Distribuované zhlukovanie textových dokumentov v prostredí Gridu

Martin Sarnovský, Peter Butka, Vladimír Safko

Katedra kybernetiky a umelej inteligencie, Fakulta elektrotechniky a informatiky,
Technická univerzita v Košiciach, Letná 9, 040 01 Košice, Slovensko
{martin.sarnovsky, peter.butka}@tuke.sk, safio@gmail.com

Abstrakt. Táto práca sa zaoberá zhlukovaním textových dokumentov, zhlukovacími algoritmami a možnosťou ich distribúcie v prostredí výpočtového Gridu. Algoritmus GHSOM (Growing Hierarchical Self Organizing Map) je rozšírením štandardného prístupu k zhlukovaniu na báze SOM, ktorý kombinuje adaptivitu rozširovania mapy a hierarchického zhlukovania tvorením nových máp. Znamená to, že každá vrstva hierarchickej štruktúry pozostáva z množiny nezávislých máp, ktoré adaptujú svoju veľkost s ohľadom na požiadavky vstupných dát (príkladov prislúchajúcich danej mape). V našom príspevku je popísaný návrh distribúcie algoritmu GHSOM a jeho implementácia v integrovanom prostredí pre distribuované dolovanie textových dokumentov na Gride - GridMiner. Obsahom práce sú aj výsledky a popis experimentov s distribuovanou verziou algoritmu, ktoré boli vykonané na výpočtovom Gride.

1 Úvod

Problematika reprezentácie a zhlukovania textových dokumentov je v súčasnosti predmetom mnohých výskumov. Spoločným znakom úloh reprezentácie a zhlukovania textových dokumentov je, že algoritmy používané ich riešenie sú časovo a výpočtovo náročné. Narastajúci objem dát predstavuje veľký problém, pretože pri využití klasických výpočtových prostriedkov, môže riešenie takýchto úloh trvať neúnosne dlho.

Vhodným riešením tohto problému je distribúcia algoritmov a dát s využitím Gridových technológií [2]. Ich výhoda spočíva v tom, že pre ich nasadenie je možné využiť už existujúce výpočtové kapacity a infraštruktúru (napr. internet).

Algoritmus GHSOM (Growing Hierarchical Self organizing map [3]) je možné použiť aj na zhlukovanie textových dokumentov, vychádza z architektúry SOM. V súčasnosti existuje mnoho systémov pre zhlukovanie a vizualizáciu textových dokumentov a mnohé z nich fungujú na základe princípu SOM. Základnou vlastnosťou modelov na báze architektúry SOM je ich schopnosť zachovávať topológiu mapovania vstupného priestoru vo výstupnom priestore (mape). Často je problémom klasickej architektúry SOM (Kohonenova mapa) jej pevná štruktúra (po začatí učenia). Model blízky klasickému modelu SOM, Growing Grid, umožňuje zväčšovanie pôvodnej SOM dynamicky počas procesu učenia (pridávaním riadkov resp. stĺpcov

© Václav Snášel (Ed.): Znalosti 2008, pp. 192–203, ISBN 978-80-227-2827-0.
FIIT STU Bratislava, Ústav informatiky a softvérového inžinierstva, 2008.

nových neurónov). Druhá adaptívna metóda je založená na použití hierarchickej štruktúry nezávislých máp, kde pre každý prvok mapy (neurón) je na novej úrovni pridaná nová mapa, ktorá potom podrobnejšie rozdeľuje príklady nadradeného (rodičovského) neurónu. Táto architektúra sa nazýva Hierarchical Feature Map (HFM). Algoritmus GHSOM potom kombinuje postupy týchto dvoch modelov neurónových sietí. Znamená to, že každá vrstva hierarchickej štruktúry pozostáva z množiny nezávislých máp, ktoré adaptujú svoju veľkost s ohľadom na požiadavky vstupných dát (príkladov prislúchajúcich danej mape).

Pri veľkom objeme dát jeho činnosť môže trvať neúnosne dlho. Zrýchlenie sa dá dosiahnuť jeho modifikáciou do distribuovanej podoby. Keďže pozostáva zo samostatných zhlukovacích úloh na čiastkových zhlukoch, je vhodný pre distribúciu na výpočtovom Gride a paralelizáciu.

2 Grid

Napriek tomu, že výkon a kapacita výpočtových prostriedkov rastie skoro exponenciálne, súčasné kapacity nie sú schopné udržať krok s extrémnymi nárokmi, ktoré sú na nich kladené pri vedeckých výpočtoch. Ani superpočítače a paralelné systémy (klastre) už nie sú schopné priniesť uspokojivé výsledky. Tam kde končia ich možnosti, začínajú možnosti distribuovaných systémov, ktoré sú schopné spojiť teoreticky neobmedzené množstvo počítačov, superpočítačov a klastrov bez ohľadu na ich geografickú polohu.

Takýmto systémom hovoríme Grid. Týmto pojmom sa označuje technológia, ktorá umožňuje z geograficky distribuovaných výpočtových a pamäťových zdrojov vytvoriť univerzálny výpočtový systém s extrémne veľkým výkonom a pamäťou [1]. Slovo Grid bolo vybrané z anglického výrazu pre elektrickú sieť – electric power grid, pretože táto symbolizuje komplexné poprepájanie, všeobecnú dostupnosť, skladá sa z heterogénnych komponentov a má veľký vplyv na možnosti ľudí a spoločnosti ako celok.

Grid je infraštruktúra, ktorá je schopná spojiť čiastkové kapacity jednotlivých počítačov a vytvoriť globálny celosvetový počítač, kde jeho časti sú spojené prostredníctvom siete Internet. Vytvára rozhranie, ktoré umožňuje užívateľom priamo pristupovať k prostriedkom jednotným spôsobom, čím poskytuje komplexnú a výkonnú platformu pre globálne výpočty a správu dát [2]. Zdieľanie prostriedkov je prísne kontrolované. Jasne definuje čo sa zdieľa, kto má dovolené zdieľať a podmienky za akých sa zdieľanie odohráva. Jednotlivci alebo inštitúcie definované pravidlami zdieľania vytvárajú virtuálne organizácie (VO). Grid pozostáva z definícii protokolov, služieb, pomocných programov a pokrýva aj otázku bezpečnosti zdieľania zdrojov. Protokoly sú postavené prevažne na báze súčasných internetových protokolov, ktoré sú doplnené a rozšírené.

2.1 Využitie Gridu pri objavovaní znalostí v dátach a textoch

Technológie pre distribuované objavovanie znalostí sú obzvlášť vhodné pre aplikácie, ktoré zvyčajne spracúvajú veľké objemy dát alebo roztrúsené dátové zdroje (digitálne knižnice, vedecké simulácie, obchodné dáta...), ktoré nie je možné analyzovať v prijateľnom čase na klasických počítačoch. Výpočtový Grid v sebe integruje distribuované a paralelné počítanie, teda predstavuje vhodný nástroj pre distribuovanú analýzu dát. Gridové služby objavovania znalostí umožňujú podnikom a spoločnostiam distribuovať výpočtovo náročnú analýzu dát na veľký počet vzdialených zdrojov (počítačov, klastrov, super počítačov). To vedie aj ku vzniku nových algoritmov a techník, ktoré umožňujú analyzovať dáta tam, kde sú uložené, čo je v kontraste s klasickými metódami, kde je potrebné presunúť dáta na centrálne miesto, kde sa vykonáva ich analýza (čo v prípade objemných dát môže klásť neúnosné nároky na počítačovú sieť).

V súčasnosti existuje niekoľko prostredí pre podporu distribuovaného dolovania dát na Gride. Niektoré z nich sú všeobecné prostredia, ktoré podporujú vykonávanie úloh dolovania dát na uzloch Gridu (projekt TeraGrid[1] a projekt DataMiningGrid[2]). Niektoré z prostredí poskytujú nástroje pre kompletný proces dolovania dát pre špecifické aplikácie, ktoré boli upravené pre beh na Gride (KnowledgeGrid[3], Discovery-Net[4], GridMiner[5]) a iné sú konkrétnou implementáciou jednotlivých algoritmov dolovania dát. Okrem tohto projektu sa problematike dolovania v textoch venuje ešte NaCTeM (UK National Centre for TextMining), kde sa zaoberajú dolovaním v medicínskych databázach (aj textových) ktoré takisto navrhuje systém dolovania z dát a textov využívajúci možnosti Gridu [8].

3 Zhlukovanie textových dokumentov

Zhlukovanie je všeobecný logický postup, pomocou ktorého sa zlučujú objekty do skupín (zhlukov) navzájom si podobných objektov na základe ich podobnosti alebo rozdielnosti. Z pohľadu strojového učenia zhluky zodpovedajú skrytým vzorom v dátach, hľadanie zhlukov je nekontrolované učenie a výsledkom je určitý dátový koncept. Zhlukovanie sa realizuje na základe množiny príznakov, ktorá je priradená každému zhlukovanému objektu. Príznaky by mali byť elementárne (ďalej nedeliteľné), aby im bolo možné priradiť rovnakú váhu. V prípade zhlukovania textových dokumentov sú objektmi, s ktorými pracuje zhluková analýza dokumenty (ich vektory) a výstupom sú zhluky dokumentov. Kolekcia dokumentov usporiadaná do zhlukov umožňuje:

- prezeranie (browsing) kolekcie dokumentov bez tvorenia otázok
- pre daný dokument je možné rýchlo nájsť jemu podobné v rámci zhluku

[1] http://www.teragrid.org
[2] http://www.datamininggrid.org
[3] http://grid.deis.unical.it/kgrid
[4] http://www.discovery-on-the.net
[5] http://www.gridminer.org

- otázku zadanú pre kolekciu dokumentov je možné klásť zástupcom zhlukov tejto kolekcie, čím je možné urýchliť vyhľadávanie.

3.1 Algoritmus SOM

Samoorganizujúce sa mapy (Self-Organizing Maps, SOM) [4] patria k metódam sekvenčného nehierarchického zhlukovania. Sú založené na princípe nekontrolovaného konkurenčného učenia. Tento model mapuje zvyčajne vysoko-rozmerný príznakový priestor do (zvyčajne) dvojrozmerného priestoru nazývaného mapa.

Mapa (neurónová sieť) je vytvorená zo základných elementov – neurónov. Každému neurónu je priradený n – rozmerný váhový vektor mi .Váhový vektor má rovnaký rozmer ako vstupné vzorky. Špecifickou črtou SOM je realizácia zobrazenia zachovávajúceho topológiu. Neuróny sú usporiadané v pravidelnej štruktúre (mriežke). Takéto usporiadanie predstavuje výstupný priestor. Vzdialenosť dvoch neurónov v tomto priestore je obyčajne euklidovskou vzdialenosťou vektorov ich súradníc v uvažovanej štruktúre. Zobrazenie ktoré vznikne po natrénovaní SOM má tú vlastnosť, že ľubovoľné dva vzory blízke vo vstupnom priestore evokujú v sieti odozvy na neurónoch, ktoré sú si tiež fyzicky blízke.

Trénovací proces pozostáva z dvoch častí: prezentácia vstupnej vzorky na vstup neurónovej siete a adaptácia váhového vektora. Každá iterácia pozostáva z náhodného výberu jednej vstupnej vzorky. Táto vstupná vzorka sa privedie na vstup SOM a následne sa nastaví aktivácia neurónov. Zvyčajne sa používa na výpočet aktivácie Euklidovská vzdialenosť medzi váhovým vektorom a vstupnou vzorkou. Neurón s najnižšou aktiváciou je víťazom iterácie. Nakoniec sa adaptujú váhy váhového vektora víťazného neurónu ako aj váhy vybraných neurónov v blízkosti víťaza. Adaptácia je realizovaná ako postupné znižovanie rozdielu medzi vstupnou vzorkou a váhovým vektorom.

Z geometrického hľadiska to znamená, že váhové vektory adaptovaných neurónov sa o niečo priblížili vstupnej vzorke. Veľkosť tohto priblíženia je riadená učiacim parametrom α, ktorý v čase klesá. Počet neurónov, ktorých vektory sa adaptujú, je určený na základe funkcie susedstva. Tento počet tiež klesá v čase. Následkom tohto pohybu sa Euklidovská vzdialenosť medzi vektormi zmenšuje a teda váhové vektory neurónov sa stávajú viac podobnými vstupnej vzorke. Príslušný neurón má väčšiu šancu stať sa víťazom v ďalšom cykle. Tým, že sa neadaptuje len samotný víťazný neurón, ale aj niekoľko neurónov v jeho susedstve, dochádza k priestorovému zhlukovaniu podobných vstupných vzoriek v susedných častiach mapy. Podobnosti medzi vstupnými vzorkami, ktoré sa vyskytujú v n-rozmernom vstupnom priestore sa postupne namapujú do dvojrozmerného výstupného priestoru samoorganizujúcej sa mapy, kde vzorky, ktoré sú si podobné v rámci vstupného priestoru, sú umiestnené do geometricky blízkych oblastí.

3.2 Algoritmus GHSOM

Jednou z nevýhod algoritmu SOM je, že štruktúra mapy musí byť definovaná a-priori. Existujú modifikácie, ktoré umožňujú dynamicky rozširovať mapu podľa potrieb vstupného príznakového priestoru (vstupných dát). Ich nevýhodou však je, že nové neuróny sa nemôžu pridávať samostatne, ale iba v celých „pásoch“ (musí sa zachovať tvar rovnomernej mriežky). Ďalšou nevýhodou je, že výsledné mapy sú príliš veľké a je problém s nimi pracovať. Tieto dôvody viedli k vzniku algoritmu GHSOM (Growing Hierarchical Self Organizing Map) [3] , kde mapa rastie:

- hierarchicky – podľa distribúcie dát, čo umožňuje hierarchickú dekompozíciu a navigáciu v podmapách
- horizontálne – veľkosť mapy sa mení tak, aby sa prispôsobila požiadavkám vstupného priestoru

Algoritmus GHSOM má nasledovné kroky:

1. Najskôr sa vyráta celková odchýlka vektora vstupných dát na vrstve 0-tej úrovne. Tejto jednotke sa priradí váhový vektor $m_0 = [\eta_{01}, \eta_{02}, ..., \eta_{0n}]^T$, ktorého zložky sú priemerom zložiek všetkých vstupných dát (vektorov). Stredná kvadratická chyba 0.tej vrstvy sa vyráta podľa vzťahu:
$$mqe_0 = \frac{1}{d} \cdot \|m_0 - x\|,$$
kde *d* je počet prvkov vstupných dát *x* (počet prvkov vstupného vektora).
2. Trénovanie GHSOM začína vrstvou 1. úrovne. Táto mapa je zvyčajne relatívne malá. Každému neurónu *i* sa priradí *n*-rozmerný váhový vektor
$$m_i = [\eta_{i1}, \eta_{i2}, ..., \eta_{in}]^T, m_i \in \Re^n,$$
ktorý je inicializovaný náhodnými hodnotami. Váhový vektor má rovnaký rozmer ako vstupné vektory.
3. Učenie samoorganizujúcej sa mapy je vlastne súperenie medzi neurónmi o to, ktorý najlepšie reprezentuje vstupný vektor (konkurenčné učenie). Neurón s váhovým vektorom, ktorý je najbližšie prezentovanému vektoru na vstupe vyhráva. Váhový vektor víťaza ako aj neuróny v jeho blízkosti sa adaptujú tak, aby sa čo najviac podobali na vstupný vektor. Miera adaptácie je riadená učiacim parametrom α , ktorý v čase klesá. Počet susedných neurónov, ktoré sa adaptujú spolu s víťazom tiež klesá v čase: na začiatku učenia sa adaptuje veľa susedných neurónov a na konci už iba víťaz. To, ktoré neuróny sa adaptujú určuje funkcia susednosti h_{ci}, ktorá je založená na vzdialenosti neurónov k víťazovi meranej v dvojrozmernej mriežke vytvorenej neurónovou sieťou. Kombináciou týchto princípov vzniká učiace pravidlo:
$$m_i(t+1) = m_i(t) \cdot \alpha(t) \cdot h_{ci}(t) \cdot [x(t) - m_i(t)],$$
kde *x* je aktuálna vstupná vzorka a *c* je víťaz v iterácii *t*.
4. Po určitom počte iterácií (daný parametrom λ) sa vyráta stredná kvadratická chyba mapy podľa vzťahu:
$$MQE_m = \frac{1}{u} \cdot \sum_i mqe_i ,$$

kde *u* je počet neurónov *i* v mape *m*, mqe_i je stredná kvadratická chyba neurónu *i* v mape *m*. Každá vrstva GHSOM je zodpovedná za vysvetlenie určitej časti odchýlky vstupných dát z predchádzajúcej vrstvy. To sa dosiahne pridávaním neurónov do mapy na každej vrstve až pokiaľ sa nedosiahne vhodná veľkosť mapy. Mapy na každej úrovni rastú pokiaľ odchýlka neurónu predchádzajúcej vrstvy nie je zredukovaná aspoň na τ_m percent. Teda čím menší parameter τ_m , tým väčšia bude výsledná mapa. Pokiaľ platí pre vrstvu m podmienka:

$$MQE_m \geq \tau_m \, . \, mqe_0$$

pridáva sa k mape nový riadok alebo nový stĺpec neurónov. Vloží sa tak, aby susedil s chybovým neurónom e (neurón, ktorý má najväčšiu chybu). To, či sa pridá nový riadok alebo nový stĺpec závisí od toho, kde sa nachádza neurón, ktorý je najmenej podobný s chybovým neurónom. Miera podobnosti sa určuje vo vstupnom priestore. Hodnota vektorov váh nových neurónov sa nastaví na priemer hodnôt váh susedov. Po vložení sa učiaci parameter alfa a funkcia susednosti h_{ci} nastaví na počiatočnú hodnotu a učiaci proces pokračuje ďalej.

5. Akonáhle skončí učenie mapy prvej úrovne, teda:

$$MQE_m < \tau_m \cdot mqe_0$$

zisťuje sa, ktoré neuróny je potrebné expandovať do ďalšej mapy. Sú to neuróny, ktoré majú veľkú kvadratickú chybu, tá sa porovnáva s kvadratickou chybou mapy 0. tej úrovne (mqe_0). Na určenie jemnosti (granularity) rozlíšenia vstupných dát v koncových mapách. Každý neurón *i*, vyhovujúci tejto podmienke, bude následne expandovaný:

$$mqe_i > \tau_u \cdot mqe_0$$

6. Učiaci proces pokračuje ďalej pre každú novovytvorenú mapu tak isto, ako v predchádzajúcich krokoch. Jediným rozdielom je, že do mapy vstupuje už iba zlomok vstupných dát prislúchajúci neurónu nadradenej mapy a chyba mqe_0, ktorá určuje rast mapy je nahradená chybou nadradeného neurónu (teda podmapa sa snaží minimalizovať chybu nadradeného neurónu).
7. Učiaci proces GHSOM končí, keď už žiadny neurón nie je potrebné expandovať, resp. sa dosiahne definovaná maximálna hĺbka hierarchie.

Parameter τ_u určuje celkovú úroveň detailu (jemnosti) rozlíšenia vstupných dát. Čím menší je tento parameter, tým viac hierarchických úrovní sa vytvorí. Parameter τ_m určuje úroveň detailu rozlíšenia vstupných dát na konkrétnej mape. Čím je parameter τ_m menší, tým väčšie mapy vzniknú na jednotlivých úrovniach (a hierarchia bude „plytšia“).

4 Distribuovaný algoritmus a jeho implementácia

Distribuovaný algoritmus GHSOM je implementovaný v jazyku Java 1.5 ako služba systému GridMiner [6] na gridovej vrstve tohto systému s využitím knižnice JBowl [5]. Implementácia zahŕňa okrem samotného distribuovaného algoritmu aj proces predspracovania textových dokumentov a ponúka možnosť vizualizácie výsledného modelu. Proces zhlukovania dokumentov pomocou algoritmu GHSOM môže pri

analýze veľkého množstva dokumentov predstavovať vážny problém z časového hľadiska. Motiváciou tohoto návrhu bolo znížiť jeho časovú náročnosť. Algoritmus GHSOM je ideálny kandidát pre paralelizáciu a distribúciu na Gride. Po vytvorení mapy GSOM 1. úrovne vzniká niekoľko samostatných zhlukovacích procesov – budovanie hierarchických podstromov GHSOM pozostávajúcich z hierarchicky usporiadaných máp GSOM. Základnou myšlienkou návrhu je paralelné vykonávanie týchto zhlukovacích procesov na uzloch Gridu. Postup je potom možné popísať nasledovne [7]:

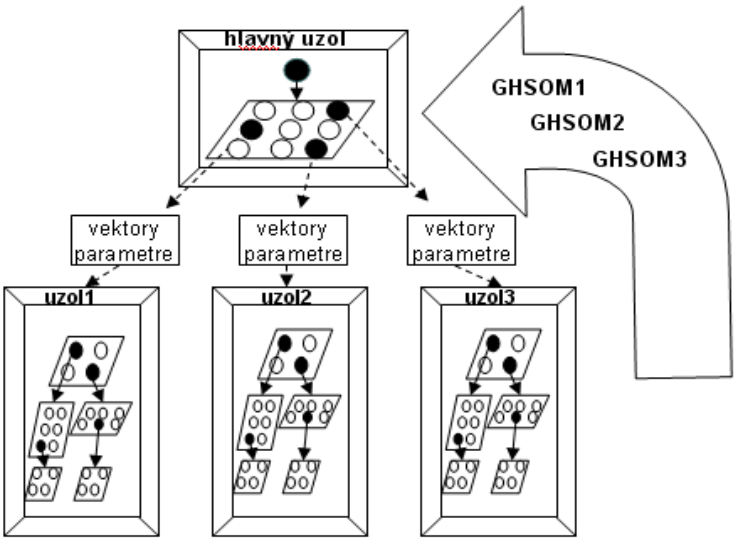
1. Na hlavnom uzle sa vyráta celková odchýlka vstupných dát, vytvorí sa mapa 1. úrovne a označia sa tie neuróny, ktoré spĺňajú podmienku expanzie do podmapy: expanduje sa ten neurón, ktorého stredná kvadratická chyba je väčšia ako τ_u násobok strednej kvadratickej chyby mapy prvej úrovne a zároveň počet vektorov prislúchajúci danému neurónu je väčší ako definovaný minimálny počet vektorov nutných pre vytvorenie ďalšej podmapy. Z množiny vstupných vektorov sa následne vyberú vektory prislúchajúce neurónom, ktoré je potrebné expandovať. Tieto vektory sa rozpošlú na jednotlivé uzly Gridu.
2. Na uzloch Gridu sa vektory stanú vstupom pre algoritmus GHSOM, ktorý vytvorí celý hierarchický podstrom. Keď sa na uzloch Gridu splní podmienka ukončenia algoritmu GHSOM, teda dosiahne sa definovaná maximálna hĺbka hierarchie alebo sa už nenašiel neurón, ktorý spĺňa podmienku expanzie, odošle sa celý čiastkový model GHSOM naspäť na hlavný uzol.
3. Na hlavnom uzle sa prijaté čiastkové modely GHSOM spoja do jedného výsledného hierarchického modelu GHSOM.

Schéma distribúcie je znázornená na Obr.1. Vstupom prvej fázy je kolekcia dokumentov, na ktorých najskôr prebehne lexikálna analýza – tokenizácia, ktorá sa realizuje pomocou štandardného tokenizéru (dokáže rozlíšiť viacero prvkov v texte ako čísla, slová, dátumy, akronymy, adresy elektronickej pošty, interpunkciu). Následne sa konvertujú všetky písmená na malé pomocou „Lower Case Filtra“. Po tomto kroku sa vykoná úprava slov na základný tvar (stemming). Posledným krokom prvej fázy je odstránenie nevýznamových slov aplikáciu „Stop filtra“. Odstránia sa tie slová, ktoré sa nachádzajú v zozname slov filtra. Výstupom prvej fázy sú štatistiky, frekvencie výskytu a slovník termov, ktoré sa uložia do samostatných súborov.

Vstupom druhej fázy sú potrebné štatistiky a dokumentové frekvencie indexovaných termov. Na ich základe sa uskutoční proces orezania termov (voľba termov) pomocou frekvenčného filtra balíka JBowl. Užívateľ určí hranice pre minimálnu a maximálnu povolenú dokumentovú frekvenciu termu. Ten atribút, ktorý nepatrí svojou dokumentovou frekvenciou do tohto intervalu, nie je pre vektorovú reprezentáciu použitý. Následne sa pomocou termov, ktoré ostali, vytvorí vektorová reprezentácia (váhovanie na základe tfidf schémy) a výsledné vektory sa normalizujú. Normalizované vektory sa uložia špeciálnym spôsobom (ukladajú sa len nenulové prvky tak, aby bolo jasné, ktorému termu odpovedajú) do výsledného súboru, ktorý tak obsahuje tfidf profily dokumentov a je základným vstupom pre blok distribúcia spájanie. Zo vstupných vektorov dokumentov sa použitím tried knižnice JBowl vytvorí mapa GSOM prvej úrovne.

V ďalšom kroku sa vytvorí zoznam neurónov, ktoré spĺňajú podmienku expanzie do mapy nižšej úrovne. S využitím metód systému GridMiner sa následne zistí počet

momentálne dostupných uzlov Gridu a vytvorí sa zoznam ich ukazovateľov. Počet uzlov Gridu je kľúčovým pri vytváraní fronty úloh uzlov.



Obr. 1. Schéma distribúcie algoritmu GHSOM v prostredí Gridu

Každá zhlukovacia úloha obsahuje identifikátor neurónu v rámci 1. mapy, zoznam (identifikátorov) vektorov namapovaných na daný neurón a parametre algoritmu GHSOM: tau1(τ_u), tau2 (τ_m), minimálny počet vektorov potrebný pre vytvorenie ďalšej podmapy, maximálna hĺbka hierachie. Priradzovanie zhlukovacích úloh uzlom Gridu prebieha nasledovne:

Nech n je počet expandovateľných neurónov a u nech je počet dostupných uzlov Gridu, potom

1. Ak n ≤ u, do fronty úloh prvých n uzlov sa priradí práve jedna zhlukovacia úloha.
2. Ak n > u, v prvej iterácii sa prvých u zhlukovacích úloh priradí prvým u uzlom, v nasledovných iteráciách sa priradzujú zvyšné zhlukovacie úlohy dovtedy, kým už nie je čo priradzovať.

Po naplnení fronty úloh, sa postupne z týchto front rozposielajú zhlukovacie úlohy na jednotlivé uzly Gridu. Na uzloch Gridu sa zo vstupných vektorov vytvoria čiastkové hierarchické modely GHSOM použitím definovaných parametrov. Po vytvorení modelu sa tento model odošle späť na hlavný uzol, kde sa uloží a následne sa na ten istý uzol Gridu odošle nasledujúca zhlukovacia úloha z fronty úloh (ak je fronta prázdna, výpočet na danom uzle končí). Keď sa vyprázdnia všetky fronty úloh a sú prijaté všetky čiastkové hierarchické modely GHSOM, realizuje sa spájanie týchto modelov do výsledného hierarchického modelu. Spájanie modelov prebieha nasledovne: referencie na rodiča mapy uzla 1. úrovne čiastkového modelu sa nastavia tak, aby odkazovali na mapu 1. úrovne vytvorenej na začiatku procesu. Následne sa zmenia referencie na „deti“ mapy uzla 1.úrovne a to tak, aby ukazovali na mapy uzlov 1. úrovne čiastkových modelov. Po tejto zmene sa uloží výsledný model GHSOM ako perzistentný serializovaný Java objekt do súboru.

5 Experimenty

Cieľom experimentov bolo porovnať časovú náročnosť klasického – sekvenčného algoritmu GHSOM s jeho distribuovanou verziou. Experimenty sme realizovali v prostredí, ktoré nebolo izolované (na použité servery a pracovné stanice mal prístup prakticky ktokoľvek a kedykoľvek) a bežali na ňom aj iné služby. Kvôli získaniu čo najpresnejšieho času výpočtu sme každý experiment pre tie isté nastavenia parametrov algoritmu GHSOM opakovali tri krát. Výsledný čas sme určili ako priemer časov získaných pri týchto opakovaných experimentoch. V niektorých prípadoch nastala situácia, že výsledný čas pre jedno opakovanie bol výrazne vyšší ako zvyšné dva časy („outlier"), to mohlo byť spôsobené náhlym vyťažením procesora inou službou servera resp. užívateľom. Takýto čas sme potom nahradili priemerom zvyšných dvoch časov. V experimentoch sa menil počet uzlov a parameter *tau1*, ktorý riadi výslednú kvalitu zhlukov, čím ovplyvňuje veľkosť vytváraných máp GSOM a celkovú hĺbku hierarchie modelu GHSOM. Čím menší je parameter *tau1*, tým väčšie mapy sa vytvárajú a hierarchia je plytšia. A naopak, čím je tento parameter väčší, tým menšie sú mapy a výsledná hierarchia je hlbšia.

Distribuovaná verzia bola testovaná v testovacom gridovom prostredí pozostávajúceho zo servera (4 x UltraSPARC-III 750MHz, 8GB RAM), ktorý predstavoval hlavný uzol Gridu a šiestich pracovných staníc SUN, ktorých konfigurácia je v nasledujúcej tabuľke. Uzly Gridu boli prepojené sieťou 100Mbit/s. Experimenty sme realizovali na dvoch kolekciách textových dokumentov: Times60 (420 dokumentov) a Reuters-21578 (12 902 dokumentov).

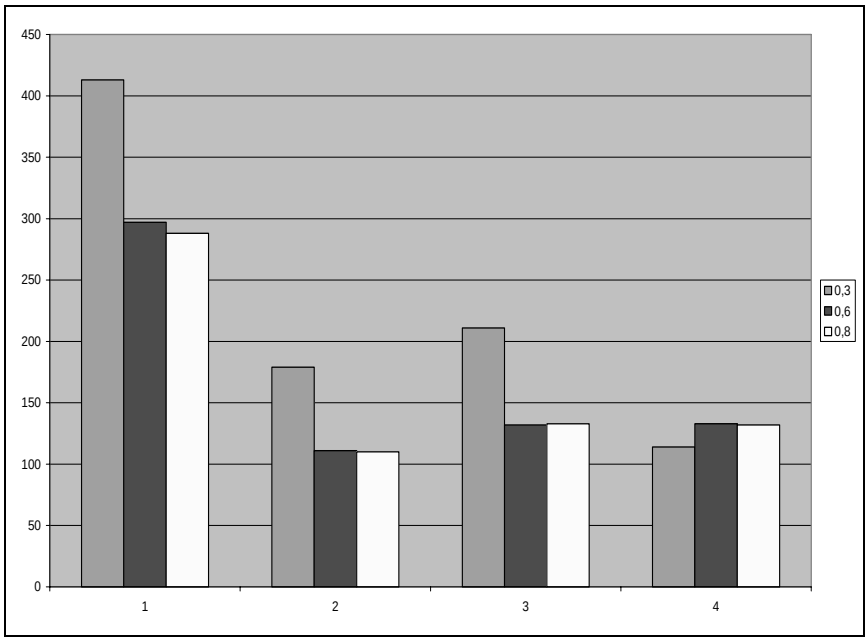
Times60 je kolekcia dokumentov – článkov magazínu Time zo šesťdesiatych rokov minulého storočia. Obsahuje 420 dokumentov. Články sa týkajú medzinárodných vzťahov, ekonomickej situácie a histórie krajín ako Rusko, Veľká Británia, Francúzsko, Malajzia, Egypt a Sýria. Hranica minimálnej dokumentovej frekvencie termov bola nastavená na 10 a maximálna na 100. Tomuto kritériu vyhovovalo 1925 termov.

Kolekcia dát Reuters[6] je štandardný voľne dostupný korpus textových dokumentov. Obsahuje 12,902 dokumentov. Každý dokument je novým článkom s určitou témou, väčšinou ekonomického charakteru. Pre účel nášho testovania boli vybrané dokumenty z roku 1987, ich počet bol 7769. Pri tejto textovej kolekcii sme nastavili hranicu minimálnej dokumentovej frekvencie termov na 20 a maximálnej na 100. Veľkosť vstupných vektorov dokumentov potom bola 1362.

5.1 Dáta Times

Experimenty na dátach Times na výpočtovom Gride sme vykonali s nastaveniami parametra tau1 0.3, 0.6 a 0.8. Najskôr sme testovali sekvenčnú verziu algoritmu na jednom uzle Gridu. S nastavením parametra tau1 0.3 sme potom testovali distribuovanú verziu postupne pre 2, 3, 4, 5 a 6 uzlov Gridu (z mapy 1. úrovne sa expandovalo 12 zhlukov). Pre parametre tau1 0.6 a 0.8 sme testovali na 2, 3 a 4 uzloch Gridu, keďže z mapy 1. úrovne sa expandovali 4 neuróny.

[6] http://www.daviddlewis.com/resources/testcollections/reuters21578



Obr. 2. Graf dosiahnutých časov výpočtu (v sekundách, os y) na dátach Times60 pre rôzny počet použitých uzlov (os x) Gridu pri rôznych hodnotách parametra tau1 (viď. Legenda – hodnoty 0.3, 0.6, 0.8).

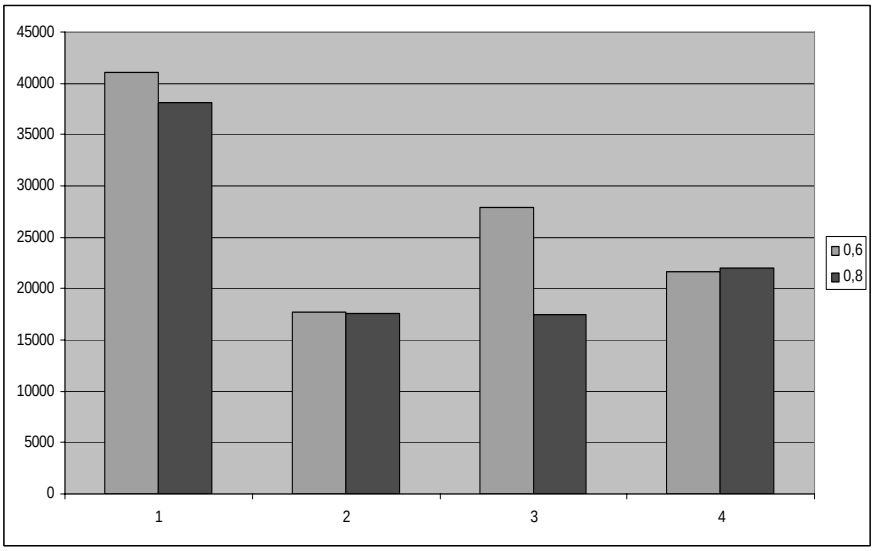
Časové zefektívnenie algoritmu s parametrom tau1 0.3 predstavovalo pri výpočte na dvoch uzloch 59%, na troch uzloch 50%, na štyroch uzloch 73%, na piatich uzloch 70% a na šiestich uzloch 66%. Časové zefektívnenie algoritmu s parametrom tau1 0.3 a použitím optimalizácie predstavovalo pri výpočte na dvoch uzloch 66%, na troch uzloch 73%, na štyroch uzloch 76%, na piatich uzloch 71% a na šiestich uzloch 76%. Časové zefektívnenie algoritmu s parametrom tau1 0.6 predstavovalo pri výpočte na dvoch uzloch 63%, na troch uzloch 55% a na štyroch uzloch 55%. Časové zefektívnenie algoritmu s parametrom tau1 0.6 a použitím optimalizácie predstavovalo pri výpočte na dvoch uzloch 63%, na troch uzloch 67% a na štyroch uzloch 56%. Grafické znázornenie pre 1 až 4 uzly Gridu je na na Obr.2.

5.2 Dáta Reuters

Pri experimentoch na kolekcii Reuters sme použili hodnoty parametra tau1 0.6 a 0.8. Obdobne ako pri testoch na dátach Times sme najskôr testovali sekvenčnú verziu algoritmu na jednom uzle a následne distribuovanú verziu postupne na dvoch, troch a štyroch uzloch Gridu.

Časové zefektívnenie algoritmu s parametrom tau1 0.6 predstavovalo pri výpočte na dvoch uzloch 56%, na troch uzloch 58% a na štyroch uzloch 49%. Časové zefektívnenie algoritmu s parametrom tau1 0.6 a použitím optimalizácie predstavovalo pri výpočte na dvoch uzloch 57%, na troch uzloch 55% a na štyroch uzloch 47%. Časové zefektívnenie algoritmu s parametrom tau1 0.8 predstavovalo pri výpočte na dvoch

uzloch 53%, na troch uzloch 55% a na štyroch uzloch 43%. Časové zefektívnenie algoritmu s parametrom tau1 0.6 a použitím optimalizácie predstavovalo pri výpočte na dvoch uzloch 57%, na troch uzloch 59% a na štyroch uzloch 42%. Grafické znázornenie je na Obr.3.



Obr. 3. Graf dosiahnutých časov výpočtu (v sekundách, os y) na dátach Reuters pre rôzny počet použitých uzlov (os x) Gridu pri rôznych hodnotách parametra tau1 (viď. Legenda – hodnoty 0.6, 0.8).

6 Záver

Hlavnou úlohou tejto práce bolo analyzovať algoritmus GHSOM z hľadiska možnosti jeho distribúcie v prostredí výpočtového Gridu a implementovať jeho distribuovanú verziu. Výsledky ukazujú celkové urýchlenie algoritmu GHSOM pri jeho distribuovanej verzii, ako aj to, že urýchlenie nebolo vždy plynulé, t.j. nie vždy vyšší počet uzlov znamenal urýchlenie oproti nižšiemu počtu uzlov. Jednou z príčin je nerovnomerné rozdelenie dát na jednotlivé uzly a tiež veľkosť chyby, ktorú daný uzol znižoval. Ďalšou príčinou bol rôzny výpočtový výkon jednotlivých uzlov testovacieho Gridu (frekvencia procesorov na prvých dvoch uzloch bola o 500MHz vyššia ako zvyšných uzlov.). K nerovnomernej časovej závislosti mohla prispieť aj náhodná inicializácia váh neurónov. Jednoduchá optimalizácia, ktorú sme navrhli za účelom rovnomerného zaťaženia uzlov, neprinášala vždy zlepšenie oproti neoptimalizovanej distribúcii. Ďalšou možnosťou ako rovnomernejšie a efektívnejšie využívať uzly Gridu, by mohla byť taká distribúcia algoritmu GHSOM, pri ktorej by sa na uzloch Gridu vytvárali iba mapy GSOM a nie celé podstromy GHSOM. Takýto spôsob by síce zvýšil komunikačné nároky a tým aj nároky na priepustnosť počítačovej siete, ale zároveň by umožnil využiť výpočtový výkon uzlov, ktoré po ukončení výpočtu mrha-

jú svojím výpočtovým časom pri čakaní na pomalšie uzly. Kritickým miestom pri distribúcii algoritmu GHSOM je vytváranie mapy 1.úrovne. Tento proces môže pri určitých dátach a nastaveniach algoritmu predstavovať viac ako 50% celkového času výpočtu. Kombinácia paralelizácie vytvárania mapy GSOM 1. úrovne (napr. na počítačovom klastri) a následná distribúcia zhlukov tejto mapy na uzly Gridu by mohla priniesť veľmi zaujímavé výsledky z hľadiska celkového urýchlenia algoritmu.

Práca prezentovaná v tomto príspevku vznikla za podpory Vedeckej grantovej agentúry Ministerstva školstva SR a Slovenskej akadémie vied v rámci projektu VEGA č.1/4074/07 s názvom "Metódy anotovania, vyhľadávania, tvorby a sprístupňovania znalostí s využitím metadát pre sémantický popis znalostí" a za podpory Agentúry na podporu výskumu a vývoja na základe zmlúv č. RPEU-0011-06 (projekt PoZnaŤ) a č. APVV-0391-06 (projekt SEMCO-WS).

Literatúra

1. Foster I., Kesselman, C.: Computional Grids, The Grid – Blueprint for a new Computing Infrastructure. Morgan Kaufmann, 1999.
2. Berman, F., Fox, G.: Grid Computing, Making the global Infrastructure a Reality. Wiley, 2003.
3. Ditttenbach, M. Rauber, A. Merkl, D.: The Growing Hierarchical Self-Organizing Map. Proceedings of International Joint Conference on Neural Networks, Como, Taliansko, 2000.
4. Kohonen, T.: Self-organizing maps. Springer-Verlag, Berlín, 1995.
5. Bednar, P., Butka, P., Paralic, J.: Java Library for Support of Text Mining and Retrieval. In Proceedings of Znalosti 2005, pp. 162-169, Stará Lesná, 2005.
6. Brezany, I., Janciak, A., Sarnovsky, M.: Text mining within the GridMiner Framework, 2nd Dialogue Workshop, Edinburg, 2006.
7. Safko, V.: Distribuované zhlukovanie textových dokumentov v prostredí Gridu, Diplomová práca, KKUI FEI TU Košice, 2007.
8. Rea, B., Ananiadou, S.: Text Mining Services to Support e-Research, UK e- Science All Hands Meeting, Nottingham, UK, September 2007.

Annotation:

Distributed clustering of textual documents in the Grid environment

This work deals with the clustering of the textual documents, clustering algorithms and its potential distribution in Grid computing environment. GHSOM algorithm (Growing Hierarchical Self Organizing Map) is an extension of the standard approach to clustering based on SOM algorithm, which combines the adaptability of map extension with hierarchical clustering by creation of new maps. Basically it means that each layer of hierarchical structure consists of the set of independent maps, which adapts its size with respect to requirements set by input data (examples that belongs to particular map). In this paper we propose the distributed GHSOM algorithm ant its implementation in the integrated distributed text-mining environment on top of the Grid – GridMiner. Paper also presents the results and experiments description obtained with the distributed version of the algorithm that were executed on the computational Grid.