Razdalje urejanja 1. del

Hammingova in Levenshteinova razdalja

↓↓↓
Mladen Borovič, Jani Dugonik

→ **V sistemih procesiranja naravnega jezika, ki se ukvarjajo z obdelavo jezika v obliki besedila, se večinoma ukvarjamo z analizo besedil. Besedila praviloma najprej pošljemo skozi cevovod predobdelave besedil, kjer kot rezultat dobimo značilke, ki jih lahko uporabimo v nadaljnjih korakih obdelave.**

Določena opravila na področju procesiranja naravnega jezika zahtevajo primerjave med različnimi besedili. Dober primer tega sta preverjanje črkovanja in implementacija iskalnikov. Pri preverjanju črkovanja s primerjavo dveh besed iščemo napake v vhodnem besedilu, pri implementaciji iskalnikov pa iščemo tista besedila, ki so najbolj podobna vhodnemu iskalnemu vnosu. Pri izvajanju primerjave med besedili se lahko poslužujemo razdalj urejanja *(angl. edit distances)*, znane tudi kot urejevalne razdalje, ki na vhodu ponavadi prejmejo dva niza, na izhodu pa vrnejo število, ki pomeni razliko med njima.

V tem prispevku bomo na primerih podrobneje predstavili delovanje Hammingove razdalje [1], ki spada med preprostejše razdalje urejanja, in Levenshteinove razdalje [3], ki je predstavnik bolj dovršenih razdalj urejanja. Pogledali si bomo tudi nekatere omejitve, ki nastopajo ob uporabi omenjenih razdalj urejanja. Ker obstaja kar nekaj različnih razdalj urejanja, bomo nekatere druge podrobneje predstavili v naslednjem delu prispevka.

Razdalje urejanja so za določene naloge v procesiranju naravnega jezika ključnega pomena, zato se uporabljajo kot prvi korak obdelave besedila po predobdelavi. Da bi bolje razumeli uporabo razdalj urejanja določenih nalog procesiranja naravnega jezika, si najprej preglejmo področje razdalj urejanja in določene definicije operacij, ki se pojavljajo v izračunih takšnih razdalj.

Razdalje urejanja

Vsaka razdalja urejanja je lahko formalno definirana na vhodnih nizih a in b ter na abecedi Σ, ki predstavlja nabor znakov, ki se lahko pojavijo v nizih a in b. Razdalja urejanja $d(a, b)$ je po definiciji najmanjše število operacij, ki jih potrebujemo, da iz enega niza dobimo drugega [4]. Operacije, ki jih pri tem lahko uporabimo so vstavljanje znaka *(ang. insertion)*, brisanje znaka *(ang. deletion)* in zamenjava znaka *(ang. substitution)*. Dodatno poznamo še posebno obliko zamenjave znaka, transpozicijo *(ang. transposition)*, kjer se zamenjata dva poljubna znaka v nizu, vsi ostali pa ostanejo nespremenjeni. Razlika med tema dvema tipoma zamenjave je, da smo s transpozicijo omejeni na zamenjavo znakov, ki so v nizu, pri navadni zamenjavi pa lahko znak zamenjamo s poljubnim znakom, tudi takšnim, ki se ne pojavi v nizu.

Obstaja več tipov razdalj urejanja, ki dovoljujejo različne nabore operacij. Hammingova razdalja dovoljuje samo zamenjavo znakov, medtem ko Levenshteinova razdalja dovoljuje vstavljanje, brisanje in zamenjavo znakov. Njena različica, imenovana Damerau-Levenshteinova razdalja, ob vseh že omenje-

nih operacijah dovoljuje še transpozicijo znakov. Na koncu lahko omenimo še Jarovo in Jaro-Winklerjevo razdaljo, ki dovoljujeta samo transpozicijo znakov.

Predmet tega prispevka sta Hammingova in Levenshteinova razdalja, zato si podrobneje poglejmo njuno delovanje na primerih.

Hammingova razdalja

Hammingova razdalja, poimenovana po ameriškem matematiku Richardu Hammingu, je najpreprostejša razdalja urejanja. Ta razdalja urejanja dovoljuje samo zamenjavo znakov, kar pomeni, da lahko deluje samo nad dvema nizoma iste velikosti. To je tudi njena glavna slabost, saj vemo, da so besede v kateremkoli naravnem jeziku lahko različnih velikosti. Kljub tej slabosti pa je ta razdalja zelo pomembna v teoriji kodiranja, predvsem na področjih stiskanja podatkov, kriptografije, zaznavanja in popravljanja napak ter pri uporabi na kvantnih računalnikih [2]. Na področju procesiranja naravnega jezika se ta razdalja sicer v praksi redko uporablja, vendar predstavlja odlično osnovo za razlago vseh ostalih v praksi bolj uporabljenih razdalj urejanja.

Delovanje Hammingove razdalje za primerjavo dveh enako dolgih nizov znakov oziroma besed je izjemno preprosto. Niza poravnamo po znakih, nato pa primerjamo istoležna znaka med seboj. Če sta znaka enaka, je utež enaka 0, sicer pa 1. Hammingovo razdaljo lahko tako definiramo tudi z uporabo logične funkcije ekskluzivni ali oziroma XOR. Zapisa $\mathrm{ham}(a,b)$ in $a \oplus b$ sta torej ekvivalentna, vendar se prvi pogosteje uporablja na področju procesiranja naravnega jezika, drugi pa na področju teorije kodiranja. Na koncu seštejemo vse uteži in rezultat je vrednost, ki predstavlja razdaljo med nizoma a in b. V spodnjem zgledu si poglejmo delovanje Hammingove razdalje nad različnimi vhodnimi nizi.

Zgled

Imejmo vhodna niza a = rezultat in b = konzulat. Kot vidimo, sta niza a in b enake dolžine ($|a| = |b| = 8$), saj sicer ne moremo izračunati Hammingove razdalje. Niza smo po znakih istoležno zapisali v tabelo. Ujemajoči znaki so označeni z modro barvo, uteži neujemanja (1) so označene z rdečo barvo, uteži ujemanja (0) pa z zeleno barvo (slika 1).

a	r	e	z	u	l	t	a	t	
b	k	o	n	z	u	l	a	t	
$\mathrm{ham}(a,b)$	1	1	1	1	1	1	0	0	$= 6$

SLIKA 1.
Primer izračuna Hammingove razdalje za vhodna niza a = rezultat in b = konzulat

Izračun razdalje začnemo s primerjavo prvega znaka v obeh nizih. V našem primeru sta to znaka »r« in »k«, ki nista enaka, zato je utež enaka 1. Nato nadaljujemo z drugim znakom v obeh nizih. To sta znaka »e« in »o«, ki znova nista enaka in zato je utež znova enaka 1. Podobno nadaljujemo vse do sedmega znaka v obeh nizih, kjer ugotovimo ujemanje v znaku »a«. Utež je tukaj enaka 0. Enako se zgodi pri osmem znaku v obeh nizih, kjer se niza ujemata v znaku »t«. Po prehodu smo ugotovili, da imamo med nizoma a in b šest neujemanj in dve ujemanji. Sledi še samo zadnji korak izračuna razdalje, ki je seštevek vseh uteži. Razdalja med nizoma $\mathrm{ham}(a,b)$ je torej enaka 6, kar pomeni, da sta niza različna.

a	r	e	z	u	l	t	a	t	
b	r	e	z	u	l	t	a	t	
$\mathrm{ham}(a,b)$	0	0	0	0	0	0	0	0	$= 0$

SLIKA 2.
Primer izračuna Hammingove razdalje za vhodna niza a = rezultat in b = rezultat

Če primerjamo dva enaka niza, lahko hitro ugotovimo, da je Hammingova razdalja enaka 0. Takšno situacijo lahko vidimo na sliki 2, kjer smo uporabili vhodna niza a = rezultat in b = rezultat. Na sliki 3, kjer smo uporabili vhodna niza a = rezultat in b = galerija, lahko vidimo, da je Hammingova razdalja enaka dolžini niza a ali b. To pomeni, da sta niza a in b povsem različna.

S Hammingovo razdaljo lahko ugotavljamo podobnosti med nizi iste velikosti, kar pa zelo omeji smiselnost njene uporabe pri naravnem jeziku, kjer so besede različnih velikosti. Hammingova razdalja v tem primeru ni primerna, saj ne dovoljuje vstavljanja ali brisanja znakov. Zaradi tega potrebu-

a	r	e	z	u	l	t	a	t	
b	g	a	l	e	r	i	j	a	
ham(a, b)	1	1	1	1	1	1	1	1	$= 8$

SLIKA 3.
Primer izračuna Hammingove razdalje za vhodna niza a = rezultat in b = galerija

jemo razdaljo urejanja, ki bo pri izračunu razdalje dovoljevala tudi ti dve operaciji in nam s tem omogočila primerjavo nizov različnih velikosti. Ena najbolj uporabljenih razdalj urejanja s takšno lastnostjo je Levenshteinova razdalja, ki jo bomo opisali v nadaljevanju.

Levenshteinova razdalja

Začetki ideje o razdalji med nizi različnih velikosti, ki upošteva operacije vstavljanja, brisanja in zamenjave znakov, segajo v leto 1965, ko je sovjetski matematik Vladimir Levenshtein predstavil novo razdaljo, imenovano Levenshteinova razdalja. Ta je zaradi upoštevanja večjega nabora operacij hitro postala tista razdalja, ki se je začela uporabljati tudi v praksi. Na področju procesiranja naravnega jezika se ta razdalja pogosto uporablja v implementacijah popravljanja pravopisa, saj lahko z njo določimo napake v črkovanju in jih glede na vsebino besedila tudi ustrezno zamenjamo s pravilno črkovanimi besedami.

Levenshteinova razdalja ima nekaj zanimivih matematičnih lastnosti:

- Spodnjo mejo vrednosti razdalje predstavlja absolutna razlika velikosti dveh primerjanih nizov.
- Zgornjo mejo vrednosti razdalje predstavlja velikost daljšega izmed obeh primerjalnih nizov.
- Vrednost razdalje bo enaka 0 takrat, ko sta oba niza enaka.
- Če sta niza enakih velikosti, bo vrednost Hammingove razdalje tudi zgornja meja vrednosti Levenshteinove razdalje.
- Za Levenshteinovo razdaljo velja tudi trikotniška neenakost, in sicer Levenshteinova razdalja med dvema nizoma ne bo nikoli večja od vsote njunih vrednosti Levenshteinovih razdalj v primerjavi s tretjim nizom.

Primeri za te matematične lastnosti so podani v kodi na javno dostopnem repozitoriju GitHub [5].

Izračun Levenshteinove razdalje lahko implementiramo rekurzivno ali iterativno. Rekurzivni način je zelo naiven in neučinkovit, saj v tem primeru večkrat računamo vrednost Levenshteinove razdalje pri istih podnizih. Zaradi tega se bomo v tem prispevku osredotočili na iterativni način izračuna Levenshteinove razdalje, ki je bistveno bolj učinkovit in se prav tako uporablja v praktičnih implementacijah. V iterativnem načinu izračuna uporabimo pristop dinamičnega programiranja, kjer uteži za vsako operacijo nad nizi shranimo v pomožno matriko. V tem prispevku bomo privzeli, da je uporaba vseh operacij enakovredna, torej bo uporaba vseh operacij ovrednotena z utežjo 1. Vrednost te uteži lahko po želji tudi spremenimo, če želimo dodatno obtežiti uporabo katere od operacij. Iterativni način implementacije, ki si ga bomo pogledali na zgledu, je sicer zelo podoben postopku, ki se uporablja v algoritmih za poravnavo sekvenc DNK, kot so algoritmi Needleman-Wunsch, Wagner-Fischer ali Smith-Waterman.

Zgled

Imejmo dva vhodna niza a = telefon in b = lepota. Vidimo, da sta niza različnih dolžin ($|a| = 7$, $|b| = 6$). V primeru bomo uporabili iterativni način izračuna Levenshteinove razdalje. Z algoritmom za izračun Levenshteinove razdalje bomo želeli ugotoviti skupno število operacij, ki jih potrebujemo, da niz a spremenimo v niz b. Najprej pripravimo pomožno matriko, kamor bomo shranjevali vmesne vrednosti Levenshteinove razdalje. Niz a po črkah razporedimo v prvi stolpec, niz b pa po črkah razporedimo v prvo vrstico. Pri tem vsako črko opremimo s številom, ki predstavlja indeks in se začne z 1. Po tem koraku dobimo matriko prikazano na sliki 4 levo.

Izračun Levenshteinove razdalje nadaljujemo s prehodom skozi vsak element pomožne matrike in izračunom vrednosti vseh praznih elementov. To storimo z operatorjem, ki obsega 4 elemente in ga predstavimo kot matriko velikosti 2×2. Spodnji desni element operatorja poravnamo s praznim elementom pomožne matrike, ki ga želimo izračunati. Vidimo, da operator prekriva vrednosti pomožne matrike, ki jih bomo uporabili za izračun praznega

		l	e	p	o	t	a
	0	1	2	3	4	5	6
t	1						
e	2						
l	3						
e	4						
f	5						
o	6						
n	7						

		l	e	p	o	t	a
	0	1	2	3	4	5	6
t	1	1					
e	2						
l	3						
e	4						
f	5						
o	6						
n	7						

SLIKA 4.
Levo: Inicializacija pomožne matrike za niza a = telefon in b = lepota. Desno: Prekrivanje operatorja in prvi izračun praznega elementa v pomožni matriki.

elementa pomožne matrike. To prikazuje slika 4 desno, kjer smo operator poravnali tako, da primerjamo prvi črki obeh nizov. To sta črki »t« in »l«, ki sta označeni s svetlo sivo barvo. Z modro barvo so označene vrednosti v pomožni matriki, ki jih prekriva operator. Z zeleno barvo je označen prazen element, katerega vrednost želimo izračunati. Sledi izračun vrednosti praznega elementa pomožne matrike. Če se črki, ki ju primerjamo, razlikujeta, potem poiščemo minimalno vrednost elementov, ki jih prekriva operator, in prištejemo utež. Utež pove za kolikšno vrednost kaznujemo razlikovanje zaradi dodajanja, brisanja ali zamenjave znaka v primeru neujemanja. Ker smo se odločili, da bomo vse operacije tretirali kot enakovredne, bo utež v našem primeru vedno enaka 1. Na sliki 4 desno vidimo, da smo za prazen element pomožne matrike (označen z zeleno barvo) na prej opisan način dobili vrednost 1. Izbirali smo minimalno vrednost med tremi prekritimi vrednostmi pomožne matrike (označenimi z modro barvo) in sedaj vidimo, da je minimalna vrednost enaka 0. Tej vrednosti prištejemo utež, ki je enaka 1, in s tem dobimo končno vrednost elementa, ki je enaka 1. Slika 5 levo prikazuje nadaljevanje izračuna, kjer primerjamo črki »t« in »e«, na sliki 5 desno pa vidimo izračunano celotno prvo vrstico pomožne matrike, kjer smo med izračuni naleteli na ujemanje v črki »t«.

Če se črki, ki ju primerjamo, ujemata, potem je vrednost praznega elementa pomožne matrike enaka vrednosti elementa pomožne matrike, ki se nahaja levo diagonalno od praznega elementa pomožne matrike. To pomeni, da prevzamemo dosedanjo vrednost razdalje, saj smo naleteli na ujemanje. Na sliki 5 desno vidimo, da smo za prazen element pomožne matrike (označen z zeleno barvo) prevzeli vrednost 4 iz elementa, ki je levo diagonalno od njega (označen z oranžno barvo).

		l	e	p	o	t	a
	0	1	2	3	4	5	6
t	1	1	2				
e	2						
l	3						
e	4						
f	5						
o	6						
n	7						

		l	e	p	o	t	a
	0	1	2	3	4	5	6
t	1	1	2	3	4	4	5
e	2						
l	3						
e	4						
f	5						
o	6						
n	7						

SLIKA 5.
Nadaljevanje izračunov vrednosti praznih elementov v prvi vrstici pomožne matrike. Levo: Izračun vrednosti pri primerjavi črk »t« in »e«. Desno: Izračun vrednosti pri ujemanju v črki »t«.

Postopek nadaljujemo tako dolgo, dokler ne izračunamo vseh vrednosti praznih elementov v pomožni matriki. Končno stanje vrednosti pomožne matrike prikazuje slika 6 levo. Ko izračunamo vse vrednosti pomožne matrike, odčitamo spodnji desni element pomožne matrike (označen z rdečo barvo). Ta vrednost predstavlja Levenshteinovo razdaljo med nizoma a in b, ki pove, da je za pretvorbo niza a v b potrebnih pet operacij. Za nekatere naloge v procesiranju naravnega jezika je ta vrednost že dovolj, saj lahko definiramo metriko podobnosti. Naj bo $\mathrm{lev}(a,b)$ Levenshteinova razdalja med nizoma a in b. Levenshteinovo podobnost med nizoma a in b označimo kot $\mathrm{sim}_{\mathrm{lev}}(a,b)$ in definiramo z enačbo

$$\mathrm{sim}_{\mathrm{lev}}(a,b) = \frac{|a| + |b| - \mathrm{lev}(a,b)}{|a| + |b|}, \qquad (1)$$

kjer sta $|a|$ in $|b|$ dolžini nizov a in b. Vrednost $\mathrm{sim}_{\mathrm{lev}}(a,b)$ je na intervalu $[0,1]$ in predstavlja po-

		l	e	p	o	t	a
	0	1	2	3	4	5	6
t	1	1	2	3	4	4	5
e	2	2	1	2	3	4	5
l	3	2	2	2	3	4	5
e	4	3	2	3	3	4	5
f	5	4	3	3	4	4	5
o	6	5	4	4	3	4	5
n	7	6	5	5	4	4	**5**

		l	e	p	o	t	a
	0	1	2	3	4	5	6
t	**1**	1	2	3	4	4	5
e	**2**	2	1	2	3	4	5
l	3	**2**	2	2	3	4	5
e	4	3	**2**	3	3	4	5
f	5	4	3	**3**	4	4	5
o	6	5	4	4	**3**	**4**	5
n	7	6	5	5	4	4	**5**

SLIKA 6.
Izračun vrednosti Levenshteinove razdalje s polno pomožno matriko in ponazoritev poti z minimalnim številom operacij nad nizi. Levo: Končno stanje vrednosti v pomožni matriki. Desno: Ena izmed možnih poti, ki ponazarja minimalno število operacij.

dobnost med nizoma a in b. Za naš primer izračunamo vrednost 0,615, kar interpretiramo kot 61,5 % podobnost med nizoma a in b.

zamenjava	brisanje
vstavljanje	*

SLIKA 7.
Operator za ugotavljanje zaporedja uporabljenih operacij. Znak * označuje pozicijo trenutnega elementa.

V primeru, da bi želeli vedeti, katere operacije so bile potrebne za pretvorbo niza a = telefon v niz b = lepota, moramo rekonstruirati zaporedje izvedenih operacij. To lahko storimo tako, da začnemo v skrajnem spodnjem desnem elementu pomožne matrike in znova upoštevamo matrični operator velikosti 2×2 (slika 7). Cilj je priti v zgornji levi element matrike z vrednostjo 0. Tvorili bomo torej pot po pomožni matriki, odvisno od premika pa bomo izvedeli, katera operacija je bila izvedena v tistem koraku. Pri tem bomo niz a spreminjali v smeri iz desne proti levi.

Ob ujemanju znakov se premaknemo levo diagonalno in ne izvedemo nobene operacije, saj gre za ujemanje. Ob neujemanju znakov bo premik v levo pomenil operacijo vstavljanja, premik navzgor bo pomenil operacijo brisanja, premik po levi diagonali pa bo pomenil operacijo zamenjave. Premiki si vedno sledijo v smeri elementa z nižjo vrednostjo od vrednosti elementa, v katerem se nahajamo. Pri tem upoštevamo znake iz nizov a in b, ki predstavljajo glavo stolpcev in vrstic. Grafični potek ene takšne poti prikazuje slika 6 desno, izvedbo zaporedja operacij pa prikazuje slika 8.

Zaključek

V tem prispevku smo predstavili osnovne lastnosti razdalj urejanja, ki jih lahko uporabimo na področju procesiranja naravnega jezika. Podrobneje smo si pogledali delovanje Hammingove in Levenshteinove razdalje urejanja. S Hammingovo razdaljo smo spoznali delovanje operacije zamenjave, z Levenshteinovo razdaljo pa tudi operaciji vstavljanja in brisanja. Primera implementacije Hammingove in Levenshteinove razdalje v programskem jeziku Python sta na voljo na javno dostopnem repozitoriju GitHub [5].

#	Operacija	Rezultat
1	zamenjava črke »n« z »a«	telefo**n** → telefo**a**
2	vstavljanje črke »t«	telefoa → telefo**t**a
3	ni operacije (ujemanje v črki »o«)	telef**o**ta
4	zamenjava črke »f« s »p«	telef**o**ta → telep**o**ta
5	ni operacije (ujemanje v črki »e«)	tel**e**pota
6	ni operacije (ujemanje v črki »l«)	t**l**epota
7	brisanje črke »e«	t**e**lepota → tlepota
8	brisanje črke »t«	**t**lepota → lepota

SLIKA 8.
Izvedba zaporedja operacij ob spremembi niza a = telefon v niz b = lepota

Čeprav operacije transpozicije v tem prispevku nismo podrobneje obravnavali, je vredno omeniti, da obstaja razširitev Levenshteinove razdalje, ki dovoljuje tudi to operacijo. To je Damerau-Levenshteinova razdalja, ki se na področju procesiranja naravnega jezika pogosto uporablja za popravljanje črkovanja. Ker pa to ni edina zanimiva uporaba operacije transpozicije, si bomo v drugem delu podrobneje pogledali dve razdalji urejanja, ki dovoljujeta izključno to operacijo – to sta Jarova in Jaro-Winklerjeva razdalja.

Literatura

[1] R. W. Hamming, *Error detecting and error correcting codes*, The Bell System Technical Journal **29.2** (1950), 147–160. doi: 10.1002/j.1538-7305.1950.tb00463.x.

[2] M. Khan in A. Miranskyy, *String Comparison on a Quantum Computer Using Hamming Distance*, arXiv: 2106.16173 [cs.ET], 2021.

[3] V. Iosifovich Levenshtein, *Binary codes capable of correcting deletions, insertions and reversals*, Soviet Physics Doklady **10.8** (1966), Doklady Akademii Nauk SSSR, V163 No.4 845-848 1965, 707–710.

[4] G. Navarro, *A Guided Tour to Approximate String Matching*, ACM Comput. Surv. **33.1** (2001), 31–88. ISSN: 0360-0300, doi: 10.1145/375360.375365, dostopno na doi.org/10.1145/375360.375365, ogled 10. marca 2022.

[5] *Procesiranje naravnega jezika – GitHub*, dostopno na github.com/procesiranje-naravnega-jezika/example-code/blob/main/1b%5C%20-%5C%20Razdalje%5C%20urejanja%5C%201.%5C%20del%5C%20-%5C%20Hammingova%5C%20in%5C%20Levenshteinova%5C-%20razdalja, ogled 10. marca 2022.

REŠITEV NAGRADNE KRIŽANKE

PRESEK 49/4

→ Pravilna rešitev nagradne križanke iz četrte številke Preseka letnika 49 je **Predobdelava besedila**. Med pravilnimi rešitvami smo izžrebali naslednje reševalce: Ana Pestotnik Stres iz Ljubljane, Marko Belingar iz Solkana in Neda Tompa iz Odrancev, ki bodo nagrade prejeli po pošti.