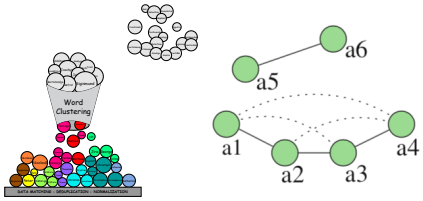
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2019
http://excel.fit.vutbr.cz

Poloautomatická normalizace slov z matričních záznamů

David Hříbek*



Abstrakt

V této práci je řešeno shlukování slov z matričních záznamů tak, aby v každém shluku byla slova, která jsou si navzájem velmi podobná.

V řešení bylo použito procesu deduplikace záznamů, který byl pozměněn tak, aby odpovídal požadavkům práce. Díky roztřídění záznamů do shluků bylo následně možné tyto shluky normalizovat, tedy přidělit celým shlukům nebo jednotlivým slovům jejich normalizované názvy.

V rámci práce byla přidána možnost normalizace slov do již existující webové aplikace, pro správu matričních záznamů, DEMoS.

Díky normalizaci slov jsou matriční záznamy lépe čitelné a je mezi nimi možné snadněji vyhledávat.

Klíčová slova: Data-Matching — Deduplication — Parish Records – Normalization

Přiložené materiály: DEMoS

*xhribe02@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Úvod

V minulosti byly matriční záznamy zapisovány pouze do matričních knih (matrik). V dnešní době se často matriční záznamy digitalizují a ukládají do elektronických databází, díky čemuž je možné mezi těmito záznamy snadněji vyhledávat a třídit je.

Matriční záznamy se zapisují do knih, které mají pevně danou strukturu (tabulka s pojmenovanými sloupci). Každý záznam mohl být ale zapsán jiným člověkem, což má za následek, že některé záznamy mohou být obsáhlejší, či méně čitelné, než jiné. Mohou také obsahovat chyby, překlepy nebo různé varianty stejného slova. Během staletí se taktéž mohla změnit podoba některých slov, která jsou do matrik zapisována. Například název města, který se kdysi zapisoval jako „Bozskowycze“ je dnes zapisován jako „Boskovice“. Totéž platí o křestních jménech, příjmeních, povoláních, atd. Podoba slov v matričních záznamech také záleží na době zápisu. Během let se na stejném místě mohly zapisovat matriční knihy v různých jazycích jako např. čeština 17. a 18. století, latina, němčina nebo moderní čeština. Tímto vznikají různé variace slov téhož významu, což ztěžuje možnost vyhledávání mezi záznamy.

V této práci je řešeno rozšíření již existující webové aplikace DEMoS, která se zabývá správou matričních záznamů, o možnost normalizace jednotivých slov v těchto záznamech. Normalizací slova je myšleno přiřazení danému slovu jeho dnešní podoby zápisu. Pro slova ’Josephus’ a ’Joseph’ by dnešní podoba zápisu byla například ’Josef’. Díky normalizaci je zvýšena efektivita vyhledávání a čitelnost matričních záznamů. Mezi typy slov, která jsou normalizována patří například jména, příjmení, povolání, obce a další.

Řešení tohoto problému se skládá z dvou částí – nalezení slov, která jsou si navzájem podobná (např.

jména 'Petr' a 'Peter' nebo 'František' a 'Francziska') a jejich roztřídění do shluků s následným přiřazením normalizovaného názvu jednotlivým slovům ve shlucích nebo celým shlukům. V první části je využit proces porovnávání dat (*Data-Matching*). Jelikož jsou hledány podobné záznamy v rámci jedné databáze, jedná se o speciální případ procesu porovnávání dat – deduplikace (*Deduplication*). Výsledkem tohoto procesu je, že každé slovo určené k normalizaci (jméno, příjmení, obec, povolání, atd.) je označeno číslem shluku, do kterého patří. Jednotlivým slovům je pak možné pomocí webové aplikace DEMoS přiřadit normalizovanou variantu a díky vytvořeným shlukům slov je možné tyto normalizované varianty sdílet v rámci daného shluku a nabízet tak doporučené normalizované varianty.

2. Použité technologie

Webové aplikace DEMoS je napsána v jazyce PHP (Nette Framework). Algoritmus pro shlukování slov je napsán v jazyce Python. Normalizace slov během vytváření nebo úpravy záznamů je řešena asynchronně, je zde využita technologie AJAX a serializační formát JSON. Dále je použit programovací jazyk JavaScript a knihovna JQuery.

3. Proces shlukování záznamů

Tato část bude věnována využití shlukové analýzy při procesu shlukování záznamů. Bude zde čerpáno z knihy [2] od autora Petera Christena, která vychází z více než 300 článků na toto téma a shrnuje tak celou problematiku. Pokud není řečeno jinak, je v této části čerpáno z této knihy.

3.1 Předzpracování záznamů

Předzpracování záznamů [2] (*Data Pre-Processing*) je důležitou součástí shlukové analýzy. Data, která vstupují do procesu shlukování se mohou lišit ve formátu a struktuře, a také mohou obshovat chyby. Hodnoty záznamů nemusí být vždy zadány v prvním pádu. Navíc mohou obsahovat nadbytečná slova, která nepředstavují informace užitečné pro shlukování. Mezi taková slova patří například předložky. To, jakým způsobem byla data pořízena také ovlivňuje jejich kvalitu. Data mohla být přepisována z hůře čitelných zdrojů nebo naskenována a získána pomocí optického rozpoznávání znaků. Chyby při zaznamenávání dat také nastávají pokud byla 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**. Dekódování znaků v kódování UNICODE do kódování ASCII. 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**. Informace o velikostí 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) je nutné odstranit. Mezi takové znaky patří například všechny nealfanumerické znaky, kromě mezer. Mezery oddělují jednotlivá slova, 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**. Podobnost některých znaků je větší než podobnost jiných znaků. Znaky jako *y* a *i* nebo *w* a *v* jsou si velice podobné, a proto je možné vybrat jeden z těchto znaků a nahradit jím všechny výskyty druhého znaku. V závislosti na typu dat je možné sestavit více pravidel pro nahrazení znaků nebo sekvencí znaků, a uložit je do vyhledávací tabulky (*lookup table*)
- **Číslice**. Pokud shlukujeme například názvy obcí nebo jména osob, číslice je také možno odstranit.
- **Nadbytečná slova**. Hodnoty záznamů mohou obsahovat slova, která nepřínášejí žádnou informaci, užitečnou pro shlukování – „stop slova" (*stop words*[1]) – mezi taková slova patří běžně užívaná slova v různých jazycích.
  Existují předpřipravené slovníky se „stop slovy" pro různé jazyky. V českém jazyce mezi stop slova patří například: *a, aby, byl, byla, pro, vsak, ktere*.

Proces čištění dat je závislý na typu dat. Je proto nutné zvolit způsob čištění dat tak, aby daná data nebyla znehodnocena. Důležité také je, aby při procesu čištění dat nebyla ztracena původní vstupní data.

[1] stop words - https://en.wikipedia.org/wiki/Stop_words

3.2 Indexace

Indexace [2] je důležitým krokem před samotným porovnáváním záznamů. Pročištěná a standardizovaná data jsou nyní připravena na porovnávání. Při shlukování dvou databází by tedy bylo nutné porovnat každý záznam z první databáze s každým záznem z druhé databáze. Pokud je shlukována jedna databáze, musí být každý záznam porovnán se všemi ostatními záznamy v dané databázi. Proces porovnávání má tedy kvadratickou časovou složitost. Při každém porovnání dvojice záznamů je navíc nutné porovnat všechny atributy obou záznamů. Porovnávání je nejnáročnější operací procesu shlukování záznamů.

Když je porovnáván jeden záznam se všemi ostatními záznamy, většina porovnání dopadne neúspěšně. 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.

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

3.3 Porovnání záznamů

Dalším krokem v procesu shlukování záznamů je samotné porovnávání [2] jednotlivých záznamů mezi sebou (*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 (1). Jedná se o funkci pro výpočet podobnosti.

$$s = sim(a_i, a_j) \tag{1}$$

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 si do jisté míry podobné.

To, co znamená, že jsou si 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:

$$sim_{exact}(a_1, a_2) = \begin{cases} 1.0 & pokud\ a_1 = a_2 \\ 0 & pokud\ a_1 \neq a_2 \end{cases} \tag{2}$$

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**. Nejznámější implementací editační vzdálenosti je Levenshteinova editační vzdálenost. 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[2]. Časová složitost je kvadratická. Metoda nezohledňuje ve které části jsou řetězce odlišné – odlišnosti 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.

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

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

  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 **Demerau-Levenshtein**, která přidává čtvrtou operaci — výměnu dvou sousedních znaků. Výměna dvou sousedních znaků je častou chybou při získávání dat.

[2] Dynamické programování - https://cs.wikipedia.org/wiki/Dynamick%C3%A9_programov%C3%A1n%C3%AD

		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**	2	3
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	**2**	3
5	**a**	5	4	3	2	3	**4**

Tabulka 1. Ukázka výpočtu Levensthteinovy editační vzdálenosti dvou jmen. V první tabulce je vypočtena vzdálenost mezi jmény 'pavla' a 'pavel' rovna 2. V druhé tabulce je vypočte vzdálenost mezi jmény 'radim' a 'radka' rovna 4. Tučné číslice vyznačují cestu ke konečnému výsledku. Tyto tabulky byly převzaty a upraveny z knihy [2].

- **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 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 a c_2 je počet Q-gram řetězců vzniklých z řetězce s_1 resp. s_2:

$$sim_{overlap}(s_1, s_2) = \frac{c}{min(c_1, c_2)} \quad (3)$$

$$sim_{jaccard}(s_1, s_2) = \frac{c}{(c_1 + c_2) - c} \quad (4)$$

$$sim_{dice}(s_1, s_2) = \frac{2 \times c}{c_1 + c_2} \quad (5)$$

  Možným rozšířením metody Q-gram je zohledněné 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. Následující tabulka zobrazuje rozdělení vstupního řetězce na jednotlivé Q-gram řetězce:
- **Jaro, Jaro-Winkler**. 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 (6). 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.

$$sim_{jaro}(s_1, s_2) = \frac{1}{3}\left(\frac{c}{|s_1|} + \frac{c}{|s_2|} + \frac{c-t}{c}\right) \quad (6)$$

  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.
- **Longest Common Substring**. 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 jichž dříve zmíněných koeficientů:

$$sim_{lcs\_overlap}(s_1, s_2) = \frac{l_c}{min(c_1, c_2)}, \quad (7)$$

$$sim_{lcs\_jaccard}(s_1, s_2) = \frac{l_c}{(|s_1| + |s_2|) - |l_c|}, \quad (8)$$

$$sim_{lcs\_dice}(s_1, s_2) = \frac{2 \times l_c}{|s_1| + |s_2|}. \quad (9)$$

  Časová složitost výpočtu sumy (l_c) všech délek nejdelších společných podřetězců je kvadratická. Nevýhodou tohoto algoritmu je, že vypočtená podobnost vstupních řetězců nemusí být vždy symetrická – pokud jsou vstupní řetězce prohozeny, výsledná míra podobnosti se může lišit.

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

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 $sim_{lcs} = 0.5$. Pokud jsou vstupní řetězce vyměněny, tedy $s_1 =$ '$jnja$' a $s_2 =$ '$janj$', jako první nejdelší podřetězec je nalezen řetězec 'nj', po jehož odstranění vzniknout 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 ($l_{min} = 2$), proto $l_c = 4$ a $sim_{lcs} = 1$.
Pro vyřešení tohoto problému je nutné vypočítat podobnost vstupních řetězců v obou pořadích a výsledek zprůměrovat.

3.3.1 Fonetické kódování atributů

Cílem funkcí pro fonetické kódování řetězců [2] je převést vstupní řetězec na kód, který odpovídá tomu, jak by daný řetězec byl vysloven. Tyto funkce jsou závislé na konkrétním jazyce, existují proto varianty těchto funkcí pro konkrétní jazyky. Fonetické kódování je možné použít také při výpočtu podobnosti mezi řetězci, kdy jsou nejdříve jednotlivé řetězce foneticky zakódovány a poté jsou tyto kódy porovnány. Fonetické kódování se používá u jmen, příjmení a názvů. Nejznámější algoritmus pro fonetické kódování se nazývá Soundex [3]. Od tohoto algoritmu jsou poté odvozeny další varianty jako např. Phonex, Phonix nebo Double-Metaphone.

Soundex Výsledný kód vznikne převodem vstupního řetězece (např. jméno) na kód ve tvaru počátečního písmene řetězce, následovaného několikamístným číselným kódem (většinou je použit 3 místný 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.

Double-Metaphone Hlavní nevýhodou předešlého algoritmu je jeho závislost na konkrétním jazyce. V reálných databázích se nenacházejí jen anglické názvy (např. jména), 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 předzpracování vstupního řetězce, kde je aplikováno velké množství pravidel pro modifikaci řetězce. Rozdílem tohoto algoritmu oproti předešlému je, že výsledný kód neobsahuje číslice, pouze znaky. Pro některá jména jsou navíc navráceny dva výsledné fonetické kódy, místo jednoho. Tyto dva kódy vzniknou odlišným pořadím aplikování fonetických transformačních pravidel.

3.4 Klasifikace

Záznamy, které byly vyfiltrovány v kroku Indexace a detailně porovnány v předešlém kroku Porovnání záznamů je nyní možné klasifikovat [2]. Záznamy lze klasifikovat do 3 tříd: shodné (*Matches*), neshodné (*Non-matches*) a potenciálně shodné (*Potential matches*).

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 následující tabulce.

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

Tento vektor je dále použit při klasifikaci. Základním způsobem klasifikace záznamů do tříd je **Prahování**. Využívají se zde dva práhy t_{up} a t_{low}, a suma podobnosti záznamů (*SimSum*). Klasifikace do tříd pak probíhá podle těchto pravidel:

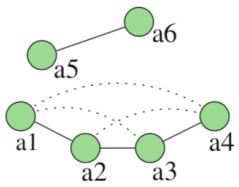
$$\begin{aligned} SimSum[r_i, r_j] \geq t_{up} &\implies [r_i, r_j] \rightarrow Match, \\ t_{low} \leq SimSum[r_i, r_j] \leq t_{up} &\implies [r_i, r_j] \rightarrow Potential\ Match, \\ SimSum[r_i, r_j] \leq t_{low} &\implies [r_i, r_j] \rightarrow Non-Match, \end{aligned}$$

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 se o klasifikaci do dvou tříd. Klasifikace do dvou tříd pak probíhá podle těchto pravidel:

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

3.5 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_2, a_1) a (a_2, a_3) byly klasifikovány jako shodné (Matches) a dvojice záznamů (a_1, a_3) byla klasifikována jako neshodná (Non-Matches), je nutné řešit tranzitivní uzavření těchto záznamů. Příklad takové situace je zobrazen na obrázku 1.



Obrázek 1. Ukázka klasifikace záznamů do dvou tříd (Matches, Non-Matches). 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 převzat z knihy [2].

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).

4. Testování kvality shlukování

Posledním krokem procesu shlukování záznamů je měření kvality shlukování [2]. 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 dvojice záznamů pak lze zařadit do jedné z následujících kategorií:

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

Cílem shlukování záznamů je, aby počty dvojic v kategoriích TP a TN byly co nejvyšší, a zároveň počty dvojic záznamů v kategoriích FP a FN byly co nejnižší. Ná 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 záznamu v této kategorii.

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

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

$$precision = \frac{TP}{TP+FP}. \quad (10)$$

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

$$recall = \frac{TP}{TP+FN}. \quad (11)$$

- **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. Vztah pro výpočet tého metriky zobrazuje následující rovnice:

$$f-measure = 2 \times \frac{precision \times recall}{precision + recall}. \quad (12)$$

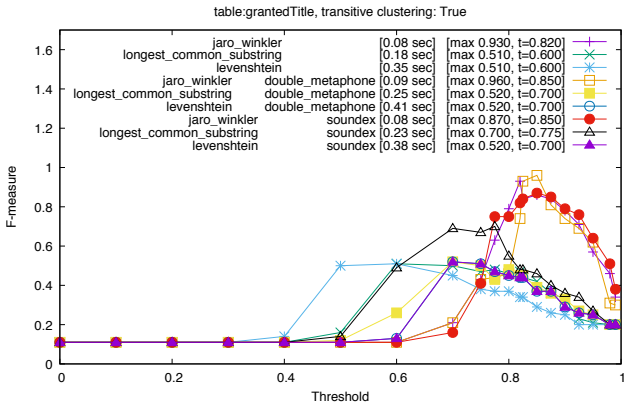
Tato metrika je často využívána pro měření kvality shlukování.

4.1 Výběr algoritmů na základě testování

Parametry a výsledky testování shlukování jsou zobrazeny v tabulce 4. Pro testování byly použity data z reálných matričních záznamů, které jsou psány češtinou 17. a 18. století, latinsky, německy a moderní češtinou. Úspěšnost shlukování je posouzena vzhledem k referenčnímu roztřídění slov do shluků (*ground-truth*) pomocí metriky *F-measure*, která je popsána níže. Finální použité funkce podobnosti řetězců, funkce pro fonetické kódování atributů a práh podobnosti byly vybrány na základě testování různých kombinací těchto parametrů pro každý typy slov. Ukázka výsledků testování pro tituly farářů je zobrazena v grafu na obrázku 2.

Typ slova	Fon. Kód.	Funkce podobnosti řetězců	Práh	Počet slov	Úspěšnost shlukování (F-meassure)
Jméno		JW	0.90	1573	∼ 0.87
Příjmení	S	LCS	0.85	2470	∼ 0.84
Povolání	DM	LCS	0.85	654	∼ 0.96
Obec		JW	0.90	995	∼ 0.92
Titul faráře	DM	JW	0.85	85	∼ 0.96

Tabulka 4. Parametry nejlepších naměřených výsledků tesování pro jednotivé typy slov. Význam zkratek: S (Soundex), DM (Double-Metaphone), JW (Jaro-Winkler), LCS (Longest Common Substring).



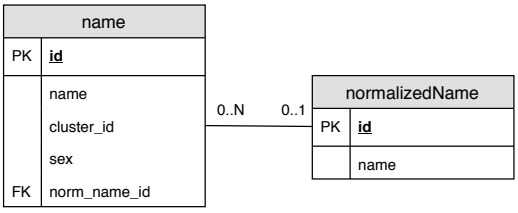
Obrázek 2. Graf závislosti kvality shlukování na práhu podobnosti pro slova typu titul faráře. Pro každou kombinaci zvolené funkce podobnosti řetězců a fonetického kódování je vykreslena samostatná křivka. Při kombinaci fonetického kódování a funkce podobnosti řetězců je výsledná podobnost vypočtena poměrem 65:35. Z toho grafu lze zjistit, že nejlepší kombinací parametrů shlukování pro slova tituly faráře je kombinace Jaro-Winkler, Double Metaphone a práh podobnosti 0.85.

5. Návrh řešení normalizace

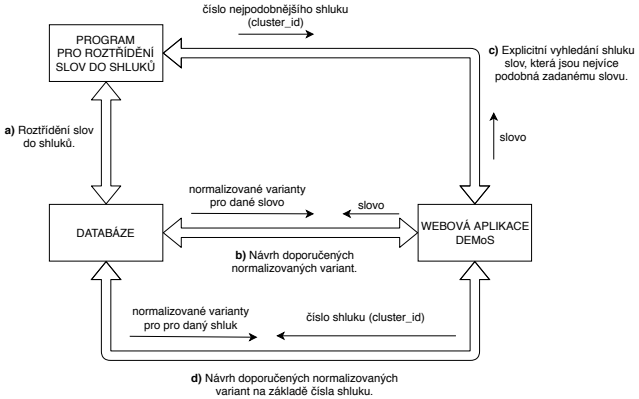
Shluky, vzniklé při procesu shlukování, jsou využity pro normalizaci slov z matričních záznamů. Pro každý typ slov, které je potřeba normalizovat, je vytvořena databázová tabulka s původními slovy a tabulka s normalizovanymi variantami pro daný typ slov. Mezi těmito dvěmi tabulkami je vazba 1:N (jedno normalizované slovo může být přiřazeno k více originálním slovům). Schéma databáze pro normalizaci jmen je zobrazeno na obrázku 3.

5.1 Obecný princip normalizace slov

Princip navržené normalizace slov z matričních záznamů je zobrazen na obrázku 4. Navržené řešení se skládá ze čtyř částí, které jsou v obrázku označeny jako část **a**, **b**, **c** a **d**. Následující text popisuje jednotlivé části navrženého řešení.



Obrázek 3. Schéma databáze pro uložení jmen z matričních záznamů a jejich normalizovaných variant. Toto schéma je aplikováno i pro ostatní typy dat (příjmení, povolání, obce, atd.).



Obrázek 4. Schéma obecného principu normalizace slov z matričních záznamů.

Originální slova z matričních záznamů budou nejdříve roztříděna do shluků (část **a** na obrázku 4), podle své podobnosti s ostatními slovy. K tomuto bude využita shluková analýza a proces shlukování záznamů, popsaný v kapitole č. 3. Výsledkem roztřídění záznamů do shluků bude u každé entity, z entitní množiny s originálními slovy, vyplněné číslo shluku v atributu `cluster_id`. Entity se stejným číslem shluku budou poté chápány jako shodné (představující stejný objekt reálného světa — různé varianty stejného jména, povolání, příjmení, atd.). Proces roztřídění slov do shluků bude spouštěn automaticky v určitém časovém intervale (např. každý den v nočních hodinách) pomocí softwarového démona CRON[3].

Při vytváření nebo editaci záznamů, pomocí webové aplikace DEMoS, budou uživateli po zadání konkrétního slova nabídnuty (v obdobě formulářového výběrového prvku) doporučené normalizované varianty pro daný typ a pohlaví slova. Tyto varianty budou získány porovnáním zadaného slova s již uloženými originálními slovy v databázi. Pokud zadané slovo již existuje v databázi (část **b** na obrázku 4), budou prioritně doporučeny normalizované varianty, které již byly použity u tohoto slova, následně pak normalizované varianty slov, která se nacházejí ve shluku se stejným číslem jako toto slovo. Pokud zadané slovo ještě neexistuje v databázi, uživatel bude mít možnost explicitně,

[3]CRON - https://cs.wikipedia.org/wiki/Cron

pomocí tlačítka, požádat o vyhledání shluku slov (část **c** na obrázku 4), které jsou nejvíce podobné s tímto slovem. Z tohoto shluku budou poté nabídnuty normalizované varianty slov (část **d** na obrázku 4). Uživatel bude mít vždy možnost zadat vlastní normalizovanou variantu daného slova ručně. Výsledkem normalizace slova je vyplnění atributu `norm_name_id` dané entity.

V případě, že dvěma slovům v odlišných shlucích (odlišná hodnota atributu `cluster_id`) bude 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 normalizované varianty.

5.2 Čas normalizace

Slova z matričních záznamů je možné normalizovat již při vytváření nebo úpravě záznamů (ukázka zobrazena na obrázku 7). Pro hromadnou normalizaci slov je vytvořena samostatná stránka, kde lze slova filtrovat podle typu (jméno, příjmení, atd.), obsahu, autra, pohlaví (pouze jména a příjmení) a stavu normalizace (normalizováno/nenormalizováno). Ukázka této stránky je zobrazena na obrázu 9.

6. Implementace

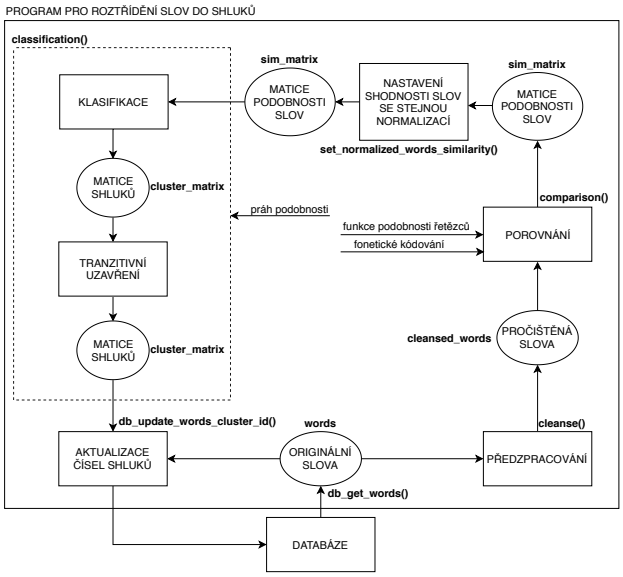
V této kapitole bude popsána implementace normalizace slov z matričních záznamů podle návrhu, který byl popsán v předešlé kapitole. Implementace programu pro roztřídění slov do shluků je částečně inspirována postupem [1].

Roztřídění slov do shluků

V této sekci je popsána část **a)** návrhu řešení, který je zobrazen na obrázku 4. V rámci této části je implementován *Proces shlukování záznamů*, který byl popsán v kapitole č. 3. Na obrázku 5 je zobrazeno schéma programu pro roztřídění slov do shluků.

Pro roztřídění slov do shluků byly využity pouze kroky *Předzpracování záznamů*, *Porovnání záznamů* a *Klasifikace*. 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*.

Proces roztřídění slov do shluků, který je zobrazen na obrázku 5 a byl implementován v rámci třídy `Words-Clustering`, je spuštěn pomocí metody `execute`. Nejdříve jsou z databáze načteny slova ke shlukování (na základě předaných parametrů při instanciaci), která jsou uložena v podobně seznamu do instanční proměnné `words`. Podle počtu načtených slov jsou pomocí



Obrázek 5. Schéma programu pro roztřídění slov do shluků.

knihovny `NumPy`[4], 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. Výsledek je uložen do příslušné buňky matice podobnosti `sim_matrix` (příklad vypočtené matice podobnosti je zobrazen v tabulce 5, vlevo). Jelikož je matice podobnosti symetrická, stačí vypočítat pouze část nad nebo pod diagonálou, včetně.

Po vypočtení podobnosti pro každou dvojici slov je matice podobnosti upravena tak, aby slova která již jsou normalizována a zároveň jejich normalizace jsou shodné (atribut `norm_name_id`), měla podobnost 1. Toto zajišťuje metoda `set_normalized_words_similarity` (příklad matice podobnosti před a po explicitním nastavením podobnosti slov je zobrazen v tabulce 5).

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. 3.4. Klasifikaci zajišťuje metoda `classification`. Po invokaci této metody je v matici podobnosti `sim_matrix` porovnána podobnost každé dvojice s prahem

[4] NumPy - http://www.numpy.org/

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 (příklad vypočtené matice shluků `cluster_matrix` je zobrazen v tabulce 6, vlevo). Posledním krokem klasifikace je tranzitivní uzavření, které je aktivováno na základě parametru `transitive_closure`, při instanciaci. Algoritmus pro tranzitivní uzavření jsem narvhl tak, že matice `cluster_matrix` je procházena po sloupcích, od konce. Pro každý sloupec je zjištěna nejmenší hodnota `col_min`, která se v něm nachází. Pokud se v daném sloupci nacházejí i buňky s hodnotou větší něž nalezená nejmenší hodnota, jsou hodnoty těchto buněk, včetně buněk na stejném řádku, 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 6.

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. Získání čísel shluků pro jednotlivá slova je zobrazeno v tabulce 6. U každého slova v seznamu `words` je pak v databázi aktualizováno číslo shluku `cluster_id`.

Výpočet nejpodobnějšího shluku pro zadané slovo

V této sekci je posána část **c)** návrhu řešení, který je zobrazen na obrázku 4. Cílem je nalézt pro zadané slovo číslo shluku, ve kterém se nachází nejvíce podobné slovo se zadaným slovem. Tento algoritmus implementuje metoda `get_best_cluste_id_of_unknown_word`. Vstupním parametrem je pouze zadané slovo `word`, pro které má být nalezeno číslo nejpodobnější shluk. Databázová tabulka, fonetické kódování, funkce podobnosti řetězců a práh podobnosti jsou zadány při instanciaci třídy `WordsClustering`. Tato metoda je volána z webové aplikace DEMoS asynchronně, pomocí technologie AJAX[5]. Výsledkem je buď to číslo nalezeného shluku, nebo v případě neúspěchu hodnota `-1`.

Po invokaci této metody jsou pomocí již zmíněné metody `db_get_words` načtena slova z databáze. Použitím parametru `clustered_only` je zajištěno načtení pouze slov, která již byla shlukována (mají vyplněný atribut `cluster_id`). Každé z těchto slov je poté porovnáno se zadaným slovem `word` s využitím parametrů porovnávání, které byly zvolených při instanciaci (pro každý typ slov byly zvoleny parametry porovnávání na základě testování uvedeném v kapitole č. 4). Pro slovo s největší podobností ke vstupnímu slovu `word` je pak pomocí jazyka SQL[6] nalezeno v databázi číslo shluku `cluster_id`, jehož hodnota je vrácena jako výsledek.

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. Část údajů v těchto sekcích je určena k normalizaci. Ukázka tří sekcí tohoto formuláře je zobrazena na obrázku 6.

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 dané poli normalizovat:

- 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 norm. variant použitých u slov v tomto shluku), které je zobrazeno pouze pokud zadané slovo nebylo nalezeno v databázi.

Tyto prvky jsou rovněž rozděleny do sekcí, podle původního formuláře. Jednotlivá pole v normalizační části jsou viditelná pouze pokud je původní pole formuláře vyplněno. Pokud není v sekci vyplněno žádné pole, tato sekce je v normalizační části skryta. Na obrázku 7 je zobrazena ukázka normalizační části formuláře pro již vyplněná pole původního formuláře na obrázku 6.

Normalizace probíhá asynchronně na pozadí. Vše je interaktivní – formulář se skrývá, odkrývá a doplňuje v reálném čase podle aktuálně zadaných slov v polích původního formuláře. Vyplněná normalizační část

[5] AJAX - https://cs.wikipedia.org/wiki/AJAX

[6] SQL - https://cs.wikipedia.org/wiki/SQL

formuláře je odeslána na server spolu s původním formulářem při uložení. 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 to 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 již zmíněného jazyka JavaScript a knihovny JQuery. Po načtení stránky s formulářem jsou nalezeny pomocí identifikátorů 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.

Aktualizace doporučených normalizovaných variant při zadání slova Jak už bylo řečeno, při zadání slov do polí původního formuláře dojde k aktivaci události `onChange` a tedy i k návrhu doporučených normalizovaných variant. Toto je provedeno odesláním zadaného slova a názvu daného formulářového pole (atribut `name`) pomocí již zmíněných API endpoint url na server, asynchronně použitím technologie AJAX. Na straně serveru je invokována metoda `getNormalizedVariantsOfWordAndWordsInSameCluster`, které je předáno zadané slovo a název daného formulářového pole. Na základě názvu pole jsou pak na serveru díky třídě `TableNames` zjištěny dodatečné informace o tomto poli a poté vyhledány doporučené normalizované varianty, které jsou pak odeslány zpět ve formátu JSON. Po přijetí odpovědi je pak aktualizováno výběrové pole s doporučenými variantami v normalizační části formuláře pomocí jazyka JavaScript. Jedná se o implementaci části **b** návrhu řešení, který je zobrazen na obrázku 4.

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 nebo pro toto slovo nejsou k dispozici žádné normalizované varianty. 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 4.

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 díky třídě `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 práh podobnosti, je zjištěno číslo shluku. Pomocí metody `getNormalizedVariantsOfWordsInSameClusterAsWord` jsou pak získány normalizované varianty slov v tomto shluku, které jsou následně odeslány zpět jako odpověď ve formátu JSON. Po přijetí odpovědi je pak rovněž aktualizováno výběrové pole s doporučenými normalizovanými variantami v normalizační části formuláře pomocí jazyka JavaScript.

Proces porovnání zadaného slova s ostatními slovy je implementován v rámci třídy `WordsClustering` v souboru `WordsClustering.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.

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 fomátu 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 vloženy varianty z kolekce `word` (mají větší prioritu). Následně pak do samostatné sekce varianty z kolekce `cluster`. Jako poslední je vložena možnost „Nestandardizovat“, pokud by uživatel nechtěl dané slovo normalizovat.

V závorce u každé doporučené varianty je zobrazen počet použití dané 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 primárně seřazeny podle pohlaví – nejdříve jsou nabídnuty varianty použité u slov stejného pohlaví jako je dané formulářové pole, poté další přípustné varianty (např. u formulářového

pole pro mužské jméno budou nejdíve nabídnuty mužské varianty a poté bezpohlavní varianty). Sekundární seřazení je podle počtu použití. První doporučená normalizovaná varianta tedy odpovídá nejvhodnější normalizované variantě pro dané slovo a je vybrána automaticky. Uživatel tedy nemusí normalizační část formuláře otevírat pokud nechce – budou automaticky vybrány nejvhodnější normalizované varianty pro zadaná slova.

Hromadná normalizace slov z matričních záznamů

Hromadnou normalizaci slov je myšlena možnost vypsání slov k normalizaci ze všech záznamů na jedné stránce, kde tyto slova bude možné normalizovat. Součástí této stránky je filtrační formulář, díky kterému je možné vypsat pouze požadovaná slova. Slova lze filtrovat podle typu (jména, příjmení, povolání, ...), pohlaví, autora normalizace, stavu (normalizováno/nenormalizováno) a obashu. Aby nebylo vypsáno najednou příliš mnoho slov, jsou slova roztříděna do stránek mezi kterými lze přepínat. Počet slov na jedné stránce lze také nastavit. Na obrázku 9 je zobrazena ukázka stránky pro hromadnou normalizaci slov.

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`[7], parametry filtrování jsou předány v rámci `URL`[8] adresy. Tyto parametry jsou následně předány již zmíněné metodě `getWordsForNormalization`, která vrátí požadovaná slova pro hromadnou normalizaci. Pro každé z těchto slov jsou dodatečně získány doporučené normalizované varianty voláním metody `getNormalizedVariantsOfWordAndWordsInSameCluster`, které je předáno dané slovo, pohlaví a název databázové tabulky, ve které je toto slovo uloženo. Pro slova, která zatím nebyla normalizována, je navíc pomocí metody `getMostFrequentWordOfCluster` zjištěno slovo které se nachází ve stejném shluku jako zadané slovo a je nejčastěji používané. 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 řečeco, 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.

Interaktivní změny formuláře, které zahrnují 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, je nalezena stejná již existující normalizovaná varianta popř. vytvořena nová normalizovaná varianta. Pokud uživatel nezadal vlastní normalizovanou variantu, je k dispozici identifikátor vybrané normalizované varianty. Takto získaný identifikátor normalizované varianty je pak uložen do atributu `norm_name_id` záznamu tohoto slova, čímž je slovo normalizováno.

7. Závěr

V této práci bylo řešeno rozšíření webové aplikace DEMoS, pro správu matričních záznamů, o možnost přiřazení slovům v těchto záznamech jejich normalizovanou podobu a zvýšit tak čitelnost a efektivitu vyhledávání těchto záznamů. Problém byl vyřešen roztříděním slov do shluků, které následné slouží pro sdílení normalizovaných variant slov v rámci daného shluku. Při vytvíření nebo editaci záznamů jsou pak nabízeny uživateli normalizované varianty podle toho v jakém shluku se slovo nachází, nebo by se nacházelo.

Shlukování slov bylo otestováno na reálných datech z matričních záznamů. Úspěšnost shlukování byla posuzována vzhledem k referenčnímu rozložení slov do shluků (*ground-truth*) a pohybovala se okolo 90 %.

Dalším krokem této práce by mohlo být například přidání veřejně dostupné databáze povolání (HISCO) a zvýšení úspěšnosti shlukování.

8. Poděkování

Rád bych poděkoval vedoucímu práce Ing. Jaroslavu Rozmanovi Ph.D. za odbornou pomoc a informace ohledně matričních záznamů, které mi během tvorby této práce poskytl.

[7] HTTP - https://cs.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[8] URL - https://cs.wikipedia.org/wiki/Uniform_Resource_Locator

Literatura

[1] *Words clustering using Python Clustering list of words*. Únor 2017, [Online; navštíveno 7.11.2018].
URL https://rapply.weebly.com/word-clustering/clustering-a-list-of-words/

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

[3] Zobel J., D. P.: *Phonetic string matching: Lessons from information retrieval.* ACM SIGIR, Zürich, Switzerland, 1996.

	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

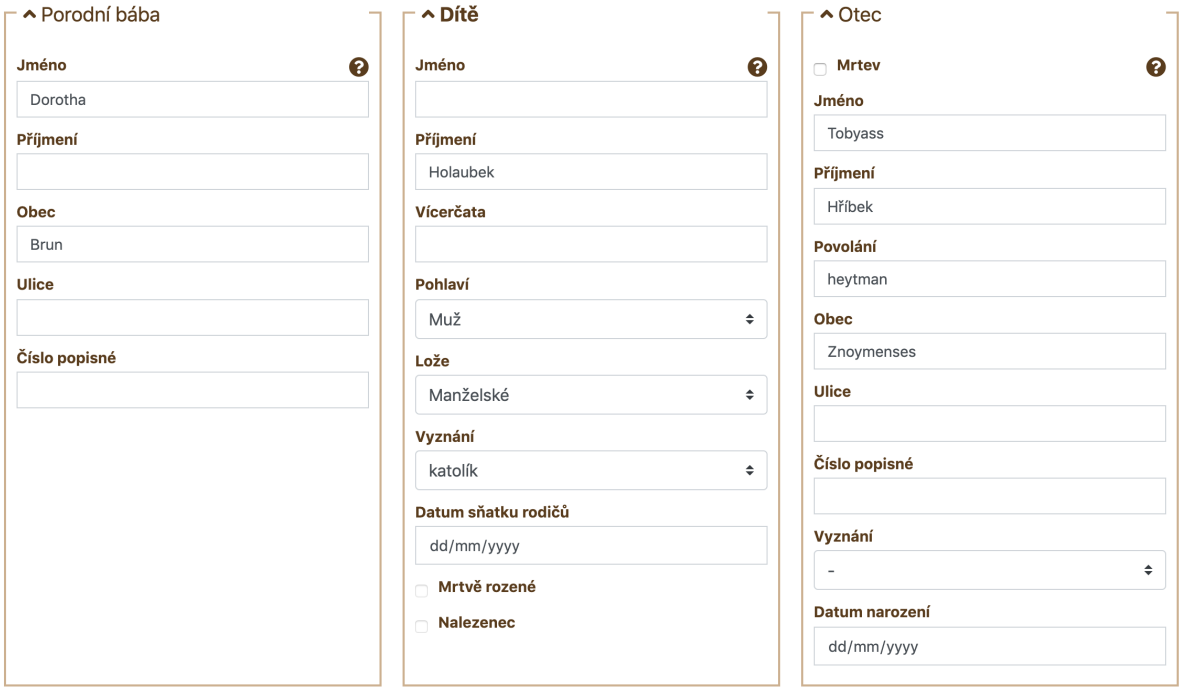
	Joan	Johanes	Johan	Petr	Peter
Joan	**1**	**1**	0.88	0	0
Johanes	0	1	0.83	0.18	0.16
Johan	0	0	1	0	0
Petr	0	0	0	1	0.88
Peter	0	0	0	0	1

Tabulka 5. Ukázka vypočtené matice podobnosti pro pět jmen. V levé tabulce je zobrzena původní vypočtená matice podobnosti. V pravé tabulce je zobrazena upravená matice podobnosti tak, aby slova která jsou již normlizována 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 upravním původní matice podobnosti.

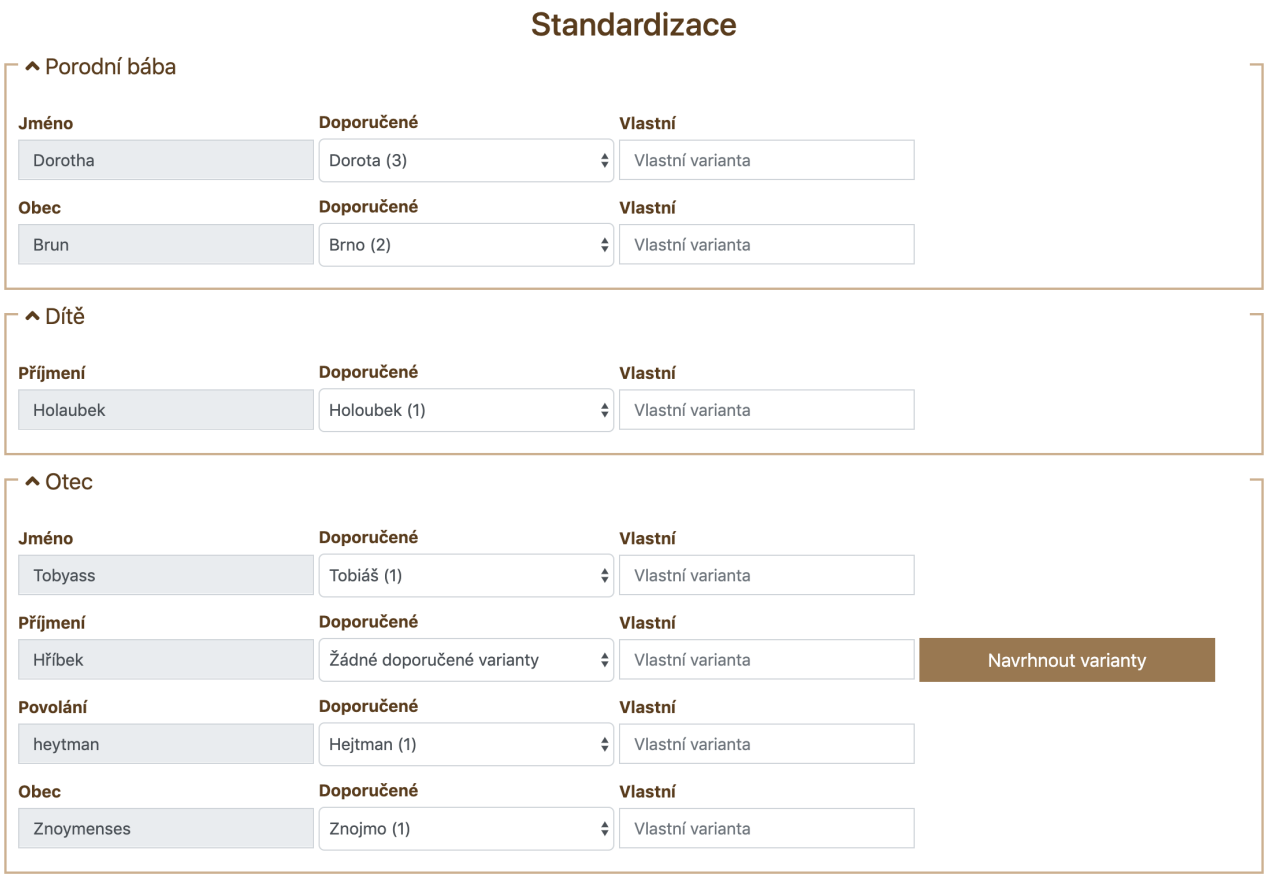
	Joan	Johanes	Johan	Petr	Peter
Joan	1	0	1	0	0
Johanes	0	2	2	0	0
Johan	0	0	3	0	0
Petr	0	0	0	4	4
Peter	0	0	0	0	5
číslo shluku	**1**	**2**	**1**	**4**	**4**

	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**

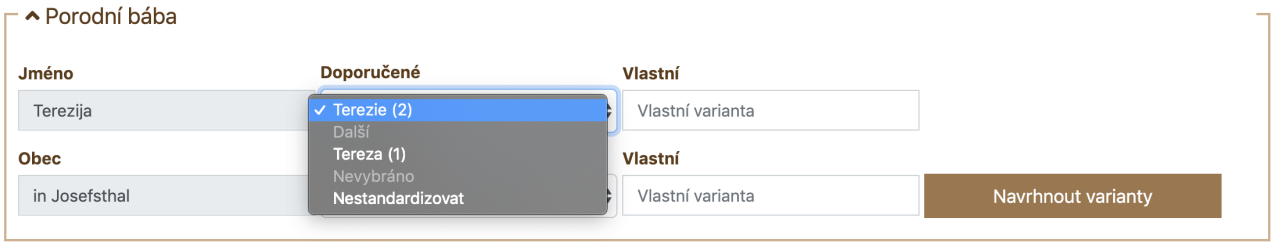
Tabulka 6. Tabulka vlevo zobrazuje vypočtenou matici shluků (`cluster_matrix`) vznilou z původní matice podobnosti, která je zobrazena v tabulce 5, vlevo. Tabulka vpravo zobrazuje původní matici shluků (vlevo) po tranzitivním uzavření. Díky tomu, že dvojice jmen (Joan, Johan) a (Johan, Johanes) byly 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. Poslední řádek obou tabulek obsahuje výsledná čísla shluků jednotlivých slov. Ty jsou získány jako nejmenejší číslo ve sloupci.



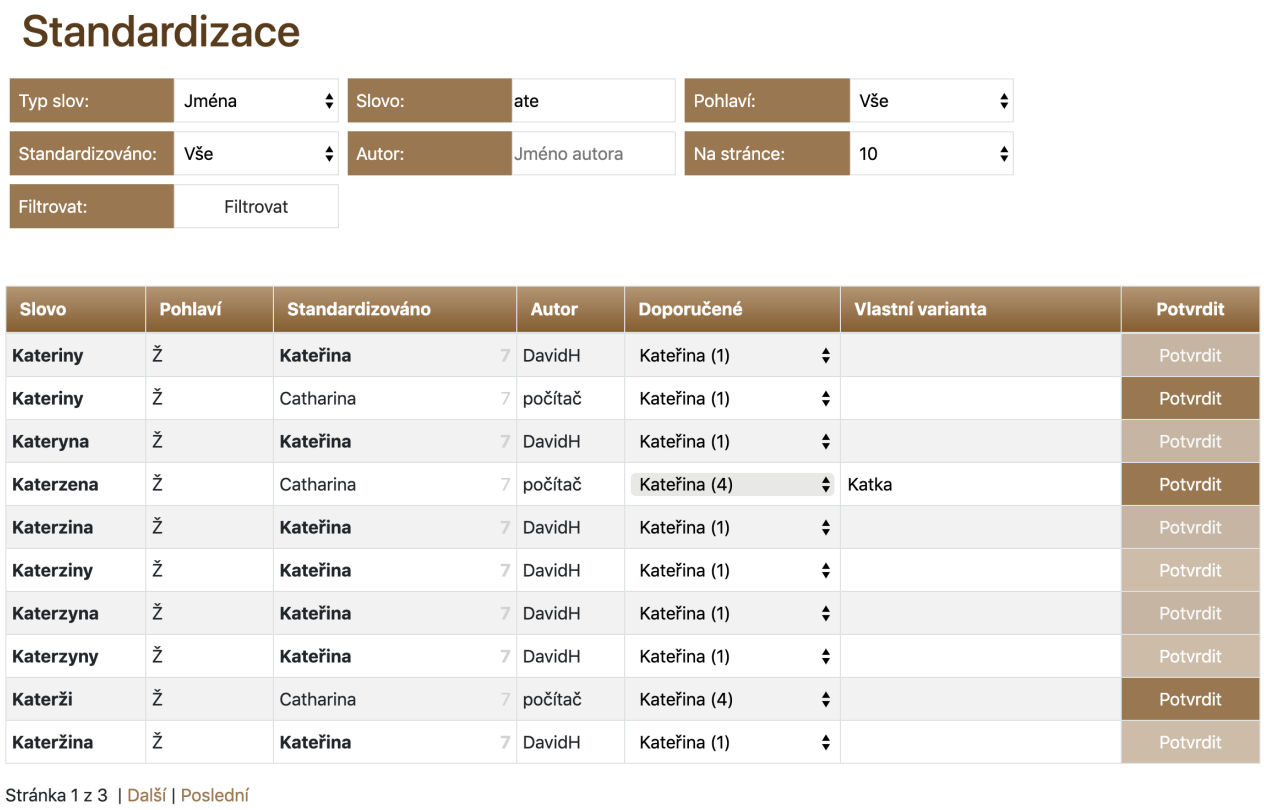
Obrázek 6. Ukázka části již existujícího formuláře pro vytvoření/editaci záznamu ve webové aplikaci DEMoS. Formulář je rozdělen do sekcí.



Obrázek 7. Ukázka normalizační části formuláře pro vytvoření/editaci záznamu ve webové aplikaci DEMoS. Formulář je rozdělen do sekcí. V prvním sloupci jsou vyplněny slova z původních polí formuláře. Druhý sloupec obsahuje doporučené normalizované varianty pro daná 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. Pro všechna slova, krom příjmení Hříbek, jsou dostupné normalizované varianty.



Obrázek 8. Výběr doporučených normalizovaných variant z fomulářového výběrového prvku. Pro danané jméno Terezija jsou na výběr dvě doporučené normalizované varianty. Slovo Terezija již bylo dvakrát normalizováno jako Terezie, proto varianta Terezie je jako první na výběr a je tedy i automaticky vybrána. Slovo Terezija se také nachází ve shluku slov, kde jedno z těchto slov již bylo normalizováno jako Tereza, proto v sekci Další je na výběr varianta Tereza. Poslední možností je nezvolit žádnou normalizovanou variantu.



Obrázek 9. Stránka pro hromadnou normalizaci slov z matričních záznamů. V horní části se nachází filtrační formulář. Pod tímto fomulářem je tabulka pro normalizaci slov. Každý řádek tabulky odpovídá jednomu slovu k normalizaci. Ve slopci Standardizováno jsou zobrazeny zvolené normalizované varianty pro jednotlivá slova (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. Vlastní normalizovaná varianta má přednost před vybranou normalizovanou variantou ve sloupci doporučeno, na toto je uživatel upozorněn změnou barvy doporučených normalizovaných varianty při zadání vlastní varianty. Tlačítko potvrdit je možné stisknout pouze pokud je pro dané slovo vybrána normalizovaná varianta, která by vedla ke změne jeho normalizace.