MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Aplikace shlukovacích metod na textové dokumenty

DIPLOMOVÁ PRÁCE

Bc. Kamil Čupr

Brno, 2007

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Poděkování

Poděkování patří panu doc. Ing. Janu Žižkovi, CSc. za cenné rady a příkladný teoretický úvod k tématu během jeho přednášek.

Shrnutí

Tato práce se zabývá reálnou aplikací vybraných shlukovacích metod Kohonenovy sítě a K-means na množinu textových dokumentů a přibližuje jejich použití běžnému uživateli.

Klíčová slova

Shlukování, Kohonenovy mapy, K-Means

Obsah

Kapitola 1

Úvod

Na světě je mnoho knih. Mnohem víc, než kdy bude jedinec schopen za život přečíst. A neustále vznikají nové. Při výběru knihy většinou vycházíme ze znalosti stylu, názorů autora a nebo ze zkušeností ostatních – přátel (či recenzentů), kteří již knihu přečetli a líbila se jim. Pokud navíc mají podobný pohled na svět jako my, je šance, že se kniha bude líbit i nám. Alespoň mnohem větší, než u náhodně vybrané publikace.

Podobná situace je i ve virtuálním prostředí Internetu. Denně vznikají tisíce různých článků (kratších článků vzniká mnohem více než obsáhlých publikací) a my jsme nuceni vybírat, čemu jsme ochotni věnovat svůj(!) čas.

S rozvojem informačních technologií a zejména webu je zveřejňování informací pro jedince snažší než kdykoli předtím. Takřka kdokoli s přístupem na Internet (a takových je stále víc a víc) může svoji myšlenku dát ostatním k dispozici téměř okamžitě.

Jednoduchost publikování má obrovský význam. Lidé s podobnými zájmy si tímto způsobem mohou rychle vyměňovat poznatky a zkušenosti, i když žijí v geograficky vzdálených oblastech. Bohužel s jednoduchým publikováním roste i počet autorů, kteří zaplavují společný informační kanál ne vždy kvalitním obsahem.

Kritérií pro výběr obsahu, který „stojí za to číst", je celá řada. Např. místo výskytu (diskusní skupina na dané téma), renomé autora, hodnocení uživatelů, klíčová slova . . . jak vidíme, některé principy z tištěného světa lze použít i v tom elektronickém. Jiné (hodnocení uživatelů) mají smysl jen při dostatečném počtu hodnotících. Jenže co když je obtížné jich dostatečný počet z různých důvodů zajistit?

My se v dalších částech tohoto textu budeme snažit přispět dalším

použitelným způsobem k usnadnění předvýběru velkého množství potenciálně relevantního obsahu, a tím ušetřit (drahý, odborný) čas čtenáře, uživatele, odborníka.

Zadání práce klade důraz zejména na praktické využití již popsaných metod, součástí řešení je klientská aplikace umožňující uživatelům aktivně využívat diskutované metody, a tedy i struktura práce bude podobná postupu implementačních prací.

V druhé kapitole se seznámíme se shlukovou analýzou (shlukováním) obecně, představíme shlukování pomocí Kohonenových map a metody k-means.

Ve třetí kapitole si ujasníme některá témata, která jsou zajímavá z implementačního hlediska.

Ve čtvrté kapitole představíme aplikaci samotnou se stručným návodem k použití a zkusíme si rozdělit na shluky pomocí obou metod modelová data z medicínského prostředí.

V závěru se pokusíme zhodnotit dosažené výsledky a porovnat obě metody z uživatelského hlediska.

Kapitola 2

Úvod do problematiky

Shlukem rozumíme skupinu prostorově blízkých objektů. Pro lepší představu – typicky se hustota objektů s rostoucí vzdáleností od středu námi identifikovaného shluku zmenšuje. Metody shlukové analýzy se zabývají jednak identifikací a rozborem jednotlivých shluků, a zejména také postupy vedoucími ke zobrazení nepřehledné či složité množiny vstupních dat do vhodnějších prostorů, kde je snažší shluky nalézt a analyzovat.

V našem případě se budeme snažit aplikovat vybrané shlukovací metody na množinu blíže neurčených textových dokumentů. Cílem je usnadnění procesu třízení uživatelem sesbíraných (pro něj potenciálně zajímavých textů) na třídy dokumentů s podobným obsahem. Podobné dokumenty by po aplikování příslušné metody měly být reprezentovány právě shluky.

V dalším textu si stručně přiblížíme nezbytný teoretický úvod k implementovaným metodám – k-means a Kohonenovy mapy.

2.1 Teorie, k-means

Metoda k-means (česky k-průměrů) je jedním z velkého počtu shlukovacích algoritmů. Je založen na vzdálenosti bodů v mnohorozměrném prostoru. Každý hodnocený objekt je reprezentován právě jedním bodem, každý sledovaný atribut pak jednou souřadnicí. O konkrétní vazbě bodu na objekt (v našem případě na textový dokument) se zmíníme později. Metoda k-means je vhodná zejména na velké množiny dat, které mají být roztřízeny do malého počtu shluků.

Algoritmus je iterační. V inicializační části nastavíme počet shluků k, které na výstupu požadujeme (odtud **k**-means). Je vytvořeno k bodů s náhodnými souřadnicemi (budoucí středy shluků – tzv. cent-

roidy).

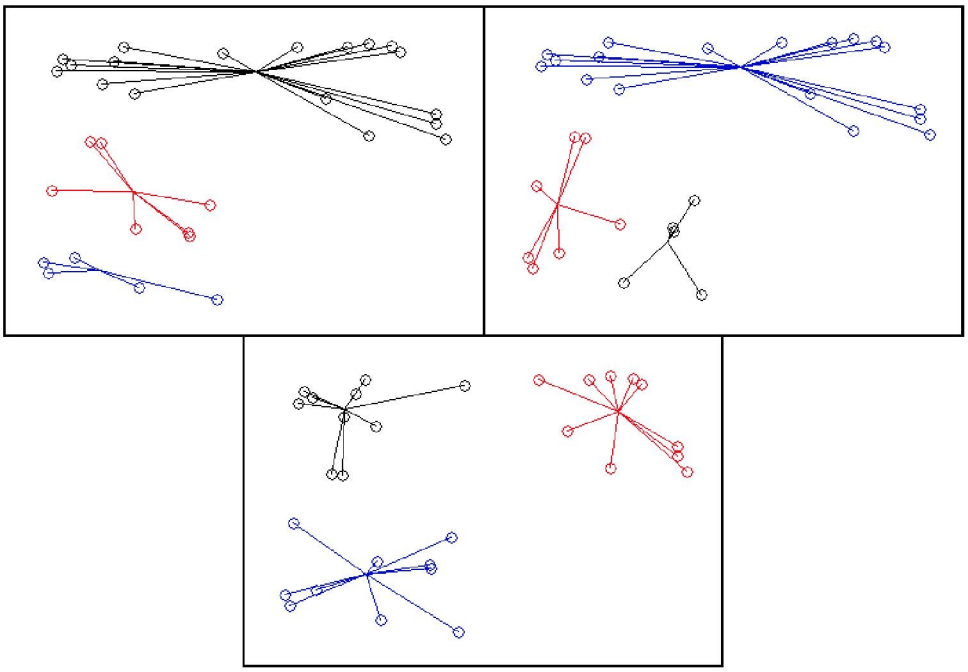
V iterační části je v každém kroku nejdříve každý bod přiřazen nejbližšímu centroidu (shluk je tedy reprezentován všemi body, které jsou nejblíž stejnému centroidu). Poté jsou všechny centroidy aktualizovány – umístěny do středu jim přiřazených bodů - „svých" shluků (střed shluku = průměrný bod, odtud k-**means**, čili k-průměrů). Iterace pokračuje, dokud se mění příslušnost některých bodů k centroidům. Je dokázáno, že algoritmus je konečný (více [MacQueen]).

2.1.1 k-means: algoritmus

- vstup: počet požadovaných shluků k; body (vektory), které chceme rozdělit do shluků
- výstup: k shluků (disjunktních množin vstupních bodů)
- algoritmus:
  - i inicializuj k centroidů náhodnými sořadnicemi
  - ii dokud se mění nejbližší centroid libovolného bodu:
    - pro každý bod:
      - přiřaď bod nejbližšímu centroidu
    - změň polohu centroidu - nastav ji jako průměr všech bodů tomuto centroidu přiřazených
  - iii body přiřazené stejnému centroidu tvoří jeden shluk, máme k shluků

2.1.2 k-means: doplnění

Algoritmus nezaručuje výběr nejlepšího řešení (nalezne pouze lokální optimum) a ani nevede pokaždé k témuž výsledku (jinou inicializací centroidů můžeme dostat jiný výsledek - viz. obr. 2.1). Proto většinou počítáme mnohokrát s různým počátečním natavením centroidů a uchováváme nejlepší výsledek, který nakonec prezentujeme. Pro výběr nejlepšího dosaženého řešení lze použít vhodně zvolená metrika (tzv. „energetická funkce") – např. součet všech vzdáleností



Obrázek 2.1: k-means: stejná data, různé výsledky, dole nejlepší

od jednotlivých bodů k jejich centroidům. Řešení s nejmenší takto vypočítanou hodnotou je nejlepší námi dosažené (ale opět pouze lokální optimum).

2.2 Teorie, Kohonenovy mapy

Kohonenova mapa je samoorganizující se (učení bez učitele) neuronová síť. Vstupní data jsou postupně předkládána síti, síť během procesu učení podle jejich hodnot mění svou strukturu. Po předložení vzoru na vstup sítě je vzor zvážen váhovým vektorem každého neuronu. Neuron s nejlepším výsledkem (neboli vítězný neuron, též pouze „vítěz") poté modifikuje své váhy tak, aby lépe odpovídaly předloženému vstupu – adaptuje se. Neurony jsou uspořádány v jedné vrstvě. Každý neuron si uchovává kromě hodnot svých vah také informaci o svém okolí – sousedících neuronech, sousedech. „Adaptací" se rozumí úprava vah tak, aby byla odchylka od zpra-

covávaného vzoru menší. Během učení jsou adaptovány nejen váhy vítězného neuronu, ale i jeho okolí. (Během učení se adaptací zasažené okolí může zmenšovat, neurony vzdálenější od vítěze mohou být adaptovány méně, adaptace se může zmenšovat i s postupujícím časem – všechny tyto možnosti jsou umožněny implementací vlastní adaptace a procesu učení, konkrétně je popíšeme v poznámkách k implementaci). Naučená síť je posléze schopna klasifikovat i dříve nepředložené (neviděné) vzory. Vzory předkládané během procesu učení nazýváme též „trénovací data", vzory neviděné pak „testovací data".

2.2.1 kohonenova mapa: algoritmus učení

Algoritmus je opět iterační, předem musíme definovat podmínky ukončení (např. zvolený počet iterací, uspokojivá chyba při klasifikaci, interakce uživatele).

- vstup: počet neuronů n; množina bodů (vektorů) trénovacích dat T
- výstup: naučená neuronová síť
- algoritmus:
  - i inicializuj náhodně váhy všech neuronů
  - ii dokud není splněna podmínka ukončení
    - pro každý bod z T:
      - najdi „vítěze"
      - adaptuj váhy vítěze a jeho okolí
  - iii naučená síť je schopna klasifikovat i vzory z množiny testovacích dat – podobné vzory mají blízké vítěze

2.2.2 kohonenova mapa: okolí

Definice „okolí" v sobě skrývá vztah mezi grafickou reprezentací a prostorovou blízkostí neuronů.

Mějme např. n neuronů v poli $[1..n]$

- Definujeme-li sousedy ve vzdálenosti d od k-tého neuronu jako $k - d$ a $k + d$ (neurony vpravo a vlevo), graficky bude vhodné síť reprezentovat na přímce.

- Definujeme-li sousedy ve vzdálenosti d od k-tého neuronu jako $((k - d)\ mod\ n) + 1$ a $((k + d)\ mod\ n) + 1$ (neurony vpravo a vlevo, 1 sousedí s n), bude vhodné síť reprezentovat jako kruh.

- Zavedelme-li nové parametry pro šířku a výšku mřížky, můžeme podobně definovat různým způsobem okolí pro dvourozměrnou mřížku (např. neurony vpravo, vlevo, nad, pod) s různým počtem sousedů (např. 4, 8 $\Rightarrow$ klasická pravoúhlá mřížka X x Y; 6 $\Rightarrow$ šestiúhelníkový „včelí plást"; atd.) Obdobně se lze dostat ke třem a více rozměrům, pro grafickou reprezentaci nám však ony dva budou ve většině případů stačit.

Podrobněji se zde rozepisovat nebudeme, naším cílem je metody nastínit a použít, nikoli znovu precizně popsat již mnohokrát popsané. Více viz [KOHO].

Kapitola 3

Poznámky k implementaci

3.1 Příprava vstupních dat

Naším úkolem je klasifikovat textové dokumenty – tj. soubory slov. Obě zde popisované metody ovšem pracují na svých vstupech s množinami bodů (nebo též vektorů). Nezbývá nám tedy než každý zpracovávaný textový dokument převést na odpovídající vektor a dále zpracovávat tento. Nebudeme se zabývat žádným jazykově závislým předzpracováním (různé tvary téhož slova, synonyma, ...) – zde je volný prostor pro jiné specializované projekty – přednost před češtinou tedy ve zdrojových dokumentech dostane záludností prostší angličtina. Způsobů převodu dokumentu na vektor je opět celá řada. My použijeme ten nejjednodušší – předem vybereme množinu zajímavých slov s daným uspořádáním (pořadí slova odpovídá souřadnici vektoru) a souřadnice vektoru pro příslušný dokument poté vyjadřuje, zda se slovo na odpovídající pozici v onom dokumentu vyskytuje či nikoliv (0 nebo 1). Dimenze d takto vytvořených vektorů bude zřejmě rovna počtu vybraných slov a vektory budou ležet v rozích d-rozměrné krychle. Pro zjednodušení výpočtů lze pak místo euklidovské vzdálenosti mezi dvěma body použít vzdálenost hranovou (tj. minimální počet hran pro přechod z jednoho bodu do druhého, stačí sečíst počet lišících se souřadnic).

Příklad:

- množina zajímavých slov: (red, green, blue, purple)
- soubor 1: [blue sky birds everywhere] bude mít souřadnice (0,0,1,0)

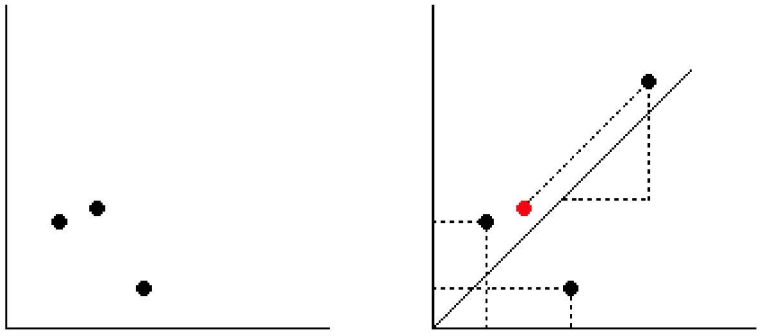
- soubor 2: [rgb is acronym for red green blue] bude mít souřadnice (1,1,1,0)
- soubor 3: [deep purple rocks] bude mít souřadnice (0,0,0,1)

Proces výběru zajímavých slov je v programu tvořen těmito kroky:

- vytvoření globálního slovníku všech slov ze všech dokumentů
- pro každé slovo je uchován počet dokumentů, ve kterých se toto slovo vyskytuje
- výběr zajímavých slov jako základ pro vektory je ponechán v režii uživatele, uživatel má k dispozici nástroje jako výběr na základě slovníku, ignorování na základě slovníku, uložení výběru jako nového slovníku a podobně, náhodný výběr, prvních n, dovýběr do n, dalších n, kombinace předchozích – tímto způsobem lze vynechat nejčastější nebo neutrální slova (slovo obsažené ve všech dokumentech je stejně nezajímavé jako třeba spojky nebo členy)
- na základě uživatelem vybrané množiny slov je vygenerován vektor stejné dimenze pro každý dokument; aby naše počínání mělo smysl, musíme si pamatovat, který vektor reprezentuje který dokument – na základě shluku vektorů vygenerujeme seznam dokumentů
- množina vektorů je dále zpracovávána vybranou metodou (k-means nebo kohonenovou sítí), dimenze vektorů je jedním ze vstupů pro danou metodu

3.2 K-means

Cesta od teorie k implementaci je v případě k-means poměrně přímočará (viz. 2.1). Vstupních parametrů je opravdu málo – pouze počet chtěných shluků, případně počet pokusů o shlukování, ze kterých se bude vybírat nejlepší případ. Z nejlepšího případu si stačí pamatovat polohu centroidů, nový výpočet shluků k již ustáleným centroidům je rychlý. Jediným problémem zůstává grafická reprezentace výsledků, typický to problém projekce mnoharozměrných dat do



Obrázek 3.1: vzdálenost ve 2D projekci, 3D originál

dvou-tří dimenzí. Pokud použijeme nejjednodušší způsob – ignorování přebytečných dimenzí a použití pouze prvních dvou souřadnic, už u tří-dimenzionálního prostoru se může stát, že v zobrazení nejbližší body budou ve skutečnosti ty nejvzdálenější (viz. obr. 3.1).

Existují různé metody redukce dimenze, ale pro zobrazení z prostoru s několika stovkami dimenzí (závisí na počtu sledovaných slov) do dvou, příp. tří dimenzí to jen kvůli lepšímu zobrazení nemá smysl. Vhodnějším místem pro použití takových metod by byla fáze předzpracování dokumentů – během výběru vhodných slov (například vybírat vždy jen jednoho zástupce z množiny slov, která svým výskytem ve všech dokumentech významným způsobem korelují).

Nakonec jsem tedy zvolil vlastní přístup založený na myšlence zachování nejkratších vzdáleností (zachování nejméně vzdálenosti k nejbližšímu sousedovi). Bude k tomu využita minimální kostra grafu, v algoritmu popsaném následujícími kroky:

i vygeneruj minimální kostru úplného grafu daného body a vzdálenostmi (body = uzly, vzdálenosti = hrany)

ii nakresli první uzel na náhodné místo

iii nakresli ještě nenakreslené hrany (včetně cílových uzlů) k ještě nenakresleným uzlům vedoucí z již nakreslených uzlů se zachováním délky hrany pod náhodným úhlem

iv opakuj iii dokud existují nenakreslené uzly

Ono „vygenerování minimální kostry úplného grafu" je paměťově náročné (protože úplný graf o n uzlech má $\frac{n}{2} * (n-1)$ hran, které je potřeba všechny zkontrolovat, což je při desítkách tisíc dokumentů stovky milionů hran), lze však zvolit (a tedy byl zvolen a je implementován) přijatelný postup, kdy není třeba mít v paměti všechna data najednou:
algoritmus:

- vytvoř graf obsahující pouze uzly
- setřiď všechny hrany od nejmenší k největší (vícecestným slučováním, na disku, v paměti jsou tedy jen právě porovnávané elementy)
- dokud existují izolované podgrafy procházej všemi hranami od nejmenší k největší
  - pokud by přidáním hrany do grafu vznikl cyklus, ignoruj hranu, jinak přidej hranu do grafu

Pro vícecestné slučování jsem navrhl a implementoval hybridní postup využívající třízení haldou. Algoritmus třízení haldou vypadá následovně:
algoritmus:

i vytvoř z nesetřízených dat haldu

ii odeber prvek z vrcholu haldy, zařaď jej na konec již setřízených dat a nahraď jej nejpravějším nejdolnějším prvkem z haldy (tím se poruší halda)

iii uprav zbylá data tak, aby znovu tvořila haldu

iv opakuj kroky ii a iii, dokud exitují nesetřízená data

V mé modifikaci nejdříve setřídím konvenčním algoritmem menší bloky, které se snadno vejdou do paměti, a ty potom zatřizuji dohromady právě s využitím třízení haldou.
algoritmus:

1 rozděl množinu nesetřízených dat na bloky, které se pohodlně vejdou do paměti

2 setřiď (libovolným jiným algoritmem dostupným z programovacího prostředí) každý blok zvlášť

3 z prvních prvků všech bloků vytvoř haldu

4 odeber prvek z vrcholu haldy, zařaď jej na konec již setřízených dat a nahraď jej dalším z téhož bloku – pokud je blok prázdný, nahraď jej nepravějším nejdolnějším prvkem z haldy (jako v ii z předchozího algoritmu)

5 obnov haldu

6 opakuj kroky 4 a 5, dokud existuje neprázdný blok

Uvedeným postupem se vyhneme potížím s nedostatkem paměti a jeho časová složitost zůstává ve třídě $O(n * log\ n)$ (ovšem za předpokladu, že jsme na setřizování menších bloků použili konvenční algoritmus ve stejné třídě časové složitosti)

Test na přítomnost cyklu lze provést optimálně tak, že si všechny izolované podgrafy a počet uzlů v nich evidujeme: na počátku je jich stejně jako uzlů, každý podgraf obsahuje jeden uzel (v grafu jsou pouze uzly), každý uzel nese informaci o příslušnosti k podgrafu. Spojuje-li právě zpracovávaná hrana uzly se stejného podgrafu, vytvořila by cyklus a tudíž ji nepřidáváme. Jinak ji přidáme a ze dvou různých podgrafů vznikne jeden – ten menší (s menším počtem uzlů) přidáme do většího (kvůli optimalizaci – musíme jím projít a aktualizovat uzly). Algoritmus končí v okamžiku, kdy existuje právě jeden podgraf.

Grafická prezentace shluků z mnoharozměrného prostoru převedená do dvou dimenzí v naprosté většině případů bude spíše zavádějící. Je tedy na povážení, zda vůbec běžného uživatele obtěžovat

pro laika nezajímavým obrázkem, i když vygenerovaným pomocí z odborného hlediska zajímavých matematických a geometrických inovativních přístupů. Výsledek je sice lepší než náhodný výběr dvou souřadnic, nicméně stále nedosahuje dostatečné vypovídající hodnoty jako u zobrazení shlukování provedeného přímo nad dvourozměrnými daty. Proto si myslím, že mnohem větší službu odvedou prosté seznamy shlukům odpovídajích dokumentů.

3.3 Kohonenova mapa

Implementace kohonenovy sítě dává prostor programátově fantazii mnohem více. Oproti k-means se při inicializaci nastavuje větší počet parametrů, lze experimentovat jak s vlastními parametry, tak se způsobem jejich použití.

Výběr vítěze zůstává jednoduchý – pro daný vstupní vektor $\overrightarrow{x}$ z n-rozměrného prostoru všech vstupů je to neuron s minimální hodnotou euklidovské vzdálenosti od vektoru vah $\overrightarrow{w}$, čili součet rozdílů odpovídajících si souřadnic $\sum_{i=1}^{n}(w_i - x_i)$ (odmocninu lze v implementaci vynechat, protože nás zajímá nejmenší hodnota a jelikož platí $x > y \Leftrightarrow \sqrt{x} > \sqrt{y}$, pro $x \geq 0$ a $y \geq 0$).

Adaptace i-té složky vah j-tého neuronu (vítězný neuron a jeho okolí) v čase $t + 1$ probíhá podle

$$w_{ij}(t + 1) = w_{ij}(t) + \eta(t) * \alpha(t)[x_i(t) - w_{ij}(t)],$$

kde

- α je tzv. koeficient učení (nebo též bdělostní koeficient), je z inervalu $(0, 1)$ a klesá v čase (implementováno jako lineární snižování mezi zadanými hodnotami v závislosti na čísle iterace)
- η je funkcí sousedství (neighbourhood) – vrací hodnotu z intervalu $(0, 1)$ a může klesat s rostoucí vzdáleností od vítěze v definovaném okolí ... také velikost onoho okolí může klesat v čase (obě funkce implementovány jako lineární klesání v uživatelem definovaných intervalech)

Nastavení iniciální hodnot parametrů je ponecháno na rozhodnutí uživatele – jelikož uživatel má hrubou představu o množině

vstupních souborů. Může zkoušet různé vstahy mezi velikostí sítě, velikostí sousedství v čase, velikostí bdělostního parametru v čase a příp. i topologií sítě (je dána definicí sousedních neuronů pro obecný neuron, jak bylo zmíněno v 2.2.2).

K použití je naimplementováno několik základních topologií na přímce a v rovině:

1 1-D řetěz, neuron n s přímými sousedy $n+1$ a $n-1$

2 1-D řetěz, konci spojený (jako 1. + 1 sousedí s n, tedy kruh)

3 2-D mřížka, neuron (x, y) s přímými sousedy nad $(x, y-1)$, pod $(x, y+1)$, vpravo $(x+1, y)$ a vlevo $(x-1, y)$

4 2-D mřížka, jako 3., přidány neurony $(x+1, y+1)$ a $(x+1, y-1)$ (6 sousedů, jako včelí plást)

5 2-D mřížka, jako 4., přidány neurony $(x-1, y+1)$ a $(x-1, y-1)$ (8 sousedů, okraje mřížky 3x3)

Funkce sousedství poté dostane jako parametr maximální vzdálenost od daného neuronu. Neurony ve vzdálenosti 1 jsou přímí (definovaní) sousedé, neurony ve vzdálenosti $d+1$ jsou generovány iterativně prohledáváním do šířky jako ještě neprocházení sousedé neuronů ve vzdálenosti d.

Síť je po natrénování schopna klasifikovat i dříve neviděné vzory, vlastní klasifikace spočívá pouze ve výběru vítěze. Podobné vzory budou mít stejného vítěze, a jelikož při adaptaci je k neuronu „přitahováno" i jeho okolí, je patrná i tendence zvýšené podobnosti vzorů mezi bližšími sousedy. Ve výstupu pro uživatele je tedy vhodné zvýraznit neurony s nejvyšším počtem vítězství po klasifikaci celé množiny testovacích dat včetně seznamu vektorů (dokumentů), pro které daný neuron zvítězil – to jsou tedy ony kýžené shluky.

Kapitola 4

Poznámky k aplikaci

Pro implementaci jsem zvolil programovací jazyk Java ve verzi 1.5. Důvodů pro tuto volbu je hned několik – snadná přenositelnost mezi různými operačními systémy, objektový přístup, bohatý a dobře dokumentovaný framework a v neposlední řadě příjemné grafické prostředí Swing. Vývoj byl realizován ve vývojovém prostředí Netbeans, které nabízí nadstandardní komfort svou provázaností s dokumentací a také výborným GUI-builderem Matisse.

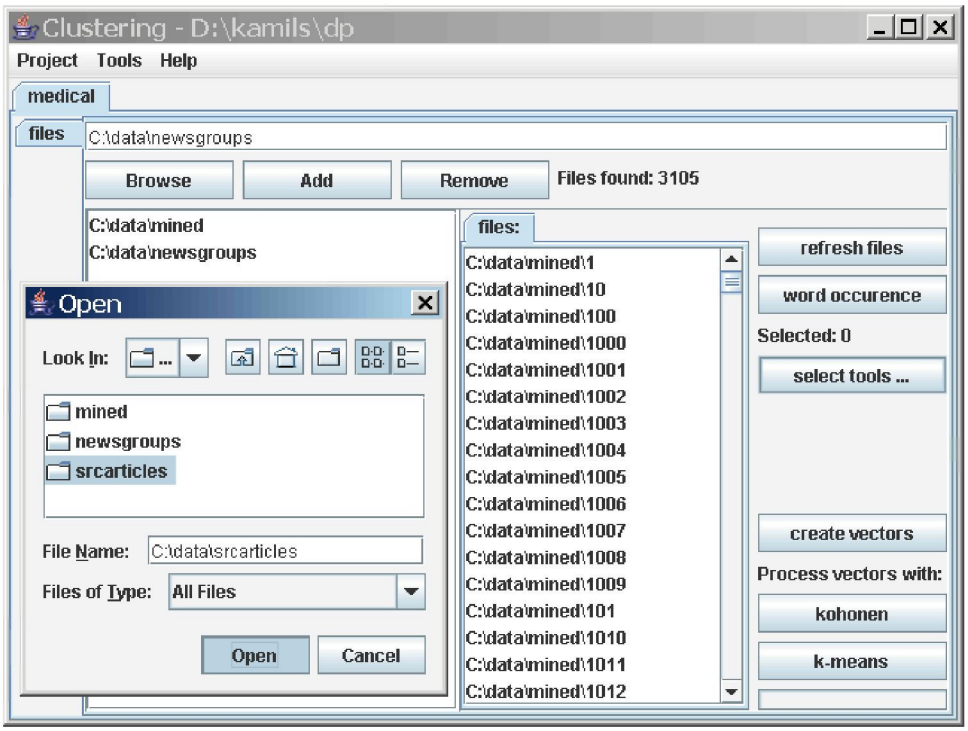
(Na tomto místě bych se rád omluvil za občasné používání anglických výrazů. V programátorské praxi se jich z pochopitelných důvodů užívá hojně ať už proto, že se český ekvivalent obtížně hledá, skoro nikdo by mu nerozuměl, a anglický výraz by musel následovat v závorce pro vysvětlení, a nebo proto, že prostě neexistuje.)

4.1 Spuštění

Spuštění se provádí pomocí příkazu

```
java -jar clustering.jar -Xmx512m
```

Poslední přepínač je pro java framework udávající velikost paměti, kterou je možno programu přidělit (samozřejmně lze použít veškeré přepínače java frameworku). Při podobných úlohách jako je ta naše platí „čím více, tím lépe", 512 megabajtů je v tomhle případě doporučené minimum. Mělo by stačit na shlukování řádově tisíců dokumentů převáděných na stovky dimenzí (stovky sledovaných slov).



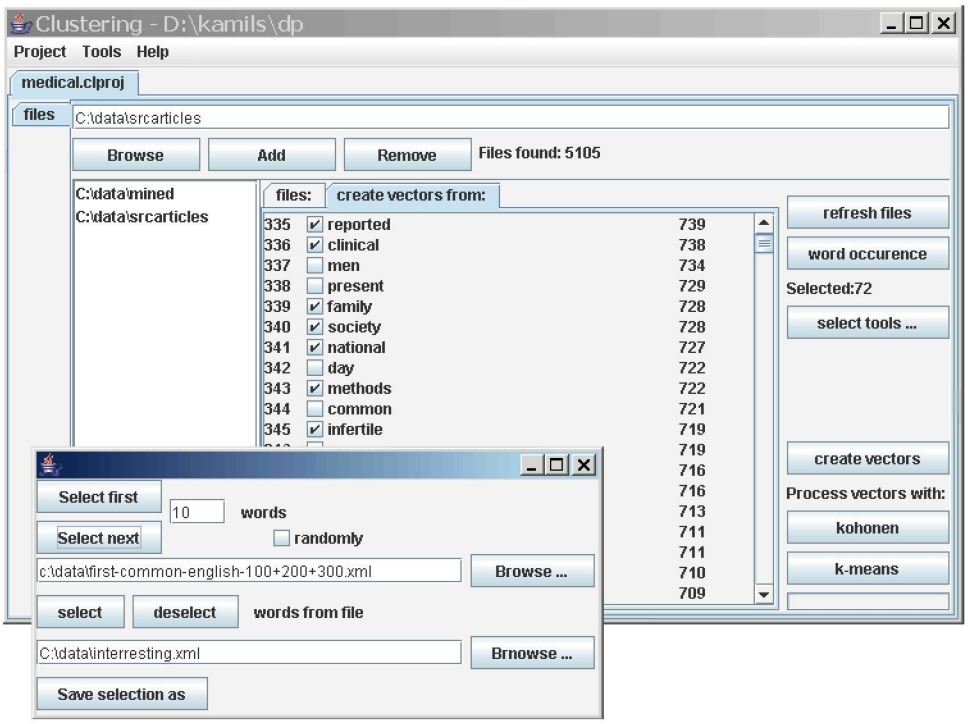
Obrázek 4.1: První krok

4.2 Společný základ

První fáze, totiž analýza zdrojových dokumentů, výběr zajímavých slov a převod dokumentů na vektory je společná pro obě metody. Veškerá dosud spočítaná data jsou uložena v projektovém souboru (ve formátu XML), který má formát ¡název projektu¿.clproj a v pomocném adresáři s názvem .metadata-¡název projektu¿.

Prvním krokem po založení nového projektu (z hlavního Menu Projekt-¿New)je výběr zdrojových adresářů, jejichž celý obsah se bude posléze analyzovat. Dokumenty k analýze je tedy potřeba mít uložené v adresáři (více adresářích, viz. obr. 4.1).

Poté je nutno pokračovat stiskem tlačítka "word occurence" – to spustí analýzu nalezených souborů. Do nové záložky s názvem "create vectors from" je vygenerován slovník všech slov a nabídnut uživateli setřízen podle počtu dokumentů, v kolika se dané slovo

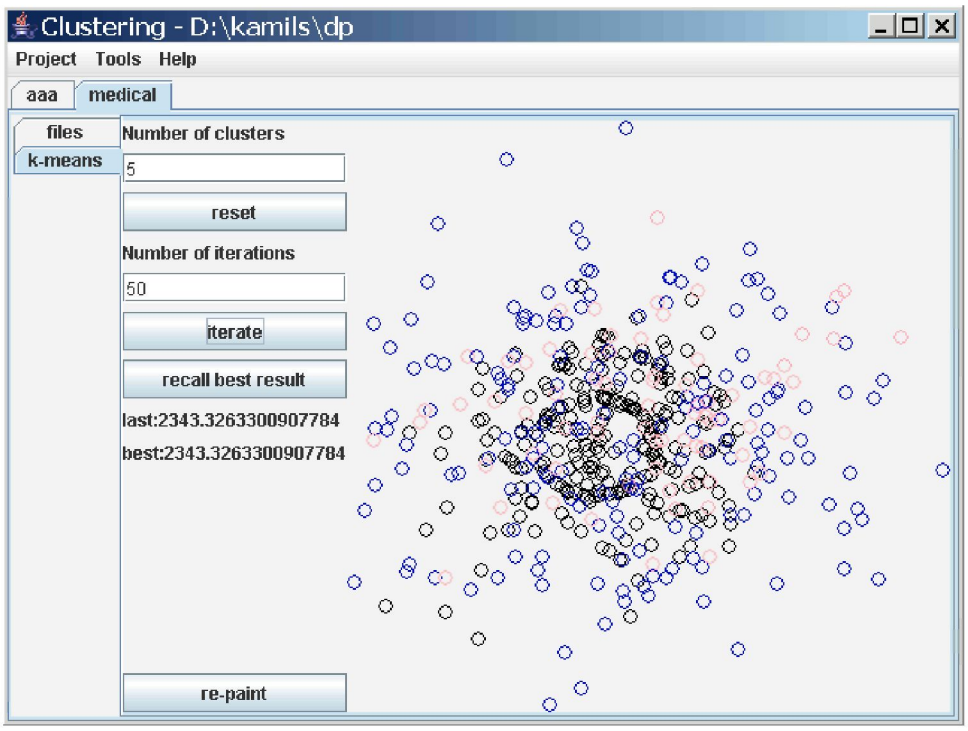


Obrázek 4.2: výběr slov jako základu pro tvorbu vektorů

vyskytuje (nejčastější nahoře). Uživatel poté vybere slova, na základě kterých se budou pro jednotlivé dokumenty tvořit vektory. Může použít pomocné označovací nástroje (viz. obr. 4.2), zejména označení (odznačení) na základě jiného (ze souboru načteného) slovníku. Vybraná slova lze do takového souboru uložit k dalšímu použití. Jako základ pro vektory je rozumné vybírat řádově maximálně stovky slov.

Výsledkem je „polotovar", který je vhodné uložit a vyhnout se tak zbytečnému absolvování téhle části opakovaně.

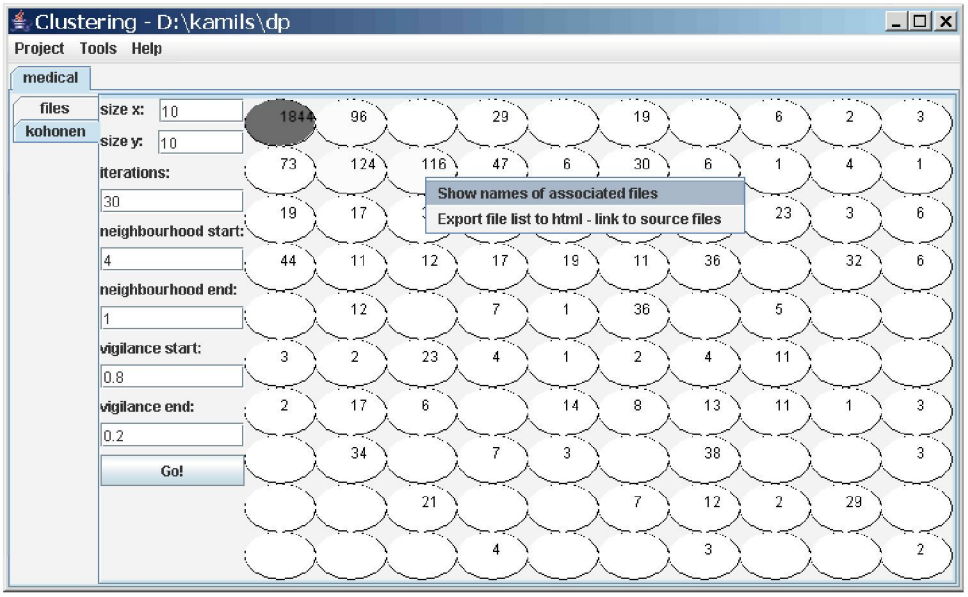
Jakmile jsou k dispozici vygenerované vektory (buď předchozím generováním nebo již hotové po načtení z projektového souboru), lze přikročit k další fázi, totiž vlastnímu shlukování, a to výběrem příslušné metody jednoduše stisknutím tlačítka s jejím názvem. Každá metoda má své vlastní rozhraní, u každé se nastavují jiné parametry.



Obrázek 4.3: Výstup shlukování pomocí k-means

4.3 K-Means

U k-means shlukování lze nastavit počet shluků a počet iterací. Grafický výstup nemá přílišnou vypovídající hodnotu, jelikož je ale v zobrazovacím algoritmu (který zachovává minimální vzdálenosti nalezené pomocí minimální kostry) zastoupen náhodný prvek, je zde možnost „překreslení" – a obrázek bude vypadat jinak při stejných hodnotách vstupů. Nechybí možnost projít seznam souborů příslušejících nalezeným shlukům včetně exportu takového seznamu do html.



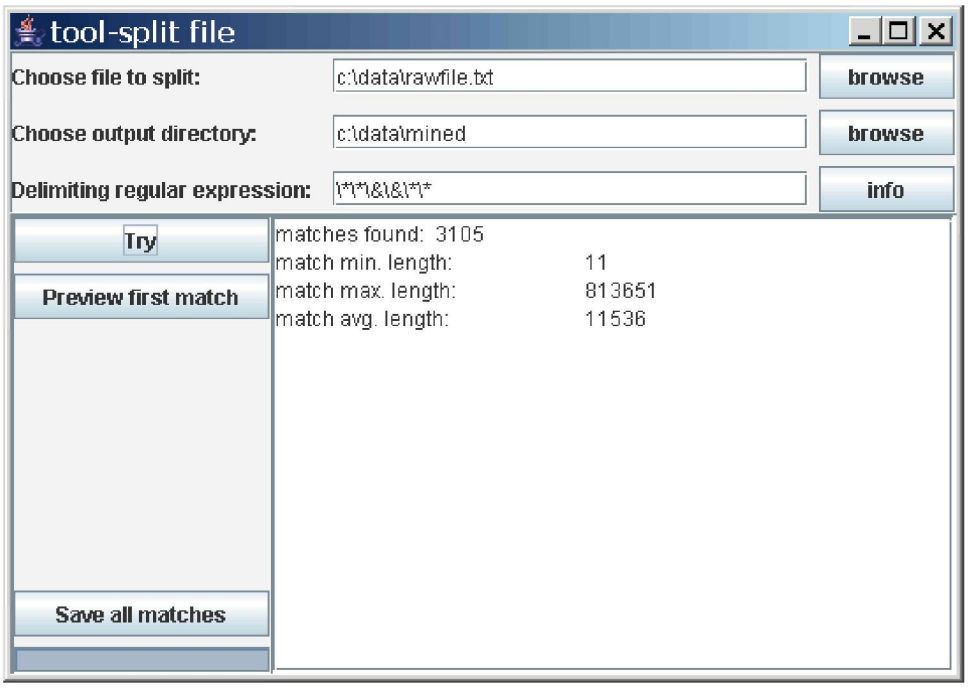
Obrázek 4.4: Výstup shlukování pomocí kohonenovy mapy

4.4 Kohonenova mapa

Na ovládacím panelu kohonenovy mapy lze před shlukováním různé parametry: rozměry sítě, počet iterací, bdělostní koeficient na počátku (při první iteraci) a na konci (při poslední iteraci), velikost ovlivňovaného sousedství. Tlačítkem "Learn!" se spouští proces učení se na projektových datech s nastavenými parametry (viz obr. 4.4. Tlačítkem classification lze načíst soubory z jiného projektu (speciálně pro tento účel vytvořeného) pouze pro klasifikaci již naučenou sítí.

4.5 Společný závěr

Výsledkem obou metod je (kromě pokusu o grafické znázornění) seznam shluků setřízený podle významnosti (počtu dokumentů do shluku příslušejících), z každého shluku jsou přístupné názvy a obsahy všech „jeho" dokumentů. Je zde možnost exportu do HTML – prostředí internetového prohlížeče nabídne lepší komfort při pročítání se výsledky.



Obrázek 4.5: Rozdělovač souborů

4.6 Rozdělovač souborů

Některá dodaná medicínská data byla obsažena v jediném souboru s oddělovači. Lze předpokládat, že to nebyl jen ojedinělý úkaz, a tudíž je v aplikaci obsažen a z menu dostupný jednoduchý nástroj na rozdělení souboru podle regulárního výrazu. Uživatelské rozhraní je minimalistické, přímočaré (viz. obr. 4.5)

Kapitola 5

Závěr

Obě metody mají své výhody a nevýhody. K-means je vhodnější pokud nám na výstupu stačí malý počet shluků a je-li předpoklad, že takové shluky v datech skutečně existují. Horší je to s vizuální prezentací dosažených výsledků.

Kohonenovy mapy jsou mnohem složitější v iniciační části – uživatel je nucen pochopit aspoň částečně teorii, aby mu byl jasný dopad všech parametrů, které nastavuje. Výsledky je možno ihned nabídnout zpracované v grafice, celý proces se jeví uživateli transparentněji.

Na základě zkušeností z nesčetných experimentů s různými množinami souborů mohu deklarovat jako vhodnější metodu pro uživatele shlukování pomocí kohonenovy mapy. Nevyžaduje totiž žádnou znalost o datech.

Aplikace může dobře posloužit jak pro seriózní práci s reálnými daty, tak pro výukové účely pro demonstraci fungování obou metod.

Na přiloženém cd najdeme zdrojové kódy programu, přeložený program v binárním formátu, některé grafické výstupy, zdrojová data a text diplomové práce v elektronické podobě.

Literatura

[MacQueen] *MacQueen, J. B.:* **"Some Methods for classification and Analysis of Multivariate Observations" In Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability** Berkeley, University of California Press, 1967, 1:281-297

[KOHO] *Kohonen, T.:* **Self-Organizating Maps** New York, Springer-Verlag, 1997