Avaliação de medidas de similaridade de texto para remoção de ambiguidade de nome de autores *

Matheus Lemos[1], **Daniel R. Figueiredo**[1], **Fábio H. Botler**[1]

[1]Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro (UFRJ)

lemos25matheus@poli.ufrj.br, daniel@cos.ufrj.br, fbotler@cos.ufrj.br

Abstract. *Name disambiguation consists of identifying different names that appear in a bibliographic database that refer to the same author. One of the primitives to tackle this problem consists of string similarity metrics applied to pairs of author names. This article evaluates the performance of three string similarity measures (Levenshtein, LCS, TLSH) using a real database with more than 10 thousand authors with more than one name and a universe of 7.3 million different names. A methodology based on ordering the distances of the names is applied to more accurately compare the different similarity metrics. Results clearly indicate that the LCS is superior to the others, but still does not adequately identify the synonymous names in a large fraction of cases.*

Resumo. *A remoção de ambiguidade de nome consiste em identificar nomes diferentes que aparecem em uma base bibliográfica que remetem ao mesmo autor. Uma das primitivas para atacar este problema consiste de medidas de similaridade de string aplicada a pares de nomes dos autores. Este artigo avalia o desempenho de três medidas de similaridade de string (Levenshtein, LCS, TLSH) utilizando uma base de dados real com mais de 10 mil autores com mais de um nome e um universo de 7,3 milhões de nomes distintos. Uma metodologia baseada na ordenação das distâncias dos nomes é utilizada para comparar mais justamente as diferentes medidas de similaridade. Resultados claramente indicam que a LCS é superior as demais, mas ainda assim não identifica adequadamente os nomes sinônimos em uma grande fração de casos.*

1. Introdução

Muitas bases bibliográficas de artigos científicos possuem autores cujos nomes aparecem mais de uma vez com grafias diferentes. Em particular, ao considerar diferentes artigos científicos de coautoria de uma mesma pessoa, seu nome pode aparecer com diferentes grafias na lista de coautores dos diferentes artigos. Por exemplo, o ilustre John von Neumann possui publicações registradas como "J Neumann" e "J von Newmann" na base bibliográfica do Google Scholar. Identificar os indivíduos que estão associados aos nomes que aparecem na lista de coautores de artigos científicos é um problema clássico,

*O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. M. Lemos foi parcialmente apoiado pelo CNPq (PIBIC/UFRJ), D. R. Figueiredo foi parcialmente apoiado pelo CNPq (312552/2020-3), F. Botler foi parcialmente apoiado pelo CNPq (423395/2018-1), FAPERJ (211.305/2019) e UFRJ (23733). FAPERJ é a Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro. CNPq é o Conselho Nacional de Desenvolvimento Científico e Tecnológico.

conhecido por *remoção de ambiguidade* ou *resolução de entidade*. Este problema é fundamental para consolidar os dados e atribuir crédito à coautoria dos artigos científicos.

Muitas técnicas foram e vem sendo propostas para atacar o problema de remoção de ambiguidade de nome, focando em diferentes abordagens, tais como o uso de algoritmos de aprendizado de máquina [Zhou et al. 2004, Han et al. 2004], e o uso de informações estruturais induzidas pela rede de colaboração [Gomide et al. 2021, Yang et al. 2008]. Entretanto, praticamente todas as abordagens propostas utilizam uma função para medir a similaridade entre dois nomes. Ou seja, dado duas sequências de caracteres (strings) que representam a grafia de dois nomes, n_1 e n_2, uma função $f(n_1, n_2)$ determina a distância entre os dois nomes. Por exemplo, f pode ter propriedades tais como $f(n_1, n_2) = 0$ apenas quando as strings n_1 e n_2 são idênticas. Em muitas abordagens, a função f é um parâmetro do método ou até mesmo uma mistura ponderada de diferentes funções de similaridade [Ferreira et al. 2012, Sanyal et al. 2021].

A forma específica de f tem um papel central no processo de remoção de ambiguidade, mas sua definição não é trivial, uma vez que as strings n_1 e n_2 correspondem a nomes de pessoas e não são strings quaisquer. Por exemplo, considere as strings n_1 ="J Neumann", n_2 ="J von Newmann" e n_3 ="J Neilman". Intuitivamente, os nomes n_1 e n_3 parecem remeter a pessoas distintas (por conta da diferente grafia do sobrenome), enquanto os nomes n_1 e n_2 parecem remeter a mesma pessoa (por conta da ausência do conectivo "von"). Curiosamente, funções clássicas de similaridade de strings, tal como a distância de edição, irão concluir precisamente o contrário!

O objetivo deste trabalho é avaliar a eficácia de diferentes métricas de similaridade de strings quando aplicadas a nome de pessoas. Em particular, determinar a proximidade relativa de um par de nomes com grafias distintas de uma mesma pessoa para diferentes métricas de similaridade, em comparação aos outros nomes que aparecem em uma mesma base bibliográfica real. No exemplo acima, assumindo que n_1 e n_2 remetem a mesma pessoa, gostaríamos de avaliar a relação de $f(n_1, n_2)$ com $f(n_1, n_3)$, considerando também diferentes nomes n_3 e diferentes funções de similaridade. Entretanto, os valores de distância de diferentes medidas de similaridade podem não ser comparáveis, e por isso uma metodologia baseada na ordenação dos nomes será apresentada e utilizada na avaliação.

O estudo realizado neste trabalho avalia três métricas de similaridade de strings relativamente ortogonais tendo em vista a abordagem usada para medir similaridade: (i) distância de *Levenshtein* [Levenshtein 1966], (ii) Maior Subsequência Comum (LCS, e.g., [Cormen et al. 2009]), (iii) TrendMicro *Locality Sensitive Hash* (TLSH) [Oliver et al. 2013]. Tais métricas foram avaliadas na base bibliográfica DBLP [Ley 2002], na qual é possível identificar pessoas que possuem nomes com grafias diferentes (utilizando o ORCID), e fornecer também o universo de nomes distintos para determinar a proximidade relativa (através da ordenação) entre os diferentes nomes da mesma pessoa (veja Seção 2).

O restante deste artigo está organizado da seguinte maneira. A Seção 2 apresenta a metodologia de comparação entre as métricas similaridade, junto à descrição do dataset utilizado na avaliação. A Seção 3 apresenta as três métricas de similaridade que foram avaliadas juntamente com um pequeno exemplo. Detalhes do software e algoritmos de-

senvolvidos para a avaliação das métricas são apresentados na Seção 4. Os resultados são apresentados e discutidos na Seção 5. Por fim, a Seção 6 apresenta uma breve conclusão e discussão do trabalho.

2. Metodologia e Dataset

Inicialmente, dada uma base bibliográfica, devemos determinar um conjunto de pares de nomes (grafias) distintos que certamente remetem a uma mesma pessoa. Naturalmente, resolver este problema não é trivial. Entretanto, muitas bases bibliográficas utilizam o *registro ORCID*[1], um identificador único universal para cada pessoa, independentemente de outros aspectos tais como grafia do nome ou afiliação. Dessa forma, o ORCID foi utilizado para construir o gabarito de pessoas que possuem mais de um nome com grafia distinta, e seus respectivos nomes, que é o seu objetivo original.

Cada nome do gabarito será utilizado como âncora no processo de avaliação da métrica de similaridade. Considere uma função f e dois nomes distintos de uma mesma pessoa, n_1 e n_2. Considerando n_1 como o nome âncora, determinamos $f(n_1, x)$ para todo nome x que aparece na base bibliográfica (independentemente do nome estar associado a um ORCID). Os nomes são então ordenados de forma crescente de acordo com tais distâncias. Repare que n_2 é um dos nomes da base, e logo possui uma posição, digamos $g(n_1, n_2)$, nesta ordenação. Esta posição será usada para medir a proximidade entre n_1 e n_2. Repare que o objetivo não é avaliar o valor absoluto de $f(n_1, n_2)$, mas sim a ordenação induzida por f e a posição de n_2 nesta ordenação. Uma boa função de similaridade deve produzir baixos valores para g. É importante notar que apesar das medidas de similaridade (a função f acima) serem simétricas, usar n_1 como âncora para avaliar a posição de n_2 na ordenação é diferente de usar n_2 como âncora para avaliar a posição de n_1 na ordenação, e, consequentemente, $g(n_1, n_2) \neq g(n_2, n_1)$. Desta forma, ambos os nomes devem ser tomados separadamente como âncoras na avaliação.

Um outro aspecto importante é o empate nos valores de f. Por exemplo, $f(n_1, n_2)$ pode ser igual a $f(n_1, x)$ para diferentes valores de x. Para isso, consideramos duas abordagens na avaliação. Na primeira, chamada de *relativa* apenas valores distintos de f são utilizados para definir a posição na ordenação, i.e., cada conjunto de empates é tratado como um único nome, ocupando uma única posição na ordenação. Por exemplo, $g(n_1, n_2) = 10$ se $f(n_1, n_2)$ é o décimo valor *distinto* de f na ordenação, e então há exatamente nove valores distintos menores que $f(n_1, n_2)$, enquanto pode haver mais do que nove nomes n_3 com $f(n_1, n_3) < f(n_1, n_2)$. Na segunda, chamada de *absoluta*, todos os nomes são considerados individualmente na ordenação e a posição de n_2 é a última dentre os nomes que empatam com n_2 (pior caso). Repare que a abordagem absoluta é mais realista, tendo em vista que não estamos considerando nenhuma informação adicional (que pudesse ser explorada para questões de desempate).

2.1. Dataset: *DBLP*

O dataset utilizado neste trabalho é a base bibliográfica do *DBLP*[2] que há décadas registra artigos científicos publicados principalmente na área da Computação. O DBLP é uma base curada (eletronicamente e manualmente), de acesso livre, e amplamente utilizada,

[1]ORCID Team, detalhes em `https://orcid.org/`

[2]The DBLP Team, detalhes em `https://dblp.org/`

inclusive no estudo de remoção de ambiguidades. Neste trabalho utilizamos o snapshot de dezembro de 2020 da base. Ao todo, 2.400.000 artigos foram analisados, dos quais obtivemos 7.334.613 nomes distintos de autores (que será usado como o universo de nomes), havendo 387.316 registros ORCID distintos, dos quais 376.935 ($\sim$97%) possuem apenas uma grafia de nome na base. Os demais 10.381 ($\sim$3%) registros ORCID (autores) possuem mais de uma grafia distinta e foram utilizados como gabarito para os pares de sinônimos.

3. Medidas de Similaridade

O problema de comparar a similaridade entre duas strings é fundamental e antigo na computação, e não há um consenso sobre o melhor algoritmo para realizar esta comparação. A melhor medida de similaridade (algoritmo) para comparar duas strings depende da tarefa (contexto), por isso diferentes medidas de similaridades surgem e são empregadas na literatura. Estamos interessados em comparar nomes de pessoas, de forma que dois nomes que remetam à mesma pessoa tenha maior similaridade (menor distância). Neste contexto, a string "Celina Miraglia Herrera de Figueiredo" é muito similar à string "Celina M. H. de Figueiredo", pois muito provavelmente as duas strings remetem à mesma pessoa (tendo em vista as convenções de nomes da língua portuguesa e abreviações). Entretanto, sintetizar esta intuição em uma função de similaridade de string não é algo trivial.

Neste trabalho, três diferentes medidas de similaridade de string serão consideradas, cada qual com um aspecto distinto para aferir a similaridade (distância) entre dois strings (a ser discutido). No que segue, as medidas serão apresentadas juntamente com um exemplo. É importante ressaltar que todas as métricas usadas são simétricas, ou seja, $f(n_1, n_2) = f(n_2, n_1)$.

3.1. Distância de *Levenshtein*

A distância de *Levenshtein* é uma distância de edição clássica que permite operações de remoção, inserção, e substituição (cada uma com um custo) ao comparar duas strings. A comparação de strings de tamanhos diferentes será penalizado (distância será aumentada) com operações de inserção ou remoção. Apesar da influência no valor da métrica, neste trabalho os custos de cada operação será considerado unitário.

n_1	n_2	$LEV(n_1, n_2)$	Edição
Maria	Maria	0	-
Maria	Mario	1	*substituição*
Maria	Mariana	2	*adição (2)*
Maria	Mery	3	*remoção (1) e substituição (2)*

Tabela 1. Exemplo da distância de *Levenshtein* com custos unitários para as operações (valores maiores são mais diferentes).

3.2. Maior Subsequência Comum (LCS)

A distância de Maior Subsequência Comum (LCS, do inglês *Longest Common Subsequence*) é dada pelo tamanho da maior subsequência em comum entre duas strings. A LCS

está relacionada a distância de *Levenshtein* mas sem permitir a operação de substituição (apenas remoção e inserção são permitidas). É importante notar que a LCS não necessita que a subsequência seja contínua, como no caso do problema da maior substring comum. Além disso, o valor de LCS é sempre menor ou igual ao tamanho da menor string, sendo zero apenas quando não há caracteres em comum entre as duas strings. Por isso, $LCS(n_1, n_1) = LCS(n_1, n_2)$ se n_1 está contido em n_2 e n_2 é maior do que n_1.

n_1	n_2	$LCS(n_1, n_2)$	Subsequência
Maria	Maria	5	*Maria*
Maria	Mario	4	*Mari*
Maria	Mariana	5	*Maria*
Maria	Mery	2	*Mr*

Tabela 2. Exemplo da distância de LCS (valores menores são mais diferentes).

3.3. *Trend Micro Locality Sensitive Hash* (TLSH)

A ideia central de técnicas de *Locality Sensitive Hash* é gerar valores de hash que sejam próximos (no espaço de hashes) quando os valores das chaves são próximas (no espaço das chaves). No contexto de strings, a distância entre o valor de hash de duas strings distintas deve ser proporcional à distância de edição entre as duas strings. A *Trend Micro Locality Sensitive Hash* (TLSH) tem este propósito e é tal que $TLSH(n_1, n_2)$ retorna a distância entre os valores de hash das strings n_1 e n_2. O funcionamento específico da TLSH está fora do escopo deste trabalho [Oliver et al. 2013].

Assim como a distância de *Levenshtein*, $TLSH(n_1, n_2) = 0$ se as strings n_1 e n_2 são idênticas, e seu valor cresce conforme aumentam as diferenças nas strings n_1 e n_2. É importante ressaltar que a TLSH foi desenvolvida para comparação de uma quantia mínima de dados que possuem uma certa "aleatoriedade" em sua geração. Na implementação utilizada neste trabalho, essa quantia mínima é de 50 bytes, valor superior ao comprimento de todos (100%) os nomes. Dessa forma, os nomes foram completados ao final com uma sequência determinística, baseada na posição do caractere em um vetor de 50 bytes, para terem o tamanho mínimo. Por exemplo, dois nomes de tamanhos quaisquer terão o último caractere de complemento idêntico. Essas limitações impactam drasticamente a performance do algoritmo, visto que este foi projetado para trabalhar com dados maiores e menos uniformes.

n_1	n_2	$TLSH(n_1, n_2)$	Diferença entre os hashes
Maria	Maria	0	-
Maria	Mario	9	*9 caracteres*
Maria	Mariana	24	*24 caracteres*
Maria	Mery	16	*16 caracteres*

Tabela 3. Exemplo da distância de TLSH (valores maiores são mais diferentes).

Os exemplos das distâncias das três medidas de similaridade ilustra como os valores e suas magnitudes dependem da medida de similaridade, não sendo possível uma comparação direta entre as distâncias. Isso reforça a metodologia de ordenação proposta e utilizada neste trabalho, pois permite uma comparação mais justa (baseada na ordenação da própria medida) entre as medidas de similaridade.

4. Processamento dos Dados

Os dados providenciados pelo *DBLP* consistem de um arquivo XML contendo dados de todas as publicações, e um arquivo DTD para especificar os atributos do XML. O arquivo avaliado neste trabalho foi o dblp-2020-12-01.xml que possui tamanho 2,98 GB. As informações desse arquivo foram extraídas pelo programa em Python 3 "DBLP Dataset Parser"[3] e modificado para fins deste artigo. Os dados adquiridos foram extraídos para diversos arquivos JSON que foram subsequentemente processados por um software em C++11 desenvolvido no âmbito deste artigo. Decidiu-se utilizar C++ para o processamento devido à imensa quantia de dados e a necessidade de haver maior controle do fluxo de dados.

O software desenvolvido possui três etapas principais: extração, filtragem e processamento. A extração ocorre ao percorrer o arquivo fornecido pelo DBLP por meio do DBLPParser que gera arquivos em formato intermediário para o software em C++. A filtragem começa a partir de variáveis que são extraídas juntamente com os dados. Filtram-se os artigos a fim de gerar *nós* de autores para logo em seguida capturar seus nomes e ORCID's a fim de montar uma tabela *hash* na qual as chaves são os ORCID's, e os valores são listas de nomes distintos atrelados a tais ORCID's. Finalmente, uma lista separada com todos os nomes distintos de autores da base de dados é gerada.

O processamento das medidas de similaridade se dá pelo uso da tabela *hash* previamente mencionada. Para cada par (n_1, n_2) de nomes sinônimos de um dado ORCID na tabela, são calculadas as distâncias entre n_1 e todos os demais nomes distintos da base dados, e guardadas em uma lista de distância-nome. Ordena-se tal lista pelo valor das distâncias e procura-se a posição de n_2 nessa ordenação. Dois valores de ordenação são calculados: absoluto e relativo. O valor absoluto (R_a) é a quantidade de nomes com distâncias menores ou iguais à distância do par (n_1, n_2); enquanto o relativo (R_r) é quantidade de distâncias distintas menores ou iguais à distância do par (n_1, n_2). O nome de referência é chamado de âncora e o outro é chamado de alvo. Por exemplo, ao considerar o par (n_1, n_2), n_1 é ancora e n_2 é alvo, e vice-versa ao considerar o par (n_2, n_1). O Algoritmo 1 apresenta o pseudo-código do processamento que é realizado para geração das duas ordenações, para cada medida de similaridade. Cada uma das medidas de similaridade será utilizada na linha 7 do pseudo-código, para gerar as distâncias do âncora para todos os nomes. As medidas de similaridade de Levenshtein e LCS foram implementadas em C++ usando algoritmos clássicos baseados em programação dinâmica, enquanto a TLSH foi calculada utilizando a biblioteca disponibilizada publicamente[4].

Retomando aos sinônimos "Celina M. H. de Figueiredo" (n_1) e "Celina Miraglia Herrera de Figueiredo" (n_2). Temos $LEV(n_1, n_2) = 13$ e obtemos $R_r(n_1, n_2) = 8$

[3]Criado por Zhang Hao, disponível em `https://github.com/IsaacChanghau/DBLPParser`

[4]TLSH - Trend Micro Locality Sensitive Hash, disponível em `https://github.com/trendmicro/tlsh`

Algorithm 1 Pseudo-código do algoritmo para processamento das medidas de similaridade e geração das ordenações.

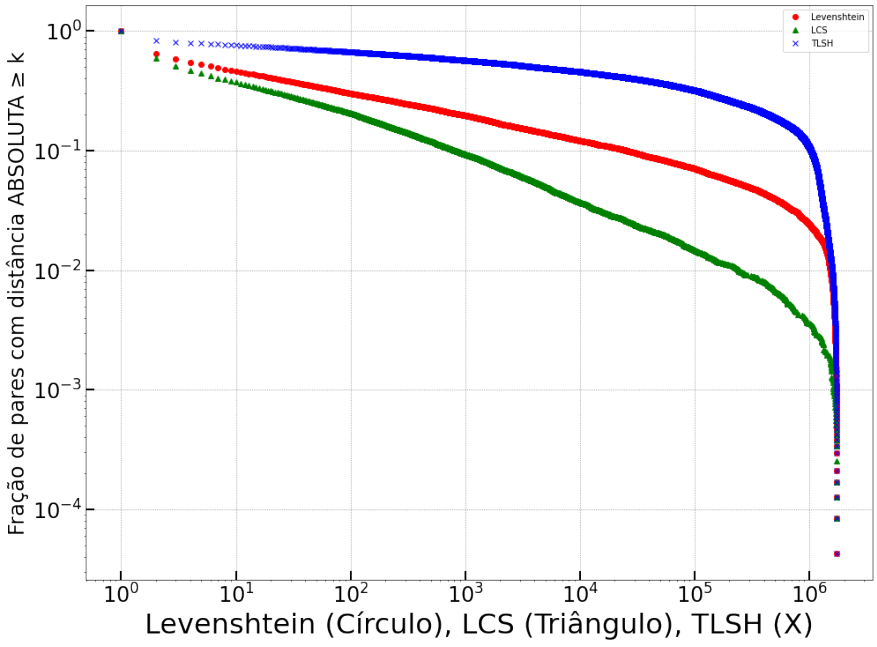
```
    let r_rank //Relative Rank
    let a_rank //Absolute Rank
 3: function PROCESS
        for each pair {ORCID : synonyms} do
            for each name of synonyms do
 6:             anchor = name
                get distance from anchor to every name of name_author_nodes
                save (distances, names) to an array DistName
 9:             sort(DistName) by distance values
                for each alt_name of synonyms do
                    if alt_name ≠ anchor then
12:                     target = alt_name
                        find position of target in DistName
                        r_rank = amount of different distances up until target's distance
15:                     a_rank = amount of names up until target's distance
                        save { anchor, target, r_rank, a_rank } to json
                    end if
18:             end for
            end for
        end for
21: end function
```

e $R_a(n_1, n_2) = 121$. Em outras palavras, há 8 **distâncias** distintas, enquanto há 121 **nomes** distintos, até n_2 aparecer na lista ordenada. E, a exemplo da falta de simetria nas ordenações, obtemos $R_r(n_2, n_1) = 0$ e $R_a(n_2, n_1) = 1$. A ordenação relativa indica ser mais promissora, por oferecer valores menores, mas em contrapartida, a ordenação absoluta oferece mais informação.

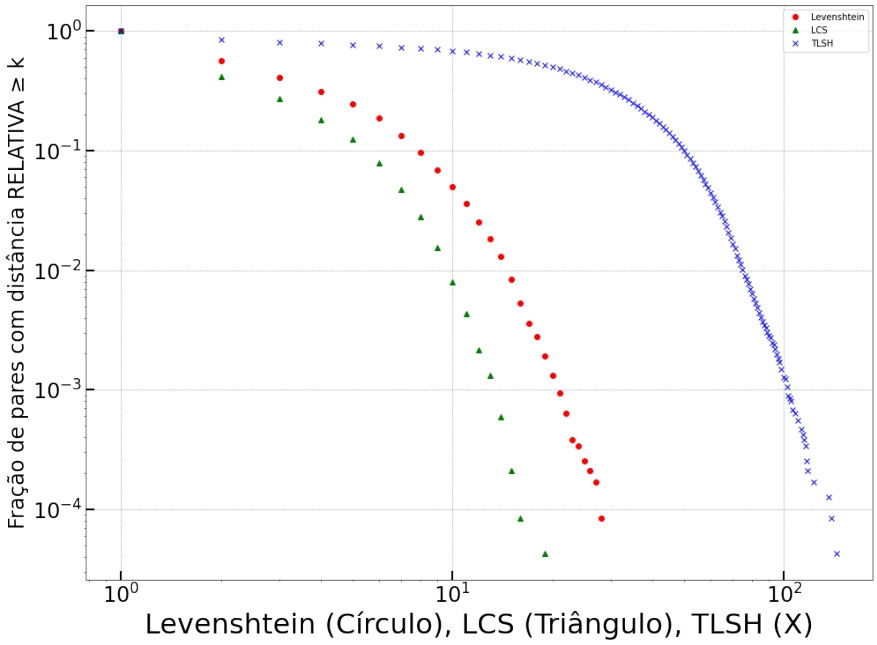
As máquinas utilizadas para o processamento possuem processadores Intel i7 4770 e Intel i5 8600K, em ambientes GNU/Linux Mint e Ubuntu respectivamente. A primeira máquina em questão foi utilizada para fins de desenvolvimento e testes iniciais, enquanto a segunda foi utilizada somente para processamento final. O processamento dos dados depende da medida de similaridade, e levou ∼23 horas para processar com a Levenshtein (83054s), ∼25.5 horas com a LCS (91619s), e ∼74 horas com a TLSH (265083s). Repare que as medidas de Levenshtein e LCS possuem tempos de execução similares (e foram implementadas no contexto deste trabalho), enquanto a TLSH levou aproximadamente três vezes mais tempo.

5. Resultados

Nesta sessão serão discutidos os resultados obtidos pelas três medidas de similaridade por meio dos dois métodos, relativo e absoluto. Como os valores obtidos das distâncias entre os pares de sinônimos induzem um espaço amostral de valores muito grande, os gráficos foram criados na escala logarítmica; com exceção do gráfico Levenshtein x LCS relativo.



(CA) Distância absoluta na ordenação



(CR) Distância relativa na ordenação

Figura 1. Distribuição complementar cumulativa (CCDF) das distâncias na ordenação absoluta e relativa das três medidas de similaridade, em escala $\log - \log$**.**

5.1. Distribuição das distâncias

A Figura 1 apresenta a distribuição complementar cumulativa (CCDF) das distâncias absoluta e relativa para as três medidas de similaridade. O gráfico está em escala $\log - \log$ e mostra a fração de pares de sinônimos que possui uma distância maior do que o valor do eixo-x (cada curva corresponde a uma métrica). As três medidas possuem comportamento

bem diferentes, tanto na ordenação absoluta quanto relativa. Em particular, a LCS decai mais rapidamente e assim possui o melhor desempenho, enquanto a TLSH decai mais lentamente e assim possui o pior desempenho. Entretanto, a distribuição das três medidas possui cauda pesada, atingindo valores muito maiores que a média e indicando que não podem ser usadas satisfatoriamente para encontrar o nome sinônimo. O valor limite do eixo-x, de pouco mais de 7,3 milhões, corresponde ao número de nomes distintos da base. O fato de haver pontos da distribuição das três medidas com este valor indica que para alguns pares de sinônimos, o nome alvo foi o último nome na ordenação (possivelmente empatado com outros). Acredita-se que o desempenho das métricas em identificar alguns nomes alvos é muito ruim para alguns pares de sinônimos específicos, e que os valores de ordenação nestes casos seriam ainda mais elevados caso a base de nomes distintos fosse maior.

Para ilustrar o desempenho das métricas, considere o gráfico de distância absoluta e o valor de 10^3, representando a posição mil na ordenação. Note que por volta de 10% de todos aos pares na medida LCS possui valor maior a 10^3. A Levenshtein e a TLSH são ainda piores com, respectivamente, $\sim 10\%$ dos pares com distâncias iguais ou maiores a 10^4, e $\sim 10\%$ dos pares com distâncias iguais ou maiores a 10^6. Ou seja, na LCS (a melhor medida) em 10% dos casos o nome alvo não estaria entre os 1000 nomes mais próximos do nome âncora. Além disso, na LCS em apenas 60% dos casos o nome alvo está entre os 10 nomes mais próximos do nome âncora (e apenas 20% para a TLSH). Isso indica que as medidas de similaridade avaliadas não podem ser utilizada satisfatoriamente para revelar os sinônimos.

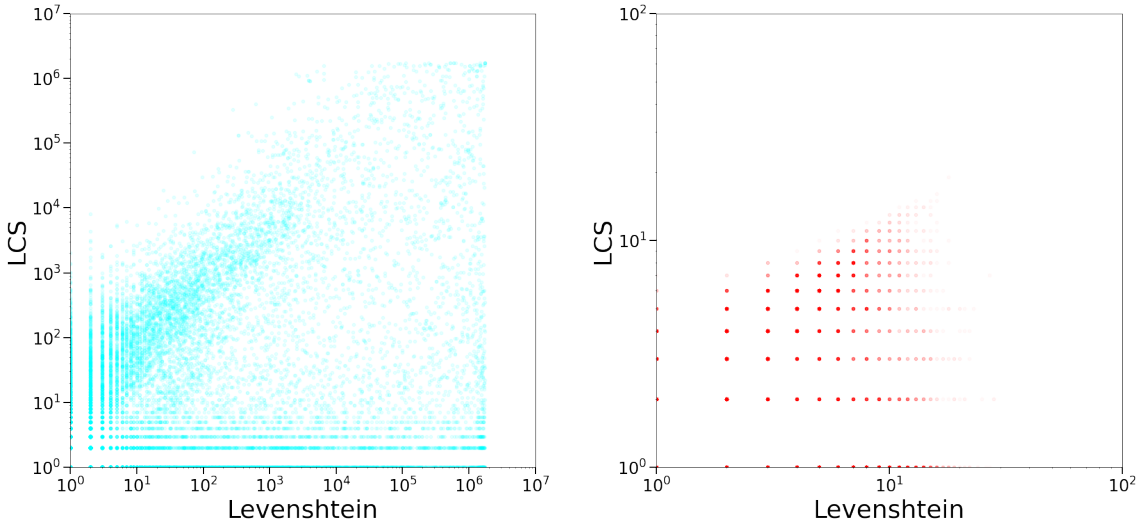
Apesar da mesma relação entre as medidas de similaridade, o gráfico das distâncias relativas mostra que nem todas as medidas atingem o mesmo limite de valor máximo de cauda. Em particular, a distribuição da LCS decai bem mais rapidamente que a TLSH, e não apresenta valores superiores a 20, enquanto a TLSH apresenta valores superiores a 100. Por serem valores relativos, é possível que o número de nomes que empatam com um mesmo valor de distância seja maior na medida LCS, dando origem a valores relativos menores, em comparação com a TLSH que possui maior diversidade de distâncias (e menos nomes empatando com o mesmo valor de distância).

Para ilustrar as diferenças, note que $\sim 1\%$ dos pares de nomes da LCS possuem distâncias iguais ou superiores a 10, com a fração de pares decrescendo rapidamente após essa faixa. Enquanto isso, a Levenshtein possui mais de 5% das distâncias igual ou maior que 10, e a TLSH praticamente 90% das suas distâncias iguais ou maiores que 10. Apesar da LCS demonstrar bom resultado (em $\sim 99\%$ dos casos o nome alvo está entre os 10 nomes mais próximos do âncora) as distâncias relativas podem esconder a real dificuldade de encontrar o sinônimo, tendo em vista os empates para um mesmo valor de distância.

5.2. Comparação direta das medidas

As medidas de similaridade foram comparadas para caracterizar a correlação entre elas, utilizando gráficos do tipo *s*catter-plot. Cada par de sinônimo possui uma distância calculada por cada combinação *(métrica, ordenação)*, onde as métricas são Levenshtein, LCS e TLSH, e as ordenações são Absoluto e Relativo. Todos os pares de sinônimos são considerados em todas as combinações de métrica e ordenação.

As Figuras 2 e 3 mostram os seis gráficos scatter-plots das combinações de métrica



(A1) Levenshtein × LCS Absoluta (R1) Levenshtein × LCS Relativa

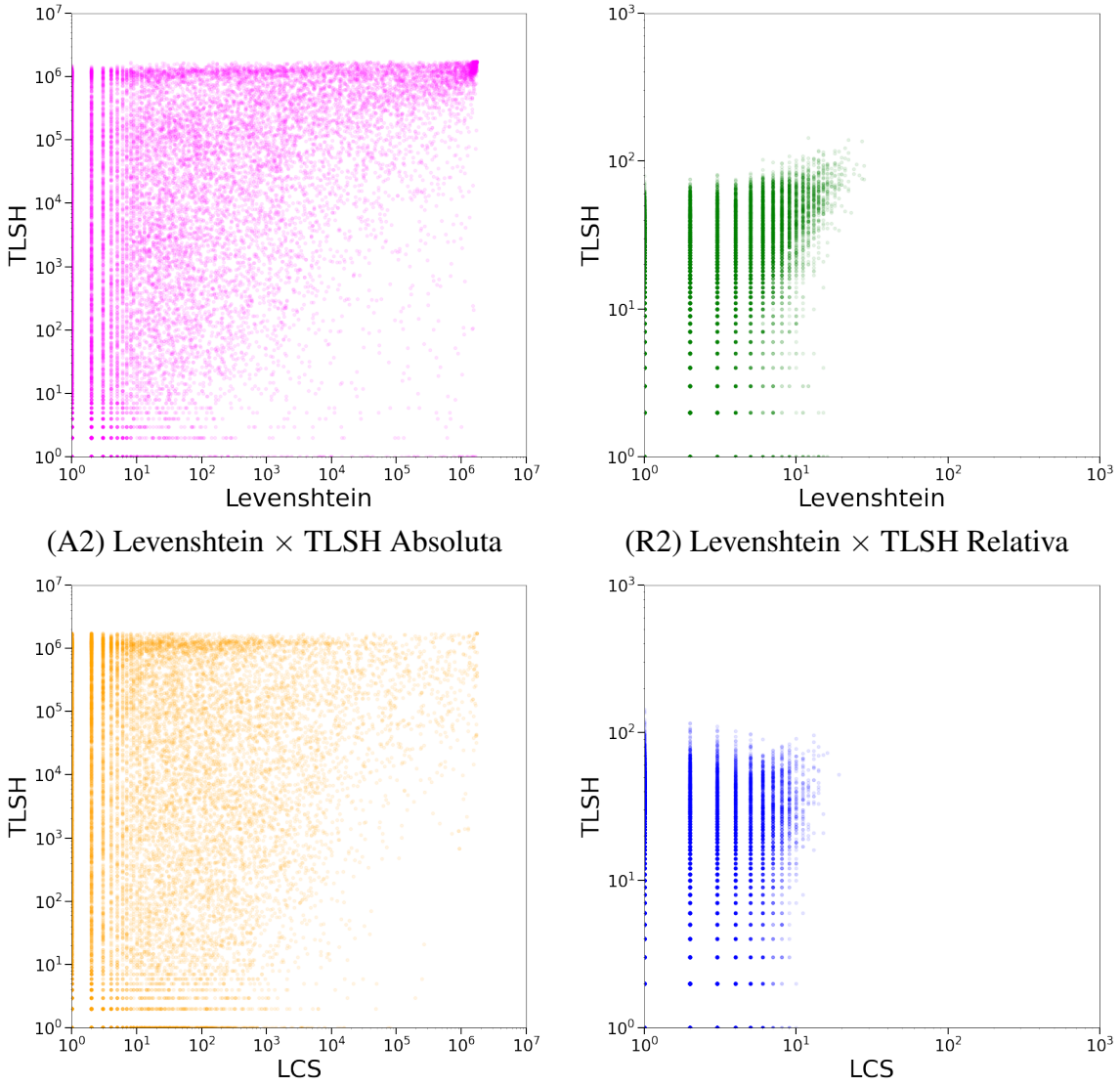
Figura 2. Comparações das distâncias de ordenação para combinação de métricas Levenshtein e LCS (*scatter-plot* em escala $\log - \log$).

e ordenação. Note que os valores dos pares dos gráficos A1, A2, A3, possuem uma distribuição mais esparsa, enquanto os gráficos R2, R3, e especialmente o R1 possuem uma concentração muito maior de valores de distâncias. Pode-se notar como A1 e R1 demonstram uma correlação de valores maior em relação aos demais gráficos, o que indica que tanto no caso Absoluto quanto no Relativo, a Levenshtein e a LCS apresentam um comportamento mais similar.

Ainda assim, note como em A1 a LCS não apresenta valores muito elevados (maiores que 10^4 quando valores de Levenshtein são baixos (menores que 10^2), mas que o contrário não ocorre (valores de LCS podem ser bem elevados mesmo quando valores de Levenshtein são baixos). Apesar deste gráfico mostrar a maior correlação dentre as Absolutas, é possível observar que a correlação não é muito alta, tendo em vista os muitos pontos com altos valores para Levenshtein.

O mesmo não pode ser dito sobre a TLSH em relação às demais métricas. De fato, a TLSH é a métrica que possui os valores mais altos para as distâncias de um par de sinônimos. Curiosamente em A2, pode-se ver um cluster de distâncias extremamente altas (próximas a 10^6) para as duas métricas Levenshtein e a TLSH. Entretanto, tal cluster não ocorre com a LCS, em A3.

O gráfico R1 mostra a correlação entre a Levenshtein e a LCS. No Relativo, a correlação é similar à Absoluta, e, além disso, o espaço de valores das distâncias dos pares é muito mais restrito. Os espaços de valores distintos em R2 e R3 também são mais restritos comparados aos seus respectivos representantes do Absoluto. Por um lado isso é vantajoso, já que os sinônimos estariam dentre os nomes mais próximos, especialmente no R1, contudo podem haver muitos nomes empatados nas posições anteriores da ordenação relativa.



(A2) Levenshtein × TLSH Absoluta (R2) Levenshtein × TLSH Relativa

(A3) LCS × TLSH Absoluta (R3) LCS × TLSH Relativa

Figura 3. Comparações das distâncias de ordenação para as demais combinações de métricas (*scatter-plot* em escala $\log - \log$).

6. Conclusão

Este trabalho abordou o problema de identificar os sinônimos em nomes de autores de artigos científicos em bases bibliográficas (um autor que aparece com dois ou mais nomes). Uma das primitivas mais utiliza por mecanismos sofisticados de que atacam este problema é a comparação dos strings que representam os dois nomes. Mais especificamente, este trabalho abordou três medidas de similaridade de strings (Levenshtein, LCS, e TLSH), quando aplicada a nome de autores utilizando uma base de dados bibliográficos real (DBLP). A metodologia consiste em identificar os autores com sinônimos através do ORCID, e avaliar a posição do nome alvo na ordenação dos nomes da base a partir do nome âncora (alvo e âncora são os sinônimos). É importante notar que esta metodologia não depende dos valores de distância propriamente ditos, e apenas da ordenação dos nomes a partir das distâncias. Dessa forma, diferentes medidas de similaridade podem ser

comparadas de forma mais justa.

A avaliação foi realizada considerando 10.381 pessoas (autores) com mais de um nome em um universo de 7,3 milhões de nomes distintos. Os resultados indicam que as três medidas de similaridade possuem comportamentos distintos na ordenação absoluta e relativa, apesar de todas apresentarem uma distribuição de cauda pesada na ordenação absoluta. A LCS tem desempenho notadamente superior às demais, tanto na ordenação absoluta quanto relativa. Entretanto, a medida ainda não poderia ser utilizada satisfatoriamente para encontrar todos os sinônimos (em apenas 60% dos casos o nome alvo está entre os 10 nomes mais próximos do nome âncora). Os resultados também indicam uma correlação entre as ordenações geradas pela LCS e Levenshtein (absoluta e relativa), enquanto a TLSH possui baixa correlação com as demais.

Como trabalho futuro, novas medidas de similaridade específicas para nomes de autores em bases bibliográficas podem ser desenvolvidas. Em particular, medidas que levem o conhecimento do contexto, como abreviações e reduções (eliminação) de nomes intermediários. Planeja-se explorar demais métricas existentes da literatura, concomitantemente ao desenvolvimento de um novo algoritmo para atacar o problema de variações de nomes.

Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT press, 3rd edition.

Ferreira, A. A., Gonçalves, M. A., and Laender, A. H. (2012). A brief survey of automatic methods for author name disambiguation. *SIGMOD Rec.*, 41(2):15–26.

Gomide, J., Kling, H., and Figueiredo, D. (2021). Consolidating identities in anonymous ego-centred collaboration networks. *Journal of Complex Networks*, 9(1).

Han, H., Giles, L., Zha, H., Li, C., and Tsioutsiouliklis, K. (2004). Two supervised learning approaches for name disambiguation in author citations. In *ACM/IEEE Conference on Digital Libraries*, pages 296–305.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Ley, M. (2002). The dblp computer science bibliography: Evolution, research issues, perspectives. In *International symposium on string processing and information retrieval*, pages 1–10.

Oliver, J., Cheng, C., and Chen, Y. (2013). Tlsh – a locality sensitive hash. In *2013 Fourth Cybercrime and Trustworthy Computing Workshop*, pages 7–13.

Sanyal, D. K., Bhowmick, P. K., and Das, P. P. (2021). A review of author name disambiguation techniques for the pubmed bibliographic database. *Journal of Information Science*, 47(2):227–254.

Yang, K.-H., Peng, H.-T., Jiang, J.-Y., Lee, H.-M., and Ho, J.-M. (2008). Author name disambiguation for citations using topic and web correlation. In *International Conference on Theory and Practice of Digital Libraries*, pages 185–196.

Zhou, G., Zhang, J., Su, J., Shen, D., and Tan, C. (2004). Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics*, 20(7):1178–1190.