VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

POLOAUTOMATICKÁ NORMALIZACE SLOV Z MATRIČNÍCH ZÁZNAMŮ

SEMI-AUTOMATIC WORD NORMALIZATION IN PARISH RECORDS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DAVID HŘÍBEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAROSLAV ROZMAN, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Hříbek David**
Program: Informační technologie
Název: **Poloautomatická normalizace slov z matričních záznamů**
Semi-Automatic Word Normalization in Parish Records
Kategorie: Umělá inteligence

Zadání:

1. Nastudujte metody shlukové analýzy. Nastudujte matriční záznamy.
2. Na základě nastudovaných znalostí navrhněte metodu pro roztřídění slov objevujících se v matričních záznamech a jejich následnou ruční normalizaci.
3. Vybranou metodu implementujte jako součást genealogického webu.
4. Proveďte otestování úspěšnosti shlukování na různých typech slov (mužská a ženská jména, příjmení, povolání a obce) a dále na různých dobách zápisu (stará čeština, latina, němčina, moderní čeština).
5. Výsledky zhodnoťte a navrhněte případná vylepšení.

Literatura:

- Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall Press, Upper Saddle River, NJ, USA, 2009

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Rozman Jaroslav, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

V této práci je řešeno rozšíření webové aplikace DEMoS pro správu matričních záznamů o možnost normalizace (přiřazení normalizované podoby zápisu jednotlivým slovům) jmen, příjmení, povolání, obcí a dalších typů slov, která se vyskytují v matričních záznamech. V řešení byl použit proces detekce duplicitních záznamů, který umožnil roztřídění slov z matričních záznamů do shluků podobných slov. Díky vzniklým shlukům bylo následně možné sdílet normalizované varianty slov v rámci těchto shluků. Aplikace DEMoS tak pro uživatelem zadaná slova navrhuje normalizované varianty použité nejen u stejných slov, ale i u podobných slov. V rámci této práce bylo navrženo automatické testování úspěšnosti shlukování slov. Celkem bylo pro každý typ slov otestováno 640 různých kombinací parametrů shlukování. Následně byly pro každý typ slov vybrány nejlepší parametry shlukování. Díky normalizaci slov je v aplikaci DEMoS výrazně zvýšena efektivita vyhledávání matričních záznamů. Záznamy jsou také lépe čitelné.

Abstract

This work deals with the extension of DEMoS web application for the management of parish records by the possibility of normalization (assignment of a normalized form of writing to individual words) of names, surnames, occupations, domiciles and other types of words occurring in parish records. In the solution, a duplicate record detection process was used, which allowed sorting of the records from parish records into clusters of similar words. As a result of the clustering, it was possible to share normalized word variants within these clusters. Thus, DEMoS suggests normalized variants for words entered by users, used not only for the same words, but also for similar words. In this work, automatic testing of word clustering was proposed. In total, 640 different combinations of clustering parameters were tested for each word type. Subsequently, the best clustering parameters were selected for each word type. By normalizing words, DEMoS application significantly increases the efficiency of searching in parish records. Records are also easier to read.

Klíčová slova

matriční záznamy, porovnání dat, odstranění duplicit, normalizace, detekce duplicit, vyhledávání, DEMoS

Keywords

parish records, data-matching, deduplication, normalization, duplicate detection, searching, DEMoS

Citace

HŘÍBEK, David. *Poloautomatická normalizace slov z matričních záznamů.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

Poloautomatická normalizace slov z matričních záznamů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Rozmana, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
David Hříbek
7. května 2019

Poděkování

Rád bych poděkoval svému vedoucímu práce Ing. Jaroslavu Rozmanovi, Ph.D. za pomoc a ochotu při vedení mé bakalářské práce. Dále pak panu Ing. Marku Žákovi za spolupráci při nasazení webové aplikace na server.

Obsah

Kapitola 1

Úvod

V dnešní době je velice populární obor genealogie, který zkoumá vztahy mezi lidmi vyplývající z jejich původu. Nejpoužívanějším pramenem, který genealogové používají, jsou matriky. Matriky se v současnosti skenují a jsou tak snadněji dostupné veřejnosti. Lidé proto nemusí jezdit do archivu, díky čemuž se ještě více zvyšuje zájem o genealogii. Pro snadný přepis matričních záznamů do elektronické podoby byla vytvořena webová aplikace DEMoS, která oproti jiným aplikacím umožňuje přepis po jednotlivých záznamech, namísto celých matrik. Aby měla tato aplikace co nejširší využití, byl zvolen způsob přepisu formou transliterace (přímý přepis). Matriční záznamy jsou tedy přepsány tak jak jsou, bez jakýchkoliv úprav. I přesto, že vyhledávání mezi matričními záznamy je klíčovou funkcí aplikace DEMoS, kvůli častým chybám a různým podobám zápisu téže slov není vyhledávání mezi takovými záznamy příliš efektivní. Záznamy jsou také hůře čitelné. Tyto důvody vedle ke vzniku této práce.

V této práci je tedy řešeno rozšíření již existující webové aplikace DEMoS, o možnost normalizace jmen, příjmení, povolání, obcí a dalších slov, která se vyskytují v matričních záznamech. Normalizací slov je myšleno přiřazení normalizované varianty zápisu jednotlivým slovům. Normalizovaná varianta zápisu by měla odpovídat současné podobě zápisu daného slova. Například pro jména Josephus a Joseph je dnešní podoba zápisu Josef. Aby bylo možné efektivněji navrhovat normalizované varianty pro jednotlivá slova, byla slova z matričních záznamů automaticky roztříděna do shluků podobných slov, což umožnilo sdílení normalizovaných variant slov v těchto shlucích. Díky normalizaci je zvýšena efektivita vyhledávání a čitelnost matričních záznamů.

Celá práce je rozdělena do tří kapitol. V kapitole „Studium problematiky“ jsou popsány teoretické znalosti, které bylo nutné nastudovat pro vytvoření této práce. Je zde popis matričních záznamů, jejich vznik, struktura, formy zápisu, rozdělení do kategorií a problémy s kvalitou těchto záznamů. Dále jsou zde popsány metody shlukové analýzy a proces roztřídění databázových záznamů do shluků. V kapitole s názvem „Analýza požadavků a návrh řešení“ je nejdříve detailněji popsán cíl práce. Ve zbytku kapitoly je navrženo schéma databáze pro uložení normalizovaných variant slov a dále navržen obecný princip normalizace slov. V poslední kapitole „Implementace a testování“ je popsána implementace navrženého principu normalizace slov. Nejdříve je vysvětlena implementace programu pro roztřídění slov do shluků, následně pak integrace tohoto programu do webové aplikace DEMoS, včetně implementace webového rozhraní pro normalizaci slov v této aplikaci. Konec této kapitoly je věnován testování úspěšnosti shlukování pro různé typy slov (mužská a ženská jména, příjmení, povolání, obce a další.) a dále různé doby zápisu (stará čeština, latina, němčina, současná čeština).

Kapitola 2

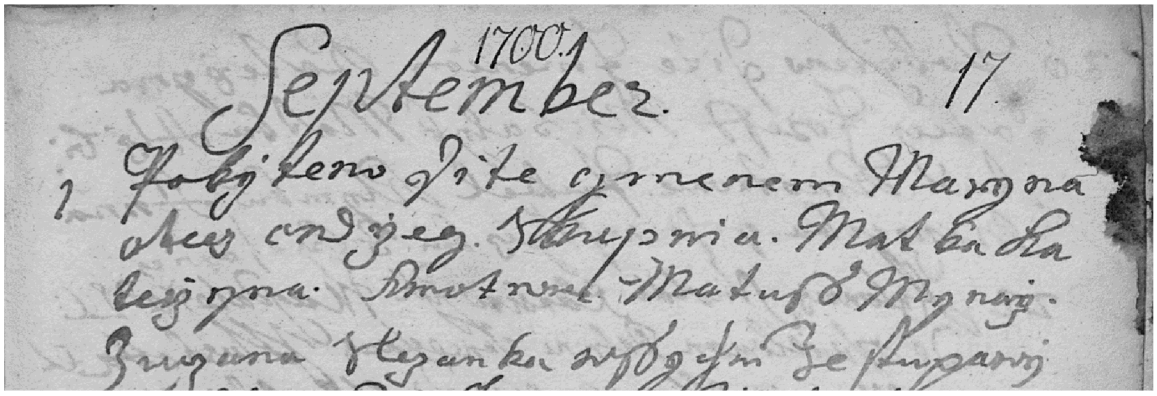
Studium problematiky

Tato kapitola se věnuje studiu informací, které bylo nutné nastudovat pro vytvoření této práce. V první části budou popsány matriční záznamy, způsob jejich získání, struktura, kvalita a jejich digitální reprezentace. Druhá část je věnována studiu shlukové analýzy. Bude zde popsáno dělení shlukové analýzy do podkategorií a popis jednotlivých podkategorií. Jako poslední bude vysvětlen proces roztřídění záznamů do shluků, který je rozdělen do pěti částí, které na sebe navazují.

2.1 Matriky a matriční záznamy

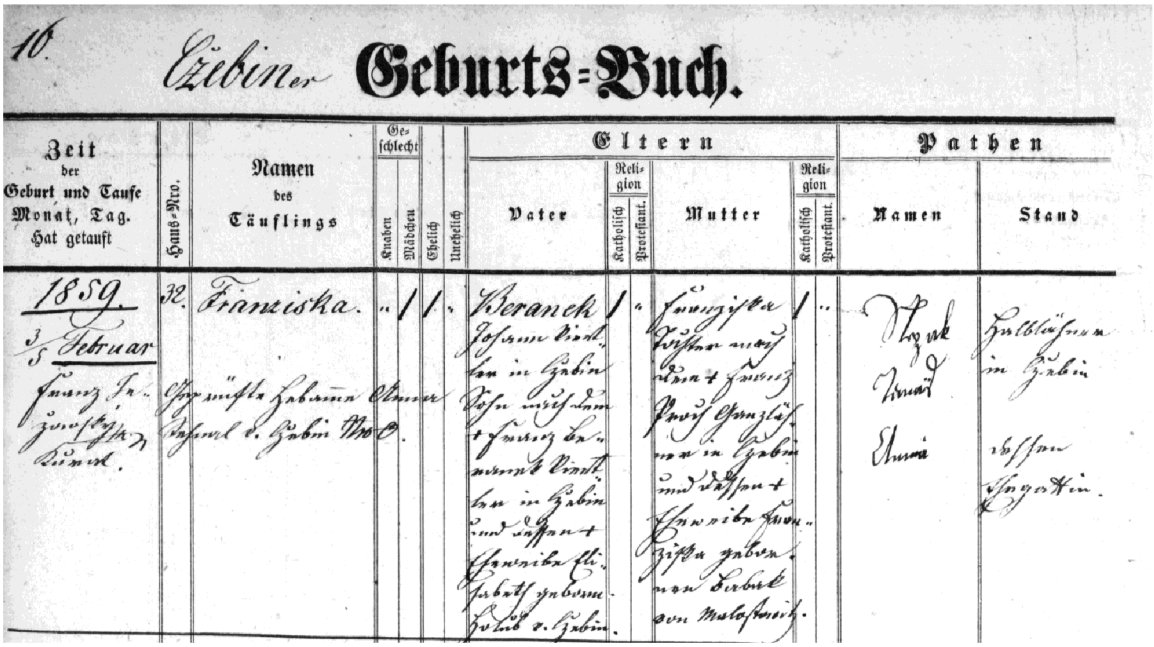
Matrika [3], je veřejně dostupný úřední dokument, který obsahuje informace o obyvatelích. V České republice jsou do matrik ukládány informace o narozeních, sňatcích, registrovaných partnerstvích a úmrtích občanů. Matriční knihy jsou vedeny na matričních úřadech.

První matriky pravděpodobně vznikly již ve středověku. Jednalo se o seznamy šlechty, které zahrnovaly především povinnosti vůči panovníkovi (např. dluhy v podobě dávek nebo počtů ozbrojenců). Později se začaly zaznamenávat i seznamy studentů na univerzitách. Se stále narůstající byrokracií rostl i počet matričních knih. Vznikly matriky kostelů, jmen, narození, úmrtí, sňatků atd. Každá matrika se vztahuje k určitému místu (např. obec, město, kostel).



Obrázek 2.1: Ukázka matričního záznamu zapisovaného nestrukturovaně. Tento naskenovaný záznam byl následně přepsán členy Masarykovy univerzity do aplikace DEMoS.

Matriční knihy původně nebyly strukturovány. Jednotlivé záznamy byly zapisovány jako souvislé věty za sebou. Později byly matriky organizovány ve formě tabulek, kde každý sloupec má svůj význam. Tyto tabulky byly často předtištěny. Na obrázku 2.1 je zobrazen matriční záznam zapisovaný nestrukturovaně. Na obrázku 2.2 je zobrazen strukturovaný zápis matričního záznamu. Matriční záznamy [14] obsahují životopisné údaje, a jsou proto důležité pro genealogický výzkum. Záznamy jsou do matričních knih zapisovány chronologicky.



Obrázek 2.2: Ukázka zápisu matričního záznamu pomocí tabulky. Jedná se o záznam z roku 1859 zapsaný německým jazykem. Tento naskenovaný záznam byl následně přepsán členy Masarykovy univerzity do aplikace DEMoS.

Rozdělení matrik do kategorií

- Matrika křtu a narození (Matrica baptisatorum – Geburts und Tauf Register).
- Matrika oddaných (Matrica copulatorum – Trauungsbuch).
- Matrika úmrtí (Matrica mortuorum – Sterbebuch).

Každá z těchto kategorií [14] obsahuje většinou následující informace. Matrika křtu a narození obsahuje *jméno, příjmení, místo a adresu narození, datum křtu a narození novorozence*, dále *jména, příjmení, místo narození a povolání rodičů*. Matrika oddaných obsahuje *jména, povolání a místa narození* oddávaného páru a rodičů oddávaného páru. Matrika úmrtí obsahuje *jméno, číslo domu, datum a příčinu smrti*. Jednotlivé matriky mohou obsahovat navíc i jiné údaje.

Kvalita záznamů

Náležitosti záznamů [14] se časem měnily. Každý záznam mohl být zapsán jiným člověkem, tudíž se obsah záznamů může lišit. Některé zápisy mohou být více obsáhle nebo méně čitelné

než jiné. Často se zde také vyskytují chyby (např. Pettr, Vyktor) nebo různé varianty zápisu slov téhož významu (např. město Znojmo: Znojmo, Znoimenses, Znoyma). Velkou roli hraje také doba zápisu záznamu. Na území České republiky jsou nejstarší záznamy zapisovány starou češtinou (Zbřezyna) nebo latinou (Brženzinensis, Brženzinio), později pak němčinou (von Brženyna) nebo současnou češtinou (Březina). Záznamy také obsahují nadbytečná slova jako jsou například předložky, které navíc nemusí být odděleny od slova před kterým se nachází (zBřeziny). Všechny tyto a další skutečnosti ztěžují čitelnost záznamů a jejich prohledávání.

Digitalizace

Dnešní matriky mohou být digitalizovány a uloženy v elektronické databázi. Matriční knihy jsou nejprve naskenovány a poté přepsány do textové podoby, se kterou pak lze dále pracovat. Pro přepis záznamů slouží například již zmíněna webová aplikace DEMoS. Příklady naskenovaných matričních záznamů jsou zobrazeny na obrázcích 2.1 a 2.2. Přepis údajů z takto naskenovaných matričních záznamů může probíhat ručně (každý záznam je přečten člověkem, který jej následně přepíše) nebo může být použita metoda optického rozpoznávání znaků (OCR[1]). Metoda OCR je rychlejší, ale také více zatížená chybami než ruční přepis. V obou případech mohou vzniknout chyby v přepsaných datech.

Díky digitální reprezentaci matrik máme možnost mezi matričními záznamy snadněji vyhledávat a třídit je. Aby bylo vyhledávání a další operace nad těmito záznamy co možná nejefektivnější, je nutné počítat s již zmíněnými nepřesnostmi a variacemi v zapisovaných údajích.

2.2 Shluková analýza

V této části budou popsány obecné principy shlukové analýzy a její dělení do kategorií. Dále bude uvedeno využití shlukové analýzy v praxi.

Shluková analýza patří do oblasti umělé inteligence zabývající se strojovým učením. Strojové učení se dělí do tří částí, které se nazývají:

- Učení s učitelem (Supervised learning).
- Učení bez učitele (Unsupervised learning).
- Posilované/motivované učení (Reinforcement learning).

Shluková analýza [10][11][18] spadá do částí Učení bez učitele. Cílem shlukové analýzy je odhalení skryté struktury a vzorů v neoznačených datech jejich uspořádáním do smysluplných skupin (shluků). Objekty ve stejném shluku si musí být navzájem co možná nejvíce podobné a zároveň objekty, které patří do odlišných shluků si nesmí být příliš podobné. Proces vytváření shluků se nazývá shlukování.

Podobnost objektů je určena na základě hodnot jednotlivých atributů objektů. Podobnost těchto atributů se určuje pomocí *funkce podobnosti* nebo *funkce vzdálenosti* (čím více jsou si atributy podobné, tím menší je jejich vzdálenost a větší je jejich podobnost). Většinou se nejedná o vzdálenost v euklidovském prostoru, proto nemusí platit trojúhelníková nerovnost. Funkce pro výpočet podobnosti řetězců budou popsány v kapitole č. 2.3.3.

[1]OCR - https://en.wikipedia.org/wiki/Optical_character_recognition

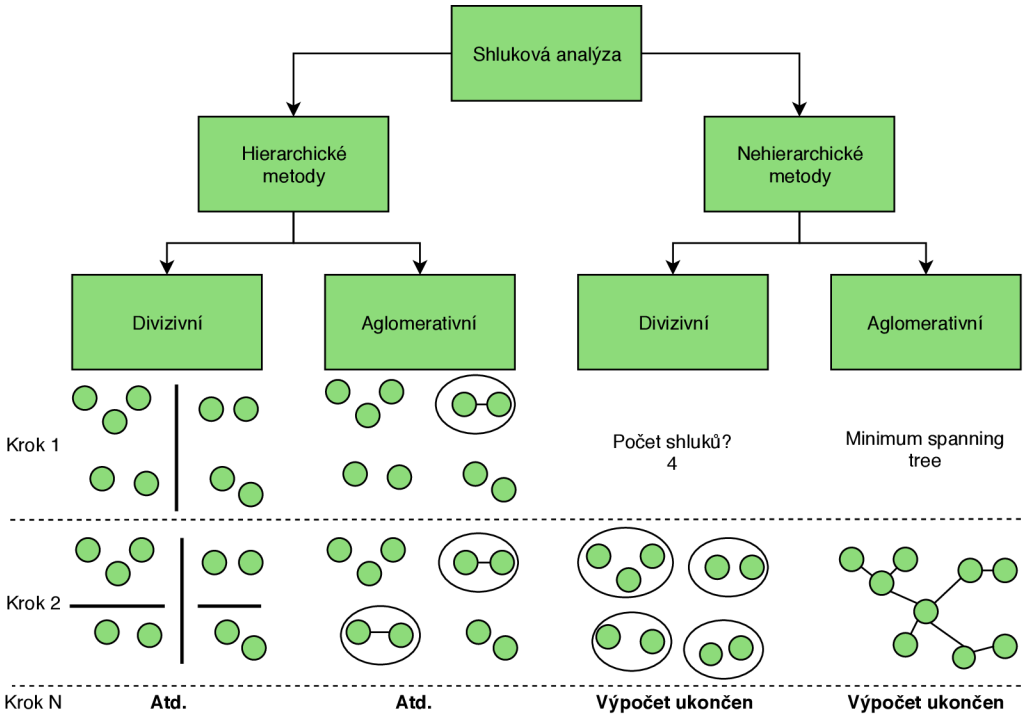
Shlukování je přirozená činnost, kterou provádíme denně — třídíme objekty kolem nás do skupin na základě své podobnosti. Existují skupiny, které je možné přesně definovat (např. rozdělení lidí podle pohlaví na muže a ženy). Pro některé skupiny objektů nemusí existovat jednoznačné rozdělení do shluků (např. roztřídění lidí podle velikosti na velké a malé nebo roztřídění seznamu nepřesně zapsaných příjmení do skupin se stejným příjmením).

Nejzákladnější využití shlukové analýzy je shlukování bodů v prostoru. Tyto body jsou rozděleny do skupin (shluků) podle své vzdálenosti. Tedy body, které leží blízko u sebe budou tvořit jeden shluk.

Shlukovou analýzu je například také možné využít při procesu získávání informací z dat. Před samotným získáváním informací z dat je nad těmito daty provedeno shlukování. Objekty ve vzniklých shlucích jsou poté chápány jako totožné a jsou dále reprezentovány jednou hodnotou. Tímto je zmenšen objem vstupních dat a umožněna efektivnější práce s těmito daty. Proces shlukování záznamů databází je popsán detailněji v kapitole č. 2.3. Tento proces bude použit pro shlukování slov z matričních záznamů. Shlukování záznamů se také používá v oblasti zdravotnictví (shlukování záznamů o pacientech), marketingu (shlukování záznamů o potenciálních zákaznících – každý zákazník je kontaktován pouze jednou), online nakupování (shlukování záznamů o produktech, srovnání cen stejných produktů) atd.

Základní dělení metod shlukové analýzy

Metody shlukové analýzy [9] se dělí do dvou základních skupin: *Hierarchické* a *Nehierarchické* metody. Každá z těchto dvou skupin dále umožňuje divizivní nebo aglomerativní přístup. Dělení metod shlukové analýzy i s příklady je podrobněji zobrazeno na obrázku 2.3.

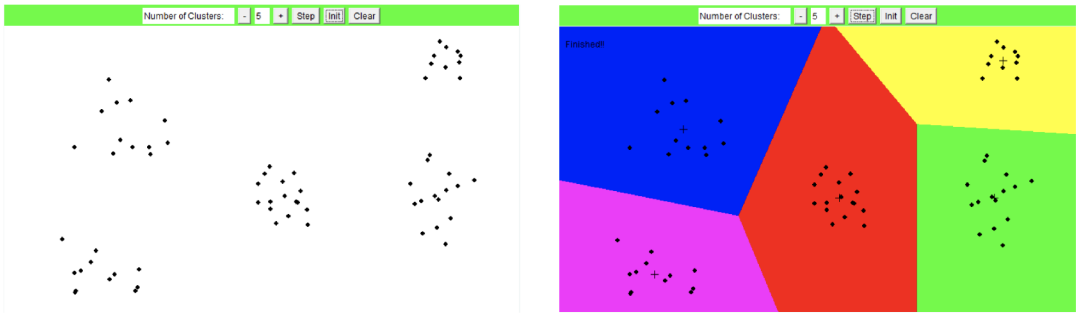


Obrázek 2.3: Dělení metod shlukové analýzy (obrázek převzat a upraven z internetové stránky [9]).

Nehierarchické metody shlukování

Nehierarchické metody shlukování [12] nevytvářejí hierarchickou strukturu, nepracují se stromovou strukturou dat. Tyto metody rozkládají množinu objektů do podmnožin pomocí předem zvoleného kritéria. Snaží se najít jediný rozklad dané množiny objektů. Výběr kritéria záleží na struktuře shlukovaných dat. Nehierarchické metody dělíme na metody s pevným počtem shluků (divizivní) a metody, které počet shluků mění během shlukování (aglomerativní).

Jednou z nejznámějších a nejčastěji používaných metod nehierarchického divizivního shlukování je metoda **K-Means** [8][10][18]. Cílem této metody je rozdělit soubor objektů v n-dimenzionálním prostoru do předem stanoveného počtu shluků `K`. Algoritmus najde takové rozdělení objektů do shluků, aby suma čtverců vzdálenosti centroidů shluků a objektů v těchto shlucích byla co nejmenší. V prvním kroku je náhodně zvoleno `K` těžišť (centroidů). Všechny objekty jsou poté přiřazeny k nejbližším centroidům. V dalším kroku je nejdříve přepočítáno těžiště u všech shluků a následně jsou znovu roztříděny objekty podle vzdálenosti k jednotlivým centroidům. Tento postup se opakuje tak dlouho, dokud nejsou objekty ve shlucích ustáleny. Výsledné řešení záleží na volbě počátečních centroidů a nemusí být proto optimální. Ukázka použití metody K-Means je zobrazena na obrázku 2.4.



Obrázek 2.4: Ukázka použití algoritmu K-Means pro shlukování objektů v prostoru. V levé části jsou zobrazeny objekty před shlukováním. V pravé části jsou zobrazeny objekty po shlukování. Prostor je barevně rozdělen podle vzdálenosti k výsledným centroidům. Obrázky převzaty z prezentace [18].

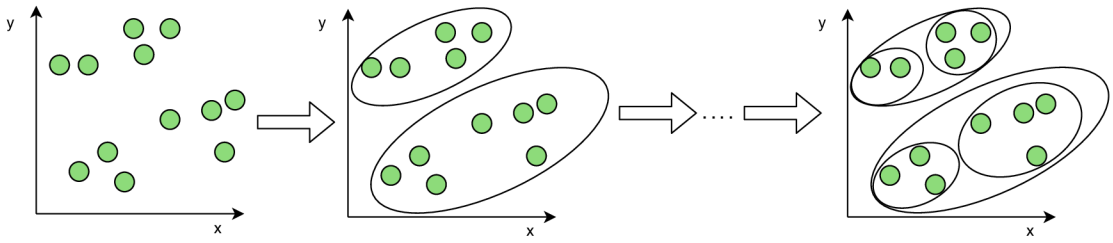
Hierarchické metody shlukování

Hierarchické metody [2][10] shlukování jsou založeny na hierarchickém rozkladu množiny objektů. Výsledkem je strom shluků. Na nejnižší úrovni stromu tvoří každý objekt vlastní shluk. Na nejvyšší úrovni stromu tvoří všechny objekty jeden shluk. Existují dva základní přístupy těchto metod – aglomerativní a divizivní.

Aglomerativní metody spojují navzájem nejpodobnější objekty do shluků. Na začátku každý objekt tvoří vlastní shluk. Postupně spojujeme nejvíce podobné shluky do větších shluků. Proces spojování shluků se opakuje tak dlouho, dokud nezůstane pouze jeden shluk obsahující všechny objekty (nejvyšší úroveň). Jedná se tedy o přístup „zdola nahoru“.

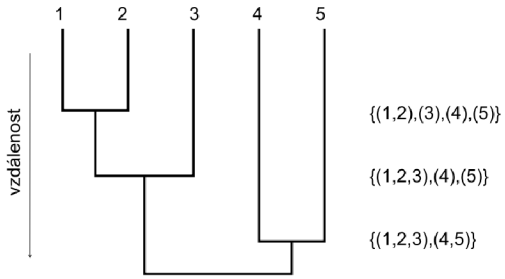
Divizivní metody pracují opačným způsobem, tedy „shora dolů“. Na počátku všechny objekty tvoří jeden shluk, který je rozdělen do dvou menších shluků. Každý nově vzniklý shluk je poté rozdělen na další, menší shluky. Princip divizivních metod je zobrazen na obrázku 2.5.

V praxi se častěji využívá aglomerativní přístup. V obou případech je nutné vybrat takovou úroveň rozkladu množiny objektů, která nejvíce vyhovuje očekávaným výsledkům.



Obrázek 2.5: Ukázka principu fungování hierarchických divizivních metod shlukování. Nejdříve všechny objekty tvoří jeden shluk. Poté v každém dalším kroku dojde k rozdělení existujících shluků do dvou menších shluků. Obrázek převzat a upraven z internetové stránky [1].

Výsledky hierarchického shlukování je možné zobrazit pomocí dendrogramu [11], jehož ukázka je zobrazena na obrázku 2.6. Jedná se o druh diagramu, který umožňuje znázornit jednotlivé kroky hierarchické shlukové analýzy.



Obrázek 2.6: Ukázka dendrogramu. Jedná se o binární strom, kde každý uzel tohoto stromu tvoří samostatný shluk. Shluky jsou spojovány podle vzdálenosti, která je zobrazena na vertikální ose. Pomocí dendrogramu lze zobrazit jednotlivé kroky hierarchické shlukové analýzy a tím nelézt optimální stupeň rozkladu. Obrázek převzat z [11].

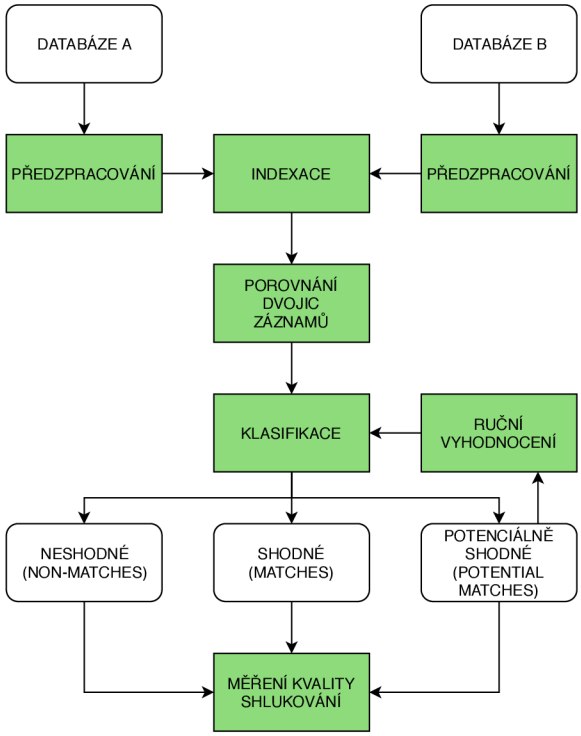
2.3 Proces roztřídění záznamů do shluků

Tato část bude věnována využití shlukové analýzy při procesu roztřídění záznamů do shluků. Bude zde čerpáno z knihy [6] od autora Petera Christena, která vychází z více než 300 článků na toto téma a shrnuje tak celou tuto problematiku. Informačním zdrojem této kapitoly je tato kniha a citované články.

Podle dělení shlukové analýzy, zobrazené na obrázku 2.3, lze tento typ shlukování zařadit do kategorie nehierarchického aglomerativního shlukování. Nevytváří se totiž hierarchická struktura dat a není ani předem znám počet vzniklých shluků.

Proces roztřídění záznamů do shluků se nazývá „Porovnávání dat“ (*Data Matching*) nebo také „Propojení záznamů“ (*Record Linkage*). Cílem tohoto procesu je identifikovat ve

dvou odlišných databázích záznamy, které odpovídají stejné entitě (objektu reálného světa). Na obrázku 2.7 je zobrazeno schéma tohoto procesu.



Obrázek 2.7: Schéma procesu roztřídění záznamů dvou odlišných databází do shluků. Tento proces lze použít také pro detekci duplicit v rámci jedné databáze. Obrázek převzat a upraven z knihy [6].

Speciálním případem „Porovnávání dat“ je detekce duplicit (*Duplicate detection*), kdy jsou hledány duplicitní záznamy pouze v rámci jedné databáze (jednoho zdroje dat). Pokud jsou nalezené duplicitní záznamy nahrazeny jedním záznamem, jedná se o proces deduplikace (*Deduplication*).

Ve zbytku této kapitoly bude vysvětlen procesu roztřídění záznamů do shluků. Tento proces se skládá z pěti částí. První část se zabývá předzpracováním dat. Budou zde uvedeny problémy s kvalitou dat a způsob jejich řešení. V druhé části jsou uvedeny časové složitosti porovnávání záznamů a metody jejich snížení. Dále pak možnost fonetického kódování řetězců, které představují jména nebo názvy. Třetí část popisuje samotné porovnávání záznamů. Jsou zde popsány funkce pro porovnávání řetězců, jejich časové složitosti a možnosti využití. Čtvrtá část se zabývá klasifikací porovnaných záznamů do tříd a vytvořením shluků podobných záznamů. Poslední část procesu roztřídění záznamů do shluků je zaměřena na měření kvality shlukování.

2.3.1 Předzpracování záznamů

Předzpracování záznamů [6] (*Data Pre-Processing*) je důležitou součástí procesu roztřídění záznamů do shluků. Data, která mají být shlukována se mohou lišit ve formátu a struktuře. Mohou také obsahovat chyby a nadbytečná slova, která neobsahují relevantní informace pro shlukování. Například názvy obcí ’von Strzemeniczko’ a ’z Bukovinky’ obsahují předložky,

které je možné odstranit, protože nejsou důležité pro shlukování. V latinském záznamu 'cum sua conjuge' neboli „s jeho chotí" je důležité pouze slovo 'conjuge'.

To, jakým způsobem byla data pořízena také ovlivňuje jejich kvalitu. Data mohla být přepisována z hůře čitelných zdrojů nebo naskenována a převedena do textové podoby pomocí optického rozpoznávání znaků. Chyby při zaznamenávání dat také nastávají pokud jsou data diktována.

Z těchto důvodů je nutné před začátkem samotného shlukování data pročistit a standardizovat. Pročištěním je zvýšena efektivita shlukování. Proces čištění dat je znám jako *Data Cleansing*. V následujícím seznamu jsou uvedeny příklady čištění dat:

- **Kódování znaků**. Převod znaků z kódování UNICODE[2] do kódování ASCII[3]. Pro každý znak v kódování UNICODE je nalezen nejvíce podobný znak v kódování ASCII. Tento krok je důležitý, pokud data mohou obsahovat znaky z cizích jazyků.

- **Jednotná velikost písmen**. Informaci o velikosti písmen je možno při shlukování zanedbat. Tato informace není při shlukování rozhodující. Je proto vhodné změnit velikost písmen na velká nebo malá písmena.

- **Nadbytečné znaky**. Znaky, které nejsou důležité pro shlukování (nepřináší žádnou novou informaci pro shlukování) je nutné odstranit. Mezi takové znaky patří například všechny nealfanumerické[4] znaky kromě mezer. Mezery oddělují jednotlivá slova a tuto informaci je nutné zachovat.

- **Sekvence opakujících se znaků**. Sekvence opakujících se znaků také většinou nepřinášejí informace, které by se měly zohledňovat při porovnávání záznamů. Například slova *auto* a *auutoooo* pravděpodobně představují stejnou entitu — auto. Proto je nutné nahradit sekvence opakujících se znaků jedním znakem.

- **Nahrazení znaků**. Některé dvojice znaků jsou si velice podobné (z hlediska porovnání skoro stejné). Znaky jako y a i nebo w a v jsou si mnohem podobnější než například znaky y a b. Podobné znaky se ve slovech často zaměňují. Toto je nutné zohlednit i při shlukování, proto je možné vybrat jeden z těchto znaků a nahradit jím všechny výskyty druhého znaku. V závislosti na typu dat je možné sestavit těchto pravidel pro nahrazení znaků nebo sekvencí znaků více.

- **Číslice**. Pokud shlukujeme například názvy obcí nebo jména osob, číslice je také možné odstranit.

- **Nadbytečná slova**. Záznamy mohou obsahovat slova, která jsou natolik obecná, že nepřinášejí žádnou informaci, užitečnou pro shlukování – „stop slova" (*stop words*[5]). Mezi taková slova patří běžně užívaná slova v různých jazycích. Existují předem připravené slovníky se „stop slovy" pro různé jazyky. V českém jazyce mezi stop slova patří například: *a, aby, byl, byla, pro, však, které.*

Proces čištění dat je závislý na typu dat. Je proto nutné zvolit způsob čištění dat tak, aby bylo odfiltrováno co nejvíce rušivých prvků, ale zároveň aby daná data nebyla znehodnocena. Důležité také je, aby při procesu čištění dat nebyla ztracena původní vstupní data.

[2]UNICODE - http://www.unicode.org/standard/WhatIsUnicode.html
[3]ASCI - http://www.asciitable.com
[4]Alfanumerické znaky - https://en.wikipedia.org/wiki/Alphanumeric
[5]Stop words - https://en.wikipedia.org/wiki/Stop_words

2.3.2 Indexace

Indexace [6] je důležitým krokem před samotným porovnáváním dvojic záznamů. Vstupem tohoto kroku jsou pročištěná a standardizovaná data, která jsou připravena na porovnávání. Při shlukování dvou databází by tedy bylo nutné porovnat každý záznam z první databáze s každým záznamem z druhé databáze. Celkem by tedy bylo nutné porovnat $m \times n$ dvojic záznamů (m a n jsou počty záznamů v jednotlivých databázích). Pokud jsou shlukovány záznamy pouze v rámci jedné databáze, musí být každý záznam porovnán se všemi ostatními záznamy v dané databázi. Celkem by tedy bylo nutné porovnat $(n \times (n-1))/2$ dvojic záznamů (každá dvojice je porovnána pouze jednou). Proces porovnávání dvojic záznamů má tedy kvadratickou časovou složitost. Při každém porovnání dvojice záznamů je navíc nutné porovnat všechny atributy obou záznamů. Porovnávání je nejnáročnější operací procesu roztřídění záznamů do shluků.

Když je porovnáván jeden záznam se všemi ostatními záznamy většina porovnání dopadne neúspěšně a jen malé procento skončí úspěchem (záznamy představují stejnou entitu). Aby bylo zredukováno potenciální velké množství kandidátních párů k porovnání, je nutné vybrat ty páry, u kterých je velká pravděpodobnost na úspěch porovnání. Tyto odfiltrované páry jsou poté předány k detailnímu porovnání. Proces vybírání kandidátních párů k detailnímu porovnání se nazývá indexace.

Existuje několik metod indexace záznamů. V následujícím seznamu je popsáná metoda *standardního blokování* a metoda *seřazení záznamů*.

- **Standardní blokování**. Principem této metody je rozdělit záznamy v každé databázi do bloků podle předem daného blokovacího klíče (*blocking key*). Blokovací klíč je aplikován na vybraný atribut záznamů. K detailnímu porovnání jsou poté předány pouze záznamy, které patří do stejného bloku.

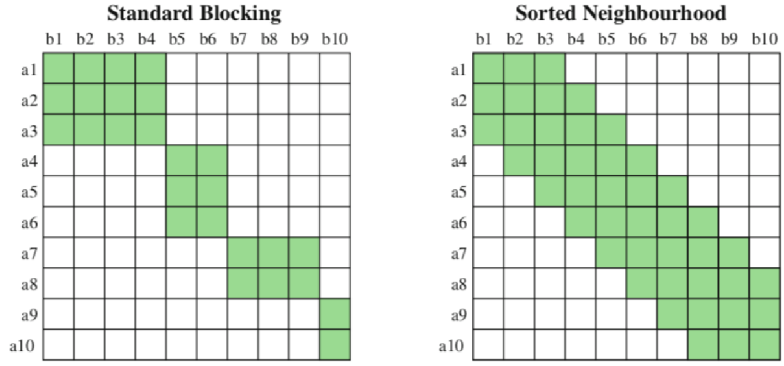
  Blokovacím klíčem může být například stejná hodnota atributu PSČ (poštovní směrovací číslo) nebo stejná hodnota foneticky zakódovaného jména, či příjmení. Funkce pro fonetické kódování atributů jsou vysvětleny níže v této části.

- **Seřazení podle podobnosti**. Principem této metody (*Sorted Neighbourhood*) je seřadit záznamy v databázi tak, aby podobné záznamy byly blíže sobě než méně podobné záznamy. Používá se zde „řadící klíč“ (*sorting key*), který je jako blokovací klíč aplikován na konkrétní atributy záznamů.

  Pro detailní porovnání jsou poté předány pouze záznamy, které leží blízko sebe v seřazené databázi. Využívá se zde „posuvné okno“ (*sliding window*) s fixní délkou w. Toto okno je posunováno po seřazené databázi, a ze záznamů, které se nacházejí v posuvném okně jsou v každém kroku generovány kandidátní páry pro detailní porovnání.

  Nevýhodou tohoto přístupu je, že množství záznamů se stejnou hodnotou řadícího klíče může být větší než velikost posuvného okna. V takovém případě nebudou vygenerovány všechny kandidátní páry s danou hodnotou řadícího klíče. Řešením je dynamická změna velikosti posuvného okna. Velikost posuvného okna se zvyšuje dokud jsou hodnoty řadícího klíče záznamů v posuvném okně podobné. Podobnost těchto záznamů je určena pomocí funkcí pro výpočet podobnosti řetězců, které jsou popsány v sekci č. 2.3.3.

Na obrázku 2.8 je zobrazen rozdíl mezi použitím metody standardního blokování a metody seřazení podle podobnosti, s velikostí posuvného okna *w=3*.



Obrázek 2.8: Na tomto obrázku je zobrazen rozdíl při generování kandidátních párů záznamů pro detailní porovnání pomocí předchozích dvou indexačních metod. Vyplněné buňky představují kandidátní páry, určené k detailnímu porovnání. Prázdné buňky představují páry, které nebudou dále porovnávány. Sjednocení prázdných a vyplněných buněk představuje páry, které by bylo nutno porovnat naivní metodou, kdy je každý záznam porovnáván se všemi ostatními záznamy. Při shlukování záznamů v rámci jedné databáze je matice symetrická, stačí proto porovnat pouze páry nad nebo pod diagonálou. Obrázek převzat z knihy [6].

Proces indexace má kvadratickou časovou složitost, jelikož je nutné porovnat všechny záznamy podle vybraného atributu mezi sebou. Využití indexace tedy nemá smysl, pokud shlukujeme záznamy na základě jednoho atributu. Indexace může naopak výrazně zmenšit čas porovnávání záznamů, pokud záznamy obsahují více atributů, které chceme zohlednit při porovnávání.

Funkce pro fonetické kódování řetězců

Funkce pro fonetické kódování řetězců [6] jsou používány při procesu indexace. Před použitím vybraného atributu jako blokovacího nebo řadicího klíče jsou nejdříve hodnoty tohoto atributu zakódovány. Cílem je, aby stejně znějící řetězce byly v procesu indexace zařazeny do stejného bloku, resp. byly seřazeny blíže u sebe. Nejčastěji jsou foneticky kódovány atributy, jako například názvy nebo jména.

Cílem funkcí pro fonetické kódování je převést vstupní řetězec na kód, který odpovídá tomu, jak by daný řetězec byl vysloven, tedy aby pro dva stejně znějící řetězce byl vygenerován stejný kód. Funkce pro fonetické kódování řetězců jsou závislé na konkrétním jazyce. Proto existují varianty těchto funkcí pro konkrétní jazyky. Fonetické kódování je také možné použít při výpočtu podobnosti mezi řetězci, kdy jsou nejdříve vstupní řetězce foneticky zakódovány a následně porovnány pomocí některé z funkcí podobnosti řetězců, které jsou popsány dále v této kapitole. V tabulce 2.2 je zobrazeno použití vybraných fonetických kódování pro zakódování různých jmen.

V následujícím seznamu jsou popsány vybrané funkce pro fonetické kódování:

- **Soundex** [19]. Tento algoritmus je jedním z nejstarších algoritmů pro fonetické kódování řetězců. Algoritmus byl patentován v roce 1918 a je založen na anglické výslovnosti. Existují však varianty tohoto kódování i pro jiné jazyky.

Výsledný kód vznikne převodem vstupního řetězce (slovo) na kód ve tvaru počátečního písmene řetězce následovaného několikamístným číselným kódem (většinou je použit třímístný číselný kód). Tento číselný kód je vytvořen pomocí transformační tabulky. Tabulka popisuje transformace znaků na číslice. Z výsledného kódu jsou odebrány sekvence duplicitních číslic a je zarovnán na požadovanou délku. Pokud je délka kódu menší než požadovaná délka kódu, kód je doplněn zprava nulami. Pokud je délka kódu větší než požadovaná délka kódu, výsledný kód je ořezán na požadovanou délku. Ukázka transformačních pravidel je zobrazena v tabulce 2.1.

- **Phonex** [13]. Kódovací algoritmus Phonex navazuje na předešlý algoritmus Soundex. Kvalita výsledného kódu je vylepšena za pomocí předzpracování (*pre-processing*) vstupního řetězce. Postupně je aplikováno 11 pravidel pro modifikaci původního řetězce. Po předzpracování je řetězec zakódován podobně jako u algoritmu Soundex. Transformační tabulka je ale pozměněná.

- **Phonix** [19]. Algoritmus Phonix navazuje na algoritmus Phonex a rozšiřuje krok předzpracování vstupního řetězce. Postupně je aplikováno více než sto pravidel pro modifikaci původního řetězce. Předzpracovaný řetězec je poté zakódován podobně jako algoritmus Soundex. Transformační tabulka pro kódování se ale liší. Ukázka transformačních pravidel je zobrazena v tabulce 2.1.

  Díky velkému množství pravidel v kroku předzpracování je tento algoritmus komplexnější, ale také více náročnější než předešlé algoritmy Phonix a Soundex.

Soundex		**Phonix**	
a, e, h, i, o u, w, y	→ 0	a, e, h, i, o u, w, y	→ 0
b, f, p, v	→ 1	b, p	→ 1
c, g, j, k, q, s, x, z	→ 2	c, g, j, k, q	→ 2
d, t	→ 3	d, t	→ 3
l	→ 4	l	→ 4
m, n	→ 5	m, n	→ 5
r	→ 6	r	→ 6
		f, v	→ 7
		s, x, z	→ 8

Tabulka 2.1: Transformační tabulka pro fonetické kódování Soundex a Phonix. Hodnoty převzaty z knihy [6].

- **NYSIIS**. Oproti předešlým algoritmům, algoritmus NYSIIS (*New York State Indentification and Intelligence System*) neprovádí převod vstupního řetězce na číselný kód. Výstupem je pouze kód tvořený znaky. Nejdříve jsou aplikována pravidla pro modifikaci původního řetězce. Vzniklý řetězec je nakonec ořezán na 6 znaků a vrácen jako výsledný kód. Tento algoritmus je po algoritmu Soundex druhým nejvíce populárním algoritmem pro fonetické kódování řetězců.

- **Double-Metaphone** [15]. Hlavní nevýhodou předešlých algoritmů pro fonetické kódování řetězců je jejich závislost na konkrétním jazyce. V reálných databázích se nenacházejí jen anglické názvy (slova), proto se algoritmus Double-Metaphone zaměřuje nejen na anglická, ale také evropská a asijská jména a názvy. Vhodný je také pro data, která obsahují chyby.

Prvním krokem je opět předzpracování vstupního řetězce, kde je aplikováno velké množství pravidel pro modifikaci daného řetězce. Rozdílem tohoto algoritmu je, že pro některá jména jsou navráceny dva výsledné fonetické kódy místo jednoho. Tyto dva kódy vzniknou odlišným pořadím aplikování fonetických transformačních pravidel.

Řetězec	**Soundex**	**Phonex**	**Phonix**	**NYSIIS**	**Double Metaphone**
peter	p360	b360	p300	pata	ptr
pete	p300	b300	p300	pat	pt
pedro	p360	b360	p360	padr	ptr
stephen	s315	s315	s375	staf	stfn
steve	s310	s310	s370	staf	stf
smith	s530	s530	s530	snat	sm0, xmt
smythe	s530	s530	s530	snat	sm0, xmt
gail	g400	g400	g400	gal	kl
gayle	g400	g400	g400	gal	kl
christine	c623	c623	k683	chra	krst
christina	c623	c623	k683	chra	krst
kristina	k623	c623	k683	cras	krst

Tabulka 2.2: Ukázka kódování jmen pomocí výše popsaných funkcí pro fonetické kódování. Tabulka převzata a upravena z knihy [6].

2.3.3 Porovnání dvojic záznamů

Dalším krokem v procesu roztřídění záznamů do shluků je samotné porovnávání jednotlivých záznamů mezi sebou [6] (*Record Comparison*). Existují tři přístupy pro porovnávání: přesné porovnání, částečné (*truncate*) a porovnání s využitím kódovací funkce.

Pro porovnání jednotlivých atributů záznamů jsou využity porovnávací funkce (*comparison functions*). Obecný předpis porovnávací funkce zobrazuje rovnice (2.1). Jedná se o funkci pro výpočet podobnosti.

$$s = sim(a_i, a_j) \tag{2.1}$$

Parametry a_i a a_j jsou atributy pro porovnání. Výsledkem je hodnota s, která určuje jak moc jsou si tyto atributy podobné. Pro podobnost s platí:

- Pokud $sim(a_i, a_j) = 1$, atributy jsou zcela shodné.
- Pokud $sim(a_i, a_j) = 0$, atributy jsou zcela odlišné.
- Pokud $0 \leq sim(a_i, a_j) \leq 1$, atributy jsou do jisté míry podobné.

Význam toho, že jsou atributy zcela shodné, zcela odlišné nebo do jisté míry podobné záleží na konkrétní použité funkci podobnosti. Lze také využít funkce pro výpočet vzdálenosti mezi atributy. Podobnost a vzdálenost jsou navzájem převoditelné a platí, že čím jsou si atributy podobnější, tím je jejich vzdálenost menší. Čím je vzdálenost atributů větší, tím méně jsou si podobné.

Výsledkem přesného porovnávání je buď hodnota 1 (atributy jsou zcela shodné) nebo hodnota 0 (atributy jsou zcela odlišné). Funkce podobnosti pro přesné porovnávání je zobrazena v následující rovnici:

$$sim_{exact}(a_1, a_2) = \begin{cases} 1.0 & pokud\ a_1 = a_2 \\ 0 & pokud\ a_1 \neq a_2 \end{cases} \tag{2.2}$$

Variantou přesného porovnávání je částečné porovnávání, kdy jsou přesně porovnávány pouze části obou atributů (prefix nebo sufix). V tomto případě musí být oba atributy typu řetězec. Pro zjištění míry podobnosti dvou řetězců je nutné využít funkce pro výpočet podobnosti řetězců. Některé z těchto funkcí jsou popsány v následujícím seznamu:

- **Editační vzdálenost**. Nejznámějším příkladem editační vzdálenosti je Levenshteinova editační vzdálenost [17]. Tato vzdálenost je definována jako nejmenší možný počet operací *vložení znaku*, *odstranění znaku* a *substituce znaku* pro změnu jednoho řetězce na druhý řetězec. Pro výpočet vzdálenosti je využito dynamické programování[6]. Metoda nezohledňuje ve které části se řetězce liší – odlišnostem řetězců na začátku nebo na konci přikládá stejnou váhu. Hodí se proto pro krátké řetězce nepředstavující jména nebo názvy. Ukázka výpočtu Levenshteinovy editační vzdálenosti pro dva páry jmen je zobrazena v následujících tabulkách.

		0	1	2	3	4	5
			p	**a**	**v**	**l**	**a**
0		**0**	1	2	3	4	5
1	**p**	1	**0**	1	2	3	4
2	**a**	2	1	**0**	1	2	3
3	**v**	3	2	1	**0**	1	2
4	**e**	4	3	2	1	**1**	2
5	**l**	5	4	3	2	1	**2**

		0	1	2	3	4	5
			r	**a**	**d**	**i**	**m**
0		**0**	1	2	3	4	5
1	**r**	1	**0**	1	2	3	4
2	**a**	2	1	**0**	1	2	3
3	**d**	3	2	1	**0**	1	2
4	**k**	4	3	2	1	**1**	2
5	**a**	5	4	3	2	2	**2**

Tabulka 2.3: Ukázka výpočtu Levenshteinovy editační vzdálenosti dvou jmen. V obou případech je vypočtená vzdálenost jmen rovna 2. Tučné číslice vyznačují cestu ke konečnému výsledku. Tyto tabulky byly převzaty a upraveny z knihy [6] (jména byla změněna na česká).

Podobnost dvou vstupních řetězců s_1 a s_2 lze pomocí Levenshteinovy vzdálenosti vyjádřit jako:

$$sim_{levenshtein}(s_1, s_2) = 1.0 - \frac{dist_{levenshtein}(s_1, s_2)}{max(|s_1|, |s_2|)}, \tag{2.3}$$

kde $dist_{levenshtein}$ je Levenshteinova vzdálenost, $|s_1|$ délka řetězce s_1, a $|s_2|$ délka řetězce s_2.

Rozšířením této metody je editační vzdálenost **Damerau-Levenshtein** [17], která přidává čtvrtou operaci — výměnu dvou sousedních znaků. Výměna dvou sousedních znaků je častou chybou při získávání dat. Časová složitost obou těchto metod je $O(|s_1| \times |s_2|)$.

[6] Dynamické programování - https://en.wikipedia.org/wiki/Dynamic_programming

- **Q-gram**. Principem této metody, která je také nazývána N-gram, je rozdělit oba vstupní řetězce na krátké podřetězce o délce q (Q-gram řetězce). Délka těchto řetězců je obvykle 2 (*bigram*) nebo 3 (*trigram*) znaky. K rozdělení vstupních řetězců na Q-gram řetězce je použit princip posuvného okna o délce q, které je posunováno od začátku řetězce na jeho konec. V každém kroku je vytvořen jeden Q-gram řetězec. Počet vzniklých Q-gram řetězců odpovídá $c = |s| - q + 1$, kde $|s|$ je délka vstupního řetězce a q je délka Q-gram řetězců.

  Podobnost vstupních řetězců s_1 a s_2 je pak možné vypočítat pomocí některého z následujících koeficientů, kde c je počet společných Q-gram řetězců, které se nacházejí v obou vstupních řetězcích a c_1, c_2 je počet Q-gram řetězců vzniklých z řetězce s_1, resp. s_2:

$$sim_{overlap}(s_1, s_2) = \frac{c}{min(c_1, c_2)} \tag{2.4}$$

$$sim_{jaccard}(s_1, s_2) = \frac{c}{(c_1 + c_2) - c} \tag{2.5}$$

$$sim_{dice}(s_1, s_2) = \frac{2 \times c}{c_1 + c_2} \tag{2.6}$$

  Možným rozšířením metody Q-gram je zohlednění pozice jednotlivých Q-gram řetězců v rámci vstupního řetězce. Při určení počtu společných Q-gram řetězců nejsou brány v úvahu Q-gram řetězce, které jsou od sebe vzdáleny více, než je maximální povolená vzdálenost.

  Výpočet podobnosti řetězců pomocí metody Q-gram je možné využit pro krátké řetězce, které nepředstavují jména. Časová složitost tohoto algoritmu je $O(|s_1| + |s_2|)$. Následující tabulka zobrazuje rozdělení vstupního řetězce na jednotlivé Q-gram řetězce:

Řetězec	**Bigram**	**Trigram**	**Bigram s pozicí**
david	da, av, vi, id	dav, avi, vid	(da,1), (av,2), (vi,3), (id,4)
pavel	pa, av, ve, el	pav, ave, vel	(pa,1), (av,2), (ve,3), (el,4)

Tabulka 2.4: Ukázka rozdělení vstupního řetězce na bigram řetězce, trigram řetězce a bigram řetězce s pozicí.

- **Jaro, Jaro-Winkler** [4][7]. Tyto metody byly speciálně vyvinuty pro výpočet podobnosti jmen. Kombinují editační vzdálenost s podobnými postupy jako u podobnosti Q-gram. Výpočet podobnosti dvou řetězců pomocí základní metody **Jaro** je zobrazen v rovnici (2.7). Pro výpočet počtu shodných znaků c a počtu transpozic t (dvojice sousedních znaků, které se liší pořadím znaků, např. 'da' a 'ad'.) je využito posuvné okno o velikosti poloviny délky většího z řetězců. Toto okno je posunováno současně po obou řetězcích. V každém kroku posunutí je zaznamenán počet znaků, které jsou shodné v obou oknech a počet transpozic.

$$sim_{jaro}(s_1, s_2) = \begin{cases} 0 & pokud\ c = 0 \\ \frac{1}{3}\left(\frac{c}{|s_1|} + \frac{c}{|s_2|} + \frac{c-t}{c}\right) & jinak \end{cases} \tag{2.7}$$

Na základě zjištění studií, že větší počet chyb ve jménech se nachází uprostřed nebo na konci řetězce, bylo vytvořeno rozšíření **Jaro-Winkler**, které klade větší váhu odlišnostem řetězců na jejich začátku, než uprostřed nebo na konci. Časová složitost tohoto algoritmu je $O(|s_1| + |s_2|)$.

- **Monge-Elkan** [4]. Tato metoda představuje obecný postup pro výpočet podobnosti mezi řetězci, které obsahují více slov. Porovnávané řetězce jsou nejprve rozděleny do dvou množin slov A a B, které obsahují slova z jednotlivých řetězců. Podobnost obou řetězců je pak vypočtena jako:

$$sim_{monge-elkan}(s_1, s_2) = \frac{1}{|A|} \sum_{i=1}^{|A|} max_{j=1}^{|B|} sim'(A_i, B_j), \tag{2.8}$$

  kde $|A|$ je počet slov prvního řetězce, $|B|$ je počet slov druhého řetězce a sim' je jakákoliv funkce pro výpočet podobnosti řetězců.

- **Jaccard, Extended-Jaccard**. Cílem Jaccardova koeficientu je vypočítat podobnost mezi dvěma množinami položek (tokenů). Výsledná hodnota je vypočtena jako počet položek v průniku těchto dvou množin, děleno počet položek ve sjednocení těchto množin. Tento vztah zobrazuje následující rovnice:

$$sim_{jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}. \tag{2.9}$$

  Stejný princip je využit u metody Q-gram, kde položky množin A a B představují jednotlivé Q-gram řetězce získané ze vstupních řetězců. Tento vztah je zobrazen v rovnici (2.5).

  Rozšíření Jaccardova koeficientu s názvem **Extended Jaccard** lze využít, pokud porovnávané vstupní řetězce obsahují více slov. Jednotlivá slova ze vstupních řetězců jsou navzájem porovnána pomocí zvolené funkce podobnosti sim'. Všechny dvojice slov, jejichž podobnost je větší než předem zvolený práh θ $(0 < \theta < 1)$, jsou uloženy do množiny S. Vztah pro získání této množiny je zobrazen zde:

$$S = \{(a_i, b_j) | a_i \in A \ \wedge b_j \in B : sim'(a_i, b_j) \geq \theta\}. \tag{2.10}$$

  Slova, jejichž podobnost není s žádným slovem z druhé množiny větší než práh θ představují množiny U_A, resp. U_B. Vztah pro získání těchto množin je zobrazen zde:

$$\begin{aligned} U_A &= \{a_i | a_i \in A \ \wedge b_j \in B \wedge (a_i, b_j) \notin S \ \}, \\ U_B &= \{b_j | a_i \in A \ \wedge b_j \in B \wedge (a_i, b_j) \notin S \ \}. \end{aligned} \tag{2.11}$$

  Rozšířený Jaccardův koeficient je pak vypočten vztahem:

$$sim_{jaccard\_extended} = \frac{|S|}{|S| + |U_A| + |U_B|}. \tag{2.12}$$

  Díky možnosti zvolit funkci podobnosti sim' jsou tyto metody využitelné pro různé typy dat.

- **Longest Common Substring** [16]. Hlavní myšlenkou tohoto algoritmu je najít a odstranit nejdelší možné podřetězce, které jsou společné v obou vstupních řetězcích. Tento postup se opakuje tak dlouho, dokud délka nejdelších společných podřetězců je větší nebo rovna předem stanovené délce l_{min} (minimální délka je většinou 2 nebo 3 znaky). Suma délek (l_c) všech nalezených společných podřetězců je pak použita pro výpočet podobnosti mezi vstupními řetězci s_1 a s_2. Lze využít jeden z již dříve zmíněných koeficientů:

$$sim_{lcs\_overlap}(s_1, s_2) = \frac{l_c}{min(s_1, s_2)}, \tag{2.13}$$

$$sim_{lcs\_jaccard}(s_1, s_2) = \frac{l_c}{(|s_1| + |s_2|) - |l_c|}, \tag{2.14}$$

$$sim_{lcs\_dice}(s_1, s_2) = \frac{2 \times l_c}{|s_1| + |s_2|}. \tag{2.15}$$

  Časová složitost výpočtu sumy (l_c) všech délek nejdelších společných podřetězců je $O(|s_1| \times |s_2|)$. Nevýhodou tohoto algoritmu je, že vypočtená podobnost vstupních řetězců nemusí být vždy symetrická – pokud jsou vstupní řetězce prohozeny, výsledná míra podobnosti se může lišit. Toto lze ukázat na výpočtu podobnosti mezi řetězci s_1 = 'janj' a s_2 = 'jnja' a $l_{min} = 2$. V prvním kroku bude jako nejdelší nalezený společný podřetězec nalezen řetězec 'ja', po jehož odstranění z obou řetězců vzniknou nové řetězce s'_1 = 'nj' a s'_2 = 'jn', dále už nejsou nalezeny žádné další společné podřetězce, tudíž $l_c = 2$ a $sim_{lcs\_dice} = 0.5$. Pokud jsou vstupní řetězce vyměněny, tedy s_1 = 'jnja' a s_2 = 'janj', jako první nejdelší společný podřetězec je nalezen řetězec 'nj', po jehož odstranění vzniknou nové řetězce s'_1 = 'ja' a s'_2 = 'ja'. V dalším kroku je nalezen společný podřetězec 'ja', po jehož odstranění z obou řetězců už nejsou nalezeny další společné podřetězce, delší než 2 znaky ($l_{min} = 2$), proto $l_c = 4$ a $sim_{lcs\_dice} = 1$. Pro vyřešení tohoto problému je nutné vypočítat podobnost vstupních řetězců v obou pořadích a výsledky zprůměrovat.

2.3.4 Klasifikace dvojic záznamů

Dvojice záznamů, které byly vyfiltrovány v kroku Indexace a detailně porovnány v předešlém kroku Porovnání záznamů je nyní možné klasifikovat [6]. Záznamy lze klasifikovat do 3 tříd: shodné (*Matches*), neshodné (*Non-matches*) a potenciálně shodné (*Potential matches*). Dvojice záznamů, které byly klasifikovány jako potenciálně shodné jsou určeny pro ruční posouzení člověkem.

Výsledkem předešlého kroku Porovnání záznamů je pro každou porovnanou dvojici záznamů vektor, který obsahuje míru podobnosti jednotlivých atributů záznamů. Ukázka výstupu porovnání dvou dvojic záznamů, které obsahují atributy jméno a příjmení, je zobrazena v tabulce 2.5. Tento vektor je dále použit při klasifikaci. Základním způsobem klasifikace záznamů do tříd je **Prahování**. Využívají se zde dva prahy t_{up} a t_{low}, a suma podobnosti záznamů (*SimSum*). Klasifikace do tříd pak probíhá podle těchto pravidel:

$$\begin{aligned} SimSum[r_i, r_j] &\geq t_{up} &&\implies [r_i, r_j] \rightarrow Match, \\ t_{low} < SimSum[r_i, r_j] &< t_{up} &&\implies [r_i, r_j] \rightarrow Potential\ Match, \\ SimSum[r_i, r_j] &\leq t_{low} &&\implies [r_i, r_j] \rightarrow Non - Match, \end{aligned} \tag{2.16}$$

kde $SimSum[r_i, r_j]$ je suma podobnosti záznamů r_i a r_j. Pokud je hodnota prahů stejná, tedy $t_{up} = t_{low} = t$, třída Potenciální shoda je odstraněna. V takovém případě se jedná

o klasifikaci do dvou tříd. Taková klasifikace pak probíhá podle těchto pravidel:

$$\begin{aligned} SimSum[r_i, r_j] &\geq t \implies [r_i, r_j] \rightarrow Match, \\ SimSum[r_i, r_j] &< t \implies [r_i, r_j] \rightarrow Non - Match. \end{aligned} \tag{2.17}$$

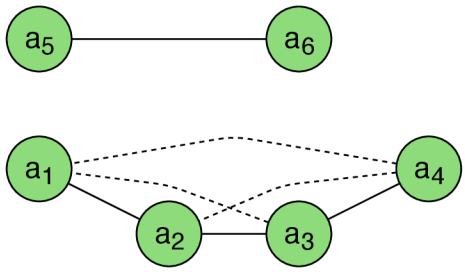
Záznam	Jméno	Příjmení	Suma podobnosti
a1	Petr	Novák	
b1	Peter	Nový	
	0.88	**0.66**	**1.54**
a2	Jan	Šimeček	
b2	Johan	Schimetzek	
	0.75	**0.58**	**1.33**

Tabulka 2.5: Ukázka porovnání dvou dvojic záznamů. Tučná čísla představují vzniklé vektory, které obsahují míru podobnosti jednotlivých atributů.

Tranzitivní uzavření

Každý záznam může být klasifikován jako shodný s více jinými záznamy. V takovém případě vzniká řetězec záznamů, které jsou si podobné. Pokud nastane situace, že dvojice záznamů (a_1, a_2) a (a_2, a_3) byly klasifikovány jako shodné (Matches), a dvojice záznamů (a_1, a_3) byla klasifikována jako neshodná (Non-Matches), je možné řešit tranzitivní uzavření těchto záznamů. Příklad takové situace je zobrazen na obrázku a tabulce 2.6.

Kandidátní páry	Suma podobnosti	Klasifikace
(a1, a2)	5.20	Shodné
(a1, a3)	3.30	Neshodné
(a1, a4)	1.15	Neshodné
(a2, a3)	5.05	Shodné
(a2, a4)	2.70	Neshodné
(a3, a4)	5.25	Shodné
(a5, a6)	6.20	Shodné



Tabulka 2.6: Ukázka klasifikace záznamů do dvou tříd (Shodné, Neshodné) s prahem podobnosti $t = 5.0$. Jednotlivé záznamy jsou zobrazeny jako zelené kruhy. Plnou čarou jsou propojeny záznamy, které byly klasifikovány jako shodné (Matches). Čárkovanou čarou jsou propojeny záznamy, které nebyly klasifikovány jako shodné (Non-Matches). Obrázek i tabulka byly převzaty a upraveny z knihy [6].

Pokud však záznam a_2 představuje stejnou entitu (objekt reálného světa) jako záznamy a_1 a a_3, tak pak i záznam a_1 musí představovat stejnou entitu jako záznam a_3. Řešením takové situace je klasifikace dvojice záznamů (a_1, a_3) jako shodné (Matches).

Váhy atributů

Hlavní nevýhodou klasifikace s využitím sumy podobností je, že tato suma nezohledňuje rozdíly mezi jednotlivými atributy. Některé atributy mohou být pro shlukování důležitější

než ostatní – například atribut příjmení může být více důležitý než atribut jméno. Aby byla rozlišena důležitost jednotlivých atributů, lze jim přiřadit váhy. Při výpočtu sumy podobnosti záznamů jsou pak nejdříve jednotlivé hodnoty podobnosti atributů vynásobeny příslušnou váhou a následně sečteny. Například, pokud by byly použity váhy atributů při výpočtu sumy podobností záznamů a_1 a b_1, z tabulky 2.5 a váha atributu jméno by byla 1.0, váha atributu příjmení by byla 2.0, výslednou sumu podobnosti těchto záznamů pak lze vypočítat jako: $SimSum[a_1, b_1] = 1.0 \times 0.88 + 2.0 \times 0.66 = 2.2$.

2.3.5 Měření kvality shlukování

Posledním krokem procesu roztřídění záznamů do shluků je měření kvality shlukování [6]. K měření kvality shlukování je nutné znát u každé porovnávané dvojice záznamů referenční výsledek klasifikace (*true match status*), který určuje, zda oba záznamy reálně představují stejnou entitu (Matches) nebo ne (Non-Matches). Takový datový set je pak nazýván *ground-truth*.

Každá klasifikovaná dvojice záznamů pak lze zařadit do jedné z následujících kategorií:

- **True positives (TP)**. Dvojice záznamů, které byly klasifikovány jako shodné, a jsou opravdu shodné.
- **False positives (FP)**. Dvojice záznamů, které byly klasifikovány jako shodné, ale nejsou shodné.
- **True negatives (TN)**. Dvojice záznamů, které byly klasifikovány jako neshodné, a jsou opravdu neshodné.
- **False negatives (FN)**. Dvojice záznamů, které byly klasifikovány jako neshodné, ale jsou shodné.

Cílem shlukování záznamů je, aby počty dvojic záznamů v kategoriích TP a TN byly co nejvyšší, a zároveň počty dvojic záznamů v kategoriích FP a FN byly co nejnižší. Na základě počtů dvojic záznamů v jednotlivých kategoriích lze vypočítat různé metriky pro měření kvality shlukování. Nejpočetnější kategorií je kategorie TN, která většinou několikanásobně převyšuje ostatní kategorie. Z tohoto důvodu nejsou pro určení kvality shlukování vhodné metriky, které zohledňují počty záznamů v této kategorii.

V následujícím seznamu jsou popsány 3 metriky pro měření kvality shlukování. Výsledkem všech těchto metrik je hodnota v intervalu 0 až 1.

- **Precision**. Precision, neboli přesnost, určuje jaký podíl z dvojic klasifikovaných jako shodné (TP, FP) byl správně klasifikován jako shodný (TP). Tento vztah zobrazuje následující rovnice:

$$precision = \frac{TP}{TP + FP}. \tag{2.18}$$

- **Recall**. Recall určuje, jaký podíl z dvojic, které jsou opravdu shodné (TP, FN) byl klasifikován jako shodný (TP). Tento vztah zobrazuje následující rovnice:

$$recall = \frac{TP}{TP + FN}. \tag{2.19}$$

- **F-Measure**. Výsledkem této metriky je harmonický průměr předešlých dvou metrik. Hodnota této metriky je vysoká pouze pokud je vysoká hodnota metriky Precision i Recall. Metrika F-Measure je nejčastější metrika používaná pro měření kvality shlukování. Vztah pro výpočet této metriky zobrazuje následující rovnice:

$$f - measure = 2 \times \frac{precision \times recall}{precision + recall}. \tag{2.20}$$

Kapitola 3

Analýza požadavků a návrh řešení

V této kapitole je nejdříve detailněji popsán cíl této práce. V další části kapitoly je navržena struktura databáze pro uložení původních slov z matričních záznamů a jejich normalizovaných variant. Nakonec je navržen obecný princip normalizace slov z matričních záznamů.

3.1 Cíl práce

Cílem této práce je navrhnout metodu pro roztřídění slov objevujících se v matričních záznamech a jejich následnou ruční normalizaci (přiřazení normalizované podoby daného slova). Dále tuto navrženou metodu implementovat do již existující genealogické webové aplikace DEMoS.

Navržená metoda umožní uživateli při vytváření nebo editaci matričních záznamů normalizovat jednotlivé typy slov (jména, příjmení, obce, povolání atd.) z těchto záznamů. Pro jednotlivá slova z matričních záznamů budou automaticky navrhovány doporučené normalizované varianty, které budou seřazeny podle počtu výskytů v ostatních záznamech. Nejčastější doporučená normalizovaná varianta daného slova bude automaticky vybrána pro normalizaci tohoto slova. Uživatel bude mít možnost výběru z navržených doporučených normalizovaných variant. Pokud s navrženými variantami nebude souhlasit, bude možné vyplnit (a tím vytvořit) vlastní normalizovanou variantu daného slova. Při zadávání slov, která jsou závislá na pohlaví (např. jméno otce), bude toto pohlaví zohledněno při návrhu normalizovaných variant.

Proces normalizace bude probíhat na pozadí. Pokud uživatel nebude chtít normalizaci řešit, nebude muset. Pro každé zadané slovo pak bude z navržených normalizovaných variant automaticky vybrána nejčastěji použitá varianta, na kterou bude dané slovo normalizováno. Pokud u některých slov nebudou navrženy žádné normalizované varianty, daná slova nebudou normalizována.

Webová aplikace také umožní přihlášeným uživatelům normalizovat slova z matričních záznamů hromadně. K tomuto bude sloužit samostatná stránka, kde bude možné slova určená k normalizaci filtrovat a normalizovat. Slova bude možné filtrovat podle typu, obsahu, pohlaví, autora normalizace a stavu normalizace (normalizováno/nenormalizováno).

Posledním úkolem je otestování úspěšnosti shlukování na různých typech slov (mužská a ženská jména, příjmení, povolání a obce) a dále na různých dobách zápisu (stará čeština, latina, němčina, moderní čeština).

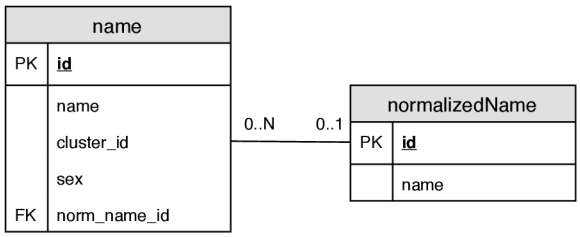
3.2 Návrh databáze

Aby bylo možné normalizovat různé typy slov, bylo nutné navrhnout obecné schéma databáze, které lze aplikovat na různé typy slov určených k normalizaci. Navržené schéma se skládá z dvou entitních množin. První entitní množina obsahuje originální slova z matričních záznamů. Druhá entitní množina obsahuje normalizované varianty těchto slov. Mezi těmito entitními množinami je vztah `1:N` (jedno normalizované slovo může představovat normalizovanou variantu pro více originálních slov). Minimální kardinalita obou entitních množin je `0`.

Entitní množina s originálními slovy obsahuje následující atributy: `ID` slova (jednoznačný identifikátor dané entity a zároveň primární klíč), `name` (slovo určené k normalizaci), `cluster_id` (číslo shluku, přidělené při procesu roztřídění záznamů do shluků, díky kterému je možné identifikovat záznamy, které patří do stejného shluku), `sex` (pohlaví, které je využito pouze u slov typu jméno a příjmení; povolené varianty: mužské `'M'`, ženské `'F'` a neznámé `'U'`), `norm_name_id` (cizí klíč vytvářející vazbu s entitou z entitní množiny s normalizovanými variantami) a `user_id`[1] (cizí klíč vytvářející vazbu s entitou z entitní množiny uživatelů – určuje autora normalizace daného slova). Pro tuto entitní množinu platí, že kombinace atributů `name, sex, norm_name_id` je unikátní.

Entitní množina s normalizovanými variantami originálních slov obsahuje pouze atribut `ID`, který je jednoznačným identifikátorem dané entity a zároveň primárním klíčem a atribut `name`, který představuje normalizovanou variantu originálních slov. Atribut `name` je unikátní v rámci této entitní množiny.

Schéma databáze pro normalizaci jmen je zobrazeno na obrázku 3.1.



Obrázek 3.1: Schéma databáze pro uložení jmen z matričních záznamů a jejich normalizovaných variant. Toto schéma je použito i pro ostatní typy dat (příjmení, povolání, obce atd.).

Entitní množina s originálními slovy představuje číselník slov daného typu v rámci celé aplikace. Entity z této entitní množiny pak mají vazby s entitami jiných entitních množin, které již jsou součástí databáze webové aplikace DEMoS.

3.3 Návrh principu normalizace slov z matričních záznamů

Navržený princip normalizace slov z matričních záznamů je zobrazen na obrázku 3.2. Výsledné řešení se skládá ze čtyř částí, které jsou na obrázku označeny jako část **a**, **b**, **c** a **d**. Následující text popisuje jednotlivé části tohoto řešení.

Originální slova z matričních záznamů budou nejdříve roztříděna do shluků (část **a** na obrázku 3.2) podle své podobnosti s ostatními slovy. K tomuto bude využita shluková analýza a proces roztřídění záznamů do shluků, popsaný v kapitole č. 2.3. Výsledkem roztřídění

[1] V době tvorby této práce nebyl tento atribut podporován v databázi aplikace DEMoS.

záznamů do shluků bude u každé entity z entitní množiny s originálními slovy vyplněné číslo shluku v atributu `cluster_id`. Entity (slova) se stejným číslem shluku budou poté chápány jako shodné (představující stejný objekt reálného světa — různé varianty stejného jména, příjmení, povolání atd.). Proces roztřídění slov do shluků bude spouštěn automaticky v určitém časovém intervale (např. každý den v nočních hodinách) pomocí softwarového démona CRON[2].

Při vytváření nebo editaci záznamů pomocí formuláře ve webové aplikaci DEMoS budou uživateli po zadání konkrétního slova pro toto slovo nabídnuty (v podobě formulářového výběrového prvku) doporučené normalizované varianty. Tyto varianty budou seřazeny podle počtu použití v ostatních záznamech a nejčastější varianta bude automaticky vybrána pro normalizaci. Při návrhu normalizovaných variant bude zohledněn typ a přípustná pohlaví formulářového pole, kde bylo toto slovo zadáno. Doporučené normalizované varianty budou získány porovnáním zadaného slova s originálními slovy stejného typu, již uloženými v databázi. Pokud zadané slovo už existuje v databázi (část **b** na obrázku 3.2), budou prioritně doporučeny normalizované varianty, které již byly použity u tohoto slova, následně pak normalizované varianty slov nacházejících se ve stejném shluku jako toto slovo. Pokud zadané slovo ještě neexistuje v databázi, uživatel bude mít možnost explicitně, pomocí tlačítka, požádat o vyhledání shluku slov (část **c** na obrázku 3.2), obsahujícího slovo, které je nejvíce podobné se zadaným slovem. Z tohoto shluku budou poté nabídnuty normalizované varianty slov (část **d** na obrázku 3.2). Uživatel bude mít vždy možnost zadat vlastní normalizovanou variantu daného slova ručně. Zadání vlastní normalizované varianty má větší prioritu než výběr z doporučených normalizovaných variant.

Po vyplnění a odeslání formuláře jsou vybrané normalizované varianty zadaných slov odeslány spolu s ostatními hodnotami tohoto formuláře na server. Výsledkem normalizace slova je vyplnění atributu `norm_name_id` příslušné entity představující toto slovo.

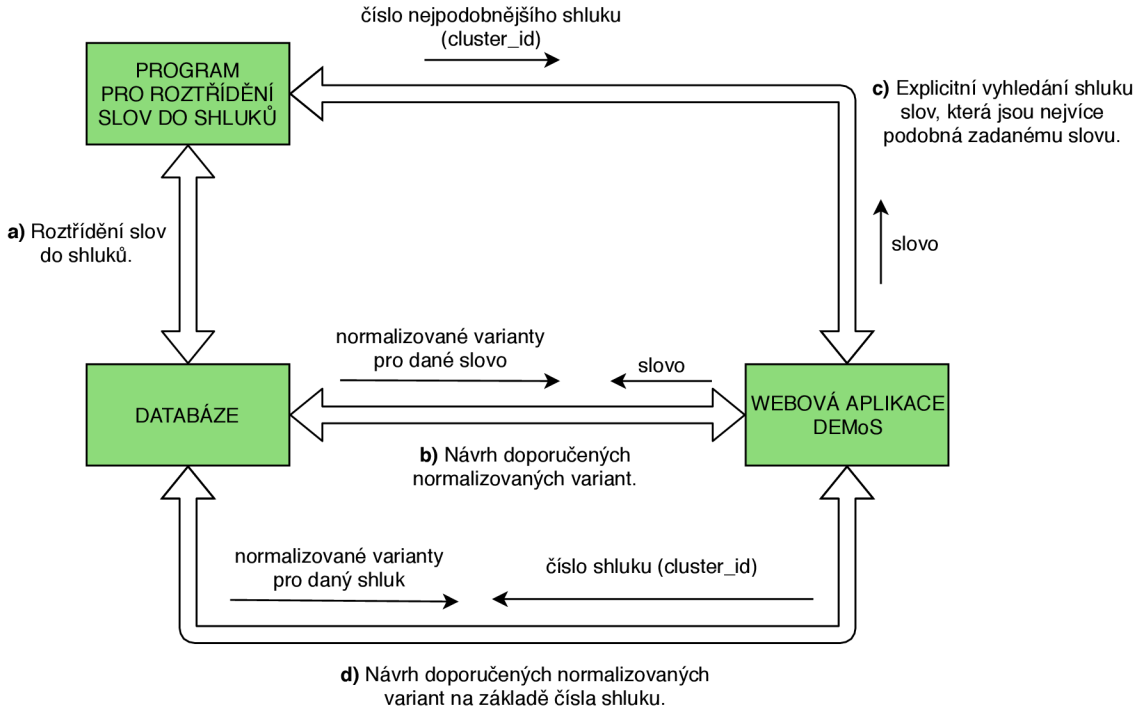
Spojení shluků slov pomocí normalizace

V případě, že dvěma slovům v odlišných shlucích (slova s odlišnou hodnotou atributu `cluster_id`) bude uživateli přidělena stejná normalizovaná varianta (stejná hodnota atributu `norm_name_id`), při následujícím plánovaném procesu roztřídění slov do shluků budou tyto shluky spojeny v jeden shluk. Tímto je možné spojovat vzniklé shluky a sdílet tak mezi nimi normalizované varianty. Například lze takto spojit shluky slov, u kterých by ke spojení během shlukování nikdy nedošlo (např. shluky se jmény George a Jiří).

Hromadná normalizace slov z matričních záznamů

Hromadná normalizace slov bude implementovaná v rámci samostatné stránky. Tato stránka se bude skládat z filtračního formuláře a tabulky se slovy určenými k normalizaci. Na každém řádku tabulky bude vypsáno slovo určené k normalizaci, pohlaví (pouze u jmen a příjmení), aktuální normalizace daného slova (pokud slovo nebylo zatím normalizováno bude vypsáno nejčastější slovo ve stejném shluku slov), autor normalizace, výběrové pole s doporučenými normalizovanými variantami (realizováno jako část **b** na obrázku 3.2), pole pro zápis vlastní normalizované varianty a tlačítko pro potvrzení normalizace daného slova.

[2]CRON - https://en.wikipedia.org/wiki/Cron



Obrázek 3.2: Schéma obecného principu normalizace slov z matričních záznamů. Toto schéma popisuje komunikaci mezi databází, webovou aplikací DEMoS a programem pro roztřídění slov do shluků.

Kapitola 4

Implementace a testování

V této kapitole bude popsána implementace normalizace slov z matričních záznamů podle návrhu, který byl uveden v předešlé kapitole. Nejdříve budou popsány použité technologie a knihovny. Zbytek této kapitoly bude rozdělen do dvou částí – implementace a testování. V části implementace bude popsána implementace programu pro roztřídění slov do shluků, včetně optimalizace tohoto programu pomocí paralelního zpracování, dále pak implementace webového rozhraní pro normalizaci slov ve webové aplikaci DEMoS. V poslední části bude provedeno testování úspěšnosti shlukování slov. Testováno bude celkem 640 kombinací parametrů shlukování (algoritmů) pro každý typ slov. Následně budou vybrány pro každý typ slov parametry s největší naměřenou úspěšností shlukování.

4.1 Použité technologie a nástroje

Technologie pro realizaci této práce byly zvoleny s ohledem na to, že se jedná o webovou aplikaci. Tato aplikace je napsána v jazyce PHP (Nette Framework[1]). Pro uložení dat je využita MySQL databáze. Program pro roztřídění slov do shluků je napsán v jazyce Python 3, který umožňuje efektivní práci s daty. Pro práci s formulářovými prvky a zajištění interaktivity normalizačního formuláře je použit programovací jazyk JavaScript a knihovna JQuery. Automatické nabízení normalizovaných variant pro zadaná slova je řešeno asynchronně, pomocí technologie AJAX. Dále byl použit jazyk CSS a HTML s frameworkem Bootstrap[2].

Knihovny použité pro roztřídění slov do shluků a testování

Za účelem čištění dat (krok Předzpracování záznamů) byla využita standardní knihovna jazyka Python pro regulární výrazy `re` a knihovna `unidecode`[3] pro převod UNICODE znaků na ASCII znaky. Pro práci s maticemi v kroku Porovnání dvojic záznamů a Klasifikace dvojic záznamů byla použita knihovna `NumPy`[4]. Funkce pro fonetické kódování řetězců byly použity z knihoven `fuzzy`[5] (NYSIIS), `pyphonetics`[6] (Soundex) a `metaphone`[7] (Double-Metaphone). Všechny funkce pro výpočet podobnosti řetězců byly použity z knihovny

[1] Nette Framework - https://nette.org/cs/
[2] Bootstrap - https://getbootstrap.com
[3] Unidecode - https://pypi.org/project/Unidecode/
[4] NumPy - http://www.numpy.org/
[5] fuzzy - https://pypi.org/project/Fuzzy/
[6] pyphonetics - https://pypi.org/project/pyphonetics/
[7] metaphone - https://pypi.org/project/Metaphone/

`textdistance`[8]. Pro paralelní výpočet podobností slov byla použita standardní knihovna `multiprocessing`. Ke změření doby potřebné pro roztřídění slov do shluků byla použita standardní knihovna `time`. Pro automatické generování grafů s výsledky testování byla použita knihovna `PyGnuplot`[9].

4.2 Program pro roztřídění slov do shluků

Jak už bylo řečeno, program pro roztřídění slov do shluku byl napsán v jazyce Python 3. Celá problematika je implementována v rámci jediné třídy `WordClustering`, která je součástí souboru `WordClustering.py`. Tato třída umožňuje:

- Načtení slov z databáze.
- Roztřídění slov do shluků.
- Aktualizaci čísel shluku v databázi pro roztříděná slova.
- Otestování úspěšnosti shlukování.
- Nalezení čísla nejpodobnějšího shluku slov pro zadané slovo.

Při instanciaci (vytvoření nového objektu dané třídy) této třídy je pomocí parametrů konstruktoru upřesněno nastavení shlukování. Jako parametr `table` je předán název databázové tabulky (typ slov) určené ke shlukování. Parametr `threshold` určuje práh podobnosti slov, parametr `comparison_function` určuje použitou metodu pro výpočet podobnosti slov, parametr `phonetic_encoding_function` určuje použité fonetické kódování slov a pomocí parametru `transitive_closure` lze nastavit, zda se má použít tranzitivní uzavření záznamů, nebo ne.

Další nepovinné parametry jsou: `sex` (pohlaví slov ke shlukování), `language` (jazyk slov ke shlukování) a `phonetic_encoding_function_weight` (váha fonetického kódování slov při výpočtu podobnosti slov).

Kromě této třídy je dále součástí skriptu `WordClustering.py` funkce `main`, která pracuje s parametry předanými skriptu a umožňuje různé využití třídy `WordClustering`. V následujícím seznamu jsou popsány možné režimy spuštění tohoto skriptu:

- `$ ./WordClustering`: Spuštění skriptu bez parametrů. Dojde k roztřídění slov do shluků a aktualizaci čísel shluku u všech typů slov. Jako parametry shlukování jsou pro každý typ slov vybrány parametry s nejlepšími výsledky shlukování, které jsou uloženy v globální proměnné `table_settings`.
- `$ ./WordClustering 'cluster_id' word sex table`: Nalezení shluku nejpodobnějších slov pro zadané slovo `word`. Vrácení čísla tohoto shluku. Tento režim je popsán v části č. 4.2.2.
- `$ ./WordClustering 'tests'`: Skript spuštěn v režimu testování – budou otestovány všechny kombinace parametrů shlukování pro každý typ slov. Nakonec budou vygenerovány grafy úspěšnosti shlukování. Tento režim je popsaný v kapitole č. 4.4.

[8] textdistance - https://pypi.org/project/textdistance/
[9] PyGnuplot - https://pypi.org/project/PyGnuplot

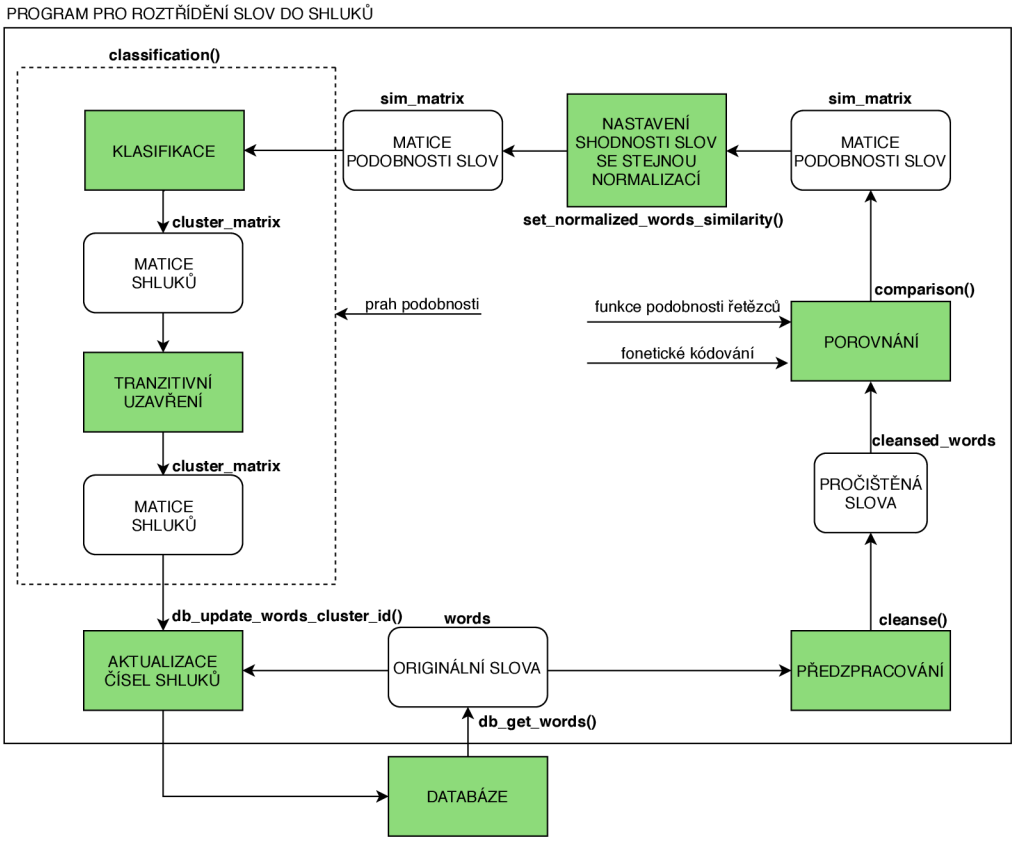
- `$ ./WordClustering 'lang_sex_tests'`: Provede otestování úspěšnosti shlukování zvlášť pro jednotlivé pohlaví a jazyky u všech typů slov. Jako parametry shlukování jsou pro každý typ slov vybrány parametry s nejlepšími výsledky shlukování, které jsou uloženy v globální proměnné `table_settings`.

4.2.1 Roztřídění slov do shluků

V této sekci je popsána implementace části **a** návrhu řešení, který je zobrazen na obrázku 3.2. V rámci této části je implementován *proces roztřídění záznamů do shluků*, který byl popsán v kapitole č. 2.3. Tento proces byl upraven pro shlukování záznamů v rámci jedné databáze – jedná se tedy o detekci duplicit (*Duplicate detection*).

Pro roztřídění slov do shluků byly využity pouze kroky *Předzpracování záznamů*, *Porovnání dvojic záznamů* a *Klasifikace dvojic záznamů*. Jelikož je v této práci řešeno roztřídění slov do shluků (= roztřídění záznamů do shluků na základě jednoho atributu – slova), je nutné porovnat každé slovo se všemi ostatními slovy. Z tohoto důvodu nebyl využit krok *Indexace*. Implementace kroků *Porovnání dvojic záznamů* a *Klasifikace dvojic záznamů* byla částečně inspirována algoritmem popsaným na stránce [5].

Pro každý typ slov, která mají být shlukována, byly zvoleny parametry shlukování (prah podobnosti, funkce podobnosti řetězců, fonetické kódování a tranzitivní uzavření záznamů) na základě testování, které je uvedeno v kapitole č. 4.4. Testovány byly pouze vybrané funkce podobnosti řetězců a fonetické kódování, které byly popsány v kapitole č. 2.3.2 a 2.3.3.



Obrázek 4.1: Schéma programu pro roztřídění slov do shluků.

Po instanciaci třídy `WordClustering` s vybranými parametry shlukování je proces roztřídění slov do shluků, který je zobrazen na obrázku 4.1, spuštěn pomocí metody `execute`. Nejdříve jsou z databáze načtena slova ke shlukování (na základě předaných parametrů při instanciaci třídy `WordClustering`), která jsou uložena v podobě seznamu do instanční proměnné `words`. Podle počtu načtených slov jsou pomocí knihovny `NumPy`, která umožňuje práci s maticemi, vytvořeny čtvercové matice `sim_matrix` a `cluster_matrix`. Tyto matice mají počet řádků i sloupců daný počtem načtených slov a jsou inicializovány nulami.

Načtená slova jsou poté předána ke předzpracování, jehož výsledkem je seznam pročištěných slov, uložený v instanční proměnné `cleansed_words`. Pročištěná slova jsou následně porovnána invokací metody `comparison`. V této metodě je pro každou dvojici slov vypočtena podobnost pomocí zvoleného fonetického kódování a funkce podobnosti řetězců[10]. Výsledek je uložen do příslušné buňky matice podobnosti `sim_matrix` (příklad vypočtené matice podobnosti je zobrazen v tabulce 4.1, vlevo). Jelikož je matice podobnosti symetrická, stačí vypočítat pouze část nad nebo pod diagonálou.

	Joan	**Johanes**	**Johan**	**Petr**	**Peter**
Joan	1	0.72	0.88	0	0
Johanes	0	1	0.83	0.18	0.16
Johan	0	0	1	0	0
Petr	0	0	0	1	0.88
Peter	0	0	0	0	1

	Joan	**Johanes**	**Johan**	**Petr**	**Peter**
Joan	**1**	**1**	0.88	0	0
Johanes	0	1	0.83	0.18	0.16
Johan	0	0	1	0	0
Petr	0	0	0	1	0.88
Peter	0	0	0	0	1

Tabulka 4.1: Ukázka vypočtené matice podobnosti pro pět jmen. V levé tabulce je zobrazena původní vypočtená matice podobnosti. V pravé tabulce je zobrazena upravená matice podobnosti tak, aby slova, která jsou již normalizována a jejich normalizace jsou shodné (v tomto případě se pro ukázku jedná o jména Joan a Johanes), měla podobnost 1. Tučná čísla v pravé matici vznikla upravením původní matice podobnosti.

Výpočet podobnosti dvojice slov zajišťuje metoda `similarity_metric` na základě parametrů shlukování. Při použití fonetického kódování slov je zvlášť vypočtena podobnost foneticky zakódovaných slov (`sim_encoded_words`)[11] a zvlášť podobnost slov bez použití fonetického kódování (`sim_not_encoded_words`). Tyto podobnosti jsou následně zkombinovány [5] v poměru 35:65. Výsledná podobnost je tedy dána vztahem $0.35 \times$ `sim_encoded_words` $+$ $0.65 \times$ `sim_not_encoded_words`. Váhu podobnosti foneticky kódovaných slov, která je implicitně nastavena na hodnotu 0.35, je možné nastavit pomocí parametru `phonetic_encoding_function_weight` předaném při instanciaci třídy. Pokud jsou slova krátká, je výsledná podobnost vypočtena pouze na základě foneticky kódovaných slov.

Po vypočtení podobnosti pro každou dvojici slov je matice podobnosti upravena tak, aby slova která již jsou normalizována a zároveň jejich normalizace jsou shodné (stejná hodnota atributu `norm_name_id`), měla podobnost 1 – tedy aby byla slova přiřazena do stejného

[10] Pro zefektivnění výpočtu jsou všechna slova foneticky zakódována a uložena do instanční proměnné `phonetic_encoded_words` před začátkem kroku Porovnání dvojic záznamů. Není tak nutné počítat pro každé slovo fonetické kódování několikrát.

[11] Výsledkem fonetického kódování Double-Metaphone mohou být až dva kódy pro každé slovo. V tomto případě jsou vypočteny podobnosti pro všechny kombinace těchto kódu (maximálně 4 kombinace) a dále je použita největší podobnost.

shluku slov[12]. Toto zajišťuje metoda `set_normalized_words_similarity` (příklad matice podobnosti před a po explicitním nastavením podobnosti slov je zobrazen v tabulce 4.1).

Dalším krokem je klasifikace porovnaných dvojic slov do tříd. Zde byla využita metoda prahování s jednotným prahem, tedy klasifikace do dvou tříd (shodné, neshodné), která byla popsána v kapitole č. 2.3.4. Klasifikaci zajišťuje metoda `classification`. Po invokaci této metody je v matici podobnosti `sim_matrix` porovnána podobnost každé dvojice s prahem podobnosti `threshold`. Pokud je daná podobnost větší nebo rovna prahu podobnosti, do ekvivalentní buňky matice `cluster_matrix` je uloženo číslo řádku matice, na kterém se daná buňka nachází [5] (příklad vypočtené matice shluků `cluster_matrix` je zobrazen v tabulce 4.2, vlevo). Posledním krokem klasifikace je tranzitivní uzavření záznamů, které je aktivováno na základě parametru `transitive_closure` předaném při instanciaci třídy. Algoritmus pro tranzitivní uzavření záznamů jsem navrhl tak, že matice `cluster_matrix` je procházena po sloupcích od konce. Pro každý sloupec je zjištěna nejmenší hodnota `col_min`, která se v tomto sloupci nachází. Pokud se v daném sloupci nacházejí i buňky s hodnotou větší než nalezená nejmenší hodnota, jsou hodnoty těchto buněk, včetně buněk na stejném řádku jako tyto buňky, změněny na hodnotu `col_min`. Procházení matice je ukončeno, pokud při předešlém průchodu nebyla změněna žádná buňka. Příklad původní matice shluků a matice shluků po tranzitivním uzavření je zobrazen v tabulce 4.2.

	Joan	**Johanes**	**Johan**	**Petr**	**Peter**
Joan	1	0	1	0	0
Johanes	0	2	2	0	0
Johan	0	0	3	0	0
Petr	0	0	0	4	4
Peter	0	0	0	0	5
číslo shluku	**1**	**2**	**1**	**4**	**4**

	Joan	**Johanes**	**Johan**	**Petr**	**Peter**
Joan	1	0	1	0	0
Johanes	0	**1**	**1**	0	0
Johan	0	0	**1**	0	0
František	0	0	0	4	4
Franta	0	0	0	0	**4**
číslo shluku	**1**	**1**	**1**	**4**	**4**

Tabulka 4.2: Tabulka vlevo zobrazuje vypočtenou matici shluků (`cluster_matrix`) vzniklou z původní matice podobnosti použitím klasifikace prahováním s prahem $t = 0.8$, která je zobrazena v tabulce 4.1, vlevo. Tabulka vpravo zobrazuje původní matici shluků (tabulka vlevo) po tranzitivním uzavření záznamů. Díky tomu, že dvojice jmen (Joan, Johan) a (Johan, Johanes) byly klasifikovány jako shodné, po tranzitivním uzavření je i dvojice (Joan, Johanes) klasifikována jako shodná. Tučně jsou zvýrazněna čísla, která byla změněna při procesu tranzitivního uzavření záznamů. Poslední řádek obou tabulek obsahuje výsledná čísla shluků jednotlivých slov. Ty jsou získány jako nejmenší číslo ve sloupci každého slova.

Posledním krokem procesu roztřídění slov do shluků je aktualizace čísel shluků v databázi. Toto zajišťuje metoda `db_update_words_cluster_id`. Nejdříve jsou z vypočtené matice shluků `cluster_matrix` získány čísla shluků jednotlivých slov. Číslo shluku odpovídá nejmenšímu číslu ve sloupci daného slova [5]. Získání čísel shluků pro jednotlivá slova je zobrazeno v tabulce 4.2. U každého slova v seznamu `words` je pak v databázi aktualizováno číslo shluku `cluster_id`.

[12]Tímto je zajištěno spojení shluků slov, které by nikdy spojeny při shlukování nebyly (např. shluky se slovy George a Jiří) – pokud je použito tranzitivní uzavření záznamů.

Spojení shluků slov pomocí normalizace

Jak už bylo řečeno — pokud dvěma slovům v odlišných shlucích (odlišná hodnota atributu `cluster_id`) byla přiřazena stejná normalizovaná varianta (stejná hodnota atributu `norm_name_id`), mají být tyto shluky spojeny v jeden shluk. Tohoto je možné docílit provedením metody `set_normalized_words_similarity`, která nastaví podobnost všech slov se stejnou normalizací na 1.0, čímž vzniknou vazby mezi shluky ve kterých se tyto slova nacházejí. Při použití tranzitivního uzavření záznamu budou pak tyto shluky spojeny v jeden shluk. Pokud však není využito tranzitivní uzavření záznamů, metodou `set_normalized_words_similarity` bude pouze docíleno umístění slov se stejnou normalizací do jednoho shluku, nikoliv spojení shluků ve kterých se tyto slova nachází.

Aby mohly být spojeny shluky ve kterých se nachází slova se stejnou normalizací i bez použití tranzitivního uzavření záznamů, je všem slovům v těchto shlucích přiřazena podobnost 1.0.

Optimalizace a urychlení shlukování

Nejnáročnější část výše popsané implementace procesu roztřídění slov do shluků je část Porovnání, kterou implementuje metoda `comparison`. V této části je nutné vypočítat míru podobnosti pro každou dvojici slov a tuto hodnotu uložit do matice podobnosti `sim_matrix`. Pro roztřídění 3947 slov do shluků je nutno vypočítat podobnost pro 7 787 431 dvojic slov. Roztřídění těchto slov do shluků zabere na serveru `perun.fit.vutbr.cz` $\sim$ 38 minut, z toho část Porovnání $\sim$ 37 minut.

Prvním krokem pro urychlení výpočtu shlukování je snížení počtu slov pro porovnání – databáze slov z matričních záznamů je navržena tak, že může obsahovat několik stejných slov, která se liší pouze pohlavím nebo aktuální přiřazenou normalizovanou variantou. Při výpočtu podobnosti se zohledňuje pouze vstupní slovo, nikoliv pohlaví (to zohledňuje až webová aplikace) – stejná slova budou tedy přiřazena do stejného sluku. Z tohoto důvodu je možné načíst pro shlukování každé slovo pouze jednou a přiřazené číslo shluku aktualizovat v databázi u všech stejných slov. Z původních 3947 slov bylo následně načteno pouze 3126 unikátních slov. Celkem se v tomto případě snížil počet dvojic slov k porovnání o 2 903 056, na 4 884 375 a čas potřebný k roztřídění těchto slov do shluků o $\sim$ 13 minut, na $\sim$ 24 minut.

Další možností pro urychlení roztřídění slov do shluků je zavedení paralelismu. Výpočet podobnosti jednotlivých dvojic slov je na sobě nezávislý. Je tedy možné rozdělit tyto dvojice mezi více procesů a počítat jejich podobnost paralelně. V tabulce 4.3 je zobrazeno rozdělení jednotlivých řádků matice podobnosti mezi více procesů (počet procesů je daný počtem jader procesoru).

Při použití paralelního výpočtu matice podobnosti byl snížen čas potřebný pro roztřídění předešlých 3126 unikátních slov o $\sim$ 20 minut, na $\sim$ 4 minuty. Celkově tedy v tomto případě optimalizace snížila čas shlukování z původních $\sim$ 38 minut na $\sim$ 4 minuty.

Paralelismus byl implementován pomocí knihovny `multiprocessing`. Nejdříve je zjištěn počet jader procesoru a vytvořeno právě tolik procesu. Každému procesu je předána metoda `comparison` a parametry `process_number` a `process_count`, které určují které řádky matice `sim_matrix` má daný proces zpracovat. Po vypočtení všech potřebných buněk matice podobnosti jsou tyto procesy ukončeny a dále pokračuje pouze hlavní proces. Aby bylo možné sdílet paměť mezi procesy, matice podobnosti musela být implementována jako třída `multiprocessing.Array`, která je následně namapována na `numpy.Array`. Kvůli

GIL[13] není možný souběžný běh více vláken, proto byl paralelismus implementován pomocí procesů. Ukázka naměřených časů shlukování pro různé typy slov je zobrazena v tabulce 4.4.

Číslo procesu		**Joan**	**Johanes**	**Johan**	**Petr**	**Peter**
proces 1	**Joan**	1	0.72	0.88	0	0
⋮	**Johanes**	0	1	0.83	0.18	0.16
proces N	**Johan**	0	0	1	0	0
proces 1	**Petr**	0	0	0	1	0.88
⋮	**Peter**	0	0	0	0	5

Tabulka 4.3: Rozdělení řádků matice podobnosti mezi více procesů. Každý proces má přidělené řádky obsahující dvojice slov, jejichž podobnost musí vypočítat. Jelikož se na horních řádcích nachází více dvojic slov než na nižších – řádky jsou rozděleny tak, aby byly počty dvojic rozděleny mezi procesy co nejvíce rovnoměrně. Počet procesů N je dán počtem dostupných jader procesoru.

Typ slov	**Čas shlukování**	
	1 proces	**12 procesů**
Jméno	6.63 min	26 sec
Příjmení	14. 36 min	1.09 min
Povolání	1.36 min	12.73 sec
Obce	2.98 min	27.87 sec
Vztahy mezi lidmi	41.71 sec	2.61 sec

Tabulka 4.4: Tabulka zobrazuje naměřené časy shlukování s použitím paralelismu a bez použití paralelismu. Všechny shlukování byly provedeny s fonetickým kódováním Double-Metaphone a funkcí pro výpočet podobnosti řetězců Longest Common Substring.

4.2.2 Výpočet nejpodobnějšího shluku pro zadané slovo

V této sekci je popsána implementace části **c** návrhu řešení, který je zobrazen na obrázku 3.2. Cílem je nalézt pro zadané slovo číslo shluku, ve kterém se nachází nejvíce podobné slovo se zadaným slovem. Tento algoritmus implementuje metoda `get_best_cluster_id_of_unknown_word`. Vstupním parametrem je pouze zadané slovo `word`, pro které má být nalezeno číslo nejpodobnějšího shluku slov. Databázová tabulka, fonetické kódování, funkce podobnosti řetězců a prah podobnosti jsou zadány při instanciaci třídy `WordClustering`. Tato metoda je volána z webové aplikace DEMoS asynchronně, pomocí technologie AJAX. Výsledkem je buď číslo nalezeného shluku, nebo v případě neúspěchu hodnota `-1`.

Po invokaci této metody jsou pomocí metody `db_get_words` načtena slova z databáze. Použitím parametru `clustered_only` je zajištěno načtení pouze těch slov, která již byla shlukována (mají vyplněný atribut `cluster_id`). Každé z těchto slov je poté porovnáno se

[13]GIL - https://wiki.python.org/moin/GlobalInterpreterLock

zadaným slovem `word` s využitím parametrů porovnávání, které byly zvoleny při instanciaci třídy. Pro slovo s největší podobností ke vstupnímu slovu `word` je pak pomocí jazyka SQL nalezeno v databázi číslo shluku `cluster_id`, jehož hodnota je vrácena jako výsledek.

4.3 Webové rozhraní pro normalizaci slov

Ve webové aplikaci DEMoS je možné slova z matričních záznamu normalizovat buďto přímo při vytváření/editaci záznamů, nebo hromadně, na samostatné stránce k tomu určené. V první části této podkapitoly bude popsáno aplikační rozhraní pro normalizaci slov. Druhá část bude věnována popisu normalizace slov při vytváření/editaci záznamů. V poslední části bude pak popsána implementace hromadné normalizace slov. V rámci aplikace je normalizace nazývána jako standardizace. Jak už bylo řečeno, webová aplikace DEMoS je napsána v jazyku PHP.

4.3.1 Aplikační rozhraní pro normalizaci slov

V této podkapitole budou popsány důležité třídy a metody pro normalizaci slov v aplikaci DEMoS. Pro zajištění jednotného přístupu k datům při normalizaci byla vytvořena třída `NormalizationManager`, která zapouzdřuje veškerou práci s databází, spojenou s normalizací. Součástí třídy jsou metody pro získání normalizovaných variant podle různých kritérií, metoda pro aktualizaci normalizované varianty daného slova a další. V následujícím seznamu jsou tyto metody popsány:

- `getNormalizedVariantsOfWord` – získání normalizovaných variant, které již byly použity pro stejná slova jako zadané slovo.
- `getNormalizedVariantsOfWordsInSameClusterAsWord` – získání normalizovaných variant použitých u slov ve stejném shluku jako zadané slovo.
- `getNormalizedVariantsOfWordAndWordsInSameCluster` – kombinace dvou předešlých metod. Výsledkem je asociativní pole obsahující zvlášť normalizované varianty (pole s názvem `word`) použité u stejných slov jako zadané slovo a zvlášť normalizované varianty (pole s názvem `cluster`) použité u slov ve stejném shluku jako zadané slovo. Pokud existuje normalizovaná varianta v poli `word`, není již pak přidána do pole `cluster`. Metoda také umožňuje zadat číslo shluku místo slova. V tomto případě je výsledné pole `word` prázdné.
- `getNormalizedVariantsOfUnknownWord` – metoda vypočte nejpodobnější shluk pro zadané slovo a vrátí normalizované varianty použité u slov v tomto shluku. Při výpočtu je spuštěn skript pro roztřídění slov do shluků s prvním parametrem 'cluster_id'. Jedná se o implementaci části **c** a **d** návrhu řešení, který je zobrazen na obrázku 3.2.
- `getWordsForNormalization` – načtení slov pro hromadnou normalizaci. Metoda podporuje stránkování a filtrování podle atributů.
- `updateNormalizedVariantOfWord` – aktualizace normalizované varianty zadaného slova. Tuto variantu lze zadat pomocí svého číselného identifikátoru nebo v podobě řetězce. Při zadání řetězce je buďto vytvořena nová normalizovaná varianta nebo nalezena stejná, již existující normalizovaná varianta. Danému slovu je pak tato normalizovaná varianta přiřazena.

- `getWordFrequency` – získání počtu použití daného slova v matričních záznamech.
- `getMostFrequentWordOfCluster` – získání slova, které je nejčastěji použito v matričních záznamech v rámci zadaného shluku slov. Tento shluk slov je identifikován číslem shluku.

Aby bylo možné navrhovat normalizované varianty zadaných slov asynchronně na straně klienta, aniž by musela být znovu načtena celá stránka, byly pro tyto metody vytvořeny API endpoint url, které vrací výsledky ve formátu JSON.

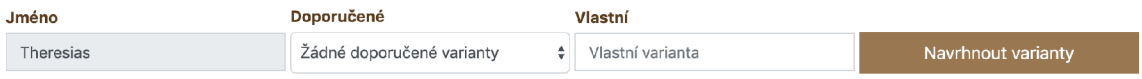
Pro umožnění návrhu normalizovaných variant slov na základě formulářových polí, do kterých byla tato slova zapsána ve webové aplikaci DEMoS, je nutné znát u těchto polí dodatečné informace. Z tohoto důvodu byla rozšířena již existující třída `TableNames` o seznam názvů jednotlivých formulářových polí, které jsou určeny k normalizaci. Pro každé pole je dostupný název databázové tabulky, ve které je slovo uloženo (název databázové tabulky s normalizovanými variantami slov je možné odvodit z názvu této tabulky), přípustné pohlaví a názvy databázových tabulek pomocí kterých je dané slovo navázáno na konkrétní osobu (slouží pro určení frekvence výskytu v záznamech).

Pohlaví daného pole je důležité pouze u některých typů slov (jména a příjmení). U těchto typů slov je nutné při návrhu doporučených normalizovaných variant vybrat pouze ty varianty, které jsou použity u slov daného pohlaví. Jak už bylo řečeno, v aplikaci existují tři typy pohlaví – mužské (`'M'`), ženské (`'F'`) a neznámé ('U'). Pokud formulářové pole zadaného slova je pohlaví typu `'M'` nebo `'F'`, vyhledány jsou varianty pro toto pohlaví a navíc varianty pro pohlaví `'U'`. Pokud je pohlaví pole typu `'U'`, není typ pohlaví zohledněn. Navržené varianty jsou pak seřazeny podle frekvence výskytu v záznamech.

4.3.2 Normalizace slov při vytváření/editaci matričních záznamů

Součástí webové aplikace DEMoS již je formulář pro vytvoření/editaci matričního záznamu. Tento formulář obsahuje sekce pro zápis údajů o křestiteli, otci, matce, kmotrech a dalších. Jména, příjmení, povolání, obce a další údaje v těchto sekcích jsou určeny k normalizaci. Ukázka tří sekcí (porodní bába, dítě, otec) tohoto formuláře je zobrazena na obrázku 4.3.

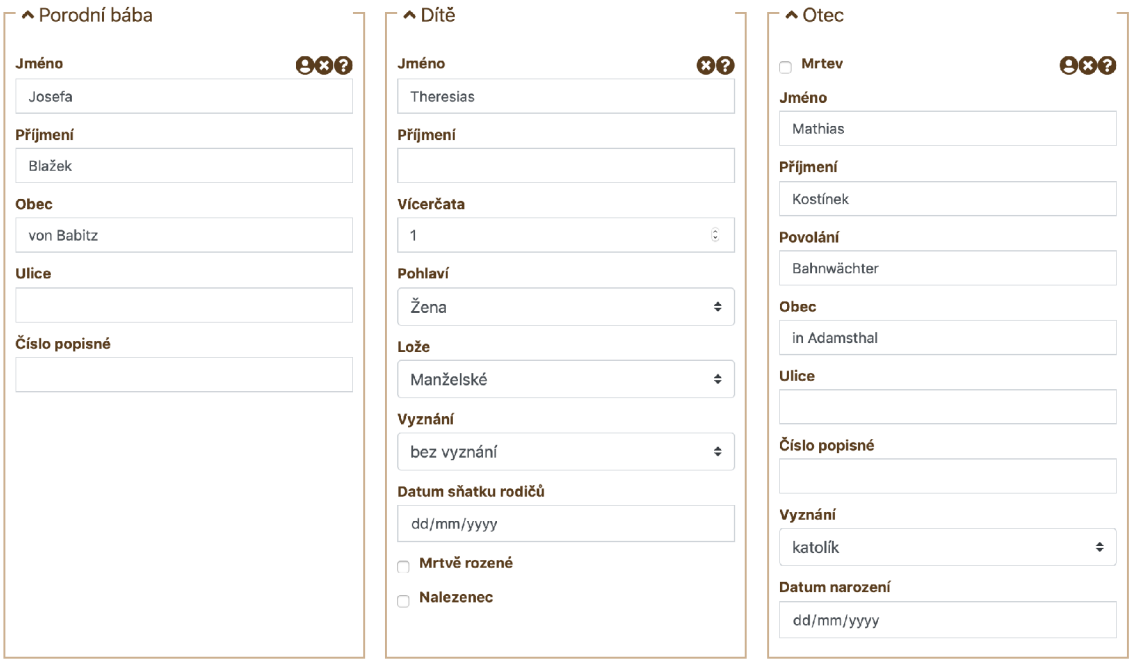
Tento formulář byl rozšířen o normalizační část, která se nachází na jeho konci. Ve výchozím stavu je tato část formuláře skryta – lze ji zobrazit pomocí tlačítka „Zobrazit standardizaci“, umístěném na konci formuláře. Pro každé pole původního formuláře, které je určeno k normalizaci, jsou v normalizační části přidány čtyři formulářové prvky, pomocí kterých je možné slovo v daném poli normalizovat:



Obrázek 4.2: Formulářové prvky pro normalizaci uživatelem zadaného slova.

- neměnné textové pole (`input type="text"`) s kopií zadaného slova určeného k normalizaci,
- výběrové pole (`select`) pro výběr doporučených normalizovaných variant, které již v databázi existují,
- textové pole pro možnost zadání vlastní normalizované varianty,

- tlačítko pro možnost explicitního návrhu normalizovaných variant (výpočet nejpodobnějšího shluku pro zadané slovo a navržení normalizovaných variant použitých u slov v tomto shluku), které je zobrazeno pouze pokud zadané slovo nebylo nalezeno v databázi. Jedná se o implementaci části **c** a **d** návrhu řešení, který je zobrazen na obrázku 3.2.



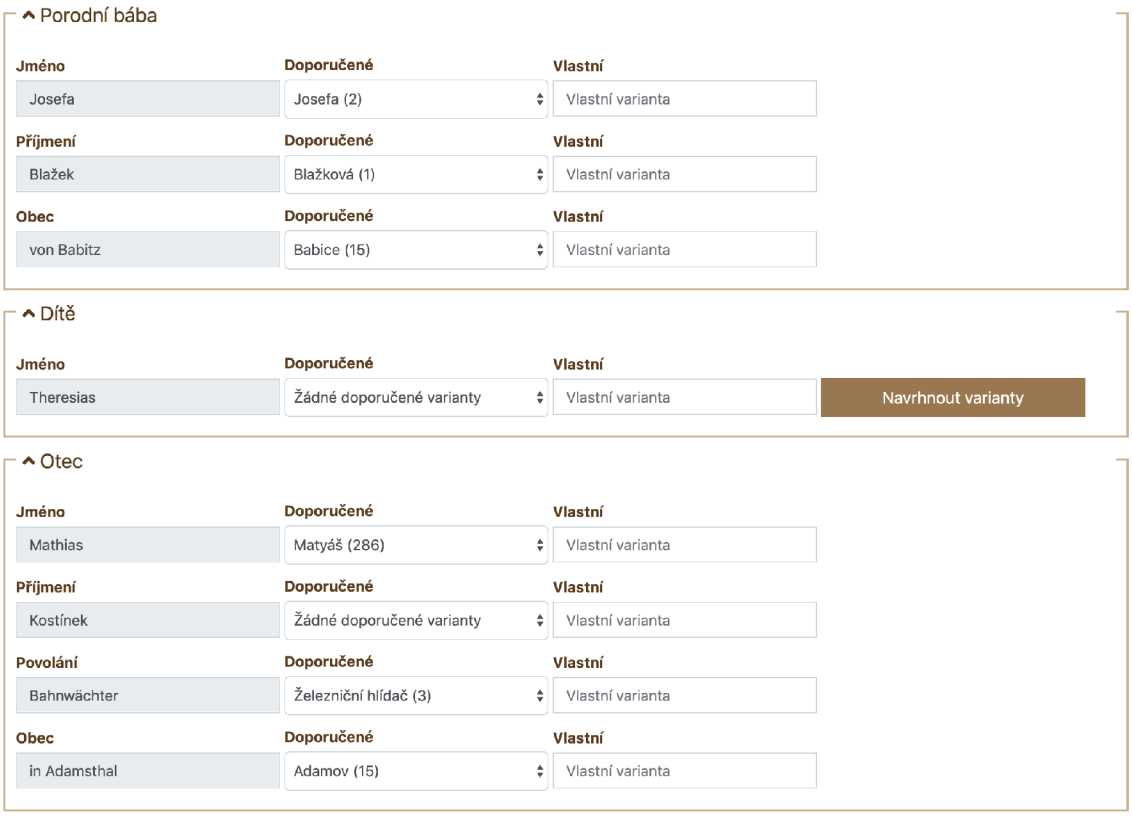
Obrázek 4.3: Ukázka části již existujícího formuláře pro vytvoření/editaci matričního záznamu ve webové aplikaci DEMoS. Formulář je rozdělen do sekcí.

Formulářové prvky pro normalizaci slov jsou rovněž rozděleny do sekcí podle původního formuláře a jsou viditelné pouze pokud bylo vyplněno příslušné slovo v části původní formuláře. Pokud není v sekci (např. otec, matka atd.) původního formuláře vyplněno žádné pole, tato sekce je v normalizační části formuláře skryta. Na obrázku 4.4 je zobrazena ukázka normalizační části formuláře pro již vyplněná pole původního formuláře, který je zobrazen na obrázku 4.3.

Normalizace probíhá asynchronně na pozadí. Vše je interaktivní – pole v normalizační části formuláře se skrývají, odkrývají a doplňují v reálném čase podle aktuálně zadaných slov v polích původního formuláře. Vyplněná normalizační část formuláře je odeslána na server spolu s původním formulářem při uložení (událost submit). V případě doporučené normalizované varianty je odeslán číselný identifikátor dané varianty na kterou má být slovo normalizováno. U vlastní normalizované varianty je odesláno celé slovo, na jehož základě je pak buď vytvořena nová normalizovaná varianta, nebo nalezena již existující varianta v databázi. Na tuto variantu je pak zadané slovo normalizováno (její identifikátor je přiřazen do atributu `norm_name_id` příslušného slova). Zadání vlastní normalizované varianty má větší prioritu než výběr z doporučených normalizovaných variant.

Veškeré chování spojené s normalizaci slov při vkládání/editaci záznamů je naprogramováno v souboru `normalizationAddEditRecord.js` pomocí jazyka JavaScript a knihovny JQuery. Po načtení stránky s formulářem jsou pomocí identifikátorů nalezeny jednotlivá pole původního formuláře, která jsou určena k normalizaci. Na tyto je navázána událost

`onChange`, která při zadání slova vyvolá jeho předání do normalizační části formuláře, zobrazení/skrytí příslušné normalizační části formuláře a aktualizaci doporučených normalizovaných variant pro zadané slovo.



Obrázek 4.4: Ukázka normalizační části formuláře pro vytvoření/editaci záznamu ve webové aplikaci DEMoS. Formulář je rozdělen do sekcí. V prvním sloupci jsou vyplněna slova z původních polí formuláře. Druhý sloupec obsahuje doporučené normalizované varianty pro tyto slova. Ve třetím sloupci je možné zadat vlastní normalizovanou variantu, pokud by žádná doporučená varianta uživateli nevyhovovala. Poslední sloupec obsahuje tlačítka pro explicitní navržení normalizovaných variant. Všechna zadaná slova, kromě jména Theresias, již existují v databázi. Příjmení Kostínek sice existuje v databázi, ale nejsou pro něj dostupné žádné normalizované varianty. Jméno Theresias neexistuje v databázi a je tedy pro něj možné vyhledat nejpodobnější shluk slov v databázi a poté navrhnout normalizované varianty použité u slov v tomto shluku.

Aktualizace doporučených normalizovaných variant při zadání slova

Jak už bylo řečeno, při zadání slova do pole původního formuláře dojde k aktivaci události `onChange` a tedy i k návrhu doporučených normalizovaných variant pro toto pole. Návrh je realizován odesláním zadaného slova a názvu formulářového pole do kterého bylo toto slova zadáno (atribut `name`) pomocí již zmíněných API endpoint url na server, asynchronně použitím technologie AJAX. Na straně serveru dojde k invokaci metody `getNormalizedVariantsOfWordAndWordsInSameCluster`, které je předáno zadané slovo a název daného formulářového pole. Na základě názvu pole jsou pak na serveru pomocí třídy `TableNames`

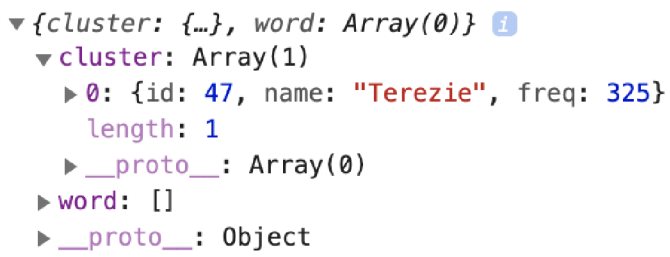
zjištěny dodatečné informace o tomto poli a poté vyhledány doporučené normalizované varianty, které jsou následně odeslány zpět klientovi ve formátu JSON. Po přijetí odpovědi je pak v normalizační části formuláře, pomocí jazyka JavaScript, aktualizováno příslušné výběrové pole s doporučenými normalizovanými variantami. Jedná se o implementaci části **b** návrhu řešení, který je zobrazen na obrázku 3.2. Ukázka možné odpovědi ve formátu JSON je zobrazena na obrázku 4.5.

Aktualizace doporučených normalizovaných variant při explicitním požádání

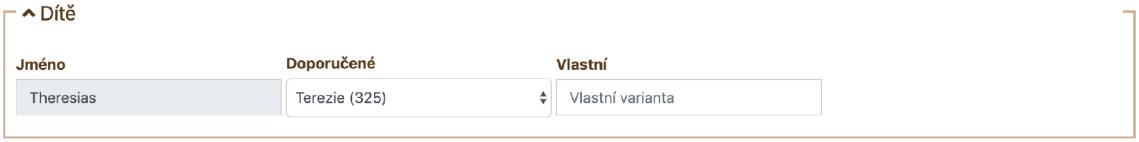
Po zadání slova do původního formuláře nemusí být vždy pro toto slovo nalezeny doporučené normalizované varianty. Toto může nastat v případě, že zadané slovo v databázi ještě neexistuje (případ jména Theresias na obrázku 4.4) nebo existuje, ale nejsou pro toto slovo k dispozici žádné normalizované varianty (případ příjmení Kostínek na obrázku 4.4). V prvním z těchto případů je zobrazeno tlačítko „Navrhnout varianty“, pomocí kterého má uživatel možnost explicitně požádat o návrh doporučených normalizovaných variant. Jedná se o implementaci částí **c** a **d** návrhu řešení, který je zobrazen na obrázku 3.2.

Po kliknutí na toto tlačítko je zadané slovo a název formulářového pole odeslán asynchronně na server pomocí zmíněných API endpoint url. Na straně serveru dojde k invokaci metody `getNormalizedVariantsOfUnknownWord`, které je předáno zadané slovo a název formulářového pole. Na základě názvu pole jsou pak na serveru pomocí třídy `TableNames` zjištěny dodatečné informace o tomto formulářovém poli. Zadané slovo je pak porovnáno se všemi slovy daného typu v databázi, která jsou roztříděná do shluků. U slova, které je nejvíce podobné zadanému slovu a zároveň tato podobnost je větší než stanovený prah podobnosti, je zjištěno číslo shluku. Pomocí metody `getNormalizedVariantsOfWordsInSameClusterAsWord` jsou pak získány normalizované varianty slov v tomto shluku, které jsou následně odeslány zpět klientovi jako odpověď ve formátu JSON. Po přijetí odpovědi je pak rovněž aktualizováno, pomocí jazyka JavaScript, výběrové pole s doporučenými normalizovanými variantami v normalizační části formuláře.

Proces porovnání zadaného slova s ostatními slovy a navrácení čísla nejpodobnějšího shluku je implementován v rámci třídy `WordClustering`, která je součástí souboru `WordClustering.py` pomocí metody `get_best_cluster_id_of_unknown_word`. Jelikož se nejedná o pouhý databázový dotaz, ale je nutné spustit externí skript, není tato akce provedena automaticky, ale až po kliknutí na zmíněné tlačítko „Navrhnout varianty“. Na obrázku 4.5 je zobrazena odpověď ve formátu JSON, přijatá po klinutí na tlačítko „Navrhnout varianty“ nacházejícím se na obrázku 4.4. Obrázek 4.6 zobrazuje aktualizované doporučené normalizované varianty pro jméno Theresias.



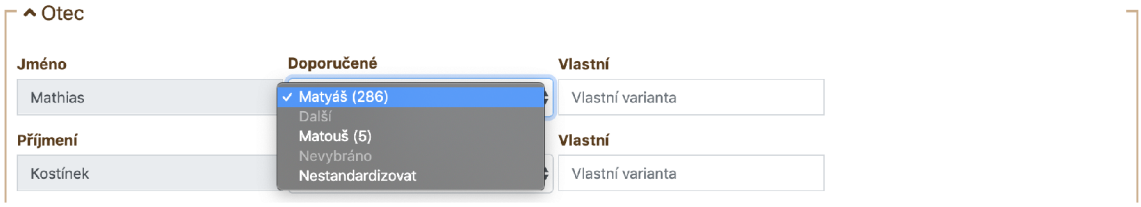
Obrázek 4.5: Ukázka odpovědi s navrženými normalizovanými variantami ve formátu JSON. U každé normalizované varianty je dostupné id, název této varianty a počet použití v ostatních záznamech.



Obrázek 4.6: Ukázka aktualizovaného výběrového pole s doporučenými normalizovanými variantami.

Výběr normalizované varianty

Po přijetí odpovědi s normalizovanými variantami pro zadané slovo jsou tyto varianty vloženy do příslušného formulářového výběrového prvku v normalizační části formuláře. Odpověď ve formátu JSON obsahuje kolekci variant (`word`) použitých u stejných slov jako zadané slovo a kolekci variant (`cluster`) použitých u slov ve stejném shluku jako zadané slovo. Nejdříve jsou do výběrového prvku vloženy varianty z kolekce `word` (mají větší prioritu). Následně pak do samostatné sekce s názvem „Další“ varianty z kolekce `cluster`. Jako poslední je vložena možnost „Nestandardizovat“, pokud by uživatel nechtěl dané slovo normalizovat. Ukázka vyplněného výběrového prvku s navrženými normalizovanými variantami je zobrazena na obrázku 4.7.



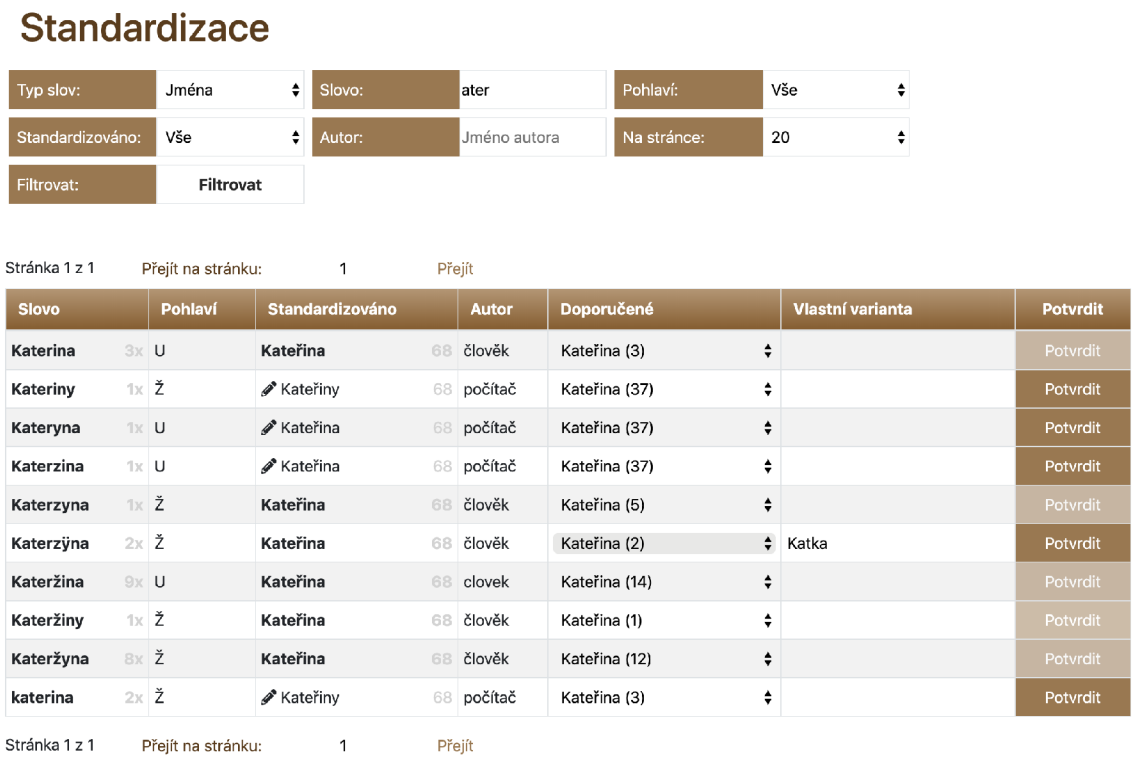
Obrázek 4.7: Výběr doporučených normalizovaných variant z formulářového výběrového prvku. Pro zadané jméno Mathias jsou na výběr dvě doporučené normalizované varianty. Slovo Mathias již bylo u 286 záznamů normalizováno jako Matyáš, proto tato varianta je jako první na výběr a je tedy i automaticky vybrána. Slovo Mathias se navíc nachází ve shluku slov, kde pět z těchto slov již bylo normalizováno jako Matouš, proto v sekci Další je na výběr varianta Matouš. Poslední možností je nezvolit žádnou normalizovanou variantu.

V závorce u každé doporučené varianty je zobrazen počet použití dané normalizované varianty pro zadané slovo (varianty v kolekci `word`), resp. pro slova ve stejném shluku jako je zadané slovo (varianty v kolekci `cluster`). Navržené varianty jsou seřazeny podle počtu použití. První doporučená normalizovaná varianta tedy odpovídá nejvhodnější normalizované variantě pro zadané slovo a je vybrána automaticky. Uživatel tedy nemusí normalizační část formuláře otevřít, pokud nechce – automaticky budou vybrány nejvhodnější normalizované varianty pro zadaná slova.

4.3.3 Hromadná normalizace slov z matričních záznamů

Hromadnou normalizací slov je myšlena možnost vypsání slov k normalizaci ze všech záznamů na jedné stránce, kde tyto slova bude možné normalizovat. Tato funkce je implementována na stránce `http://perun.fit.vutbr.cz/demos/standardization`. Součástí této stránky je filtrační formulář, díky kterému je možné vypsat pouze požadovaná slova. Slova lze filtrovat podle typu (jména, příjmení, povolání, obce atd.), pohlaví, autora normalizace, stavu (normalizováno/nenormalizováno) a obsahu. Aby nebylo vypsáno najednou

příliš mnoho slov, je využito stránkování obsahu. Počet slov na jedné stránce lze také nastavit. Na obrázku 4.8 je zobrazena ukázka stránky pro hromadnou normalizaci slov.



Obrázek 4.8: Stránka pro hromadnou normalizaci slov z matričních záznamů. V horní části se nachází filtrační formulář. Pod tímto formulářem je tabulka pro normalizaci slov. Každý řádek tabulky odpovídá jednomu slovu k normalizaci. Ve sloupci Standardizováno jsou zobrazeny aktuální normalizované varianty jednotlivých slov (šedou barvou je zobrazeno číslo shluku tohoto slova). Ve sloupci Autor je vypsán autor této normalizace. Dále je zde sloupec s doporučenými normalizovanými variantami a sloupec pro zápis vlastní varianty. Zadání vlastní normalizované varianty má větší prioritu než výběr z doporučených normalizovaných variant. Na toto je uživatel upozorněn změnou barvy doporučených normalizovaných variant při zadání vlastní varianty. Tlačítko Potvrdit je možné stisknout pouze, pokud je pro zadané slovo vybrána normalizovaná varianta, která by vedla ke změně jeho normalizace.

Příprava slov pro hromadnou normalizaci

Proces vykreslení této stránky je implementován metodou `renderDefault`, která se nachází v presenteru `StandardizationPresenter`. Jelikož je pro odeslání filtračního formuláře zvolena metoda `GET`, protokolu `HTTP`, parametry filtrování jsou předány v rámci `URL` adresy. Tyto parametry jsou následně předány již zmíněné metodě `getWordsForNormalization`, která načte požadovaná slova pro hromadnou normalizaci. Pro každé z těchto slov jsou dodatečně získány doporučené normalizované varianty voláním metody `getNormalizedVariantsOfWordAndWordsInSameCluster`, které je předáno příslušné slovo, pohlaví a název databázové tabulky, ve které je toto slovo uloženo (typ slova). Dále je u každého slova voláním metody `getWordFrequency` získán počet použití daného slova v záznamech. Pro slova, která zatím nebyla normalizována, je navíc pomocí metody `getMostFrequentWord-`

`OfCluster` zjištěno slovo, které se nachází ve stejném shluku jako zadané slovo a je nejčastěji používané v záznamech. Toto slovo bude poté vypsáno ve sloupci Standardizováno u slov, která zatím nebyla normalizována (autorem bude uveden počítač).

Vykreslení slov pro hromadnou normalizaci

Jak už bylo řečeno, slova pro hromadnou normalizaci jsou vypsána pomocí tabulky, kde každý řádek představuje jedno slovo určené k normalizaci. Po vykreslení stránky již nejsou dodatečně doporučovány další normalizované varianty pomocí asynchronní komunikace se serverem, jako tomu bylo u normalizace při vytváření/editaci matričních záznamů.

Interaktivní změny formuláře jako například zapínání/vypínání tlačítka Potvrdit a výběrového prvku s doporučenými normalizovanými variantami, jsou implementovány pomocí jazyka JavaScript v souboru `standardizationPage.js`. V rámci toho souboru je také implementována reakce na stisknutí tlačítka Potvrdit – vybraná normalizovaná varianta, resp. zadaná vlastní normalizovaná varianta, je odeslána asynchronně na server pomocí zmíněných API endpoint URL, kde je následně invokována metoda `updateNormalizedVariantOfWord`. Této metodě je předán: identifikátor slova, které má být normalizováno, název databázové tabulky, ve které je toto slovo uloženo a zadané normalizované varianty. Pokud uživatel zadal vlastní normalizovanou variantu, nejdříve je zjištěno jestli taková normalizovaná varianta v databázi již existuje, popř. je vytvořena nová normalizovaná varianta. Pokud uživatel nezadal vlastní normalizovanou variantu, je k dispozici identifikátor normalizované varianty, která byla vybrána z doporučených normalizovaných variant. Takto získaný identifikátor normalizované varianty je pak uložen do atributu `norm_name_id` záznamu tohoto slova, čímž je slovo normalizováno.

4.4 Testování úspěšnosti shlukování slov

V této podkapitole bude otestována úspěšnost shlukování na různých typech slov (jména, příjmení, povolání, obce, vztahy mezi lidmi) a různých dobách zápisu těchto slov (stará čeština, latina, němčina, současná čeština). Pro každý typ slov bude otestováno několik kombinací nastavení shlukování (prah podobnosti, funkce pro fonetické kódování atributů, funkce podobnosti řetězců, tranzitivní uzavření záznamů). Na základě tohoto testování bude pro každý typ slov vybráno nejlepší nastavení shlukování, které bude použito pro finální shlukování jednotlivých typů slov.

Pro testování byla využita reálná slova z matričních záznamů, která byla přepsána členy projektu DEMoS z Masarykovy univerzity. Celkem bylo použito 6235 slov. Jedná se o slova, u kterých byla známa doba zápisu. Tato slova byla ručně roztříděna do shluků, čímž bylo vytvořeno referenční roztřídění slov *ground-truth*, které bylo využito při vyhodnocení testování úspěšnosti shlukování slov.

Pro určení úspěšnosti shlukování byla využita teorie popsaná v kapitole č. 2.3.5. Po každém procesu roztřídění slov do shluků byly oproti referenčnímu roztřídění slov do shluků *ground-truth* vypočteny hodnoty *true positives, false positives, true negatives a false negatives*, na jejichž základě byly vypočteny hodnoty metrik *Precision, Recall* a *F-measure*. Hodnota metriky *F-measure* byla použita pro určení kvality daného shlukování.

Proces roztřídění slov do shluků s následným určením kvality tohoto shlukování (hodnota *F-measure*) byl opakován pro jednotlivé typy slov s různým nastavením shlukování. Díky tomu bylo možné výsledky těchto shlukování vynést do grafů. Pro každý typ slov byly vytvořeny dva grafy zobrazující závislost úspěšnosti shlukování na prahu podobnosti

– s použitím a bez použití tranzitivního uzavření záznamů. Na každém z těchto grafů je vykreslena křivka pro každou kombinaci funkce podobnosti řetězců a fonetického kódování. U každé křivky je navíc vypsána maximální naměřená hodnota a prah podobnosti, který této hodnotě odpovídá. Dále je zde uveden průměrný čas jednoho shlukování slov při použití dané kombinace parametrů shlukování. Z těchto grafů lze tedy zjistit, jaké je nejlepší nastavení shlukování pro daný typ slov. Grafy úspěšnosti shlukování jsou součástí přílohy B.

Testované algoritmy

Pro testování byly vybrány algoritmy popsané v sekci č. 2.3.2 a 2.3.3. Z fonetických kódování byly vybrány tři nejznámější algoritmy: Soundex, Double-Metaphone a NYSIIS. Testována je i varianta bez fonetického kódování. Algoritmy Phonex a Phonix nebyly použity, jelikož se v obou případech jedná o rozšíření algoritmu Soundex, které nejsou tak často používané.

Pro výpočet podobnosti řetězců byly použity metody Damerau-Levenshtein, Q-gram, Jaro-Winkler a Longest Common Substring. Základní metoda Levenshtein nebyla použita, protože její výsledky jsou velice podobné rozšiřující metodě Damerau-Levenshtein. Metoda Q-gram byla použita v kombinaci s Jaccardovým koeficientem (2.5) a délkou Q-gram řetězců dva znaky. Metoda Longest Common Substring využívá koeficient Dice (2.15) a byla vypočtena pro každou kombinaci slov dvakrát (není symetrická).

Pro každý typ slov byly testovány všechny kombinace těchto parametrů na dvaceti prazích podobnosti slov, s použitím a bez použití tranzitivního uzavření záznamů. Dohromady tedy pro každý typ slov bylo provedeno 640 ($4 \times 4 \times 20 \times 2 = 640$) shlukování s otestováním úspěšnosti shlukování. Testování probíhalo na serveru `perun.fit.vutbr.cz`.

Implementace automatického testování

Pro účely testování bylo referenční roztřídění slov do shluků realizováno pomocí vyplnění atributu `norm_name_id` u každého slova (záznamu). V produkční databázi díky tomu vznikne uživateli definované *ground-truth*. Slova se stejnou hodnotou tohoto atributu jsou chápána jako navzájem shodná (Matches). Slova s odlišnou hodnotou tohoto atributu jsou brána jako navzájem neshodná (Non-Matches).

Pro otestování úspěšnosti aktuálního roztřídění slov do shluků byla implementována metoda `test`, která je součástí třídy `WordClustering`. Testování je prováděno nad databázovou tabulkou (typem slov), jejíž název byl předán při instanciaci třídy. Po invokaci této metody jsou z databáze načtena požadovaná slova, která již byla roztříděna do shluků. Poté jsou vypočteny hodnoty *true positives, false positives, true negatives, false negatives* a na jejich základě hodnoty metrik *Precision, Recall* a *F-Measure*. Vypočtené hodnoty jsou následně vypsány na standardní výstup. Lze aktivovat také výpis ve formátu CSV pomocí parametru této metody.

Testování různých kombinací parametrů shlukování s následným vygenerováním grafů pro každý typ slov je implementováno ve funkci `main` v souboru `WordClustering.py`. Toto testování lze aktivovat spuštěním skriptu s prvním parametrem `'tests'`. Pro každý typ slov jsou spuštěny a otestovány všechny možné kombinace zadaných parametrů shlukování (prah podobnosti, funkce podobnosti řetězců, funkce pro fonetické kódování řetězců, tranzitivní uzavření záznamů). Při každém testování je vždy vytvořena instance třídy `WordClustering` s aktuálními parametry shlukování. Následně je spuštěno samotné shlukování pomocí metody `execute` a nakonec je otestována úspěšnost daného shlukování pomocí metody `test`. Výsledky jednotlivých testů jsou uloženy do CSV souborů, které jsou pojmenovány podle

použité funkce pro fonetické kódování, funkce podobnosti řetězců a tranzitivního uzavření záznamů. Tyto soubory jsou automaticky roztříděny do složek podle názvu testované databázové tabulky (typu slov). Na základě dat v těchto souborech jsou následně vygenerovány grafy, které zobrazují úspěšnost shlukování (hodnota *F-Measure*) v závislosti na prahu podobnosti pro jednotlivé kombinace funkcí podobnosti řetězců a funkcí pro fonetické kódování řetězců. Pro generování grafů je použita knihovna `PyGnuPlot`.

Testování všech různých kombinací shlukování pro všechny typy slov je časově velmi náročné. Z tohoto důvodu je část vytváření CSV souborů oddělena od části generování grafů z dat v těchto souborech. Díky tomu není nutné počítat znovu data, která již jsou k dispozici.

Výsledky testování úspěšnosti shlukování

V tabulce 4.5 jsou pro jednotlivé typy slov vypsány nejlepší naměřené výsledky shlukování, společně s parametry použitými u těchto shlukování. Tyto parametry byly získány z grafů testování úspěšnosti shlukování, které jsou součástí přílohy B.

Typ slov	Funkce podobnosti řetězců	Fonetické kódování řetězců	Prah podobnosti	Tranzitivní uzavření záznamů	Úspěšnost shlukování (F-Measure)
jména	LCS	S	0.82	Ano	0.95
příjmení	LCS	S	0.825	Ano	0.85
povolání	LCS	DM	0.85	Ano	0.96
obce	JW	N	0.9	Ano	0.93
vztahy mezi lidmi	JW	N	0.85	Ne	0.96

Tabulka 4.5: Parametry s nejlepšími naměřenými výsledky shlukování pro jednotlivé typy slov. Tyto parametry jsou použity pro shlukování jednotlivých typů slov ve finální aplikaci.

S použitím nalezených nejlepších parametrů shlukování byla u jednotlivých typů slov otestována úspěšnost shlukování pro konkrétní pohlaví a jazyky. Výsledky těchto testování jsou uvedeny v tabulce 4.6.

Typ slov	Pohlaví			Jazyk			
	Muži	Ženy	Neznámé	stará čeština	němčina	latina	současná čeština
jména	0.94	0.94	0.94	0.92	0.92	0.97	0.97
příjmení	0.86	0.82	0.82	0.89	0.89	0.87	0.93
povolání				1.0	0.95	0.98	0.96
obce				0.97	0.98	0.86	1.0
vztahy mezi lidmi				1.0	0.89	0.96	0.99

Tabulka 4.6: Testování úspěšnosti shlukování slov jednotlivě pro konkrétní pohlaví a jazyky u všech typů slov. Jedná se o hodnoty metriky F-Measure. Použité parametry shlukování jednotlivých typů slov jsou zobrazeny v tabulce 4.5.

U všech typů slov kromě příjmení se pohybovala úspěšnost shlukování okolo hodnoty F-Measure 0.95. Horší úspěšnost shlukování byla zjištěna pouze u příjmení, kde byla namě-

řena nejlepší hodnota F-Measure 0.85. Tyto naměřené hodnoty úspěšnosti shlukování jsou dostatečně vysoké.

Uživateli definované referenční roztřídění slov do shluků

Nalezené nejlepší parametry shlukování nemusí být optimální, pokud dojde k výrazné obměně dat v databázi. Proto je vhodné tyto parametry časem změnit tak, aby odpovídaly současným datům v databázi.

Díky tomu, že v produkční databázi budou hodnoty atributů `norm_name_id` (hodnoty tohoto atributu jsou použity jako *ground-truth*) vyplňovat sami uživatelé prostřednictvím normalizace slov, lze pak pro toto uživateli definované referenční roztřídění slov do shluků spustit automatické testování úspěšnosti shlukování a zjistit tak aktuální nejlepší parametry shlukování.

Chyby shlukování

Většina chyb shlukování je způsobena spojením dvou shluků s velice podobnými slovy do jednoho většího shluku, nebo naopak rozdělením velkého shluku slov do několika menších shluků s navzájem více podobnými slovy. V tabulce 4.7 je zobrazen příklad chybného shlukování jmen odpovídajících jménu Ludmila. Pro detekci a výpis takovýchto problémových shluků byl vytvořen databázový dotaz, který je součástí souboru `/data/tests_queries.sql`. V tomto souboru se nachází i dalších dotazy pro kontrolu shlukovaných dat.

Slovo	**Referenční roztřídění**	**Výsledek shlukování**
Lidmilla	155	22
Lydmila	155	22
Lydmilla	155	22
Lydmyla	155	22
Ludmila	155	557
Ludmilla	155	557
Ludmillæ	155	557
ludmilla	155	557

Tabulka 4.7: Ukázka chybného shlukování jmen Ludmila. Všechna tato slova byla označena stejným číslem referenčního roztřídění slov do shluků. Předpokládá se tedy, že je shlukovací program umístí do stejného shluku. Výsledkem ale je, že při shlukování byla tato slova rozdělena do dvou shluků, které obsahují navzájem více podobná slova (řetězce byly porovnány metodou Jaro-Winkler, která klade větší důraz na rozdíly na začátku řetězců než na konci). Problém lze vyřešit snížením prahu podobnosti slov.

Úspěšnost shlukování je posuzována podle referenčních roztřídění slov do shluků (*ground-truth*). Určení správného roztřídění slov do shluků je obtížné. Pro každý typ slov nemusí existovat pouze jedno správné roztřídění slov. U některých slov například nelze jednoznačně rozhodnout, zda mají patřit do stejného shluku nebo ne (např. příjmení Chlebičžek a Chlebowský nebo Wybihal a Wybiral). V některých případech se slova nemusí příliš lišit a přesto představují dvě odlišné entity, nebo se naopak mohou slova lišit v několika znacích a přesto představují stejnou entitu (např. příjmení Wallaßzeckh a Walasek).

Další problém může nastat, pokud bylo použito pro shlukování slov tranzitivní uzavření záznamů. V takovém případě mohou být spojeny i velké shluky slov, mezi kterými existuje

pouze jediná vazba (slovo jednoho shluku klasifikované jako shodné se slovem druhého shluku). Pokud taková situace nastane, je možné ji vyřešit vypnutím tranzitivního uzavření záznamů a snížením prahu podobnosti. Vypnutí tranzitivního uzavření záznamů bylo nutné využít v produkční databázi.

4.5 Návrhy na rozšíření této práce

V této části budou popsány možnosti dalšího rozvoje této práce.

Databáze povolání a obcí

Tuto práci lze dále rozšířit integrací veřejně dostupných databází povolání (HISCAM) a obcí (RÚIAN). Při návrhu normalizovaných variant u povolání a obcí by byly navrhovány pouze varianty z těchto databází. Normalizované názvy obcí by následně navíc bylo možné spojovat s konkrétními obcemi (může existovat několik obcí se stejným názvem, které se nacházejí v jiných oblastech). Díky tomuto spojení by pak bylo umožněno podle názvu obce zobrazit záznamy o osobách na mapě.

Shlukování záznamů

Rozšířit lze také program pro roztřídění slov do shluků. V části předzpracování záznamů je možné přidat další pravidla pro čištění dat tak, aby ještě lépe odpovídala datům v matričních záznamech nebo rozšířit seznamy se „stop slovy“. V části porovnání záznamů je možné přidat další fonetická kódování a funkce pro porovnání řetězců.

Kapitola 5

Závěr

Cílem této práce bylo rozšířit webovou aplikaci pro správu matričních záznamů DEMoS o možnost roztřídění slov objevujících se v matričních záznamech a jejich následnou normalizaci (přiřazení normalizované podoby daným slovům).

Tento cíl byl splněn – uživatelé webové aplikaci mají možnost normalizace slov buď při vytváření/editaci matričních záznamů nebo navíc na samostatné webové stránce, kde lze normalizovat slova z matričních záznamů hromadně na jednom místě. Díky roztřídění slov objevujících se v matričních záznamech do shluků podobných slov jsou v rámci těchto shluků sdíleny normalizované varianty slov. Aplikace tak pro zadaná slova navrhuje normalizované varianty použité nejen u stejných slov, ale také i u podobných slov.

Pro účely normalizace slov během vytváření/editace matričních záznamů byl rozšířen původní formulář pro vytvoření/editaci matričního záznamu o normalizační část. Tato část se nachází na konci tohoto formuláře a poskytuje uživateli rozhraní pro normalizaci slov daného matričního záznamu. Veškeré informace o normalizaci aktuálního záznamu jsou tak uživateli dostupné na jednom místě. Normalizace slov byla zároveň navržena tak, aby mohla probíhat automaticky na pozadí a uživatel ji nemusel řešit, pokud nechce. Z tohoto důvodu je normalizační část formuláře ve výchozím stavu skryta. Lze ji zobrazit stisknutím tlačítka. Normalizované varianty v normalizační části formuláře jsou navrhovány v reálném čase, ihned po vyplnění slova uživatelem. Zároveň jsou seřazeny podle počtu použití v ostatních záznamech a nejčastěji použitá varianta je automaticky vybrána. Pro zadaná slova jsou tak automaticky vybrány nejvhodnější normalizované varianty.

Pro roztřídění slov z matričních záznamů do shluků podobných slov bylo nutné nastudovat proces porovnání dat (*Data-Matching*). Tento proces byl vylepšen o paralelní výpočet podobnosti jednotlivých dvojic slov. Díky tomuto rozšíření bylo zrychleno shlukování slov až 16krát. Vhodné parametry shlukování byly vybrány na základě automatického testování úspěšnosti shlukování, při kterém bylo pro každý typ slov otestováno celkem 640 kombinací parametrů shlukování. Výsledky tohoto testování jsou zobrazeny v podobě automaticky generovaných grafů. Úspěšnost shlukování s již nalezenými nejlepšími parametry shlukování byla následně testována zvlášť pro jednotlivé jazyky a pohlaví všech typů slov. Pokud dojde k větší obměně dat v databázi, nemusí již být aktuální parametry shlukování optimální. Díky automatickému testování je ale možné nalézt nové optimální parametry shlukování.

Vyhledávání v matričních záznamech je stěžejní funkcí webové aplikace DEMoS. Díky normalizaci slov objevujících se v matričních záznamech byla zvýšena čitelnost a efektivita vyhledávání v těchto záznamech. Aplikace DEMoS je dostupná na internetové adrese http://perun.fit.vutbr.cz/demos. Tato práce byla součástí studentské konference inovací, technologií a vědy v IT Excel@FIT2019

Literatura

[1] *Matematickabiologie.cz: Hierarchické divizivní shlukování.* [Online; navštíveno 12.1.2019].
URL http://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--vicerozmerne-metody-pro-analyzu-dat--shlukova-analyza--shlukova-hierarchicka-analyza--hierarchicke-shlukovani--hierarchicke-divizivni-shlukovani

[2] *Matematickabiologie.cz: Hierarchické shlukování.* [Online; navštíveno 12.1.2019].
URL
http://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--vicerozmerne-metody-pro-analyzu-dat--shlukova-analyza--shlukova-hierarchicka-analyza--hierarchicke-shlukovani

[3] *Wikipedia: Matrika.* Prosinec 2018, [Online; navštíveno 13.12.2018].
URL https://cs.wikipedia.org/wiki/Matrika

[4] ANGELES, M. D. P.; ESPINO-GAMEZ, A.: Comparison of methods Hamming Distance, Jaro, and Monge-Elkan. Květen 2015, str. 64.

[5] CHAHOR, D.: *Clustering a list of words.* Únor 2017, [Online; navštíveno 5.11.2018].
URL
https://rapply.weebly.com/word-clustering/clustering-a-list-of-words

[6] CHRISTEN, P.: *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer Publishing Company, Incorporated, 2012, ISBN 978-3-642-31163-5.

[7] DRESSLER, K.; NGOMO, A.-C. N.: Time-efficient Execution of Bounded Jaro-Winkler Distances. In *Proceedings of the 9th International Conference on Ontology Matching - Volume 1317*, OM'14, Aachen, Germany, Germany: CEUR-WS.org, 2014, s. 37–48.
URL http://dl.acm.org/citation.cfm?id=2878740.2878744

[8] JAIN, A. K.: Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, ročník 31, č. 8, 2010: s. 651 – 666, ISSN 0167-8655, doi:https://doi.org/10.1016/j.patrec.2009.09.011, award winning papers from the 19th International Conference on Pattern Recognition (ICPR).
URL http://www.sciencedirect.com/science/article/pii/S0167865509002323

[9] JARKOVSKÝ, J.; LITTNEROVÁ, S.: *Vícerozměrné statické metody.* [Online; navštíveno 13.12.2018].

URL http://www.iba.muni.cz/esf/res/file/bimat-prednasky/vicerozmerne-statisticke-metody/VSM-05.pdf

[10] KAPAVÍK, R.: *Získávání znalostí z dat - shlukovací algoritmy*. Brno, 2008. bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D., [Online; navštíveno 12.1.2019].
URL https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=116197

[11] KELBEL, J.; ŠILHÁN, D.: *Shluková analýza*. Květen 2002, [Online; navštíveno 12.1.2019].
URL http://cmp.felk.cvut.cz/cmp/courses/recognition/zapis_prednasky/zapis_02/13/shlukovani.pdf

[12] KUČERA, J.: *Nehierarchické metody shlukování*. [Online; navštíveno 14.1.2019].
URL https://is.muni.cz/th/w8lgz/5739129/web/web/nehiermet.html

[13] LAIT, A. J.; RANDELL, B.: An assessment of name matching algorithms. Duben 2019.

[14] PECHÁČEK, J.: *Jak vypadají matriční záznamy*. Červenec 2018, [Online; navštíveno 16.03.2019].
URL https://www.odkudjsme.cz/blog/jak-vypadaji-matricni-zaznamy/

[15] PHILIPS, L.: The Double Metaphone Search Algorithm. *C/C++ Users J.*, ročník 18, č. 6, Červen 2000: s. 38–43, ISSN 1075-2838.
URL http://dl.acm.org/citation.cfm?id=349124.349132

[16] RATCLIFF, J. W.; METZENER, D. E.: Pattern Matching: The Gestalt Approach. Leden 2009.
URL http://collaboration.cmc.ec.gc.ca/science/rpn/biblio/ddj/Website/articles/DDJ/1988/8807/8807c/8807c.htm

[17] SETIADI, I.: Damerau-Levenshtein Algorithm and Bayes Theorem for Spell Checker Optimization. Prosinec 2013, doi:10.13140/2.1.2706.4008.

[18] ZBOŘIL, F. V.; ZBOŘIL, F.: *IZU - Strojové učení*. [Online; navštíveno 13.12.2018].
URL https://www.fit.vutbr.cz/study/courses/IZU/private/1819izu_8.pdf

[19] ZOBEL, J.; DART, P.: Phonetic String Matching: Lessons from Information Retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, New York, NY, USA: ACM, 1996, ISBN 0-89791-792-8, s. 166–172, doi:10.1145/243199.243258.
URL http://doi.acm.org/10.1145/243199.243258

Příloha A

Obsah přiloženého paměťového média

`/src/www` – Zdrojové soubory normalizační části webové aplikace.

`/src/clustering` – Zdrojové soubory programu pro roztřídění slov do shluků.

`/data/tests.sql` – Databáze použitá pro testování úspěšnosti shlukování. Obsahuje referenční roztřídění slov do shluků.

`/data/tests_queries.sql` – Soubor s SQL dotazy pro výpis problémových shluků atd.

`/data/tests` – Složka obsahující CSV soubory a grafy s výsledky testování.

`/doc/src` – Zdrojové soubory pro vytvoření technické zprávy.

`/doc/xhribe02-normalizace.pdf` – Technická zpráva ve formátu PDF.

Příloha B

Grafy testování úspěšnosti shlukování

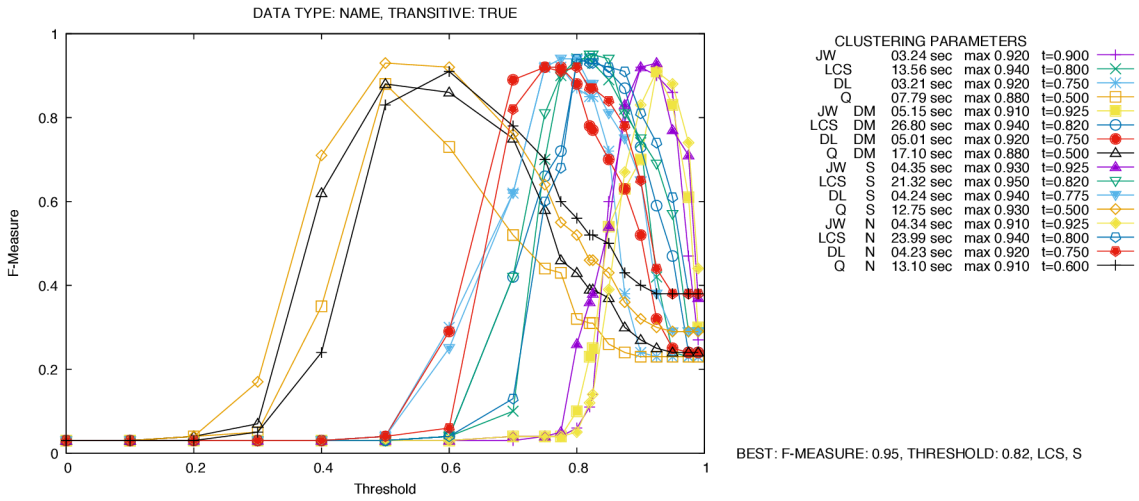
Pro každý typ slov bylo provedeno testování úspěšnosti shlukování s různými kombinacemi parametrů shlukování (práh podobnosti, fonetické kódování, funkce pro vypočet podobnosti řetězců, tranzitivní uzavření záznamů). Pro každý typ slov byly vygenerovány dva grafy zobrazující výsledky testování (s použitím a bez použití tranzitivního uzavření záznamů). Grafy jsou generovány automaticky při testování. Význam zkratek v těchto grafech:

- JW (Jaro-Winkler),
- LCS (Longest Common Substring),
- DL (Damerau-Levenshtein),
- Q (Q-Gram),
- DM (Double-Metaphone),
- S (Soundex),
- N (NYSIIS).

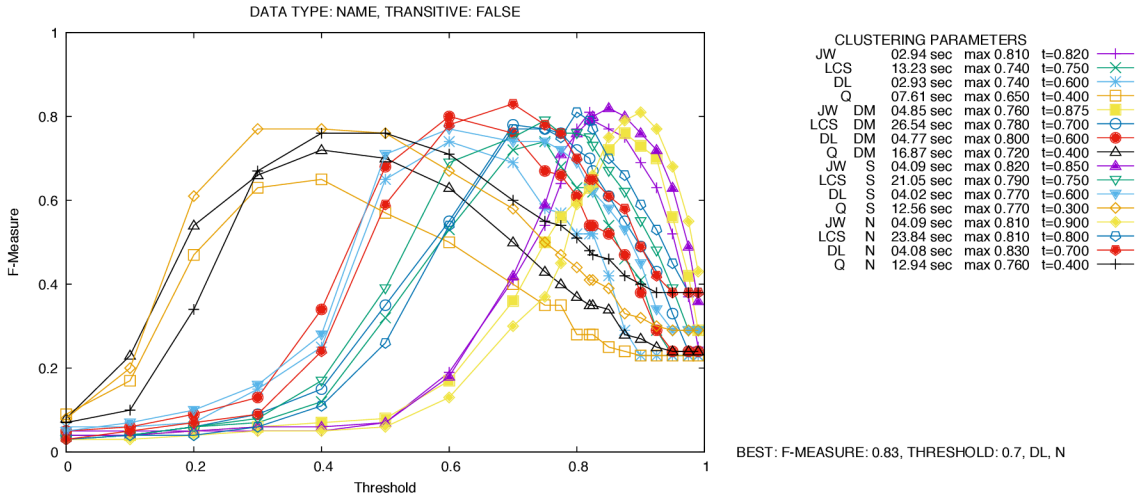
V pravé části každého grafu je umístěna legenda. U každé kombinace parametrů fonetického kódování a funkce podobnosti řetězců je navíc vypsána průměrná doba jednoho shlukování a maximální naměřená úspěšnost shlukování společně s prahem, na kterém byla tato úspěšnost naměřena.

Jména

Celkem použito 1573 slov, která jsou psána starou češtinou (398), latinsky (618), německy (296) a současnou češtinou (261).



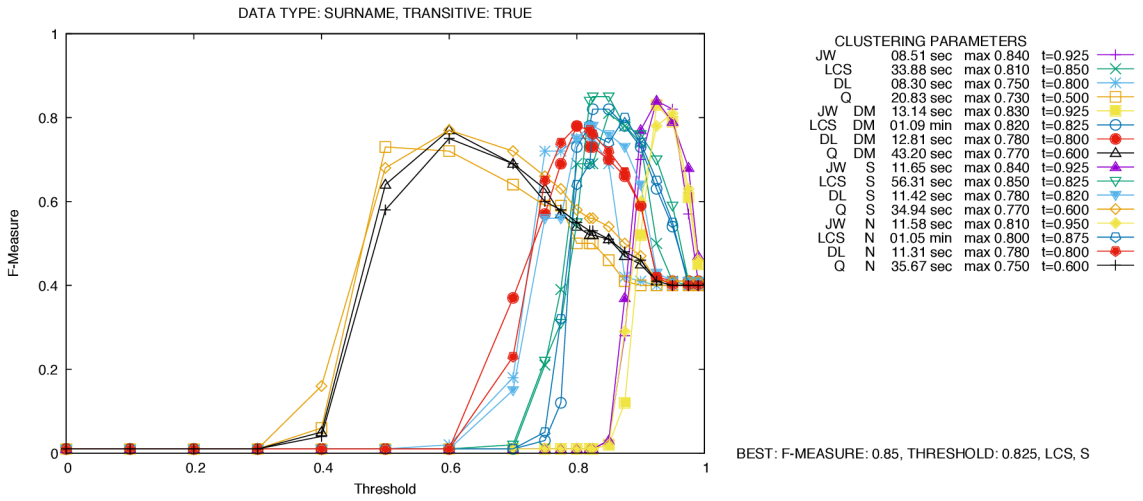
Obrázek B.1: Graf testování úspěšnosti shlukování jmen s použitím tranzitivního uzavření záznamů. Nejlepší úspěšnost shlukování 0.95 byla naměřena pro kombinaci parametrů shlukování: Longest Common Substring, Soundex, práh podobnosti 0.82.



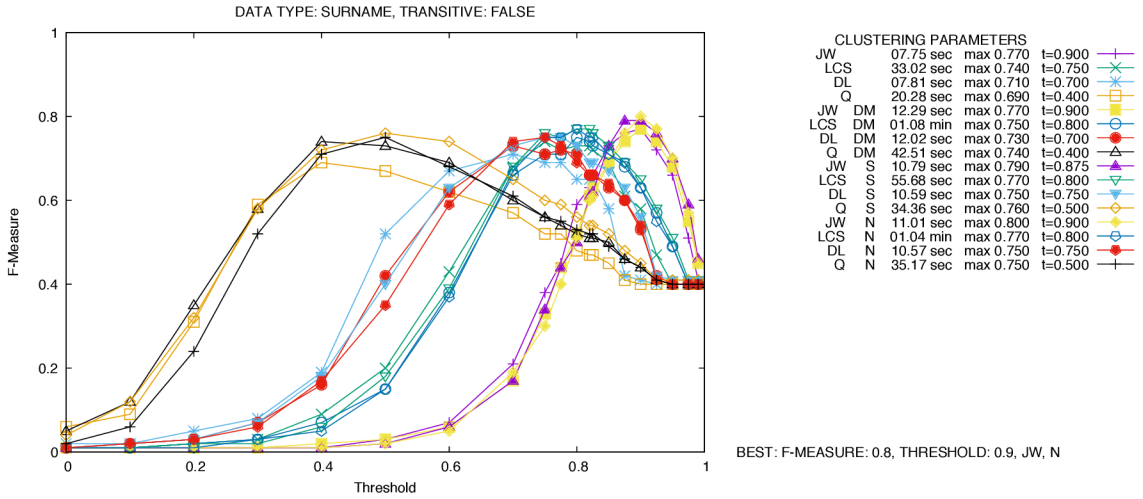
Obrázek B.2: Graf testování úspěšnosti shlukování jmen bez použití tranzitivního uzavření záznamů. Nejlepší úspěšnost shlukování 0.83 byla naměřena pro kombinaci parametrů shlukování: Damerau-Levenshtein, NYSIIS, práh podobnosti 0.7.

Příjmení

Celkem použito 2470 slov, která jsou psána starou češtinou (795), latinsky (816), německy (419) a současnou češtinou (440).



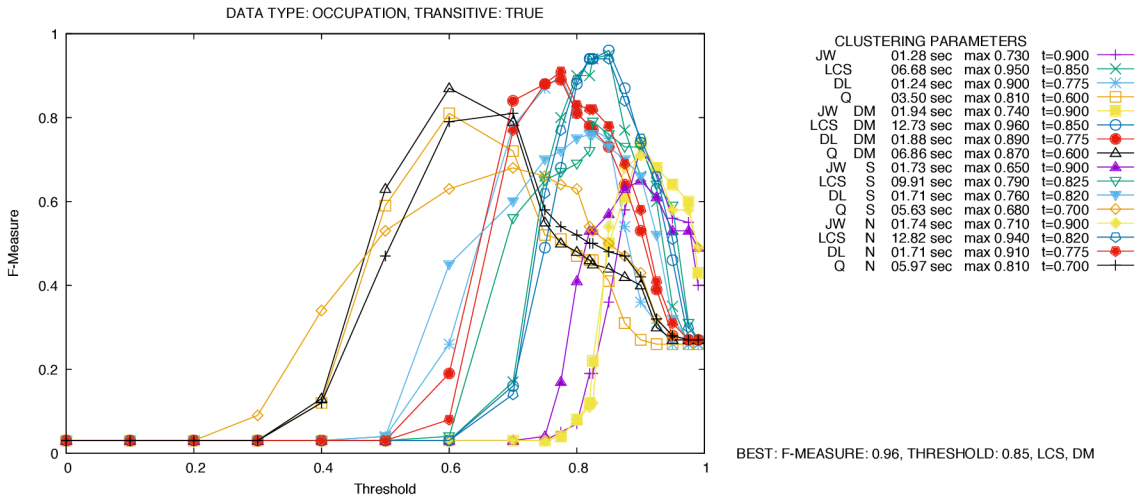
Obrázek B.3: Graf testování úspěšnosti shlukování příjmení s použitím tranzitivního uzavření záznamů. Nejlepší úspěšnost shlukování 0.85 byla naměřena pro kombinaci parametrů shlukování: Longest Common Substring, Soundex, práh podobnosti 0.825.



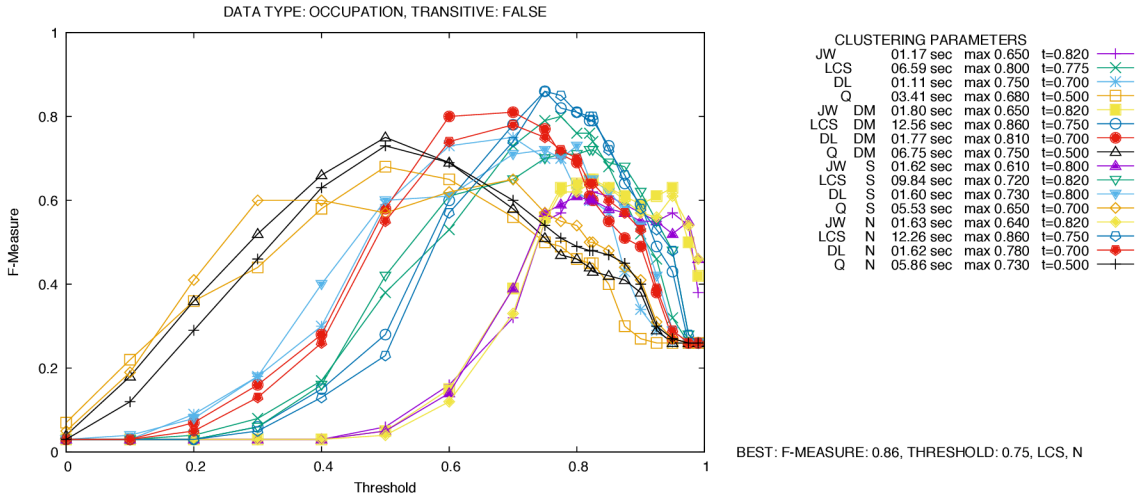
Obrázek B.4: Graf testování úspěšnosti shlukování příjmení bez použití tranzitivního uzavření záznamů. Nejlepší úspěšnost shlukování 0.8 byla naměřena pro kombinaci parametrů shlukování: Jaro-Winkler, NYSIIS, práh podobnosti 0.9.

Povolání

Celkem použito 654 slov, která jsou psána starou češtinou (31), latinsky (90), německy (334) a současnou češtinou (199).



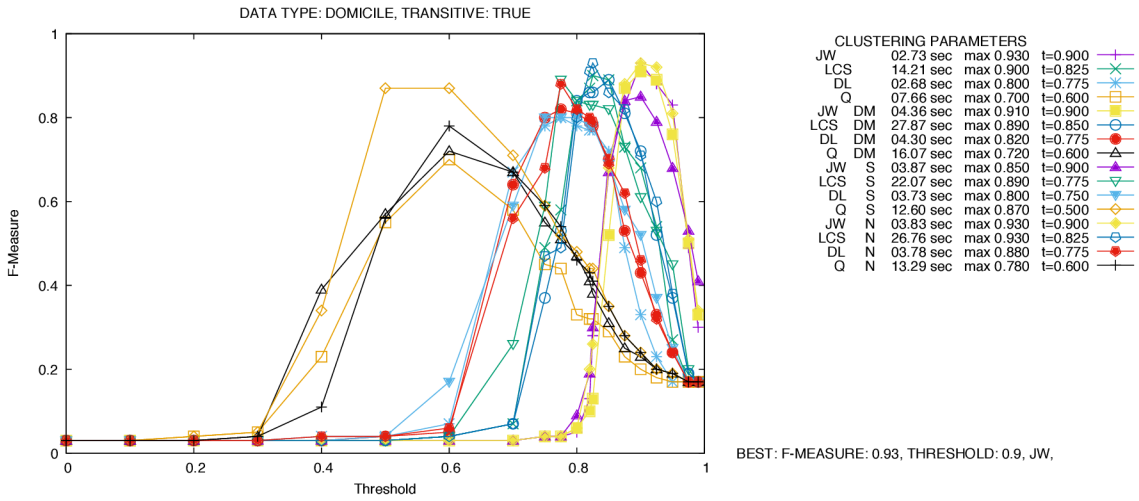
Obrázek B.5: Graf testování úspěšnosti shlukování povolání s použitím tranzitivního uzavření záznamů. Nejlepší úspěšnost shlukování 0.96 byla naměřena pro kombinaci parametrů shlukování: Longest Common Substring, Double-Metaphone, práh podobnosti 0.85.



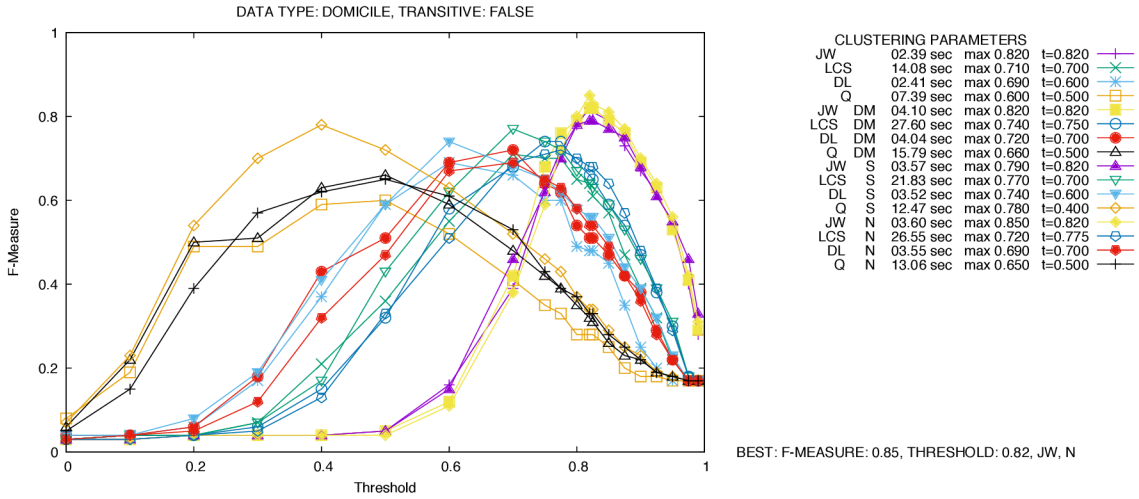
Obrázek B.6: Graf testování úspěšnosti shlukování povolání bez použití tranzitivního uzavření záznamů. Nejlepší úspěšnost shlukování 0.86 byla naměřena pro kombinaci parametrů shlukování: Longest Common Substring, NYSIIS, práh podobnosti 0.75.

Obce

Celkem použito 995 slov, která jsou psána starou češtinou (367), latinsky (305), německy (214) a současnou češtinou (109).



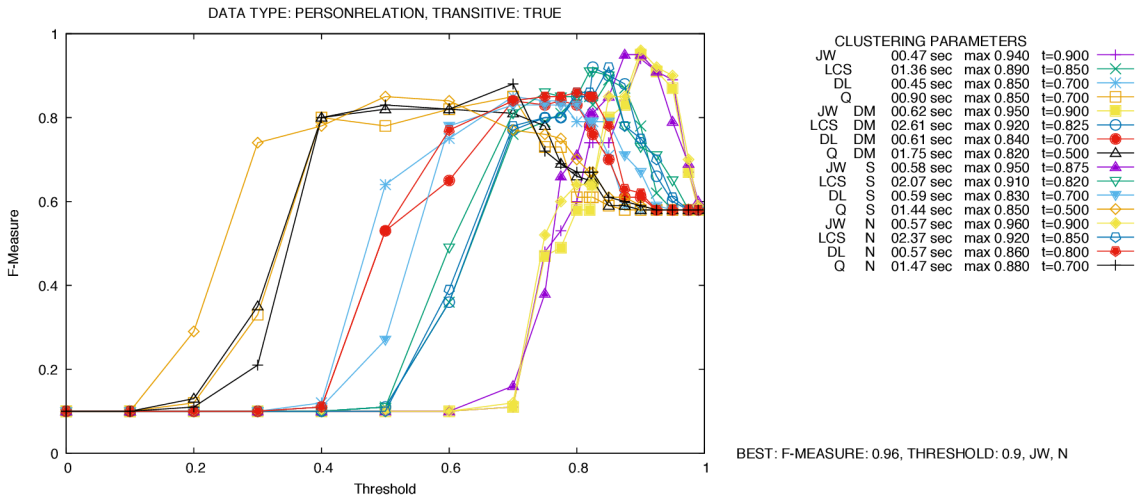
Obrázek B.7: Graf testování úspěšnosti shlukování obcí s použitím tranzitivního uzavření záznamů. Nejlepší úspěšnost shlukování 0.93 byla naměřena pro kombinaci parametrů shlukování: Jaro-Winkler, bez fonetického kódování, práh podobnosti 0.9.



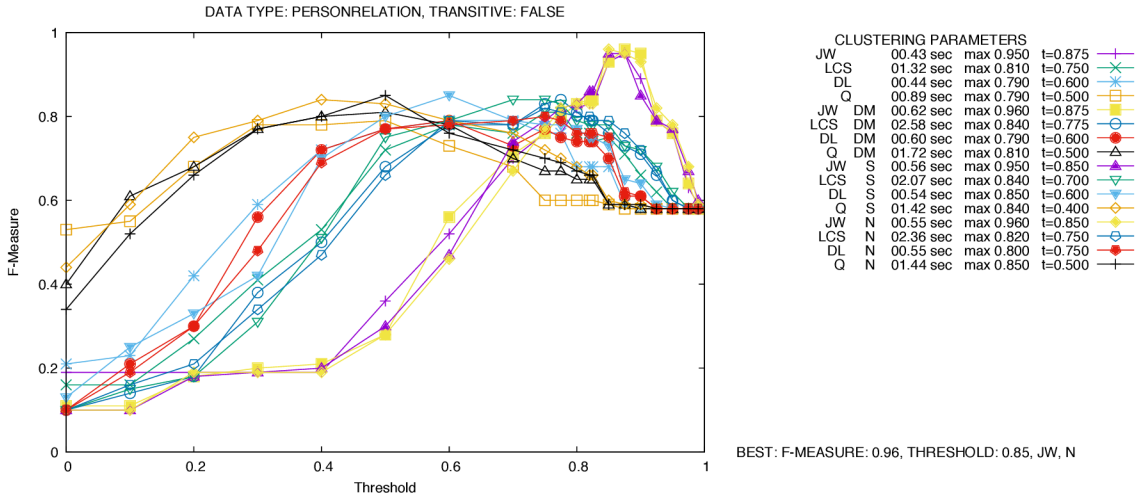
Obrázek B.8: Graf testování úspěšnosti shlukování obcí bez použití tranzitivního uzavření záznamů. Nejlepší úspěšnost shlukování 0.85 byla naměřena pro kombinaci parametrů shlukování: Jaro-Winkler, NYSIIS, práh podobnosti 0.82.

Vztahy mezi lidmi

Celkem použito 543 slov, která jsou psána starou češtinou (25), latinsky (213), německy (207) a současnou češtinou (98).



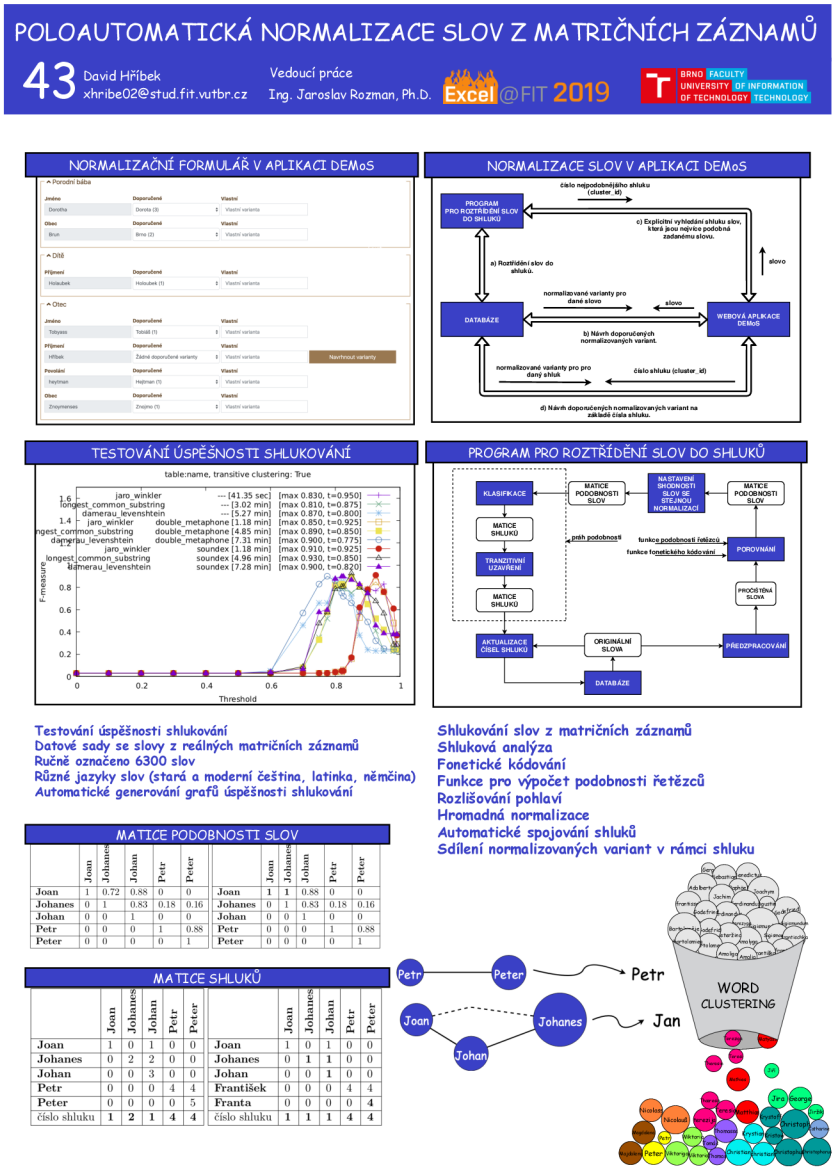
Obrázek B.9: Graf testování úspěšnosti shlukování vztahů mezi lidmi s použitím tranzitivního uzavření záznamů. Nejlepší úspěšnost shlukování 0.96 byla naměřena pro kombinaci parametrů shlukování: Jaro-Winkler, NYSIIS, práh podobnosti 0.9.



Obrázek B.10: Graf testování úspěšnosti shlukování vztahů mezi lidmi bez použití tranzitivního uzavření záznamů. Nejlepší úspěšnost shlukování 0.96 byla naměřena pro kombinaci parametrů shlukování: Jaro-Winkler, NYSIIS, práh podobnosti 0.85.

Příloha C

Plakát



Obrázek C.1: Plakát. Práce byla součástí konference Excel@FIT2019.