Relatório de Estágio: Desenvolvimento e Aplicação de Métodos Automáticos de Avaliação de Impacto de Pesquisadores

Aluna: *Dânia N. O. Meira*
Professor: Ricardo da Silva Torres
Responsável: Leonardo B. Oliveira
Empresa: Elabora Consultoria

4 de novembro de 2011

Resumo

Este projeto tem por objetivo o desenvolvimento de um sistema de classificação automática de pesquisadores através de métricas que medem o impacto de suas publicações. A lista de publicações é extraída do currículo Lattes dos pesquisadores e as métricas podem ser calculadas a partir de duas bases de dados: Journal Citations Report (JCR) da Thomson Reuters e Qualis da CAPES. Após a obtenção de tais métricas pretende-se montar sistemas de classificação para acompanhar o desenvolvimento de grupos de pesquisadores.

1 Introdução

Os pesquisadores brasileiros disponibilizam através da Plataforma Lattes informações sobre sua vida acadêmica como por exemplo dados sobre sua formação, atuação profissional, linhas de pesquisa, área de atuação, produção científica, tecnológica e artística, participação em eventos entre outros. A partir do Currículo Lattes do pesquisador é possível então a obtenção de uma lista de publicações de artigos científicos por ele publicados.

Depois da obtenção desta lista de publicações é necessário associar a estas uma métrica adequada para classificar os pesquisadores conforme sua produção científica. Esta métrica pode ser o Fator de Impacto dos canais de publicação em que foram publicados seus trabalhos, encontrado no JCR ou o estrato que é dado pelo Qualis, também por canal de publicação.

Uma das principais dificuldades encontradas neste trabalho é fazer a associação ao realizar uma busca dentro das bases de dados, seja dos nomes das publicações ou dos nomes dos pesquisadores, por ser necessário o casamento automático entre as *strings*, ou seja, como saber que *string1* corresponde a *string2*.

Isto abrange lidar com nomes que podem ser iguais mas não serão um casamento exato uma vez que o casamento exato só acontece nos casos em que exatamente todos os caracteres das duas *strings* são os mesmos, e na mesma ordem. Os nomes podem ser iguais mas diferenciar, por exemplo em

- *Maiúsculas/minúsculas:* Journal of Information and Data Management / journal of information and data management / JOURNAL OF IINFORMATION AND DATA MANAGEMENT
- *Acentuação:* Revista Eletrônica de Iniciação Científica / Revista Eletronica de Iniciacao Cientifica
- *Abreviações:* International Journal Of Information Technology & Decision Making / Int J Inf Tech Decis
- *Espaços duplos:* Computer Networks / Computer Networks

Contribuição: O objetivo deste trabalho é desenvolver um algoritmo que realize o casamento aproximado entre *strings* de maneira mais eficiente possível, ou seja, mais próximo possível da decisão humana se duas *strings* estão se referindo ao mesmo nome ou não o que tornará possível encontrar com maior exatidão os índices referentes a cada canal de publicação indexado nas bases de dados.

Organização: Este trabalho está organizado da seguinte maneira: na Seção 2 encontra-se uma explicação sobre os métodos e técnicas que serão abordados para a execução deste projeto. Em seguida, na Seção 3, está apresentado o cronograma de trabalho. Finalmente na Seção 4 conclui-se com as referências bibliográficas.

2 Revisão Teórica

Iniciando pelo problema do casamento automático entre *strings* primeiramente serão estudados tratamentos de *strings* para melhoria do casamento exato. Alguns tratamentos são, por exemplo, alteração para somente maiúsculas ou minúsculas; remoção de espaços em branco antes ou depois da string, acentuação, palavras entre parêntesis.

Após obtenção do melhor resultado para casamento exato, será atacado o problema do casamento aproximado entre *strings*. Para isto, três algoritmos diferentes serão analisados: Metaphone, Soundex e Levenstein. O melhor resultado será aquele que se aproximar melhor da escolha humana: aceitar as *strings* como iguais quando realmente são e diferentes quando não são. Dizemos ser um falso positivo quando o algoritmo toma as *strings* como iguais quando elas realmente não são; e um falso negativo quando ele assume não serem iguais porém na realidade são.

2.1 Algoritmos de Casamento Aproximado Entre *Strings*

Segue descrição dos três algoritmos estudados:

1. **Levenshtein**

   A distância Levenshtein ou distância de edição entre duas *strings* é uma medida, dada pelo número mínimo necessário de operações: inserção, deleção ou substituição de caracteres, para transformar uma *string* na outra. Quanto maior a distância Levenshtein mais diferentes as *strings* são.

   Em geral, o algoritmo para calcular a distância Levenshtein apresenta uma solução que usa programação dinâmica. Inicia-se com uma matriz de tamanho m x n onde m e n correspondem ao tamanho de cada uma das duas *strings* que se deseja comparar. Tal matriz é preenchida a partir do canto superior esquerdo até o canto inferior direito. Cada salto horizontal ou vertical corresponde a uma inserção ou exclusão, respectivamente. O

salto diagonal pode implicar as duas operações, se os dois caracteres na linha e na coluna não coincidem, ou nenhuma operação se eles forem iguais. O custo é normalmente definido como 1 para cada uma das operações. Cada célula sempre minimiza o custo localmente. Desta forma, o número no canto inferior direito é a distância Levenshtein entre ambas as palavras.

Por exemplo a comparação entre “meilenstein” e “levenshtein”.

		m	e	i	l	e	n	s	t	e	i	n
	0	1	2	3	4	5	6	7	8	9	10	11
l	1	1	2	3	3	4	5	6	7	8	9	10
e	2	2	1	2	3	3	4	5	6	7	8	9
v	3	3	2	2	3	4	4	5	6	7	8	9
e	4	4	3	3	3	3	4	5	6	6	7	8
n	5	5	4	4	4	4	3	4	5	6	7	7
s	6	6	5	5	5	5	4	3	4	5	6	7
h	7	7	6	6	6	6	5	4	4	5	6	7
t	8	8	7	7	7	7	6	5	4	5	6	7
e	9	9	8	8	8	7	7	6	5	4	5	6
i	10	10	9	8	9	8	8	7	6	5	4	5
n	11	11	10	9	9	9	8	8	7	6	5	4

Figura 1: Matriz distância Levenshtein

O preenchimento desta matriz é realizado da seguinte maneira: Coloca-se uma das palavras na coluna e outra na linha, e inicia-se o preenchimento desta matriz pelas primeira linha e coluna. Primeiro temos zero operações na primeira posição da matriz pois iniciamos com nenhum caractere. Ao avançar na primeira linha, a segunda posição totaliza uma operação, que é de inserção (nenhuma letra para letra ´´m”); a terceira posição soma mais uma operação, de substituição de caracteres (letra ´´m” para letra ´´e”), totalizando duas operações e assim por diante até completarmos a palavra “meilenstein”. Da mesma forma é o preenchimento para a primeira coluna.

Continuando o preenchimento na segunda linha, por exemplo, cada posição é o número total de operações necessárias para ir da letra ´´l” de “levenshtein” (que es colhemos colocar na linha) para cada uma das letras da palavra “meilenstein” (que colocamos na coluna): de ´´l” até ´´m” uma substituição; de ´´l” até ´´me” temos a substituição anterior mais uma, totalizando duas operações; de ´´l” até ´´mei” temos que fazer as substituições anteriores mais outra, totalizando três operações; de ´´l” até ´´meil” precisamos fazer três inserções, total de três operações, e assim por diante. O procedimento pode ser realizado analogamente se é desejado preencher pelas colunas em vez das linhas.

Desta forma, o valor na última posição da última linha corresponde ao número de operações feitos para sair de uma das palavras e chegar à outra.

2. **Soundex**

   O algoritmo Soundex é baseado num sistema de codificação a partir do som de consoantes e foi criado no século 19 pelo governo americano com o objetivo de estudar dados censitários. Por exemplo "Smith" e "Smyth" são considerados iguais porque, apesar de terem grafia diferente, tem a mesma pronúncia e neste caso o mesmo código.

   O código Soundex para um nome é composto por uma letra seguida de três dígitos numéricos. A letra é a primeira letra do nome, e os dígitos codificam as consoantes restantes. Consoantes que soam similares compartilham a mesma codificação então por exemplo as consoantes labiais B, F, P, V são codificadas com o número 1. Vogais podem afetar a codificação mas não são codificadas exceto se forem a primeira letra. No entanto, se "h" ou "w" separar duas consoantes que tem o mesmo código soundex, a consoante à direita da vogal não é codificada.

Número	Representa as letras
1	B, F, P, V
2	C, G, J, K, Q, S, X, Z
3	D, T
4	L
5	M, N
6	R

Tabela 1: Código Soundex

3. **Metaphone**

   Metaphone é um algoritmo fonético publicado em 1990 para indexar palavras pela sua pronúncia em inglês. É mais preciso do que o Soundex porque utiliza um conjunto maior de regras para a pronúncia em inglês. Seu código utiliza 16 caracteres: 0BFHJKLMNPRSTWXY onde o ´´´0" representa o fonema "th", ´´X" representa "sh" ou "ch" e as outras consoantes representam sua pronúncia usual em inglês. As vogais AEIOU também são utilizadas, porém somente no início do código. Segue o procedimento completo:

   (a) Desconsiderar a segunda letra de letras duplicadas, exceto pela letra C;

   (b) Se a palavra iniciar por 'KN','GN','PN','AE','WR', desconsiderar a primeira letra;

   (c) Desconsiderar 'B' se vier depois de 'M' e se estiver no final da palavra;

   (d) Trocar 'C' por 'X' se vier seguido de 'IA' ou 'H'; por 'S' se vier seguido de 'I', 'E' ou 'Y'; por 'K' nos outros casos;

   (e) Trocar 'D' por 'J' se seguido de 'GE', 'GY' ou 'GI'; por 'T' nos outros casos;

(f) Desconsiderar ‘G’ se seguido de ‘H’ e se ‘H’ não estiver no final da palavra nem antes de uma vogal e se seguido de ‘N’ ou ‘NED’ e estiver no final da palavra;

(g) Trocar ‘G’ por ‘J’ se estiver antes de ‘I’, ‘E’ ou ‘Y’; por ‘K’ nos outros casos;

(h) Desconsiderar ‘H’ se seguido de vogal mas não depois de uma vogal;

(i) Trocar ‘CK’ por ‘K’;

(j) Trocar ‘PH’ por ‘F’;

(k) Trocar ‘Q’ por ‘K’;

(l) Trocar ‘S’ por ‘X’ se seguido de ‘H’, ‘IO’ ou ‘IA’;

(m) Trocar ‘T’ por ‘X’ se seguido de ‘IO’ ou ‘IA’; Trocar ‘TH’ por ‘0’; desconsiderar ‘T’ se seguido de ‘CH’;

(n) Trocar ‘V’ por ‘F’;

(o) Se a palavra começa com ‘WH’ desconsiderar ‘H’; Desconsiderar ‘W’ se não seguido de uma vogal;

(p) Se a palavra começa com ‘X’ então trocar ‘X’ por ‘S’ senão trocar por ‘KS’;

(q) Desconsiderar ‘Y’ se não seguido por vogal;

(r) Trocar ‘Z’ por ‘S’;

(s) Vogais são mantidas somente quando são a primeira letra da palavra;

(t) Em todos os outros casos as letras são mantidas as mesmas.

Todos os três algoritmos estão implementados em Python: Levenshtein.ratio que recebe como parâmetro as duas *strings* e retorna a porcentagem de similaridade entre elas, calculada a partir da sua distância de edição; módulo *advas*, que fornece algoritmos para buscas avançadas em geral utilizados para recuperação de informação e linguística, tem implementação de ambos Soundex e Metaphone

Ao analisar todas as possibilidades foi decidida a utilização do algoritmo Levenshtein em vez do Soundex ou Metaphone pois estes são algoritmos fonéticos que funcionam muito bem com palavras na língua inglesa e o conjunto de palavras aqui não é somente de palavras em inglês. Desta forma o algoritmo Levenshtein se mostra a melhor escolha uma vez que é universal (funciona para quaisquer linguagens/fonemas).

2.2 Métricas de Classificação

Os dois principais bancos que foram utilizados para fazer classificações foram o Journal Citations Report (JCR), que contém a informação do Fator de Impacto por canal de publicação e o Qualis, que contém o estrato do canal de publicação. Apresentamos estas métricas a seguir:

1. **Thomson ISI Journal Impact Factor (JIF)** *(Garfield, 1960's)*

   Reflete o número médio de citações a um artigo de um dado periódico dentro do período de 1-2 anos depois de sua publicação.

   Definição: nº de citações num ano particular (2009 por ex.) a todos os artigos publicados num dado periódico nos dois anos anteriores (2007 e 2008 por ex.) dividido pelo nº total de "source items" publicados nos dois anos anteriores (2007 e 2008 por ex.)

   [1]

2. **Estratos do Qualis**

   Os canais de publicação são separados em Estratos: A1, A2, B1, B2, B3, B4, B5 e C.

   Cada estrato tem sua definição específica por área, e tal definição está descrita detalhadamente nos arquivos de Critérios disponíveis no site do WebQualis.

   Em geral as definições seguem recomendação do CTC-ES (Conselho Técnico Científico da Educação Superior) no que diz respeito a requisitos para o periódico ser avaliado (ter registro no ISSN por exemplo) e dos limites da cota de cada estrato (percentual de periódicos em cada estrato não superar alguma porcentagem), utilização do fator de impacto calculado pelo JCR (mediana por exemplo), além de utilizar informações de periódicos indexados em outras bases (Scopus, Google Scholar e outras que são mais relevantes para a área específica). As informações de outras bases podem ser tanto a existência ou não do periódico na base quanto o índice-h.

   As recomendações estão no Ofício Circular nº 049/2009 - PR/CAPES de 10/2/2009, aprovado formalmente pelo CTC-ES.

3 Metodologia e Resultados

O estudo seguiu os seguintes passos experimentais:

3.1 Retirada dos nomes dos canais de publicação a partir do Currículo Lattes

A partir de uma seleção de 889 professores de programas de pós-graduação em Ciência da Computação foram obtidos 2103 diferentes canais de publicação. Estes correspondem aos canais de publicação nos quais os pesquisadores tiveram seus artigos publicados, relatados nos seus Currículos Lattes. Os currículos deste conjunto foram armazenados no formato xml.

[1] "source items" são artigos regulares, ou seja, excluem-se cartas, resenhas e editoriais de livros, resumos de conferências.

Para a retirada dos canais de publicação foi utilizado um script em perl que varre o xml do Currículo Lattes em busca das produções bibliográficas ao utilizar expressões regulares como por exemplo ´´produçãoemc_t_and_a", ´´producao-bibliografica", ´´artigos_completos_publicados_em_periódicos", e a partir disto separa o nome do periódico de outras informações (autores e nome do artigo entre outros).

3.2 Utilização do filtro para homogeneizar as strings

Para fazer a associação da *string* que corresponde ao nome do periódico com o estrato do Qualis e com o Fator de Impacto do JCR tanto a amostra quanto os bancos, que contém um campo tipo texto para o nome do periódico e outro campo que contém a classificação, tiveram as strings primeiramente homogeneizadas a partir de um filtro implementado em Python que realiza os seguintes tratamentos:

- retirada de espaços em branco antes e depois da *string*;
- mudança de todos os caracteres para minúsculos;
- retirada de acentuação;
- remoção de informação entre parêntesis;
- remoção da expressão regular 'v.';
- substituição de '&' por 'and'.

3.3 Retirada de *stopwords* em inglês

Para um melhor desempenho do casamento aproximado também foi feito um estudo sobre *stopwords*. Dado que dentro do nosso escopo existem nomes de periódicos tanto em português quanto em inglês foi realizado um estudo para cada uma destas línguas. Ao retirar *stopwords* em inglês houve um ganho de aproximadamente 20% enquanto que as *stopwords* em português resultaram numa perda de quase 5% no casamento aproximado. Investigações indicam que isto ocorre porque a amostra está predominantemente em inglês e ainda as *stopwords* em português estão presentes dentro da *string* em inglês o que causa uma perda de partes do nome do periódico. Portanto é utilizado somente o tratamento para o inglês.

3.4 Calibragem do algoritmo Levenshtein

O algoritmo Levenshtein retorna uma porcentagem de similaridade entre duas *strings*, mas como saber o que esta porcentagem significa? Implica que as palavras são iguais ou que são diferentes? Para resolver isto é preciso encontrar um valor que faça esta distinção, que a partir dele as palavras sejam, na maioria dos

casos, iguais e antes dele sejam diferentes. Ou seja, um limiar de similaridade entre as *strings*.

Saber se as palavras são ou não iguais faz parte de um julgamento humano. Por isto é necessário trabalhar com uma amostra manual para definição do limiar. Esta amostra deve permitir que a partir dela o estudo seja estendido e portanto o valor encontrado funcione como um bom limiar para o conjunto total, também referido como população. Vale ressaltar aqui que o limiar ótimo encontrado dependerá portanto dos conjuntos de palavras a serem comparadas e por este motivo a cada mudança em quaisquer um dos conjuntos é necessária uma nova calibragem.

Sejam:

n = tamanho da amostra;
N = tamanho total da população;
e = erro aceitável;
$m = \frac{1}{e^2}$;

O tamanho da amostra[2] é dado por :
$n = \frac{m.N}{m+N}$

Podemos definir a taxa de erro em 5%, que é o caso mais geral, portanto $e =$ 0,05. O tamanho total da população é igual a todos os canais de publicação pertencentes à lista retirada dos Currículos Lattes, menos os canais de publicação que já são encontrados em cada universo com o casamento exato de *strings*, uma vez que estes não precisam do algoritmo de casamento aproximado.

Para selecionar esta amostra vamos pegar números aleatórios gerados pelo site *random.org* [3]: gerar n números aleatórios dentro do intervalo [1,2103] sem nenhuma repetição. Os números aleatórios obtidos correspondem às linhas do arquivo de saída do algoritmo Levenshein que serão o conjunto trabalhado manualmente.

Este arquivo de saída do algoritmo é um .txt separado por tabulações com três colunas, sendo a primeira o nome do veículo de publicação dado como entrada, a segunda a porcentagem de similaridade calculada, e a terceira o nome encontrado na lista do universo. Por exemplo uma linha seria: revista brasileira odontologia 0.873 Revista Brasileira de Odontologia.

A checagem manual significa olhar as duas strings relacionadas pelo algoritmo e verificar se elas correspondem ao mesmo nome de periódico ou não. Registra-se esta informação numa nova coluna, preenchida com 1 quando as strings são correspondentes e 0 quando não são.

A partir deste arquivo com a checagem manual e com o arquivo de saída do algoritmo é montada uma matriz de comparação que reúne as informações do algoritmo e a informação humana para avaliar quando esse se aproxima mais do último, seria a chamada matriz de avaliação:

[2]Estatística Aplicada às Ciências Sociais, Cap. 3 Pedro Alberto Barbetta. Ed. UFSC, 5ª Edição, 2002

[3]http://www.random.org/sequences/

	M	L0.600	L0.625	L0.650	...	L0.975
Nome1	1	0	0	0	...	1
Nome2	1	0	0	1	...	1
Nome3	1	0	0	0	...	1
...						

Onde:

M : coluna que será preenchida manualmente com um valor binário sendo 1 quando os nomes forem iguais e 0 quando não forem.

L0.x : colunas que serão preenchidas automaticamente com um valor binário sendo 1 quando a porcentagem de similaridade dada por Levenshtein (encontrada no arquivo de saída do algoritmo) for maior ou igual ao valor correspondente da coluna e 0 quando for menor.

No exemplo dado acima, como a porcentagem calculada pelo algoritmo de Levenshtein é 0.873, e manualmente podemos verificar que os nomes são iguais, teremos a seguinte linha correspondente na matriz:

	M	L0.600	...	L0.850	L0.875	...	0.975
Revista Brasileira de Odontologia	1	1	1	1	0	0	0

Agora, podemos verificar se o algoritmo acertou ou errou a partir da porcentagem que ele calculou. Para isto vamos comparar, em cada linha, os valores contidos na coluna M com os valores em cada uma das demais colunas L0.x. Um contador será criado para cada uma das porcentagens Levenshtein e, quando a comparação disser que os valores são iguais vamos adicionar uma unidade no contador, dizendo que é mais um acerto do algoritmo, ou seja, vamos contar todos os verdadeiros positivos e também os verdadeiros negativos.

Ao final da leitura da matriz teremos em cada contador a respectiva nota, ou seja, a quantidade de acertos que Levenshtein tem para cada porcentagem. Dividindo este valor por n, o tamanho da amostra, temos a média de acertos e pegando a maior média temos a correspondente porcentagem que indica o melhor resultado (o mais próximo possível de um matching manual) que podemos obter com o algoritmo de Levenshtein.

3.5 Realização do casamento aproximado de *strings* utilizando o algoritmo Levenshtein calibrado

Encontrado o limiar basta que este seja implementado no script do algoritmo Levenshtein de forma que cada vez que for calculada a porcentagem de similaridade entre duas strings, este valor seja comparado com o limiar. Quando a porcentagem é maior do que o limiar o algoritmo diz que as strings comparadas são iguais e quando a porcentagem é menor então o algoritmo diz que elas não são iguais.

Lembrando que a saída do algoritmo é um arquivo .txt separado por tabulações no qual cada linha corresponde a uma dupla de strings que foi comparada e o valor 0 ou 1 dependendo do resultado da comparação, para gerar a lista que relaciona somente as strings que foram encontradas no universo, ou seja, somente as comparações que deram igual, basta utilizar o comando *awk* para remove as linhas que tem o valor zero entre as strings, ficando somente com as linhas que tem valor um.

3.6 Associação do índice de classificação com o nome do canal de publicação

Agora temos dois arquivos: uma lista com as strings encontradas dentro do universo e a lista do universo que contém o canal de publicação e sua dada métrica.

Para relacionar os canais de publicação encontrados com a sua métrica é utilizado um script que com o auxílio do comando *awk* realiza uma busca da string contida na primeira coluna do arquivo de canais de publicações encontrados na primeira coluna do arquivo do universo, e retornando o valor da segunda coluna do arquivo do universo.

Para o universo JCR foram encontrados 735 periódicos utilizando apenas o casamento exato, correspondendo a 35,0% do total de periódicos trabalhados. Portanto o tamanho total da população utilizado foi $N = 2103$ - $735 = 1368$, e a amostra correspondente teve tamanho $n = 309$.

A partir disto, o limiar ótimo do algoritmo Leveshtein encontrado para esta amostra foi 92,5%, como mostram os gráficos nas Figuras 2 e 3.

Com o algoritmo Levenshtein foram encontrados mais 48 periódicos, um ganho portanto de 3,5%. O total de periódicos encontrados foi então 783, o que corresponde a 37,2% do total.

Já para o universo Qualis foram encontrados 639 periódicos utilizando apenas o casamento exato, correspondendo a 30,1% do total de periódicos trabalhados. Portanto o tamanho total da população utilizado foi $N = 2103$ - $639 = 1464$, e a amostra correspondente teve tamanho $n = 314$.

Então para esta amostra, a partir dos gráficos nas Figuras 4 e 5, o limiar ótimo do algoritmo Leveshtein encontrado foi 90,0%.

Com o algoritmo Levenshtein foram encontrados mais 75 periódicos, um ganho portanto de 5,1%. O total de periódicos encontrados foi então 714, o que corresponde a 34,0% do total.

4 Conclusão

Para avaliar o quão satisfatório foi o número encontrado de periódicos, em cada um dos casos, também foi realizado um estudo com amostra manual para a existência dos nomes nos universos JCR e Qualis.

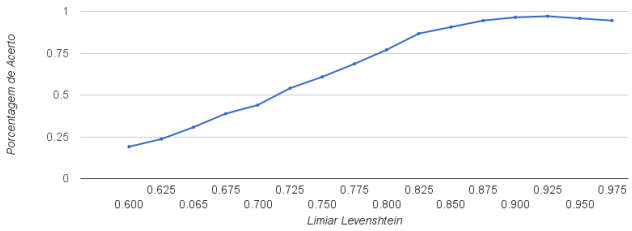

Figura 2: Possíveis Limiares Ótimos: Universo JCR

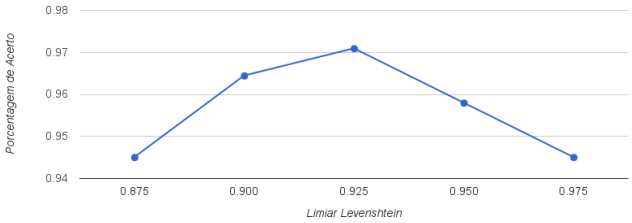

Figura 3: Limiar Ótimo Encontrado: Universo JCR

Foi descoberto então que, a partir da amostra manual do universo JCR, aproximadamente 38,4% dos periódicos retirados dos Currículos Lattes estão indexados neste universo. Deste total de 38,4% conseguimos encontrar 37,2%, o que implica um resultado final de 96,98%. Tal resultado é consistente e podemos observar isto na inexistência de falsos positivos dentro da amostra manual, ou seja, pela amostragem não deve haver nenhum ganho falso.

Para o universo Qualis, a partir da amostra manual, aproximadamente 36,6% estão indexados. Isto implica que conseguimos encontrar 34% a partir do total de 36,6% que poderíamos encontrar, atingindo um resultado de 92,75%. Este resultado se mostra consistente quando se verifica dentro da amostra manual do estudo de calibragem do algoritmo que apenas 1 das 314 *strings* é um falso positivo, ou seja, existe um falso ganho de 0,3% apenas.

Apesar das métricas aqui utilizadas já conseguirem caracterizar o desempenho acadêmico ainda existem outras possíveis, que são calculadas utilizando o número de citações de cada artigo em vez de considerar o veículo de publicação do artigo. Uma delas por exemplo é o h-index. Para estudos futuros este número de citações será buscado em diversos meios, por exemplo Google Scholar [4], ArnetMiner [5], Microsoft Academic Search [6] para geração de diferentes *rankings* de classificação, dados diferentes pontos de vista.

[4]http://scholar.google.com.br/
[5]http://arnetminer.org/
[6]http://academic.research.microsoft.com/

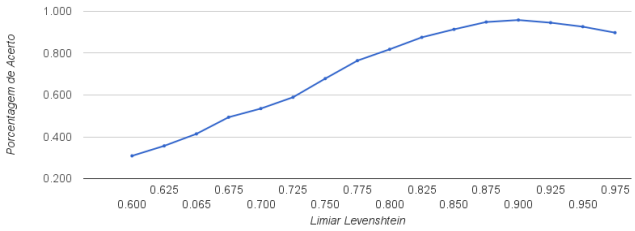

Figura 4: Possíveis Limiares Ótimos: Universo Qualis

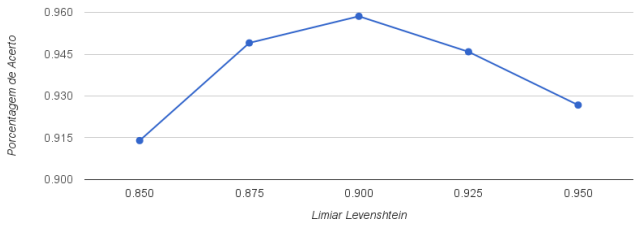

Figura 5: Limiar Ótimo Encontrado: Universo Qualis

Referências

[1] `http://docs.python.org`

[2] `http://www.postgresql.org`

[3] `http://www.levenshtein.net/`

[4] `http://www.archives.gov/research/census/soundex.html`

[5] `http://www.capes.gov.br/avaliacao/qualis`

[6] `http://qualis.capes.gov.br/webqualis`

[7] `http://www.scientific.thomson.com/tutorials/jcr4/jcr4tut5a.html`

[8] `http://thomsonreuters.com/products_services/science/free/essays/impact_factor`

[9] Reis, D.C. e Fonseca, B.M. (2002) O fantástico mundo da distância de edição