Comparação de similaridade entre textos utilizando os algoritmos *Dice's coefficient* e *Longest Common Substring*

Sandro Henrique Uliana Catabriga[1]

[1]Departamento de Informática
Universidade Estadual de Maringá (UEM) - Maringá, PR - Brazil

ra98397@uem.br

Abstract. *This paper aims to present analyzes and results of the performance of the Dice's coefficient and Longest Common Substring algorithms to verify similarity between texts, in the processing of four methodically selected base cases. Throughout the work, the concepts that involve the algorithms are clarified, as well as the techniques used for evaluation. The obtained results revealed superior performance of the Longest Common Substring algorithm when tested in the test cases.*

Resumo. *Este trabalho visa apresentar análises e resultados do desempenho dos algoritmos Dice's coefficient e Longest Common Substring para verificação de similaridade entre textos, no processamento de quatro casos base selecionados metodicamente. Ao decorrer do trabalho são esclarecidos os conceitos que envolvem os algoritmos, bem como as técnicas utilizadas para avaliação. Os resultados obtidos revelaram um superior desempenho do algoritmo Longest Common Substring quanto posto à prova nos casos de teste.*

1. Introdução

Medir a similaridade entre palavras, sentenças ou textos, vem se tornando um importante elemento em diversas aplicações, tais como classificação de textos, recuperação de informações, determinação de pontuação para redações, entre outras [Gomaa and Fahmy 2013]. Encontrar similaridade entre palavras é um aspecto fundamental para verificação de similaridade entre textos [Gomaa and Fahmy 2013], e, devido a isso, utilizar um algoritmo que realize um processamento preciso e confiável das palavras é essencial.

Com poucas exceções, a abordagem típica para encontrar a similaridade entre dois textos é a utilização de um método léxico de *matching*, produzindo um score de similaridade baseado no número de unidades léxicas que ocorrem em ambos os textos [Mihalcea et al. 2006].

Este trabalho buscar analisar, por meio de comparação de resultados, a aplicação de dois algoritmos léxicos *String-Based*[1], o *Longest Common Substring* - do tipo *Character-Based*, e o *Dice's coefficient* - do tipo *Term-Based*. Para tal análise, realizou-se testes a fim de inferir qual algoritmo obteve melhor performance quando comparado aos resultados esperados.

[1]Algoritmos léxicos utilizam sequências de caracteres para inferir similaridade. *String-Based* se refere às operações de comparação utilizando sequências de palavras e composição de caracteres.

2. *Dice's coefficient*

2.1. Descrição do algoritmo

Dice's coefficient é um algoritmo estatístico, originalmente desenvolvido com princípios aplicativos à biologia, porém, muito utilizado em outras áreas do conhecimento, dentre elas, subdivisões que aplicam e necessitam de análise de padrões e similaridade entre textos. E ainda como cita [Duarte et al. 1999], o algoritmo possui extrema eficiência de projeção no espaço bidimensional, como é caso da similaridade entre textos analisada neste trabalho.

No contexto abordado neste trabalho, o algoritmo é utilizado para medir a similaridade entre dois textos baseando-se no número de *bigrams*[2] resultante da interseção entre os excertos. A medida estatística do algoritmo pode ser definida como observado na equação 1, em que n_t representa o número de *bigrams* encontrados nos dois textos, ou seja, a interseção, n_x representa o número de *bigrams* do texto x e n_y o número de *bigrams* do texto y.

$$d = \frac{2n_t}{n_x + n_y} \tag{1}$$

2.2. Implementação

O algoritmo foi implementado utilizando o paradigma de programação funcional, por meio da linguagem Racket.

Buscou-se abstrair a utilização do método por meio de um *template*, alimentado por dois parâmetros, *texto1* e *texto2*, sendo estes os elementos a serem processados pela técnica descrita anteriormente.

Para processamento interno, fez-se uso de dois procedimentos principais, o primeiro para busca de *bigrams* dos textos de entrada, e o segundo para cálculo da medida estatística, sendo este último alimentado com os resultados do primeiro procedimento.

3. *Longest Common Substring* (LCS)

3.1. Descrição do algoritmo

Longest Common SubString (LCS) é um algoritmo que considera a similaridade entre duas strings baseado no tamanho da cadeia contígua de caracteres existentes em ambas as strings [Gomaa and Fahmy 2013]. [Kondrak 2005] define a formulação padrão do LCS como segue:

> "Dada uma sequência $X = x_1...x_k$, outra sequência $Z = z_1...z_m$ é subsequência de X se existe uma sequência estritamente crescente $i_1...i_m$ de índices de X tal que para todo $j = 1, ..., m$, temos $x_{i_j} = z_j$."

O algoritmo é popularmente implementado de forma recursiva, com recursão apresentada na figura 1. Uma normalização do resultado também foi necessária, pois o algoritmo retorna o tamanho da subsequência. A equação 2 representa a normalização, em

[2] *Bigrams* podem ser definidos como pares de caracteres adjacentes em um string.

que l_{xy} é o tamanho da subsequência retornado pelo algoritmo LCS, l_x é o tamanho da string X e l_y o tamanho da string Y.

$$LCS(X_i, Y_j) = \begin{cases} \emptyset & \text{if } i = 0 \text{ or } j = 0 \\ LCS(X_{i-1}, Y_{j-1}) \hat{} x_i & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max\{LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)\} & \text{if } i, j > 0 \text{ and } x_i \neq y_j. \end{cases}$$

Figura 1. Recursão do algoritmo LCS

$$d = \frac{l_{xy} \times 0.5}{l_x} + \frac{l_{xy} \times 0.5}{l_y} \tag{2}$$

3.2. Implementação

Da mesma forma que o algoritmo anterior, o LCS foi implementado utilizando o paradigma de programação funcional, por meio da linguagem Racket.

Para fins de abstração, utilizou-se um procedimento alimentado por dois elementos, *texto1* e *texto2*, que internamente realiza o processo exemplificado na figura 1 e posteriormente realiza a normalização, como descrita na equação 2.

4. Metodologia

Neste trabalho, busca-se conjecturar a respeito da eficiência dos algoritmos *Dice's coefficient* e LCS na verificação de similaridade entre textos, quando comparados a resultados esperados. Devido a isso, fez-se necessário o uso de mecanismos ou métricas para que uma análise dos resultados seja aferida.

Para tal, definiu-se um conjunto base de oito textos, nos quais a cada dois textos fora inferido um resultado prévio tomado como resultado ótimo, ou seja, quatro pares de textos para aplicação de métricas de avaliação dos algoritmos. Utilizou-se, então, dois mecanismos de avaliação: erro relativo percentual entre resultados unitários e coeficiente de correlação de Pearson.

O erro relativo percentual entre resultados unitários é definido na equação 3, em que v_a é o valor obtido como resultado do algoritmo, e v_e é o valor exato, isto é, o resultado esperado. Esta métrica é capaz de nos informar, individualmente, o quão distante cada processamento realizado pelos algoritmos está do resultado esperado. Para casos em que o resultado esperado é 0, consideraremos um erro relativo percentual de $v_a \times 100$, apenas para fins de análise.

$$d = \frac{|v_a - v_e|}{v_e} \times 100 \tag{3}$$

Em pesquisas geralmente busca-se verificar se há relação entre diferentes conjuntos aleatórios de dados, isto é, descobrir se alterações ocorridas em um dos conjuntos acompanham alterações ocorridas no outro. Partindo de tal princípio, fez-se uso do coeficiente de correlação de Pearson para que posteriormente fosse possível inferir se os resultados obtidos por meio dos algoritmos acompanham de alguma forma os resultados esperados.

Os textos utilizados para teste e os resultados esperados foram constituídos como segue:

- Textos 1 e 2: foram elaborados de forma a terem palavras parecidas, porém, contextos totalmente diferentes. Impôs-se um resultado esperado de $0,0$.
- Textos 3 e 4: o texto 4 contém 50% do conteúdo do texto 3, sendo seu restante um conteúdo distinto. Estipulou-se um resultado esperado de $0,50$.
- Textos 5 e 6: simulam a cópia de referências em um trabalho. Cada texto contém três parágrafos, em que cada parágrafo do texto 6 possui uma referência do texto 5. As referências representam uma composição de aproximadamente um terço do parágrafo. Resultado esperado de $0,33$.
- Textos 7 e 8: são textos perfeitamente iguais, portanto o resultado esperado é $1,00$.

5. Resultados e Discussão

Nesta seção, devotaremo-nos a apresentar resultados obtidos na simulação dos algoritmos, e então analisá-los baseando-se nas métricas apresentadas anteriormente.

5.1. Coeficiente de correlação de Pearson

Com base na execução dos testes propostos na seção anterior, podemos notar por meio da tabela 1, que o algortimo LCS aproximou-se mais do resultado ótimo em 75% dos casos se comparado ao algoritmo *Dice's coefficient*. Constatamos também que o LCS obteve melhor coeficiente de correlação, isto é, o conjunto de resultados produzido pelo LCS possui uma variação que acompanha mais fielmente os resultados ótimos.

Pautando-se nos resultados aferidos, é evidente o melhor desempenho do algoritmo LCS para verificação de similaridade de textos dos casos teste propostos. Isso pode ser explicado devido ao fato de strings isoladas e iguais em ambos os textos não necessariamente fazerem parte de uma solução para o LCS, pois podem não fazer parte da cadeia crescente, no entanto, seus *bigrams* necessariamente farão parte de uma solução para o *Dice's coefficient*. Tal aspecto pode impactar negativamente os resultados produzidos pelo *Dice's coefficient*.

	Dice's coefficient	LCS	Resultado esperado
Textos 1 e 2	0,6160	0,4063	0,00
Textos 3 e 4	0,8024	0,5217	0,50
Textos 5 e 6	0,8815	0,4933	0,33
Textos 7 e 8	1,00	1,00	1,00
Correlação	0,9042	0,9435	

Tabela 1. Tabela de resultados

5.2. Erro relativo percentual entre resultados unitários

Partindo dos dados apresentados na figura 2, podemos notar que, para cada teste realizado, com exceção de um caso, o algoritmo LCS fica menos distante do resultado esperado.

Utilizando uma análise restrita e individual dos resultados obtidos pelos algoritmos, podemos conjecturar que o LCS possui um desempenho melhor que seu concorrente,

errando percentualmente menos em 75% dos testes. Isso pode ser explicado devido a estatística utilizada pelo *Dice's coefficient*, que busca por padrões, os *bigrams*, que acabam ocorrendo de forma mais numerosa que os buscados pelo LCS. Tal fato acabou depreciando os resultados obtidos pelo *Dice's coefficient* quando posto à prova nos casos teste. Vale a ressalva que, em um outro contexto, os resultados dos algoritmos poderiam se inverter.

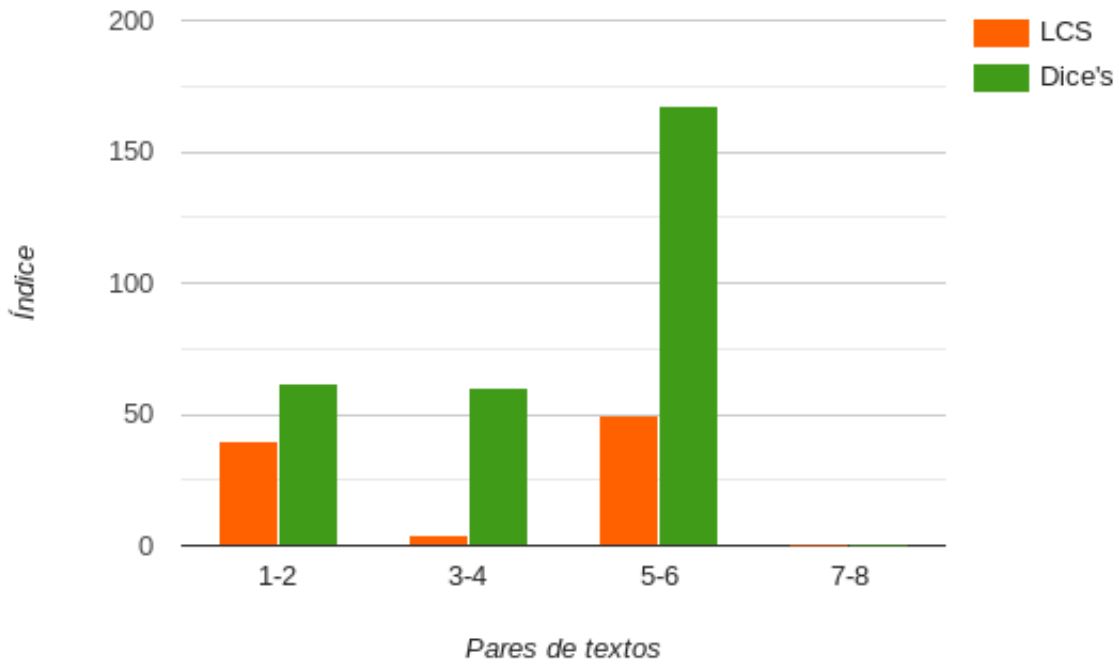

Figura 2. Erro relativo percentual entre os resultados unitários

6. Conclusão

Algoritmos de verificação de similaridade entre textos possuem inúmeras aplicações no mercado e na academia. Devido a isso, a escolha de um algoritmo que obtenha resultados precisos é extremamente importante.

Como discutido por meio dos resultados, o algoritmo LCS se mostrou mais acurado na verificação de similaridade entre textos, levando em consideração uma análise precedente para obtenção de resultados ótimos. Isso demostra que a técnica adotada pelo LCS, a *Character-Based*, consegue um desempenho melhor que a do seu concorrente, ainda que muito parecidas.

Seria necessária a aplicação de um maior número de testes com algoritmos que utilizem outras técnicas para podermos inferir uma conclusão mais precisa. Ainda assim, podemos perceber que técnicas parecidas - *Charactere-Based* e *Term-Based*, aplicadas em algoritmos com princípios claramente diferentes, podem produzir resultados completamente desiguais.

Referências

Duarte, J. M., Santos, J. B. d., and Melo, L. C. (1999). Comparison of similarity coefficients based on rapd markers in the common bean. *Genetics and Molecular Biology*, 22(3):427–432.

Gomaa, W. H. and Fahmy, A. A. (2013). A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13):13–18.

Kondrak, G. (2005). N-gram similarity and distance. In *International symposium on string processing and information retrieval*, pages 115–126. Springer.

Mihalcea, R., Corley, C., Strapparava, C., et al. (2006). Corpus-based and knowledge-based measures of text semantic similarity. In *Aaai*, volume 6, pages 775–780.