**INSTITUTO INFNET – COMPARAÇÃO ENTRE ALGORITMOS DE CHAVE FONÉTICA PARA NOMES PRÓPRIOS NA LÍNGUA PORTUGUESA**

**Milton Bahia1**

**Roberto Sauer de Paiva Rio1**

**Vinícius Martins Rosa1**

**Raul Sena Ferreira2**

**Eduardo Morelli2**

**Resumo**

Este artigo tem como objetivo abordar, baseados nos conhecimentos de Text Mining e Information Retrieval, a eficiência dos algoritmos de chave fonética que fazem o processamento de linguagem natural para nomes próprios na língua portuguesa. Através de uma análise exploratória e experimental o artigo tem como objetivo demonstrar e comparar o nível de eficiência de alguns algoritmos de chave fonética existentes no mercado. Apresentaremos à conceituação e aplicação de algumas técnicas de Text Mining existentes no mercado, a conceituação e aplicação do processo de teste do experimento e os resultados do experimento. Também apresentaremos de forma introdutória um caso de sucesso e os resultados após a aplicação de um algoritmo de fonética próprio. O cenário atual aponta uma convergência cada vez maior das empresas para aplicação de Text Mining e Information Retrieval, porém como tratamos de um campo muito rico com várias possibilidades, só atingimos a camada superficial de suas aplicações. Dada a variação em larga escala de possibilidades, é preciso juntar a parte conceitual, aplicabilidade de técnicas, ferramentas e fazer com que os envolvidos neste campo continuem a criar e aprimorar novas descobertas para que seja possível obter maiores índices de sucesso e melhorar o apoio a decisão com processamento em larga escala que uma máquina com inteligência aplicada é capaz de realizar.

**Palavras-Chave:** Algoritmo de Chave Fonética. Nomes Próprios. Processamento de Linguagem Natural. Língua Portuguesa. Text Mining. Information Retrieval.

1Pós Graduando do Curso de MIT em Big Data no Instituto INFNET – ([milton.bahia@gmail.com](mailto:milton.bahia@gmail.com)); (robertosprio@hotmail.com); (viniciusmartinsrosa@gmail.com ).

2 MESTRE EM EDUCAÇÃO

3 MESTRE EM EDUCAÇÃO

**1. Introdução**

Uma das finalidades centrais de um **algoritmo de fonética** é permitir a leitura de palavras similares através de uma aproximação para facilitar o entendimento, retorno de consulta e leitura de máquina. Como este campo possui uma grande variedade de oportunidades, também existem diversas barreiras como o idioma (no caso deste artigo **idioma português**) e também as propriedades das palavras, tais como: verbos, adjetivos, advérbios, conjunções e os **nomes próprios**.

Ao longo da evolução da tecnologia, muitas empresas no Brasil e no mundo registraram e armazenaram seus dados sem os devidos controles. As interações para inclusão dos dados foi efetuada por operações humanas. No caso dos nomes próprios, quantos bancos de dados possuem mais de um registro composto por dados similares (digitação incorreta na grafia), mas que podem ser o mesmo dado? Por exemplo: *Nome =* ***Jose;*** *CPF =* ***111.111.111-11;*** *Data de Nascimento =* ***10/10/2010***; e *Nome =* ***Joze;*** *CPF =* ***111.111.111-11;*** *Data de Nascimento =* ***10/10/2010***. A digitação incorreta da grafia acabou se tornando um hábito incorreto e encheu o bando de dados com milhares de casos como no exemplo citado.

De acordo com um **caso real ocorrido em uma Seguradora**, existiam no **banco de dados** aproximadamente **8 milhões de registros**, sendo o **nome da pessoa a chave primária** de busca para o **pagamento das indenizações**. Diante dos erros de grafia o risco de pagamento em duplicidade era elevado. Uma das soluções aplicadas foi a implementação de um algoritmo de fonética criado dentro da própria Seguradora. **Após a implementação do algoritmo** foi possível identificar que houve uma **redução de 8 milhões** de registros **para 6.5 milhões**, ou seja, a **qualidade** apresentada na base era somente de **79%**. Hoje o **nível de assertividade do algoritmo de fonética da Seguradora, supera 90%** evitando o pagamento de indenizações indevidas bem como detectando as quadrilhas fraudadoras do seguro.

Diante do exposto fica evidente que os algoritmos de chave fonética podem influenciar o modo como tratamos os dados. Sendo assim o objetivo deste artigo é apresentar, em caráter exploratório, a comparação de eficiência do algoritmo próprio da Seguradora com alguns algoritmos de chave fonética e técnicas de stemmer disponíveis no mercado. Demonstrando a importância da implementação para aumentar a probabilidade de qualidade e mitigação dos riscos.

Este artigo tem como método de pesquisa a revisão bibliográfica visando adquirir conhecimento sobre a temática citada, e especificamente realizando uma pesquisa exploratória e experimental (testes) através de livros, artigos e internet assim como a própria experiência profissional.

**2. Conceitos de Text Mining**

Para o desenvolvimento deste artigo se faz necessário explicar, mesmo que de forma sucinta, a conceituação dos seguintes temas: Text Mining, Lematização e Stemming.

Segundo Marti Hearst, UC Berkeley, “**Text Mining** é a descoberta do computador de informação nova, até então desconhecida, por meio da extração, para relacionar informações de diferentes fontes escritas e revelar outra forma significados ocultos automaticamente".

A mineração de texto frequentemente usa algoritmos computacionais baseados em processamento de linguagem natural, ou NLP, para permitir que um computador possa "ler" e analisar a informação textual. NLP interpreta o significado do texto e identifica, extrai, sintetiza e analisa fatos e relações relevantes que respondem diretamente a sua pergunta. As consultas baseadas em NLP poderosos podem ser executadas em tempo real, através de milhões de documentos.

Dentre as aplicações da mineração de texto, podemos destacar: A gestão de riscos; A gestão do conhecimento; Prevenção Ciber crime; Serviço de Atendimento ao Cliente; Detecção de fraudes; Publicidade contextual; Business Intelligence; Filtragem de spam; Análise de dados de mídia social.

A normalização morfológica ocorre quando há redução das formas flexionais de uma palavra, através de conflação, a uma forma única que procura representar um conceito ou uma classe de conceitos. Os processos mais comuns de conflação são o **stemming** [FRA 92, KRO 93, ALL 03] e a **lematização** [ARA 00, KOR 04].

**Stemming** é um processo que reduz ao mesmo stem (parte fundamental semelhante ao radical) palavras que se diferenciam basicamente pela flexão, como: ***stemming (livro) = stemming (livros) = livr.***

**Lematização** reduz as palavras variáveis à correspondente forma canônica: verbos no infinitivo e palavras, como substantivos e adjetivos, no singular e, quando existir, masculino. São exemplos: ***lematização (livro) = lematização (livrinho) = livro.***

Diante do exposto, podemos perceber uma variedade de oportunidades de estudo, evolução e exploração. Mas observando a ótica do case citado, como podemos fazer mineração de texto somente para nomes próprios na língua portuguesa?

**3. Tipos de Algoritmo de Chave Fonética**

Um **algoritmo de fonética combina palavras diferentes com pronúncia similar** **ao mesmo código (chave)**, o que permite a comparação através da chave fonética de palavras baseada em semelhança.

Esses algoritmos são muito úteis para pesquisar em listas de pessoas em bancos de dados, bem como para usar em um verificador ortográfico. Eles são freqüentemente usados ​​em combinação com os algoritmos de busca fuzzy, proporcionando aos usuários uma busca acessível por nome e sobrenome em bancos de dados, listas de pessoas e assim por diante. Os algoritmos de codificação fonética visam basicamente: **Identificar palavras com pequenas diferenças entre si; Eliminar erros de escrita; Comprimir dados**.

Os primeiros algoritmos fonéticos foram desenvolvidos para língua inglesa. Destacam-se entre eles o **Soundex e o Metaphone**.

O **Soundex** foi criado em 1918 por Robert Russel e Margaret King Odell com objetivo representar de forma única palavras que tem proximidade fonética mas apresentam grafias diferenciadas. Ele foi utilizado na década de 1930 para análise dos dados do censo americano do período de 1890 a 1920. O Soundex ficou famoso nos anos 1960 em função da publicação de diversos artigos na “Association for Computing Machinery”, especialmente o artigo escrito por Donald Knuth em The Art of Computer Programming. Os principais SGBDS disponibilizam implementações do Soundex, tais como DB2, Oracle, SqlServer e PostGreSQL.

O código Soundex é uma sequência de quatro dígitos. O primeiro dígito é a primeira letra da palavra. Os três dígitos restantes representam as demais consoantes existentes na palavra. A formação do código Soundex segue regras a seguir:

1. Retém a primeira letra da palavra;
2. Ignora as vogais e as letras y, h e w;
3. Troca as consoantes pelo dígito conforme tabela abaixo.
4. Caso existam duas letras consecutivas com o mesmo código, elas são consideradas como uma única letra;
5. Caso seja necessário, o código é completado com zeros para formar os três dígitos.
6. Caso existam mais de três consoantes, elas serão ignoradas.

|  |  |
| --- | --- |
| Lista de Consoantes | Código |
| B, P, F, V | 1 |
| C, G, J, K, Q, S, X, Z | 2 |
| D, T | 3 |
| L | 4 |
| M, N | 5 |
| R | 6 |

Tabela 1- Conversão de Consoantes no Soundex

Foram desenvolvidas variações do Soundex com o objetivo resolver algumas das suas deficiências, dentro os quais o Reverse Soundex, NYSIIS (New York State Identification and Intelligence Systems) e o Daitch–Mokotoff Soundex.

Em 1990, Lawrence Philips apresentou um novo algoritmo fonético denominado de Metaphone, o qual também procura tratar as deficiências do Soundex. Esse algoritmo tem maior precisão para língua inglesa e tem grande aceitação, tendo sido incorporado à biblioteca do PHP.

O Metaphone elimina todas as vogais e considera o conjunto de 16 símbolos: B, X, S, K, J, T, F, H, L, M, N, P, O, W, Y,0; onde o 0 representa o ‘TH’. Letras repetidas são deletada, com a exceção do ‘C’. Seguem as transformações realizadas pelo Metaphone:

|  |  |  |
| --- | --- | --- |
| Letra | Código | Regra |
| B | B | Exceto no final da palavra |
| C | X | Se aparece em –cia-, -ch- |
| S | Se aparece em –ci-, -ce-, -cy- |
| K | nos demais casos |
| D | J | se está dentro de –dge-, -dgy-, -dgi- |
| T | nos demais casos |
| F | F |  |
| G | Nulo | Se aparece em –gh- |
| J | Se está diante de –i-, -e-, -y- |
| K | Nos demais casos |
| H | Nulo | Se aparece depois de vogal e não seguida por vogal |
| H | nos demais casos |  |
| J | J |  |
| K | Nulo | Se aparece depois de vogal e não seguida de vogal |
| K | Nos demais casos |
| L | L |  |
| M | M |  |
| N | N |  |
| P | F | Se aparece diante de H |
| P | Nos demais casos |
| Q | K |  |
| R | R |  |
| S | X | Se aparece antes de –h- ou dentro de –sio-, -sia- |
| S | Nos demais casos |
| T | X | Se aparece em –tia-, -tio |
| 0 | Se aparece diante de –h- |
| T | Nos demais casos |
| V | F |  |
| W | Nulo | Se não está seguida de vogal |
| W | Se está seguida de vogal |
| X | KS |  |
| Y | Nulo | Se não está seguida por vogal |
| Y | Se está seguida por vogal |
| Z | S |  |

Tabela - Transfomações no Metaphone

Posteriormente, Lawrence Philips desenvolveu versões melhoradas do Metaphone, o Double Metaphone e o Metaphone 3. Entretanto, esses algoritmos ainda estão restritos à língua inglesa.

A partir do trabalho de Lawrence Philips, foram desenvolvidas versões do Metaphone para outras línguas, tais como espanhol, alemão, russo, bengali (Bangladesh), amárica (Etiópia) e português.

A versão em espanhol foi desenvolvida em 2005 por Israel J. Sustaíta em PHP. A partir desta versão a equipe da Prefeitura de Várzea Paulista desenvolveu uma versão em português em 2009, dentro do contexto do trabalho realizado para a Promenino Fundação Telefônica.

O Metaphone em português elimina todas as vogais, exceto no início da palavra. Todas as acentuações são retiradas.

Ele considera 19 sons consonantais: “b”(bacia), “d”(conde), ”f”(foca), “g”(fogo), “j”(gente), “k”(doca), ”l”(lote), “λ”¹(alho), “m”(maçã), “n”(cana), “ŋ”²(nhoque), “p”(copo), “r”(tora), “R”(arroz), “s”(centro), “t”(ponte), “v”(vila), “x”(chão), “z”(asa). As consoantes duplicadas que não alteram o fonema são ignoradas.

Segue tabela de conversões do Metaphone em português:

|  |  |  |
| --- | --- | --- |
| Letra | Código | Regra |
| B | B |  |
| C | X | No dígrafo CH |
|  | S | Sucedida por E ou I |
|  | K | Sucedida por A, O, U, R |
| Ç | S |  |
| D | D |  |
| F | F |  |
| G | J | Precedida de E |
|  | G | Demais casos |
| H | NULO | Exceto nos dígrafos CH, NH, LH |
| J | J |  |
| L | 1 | No dígrafo LH |
|  | L | Nos demais caso |
| M | M |  |
| N | M | Se está no final da palavra |
|  | 3 | No dígrafo NH |
|  | N | Nos demais casos |
| P | F | Sucedida por H (PH) |
|  | P | Nos demais casos |
| Q | K |  |
| R | 2 | No caso de RR, no início ou no final da palavra |
|  | R | Nos demais casos |
| S | Z | Caso esteja entre vogais |
|  | X | No caso de SH |
|  | S | Demais casos |
| T | T |  |
| V | V |  |
| W | V |  |
| X | X | 1. No final de palavras 2. Nos casos –exe-, -exi- 3. Precedido por encontro vocálico 4. Precedida pelas sílabas formadas por [C,K,G,L,R,X] +Vogal |
|  | S | Nos casos de –xc-, -xp-, -xt- |
|  | KS | * Nos casos –exa-, -exo-,-exu-, * Precedida por vogal mas não está entre os casos anteriores (3,4) |
|  | X | Em todos os demais caso que não estão nas regras acima |
| Y | I | Recebe o mesmo tratamento das demais vogais |
| Z | S | No final da palavra |
|  | Z | Nos demais casos |

Tabela 3 - Tabela de Conversões do Metaphone BR

Outro algoritmo fonético especificamente desenvolvido para a língua portuguesa foi o BuscaBR. Ele foi proposto por Fred Jorge Tavares da Unibratec em 2007 através de uma análise fonética da língua portuguesa com o objetivo de substituir o Soundex. Ele sofreu melhorias em 2010, através do trabalho de Marcos Rodrigues em 2010 (Caso Unicid), e em 2011 por Gabriel Sobrinho em 2011.

Seguem as regras de transformação do BuscaBR:

1. Converter palavra para letras maiúsculas;
2. Remover acentuação e pontuação (hífen, apóstrofo, etc.);
3. Realizar as substituições conforme a Tabela 4:
4. Remover vogais (A, E, I, O, U) e H;
5. Se a última letra da palavra for M, R ou S, ela deve ser removida;
6. Remover letras duplicadas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DE | PARA | DE | PARA | DE | PARA |
| BR,BL | B | L | R | RM | SM |
| CA | K | N, MD | M | RJ | J |
| CE, CI | S | MG | G | ST, TR, TL | T |
| CO, CU, CK | K | MJ | J | TS | S |
| Ç, CH | S | PH | F | W | V |
| CT | T | PR | P | X | S |
| GE, GI | J | Q | K | ST | T |
| GM | M | RG | G | Y | I |
| GL, GR | F | RS | S | Z | S |
|  |  | RT | T |  |  |

Tabela 4 - Tabela de conversões do BuscaBR

É possível implementar uma variação do BUSCABR com retirada de todas as vogais. Nesse artigo denominamos essa variante de BUSCABR\_SV

Existe uma série de algoritmos de chave fonética para língua inglesa, porém trataremos neste artigo alguns algoritmos de chave fonética e técnicas de stemmer voltados para língua portuguesa, que são: Técnicas de stemmer : **RSLP; SNOW**; Algoritmo de chave fonética: **CHAVES SOUNDEX**; **BUSCABR\_SV**; **BUSCABR**; **METAPHONEBR** e o **ALGORITMO DA SEGURADORA** (citado no case).

**4. Métodos de Teste – Nível de Assertividade**

Durante o processo de criação de um modelo é preciso medir a qualidade dele de acordo com o objetivo da tarefa. Existem funções matemáticas que nos ajudam a avaliar a capacidade de erro e acerto dos nossos modelos. Algumas métricas são mais simples e outras mais complexas, algumas que funcionam melhor para datasets com determinadas características e outras personalizadas de acordo com o objetivo final do modelo. Para esse artigo levaremos em consideração as métricas mais comuns na área de Recuperação da Informação: Precision, Recall, F1 Score e Matriz de Confusão.

A métrica de **Precision** representa a fração dos resultados que são relevantes em uma busca. No que se refere a esse artigo, a relevância está relacionada com a semelhança fonética entre o nome passado como parâmetro de busca e o os resultados retornados na consulta no banco de dados de nomes. O valor é calculado pela equação abaixo, onde VP significa Verdadeiros Positivos e FN significa Falso Negativos.

A métrica **Recall** representa a fração das instancias relevantes que são retornadas numa busca. Como a relevância está relacionada com a semelhança fonética entre o nome passado como parâmetro de busca e o resultado da consulta no banco de dados de nomes, essa métrica indica a fração dos nomes semelhantes recuperados na busca em relação ao total de nomes semelhantes presentes no banco de dados de nomes. O valor é calculado pela equação abaixo, onde VP significa Verdadeiros Positivos e FN significa Falso Negativos.

A métrica **F1 Score** é uma média harmônica entre **Precision** (que, apesar de ter o mesmo nome, não é a mesma citada acima) e **Recall**. Ela é muito boa quando você possui um dataset com classes desproporcionais, e o seu modelo não emite probabilidades. Isso não significa que não possa ser usada com modelos que emitem probabilidades, tudo depende do objetivo de sua tarefa. Em geral, quanto maior o F1 score, melhor.

A técnica **Matriz de confusão** é uma tabela que permite a visualização do desempenho de um algoritmo de aprendizado, tipicamente em um algoritmo supervisionado. Cada coluna da matriz representa as instâncias de uma classe prevista, enquanto as linhas representam os casos de uma classe real. O nome é originado do fato de que a matriz torna mais fácil verificar se o algoritmo está confundindo duas classes.

|  |  |  |
| --- | --- | --- |
|  | Positivo | Negativo |
| Positivo | Verdadeiro Positivo | Falso Positivo |
| Negativo | Falso Negativo | Verdadeiro Negativo |

**5. O experimento**

O experimento deste artigo consistiu em comparar a eficiência de algoritmos de chave fonética e técnicas de stemmer (**RSLP; SNOW**; **CHAVES SOUNDEX**; **BUSCABR\_SV**; **BUSCABR**; **METAPHONEBR** e o **ALGORITMO DA SEGURADORA)** através das métricas de **Precision, Recall, F1score e Matriz de Confusão**. Além dos algoritmos citados, foi avaliado o efeito da aplicação de um algoritmo de distância de edição de Levenshtein de forma subsequente aos algoritmos de citados.

Para realização do experimento foram realizadas as seguintes etapas:

* Seleção de uma base de dados com nomes próprios.
* Geração das chaves fonéticas de cada algoritmo para todos os nomes.
* Preparação dos cenários de testes com a busca e o retorno esperado
* Realização de consultas com a chave gerada por cada algoritmo
* Comparação dos resultados obtidos com o retorno esperado
* Cálculo das métricas para cada algoritmo

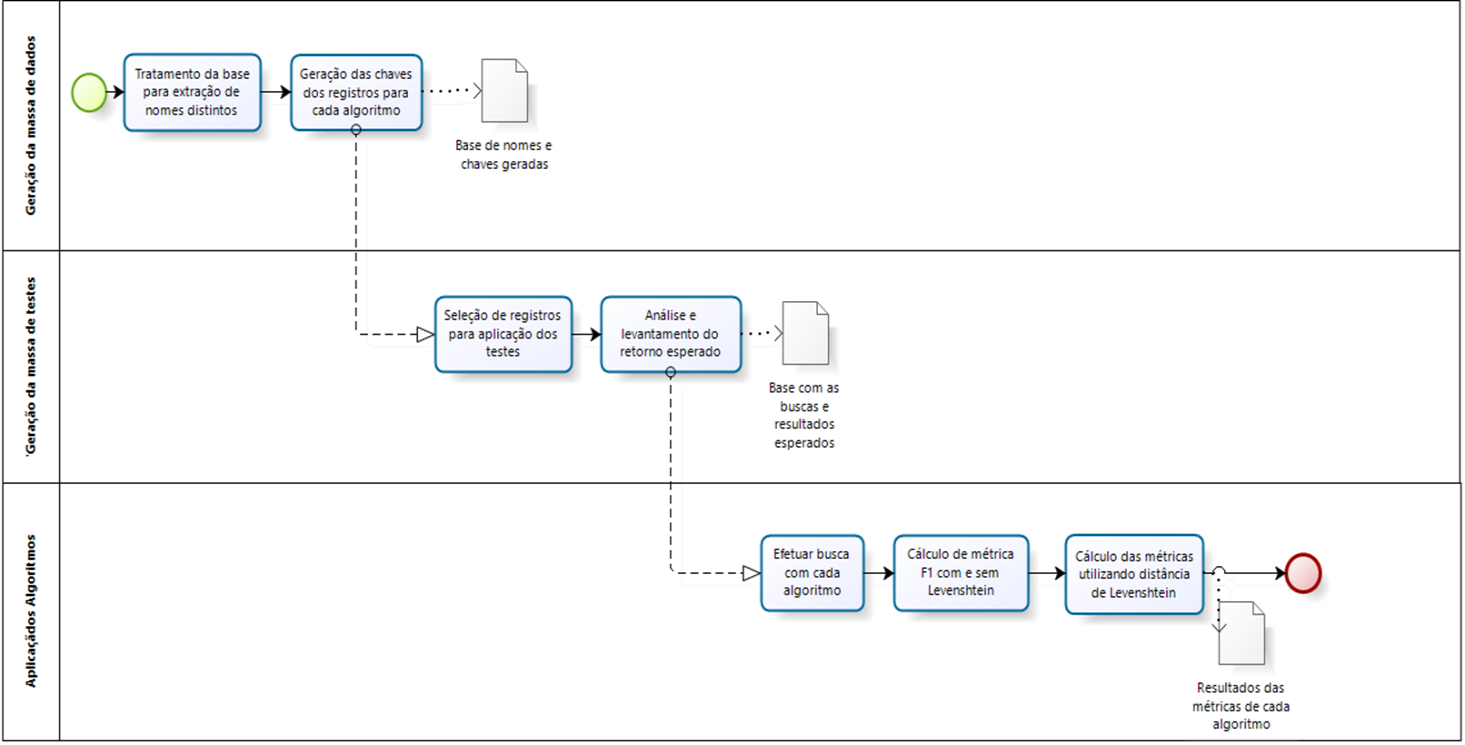


Figura - Fluxo de realização do experimento

O banco de dados de nomes próprios continha 3290 registros. Não foram utilizados nomes compostos e a acentuação foi retirada previamente. Para cada nome no banco de dados foi gerada a chave de todos os algoritmos de chave fonética avaliados, conforme exemplo na Tabela 5. O objetivo da geração prévia dessas chaves foi possibilitar as buscas na etapa de testes.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **NOME** | **CHAVES GERADAS POR CADA ALGORITMO** | | | | | | |
| **BUSCABR\_SV** | **BUSCABR** | **SEGURADORA** | **METAPHONEBR** | **RSLP** | **SNOW** | **CHAVESOUNDEX** |
| ABDALA | BDR | ABDARA | BDLA | ABDL | ABDAL | ABDAL | A134 |
| ABDIAS | BD | ABDIA | BDIAS | ABDS | ABD | ABDI | A132 |
| BENJAMIN | BJM | BEJAMI | BNJMIN | BNJMM | BENJAMIN | BENJAMIN | B525 |
| BENTO | BMT | BEMTO | BINT | BNT | BENT | BENT | B53 |
| BERENICE | BRM | BEREMI | BRNSI | BRNS | BEREN | BERENIC | B652 |
| CATARINA | KTRM | KTARIMA | CTRNA | KTRN | CATARIN | CATARIN | C365 |
| CELESTE | SRT | SRETE | SLSTI | SLST | CEL | CEL | C423 |
| CELIA | SR | SRIA | SLIA | SL | CEL | CEL | C4 |
| DANIELA | DMR | DAMIERA | DNLA | DNL | DANIEL | DANIEL | D54 |
| DANILO | DMR | DAMIRO | DNLO | DNL | DANIL | DANIL | D54 |
| EDGAR | DG | EDGA | IDGR | EDG2 | EDG | EDGAR | E326 |
| EDMUNDO | DM | EDMUMO | DMNDO | EDMND | EDMUND | EDMUND | E353 |
| LUISA | RS | RUISA | LUIS | LZ | LUIS | LUIS | L2 |
| LUIZ | R | RUI | LUIS | LS | LUIZ | LUIZ | L2 |
| MANOEL | M | MAMOE | MNOIL | MN | MANOEL | MANOEL | M54 |
| MANOELA | MR | MAMOERA | MNOILA | MNL | MANOEL | MANOEL | M54 |
| RAFAEL | RF | RAFAE | RFAIL | 2RF | RAFAEL | RAFAEL | R14 |
| SILVIA | SRV | SIRVIA | SLIA | SV | SILV | SILV | S41 |
| VAGNER | VM | VAME | UAGNR | VGN2 | VAGN | VAGN | V256 |
| WALDIR | VRD | VARDI | UALDR | VD2 | WALD | WALD | W436 |
| ZACARIAS | SKR | SAKRIA | SCRIAS | ZKRS | ZAC | ZAC | Z262 |
| ZULEIDE | SRD | SUREIDE | SLDI | ZLD | ZULEID | ZULEID | Z43 |

Tabela - Banco de Nomes com Chaves Fonéticas

Para avaliação do desempenho de cada algoritmo, foi montado um conjunto de duzentos casos de testes com o nome de referência para busca e o retorno esperado. No retorno esperado podem existir múltiplos nomes similares à referência passada para o algoritmo de testes (vide Tabela 6).

A execução dos testes consistiu na realização das buscas dos nomes de referência no Banco de Dados de Nomes utilizando a chave gerada por cada algoritmo. Para cada sequência de testes foram avaliados o número de acertos (Positivos) e erros (Falsos Positivos e Falsos Negativos). Esses valores foram utilizados para cálculo das métricas de desempenho.

De forma simultânea foi avaliado o aumento de eficiência utilizando o algoritmo de distância de edição de Levenshtein. O cálculo de distância foi aplicado para cada resultado das buscas com o objetivo de reduzir o conjunto de valores retornados. Foi utilizado como parâmetro de comparação uma distância de edição menor ou igual a 2. As métricas propostas para o experimento foram novamente avaliadas após a aplicação do algoritmo de distância.

|  |  |
| --- | --- |
| **BUSCA** | **RETORNO ESPERADO** |
| ADEMAR | ADEMHAR , ADEMAR , ADHEMAYR , ADHEMAR |
| ALESSANDRA | ALESSANDRA , ALEXANDRA |
| BENEDITO | BENEDITO , BENEDICTO , BENEDETTO |
| BERNADETE | BERNARDETH , BERNARDET , BERNADETE |
| CHRISTIAN | CHRISTIAN , CRISTIAN |
| CLEUSA | CLEUSA , CLEUZA , CREUSA , CREUZA |
| CRISTIANO | CHRISTIANNO , CRISTIANO |
| DONISETE | DONISETE , DONIZETE , DONIZETTE , DONIZETTI |
| EDER | EDER , HEDER |
| EDGAR | EDGAR , EDGARD |
| ELIANA | HELIANA , ELIANA , HELEANA , ELIANE |
| ELIZABETE | ELIZABETE , ELIZABETH , ELISABETE |
| HEBER | HEBER , EBER , HEBERT |
| HELOISA | HELOISA , HELOIZA , ELOISA , ELOIZIA , ELUIZA |
| IVETE | IVETTE , IVETE , IVETH , YVETE , YVETTE |
| MATEUS | MATEUS , MATHEUS |
| OSEIAS | OSEIAS , OSEAS , OZEAS , OZIAS |
| ROSEMERI | ROSEMARY , ROSEMERI , ROSEMARIE , ROSEMARI , ROZIMERE , ROSIMARY |
| VANDERLEI | VANDERLEI , WANDERLEI , WANDERLEY |
| WELINGTON | WELINGTON , WELITON , WELLINGTON , WELLINGUITON |

Tabela - Exemplos de casos de testes

**6. Análise dos resultados**

De acordo com os testes realizados, conseguimos verificar as diferenças entre cada algoritmo, sendo notável a semelhança da maior parte dos algoritmos de cada classe, sendo elas stemmers ou algoritmos fonéticos e que a aplicação de soluções auxiliares, como o de Distância de Levenshtein nos dá um ganho significativo na abrangência do algoritmo.

Vimos também que o algoritmo mais eficiente varia de acordo com a massa de dados utilizada para os testes, portanto quanto maior a massa de testes utilizada, mais conseguimos retirar o viés de um ou outro algoritmo.

Como é possível verificar no gráfico da Figura 2, a maior precisão foi obtida no algoritmo do BuscaBR, entretanto os algoritmos de stemmer (Snowball e RLSP) tiveram um desempenho superior aos outros algoritmos de chave fonética. Esse resultado dos algoritmos é coerente pois eles fazem uma pouca ou nenhuma redução nos nomes próprios, de tal forma que a quantidade de resultados é muito próxima ao valor exato da busca.

No gráfico da Figura 3, referente à métrica de Recall (abrangência), verificamos uma diferença acentuada entre os algoritmos de stemmer em relação aos de chave fonética. Esse resultado é esperado pois esse tipo de algoritmos recupera um conjunto de valores muito restrito, não conseguindo realizar um agrupamento de nomes semelhantes. Novamente o algoritmo com melhor desempenho nessa métrica foi o BuscaBR. O Soundex teve um desempenho ruim nessa métrica pois ele foi elaborado para o idioma inglês. A retirada da vogal do algoritmo do BuscaBR\_SV diminui muito a sua precisão, em função da redução da chave e do alto grau de agrupamento de nomes para uma mesma chave.

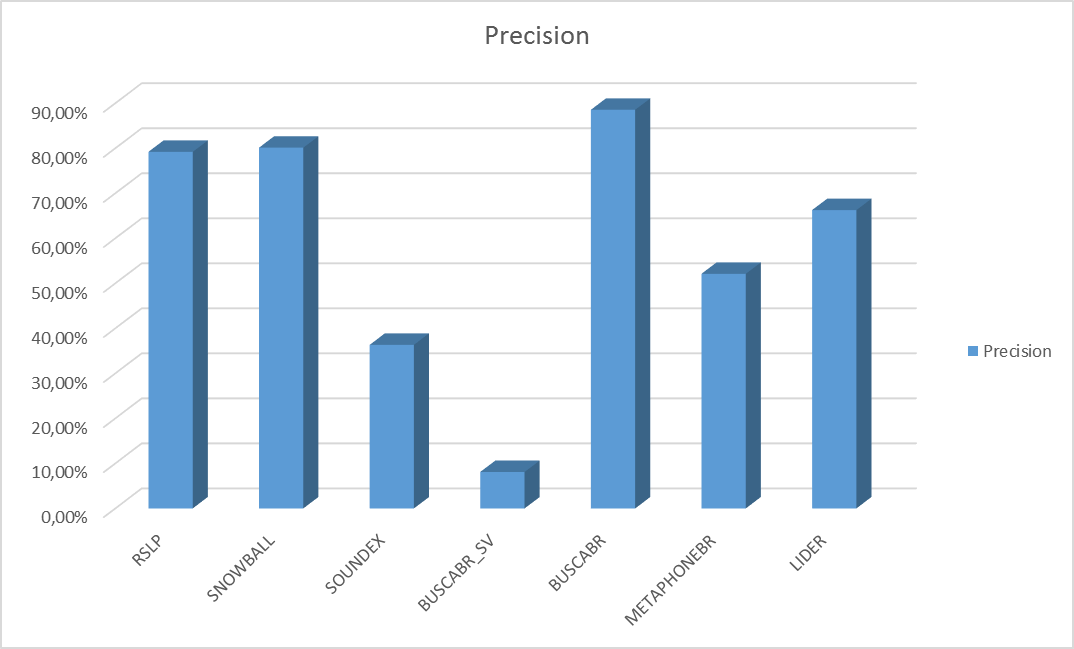


Figura - Gráfico de resultado da métrica Precision

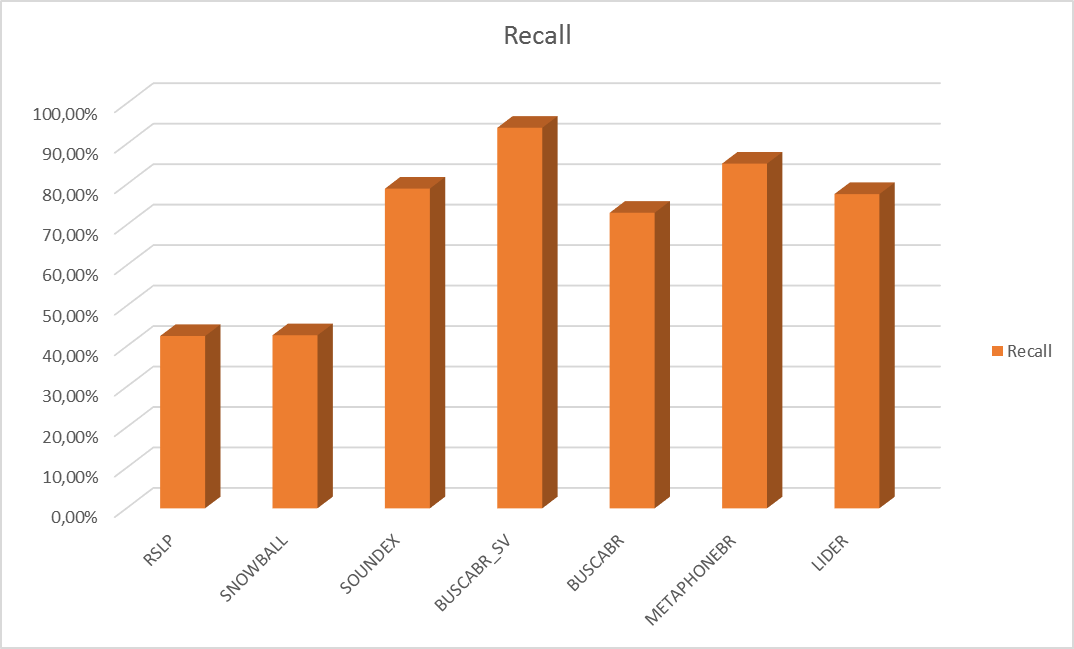
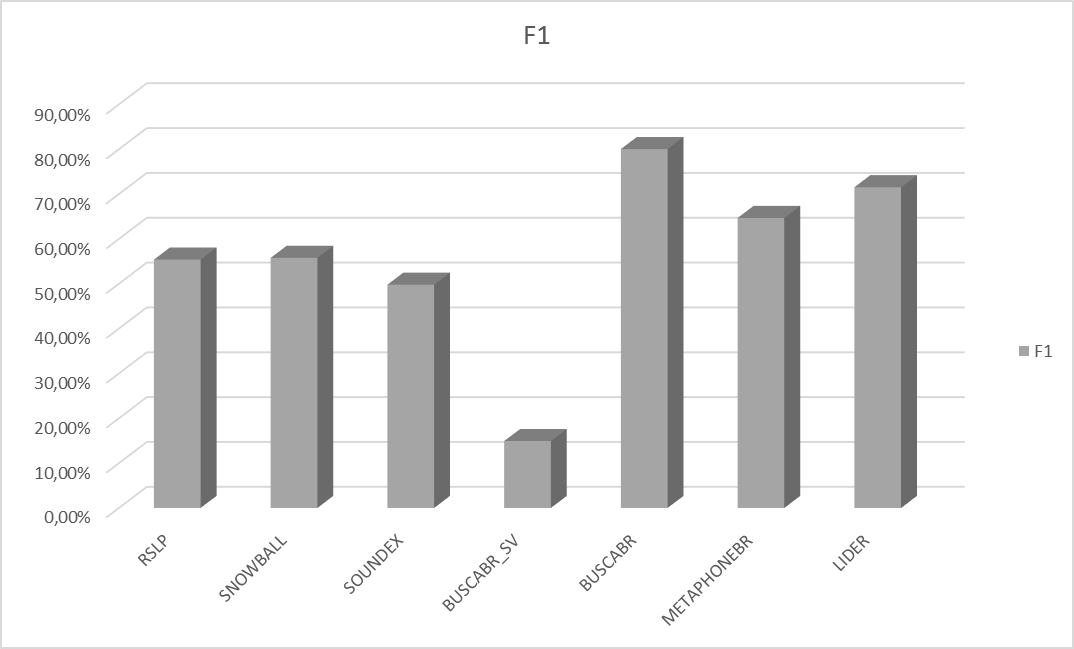


Figura - Gráfico da métrica Recall

Na xx apresentamos o gráfico referente à métrica F1. Para a massa utilizada nos testes o algoritmo que melhor se saiu foi o **BUSCABR**(sem retirada de vogais). Os algoritmos de stemmer tem um desempenho inferior aos algoritmos de chave fonética. Novamente os algoritmos do Soundex e do BuscaBR\_SV não tem um desempenho satisfatório.



A utilização da distância de Levenshtein como auxiliar, melhora o desempenho de todos os algoritmos.

**7. Conclusão**

Após o término de nosso experimento foi possível constatar que cada algoritmo de fonética tem uma particularidade que se aplica melhor a cada cenário, por exemplo, quando estamos tentando detectar possíveis erros de escrita em nomes, trocas comuns na língua portuguesa, os algoritmos **METAPHONE** e **BUSCABR,** já quando a necessidade é de diferenciação de fonemas, casos em que um mesmo nome pode ser escrito de maneiras diferentes porém tendo uma estrutura fonética similar, o algoritmo da **LIDER** ganha o seu destaque.

Constatamos também que de acordo com a composição dos nomes na massa de dados na qual serão aplicados os algoritmos, os resultados podem variar, uma vez que cada algoritmo tende a resolver um problema ligeiramente diferente do outro, e que na maioria dos casos essas regras para solução acabam entrando em conflito.

A aplicação de algoritmos auxiliares, como foi o caso da distância de Levenshtein podem ajudar a aumentar a eficiência do algoritmo nos casos em que estamos buscando nomes em uma base, porém em cenários onde a comparação de nomes é feita para identificar registros em duplicidade, isso pode ser um ofensor e acabar resultando em falsos positivos.

**Abstract**

This article aims to address the efficiency of the phonetic key algorithms that make the natural language processing for proper names in the Portuguese Language, based on the knowledge of Text Mining and Information Retrieval. Through an exploratory and experimental analysis the article aims to demonstrate and compare the level of efficiency of some phonetic key algorithms available in the market. We will present to the conception and application of some existing techniques of Text Mining available in the market, the conception and application of the test process and the results of the experiment. We will also present an introductory case of success and the results after the application of a proper phonetic algorithm. The current scenario indicates an increasing convergence of companies for the application of Text Mining and Information Retrieval, but as we deal with a very rich field with several possibilities and we only reach the superficial layer of its applications. Given the large-scale variation of possibilities, it is necessary to combine the conceptual part, applicability of techniques, tools and make those involved in this field continue to create and improve new discoveries to achieve higher success rates and improve support to large-scale processing that a machine with applied intelligence is capable of performing.

**Key Words:** Phonetic Key Algorithm. Proper Names. Natural Language Processing. Portuguese Language. Text Mining. Information Retrieval.

**8. Referências**

http://mariofilho.com/as-metricas-mais-populares-para-avaliar-modelos-de-machine-learning/