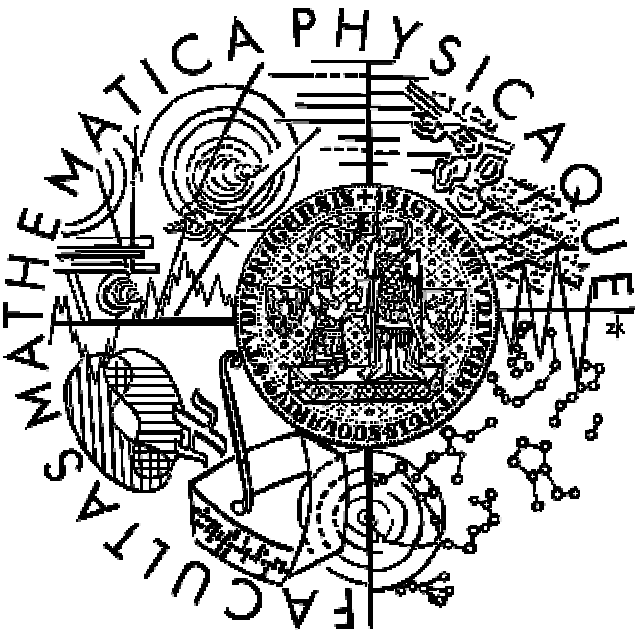
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Tomáš Dzetkulič

Využití klastrovacích technik při monitorování inzerce

Katedra aplikované matematiky

Vedoucí diplomové práce: Mgr. Petr Kolman, Ph.D.
Konzultant: Ing. Martin Cakl

Studijní program: Informatika
2007

Ďakujem svojmu vedúcemu Mgr. Petru Kolmanovi, Ph.D. za jeho podporu, rady a komentáre k práci. Ďakujem Ing. Martinu Caklovi za poskytnutie skúseností, rád a komentárov z praxe monitorovania inzercie a za poskytnutie reálnych dát na testovacie účely.

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

V Praze dne 10.8.2007 Tomáš Dzetkulič

Abstrakt:

Název práce: Využití klastrovacích technik při monitorování inzerce
Autor: Tomáš Dzetkulič
Katedra: Katedra aplikované matematiky
Vedoucí diplomové práce: Mgr. Pert Kolman, Ph.D.
e-mail vedoucího: kolman@kam.mff.cuni.cz
Abstrakt: Práca sa zaoberá možnosťami klastrovania inzercie so zameraním na realitnú inzerciu. V prvej časti práce definujeme čo to je klastrovanie, kde sa používa a aké sú typické požiadavky na klastrovacie algoritmy. Popíšeme existujúce klastrovacie metódy, ich vlastnosti a použitie. Posúdime ich vhodnosť pre oblasť inzercie a vyberieme najvhodnejší algoritmus pre klastrovanie rádovo miliónov inzerátov. V ďalšej časti detailne popíšeme interpretáciu inzerátu ako prvku vektorového priestoru s vysokou dimenziou a algoritmus klastrujúci prvky takéhoto vektorového priestoru založený na rodinách lokálnych hašovacích funkcií. Popíšeme jeho vlastnosti, časovú a pamäťovú zložitosť, jeho parametre a očakávané výsledky behu algoritmu. V implementačnej časti rozoberieme detaily implementácie v programovacom jazyku Java a navrhneme vhodné uloženie dát v relačnej databázi. V časti venovanej testom potom zhodnotíme výsledky behu algoritmu na reálnych dátach a porovnáme ich s očakávaným výstupom algoritmu. V závere práce posúdime možnosti ďalšieho rozšírenia použitej klastrovacej metódy.
Klíčová slova: klastrováni, inzerce, Java

Title: Clustering techniques for ads monitoring
Author: Tomáš Dzetkulič
Department: Department of applied mathematics
Supervisor: Mgr. Pert Kolman, Ph.D.
Supervisor's e-mail address: kolman@kam.mff.cuni.cz
Abstract: This thesis surveys possibilities of clustering of advertisements, especially those for real estates. It defines clustering itself, its usage and typical requirements for clustering algorithms. We provide list of existing clustering methods and approaches, their properties and suitable application. We consider possiblity of using them for clustering of milions of advertisements and based on that, we choose most suitable algorithm for this problem. We describe how to interpret advertisement as the point in multi dimensional vector space and this algorithm for clustering such points using locality of families of hash functions. We describe algorithm in detail, listing all of its parameters, estimating its complexity and expected results. In the following chapters we describe implementation of the algorithm in Java. We also describe database structure of underlying relational database. In the next chapter we present results of the algorithm based on real data and we compare the results with the expected results of the algorithm. In the end, we discuss possibilities for future extension of the clustering method.
Keywords: clustering, advertisement, Java

Obsah:

Zoznam obrázkov:

1. Úvod

V súčasnosti sa množstvo informácií v rôznych informačných systémoch rozrastá obrovským tempom. Na internete sa objavujú projekty digitalizácie dát zo všetkých oblastí a postupne si zvykáme na to, že namiesto knihy, alebo učebnice hľadáme informácie v digitálnej podobe. S rozrastajúcim sa množstvom informácií sa rozvíjajú i klastrovacie metódy. Klastrovanie je proces v ktorom sa objekty rozdelia na skupiny, pričom prvky jednej skupiny majú rovnaké, alebo aspoň podobné vlastnosti. Objektom klastrovania môže byť dokument, internetová stránka, vektor, inzerát, obrázok, alebo iný objekt. Rozvoj klastrovania je spôsobený hlavne potrebou triediť výsledky webových vyhľadávačov a iných aplikácií v ktorých sa ukladajú dokumenty. Cieľom týchto vyhľadávačov je poskytnúť čo najrelevantnejšiu množinu dokumentov k dotazu od užívateľa. Na internete sa pritom množstvo informácií nachádza v množstve kópií, alebo iba trochu pozmenené. Vyhľadávač nesmie zahltiť užívateľa množstvom dokumentov, ktoré všetky obsahujú rovnaké informácie. Snaží sa preto dokumenty klastrovať a poskytnúť užívateľovi pre každú skupinu podobných dokumentov jedného relevantného zástupcu. Ostatné prvky skupiny sú potom umiestnené medzi výsledkami nižšie, takže užívateľ na niekoľkých prvých miestach vidí dokumenty z rôznych oblastí.

1.1 Motivácia

Táto práca sa zaoberá klastrovaním inzerátov na nehnuteľnosti. Vychádza z prác, ktoré sa zaoberajú klastrovaním výsledkov vyhľadávania webových vyhľadávačov, pretože hlavný prúd výskumu sa uberá týmto smerom, vďaka obrovským ziskom spoločností, ktoré sa zaoberajú webovým vyhľadávaním.

Tak ako i v iných oblastiach, i v oblasti inzercie sa v súčasnosti môžeme stretnúť s inzerciou v digitálnej podobe na internete. Na internetových stránkach, ktoré sa inzerciou zaoberajú pribúdajú denne tisíce inzerátov na nehnuteľnosti. Tieto sú typicky triedené podľa lokality danej nehnuteľnosti, podľa jej ceny a iných vlastností ako napríklad počet izieb. Napriek tomu je však potrebné tieto inzeráty klastrovať,

pretože v praxi sa často stretávame so stále sa opakujúcimi inzerátmi na rovnakú nehnuteľnosť. Pri pohľade na bežnú internetovú stránku zaoberajúcu sa inzerciou máme zvyčajne veľmi malé možnosti vyhľadávania. Je typické, že inzeráty sú utriedené podľa času zadania. Často sa preto stáva, že jeden inzerát vidíme na každej stránke niekoľko krát. Ešte horšia je situácia v prípade, ak si prezeráme niekoľko rôznych stránok. Je typické, že rovnaký inzerát nájdeme na každej z nich. Táto situácia komplikuje vyhľadávanie relevantných nových inzerátov užívateľom aj realitným kanceláriám. Túto situáciu by pomohlo vyriešiť klastrovanie inzerátov.

1.2 Cieľ diplomovej práce

Cieľom diplomovej práce je navrhnúť a implementovať vhodný algoritmus na klastrovanie inzerátov. Algoritmus musí spĺňať niekoľko požiadaviek špecifických pre oblasť inzercie. Hlavnou požiadavkou je rýchle spracovanie nového inzerátu. V praxi sa denne stretávame s tisíckami inzerátov z ktorých menej ako desatina sú s ohľadom na historické dáta skutočne novými inzerátmi. Algoritmus by mal pre každý inzerát poskytnúť odpoveď na otázku, či sa jedná o nový inzerát, poprípade by mal poskytnúť zoznam podobných inzerátov.

1.3 Štruktúra diplomovej práce

V prvej časti diplomovej práce definujeme problém klastrovania, jeho využitia a algoritmov, ktoré sa pri klastrovní používajú. Na základe požiadaviek na klastrovanie inzerátov navrhneme vhodný algoritmus, popíšeme jeho teoretický základ a očakávané výsledky. V ďalšej časti popíšeme detaily implementácie, problémy na ktoré sme narazili počas implementácie a výsledky na skutočných dátach.

2. Klastrovanie

Klastrovanie je neriadený proces v ktorom sa objekty rozdelia na skupiny, pričom prvky jednej skupiny majú rovnaké, alebo aspoň podobné vlastnosti. Reprezentácia dát pomocou menšieho počtu klastrov stráca niektoré detaily uložených objektov, ale prináša zjednodušenie a štruktúrovanie objektov. Objektom klastrovania môže byť dokument, internetová stránka, vektor, inzerát, obrázok, alebo iný objekt. Objekt je zvyčajne reprezentovaný vektorom, alebo bodom v priestore s veľkou dimenziou. Objekty v jednej skupine, alebo klastri, by mali byť navzájom podobné a mali by byť odlišné od objektov v iných klastroch. Dôležitou vlastnosťou klastrovania je to, že ide o neriadený proces. Týmto sa líši od iných techník, ktoré majú dopredu definovaný počet tried objektov a snažia sa priradiť objekt do niektorej z tried. Klastrovanie naproti tomu vytvára skupiny autonómne. Často sa stáva, že samotné skupiny, teda klastre, sa menia i v priebehu výpočtu. Celý proces je riadený dátami na vstupe algoritmu.

2.1 Použitie klastrovania

Klastrovanie sa používa v mnohých oblastiach informačných technológií od získavania informácií (information retrieval), kategorizácii dát získaných webovým vyhľadávačom, kategorizácia obrázkov a rôznych vzorov, rozpoznávanie objektov a znakov a mnohých ďalších oblastiach.

2.1.1 Požiadavky na klastrovacie algoritmy

Od klastrovacieho algoritmu požadujeme hlavne rozdelenie dát do klastrov, tak, že podobné objekty sú priradené do rovnakého klastra a rozdielne objekty sa nachádzajú v rôznych klastroch. V niektorých prípadoch potrebujeme pre jednotlivé klastre priradiť nadpis, alebo označenie, ktoré reprezentuje dáta v danom klastri. Ďalej požadujeme, aby bol algoritmus škálovateľný i pre veľké množstvá dát. Veľkým množstvom sú v našom prípade desať tisíce až milióny inzerátov, ale pre použitie pri

klastrovaní dokumentov v knižniciach, alebo pri klastrovaní internetových stránok to môže byť ešte väčší počet klastrovaných objektov.

2.1.2 Požiadavky na klastrovanie inzerátov

Pri klastrovaní inzerátov ešte navyše požadujeme možnosť pridania nového inzerátu k už klastrovaným dátam. Výhodou pri inzerátoch je, že nemusíme jednotlivým klastrom priradzovať popis dát, pretože klaster môžeme reprezentovať jedným z jeho prvkov. Inzerát totiž zvyčajne pozostáva iba z malého počtu slov. Musíme predpokladať, že počet klastrov – jednotlivých nehnuteľností bude lineárne závislý na celkovom počte inzerátov, preto nemôžeme využiť algoritmy, ktoré predpokladajú malý počet klastrov, alebo veľké množstvo dát v klastri, ktoré by tento klaster detailne definovalo. Takisto nemôžme predpokladať, že sa skutočne v každom prípade podarí určiť, že ide o rovnakú nehnuteľnosť. Túto informáciu často nevie rozoznať ani človek, pretože typický inzerát obsahuje skutočne minimum informácií. Preto by sme okrem samotného klastrovania mali pre každú dvojicu inzerátov poskytnúť aj informáciu ako veľmi sú podobné. Pri výbere algoritmu zvolíme z tohoto dôvodu radšej fuzzy algoritmus. Fuzzy klastrovacie algoritmy sú skupinou algoritmov, ktoré vytvárajú prekrývajúce sa klastre, a určujú ako veľmi objekt prislúcha k jednotlivým klastrom v ktorých sa nachádza.

2.2 História klastrovania

Aj keď klastrovanie sa v súčasnosti používa primárne na kategorizáciu dokumentov a rozpoznávanie vzorov, má bohatú históriu v oblasti biológie, psychiatrie, archeológie, geológie, geografie a marketingu [1]. V minulosti sa poznatky v týchto odboroch museli kategorizovať, a klastrovanie poznatkov prebiehalo zvyčajne bez použitia počítačovej techniky. Klastrovaním môže byť označené i neriadené učenie, číselná taxonómia, alebo vektorová kvantizácia. Klastrovacie techniky boli podľa práce [26] použité na konštrukciu minimálnej kostry, alebo na segmentáciu obrázkov.

V poslednom desaťročí sa klastrovanie rozvíja vďaka potrebe klastrovať výsledky vyhľadávania internetových stránok.

2.3 Priebeh klastrovania - Predspracovanie

Klastrovanie objektov prebieha zvyčajne v niekoľkých krokoch. Na začiatku musíme každý objekt predspracovať, určiť jeho vlastnosti a tieto vlastnosti vhodne reprezentovať pre klastrovací algoritmus. Zvyčajne počas predspracovania vytvoríme z objektu vektor v n-dimenzionálnom vektorovom priestore na základe niektorého z dátových modelov. Ak sa jedná o dokument, inzerát, alebo webovú stránku, toto predspracovanie je prakticky nezávislé na voľbe algoritmu. Popíšeme si kroky predspracovania dokumentov v textovej podobe.

2.3.1 Získanie náhľadu

Ak je dokument rozsiahly, musíme ho popísať menšou množinou slov, slovných spojení a viet, pretože ak by sme pracovali s celým dokumentom, mohli by sme naraziť na problémy s výkonom celého algoritmu. Z dokumentu vytvoríme náhľad. Sem zvyčajne zaradíme nadpis dokumentu, tak ako to doporučuje práca [27], meno autora dokumentu, slová a slovné spojenia, ktoré sa v dokumente vyskytujú najčastejšie, ďalej sem patria časti dokumentu, ktoré sú v dokumente zvýraznene väčším, alebo tučným písmom, alebo podčiarknutím. Pri monitorovaní inzercie sa nestretávame s rozsiahlymi dátami, preto môžeme tento krok vynechať a ďalej pracovať s celým textom inzerátu

2.3.2 Filtering

Dokument zvyčajne okrem samotného textu obsahuje časti, ktoré nenesú žiadnu relevantnú informáciu. Napríklad v prípade dokumentu vo forme HTML sú to rôzne HTML tagy, na stránke sa môže vyskytovať reklama, alebo komentáre dokumentu od anonymných užívateľov uvedené pod dokumentom. Počas filtrovacej fázy by sme mali takéto informácie z dokumentu odstrániť.

2.3.3 Tokenizácia (Tokenisation)

Počas tokenizácie vytvoríme z textu získaného z predchádzajúcich dvoch krokov postupnosť tokenov. Token je informačnou jednotkou v texte. Môže to byť jedno slovo, číslo, skratka, alebo dátum. Pri vytváraní tokenov sa bežne používa takzvaný stop list, čo je zoznam slov, ktoré nenesú žiadnu informáciu, ale často sa vyskytujú v dokumentoch. Typicky ide o spojky, častice, citoslovcia, ale aj osobné zámená a podobne. Často sa pri tokenizácii textu úplne ignorujú čísla. Pri inzerátoch je situácia úplne odlišná a čísla v texte sú zvyčajne hlavnými rozpoznávacími znakmi medzi jednotlivými inzerátmi. Určujú koľko má byt izieb, aká je jeho plocha a aká je jeho cena. Spracovaniu čísel preto musíme venovať osobitú pozornosť.

2.3.4 Rozpoznanie jazyka

Veľmi dôležitou informáciou o texte je jazyk v ktorom je text napísaný. Pri klastrovaní inzercie budeme vždy pracovať s textom v češtine, ale pri iných aplikáciách klastrovania sa často stretávame s týmto problémom. V súčastnosti sa venuje tejto téme veľa pozornosti, pretože bráni najväčším svetovým vyhľadávačom ako google, alebo yahoo presadiť sa na špecifických trhoch v krajinách, kde sa gramatika výrazne líši od anglickej gramatiky. Jazyk textu môžeme určiť na základe početnosti písmen. Napríklad v angličtine, španielčine, nemčine a francúštine je najčastejším písmenom v texte písmeno ‘e’. Kým v angličtine je frekvencia tohto písmena 12,7%, v nemčine je to až 17.4% ako uvádza článok [21] a na základe tejto informácie by sme mohli automaticky rozoznať text v týchto dvoch jazykoch. Rozdiely by sme našli i pri ďalších písmenách a napríklad dva texty v angličtine a češtine by sme asi najjednoduchšie odlíšili podľa početností písmen ‘q’ a ‘w’. Ako ešte lepšia metóda sa ukázala metóda založená na početnosti dvojíc písmen. Táto metóda dokáže rozlíšiť jazyk textu už pri niekoľkých slovách. Najpresnejšia je však podľa článku [28] slovníková metóda, kde máme pre každý rozpoznávaný jazyk zoznam slov v tomto jazyku. Pre každý jazyk potom spočítame počet slov textu, ktoré patria do tohto jazyka a vyberieme jazyk s najväčším počtom.

2.3.5 Lemmatizácia

Ak poznáme jazyk v ktorom je text dokumentu napísaný, môžeme jednotlivé tokeny previesť na lemmy, teda previesť slovo jazyka na základný tvar. Predpokladáme pritom, že dva dokumenty z ktorých jeden obsahuje napríklad slovo "inzerátom" a druhý slovo "inzerátu" sú podobné i keď slová nájdené v textoch nie sú zhodné. Každé slovo preto prevedieme na jeho základný tvar pričom využívame gramatiku jazyka. Po prevedení na základný tvar potom môžeme jednotlivé slová porovnávať na úplnú zhodu. Ďalšou výhodou je to, že sa nám takto zmenší množina rôznych slov a to potom často vedie k rýchlejšiemu algoritmu. Pri lemmatizácii musíme využívať gramatiku jazyka, pretože rôzne tvary sa v jednotlivých jazykoch tvoria odlišne. Preto je dôležité správne identifikovať jazyk pred začatím lemmatizácie.

2.4 Modely reprezentácie dát

Po predspracovaní dokumentu máme pre každý dokument množinu tokenov, lemm, alebo vo všeobecnom prípade množinu vlastností, ktoré reprezentujú daný objekt. Tieto vlastnosti potom ďalej v algoritme reprezentujeme zvyčajne ako vektor vo vektorovom priestore s vysokou dimenziou. K tomu nám slúži jeden z týchto modelov:

2.4.1 Model vektorového priestoru

Pri snahe použiť na klastrovanie dokumentov rovnaké algoritmy ako pri klastrovaní numerických dát musíme dáta zo vstupu na začiatku interpretovať ako vektory z nejakého vektorového priestoru. Po prevedení dokumentov na tokeny, vytvoríme množinu všetkých tokenov. Nech má táto množina n prvkov. Potom môžeme definovať zobrazenie dokumentu do n-dimenzionálneho priestoru všetkých tokenov, tak, že:

$\mathbf{f(doc_j) = (w_{j,1}, w_{j,2}, \ldots, w_{j,n})}$

kde $w_{j,i}$ je váha i-tého tokenu pre dokument j. Váha tokenu v dokumente môže byť definovaná ako 0 ak sa token v dokumente nevyskytuje, alebo 1 ak sa v dokumente

nachádza. Ako váhu môžeme použiť i počet výskytov tokenu i v dokumente j. Často sa používajú normalizované váhy tokenov. Napríklad token, ktorý sa vyskytuje vo všetkých dokumentoch (napríklad predložky, spojky, zámená) majú priradenú malú, alebo dokonca nulovú váhu pretože nerozlišujú jednotlivé dokumenty medzi sebou a nemajú teda na výsledok klastrovania žiaden vplyv.

2.4.2 Model invertovaného indexu

V modeli vektorového priestoru sme dokumentom priradili vektory váh tokenov v danom dokumente. V modeli invertovaného indexu naopak každému tokenu priradíme vektor, ktorý obsahuje váhu daného tokenu vo všetkých dokumentoch. Ak je váha binárna, teda ak je to buď 0, alebo 1, podľa toho či je token relevantný pre daný dokument, potom získame pre každý token bitový vektor. Takáto štruktúra je vhodná ak v niektorom okamihu behu algoritmu chceme získať množinu dokumentov, ktoré spĺňajú nejaký logicý výrok. Ak by sme chceli napríklad dokumenty pre ktoré platí:

$r(t_1)$ OR ($r(t_2)$ AND NOT $r(t_3)$)

kde $r(t_i)$ znamená, že token t_i je relevantný pre dané dokumenty, potom nám stačí vybrať bitový vektor odpovedajúci tokenu t_3, znegovať ho, použiť operáciu AND na výsledok a vektor tokenu t_2 a na tento výsledok použiť operáciu OR s bitovým vektorom tokenu t_1. Takto získame bitový vektor a na miestach, kde sa nachádzajú jedničky sú dokumenty, ktoré zodpovedajú nášmu dotazu. Výhodou je to, že operácie na bitových vektoroch sú pre dnešné počítače najrýchlejšími operáciami a v jednom procesorovom cykle sa môže spracovať aj 128 bitov dlhý reťazec.

2.4.3 Modelovanie pomocou hammingovho priestoru

Ak je objekt popísaný množinou vlastností, pričom o každom objekte vieme povedať, či danú vlastnosť má, alebo nie, potom môžeme objekt reprezentovať ako n-bitový vektor, kde n je počet všetkých vlastností. Na pozícii i vektoru sa bude nachádzať 1 ak objekt obsahuje vlastnosť i a 0 v prípade, že objekt vlastnosť neobsahuje. V takomto prípade nie je vhodné medzi dvoma objektmi použiť ako mieru odlišnosti euklidovskú

vzdialenosť vo vektorovom priestore, pretože tá by bola rovná druhej odmocnine z počtu odlišných vlastností. Omnoho vhodnejší sa zdá byť samotný počet týchto odlišných vlastností. Takáto miera odlišnosti sa nazýva hammingova vzdialenosť vektorov a je popísaná v práci [29].

2.5 Miery podobnosti a rôznosti objektov

Keďže podobnosť objektov je základom pre definovanie klastrovania, je dôležité definovať čo to znamená, že dva objekty sú podobné. Výber metódy, ktorou sa bude určovať podobnosť dvoch objektov je základom úspechu algoritmu. K dispozícii máme pritom množstvo rôznych mier podobnosti, alebo nepodobnosti dvoch objektov. Ak sú objekty reprezentované vektormi, potom najjednoduchšou možnosťou voľby miery odlišnosti objektov je metrika Euklidovského priestoru:

$$d(x, y) = \sqrt{\sum_{i=0}^{n} (x_i - y_i)^2}$$

Predpokladáme, že čím sú objekty od seba ďalej, tým sú rôznejšie. Nevýhodou použitia metriky v Euklidovom priestore je prípad, keď jedna zložka vektoru dominuje nad ostatnými zložkami. Riešenie môže spočívať v tom, že už pri vytváraní vektorov priradíme jednotlivým vlastnostiam, alebo tokenom váhu tak, aby nám nevznikaly zložky, ktoré budú dominovať nad ostatnými. Ak napríklad klastrujeme dokumenty a v každom dokumente sa vyskytuje slovo ‘ja’, potom toto slovo neodlišuje dokumenty a bolo by dominantnou zložkou každého vektoru po prevedení dokumentov do modelu vektorového priestoru. Preto je vhodné ak tokenu ‘ja’ priradíme malú, alebo nulovú váhu.

Inou možnosťou je použiť ako mieru podobnosti kosínus medzi vektormi. Toto je vhodné v prípade, že nechceme, alebo nemôžeme normalizovať vektory, ale pritom vektory smerujúce v podobnom smere reprezentujú podobné objekty. Kosínus v takom prípade určíme zvyčajne z kosínusovej vety:

$$s(x, y) = \cos(x, y) = \frac{\sum_{i=0}^{n} x_i y_i}{\sqrt{\sum_{i=0}^{n} x_i^2}\sqrt{\sum_{i=0}^{n} y_i^2}}$$

Existujú i miery odlišnosti objektov pri ktorých záleží i na okolitých objektoch. Tieto objekty sú potom nazývané kontext. Podobnosť dvoch objektov je potom daná ako funkcia týchto objektov a množiny okolitých objektov:

$$s(x,y)=f(x,y,\Phi) ,$$

kde Φ je množina okolitých objektov. Príkladom na takúto mieru je napríklad vzájomná susedná vzdialenosť (mutual neighbor distance) popísaná v práci [3]:

$$MND(x,y)= NN(x,y) + NN(y,x) ,$$

kde NN(x,y) je poradie y medzi susedmi x. Predstavme si zoznam susedov bodu x utriedený napríklad podľa Euklidovskej vzdialenosti od bodu x. Ak sa bod y nachádza v tomto zozname na k-tom mieste, potom NN(x,y)=k. Nevýhodou tejto miery je, že táto miera nie je metrikou pretože nesplňuje trojuholníkovú nerovnosť. [4]

Ak objekt nie je reprezentovaný vektorom ale napríklad reťazcom bitov dĺžky n, potom môžme ako mieru podobnosti medzi dvoma reťazcami považovat počet pozícii na ktorých sa dané dva reťazce zhodujú. Rovnakú mieru môžme použiť i pre reťazce znakov. Ak však ide o text, potom táto miera nie je vhodná. Pri texte sa často definujú elementárne operácie, ktoré transformujú text a mierou odlišnosti dokumentov je potom počet operácií, ktoré musíme urobiť aby sme jeden dokument transformovali na druhý. Medzi elementárne operácie typicky patrí zmena jedného znaku za iný, pridanie znaku na ľubovoľné miesto v texte, zmazanie ľubovoľného znaku, prípadne ešte prehodenie dvoch po sebe idúcich znakov. Takáto miera odlišnosti dvoch reťazcov sa dá určiť pomocou dynamického programovania, pričom je veľmi vhodná pri klastrovaní textov v ktorých môžu byť preklepy.

Pre dokumenty môžeme zaviesť inú mieru podobnosti, založenú na tokenoch, ktoré sa v dokumentoch vyskytujú. Ak S(A) je množina tokenov relevantných pre dokument A, potom podobnosť dokumentov môžme definovať ako:

$$p(A, B) = \frac{|S(A) \cap S(B)|}{|S(A) \cup S(B)|}$$

2.6 Klastrovacie metódy

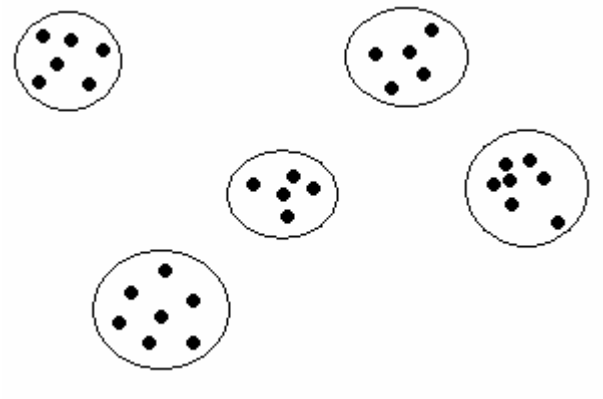
V súčasnosti existuje veľké množstvo rôznych klastrovacích techník a ich variácií na základe toho akú použijeme mieru podobnosti objektov, prípadne akým modelom reprezentujeme objekty. V tejto časti práce popíšeme niektoré z nich.

2.6.1 Prekrývajúce sa klastre

Klastrovacie metódy môžeme rozdeliť na dva druhy podľa toho, či povoľujú, aby sa jeden objekt nachádzal v niekoľkých rôznych klastroch, alebo nie. Algoritmy, ktoré prekrývanie nepovoľujú priradia každý objekt do práve jedného klastra. Naopak algoritmy, ktoré počítajú s prekrývajúcimi sa klastrami (fuzzy algoritmy) môžu priradiť objekt do viacerých klastrov, prípadne môžu poskytnúť maticu príslušnosti ($u_{i,j}$), kde prvok matice je reálne číslo z intervalu [0,1] určuje ako veľmi i-tý objekt patrí do j-tého klastra.

2.6.2 Klastrovanie numericých dát

Prvé klastrovacie algoritmy vznikli pri potrebe klastrovať body vo vektorovom priestore. Jedným z prvých takýchto algoritmov je **k-means algoritmus,** popísaný v práci [31], ktorý funguje v iteráciach. Vstupom algoritmu je číslo k, čo je počet požadovaných klastrov. Na začiatku sa v priestore vygeneruje k náhodných bodov, ktoré sa stanú stredmi klastrov. Tento stred nazývame centroidom. Každý objekt je priradený k najbližšiemu stredu. Následne sú prepočítané stredy klastrov tak, aby odpovedali ťažisku bodov priradených tomuto klastru a pokračuje sa ďalšou iteráciou. Tento algoritmus je veľmi vhodný ak dáta na vstupe skutočne tvoria skupiny okolo nejakých centroidov, teda pre prípad aký vidíme na obrázku 1. Ak majú klastre nepravidelný tvar, alebo ak nepredpokladáme, že dáta budú mať takéto vlastnosti, potom algoritmus nie je vhodné použiť.



Obrázok 1

Pri klastrovaní objektov iného ako numerického typu je možné použiť rovnaký algoritmus, vstupné data však musíme vedieť reprezentovať ako vektory. K tomu nám pomáha práve už popísaný model vektorového priestoru.

2.6.3 Hierarchické klastrovanie

Hierarchické klastrovanie patrí tiež medzi najstaršie klastrovacie metódy a stále je veľmi rozšírené a používa sa ako súčasť sofistikovanejších klastrovacích metód. Hierarchické klastrovanie delíme podľa toho či sa klastre vyrábajú zhora, alebo zdola na zhromažďujúce (agglomerative), alebo deliace (divisive). Pri aglomeratívnom klastrovaní začíname s množinou klastrov rovnou množine objektov. Potom postupne spájame menšie klastre do vačších. Naopak pri deliacom hierarchickom klastrovaní začíname s jedným veľkým klastrom, obsahujúcim všetky objekty a postupne ho delíme na menšie klastre. Výhodou hierarchického klastrovania je to, že nám na konci vznikne strom všetkých klastrov, takže užívateľovi môžeme poskytnúť zoznam všetkých objektov v najväčšom klastri a on postupne môže vyhľadávať tak, že si stále vyberie ten subklaster, ktorý ho najviac zaujíma. Nevýhodou je to, že dáta sú v klastroch priradené staticky a ak sa algoritmus v jednom okamihu rozhodne pre nesprávne delenie/zlúčenie klastrov, potom sa táto chyba už ďalej v priebehu algoritmu nenapraví. Takisto nie je možné jednoducho pridať nový objekt, preto sa táto metóda nehodí na klastrovanie inzercie.

Väčšina algoritmov, ktoré využívajú hierarchické klastrovanie používa metódu single link popísanú v práci [5], alebo complete link popísanú v práci [6]. Pri metóde single

link je vzdialenosť medzi dvoma klastrami určená ako minimum vzdialeností medzi dvojicami objektov, pričom prvý objekt je prvkom prvého klastra a druhý objekt je z druhého klastra. Pri complete link sa využíva maximum zo vzdialeností. V obidvoch prípadoch ide o aglomeratívne klastrovanie, takže dva klastre s najmenšou hodnotou vzdialenosti sú zlúčené do nového klastra. Nevýhodou single link je takzvaný reťazový efekt. Ak máme objekty, ktoré sú rozložené rovnomerne na jednej priamke vo vektorovom priestore, potom každá dvojica po sebe idúcich objektov je podobná, ale objekty, ktoré sú na opačných stranách takto vzniknutej reťaze nemusia mať nič spoločné. Algoritmus založený na single link vzdialenosti však takéto objekty zaradí do jedného klastra.

2.6.4 Problém ukončenia klastrovania

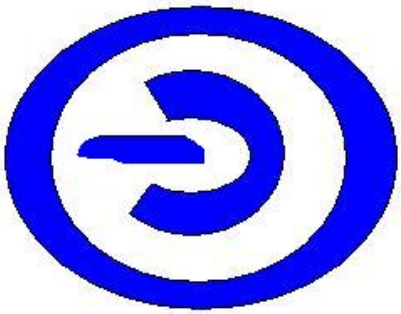
Problém, ktorý sa spája s hierarchickým klastrovaním, ale i s ďalšími metódami spočíva v tom, pri akom počte klastrov máme zastaviť beh algoritmu. Problémom sa zaoberá [7] a navrhuje niekoľko techník na riešenie tohto problému. Najčastejšie používaným kritériom na vhodné klastrovanie je kvadratická chyba klastrovania:

$$e^2 = \sum_{j=1}^{N} \sum_{i=1}^{n_j} d\,(x_{i,j},\, c_j)^2$$

kde N je počet klastrov, n_j je počet objektov v j-tom klastri a d(x,y) je euklidovská vzdialenosť objektov. Pre rôzny počet existujúcich klastrov môžeme vyhodnotiť hodnotu tejto chyby a určíme ešte priemernú vzdialenosť medzi stredmi dvoch klastrov s. Potom podiel s/e udáva separáciu klastrov a je vhodné aby bola táto hodnota čo najväčšia. Ak teda algoritmus počas svojho behu generuje rôzny počet klastrov, potom ako výsledok klastrovania označíme ten výsledok, pre ktorý je podiel maximálny. Túto chybu môžme použiť i s k-means klastrovaním, pričom algoritmus niekoľko krát reštartujeme s rôznym počtom požadovaných klastrov a pri voľbe výsledku sa rozhodujeme pre výsledok jedného z behov algoritmu analogicky ako pri hierarchickom klastrovaní.

2.6.5 Metódy založené na hustote pokrytia priestoru

Problémom k-means metód klastrovania je to, že neposkytujú riešenie pre prípad v ktorom sa podobné objekty vo vektorom priestore vyskytujú v nepravidelných tvaroch, napríklad sú na niekoľkých rovnobežných priamkach, alebo pre príklad na obrázku 2. Takýto problém sa snažia riešiť metódy založené na hustote pokrytia priestoru objektami. Metódy predpokladajú, že kým je hustota objektov v nejakej spojitej podmnožine priestoru (hoci nepravidelnej) vysoká, potom objekty patria do jedného klastra. Naopak, miesta s nízkou hustotou objektov sú hranicami medzi klastrami. Na obrázku 2 vidíme 3 oblasti pokryté objektami. Pri klastrovaní takýchto dát je vhodné použiť práve algoritmy založené na hustote pokrytia.



Obrázok 2

Problémom týchto algoritmov je zvyčajne to, že kritická hustota, pri ktorej sa vytvára hranica medzi klastrami musí byť zvyčajne dodaná do algoritmu zvonka ako parameter algoritmu.

Príkladom takéhoto algoritmu je algoritmus DBSCAN navrhnutý a popísaný v [8]. Algoritmus slúži na nájdenie klastrov v priestorových databázach, pričom je navrhnutý tak aby si dokázal poradiť so šumom vo vstupných dátach. Algoritmus má na vstupe dva parametre: $\in$ a k, kde epsilon je maximálna vzdialenosť medzi bodmi v ktorej hľadáme susedov bodu a k je minimálny počet bodov v epsilonovom okolí bodu. Potom definujeme priamu hustotovú dostupnosť bodu x od bodu y ak sa v epsilonovom okolí bodu x nachádza aspon k bodov a bod y je medzi nimi. Body x a y sú dostupné nepriamo, ak existuje postupnosť bodov $x_1,...x_n$, tak že z x je priamo dostupný bod x_1, z bodu x_i je priamo dostupný bod x_{i+1} a z bodu x_n je priamo dostupný

bod y. Dva body x a y sa budú nachádzať v jednom klastri ak existuje tretí bod z z ktorého sú obidva body dostupné.
Ďalšími algoritmami založenými na tomto prístupe sú algoritmy OPTICS [9], ktorý je rozšírením DBSCANu a algoritmus CLIQUE [10], ktorý je založený na mriežkach vo viacrozmernom priestore.

2.6.6 Suffix tree clustering

Suffix Tree Clustering je metóda na klastrovanie dokumentov, ktorá zhlukuje dokumenty do klastrov podľa toho, koľko obsahujú spoločných slovných spojení-fráz. Predpokladáme, že samostatné slová nemajú takú veľkú hodnotu ako frázy. Pri suffix tree klastrovaní sa snažíme o zachovanie informácií o poradí slov v dokumente. Táto metóda je popísaná v práci [23] a pozostáva z dvoch fází. Počas prvej fázy vytvoríme suffixový strom, ktorý obsahuje všetky vety z klastrovaných dokumentov. Uzlami tohto stromu sú jednotlivé slová. Keď máme vytvorený tento strom, každý uzol predstavuje frázu a obsahuje informáciu o tom v ktorých dokumentoch sa táto fráza vyskytuje. Frázy, ktoré sú dostatočne dlhé, alebo dostatočne časté v jednotlivých dokumentoch sa ponechajú a ostatné frázy sa z klastrovania vyradia. Tento krok je podľa článku [24] najväčším problémom celého algoritmu, pretože nie je jednoduché nastaviť správne hodnotiacu funkciu podľa ktorej budeme určovať frázy, ktoré ponecháme. Často sa nám môže stať, že ponecháme frázy, ktoré popisujú málo dokumentov, alebo naopak krátke frázy, ktoré sa vyskytujú vo všetkých dokumentoch. V druhej fázi algoritmu potom na základe informácií o frázach a dokumentoch, ktoré v nich patria zhlukujeme jednotlivé frázy, ktoré sa vyskytujú v rovnakých dokumentoch, aby sme vytvorili výsledné klastre. Výhodou tohto algoritmu je, že zachováva informáciu o poradí slov vo frázach a že frázy môžeme nesôr použiť na popis daného klastra. Nevýhodou potom je, že nie je jednoduché nastaviť správne parametre algoritmu a v článku [25] sa ukazuje, že často nenájdeme dokumenty, ktoré neobsahujú žiadnu z relevantných frází, no aj tak môžu byť relevantné.

2.6.7 Klastrovanie za pomoci zaokrúhľovacích algoritmov

Tieto metódy slúžia na klastrovanie objektov, ktorých vlastnosti sú reprezentované bitovým vektorom konštantnej dĺžky. Podobnosť objektov je definovaná ako počet miest na ktorých sa dva bitové vektory zhodujú. V [11] je popísaná dátová štruktúra, pre aproximatívny algoritmus na hľadanie najbližšieho suseda. Tento algoritmus môže byť ďalej rozšírený na klastrovanie týchto prvkov. Popísaný algoritmus redukuje problém na inštanciu problému PLEB (Point Location in Equal Balls). Pre každé ϵ potom existuje algoritmus, ktorý rieši ϵ–PLEB v čase $O(n^{1/(1+\epsilon)})$. Táto práca bola rozšírená v [12], kde za pomoci jednoduchšieho algoritmu získame rovnaký výsledok. Tento algoritmus si teraz popíšeme: Majme množinu bitových vektorov, ktorý každý pozostáva z d bitov. Zvolíme presnosť výpočtu ϵ, a na základe ϵ určíme $N = O(n^{1/(1+\epsilon)})$ náhodných permutácií bitov. Pre každú permutáciu p vytvoríme utriedený zoznam O_p všetkých bitových vektorov zo vstupu po aplikovaní permutácie p. Pre bitový vektor q, pre ktorý chceme určiť najbližšieho suseda postupujeme takto: Pre každú permutáciu p nájdeme pomocou binárneho vyhľadávania najbližší lexikograficky väčší a najbližší menší bitový vektor k vektoru q v O_p. Z 2N takto získaných vektorov vrátime ten ktorý má najmenšiu hammingovu vzdialenosť od q. Výhodou tohto algoritmu oproti PLEB je, že pre každý hľadaný vektor q môžeme použiť rovnakú množinu permutácií. Pri použití PLEB by sme museli skonštruovať vždy novú inštanciu tohto problému.

2.6.8 Iné klastrovacie metódy

Sú známe i klastrovacie algoritmy založené na evolučných metódach ako genetické algoritmy, simulované ochladzovanie (simulated annealing), neurónových sieťach, alebo klastrovanie založené na zadaných podmienkach (constraint based). Tieto metódy sú vhodné na veľmi špecifické použitie a často majú vyššiu časovú zložitosť ako metódy detailnejšie popísané v tejto práci. Na klastrovanie inzercie nie sú teda vhodné.

2.7 Klastrovanie výsledkov webového vyhľadávania

Pri klastrovaní výsledkov webového vyhľadávania máme na vstupe rádovo stovky dokumentov, v ktorých sa full-textovým vyhľadávaním našli slová, alebo slovné spojenia ktoré boli predmetom hľadania. Tieto stránky môžu tvoriť až desiatky tém a pri klastrovaní týchto výsledkov by sme chceli tieto témy odhaliť a užívateľovi dať k dispozícii zoznam tém z ktorého by si mohol vybrať tú tému, ktorá ho najviac zaujíma. Každá téma by mala obsahovať niekoľko dokumentov. Tém by nemalo byť príliš veľa, aby si užívateľ mohol ľahko a rýchlo vybrať.

2.8 Klastrovanie pri monitorovaní inzercie

Pri monitorovaní inzercie sa stretávame s množstvom rozdielov oproti typickému použitiu klastrovania – klastrovaniu výsledkov webového vyhľadávania. Od algoritmu požadujeme hlavne schopnosť zobraziť klaster pre zvolený inzerát a potom odhaliť, či ide o nový, alebo už zadaný inzerát s ohľadom na existujúcu databázu historických inzerátov. Musíme vychádzať z praktických požiadaviek, ktoré sú:

- počet klastrovaných inzerátov je relatívne vysoký a denne pribudne približne 5000 až 10000 nových inzerátov
- musíme predpokladať, že výsledný počet klastrov bude lineárne závislý na celkovom počte inzerátov
- inzerát obsahuje malý počet slov, aby bola dĺžka inzerátu krátka
- požadujeme, aby sme o každom novom inzeráte dokázali rozhodnúť, či ide o novú nehnuteľnosť na ktorú ešte neexistuje v databázi inzerát
- požadujeme aby sme vedeli efektívne k inzerátu zobraziť zoznam podobných inzerátov. Tento zoznam by mal byť utriedený od najpodobnejších po menej podobné inzeráty.

2.8.1 Hlavné nevýhody popísaných klastrovacích metód

Medzi hlavné nevýhody využitia popísaných metód pri klastrovaní inzercie patrí:

- algoritmy predpokladajú malý počet pomenovaných klastrov, tento počet je často parametrom algoritmu. Pri inzerátoch tento predpoklad nemôžeme uplatniť.

- algoritmy predpokladajú, že objekty tvoria v priestore klastre s vyššou hustou v okolí stredu klastra a s menšou hustotou záznamov v oblasti medzi klastrami. Na základe tohto predpokladu sú potom nastavené ukončovacie podmienky (napríklad kvadratická chyba klastrovania a hodnota separácie klastrov). Tento prístup je i základom klastrovacích techník založených na hustote pokrytia priestoru. Pri monitorovaní inzercie tiež nemôžeme predpokladať, že inzeráty budú vo vektorovom priestore nejako separované.
- algoritmy často nepodporujú jednoduché pridanie nových objektov do klastrovania a celý algoritmus je nutné reštartovať pre novú množinu vstupných dát. Naopak pri inzercii musíme predpokladať, že inzeráty pribúdajú priebežne a celkový počet inzerátov je rádovo niekoľko miliónov.

Celkovo sa teda klastrovanie inzerátov podstatne odlišuje od ostatných aplikácií klastrovania, hlavne od klastrovania webových stránok.

2.8.2 Zhrnutie rozdielov klastrovania inzercie a webových stránok

Základným rozdielom pri inzercii oproti klastrovaniu webových stránok je práve počet očakávaných klastrov. Pri klastrovaní stránok predpokladáme istý obmedzený počet tém, ktorých počet sa nezvyšuje a nové stránky sa venujú téme o ktorej už existujú iné stránky. Pri klastrovaní webových stránok by mal výsledok klastrovania obsahovať maximálne desiatky klastrov, aby si užívateľ vedel vybrať klaster, ktorý ho najviac zaujíma.

Naopak, pri klastrovaní inzerátov je počet klastrov lineárne závislý na počte inzerátov. Vzniknuté klastre sa prekrívajú, pretože často ani človek nedokáže rozhodnúť o dvoch inzerátoch, či ide o rovnakú nehnuteľnosť. Podobnosť treba definovať ako funkciu pre každú dvojicu inzerátov. Každý inzerát potom predstavuje klaster a inzeráty, ktoré sú podobnejšie ako zvolená hraničná hodnota podobnostnej funkcie budú patriť do klastra s týmto inzerátom.

3. Existujúce aplikácie používajúce klastrovanie

Autorovi práce nie sú známe žiadne práce, alebo projekty, ktoré by sa špecializovali na klastrovanie inzercie. Ako podobný projekt môžeme označiť prácu [13], ktorá sa zaoberá klastrovaním krátkych textov. Pri klastrovaní inzercie sa stretávame s problémom, že klastrovaný text má maximálne desiatky slov a to robí klastrovanie zložitejším. V spomínanej práci je navrhnutý a popísaný spôsob ako rozšíriť informáciu na vstupe do algoritmu za pomoci ďalšieho zdroja informácií. Ako zdroj je pritom použitá online encyklopédia Wikipedia. Pre krátky text, ktorý chceme klastrovať sa jednotlivé tokeny textu hľadajú v texte článkov na Wikipédii a nájdené články sa potom použijú na rozšírenie informácie obsiahnutej v primárnom texte. Ak napríklad krátky text obsahuje podreťazec "stredný východ", potom výsledkom hľadania na wikipédii môže byť zoznam krajín, ktoré k strednému východu patria. Ak potom ďalší krátky text obsahuje priamo mená niektorých štátov, potom po použití tejto metódy novo získané informácie pomôžu priradiť takto tieto dva objekty do rovnakého klastra.

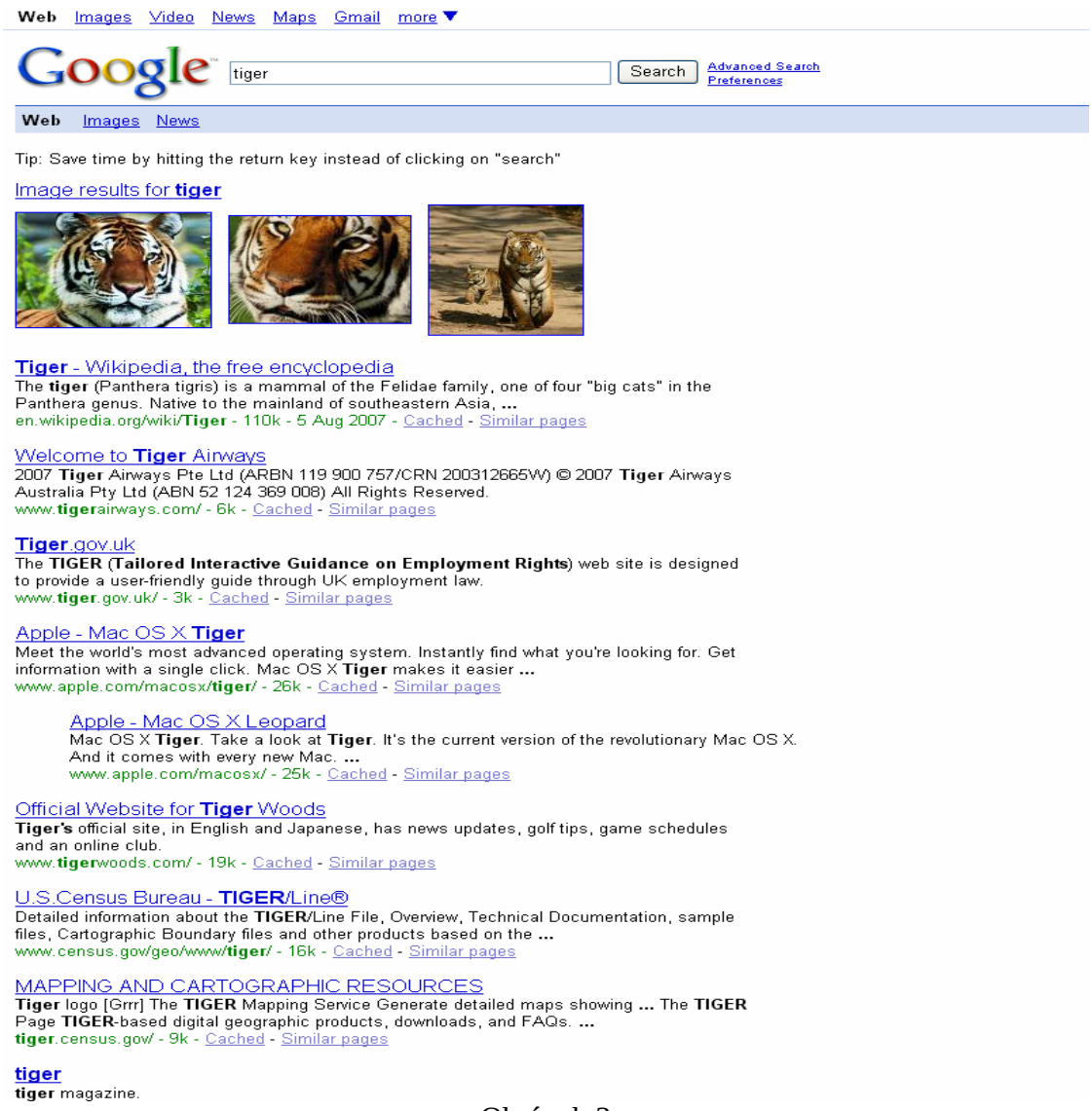
Podobná technika by sa dala použiť i pri monitorovaní inzercie, museli by sme však mať špecifický zdroj informácií zameraný na prostredie, ktorého sa inzercia týka. Výhodou by bolo ak by sme vedeli interpretovať skratky, ktoré sa často vyskytujú v inzercii, aby bol inzerát čo najkratší. Podobne by bolo dobré vedieť k ulici, alebo adrese priradiť mestskú časť, alebo kraj v ktorom sa daná oblasť nachádza. Takýto zdroj informácií sme však nemali k dispozícii.

Keďže neexistuje veľké množstvo podobných projektov, popíšeme si niektoré projekty, ktoré sa zaoberajú najbežnejším použitím klastrovacích metód, teda klastrovaním výsledkov hľadania dokumentov a internetových stránok.

3.1 Google

Keď hovoríme o hľadaní dokumentov a internetových stránok, nemôžme nespomenúť internetový vyhľadávač Google. I keď Google nie je nástrojom špecificky určeným na klastrovanie objektov, je zrejmé, že klastrovanie vo veľkej miere využíva k poskytnutiu presnejšieho výsledku hľadania. Klastrovacie si môžeme predviesť

zadaním hľadaného reťazca “tiger” do vyhľadávača. Výsledok dotazu vidíme na Obrázku 3.



Obrázok 3

Prvá stránka s výsledkami hľadania neobsahuje žiadne dva odkazy, ktoré by reprezentovali ten istý význam slova tiger. Vidíme tu stránku o tigrovi ako zvierati, stránku Tiger airlines a Tigera Woodsa, ako aj odkaz na operačný systém Tiger od firmy Apple. Google ako jeden z najväčších spoločností zaoberajúcich sa vyhľadávaním stále vylepšuje presnosť a rôznorodosť poskytnutých informácií aj za pomoci klastrovania. Na internetovej stránke [35] je popísaná prednáška predstaviteľa

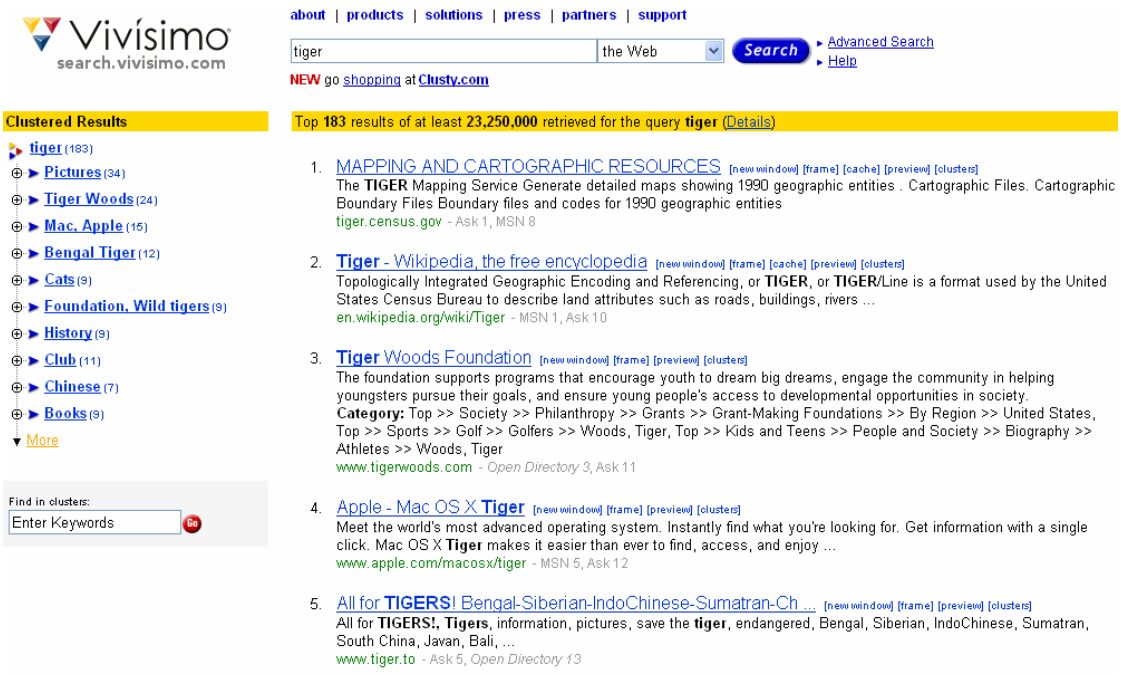
spoločnosti Google, ktorá sa zaoberá tým, ako sa Google snaží poskytnúť lepšie výsledky wehľadávania webových stránok. Google sa snaží porozumieť významu hľadaného reťazca. K tomu mu slúži:

- štatistický strojový preklad
- pomenované entity
- klastrovanie slov

Pomocou štatistického strojového prekladu sa spoločnosť snaží porozumieť sémantike a syntaxi rôznych jazykov, vrátane strojových jazykov. Pomocou pomenovaných entít sa snaží zistiť, ktoré výsledky full-textového hľadania síce obsahujú všetky hľadané slová, tieto však nie sú v spojitosti. Ako tretiu skúmanú oblasť uviedol predstaviteľ spoločnosti práve klastrovanie. Problémom podľa Google je, že zadané slová môžu mať rôzne významy a stránky zobrazené medzi prvými nájdenými nemusia reprezentovať ten význam na ktorý sme mysleli. Google preto pracuje na databázi slovných klastrov, ktorá by mala odhaliť najpravdepodobnejší význam slova ktoré sme chcli vyhľadávať. Detailnejší popis toho, ako klastrovanie v Google funguje však článok neobsahuje.

3.2 Vivísimo

Vivísimo je ďalším komerčným projektom. Algoritmus nie je známy, ale mal by byť založený na label based algoritmoch. Pri vytváraní klastrov najskôr vygenerujeme popisy (label), ktoré pokrývajú nájdené dokumenty a dokumenty sú potom zaradené pod jednotlivé nadpisy. Výsledok hľadania rovnakého dotazu ako pri vyhľadávači google vidíme na obrázku 4.

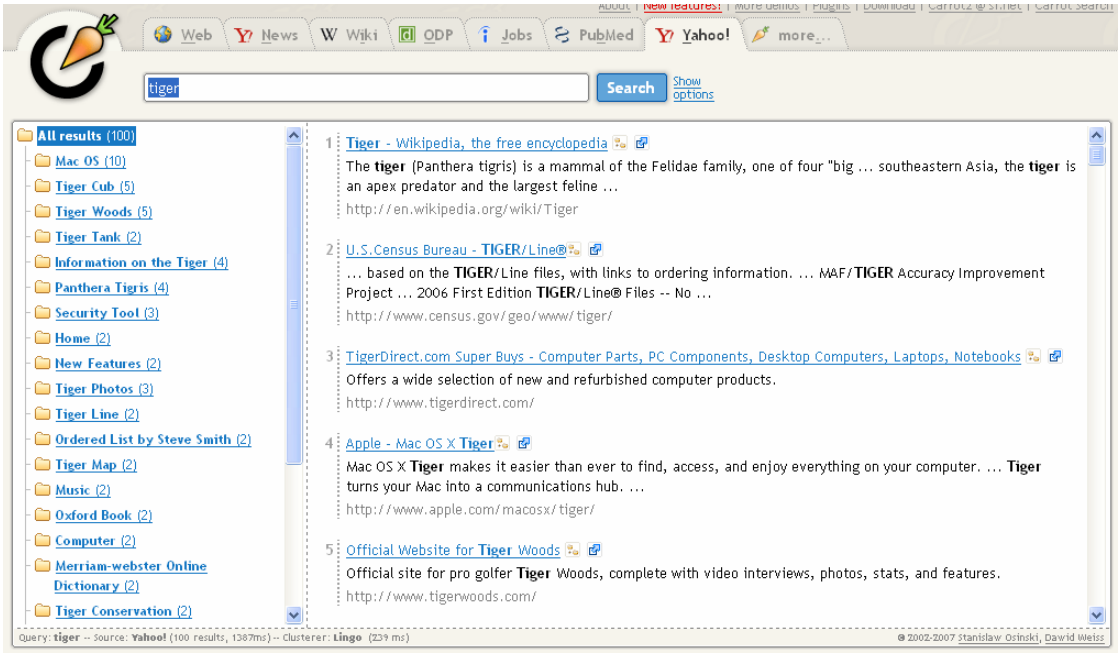


Obrázok 4

Vivísimo vytvára klastre tak, aby každý z nich obsahoval dostatočné množstvo odkazov, ktoré špecifikujú jednotlivé klastre. Popisy, alebo labely klustrov však nie sú určené úplne najvhodnejšie. Vidíme samostatnú skupinu o knihách, samostatná skupina pre obrázky, alebo pre históriu, pričom nie je špecifikované o aký význam hľadaného slova sa jedná. Takisto skupina "Cats" by mohla byť zlúčená s niektorou inou skupinou. Na pravej strane medzi samotnými výsledkami hľadania vidíme hneď na prvom mieste odkaz, ktorý sa nezdá byť relevantným. Celkovo teda Vivísimo neposkytlo také dobré výsledky ako Google.

3.3 Carrot²

Výsledky pre rovnaký hľadaný reťazec v systéme Carrot² vidíme na obrázku 5. Carrot² je rozšírením klastrovacieho frameworku Carrot. Je to open source projekt, ktorý je popísaný na svojich webových stránkach [33]. Carrot² je teda framework, ktorý obsahuje dva klastrovacie algorimy: Suffix tree clustering Lingo a Tollerance Rough Set Clustering popísaný v práci [33].



Obrázok 5

Carrot[2] generuje viac klastrov ako systém Vivísimo, ale niektoré z týchto klastrov potom obsahujú malý počet odkazov. Takéto klastre sú potom často charakterizované popisom u ktorého nevidíme spojitosť s pôvodne hľadaným reťazcom. Pozitívom algoritmu je naopak to, že na niekoľkých prvých pozíciách vidíme skutočne dobre formované a dobre zaplnené klastre.

3.4 Lingo

Lingo je suffix tree fuzzy klastrovací algoritmus, popísaný v práci [14]. Autor algoritmu tvrdí, že dôležitejšie pre dobre formované klastre nie je ani tak správne priradenie objektov do klastrov, ale omnoho dôležitejšie je, aby klaster popisovala fráza, ktorá bude zrozumiteľná človeku, teda nie len zhluk často vyskytujúcich sa tokenov. Pracuje v niekoľkých fázach. V prvej fáze sa algoritmus snaží nájsť množinu frází vo vstupných dokumentoch, ktoré by čo najlepšie pokrývali množinu dokumentov. Tieto fráze sú potom použité ako popis (label) jednotlivých klastrov. Tento popis je v systéme Lingo nazývaný abstraktným konceptom klastra. Lingo používa model vektorového priestoru, ale i niektoré netriviálne prístupy ako Singular Value Decomposition matice príslušnosti termu k dokumentu a metódu redukcie

dimenzie vektorového priestoru. Lingo vytvára veľmi dobré abstraktné koncepty pre jednotlivé klastre, ale tieto môžu byť v niektorých prípadoch veľmi špecifické a preto sa potom pre niektoré z klastrov podarí priradiť iba málo dokumentov.

3.5 Lucene

Lucene je projekt zameraný na fultextové vyhľadávanie v dokumentoch. Je popísané v článku [22]. Ide o skutočne rýchly full-textový vyhľadávač naprogramovaný v Jave, ktorý je navyše dostupný ako open-source. Jeho použitiu na vyhľadávanie podobných dokumentov pri monitoringu inzercie však bráni to, že Lucene neposkytuje toľko nástrojov na spravovanie dokumentov ako relačná databáza. Ak máme inzeráty uložené v relačnej databáze, museli by sme mať ich redundantnú kópiu ešte v Lucene.

3.6 Egocentric

Egocentric je projekt, ktorý vznikol na Karlovej univerzite a zaoberá sa takisto klastrovaním výsledkov hľadania na webe. Je určený pre vyhľadávač Egothor. Egocentric je klastrovacím frameworkom podobne ako Carrot^2 a implementuje niektoré známe klastrovacie algoritmy ako k-means, alebo jeden z label based algoritmov, teda algoritmov, ktoré najskôr vytvoria popisy klastrov a potom samotné klastre. Tento framework je popísaný v práci [36].

4.Návrh metódy na klastrovanie inzercie

Pri návrhu sme vychádzali z predpokladov uvedených v druhej kapitole. Na základe nich sme zvolili algoritmus, ktorý popíšeme v tejto časti práce.

4.1 Výber algoritmu

Keďže predpokladáme vysoký počet objektov ktoré musíme klastrovať ako i vysoký počet klastrov, musíme zvoliť algoritmus, ktorý nie je kvadraticky závislý na počte inzerátov. Algoritmus musí pracovať v jednom, alebo niekoľkých prechodoch vstupných dát. S ohľadom na ďalšie požiadavky, ako potreba priebežne pridávať nové inzeráty a potreba určiť podobnosť dvoch vybraných inzerátov sme zvolili špecifický algoritmus založený na práci [12]. Táto práca sa venuje hľadaniu najbližšieho suseda a určovaním miery podobnosti všeobecných objektov. Hľadanie najbližšieho suseda potom môžeme rozšíriť na klastrovací algoritmus, ak predpokladáme malú veľkosť klastra a nájdeme niekoľko najbližších susedov. Tento algoritmus si teraz bližšie popíšeme.

4.2 Popis metódy hľadania podobných objektov

V práci [12] je popísaná metóda hľadania podobných objektov. Majme podobnostnú funkciu sim(x,y), ktorá priradzuje objektom hodnotu z intervalu [0,1], kde 0 znamená že objekty sú úplne odlišné a 1 znamená, že objekty sú úplne rovnaké. Predpokladáme, že k tejto podobnosstnej funkcii existuje rodina lokálnych hašovacích funkcií F, ktorá podobným objektom priradzuje s určitou pravdepodobnosťou rovnaké hodnoty hašovacích funkcií. Túto vlastnosť popisuje podmienka:

$$\mathbf{P_{h \in F}(h(x)=h(y)) = sim(x,y)\ ,}$$

kde F je rodina hašovacích funkcií, h je jedna z týchto funkcií, sim je zvolená podobnostná funkcia a x a y sú dva objekty, ktoré chceme porovnať. $P_{h \in F}$ je pravdepodobnosť cez všetky funkcie h z F. V práci [12] sa ďalej nachádza dôkaz existencie takýchto rodín hašovacích funkcií a spôsob ich využitia pre hľadanie najbližšieho suseda.

Ak by sme mali takúto rodinu hašovacích funkcií F, ktorá splňuje rovnosť uvedenú vyššie, potom môžeme vybrať niekoľko zástupcov tejto rodiny (h_1,h_2,...h_n). Pre každý objekt a každú hašovaciu funkciu potom určime hodnoty $h_i(x)$. Pre každé dva objekty označme p(x,y) počet indexov i, $0 < i <= n$, pre ktoré platí $h_i(x)=h_i(y)$. Ak boli hašovacie funkcie vybrané náhodne, potom môžeme funkciu sim(x,y) aproximovať hodnotou p(x,y)/n, čo je vlastne hammingova vzdialenosť medzi vektormi ($h_1(x)$,$h_2(x)$,...$h_n(x)$) a ($h_1(y)$,$h_2(y)$,...$h_n(y)$) vydelená počtom zvolených funkcií. Hľadanie najbližšieho suseda potom môžeme založiť na hľadaní v hammingovom priestore popísanom v tejto práci v časti 2.6.7. V ďalšej časti tejto kapitoly si popíšeme ako nájsť takéto rodiny hašovacích funkcií a ako rozšíriť hľadanie najbližšieho suseda na klastrovací algoritmus.

4.3 Rozšírenie algoritmu pre použitie v klastrovaní

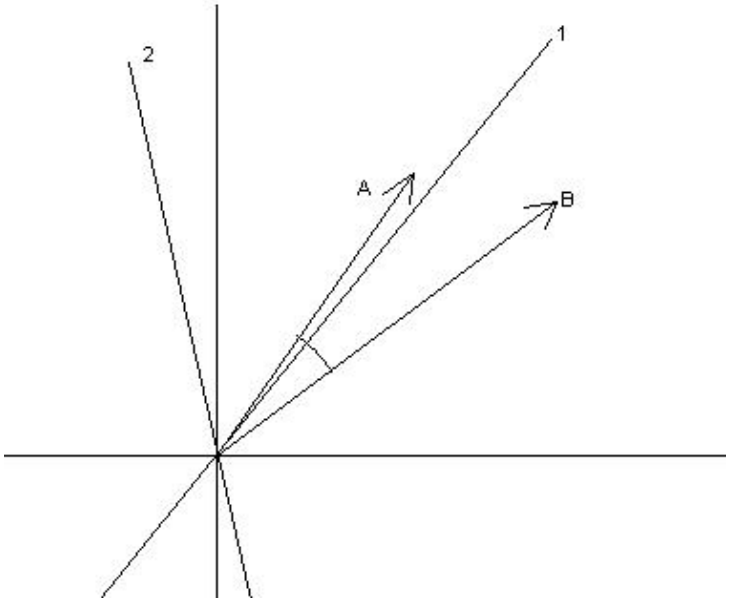
Vychádzame z predpokladu, že počet klastrov je lineárne závislý na počte objektov klastrovania a že počet objektov v jednom klastri je relatívne malý. Ak vieme pre objekt určiť jeho najbližšieho suseda pomocou metódy popísanej vyššie, potom môžeme nájsť v druhom hľadaní druhého najbližšieho suseda, ak z hľadania vylúčime objekt, ktorý bol najbližším susedom. Tento postup opakujeme, kým nemáme dostatočný počet objektov na zobrazenie užívateľovi, alebo kým niektorý z nájdených objektov už nie je taký rozdielny, že ho už podľa zvolených kritérií nepovažujeme za podobný objekt. Ak hľadanie robíme na základe zaokrúhľovacích algoritmov z časti 2.6.7 tejto práce, potom môžeme vyberať niekoľko najbližších objektov a potom určiť celý výsledok hľadania po jednom kroku algoritmu.

4.4 Konštrukcia rodiny lokálnych hašovacích funkcií na báze náhodných nadrovín

V tejto časti si popíšeme ako nájsť rodiny hašovacích funkcií, ktoré splňujú podmienku.

$$\mathbf{P_{h \in F}(h(x)=h(y)) = sim(x,y)}$$

Predpokladajme, že objekty sú popísané vektormi s počiatkom v počiatku súradnicovej sústavy a podobnosť objektov je daná uhlom medzi objektami. Čím je uhol medzi dvoma vektormi väčší, tým viac sú dané dva vektory rozdielne. Majme dva vektory A a B, ktoré neležia na jednej priamke. Tieto dva vektory nám určujú rovinu ρ. Práve v tejto rovine meriame uhol medzi týmito vektormi. V práci [37] je navrhnutá metóda konštrukcie lokálnych hašovacích funkcií založená na náhodných nadrovinách v n-rozmernom vektorovom priestore. Nech má náš vektorový priestor dimenziu n>1. Zvolíme v ňom náhodnú nadrovinu dimenzie n-1, ktorá prechádza počiatkom súradnicovej sústavy. Prienik náhodnej nadroviny a roviny ρ určenej vektormi A a B môže byť priamka, alebo v špeciálnom prípade to môže byť celá rovina ρ. Prípad ak je prienikom celá rovina nebudeme uvažovať a budeme ho riešiť ako špeciálny prípad. Predpokladajme teda, že prienikom n-1 dimenzionálnej náhodnej nadroviny s rovinou ρ je priamka. Celú situáciu vidíme na obrázku 6. Na obrázku sú zakreslené vektory A a B, uhol medzi nimi a sú tu ako priamky premietnuté dve náhodné nadroviny označené 1 a 2.



Obrázok 6

Každá z priamok 1 a 2 nám rozdeľuje rovinu na dve polroviny. Pravdepodobnosť, že sa vektory A a B nachádzajú v rôznych polrovinách je rovná pravdepodobnosti, že smerový vektor priamiek 1 a 2 sa nachádza medzi vektormi A a B. Táto

pravdepodobnosť je teda priamo úmerná veľkosti uhla medzi vektormi A a B. Definujme teraz rodinu hašovacích funkcií na základe množiny všetkých n-1 dimenzionálnych nadrovín prechádzajúcich počiatkom súradnicovej sústavy v n-dimenzionálnom vektorovom priestore. Každá nadrovina nám delí priestor na dve polroviny. Hašovacia funkcia určená nadrovinou bude priradzovať objektu hodnotu 0 ak sa nachádza v jednej z polrovín a hodnotu 1 ak sa nachádza v druhej polrovine. Pravdepodobnosť, že majú dva objekty rôznu hodnotu hašovacej funkcie je priamo úmerná veľkosti uhlu medzi vektormi. Veľkosť uhlu pritom skutočne určuje rôznosť objektov, preto takáto rodina hašovacích funkcií spĺňa podmienku

$\mathbf{P_{h \in F}(h(x)=h(y)) = sim(x,y)}$.

Problémom teraz zostáva určiť spôsob ako generovať náhodné nadroviny v priestore s vysokou dimenziou. Náhodná nadrovina, ktorá prechádza počiatkom súradnicovej sústavy môže byť popísaná svojím normálovým vektorom v n rozmernom priestore. Aby išlo skutočne o náhodnú nadrovinu, musia tieto vektory rovnomerne pokrývať n-rozmernú hyperguľu s polomerom 1.

4.4.1 Výber bodu na jednotkovej guli

Výberom náhodného bodu na jednotkovej guli sa zaoberá článok [15]. Už na dvojrozmernom priestore vidíme, že nestačí vybrať pre každú súradnicu náhodnú hodnotu z intervalu [-1,1] a takto získaný vektor normalizovať. Ak by sme postupovali týmto spôsobom, väčšina bodov by bola sústredená v strede kvadrantov, teda v okolí uhlov $\pi/4$, $3*\pi/4$, $5*\pi/4$ a $7*\pi/4$. Pri dvojrozmernom prípade by nám stačilo zvoliť náhodný uhol Θ z intervalu $[0, 2\pi)$ a potom zvoliť:

x=cos(Θ)

y=sin(Θ)

Takto by sme kružnicu pokryli rovnomerne. Už v trojrozmernom prípade však analogický postup zlyháva. Ak chceme vybrať náhodný bod na jednotkovej guli v trojrozmernom priestore, bolo by nesprávne vybrať uhol Θ z intervalu $[0,2\pi)$ a uhol ϕ z intervalu $[0,\pi]$ a potom použiť tieto uhly ako sférické súradnice:

x=cos(Θ)*sin(ϕ)

$y=\sin(\theta)*\sin(\phi)$

$z=\cos(\phi)$

Ak by sme generovali body takýmto spôsobom, potom by bol element plochy gule vyjadrený ako:

$d\Omega = \sin(\phi)* d\theta * d\phi$

Hustota bodov by teda bola funkciou uhla ϕ. Na póloch gule by sme získali vysokú hustotu bodov a naopak v okolí roviny xy by bola hustota bodov menšia. Marsaglia v práci [17] ukázal spôsob ako vybrať náhodný bod na guli v trojrozmernom priestore. Najskôr vygenerujeme dve náhodné čísla x_1 a x_2 z intervalu (-1,1), pričom vylúčime dvojice pre ktoré platí:

$$(x_1^2 + x_2^2) \geq 1$$

Potom body:

$$x = 2 * x_1 * \sqrt{1 - x_1^2 - x_2^2}$$
$$y = 2 * x_2 * \sqrt{1 - x_1^2 - x_2^2}$$
$$z = 1 - 2 * (x_1^2 + x_2^2)$$

tvoria rovnomerné pokrytie jednotkovej gule v trojrozmernom priestore. V práci [17] sa ďalej dokáže, že sa táto metóda dá rozšíriť i na viacrozmerný priestor.

Ďalšou možnosťou ako vybrať náhodný bod na jednotkovej n-rozmernej guli je vygerovať n náhodných Gausových čísel $x_1,x_2,\ldots x_n$. Vektory:

$$\frac{1}{\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

potom rovnomerne pokrývajú jednotkovú guľu v n-rozmernom priestore [17,18]. Táto metóda je vhodná pre konštrukciu náhodných nadrovín, ktoré potrebujeme v časti 4.4 tejto práce. Pri pridávaní vektorov, ktoré popisujú inzeráty nám môže rásť dimenzia priestoru všetkých tokenov. Od generátora náhodných nadrovín teda požadujeme možnosť rozšírenia dimenzie v prípade, že musíme spracovať nový inzerát, ktorý obsahuje doteraz nenájdený token. Náhodná nadrovina je definovaná svojim normálovým vektorom. Výhodou je, že tento normálový vektor nemusí mať jednotkovú veľkosť. Môžeme teda vynechať normalizáciu vektora, čím ušetríme časť

výpočtového vykonu, ale ešte dôležitejšie je, že rozšírenie dimenzie môžeme spraviť jednoduchým rozšírením vektora o nové nahodné Gaussovo číslo.

4.4.2 Generátor náhodných Gaussových čísel

Problematike generovania pseudo-náhodných čísel s gaussovým rozdelením sa obšírne venuje práca [20].

Metóda generovania týchto čísel má na vstupe pseudo-náhodné čísla x_1,x_2 z intervalu [0,1] a transformuje ich na čísla y_1,y_2 s Gausovou distribúciou (normálnou distribúciou) so strednou hodnotou 0 a odchýlkou 1. Základný tvar tejto transformácie je uvedený v článku [16] ako:

$$y_1 = \sqrt{-2 * \log(x_1)} * \cos(2 * \pi * x_2)$$
$$y_2 = \sqrt{-2 * \log(x_1)} * \sin(2 * \pi * x_2)$$

Tento tvar však podľa [16] nie je vhodný pretože sa pri ňom vykoná veľa matematických operácií a môže mať problémy s numerickou stabilitou, ak sa x_1 blíži k nule. Vhodnejšie je použiť polárnu formu transformácie popísanú v [19] (Box-Mullerova transformácia). Na začiatku zvolíme x_1,x_2 z intervalu [-1,1]. Vylúčime dvojice pre ktoré platí:

$$(x_1^2 + x_2^2) \geq 1$$

Transformácia má potom tvar:

$$y_1 = x_1 * \sqrt{\frac{-2 * \log(x_1^2 + x_2^2)}{(x_1^2 + x_2^2)}}$$
$$y_2 = x_2 * \sqrt{\frac{-2 * \log(x_1^2 + x_2^2)}{(x_1^2 + x_2^2)}}$$

Generátor pseudo-náhodných čísel s gaussovou distribúciou môžeme použiť na efektívne generovanie náhodných nadrovín.

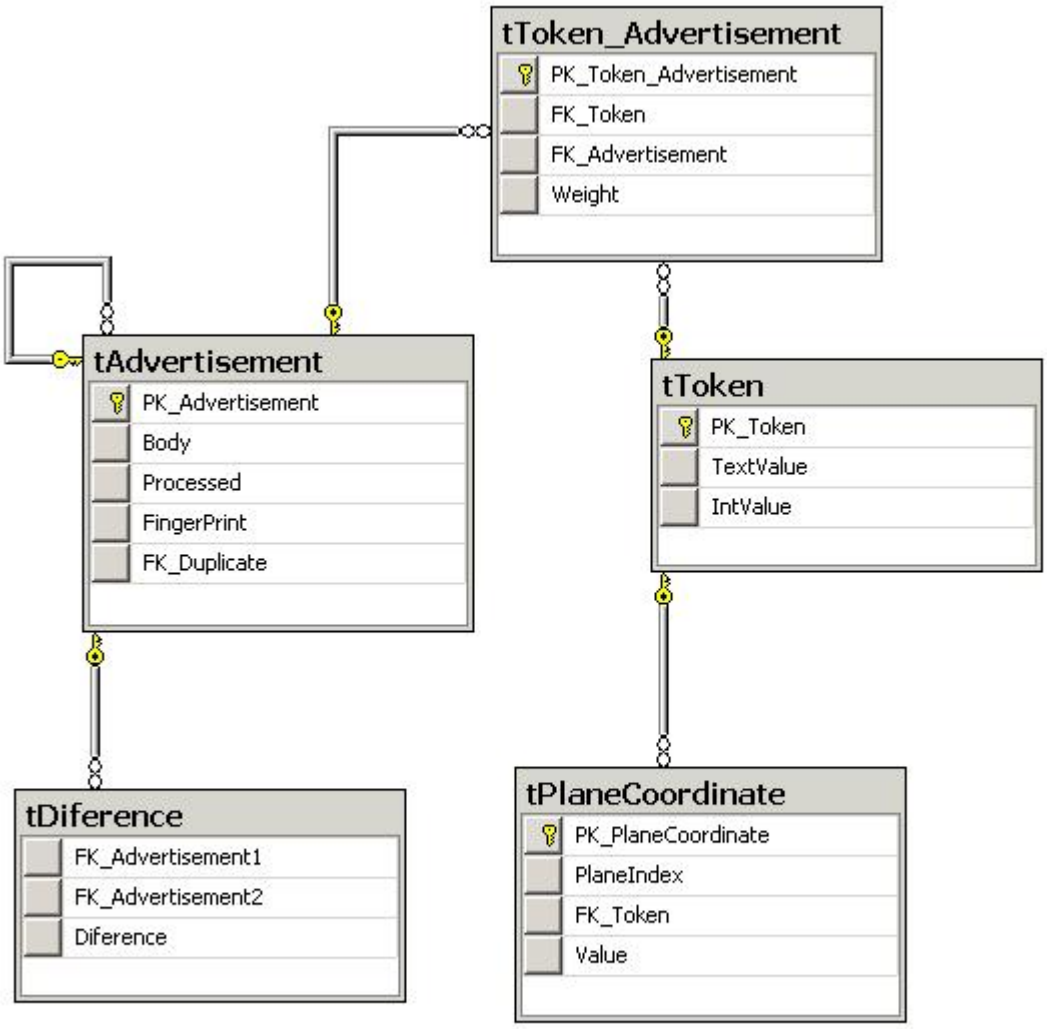
5. Implementácia

Pre implementačnú časť diplomovej práce som sa rozhodol použiť programovací jazyk Java a relačnú databázu PostgreSQL. Výhodou použitia PostgreSQL je to, že pre tento databázový engine existuje modul s názvom TSearch2, ktorý podporuje parsovanie dokumentov a full-textové vyhľadávanie v dokumentoch v češtine. Použitie Javy je potom spôsobené hlavne multiplatformnosťou jazyka a tým, že pre kombináciu PostgreSQL a Java existujú vhodné programovacie nástroje. S programovacím jazykom Java takisto ako s daným databázovým enginom som pred touto prácou nemal žiadne praktické skúsenosti, no počas práce sa voľba ukázala byť vhodná na tento problém. S rovnakými vývojovými nástrojmi a programovacím jazykom pracuje spoločnosť ktorá mi poskytla reálne testovacie dáta. To ma nakoniec ovplyvnilo použiť tieto nástroje na implementáciu algoritmu na klastrovanie inzerátov.

5.1 Databázový model

Použitý databázový model vidíme na obrázku 7. Hlavnou tabuľkou je tabuľka **tAdvertisement** v ktorej sú uložené jednotlivé inzeráty v textovej podobe. Tabuľka obsahuje pre každý inzerát informáciu o tom, či je inzerát už zpracovaný, aký je jeho fingerprint (odlačok), čo je vlastne hodnota náhodne vybraných hašovacích funkcií z množiny lokálnych hašovacích funkcií, spojená do jedného reťazca. Tabuľka obsahuje ešte stĺpec FK_Duplicate, ktorý je vyplnený vtedy, ak už v databázi existuje inzerát s úplne rovnakým textom. Inzeráty, ktoré majú úplne rovnaký text, totiž budú mať aj rovnakú hodnotu odtlačku a nie je potrebné takýto inzerát klastrovať opakovane. V prípade nájdenia inzerátu, ktorý už bol zaklastrovaný sa preto len vyplní príslušný primárny kľúč pôvodného inzerátu do položky FK_Duplicate.

Ďalšou tabuľkou je tabuľka **tToken**, ktorá obsahuje jednotlivé tokeny ktoré sa v inzerátoch vyskytujú. Tokenom pritom môže byť buď jedno číslo, alebo jedno slovo v texte. Generovanie tokenov si bližšie popíšeme v samostatnom odseku. Počet všetkých tokenov n nám určuje dimenziu vektorového priestoru do ktorého sa na základe modelu vektorového priestoru zobrazujú vektory odpovedajúce inzerátom.



Obrázok 7

Väzbu medzi tokenom a inzerátom obsahuje tabuľka **tToken_Advertisement**. Táto tabuľka obsahuje primárne kľúče inzerátu a tokenu a váhu tokenu v danom inzeráte. Tabuľka neobsahuje riadky, ktoré majú nulovú váhu. Keďze inzerát je zvyčajne krátky, táto tabuľka obsahuje pre každý inzerát len niekoľko desiatok záznamov.

Ďalšou tabuľkou je **tPlaneCoordinate**, ktorá obsahuje normálové vektory náhodných nadrovín. Tieto nadroviny nám určujú m vybraných hašovacích funkcií. Ak je dimenzia vektorového priestoru n a počet hašovacích funkcií m, potom tabuľka bude obsahovať m*n náhodných gaussových čísel.

Poslednou tabuľkou je **tDiference**, v ktorej sú uložené hammingove vzdialenosti medzi odtlačkami inzerátov. Neukladáme sem tieto vzdialenosti pre všetky dvojice, pretože veľkosť tabulky by potom bola obrovská, ale len vzdialenosti dvojíc ktoré algoritmus vyhodnotí ako podobné. Ktoré to sú si tiež povieme neskôr.

5.2 Program

Program pozostáva z dvoch modulov. Prvým je modul určený na import dát do databázy a druhý modul slúži priamo na klastrovanie inzerátov.

5.2.1 Import dát

Množinu 10000 inzerátov na testovacie účely som získal od spoločnosti, ktorá sa zaoberá monitorovaním inzercie. Inzeráty boli dodané vo formáte xml, kde v poli "pole", ktoré obsahovalo atribút "jméno" s hodnotou "text" sa nachádzal samotný text inzerátu. Príklad takéhoto súboru vidíme na nasledujúcich riadkoch:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <inzerat>
  <pole jmeno="indexed">2006081807</pole>
  <pole jmeno="datum">20060818</pole>
  <pole jmeno="kraj" />
  <pole jmeno="text">Prodej, byt, Praha 3-Žižkov, 4+1, 83 m2, panel, osobní
    vlastnictví, cena 2.700.000 Kč, Tonza reality spol. s r.o., Mezibranská
    1579/4, Praha 1- Nové Město, email:</pole>
  <pole jmeno="cena">nezadána</pole>
  <pole jmeno="ntyp">byty</pole>
  <pole jmeno="source">Annonce Byty od 2 do 3 milionu - nabidka</pole>
  <pole jmeno="np">nabidka</pole>
  <pole jmeno="telefon">INZANCP36024</pole>
  </inzerat>
```

Modul určený na import dát má na vstupe meno adresára ktorý obsahuje xml súbory s inzerátmi. Tento modul postupne prechádza jednolivé xml súbory v adresári a ak nájde text inzerátu, potom, ho vloží do databáze s tým že bit Processed je nastavený na hodnotu false.

5.2.2 Klastrovací modul

Klastrovací modul pozostáva z niekoľkých tried. Trieda **Token** je určená na uchovávanie dát jednotlivých tokenov a neobsahuje prakticky žiadne metódy. Trieda **Parser** je určená na vygenerovanie tokenov z textu inzerátu, tento proces si popíšeme bližšie. Ďalšia trieda Gaussian_random_number_generator obsahuje implementovanú Box-Mullerovu metódu generovania pseudo-náhodných čísel s gaussovým rozdelením. Túto triedu som v programe nakoniec nepoužil, pretože samotný jazyk Java obsahuje generátor takýchto pseudonáhodných čísel. Poslednou triedou je trieda **Clustering**, ktorá obsahuje všetky klastrovacie metódy. Aj túto triedu si detailne popíšeme.

5.2.3 PostgreSQL modul TSearch2

Na parsovanie inzerátu som chcel použiť modul TSsearch2, ktorý je súčasťou PostgreSQL a obsahuje funkciu **parse**. Táto funkcia však nemá prakticky žiadne možnosti nastavenia a neukázala sa ako vhodná na tokenizáciu inzerátov. Inzeráty v češtine často obsahujú číslovky v ktorých sú bodkami vyznačené tisícky a milióny, napríklad “2.700.000Kč”. Funkcia parse v takomto reťazci nenájde číslovku. Ďalšou dôležitou informáciou je potom informácia o počte izieb, napríklad “3+1kk”. Takýto reťazec funkcia parse rozloží na 3 tokeny “3”, “+”, “1kk”. To aby tento reťazec bol spracovaný ako samostatný token sa však ukazuje ako kritická vlastnosť ktorú požadujeme po implementácii klastrovania, pretože často iba na základe tohto jedného reťazca môžeme odlíšiť dva inzeráty. Ak má totiž byt určitý počet izieb, potom úplne rovnaký počet izieb bude uvedený vo všetkých inzerátoch na túto nehnuteľnosť. Z tohoto dôvodu som sa nakoniec rozhodol implementovať vlastný parser, ktorý je zameraný práve na to, aby dobre zvládal číslovky v uvedenom tvare a počty izieb bytov.

Modul TSearch2 však obsahuje ešte jednu funkciu s názvom **lexize**, ktorá prevedie slovo na jeho základný tvar. Použitie lemmatizace je veľmi výhodné, pretože sa nám zmenší dimenzia vektorového prestoru v ktorom pracujeme a navyše označíme ako podobné dva inzeráty ktoré obsahujú to isté slovo, ale v rôznych slovných tvaroch.

Funkciu lexize som preto pri implementácii použil s mojím parserom na vytvorenie množiny tokenov.

5.2.4 Parsovanie inzerátu

Počas parsovania inzerátu sa vytvára množina tokenov ktoré daný inzerát obsahuje. Na začiatku sa snažíme v texte identifikovať číslovky, potom text rozdelíme na jednotlivé slová, pričom oddeľovačmi sú medzery, bodky a čiarky. Na konci priradím jednotlivým tokenom váhy, podľa počtu výskutov v texte.

5.2.5 Parsovanie čísel

V češtine sa na rozdiel od iných jazykov používa desatinná čiarka a nie desatinná bodka. V praxi sa však stretávame s rôznymi zápismi čísel. Často i české inzeráty obsahujú desatinné bodky, alebo naopak, obsahujú bodky na zvýraznenie tisícov a miliónov (“2.700.000Kč”). Pri parsovaní čísel som sa snažil zachytiť všetky možnosti, na ktoré som v svojich dátach narazil.

Problémom však zostáva, čo s číslami, ktoré sa v texte podarí identifikovať. Čísla v inzerci tvoria najdôležitejšiu informáciu. Ide hlavne o cenu nehnuteľnosti, jej rozlohu a podobne. Problémom je, že cena sa môže postupom častu zvyšovať, alebo znižovať a rovnako rozloha môže byť v dvoch inzerátoch zaokrúhlená rôznym spôsobom. V práci som sa zameral aj na to, ako odhaliť podobnosť medzi podobnými číslami v inzeráte. Uvažujme napríklad dvojicu čísel 5 a 6 a inú dvojicu 1000 a 1001. Aj keď v každej z dvojíc je rozdiel medzi číslami rovnaký, prvú dvojicu by sme nechceli označiť ako podobné čísla, druhá dvojica však podobné čísla sú. Zdá sa teda vhodné zaviesť na čísla logaritmickú mierku, pretože potom môžme za podobnosť čísel označiť rozdiel ich logaritmov. Túto logaritmickú mierku rozdelíme na úseky konštantnej dĺžky a čísla v jednom úseku sú podobné. Takýto úsek na logaritmickej mierke nám potom predstavuje jeden token. Špeciálne musíme vyriešiť prípady čísel, ktoré sú takmer rovnaké, ale na logaritmickej mierke ich bude oddeľovať hranica nami zvolených intervalov. Problém môžeme vyriešiť ak zvolíme menšie intervaly. Ak číslo patrí do i-tého intervalu, potom priradíme najväčšiu váhu intervalu i, ale nenulové

váhy priradíme aj niekoľkým okolitým intervalom. Dve čísla ktoré sú v susedných intervaloch potom budú obsahovať spoločné tokeny s nenulovou váhou. Takéto inzeráty potom budú označené za podbné i keď čísla v nich sa nezhodujú úplne.

5.2.6 Samotné klastrovanie

Po spustení klastrovacieho modulu sa z databázy vyberú nové inzeráty. Sú to tie, ktoré majú bit processed nastavený na 1. Tieto inzeráty sú potom zparsované a každému je priradená množina tokenov ktoré inzerát obsahuje. V prípade, že inzerát obsahuje nový token, tento je pridaný do databáze. Proces pridávania tokenu si popíšeme neskôr. Na základe tokenov a existujúcich náhodných nadrovín v priestore tokenov sú vyhodnotené hodnoty lokálnych hašovacích funkcií a spojením týchto hodnôt do vektora vznikne odlačok inzerátu. Odtlačky sú potom porovnané s historickými dátami v databáze a do tabuľky tDiference sú pridané informácie o podobných inzerátoch.

5.2.7 Vytvorenie tokenu

Pri vytváraní tokenu nestačí pridať záznam do tabuľky tToken. Keďže sa nám zvýšila dimenzia priestoru všetkých tokenov, musíme navyše rozšíriť existujúce náhodné nadroviny o novú dimenziu. To znamená, že musíme vygenerovať m geussových pseudo-náhodných čísel a pridať ich do tabuľky tPlaneCoordinate. Tým je vytváranie tokenu kompletné.

5.2.8 Časová a pamäťová zložitosť jednotlivých operácií

Označme n počet všetkých tokenov, c počet inzerátov, m počet zvolených náhodných nadrovín a d priemerný počet tokenov v jednom inzeráte. Pamäťová zložitosť algoritmu je O(c*d + m*n), pretože pre každý inzerát si musíme pamätať zoznam tokenov ktoré obsahuje (c*d) a pre každý rozmer priestoru tokenov musíme mať súradnicu náhodnej nadroviny (m nadrovín v n-dimenzionálnom priestore). Časová zložitosť je O(c*d*m), pretože pre každý inzerát musíme spracovať každý term a to odpovedá pridaniu a vyhodnoteniu m-záznamov z tabuľky tPlaneCoordinate. Po

prvom zaklastrovaní historických inzerátov však vieme každý nový inzerát klastrovať v čase O(d*m+c*m), čo je čas potrebný na vytvorenie odtlačku (d*m) a porovnanie odtlačku s historickými dátami (c*m). Vidíme, že pridanie nového inzerátu je závislé na počte historických inzerátov, preto pri spracovaní všetkých inzerátov je skutočne algoritmus kvadratický. Túto operáciu však nikdy nebudeme v praxi robiť. Pre historické dáta určíme ich odtlačky a iba nové inzeráty budeme klastrovať s ohľadom na historické dáta. Operácie nad bitovými vektormi sú pritom jedny z najrýchlejších operácií v počítači, preto pridanie nového inzerátu prebehne skutočne rýchlo. Ak by však rýchlosť algoritmu nepostačovala, je možné použit rýchlejšie hľadanie podobných dokumentov popísané v časti 2.6.7 tejto práce.

5.3 Spustenie programu

Pred spustením programu je potrebné nainštalovať PostgreSQL databázu a Java Runtime Enviroment. Ak chceme použiť lemmatizáciu z modulu TSearch2, musíme nainštalovať aj tento modul a nastaviť v ňom cesty k slovníkom českého jazyka. Popis týchto nastavení sa nachádza v dokumentácii tohto modulu.

5.3.1 Nastavenia programu

Pred spustením programu musíme ešte vytvoriť v databázi štruktúru tabuliek. K tomu slúži dodaný skript, ktorý vytvorí všetky potrebné tabuľky a indexy nad tabuľkami. Pri vytváraní tabuľky tAdvertisement si musíme zvoliť dĺžku odtlačku, ktorá musí byť rovnako nastavená v databázi aj v programe. V programe musíme potom správne nastaviť connection string k databáze. Potom môžme spustiť import dát. Na príkazovom riadku aplikácia očakáva ako paraeter meno adresára s inzerátmi v XML podobe. Po naimportovaní dokumentov spustíme klastrovaciu aplikáciu a výsledky sú po jej ukončení uložené v tabuľke tDiference. Pre dvojice podobných inzerátov sa tu nachádzajú primárne kľúče oboch inzerátov z tabuľky tAdvertisement a potom početrozdielnych bitov v odtlačkoch inzerátov. Do tejto tabuľky sú pritom uložené len tie hodnoty, kde tento počet nepresahuje určitú hraničnú hodnotu. Nad touto hodnotou sú už inzeráty označené ako rozdielne.

6. Testy

Implementáciu tohto klastrovacieho algoritmu som mal možnosť testovať na vzorke dát o veľkosti približne 10000 inzerátov, ktoré boli dodané spoločnosťou ktorá sa prakticky zaoberá monitorovaním realitnej inzercie.

6.1 Návrh testov

Testy ktoré som robil boli zamerané na dve oblasti. Prvou bol výkon celého algoritmu, jeho čas behu pre rôzne veľké vstupné dáta a veľkosť dát ktoré algoritmus ukladá do databáze. Druhou skupinou testov boli testy zamerané na skutočné výsledky klastrovania. Tieto testy sa nedajú robiť automaticky a teda spočívali v opakovanom spúštaní klastrovania a prezeraní si výsledkov behu algoritmu.

6.2 Výsledky testov

Počas behu algoritmu sa v databázi vygenerovalo z 10000 inzerátov približne 30000 záznamov v tabuľke tToken. Inzeráty teda obsahovali 30000 navzájom rôznych slov, alebo čísel. Veľkosť vstupných dát bola približne 150MB. Databáza mala po behu algoritmu veľkosť okolo 1.2GB. Z toho vyplýva aj relatívne dlhší čas behu algoritmu, pretože niektoré operácie vyžadovali prístupy na disk.

6.2.1 Čas behu algoritmu

Čas behu algoritmu je závislý na veľbe počtu náhodných nadrovín, ktorými budeme hašovať inzeráty. Čas je však ďalej závislý i na rýchlosti parsovania textu a na počte novo pridaných tokenov. Počet náhodných nadrovín potom lineárne ovplyvňuje veľkosť tabuľky tPlaneCoordinate a to sa tiež prejaví na rýchlosti algoritmu. Pri klastrovaní rádovo miliónov inzerátov nie je vhodné použiť dĺžku odtlačku menšiu, alebo rovnú 32, pretože 32 bitov by nevytvorilo dostatočný počet kategórií inzerátov. Pre dĺžku 64 pracoval algoritmus 15 minút, pre dĺžku 128 potom 24 minút. Pre dĺžky nad 256 bitov už veľkosť databáze presahovala 2GB a tieto dĺžky nám už nepriniesli

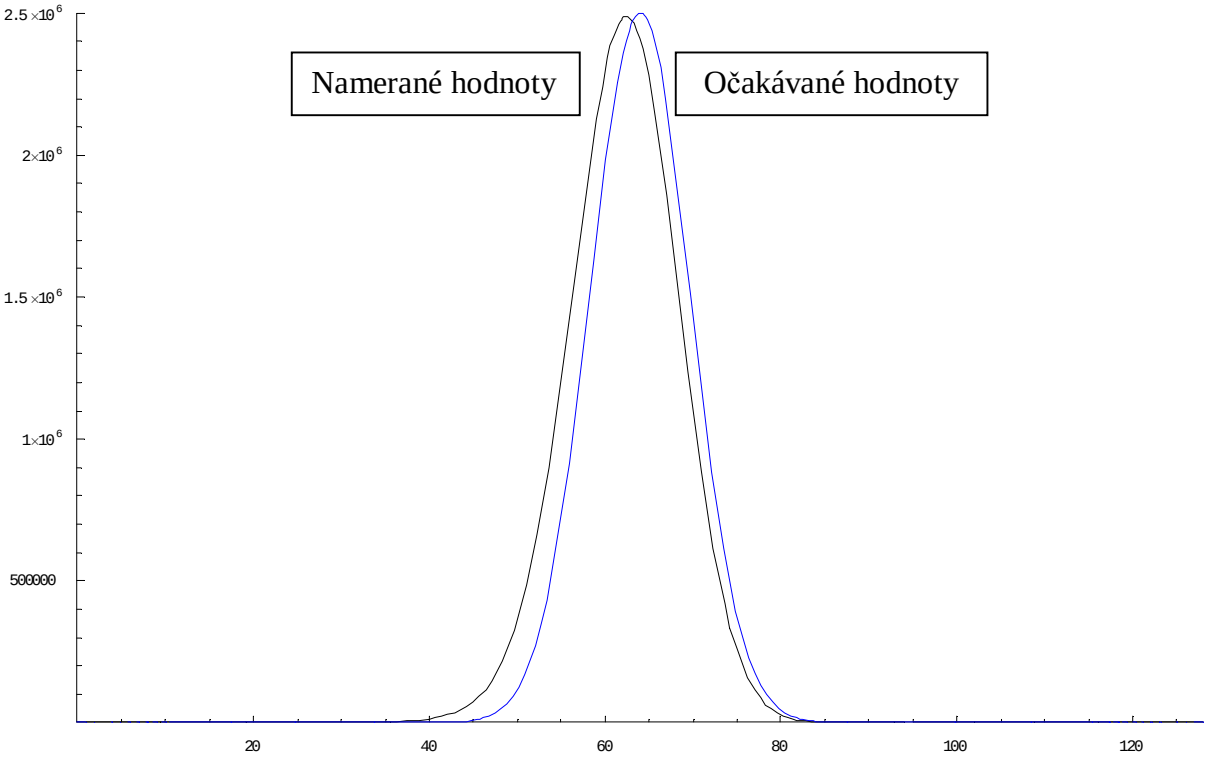
žiadne zlepšenie výsledkov, pretože už dĺžka 128 bitov vytvorí dostatočný počet rôznych odtlačkov. Veľkosť databáze závisí hlavne na počte rôznych tokenov a dĺžke odtlačku. Najväčšou tabuľkou teda bude tabuľka tPlaneCoordinate. Predpokladáme, že pridávaním ďalších inzerátov nad 10000 už nezískame veľké množstvo nových tokenov, takže rýchlosť rastu databáze by sa mala zmenšiť.

6.2.2 Výsledky manuálnych testov

Počas prezerania inzerátov v databáze som sa snažil pre každú zle klastrovanú dvojicu odhlaiť dôvod, prečo sa nepodarilo inzeráty klastrovať správne. Počas tohto testovania som vylepšil parsovací algoritmus, hlavne čo sa týka parsovania čísel, keďže sa ukázalo, že s rozpoznaním čísel a určením ich hodnoty mal algoritmus najväčšie problémy. Výsledky testov potom skutočne odpovedali očakávaným výsledkom na základe počtu rovnakých slov v dvojici inzerátu.

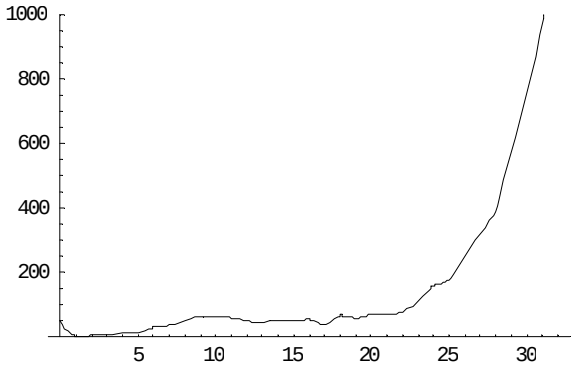
6.3 Štatistické výsledky

Ak by bol vyskyt tokenov v jednotlivých inzerátoch náhodný, potom by i odtlačok inzerátu mal byť reťazcom náhodných bitov. Ak porovnávame dva náhodné bitové reťazce, potom pravdepodobnosť že počet zhodných pozícií je x je priamo úmerna kombinačnému číslu x!/(m!*(m-x)!) kde m je počet bitov reťazca. Najpravdepodobnejšie by malo byť, že reťazce sú zhodné v m/2 bitoch a situácia, že reťazce sú úplne rovnké, alebo úplne rozdielne je najmenej pravdepodobná. Na algoritme som skúmal ako je to s touto pravdepodobnosťou v prípade inzerátov. Ukazuje sa, že rozloženie pravdepodobnosti je oproti očakávaným hodnotám posunuté k viac podobným inzerátom. Je to spôsobené tým, že inzeráty neobsahujú jednotlivé tokeny náhodne, ale niektoré tokeny sú častejšie ako ostatné. Rozloženie pravdepodobnosti je zobrazené na obrázku 8.



Obrázok 8

Ľavá spodná časť obrázku 8 je potom detailnejšie zobrazená na obrázku 9.



Obrázok 9

Z týchto údajov môžeme určiť hranicu podobnosti pri ktorej označíme dva inzeráty za podobné. Pre hodnoty väčšie ako 25 sa graf správa ako štatistické náhodné rozdelenie inzerátov, no pre hodnoty menšie ako 25 sa výrazne odchyľuje a v tejto oblasti vidíme inzeráty ktoré sú podobné. Ako hranicu kedy sú inzeráty podobné by sme mali pri dĺžke odtlačku 128 zvoliť hammingovu vzdialenosť okolo 25.

7. Záver

Pri písaní tejto práce som sa oboznámil s množstvom klastrovacích algoritmov, metód a optimalizácií, ktoré sa dajú pri klastrovaní použiť. Takisto som sa oboznámil s problematikou klastrovania výsledkov webového vyhľadávania a aj s problematikou monitoringu inzercie. Oboznámil som sa i s rôznymi prístupmi k určovaniu podobnosti a rozdielnosti objektov. Venoval som sa aj štúdiu okrajových tém pri spracovaní textu ako je rozpoznávanie jazyka, tokenizácia, lemmatizácia, modely reprezentácie dát pre klastrovanie aj pre full-textové vyhľadávanie dokumentov. V práci som popísal existujúce klastrovaci metódy a zhodnotil som, ako sa dajú použiť pri klastrovaní inzercie. Vychádzajúc z požiadavkov na klastrovanie inzercie som navrhol použiť fuzzy klastrovací algoritmus založený na rodinách lokálnych hašovacích funkcií. Tento algoritmus som implementoval v programovacom jazyku Java a s použitím relačnej databázy. Pri programovaní som veľkú pozornosť venoval priebehu parsovania textu, tak aby tento proces čo najviac odpovedal potrebám klastrovania inzercie. Výsledný program dokáže rýchlo pridať nový inzerát medzi už klastrované inzeráty a rieši požiadavky kladené na monitoring inzercie. Ďalší smer vývoja tohto algoritmu ukáže až jeho nasadenie v praxi.

7.1 Ciele do budúcnosti

V budúcnosti by som chcel popísanú klastrovaciu metódu implementovať v praxi nad väčším množstvom dát. Ak by sa algoritmus dostal do skutočnej prevádzky a používal by sa pri monitoringu skutočnej inzercie, užívatelia by určite odhalili ďalšie oblasti v ktorých by sa mal algoritmus vylepšiť. Chcel by som tiež implementovať vyhľadávanie podobných inzerátov na základe zaokrúhľovacích algoritmov tak ako je to popísané v práci [12], alebo v časti 2.6.7 tejto práce.

8. Literatúra

[1] Jain A.K., Murty M.N., Flynn P.J. (1999) Data Clustering: A Review

[2] Michalski R., Stepp R.E. Diday E. (1983) Automated construction of classifications: conceptual clustering versus numerical taxonomy. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-5, 5 (Sept.), 396–409

[3] Gowda K.C., Krishna G. (1977) Agglomerative clustering using the concept of mutual nearest neighborhood. Pattern Recogn. 10, 105–112

[4] Zhang K. (1995) Algorithms for the constrained editing distance between ordered labeled trees and related problems. Pattern Recogn. 28, 463–474

[5] Sneath P.H., Sokal R.R. (1973) Numerical Taxonomy. Freeman, London, UK

[6] King B. (1967) Step-wise clustering procedures. J. Am. Stat. Assoc. 69, 86–101

[7] Dubes R.C. (1987) How many clusters are best?—an experiment. Pattern Recogn. 20, 6 (Nov. 1, 1987), 645–663

[8] Ester M., Kriegel H.-P., Sander J., Xu X. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd ACM SIGKDD, 226-231, Portland, Oregon

[9] Ankerst M., Breunig M., Kriegel H.-P., Sander J. (1999) OPTICS: Ordering points to identify clustering structure. In Proceedings of the ACM SIGMOD Conference, 49-60, Philadelphia, PA

[10] Agrawal R., Gehrke, J., Gunopulos D., and Raghavan P. (1998) Automatic subspace clustering of high dimensional data for data mining applications. In Proceedings of the ACM SIGMOD Conference, 94-105, Seattle, WA

[11] Indyk, P., Motwani, R. (1998) Approximate nearest neighbors: towards removing the curse of dimensionality. Proc. 30th STOC pp. 604–613

[12] Charikar M. S. (2002) Similarity Estimation Techniques from Rounding Algorithms

[13] Banarjee S., Ramanathan K., Gupta A. (2007) Clustering Short Texts using Wikipedia

[14] Osiński S. (2003) An Algorithm for Clustering of Web Search Results, Master thesis, Poznań University of Technology

[15] http://mathworld.wolfram.com/SpherePointPicking.html

[16] http://www.taygeta.com/random/gaussian.html

[17] Marsaglia, G. (1972) Choosing a Point from the Surface of a Sphere. Ann. Math. Stat. 43, 645-646

[18] Muller M. E. (1959) A Note on a Method for Generating Points Uniformly on N-Dimensional Spheres, Comm. Assoc. Comput. Mach. 2, 19-20

[19] Box G.E.P, Muller M.E. (1958) A note on the generation of random normal deviates, Annals Math. Stat, V. 29, pp. 610-611

[20] Rubinstein, R.Y. (1981) Simulation and the Monte Carlo method, John Wiley & Sons, ISBN 0-471-08917-6

[21] http://en.wikipedia.org/wiki/Letter_frequencies

[22] http://lucene.apache.org/

[23] Zamir O.E. (1999) Clustering Web Documents: A Phrase-Based Method for Grouping Search Engine Results. Doctoral Dissertation, University of Washington

[24] Stefanowski J., Weiss D. (2003) Web search results clustering in Polish: experimental evaluation of Carrot. Advances in Soft Computing, Intelligent Information Processing and Web Mining, Proceedings of the International IIS: IIPWM´03 Conference, Zakopane, Poland, vol. 579 (XIV), pp. 209-22

[25] Zhang D., Dong Y. (2004) Semantic, Hierarchical, Online Clustering of Web Search Results Proceedings of the 6th Asia Pacific Web Conference (APWEB) Hangzhou, China

[26] Lee R. C. T. (1981). Cluster analysis and its applications. In Advances in Information Systems Science, J. T. Tou, Ed. Plenum Press, New York, NY

[27] Riboni D. (2002) Feature Selection for Web Page Classification

[28] Grefenstette G. (1995) Comparing two language identification schemes. 3rd International Conference on Statistical Analysis of Textual Data, Rome

[29] Hamming R.W. (1950) Error Detecting and Error Correcting Codes, Bell System Technical Journal 26(2):147-160

[30] Spertus E., Sahami M., Buyukkokten O. (2005) Evaluating Similarity Measures: A LargeScale Study in the Orkut Social Network

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

[32] Osiński S. (2004) Dimensionality reduction techniques for search results clustering

[33] http://project.carrot2.org/

[34] Lang N.C. (2004) A tolerance rough set approach to clustering web search results. Master thesis, Faculty of Mathematics, Informatics and Mechanics, Warsaw University

[35] http://www.searchenginelowdown.com/2004/10/web-20-exclusive-demonstration-of.html

[36] Broder A.Z., Glassman S.C., Manasse M.S., Zweig G. (1997) Syntactic clustering of the Web. Proc. 6th

Int'l World Wide Web Conference, pp. 391–404

[37] Cetkovský M. (2006) Clustering Framework, Banchelor thesis

[38] Goemans M. X. Williamson D. P. (1995) Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. JACM 42(6): 1115-1145

[39] Broder A.Z. (1998) Filtering near-duplicate documents. Proc. FUN 98

[40] Cohen S., Guibas L. (1999) The Earth Mover's Distance under Transformation Sets. Proc. 7th IEEE Intnl. Conf. Computer Vision

[41] Gionis A., Indyk P., Motwani R.. (1999) Similarity Search in High Dimensions via Hashing. Proc. 25th VLDB pp. 518-529

Dodatok A: Obsah CD

CD obsahuje tieto adresáre:

- THESIS: obsahuje prácu vo formáte PDF.

- REFERENCES: adresár obsahuje literatúru, ktorá je voľne prístupná v elektronickej podobe

- PROGRAM: obsahuje zdrojový kód programu a databázové skripty, ktoré vytvoria tabuľky v prostredí Postgre databázy.

- INSTALLS: obsahuje inštalačné súbory, ktoré sú potrebné pre spustenie programu. Ide o Postgre SQL server s modulom TSearch2, Java runtime enviroment a Postgre SQL JDBC. Všetok potrebný software je voľne šíriteľný a je dodaný pre operačné systémy Windows i Unix (Linux).

- SAMPLE DATA: Adresár obsahuje príklad reálnych vstupných dát programu

Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE

2007 Tomáš Dzetkulič