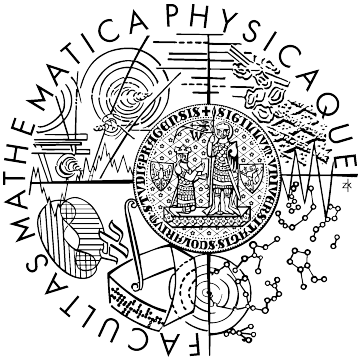
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Petr Vávra

Deduplikační metody v databázích

Katedra softwarového inženýrství

Vedoucí diplomové práce: Ing. Vladimír Kyjonka

Studijní program: Informatika, Softwarové systémy

2010

Děkuji Vladimíru Kyjonkovi za ochotu a vynaložený čas při vedení mé diplomové práce. Rovněž děkuji Ondřeji Rottovi za přínosné konzultace a společnosti SAS Institute ČR za poskytnutý software.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 28. 7. 2010 Petr Vávra

Obsah

1 Úvod a motivace **6**

2 Datové sklady, business intelligence a datová kvalita v souvislostech **8**
2.1 Co to je datová kvalita....8
2.2 Zasazení do souvislostí....8
2.3 Příčiny špatné kvality dat....9
2.4 Důsledek špatné kvality dat....11

3 Unifikace datových záznamů **12**
3.1 Čištění dat....12
3.2 Standardizace....13
3.3 Ohodnocení čistícím skórem....13
3.4 Proces unifikace....13
3.4.1 Tranzitivní unifikace....15
3.4.2 Unifikace při vstupu do systému....16
3.4.3 Unifikace s výjimkami....16
3.4.4 Identifikace domácností....16
3.5 Deduplikace....16
3.6 Ohodnocení výsledků unifikace....17

4 Deduplikační metody **18**
4.1 Metody slučování jednoatributových záznamů....18
4.1.1 Editační vzdálenost....18
4.1.2 Metrika Affine gap....19
4.1.3 Metrika Smith-Waterman....19
4.1.4 Metrika Jaro....19
4.1.5 N-gramy....20
4.1.6 Atomické řetězce....20
4.1.7 WHIRL....20
4.1.8 N-gramy a tf-idf....21
4.1.9 Soundex....21
4.1.10 Shrnutí metrik....21
4.2 Metody slučování víceatributových záznamů....22
4.2.1 Pravděpodobnostní metody....22
4.2.2 Metody učení s učitelem....23
4.2.3 Metody aktivního učení....24
4.2.4 Metody učení bez učitele....24
4.2.5 Vzdálenostní metody....25
4.2.6 Metody založené na pravidlech....26

4.3 Kombinování metod....27

5 Robustní deduplikační metoda 28
5.1 Principy metody....28
5.2 Implementace....31
5.2.1 Fáze 1: spočítání vzdáleností mezi záznamy....31
5.2.2 Fáze 2: vytvoření seznamu nejbližších sousedů....33
5.2.3 Fáze 3: nalezení kompaktních množin....34
5.2.4 Fáze 4: nalezení duplicit....35
5.3 Shrnutí robustní deduplikační metody....37

6 Porovnání deduplikačních metod 39
6.1 Přehled výhod a nevýhod....40
6.1.1 Pravděpodobnostní metody....40
6.1.2 Metody s učením....41
6.1.3 Metody založené na pravidlech....42
6.1.4 Vzdálenostní metody....42
6.2 Srovnávací metoda pomocí porovnání množin....43
6.2.1 Specifikace metody Compare....43
6.2.2 Výstup metody Compare....44
6.2.3 Index pokrytí....46
6.3 Srovnání metody DataFlux a Robustní deduplikační metody....47
6.3.1 Metoda DataFlux....48
6.3.2 Srovnání....48

7 Závěr 54

Název práce: Deduplikační metody v databázích
Autor: Petr Vávra
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí diplomové práce: Ing. Vladimír Kyjonka
E-mail vedoucího: kyjonka@mff.cuni.cz

Abstrakt: V této práci studujeme úlohy odhalování duplicit v databázích v rámci datové kvality. Za duplicity považujeme ty záznamy, které se sice mohou syntakticky lišit, ale které sémanticky představují tentýž objekt reálného světa. Hlavním cílem této práce je shrnout současné deduplikační metody z hlediska jejich nároků, výsledků a využitelnosti v praxi. Detailněji se zaměříme na porovnání dvou kategorií deduplikačních metod – těch, které vyžadují detailní informace o doméně, a těch, které se bez nich naopak dokáží obejít. Praktickou částí této práce je proto implementace vlastní metody z rodiny vzdálenostních metod nevyžadující žádné znalosti, jejíž výsledky porovnáme s výsledky komerčního nástroje používaného v praxi, který naopak využívá detailních znalostí dat, ve kterých jsou hledány duplicity.

Klíčová slova: unifikace, deduplikace, hledání duplicit, datová kvalita

Title: Deduplication methods in databases
Author: Petr Vávra
Department: Department of Software Engineering
Supervisor: Ing. Vladimír Kyjonka
Supervisor's e-mail address: kyjonka@mff.cuni.cz

Abstract: In the present work we study the record deduplication problem as an issue of data quality. We define duplicates as records having different syntax and the same semantics and which are representing the same real-world entity. The main goal of this work is to provide the overview of existing deduplication methods according to their requirements, results and usability. We focus on the comparison of two groups of record deduplication methods – with and without the domain knowledge. Therefore, the second part of this work is dedicated to the implementation of our method which does not utilize any domain knowledge and compare its results with the results of commercial tool deeply utilizing the domain knowledge.

Keywords: deduplication, record linkage, instance identification, data quality

1 Úvod a motivace

Databáze hrají velmi důležitou roli v dnešním světě, který se stává čím dál tím více závislý na informačních technologiích. Zájem firem o podrobnou analýzu firemních dat roste závratným tempem. Kvalitní a podrobné informace umožňují snižovat ztráty, zvyšovat ziskovost, analyzovat zákazníky a produkty, provádět cílený marketing nebo prostě jen zefektivnit běžný provoz společnosti.

Přitom se ale informační systémy větších společností čím dál tím více rozrůstají. Skládají se z bezpočtu databází, mezi kterými se data neustále přesouvají. Vstupní data pocházejí z různých zdrojů, kde jsou uložena v různých formátech. Mezitím se data průběžně mění stejně tak, jako se mění samotné databáze a aplikace, které s těmito daty pracují. Výsledkem toho je, že informační systémy se stávají čím dál tím lepšími, ale naproti tomu kvalita dat se stále více zhoršuje. Přitom právě kvalita dat je hlavním faktorem ovlivňující prospěšnost daných informačních systémů jako nástrojů pro shromažďování, zpracování a prezentaci informací.

Datová kvalita je ovlivněna nedostačující kontrolou vstupních i stávajících dat v databázi, tak i rozdílnou nebo nedostačující správou dat na fyzické i logické úrovni v rámci jedné nebo i více databází. Důsledkem toho je, že se v databázích vyskytují překlepy či úplně chybné údaje (*Univrzita Karolva* místo *Univerzita Karlova*), chybí integritní omezení (atributu *Věk zaměstnance* je umožněno nastavit hodnotu 547) nebo existuje více formátů (*Ul. Malinová 1b* oproti *1/213 Malinová ulice*). K této tzv. lexikální rozdílnosti v datech se přidává i rozdílnost v samotné struktuře databází. Např. adresní údaje mohou být uloženy v jednom atributu jako dlouhý řetězec, a v jiné databázi ve více atributech (*ulice, čp, město, psč* apod.).

V naší práci se zaměříme na rozdílnost lexikální a s tím související problematiku hledání duplicit. Za duplicitní budeme považovat takové záznamy, které se sice mohou syntakticky lišit, avšak sémanticky reprezentují tentýž objekt reálného světa. Tento problém byl definován již před několika dekádami autorem Newcombe a kol. [9] a později formalizován I. P. Fellegim a A. B. Sunterem [29]. Od té doby vzniklo nespočet různých metod snažící se řešit tento problém.

Cílem naší práce je zmapovat tato řešení a provézt jejich porovnání z hlediska nároků, výsledků a využitelnosti v praxi. Přitom se detailněji zaměříme na srovnání dvou deduplikačních metod: metody používané v nástroji DataFlux od společnosti SAS a námi implementované metody z kategorie vzdálenostních metod.

Nástroj DataFlux velmi efektivně využívá detailních znalostí domény včetně využití číselníků, databáze překlepů atd., zatímco naše metoda spoléhá čistě na syntaktickou podobnost záznamů bez využití jakýchkoli znalostí o datech a chybách v nich obsažených. Pokusíme se proto zjistit, zda i tato metoda může dosahovat stejně dobrých výsledků. Cílem této práce však není vyvinutí deduplikačního nástroje, který by byl okamžitě využitelný v praxi. Spíše se zaměříme na porovnání výsledků obou metod, jejich výhod a nevýhod.

Struktura této práce je následující. V příští kapitole se budeme zabývat pojmy, jako jsou datové sklady, business intelligence a jejich zasazením do souvislostí v rámci datové kvality. Ve třetí kapitole zformalizujeme problematiku unifikace datových záznamů. Ve čtvrté kapitole shrneme hlavní skupiny současných deduplikačních metod. Praktickou část naší práce popisující specifika námi implementované metody představíme v kapitole páté. V šesté kapitole porovnáme jednotlivé metody a zdůrazníme jejich přednosti a nároky pro využití v praxi. Zároveň zde představíme námi navržený algoritmus pro porovnání výsledků dvou deduplikačních metod, s jehož pomocí popíšeme rozdíly mezi metodou naší a metodou DataFluxu. Konečně v sedmé kapitole shrneme naše poznatky a výsledky.

2 Datové sklady, business intelligence a datová kvalita v souvislostech

Datová kvalita už svým názvem napovídá, že je jedním z hlavních oborů Informatiky. Ještě na počátku devadesátých let tomu tak však nebylo. V odborných kruzích se tímto tématem nikdo nezabýval. Povědomí o tomto oboru bylo všeobecně malé a pokud vůbec se nějaká firma zabývala kvalitou svých dat, málokdy se tím navenek chlubila. Přitom špatná kvalita dat může stát firmu nemalou část jejího zisku.

V této kapitole nejprve zavedeme pojem datové kvality (dále jen DQ) a následně se krátce zmíníme o tom, jak DQ souvisí s primárními systémy, datovými sklady a pojmem business intelligence. Na konci této kapitoly se budeme zabývat příčinami špatné kvality dat, protože jedním z jejích důsledků je právě vznik duplicit a nutnost jejich odhalování.

2.1 Co to je datová kvalita

Datová kvalita je pojem popisující míru korektnosti daných dat. Korektností je v tomto případě myšleno to, jak dobře daná data reprezentují objekty reálného světa. Datová kvalita je tedy tím větší čím lépe data popisují objekty reálného světa a zároveň pokud vyhovují operacím, které se s nimi provádí.

2.2 Zasazení do souvislostí

Životní cyklus informací může začínat u desktopových či webových aplikací, které tvoří rozhraní pro tzv. primární či provozní systémy. Do těchto systémů vstupují data důležitá pro každodenní chod společnosti. Tato data jsou detailní, aplikačně orientovaná a většinou transakčně zpracovávaná.

Pro účely získávání dalších informací, které primární systémy neposkytují, vznikají datové sklady, které mají za úkol doslova vytěžit informační bohatství. Oproti primárním systémům jsou zde data většinou agregovaná a subjektově orientovaná. Změny a mazání dat v nich nemá příliš velký význam. Ba právě naopak, znalost historie dat je jedním z hlavních cílů datových skladů. Datový sklad může být tedy chá-

pán jako další firemní systém, který je zaměřený na analýzu dat a který slouží potřebám manažerů především pro podporu jejich rozhodování.

Business intelligence je pojem završující úsilí datových skladů. Zahrnuje jak podnikové procesy a znalosti, tak i analýzu informací, jež bývají poskytovány právě datovým skladem. Business intelligence poskytuje ucelený pohled na historické i současné obchodní informace. Může však také dávat i předpovědi různých ekonomických aspektů. Základem ale je, že tyto informace jsou srozumitelné a snadno použitelné. Jako disciplíny spadající pod business intelligence jmenujme např. OLAP (Online Analytical Processing) [41], CRM (Customer relationship management) [55], Risk Management [42], Fraud Detection [55] aj.

A jak s tím souvisí DQ? DQ zde představuje jakýsi pilíř podpírající jak primární systémy, tak datové sklady a samozřejmě i business intelligence. V primárních systémech je nutné, aby data správně popisovala objekty reálného světa. Vznikají-li zde nějaké nepřesnosti a nekonzistence, přestávají data vyhovovat požadavkům systémů a potažmo i uživatelům. V datových skladech je snaha co nejvíce omezit jakékoli chyby a nepřesnosti, protože i drobné chyby se mohou během agregací projevit ve velké míře a i následné analýzy pro účely business inteligence mohou být velmi zkreslené. Proto se DQ řeší na úrovni primárních systémů ať už preventivními omezeními a pravidly nebo jednorázovým čištěním. To se může také provádět během procesu ETL (Extract, Transform, Load) [3], při vstupu dat do datových skladů.

2.3 Příčiny špatné kvality dat

V této sekci popíšeme hlavní příčiny vzniku chyb v datech, které ovlivňují datovou kvalitu. Patří sem konverze systémů; ruční zadávání dat; dávková zpracování; real-time rozhraní; změny, prořezávání a stárnutí dat aj.

Konverze systémů

Při vytváření nových systémů málokdy vznikají prázdné databáze. Většinou jsou hned při vzniku naplněny daty z jiných starších systémů. Je ale nutné zmínit, že informace, které mají uživatelé k dispozici, jsou výsledkem zpracování dat aplikační vrstvou nad samotnou databází. Při konsolidaci takových dat je tedy nutné přihlédnout i k dané aplikační vrstvě stejně tak jako k celkovým business procesům firmy.

Dalším problémem také je, že do nových databází se stará data často agregují, a tak i několik malých chyb může způsobit velké nepřesnosti, které je následně velmi obtížné vystopovat. Tomu všemu negativně napomáhá i špatný stav metadat[1], která jsou v naprosté většině případů nekompletní nebo jsou zastaralá či dokonce chybná.

1 Metadata jsou tzv. data o datech. Vypovídají, jaká je jejich struktura, význam, původ apod.

Ruční zadávání dat
Velké množství dat je zadáváno do databáze lidmi pomocí formulářů desktopových či webových aplikací. Základním problémem pro kvalitu dat jsou samozřejmě překlepy, ale chyby vznikají i zadáváním údajů do nesprávných políček nebo tím, že uživatel neví, co má do políčka zadat případně v jakém formátu. Do toho přistupuje i snaha vyplnit formulář co nejrychleji a i když aplikace nedovolí některá políčka vynechat, uživatel mnohdy raději vyplní nesmyslná data, jen aby měl vyplňování co nejrychleji za sebou.

Dávková zpracování
Uvnitř firmy či mezi dvěma firmami často dochází k pravidelným dávkovým přesunům dat z jedné databáze do druhé. Problém vzniká, pokud se primární databáze mění, což se děje neustále a to jak architekturou tak obsahem. Cílová databáze pak přijímaná data buď nezpracuje vůbec nebo špatně. Navíc není pravidlem, že by cílová databáze příchozí data jen uložila a nic dál s nimi nedělala. Většinou se s těmito daty provádí okamžitě další operace a tím se i sebemenší chyba z primární databáze propaguje dále a páchá ještě větší škody.

Real-time rozhraní
V zájmu zachování databází v co nejaktuálnějším stavu je snaha přenášet data mezi nimi co nejrychleji. Vložení záznamu do první databáze spustí vkládání do druhé atd. Na úkor toho je ale opomíjena kvalita dat. Pokud se však už zjistí nějaký problém, data musí být opravena automaticky a nebo je jejich vkládání pozastaveno, což ale může vést k tomu, že už nebudou vložena nikdy.

Změny dat
Ke změnám dat dochází v databázi běžně a je samozřejmé, že i při tom budou v datech vznikat chyby ať už jsou to běžné každodenní operace nebo velká úprava týkající se miliónů záznamů prováděná např. vždy na konci měsíce. Někdy dojde i ke změně struktury databáze nebo i aplikace pracující s daty. Neopatrnost při těchto úpravách pak může vést k velkým ztrátám či nepřesnostem.

Prořezávání dat
Stará či nepotřebná data se mohou občas z databáze automaticky či ručně odstraňovat. Problém je v tom, že takové prořezávání může zanechat databázi v nekonzistentním stavu, a tak musí být prováděno velmi obezřetně. Nehledě na to, že chyby v datech mohou vést k tomu, že byla prořezána data, která být prořezána neměla, či naopak.

Stárnutí dat
V databázích je uchovávána určitá statická podoba reálného světa, který se však neustále mění. Informace o těchto změnách se ne vždy dostanou do všech systémů,

ve kterých jsou vedeny. Tím se postupně data „kazí“, až se může stát, že naprosto neodpovídají skutečnosti. Záznamy jsou pak chybně interpretovány, zpracovávány, v databázích vznikají duplicity atd.

Ostatní
Jako další příčiny špatné kvality dat zmiňme úpravy systémů; nové, ale nevhodné využití stávajících dat; ztráta znalostí, metadat nebo automatizace zpracování dat.

2.4 Důsledek špatné kvality dat

Jedním z důsledků špatné kvality dat je výskyt duplicitních záznamů. Hlavními faktory ovlivňující jejich vznik jsou jak reálné chyby, tak i integrace více firemních systémů. V těchto aplikačně zaměřených systémech se např. mohou vyskytovat zákazníci využívající nějaký produkt. Pokud pak přihlédneme k rozdílným způsobům uložení, různým formátům a standardům dat podtrhnutých ne příliš dobrou kvalitou, je těžké odpovědět i na tak jednoduchý dotaz, kolik zákazníků vlastně firma má. Tyto i jiné duplicity je nutné rozpoznat a případně sloučit. Touto problematikou se budeme detailněji zabývat v následující kapitole.

3 Unifikace datových záznamů

Zřejmě proto, že se jedná o disciplínu poměrně úzce specializovanou, názvosloví v této oblasti není nikterak jednoznačně vyhraněné. Pojem unifikace je v různých zdrojích uváděn pod různým názvem. Pro přehled uveďme např. deduplikaci, matching, record linkage, instance identification nebo integration. V našem případě se pod pojmem unifikace rozumí rozpoznávání a slučování záznamů, které reprezentují tentýž objekt reálného světa. Takové záznamy nazýváme duplicitami.

V této kapitole se zaměříme na úlohy, které se mohou unifikace účastnit. Jsou jimi čištění dat, standardizace, ohodnocení čistícím skórem a deduplikace. Nakonec nastíníme možnosti ohodnocení výsledků unifikace.

3.1 Čištění dat

Existují dva možné přístupy k čištění dat. Prvním je ruční čištění. To je však činnost časově velmi náročná a navíc ji může provádět jen odborník, který rozumí daným datům a zná jejich vztahy jak mezi sebou, tak mezi aplikační či business logikou. Je proto nutné zavést automatizované čištění. To sice nemusí dosáhnout takových výsledků jako ruční čištění, zato je v praxi lépe použitelné. Navíc s přihlédnutím k tomu, jakými způsoby chyby v datech vznikají, může automatizované čištění odhalit určité vzory chyb a na ty se soustředit.

Průběh čištění
První úlohou čištění, která přichází hned po analýze vstupních dat, je parsing, který zjišťuje, jestli jednotlivé datové položky odpovídají dané syntaxi. Existuje několik přístupů k parsingu, pro ukázku uveďme Markov models for segmentation [4] nebo Automatic segmentation [5]. Nejprve je daná datová položka předzpracována, provede se rozřezání na slova a mohou být např. odstraněny nevhodné znaky. Například mezera v telefonním čísle. Dále se rozpoznávají jednotlivé tokeny[2] a rozřazují se do kategorií. K tomu parsing využívá kromě definic sloupců a jiných metadat i gramatiky, tj. znalosti toho, jak mohou být jednotlivé tokeny spojovány v celek.

Tím je otevřena cesta k tomu, aby byla slova vhodně opravena. Lze již použít příslušné slovníky, číselníky, seznamy překlepů apod. a podle nich data opravit.

2 Řetězec mající jednoznačný smysl definovaný gramatikou nebo regulárním výrazem. Může se skládat z jednoho, ale i více slov.

V případě jednoslovných datových položek se ke zjištění kategorie slova může využít regulárních výrazů[3]. Slovníky pocházejí jak z vlastních zdrojů, a to z nejčastějších údajů vyskytujících se v databázi, tak i z těch vnějších veřejně či komerčně dostupných.

3.2 Standardizace

Čištěním mohou být opraveny nejčastější syntaktické chyby. Následujícím krokem může být standardizace dat, neboli uvedení dat do jednotného tvaru pro účely dalšího využití. Dojde např. k rozbalení zkratek, nahrazení zastaralých nebo familierních názvů, seřazení tokenů do správného pořadí apod. Dále se mohou doplňovat chybějící údaje odvozením z těch známých. Např. doplnění čísla popisného podle ostatních položek adresy (s pomocí např. českého registru adres [48]).

3.3 Ohodnocení čistícím skórem

V dalším kroku se mohou opravené a standardizované záznamy označit čistícím skórem. To je tím vyšší, čím více editačních úprav bylo se záznamem provedeno a tím menší má tedy vypovídací hodnotu pro účely dalšího zpracování. Čistící skóre může být využito během úlohy deduplikace, kterou se ještě budeme zabývat v této kapitole.

Nakonec se celá čistící fáze vyhodnotí. Určí se její úspěšnost, data se znovu zanalyzují a vznikne hodnotící zpráva o provedeném čištění. Případně se upraví priority editačních pravidel, gramatik a regulárních výrazů a čištění se provede znovu. Výstupem také mohou být záznamy, u kterých se určila nutnost ruční opravy.

3.4 Proces unifikace

Proces unifikace by byl bezpochyby značně jednoduchý, pokud by se jednalo o duplicity triviální, tedy ty, které si jsou stoprocentně totožné. Jak ale vyplývá z předchozí kapitoly, není výjimkou, že se v datech mohou vyskytovat v hojné míře i záznamy, které jsou syntakticky odlišné, ale svou sémantikou jsou si velmi podobné.

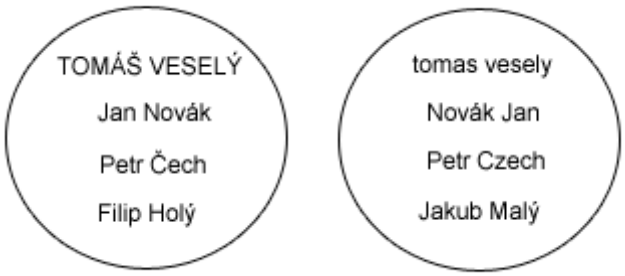
Takové záznamy by se daly nazvat synonymy a úkolem unifikace je jejich nalezení a rozřazení do skupin. Do těchto skupin, které budeme označovat jako clustery, jsou zařazena synonyma s jistou mírou pravděpodobnosti, neboli podle teorie fuzzy

3 Regulární výraz znázorňuje určitou masku nebo vzor pro řetězce nebo jejich části. Umožňuje tak jejich popis, kontrolu či vyhledávání v podřetězcích.

množin [40] s určitým stupněm příslušnosti. Pokud je tato míra dostatečná, lze daná synonyma prohlásit za duplicity.

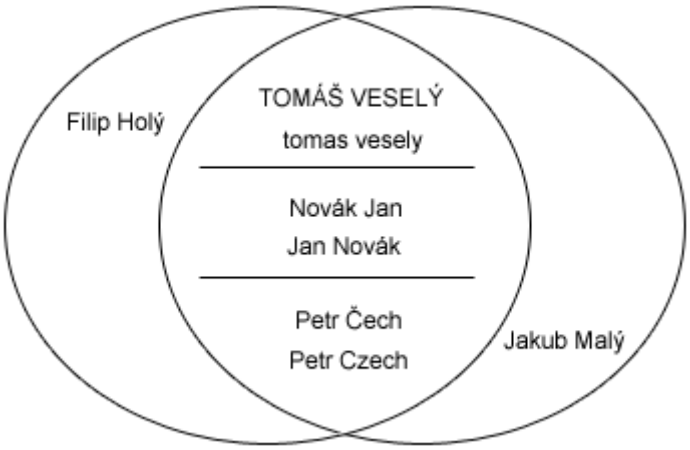
Jak unifikovat záznamy

Obrázek 3.1 znázorňuje dvě relace obsahující záznamy s rozdílnou syntaxí, ale v některých případech se stejnou sémantikou.



Obrázek 3.1: Dvě relace před sloučením

Po sjednocení těchto relací, které je zobrazeno na obrázku 3.2, stojí za povšimnutí jejich průnik, v němž se vyskytují dvojce synonym. Jejich nejednoznačnost je dána jak rozdílným formátem zápisu (např. „jméno příjmení“ vs. „příjmení jméno“), tak i šumem apod. V doplňcích tohoto průniku jsou potom záznamy unikátní.



Obrázek 3.2: Průnik dvou relací s vyznačením dvojic synonym

Způsobů, jak nalézt synonyma, je velká škála. Obecně lze však říci, že o rozhodnutí, zda a do jakého clusteru dané záznamy patří, rozhoduje jejich vzájemná podobnost. Jsou-li si dva záznamy velmi podobné, budeme říkat, že jsou od sebe málo vzdálené a naopak.

Výpočet vzájemné vzdálenosti dvou záznamů je určen podobností v hodnotách odpovídajících atributů. Právě proto může být velmi výhodné, před tímto krokem

provézt výše zmíněné čištění a standardizaci. Přitom je možné pro tento výpočet využít skupinu či skupiny jen některých ze všech atributů záznamu a úplně pominout některé nedůležité atributy. Mohou být například uvažovány tyto skupiny atributů (jméno, přijmení, rodné číslo) nebo (jméno, příjmení, ulice, město, psč) a podle podobnosti, či dokonce shodě v hodnotách těchto atributů je vypočítána výsledná vzdálenost (podobnost) dvou záznamů.

Existuje však více přístupů k této problematice. Některé z nich například vždy využívají všech atributů. Jiné zase mohou sloučit hodnoty všech atributů v jeden řetězec a počítat podobnost takto vzniklých řetězců. Konkrétní metody výpočtu vzdálenosti hodnot dvou atributů a následně i dvou záznamů v rámci unifikace shrneme v následující kapitole.

U této doposud popsané unifikace se může uplatnit několik zajímavých variant či rozšíření, které si nyní stručně popíšeme. Jedná se o tranzitivní unifikaci, unifikaci při vstupu do systému, unifikaci s výjimkami a identifikaci domácností.

3.4.1 Tranzitivní unifikace

Tranzitivní unifikace umožňuje vytvářet clustery duplicit na základě shody v hodnotách některých atributů a tyto clustery následně slučovat. Jedná se tedy o tranzitivní uzávěr relace, která udává, zda jsou dva prvky duplicitní. V tabulce 3.1 je znázorněna unifikace nejprve podle RČ a posléze podle jména a data narození. Tím vzniknou dva clustery (Id klient = 1 a Id klient = 2) Nakonec se odvodí výsledný cluster se třemi unifikovanými klienty.

Význam tranzitivní unifikace tkví v možnosti vytvářet skupiny duplicit, které by s využitím jednorázové unifikace podle velkého počtu atributů nemohly vzniknout, protože shoda v tolika atributech bývá výjimečná i u sémanticky stejných záznamů.

RČ	Jméno	Datum narození	Id klient	Id kandidát
5502101547			1 – 1	1
5502101547	Karel Novák	10.2.1955	1 – 2	1
	Karel Novák	10.2.1955	2 – 2	1

Tabulka 3.1: Ukázka hierarchické unifikace

Všimněme si, že princip slučování na základě části atributů přináší závažné důsledky. Pokud je použito mnoho atributů, nemusí dojít téměř k žádnému sloučení a záznamy budou považovány za unikátní, tzv. falešné nesloučení (false negative). Opačným případem je, pokud je použito nedostatečné množství atributů, které nemají dostatečnou rozlišovací sílu. Tím dojde k přemrštěnému sloučení záznamů a to i v případech, kdy by k němu dojít správně nemělo tzv. falešné sloučení (false positive). Tento problém může být mnohem závažnější než ten prve zmíněný, protože při odhalení

chyby je nutné použít historii sloučených záznamů a zpětně se i musí dohledávat původní vazby na další relace.

3.4.2 Unifikace při vstupu do systému

Specifickým případem unifikace je její varianta, kdy je potřeba zajistit unikátnost již jednou unifikované relace i v případě, kdy do ní vstupují nové záznamy. Při tom se navíc může stát, že vložení jednoho záznamu je vyhodnoceno jako vložení unikátního záznamu, ale další vložení může vyvolat sjednocení těchto dvou nových záznamů s některým již dříve se nacházejícím v dané relaci (například podle pravidel tranzitivní unifikace).

3.4.3 Unifikace s výjimkami

K tomuto specifickému případu unifikace dochází, pokud o slučování záznamů nerozhoduje výhradně vzájemná podobnost. Mezi tyto případy se dá zahrnout například ruční či automatické určení duplicit na základě vnějších explicitních pravidel (například sloučení poboček téže společnosti). Další významná skupina výjimek souvisí s historicky prováděnou unifikací, které je nutné zachovat.

3.4.4 Identifikace domácností

Identifikace domácností neboli householding je variantou unifikace, která je v poslední době čím dál tím více využívána v praxi. Tento proces vyžaduje skloubení unifikace klientů, adres a případně i dalších podpůrných informací. Nejedná se však o hledání duplicit tak, jak bylo výše popsáné, ale o nalezení osob, které sdílejí společnou domácnost nebo jiný objekt, což přináší mnohé informační výhody. Pro tyto účely bývá nutné rozšířit adresní záznamy např. o číslo bytu, podlaží apod. Další variantou je tzv. komerční householding, který identifikuje vztahy mezi firmami, případně mezi fyzickými a právnickými osobami.

3.5 Deduplikace

V momentě, kdy již jsou nalezeny skupiny (clustery) duplicit, může dojít k rozhodnutí, které z duplicitních záznamů mají být odstraněny, či sloučeny v jeden reprezentující záznam tak, aby vznikla nová relace pouze s unikátními záznamy. Tento proces se nazývá deduplikace. Není to však krok nutný. Někdy jsou nalezené duplicity jen označeny a je ponecháno na uživateli, jak s nimi naloží.

Uveďme tedy jen, že pro rozhodnutí, který záznam má být rozhodující pro vytvoření unikátního záznamu v rámci jednoho clusteru, existuje několik variant. Uveďme například:

- výběr kvalitativně nejlepšího záznamu. (Ten může být určen nejlepším čistícím skórem, viz 3.3 Ohodnocení čistícím skórem);

- výběr záznamu, jež nejlépe vyhovuje některým závazným číselníkům;
- výběr nejnovějšího nebo naopak nejstaršího záznamu (dle specifik daného systému);
- upřednostnění podle prioritního seznamu daného systému;
- kombinace předchozích aj.

Vyjasnění pojmů
Jak již bylo řečeno, pod pojmem *deduplikace záznamů* se rozumí odstraňování duplicit v rámci clusterů nalezených během unifikace. Přesto se deduplikací v praxi i v odborné literatuře většinou označuje právě samotný proces unifikace. Proto je tedy běžné, že se pojmy *deduplikace* nebo *deduplikační metody* zabývají hledání duplicit nikoli jejich odstraňováním. Abychom zachovali stejné názvosloví s odbornou literaturou zabývající se touto problematikou, budeme i my pro zbytek této práce používat termín deduplikace ve významu hledání duplicit.

3.6 Ohodnocení výsledků unifikace

Jak lze tušit z předchozího, úloha unifikace není jednoduchá. Navíc ohodnocení jejích výsledků nemusí být vždy jednoznačné. Například k případu, kdy dva záznamy nebyly označeny jako duplicitní, mohlo dojít ze dvou příčin. Tou první je, že se skutečně nejedná o duplicity, a proces unifikace se tedy zachoval správně. Tou druhou je, že sice o duplicity jde, ale jsou si natolik syntakticky odlišné, že sloučeny nebyly, a metoda se tedy zachovala špatně. To, o jaký případ se jedná, nelze vždy snadno určit.

Přesto, pokud to možné z větší části případů je, jsou pro ohodnocení výsledků unifikace použity některé metriky, jako jsou např.: přesnost (podíl správně nalezených duplicit a všech nalezených duplicit), úplnost (podíl nalezených duplicit a všech duplicit skutečně obsažených v relaci), již dříve zmíněný počet falešně sloučených a falešně nesloučených záznamů aj. Podrobně se budeme touto problematikou zabývat v šesté kapitole. Pro účely následujících dvou kapitol však bude postačovat, pokud pod pojmem dobrá či účinná metoda budeme rozumět metodu vykazující dobrých výsledků v těchto výše zmíněných metrikách.

V následující kapitole se budeme zabývat způsoby měření vzdálenosti mezi dvěma záznamy a popíšeme charakteristiky konkrétních deduplikačních metod.

4 Deduplikační metody

Jak vyplývá z předchozí kapitoly, deduplikační metody ke svému výpočtu vyžadují porovnání hodnot všech nebo jen některých atributů dvou záznamů. Na základě tohoto porovnání je možné určit vzdálenost dvou záznamů a rozhodnout, zda jde o duplicity. V této kapitole proto nejprve detailněji zmapujeme jedny z hlavních současných metod pro měření podobností dvou řetězců a následně se zaměříme na metody řešící slučování celých záznamů s více atributy.

4.1 Metody slučování jednoatributových záznamů

Nyní představíme metody, také nazývané metriky, vhodné pro zjištění vzdáleností dvou textových hodnot. Principem všech níže zmíněných metod je nalezení hodnoty, která by dostatečně vypovídala o vzájemné podobnosti dvou řetězců. Existují i metody, které se snaží řešit vzdálenost dvou číselných hodnot, nicméně tyto metody jsou poměrně na okraji zájmu, a tak se v nemalé míře k číselným hodnotám přistupuje jako k textovým řetězcům a jsou na ně použity příslušné řetězcové metriky. Konkrétně se budeme zabývat editační vzdáleností, metrikou Affine gap, Smith-Waterman, Jaro, dále pak metrikou s využitím N-gramů, atomických řetězců, metrikou WHIRL, TF-IDF a Soundex.

4.1.1 Editační vzdálenost

Editační vzdálenost mezi dvěma řetězci α_1 a α_2 je definována jako minimální počet editačních úprav nutných k tomu, aby se upravený řetězec α_2 shodoval s α_1. Existují tyto druhy editačních úprav:

- vložení nového znaku;
- smazání znaku v řetězci;
- náhrada znaku za jiný znak.

Tato verze editační vzdálenosti, kdy každá z uvedených operací má stejnou váhu, bývá označována také jako Levenshteinova vzdálenost [6]. Kromě této základní verze však existují i další modifikace. Needleman a Wunsch [8] například přišli s vylepšením, které každé editační úpravě přiřazuje jinou váhu. Zvýhodnit lze například případ záměny „O“ a „0“ oproti záměně „A“ a „M“ apod.

Editační vzdálenost je vhodná spíše pro jednoslovné hodnoty a pro drobné syntaktické odchylky, jako jsou překlepy. Ve víceslovných atributech, kde dochází ke změnám pořadí jednotlivých slov nebo některá slova přebývají, jsou použity zkratky, přípony, předpony apod., je editační vzdálenost velmi málo prospěšná.

4.1.2 Metrika Affine gap

Metrika Affine gap [54] je rozšířením editační vzdálenosti, ke které přidává nové editační operace. Těmi jsou otevření „trhliny“ a prodloužení „trhliny“. Touto „trhlinou“ se rozumí neshoda n-tého znaku v řetězci α_1 s n-tým znakem řetězce α_2. Přitom cena prodloužení „trhliny“ bývá mnohem menší než cena otevření. Tím je neshoda více znaků v řadě za sebou penalizována celkově méně než v případě prosté editační vzdálenosti. Díky tomu je například vzdálenost mezi řetězci „John R. R. Tolkien“ a „John Ronald Reuel Tolkien“ mnohem menší, než jakou by vypočítala editační vzdálenost.

4.1.3 Metrika Smith-Waterman

Jedná se o další rozšíření editační metriky, kterou navrhli T. F. Smith a M. S. Waterman [10]. Tato metrika má za cíl penalizovat neshody uvnitř řetězců více než na jejich okrajích. Tím si budou řetězce, lišící se jen v začátcích nebo koncích, předponách nebo příponách, vzdálenostně mnohem bližší. Příkladem může být „prof. Jiří Novák, Univerzita Karlova“ a „Jiří Novák, prof.“

Je zřejmé, že obě předchozí metriky mají snahu omezit nevýhody jednoduché verze editační vzdálenosti. Stojí však za povšimnutí, že i přes tuto snahu není ani jedna z těchto metrik dokonalá. Na zmíněných příkladech je vidět, že to, co ohodnotí metrika Affine gap jako řetězce velmi si blízké, může být metrikou Smith-Waterman ohodnoceno jako velmi si vzdálené a naopak.

4.1.4 Metrika Jaro

Tato metrika [11] je vhodná především na měření vzdáleností mezi řetězci s osobními jmény. Výpočet probíhá v těchto krocích:

- Spočítá se délka obou vstupních řetězců $|\alpha_1|$ a $|\alpha_2|$.
- Nalezne se množina C všech „společných znaků“ takových, že $\alpha_1[i] = \alpha_2[j]$ a $|i - j| \leq ½ \min\{|\alpha_1|,|\alpha_2|\}$
- Spočítá se počet transpozic t. Transpozice je případ, kdy pozice společného znaku v řetězci α_1 se neshoduje s pozicí v řetězci α_2.

$$\text{Výsledná hodnota je: } \text{Jaro}(\alpha_1, \alpha_2) = \frac{1}{3}\left(\frac{|C|}{|\alpha_1|} + \frac{|C|}{|\alpha_2|} + \frac{|C| - t/2}{|C|}\right)$$

W. E. Winkler a Y. Thibaudeau [12] přidali k této metodě penalizaci v případě neshody znaků na začátku řetězce, protože právě rozdílnost jmen a příjmení v prvních písmenech je mnohem více významná, než uprostřed či na konci.

4.1.5 N-gramy

N-gramy (někdy nazývané q-gramy) řetězce α jsou všechny jeho podřetězce délky n. Metoda porovnání dvou řetězců pomocí n-gramů [13] stojí na předpokladu, že dva velmi si blízké řetězce budou sdílet velké množství n-gramů. N-gramy jsou vytvořeny pomocí okénka velikosti n, které se posouvá po řetězci zleva doprava hned od prvního znaku. Přitom lze první a poslední n-gramy, které ještě neobsahují n znaků z řetězce α doplnit pomocí až n-1 speciálních symbolů, které se nevyskytují v původní abecedě dané domény. Existuje mnoho rozšíření této metriky. Uveďme například Gravano a kol. [14], kde je spolu s n-gramy využito i informace o jejich pozici. Ta je následně zohledněna při výpočtu překrývajících se n-gramů obou řetězců.

4.1.6 Atomické řetězce

Metrika založená na atomických řetězcích byla navržena autory A. E. Monge a C. P. Elkan [15]. Atomický řetězec je souvislý podřetězec alfanumerických znaků ohraničený mezerou nebo jiným oddělujícím znakem. Zjednodušeně řečeno se tedy jedná o slova. Slova řetězce α_1 jsou spárována se slovy řetězce α_2, pokud se rovnají, nebo pokud je jedno slovo prefixem (předponou) druhého. Výsledná metrika je poměr takto spárovaných dvojic atomických řetězců ku průměrnému počtu atomických řetězců v α_1 a α_2.

4.1.7 WHIRL

Tato metrika [16], založená na kosinové vzdálenosti [7], je dalším představitelem metriky využívající místo znaků celá slova. Každému slovu w je přiřazena jeho váha podle tf-idf schématu:

$$v_\alpha(w) = \log(tf_w + 1) \cdot \log(idf_w),$$

kde tf_w je počet výskytů slova *w* v řetězci α a idf_w je $\frac{|D|}{n_w}$, kde n_w je počet záznamů v databázi D, které obsahují *w*. Všimněme si, že tato váha je vysoká, pokud se toto slovo v daném řetězci vyskytuje často a zároveň zřídka v rámci celé kolekce záznamů. Tím se vyloučí často se opakující slova, která nemají velký rozlišovací význam (spojky, předložky apod.) Na poli unifikace názvů firem by např. slovům jako „s.r.o.“ nebo „a.s.“ byla obdobně přiřazena velmi malá váha. Také si povšimněme, že tato váha nikterak nezohledňuje pozici slova v řetězci, což také může být velkou předností této metriky. Výsledná podobnost se nechá spočítat jako:

$$sim(\alpha_1, \alpha_2) = \frac{\sum_{j=1}^{|D|} v_{\alpha_1}(j)\, v_{\alpha_2}(j)}{\|v_{\alpha_1}\| \cdot \|v_{\alpha_2}\|}$$

Tato metrika je vhodná pro řetězce obsahující více slov, které se mohou umisťovat na různých pozicích. Zároveň výskyt vysoce frekventovaných (a tudíž malou vahou ohodnocených) slov tolik neovlivní výslednou podobnost. Např. řetězce „Mr. John Smith“ a „Smith John“ budou mít podobnost velmi se blížící jedné. Nevýhodou této metriky je malá odolnost vůči šumu, kde i malá syntaktická odchylka způsobí, že výsledná podobnost bude blízká nule. Jako příklad může sloužit „Univrzita Karlova v Praze“ a „Univerzita Kalrova v Praze“. Tento problém se snaží řešit M. Bilenko a kol. svou metrikou SoftTF.IDF [17], která zohledňuje i páry slov, které nejsou úplně totožné. Jejich váha je ve výsledné podobnosti dvou záznamů vynásobena příslušnou hodnotou z intervalu [0,1], která se odvíjí od vzájemné podobnosti daných slov.

4.1.8 N-gramy a tf-idf

Toto zajímavé rozšíření předchozí metriky nastínil L. Gravano [18], kde místo slov jsou použity n-gramy. To má za důsledek, že drobné syntaktické rozdíly tolik neovlivní výslednou podobnost dvou záznamů, jelikož celkové množství n-gramů, které sdílejí, je vysoké. Taktéž je zřejmé, že tato metrika je imunní vůči změnám pořadí slov v řetězci. A konečně slova potažmo n-gramy často se vyskytující v databázi jsou ohodnocena nízkou vahou obdobně jako v předchozí metrice.

4.1.9 Soundex

Soundex [19] je metrika, která se úplně odlišuje od všech předchozích. Podobnost dvou řetězců totiž nevypočítává na základě porovnávání jednotlivých znaků nebo celých tokenů, ale všímá si fonetické podobnosti. Některá slova se totiž stejně nebo velmi podobně čtou navzdory tomu, že je jejich psaná podoba odlišná. Princip této metriky spočívá v tom, že slovům jsou přiřazeny alfanumerické kódy. První písmeno bývá bráno jako prefix, některá písmena jsou úplně zanedbána, většině je však přiřazen nějaký číselný kód. Např. písmena „V“a „B“ nebo „M“ a „N“ jsou označena stejným kódem. Nakonec jsou tedy místo konkrétních slov porovnávány tyto přiřazené kódy. Jde o velmi zajímavý přístup, který nezůstal ojedinělý a brzy vznikaly další verze a rozšíření této metody. Zmiňme například NYSIIS [20] nebo ONCA [21].

4.1.10 Shrnutí metrik

Na poli unifikace záznamů existuje nespočet variací různých metrik. Výše zmíněný přehled metrik tak jistě není úplný. Přesto zde byly zmíněny hlavní přístupy, se kterými je možné se setkat v praxi. Každá má své výhody a nevýhody. Editační metrika se hodí na krátké, nejlépe jednoslovné řetězce, zatímco tokenově zaměřené metody, např. WHIRL, si poradí i s víceslovnými záznamy, ovšem jen pokud se v nich nevyskytují syntaktické chyby. Je zřejmé, že neexistuje jediná absolutně nejlepší metrika, která by byla nejvhodnější pro všechny možné databáze. Je nutné vždy přihlédnout

ke znalostem o daných datech a použít tu nejpřínosnější metriku i s ohledem na to, jaká metoda bude použita pro unifikaci víceatributových záznamů. A právě o těchto metodách pojednává následující sekce.

4.2 Metody slučování víceatributových záznamů

Všechny metody zmíněné v předchozí sekci zjišťují podobnost pouze záznamů obsahující jeden atribut. V reálných situacích jsou však záznamy obsahující pouze jeden atribut spíše výjimkou. Proto je nutné řešit poněkud složitější problém, kterým je unifikace víceatributových záznamů.

Formální zápis tohoto problému může být následující. Mějme relace A a B, které mají n společných atributů. Každá dvojce záznamů $[a,b]$ ($a \in A$, $b \in B$) je přiřazena do jedné z tříd M nebo U. Třída M obsahuje ty páry záznamů, které jsou duplicitní, a třída U reprezentuje množinu těch párů, které jsou unikátní. Přitom každý pár záznamů $[a,b]$ lze reprezentovat porovnávacím vektorem $x = [x_1,..,x_n]^T$, kde n je počet atributů záznamů a x_i je míra podobnosti i-tých atributů záznamů a a b.

V následující části představíme základní modely deduplikace víceatributových záznamů. Jsou jimi pravděpodobností metody, metody učení s učitelem i bez učitele, metody aktivního učení, vzdáleností metody a metody založené na pravidlech.

4.2.1 Pravděpodobnostní metody

I. P. Fellegi a A. B. Sunter [29] formalizovali problém zařazení páru záznamů $[a,b]$ do třídy M nebo U pomocí jejich porovnávacího vektoru x. Předpokládá se, že hodnoty tohoto vektoru jsou náhodné a jejich rozložení je pro každou z tříd M a U jiné. Pravidlo rozhodující o tom, zda dvojce $[a,b]$ náleží do M nebo U lze napsat jako:

$$[a,b] \in M \quad \text{pokud } p(M|x) \geq p(U|x),$$
$$[a,b] \in U \quad \text{v ostatních případech}$$

Toto pravidlo zjednodušeně řečeno značí, že daná dvojce záznamů je duplicitní v případě, že pravděpodobnost zařazení vektoru x do třídy M je větší než pravděpodobnost zařazení do třídy U. Dle Bayesovy věty lze toto pravidlo přepsat na:

$$[a,b] \in M \quad \text{pokud } l(x) \geq \frac{p(U)}{p(M)}$$

kde $l(x) = \frac{p(x|M)}{p(x|U)}$ a $\frac{p(U)}{p(M)}$ slouží jako „práh sloučení“.

Otázkou nyní je, jak spočítat hodnoty $p(x|M)$ a $p(x|U)$. Metoda nazývaná Naivní Bayes předpokládá vzájemnou nezávislost náhodných proměnných x_i. Potom lze hodnoty $p(x|M)$ a $p(x|U)$ spočítat takto:

$$p(x|M) = \prod_{i=1}^{n} p(x_i|M)$$

$$p(x|U) = \prod_{i=1}^{n} p(x_i|U)$$

Hodnoty $p(x_i|M)$ a $p(x_i|U)$ mohou být spočítány pomocí ohodnoceného vzorku dat nebo, jak navrhl M. A. Jaro [30], pomocí EM (Expectation maximization) algoritmu, který v binárním modelu dokáže spočítat $p(x_i = 1|M)$. Pravděpodobnosti $p(x_i = 1|U)$ jsou spočítány podle náhodně vybraných záznamů, u kterých je zřejmé, že s velkou jistotou náleží do U.

V doposud zmíněných pravděpodobnostních metodách se vyskytuje podmínka vzájemné nezávislosti jednotlivých atributů. To je však v reálných datech velmi nepravděpodobná situace. Proto W. E. Winkler [31] přichází s obecným EM algoritmem pro výpočet hodnot $p(x|M)$. Jeho metoda se opírá o tyto předpoklady:

- Data obsahují poměrně velké množství duplicit (více než 5%).
- Duplicitní páry se výrazně liší od ostatních záznamů.
- Výskyt syntaktických chyb je malý.
- Záznamy obsahují více atributů, které pomáhají odstínit syntaktické odlišnosti v některých jiných atributech.

Jak tvrdí W.E Winkler [31] výše zmíněné podmínky lze velmi dobře uvolňovat a stále dochází k dobrým výsledkům deduplikace. Zajímavé však je, že výsledky metod předpokládající vzájemnou nezávislost atributů nejsou výrazně horší než výsledky EM algoritmu povolující vzájemné ovlivnění atributů.

4.2.2 Metody učení s učitelem

Druhou skupinou metod, která začala vznikat jako náhrada pravděpodobnostních metod, jsou metody učení s učitelem. Tyto metody předpokládají dostupnost trénovacích dat. Ta jsou v podobě dvojic záznamů [a,b] ručně označených, zda patří do třídy M – třídy unifikovaných dvojic, nebo do třídy U – třídy unikátních záznamů.

Konkrétních technik je velká škála. Např. metoda CART [22] nebo Algoritmus lineárních diskriminantů [23], který vytváří parametry a ty lineárně kombinuje pro zjištění výsledného zařazení do správné třídy. Dále jmenujme například metodu vektorové kvantizace, která je zobecněním metody nejbližších sousedů [24].

Metod zabývajících se tímto druhem unifikace je opravdu mnoho. Naprosté většině je však společné, že považují záznamy za vrcholy grafu a relace značící du-

plicitu tvoří v grafu hranu. Potom na konci výpočtu musí dojít ke tranzitivnímu uzávěru nalezených relací. Tj. například pokud jsou dvojce [*a*,*b*], [*a*,*c*] označeny jako duplicitní, bude v tomto posledním kroku jako duplicitní označena také dvojce [*a*,*c*]. Tento krok, ač bývá nutný, může přinášet problém. Tím je nekonzistentnost daného algoritmu v případě, že dvojce [*a*,*c*] má být zároveň označena jako dvojce unikátních záznamů.

Výsledkem metod by mělo být rozložení těchto grafů na podgrafy, které minimalizují počet takových nekonzistentních případů. Ačkoli se jedná o NP-úplný problém, existují algoritmy, které ho aproximují. Jsou jimi např. Correlation clustering [35] nebo Large high-dimensional data clustering [36].

Zástupcem metod, které nepoužívají záznamy jako vrcholy grafu, může být algoritmus autorů P. Sigla a P. Domingos [25]. Tento algoritmus naopak používá jako vrcholy samotné atributy záznamů, na čemž se ukazuje, že je možné „propagovat“ informace o jednotlivých atributech a vylepšit tím tak celý proces unifikace. Např. jsou-li sloučeny záznamy [10 Downing street, LA, California] a [10 Downing street, Los Angeles, California] může být pro další záznamy využito vztahu hodnot „LA“ a „Los Angeles“. Je však nutné dodat, že tato metoda předpokládá, že rozdílnost v syntaxi takovýchto případů není způsobena chybou, ale prostým využitím jiného pojmenování.

4.2.3 Metody aktivního učení

Variantou metod využívající trénování jsou tzv. metody aktivního učení. Jejich specifikem je, že nepotřebují tak velké množství trénovacích dat. Velké množství dat lze poměrně snadno rozdělit do skupiny zřejmých duplicit a zřejmých unikátních záznamů. K ohodnocení ostatních je částečně nutné rozhodnutí uživatele. Tyto metody aktivně vyberou některé z těchto těžko ohodnotitelných dat a nechají uživatele, aby rozhodl, zda jde o duplicity. Díky tomu tyto metody dokáží ohodnotit ostatní podobné případy. Typickým představitelem těchto metod je ALIAS [37], jehož hlavním cílem je zaměření se na to, jak vybrat ten nejlepší vzorek dat pro ohodnocení uživatelem, aby toto ohodnocení co nejvíce pomohlo k deduplikaci ostatních složitých případů.

4.2.4 Metody učení bez učitele

Další kategorií metod s učením jsou tzv. metody učení bez učitele. Ty se snaží úplně se vymanit ze závislosti na trénovacích datech a to pomocí předpokladu, že podobné porovnávací vektory patří do stejné třídy, podobně jako předpokládá rodina pravděpodobnostních metod. Viz sekce 4.2.1.

Do této kategorie se dá zařadit metoda Approximate record matching [34], která využívá učení založené na shlukování a cotrainingu [38]. Tato metoda přeci jenom využívá několik málo ohodnocených dat. Ta však postačují opravdu jen v malém množství. Jak již bylo zmíněno, využívá se toho, že porovnávací vektory s podobnou charakteristikou patří do stejných tříd. Tudíž i dvojice záznamů příslušného po-

rovnávacího vektoru patří do stejných tříd. Tím pádem stačí jen málo ohodnocených dat v každém shluku, aby bylo možné rozhodnout o všech jeho prvcích, zda jde o duplicity nebo ne.

4.2.5 Vzdálenostní metody

Vzdáleností metody jsou jednou ze skupin deduplikačních metod, které nepotřebují žádná testovací data. Jediné, co tyto metody potřebují, je definice nějaké metriky a jedna hraniční hodnota, která rozhodne, zda dvojici záznamů sloučit nebo ne.

Jedna z variant autorů A. E. Monge a C. P. Elkan [39] slučuje všechny atributy záznamu v jeden dlouhý řetězec, na který je použita vhodná řetězcová metrika. Taková metrika by měla být schopná vypořádat se s různým pořadím slov, s nadbytečnými slovy apod. W. W. Cohen [40] pro tento případ unifikace používá metriku „tf-idf“. Viz sekce 4.1.

Tyto metody, které slučují všechny atributy v jeden mají tu výhodu, že výpočet vzdálenosti je zjednodušen na výpočet podobnosti jediného pole, (jak je popsáno v sekci 4.1). Výhoda je také v tom, že tato metoda si poradí s prohozenými hodnotami zapsanými ve špatných polích. Na druhou stranu tyto metody přicházejí o podstatné informace, které vycházejí z podstaty atributů.

Metoda Entity matching in heterogeneous databases [26] již pracuje s každým atributem zvlášť. Atributům je přiřazena váha a podle podobnosti hodnot v jednotlivých atributech je vypočítána vážená podobnost (tedy vzdálenost) dvou záznamů. Zajímavé je, že tato metoda se z hlediska hledání vah atributů a celkového výpočtu velmi podobá pravděpodobnostním metodám popsaných v sekci 4.2.1.

Jiný způsob výpočtu představil S. Guha a kol. [27]. Princip jejich algoritmu spočívá v tom, že pro každý atribut lze snadno vytvořit seznam nejpodobnějších záznamů. Tento seznam je vytvořen pro každý z atributů a je seřazen od záznamů, co jsou k sobě nejblíž až k těm, co jsou si k sobě nejvzdálenější. Dále je nutné spárovat záznamy tak, aby celková cena párování byla co nejmenší. Přitom cena spárování se odvíjí od toho, na jaké pozici se záznamy v seznamech vyskytují. (Čím horší pozice, tím vyšší cena). K výpočtu minimalizace této ceny slouží Hungarian algorithm [32].

Většina doposud zmíněných vzdálenostních metod vždy vyžaduje nějakou hranici, se kterou jsou porovnány podobnosti dvou záznamů a podle které je rozhodnuto o tom, zda jde o duplicity nebo o záznamy unikátní. Tento práh může být zadán uživatelem nebo ho lze získat pomocí trénovacích dat. To by ale ubralo na kráse této skupině algoritmů, které jinak žádná trénovací data nepotřebují.

S. Chaudhuri a kol. přichází se zajímavou metodou Robust identification of fuzzy duplicates [28], která nepotřebuje ke svému výpočtu žádný takový práh. K rozhodnutí o tom, zda mají být dva záznamy sloučeny, totiž ani tak nepřihlíží k jejich vzájemné podobnosti, jako si spíše všímá okolí těchto dvou záznamů. Pro tyto účely jsou definovány dvě vlastnosti, které musí splňovat záznamy, aby mohly být sloučeny:

- „kompaktnost“: záznamy si musí být blízké, avšak ne z hlediska nějaké absolutní slučovací hranice, ale podle toho, že mají k sobě blíž než k jiným záznamům.
- „řídké okolí“: počet záznamů v blízkém okolí záznamů musí být malý.

S. Chaudhuri a kol. [28] ukazují, metoda s takto zvolenými vlastnostmi vykazuje lepší výsledky než běžné vzdálenostní metody spoléhající se na jednu konkrétní hranici slučování.

4.2.6 Metody založené na pravidlech

Snahou těchto metod je sloučit záznamy na základě stanovených pravidel. Tato pravidla říkají, zda sloučit nebo nesloučit záznamy například podle toho, jestli se rovnají v prvním a ve třetím atributu apod. Tyto metody tedy mohou být považovány za speciální případ vzdálenostních metod, kde vzdálenost záznamů je vždy 1 nebo 0. Velký rozdíl je však v tom, že tyto metody nutně vyžadují dokonalou znalost domény. To opět může být jak výhoda, tak nevýhoda. Např. vzdálenostní metoda (nevyužívající metriku tf-idf) si neporadí s případem, kdy je v záznamu prohozeno jméno a příjmení. Metoda založená na pravidlech si s tím snadno poradí přidáním dalšího pravidla. Ovšem, a to je ona nevýhoda, je nutné, aby expert vytvářející tato pravidla pokryl tento i všechny ostatní případy, které v reálných datech mohou nastávat.

Příklad pravidla pro slučování osob:

- Úplná shoda v rodném čísle, max. 1 rozdílný znak ve jménu (ne na začátku)
- Rozdílná příjmení
- Pohlaví = Žena

Všimněme si, že toto pravidlo bylo sestaveno pro případ, kde se žena vdá. Tím se v databázi mohou vyskytovat dva zdánlivě rozdílné záznamy, které ve skutečnosti zastupují jednu osobu a které by jiné metody jen těžko slučovaly. Podrobných závislostí z reálného světa je nepřeberné množství. Jednou z metod, která jich využívá, je [2], kde jsou dokonce začleněna pravidla jako např.: pokud je věk > 26, nejedná se o studenta.

Metoda AJAX [33] navrhuje systém s podporou deklarativních jazyků, které umožňují zapisování takových pravidel s podporou SQL a jednoduchých příkazů pro čistění dat.

V rámci pravidel je samozřejmě možné pro dvojici atributů použít i složitější metriku, než je „úplná shoda“. Metoda Learning based fusion [43] používá dvě sady pravidel. Nejprve tu, která je výpočetně jednodušší, a až v případě shody je použita druhá sada pravidel, která je detailnější a výpočetně složitější. Tím je vlastně docíleno jistého shlukování. Zároveň jsou pravidla druhé sady ohodnocena vahou, a tak o sloučení rozhoduje vážený součet splněných pravidel. Pro účely stanovení slučova-

cí hranice se v tomto případě přistupuje k natrénování metody na vzorku předem ohodnocených dat. Povšimněme si, že tato metoda je vlastně vzdálenostní metodou využívající pravidla a trénování. Jak je vidět, rozmanitosti metod nejsou kladeny žádné meze.

4.3 Kombinování metod

Motivací ke kombinování či slučování více metod je využití jejich výhod a případně potlačit některé jejich nevýhody. Např. již dříve zmíněná metoda Learning based fusion [43] využívá jak pravidel, tak vzdálenostních metrik a konečně i trénování k nastavení slučovací hranice. Tato metoda tak šikovně využívá výhod pravidel, které dokáží postihnout velkou škálu chyb způsobující duplicity. Tato pravidla ohodnocuje vahou a aby bylo dosaženo nejlepších výsledků na konkrétních datech bez nutnosti ručního nastavování slučovací hranice, je využito učení na vzorku trénovacích dat. Je samozřejmě pravdou, že trénování přináší i své nevýhody, ale takováto metoda jako celek může dosahovat lepších výsledků než obyčejná vzdálenostní metoda nebo metoda založená na pravidlech.

Jiným případem kombinování metod, ke kterému se poslední dobou směřuje, je využití trénovacích dat učícími metodami pro účely vytvoření slučovacích pravidel. Na základě vzorku ohodnocených dat tak vzniknou pravidla, která mohou být následně škálována a využita pro samotnou deduplikaci. Odpadá tím pracnost při vymýšlení pravidel expertem. Navíc je možné, že ani expert s dobrou znalostí domény by nenalezl všechna takto vytvořená pravidla.

Dále mezi kombinování metod můžeme zařadit i prosté použití více druhů metrik pro porovnávání dvou atributů, což bývá základem pro většinu deduplikačních metod. S využitím znalostí o datech, lze použít vhodnou metriku. Na krátké jednoslovné řetězce bude výhodnější jiná metrika než na dlouhé řetězce s více slovy, jiná bude použita na čísla apod. Pokud nemáme dobrou znalost dat, lze dokonce využít více metrik najednou a jako výsledek použít tu, která dává nejlepší výsledek.

V neposlední řadě se nabízí využití některých nástrojů, které zahrnují celou řadu metrik a deduplikačních metod. Výběr těchto metod je ponechán výhradně na uživateli, což není úplně přívětivé. Existuje tím ale možnost spuštění více metod najednou. Následně lze ty duplicity, které byly odhaleny všemi spuštěnými metodami, označit za jednoznačně pravdivé. Všechny ostatní mohou být ponechány k ohodnocení uživateli. Mezi tyto nástroje, obsahující jak prosté deduplikační metody tak přímo hybridní metody, patří TAILOR [51], FEBRL [52] nebo STEM [53].

5 Robustní deduplikační metoda

Nyní, když máme dostatečné teoretické znalosti z oblasti hledání duplicit, se můžeme posunout k praktické části této práce. Jedním z našich hlavních cílů je totiž srovnání dvou kategorií deduplikačních metod: metod založených na pravidlech a metod nevyžadující znalost domény.

Metody založené na pravidlech jsou velmi oblíbené v praxi pro svou jasnou a průhlednou interpretovatelnost a možnost postihnout velké množství příčin, kdy dva syntakticky odlišné záznamy reprezentují jeden objekt. Naproti tomu stojí jiné metody, které nevyužívají znalosti o datech. Například nedokáží jednoduše postihnout případy, kdy je jméno a příjmení osoby zadáno ve špatném atributu. Jejich nezvratnou předností je však snadná implementace a možnost rychlého nasazení. Ta je umocněna v případě, kdy není zapotřebí trénovacích dat.

Proto jsme zvolili algoritmus z rodiny vzdálenostních metod. Jde totiž o metody, které nevyužívají trénovací data, a tak je jejich nasazení velmi jednoduché a univerzální pro velké množství relací. Konkrétně byl implementován algoritmus z velké míry inspirovaný již dříve zmíněnou metodou Robust identification of fuzzy duplicates [28].

V této kapitole nejprve představíme princip této metody a následně ukážeme detailní specifika naší implementace.

5.1 Principy metody

Jako u všech vzdálenostních deduplikačních metod i v tomto případě je cílem rozpoznat duplicity na základě jejich vzájemné vzdálenosti, která je závislá na míře podobnosti příslušných dvojic atributů daných záznamů. Naše Robustní deduplikační metoda (dále jen RDM) si však všímá i jiných aspektů. Sleduje, jak moc jsou si dva záznamy blízké, avšak relativně vhledem k tomu, jak blízko mají k dalším sousedům. K tomu jsou definovány dvě vlastnosti pro množiny potenciálních duplicit: kompaktnost a řídké okolí. Formálně zavedeme tyto pojmy níže. Využití těchto vlastností k deduplikaci je mnohem výhodnější než využití jedné globální slučovací hodnoty použité u tradičních vzdálenostních metod, protože tyto vlastnosti vycházejí ze zkušeností, že i skutečné duplicitní záznamy mají tyto vlastnosti:

- Duplicity mají k sobě blíže než k ostatním záznamům.
- V jejich okolí se vyskytuje jen málo záznamů.

V tabulce 5.1 je možné vidět několik množin (clusterů) duplicitních záznamů: {1, 2}, {3, 4}, {5, 6}. Dále si všimněme, že záznamy {7, 8, 9, 10} by pravděpodobně byly sloučeny tradičními vzdálenostními metodami, protože například záznam 7 je nejbližší záznamu 8, 8 je nejbližší záznamu 9 atd. Tradiční metody předpokládají, že relace „být nejbližší soused“ je relací tranzitivní. Tj. pokud záznam *b* je nejbližší záznamu a a záznam *c* je nejbližší *b*, pak je celá množina {*a, b, c*} označena za duplicitní i přesto, že vzdálenost záznamů a a *c* může být obrovská. Tím pádem může být výsledkem takových metod velké množství neoprávněně sloučených záznamů.

Snahou RDM je naopak považovat za duplicity jen ty záznamy, jež jsou si vzájemně svými nejbližšími sousedy. Z toho plyne použití první vlastnosti potenciálně duplicitních záznamů – kompaktnosti. Tudíž mohou být sloučeny jen ty záznamy, které mají k sobě blíže než k ostatním, tedy ty, které jsou si svými vzájemnými nejbližšími sousedy.

Tato vlastnost však není postačující pro hledání duplicit. Uvědomme si, že množina všech záznamů relace vykazuje vlastnost kompaktnosti, přesto není možné celou tuto skupinu označit za duplicity. Proto je použita i vlastnost řídkého okolí, která zamezuje tomuto extrémnímu případu. Není to ale samozřejmě jediný případ, kterému je nutné zabránit. Představme si velkou skupinu záznamů vzájemně velmi podobných, přesto se nejedná o duplicity (například seznam jmen v telefonním seznamu u příjmení Novák nebo záznamy {7, 8, 9, 10} v tabulce 5.1) V řeči metriky jsou si tyto záznamy velmi blízké díky podobné syntaxi. Zřejmě se však nejedná o duplicity, v jejichž sloučení zabrání právě použití vlastnosti řídkého okolí. Tudíž mohou být sloučeny jen ty záznamy, které nejsou obklopeny velkým množstvím záznamů.

ID	Autor	Název písně
1	The Doors	LA Woman
2	Doors	LA Woman
3	The Beatles	A Little Help from My Friends
4	Beatles, The	With A Little Help From My Friend
5	Shania Twain	Im Holdin on to Love
6	Twian, Shania	I'm Holding On To Love
7	4th Elemynt	Ears/Eyes
8	4th Elemynt	Ears/Eyes - Part II
9	4th Elemynt	Ears/Eyes - Part III
10	4th Elemynt	Ears/Eyes - Part IV

Tabulka 5.1: Ukázka dát autorů a jejich písní. (Převzato z článku Robust identification of fuzzy duplicates [28])

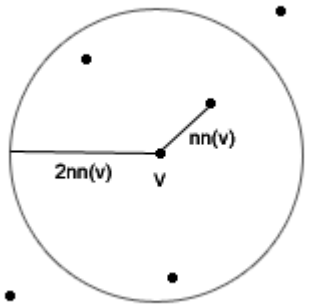
Nyní formálně zapíšeme vlastnost kompaktnosti a řídkého okolí. K definování těchto vlastností zavedeme nejprve následující značení. Mějme relaci R a symetrickou vzdálenostní funkci d: R x R → [0,1]. Bez újmy na obecnosti předpokládejme, že množiny potenciálních duplicit obsahují vždy dva nebo více záznamů.

Kompaktnost
Množina záznamů S je kompaktní, pokud pro každý záznam v ∈ S platí: d(v, v') < d(v, v''), kde $v' \in S$ a $v'' \in R - S$. Tedy množina záznamů je kompaktní, pokud vzdálenost mezi nimi je vždy menší než vzdálenost k ostatním záznamům v relaci.

Řídké okolí
Pro tento pojem je nejprve nutné zavézt funkce *nn* a *ng*:

- nn(v) je vzdálenost mezi v a jeho nejbližším sousedem.
- ng(v) = |{u | d(u, v) < p · nn(v) }|, kde *p* je v našem případě konstanta zafixovaná na hodnotu 2. Jedná se tedy o počet záznamů, které se vyskytují od *v* ve vzdálenosti menší dvojnásobku vzdálenosti k nejbližšímu sousedovi *v*. Viz obrázek 5.1.

Množina záznamů S má řídké okolí, pokud pro její záznamy platí: agg(ng(v)) < *c*, kde *c* je konstanta a *agg* je agregační funkce, v našem případě maximum. Zjednodušeně řečeno, množina S má řídké okolí, pokud pro její záznamy v platí, že počet sousedů v určité vzdálenosti od v není velký. Přitom tato vzdálenost není jednotně určena pro všechny záznamy, nýbrž se odvíjí od toho, jak daleko má každý záznam svého nejbližšího souseda. Ještě dodejme, že místo maxima je možné použít i jinou agregační funkci, jako je například průměr.

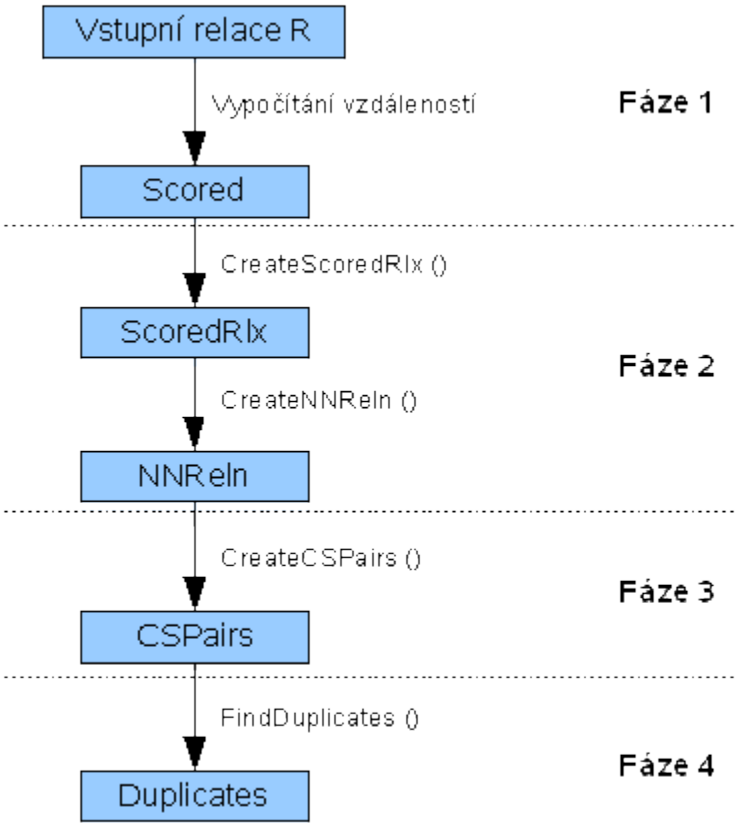


Obrázek 5.1: Znázornění výpočtu hodnoty řídkého okolí. Zde ng(v) = 3.

5.2 Implementace

Jak vyplývá z předchozí sekce, cílem RDM je nalezení všech množin S záznamů z relace *R*, pro které platí vlastnost kompaktnosti i řídkého okolí. Do výpočtu ještě vstupuje konstanta *c*, která je využita ve vlastnosti řídkého okolí. Zmiňme, že pro naše účely byl algoritmus navržen odlišným způsobem, než je ten původní [28], a tak se v některých krocích výrazně liší. Jako vývojové nástroje byl využit nástroj SAS a MS SQL. Nyní tedy popíšeme naší verzi algoritmu, který probíhá v těchto fázích (viz obrázek 5.2):

1. spočítání vzdáleností mezi každými dvěma záznamy;
2. vytvoření seznamu nejbližších sousedů;
3. nalezení kompaktních množin;
4. nalezení duplicit.



5.2.1 Fáze 1: spočítání vzdáleností mezi záznamy

Tento první krok naší metody byl implementován v nástroji SAS. Implementační detaily zle vyčíst ze zdrojových kódů a jejich komentářů na přiloženém CD. Oproti původní metodě [28], ve které lze využívat jen editační a kosinová metrika [7], je v naší

metodě možné využít jakoukoli. Přesto jsme pro výpočet vzdáleností mezi záznamy vybírali jen ty „základní“ metriky (z kategorie editačních metrik) a jen mírně jsme je přizpůsobili našim požadavkům. Důvodem, proč jsme nevolili složitější metriky a proč nebyly přizpůsobovány přímo na míru daným datům je, že jsme chtěli ukázat, že i bez znalosti domény je možné dosáhnout dobrých výsledků.

Hlavní metrikou, která byla využita, je Spedis [45] nástroje SAS. Jde o upravenou editační metriku, kde každá operace má svou váhu. Mezi její další hlavní specifika patří:

- Váha záměny dvou písmen je menší než váha vložení či mazání znaku.
- Rozdílnost dvou řetězců na začátku slova je penalizována více něž na konci, což koresponduje s tím, že váha písmen na začátku slov je mnohem vyšší a že překlepy se spíše stávají uprostřed nebo na konci slov.
- Editační metrika je relativně upravena vůči délce řetězce. Tj. rozdíly v krátkých slovech jsou penalizovány více než rozdíly v dlouhých slovech.

Tato metrika je využita pro většinu atributů. Pro některé důležité atributy, jako je například příjmení, nebyla volena jiná metrika, jen byla kombinována s dalšími úpravami, (které částečně zastupují čištění dat, viz 3.1):

- Odstranění diakritiky, což je v českém prostředí často výhodné.
- Převedení písmen na velká písmena.
- Oříznutí delšího slova na délku kratšího slova (s připočtením penalizace, která se odvíjí od rozdílu délek obou slov a délky nejdelšího slova).

Pokud bylo kombinováno více úprav, vždy byly spočítány metriky s nimi a bez nich a jako výsledek byla použita minimální nalezená vzdálenost.

Váhy atributů

Výsledná vzdálenost dvou záznamů *u, v* je spočítána jako vážený součet podobností jejich atributů:

$$d(u,v)=\sum_{i=1}^{K} s(u_i, v_i) w_i \quad ,$$

kde $s(u_i, v_i)$ je podobnost i-tých atributů záznamů u a *v*; K je počet společných atributů a w_i je váha i-tého atributu. Nakonec je vzdálenost normována na interval [0,1]. Hodnoty w_i jsou získány od uživatele, což poněkud narušuje naší koncepci omezit nutnost znalosti domény při nasazení metody. Na druhou stranu lze podotknout, že pro nejčastější deduplikační úkoly (deduplikace osob nebo adres) je možné váhy nastavit jednorázově už při implementaci metody. Pro optimální nastavení vah existuje několik metod, například Economic and multiattribute evaluation [46] nebo A process to determine priorities and weights for large-scale linear goal programming [47]. V našem případě se však ukázalo, že pro dosažení dobrých výsledků v deduplikaci osob a adres postačilo použít intuitivní váhy odpovídající důležitosti atribu-

tů a ty jen následně znormovat (tak, aby výsledná vzdálenost d*(u, v)* spadala stále do intervalu [0,1]). Je zde důležité hlavně to, aby se ve vahách odrážela rozlišovací síla atributů. Např. PSČ nemá takovou rozlišovací sílu, protože stejné PSČ se vyskytuje u mnoha (unikátních) záznamů. Opakem je např. příjmení.

Tabulka 5.2 znázorňuje nastavení vah pro atributy osob. Tabulka 5.3 obsahuje ukázku výstupu této první fáze metody RDM do databázové tabulky nazvané „Scored“. Obsahuje id dvou záznamů a jejich vzdálenost. Všimněme si, že pro urychlení výpočtu a zamezení dvojího porovnávání vždy měříme vzdálenost mezi záznamy v_i a v_j, pokud i < j.

jméno	0,8
příjmení	1
ulice	0,7
město	0,5
psč	0,1
telefon	0,3
email	0,3

Tabulka 5.2: Nastavení vah pro atributy osob (před normalizací)

ID1	ID2	d
1	2	0,1
2	3	0,2
1	3	0,3
1	4	0,9
2	4	1
3	4	1

Tabulka 5.3: Relace Scored: Výstup prního kroku deduplikační metody

5.2.2 Fáze 2: vytvoření seznamu nejbližších sousedů

Od tohoto kroku se výpočet provádí v databázi s využitím tabulek, SQL dotazů a T-SQL v prostředí MS SQL. To nám umožňuje využít mnoha výhod, které databáze nabízejí: rychlé a pohodlné zpracování dat, indexy pro rychlé vyhledávání v tabulkách, primární klíče, dotazovací sílu SQL aj.

V původní metodě [28] byl mimo databázi vytvářen index nejbližších sousedů postavený na přibližném výpočtu [44], který je možný pouze pro některé metriky. Pro naše účely není tento index nutný, a tak vytváříme přesný seznam nejbližších sousedů.

Vstupem pro tuto fázi je výstup předchozí fáze, tedy seznam dvojic záznamů a jejich vzdáleností z intervalu [0,1], jak je možné vidět v tabulce 5.3.

Pro připomenutí uveďme, že vlastnost kompaktnosti mimo jiné o množině záznamů tvrdí, že tyto záznamy si jsou vzájemně svými nejbližšími sousedy. Tato vlastnost se nejlépe ověří pomocí seznamů nejbližších sousedů a jejich vytvoření je úkolem této fáze.

Postup vytvoření seznamu nejbližších sousedů:

1. Z dat vstupní tabulky (tabulka 5.3) je vytvořena obdobná tabulka „ScoredRlx“, která doplňuje tu původní na reflexivní relaci. Tj. pokud je v původní tabulce uvedeno, že d(1,2) = 0,1, tak v této nové tabulce bude uve-

deno i d(2,1) = 0,1. Na této tabulce jsou nastaveny indexy, které nám umožní rychlé vyhledávání.
- Tento krok je řešen procedurou CreateScoredRlx(), viz obrázek 5.1.

2. Pro každý záznam je nalezen seznam k sousedů seřazen od nejbližších po nejvzdálenější. Výsledek je uložen v relaci NNReln. V našem případě vycházejícím z tabulky 5.3 jsou pro záznam 1 nejbližšími sousedy záznamy 2, 3.
   - Tento krok je řešen procedurou CreateNNReln(), viz obrázek 5.1.
3. Současně při výpočtu nejbližších sousedů je spočtena hodnota ng(*v*), tedy počet záznamů nacházejících se v nejbližším okolí *v.* Viz vlastnost řídkého okolí v sekci 5.1. V našem případě ng(1) = 1; ng(2) = 2. Tabulka 5.4 ukazuje celkový výstup této fáze pro data použitá v první fázi.

Nastavení konstanty *k*
Jak jsme již zmínili, pro každý záznam je nalezen seznam nejbližších sousedů velikosti *k*. Metodu RDM jsme implementovali pro hodnoty k = 2 nebo k = 3. To nám umožňuje nalézt množiny duplicit o velikosti až 3 záznamů (pro hodnotu k = 2) resp. 4 záznamů (pro hodnotu k = 3).

ID	NN1	NN2	NG
1	2	3	1
2	1	3	2
3	2	1	2
4	1	2	3

Tabulka 5.4: Relace NNReln obsahující pro každý záznam seznam 2 nejbližších sousedů a hodnotu ng(v)

5.2.3 Fáze 3: nalezení kompaktních množin

Nyní, když máme k dispozici seznam nejbližších sousedů, je možné nalézt kompaktní množiny. Tedy množiny záznamů, jež mají společné nejbližší sousedy. V prvním kroku jsou vytvořeny jen množiny o dvou záznamech. Dále však uvidíme, že už díky těmto malým množinám je možné snadno určit kompaktní množiny i o více záznamech.

Výsledek této fáze je relace CSPairs, jejíž ukázka je v tabulce 5.5. Tato relace obsahuje id dvou záznamů, hodnoty CS2 resp. CS3 značí, zda se dvou resp. tří prvkové množiny nejbližších sousedů obou záznamů shodují.

Vytvoření relace CSPairs je velmi jednoduché. Stačí provézt spojení relace NNReln se sebou za podmínky, že ID1 z relace NNReln1 je menší než ID2 z relace NNReln2 a že ID1 se vyskytuje v seznamu nejbližších sousedů ID2 a naopak. Protože jako agregační funkci pro výpočet *ng* používáme maximum, je již zde oproti původní metodě [28] možné vyloučit ty páry, které podmínku řídkého okolí nesplňují.

Tato fáze je provedena procedurou CreateCSPairs(), jak je možné vidět na celkovém přehledu obrázku 5.1.

Tabulka 5.5 ukazuje, že záznamy {1,2} tvoří kompaktní množinu. Množina jejich nejbližších sousedů (včetně jich samých) o velikosti 2 se shoduje (CS2 = 1) a dokonce se shoduje i množina jejich tří nejbližších sousedů (CS3 = 1). Stejně tak i množina {1,3} je kompaktní, protože CS3 = 1. Díky tomu lze dokonce usoudit, že množina {1,2,3} je kompaktní.

ID1	ID2	CS2	CS3	MAX_NG
1	2	1	1	2
1	3	0	1	2
2	3	0	1	2

Tabulka 5.5: Relace CSPairs zobrazující shodu množin nejbližších sousedů

Vlastnosti relace CSPairs

Tato relace je asi nejdůležitějším mezikrokem při hledání duplicit. V této tabulce je možné vypozorovat hned několik důležitých faktů:

- Hodnota CS2 = 1 značí, že dva záznamy ID1 a ID2 si jsou svými nejbližšími sousedy.
- Hodnota CS2 = 0 značí, že záznam ID1 není nejbližší záznamu ID2 nebo naopak.
- Jak je vidět v tabulce 5.5, je možné že CS2 = 0 a zároveň CS3 = 1. Záznam 3 není nejbližší soused záznamu 1 (proto CS2 = 0), avšak platí, že díky záznamu 2 mají shodný seznam sousedů (včetně záznamů samých) o velikosti tři: {1:2,3} = {3:2,1}.
- Pokud by byla počítána i hodnota CS4, může se přihodit, že CS4 = 0 i přesto, že CS3 = 1. Tento jev nastává, pokud se ke kompaktní množině o velikosti 3 „přiblížil“ čtvrtý záznam, který porušuje podmínku kompaktnosti.
- Dále si povšimněme, že jsme o množině {1,2,3} usoudili, že je kompaktní jen na základě kompaktních množin {1,2} a {1,3} bez posuzování množiny {2,3}. Tento fakt vyplývá z toho, že pokud mají množiny {1,2} a {1,3} shodné nejbližší sousedy, musí je mít shodné i dvojce záznamů {2,3}.
- Důsledkem předchozího je, že díky údajům o kompaktních párech záznamů lze nalézt kompaktní množiny libovolné velikosti a navíc bez ověřování všech jejich dvou prvkových podmnožin.

5.2.4 Fáze 4: nalezení duplicit

Úkolem této poslední fáze je vhodně seskupit kompaktní páry z relace CSPairs tak, aby se nalezly největší množiny kompaktních párů. Protože je relace CSPairs tranzitivní a všechny její kompaktní páry (ID1, ID2) mají vlastnost ID1 < ID2, stačí ověřovat jen ty páry, které začínají stejným ID1. V našem případě jsou to dvojce (1,2)

a (1,3). U každé takové skupiny párů je ověřeno, zda dohromady splňují vlastnost kompaktní množiny. Pokud ne, je vybrána jejich největší podmnožina, která tuto vlastnost splňuje. Hledáme tedy množinu záznamů G = $\{v_1,...,v_m\}$, kde m ≤ K (počet společných atributů) a $v_i < v_j$ (i < j) za podmínky, že množiny $\{v_1,v_2\},..\{v_1,v_m\}$ mají společné sousedy.

Nalezení největší kompaktní podmnožiny

V ideálním případě tvoří všechny páry $\{v_1,v_2\},..\{v_1,v_m\}$, (které mají stejný první prvek), kompaktní množinu. K tomu dojde, pokud v této relaci obsahují všechny páry ve sloupci CS2 nebo CS3 jedničku. V praxi ale dochází k tomu, že některý ze záznamů v_i do kompaktní množiny nezapadá, tj. neexistuje sloupec, který by obsahoval pro všechny páry $\{v_1, v_i\}$ jedničku. Abychom nalezli všechny ostatní páry, jež dohromady kompaktní jsou, musí být zvolen sloupec, který obsahuje nejvíce jedniček pro skupinu párů $\{v_1, v_i\}$. K tomu jsme opět využili sílu SQL. Jednoduše lze totiž sečíst pro každou skupinu potenciálních kompaktních množin *G* hodnoty ve sloupci CS2 a ve sloupci CS3 a porovnat výsledky. Sloupec, který obsahuje nejvyšší počet jedniček, je určující pro nalezení kompaktní podmnožiny.

Po nalezení rozhodujícího sloupce, jsou již snadno vybrány ty záznamy, jež mají v daném sloupci jedničku. To je provedeno pro každou skupinu párů $\{v_1, u_i\}$ začínající stejným záznamem. Tento proces zajišťuje procedura FindDuplicates().

Výsledné množiny duplicit jsou ukládány do relace Duplicates (viz tabulka 5.6), která se skládá z id záznamu a id clusteru (shluku), kam daný záznam patří. Podotkněme, že pro snadnější a rychlejší výpočet je využito primárního klíče pro id záznamu v této tabulce. Tím je zabráněno přiřazení jednoho záznamu do dvou různých shluků duplicit.

ID	CLUSTER
1	1
2	1
3	1

Tabulka 5.6: Relace Duplicates obsahující zařazení záznamu do skupiny duplicit.

Poznámky k hledání duplicit

- Je možné, že jeden záznam může náležet do více potenciálních množin duplicit. Potom jsou to ale podmnožiny různě velké a je vždy vybrána největší z nich, nebo jsou to podmnožiny stejně velké, kdy je vybrána ta podmnožina, jež je určena lepším „ukazatelem kompaktnosti“. Lepší ukazatel než CS2 je CS3, CS4 je lepší než CS3 atd. To je proto, že např. záznamy mající v CS3 jedničky mají více společných sousedů než záznamy, co mají jedničky ve sloupci CS2.
- Pokud je nejprve záznam přiřazen do nějaké výsledné množiny duplicit a následně by měl být označen znovu do nějaké nové skupiny duplicit, je tomu

zabráněno právě kvůli primárnímu klíči v relaci Duplicates. V našem příkladě by bylo po nalezení clusteru {1,2,3} zabráněno vložení množiny {2,3}.

- Vždy platí, že záznam je zařazen do největší možné skupiny duplicit G = $\{v_1,...,v_m\}$, protože jinak by v_1 nemohl být nejmenší prvek v G.
- Výpočet pro konstantu k = 3, která umožňuje nalezení až čtyř prvkových množin duplicit, probíhá obdobně jen s menšími rozšířeními v jednotlivých tabulkách a procedurách, jejichž název obsahuje příponu „K3“. Viz zdrojové soubory na přiloženém CD.

5.3 Shrnutí robustní deduplikační metody

Nyní, když máme představu o funkčnosti naší metody, shrneme ta nejdůležitější fakta a detaily nutné k praktickému použití. Naše metoda spadá do kategorie vzdálenostních metod. Přesto nejde o klasický příklad, protože zde není nutné stanovit jeden konkrétní práh podobnosti nutný pro rozhodnutí o slučování záznamů. Využíváme poznatku, že duplicitní záznamy se nacházejí z hlediska podobnosti blízko sebe a naopak daleko od ostatních unikátních záznamů či jiných duplicitních skupin. Tyto vlastnosti, které jsme definovali jako kompaktnost a řídké okolí, jsou využity v samotném procesu hledání duplicit a to v podobě ověřování seznamů nejbližších sousedů a počtu záznamů v blízkém okolí (hodnota *ng*). Tento výpočet je velmi efektivní a navíc díky nepřítomnosti jedné konkrétní slučovací hranice dokáže rozpoznat i duplicity, jejichž syntaxe je velmi rozdílná, a naopak nedovolí sloučení unikátních záznamů, ač si jsou velmi podobné. Viz tabulka 5.1.

Ve výpočtu přesto musí být použito několik konstant. Nutné je však podotknout, že ku prospěchu uživatelské přívětivosti a snadného nasazení většinou není nutné tyto konstanty ladit. Nyní zrekapitulujeme všechny konstanty, které do výpočtu vstupují:

- k – Pro velikost seznamu nejbližších sousedů. Udává i maximální velikost nalezených skupin duplicit. Ta je vždy $k + 1$. V naší metodě nastavujeme nejčastěji na hodnotu 2 (výjimečně na hodnotu 3, podle toho, kolik duplicit v relaci očekáváme). Viz sekce 5.2.2.
- *p, c* – Konstanty použité pro ověření vlastnosti řídkého okolí. Viz sekce 5.1. Pro naší metodu bylo zvoleno p = 2 a c = 5. Uveďme, že hodnotu konstanty *c* lze stanovit na základě očekávaného množství duplicit v relaci. Detailní způsob výpočtu je uveden v článku Robust identification of fuzzy duplicates [28].
- Váhy atributů – Ty jsou použity pro výpočet podobnosti dvou záznamů. Tyto váhy, jak bylo zmíněno v sekci 5.2.1, lze pro dané druhy úloh nastavit jednorázově.

– agg – Nejedná se přímo o konstantu ale o agregační funkci pro výpočet řídkého okolí celé skupiny záznamů. V našem případě je použito maximum, což nám i dovolilo urychlit běh algoritmu.

Minimalizace kompaktních množin

Naše metoda vytváří skupiny duplicit o velikosti až $k + 1$. Specifikem naší metody zároveň je, že nehledá duplicity čistě na základě vzájemné podobnosti, ale na základě shodnosti seznamů sousedů. Jelikož hledáme maximální kompaktní množiny, může nastat následující jev. Mějme čtyři záznamy A, B, C, D a hledejme v nich duplicity s parametrem k = 3. Záznam A je velmi podobný záznamu B a záznam C záznamu D. Měly by tedy vzniknout dvě výsledné množiny duplicit {A, B}, {C, D}. Zároveň uvažujme, že podobnosti mezi záznamy A–C, A–D, B–C, B–D nejsou nulové. Tudíž při stávající volbě k = 3 dojde k vytvoření jedné kompaktní množiny {A, B, C, D}, protože seznamy nejbližších sousedů se budou shodovat. V extrémním případě by při volbě k = |R| - 1 došlo ke sloučení všech záznamů v relaci R. Proto je vhodné v naší metodě nastavovat konstantu k na menší hodnoty, obzvlášť pokud víme, že se v relaci nevyskytují větší skupiny duplicit. Další způsob, jak se vyhnout tomuto jevu, je minimalizace výsledných kompaktních množin. Na konci našeho algoritmu by bylo nutné zkontrolovat, zda výsledné množiny nejdou rozdělit na menší stále kompaktní podmnožiny. Protože ale k tomuto jevu dochází v praxi velmi zřídka, nebylo ani pro naše účely nutné toto ověřování v algoritmu zahrnout.

Clustrování vstupních záznamů

Aby došlo k urychlení výpočtu, je vhodné rozdělit velké množství vstupních dat do skupin a až v nich provézt samotné hledání duplicit. Tento krok se používá u většiny vzdálenostní metod, kde je nutné měřit vzdálenost každého záznamu s každým. Je nutné zmínit, že tento krok vždy zvyšuje riziko výskytu falešně nesloučených záznamů. V našem případě jsme pro naše testovací data s osobními údaji použili jednoduché clustrování podle prvního písmena příjmení. Více o použitých testovacích datech je uvedeno v sekci 6.3 zaměřené na srovnání výsledků.

Výsledky metody

Robustní deduplikační metoda vykazuje velmi dobré výsledky z hlediska přesnosti a úplnosti. Jak je možné vidět v konkrétních měřeních článku Robust identification of fuzzy duplicates [28], tato metoda dosahuje lepších výsledků než metody založené na použití jednoho slučovacího prahu.

V následující kapitole shrneme vlastnosti deduplikačních metod a detailněji se zaměříme na srovnání této metody nevyužívající znalosti domény s metodou využívanou v praxi, která se naopak bez detailních znalostí o datech neobejde.

6 Porovnání deduplikačních metod

Základní rozdíly v principech deduplikačních metod byly představeny ve čtvrté kapitole. Nyní srovnáme detailněji jejich výhody, nevýhody a vhodnost použití v konkrétních případech.

Nejprve je však vhodné uvědomit si několik základních faktů důležitých při srovnávání metod. Snad tím nejzákladnějším je, že srovnávání metod může být nejenže velmi složité, ale i subjektivní. Tabulka 6.1 znázorňuje dva potenciálně duplicitní záznamy. Mohou nastat dva případy: Jedná se o tutéž osobu, ať už z důvodu chyby v hodnotě příjmení, nebo z důvody přejmenování, a tudíž jde o duplicity, nebo se jedná o dva unikátní záznamy, přičemž se alespoň v jednom z rodných čísel vyskytuje chyba nebo obě osoby mají přidělené stejné RČ. Je tedy vidět, že bez dalších znalostí nelze u mnoha případů o metodě říct, zda rozhodla správně nebo chybně, ať už dané záznamy sloučí nebo ne.

RČ	Jméno	Příjmení	Pohlaví
7008064869	Tomáš	Veselý	M
7008064869	Tomáš	Ženatý	M

Tabulka 6.1: Relace obsahující dvojici potenciálních duplicit.

Deduplikační metody sice mohou zohlednit míru pravděpodobnosti jednotlivých situací (např. přejmenování muže po svatbě nebo výskyt dvou stejných RČ u dvou různých osob je velmi málo pravděpodobné), přesto nelze nikdy vyloučit jistou míru subjektivity v rozhodování jednotlivých metod. Skutečně správné rozhodnutí, zda metoda nalezla duplicity správně, by dokázal dát jen ten, kdo reálně zná oba objekty vstupující do databáze a nejlépe i procesy účastnící se zpracování daných záznamů.

Dále si uvědomme, že neexistuje jedna globálně nejlepší metoda, která by byla nejúspěšnější na všech relacích. To částečně platí i pro metody slučující jednoatributové záznamy. Zde je však situace o něco jednodušší, protože lze celkem snadno nalézt jednu z vhodných metod s přihlédnutím k daným řetězcům. Např. pro porovnávání jmen se ukazuje jako nejlepší metrika Jaro [11], a pro delší řetězce jsou vhodné kontextové metriky (viz kapitola 4.1). Obdobně lze nalézt vhodnou metriku pro atributy číselné, obsahující telefon, email atd.

U slučování víceatributových záznamů je situace složitější o to, že se k problému přidává různá struktura relací, rozdílnosti ve specifikacích metod a jejich podmín-

kách na vstupní data, rozdílné požadavky na trénovací data nebo jiné uživatelské úsilí nutné ke spuštění metody apod.

V další části se tedy budeme zabývat tím, jaké jsou možnosti ohodnocení metod a srovnáme výhody a nevýhody jejich hlavních kategorií. Následně představíme vlastní algoritmus pro srovnání výsledků dvou metod, s jejíž pomocí se v závěru této kapitoly zaměříme na porovnání DataFluxu a RDM.

6.1 Přehled výhod a nevýhod

K ohodnocení metod se často používá těchto metrik:

- přesnost – podíl správně nalezených duplicit a všech nalezených duplicit,
- úplnost – podíl nalezených duplicit a všech duplicit skutečně obsažených v relaci,
- efektivita – rychlost běhu algoritmu,
- obecnost – schopnost nasazení metody na různé případy,
- nutnost uživatelských vstupů – množství uživatelských zásahů nutných před a během běhu algoritmu (nastavování různých prahů sloučení, ohodnocování trénovacích dat atd.)

Uveďme, že většina tvůrců metod vykazuje velmi vysokou přesnost i úplnost. Jenže tyto ukazatelé jsou spjaty s daty a dalšími výše zmíněnými podmínkami. Jako příklad uveďme metodu Entity matching in heterogeneous databases [26], která velmi účinně vyhledává duplicity během integrace dvou relací, které uvnitř sebe obsahují pouze unikátní záznamy. Na všech jiných datech by tato metoda použít nešla, nebo by vykazovala špatné výsledky. Přesto existují srovnávací studie (např. Frameworks for entity matching [49]), které porovnávají přesnost, úplnost a další ukazatele u metod, z nichž většina nepracuje se stejnými daty.

My se nyní spíše zaměříme na srovnání obecných výhod a nevýhod hlavních skupin deduplikačních metod z hlediska jejich nároků, výsledků a využitelnosti v praxi. Použili jsme rozdělení na tyto hlavní skupiny:

- pravděpodobnostní metody,
- metody s učením (včetně aktivního učení, učení s učitelem apod.),
- metody založené na pravidlech,
- vzdálenostní metody.

6.1.1 Pravděpodobnostní metody

Pravděpodobnostní metody jsou jednou z nejstarších metod vůbec. Jedná se o metody, které k rozhodnutí o spárování dvou záznamů využívají pravděpodobnostního rozdělení porovnávacích vektorů. Základní varianta se velmi podobá vzdálenostním

metodám, protože také využívá jedné slučovací hranice, kterou je nutné ručně nastavit.

Asi proto, že jsou tyto metody špatně škálovatelné a jejich výpočet málo průhledný a často vyžadující vzorek ohodnocených dat, se tyto metody v praxi příliš neuchytily. Existují však zajímavé varianty, které například dokáží uvažovat cenu pro chybné sloučení a chybné nesloučení (false positive a false negative). Tím je proces deduplikace převeden na minimalizaci této ceny v rámci všech záznamů. Obdobně si počíná metoda Entity matching in heterogeneous databases [26], která dokazuje vzájemnou podobnost pravděpodobnostních a vzdálenostních metod. Tato metoda se jasně řadí mezi vzdálenostní metody a také minimalizuje cenu chybných sloučení a nesloučení. Dokonce lze při výpočtu nastavit poměr obou těchto chyb, což může být velmi užitečné, chceme-li škálovat „volnost“ či „agresivitu“ slučování.

6.1.2 Metody s učením

Tyto metody jsou logickým nástupcem pravděpodobnostních metod. V různě velké míře využívají předem ohodnocená trénovací data, což přináší určitá rizika. Kromě toho, že trénovací data nemusí být vůbec dostupná, existuje i riziko, že množství trénovacích dat nebude dostačující nebo že nepokryjí všechny možné případy, kdy by mělo či nemělo dojít u unifikaci. Je sice jednodušší uměle vygenerovat velké množství trénovacích dat, ty však vykazují jisté podobné vzorce a často nepokrývají rozmanitou škálu chyb, které se vyskytují v reálných databázích. Také obsahují z velké části skupiny zřejmých duplicit a zřejmých unikátních záznamů. Naopak ty vzorky dat, které by skutečně byly vhodné pro učení, se podaří získat jen v malém množství. Proto se může stát, že tyto metody nebudou příliš účinné, nebo že budou „přeučené“ jen na konkrétní trénovací data.

Přesto některé srovnávací studie (např. Framework for entity matching [49]) v mnoha případech ukazují, že metody, které jsou dobře nastavené a podpořené kvalitním vzorkem trénovacích dat v dostačujícím množství, vykazují často lepší výsledky než metody fungující na jiném principu.

Opět ale musíme zmínit, že vykazovaná „účinnost“ některých metod je velmi subjektivní. Pro příklad uveďme metodu Multiple classifier system [50], která dosahuje výjimečně dobrých výsledků, ale jen díky tomu, že pro deduplikaci 25 000 záznamů používá ohromné množství trénovacích dat, jež tvoří 65 % velikosti původních dat. Dobrá „účinnost“ je tedy vyvážena nutností náročného uživatelského úsilí v podobě ručního ohodnocení dat.

Další nevýhodou učících metod je, že výpočet probíhá pro uživatele velmi neprůhledně. V případě vrácení chybných výsledků lze těžko upravovat konkrétní parametry nebo kroky algoritmu. Nelze jednoduše říci, jak a proč algoritmus vydal právě ty které výsledky. V praxi z toho pramení obliba spíše v metodách založených na pravidlech.

6.1.3 Metody založené na pravidlech

Tyto metody jsou hojně využívány v praxi díky možnosti postihnout většinu případů, které způsobují nekonzistentnost v datech. Využívá se při tom znalosti domény, dat, kulturního a historického pozadí apod. Tím je také jejich účinnost velmi vysoká.

S tím souvisí i další výhoda těchto metod a to v případě, kdy se při vývoji, testování nebo nasazení této metody objeví nějaký případ, na kterém metoda selhala. Tj. sloučila očividně unikátní záznamy nebo naopak. Je totiž okamžitě možné odstranit nebo upravit nevhodná pravidla, která se podílela na neoprávněném sloučení nebo nasadit úplně nová pravidla. Velké množství jiných metod nedokáže často jednoznačně určit, proč dva záznamy označila jako unikátní nebo duplicitní, a pokud to dokáže, tak jen v řeči matematických formulí nebo pro uživatele „nicneříkající“ hodnotou podobnosti apod.

Na druhé straně stojí velmi náročná pracnost při hledání a vytváření všech pravidel. Pro představu uveďme, že pravidla pro slučování adres, které typicky vykazují nespočet různých druhů zápisů, se počítají na tisíce. Zároveň je nutné podotknout, že většina klasických metod využívající pravidla potřebuje provézt kvalitní čištění dat. V příloze A na konci této práce je uvedena tabulka využitá při takovém čištění. Z ní je více než patrné, jak náročné je nasazení těchto metod.

Pokud jsou ale již jednou slučovací pravidla vytvořena odpadá nutnost zásahu uživatele kvůli ladění nějakých slučovacích hranic apod. To je na jednu stranu velmi výhodné. Uživatel nemusí disponovat znalostí daného algoritmu a všech jeho implementačních detailů. Na druhou stranu je pravda, že je tím uživatel omezen z hlediska škálovatelnosti slučování, protože z pohledu slučovacích pravidel je svět černobílý. Tj. záznamy jsou buď na 100% duplicitní nebo na 100% unikátní. Ale samozřejmě existují i výjimky u metod, které kombinují více slučovacích principů (např. výše zmíněná metoda Learning based fusion [43]) nebo které využívají metrik umožňujících nastavení citlivosti.

6.1.4 Vzdálenostní metody

Vzdálenostní metody slučují záznamy na základě jejich vzájemné vzdálenosti, která je určena podobností jednotlivých atributů. Jejich velkou předností je, že většina těchto metod ke svému výpočtu nevyžaduje ani trénovací data ani výraznou znalost domény. To umožňuje snadnou implementaci a rychlé nasazení do praxe bez nutnosti složitého ladění.

Nevýhodou může být nutnost nastavování některých parametrů. Zde musíme zmínit například hranici podobnosti, nad kterou jsou dva záznamy považovány za duplicitní nebo váhy atributů, které je však nutné nastavovat i ve většině ostatních metod.

Existuje však mnoho variací těchto metod, které s různou mírou úspěšnosti eliminují předchozí nevýhody. Naše Robustní deduplikační metoda popsaná v této práci například nevyžaduje nastavení žádné slučovací hranice. Jiné metody se také dokáží

vyhnout tomuto nastavování, ale zas na úkor toho, že musí použít trénovací data apod.

Celkově je tato skupina metod velmi dobře použitelná v praxi, jak ukazuje i srovnání naší metody a metody v praxi již využívané – DataFlux. Viz sekce 6.3. Nejprve však představíme námi navrženou metodu pro účely porovnání nalezených clusterů dvou metod.

6.2 Srovnávací metoda pomocí porovnání množin

Pro účely porovnání dvou deduplikačních algoritmů jsme navrhli metodu, jejíž cílem je vzájemně porovnat jejich výsledky. V tento moment se nezajíme o další kritéria, jako je časová a prostorová náročnost, možnost využití metody on-line, přesnost, úplnost aj. Zaměřili jsme se čistě na objektivní srovnání výstupů obou metod, přesněji řečeno na nalezené unikátní záznamy a nalezené množiny unifikovaných záznamů, které daná metoda považuje za duplicitní a tudíž za ty, které by v reálném světě měly představovat tentýž objekt. Tyto nalezené množiny budeme dále označovat jako clustery.

Často se stává, že v reálných datech nelze manuálně a hlavně objektivně rozhodnout, které záznamy jsou duplicitní. Z toho plyne, že se na reálných datech nedá spočítat ani přesnost či úplnost použitých deduplikačních metod, která je oblíbeným měřítkem pro jejich srovnávání. Proto může být výhodné srovnávat výstupy metod mezi sebou než k nějakému výchozímu manuálně vytvořenému seznamu duplicit. Jindy se zase stává, že jsou porovnávány metody, které pracují na různých datech. Některé metody jsou tak úzce zaměřeny na konkrétní data, že je téměř nemožné je z hlediska úspěšnosti srovnávat s ostatními.

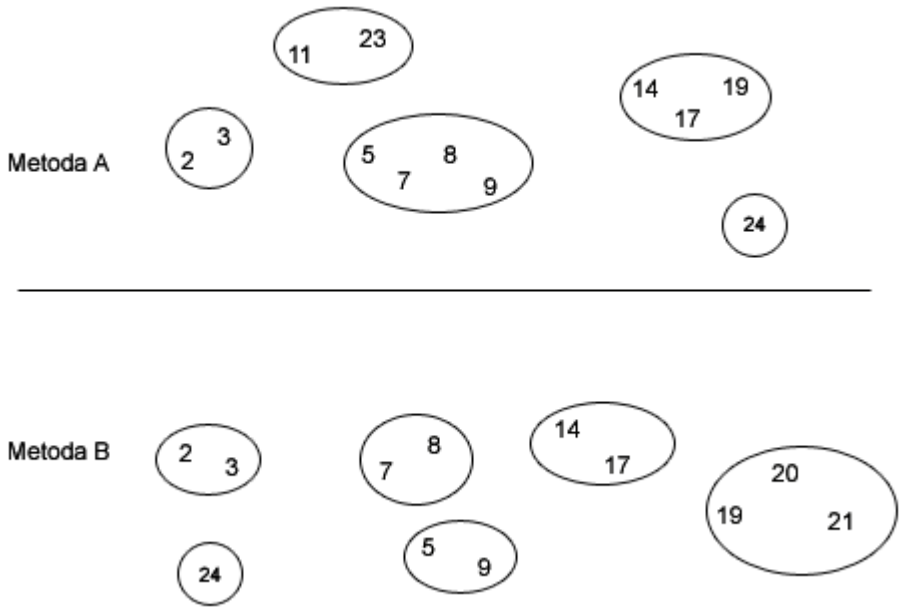
Náš algoritmus, nazvaný *Compare*, porovnává výhradně ty metody, které dokáží pracovat se stejným vstupem, čímž se rozumí stejný formát a stejné atributy vstupních relací, stejné množství dat apod.

6.2.1 Specifikace metody Compare

Algoritmus výpočtu vyžaduje dva vstupní seznamy záznamů. Jeden od každé metody. U každého záznamu musí mít určeno, do jakého clusteru patří. Tím je definováno rozčlenění na disjunktní množiny záznamů, které jsou označeny jako unifikované (duplicitní). Přitom ty clustery, které obsahují právě jeden záznam, značí, že tento záznam je unikátní.

Jelikož výstupem metod jsou clustery netriviálních duplicit, které jsou vyhledávány s určitou mírou nejistoty, není překvapením, že výsledek dvou metod pracujících s týmiž daty nebývá totožný. V ideálním případě jsou všechny výstupní clustery jedné metody shodné s clustery druhé metody. Může se však snadno stát, že některé duplicity může nalézt jen jedna z obou metod. Některé clustery mohou být nalezeny

druhou metodou jen z části, či se mohou dva clustery různě překrývat (tj. mají neprázdný průnik, ale nejsou shodné). Všechny možné případy znázorňuje obrázek 6.1 a níže uvedený seznam.



Obrázek 6.1: Schématické znázornění deduplikačního výsledku dvou metod

Jak je vidět, mohou nastat tyto případy:

- Clustery se rovnají: $\{2, 3\}_A$ a $\{2, 3\}_B$, v případě množin $\{24\}_A$ a $\{24\}_B$ se jedná o shodné nalezení unikátních záznamů.
- Cluster se ve výstupu druhé metody nevyskytuje: $\{11, 23\}_A$.
- Cluster má v druhé metodě pouze svou podmnožinu $\{5, 7, 8, 9\}_A$ a $\{7, 8\}_B$.
- Clustery se „překrývají“ (mají neprázdný průnik, ale ani jeden není podmnožinou druhého): $\{14, 17, 19\}_A$ a $\{19, 20, 21\}_B$.
- Mohou samozřejmě nastati i kombinace předchozích dvou případů.
  - Cluster může mít více podmnožin a to i různě velkých.
  - Cluster může mít jak podmnožinu, tak se může s dalším clusterem či více clustery překrývat: $\{14, 17, 19\}_A$, $\{14, 17\}_B$, $\{19, 20, 21\}_B$.

6.2.2 Výstup metody Compare

Metoda Compare zpracovává clustery obou zkoumaných metod. Výstupem je několik statistik, které slouží k detailnímu porovnání daných metod. Všechny údaje jsou ukládány do textových souborů. Viz výsledky na přiloženém CD. Nyní uvedeme seznam výstupních souborů, jejich obsah a význam.

resultsSummary.csv – Celkové srovnání obou metod A a B. Pro každou metodu obsahuje tyto hodnoty:

- počet duplicitních záznamů (tj. počet všech záznamů v clusterech o velikosti alespoň 2);
- počet clusterů o velikosti alespoň 2,
- průměrná velikost clusteru,
- počet shodných clusterů nalezených metodou A a B;
- počet clusterů nenalezených druhou metodou. Toto číslo např. značí, kolik clusterů metoda A nalezla a metoda B nikoli. Navíc žádný z clusterů metody B nemá ani žádný společný záznam s clustery metody A,
- počet clusterů částečně pokrytých clustery druhé metody. To jsou takové množiny duplicit, pro které ve výstupu druhé metody existuje alespoň jedna její podmnožina;
- počet shodných unikátních záznamů (tj. počet shodných clusterů velikosti 1),
- index pokrytí – číslo z intervalu [0,1] které značí vzájemnou podobnost nalezených clusterů oběma metodami. Podrobněji se touto hodnotou budeme zabývat níže.

resultsFullMatch.csv – Detailní přehled clusterů, které se úplně shodují jak ve výstupu metody A tak ve výstupu metody B. Jednotlivé sloupce obsahují:

- id clusteru metody A,
- id clusteru metody B (Očíslování clusteru obou metod se nemusí shodovat),
- záznamy daného clusteru z metody A,
- záznamy daného clusteru z metody B (Zde se jedná o stejné hodnoty jako v předchozím bodě.)

resultsPartialMatchA.csv – Detailní přehled clusterů z metody A, které jsou částečně pokryty clustery z metody B. Jednotlivé sloupce obsahují:

- id clusteru metody A;
- id jednoho nebo více clusterů metody B, které jsou podmnožinami předchozího clusteru;
- záznamy daného clusteru metody A;
- záznamy daných clusterů metody B.

resultsPartialMatchB.csv – Obdobné jako resultsPartialMatchA.csv. Zde se však sledují clustery metody B, které jsou částečně pokryty clustery z metody A.

resultsDistinctGroupsA.csv – Jde o seznam clusterů a jejich záznamů z metody A, které vůbec nebyly nalezeny metodou B a to ani částečně. Tedy žádná dvojce duplicit označená metodou B se nevyskytuje v těchto clusterech označených jako duplicitní metodou A. Jednotlivé sloupce obsahují:

- id clusteru A,
- záznamy daného clusteru A.

resultsDistinctGroupsB.csv – Obdobné jako resultsDistinctGroupsA.csv pro metodu B.

resultsWrongMatch.csv – Tento seznam uvádí clustery z metody A a clustery z metody B, které se vzájemně překrývají a to tak, žádný cluster jedné metody není podmnožinou clusteru z druhé metody.

Tato varianta je z hlediska porovnání obou metod tou nejméně přínosnou kromě případu úplně rozdílných clusterů. Každý takový případ značí, že druhá metoda nalezla jen část duplicit první metody a navíc k nim přidala úplně jiné záznamy.

6.2.3 Index pokrytí

Všechny výše uvedené výstupy metody Compare slouží k detailnímu přehledu toho, v čem a jak se výsledky daných dvou metod liší. Pro snadné a rychlé srovnání by však bylo vhodné zavést jakousi míru shodnosti obou metod. Nějaké konkrétní číslo jako je přesnost nebo úplnost. V našem případě nelze říci, jak moc je prvá či druhá metoda přesná nebo úplná. Možné je však určit, jak moc se oba výsledky metod shodují. Proto jsme se přiklonili k výpočtu konkrétního čísla, které by udávalo míru pokrytí clusterů obou metod. Tuto míru jsme nazvali Indexem pokrytí. Jeho výpočet probíhá následovně.

Postupně jsou porovnány všechny výstupní clustery obou metod mezi sebou, přičemž se sleduje, zda a jak moc se clustery první metody pokrývají s clustery druhé metody. Tj. kolik clusterů se shoduje úplně, kolik částečně a kolik vůbec. Konkrétně je každé pokrytí ohodnoceno číslem z intervalu [0,1] ve všech možných případech takto:

1. Dva clustery se úplně shodují - ohodnocení: 1 (To platí i pro clustery velikosti 1. Další body jsou uvažovány pouze pro clustery velikosti alespoň 2.)
2. Cluster v_1 je vlastní podmnožinou clusteru v_2 - ohodnocení: $\frac{|v_1|}{|v_2|}$
3. Existuje-li více podmnožin o různých kardinalitách, pro účely ohodnocení rozhoduje ta největší. Např. pro clustery: $\{3, 4, 5, 6, 7, 8\}_A$ a $\{3, 4, 5\}_B$, $\{6, 7\}_B$ je ohodnocení spočítáno jako 3/6 tedy 0,5.
4. Pokud se dva clustery překrývají (tj. mají neprázdný průnik, ale ani jeden není podmnožinou druhého) je toto ohodnocení nastaveno na nulu.
5. Obdobně pokud pro cluster jedné metody neexistuje cluster druhé metody, který by se s ní shodoval, nebo byl alespoň jeden podmnožinou druhého, je tento případ také ohodnocen nulou.

6. Pokud pro jeden cluster existuje její podmnožina a zároveň se překrývá s dalším clusterem, je překrývající se cluster zanedbán a ohodnocení se řídí výše zmíněným bodem 2 respektive 3 podle množství podmnožin.

Součet těchto ohodnocení je následně vydělen počtem clusterů, jež jsou výsledkem té metody, která jich nalezla více. Přitom jsou do tohoto počtu zahrnuty všechny clustery dané metody kromě těch, které jsou v pořadí druhou, třetí, atd. podmnožinou nějakého clusteru nalezeného druhou metodou. Na příkladu, jež je zobrazen na obrázku 6.1, má metoda A celkem 5 clusterů, metoda B celkem 6 clusterů. Protože ale clustery $\{7, 8\}_B$ a $\{5, 9\}_B$ jsou oba podmnožinou téhož clusteru $\{5, 7, 8, 9\}_A$, je počítáno s metodou B jako by měla clusterů pouze 5. Nyní je tedy zřejmé, že Index pokrytí metody A a B je: $(1 + 1 + 1/2 + 2/3) / 5 = 0{,}63$. Za povšimnutí stojí, že toto číslo poměrně dobře vystihuje povahu obou výsledků metod A a B. Jen ve dvou clusterech se obě metody shodují. V jednom se shodují „napůl“, v dalším ze 2/3 a jeden cluster metody A druhá metoda nenalezla vůbec.

Další vlastnosti Indexu pokrytí
Důležité je uvědomit si, že Index pokrytí neurčuje, zda je jeden algoritmus lepší či horší než ten druhý. To, co toto číslo udává, je vzájemná relativní podobnost dvou deduplikačních metod. I malá hodnota nemusí znamenat, že by byl jeden z algoritmů výrazně špatný. Značí to, jak moc se liší v množství nalezených duplicit a unikátních záznamů.

Index pokrytí má tak svou vypovídací hodnotu a to zejména v případě, kdy při výpočtu na reálných datech není možné určit přesnost a úplnost a kdy je záhodno porovnat výstupy dvou metod mezi sebou. Index pokrytí blížící se k 1 naznačuje, že nalezené clustery duplicit obou metod jsou víceméně shodné (rovnají se, jen výjimečně jsou vlastní podmnožinou clusteru z druhé metody a téměř vůbec se nepřekrývají). Naopak Index pokrytí blížící se k 0 nám říká, že clustery obou metod se zjednodušeně řečeno velmi liší. Jde o úplně jiné clustery nebo se clustery překrývají nebo se jen málokdy jedná o podmnožiny.

6.3 Srovnání metody DataFlux a Robustní deduplikační metody

Nástroj DataFlux neslouží pouze pro úlohy deduplikace, přesto v naší práci jsme se zaměřili pouze na tuto funkčnost. Pro zjednodušení tedy nadále chápejme pojmem DataFlux jako deduplkační metodu používanou v tomto nástroji.

V této sekci se zaměříme na srovnání výsledků naší metody RDM a DataFluxu. Nejprve představíme podstatu hledání duplicit DataFluxem a potom uvedeme srovnání výsledků obou metod.

6.3.1 Metoda DataFlux

V tomto případě se k deduplikaci používá specifický druh metody založené na pravidlech. Pravidla však nepracují s prostými hodnotami atributů ale s tzv. párovacími kódy (matchcodes). Tyto párovací kódy jsou obdobou SOUNDEXu (viz 4.1.9), kde je každý řetězec nahrazen alfanumerickým kódem. Při tvorbě párovacích kódů se však přihlíží i k těmto okolnostem:

- Je využit fonetický základ slova, obdobně jako dělá SOUNDEX.
- Možnost parsování řetězců, čímž je umožněno použití dalších kroků jako např. vyřazení nepodstatných slov (předložky, spojky, tituly před jménem, oddělovače, …)
- Možnost nastavení citlivosti. Při malé citlivosti nebude tolik záležet na rozdílnosti dvou řetězců. (Jejich párovací řetězce se budou shodovat i v případě poměrně velkého množství rozdílů v syntaxi obou řetězců.)
- Možnost využít tzv. znalostní bázi. Jedná se o rozsáhlou sadu datových typů; číselníků pro různá pojmenování, zkratky apod; seznamy nejčastějších překlepů,…

Tvorba pravidel

Při samotné tvorbě pravidel je už jen potřeba nadefinovat, jaké skupiny párovacích řetězů se musí shodovat, aby bylo možné označit dva záznamy za duplicitní. Nejstriktnější pravidlem by bylo, že se musí rovnat všechny matchcody všech příslušných atributů dvou záznamů. K tomu lze ale přidat např. pravidlo pouze na rovnost dvojic matchcodů příjmení a emailu. Takovýchto pravidel je možné vytvářet celá řada. Zároveň je výhodné pro různou citlivost párovacích řetězců použít různá pravidla.

6.3.2 Srovnání

Nyní uvedeme výsledky naší metody RDM a srovnáme je s výsledky DataFluxu pomocí metody Compare. Pro tyto účely jsme použili tři sady testovacích dat:

1. seznam osob (1000 záznamů, z prostředí USA, obsahuje 500 uměle vytvořených duplicitních dvojic záznamů),
2. seznam restaurací (864 záznamů z prostředí USA, reálná data),
3. seznam zákazníků (7321 záznamů z prostředí České republiky, reálná data).

První dvě sady testovacích dat pochází z veřejně dostupného zdroje Riddle repository [56]. Třetí sada dat pochází z databáze zákazníků jednoho českého internetového obchodu. Výsledky pro každý seznam testovacích dat jsou po řadě uvedeny v tabulkách 6.2, 6.3 a 6.4.

DataFlux	RDM	Ukazatel
716	1000	Celkem duplicit
358	500	Počet clusteru (velikosti > 1)
2	2	Průměrná velikost clusteru
358	358	Schodných clusterů
0	142	Počet clusterů nenalezených druhou metodou
0	0	Počet clusterů částečně pokrytých druhou metodou
0	0	Shodných unikátních záznamů
	0,72	Index pokrytí

Tabulka 6.2: Výsledky metody Compare pro seznam osob

DataFlux	RDM	Ukazatel
188	216	Celkem duplicit
91	107	Počet clusteru (velikosti > 1)
2,07	2,02	Průměrná velikost clusteru
79	79	Schodných clusterů
8	24	Počet clusterů nenalezených druhou metodou
0	2	Počet clusterů částečně pokrytých druhou metodou
626	626	Shodných unikátních záznamů
	0,96	Index pokrytí

Tabulka 6.3: Výsledky metody Compare pro seznam restaurací

DataFlux	RDM	Ukazatel
716	799	Celkem duplicit
334	375	Počet clusteru (velikosti > 1)
2,14	2,13	Průměrná velikost clusteru
240	240	Schodných clusterů
71	110	Počet clusterů nenalezených druhou metodou
8	12	Počet clusterů částečně pokrytých druhou metodou
6358	6358	Shodných unikátních záznamů
	0,98	Index pokrytí

Tabulka 6.4: Výsledky metody Compare pro seznam zákazníků

Všimněme si, že na testovacích datech s údaji osob metoda RDM nalezla všech 500 clusterů, zatímco DataFlux pouze 358 (viz tabulka 6.2). To je důsledek použití pouze dvou jednoduchých pravidel. Proto je také index pokrytí v tomto případě jen 0,72. Tento příklad dokazuje i to, že metoda RDM bez obtížného ladění nalezla správný výsledek (dle specifikace Riddle repository [56]), k jehož dosažení by DataFlux vyžadoval náročnější hledání pravidel a ladění síly matchcodů.

Přesto je však z uvedených výsledků zřejmé, že metoda RDM dosahuje velmi podobných výsledků jako metoda DataFlux. To dokazují jednak vypočtené indexy pokrytí (0,72; 0,96; 0,98), které se blíží k 1, tak i počet společně nalezených clusterů a unikátních záznamů.

Přesto se ve výsledcích objevují clustery, které byly nalezené jen jednou z metod. Díky tomu, že výstupem naší metody Compare jsou rovněž seznamy takových clusterů, můžeme je analyzovat. Zde si nyní ukážeme pouze na malém vzorku testovacích dat zákazníků několik příkladů, kde vyniknou přednosti i nevýhody obou metod. Nejprve uvedeme dvojce duplicit, které nalezla RDM, ale DataFlux nikoli (tabulka 6.5) a následně dvojce duplicit nalezené DataFluxem, jež se nepodařilo najít metodě RDM (tabulka 6.6).

Jelikož se jedná o reálná data, jsou uvedené hodnoty z legislativních důvodů záměrně pozměněné. Tyto změny však byly provedeny s ohledem na to, aby zůstala zachována rozdílnost uvedených záznamů z hlediska chyb, překlepů apod.

Cluster	Jméno	Příjmení	Ulice	Město	PSČ	Telefon	Email
1	Karel	Novák	Říční 12/3000	Hornice	52512	555123123	hafhaf@seznam.cz
1	Karel	Novák	Horní 639	Hrmice	51115	555123123	hafhaf@seznam.cz
2	Petr	Roubal	Osobní odběr	Olomouc	67225	666123123	rup@seznam.cz
2	Petr	Roubal	Salaš 12	Salaš 12	67225	666123123	rup@seznam.cz
3	David	Molovice 87			666 123 123	rtst@seznam.cz	David
3	David	Molovice 87	Chrudim	44225	666 123123	rtst@seznam.cz	David

Tabulka 6.5: Páry duplicit identifikované pouze metodou RDM

Popis výsledků metody RDM

V tabulce 6.5 je možné vidět, že v clusteru č. 1 se vyskytuje dvojce poměrně jistých duplicit. Díky tomu, že názvy měst Hornice a Hrmice si jsou velmi podobné a většina ostatních údajů je úplně shodná, mohla vzdálenostní metoda RDM rozhodnout o sloučení. DataFlux vyhodnotil adresu jako odlišnou a tak nesloučil vůbec.

Podobným případem je cluster č. 2, ve kterém pravidla DataFluxu neumožnila sloučení, ale RDM si dokázala s rozdílností v adresních údajích „poradit“ díky stále velké podobnosti obou záznamů a malému počtu jiných záznamů v jejich blízkém okolí.

Při slučování třetího clusteru opět DataFlux neuspěl z toho důvodu, že nebylo použito pravidlo, které by si poradilo s rozdílnou či neuvedenou adresou.

Cluster	Jméno	Příjmení	Ulice	Město	PSČ	Telefon	Email
1	Tomáš	Malý	Teplická 555/12	Praha 6 - Dejvice	15000	888666444,777866866	aaa@seznam.cz
1	Tomáš	Malý	Evropská 2400/5	Praha 6	16600	888666444	aaa@seznam.cz
2	Pan Hora	b	Malá 5	Praha	15000	888666444	bbb@seznam.cz
2	p. Hora	b	Malá 5	Praha	15000	888666444	bbb@seznam.cz
3	pan	Jiří ,HOLEŠ	Na sloupcích	Příbram	15045	999666444	kak@seznam.cz
3	Jiří	HOLEŠ	Na sloupcích	Příbram	15045	999666444	kak@seznam.cz

Tabulka 6.6: Páry duplicit identifikované pouze metodou DataFlux

Popis výsledků metody DataFlux

V případě clusteru 1. se jedná určitě o tutéž osobu. DataFlux oba záznamy sloučil zřejmě i na základě toho, že díky parsingu určil shodu v hodnotách atributů města a telefonu. Metodu RDM tyto textové odlišnosti „zmátly“ natolik, že vyhodnotil podobnost obou záznamů jako nízkou. Ty vyplývá z použité editační vzdálenosti. Viz implementace metody RDM 5.2.

V záznamech clusteru č. 2 je možné spatřit klasický případ, kdy parsing velmi pomohl při vyloučení nepodstatných výrazů „pan“ a „p.“. Díky tomu DataFlux oba záznamy mohl bez problémů sloučit.

V posledním clusteru se vyskytuje velmi podobný případ. Zde parsing nevyloučil pouze nepodstatné výrazy, ale dokonce rozdělil řetězec „Jiří, Holeš“ na jméno a příjmení. Tím DataFlux snadno zjistil, že jde o tutéž osobu. Naproti tomu metoda RDM pouze vyhodnotila malou podobnost v obou tak důležitých atributech jako je jméno a příjmení a oba záznamy tak nemohly být sloučeny.

Z předchozích ukázek je tedy patrné, že obě metody mají své výhody a nevýhody, jež vyplývají z jejich rozdílných konceptů. Nyní kompletně shrneme jaké přednosti má DataFlux oproti RDM a naopak.

Přednosti DataFluxu

Použitá pravidla spolu s matchcody dokáží v DataFluxu popsat velkou škálu chyb a syntaktických rozdílností, které se v sémanticky stejných záznamech mohou vyskytovat. Když se navíc během úlohy unifikace využije procesu standardizace, stává se DataFlux velmi silným nástrojem.

Spárování např. dvojic (pan Novák, ing Novák) nebo (Uh. Hradiště, Uherské Hradiště) není pro DataFlux žádný problém s využitím různých číselníků či jiných znalostních bází podpořených čištěním dat včetně parsingu.

Porozumění datům a jeho využití je zkrátka velkou výhodou DataFluxu oproti naší metodě RDM. Nejen z výše zmíněných příkladů, ale i z detailních seznamů nalezených clusterů metodou Compare (viz soubory na přiloženém CD), je patrné, že některé duplicity dokáže nalézt pouze metoda DataFluxu. Metoda RDM tyto duplicity jen na základě textové podobnosti neodhalí.

Další nemalou výhodou metody DataFlux je rychlý výpočet. Vytvoření matchcodů a nasazení slučovacích pravidel je výpočetně mnohem méně náročné než hledání duplicit metodou RDM, která pro větší objemy dat vyžaduje jejich prvotní rozčlenění do menších částí (tj. clusterování). Navíc takové clusterování vždy zvyšuje počet falešně nesloučených záznamů.

Přednosti RDM

Hlavní výhoda RDM tkví v jednoduchosti implementace a ve snadném použití metody v praxi. I když je samotné použití DataFluxu také poměrně snadné, jeho využití v praxi musí předcházet mnohdy pracné úlohy standardizace. Následuje neméně pracné vytváření slučovacích pravidel spolu se škálováním citlivosti matchcodů. Tento krok navíc mnohdy nestačí provézt pouze jednou. Naopak často je nutné iterativně přidávat či upravovat slučovací pravidla v závislosti na tom, jakých výsledků bylo aktuálně dosaženo.

RDM se vydává jiným směrem, kde podobné nebo stejné hodnoty atributů přibližují z pohledu podobnosti dva záznamy k sobě a rozdílné je oddalují. Tím je možné sloučit dva záznamy, jež mají shodnou nebo velmi podobnou většinu atributů. K tomu není potřeba žádná znalost o datech, což činí tuto metodu velmi univerzální. O tom jsme se mohli přesvědčit během praktické části této práce, kdy byla metoda

RDM použita pro anglické i české prostředí a to bez nutnosti jakýchkoli změn v implementaci, škálování vah atributů nebo získávání nějakých číselníků z jiného národního prostředí.

Další výhodou je „volnost“ při slučování záznamů. DataFlux totiž striktně nesloučí dva záznamy, pokud nevyhovují alespoň jednomu pravidlu. To se může snadno stát, pokud se záznamy liší (např. v hodnotě nějakého důležitého atributu) více, než povoluje citlivost matchcodů. Naopak RDM dokáže na základě své slučovací metodiky, postavené na vlastnostech kompaktnosti a řídkého okolí, sloučit i takovéto záznamy. Navíc metoda RDM nevyužívá jedné slučovací hranice. I to umožňuje slučovat záznamy duplicity, jež se výrazně liší v některé hodnotě atributu. Naopak DataFlux takové duplicity odhalit nemusí, kvůli svým striktním pravidlům.

7 Závěr

V této práci jsme studovali hlavní skupiny deduplikačních metod jak z hlediska jejich implementačních specifik, tak z hlediska jejich výhod a nevýhod. Představili jsme jejich slabiny i přednosti a ukázali, jakou skupinu metod pro praktické použití vybrat v závislosti na daných požadavcích a situacích.

Jedním z cílů této práce byla i implementace deduplikační metody a její porovnání s metodou DataFlux. Pro tyto účely byla implementována Robustní deduplikační metoda (RDM) a pro účely porovnání výsledků metoda Compare. Dále shrneme přednosti těchto metod, zrekapitulujeme výsledky porovnání RDM a DataFluxu a nastíníme další možný vývoj v této oblasti.

Metoda RDM
Námi implementovaná metoda RDM vychází z algoritmu Robust identification of fuzzy duplicates [28]. V několika krocích algoritmu jsme však navrhli vlastní, pro naše účely výhodnější, způsob výpočtu odlišující se od původního konceptu. Patří sem výpočet vlastní vzdálenostní metriky kombinující více vzdáleností. Dále je počítán přesný seznam nejbližších sousedů (nikoli přibližný) a to pomocí T-SQL. To oproti původnímu návrhu metody umožňuje použití libovolné vzdálenostní metriky. I zbytek výpočtu naší metody je proveden v databázi, kde jsme využili jejích výhod, např. indexů nad tabulkami. Provedení většiny výpočtu v databázi nám přineslo výhodu i v tom, že jinak složité kroky s většími objemy dat jsou provedeny velmi efektivně a snadno.

Metoda Compare
Cílem návrhu této metody bylo srovnání dvou deduplikačních metod z hlediska jejich výsledků. S pomocí metody Compare lze snadno zjistit, která z metod nalezla více duplicit, v kolika clusterech, kolik nalezla unikátních záznamů apod. Pro účely srovnání dvou metod slouží výstupní hodnoty, jako jsou počet shodně nalezených unikátních záznamů, počet shodně nalezených clusterů a v neposlední řadě počet clusterů nenalezených druhou metodou. Samozřejmostí je i detailní výpis těchto clusterů včetně záznamů v nich obsažených. Pro rychlé porovnání výstupů dvou metod jsme dále navrhli tzv. Index pokrytí, který udává v jaké míře se shodují clustery nalezené oběma metodami.

Metodu Compare lze výhodně využít i během vývoje deduplikační metody či škálování jejích parametrů, protože lze snadno poměřit zda a v jaké míře se změnily výsledky metody po její úpravě.

Porovnání metody RDM a DataFlux

Pomocí metody Compare jsme provedli několik měření na uměle vytvořených i reálných datech. Na jejich výsledcích se podařilo ukázat, že metoda RDM dokáže dosahovat srovnatelných výsledků jako metoda DataFlux. Tím jsme i ukázali, že se poměrně jednoduchá metoda, která nevyužívá znalostí o zpracovávaných datech, vyrovná složitější metodě DataFlux, která využívá rozsáhlou znalostní bázi. Přesto ze srovnání dále vyplynulo, že každá z obou metod má své výhody a nevýhody a že se v praxi vyskytují duplicity, které dokáže odhalit pouze metoda RDM, ale DataFlux nikoli a naopak.

Další vývoj metody RDM

V naší práci jsme implementovali deduplikační metodu spolu s výpočtem vzdálenostních metrik. Přesto naším hlavním cílem nebylo vytvoření programu, který by byl ihned použitelný v praxi. Šlo nám spíše o předvedení, že poměrně jednoduchý algoritmus, který nevyžaduje znalost domény ani trénovacích dat, se dokáže ve výsledcích velmi dobře srovnávat s jiným algoritmem využívající znalostní bázi.

Pro finální nasazení naší metody do praxe by bylo nutné provézt několik dalších úprav, například zautomatizování výpočtu vzdálenostní funkce a výpočtu samotného hledání duplicit, což je nyní řešeno odděleně (v softwaru SAS a MS SQL). Zároveň by pro velká data bylo nutné využít nějakého sofistikovanějšího clusterování, které by redukovalo množství porovnávaných dvojic záznamů a příliš nezvyšovalo míru vzniku falešně nesloučených záznamů. Dále by bylo výhodné použít i další vzdálenostní metriky a to i s přihlédnutím k daným datům. Samozřejmě je i možné použití dalších kroků využívající znalost dat (čištění, standardzace) podobně jako v DataFluxu. To by ale samozřejmě přineslo i své nevýhody, jak bylo v této práci zmíněno, a je tedy nutné mít na paměti, že je vhodné vydat se vždy tou cestou, která nejlépe vyhovuje dané situaci a požadavkům.

Literatura

[1] Maydanchik A. (2007): Data quality assessment, *Technics Publications, LLC; New Jersey, USA*

[2] Hernández M. A., Stolfo S. J. (1998): Real-World Data Is Dirty: Data Cleansing and the Merge/Purge Problem, *Data Mining and Knowledge Discovery, vol. 2, no. 1*, 9-37

[3] Kimball R. (1998): The Data Warehouse Lifecycle Toolkit : Expert Methods for Designing, Developing, and Deploying Data Warehouses, Wiley, ISBN: 978-0471255475

[4] McCallum A, Freitag D., Pereira F.C.N. (2000): Maximum Entropy Markov Models for Information Extraction and Segmentation, *17th Int. Conf. Machine Learning (ICML '00)*, 591-598

[5] Borkar V.R., Deshmukh K., Sarawagi S. (2001): Automatic Segmentation of Text into Stuctured Records, *Int. Conf. Management of Data (SIGMOD '01)*, 175-186

[6] Levenshtein V.I. (1966): Binary Codes Capable of Correcting Deletions, Insertions and Reversals, *Doklady Akademii Nauk SSSR, 163(4)*, 845-848

[7] Tan P. N., Steinbach M., Kumar V. (2005): Introduction to Data Mining, *Addison-Wesley*, ISBN 0-321-32136-7

[8] Needleman S.B., Wunsch C.D. (1970): A General Method Applicableto the Search for Similarities in the Amino Acid Sequence of Two Proteins, *J. Molecular Biology, vol. 48, no. 3*, 443-453

[9] Newcombe H. B., Kennedy J. M., Axford S. J., James A. P. (1959): Automatic linkage of vital records, *Science 130,* 954–959.

[10] Smith T.F., Waterman M.S. (1981): Identification of Common Molecular Subsequences, *J. Molecular Biology, vol. 147*, 195-197

[11] Jaro M.A.(1976): Unimatch: A Record Linkage System: User's Manual, *technical report, US Bureau of the Census, Washington, D.C.*

[12] Winkler W.E., Thibaudeau Y. (1991): An Application of the Fellegi-Sunter Model of Record Linkage to the 1990 US Decennial Census, *Technical Report Statistical Research Report Series,* RR91/09, US Bureau of the Census, Washington, D.C.

[13] Ullmann J.R. (1977): A Binary n-Gram Technique for Automatic Correction of Substitution, Deletion, Insertion, and Reversal Errors in Words, *The Computer J., vol. 20, no. 2*, 141-147

[14] Gravano L., Ipeirotis P.G., Jagadish H.V., Koudas N., Muthukrishnan S., Pietarinen L., Srivastava D. (2001): Using q-Grams in a DBMS for Approximate String Processing, *IEEE*

[15] Monge A.E., Elkan C.P. (1996): The Field Matching Problem: Algorithms and Applications, *Proc. Second Int'l Conf. Knowledge Discovery and Data Mining (KDD '96)*, 267-270

[16] Cohen W.W. (1998): Integration of Heterogeneous Databases without Common Domains Using Queries Based on Textual Similarity, *Proc. 1998 ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '98)*, 201-212

[17] Bilenko M., Mooney R.J., Cohen W.W., Ravikumar P., Fienberg S.E. (2003): Adaptive Name Matching in Information Integration, *IEEE Intelligent Systems, vol. 18, no. 5*, 16-23

[18] Gravano L., Ipeirotis P.G., Koudas N., Srivastava D. (2003): Text Joins in an RDBMS for Web Data Integration, *Proc. 12th Int'l World Wide Web Conf. (WWW12)*, 90-101

[19] Russell R.C. (1918): Soundex Index, U.S. Patent 1.261.167, http://patft.uspto.gov/netahtml/PTO/srchnum.htm

[20] Taft R.L. (1970): Name Search Techniques, *Technical Report Special Report No. 1, New York State Identification and Intelligence System, Albany, N.Y.*

[21] Gill L.E. (1997): OX-LINK: The Oxford Medical Record Linkage systém, *Proc. Int'l Record Linkage Workshop and Exposition,* 15-33

[22] Breiman L., Friedman J.H., Olshen R.A., Stone C.J. (1984): Classification

and Regression Trees, *CRC Press*

[23] Hastie T., Tibshirani R., Friedman J. H. (2001): The Elements of Statistical Learning, *Springer Verlag*

[24] Zezula P., Amato G., Dohnal V., Batko M. (2006): Similarity Search - The Metric Space Approach

[25] Singla P., Domingos P. (2004): Multi-Relational Record Linkage, *Proc. KDD-2004 Workshop Multi-Relational Data Mining*, 31-48

[26] Dey D., Sarkar S., De P. (1998): Entity Matching in Heterogeneous Databases: A Distance Based Decision Model, *Proc. 31st Ann. Hawaii Int'l Conf. System Sciences (HICSS '98)*, 305-313

[27] Guha S., Koudas N., Marathe A., Srivastava D. (2004): Merging the Results of Approximate Match Operations, *Proc. 30th Int'l Conf. Very Large Databases (VLDB '04)*, 636-647

[28] Chaudhuri S., Ganti V., Motwani R.(2005): Robust Identification of Fuzzy Duplicates, *Proc. 21st IEEE Int'l Conf. Data Eng. (ICDE '05),* 865-876

[29] Fellegi I.P., Sunter A.B. (1969): A Theory for Record Linkage, *J. Am. Statistical Assoc., vol. 64, no. 328*, 1183-1210

[30] Jaro M.A. (1989): Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida, *J. Am. Statistical*

Assoc., vol. 84, no. 406, 414-420

[31] Winkler W.E. (1993): Improved Decision Rules in the Felligi-Sunter Model of Record Linkage, *Technical Report Statistical Research Report Series RR93/12, US Bureau of the Census, Washington*

[32] Ahuja R. K., Magnanti T. L., Orlin J. B. (1993): Network Flows: Theory, Algorithms, and Applications, first ed., Prentice Hall

[33] Galhardas H., Florescu D., Shasha D., Simon E., Saita C. A. (2001): Declarative Data Cleaning: Language, Model, and Algorithms, *Proc. 27th Int'l Conf. Very Large Databases (VLDB '01)*, 371-380

[34] Verykios V. S., Elmagarmid A. K., Houstis E. N. (2000): Automating the Approximate Record Matching Process, *Information Sciences, vol. 126, nos. 1-4*, 83-98

[35] Bansal N., Blum A., Chawla S. (2004): Correlation Clustering, *Machine Learning, vol. 56, nos. 1-3*, 89-113

[36] Cohen W.W., Richman J. (2002): Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integration, *Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '02)*

[37] Sarawagi S., Bhamidipaty A. (2002): Interactive Deduplication Using Active Learning, *Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '02)*, 269-278

[38] Blum A., Mitchell T. (1998): Combining Labeled and Unlabeled Data with Co-Training, *COLT '98: Proc. 11th Ann. Conf. Computational Learning Theory*, 92-100

[39] Monge A.E., Elkan C.P. (1997): An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records, *Proc. Second ACM SIGMOD Workshop Research Issues in Data Mining and Knowledge Discovery (DMKD '97)*, 23-29

[40] Novák V. (1990): Fuzzy množiny a jejich aplikace, matematický seminář SNTL, Praha, ISBN 80-03-00325-3

[41] Chaudhuri S., Dayal U. (1997): An Overview of Data Warehousing and OLAP Technology, ACM SIGMOD

[42] Gorrod M. (2004): Risk Management Systems: Technology Trends (Finance and Capital Markets). Basingstoke: Palgrave Macmillan. ISBN 1-4039-1617-9.

[43] Dinerstein J., Dinerstein S., Egbert P. K., Clyde S. W. (2008): Learning-based Fusion for Data Deduplication, *Seventh International Conference on Machine Learning and Applications*

[44] Navarro G., Baeza-Yates R., Sutinen E., Tarhio J. (2001): Indexing methods for approximate string matching. *IEEE Data Engineering Bulletin*, 24(4):19-27

[45] SAS Character functions manual http://support.sas.com/publishing/pubcat/chaps/59343.pdf

[46] Canada J. R., Sullivan W. G. (1989): Economic and Multiattribute Evaluation of Advanced Manufacturing Systems, *Prentice-Hall, Englewood Cliffs, NJ*

[47] Gass S. (1985): A Process to Determine Priorities and Weights for Large-Scale Linear Goal Programming, *Proceedings of the 12th International Symposium on Mathematical programming, Boston*

[48] UIR-ADR, Územně identifikační registr adres, http://forms.mpsv.cz/uir/

[49] Kopcke H., Rahm E. (2009): Frameworks for entity matching: A comparison, *Data Knowl. Eng. Doi: 10.1016/h.datak.2009.10.003*

[50] Zhao H., Ram S. (2005): Entity identification for heterogeneous database integration, *Inf. Syst. 30 (2) (2005)*, 119-132

[51] Elfeky M. G., Elmagarmid A. K., Verykios V. S (2002): TAILOR: a record linkage toolbox, *Proceedings of the 18th Int. Conf. On Date Engineering (ICDE '02)*, 17 - 28

[52] Christen P. (2008): FEBRL: a freely available record linkage system, Proceedings of the second Australasian workshop on Heath Data and Knowledge management, 17 – 25

[53] Kopcke H., Rahm E. (2008): Training selection for tuning entity matching, *Proceedings in the Sixth Int. Workshop on Quality in Databases and Management of Uncertain data*, 3 – 12

[54] Waterman M.S., Smith T. F., Beyer W.A. (1976): Some Biological Sequence Metrics, *Advances in Math., vol. 20, no. 4*, 367-387

[55] Wang X. Z. (1999): Data mining and knowledge discovery for process monitoring and control, Springer, London.

[56] M. Bilenko. RIDDLE: Repository of information on duplicate detection, record linkage, and identity uncertainty, http://www.cs.utexas.edu/users/ml/riddle/index.html

Příloha A

Seznam různých zápisů města „Rožnov pod Radhoštěm“ pro účely standardizace a deduplikace adresních bodů.

ROŽNOV
ROŽNOV OD RADH.
ROŽNOV OD RADHOŠTĚM
ROŽNOV P OD RADHOŠTĚ
ROŽNOV P R.
ROŽNOV P RADH.
ROŽNOV P RADHOŠTĚM
ROŽNOV P. R
ROŽNOV P. R-
ROŽNOV P. R,
ROŽNOV P. R.
ROŽNOV P.- R.
ROŽNOV P. R..
ROŽNOV P. RAD.
ROŽNOV P. RADH
ROŽNOV P. RADH-
ROŽNOV P. RADH.
ROŽNOV P. RADHOSTĚM
ROŽNOV P. RADHOŠTEM
ROŽNOV P. RADHOŠTĚM
ROŽNOV P. RADHOŠŤEM
ROŽNOV P. RADHOŠTĚM0
ROŽNOV P.. R.
ROŽNOV P.R
ROŽNOV P.R-
ROŽNOV P-.R-
ROŽNOV P.R,
ROŽNOV P.R.
ROŽNOV P.R.;
ROŽNOV P.RAD
ROŽNOV P.RADH
ROŽNOV P.RADH.
ROŽNOV P.RADH..
ROŽNOV P.RADHO
ROŽNOV P.RADHOŠTĚM
ROŽNOV P.ROŽNOV
ROŽNOV P/R
ROŽNOV P/R.
ROŽNOV P/RAD.
ROŽNOV P/RADH.
ROŽNOV P/RADHO
ROŽNOV P/RADHOŠTĚM
ROŽNOV P:R.
ROŽNOV PD RADH.
ROŽNOV PDO RADHOŠTĚM
ROŽNOV PDOD RADHOŠTĚM
ROŽNOV PO RADHOŠTĚM
ROŽNOV POD
ROŽNOV POD ADHOŠTĚM
ROŽNOV POD EADHOŠTĚM
ROŽNOV POD R.
ROŽNOV POD RA
ROŽNOV POD RAD
ROŽNOV POD RAD.
ROŽNOV POD RADH
ROŽNOV POD RADH,
ROŽNOV POD RADH.
ROŽNOV POD RADH.,
ROŽNOV POD RADHO3T2M
ROŽNOV POD RADHOĚTĚM
ROŽNOV POD RADHOSTĚM
ROŽNOV POD RADHOŠ´TEM
ROŽNOV POD RADHOŠEM
ROŽNOV POD RADHOŠTĚ
ROŽNOV POD RADHOŠTEM
ROŽNOV POD RADHOŠTĚM
ROŽNOV POD RADHOŠŤEM
ROŽNOV POD RADHOŠŤĚM
ROŽNOV POD RADHOŠTĚM 1
ROŽNOV POD RADHOŠTĚM 3
ROŽNOV POD RADHOŠTĚM
OKR. VSETÍN
ROŽNOV POD RADHOŠTĚM,
CZ
ROŽNOV POD RADHOŠTĚM;
ROŽNOV POD RADHOŠTĚM0
ROŽNOV POD RADHOŠTĚN
ROŽNOV POD RADHOŠTLĚM
ROŽNOV POD RADHOŠTM
ROŽNOV POD RADHOŠTTĚM
ROŽNOV POD RADHOTĚM
ROŽNOV POD RADHOTŠĚM
ROŽNOV POD RADNOŠTĚM
ROŽNOV POD RADOŠTĚM
ROŽNOV POD RADSHOŠTĚM
ROŽNOV POD RAHOŠTĚM
ROŽNOV POD RDAHOŠTĚM
ROŽNOV POD RDHOŠTĚM
ROŽNOV POD. RADH.
ROŽNOV POD. RADHOŠTĚM
ROŽNOV POD. RO-
DAHOŠTĚM
ROŽNOV POD.RADH.
ROŽNOV POD.RADHOŠTĚM
ROŽNOV PODRADHOŠTĚM
ROŽNOV POPD RADH.
ROŽNOV POSD RADHOŠTĚM
ROŽNOV PPOD RADH.
ROŽNOV PPOD RADHOŠTĚM
ROŽNOV PR.
ROŽNOV P-R.
ROŽNOV.P.RADH.
ROŽNOV/RADHOŠT
ROŽNOV/RADHOŠTĚM
ROŽNOVA POD RADHOŠTĚM
ROŽNOVB POD RADHOŠTĚM
ROŽNOVP OD RADH.
ROŽNOVP POD RADH.
ROŽNOVP. R.
ROŽNOVP.R.
ROŽNOVPOD RADH
ROŽNOVPOD RADH.
ROŽNOVPOD RADHOŠTĚM
ROŽNOVVPOD RADHOŠTĚM