Delovanje algoritma Jaro-Winkler glede na mesto pojavljanja tipografskih napak

Uroš Godnov
Fakulteta za management, Cankarjeva 5, 6000 Koper
uros.godnov@fm-kp.si

Izvleček

Z decentralizacijo podatkovnih zbirk se je pojavila potreba po integraciji podatkov. Ob odsotnosti lastnosti, ki bi enolično določala zapis podatkov, in ob prisotnosti različnih tipografskih napak se pojavi problem učinkovite integracije podatkov. V članku predstavljamo simulacijo algoritma Jaro-Winkler, ki omogoča samodejno spajanje podatkov s tipografskimi napakami. Želeli smo preveriti, kako se algoritem Jaro-Winkler obnese glede na mesto pojavljanja tipografske napake. Poleg tega smo simulirali tudi delovanje hibridne različice algoritma Jaro-Winkler in rezultate primerjali med seboj. Simulacija je pokazala, da je hibridni algoritem Jaro-Winkler neobčutljiv na mesto pojavljanja tipografskih napak, osnovni pa deluje bolje, če se tipografske napake ne pojavljajo na začetku podatkov, ki jih integriramo.

Ključne besede: algoritem Jaro-Winkler, tipografske napake, hibridni algoritem Jaro-Winkler, kakovost podatkov, integracija podatkov.

Abstract

Simulation of the Jaro-Winkler Algorithm Depending on the Position of Typographical Errors

Database decentralization created the need for data integration. Where no unique data attributes are available and various typographical errors are present, efficient data integration can present a problem. This article describes a simulation of the Jaro-Winkler algorithm, which facilitates automatic matching of data with typographic errors. Our goal was to test the Jaro-Winkler algorithm with regard to the position of the typographical error. We have additionally tested the hybrid version of the Jaro-Winkler algorithm and compared the results of both tests. Our simulation has shown that the hybrid Jaro-Winkler algorithm is not sensitive to the position of the error, however the basic Jaro-Winkler algorithm performs better if typographical errors do not occur at the beginning of the data we are integrating.

Keywords: Jaro-Winkler, typographical mistakes, hybrid Jaro-Winkler, data quality, data integration.

1 UVOD

Pomembnost področja kakovosti podatkov že desetletje nakazujejo različne raziskave (Russom, 2006). Med novejšimi izstopa raziskava iz leta 2009 z naslovom *The State of Data Quality Today* (Waddington, 2009) ter Gartnerjeve ugotovitve iz leta 2010. Raziskava iz leta 2009 podaja te ključne ugotovitve:

- **tretjina anketirancev ocenjuje kakovost podatkov kot slabo in le štirje odstotki kot odlično;**
- **63 odstotkov anketirancev ni znalo oceniti stroškov nekakovostnih podatkov;**
- **ključni prepreki uvedbi upravljanja s kakovostjo podatkov sta »menedžment področja kakovosti podatkov ne vidi kot pomembno področje« ter »težko je predstaviti področje kakovosti podatkov kot področje, ki potrebuje pozornost oz. ukrepanje«;**
- **ključni trije problemi podatkov v smislu kakovosti so, ker »podatki niso standardizirani«, »podatki manjkajo in potrebujejo dopolnitev« ter »podatki so netočni in potrebujejo popravek«.**

Standardizacija, dopolnitve in popravki zahtevajo ogromno časa in običajno pomenijo največji delež aktivnosti pri izgradnji analitičnega sistema, še posebno ko moramo integrirati podatke iz več virov. Fayyad idr. (1996) navajajo, da je največ dela (okoli 90 %) namenjenega odstranjevanju podvojenih zapisov.

Integracija podatkov ter identifikacija podvojenih zapisov sta izhodišče našega raziskovanja. Z nastankom relacijskih podatkovnih zbirk so se pojavili tudi primarni ključi, ki enolično določajo posamezno vr-

stico. Vendar moramo pogosto združevati podatke iz različnih virov, v katerih ne obstaja primarni ključ oz. ni niti atributa, ki bi bil lahko kandidat za primarni ključ. V tem primeru se moramo zanesti na kombinacijo atributov, ki najbolje določajo posamezni zapis. Tako bi lahko npr. posamezno stranko določili s kombinacijo imena, priimka in naslova. Na težave naletimo, ko so atributi zamenjani (»John Smith« vs. »Smith John«), vsebujejo tipografske napake (»John Smith« vs. »John Smiht«), so združeni (»John Smith« vs. »Mr. John Smith Jr.«) ali pa kombinacija vsega naštetega (»John Smith« vs. »Mr. Smiht John Jr.«).

Povezovanje zapisov z omenjenimi značilnostmi je bilo v preteklosti zamudno ročno delo, ki pa se je s pojavom sodobnih orodij za povezovanje in odstranjevanje podvojenih podatkov v veliki meri avtomatiziralo. Za povezovanje podatkov ter odstranjevanje podvojenih podatkov uporabljajo orodja različne algoritme (npr. Levensthein distance, Jaccard index, algoritem Jaro-Winkler, Simil index, q-grams, hibridne modele). V članku se bomo osredinili na algoritma edit distance in Jaro-Winkler, skupaj z njunima hibridnima izpeljankama. Algoritmu Jaro-Winkler Cohen idr. (2003) ter Bilenko idr. (2003) pripisujejo pomen predvsem pri povezovanju in odstranjevanju podvojenih podatkov krajših zapisov, tj. imen in priimkov, vendar ne postrežejo z empiričnimi podatki.

Članek sestoji iz treh delov. V prvem delu se bomo posvetili teoretičnemu ozadju povezovanja podatkov ter odstranjevanja podvojenih podatkov. Sledi simulacija delovanja algoritma edit Jaro-Winkler skupaj s hribridno različico na različno okvarjenih zapisih. Tretji del bomo sklenili s ključnimi ugotovitvami in z razpravo.

2 TEORETIČNO OZADJE

Pred letom 1990 so v ameriškem statističnem uradu pri povezovanju in odstranjevanju podvojenih podatkov v aplikaciji Decennial Census potrebovali 3000 uradnikov v obdobju treh mesecev. Uporaba računalniško podprtih algoritmov je zmanjšala potrebo po uradnikih na 200 v obdobju šestih tednov. Podobno je bilo na področju statistike kmetijstva. Pred letom 1992 so za delo pri integraciji podatkov potrebovali 75 uradnikov v obdobju treh mesecev, kar znese ob upoštevanju 168-urnega mesečnega dela 37800 ur. Potrebni čas so z uvedbo računalniško podprtih algoritmov zmanjšali na 6500 ur (Winkler 1995).

Integracija podatkov ima v literaturi različna poimenovanja. McCallum in Wellner (2003) jo poimenujeta čiščenje podatkov, Tejada idr. (2001) identifikacijo objektov, Winkler (2006) kot povezovanje zapisov ter Gravano idr. (2001) s približnim usklajevanjem zapisov. Integracija podatkov je predmet raziskovanj različnih področij, npr. statistike, podatkovnih zbirk, digitalnih knjižnic ter podatkovnega rudarjenja (Bilenko idr. 2003).

Potreba in s tem težave pri integraciji podatkov so se pojavile šele takrat, ko so se v organizacijah pojavile decentralizirane uporabniške rešitve. V obdobju osrednjih računalnikov teh težav ni bilo, saj je obstajala osrednja podatkovna zbirka pod budnim očesom računalniških inženirjev. Decentralizirane uporabniške rešitve so prinesle večjo svobodo uporabnikov ter boljšo prilagodljivost. Hkrati pa so se pojavile tudi različne inačice posameznih podatkov, kar je s pojavom potrebe po analitičnih rešitvah prineslo nujnost odločitve, katera inačica zapisa je prava. Poleg omenjene odločitve sta se pojavili še dve potrebi, in sicer potreba po odstranjevanju podvojenih podatkov ter potreba po dopolnitvi in/ali popravkih osrednjih zapisov iz drugih informacijskih virov oz. vice versa.

Razlike v zapisih iz različnih informacijskih virov po navedbi Bilenka idr. (2003) ter Jina idr. (2003) nastanejo zaradi različnih načinov shranjevanja podatkov, tipografskih napak, skeniranja dokumentov ter krajšav. Manjše tipografske napake so posebno pogoste pri skeniranju ročno pisanih dokumentov (Winkler 2006). Winkler (2006) je pokazal, da ima tudi integracija podatkov iz zelo kakovostnih informacijskih virov lahko več kot 20 odstotkov napak pri povezovanju imen ter več kot 10 odstotkov napak pri povezovanju priimkov.

Model povezovanja podatkov je prvi predstavil Newcombe s soavtorji (1959), Fellegi in Sunter (1969) pa sta ga leta 1969 matematično opisala. Model temelji na verjetnostih, testiranju hipotez ter relativnih frekvencah. Največ napredka pri razvoju algoritmov je bilo na področju algoritmov za povezovanje znakovnih zapisov s tipografskimi napakami.

Za samodejno povezovanje podatkov obstaja več algoritmov, ki bi jih lahko ločili med seboj glede na prisotnost/odsotnost tipografskih napak ter prisotnost/odsotnost vedenja, kako se pojavljajo napake. V članku se osredinjamo na algoritme iz skupine, v kateri se pojavljajo tipografske napake, hkrati pa nima-

mo vedenja o tem, ali obstaja kakšen vzorec, kako se pojavljajo te napake. Najbolj znana algoritma iz omenjene skupine sta algoritma Jaro-Winkler[1] ter edit distance (Winkler 2006). Algoritem edit distance naj bi bil po mnenju Cohena idr. (2003) trenutno najbolj učinkovit algoritem za povezovanje podatkov iz omenjene skupine, čemur pa nasprotujeta npr. Sarka in Mauri (2011), ki favorizirata algoritem Jaro-Winkler.

Pojavljajo se tudi algoritmi za povezovanje zapisov s tipografskimi napakami, ki temeljijo na drugačnih pristopih, npr. na strojnem učenju (Stenetorp idr. 2011).

3 NAMEN ČLANKA

V članku smo se osredinili na algoritem Jaro-Winkler, katerega delovanje sta simulirala Sarka in Mauri (2011). Vendar je njuna simulacija temeljila samo na enem pokvarjenem nizu podatkov, hibridne različice algoritma Jaro-Winkler pa se sploh nista lotila. Naša simulacija je veliko bolj obsežna in zato tudi natančna, saj je bila analiza narejena glede na mesto pojavljanja tipografskih napak, hkrati pa smo analizo naredili tudi na 150 oz. pri hibridni različici celo na 200 nizih pokvarjenih podatkov.

V literaturi nismo našli tako podrobne simulacije algoritma Jaro-Winkler, še posebno ne njene hibridne različice, sploh pa ne glede na mesto pojavljanja tipografskih napak, o čemer nismo našli nobene simulacije. Poleg tega smo želeli preveriti, ali hibridna različica vpliva na občutljivost algoritma Jaro-Winkler na pojavljanje napak na začetku zapisov in kako.

Namen članka je torej trojen:

- podrobna analiza delovanja algoritma Jaro-Winkler glede na mesto pojavljanja tipografskih napak;
- podrobna analiza delovanja hibridne različice algoritma Jaro-Winkler glede na mesto pojavljanja tipografskih napak;
- preverjanje občutljivosti hibridne različice algoritma Jaro-Winkler glede na mesto pojavljanja tipografskih napak.

4 METODOLOGIJA

Za simulacijo učinkovitosti delovanja algoritma Jaro-Winkler smo uporabili podatkovno zbirko AdventureWorks, ki je podatkovna zbirka strežnika MS SQL 2005 (ter novejših različic), namenjena učenju. Uporabili smo lastnosti Ime ter Priimek entitete Oseba (dbo.DimCustomer). Uporabljeni lastnosti smo »pokvarili« nadzorovano, in sicer smo območje napak razdelili na tri območja, tj. začetno območje (prva tretjina), druga tretjina (osrednji del) ter končno območje (zadnja tretjina).

Podatke so pokvarili takole: v zanki smo zapise pokvarili popolnoma naključno, in sicer v prvi tretjini zapisa, v osrednjem delu in v zadnji tretjini zapisa. Mesto in »težo« napake je shranjena procedura izbrala naključno (slika 1). Tako se je lahko npr. na posameznem območju zapisa napaka pojavila na enem mestu, na dveh mestih ali več.

Ker smo tipografske napake povzročili nadzorovano, smo vedeli, kateri zapisi spadajo skupaj, in temu primerno smo lahko tudi izračunali učinkovitost delovanja algoritma Jaro-Winkler, in sicer smo v vsakem pokvarjenem nizu podatkov izračunali odstotek napak pri povezovanju zapisov.

Zanimala nas je le pravilnost povezovanja zapisov, pri čemer smo zanemarili hitrost delovanja algoritma. Ta je problematična pri uporabi hibridne različice algoritma Jaro-Winkler, saj število potrebnih operacij eksponentno narašča. V takšnih primerih je nujna uporaba inteligentno predizbranih zapisov, v našem primeru smo uporabili kartezijski produkt. V realnih poslovnih okoliščinah bi bil takšen pristop tako rekoč neučinkovit, saj bi postopek obdelovanja zapisa potekal prepočasi. Ob upoštevanju 1000 zapisov, potrebnih povezovanja, in velikosti zapisov 20 znakov (podnizi so dolgi polovico dolžine celotnega zapisa + 1), mora hibridni algoritem obdelati 1000 × 1000 × 11 × 11 = 121,000.000 zapisov.

4.1 Algoritem Jaro-Winkler

Algoritem Jaro kombinira število skupnih znakov ter število potrebnih operacij, da niz s pretvorimo v niz t.

$$D(s, t) = \frac{1}{3}\left(\frac{m}{|s|} + \frac{m}{|t|} + \frac{m-n}{m}\right) \quad (1)$$

m število skupnih znakov
$|s|$ ter $|t|$ dolžina nizov
n število potrebnih operacij (transpositions)

Za izračun m pridejo v poštev le znaki, ki so blizu skupaj. Bližina je določena kot razdalja med položaji posameznih znakov (angl. by character position distance, CPD). Bližina algoritmu pove, v kakšnem

[1] Algoritem Jaro-Winkler je nadgradnja algoritma Jaro, ki ga je predstavil Jaro (1989) za povezovanje podatkov s tipografskimi napakami.

```
DECLARE @i AS INT = 0,
    @j AS INT = 0,
    @m AS INT ;
WHILE ( @i < 3 )
    BEGIN
        SET @i += 1 ;
        SET @j = @i - 2 ;
        WITH    RandomNumbersCTE
                  AS ( SELECT   CustomerId,
                                RAND(CHECKSUM(NEWID()) % 1000000000
                                     + CustomerId) AS RandomNumber1,
                                RAND(CHECKSUM(NEWID()) % 1000000000
                                     + CustomerId) AS RandomNumber2,
                                RAND(CHECKSUM(NEWID()) % 1000000000
                                     + CustomerId) AS RandomNumber3,
                                Random,
                                FirstThird,
                                CentreThird,
                                LastThird,
                                Reverse,
                                LongString,
                                UpDated
                       FROM     dbo.CustomersTargetAllInOne
                     )
            UPDATE  RandomNumbersCTE
                                    SET     Random = dbo.MakeDirtyData(Random,
                                    CAST(CEILING(RandomNumber1
                                                 * LEN(Random)) AS INT),
                                    RandomNumber1, RandomNumber2,
                                    RandomNumber3,
                                    CHAR(CEILING(RandomNumber2 * 26)
                                         + 96)),
                    FirstThird = dbo.MakeDirtyData(FirstThird,
                                       ABS(CAST(LEN(FirstThird) / 3 AS INT)+2
                                           - CAST(CEILING(RandomNumber1 * LEN(FirstThird)/3) AS INT)),
                                       RandomNumber1, RandomNumber2,
```

Slika 1: **Del shranjene procedure za kreiranje »šuma« v podatkih**

obsegu naj v primerjanem nizu išče znak iz prvega niza.

$$\mathrm{CPD} = \left\lfloor \frac{Max\,(|s|, |t|)}{2} \right\rfloor - 1 \qquad (2)$$

Če npr. primerjamo niz ALEŠ z nizom ANDREJ, je obseg iskanja 6/2 – 1 = 2. Torej bo algoritem črko A iz prvega niza iskal na prvem in drugem mestu v nizu ANDREJ. Črko L bo iskal na prvem, drugem in tretjem mestu, črko E pa na drugem, tretjem in četrtem mestu v nizu ANDREJ.

Winkler je kasneje algoritem Jaro dopolnil z mero lp, ki nam da boljše rezultate v primerih, ko imata dva niza identičen začetek (do prvih štirih znakov).

$$\mathrm{JW}(s,t) = d_j + (l_p \times (1 - d_j)) \qquad (3)$$

d_j Jaro algoritem
l_p korekcijski faktor za največ prve štiri skupne znake; vrednost ne sme biti večja od 0,25

Vrednost lp dobimo kot max (št. začetnih skupnih znakov, 4) / 10.
Če imamo niza NOVAK ter NOVKA, je Jaro-Winkler index (enačba 3) = 0,8.

4.2 Hibridni algoritem Jaro-Winkler

Hibridni algoritem, predstavljen v enačbi 4, za osnovo uporablja podnize posameznega zapisa, nato pa med posameznimi kombinacijami podnizov uporablja določen osnovni algoritem.

$$\mathrm{sim}(s, t) = \frac{1}{K} \sum_{i=1}^{K} max_{j=1}^{L}\, SIM\,(A_i, B_j) \qquad (4)$$

K število podnizov v zapisu s
L število podnizov v zapisu t
A_i podniz zapisa s
B_j podniz zapisa t
SIM osnovni algoritem (npr. Jaro-Winkler)

Za dolžino podnizov 3 bi z uporabo algoritma Jaro-Winkler kot osnovnega algoritma iz enačbe 4 na podatku iz preglednice 1 dobili:

$$\text{sim (NOVAK, NOVKA)} = \frac{1}{3}(\max(SIM(NOV, NOV), SIM(NOV, OVK), SIM(NOV, VKA)) +$$
$$\max(SIM(OVA, NOV), SIM(OVA, OVK), SIM(OVA, VKA)) + \max(SIM(VAK, NOV), SIM(VAK, OVK),$$
$$SIM(VAK, VKA))) = \frac{1}{3}(1 + 0{,}67 + 0{,}67) = 0{,}78$$

Z dolžino podnizov 4 pa bi bila vrednost hibridnega algoritma Jaro-Winkler še nižja, in sicer 0,75.

Preglednica 1: **Primer podnizov različnih dolžin**

NOVAK		**NOVKA**	
Dolžina podniza			
3	4	3	4
NOV	NOVA	NOV	NOVK
OVA	OVAK	OVK	OVKA
VAK		VKA	

5 SIMULACIJA UČINKOVITOSTI ALGORITMA JARO-WINKLER

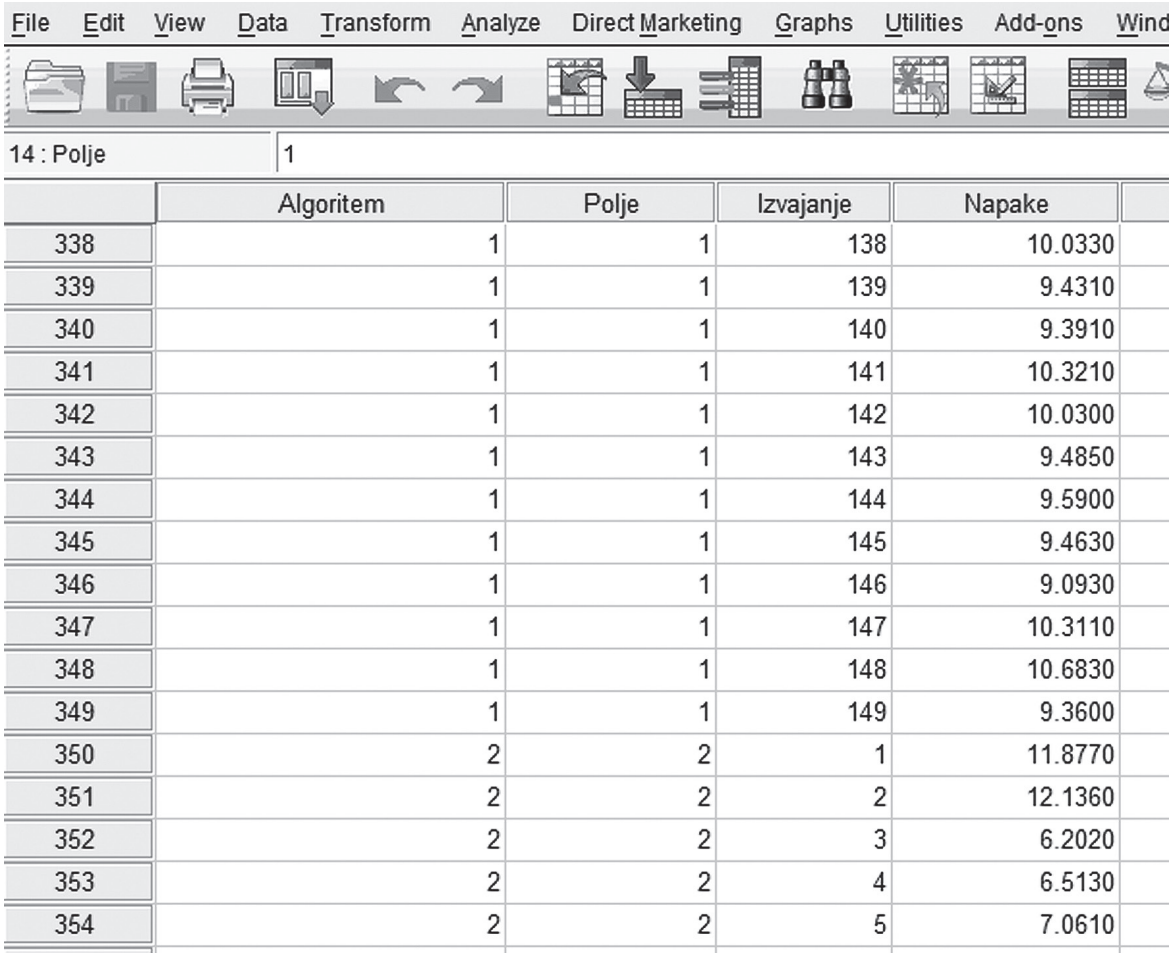
V podatkovni zbirki AdventureWorks smo nadzorovano pokvarili približno tretjino od 18.000 zapisov entitete Osebe. Območja napak smo razdelili na tri dele, in sicer na začetni, osrednji in zadnji del. Posamezni del je bil širok približno tretjino celotne dolžine posameznega podatka (preglednica 2). Poleg tega smo pokvarili podatke tudi popolnoma naključno, torej brez vedenja o mestu pojavljanja tipografske napake.

Preglednica 2: **Prikaz »pokvarjenih« podatkov glede na območje pojavljanja tipografske napake**

Pravilni podatek	**Naključne napake**	**Napake v prvi tretjini**	**Napake v sredini**	**Napake v zadnji tretjini**
Blake Edwards	hlake Edwayds	Byakh Edwards	Blaky Ehwards	Blake Edhayds
Wyatt Turner	Wyattqqurner	Wyqqt Turner	Wyattqqurner	Wyatt Turqer
Brad Deng	arad Dejg	Bjaa Deng	Braj aeng	Brad Dajg
Andres Shen	izdres Shen	Andies Shen	Andrei Shen	Andres ihen
Virginia Srini	Vijginia Srinx	xirgjnia Srini	Virxinij Srini	Virginia jrixi
Angela Flores	Angelk Fllres	Algkla Flores	AngelakFlores	Angela Flkles
Dalton Gonzalez	Dalton Gonzdlsz	Dsdton Gonzalez	DaltosdGonzalez	Dalton Gonzalsz

Ker smo želeli doseči naključnost, smo za vsak izračun/polje posebej pokvarili podatke. Za preverjanje algoritma Jaro-Winkler brez uporabe hibridne različice smo naredili 150 izračunov za vsako polje, torej skupno 600 izračunov. Pri simulaciji hibridne različice smo uporabili manjši vzorec zapisov entitete Oseba, zato smo naredili 50 izračunov več, tj. 200 izračunov za vsako polje, tj. 800 izračunov. Skupaj je bilo opravljenih 1400 izračunov.

Posamezni izračun je torej vseboval proces naključnega ustvarjanja napak v posameznem območju zapisa, uporabo algoritma Jaro-Winkler (osnovne ter hibridne različice) za povezovanje zapisov med seboj ter računanje odstotka napak pri povezovanju zapisov. Rezultate simulacije smo vnesli v program SPSS (slika 2).



File Edit View Data Transform Analyze Direct Marketing Graphs Utilities Add-ons Wind

14 : Polje 1

	Algoritem	Polje	Izvajanje	Napake
338	1	1	138	10.0330
339	1	1	139	9.4310
340	1	1	140	9.3910
341	1	1	141	10.3210
342	1	1	142	10.0300
343	1	1	143	9.4850
344	1	1	144	9.5900
345	1	1	145	9.4630
346	1	1	146	9.0930
347	1	1	147	10.3110
348	1	1	148	10.6830
349	1	1	149	9.3600
350	2	2	1	11.8770
351	2	2	2	12.1360
352	2	2	3	6.2020
353	2	2	4	6.5130
354	2	2	5	7.0610

Slika 2: **Rezultati simulacije Jaro-Winkler algoritma v SPSS-u**

V ta namen smo ustvarili shranjeno proceduro ter uporabniško določeno funkcijo, ki nam omogočata nadzorovano simulacijo posameznega algoritma. Hibridna različica algoritma v MS SQL-u ne obstaja, zato smo jo morali napisati sami (slika 3). Hibridna različica uporablja obstoječi sistemski funkciji mdq.Similarity ter mdq.NGrams. Prva vsebuje štiri najbolj razširjene algoritme za povezovanje kratkih zapisov s tipografskimi napakimi (algoritem edit distance, algoritem Jaro-Winkler, algoritem Jaccard ter algoritem Simil). Funkcija NGrams pa razdeli posamezni zapis na dolžino podnizov, ki jo določimo. Skupek obeh sistemskih funkcij nam omogoča izgradnjo hibridne različice omenjenih algoritmov.

Največji izziv pri izdelavi hibridnega algoritma je opredelitev dolžine podniza posameznega zapisa. V našem primeru smo uporabili izsledke, ki sta nam jih predstavila Godnov in Dular (2012). Avtorja sta pri simulaciji algoritma edit distance najboljše rezultate dosegla pri dolžini podniza min(s,t) / 2 + 1. Zato smo tudi za hibridni algoritem Jaro-Winkler uporabili predlagano dolžino podniza.

1400 izračunov smo vnesli v program SPSS, s katerim smo jih obdelali podrobneje. Najprej smo izračunali povprečje napak pri posameznem algoritmu ter posameznem polju (preglednica 3).

```
CREATE FUNCTION [dbo].[HyBridFunctionSQL](@String1 nvarchar(max), @String2 nvarchar(max),
@NGrams int, @Method int, @Bias float, @MinScore float)
RETURNS FLOAT
AS
BEGIN

DECLARE @Index FLOAT;

WITH  FirstGrams(FirstToken)
AS
  (SELECT A.*
     FROM
       (SELECT TOKEN FROM mdq.NGrams(@String1,@NGrams,1)) AS A
      WHERE LEN(RTRIM(LTRIM(A.TOKEN)))=@NGrams),

      SecondGrams(SecondGrams)
AS
   (SELECT A.*
     FROM
       (SELECT TOKEN FROM mdq.NGrams(@String2,@NGrams,1)) AS A
      WHERE LEN(RTRIM(LTRIM(A.TOKEN)))=@NGrams),

      InnerValues(Value)
AS
 (SELECT
MAX(mdq.Similarity(FirstGrams.FirstToken,SecondGrams.SecondGrams,@Method,@Bias,@MinScore))
AS INNERVALUE
   FROM FirstGrams
     CROSS JOIN SecondGrams
     GROUP BY FirstToken)


 SELECT @Index=SUM(Value)/COUNT(*)
    FROM InnerValues

 RETURN  @Index
END
```

Slika 3: **Funkcija hibridnega algoritma v MS SQL**

Preglednica 3: **Izračunana povprečje ter standardni odklon simulacije algoritma Jaro-Winkler**

Algoritem	Polje	Povprečje %	Standardni odklon
Jaro-Winkler	Prva tretjina	10,02	0,92
	Osrednji del	6,47	0,60
	Zadnja tretjina	5,01	0,22
	Naključno	11,60	0,99
Hibridni Jaro-Winkler	Prva tretjina	3,86	0,82
	Osrednji del	6,75	1,04
	Zadnja tretjina	2,45	0,71
	Naključno	5,57	0,96

Največ napak se je pojavilo pri osnovnem algoritmu Jaro-Winkler, če so se tipografske napake pojavile naključno ter v prvi tretjini zapisa. Simulacija nam je potrdila dejstvo, da je osnovni algoritem Jaro-Winkler zelo občutljiv na napake, ki se pojavijo na začetku zapisa, medtem ko hibridni Jaro-Winkler odpravi to slabost. Preglednica 3 nam tudi pokaže, da je hibridni algoritem Jaro-Winkler mnogo bolj natančen kot osnovni.

Preglednica 4: **Primer napačnega povezovanja kratkih zapisov z običajnim algoritmom Jaro-Winkler**

Izvirni zapis	Povezani zapis	Vrednost algoritma
Jonathan Henderson	Narhan Henderson	0,923611111
Jonathan Henderson	Jonathnn Henderson	0,918954248
Jonathan Henderson	Jonatfan Nelson	0,902393162

Kot je razvidno iz preglednice 4, je algoritem Jaro-Winkler napačno povezal kratka zapisa. Pravilna povezava je v osenčenih poljih. Z uporabo hibridnega algoritma Jaro-Winkler dobimo rezultate v preglednici 5. Hibridni algoritem je tokrat pravilno povezal zapisa.

Preglednica 5: **Primer pravilnega povezovanja kratkih zapisov s hibridnim algoritmom Jaro-Winkler**

Izvirni zapis	Povezani zapis	Vrednost algoritma
Jonathan Henderson	Narhan Henderson	0,930556
Jonathan Henderson	Jonathnn Henderson	0,957407
Jonathan Henderson	Jonatfan Nelson	0,638889

Razlike v uspešnosti algoritmov glede na mesto pojavljanja napak, tj. naključno, prva tretjina, osrednji del in zadnja tretjina, je potrdila tudi analiza ANOVA.[2] S posthoc testom Games-Howell smo potrdili, da je razlika v odstotkih med polji statistično značilna tako pri osnovnem algoritmu Jaro-Winkler kot pri hibridnem.[3] Algoritma sta se najbolje obnesla, če so bile tipografske napake v zadnji tretjini zapisa. Osnovni algoritem Jaro-Winkler se je najslabše obnesel, če so bile tipografske napake na začetku zapisa, s hibridno različico smo odpravili to pomanjkljivost. S hibridnim algoritmom smo uspeli prepoloviti odstotek napak pri povezovanju kratkih zapisov, pri katerem se napake pojavljajo naključno, tj. da ne vemo, kako se tipografske napake pojavljajo v zapisih.

6 SKLEP

Zaradi ogromnih količin podatkov bo njihovo povezovanje vedno večji izziv. Na srečo nam je relacijski podatkovni model prinesel lastnosti entitet, ki enolično določajo posamezni zapis določene entitete, in je povezovanje tako lažje. A še vedno se pojavljajo in se bodo pojavljale okoliščine, v katerih nimamo oz. ne bomo imeli takšnih enoličnih lastnosti. V takih primerih so nam v veliko pomoč algoritmi, ki nam pomagajo pri samodejnem povezovanju zapisov, ki imajo tipografske napake. Administrativno določiti mejo, kdaj sta dva zapisa ista, pri čemer ima eden ali pa celo oba tipografske napake, je tako rekoč nemogoče. Smiseln bi bil vrstni red, da bi najprej povezali in s tem iz procesa povezovanja odstranili zapise, pri katerih je vrednost algoritma enaka 1, tj. da so primerjani zapisi enaki. Nato pa bi postopoma zniževali mejo, ki jo mora doseči algoritem, da dva zapisa lahko smatramo kot ista zapisa.

Eden najpogosteje uporabljenih algoritmom je Jaro-Winkler in njegovo učinkovitost smo preizkušali ter jo opisali v članku. Pri preizkušanju smo izpustili inteligentni predizbor zapisov,[4] ki pa je pri večjem številu tako rekoč neizogiben. Pokazali smo, kako je hibridni algoritem Jaro-Winkler uspešnejši od običajnega z vidika natančnosti pri povezovanju, pri čemer smo zanemarili dejstvo porabe časa.

Na podlagi izračunov lahko tudi podamo priporočila za uporabo omenjenega algoritma. Kadar koli moramo povezati majhno število kratkih zapisov (< 2000), vedno uporabimo hibridno različico algoritma Jaro-Winkler, pri kateri je dolžina podniza enaka polovici dolžine krajšega zapisa + 1.

Če moramo povezati večje število kratkih zapisov in vemo, kje se pojavljajo tipografske napake, lahko v primeru, če se te napake pojavljajo v večini primerov na koncu zapisa, uporabimo običajni algoritem Jaro-Winkler. V nasprotnih primerih pa predvidevamo, da bi bila nujna uporaba hibridnega algoritma Jaro-Winkler z inteligentnim predizborom zapisov.

7 LITERATURA

[1] Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P. & Fienberg, S. (2003). Adaptive name matching in information integration. *IEEE Intelligent Systems*, *18*(5), 16–23.

[2] Cohen, W., Ravikumar, P. & Fienberg, S. (2003). A Comparison of String Distance Metrics for Name-Matching Tasks. Proceedings of IJCAI-03 Workshop on Information Integration.

[3] Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, *39*(11), 27–34.

[4] Fellegi, I. P. & Sunter, A. B. (1969, December 1). A Theory for Record Linkage. *Journal of the American Statistical Association*, *64*(328), 1183–1210.

[5] Godnov, U. & Dular, T. (2012). Povezovanje kratkih zapisov s tipografskimi napakami : primer aplikacije algoritma edit distance. Predstavljeno na konferenci Dnevi slovenske informatike 2012: Ustvarimo nove rešitve!, Portorož: Slovensko društvo Informatika.

[6] Gravano, L., Ipeirotis, P. G., Jagadish, H. V., Koudas, N., Muthukrishnan, S. & Srivastava, D. (2001). Approximate String Joins in a Database (Almost) for Free. Proceedings of the 27th International Conference on Very Large Data Bases, Rome, Italy. Pridobljeno z naslova http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.5630.

[7] Jaro, M. A. (1989). Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, *84*(406), 414–420.

[8] Jin, L., Li, C. & Mehrotra, S. (2003). Efficient Record Linkage in Large Data Sets. Proceedings of the Eighth International Conference on Database Systems for Advanced Applications. Pridobljeno z naslova http://dl.acm.org/citation.cfm?id=789250.

[2] Tako pri osnovni različici algoritma Jaro-Winkler kot tudi pri hibridni različici je pri ANOVA testu Sig.<0,001.

[3] Sig. je pri obeh algoritmih in vseh kombinacijah polj manjša od 0,001.

[4] Inteligentni predizbor je procedura, ki nam omogoča, da med seboj primerjamo manjšo količino zapisov po nekih pravilih, običajno s pristopom NGrams. Tako zožimo problemski prostor in povečamo hitrost ter natančnost algoritmov za povezovanje kratkih zapisov s tipografskimi napakami.

[9] Mccallum, A. & Wellner, B. (2003). Object Consolodation by Graph Partitioning with a Conditionally-Trained Distance Metric. Proceedings of the ACM Workshop on Data Cleaning, Record Linkage and Object Identification, Washington, D. C. Pridobljeno z naslova http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.4.3532.

[10] Newcombe, H. B., Kennedy, J. M., Axford, S. J. & James, A. P. (1959, Oktober). Automatic Linkage of Vital Records. *Science*, *130*(3381), 954–959.

[11] Russom, P. (2006). Liability and Leverage – A Case for Data Quality. Pridobljeno z naslova http://www.information-management.com/issues/20060801/1060128-1.html.

[12] Sarka, D. & Mauri, D. (2011). Data Quality and Master Data Management with SQL Server 2008 R2. Retrieved from http://www.solidq.com/ce-en/News/Pages/Data-Quality-and-Master-Data-Management-with-Microsoft-SQL-Server-2008-R2.aspx.

[13] Stenetorp, P., S. Pyysalo & Tsujii, J. (2011). SimSem: Fast Approximate String Matching in Relation to Semantic Category Disambiguation, Proceedings of BioNLP 2011 Workshop.

[14] Tejada, S. (2001). Learning object identification rules for information integration. *Information Systems*, *26*(8), 607–633.

[15] Waddington, D. (2009, July 17). The State of Data Quality Today. Pridobljeno z naslova http://www.it-analysis.com/business/quality/content.php?cid=11417.

[16] Winkler, W. (2006). *Overview of Record Linkage and Current Research Directions* (No. Statistics #2006-2). U. S. Bureau of the Census. Pridobljeno z naslova http://www.census.gov/srd/papers/pdf/rrs2006-02.pdf.

[17] Winkler, W. (1995). Matching and Record Linkage. *BUSINESS SURVEY METHODS*, 355–384.

▪

Uroš Godnov je doktoriral na področju informacijsko-upravljalskih ved. Že več kot desetletje se ukvarja s področjem podatkovnih zbirk, bil pa je tudi arhitekt in programer poslovnoinformacijskih rešitev, tako klasičnih, kot tistih v računalniškem oblaku. V zadnjih letih večino časa posveča področju kakovosti podatkov; s kolegom Dejanom Sarko sta ustanovila Inštitut za kakovost podatkov. V okviru inštituta sta Godnov in Sarka razvila tečaje s področja kakovosti podatkov, ki udeležencem pokažejo pomen kakovosti podatkov za podjetja in druge organizacije.