Klasteriranje tekstualnih dokumenata

Valentina Križ, Lucija Valentić, Jelena Kurilić

1. Uvod

Klasteriranje kolekcija dokumenata s obzirom na njihovu sličnost je dobro poznati problem. Uobičajeni način pristupa tom problemu je izdvajanje kontekstnih riječi iz kolekcije dokumenata i reprezentacija svakog dokumenta pomoću vektora koji predstavlja pojavu kontekstnih riječi u dokumentu.
Algoritam koji se često koristi za klasteriranje kolekcija dokumenata je algoritam k - sredina. Klasični algoritam k - sredina koristi Euklidsku udaljenost koja često nije prikladna za klasteriranje dokumenata.
Bolja mjera sličnosti između dokumenata je kosinusova sličnost; algoritam k - sredina u kojem se koristi ta mjera naziva se sferne k - sredine.
Jedna od mana algoritma sfernih k - sredina je da često zaglavi u lokalnom optimum posebice kod klastera malih veličina. S ciljem rješenja tog problema u radu [1] je predložen algoritam koji se sastoji od dvije faze: (a) sferne k - sredine i (b) prva varijacija. U drugoj fazi se odabire dokument čijim se premještanjem u drugi klaster funkcija cilja povećava. Upravo taj korak omogućava da se izbjegne lokalni optimum i da se dokumenti bolje klasteriraju. Ovaj ping-pong algoritam dovodi do poboljšane k - sredine koja uvijek dovodi do bolje funkcije cilja od sferne k - sredine.
U ovom radu predstavit ćemo našu implementaciju sferne k - sredine i ping-pong algoritma predloženog u radu [1] te dobivene rezultate na istim testnim podacima.

2. Sferne k - sredine

Neka je d broj dokumenta, w broj riječi i
$X = \{x_1, x_2, ..., x_d\}$ skup ne-negativnih vektora koji predstavljaju dokumente, za svaki $x_i \in R^w$ i svaki x_i leži na jediničnoj sferi. Klasteriranje kolekcije dokumenata je particioniranje u disjunktne skupove $\pi_1, \pi_2, ..., \pi_k$ takve da $X = \bigcup_{j=1}^{k} \pi_j$ i $\pi_j \cap \pi_i, j \neq i$. Za klaster π definiramo sumu $s(\pi) = \sum_{x \in \pi} x$. Centroid klastera π je definiran kao $c(\pi) = \frac{s(\pi)}{\|s(\pi)\|}$. Kvalitetu nepraznog klastera π definiramo kao $q(\pi) = \sum_{x \in \pi} x^T c(\pi) = \|s(\pi)\|$, gdje je $q(\emptyset) = 0$. Za particiju $\{\pi_j\}_{j=1}^k$ definiramo funkciju cilja kao zbroj kvaliteta pojedinih klastera:

$$Q(\{\pi_j\}_{j=1}^k) = \sum_{j=1}^{k} q(\pi_j).$$

Cilj je pronaći klasteriranje koje maksimizira vrijednost navedene funkcije cilja. Algoritam sferne k - sredine je iterativni proces koji generira niz particija takvih da vrijedi

$$Q(\{\pi_j^{(t+1)}\}_{j=1}^k) \geqslant Q(\{\pi_j^{(t)}\}_{j=1}^k).$$

3. Refinement algoritam

Sada opisujemo algoritam koji je predložen u članku i koji rješava problem zaglavljivanja u lokalnom optimumu.

Definicija 1. *Prva varijacija neke particije $\{\pi_l\}_{l=1}^k$ je particija $\{\pi_l^{'}\}_{l=1}^k$ koja je nastala tako da se jedan vektor x iz klastera π_i particije $\{\pi_l\}_{l=1}^k$ prebacio u klaster π_j particije $\{\pi_l\}_{l=1}^k$.*

U skupu svih prvih varijacija particije $\{\pi_l\}_{l=1}^k$ tražimo onu koja ima "najveći uspon", tj. tražimo $nextFV(\{\pi_l\}_{l=1}^k)$ t.d.

$$Q(nextFV(\{\pi_l\}_{l=1}^k)) \geqslant Q(\{\pi_l^{'}\}_{l=1}^k)$$

za svaku prvu varijaciju $\{\pi_l^{'}\}_{l=1}^k$.
Algoritam prvih varijacija generira niz particija $\{\pi_j^{(t)}\}_{j=1}^k, t \geqslant 0$ takve da

$$\{\pi_j^{(t+1)}\}_{j=1}^k = nextFV(\{\pi_j^{(t)}\}_{j=1}^k), t = 0, 1, ...$$

Algoriram prvih varijacija:

Neka je $tol > 0$. Radi sljedeće:

1. Započni s particijom $\{\pi_j^{(0)}\}_{j=1}^k$.
   Postavi indeks iteracije $t = 0$.

2. Izgeneriraj $\{\pi_j^{(t+1)}\}_{j=1}^k = nextFV(\{\pi_j^{(t)}\}_{j=1}^k)$.

3. Ako je $Q(nextFV(\{\pi_j^{(t)}\}_{j=1}^k)) - Q(\{\pi_j^{(t)}\}_{j=1}^k)$ veće od tol, povećaj t za 1 i odi na korak 2. Inače stani s algoritmom.

Jedna iteracija algoritma prvih iteracija dovodi do povećanja funkcije cilja. Kako bi još poboljšali algoritam možemo proširiti lokalnu pretragu tako da potražimo lanac koraka umjesto samo jednog. Ovo ideju implemetiramo pomoću Kernighan-Lin heuristike.
Prvo generiramo korak prve varijacije, zabilježimo poboljšanje funkcije cilja i označimo vektor koji smo premjestili kako ga u idućim iteracijama ne bi više premještali. Ponavljamo ovaj korak f puta, gdje je f parametar odabran od stane korisnika. Na kraju uzimamo onaj broj iteracija koji je doveo do maksimalnog povećanja funkcije cilja.

Prednost algoritma prvih variacija je da se njegovom primjenom uvijek povećava vrijednost funkcije cilja, ali najčešće je ta promjena mala. K - sredine obično dovode do većeg povećanja vrijednosti funkcije cilja. Kako bi postigli što bolje rezultate kombiniramo ova dva algoritma. Algoritam se sastoji od dva koraka koja se ponavljaju sve dok razlika u funkciji cilja između ta dva koraka ne bude premala (manja od tol). U prvom koraku se primjenjuju sferne k - sredine sve dok algoritam k - sredine ne stane. Tada se prelazi na 2. korak u kojemu se na dobivenu particiju primjenjuje Kernighan-Lin heuristika.

Kernighan-Lin heuristika:

Neka je $tol > 0$. Radi sljedeće:

1. Započni s particijom $\{\pi_j^{(0)}\}_{j=1}^k$.
   Postavi indeks iteracije $t = 0$.

2. Postavi $j = 0$, U = X i napravi f puta sljedeće:

   (a) Postavi $\{\pi_j^{(t+j+1)}\}_{j=1}^k = nextFV(\{\pi_j^{(t+j)}\}_{j=1}^k, U)$.
   Označi vektor kao pomaknut
   i makni ga iz U.

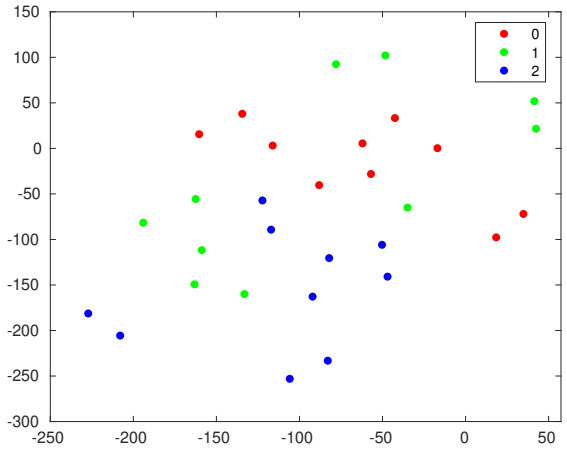
   (b) Zabilježi Q(nextFV($\{\pi_j^{(t+j)}\}_{j=1}^k), U) - Q(\{\pi_j^{(t+j)}\}_{j=1}^k)$ u `ObjChange[j]`. Inkrementiraj j za 1.

3. Postavi `MaxChange` = $max_i \sum_{j=0}^{i}$ `ObjChange[j]` i `MaxI` = $\text{arg}max_i \sum_{j=0}^{i}$ `ObjChange[j]`, $i = 0$, 1..., f - 1.

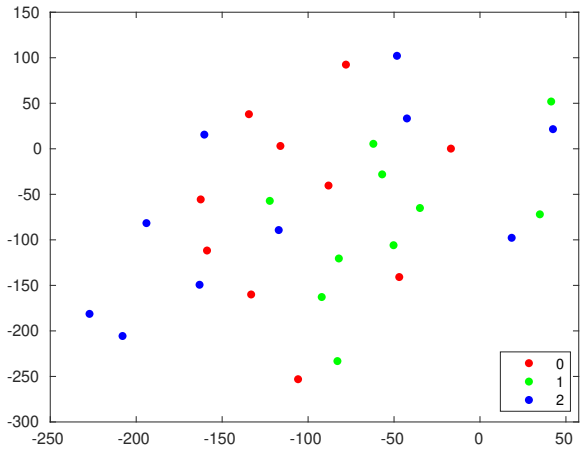
4. Ako `MaxChange` $\geq$ `tol`, inkrementiraj t sa `MaxI+1` i odi na korak 2. Inače, vrati particiju $\{\pi_j^{(t)}\}_{j=1}^k$ i `MaxI`, i stani s algoritmom.

4. Rezultati

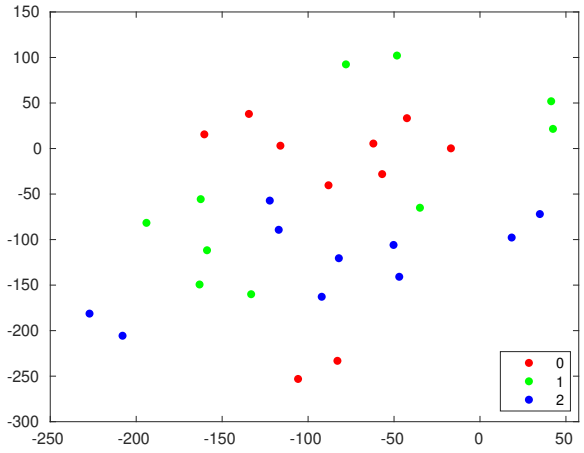
Implementaciju našeg algoritma testirali smo na kolekciji dokumenata koja se sastoji od MEDLINE, CISI i CRANFIELD kolekcije dokumenata. MEDLINE se sastoji od 1033, CISI 1460 i CRANFIELD od 1400 dokumenata. Prvo smo kreirali kolekciju od 30 dokumenata koji sadrži po 10 dokumenta iz svake skupine. Zatim smo na tom kolekcijom prvo primijenili sferne k - sredine pa refinement algoritam. Korišteni parameti $f = 50, tol = 0.0001$. Slika 1 prikazuje točnu podjelu dokumenata, Slika 2 nakon sferne k - sredine i Slika 3 nakon refinement algoritma.



Slika 1: Točna podjela



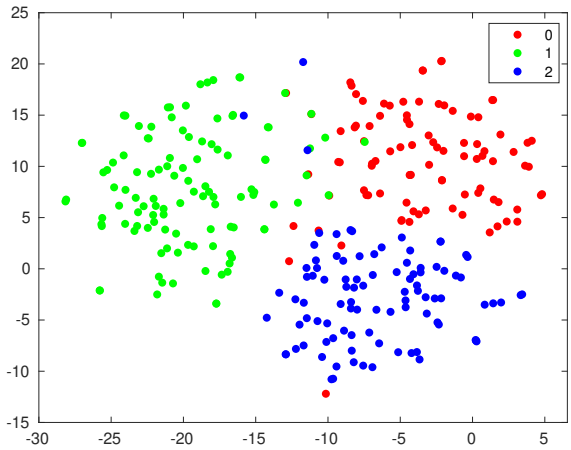
Slika 2: Nakon sferne k - sredine
$(d = 30, tol = 0.0001)$



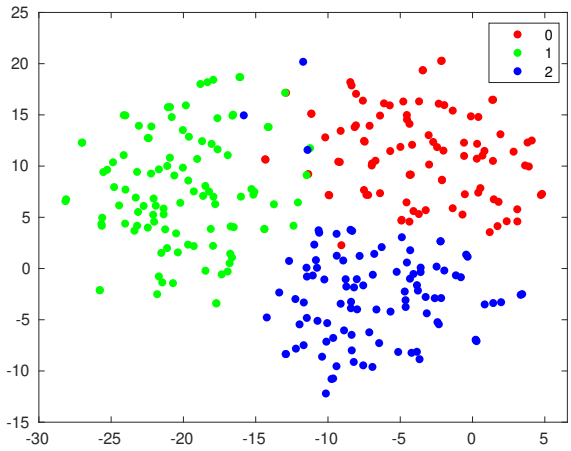
Slika 3: Nakon refinement algoritma
$(d = 30, f = 50, tol = 0.0001)$

Vrijednosti funkcije cilja nakon sferne k - sredine je 10.1566, a nakon refinement algoritma 10.9422. Iz slika možemo vidjeti da sferne k - sredine kod ovako malog broja dokumenata nisu napravile dobru podjelu. Refinement algoritam je povećao funkciju cilja i doveo do zadovoljavajućeg rezultata, većinu dokumenata je grupirao u odgovarajuće klastere.
Zatim smo testirale naš algoritam na kolekciji od 300 dokumenata iz svake kolekcije po 100. Također smo napravile sferne k - sredine i refinement algoritam. Parametre koje smo koristili $f = 3, tol = 0.001$.

Slika 4 prikazuje stanje nakon sferne k - sredine i Slika 5 nakon refinement algoritma.



Slika 4: Nakon sferne k - sredine
$(d = 300, tol = 0.001)$



Slika 5: Nakon refinement algoritma
$(d = 300, f = 3, tol = 0.001)$

Vrijednost funkcije cilja nakon sferne k - sredine je 72.9169, dok nakon refinement algoritma 73.1164. Vidimo da su već k - sredine dosta dobro klasterirale dokumente, što nije bio slučaj za $d = 30$. Primjenom refinement algoritma funkcija cilja je povećala i dobili smo još bolje klasteriranje dokumenata.

5. Zaključak

U ovome radu prezentirali smo poboljšani algoritam dobiven korištenjem lokalnog traženja nakon sfernih k-sredina. Pokazali smo da razlika u kvaliteti rješenja može biti značajna, pogotovo kada se radi s visokodimenzionalnim, ali rijetkim podacima i kada je veličina klastera mala. Algoritam također daje bolja rješenja i na većim klasterima, iako već i same sferne k-sredine daju odlične rezultate.

Literatura

[1] J Kogan IS Dhillon, Y Guan. *Iterative Clustering of High Dimensional Text Data Augmented by Local Search.*