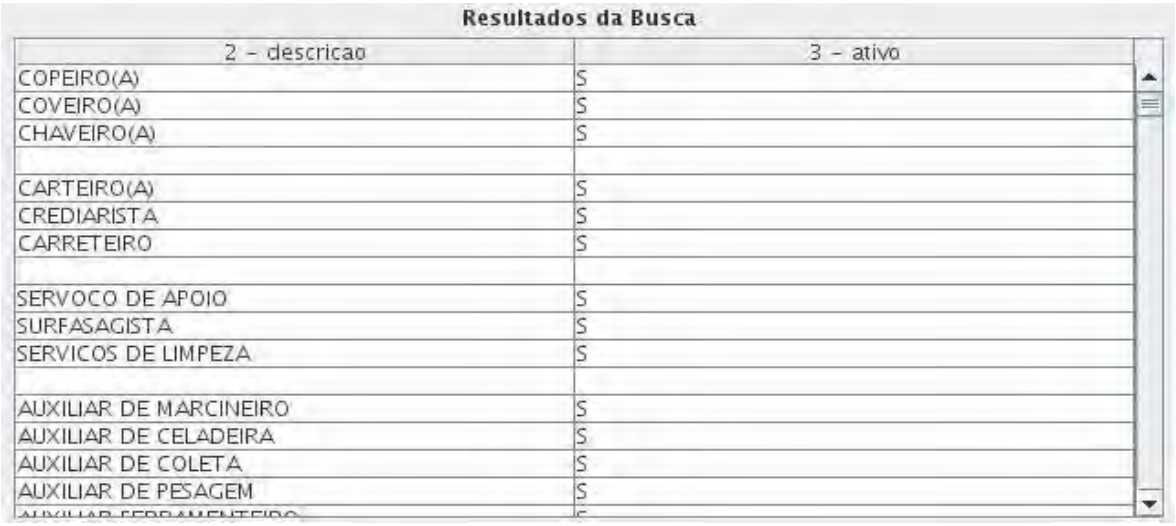


Figura 4.8 Gráfico do tempo de execução – Base SIVAT – Tabela ramo atividade.

De acordo com o gráfico apresentado na Figura 4.8, o tempo de execução do Algoritmo *Soundex* e *Metaphone* foi de aproximadamente 2 segundos, enquanto que o Algoritmo *PSI* foi executado em 40 segundos mediante os 260 registros armazenados nesta tabela do banco de dados. Ao observar os resultados, nota-se um ganho de precisão na utilização desse último algoritmo em relação aos demais, embora seu tempo de processamento seja bem maior, pois a metodologia de funcionamento é diferenciada. O algoritmo *Soundex* transforma a palavra do banco em um código de quatro caracteres, o *Metaphone* em um código de até seis caracteres, enquanto que o *PSI* trabalha com todas as letras dos dois termos das palavras buscadas. Isso explica uma demora considerável deste algoritmo, que é compensada pela precisão dos resultados obtidos.

4.3.1.3. Experimento 3 – Base SIVAT

O terceiro teste foi realizado na base de dados “SIVAT”, tabela “Ocupação” e coluna “Descrição”, executados por meio do “tipo de pesquisa automático” e o idioma “Português”. Os resultados são apresentados nas Figuras 4.9, 4.10 e 4.11.

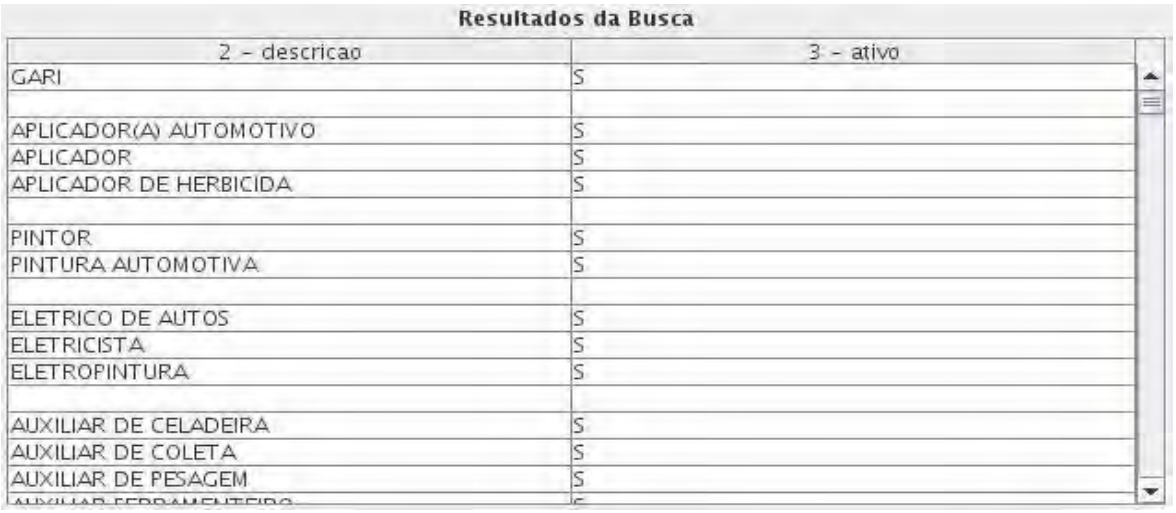


Resultados da Busca

2 - descricao	3 - ativo
COPEIRO(A)	S
COVEIRO(A)	S
CHAVEIRO(A)	S
CARTEIRO(A)	S
CREDIARISTA	S
CARRETEIRO	S
SERVOCO DE APOIO	S
SURFASAGISTA	S
SERVICOS DE LIMPEZA	S
AUXILIAR DE MARCINEIRO	S
AUXILIAR DE CELADEIRA	S
AUXILIAR DE COLETA	S
AUXILIAR DE PESAGEM	S

Figura 4.9 Busca automática na tabela ocupação utilizando o algoritmo *Soundex*.

Conforme apresentado na Figura 4.9, o Algoritmo *Soundex* mostra 66.064 resultados entre os mais variados possíveis, como “Copeiro” e “Chaveiro” que origina a codificação “C160”, e “Carteiro” e “Crediarista” com codificação igual a “C636”. Resultados aproximados também foram apresentados, como “Auxiliar de Pesagem” e “Auxiliar de Coleta”, já que esta codificação, basicamente, se restringe apenas ao termo “auxiliar”. Portanto, é possível observar que resultados aproximados e discrepantes são obtidos, o que pode caracterizar um processo problemático na identificação de registros duplicados.



Resultados da Busca

2 - descricao	3 - ativo
GARI	S
APLICADOR(A) AUTOMOTIVO	S
APLICADOR	S
APLICADOR DE HERBICIDA	S
PINTOR	S
PINTURA AUTOMOTIVA	S
ELETRICO DE AUTOS	S
ELETRICISTA	S
ELETROPINTURA	S
AUXILIAR DE CELADEIRA	S
AUXILIAR DE COLETA	S
AUXILIAR DE PESAGEM	S

Figura 4.10 Busca automática na tabela ocupação utilizando o algoritmo *Metaphone*.

Ao utilizar o Algoritmo *Metaphone*, foram obtidos 65.486 resultados, conforme apresentados na Figura 4.10. É possível observar alguns resultados mais aproximados, como “Aplicador” e “Aplicador de Herbicida”, e outros discrepantes como “Elétrico de Autos” e “Eletropintura”, o que não caracteriza similaridade fonética entre esses termos. Diante disso, novamente pode se afirmar que a identificação de registros duplicados pode ser comprometida, visto que são agrupados valores que não caracterizam similaridade entre as tuplas.

Resultados da Busca

2 - descricao	3 - ativo
GARI	S
GARIBADOR	S
CICLISTA	S
MOTOCICLISTA	S
CARVOEIRO(A)	S
CHOPEIRO(A)	S
AUXILIAR DE DIRETORIA	S
AUXILIAR DENTISTA	S
AUXILIAR DE DEPOSITO	S

Figura 4.11 Busca automática na tabela ocupação utilizando o algoritmo *PSI*.

Com a utilização do Algoritmo *PSI*, é possível observar na Figura 4.11 resultados mais precisos quando se busca identificar registros duplicados. Dos 5.092 resultados, é possível notar nos termos apresentados uma similaridade fonética entre “Gari” e “Garibador”, e “Ciclista” e “Motociclista”, o que mostra que a busca não é restrita a apenas uma parte da palavra. Com a palavra “Chopeiro”, não houve identificação de outro termo similar na base de dados, conforme apresentado na figura, diferente do que foi apresentado pelo Algoritmo *Soundex*. Além disso, vale ressaltar que a busca pelo tipo de pesquisa automático restringe a transcrição fonética em dois termos. No caso, a apresentação dos resultados “Auxiliar de Diretoria” e “Auxiliar de Depósito” não foram considerados similares foneticamente. Vale lembrar que este algoritmo elimina preposições, neste caso “de”, para que posteriormente ocorra a transcrição fonética dos termos.

O desempenho da aplicação dos algoritmos *Soundex*, *Metaphone* e *PSI* é apresentado na Figura 4.12.

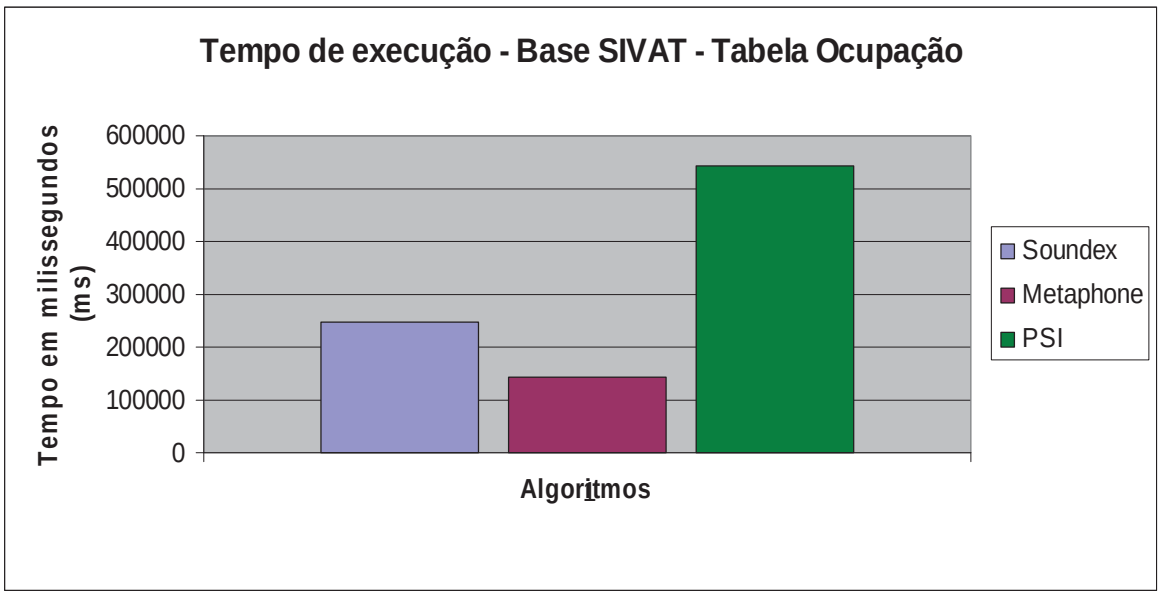
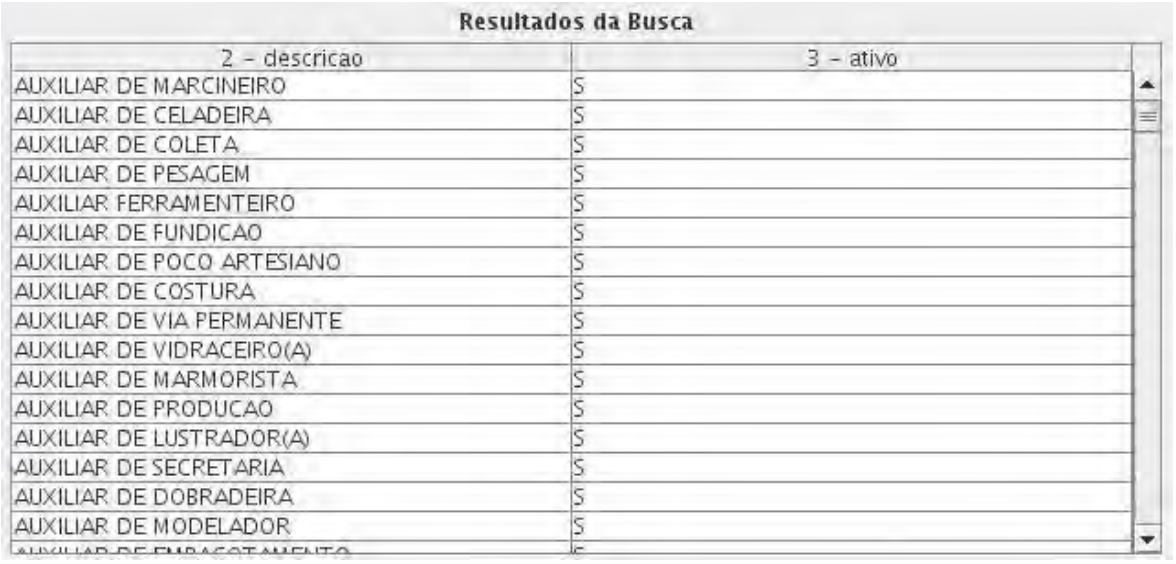


Figura 4.12 Gráfico do tempo de execução – Base SIVAT – Tabela ocupação.

Conforme é apresentado na Figura 4.12, em uma tabela que contém 1.330 registros, o Algoritmo *Soundex* processou as informações em pouco mais de 4 minutos, o Algoritmo *Metaphone* em pouco mais de 2 minutos e o Algoritmo *PSI* em aproximados 9 minutos. Nota-se que o tempo de processamento do Algoritmo *PSI* é maior em relação aos outros, mas os resultados apresentados são melhores, já que identifica apenas aqueles registros que possuem algum grau de similaridade entre eles, diferentemente dos demais algoritmos estudados. Portanto, embora o tempo de processamento seja maior, a compensação está na busca de resultados mais precisos.

4.3.1.4. Experimento 4 – Base SIVAT

O quarto teste foi realizado na base de dados “SIVAT”, tabela “Ocupação” e coluna “Descrição”, executados por meio do “tipo de pesquisa manual”. O termo buscado foi “Auxiliar Produção” e o idioma escolhido foi o “Português”. Os resultados são apresentados nas Figuras 4.13, 4.14 e 4.15.

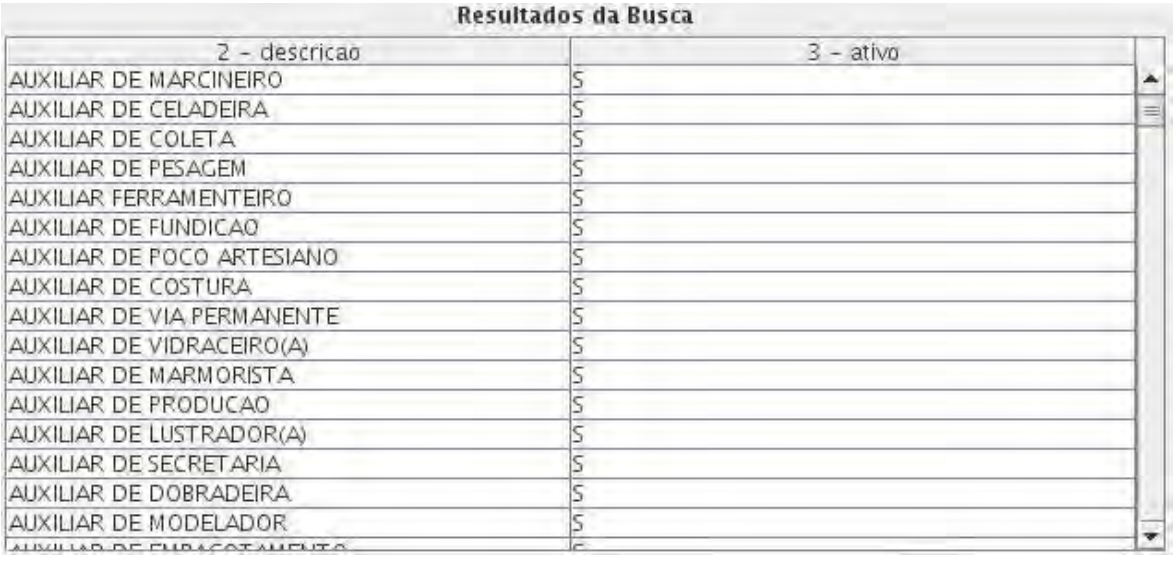


Resultados da Busca

2 - descricao	3 - ativo
AUXILIAR DE MARCINEIRO	S
AUXILIAR DE CELADEIRA	S
AUXILIAR DE COLETA	S
AUXILIAR DE PESAGEM	S
AUXILIAR FERRAMENTEIRO	S
AUXILIAR DE FUNDICAO	S
AUXILIAR DE POCO ARTESIANO	S
AUXILIAR DE COSTURA	S
AUXILIAR DE VIA PERMANENTE	S
AUXILIAR DE VIDRACEIRO(A)	S
AUXILIAR DE MARMORISTA	S
AUXILIAR DE PRODUCAO	S
AUXILIAR DE LUSTRADOR(A)	S
AUXILIAR DE SECRETARIA	S
AUXILIAR DE DOBRADEIRA	S
AUXILIAR DE MODELADOR	S

Figura 4.13 Busca manual na tabela ocupação pelo termo Auxiliar Produção utilizando o algoritmo *Soundex*.

Com a utilização do Algoritmo *Soundex*, foram obtidos 202 resultados. O termo "Auxiliar Produção" gerou uma codificação *soundex* igual a "A246", pois a codificação se restringe apenas ao termo "auxiliar" e, consequentemente, exibe todos os registros que possuem como parte integrante de um termo do banco a palavra "auxiliar", conforme é apresentado na Figura 4.13. Isso torna a busca muito discrepante, não sendo possível identificar registros duplicados por meio do termo de busca pesquisado.



Resultados da Busca

2 - descricao	3 - ativo
AUXILIAR DE MARCINEIRO	S
AUXILIAR DE CELADEIRA	S
AUXILIAR DE COLETA	S
AUXILIAR DE PESAGEM	S
AUXILIAR FERRAMENTEIRO	S
AUXILIAR DE FUNDICAO	S
AUXILIAR DE POCO ARTESIANO	S
AUXILIAR DE COSTURA	S
AUXILIAR DE VIA PERMANENTE	S
AUXILIAR DE VIDRACEIRO(A)	S
AUXILIAR DE MARMORISTA	S
AUXILIAR DE PRODUCAO	S
AUXILIAR DE LUSTRADOR(A)	S
AUXILIAR DE SECRETARIA	S
AUXILIAR DE DOBRADEIRA	S
AUXILIAR DE MODELADOR	S

Figura 4.14 Busca manual na tabela ocupação pelo termo Auxiliar Produção utilizando o algoritmo *Metaphone*.

A mesma explicação do Algoritmo *Soundex* cabe para o Algoritmo *Metaphone*. A diferença é que a codificação *metaphone* da palavra será diferenciada, mas restrita também apenas ao primeiro termo buscado. Foram obtidos 202 resultados extensos e discrepantes,

como “Auxiliar de Celadeira”, “Auxiliar de Costura” e “Auxiliar de Secretaria”, todos identificados como registros duplicados na base de dados, conforme é apresentado na Figura 4.14.

Figura 4.15 Busca manual na tabela ocupação pelo termo Auxiliar Produção utilizando o algoritmo *PSI*.

Com a utilização do Algoritmo *PSI*, é possível obter apenas o resultado pesquisado, conforme é apresentado na Figura 4.15, o que favorece uma busca mais efetiva. Isso acontece devido a transcrição fonética abranger os dois termos pesquisados, ocorrendo a divisão de cada letra para, posteriormente, formar diferentes combinações de palavras a serem buscadas. Neste caso, ocorreram 768 transcrições fonéticas do termo pesquisado, o que permite afirmar que independentemente da forma que a palavra pesquisada tenha sido registrada na base de dados, ela será demonstrada. Portanto, entre as possíveis variações pesquisadas do termo “Auxiliar Produção” cita-se “Aussiliar Produção”, “Auxiliar Produssão”, “Aussiliar Produssão”, entre outros.

Vale ressaltar que, no Algoritmo *PSI*, o tamanho do termo de pesquisa influencia o resultado. Quanto mais informação for fornecida no termo de pesquisa, melhor e mais precisa será a busca, o que favorece a uma busca mais exata.

O desempenho da aplicação dos algoritmos citados é apresentado na Figura 4.16.

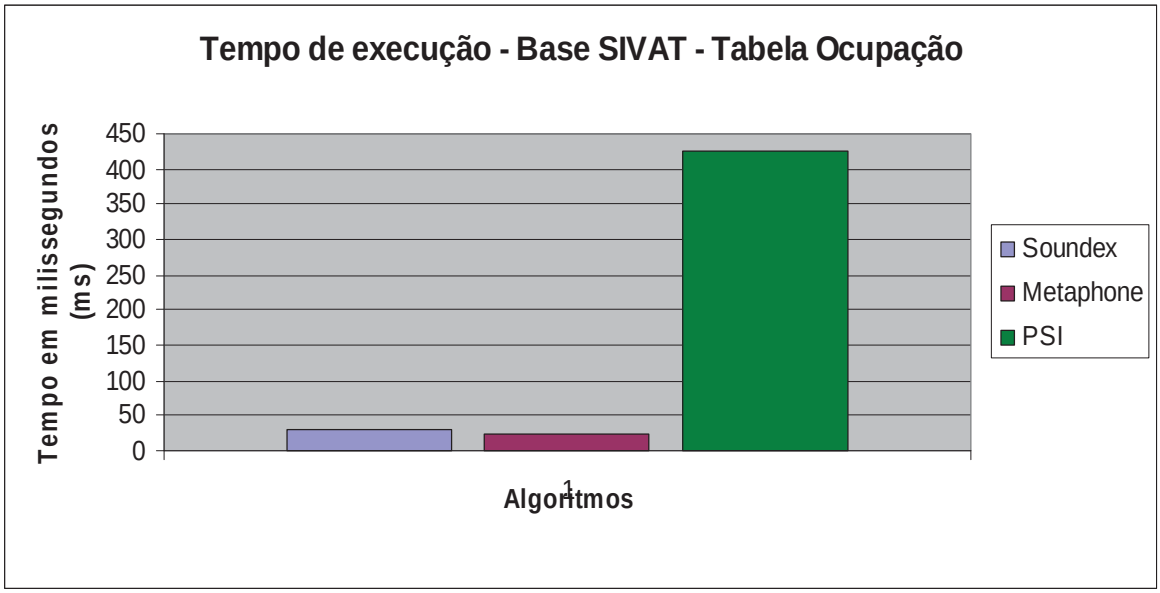
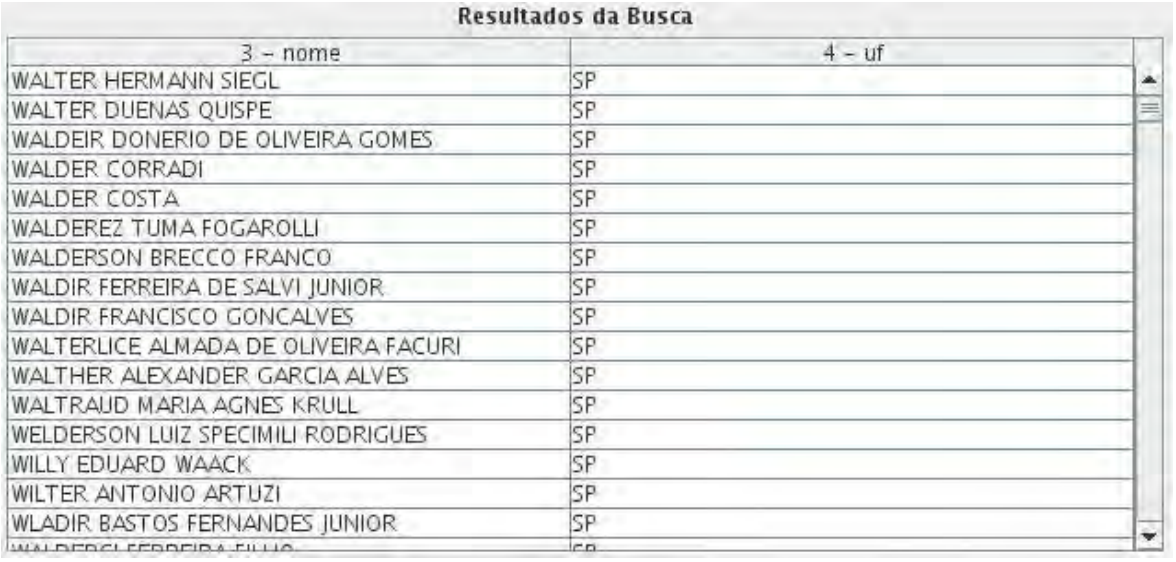


Figura 4.16 Gráfico do tempo de execução – Base SIVAT – Tabela ocupação – Termo Auxiliar Produção.

Conforme é apresentado na Figura 4.16, em uma tabela que contém 1.330 registros, o tempo de execução do Algoritmo *Soundex* foi de 31 milissegundos, o Algoritmo *Metaphone* processou em 24 milissegundos e o Algoritmo *PSI* em 427 milissegundos. O tempo de execução do algoritmo *PSI* é superior em relação aos demais que, também, trabalham com fonética, visto que sua metodologia de funcionamento é diferenciada. Embora seu tempo de trabalho seja maior, isso infere em uma precisão de tempo de pouco menos de meio segundo, o que o torna compensatório ao ser comparado com os resultados obtidos nos outros algoritmos.

4.3.1.5. Experimento 5 – Base SIVAT

O quinto teste foi realizado na base de dados “SIVAT”, tabela “Médico” e coluna “Nome”, executados por meio do “tipo de pesquisa manual”. O termo de busca foi “Walter” e o idioma escolhido foi o “Português”. Os resultados são apresentados nas Figuras 4.17, 4.18 e 4.19.

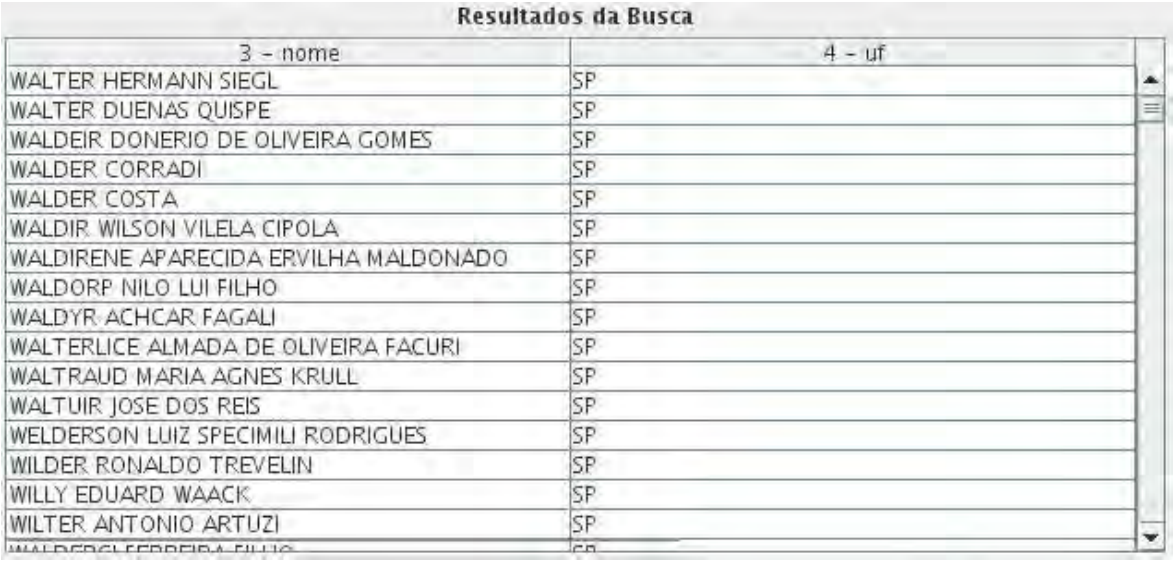


Resultados da Busca

3 - nome	4 - uf
WALTER HERMANN SIEGL	SP
WALTER DUENAS QUISPE	SP
WALDEIR DONERIO DE OLIVEIRA GOMES	SP
WALDER CORRADI	SP
WALDER COSTA	SP
WALDEREZ TUMA FOGAROLLI	SP
WALDERSON BRECCO FRANCO	SP
WALDIR FERREIRA DE SALVI JUNIOR	SP
WALDIR FRANCISCO GONCALVES	SP
WALTERLICE ALMADA DE OLIVEIRA FACURI	SP
WALTHER ALEXANDER GARCIA ALVES	SP
WALTRAUD MARIA AGNES KRULL	SP
WELDERSON LUIZ SPECIMILI RODRIGUES	SP
WILLY EDUARD WAACK	SP
WILTER ANTONIO ARTUZI	SP
WLADIR BASTOS FERNANDES JUNIOR	SP

Figura 4.17 Busca manual na tabela médico pelo termo Walter utilizando o algoritmo *Soundex*.

Conforme é apresentado na Figura 4.17, a busca pelo termo “Walter” com o algoritmo *Soundex* obteve 495 resultados. A codificação originada foi “W436” e foi possível apurar como resultado além do próprio termo de busca nomes muito discrepantes, como “Walderson”, “Waltraud”, “Wilter”, entre outros, pois o Algoritmo *Soundex* ignora a presença das vogais nas palavras e utiliza codificação similar em algumas consoantes. Observa-se que foram demonstrados apenas resultados de palavras que começam com a consoante “W”, pois não há variação fonética nessa consoante ao utilizar este algoritmo.



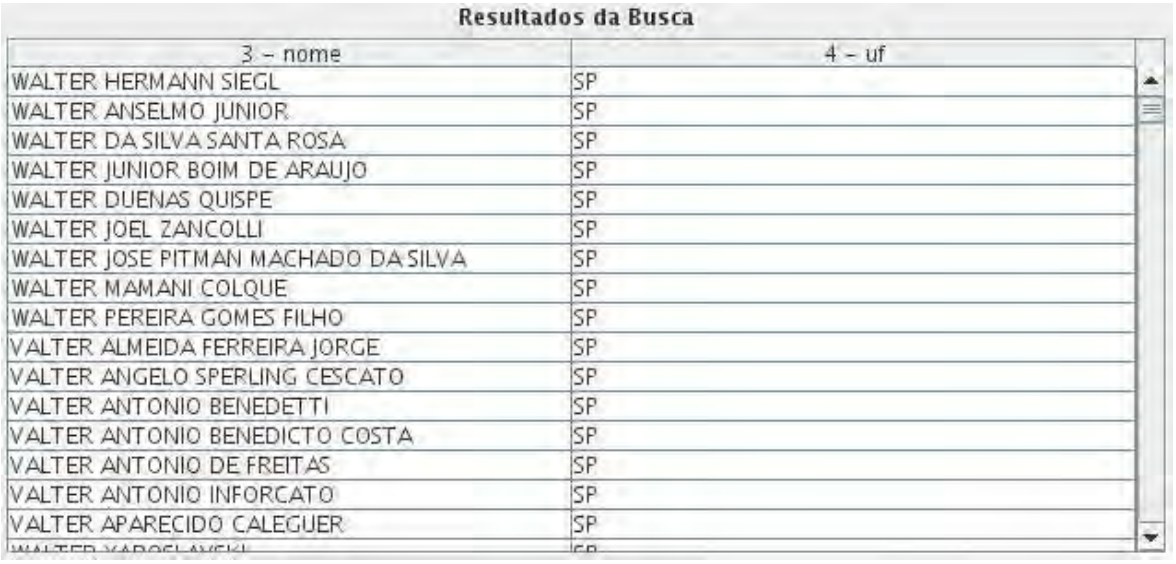
Resultados da Busca

3 - nome	4 - uf
WALTER HERMANN SIEGL	SP
WALTER DUENAS QUISPE	SP
WALDEIR DONERIO DE OLIVEIRA GOMES	SP
WALDER CORRADI	SP
WALDER COSTA	SP
WALDIR WILSON VILELA CIPOLA	SP
WALDIRENE APARECIDA ERVILHA MALDONADO	SP
WALDORP NILO LUI FILHO	SP
WALDYR ACHCAR FAGALI	SP
WALTERLICE ALMADA DE OLIVEIRA FACURI	SP
WALTRAUD MARIA AGNES KRULL	SP
WALTUIR JOSE DOS REIS	SP
WELDERSON LUIZ SPECIMILI RODRIGUES	SP
WILDER RONALDO TREVELIN	SP
WILLY EDUARD WAACK	SP
WILTER ANTONIO ARTUZI	SP

Figura 4.18 Busca manual na tabela médico pelo termo Walter utilizando o algoritmo *Metaphone*.

De acordo com a Figura 4.18, ao utilizar o Algoritmo *Metaphone* na busca manual pelo nome “Walter”, a codificação originada foi “WLTR” e exibiu 482 resultados. Além

de resultados foneticamente similares com termo de busca, nomes discrepantes, também, foram apresentados, como “Waldeir”, “Waldorp”, “Wilder”, entre outros, o que aponta resultados muito abrangentes e inexatos, assim como na aplicação do Algoritmo *Soundex*, o que desfavorece uma busca precisa na identificação de registros duplicados.



Resultados da Busca

3 – nome	4 – uf
WALTER HERMANN SIEGL	SP
WALTER ANSELMO JUNIOR	SP
WALTER DA SILVA SANTA ROSA	SP
WALTER JUNIOR BOIM DE ARAUJO	SP
WALTER DUENAS QUISPE	SP
WALTER JOEL ZANCOLLI	SP
WALTER JOSE PITMAN MACHADO DA SILVA	SP
WALTER MAMANI COLQUE	SP
WALTER PEREIRA GOMES FILHO	SP
VALTER ALMEIDA FERREIRA JORGE	SP
VALTER ANGELO SPERLING CESCATO	SP
VALTER ANTONIO BENEDETTI	SP
VALTER ANTONIO BENEDICTO COSTA	SP
VALTER ANTONIO DE FREITAS	SP
VALTER ANTONIO INFORCATO	SP
VALTER APARECIDO CALEGUER	SP

Figura 4.19 Busca manual na tabela médico pelo termo Walter utilizando o algoritmo *PSI*.

Conforme é apresentado na Figura 4.19, com a utilização do Algoritmo *PSI* na busca manual pelo nome “Walter” foram realizadas 32 transcrições e demonstrados apenas os registros que continham literalmente o termo de busca transcrito foneticamente, que varia entre “Walter” e “Valter”. Embora, foram apurados 548 resultados, com a utilização deste algoritmo é possível obter resultados mais precisos, não discrepantes do termo de busca como os obtidos na utilização dos outros dois algoritmos exemplificados. Neste caso, foneticamente, a consoante “w” pode assumir os valores “w” e “v”, a vogal “a” pode assumir o valor “a” ou “ã”, a consoante “l” pode assumir o valor “l” ou “u”, a consoante “t” apenas o seu próprio valor, a vogal “e” pode assumir o valor “e”, “é”, “ẽ” ou “i”, e por último a consoante “r” pode assumir apenas o seu valor o que pode ser visto nas ocorrências das palavras conforme apresentados na Figura 4.19.

O desempenho da aplicação dos algoritmos citados é apresentado na Figura 4.20.

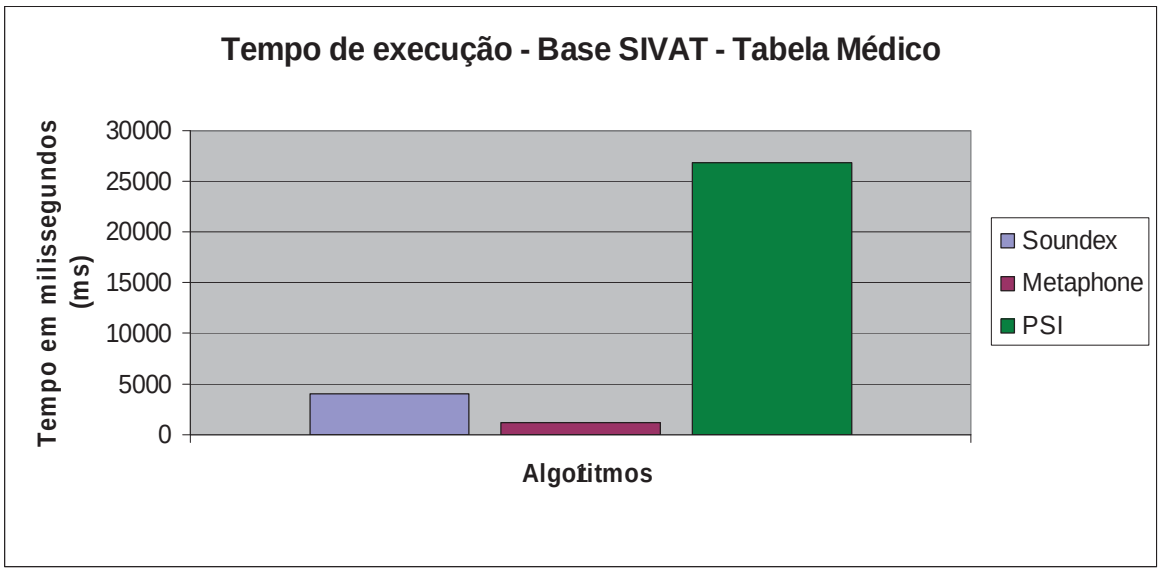
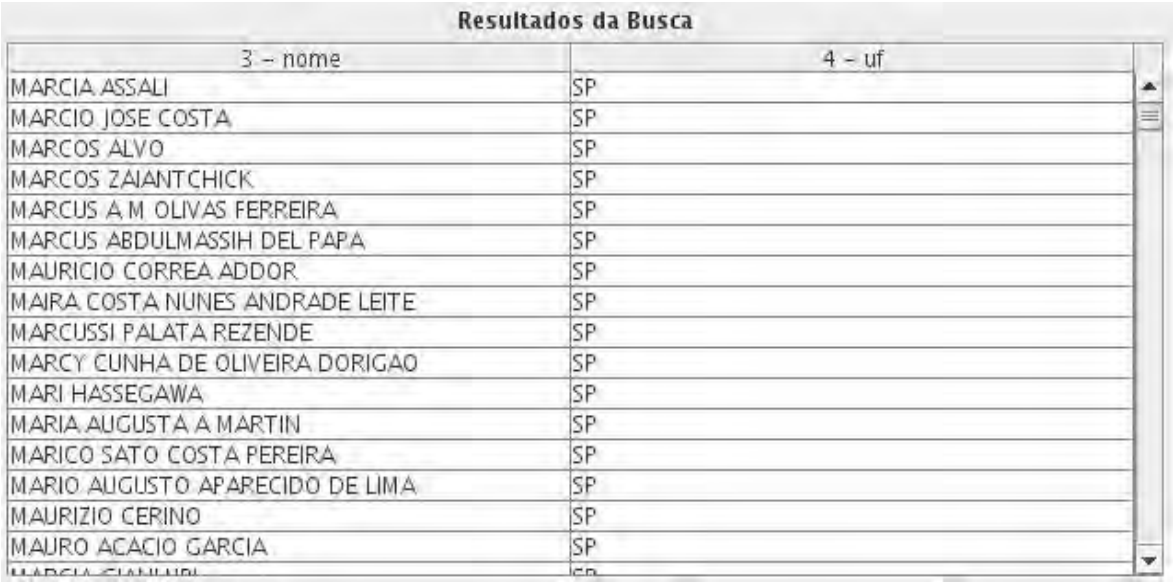


Figura 4.20 Gráfico do tempo de execução – Base SIVAT – Tabela médico – Termo Walter.

Conforme apresentado na Figura 4.20, o tempo de execução do Algoritmo *PSI* é maior em relação aos algoritmos *Soundex* e *Metaphone*, pois para cada letra do termo de busca "Walter" foi realizada uma transcrição fonética, que resulta em 32 transcrições o que, consequentemente, demanda tempo de processamento. Isso justifica um consumo maior do tempo de execução em relação aos demais algoritmos, pois *Soundex* e *Metaphone* trabalham com um número restrito de codificação do dado de entrada e dos registros do banco. Em uma tabela que contém 141.375 registros, o Algoritmo *Soundex* processou essas informações em aproximados 4 segundos, o Algoritmo *Metaphone* em pouco mais de 1 segundo, e o Algoritmo *PSI* em 26 segundos.

4.3.1.6. Experimento 6 – Base SIVAT

O sexto teste foi realizado na base de dados "SIVAT", tabela "Médico" e coluna "Nome", executados por meio do "tipo de pesquisa manual". O termo de busca foi "Marcos" e o idioma escolhido foi o "Português". Os resultados são apresentados nas Figuras 4.21, 4.22 e 4.23.

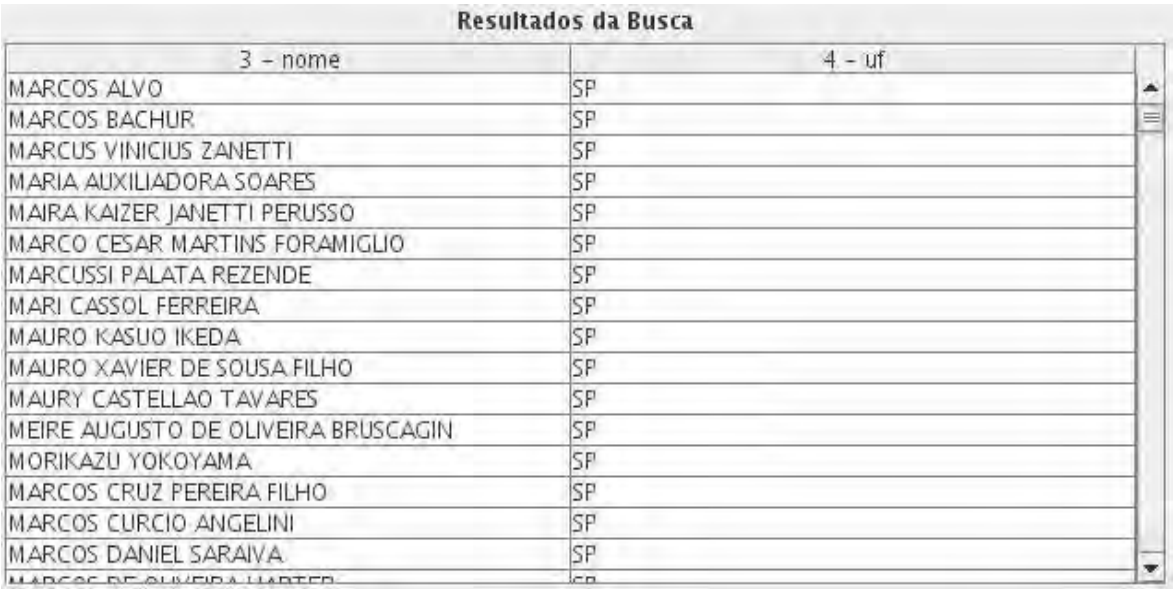


Resultados da Busca

3 - nome	4 - uf
MARCIA ASSALI	SP
MARCIO JOSE COSTA	SP
MARCOS ALVO	SP
MARCOS ZAIANTCHICK	SP
MARCUS A M OLIVAS FERREIRA	SP
MARCUS ABDULMASSIH DEL PAPA	SP
MAURICIO CORREA ADDOR	SP
MAIRA COSTA NUNES ANDRADE LEITE	SP
MARCUSSI PALATA REZENDE	SP
MARCY CUNHA DE OLIVEIRA DORIGAO	SP
MARI HASSEGAWA	SP
MARIA AUGUSTA A MARTIN	SP
MARICO SATO COSTA PEREIRA	SP
MARIO AUGUSTO APARECIDO DE LIMA	SP
MAURIZIO CERINO	SP
MAURO ACACIO GARCIA	SP

Figura 4.21 Busca manual na tabela médico pelo termo Marcos utilizando o algoritmo *Soundex*.

Com a utilização do Algoritmo *Soundex*, foram apresentados 2.460 resultados, mediante a codificação originada “M622”. De acordo com a Figura 4.21, resultados discrepantes como “Mauricio”, “Marcy” e “Mauro” foram apresentados, o que não caracteriza similaridade fonética entre esses termos registrados. Isso implica na problemática de identificação de registros duplicados na base.



Resultados da Busca

3 - nome	4 - uf
MARCOS ALVO	SP
MARCOS BACHUR	SP
MARCUS VINICIUS ZANETTI	SP
MARIA AUXILIADORA SOARES	SP
MAIRA KAIZER JANETTI PERUSSO	SP
MARCO CESAR MARTINS FORAMIGLIO	SP
MARCUSSI PALATA REZENDE	SP
MARI CASSOL FERREIRA	SP
MAURO KASUO IKEDA	SP
MAURO XAVIER DE SOUSA FILHO	SP
MAURY CASTELLAO TAVARES	SP
MEIRE AUGUSTO DE OLIVEIRA BRUSCAGIN	SP
MORIKAZU YOKOYAMA	SP
MARCOS CRUZ PEREIRA FILHO	SP
MARCOS CURCIO ANGELINI	SP
MARCOS DANIEL SARAIVA	SP

Figura 4.22 Busca manual na tabela médico pelo termo Marcos utilizando o algoritmo *Metaphone*.

De acordo com o que é apresentado na Figura 4.22, o Algoritmo *Metaphone* apresentou 1.459 resultados, com características discrepantes entre eles, como os registros “Maria”, “Morikazu” e “Maury”, entre outros. Resultados foneticamente similares também

foram encontrados, como “Marcos” e “Marcus”, mas que retornados junto aos discrepantes torna a busca pela identificação de registros duplicados complicado e com grande demanda de tempo.

Resultados da Busca

3 - nome	4 - uf
MARCOS ALBERTO PAGANI JUNIOR	SP
MARCOS ANTONIO DE FARIAS LIRA JUNIOR	SP
MARCOS APARECIDO DE TOLEDO JUNIOR	SP
MARCOS ARAUJO CHAVES JUNIOR	SP
MARCOS BARBOSA DE SOUZA JUNIOR	SP
MARCOS CARDOSO BENHAMI	SP
MARCOS CRUZ PEREIRA FILHO	SP
MARCOS CURCIO ANGELINI	SP
MARCUS FURTADO DE ANDRADE	SP
MARCUS PICORAL PINTO	SP
MARCUS TAVER COSTA DANTAS	SP
MARCUS VINICIUS AMARAL BARRETO	SP
MARIO MARCOS GUIMARAES ABEID	SP
MARKUS ANDRET CAVALCANTE GIFONI	SP
MARKUS MAX HOFSTETTER	SP
MIRIAM GONCALVES MARKOS KAWABATA	SP

Figura 4.23 Busca manual na tabela médico pelo termo Marcos utilizando o algoritmo *PSI*.

Com a utilização do algoritmo *PSI*, foram apresentados 1.569 registros que contém similaridade fonética entre os termos, conforme a Figura 4.23. Foram realizadas 64 transcrições fonéticas, e os resultados foram mais precisos se comparados aos demais algoritmos, como “Marcos”, “Marcus”, “Markus” e “Markos”. Os dois últimos citados só foram apresentados na utilização do Algoritmo *PSI*, pois não identifica similaridade entre as consoantes “c” e “k”. Portanto, para a identificação de registros duplicados, a utilização deste algoritmo tornaria a busca mais exata.

O desempenho da aplicação entre os Algoritmos *Soundex*, *Metaphone* e *PSI* é apresentado na Figura 4.24.

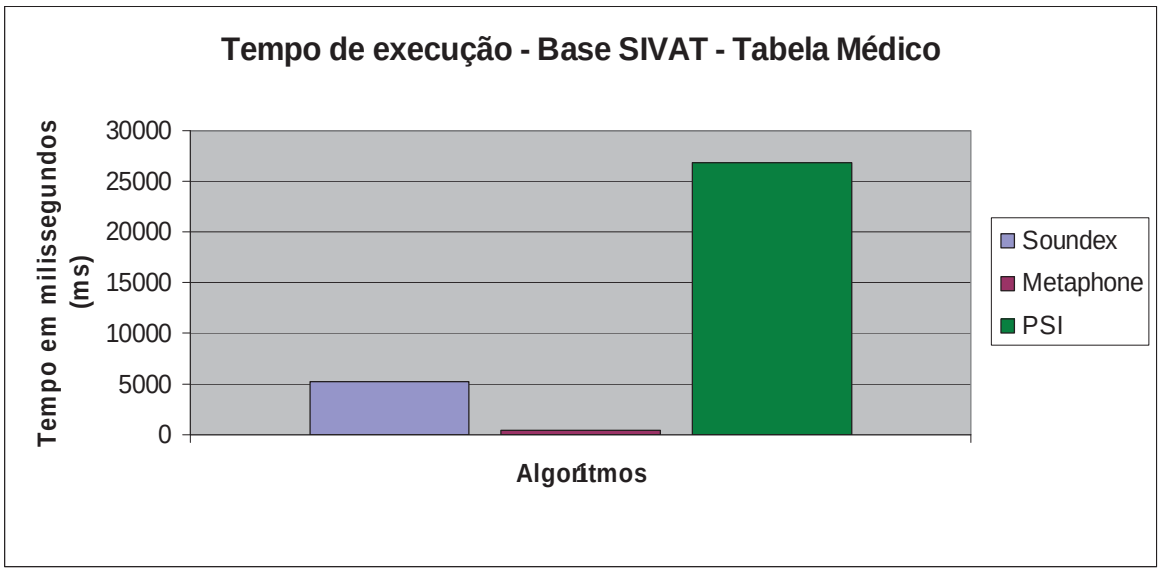
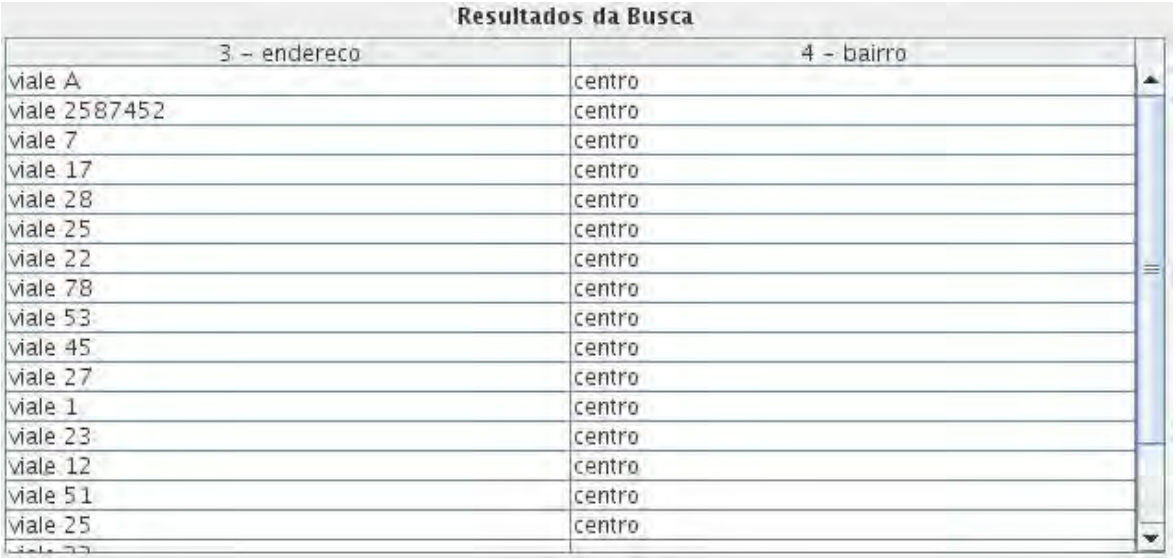


Figura 4.24 Gráfico do tempo de execução – Base SIVAT – Tabela médico – Termo Marcos.

Conforme é apresentado na Figura 4.24, em uma tabela que contém 141.375 registros, o Algoritmo *Soundex* foi executado em pouco mais de 5 segundos, o Algoritmo *Metaphone* em menos de 1 segundo e o Algoritmo *PSI* em aproximados 26 segundos. O tempo de execução do Algoritmo *PSI* é muito maior em relação aos demais algoritmos, visto que a funcionalidade deste algoritmo tem mais etapas a serem executadas, mas que reflete em uma busca mais exata, o que favorece a identificação de registros duplicados. Portanto, na busca pelo termo "Marcos", foi realizada 64 transcrições fonéticas, o que demanda um tempo muito maior de processamento.

4.3.1.7. Experimento 7 – Base FUNCIONÁRIOS

O sétimo teste foi realizado na base de dados "Funcionários", tabela "Inscritos" e coluna "Endereço", executados por meio do "tipo de pesquisa manual" para demonstração de uma pesquisa por meio de dados numéricos. O termo de busca foi "Viale 7 (*avenida* 7)" e o idioma escolhido foi o "Italiano". Os resultados são apresentados nas Figuras 4.25, 4.26 e 4.27.

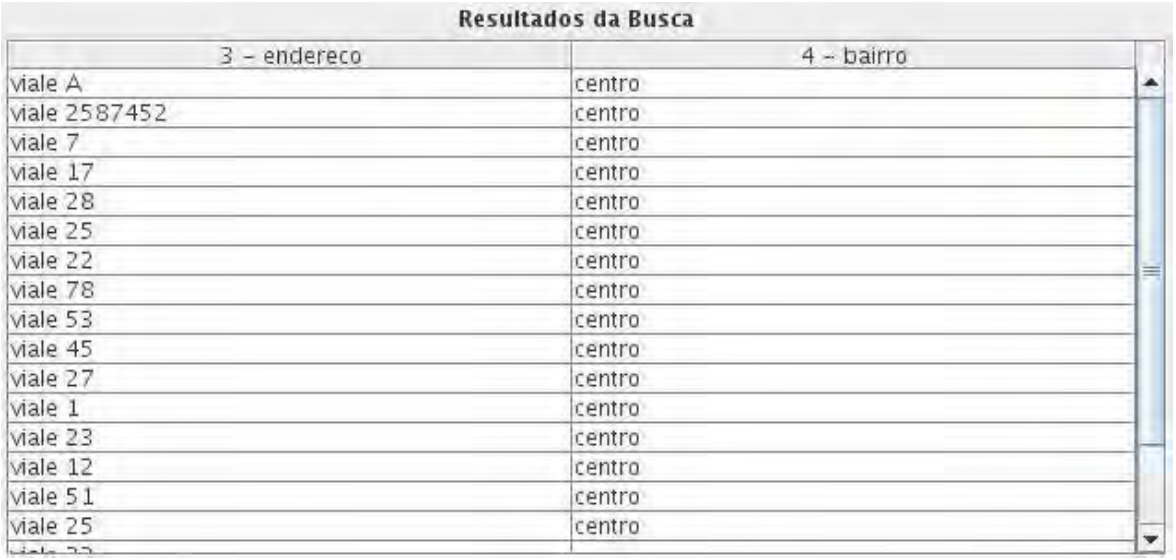


Resultados da Busca

3 – endereco	4 – bairro
viale A	centro
viale 2587452	centro
viale 7	centro
viale 17	centro
viale 28	centro
viale 25	centro
viale 22	centro
viale 78	centro
viale 53	centro
viale 45	centro
viale 27	centro
viale 1	centro
viale 23	centro
viale 12	centro
viale 51	centro
viale 25	centro

Figura 4.25 Busca manual na tabela inscritos pelo termo viale 7 utilizando o algoritmo *Soundex*.

Conforme é apresentado na Figura 4.25, com a utilização do Algoritmo *Soundex*, dos 100 registros na base de dados foram encontrados os 20 que contém como parte integrante a palavra “viale”, independente se é acompanhada ou não de um numeral, o que desfavorece a busca exata. É possível observar por meio dos registros apresentados que o Algoritmo *Soundex* opera somente sobre as letras do termo de pesquisa, desprezando os caracteres numéricos. Neste caso, se tiver 1000 registros na base de dados que contenha a palavra “viale”, serão apresentados esses 1000 registros como resultados, pois desconsidera uma parte importante do termo de pesquisa que é o numeral “7”. Portanto, o caractere “7” que faz parte do termo de pesquisa é desprezado, o que ocasiona um resultado abrangente que compromete a identificação de registros duplicados.



Resultados da Busca

3 – endereco	4 – bairro
viale A	centro
viale 2587452	centro
viale 7	centro
viale 17	centro
viale 28	centro
viale 25	centro
viale 22	centro
viale 78	centro
viale 53	centro
viale 45	centro
viale 27	centro
viale 1	centro
viale 23	centro
viale 12	centro
viale 51	centro
viale 25	centro

Figura 4.26 Busca manual na tabela inscritos pelo termo viale 7 utilizando o algoritmo *Metaphone*.

A mesma explicação na utilização do algoritmo *Soundex* pode ser utilizado para o algoritmo *Metaphone*. Os 20 resultados apresentados nas Figuras 4.25 e 4.26 são os mesmos, visto que este algoritmo também desconsidera os termos numéricos da pesquisa, o que favorece a apresentação de registros muito abrangentes e fora dos parâmetros buscados.

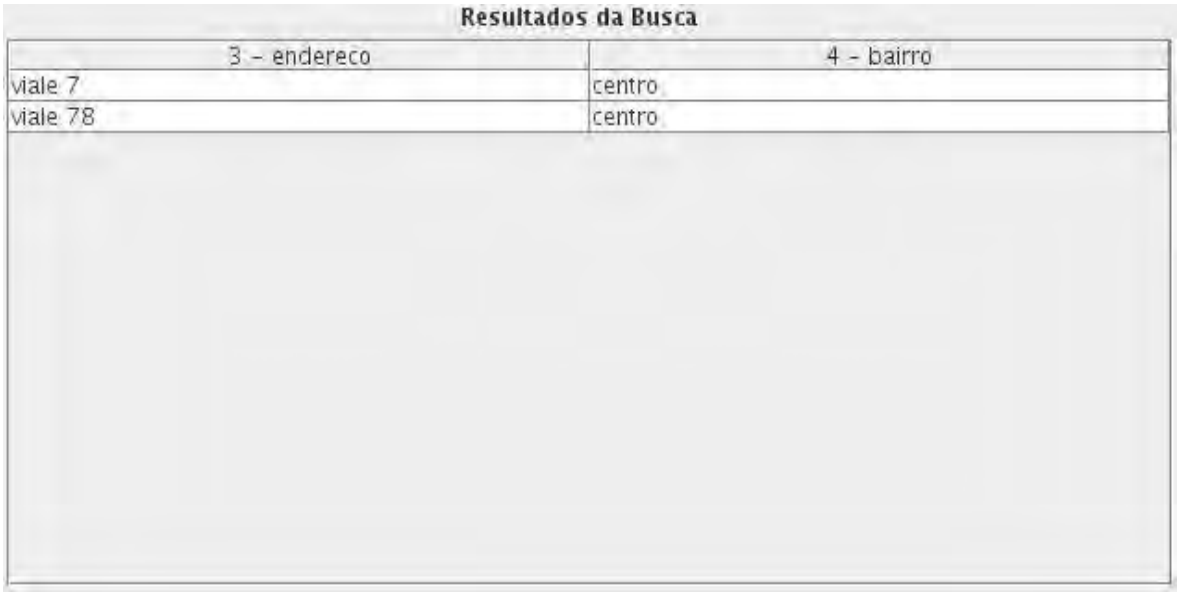


Figura 4.27 Busca manual na tabela inscritos pelo termo viale 7 utilizando o algoritmo *PSI*.

Conforme é apresentado na Figura 4.27, com a utilização do algoritmo *PSI*, os resultados obtidos por meio do termo de pesquisa “viale 7” foram mais restritos, visto que o algoritmo trabalha com os caracteres numéricos e letras. Neste caso, o algoritmo realizou a transcrição fonética do termo de pesquisa e apresentou os seguintes valores a serem buscados: “viale 7” e “vialé 7”. Diante dessas duas transcrições, os resultados da busca foram mais objetivos, pois só foram encontrados valores que contém a palavra transcrita foneticamente “viale” seguida pelo numeral “7”. Nota-se que estas transcrições realizadas podem fazer parte em qualquer posição dos registros do banco de dados, variando-se entre o inicio, meio e fim, como nos resultados apresentados “viale 7” e “viale 78”. Ressalta-se que neste algoritmo os números são tratados como *string* e não como um valor numérico.

O desempenho da aplicação entre os algoritmos é apresentado na Figura 4.28.

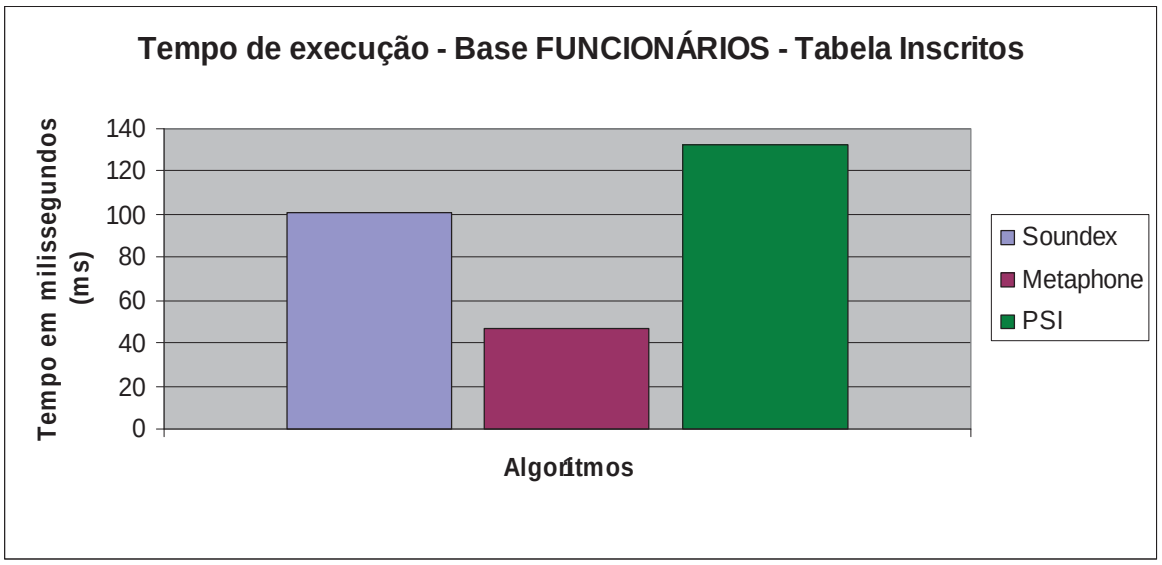
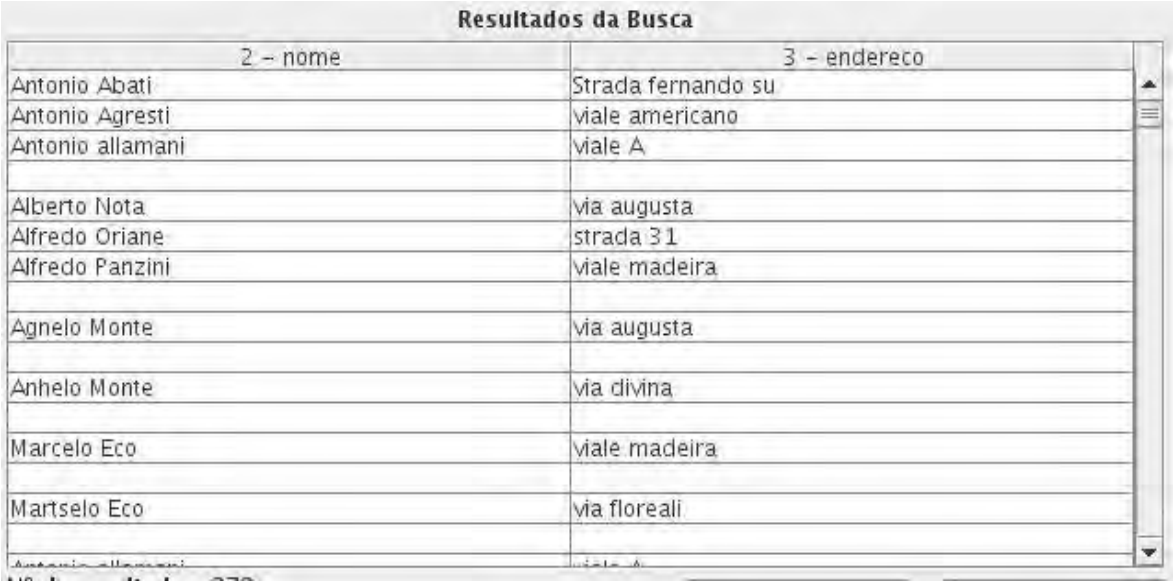


Figura 4.28 Gráfico do tempo de execução – Base FUNCIONÁRIOS – Tabela inscritos – Termo Viale 7.

De acordo com a Figura 4.28, em uma tabela que contém 100 registros, o Algoritmo *Soundex* processou as informações em 101 milissegundos, o Algoritmo *Metaphone* em 47 milissegundos e o Algoritmo *PSI* em 133 milissegundos. Esses três algoritmos operaram em menos de 1 segundo de tempo. O tempo de execução do Algoritmo *PSI* é maior em relação aos demais algoritmos, mas reflete em uma exatidão de resultados maior por trabalhar com números além de letras, o que favorece a identificação de registros duplicados.

4.3.1.8. Experimento 8 – Base FUNCIONÁRIOS

O oitavo e último teste foi realizado na base de dados “Funcionários”, tabela “Inscritos” e coluna “Nome”, executados por meio do “tipo de pesquisa automático” e idioma “Italiano”. Os resultados são apresentados nas Figuras 4.29, 4.30 e 4.31.

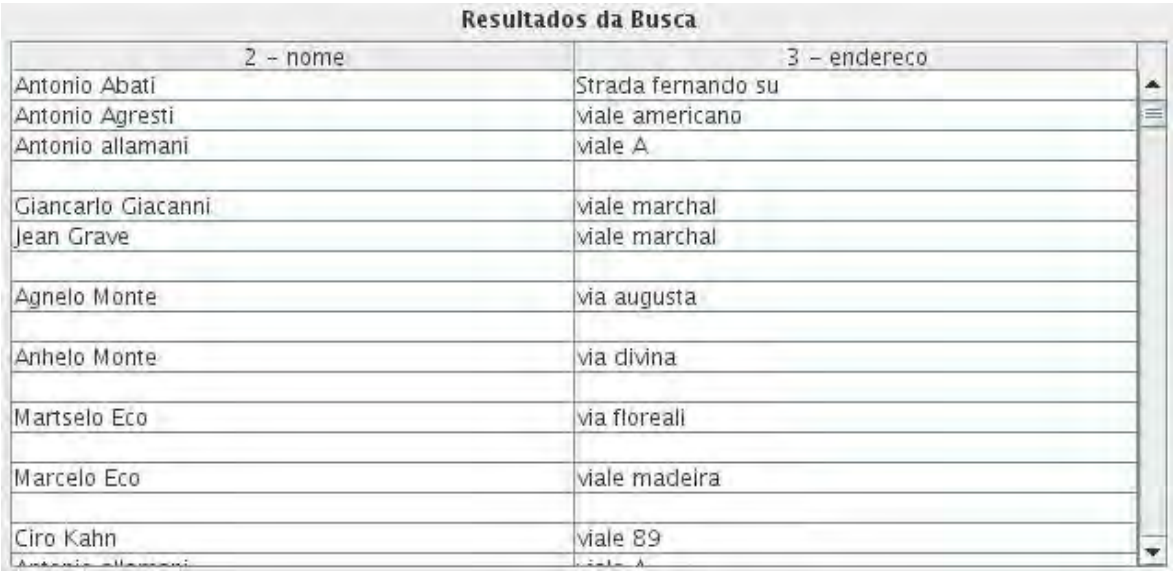


Resultados da Busca

2 – nome	3 – endereco
Antonio Abati	Strada fernando su
Antonio Agresti	viale americano
Antonio allamani	viale A
Alberto Nota	via augusta
Alfredo Oriane	strada 31
Alfredo Panzini	viale madeira
Agnelo Monte	via augusta
Anhelo Monte	via divina
Marcelo Eco	viale madeira
Martselo Eco	via floreali

Figura 4.29 Busca automática na tabela inscritos utilizando o algoritmo *Soundex*.

De acordo com a Figura 4.29, o Algoritmo *Soundex* encontrou 272 resultados. Palavras como “”Antonio Abati”, “Antonio Agresti” foram identificados como similares foneticamente, assim como “Alfredo Oriane” e “Alberto Nota”. Isso mostra uma restrição na codificação somente aos primeiros termos de busca, pois forma um código de até quatro caracteres que o algoritmo utiliza. Conseqüentemente, os resultados são abrangentes, não caracterizando similaridade fonética entre todos as palavras dos termos apresentados, o que dificulta a análise na identificação de registros duplicados. Nota-se que os termos “Agnelo Monte” e “Anhelo Monte” não são identificados como similares foneticamente, mas que de acordo com a língua italiana são considerados. O mesmo acontece com os termos “Marcelo Eco” e “Martselo Eco”, que possuem similaridade fonética na língua italiana.

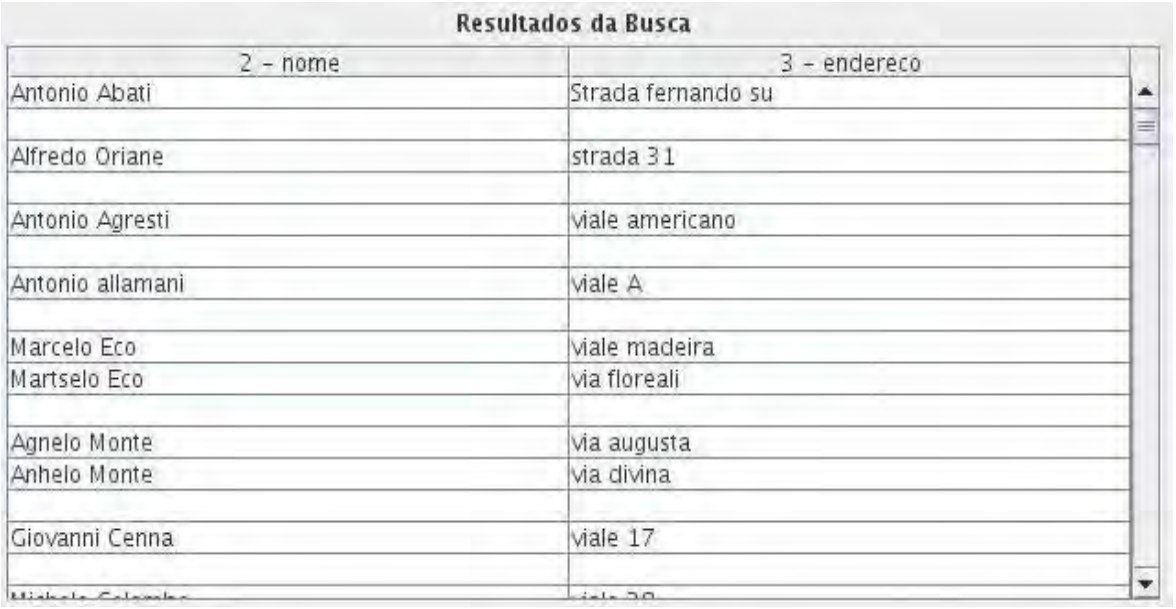


Resultados da Busca

2 – nome	3 – endereco
Antonio Abati	Strada fernando su
Antonio Agresti	viale americano
Antonio allamani	viale A
Giancarlo Giacanni	viale marchal
Jean Grave	viale marchal
Agnelo Monte	via augusta
Anhelo Monte	via divina
Martselo Eco	via floreali
Marcelo Eco	viale madeira
Ciro Kahn	viale 89

Figura 4.30 Busca automática na tabela inscritos utilizando o algoritmo *Metaphone*.

Conforme é apresentado na Figura 4.30, com a utilização do Algoritmo *Metaphone* foram apresentados 254 resultados. A mesma explicação na utilização do algoritmo *Soundex* pode ser utilizado para o algoritmo *Metaphone*, que identificou os termos compostos pela palavra “Antonio” como foneticamente similares, e não identificou os termos “Agnelo Monte” e “Anhelo Monte, e “Martselo Eco” e “Marcelo Eco” como similares foneticamente. Uma observação importante cabe aos termos identificados “Giancarlo Giacanni” e “Jean Grave”, que somente neste algoritmo foram identificados como resultados com características fonéticas similares.



Resultados da Busca

2 - nome	3 - endereco
Antonio Abati	Strada fernando su
Alfredo Oriane	strada 31
Antonio Agresti	viale americano
Antonio allamani	viale A
Marcelo Eco	viale madeira
Martselo Eco	via floreali
Agnelo Monte	via augusta
Anhelo Monte	via divina
Giovanni Cenna	viale 17

Figura 4.31 Busca automática na tabela inscritos utilizando o algoritmo *PSI*.

Com a utilização do algoritmo *PSI*, foram apresentados 193 registros. Termos identificados como foneticamente similares nos algoritmos *Soundex* e *Metaphone* não foram identificados como tal neste algoritmo, como é apresentado na Figura 4.31. Como exemplo, nota-se a separação dos termos “Antonio Abati” e “Antonio Agresti”, assim como o termo “Alfredo Oriane” que é retornado isoladamente. Nota-se que com a utilização deste algoritmo, os termos “Marcelo Eco” e “Martselo Eco”, assim como “Agnelo Monte” e “Anhelo Monte”, já foram identificados como foneticamente similares, obedecendo as características da fonética da língua italiana, conforme as suas transcrições. Portanto, neste caso o Algoritmo *PSI* mostra-se mais eficiente do que o *Soundex* e *Metaphone* com a apresentação de resultados mais precisos, favorecendo a identificação de registros duplicados na base de dados conforme as características do idioma escolhido.

O desempenho da aplicação entre os três algoritmos é apresentado na Figura 4.32.

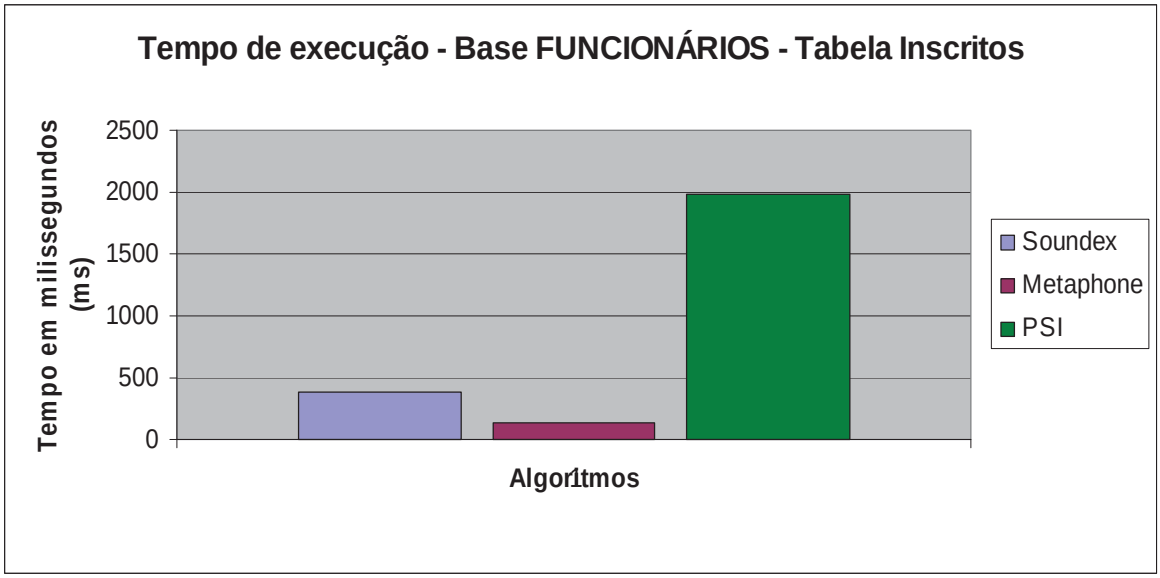


Figura 4.32 Gráfico do tempo de execução – Base FUNCIONÁRIOS – Tabela inscritos.

Conforme é apresentado na Figura 4.32, na tabela “inscritos” que possui 100 registros armazenados, o Algoritmo *Soundex* foi executado em 386 milissegundos, o Algoritmo *Metaphone* em 135 milissegundos e o Algoritmo *PSI* em 1.983 milissegundos. É possível observar que, tanto o Algoritmo *Soundex* quanto o *Metaphone* processaram as informações e listaram os resultados em menos de 1 segundo. Já o *PSI* realizou esta mesma tarefa em aproximados 2 segundos, o que comprova também um excelente tempo de execução, além de um melhor retorno dos resultados em termos de precisão comparado aos demais algoritmos.

4.3.2. Estudo comparativo do Algoritmo *PSI* com a utilização de *threads*

Nesta seção é abordada a utilização de *multithreading* na execução do Algoritmo *PSI*, a fim de constatar uma melhora de desempenho do mesmo. A utilização de *threads* neste trabalho tem funcionalidades diferentes, conforme a escolha do tipo de pesquisa do usuário, que pode variar entre automática e manual.

No “tipo de pesquisa automática”, cada *thread* atua sobre um registro da tabela e o compara com os demais. Para exemplificar este procedimento, considere uma tabela contendo 1.000 registros e um computador com dois núcleos de processamento. A primeira *thread* lê o primeiro registro da tabela, transcreve-o foneticamente e compara as transcrições com os demais registros. A segunda *thread* atua sobre o segundo registro, transcrevendo-o foneticamente e comparando com os demais registros da tabela, inclusive

com os registros antecessores. Para o caso de duas *threads*, o passo de iteração será a cada dois registros, sendo que na próxima execução a primeira *thread* atuará sobre o terceiro registro e a segunda *thread* atuará sobre o quarto registro.

No “tipo de pesquisa manual”, primeiramente é lido o número total de registros da tabela pesquisada para que, posteriormente, seja dividida essa quantidade conforme for o número de *threads* que foram criadas. Diante disso, cada *thread* atua em um número limitado de registros, sendo que há um sincronismo entre elas para a escrita na lista de resultados, pois se uma *thread* estiver listando, a outra aguarda para realizar a escrita. Para exemplificar, consideremos uma tabela contendo 1.000 registros e o computador que é executado o Algoritmo PSI contém dois núcleos de processadores. Portanto, serão criadas duas *threads*, em que cada uma é responsável pela busca das transcrições fonéticas em 500 registros dessa tabela.

Nota-se que a funcionalidade aplicada em *threads* permite a distribuição de tarefas entre os núcleos do processador. Esta divisão garante um ganho significativo no desempenho do algoritmo visto que, sem a utilização da mesma, os demais núcleos ficariam ociosos.

Para a execução dos testes foi utilizado o computador mencionado no capítulo 4.3 deste trabalho. A título de informação, vale ressaltar que o número de resultados não são apresentados nos experimentos a seguir, visto que são os mesmos obtidos com ou sem o uso de *threads* na execução do Algoritmo *PSI*, e que foram apresentados nos experimentos contidos no subcapítulo 4.3.1.

4.3.2.1. Experimento 1 – Base SIVAT – Uso de Threads

O primeiro teste realizado com a utilização de *threads* foi aplicado na base de dados “SIVAT”, tabela “Maquina Causadora” e coluna “Nome”, executados por meio do “tipo de pesquisa automático” e o idioma “Português”. O gráfico de desempenho com o uso de *threads* é apresentado na Figura 4.33.

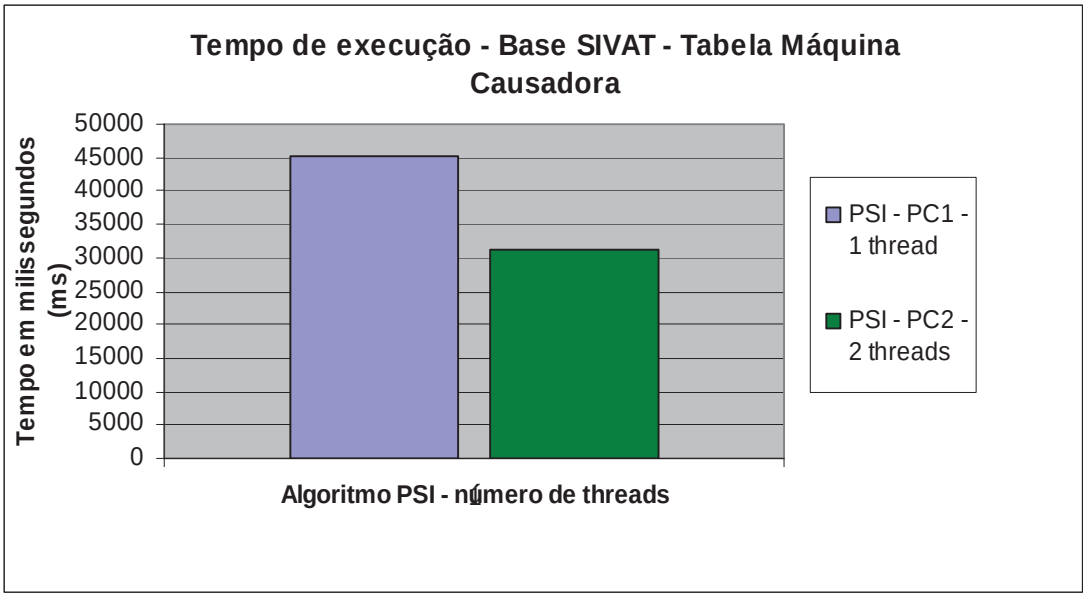


Figura 4.33 Gráfico do tempo de execução com o uso de *threads*– Base SIVAT – Tabela Máquina Causadora

De acordo com o gráfico de desempenho apresentado na Figura 4.33, o tempo de processamento do algoritmo *PSI* utilizando um núcleo de processador foi maior em relação a dois núcleos de processadores. Nota-se que com um núcleo de processador a execução demorou aproximados 45 segundos, enquanto que com dois núcleos demorou em torno de 31 segundos. A utilização dos dois núcleos favorece um desempenho de tempo melhor na busca pelos resultados, pois a divisão do processamento torna o algoritmo mais ágil.

4.3.2.2. Experimento 2 – Base SIVAT – Uso de Threads

O segundo teste realizado com a utilização de *threads* foi aplicado na base de dados “SIVAT”, tabela “Ramo atividade” e coluna “Descrição”, executados por meio do “tipo de pesquisa automático” e o idioma “Português”. O gráfico de desempenho com o uso de *threads* é apresentado na Figura 4.34.

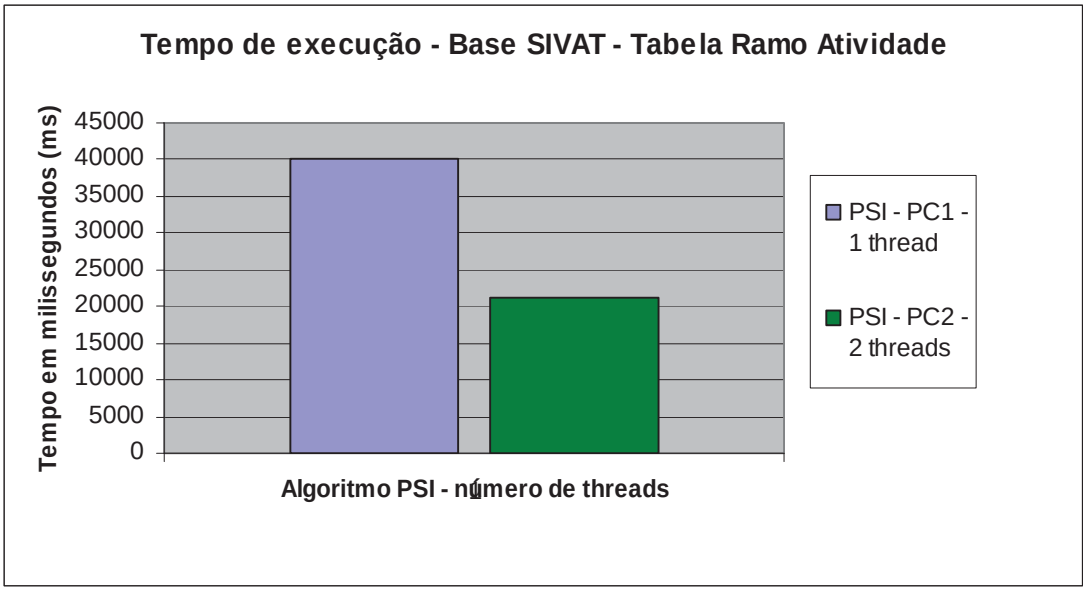


Figura 4.34 Gráfico do tempo de execução com o uso de *threads*– Base SIVAT – Tabela Ramo Atividade

Conforme é apresentando no gráfico da Figura 4.34, o tempo de processamento do Algoritmo *PSI* em um núcleo de processador foi de 40.127 milissegundos, ou seja, aproximados 40 segundos de execução, enquanto que em dois núcleos de processadores foi de 21.305 milissegundos, ou aproximados 21 segundos de execução. Neste caso, o processamento com um núcleo de processador foi quase duas vezes maior que com dois núcleos. Portanto, o fato de utilizar dois núcleos favorece a uma busca mais rápida pelos resultados.

4.3.2.3. Experimento 3 – Base SIVAT – Uso de Threads

O terceiro teste realizado com a utilização de *threads* foi aplicado na base de dados “SIVAT”, tabela “Ocupação” e coluna “Descrição”, executados por meio do “tipo de pesquisa automático” e o idioma “Português”. O gráfico de desempenho com o uso de *threads* é apresentado na Figura 4.35.

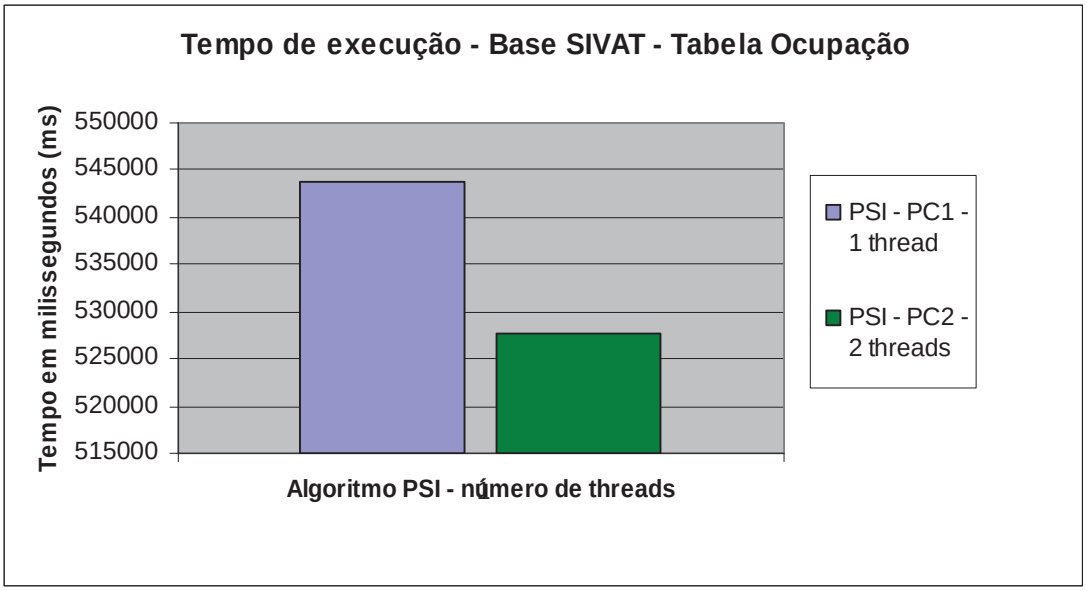


Figura 4.35 Gráfico do tempo de execução com o uso de *threads*– Base SIVAT – Tabela Ocupação

É possível observar na Figura 4.35 que o Algoritmo *PSI* executado em um núcleo de processamento demandou mais tempo de processamento do que em dois núcleos de processamento. De fato, com um único núcleo o tempo de execução foi de 543.859 milissegundos, ou seja, em torno de 9 minutos, enquanto que com dois núcleos foi de 527.671 milissegundos, aproximadamente 8 minutos.

4.3.2.4. Experimento 4 – Base SIVAT – Uso de Threads

O quarto teste realizado com a utilização de *threads* foi aplicado na base de dados "SIVAT", tabela "Ocupação" e coluna "Descrição", executados por meio do "tipo de pesquisa manual" pelo termo de pesquisa "Auxiliar Produção", no idioma "Português". O gráfico de desempenho com uso de *threads* é apresentado na Figura 4.36.

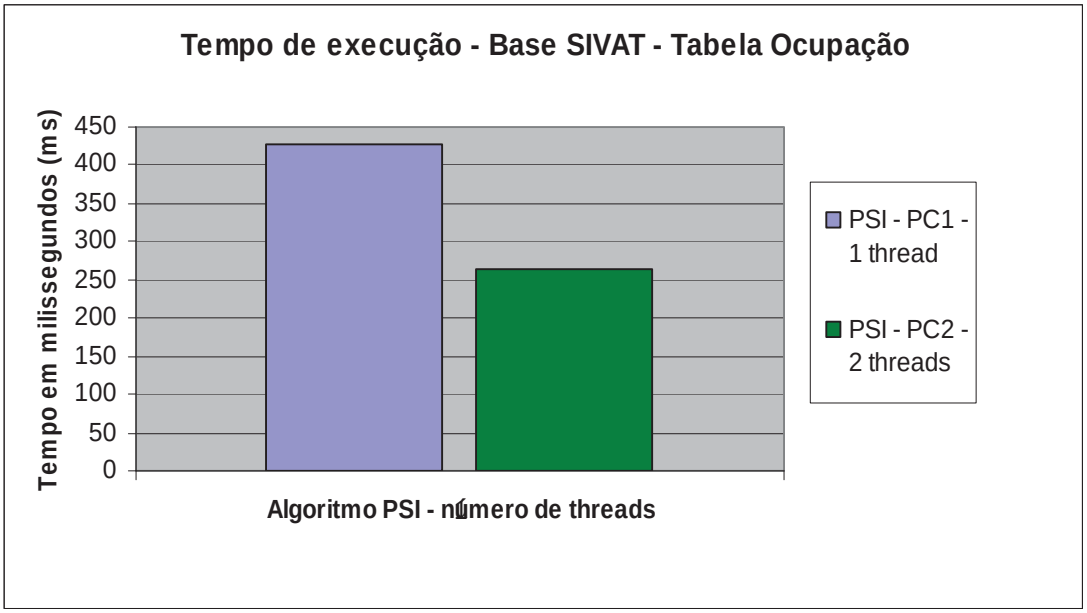


Figura 4.36 Gráfico do tempo de execução com o uso de *threads*– Base SIVAT – Tabela Ocupação – Termo Auxiliar Produção

Conforme é apresentado na Figura 4.36, a utilização do Algoritmo *PSI* em dois núcleos de processamento tem seu tempo de execução menor se comparado ao computador que possui apenas um núcleo. Isso comprova que a utilização de *threads* torna a velocidade de processamento mais eficiente, pois os dois núcleos buscaram em todos os registros e apresentaram os resultados em exatos 265 milissegundos, enquanto que em um núcleo a mesma operação foi realizada em 427 milissegundos.

4.3.2.5. Experimento 5 – Base SIVAT – Uso de Threads

O quinto teste realizado com a utilização de *threads* foi aplicado na base de dados “SIVAT”, tabela “Médico” e coluna “Nome”, executados por meio do “tipo de pesquisa manual” pelo termo de pesquisa “Walter”, no idioma “Português”. O gráfico de desempenho com o uso de *threads* é apresentado na Figura 4.37.

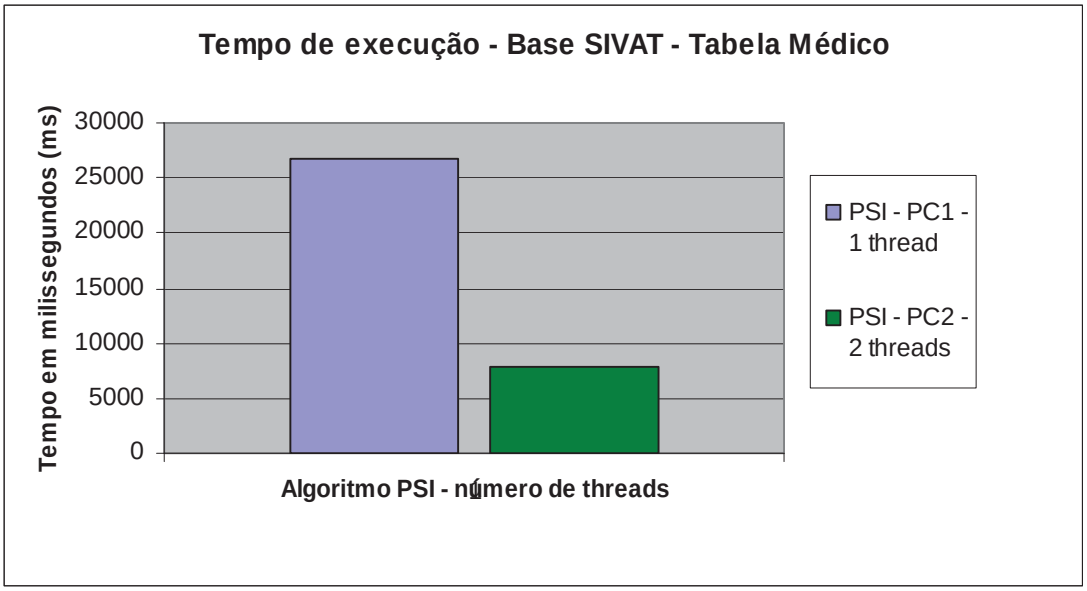


Figura 4.37 Gráfico do tempo de execução com o uso de *threads*– Base SIVAT – Tabela Médico – Termo Walter

Conforme é apresentado na Figura 4.37, o tempo de processamento do Algoritmo *PSI* em um núcleo foi de 26.751 milissegundos, ou aproximados 27 segundos de execução, enquanto que em dois núcleos foi de 7894 milissegundos, ou aproximados 8 segundos de execução. Neste caso, a divisão de processamento entre duas *threads* favoreceu um ganho de tempo quase quatro vezes menor do que com um único núcleo de processamento.

4.3.2.6. Experimento 6 – Base SIVAT – Uso de Threads

O sexto teste realizado com a utilização de *threads* foi aplicado na base de dados "SIVAT", tabela "Médico" e coluna "Nome", executados por meio do "tipo de pesquisa manual" pelo termo de pesquisa "Marcos", no idioma "Português". O gráfico de desempenho para este caso com o uso de *threads* é apresentado na Figura 4.38.

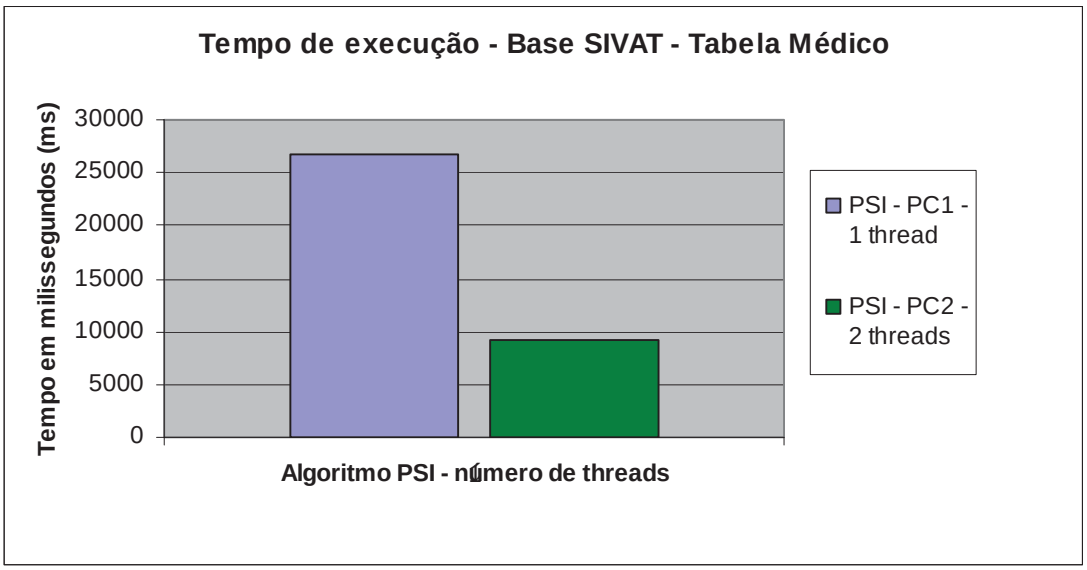


Figura 4.38 Gráfico do tempo de execução com o uso de *threads*– Base SIVAT – Tabela Médico – Termo Marcos

A partir da análise do gráfico apresentado na Figura 4.38, é possível constatar novamente que a utilização do Algoritmo *PSI* em um único núcleo de processamento demanda mais tempo que em dois núcleos de processamento. Precisamente, um único núcleo processou as informações e apresentou os resultados no tempo de 26.842 milissegundos, ou seja, aproximados 26 segundos, enquanto que em dois núcleos a mesma tarefa foi realizada em 9204 milissegundos, ou seja, 9 segundos. Isso comprova que a divisão do processamento em núcleos favorece a realização da tarefa de execução do Algoritmo *PSI* em menor tempo.

4.3.2.7. Experimento 7 – Base FUNCIONÁRIOS – Uso de Threads

O sétimo teste realizado com a utilização de *threads* foi aplicado na base de dados "Funcionários", tabela "Inscritos" e coluna "Endereço", executados por meio do "tipo de pesquisa manual" e pelo termo de pesquisa "Viale 7 (*avenida 7*)", no idioma "Italiano". O gráfico de desempenho para este caso com o uso de *threads* é apresentado na Figura 4.39.

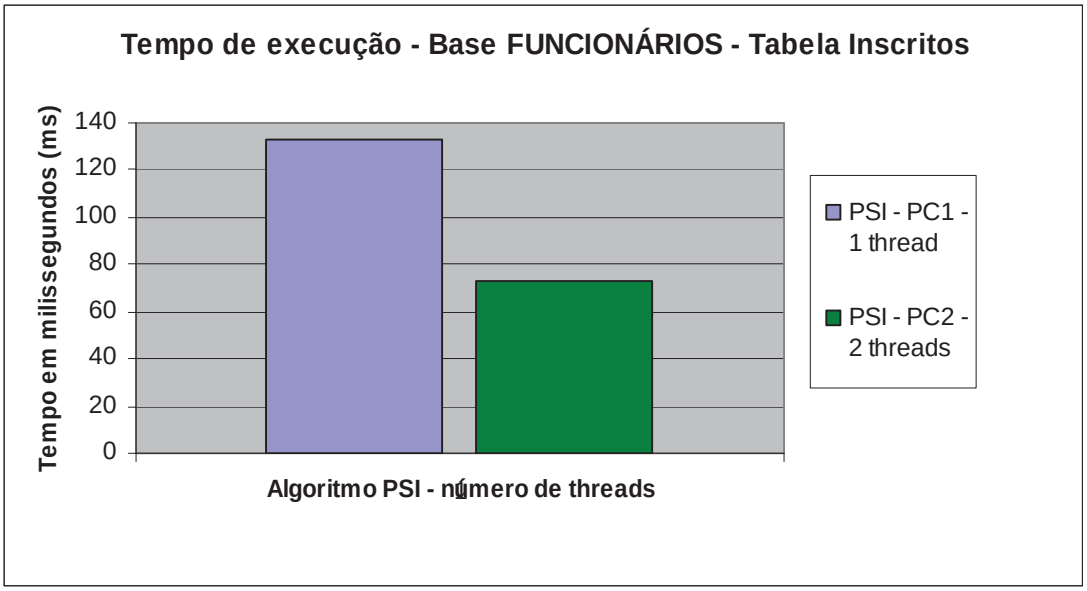


Figura 4.39 Gráfico do tempo de execução com o uso de *threads*– Base FUNCIONÁRIOS – Tabela Inscritos – Termo Viale 7

Conforme é apresentado no gráfico da Figura 4.39, nota-se que a execução do Algoritmo *PSI* em um núcleo de processador demanda mais tempo que em dois núcleos de processadores, embora o número de resultados apresentados sejam os mesmos. Um núcleo processou as informações e listou os resultados no tempo de 133 milissegundos, enquanto que os dois núcleos realizaram a mesma tarefa em 73 milissegundos.

4.3.2.8. Experimento 8 – Base FUNCIONÁRIOS – Uso de Threads

O oitavo e último teste realizado com a utilização de *threads* foi aplicado na base de dados “Funcionários”, tabela “Inscritos” e coluna “Nome”, executados por meio do “tipo de pesquisa automático” no idioma “Italiano”. O gráfico de desempenho com a utilização de *threads* é apresentado na Figura 4.40.

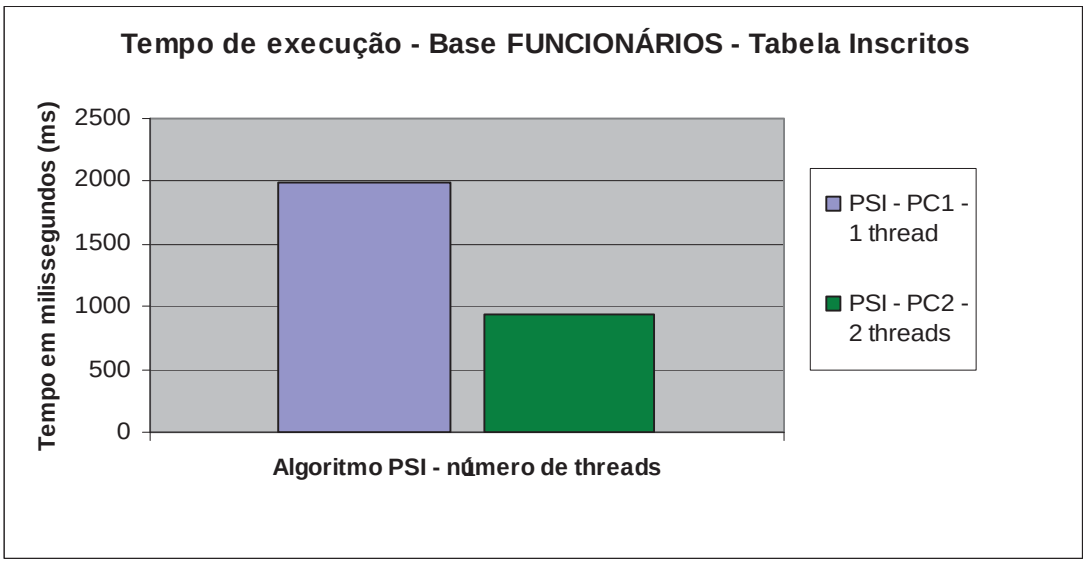


Figura 4.40 Gráfico do tempo de execução com o uso de *threads*– Base FUNCIONÁRIOS – Tabela Inscritos

De acordo com o gráfico apresentado na Figura 4.40, a utilização de *threads* no Algoritmo *PSI* melhora o grau de desempenho em relação ao tempo de execução. Observa-se que o tempo de execução do Algoritmo *PSI* em um núcleo de processador foi de 1.983 milissegundos, ou aproximados 2 segundos, enquanto que em dois núcleos foi de 935 milissegundos, ou seja, menos que 1 segundo de processamento dos mesmos resultados. Dessa forma, pode-se afirmar que a divisão do processamento das informações entre os núcleos de processador é viável, visto que há um importante ganho de tempo de execução.

4.3.3. Estudo comparativo dos Algoritmos em relação a *bytes* por segundo

Nesta seção, é abordada o tempo de processamento dos algoritmos em relação a quantidade de *bytes* de informações armazenadas pelo tempo de execução. Esses testes foram aplicados nas tabelas Máquina Causadora, Ramo Atividade, Ocupação e Médico, pertencentes ao Banco de Dados SIVAT, e na tabela Inscritos do Banco de Dados FUNCIONÁRIOS, conforme apresentadas na Tabela 4.1. Esses testes foram realizados a fim de constatar a quantidade de informações processadas em número de *bytes* por segundo.

Vale ressaltar que o tamanho da tabela em *bytes* pode variar, conforme as informações nela armazenada. O mesmo acontece com o tempo de processamento que varia conforme os dados, tamanho e o tratamento realizado por cada um dos algoritmos.

4.3.3.1. Experimento 1 – Base SIVAT – Tabela Máquina Causadora

O primeiro teste foi realizado na tabela máquina causadora, que contém 444 registros, cerca de 3.704 *bytes* de informações armazenadas. Os resultados obtidos de cada algoritmo são apresentados no subcapítulo 4.3.1.1. Com as informações contidas nesta tabela foi possível mensurar o tempo de processamento em *bytes* por segundo.

Tabela 4.2 Tempo de execução de bytes por segundo – Tabela máquina causadora

Algoritmo	Tamanho em Bytes	Tempo em segundos	Bytes por Segundo
Soundex	3.704 bytes	5 seg	740 b/s
Metaphone	3.704 bytes	1 seg	3.704 b/s
PSI (1 *thread*)	3.704 bytes	45 seg	82 b/s
PSI (2 *thread*)	3.704 bytes	31 seg	119 b/s

Conforme é apresentado na Tabela 4.2, o Algoritmo *Soundex* processou as informações na busca por registros duplicados em uma média de 740 *bytes* por segundo, enquanto que o Algoritmo *Metaphone* apresentou o processamento mais ágil, executando em média 3.704 *bytes* por segundo. O Algoritmo *PSI* executado em um núcleo de processador processou as mesmas informações em uma média de 82 *bytes* por segundo, enquanto que em dois núcleos o processamento foi em média de 119 *bytes* por segundo. Nota-se que este tempo de processamento está diretamente relacionado ao funcionamento dos algoritmos, sendo assim o algoritmo *PSI* leva um tempo maior na execução desta tarefa diante da execução mais complexa na forma de processar as informações.

4.3.3.2. Experimento 2 – Base SIVAT – Tabela Ramo Atividade

O segundo teste foi realizado na tabela ramo atividade, que contém 260 registros, cerca de 5.263 *bytes* de informações armazenadas. Os resultados obtidos de cada algoritmo são apresentados no subcapítulo 4.3.1.2. Na Tabela 4.3 é apresentada o tempo de execução de cada algoritmo em número de *bytes* por segundo.

Tabela 4.3 Tempo de execução de bytes por segundo – Tabela ramo atividade

Algoritmo	Tamanho em Bytes	Tempo em segundos	Bytes por Segundo
Soundex	5.263 bytes	2 seg	2.631 b/s
Metaphone	5.263 bytes	2 seg	2.631 b/s

PSI (1 *thread*)	5.263 bytes	40 seg	131 b/s
PSI (2 *thread*)	5.263 bytes	21 seg	250 b/s

De acordo com a Tabela 4.3, os Algoritmos *Soundex* e *Metaphone* processaram as informações na busca por registros duplicados em uma média de 2.631 *bytes* por segundo. O Algoritmo *PSI*, executado em um núcleo de processamento, processou as mesmas informações em uma média de 131 *bytes* por segundo, enquanto que em dois núcleos o processamento foi em média de 250 *bytes* por segundo.

4.3.3.3. Experimento 3 – Base SIVAT – Tabela Ocupação

O terceiro teste foi realizado na tabela ocupação, que contém 1.330 registros, cerca de 22.699 *bytes* de informações armazenadas. Os resultados obtidos de cada algoritmo são apresentados no subcapítulo 4.3.1.3. Na Tabela 4.4 é apresentada o tempo de processamento de cada algoritmo em *bytes* por segundo.

Tabela 4.4 Tempo de execução de bytes por segundo – Tabela ocupação

Algoritmo	Tamanho em Bytes	Tempo em segundos	Bytes por Segundo
Soundex	22.699 bytes	240 seg	94 b/s
Metaphone	22.699 bytes	120 seg	189 b/s
PSI (1 *thread*)	22.699 bytes	540 seg	42 b/s
PSI (2 *thread*)	22.699 bytes	480 seg	47 b/s

Conforme é apresentado na Tabela 4.4, o Algoritmo *Soundex* processou as informações contidas em uma média de 94 *bytes* por segundo e o Algoritmo *Metaphone* processou em uma média de 189 *bytes* por segundo. O Algoritmo *PSI* executou com um núcleo de processamento as mesmas informações em uma média de 42 *bytes* por segundo, enquanto que em dois núcleos o processamento foi em média de 47 *bytes* por segundo.

4.3.3.4. Experimento 4 – Base SIVAT – Tabela Médico

O quarto teste foi realizado na tabela médico, que contém 141.375 registros, cerca de 3.381.761 *bytes* de informações armazenadas. Neste caso, a busca foi restrita a um termo, “Walter”, a fim de mensurar o tempo em relação *bytes*/segundo. Os resultados

obtidos de cada algoritmo são apresentados no subcapítulo 4.3.1.5. Com as informações contidas nesta tabela foi possível mensurar o tempo de processamento em *bytes* por segundo, conforme são apresentados na Tabela 4.5.

Tabela 4.5 Tempo de execução de bytes por segundo – Tabela médico

Algoritmo	Tamanho em Bytes	Tempo em segundos	Bytes por Segundo
Soundex	3.381.761 bytes	4 seg	845.440 b/s
Metaphone	3.381.761 bytes	1 seg	3.381.761 b/s
PSI (1 *thread*)	3.381.761 bytes	27 seg	125.250 b/s
PSI (2 *thread*)	3.381.761 bytes	8 seg	422.720 b/s

Conforme é apresentado na Tabela 4.5, na busca pelo termo “Walter” na tabela médico, o Algoritmo *Soundex* resultou no processamento médio de 845.440 *bytes* por segundo, enquanto que o Algoritmo *Metaphone* processou uma média de 3.381.761 *bytes* por segundo. O Algoritmo *PSI* processou as mesmas informações em um núcleo de processador uma média de 125.250 *bytes* por segundo, enquanto que em dois núcleos o processamento foi em média de 422.720 *bytes* por segundo.

4.3.3.5. Experimento 5 – Base FUNCIONÁRIOS – Tabela Inscritos

O quinto e último teste foi realizado na tabela inscritos, que contém 100 registros, cerca de 4.492 *bytes* de informações armazenadas. Os resultados obtidos de cada algoritmo são apresentados no subcapítulo 4.3.1.8. Na Tabela 4.6 é apresentada o tempo de processamento de cada algoritmo.

Tabela 4.6 Tempo de execução de bytes por segundo – Tabela inscritos

Algoritmo	Tamanho em Bytes	Tempo em segundos	Bytes por Segundo
Soundex	4.492 bytes	0,386 seg	11,637 b/s
Metaphone	4.492 bytes	0,135 seg	33,274 b/s
PSI (1 *thread*)	4.492 bytes	1,983 seg	2,265 b/s
PSI (2 *thread*)	4.492 bytes	0,935 seg	4,804 b/s

De acordo com a Tabela 4.6, o Algoritmo *Soundex* processou uma média de 11,637 *bytes* por segundo as informações armazenadas, enquanto que o Algoritmo

Metaphone processou uma média de 33,274 *bytes* por segundo. O Algoritmo *PSI* executado com um núcleo de processador processou as mesmas informações em uma média de 2,265 *bytes* por segundo, enquanto que em dois núcleos o processamento foi em média de 4,804 *bytes* por segundo.

4.4. Considerações finais

Neste capítulo, foram apresentados os testes aplicados em diferentes tabelas das bases de dados, a fim de demonstrar o funcionamento do algoritmo proposto. Para melhor exemplificar, foram detalhados os diferentes resultados obtidos na utilização dos algoritmos *Soundex*, *Metaphone* e *PSI*, realizado um estudo de desempenho entre a aplicação do algoritmo proposto em um único núcleo de processador e dois núcleos de processadores, bem como demonstrar o tempo de processamento dos algoritmos em diferentes tabelas a fim de mensurar a quantidade de *bytes* processados por segundos de tempo.

Capítulo 5
Conclusões

5.1. Conclusões finais

Diante dos capítulos apresentados, este trabalho objetivou a construção de um algoritmo que é capaz de identificar registros duplicados em bases de dados por meio da busca por similaridade numérica e fonética, favorecendo a uma busca mais precisa dos resultados e contribuindo significativamente para a limpeza do banco com a melhoria da qualidade dos dados armazenados que são utilizados para tomadas de decisões.

A limpeza de dados tornou-se um objeto de grandes estudos devido a necessidade de melhoramento da qualidade dos dados armazenados para futuras projeções. A cada dia surgem novos algoritmos que vêm somar com as técnicas existentes para a limpeza de dados. Esses algoritmos devem ser estudados, a fim de que possam ser aprimorados em busca da realização perfeita na etapa de limpeza de dados. Dentre os algoritmos estudados, não se pode afirmar que há aquele que tem um melhor desempenho, já que isto depende de onde será aplicado e qual o resultado que se busca. Muitas vezes, faz-se necessário uma junção de vários desses algoritmos citados a fim de buscar uma satisfação de resultado, ou o mais próximo do resultado preciso para tomada de decisão.

Após o estudo dos algoritmos existentes e identificado os diversos problemas contidos nas bases de dados, surgiu à idéia de desenvolver um algoritmo capaz de auxiliar a etapa de limpeza de dados. Isto objetivou o rumo deste trabalho, que é a concretização de desenvolvimento de um algoritmo baseado na identificação de registros duplicados por meio da similaridade fonética e numérica. É nesse sentido que este trabalho apresenta o

algoritmo *PSI* para a detecção de tuplas duplicatas, com o objetivo de obter resultados mais precisos pelos sistemas de identificação por meio da similaridade. Portanto, a idéia de desenvolvimento deste algoritmo busca sanar um dos problemas levantados, a identificação de registros duplicados, de forma a contribuir com a etapa de limpeza de dados.

Para a tarefa de identificar esses registros, foram estudadas algumas das técnicas desenvolvidas e constatada a viabilidade da utilização da técnica baseada em distância, que trata cada palavra como grandes *strings*, não havendo a necessidade de usar dados treinados para a identificação. Para somar a utilização desta técnica, adotou-se a metodologia de identificação baseado na similaridade fonética e numérica. A importância de usar a fonética parte da idéia de identificar registros duplicados armazenados na base de dados conforme foram escritos, podendo ter a sua variação baseada numa citação ou fala. Isto não leva em consideração as características regionais que cada palavra pode sofrer ao ser pronunciada e, sim, a maneira que um digitador pode escrevê-la diante das variações de grafia existente. Já a necessidade de identificação dos registros numéricos duplicados surge diante da deficiência dos algoritmos estudados de não considerá-los como parte integrante da palavra armazenada. Vale destacar que o trabalho com dados numéricos pode auxiliar também no processo de identificação de valores fora do domínio.

Para contribuir na idéia de identificar registros duplicados na base de dados por meio da similaridade fonética, somou-se a possibilidade de trabalhar com diversos idiomas. No entanto, primeiramente foram realizadas transcrições fonéticas das vogais, consoantes, dígrafos e caracteres especiais do alfabeto da língua portuguesa, no intuito de elaboração de um estudo de caso, a fim de comprovar a viabilidade e eficiência do algoritmo *PSI*. Após essas transcrições, objetivou-se a transcrição fonética de uma língua européia, a citar o idioma italiano, com a finalidade de reforçar o que se pretendia com a língua portuguesa. Diante disso, testes foram realizados e bem sucedidos, o que comprova a idéia de que trabalhar com a fonética para a identificação de duplicatas é importante e de grande utilidade. A tarefa de identificação de duplicatas baseada na similaridade fonética da língua portuguesa e italiana é inovadora, não sendo encontrado nenhum material científico que caracteriza a identificação de registros duplicados por meio desses idiomas.

Contudo, o desenvolvimento do algoritmo *PSI* viabilizou a identificação de registros duplicados na base de dados, o que possibilitou a comparação, análise e a conclusão de resultados mais exatos se comparado aos algoritmos *Soundex* e *Metaphone*, pois favoreceu uma busca mais eficiente. Por meio da análise dos resultados adquiridos, foi

possível constatar que o algoritmo *PSI* foi em média 30% mais eficiente que os algoritmos *Soundex* e *Metaphone* na busca por registros duplicados, pois os resultados obtidos por este algoritmo apresentaram um maior grau de similaridade do que os algoritmos difundidos na literatura, tanto nos testes realizados na língua portuguesa como na língua italiana. Os algoritmos *Soundex* e *Metaphone* trabalham com uma metodologia de busca que demonstram resultados muito abrangentes, o que não contribui para uma representação precisa dos resultados. Vale ressaltar que estes algoritmos foram codificados baseados na versão descrita pelos respectivos autores.

Análises realizadas por meio do tempo total de execução e o número de *bytes* processados por segundo de cada algoritmo apontam que o algoritmo *PSI* tem o tempo processamento maior dos que os algoritmos *Soundex* e *Metaphone*. Embora o tempo de processamento do Algoritmo *PSI* tenha sido superior se comparado aos demais algoritmos estudados nos testes realizados, os resultados são melhores e mais precisos, o que compensa o tempo gasto para identificação. Para tanto, foram utilizadas tabelas das bases de dados com um número de registros reduzidos, ou seja, em torno de 100 registros, como, também, tabelas com números de registros expressivos, que contém cerca de 140 mil registros. Verifica-se, portanto, a importância de ter resultados mais exatos do que o tempo de processamento das informações, já que os resultados são para tomada de decisões e o tempo de processamento é relativo, conforme o tamanho das bases de dados.

A fim de agregar funcionalidades e aperfeiçoar a metodologia de funcionamento do algoritmo *PSI*, adotou-se a divisão de processamento entre núcleos de processadores dos computadores, funcionalidade esta chamada de *multithreading*, com o intuito de melhorar o desempenho do algoritmo em relação ao tempo de processamento, visto que cada etapa de transcrição fonética dos caracteres de uma palavra é altamente demorada. Por meio dessa funcionalidade, foi possível verificar uma melhora significativa no tempo de processamento do algoritmo *PSI*¸ fator que colaborou para uma apresentação mais rápida dos resultados obtidos. Portanto, a divisão da tarefa de processar as informações entre os núcleos de processadores contribui na velocidade de processamento e da listagem dos resultados. É importante destacar que a velocidade de processamento está ligada ao número de núcleos de processadores que o computador possui, pois quanto mais núcleos, mais rápida será a execução do algoritmo *PSI*.

Para concluir, vale ressaltar que o algoritmo *PSI* pode ser aplicado em qualquer base de dados, independente do tamanho, número de registros e domínio do atributo, podendo ser numérico ou alfanumérico. O objetivo deste trabalho foi alcançado, já que por

meio do desenvolvimento do Algoritmo *PSI* foi possível obter resultados satisfatórios, em um curto tempo de processamento e de listagem dos resultados.

5.2. Sugestões de trabalhos futuros

A partir do trabalho desenvolvido, é possível sugerir as seguintes opções de pesquisa e desenvolvimento:

a) *Implementação da funcionalidade de limpeza de dados no Ambiente DIBP*: por meio do algoritmo *PSI* desenvolvido, foi possível identificar os registros duplicados, e como próximo passo a efetuação da tarefa de limpeza de dados.

b) *Adição de novos idiomas no Ambiente DIBP*: para enriquecer o ambiente *DIBP*, a adição de novas transcrições fonéticas favoreceria a identificação de registros duplicados em outros idiomas. Como sugestão, a inclusão da língua inglesa é de extrema importância na continuidade deste trabalho.

c) *Módulo de adição de transcrição fonética em diversos idiomas no Ambiente DIBP*: para tornar o ambiente mais rico em funcionalidade, é sugerido a proposta de implementação deste módulo com a finalidade de oferecer uma maior variedade de transcrições fonéticas em diversos idiomas, realizando o *upload* destas transcrições sem que haja a necessidade da programação em linhas de código.

d) *Implementação de novos algoritmos para a limpeza de dados*: para tornar o Ambiente *DIBP* mais robusto, a adição de novos algoritmos na tarefa de identificar registros duplicados é importante e aconselhável, a fim de favorecer a uma melhor limpeza de dados.

e) *Explorar novas métricas para medir desempenho*: para calcular o desempenho dos algoritmos estudados e o proposto, a utilização de novas métricas é interessante, o que favorece o enriquecimento da pesquisa a fim de comprovar a eficiência do algoritmo proposto.

Referências Bibliográficas

ARASU, Arvind; KAUSHIK, Raghav. **A Grammar-based Entity Representation Framework do Data Cleaning.** ACM - 35th SIGMOD International Conference on Management of Data, Rhode Island, p. 233-244, 2009.

BABINI, Maurizio. **Fonética, Fonologia e Ortoépia da Língua Italiana.** Annablume, 2002. 144 p.

BILENKO, Mikhail; MOONEY, Raymond; COHEN, William; RAVIKUMAR, Pradeep; FIENBERG, Stephen. **Adaptive Name Matching in Information Integration.** IEEE Intelligent Systems - Computer Society, vol. 18, p. 16-23, Set/Out. 2003.

BRADLOW, Ann; CLOPPER, Cynthia; SMILJANIC, Rajka; WALTER, Mary Ann. **A perceptual phonetic similarity space for languages: evidence from five native language listener groups.** Elsevier Science Publishers - Journal Speech Communication, vol. 52, p. 930-942, Nov. 2010.

CAGLIARI, Luiz Carlos. **Análise Fonológica: introdução a teoria e prática, com especial destaque para o modelo fonêmico.** Mercado de Letras, 2002. 208 p.

CISZAK, Lukasz. **Application of Clustering and Association Methods in Data Cleaning.** IEEE - Proceedings of the International Multiconference on Computer Science and Information Technology, Warszawa, p. 97-103. Out, 2008.

CHRISTEN, Peter. **Development and User Experiences of an Open Source Data Cleaning, Deduplication and Record Linkage System.** ACM SIG*KDD* Explorations Newsletter, vol. 11, p. 39-48, Jun. 2009.

ELMAGARMID, Ahmed K.; IPEIROTIS, Panagiotis G.; VERYKIOS, Vassilios S.. **Duplicate Record Detection: A Survey.** IEEE Transactions On Knowledge And Data Engineering, vol. 19, n. 1, p. 1-16, Jan. 2007.

FAN, Wenfei; GEERTS, Floris; JIA, Xibei. **A Revival of Integrity Constraints for Data Cleaning.** ACM - Proceedings of the VLDB Endowment, vol. 1, p. 1522-1523, Ago. 2008.

FAYYAD, Usama; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic; **From Data Mining to Knowledge Discovery: An Overview.** In: FAYYAD, Usama, et al. A*dvances in Knowledge Discovery and Data Mining*. Menlo Park: AAAI Press, The MIT Press, p. 1-34, 1996.

FAYYAD, Usama; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. **From Data Mining to Discovery Knowledge in Databases**. AI Magazine – American Association for Artificial Intelligence, p. 37-54, 1996.

GÁLVEZ, Carmen. **Identificación de Nombres Personales por Medio de Sistemas de Codificación Fonética.** Encontros Bibli: Revista Eletrônica de Biblioteconomia e Ciência da Informação – Universidade Federal de Santa Catarina – UFSC, n. 22, p. 105-116, 2006.

GUIMARÃES, Ângelo de Moura; LAGES, Newton Alberto de Castilho. **Algoritmos e Estruturas de Dados.** 4. ed. Rio de Janeiro: Livros Técnicos e Científicos, 1994. 216 p.

HAN, Jiawei; KAMBER, Micheline. **Data Mining: Concepts and Techniques.** 2. ed. San Francisco: Elsevier, 2006. 743 p.

HASSANZADEH, Oktie; MILLER, Renée J.. **Creating probabilistic databases from duplicated data.** ACM – The VLDB Journal – The International Journal on Very Large Databases, p. 1141-1166, Out. 2009.

HE, Zengyou; XU, Xiaofei; DENG, Shengchun. **Clustering Mixed Numeric and Categorical Data: A Cluster Ensemble Approach**. ArXiv Computer Science e-prints (2005). Disponível em <http://arxiv.org/ftp/cs/papers/0509/0509011.pdf>. Acesso em: 27 outubro 2009.

HENRIQUES, Claudio Cezar. **Fonética, Fonologia e Ortografia: conceitos, estruturas e exercícios com respostas.** Elsevier, 2007. 138 p.

KDNUGGETS, Data Mining Community's Top Resource.1997. Disponível em: <http://www.kdnuggets.com/>. Acesso em: novembro/2009.

KONDRAK, Grzegorz. **Identifying Cognates by Phonetic and Semantic Similarity.** ACM - NAACL '01 Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies, p. 103-110, Jun. 2001.

KYUEUN, Yi; GAUDIOT, Jean-Luc. **Network Applications on Simultaneous Multithreading Processors.** IEEE Transactions on Computers, vol 59, nº 9, p. 1200-1209, Sep. 2010.

LARMAN, Craig. **Utilizando UML e Padrões.** 2. ed. Porto Alegre: Bookman, 2004. 608 p.

LIST, Johann-Mattis. **Phonetic Alignment Based on Sound Class: a new method for sequence comparison in Historical Linguistics.** ESSLLI 2010 – European Summer School of Logic, Language and Information, Dinamarca, p. 1-4, Aug. 2010.

LOWE, Robert J. **Fonologia: avaliação e intervenção: aplicações na patologia da fala.** Artes Médicas, 1996. 237 p.

LUO, Qi. **Advancing Knowledge Discovery and Data Mining.** IEEE Computer Society: Workshop on Knowledge Discovery and Data Mining, Washington, p. 3-5, 2008.

MALMBERG, Bertil. **A fonética: no mundo dos sons da linguagem.** Coleção Vida e Cultura. Livros do Brasil Lisboa, 1954. 195 p.

MITRA, Sushmita; ACHARYA, Tinku. **Data Mining: Multimedia, soft Computing and Bioinformatics.** Hoboken: John Wiley & Sons, Inc., 2003. 401 p.

NAKOV, Svetlin; PASKALEVA, Elena; NAKOV, Preslav. **A knowledge-Rich Approach to Measuring the Similarity between Bulgarian and Russian Words.** ACM - Multilingual Resources, Technologies and Evaluation for Central and Eastern European Languages, Bulgaria, p. 32-39, 2009.

OLIVEIRA, Paulo Jorge; RODRIGUES, Fátima. **Limpeza de Dados: Uma Visão Geral.** Relatório Técnico. Simpósio Doutoral do Departamento de Informática. Universidade do Minho, Braga, Portugal, 2004.

PETRIK, Stefan; DREXEL, Christina, FESSLER, Leo; JANCSARY, Jeremy; KLEIN, Alexandra; KUBIN, Gernot; MATIASEK, Johannes; PERNKOPF, Franz; TROST, Harald. **Semantic and phonetic automatic reconstruction of medical dictations.** Elsevier - Journal Computer Speech and Language, vol. 25, p. 363 – 385, Apr. 2011.

PIATETSKY-SHAPIRO, Gregory. Knowledge **Discovery in Real Databases: A Report on the IJCAI-89 Workshop**. AI Magazine – American Association for Artificial Intelligence**,** v.11, n. 5, p. 68-70, 1991.

RAHM, Erhard; DO, Hong Hai. **Data Cleaning: Problems and Current Approaches.** IEEE Data Engineering Bulletin, p. 3-13. dez. 2000.

SILVA, Gabriel P. Arquitetura de Computadores II. **Multithreading.** Universidade Federal do Rio de Janeiro – UFRJ. Disponível em <http://equipe.nce.ufrj.br/gabriel/arqcomp2/Multithread.pdf>. Acesso em: 27 junho 2011.

SILVA, Thaïs Cristófaro. **Fonética e Fonologia do Português: roteiro de estudos e guia de exercícios.** Contexto, 2007. 254 p.

WEIS, Melanie; NAUMANN, Felix; JEHLE, Ulrich; LUFTER, Jens; SCHUSTER, Holger. **Industry-Scale Duplicate Detection.** ACM - Proceedings of the VLDB Endowment, vol. 1, p. 1253-1264, Ago. 2008.

XIAO, Chuan; WANG, Wei; LIN, Xuemin; YU, Jeffrey Xu. **Efficient Similarity Joins for near Duplicate Detection**. ACM – International World Wide Web Conference, p. 131-140, 2008.

YAN, Hao; DIAO, Xing-chun. **The Desing and Implementation of Data Cleaning Knowledge Modeling.** IEEE Computer Society – International Symposium on Knowledge Acquisition and Modeling, Nanjing, p. 177-179, Dez. 2008.

Apêndice A
Fichamentos sobre a Fonética

A.1. A Fonética – Bertil Malmberg (MALBERG, 1954).

A fonética é o estudo dos sons da linguagem. É, pois, um ramo da lingüística, mas um ramo que, ao contrário dos outros, apenas se interessa pela linguagem articulada e não por outras formas de comunicação organizada (linguagem escrita, linguagem dos surdos-mudos, sinais dos marinheiros, etc). P. 09

Aliás, o emprego do alfabeto fonético internacional não se encontra generalizado em França, nem sequer entre os lingüistas. P. 11

Em português existem duas espécies de ditongos orais e nasais. Ao primeiro tipo pertencem: ai, au, ei, éi, eu, eu, iu, ui, oi, oi; do segundo tipo são: ãe, ao, am, em, õe, ~ui (representado sem til em mui e muito, por exemplo). Podemos ainda reagrupá-los em ditongos crescentes e decrescentes. P. 69

Enquanto as vogais se caracterizam acusticamente pela ausência de ruído audível e, do ponto de vista articulatório, por uma passagem de ar livre, as consoantes são – ou contém – ruídos e pronunciam-se com um fechamento ou uma constrição quando da passagem do ar. P. 73

As consoantes portuguesas p, t, c (q) são oclusivas não aspiradas. É este mesmo tipo que se encontra nas outras línguas românticas e na maior parte das línguas européias, com exceção do grupo germânico. P. 76

O simples fato de que a versificação é frequentemente baseada no número de sílabas fornece-nos uma prova de que a sílaba é uma unidade fonética da qual os sujeitos falantes estão absolutamente conscientes. P. 115

O estudo cujo objetivo é determinar as distinções fonéticas que, em dada língua, tem valor diferencial e estabelecer o sistema de fonemas e de prosodemas, é frequentemente designado por fonologia. P. 169

A fonologia, tomada nesse sentido, foi fundada em Praga há cerca de trinta anos, por um grupo de lingüistas (Troubetzkoy, Jakobson e outros), o que explica o nome da escola de Praga. P. 169

Quando de trata de saber porque muda a pronuncia, o investigador encontra-se perante dificuldades quase insuperáveis. Os sons não são os únicos elementos que mudam numa língua. Também mudam as formas, o tipo de sintaxe, o vocabulário e o estilo literário. P. 173

Umas palavras freqüentes, cotidianas, sofre mais facilmente os efeitos de uma tendência fonética do que uma palavra rara, literária ou especial. P. 176

Do que se disse lá atrás conclui-se que nem a fonética nem a lingüística são capazes de explicar por si mesmas as evoluções fonéticas. É preciso ultrapassar os limites da fonética – e até os da lingüística – para encontrar, se for possível, todos os fatores que determinam em conjunto a evolução dos sons e a das línguas. P. 183

Graças a invenção do telefone, rádio, fonógrafo, alto-falantes, gravadores de som e cinema sonoro, a língua falada substitui cada vez mais a língua escrita. P. 188

A.2. Análise Fonológica – Luiz Carlos Cagliari (CAGLIARI, 2002).

A Fonética e a Fonologia são áreas da Lingüística que estudam os sons das línguas. A Fonética preocupa-se principalmente com a descrição dos fatos físicos que caracterizam lingüisticamente os sons da fala. P. 17

A fonética descreve os sons da fala, dizendo quais mecanismos e processos de produção de fala estão envolvidos em um determinado segmento da cadeia sonora da fala. Diz, por exemplo, que um som articula-se com uma corrente de ar pulmonar, egressiva, com uma função sonora, com uma obstrução fricativa à corrente de ar, formada pela aproximação dos lábios levemente protusos. P. 17

A fonologia, por sua vez, faz uma interpretação dos resultados apresentados pela Fonética, em função dos sistemas de sons das línguas e dos modelos teóricos que existem para descrevê-los. P. 18

A fonética é basicamente descritiva e a Fonologia, interpretativa. P. 18

A análise fonológica baseia-se no valor dos sons dentro de uma língua, isto é, na função lingüística que eles desempenham nos sistemas de sons da língua. Enquanto a Fonética descreve o que acontece quando um falante fala, a Fonologia almeja a descrição da organização sistemática global dos sons da língua desse falante. P. 18

Há um limite mínimo de segmentação da cadeia da fala que permite a identificação de um composto formado de significado e significante. Menos do que isto, perde-se o significado.

O uso de símbolos fonéticos para representar os fonemas é feito para auxiliar o raciocínio fonológico, não para indicar um som fonético propriamente dito. P. 26

Os alofones, por sua vez, são os representantes fonéticos dos fonemas. Os fonemas são representados entre duas barras inclinadas e os alofones entre colchetes quadrados. P. 26

A fala realiza-se através de uma cadeia de sons, produzindo um continuo sonoro de qualidades variáveis ao longo do tempo. Um ambiente fonológico ou contexto é

constituído por um ou mais elementos que precedem ou seguem um determinado segmento da fala. P. 27

Quando a troca de um som por outro modificar o significado, esses sons estão em oposição e são classificados como fonemas. Quando não ocorrer a mudança de significado, trocando um som por outro em um determinado ambiente, não ocorre oposição fonológica e os sons são variantes de um mesmo fonema. P. 33

Tradicionalmente, a fonologia marca os sons foneticamente semelhantes (SFS) – ou pares suspeitos – circunscrevendo-os em balões. P. 39

A.3. Fonética, Fonologia e Ortografia – Cláudio Cezar Henriques (HENRIQUES, 2007).

Tomada em seu objetivo significativo, a serviço da comunicação, a Fonação – que é o ato humano de emitir sons vocais – se constitui como a própria fala. P. 6

A distinção metodológica entre Fonética e Fonologia ocorre na primeira metade do século XX, em decorrência sobretudo das idéias de Sausurre (1916), embora outros autores, como Courtenay (1895) e Jepersen (1904), já houvessem se manifestado a respeito dos valores lingüísticos distintivos dos elementos fônicos. P. 6

A fonética estuda os sons da fala; a Fonologia estuda os sons da língua. P. 6

Só a transcrição fonética expõe rigorosa e sistematicamente os fonemas empregados na pronúncia de uma palavra ou sintagma. P. 9

Os sistemas de transcrição fonética vêm sendo aperfeiçoados desde os finais do século XIX, criando alfabetos fonéticos a fim de indicar individualmente todas as nuanças sonoras produzidas pelo aparelho fonador. Um dos mais difundidos é o alfabeto fonético da *International Phonetic Association*, utilizado em livros e trabalhos de lingüística e em muitos dicionários bilíngües para ensinar como se deve pronunciar cada palavra. P. 15

A língua portuguesa possui 33 fonemas, sendo 12 vogais, 19 consoantes e 2 semivogais. Vogais: ver página 17.

A.4. Fonética e Fonologia do Português – Thaïs Cristófaro Silva (SILVA, 2007).

Todo falante de qualquer língua é capaz de reconhecer se outro falante apresenta sotaque ou não. Adicionalmente, todo falante é capaz de fazer julgamentos sobre a sua língua materna. Os ramos da ciência que estudam a sonoridade são a fonética e a fonologia**.**

A fonética se destaca pelo estudo articulatório e acústico da fala. A fonologia estuda a categorização de sons em línguas específicas e os aspectos relacionados com a percepção.

A sonoridade – tomada no sentido estrito do "barulho" da fala – é parte de conteúdo programático de disciplinas em várias áreas do conhecimento e consequentemente em vários cursos da graduação como o de Letras, Artes Cênicas, Música, Fonoaudiologia, Psicologia, e Tecnologia da Fala**.**

A Lingüística é a ciência que estuda os fenômenos relacionados à linguagem verbal humana, buscando entender quais são as características e princípios que regem às estruturas das línguas do mundo. Aos lingüistas, então, cabe estudar e formular explicações acerca das estruturas e dos mecanismos da linguagem em geral.

O lingüista pode estudar a linguagem, analisando suas modificações durante um determinado período (Lingüística diacrônica ou histórica), ou, então, estudar uma determinada língua no seu estágio de evolução atual (Lingüística sincrônica).

A Fonologia é a ciência que estuda a organização dos sistemas sonoros das línguas naturais. Transcrições fonológicas são apresentadas entre barras transversais: **/.../**. Transcrições fonológicas recebem várias denominações dependendo do modelo fonológico, como, por exemplo: representação subjacente, representação lexical, representação mental ou *output*.

Apêndice B

Transcrições Fonéticas da Língua Italiana

As tabelas B.1 a B.19 foram elaboradas pelo autor deste trabalho como uma contribuição científica e apresentam as transcrições fonéticas das vogais, consoantes, dígrafos e trígrafos da língua italiana, conforme explicações das obras de Maurizio Babini (BABINI, 2002). Essas transcrições foram realizadas para o estudo de caso da língua italiana e foram necessárias para a implementação no ambiente *DIBP*, que identifica registros duplicados em bases de dados baseado na similaridade fonética e numérica de diferentes idiomas, neste caso o italiano.

Tabela B.1 Transcrição fonética das vogais – língua italiana

Caractere	Transcrição Fonética	Exemplos
A	/a/	*capo (chefe)*
E	/e/	*pane (pão)*
	/ɛ/	*caffe (café)*
I	/i/	*mari (mares)*
O	/o/	*anno (ano)*
	/ɔ/	*dote (dote)*
U	/u/	*rumore (ruído)*

Tabela B.2 Transcrição fonética da consoante B – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
B	/b/	B+A	[ba]	*barca (barco)*
		B+E	[be]	*becco (bico)*
			[bɛ]	*bello (linda)*
		B+I	[bi]	*bica (monte)*
		B+O	[bo]	*bocca (boca)*
			[bɔ]	*bocce (bocha)*
		B+U	[bu]	*buono (bom)*

Tabela B.3 Transcrição fonética da consoante C – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
C	/tʃ/	C+E	[tʃe]	*ceffo (focinho)*
			[tʃɛ]	*cédola (cédula)*
		C+I	[tʃi]	*cielo (céu)*
	/k/	C+A	[ka]	*cane* (*cachorro*)
		C+O	[ko]	*coccio (caco)*
			[kɔ]	*codice (código)*
		C+U	[ku]	*cubo (cubo)*

Tabela B.4 Transcrição fonética da consoante D – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
D	/d/	D+A	[da]	*damasco (damasco)*
		D+E	[de]	*decalitro (decalitro)*
			[dɛ]	*deca (dezena)*
		D+I	[di]	*dito (dedo)*
		D+O	[do]	*docente (docente)*
			[dɔ]	*dócile (dócil)*
		D+U	[du]	*duale (dual)*

Tabela B.5 Transcrição fonética da consoante F – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
F	/f/	F+A	[fa]	*famoso (famoso)*
		F+E	[fe]	*feltro (feltro)*
			[fɛ]	*fé (fé)*
		F+I	[fi]	*fine (fim)*
		F+O	[fo]	*formale (formal)*
			[fɔ]	*fônico (fônico)*
		F+U	[fu]	*fuga (fuga)*

Tabela B.6 Transcrição fonética da consoante G – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
G	/g/	G+A	[ga]	*gatto (gato)*
		G+O	[go]	*goleador (goleador)*
			[gɔ]	*gonna (saia)*
		G+U	[gu]	*gusto (gosto)*
	/dʒ/	G+E	[dʒe]	*gelato (sorvete)*
			[dʒɛ]	*gesto (gesto)*
		G+I	[dʒi]	*giro (giro)*

Tabela B.7 Transcrição fonética da consoante H – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
H	Não há som			*hanno (anos)*

Tabela B.8 Transcrição fonética da consoante L – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
L	/l/	L+A	[la]	*lana (lã)*
		L+E	[le]	*leale (leal)*
			[lɛ]	*lemme (devagar)*
		L+I	[li]	*libro (livro)*
		L+O	[lo]	*locale (local)*
			[lɔ]	*logica (lógico)*
		L+U	[lu]	*lucro (lucro)*

Tabela B.9 Transcrição fonética da consoante M – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
M	/m/	M+A	[ma]	*mare (mar)*
		M+E	[me]	*mela (maçã)*
			[mɛ]	*media (média)*
		M+I	[mi]	*militare (militar)*
		M+O	[mo]	*mobilia (mobilia)*
			[mɔ]	*moda (moda)*
		M+U	[mu]	*muto (mudo)*

Tabela B.10 Transcrição fonética da consoante N – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
N	/n/	N+A	[na]	*nappo (copo)*
		N+E	[ne]	*necessario (necessário)*
			[nɛ]	*nebbia (névoa)*
		N+I	[ni]	*nido (ninho)*
		N+O	[no]	*pino (pinho)*
			[nɔ]	*nomade (nômade)*
		N+U	[nu]	*nudo (nu)*

Tabela B.11 Transcrição fonética da consoante P – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
P	/p/	P+A	[pa]	*padre (pai)*
		P+E	[pe]	*peccato (pecado)*
			[pɛ]	*pezzo (pedaço)*
		P+I	[pi]	*pianto (choro)*
		P+O	[po]	*poeta (poeta)*
			[pɔ]	*Poco (pouco)*
		P+U	[pu]	*puro (limpo)*

Tabela B.12 Transcrição fonética da consoante Q – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
Q	/k/	QU+A	[kwa]	*quattro (quatro)*
		QU+E	[kwe]	*questione (questão)*
			[kwɛ]	*quercia (ccarvalho)*
		QU+I	[kwi]	*quintale (quianta)*
		QU+O	[kwo]	*quotato (cotado)*
			[kwɔ]	*quota (cota)*

Tabela B.13 Transcrição fonética da consoante R – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
R	/r/	R+A	[ra]	*rabbia (raiva)*
		R+E	[re]	*reame (reino)*
			[rɛ]	*resto (troco)*
		R+I	[ri]	*rito (rito)*
		R+O	[ro]	*robusto (forte)*
			[rɔ]	*roba (roupa)*
		R+U	[ru]	*ruota (roda)*

Tabela B.14 Transcrição fonética da consoante S – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
S	/s/	S+A	[sa]	*sapere (talento)*
		S+E	[se]	*sera (tarde)*
			[sɛ]	*sicuro (seguro)*
		S+I	[si]	*siclo (siclo)*
		S+O	[so]	*soave (suave)*
			[sɔ]	*socio (parceria)*
		S+U	[su]	*superficie (superfície)*
	/z/	S+A	[za]	*casa (casa)*

		S+E	[ze]	*arsenale (arsenal)*
			[zɛ]	*esercito (exército)*
		S+I	[zi]	*asino (burro)*
		S+O	[zo]	*bisogno (precisão)*
			[zɔ]	*episódio (episódio)*
		S+U	[zu]	*arsura (sede)*

Tabela B.15 Transcrição fonética da consoante T – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
T	/t/	T+A	[ta]	*taglio (corte)*
		T+E	[te]	*teatrale (teatral)*
			[tɛ]	*tecnico (técnico)*
		T+I	[ti]	*timbro (selo)*
		T+O	[to]	*tombola (bingo)*
			[tɔ]	*topo (rato)*
		T+U	[tu]	*tuffo (mergulho)*

Tabela B.16 Transcrição fonética da consoante V – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
V	/v/	V+A	[va]	*vacante (vago)*
		V+E	[ve]	*venoso (venoso)*
			[vɛ]	*vela (vela)*
		V+I	[vi]	*vino (vinho)*
		V+O	[vo]	*volante (volante)*
			[vɔ]	*volta (giro)*
		V+U	[vu]	*vulcano (vulcão)*

Tabela B.17 Transcrição fonética da consoante Z – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
Z	/ts/	Z+A	[tsa]	*forza (força)*
		Z+E	[tse]	*zecca (casa da moeda)*
			[tsɛ]	*carrozzela (carrocinha)*
		Z+I	[tsi]	*zigano (cigano)*
		Z+O	[tso]	*pzzzo (louco)*
			[tsɔ]	*zoccolo (tamanco)*
		Z+U	[tsu]	*zucchero (açucar)*
	/dz/	Z+A	[dza]	*zaffir (zafira)*
		Z+E	[dze]	*zelare (zelar)*
			[dzɛ]	*zero (zero)*

		Z+I	[dzi]	*Zigomo (osso malar)*
		Z+O	[dzo]	*zodiaco (zodíaco)*
			[dzɔ]	*zótico (grosso)*
		Z+U	[dzu]	*azzurro (azul)*

Tabela B.18 Transcrição fonética dos dígrafos – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
CH	/k/	CH+E	[ke]	*cherubino (querubin)*
			[kɛ]	*archetipo (arquétipo)*
		CH+I	[ki]	*chiave (chave)*
	/ᵈʒ/	GH+E	[ge]	*guepargo (leopardo)*
			[gɛ]	*riguello (régua)*
		GH+I	[gi]	*guirlanda (guirlanda)*
GN	/ɲ/	GN+A	[ɲa]	*assegnare (atribuir)*
		GN+E	[ɲe]	*cagnetto (canino)*
			[ɲɛ]	*magnette (imã)*
		GN+I	[ɲi]	*bagnina (salva-vidas)*
		GN+O	[ɲo]	*agnolo (anjo)*
			[ɲɔ]	*calcagno (calcanhar)*
		GN+U	[ɲu]	*ignudo (nu)*
GL	/ʎ/	GL+I	[ʎi]	*glicerina (glicerina)*
CI	/tʃ/	CI+A	[tʃa]	*arancia (laranja)*
		CI+E	[tʃe]	*ciecamente (cegamente)*
			[tʃɛ]	*cielo (céu)*
		CI+O	[tʃo]	*cioccolato (chocolate)*
			[tʃɔ]	*ciocca (tufo)*
		CI+U	[tʃu]	*acciuga (anchova)*
GI	/dʒ/	GI+A	[dʒa]	*pioggia (chuva)*
		GI+E	[dʒe]	*igienico (higiênico)*
			[dʒɛ]	*igiene (higiene)*
		GI+O	[dʒo]	*adagio (devagar)*
			[dʒɔ]	*gioco (jogo)*
		GI+U	[dʒu]	*giustizia (justiça)*
SC	/ʃ/	SC+I	[ʃi]	*pesci (peixes)*
		SC+E	[ʃe]	*scelta (escolha)*
			[ʃɛ]	*scena (cena)*

Tabela B.19 Transcrição fonética dos trígrafos – língua italiana

Caractere	Trans. Fonética	Letras	Sons	Exemplos
SCI	/ʃ/	SCI+A	[ʃa]	*sciabola (alfange)*
		SCI+E	[ʃe]	*scientifico (científico)*
			[ʃɛ]	*scienza (ciência)*
		SCI+O	[ʃo]	*uscio (porta)*
			[ʃɔ]	*sciopero (greve)*
		SCI+U	[ʃu]	*asciugare (enxugar)*
GLI	/ʎ/	GLI+A	[ʎa]	*paglia (palha)*
		GLI+E	[ʎe]	*paglietta (velejador)*
			[ʎɛ]	*berasagliere*
		GLI+O	[ʎo]	*aglio (alho)*
			[ʎɔ]	*portafoglio (carteira)*
		GLI+U	[ʎu]	*tagliuzzare (retalhar)*
GNI	/ɲ/	GNI+A	[ɲa]	*spegniamo (apagamos)*
		GNI+E	[ɲe]	*
			[ɲɛ]	*
		GNI+O	[ɲo]	*Lagnio (queixar-se)*
			[ɲɔ]	*
		GNI+U	[ɲu]	*ogniuno (cada um)*

As consoantes “j”, “k”, “w”, “x” e “y” tem o uso muito baixo em italiano. A maior parte das palavras que começam com estas letras – ou simplesmente contém estas letras – é de origem estrangeira. É por esta razão que não são realizadas as transcrições fonéticas dessas letras (BABINI, 2002).

A consoante “h” (lê-se “aga”) não foi transcrita foneticamente, pois não há sonoridade para ela. Diante disso, as ocorrências da letra “h” no inicio de palavras serão ignoradas no algoritmo proposto, visto que o “h” foneticamente não tem som expressivo de forma a alterar a pronúncia de uma palavra.

É possível observar na Tabela B.19 que na coluna de exemplos possui alguns registros com asterisco (*), pois não foram encontrados no dicionário da língua italiana exemplos de palavras que utilizam tal letra ou são de menos expressividade e uso. Vale lembrar que o fato de não ter sido encontrada palavras de exemplo com estas letras não descarta a necessidade de suas transcrições fonéticas.