Využití Fuzzy Match algoritmu pro čištění dat

Ing. David Pejčoch, DiS.

Úsek pojištění motorových vozidel, Kooperativa, pojišťovna, a.s., Vienna Insurance Group,

dpejcoch@koop.cz, Templová 747, 110 01 Praha 1, Czech Republic

Katedra informačního a znalostního inženýrství, FIS-VŠE Praha.

Publikováno: 17.1.2008

Abstrakt

S enormním nárůstem objemu zpracovávaných dat v podnicích se stále častěji setkáváme s problematikou datové kvality. Nečistota v datech s sebou přináší jak ekonomické, tak analytické důsledky. V některých případech, jako je např. válečná logistika či kosmický program, nečistota dat dokonce rozhoduje o životě či smrti jednotlivců a jejich skupin.

V rámci procesu čištění dat lze identifikovat krok standardizace, v němž jsou aktuální data porovnávána s daty referenčními a při dostatečné shodě jsou zaměněna za data referenční. Pro tento účel je ve většině případů nedostačující klasický přístup založený pouze na absolutní shodě pomocí SQL JOIN. Je tedy třeba se inspirovat teorií fuzzy logiky a použít přístup, který se nazývá Fuzzy Match. Tento přístup vyhodnocuje míru shody srovnávané a referenční sady pomocí tzv. similarity function. Automaticky sloučeny jsou poté záznamy se similarity function vyšší než stanovená prahová hodnota. Ostatní případy se skóre nižším než prahová hodnota lze z hlediska sloučení posuzovat individuálně.

Klíčová slova

Data Quality, Fuzzy Match, Čistota dat, fuzzy logika.

Úvod do problematiky datové kvality a čistoty dat

V úvodu pokládám za účelné krátce se zmínit o problematice datové kvality, popsat proces datové kvality a vymezit funkci Fuzzy Merge v konkrétním jeho kroku.

Data Quality

Problematika kvality dat (angl. Data Quality) a na ní navazující problematika kvality informací (angl. Information Quality) se zejména s ohledem na enormní nárůst objemu zpracovávaných dat ve firmách dostala v posledním desetiletí do popředí zájmu. Firmy nejprve pochopily, že data pro ně v informační době znamenají velice cenné aktivum[1], účinnou zbraň v boji o udržení a získání nových zákazníků. S rostoucím uvědoměním si důležitosti vlastních dat se role kvality dat posunula od přednosti k nutnosti. Ten, kdo díky duplicitním záznamům v klientských datech frustruje své zákazníky opakovaným oslovováním v marketingových kampaních o tyto zákazníky dříve či později přijde. Ten, kdo zasílá svým zákazníkům permanentně chybně vyplněné faktury, dopadne stejně.

V další fázi firmy začaly chápat, že retroaktivní přístup k datové kvalitě pomocí ad-hoc prováděných akcí „čistá data“ se stává neúčinným, pokud nemá odraz ve změně podnikových procesů, v rámci nichž tyto nečistoty vznikají. Současně si firmy začaly uvědomovat nutnost integrace čistých dat v rámci různých datových zdrojů. Začalo se mluvit o jediném pohledu na zákazníka (resp. produkt) a skloňovat pojmy jako je Master Data Management (MDM) či Customer Data Integration (CDI)[2].

V souvislosti s datovou kvalitou se zpravidla (např. **[2]**) hovoří o vlastnostech dat jako je přesnost, objektivnost, důvěryhodnost, reputace, dostupnost, bezpečnost přístupu, relevantnost, přínos dat, včasnost, úplnost, rozsah, interoperabilita, srozumitelnost, výstižná a stručná reprezentace, konzistentní reprezentace, apod. Již z prvního pohledu je zřejmé, že část těchto vlastností dat je numericky měřitelná (hovoříme o tzv. objektivních metrikách) a část z těchto vlastností představuje

[1] Např. **[10]** tvrdí, že pokud jsou informace měnou nové ekonomiky, pak data jsou kritickou surovinou nezbytnou k úspěchu.

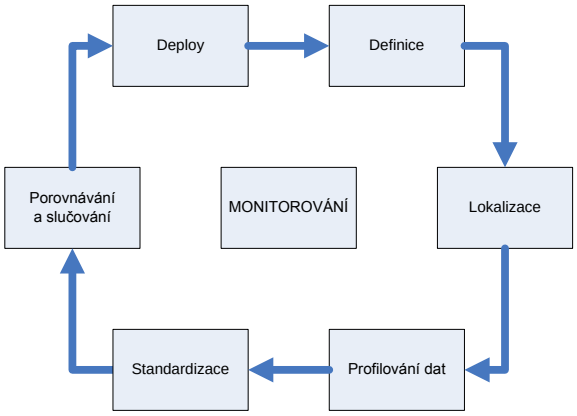
[2] Podle **[9]** je CDI soubor procesů, pravidel, mechanismů a dovedností nezbytných pro standardizaci a integraci zákaznických dat pocházejících z různých zdrojů. MDM je v podstatě totéž napříč všemi předmětnými oblastmi ve firmě.

metriky subjektivní, které lze jen obtížně numericky kvantifikovat (uvádí např. **[3]**). V souvislosti s Fuzzy Merge se v této práci věnuji zajištění vlastností dat, jako je přesnost, úplnost a konzistentní reprezentace, tedy vlastností měřitelných objektivními metrikami.

Proces Data Quality (DQ)

V této části práce bych rád jako příklad procesu DQ zmínil proces používaný pro potřeby CDI (Customer Data Integration) tak, jak jej uvádí **[9]** a v rámci něj vymezil funkci Fuzzy Match. Cílem CDI je vytvoření jednotné báze zákaznických dat, tedy zdroje jediné pravdy o zákaznících. Z tohoto důvodu je v tomto pojetí procesu DQ obsažen i krok popisující definici zamýšlené výsledné datové struktury této jednotné báze, tzv. Data Customer Hub. Tuto variantu procesu DQ uvádím z důvodu její lepší přehlednosti, jíž se odlišuje např. od poněkud komplikovaného procesu TQdM (Total Quality data Management) DQ guru Larryho Englishe, prezentovaného v **[12]**. Tak, jako mnoho jiných procesů (např. různé varianty procesu KDD[3]), je i pro proces DQ typický jeho cyklický charakter, naznačující, že se nejedná o jednorázový krok.

Schéma procesu Data Quality pro potřeby CDI



Zdroj: Překresleno podle schématu DQ uvedeném v **[9]**

Definice

V tomto kroku procesu DQ probíhá definice výsledné datové báze, formulace požadavků na datový obsah z pohledu byznysu, požadavků na formáty reprezentující tento obsah a vztah zdrojových dat k ostatnímu datovému obsahu.

Lokalizace

V rámci kroku Lokalizace probíhá shromažďování informací o disponibilních datových zdrojích a jejich následná validace. Podstatným krokem je zde nalezení referenčních datových sad.

Profilace dat (Data profiling)

Tento krok v sobě zahrnuje zkoumání struktury dat. Data jsou nejprve zkoumána z hlediska jejich shody s metadaty (z hlediska datového typu, délky atributu, unikátnosti dat a chybějících hodnot). Dále je ověřováno, zda jednotlivé atributy svou strukturou odpovídají svým předpokladům, tj. např. zda obsah atributu, který má obsahovat rodné číslo, odpovídá pravidlům pro rodné číslo (ověření modula, ověření platných variant zápisu). V dalším kroku jsou spočteny popisné statistiky a na jejich základě identifikovány odlehlé hodnoty, které představují potenciální kandidáty na chybné hodnoty.

Standardizace

V rámci standardizace je provedeno parsování dat, tj. rozdělení řetězců obsažených v jednotlivých atributech na jejich části. Např. obsah atributu adresa je rozdělen na název ulice, číslo orientační a číslo popisné, obsah atributu jméno je rozdělen na titul, křestní jméno a příjmení. Dále je provedeno sémantické sladění různých variant slov vzniklých např. v rámci různých nářečí a dále srovnání a sladění formátů s existujícími standardy[4].

Porovnání a slučování

Nyní se dostáváme k té fázi procesu DQ, v níž nachází své uplatnění technika Fuzzy Match. Cílem této fáze totiž je eliminace duplicit a eventuelní zkombinování duplicit do výsledné nejlepší verze záznamu. V případě CDI sem patří rovněž i určování domácností.

Deploy

V popisu procesu DQ uvedeném v **[9]** je opět tato fáze procesu označena za specifickou pro CDI a představuje přesun výsledné datové sady do Customer Data Hubu. Při běžném DQ projektu by tento krok představoval vrácení vyčištěných dat do zdrojového systému, resp. oprava dat ve zdrojovém souboru pro data mining.

Monitorování

Kvalitu dat je samozřejmě nutné průběžně monitorovat. A to ve všech fázích procesu DQ. Zejména je třeba sledovat zlepšující se datovou kvalitu za účelem stanovení ROI DQ projektů.

[3] Knowledge Discovery in Databases

[4] Většina nástrojů pro data quality umožňuje v USA porovnávat se standardem zapisování adres definovaným U.S. Postal Service a s dalšími poštovními standardy

Selhání klasického SQL JOIN a LOOKUP

Použití klasického SQL JOIN ve fázi porovnávání a slučování je obecně možné pouze v případě předpokladu naprosté shody porovnávacího klíče. Takovým případem je např. join přes rodné číslo nebo IČO v případě, že tento atribut obsahuje pouze verifikované validní hodnoty[5].

Příklad situace, kdy klasický SQL JOIN jako metoda pro porovnání a sloučení vzorku dat s referenčními daty nestačí, uvádí např. Charles Patridge[6] v **[13]**. Popisuje situaci, kdy bylo programátorovi zadáno zřetězení seznamu potenciálních klientů na seznam stávajících klientů. Cílem byla identifikace těch klientů ze seznamu potenciálních, kteří již v daný okamžik jsou klienty firmy. Datový soubor potenciálních klientů neměl žádný jedinečný klíč. Programátor se pokusil zřetězit jej přes kombinaci ZIP kódu a jména klienta. Výsledky však byly žalostné. Pomocí merge (obdoba join při použití SAS data step) se programátorovi podařilo zřetězit pouhých 21% záznamů, zatímco asistentce ředitele, která se o to samé pokusila pomocí ručního srovnávání, se za tu samou dobu podařilo ověřit prvních 100 záznamů (celý seznam měl řádově stovky záznamů). Příčina neúspěchu aplikace SQL JOIN spočívala v různých variantách zápisu obsahu atributu jméno klienta a ZIP kódu v porovnávaném a referenčním souboru (viz. níže).

Obdobné problémy nastávají i při použití LOOKUP, ať již je realizován jakýmkoliv způsobem (v SASu např. pomocí formátu nebo načtením referenční tabulky do paměti pomocí hash objektu).

Problémy mohou nastat zejména použitím různých variant křestního jména. Např. Robert / Bob, Charles / Chuck, v prostředí České kotliny např. Mirek / Miroslav nebo Dana / Daniela. Dále např. použitím různých variant zkratek (Road / Rd, Drive / Dr, náměstí / nám. / n.).

Navíc, v dosavadním výkladu jsem doposud abstrahoval od možnosti, že obsah atributu, přes který zřetězení provádíme, není pouze jinak zapsán, ale je chybný.

Různý způsob zápisu adres v příkladu z [13]

Seznam potenciálních klientů	Referenční seznam
Chuck Patridge P D P C, Ltd. 172 Monce Road Burlington, Ct. 06013	Patridge, Charles PDPC, Ltd. Monce Rd. Burlington, Ct. 06013-2545

Zdroj: **[13]**

Fuzzy Match

V případě nemožnosti použití klasického SQL JOIN na základě unikátního klíče (tzv. deterministický přístup) a jeho selhání v případě použití složeného klíče, je třeba použít sofistikovanějších metod, často souhrnně označovaných za pravděpodobnostní spojování. Do této skupiny patří aplikace metod strojového učení, použití Fuzzy Join operátoru a metody používající principů teorie získávání informací (IR). Do poslední uvedené skupiny patří i v této práci prezentovaná metoda Fuzzy Match.

Pojďme se nyní na chvíli zastavit nad původem pojmu Fuzzy Match. **[4]** vysvětluje význam pojmu fuzzy jako „nedostatek v přesnosti definice“, např. ve smyslu rozmazaných fotografií při pohybu fotoaparátem během expozice. **[6]** uvádí, že pojem fuzzy znamená „neurčitý, nejasný, ochmýřený“, přičemž upozorňuje, že není žádoucí tento pojem překládat a je vhodné jej používat v původním anglickém znění. Historie tohoto pojmu sahá do 60. let minulého století a je zejména spojována s postavou profesora kalifornské univerzity v Berkley, Lotfi A. Zadeha a teorií fuzzy množin., tj. matematickou disciplínou programově zaměřenou na modelování fenoménu vágnosti. V devadesátých letech minulého století došlo k takové oblíbenosti pojmu fuzzy, že (jak uvádí **[6]**) bylo toto slovo v r. 1992 vyhodnoceno jako nejpoužívanější cizí slovo v Japonštině.

[16] vysvětluje pojem Fuzzy Matching jako protiklad proti hledání přesné shody. Fuzzy Matching je při současném zachování relevantnosti méně striktní než tzv. Exact Matching. Výklad pojmu Fuzzy Match jako vylepšení funkce klasického SQL JOIN / LOOKUP se objevuje např. v **[1]** nebo jako Fuzzy Match/Merge v **[13]**. **[17]** popisuje shodné techniky, které **[1]** označuje jako Fuzzy Match pod pojmem „spojování záznamů[7] - databázový přístup“.

V této práci budu nadále za Fuzzy Match považovat aplikaci technik pro spojování záznamů při řešení problému jejich deduplikace v databázích.

[5] Pro IČO i RČ (od r. 1954) slouží jako kontrola nulový zbytek po celočíselném dělení číslem 11.

[6] Charles Patridge je průkopníkem použití techniky Fuzzy Match v nástrojích SAS.

[7] angl. Record Linkage

[1] definuje K-Fuzzy Match Problem takto: Mějme referenční relaci R a práh minimální shody c z intervalu <0;1>, funkci podobnosti f a vstupní záznam u. Nalezněme množinu FM(u) fuzzy shod s nanejvýše K referenčními záznamy z R takových, že:

(i) $fms(u,v) \geq c$ pro všechna v z FM(u)

(ii) $fms(u,v) \geq fms(u,v')$ pro některá v z FM(u) a v' z R-FM(u).

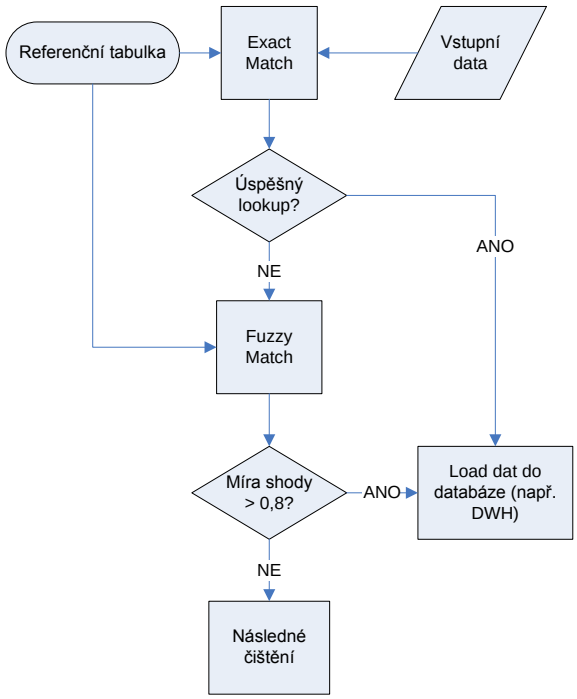
Obecně lze místo fms, jejíž přesný význam bude objasněn v dalším textu, uvažovat jakoukoliv funkci podobnosti. Cílem Fuzzy Match algoritmu je nalezení K referenčních záznamů, které jsou nejvíce podobné vstupnímu záznamu u.

Pro použití Fuzzy Match existují různé přístupy. Některé (např. **[13]**) při aplikaci Fuzzy Match používají dodatečné informace z konkrétní předmětné oblasti. Příkladem může být aplikace doménově specifického[8] slovníku synonym pro normalizaci / standardizaci dat. Zcela odlišným přístupem (publikovaným např. v **[1]**) je pokus o vývoj doménově nezávislého řešení. Tato práce se zaměřuje na prezentaci doménově nezávislých technik, neboť oblast doménové závislých řešení je natolik obsáhlá, že by zde všechna specifická pravidla nebylo možno na tomto relativně malém prostoru popsat.

Začlenění Fuzzy Match do porovnávací a slučovací fáze procesu DQ v pojetí **[1]** zachycuje níže uvedené schéma. Po neúspěšném SQL JOIN, resp. LOOKUP proti referenční tabulce je následně proveden Fuzzy Match a při dosažení předem stanovené míry shody je porovnávaný záznam nahrán do databáze (nebo nahrazen referenčním). Nutno poznamenat, že toto schéma neřeší případ, kdy by stanovenou míru shody splňovalo více referenčních záznamů. Důvody, proč není přímo aplikován Fuzzy Match a nejprve je proveden pokus o SQL JOIN jsou dle mého názoru ryze výkonové. Přes veškeré optimalizační techniky, popsané níže v této práci, je provedení Fuzzy Match lookup méně rychlé, než klasický SQL JOIN / lookup.

[8] Např. zaměřený pouze na deduplikaci adres

Schéma 1: Vzor pro aplikaci Fuzzy Match



Zdroj: **[1]**

V následujícím výkladu budu v doplňujících příkladech používat totožnou sadu referenčních záznamů a vstupních (porovnávaných) dat jaká je použita v **[1]**.

Referenční záznamy

ID	Název společnosti	Město	Stát	Zip
R1	Boeing Company	Seattle	WA	98004
R2	Bon Corporation	Seattle	WA	98014
R3	Companions	Seattle	WA	98024

Porovnávaný vzorek dat

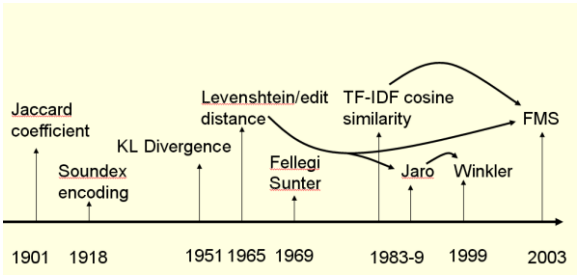
ID	Název společnosti	Město	Stát	Zip
I1	Beoing Company	Seattle	WA	98004
I2	Beoing Co.	Seattle	WA	98004
I3	Boeing Corporation	Seattle	WA	98004
I4	Company Beoing	Seattle	NULL	98004

Klasifikace měr podobnosti použitelných pro Fuzzy Match

[17] nabízí klasifikaci různých měr podobnosti, které lze používat pro porovnávání řetězců. Rozděluje je do tří kategorií. Do první skupiny řadí metriky založené na úpravách (Soundex, Levenshteinova vzdálenost / edit distance, Jarova vzdálenost a Jaro-Winklerova vzdálenost). Jejich

společným rysem je kalkulace nákladů na operace vložení, smazání a nahrazení částí řetězce při transformaci na řetězec druhý. Druhou skupinu tvoří metriky založené na tokenech (TF-IDF kosínová podobnost, Jaccardův koeficient a pravděpodobnostní modely). Třetí skupinu tvoří hybridní metriky a **[17]** do ní řadí FMS (Fuzzy Similarity Function), jíž bude v této práci věnován největší prostor. Dobu jejich vzniku a jejich vazby znázorňuje obrázek níže.

Historická osa vývoje používaných měr podobnosti



Zdroj: **[17]**

Většina těchto metrik bude stručně popsána v následujícím textu.

Jaccardův koeficient

Jaccardův koeficient je metrika podobnosti, která je aplikována na jednotlivé tokeny. Tokenizace v pojetí **[1]** představuje rozklad referenčního záznamu $R[tid, A_1, ..., A_n]$, kde $A_1,....,A_n$ jsou jednotlivé sloupce referenčního záznamu a tid představuje jeho jedinečný identifikátor, na množiny tokenů $tok(s)$ jejichž prvky jsou v referenčním záznamu oddělené nějakým oddělovačem (např. mezerou).

Např. množina tokenů $tok(v[1])$ pro referenční záznam v = R[R1, Boeing Company, Seattle, WA, 98004] je {boeing, company}.

Pokud bychom tedy uvažovali dvě množiny tokenů S = tok(v[1]) = {boeing, company} a T = tok(u[1]) = {beoing, company}, pak Jaccardův koeficient pro tyto dvě množiny by byl definován výrazem

$$Jaccard(S,T) = \frac{|S \cap T|}{|S \cup T|}.$$

Jaccardův koeficient tedy porovnává délku průniku množin S a T s délkou jejich sjednocení. V případě výše definovaných množin S a T by tedy činil 7 / (6 + 6 + 7), tedy cca 0,37. Je tedy zřejmé, že se nejedná o příliš přesnou metriku, neboť přehození jednoho písmene prakticky vylučuje shodu.

KL divergence

Kullback – Leiblerova divergence (viz. např. **[14]**), často také nazývaná informační zisk nebo relativní entropie, je mírou rozdílu dvou pravděpodobnostních rozdělení náhodných veličin P a Q. Tyto náhodné veličiny mohou být jak diskrétního typu, tak spojité. KL divergence není v pravém slova smyslu metrikou vzdálenosti (proto se nazývá divergence a ne vzdálenost), neboť není symetrická. DKL (P||Q) není rovna DKL (Q||P). Ba co více, KL divergence nesplňuje ani pravidlo trojúhelníkové nerovnosti.

Pro aplikaci DKL v oblasti IR lze použít např. tvar publikovaný v **[26]**:

$$KL(RDa \,||\, RDe) = \sum_{t \in V} p(t \mid RDa) \log \frac{p(t \mid RDa)}{p(t \mid RDe)},$$

kde RDa je aktuální dokument (v našem případě tabulka vstupních záznamů) a RDe je existující dokument (v našem případě referenční tabulka).

$$p(t \mid RDa) = \frac{n(t, RDa)}{\sum_{t \in RDa} n(t, RDa)}$$

$$p(t \mid RDe) = \frac{\sum_{d \in RDe} n(t,d) + \alpha}{\sum_{t} \left(\sum_{d \in RDe} n(t,d) + \alpha \right)},$$

kde n(t,d) je četnost řetězce t v záznamu d. Konstanta α představuje tzv. Laplaceovské vyhlazení.

Fellegi Sunter

Přístup Fellegiho a Suntera řadí **[17]**, stejně jako KL divergenci do oblasti pravděpodobnostních měr podobnosti. Jak poznamenává **[24]**, Fellegi a Sunter v r. 1969 představili formální matematický základ pro spojování záznamů. Základní pohled na tuto teorii a její použití pro spárování záznamů ze sčítání lidu v USA a dodatečných průzkumů je popsáno např. v **[25]** . Metoda uvažuje dvě populace **A** a **B**, jejichž prvky jsou a a b. Zároveň se předpokládá, že některé elementy mají tyto populace společné. Množina uspořádaných dvojic

$$AXB = \{(a,b) : a \in A,\ b \in B\}$$

je sjednocením množiny M obsahující uspořádané dvojice (*a*, *b*), pro které platí *a* = *b* a množiny U obsahující uspořádané dvojice (*a*, *b*) pro které platí $a \neq b$. Záznamy korespondující s prvky množin A a B jsou označeny $\alpha(a)$ a $\beta(b)$. Dále je definován porovnávací vektor

$$\gamma[\alpha(a), \beta(b)] = \{\gamma^1[\alpha(a), \beta(b)], ..., \gamma^K[\alpha(a), \beta(b)]\},$$

jehož každý prvek γ^i pro i = 1, ..., K, představuje specifické porovnání jednotlivých prvků množin A a B (např. porovnání atributů pohlaví, příjmení, ...).

Dále $m(\gamma)$ je definována jako podmíněná pravděpodobnost $\gamma(a,b)$ za podmínky $(a,b) \in M$ a $u(\gamma)$ jako podmíněná pravděpodobnost $\gamma(a,b)$ za podmínky $(a,b) \in U$. Podíl těchto pravděpodobností je věrohodnostní poměr R. Pokud je R > UPPER,

potom (a,b) jsou navázány. Pokud platí LOWER <= R <= UPPER, pak vazba (a, b) možná existuje. Pokud je R < LOWER, pak vazba (a,b) neexistuje. Konstanty UPPER a LOWER jsou určeny požadovanou mírou chyby.

Pro získání maximálně věrohodných odhadů podmíněných pravděpodobností potřebných k výpočtu R používá **[25]** např. algoritmus EM (Expectation Maximization).

TF-IDF kosínová podobnost

Tato metrika podobnosti je založená na výpočtu TF-IDF vah pro jednotlivé tokeny. Pro výpočet TF-IDF[9] tokenu t v daném záznamu uvádí **[17]** následující vzorec:

$$TF - IDF = \log(1+TF).\log\left(1+\frac{N}{N_t}\right)$$

Kosínovou podobnost dvou záznamů u a v, pro které jsme vytvořili množiny jejich tokenů Su a Sv a spočetli váhy jednotlivých tokenů W(t, Su) a W(t, Sv), lze určit pomocí vzorce

$$Cosine(u,v) = \sum_i W(t_i, S_u).W(t_i, S_v).$$

[17] uvádí příklad, kdy Kosínová podobnost je při aplikaci na q-tice pevné délky q = 3, vytvořené z jednotlivých tokenů, imunní vůči drobným překlepům a podobnost řetězců „Jaccard“ a „Jacard“ ohodnotí vysokým skóre. Token „Jaccard“ v tomto příkladu rozkládá na množinu q-tic {jac, acc, cca, car, ard} a token „Jacard“ na množinu q-tic {jac, aca, car, ard}.

Jarova a Jaro-Winklerova vzdálenost

Podle **[14]** je Jarova vzdálenost dvou řetězců s1 a s2 dána vzorcem $d_j = \frac{1}{3}\left(\frac{m}{|s_1|}+\frac{m}{|s_2|}+\frac{m-t}{m}\right)$, kde

m je počet znaků, které si v obou řetězcích odpovídají, t je počet transpozicí potřebných pro transformaci s1 na s2 a |s1|, |s2| jsou délky porovnávaných řetězců. Počet transformací je definován jako počet neodpovídajících si znaků v řetězci dělený číslem 2. Dva řetězce jsou označeny jako shodné, pokud Jarova vzdálenost nepřesáhne práh definovaný jako $\left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1$.

Praktické použití je možno demonstrovat na řetězcích s1 = „beoing“, s2 = „boeing“. Počet odpovídajících si znaků je m = 6, délka obou řetězců je shodně |s1| = |s2| = 6. Pro transformaci s1 na s2 je třeba zaměnit E za O a O za E. Celkový počet transpozic t je tedy 2 / 2 = 1.

	B	**E**	**O**	**I**	**N**	**G**
B	1	0	0	0	0	0
O	0	0	1	0	0	0
E	0	1	0	0	0	0
I	0	0	0	1	0	0
N	0	0	0	0	1	0
G	0	0	0	0	0	1

$$d_j = \frac{1}{3}\left(\frac{6}{6}+\frac{6}{6}+\frac{6-1}{6}\right) = 0{,}944$$

Hodnota prahu je 2. Jarova vzdálenost tedy ukazuje na shodu řetězců s1 a s2.

William Winkler (spolutvůrce rozšíření Jarovy vzdálenosti) však ve své stati **[24]** mapující v r. 1990 stav bádání a problémy výzkumu v oblasti spojování záznamů definuje tuto vzdálenost poněkud odlišně. Uvádí vzorec ve tvaru Φj(s1, s2) = 1/3 (#common/str_len1 + #common/str_len2 + 0,5 #transpositions/#common).

Jarova-Winklerova vzdálenost se od původní Jarovy liší zakomponováním tzv. škálovacího faktoru p, který zvýhodňuje shodu prvních *l* znaků. Vzorec pro výpočet má tedy tvar

$$d_w = d_j + (lp(1-d_j)).$$

Zpravidla je škálovací faktor nastaven na 0,1.

String edit distance a Levenshteinova vzdálenost

Vzdálenost ed (s1, s2) dvou řetězců s1 a s2 je minimální počet operací smazání, vložení a záměna potřebný k transformaci s1 na s2 normalizovaný maximem z délek s1 a s2. Výpočet ed v podstatě vychází z Levenshteinovy vzdálenosti tak, jak je popsán v **[18]**. Pouze ji normuje maximem délek porovnávaných řetězců.

Samotné použití ed jako kritéria pro záměnu porovnávaného záznamu za referenční není vhodné a vede často k chybám. Jako příklad je možné uvést porovnání vstupního vzorku I3 s referenčními záznamy. Jak je patrné, kritérium ed vede k závěru, že vstupní vzorek I3 je nejblíže k referenčnímu záznamu R3, neboť shledává méně nákladnou transformaci tokenu „bon“ na token „boeing“ než transformaci tokenu „company“ na token „corporation“. Ignoruje totiž fakt, že „boeing“ a „98004“ jsou tokeny s mnohem vyšší vypovídací hodnotou než je token „corporation“.

s1 = „company“

[9] Token frequency – Inverse document frequency

s2 = „corporation“

c	**o**	**m**	**p**			**a**				**n**	**y**
c	o	r	p	o	r	a	t	i	o	n	
0	0	1	0	1	1	0	1	1	1	0	1

$$ed(s_1, s_2) = \frac{7}{11} \cong 0{,}64$$

s1 = „bon“

s2 = „boeing“

b	**o**			**n**	
b	o	e	i	n	g
0	0	1	1	0	1

$$ed(s_1, s_2) = \frac{3}{6} \cong 0{,}5$$

Aplikaci ed na q-tice pro přibližný join přes proměnné typu string podrobně popisuje např. **[7]**. Při vytváření q-tic ale postupuje jiným způsobem, než **[17]** u Kosínové podobnosti a **[1]** u fms. Q-tice na počátku a konci řetězce mají kratší délku než q, a proto je doplňuje na délku q znaky # (na začátku) a $ (na konci). Současně doplňuje q-tice o informaci o pořadí a takto vzniklou uspořádanou dvojici nazývá poziční q-tice. Rozklad řetězce ‚Boeing‘ by tedy byl množinou pozičních q-tic {(1, ##b), (2, #bo), (3, boe), (4, oei), (5, ein), (6, ing), (7, ng$), (8, g$$)}

Následně **[7]** provádí dvoukrokové porovnávání, kdy v prvním kroku jednoduchým filtrem eliminují vazby ne chybně zamítnuté a v druhém kroku eliminují chybně označené vazby pomocí zakomponování ed do where podmínky. Tento postup lze aplikovat pouze v prostředí databázových systémů, které podporují uživatelem definované funkce (např. Oracle a DB2).

Query pro druhý krok by tedy vypadalo např. takto:

```
SELECT a.nazev, b.nazev
FROM a, b
WHERE edit_distance(a.nazev, b.nazev, k)
```

K je zde minimální prahová hodnota pro ed.

Nutno poznamenat, že vzhledem k tomu, že relační enginy při spuštění tohoto query vykonají ed až jako filtr po jeho proběhnutí, je tento postup značně neefektivní.

Fuzzy Similarity Function (fms)

[1] zavádí funkci FMS (Fuzzy match similarity), jejíž použití je jak univerzální v rámci všech domén, tak i odstraňuje nedostatky ED tím, že pohlíží na řetězec jako na sekvenci tokenů a uvažuje rozdílnou důležitost jednotlivých tokenů. Pro určení důležitosti jednotlivých tokenů používá IDF (Inverse document frequency). Tato metrika vychází z předpokladu, že důležitost tokenů závisí na jeho četnosti výskytu v referenčních datech. FMS zároveň uvažuje případy, kdy vstupní záznamy jsou chybné a umí se s takovou situací vypořádat.

Funkce vah IDF

Ignoranci ed vůči důležitosti jednotlivých tokenů lze řešit zakomponováním jejich váhy. Pro tento účel **[1]** používá modifikaci IDF (Inverse Document Frequency), kterou lze spočíst jako logaritmus poměru celkového počtu záznamů a četnosti konkrétního tokenu v konkrétním sloupci[10], tedy podle následujícího vzorce.

$$w(t,i) = IDF(t,i) = \log \frac{|R|}{freq(t,i)}$$

Náklady transformace

Pokud uvažujeme transformační operace nahrazení tokenu, vložení tokenu a smazání tokenu, můžeme definovat s ohledem na jednotlivé typy transformačních operací minimální náklady transformace množiny tokenů u[i] na v[i]. Součet transformačních nákladů pro všechny množiny tokenů (tedy atributy) představuje celkové náklady transformace datového záznamu u na referenční záznam v. Celkové transformační náklady tedy lze vyjádřit pomocí následujícího vzorce:

$$tc(u,v) = \sum_i tc(u[i], v[i])$$

[1] stanovuje hodnoty pro jednotlivé typy transformačních nákladů různými způsoby.

Náklady na nahrazení tokenu t1 v množině tok(u[i]) tokenem t2 z množiny tok(v[i]) lze vyjádřit jako součin vzdálenosti řetězců ed(t1,t2) a váhy tokenu t1 w(t1,i).

Náklady na vložení nového tokenu t do množiny tokenů u[i] jsou součinem konstanty cins a váhy tokenu t w(t,i). Konstantu cins **[1]** nazývá faktorem vložení tokenu a přiřazuje jí hodnotu z intervalu <0;1>.

Náklady na smazání tokenu t z množiny tokenů u[i] odpovídají jeho váze w(t,i).

Jako příklad je možné uvést kalkulaci transformačních nákladů pro převod vstupního záznamu u[Beoing Corporation, Seattle, WA, 98004] na referenční záznam v[Boeing Company, Seattle, WA, 98004]. Tato operace vyžaduje dvě dílčí transformace: nahrazení ‚beoing‘ za ‚boeing‘ a

[10] Obecně logaritmus podílu celkového počtu dokumentů v databázi a počtu dokumentů obsahujících daný výraz.

nahrazení ‚corporation' za ‚company'. Celkové transformační náklady jsou tedy součtem transformačních nákladů těchto dvou operací.

ed(‚beoing','boeing') = 0,33

w(‚beoing',1) = log (3 / 1) = 0,477

ed(‚corporation','company') = 0,64

w(‚corporation',1) = log (3 / 1) = 0,477

tc(u,v) = tc(u[1],v[1]) = ed(‚beoing','boeing') * w(‚beoing',1) + ed(‚corporation','company') * w(‚corporation',1) = 0,33 * 0,477 + 0,64 * 0,477 = 0,463

Metrika fuzzy match podobnost (fms)

Funkci fuzzy match podobnosti fms(u, v) vstupního záznamu u a referenčního záznamu v **[1]** definuje následujícím vzorcem:

$$fms(u,v) = 1 - \min\left(\frac{tc(u,v)}{w(u)}, 1\right), \text{ kde}$$

tc(u, v) jsou minimální transformační náklady vstupního záznamu u na referenční záznam v a w(u) je součet vah všech tokenů v množině tokenů tok(u) vstupního záznamu u.

V případě vstupního záznamu u[Beoing Corporation, Seattle, WA, 98004] a referenčního záznamu v[Boeing Company, Seattle, WA, 98004] je tedy výpočet fms(u, v) následující:

tc(u, v) = 0,463

w(u) = 0,125 + 0,64 + 0 + 0,125 + 0,125 = 1,015

fms(u, v) = 1 – min (0,463 / 1,015; 1) = 0,544

Vztah fms a ed

Následující přepis vztahu pro určení vzdálenosti řetězců ed(u, v) zvýrazňuje vliv implicitního přiřazení vah jednotlivých tokenů a umožňuje lepší srovnání použití ed (u, v) a fms.

$$ed(u,v) = \frac{L(u)}{\max(L(u), L(v))} \sum_i \frac{\max(|u_i|, |v_i|)}{L(u)} ed(u_i, v_i)$$

L(u) představuje součet délek všech tokenů ve vstupním záznamu u (obdobně pro referenční záznam v). Z upraveného výrazu je patrné, že ed implicitně přiřazuje váhy jednotlivým tokenům přímo úměrně jejich délce, tedy proporcionálně k poměru max(|ui|, |vi|) / L(u). Delší tokeny mají tedy při použití ed automaticky vyšší váhu.

Aproximace fms

[1] dále formuluje aproximaci fms pro případy, kdy může být různé pořadí tokenů ve vstupním a referenčním záznamu a je tedy umožněno porovnat každý token ve vstupním záznamu s každým tokenem v každém referenčním záznamu.

Míra fms^{apx} tvoří horní hranici fms. Záznamy, které se liší pouze pořadím tokenů v rámci atributů jsou fms^{apx} vyhodnoceny jako identické.

Algoritmus fms^{apx} spočívá v aplikaci fms na množiny q-tic $QG_q(s)$ vzniklé rozložením původních tokenů na sub-řetězce dané délky. Např. pro q = 3 a token s = ‚corporation' je QG3(‚corporation') množina {cor, orp, rpo, por, ora, rat, ati, tio, ion}. Do výpočtu fms^{apx} ale vstupují pouze q-tice, které představují tzv. min-hash signaturu, která je pro množinu řetězců S definována jako vektor $mh(S) = [mh_1(S), \ldots, mh_H(S)]$, jehož i-tý koordinát je definován výrazem

$$mh_i(S) = \arg\min_{a \in S} h_i(a).$$

[1] přirovnává výpočet min-hash signatury k házení šipkami na terč do zásahu elementu z S.

Aproximaci fms pro záznamy u a v definuje **[1]** pomocí následujícího výrazu:

$$fms^{apx}(u,v) = \frac{1}{w(u)} \sum_i \sum_{t \in tok(u[i])} w(t).$$

$$. \max_{r \in tok(v[i])} \left(\frac{2}{q} sim_{mh}(QG(t), QG(r)) + d_q \right), \text{ kde}$$

$sim_{mh}(QG(t), QG(r))$ je tzv. min-hash podobnost dvou q-tic vytvořených ze záznamů u a v, dq = (1-1/q).

Min-hash podobnost dvou tokenů t1 a t2 lze definovat výrazem:

$$sim_{mh}(t_1, t_2) = \frac{1}{H} \sum_{i=1}^{H} I[mh_i(QG(t_1)) = mh_i(QG(t_2))]$$

Výpočet fms^{apx} bude nejlépe demonstrovat na příkladu. Předpokládejme délku q-tic q = 3, maximální počet q-tic H = 2. Dále předpokládejme vstupní záznam v [Company Beoing, Seattle, NULL, 98004] a referenční záznam u[Boeing Company, Seattle, WA, 98004], tedy dva záznamy lišící se pouze pořadím tokenů v prvním sloupci, přesmyčkou Boeing za Beoing a chybějící zkratkou státu Washington. Předpokládejme, že min-has signatura pro vstupní záznam u je [eoi, ing], [com, pan], [sea, ttl], [980, 004] a pro referenční záznam v je [oei, ing], [com, pan], [sea, ttl], [wa], [980, 004]. Dále předpokládejme váhy jednotlivých tokenů ve vstupním záznamu w(u) = 0,25 + 0,5 +1 + 2 = 3,75.

Fms^{apx} ignoruje náklady na vložení tokenu ‚WA' a rovněž ignoruje změnu pořadí tokenů v prvním atributu. V tomto příkladu si tedy neodpovídají pouze tokeny ‚Boeing' a ‚Beoing'. $fms^{apx}(u, v)$ je tedy rovna 1 / w(u) * w(beoing) * (2/3 * 0,5 + (1 – 1/3)) = 3,75 / 3,75 * 1 = 1. Fms^{apx} tedy označila oba řetězce za totožné.

Další možnosti optimalizace použití ED

Ignorování samohlásek

[13] doporučuje při procesu porovnávání a slučování vynechání samohlásek v porovnávaných řetězcích. Zdůvodňuje to častými chybami vzniklými vlivem špatného hláskování. V anglickém jazyku zní např. Charlie stejně jako Charley. Při vynechání samohlásek by takto posouzení vzdálenosti řetězců pomocí ed ukazovalo jejich naprostou shodu.

c	**h**	**a**	**r**	**l**	**i**	**e**
c	h	a	r	l	e	y
0	0	X	0	0	X	X

Kódování pomocí algoritmu SOUNDEX

Další, poněkud komplikovanější cestou optimalizace ed, kterou okrajově zmiňuje **[13]**, je použití algoritmu SOUNDEX. Jedná se o fonetický algoritmus (viz. např. **[14]**, **[15]**), tj. algoritmus, který indexuje slova podle jejich výslovnosti. Konkrétně algoritmus SOUNDEX stejným způsobem kóduje slova, která mají velmi podobnou výslovnost. Algoritmus byl vyvinut specielně pro anglofonní prostředí a jeho použití v jiném prostředí by si vyžádalo jeho přizpůsobení. Přesný postup algoritmu je následující:

1. Zachování 1. písmene řetězce
2. Smazání písmen A,E,H,I,O z řetězce
3. Nahrazení zbývajících souhlásek čísly podle následující tabulky:

Původní písmeno	**Nahrazení číslem**
B, F, P, V	1
C, G, J, K, Q, S, X, Z	2
D, T	3
L	4
M, N	5
R	6

4. Pokud je více písmen zakódováno stejně, smaže všechna taková kromě prvního.

Dále uvádím několik příkladů použití algoritmu SOUNDEX na konkrétních dvojicích řetězců.

V prvním příkladu algoritmus SOUNDEX[11] zafunguje v anglofonním prostředí při záměně řetězců SMITH a SMYTHIE a označí je za shodné (otázkou však je, zda správně).

SMITH ➔ S53 = SMYTHIE ➔ S53

V druhém příkladu ale neoznačí za totožné řetězce KAROL a CAROL (viz. 1. bod postupu algoritmu uvedeného výše). Zde by použití výsledku algoritmu SOUNDEX jako vstupu pro ed vedlo k výsledku 1/3, zatímco aplikace ed na původní řetězce k lepšímu výsledku 1/5.

KAROL ➔ K64 != CAROL ➔ C64

Ve třetím příkladu algoritmus vede k totožnému kódování v češtině často zaměňovaných slov švec a ševc[12].

ŠEVC ➔ S12 = ŠVEC ➔ S12

Stejně tak jako ve čtvrtém a pátém příkladu shodně zakóduje podobně znějící dvojice slov kosa / koza a lup / lub.

KOSA ➔ K2 = KOZA ➔ K2

LUP ➔ L1 = LUB ➔ L1

V šestém příkladu algoritmus zafunguje i na dvojici českých jmen IVANA a IVONA.

IVANA ➔ I15 = IVONA ➔ I15

Přesto otázka reálné použitelnosti algoritmu SOUNDEX pro optimalizaci fuzzy match zůstává otevřená. Bylo by totiž nutné předem určit, na která konkrétní slova se má použít. V této souvislosti je nutné upozornit, že i v **[13]** je tato možnost uvedena pouze jako okrajová poznámka s tím, že samotného autora tohoto článku pokusy s algoritmem SOUNDEX příliš nepřesvědčily cituji: „o záruce jeho užitečnosti“.

Přiřazení vah jednotlivým sloupcům

[1] jako jedno z doplňujících vylepšení aplikace fms^{apx} uvádí možnost doplnění stávajících vah založených na IDF subjektivními vahami důležitosti jednotlivých atributů. Nové váhy použité ve výpočtu fms jsou tedy součinem původních IDF vah tokenů w(t) a sloupcových vah Wi.

[11] Pro tento příklad byla použita implementace algoritmu SOUNDEX v nástroji SAS 9.1.

[12] Můj kolega z práce se jmenuje Ševc a často je oslovovaán jako pan Švec.

Výkonnostní problémy při užití Fuzzy Match

Vzájemné porovnávání tokenů (nebo jejich q-tic) může v reálné situaci představovat značný výkonnostní problém.

Naivní algoritmus prohledává při aplikaci měr podobnosti referenční relaci R a srovnává každý její záznam se vstupním záznamem u. V rámci optimalizace je však vhodné vytvořit na referenční relaci index. Jak ovšem poznamenává **[1]**, přímou aplikaci klasických B+-tree indexů nelze uvažovat, neboť je lze použít pouze, pokud předpokládáme přesnou shodu srovnávaných atributů.

Použití M-tree indexu

Použití M-tree indexu v případě "podobnostního dotazování" v multimediálních databázových systémech, zaměřujících se na jednotnou správu záznamu hlasu, videa, obrazu, textu a numerických dat uvádí **[23]**. M-tree vytváří partition objektů na základě jejich relativní vzdálenosti měřené pomocí libovolné metriky[13]. Pro vyhodnocení podobnosti uvažuje **[23]** použití rozsahu (range) a algoritmu k - nejbližších sousedů. Range query vrací všechny indexované objekty, jejichž vzdálenost od daného objektu je menší nebo rovna zadané maximální vzdálenosti. Query používající algoritmus k - nejbližších sousedů Negativem použití této metody (stejně jako např. R-tree indexu) je její nepoužitelnost v prostředí klasických datových skladů. V podstatě lze tímto způsobem vytvořit index nad libovolnými daty, pro něž lze spočíst metriku podobnosti.

Použití Error tolerant indexu

Efektivní řešení indexace při použití fms^{apx} nabízí **[1]** v podobě tzv. Error tolerant indexu (ETI). ETI je vlastně pomocná tabulka, která přiřazuje každé min-hash signatuře její pozici v rámci atributu, číslo atributu, absolutní četnost v rámci referenční relace R a množinu identifikátorů řádků v rámci referenční relace, které signaturu obsahují[14]. Nad touto tabulkou je poté vytvořen cluster index přes atributy Q-tice, Koordinát a Sloupec.

Rozklad vstupního řetězce u na min-hash signatury o q=3, H=2

Beoing ➔ [eoi, ing] ➔ S_1, S_2

Company ➔ [com, pan] ➔ S_3, S_4

Seattle ➔ [sea, ttl] ➔ S_5, S_6

WA ➔ [wa] ➔ S_7

98004 ➔ [980, 004] ➔ S_8, S_9

Příklad vytvoření ETI pro celou referenční relaci R

Q-tice	Koordinát	Sloupec	Freq	Tid list
oei	1	1	1	{R1}
ing	2	1	1	{R1}
com	1	1	2	{R1,R3}
pan	2	1	2	{R1,R3}
bon	1	1	1	{R2}
orp	1	1	1	{R2}
ati	2	1	1	{R2}
sea	1	2	3	{R1,R2,R3}
ttl	2	2	3	{R1,R2,R3}
wa	1	3	3	{R1,R2,R3}
980	1	4	3	{R1,R2,R3}
004	2	4	1	{R1}
014	2	4	1	{R2}
024	2	4	1	{R3}

Postup Fuzzy Match má tedy následující podobu:

1. Inicializace hash tabulky pro tid skóre; Korigující člen je nastaven na nulu
2. Provedení tokenizace vstupního záznamu u a následně odvozeny min-hash signatury ze všech vzniklých tokenů
3. Přiřazení vahám jednotlivým tokenům; RemWt = součet všech vah
4. Nastavení prahové hodnoty jako součinu původního prahu minimální shody c a RemWt
5. Pro každou min-hash signaturu s pořadím l v daném tokenu a sloupci upravit korigující člen += váha příslušného tokenu * (1-1/q)
6. Pro každou min-hash signaturu proveden lookup do ETI tabulky
7. Pro každou min-hash signaturu proveden update tid skóre. Skóre existujících tid je zvýšeno o váhu dané q-tice spočtené jako podíl váhy tokenu a délky q-tice, pokud pro ní v ETI tabulce existuje tid. Pokud je RemWt vyšší nebo rovno prahové hodnotě,

[13] M-tree je po této stránce plně parametrizované a konkrétní funkce použitá jako metrika je pro něj černou skříňkou.

[14] Realizace tabulky ETI pomocí standardních SQL dotazů si samozřjemě vyžaduje vytvoření mezikroku [Q-tice, Koordinát, Sloupec, Tid].

jsou vloženy nové tid se skóre odpovídající podílu váhy tokenu a délce dané q-tice.

8. Pro každou min-hash signaturu RemWt -= váha q-tice
9. Vybrat všechny záznamy z referenční relace R, pro tid se skóre >= c – korekční člen
10. Pomocí f porovnání každého takového referenčního záznamu se vstupním záznamem u
11. Získání sady nejvýše K referenčních záznamů s podobností nad w(u)*c.

Další optimalizaci použití ETI **[1]** shledává v použití tzv. optimistického zkratování (angl. optimistic short circuiting), které radikálně snižuje počet lookupů do tabulky ETI tím, že zohledňuje váhy jednotlivých tokenů. Váha jednotlivým min-hash signaturám je určena jako podíl váhy tokenu a délky příslušné q-tice. Tato váha je pochopitelně shodná v rámci všech min-hash signatur pocházejících z jednoho tokenu.

Rozšíření tématu

Jiný příklad doménově nezávislého řešení uvádí např. **[22]**. Jedná se o řešení problému eliminace fuzzy duplicit v rámci dimenzionálních tabulek datového skladu. Tedy o retrospektivní řešení problémů, které již měly být vyřešeny např. v rámci CDI na úrovni Customer Hubu. Historicky se pro tento účel používaly buď shodné techniky založené na míře vzdálenosti, jaké byly prezentovány v této práci výše (např. ed, Kosínový koeficient, apod.), a nebo doménově specifická pravidla (např. pro deduplikaci adres v nástroji firmy Trillium). Vzhledem k chybovosti některých popsaných metrik a snaze o vývoj doménově nezávislého řešení, **[22]** nabízí vlastní algoritmus DELPHI (Duplicate ELimination in the Presence of HIerarchies). Toto řešení využívá hierarchické struktury některých dimenzí a nabízí efektivní grupovací strategii, díky níž jsou v každé relaci porovnávány záznamy pouze s malou skupinou referenčních záznamů. Např. pokud uvažujeme hierarchii Název firmy ➔ Město ➔ Stát ➔ Země[15], porovnává např. pouze záznamy v dimenzi Stát, pokud odkazují na stejnou Zemi, resp. dva záznamy pro různé Země odkazující pouze na jeden Stát.

Závěr

V dnešní době již není zcela pravdivé tvrzení publikované v **[7]**, že komerční databázové systémy samy nepodporují tzv. přibližné spojování řetězců (v podstatě další synonymum pro Fuzzy Match) a že je tedy nutné tuto problematiku řešit pomocí uživatelem vytvořených funkcí nebo koupí specializovaného software (s problémy spojenými s nutnou integrací s DB). Použití různých metod pro spojování záznamů v komerčních systémech znázorňuje následující tabulka.

Commercial System	**Record Linkage Methodology**
SQL Server Integration Services 2005	Fuzzy Lookup; Fuzzy Grouping; uses Error Tolerant Index
OracleBI Warehouse Builder 10gR2 "Paris"	match-merge rules; deterministic and probabilistic matching
IBM's Entity Analytic Solutions, QualityStage	probabilistic matching (information content); multi-pass blocking; rules-based merging

Zdroj: **[17]**

Při aplikaci Fuzzy Match je ale vždy nutné pamatovat, že spolehlivost této metody dosud není přes všemožné snahy výzkumníků vždy 100%.

V oblasti aplikace Fuzzy Match existují stále určité výzvy, kterým bych se rád dále věnoval ve své disertační práci :

1. **[17]** upozorňuje na absenci srovnávacích studií jednotlivých algoritmů z hlediska jejich kvality. Vyzývá k vývoji standardních benchmarků v této oblasti.
2. Vývoj modifikace algoritmu SOUNDEX použitelný pro shodné zakódování podobně znějících slov v českém prostředí s důsledným vymezením případů, kdy tento algoritmus lze aplikovat.
3. Další výzvou je vyšší míra automatizace aplikace algoritmů, bez nutnosti občasného lidského rozhodnutí . Např. při možné shodné hodnotě funkce podobnosti pro více referenčních záznamů.

Reference

[1] Chaudhuri S., Ganjam K., Ganti V., Motwani R.: Robust and Efficient Fuzzy Match for Online Data Cleaning, SIGMOD 2003, June 9-12, 2003 San Diego, CA.

[2] Wang, R,Y., Ziad, M., Lee, Y.,W.: Data Quality, The Kluwer International Series on Advances in Database Systems Volume 23, Springer, Berlin, 2001.

[3] Král J., Žemlička M.: Kvalita dat a informací – základní omezení IT ve veřejné správě, Systems Integration 2006, str. 215 – 222.

[4] Merriam-Webster Online Dictionary, výklad pojmu fuzzy: http://www.m-w.com/dictionary/fuzzy.

[15] Má smysl pro USA, Kanadu, UK. Pro valnou většinu zemí EU by se spíše hodila hierarchie Název ➔ Město ➔ Region ➔ Stát.

[5] Rubens N. O.: The Applicationn of Fuzzy Logic to The Construction of The Ranking Function of Information Retrieval Systems, Computer Modelling and New Technologies, 2006, Vol. 10., No. 1, str. 20-27.

[6] Novák V.: Základy fuzzy modelování, BEN – technická literatura, Praha 2000, ISBN 80-7300-009-1.

[7] Gravano L., Ipeirotis P. G., Jagadish H. V., Koudas N., Muthukrishnan S., Srivastava D.: Approximate string joins in a database (almost) for free. In Proceedings of VLDB, Roma, Italy. September 11-14 2001.

[8] Navarro G., Baeza-Yates R., Sutinen E., Tarhio J.: Indexing methods for approximate string matching. IEEE Data Engineering Bulletin, 24(4):19—24,2001.

[9] Dyché J., Levy E: Customer Data Integration – Reaching a Single Version of Truth, John Wiley & Sons, Inc., 2006, ISBN 0-471-91697-8.

[10] Eckerson W. W.: Data Warehousing Special Report: Data quality and the bottom line, 2002, http://www.adtmag.com/print.aspx?id=6321

[11] Dorr B., Herbert P.: Data Profiling: Designing the Blueprint for Improved Data Quality, SUGI 30, Data Warehousing, Management and Quality, Paper 102-30, 2005.

[12] English L.: Improving Data Warehouse and Business Information Quality – Methods for Reducing Costs and Increasing Profits, John Wiley & Sons, 1999, ISBN: 0-471-25383-9.

[13] Patridge Ch.: The Fuzzy Feeling SAS Provides: Electronic Matching of Records without Common Keys, Observations – the technical journal for SAS Software Users, 1998, SAS Institute Inc., Cary.

[14] Wikipedia – The free encyclopedia: http://www.wikipedia.com

[15] SAS Institute Inc. 2004. SAS OnlineDoc® 9.1.3. Cary, NC: SAS Institute Inc.

[16] Search Engine Dictionary – A Complete Guide to Search Engine Terminology, výklad pojmu fuzzy matching (29.12.2007): http://www.serarchenginedictionary.com/terms-fuzzy-matching.shtml

[17] Srivastava D.: Record Linkage: A Database Approach, AT&T Labs-Research: http://www.research.att.com/~divesh/

[18] http://www.levenshtein.net/

[19] Broder A.: On the resemblance and containment of documents. In Compression and Complexity of Sequences (SEQUENCES ,97), 1998.

[20] Cohen E.: Size estimation framework with applications to transitive closure and reachability. Journal of Computer and System Sciences, 1997.

[21] Cohen W.: Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity. In Proceedings of ACM SIGMOD, Seattle, WA, June 1998.

[22] Ananthakrishna R., Chaudhuri S., Ganti V.: Eliminating fuzzy duplicates in data warehouses. In Proceedings of VLDB, Hong Kong, 2002.

[23] Ciaccia P., Patella M., Zezula P.: M-Tree: An efficient access method for similarity search in metric spaces. VLDB 1997.

[24] Winkler W.: The state of record linkage and current research. *Statistics of Income Division, Internal Revenue Service Publication R99/04* http://www.census.gov/srd/papers/pdf/rr99-04.pdf.

[25] Winkler W. E., Thibaudeau Y.: An Application of the Fellegi-Sunter Model of Record Linkage to the 1990 U. S. Decennial Census, U. S. Burreau fo Census, 1991.

[26] Baillie M., Azzopardi L, Crestani F.: Towards Better Measures: Evaluation of Estimated Resource Description Quality For Distributed IR.