Gom Nhóm Văn Bản và Thuật Toán Áp Dụng

Nguyễn Lương Điền

Khoa Công nghệ thông tin

Trường Đại học Khoa học Tự nhiên

TP HCM, Việt Nam

[tom.nguyenluong@gmail.com](mailto:tom.nguyenluong@gmail.com)

Mai Khải Huy

Khoa Công nghệ thông tin

Trường Đại học Khoa học Tự nhiên

TP HCM, Việt Nam

[maikhaihuy@gmail.com](mailto:maikhaihuy@gmail.com)

Tóm tắt – Gom nhóm văn bản nhằm mục đích khám phá bản chất của các thông tin được ẩn chứa trong văn bản. Dữ liệu kiểu văn bản không giống như dữ liệu số hay hình ảnh, thường phải trải qua giai đoạn tiền xử lý rất công phu và cần chuyển đổi sang dạng thức phù hợp để khai thác, thường là Vector Space Model. Có 3 phương pháp phân cụm thường được sử dụng là connectivity-based clustering, centroid-based clustering, density-based clustering. Mỗi phương pháp đều có những thế mạnh riêng, mang lại nhiều giá trị ứng dụng trong thực tiễn.

Từ khóa – CLARA, OPTICS, Chameleon, tf-idf, Vector Space Model.

# Giới thiệu:

Gom nhóm là một bài toán điển hình trong Khai thác dữ liệu, thuộc về nhóm phương pháp *học không giám sát* (Unsupervised Learning) trong lĩnh vực Học máy. Khác với dữ liệu số và hình ảnh, dữ liệu dạng văn bản mang nhiều đặc trưng của con người hơn, do đó trong máy học phải có những phương pháp tiếp cận đặc biệt để xử lý và khai thác. Vector space model cho ta một phương thức chuyển đổi văn bản từ phi số sang dạng đặc trưng số, từ đó quy về các bài toán gom cụm truyền thống, tận dụng được các thành tựu nghiên cứu khoa học trong suốt quá trình dài trước đó, vừa làm tăng được tính tổng quát, vừa giúp giải quyết được bài toán một cách hiệu quả.

Trong bối cảnh tầm ảnh hưởng của học máy ngày càng trở nên to lớn trên nhiều mặt của đời sống xã hội, khoa học về xử lý và khai thác thông tin hứa hẹn cũng sẽ có được những bước phát triển vượt bậc, góp phần làm cho cuộc sống của con người ngày một tốt đẹp hơn.

# Vấn đề gom nhóm văn bản:

## Mục tiêu:

Mục tiêu của gom nhóm văn bản để phân nhóm các đoạn văn bản không cấu trúc. Mỗi văn bản có các đặc trưng về ngữ nghĩa, các văn bản trong cùng một nhóm sẽ có đặc trưng giống (tương tự) nhau. Sau quá trình gom nhóm, các đặc trưng này sẽ trở nên cụ thể hơn, từ đó làm cơ sở cho việc phân tích nội dung trên các văn bản khác.

Khi dữ liệu các văn bản được phân nhóm, quá trình tra cứu, quản lý số lượng rất nhiều các văn bản được lưu trữ lớn trên các hệ thống trở nên dễ dàng, nhanh chóng.

## Vấn đề xử lý ngôn ngữ tự nhiên

Dân số thế giới hiện nay vào khoảng 7 tỷ người, phân bố trên 5 châu lục, bao gồm hơn 200 quốc gia. Hầu hết mỗi quốc gia đều có một ngôn ngữ riêng cho mình, chưa kể đến sự phân hóa ngôn ngữ theo vùng miền trong chính quốc gia đó, kéo theo số lượng vô cùng lớn, lên đến hàng nghìn thứ tiếng. Hầu hết trong số chung được sử dụng một cách rất hạn chế, gần như bị tiêu vong. Phần lớn trong số chúng thậm chí còn chưa có chữ viết chính thức để sử dụng.

Chỉ có khoảng 500 ngôn ngữ đã được nghiên cứu một cách tương đối đầy đủ. Tuy nhiên đây cũng vẫn là một số lượng lớn. Mỗi ngôn ngữ có các đặc trưng riêng về từ ngữ và ngữ pháp, đòi hỏi những phương pháp tiếp cận đặc thù. Ví dụ như trong tiếng Anh, thường thấy các hiện tượng đảo ngữ, các biến thể của từ gốc tùy thuộc vào ngữ cảnh; trong tiếng Việt, các từ đơn đôi lúc lại không mang một ý nghĩa cụ thể chính xác nào, cần phải xem xét trong mối quan hệ với từ đơn khác để hiểu rõ được ý nghĩa. Điều này đặt ra thách thức không nhỏ trong khâu xử lý đối với khoa học máy tính.

Trong phạm vi tài liệu này, ngôn ngữ được sử dụng để khai thác là tiếng Anh. Đây là một ngôn ngữ phổ biến và đã được nghiên cứu rất sâu sắc. Để sử dụng được dữ liệu tiếng Anh vào khai thác dữ liệu, người ta đề xuất ra một số bước xử lý như sau, đã được kiểm chứng là có hiệu quả tốt:

* Loại bỏ stopword: stopword là những từ có tần số xuất hiện rất lớn trong tiếng Anh và thường có giá trị thấp về mặt ngữ nghĩa. Chúng có thể là các mạo từ, trạng từ, đại từ, giới từ, etc. Việc lại bỏ các từ này giúp giảm khối lượng công việc, làm tăng sự hiệu quả cho khâu khai thác.
* Stemming: Các từ tiếng Anh thường có rất nhiều biến thể, được sử dụng tùy theo ngữ cảnh khác nhau, tuy nhiên về mặt ý nghĩa thì lại tương tự nhau. Việc chuyển đổi các biến thể này về cùng một dạng gốc (origin) giúp việc xem xét sự phân bố và đóng góp về mặt nội dung của chúng được đầy đủ và chính xác hơn.

Ngoài ra, một số phương pháp xử lý tổng quát khác như tách từ, loại bỏ dấu câu và các kí tự đặc biệt cũng được sử dụng cho bước tiền xử lý văn bản.

## Vec-tơ đặc trưng:

Văn bản phi cấu trúc mang các tính chất gần với tự nhiên, gần với con người hơn, do đó việc áp dụng các phương pháp học máy để xử lý trên những đặc trưng nguyên thủy này là rất khó khăn, và thường mang lại hiệu quả không cao. Do đó, chuyển đổi văn bản phi cấu trúc sang một dạng đặc trưng khác trước khi thực hiện quá trình khai thác là một công việc vô cùng cần thiết. Để phân nhóm cho các văn bản, đầu tiên cần rút ra được các “mẫu đặc trưng” (từ khóa) trong các văn bản. Các từ khóa này thường được lựa chọn dựa theo định luật Zipf (tần số từ 3% - 50%), trong đó không bao gồm các stopword. Tập hợp các từ khóa này được gọi là Bag of Words.

Các văn bản sau đó sẽ được biểu diễn theo danh sách các từ khóa này. Có hai mô hình phổ biến thường được sử dụng:

* Mô hình Boolean: mỗi văn bản được tổ chức theo tập hợp các từ theo đúng vị trí trong Bag of Words.

Gọi *V* = {*t*0, *t*1,..., *tm*} là tập hợp các từ của tập các văn bản *D*.

Gọi *di* = {*w*0, *w*1…*wm*} (*di* thuộc *D*) là tập hợp các trọng số của văn bản *di*. *wk* có giá trị 0 khi văn bản *di* không chứa *tk*, có giá trị 1 nếu văn bản có chứa ­*tk*.

Mô hình boolean chỉ có giá trị về mặt định tính (*có* hoặc *không* tương ứng với 1 hoặc 0), do đó không thể hiện được giá trị định lượng về sự đóng góp vào nội dung của từ trong văn bản (tần số), làm cho việc xử lý vẫn gặp rất nhiều khó khăn và thiếu chính xác, vì thế mô hình này hầu như không được áp dụng vào thực tế.

* Vector space: tương tự như mô hình Boolean, tuy nhiên, các trọng số là các giá trị liên tục. Các giá trị trọng số thường được sử dụng là TF và TF-IDF:
  + TF (term frequency): giá trị thể hiện cho tần số xuất hiện của mẫu *t* trong văn bản *d*.
  + TF-IDF: giá trị được tính bằng tích của 2 hệ số TF và IDF. IDF (inverse document frequency) thể hiện cho giá trị nghịch đảo tần số văn bản *d* có chứa mẫu *t* trong tập hợp tất cả các văn bản được khai thác. Tích giữa 2 giá trị này là trọng số để biểu diễn văn bản trong không gian vector đặc trưng.
  + Ngoài cách tính trọng số như trên, người ta còn sử dụng nhiều biến thể tùy vào các mục đích khác nhau, được liệt kê trong bảng sau:

|  |  |
| --- | --- |
| **Term frequency (TF)** | **Document frequency (DF)** |
| tf*t*, *d* | n (no of document) |
|  |  |
|  |  |
|  |  |
|  |  |

Bảng 1 – các biến thể trong tính toán trọng số

## Các độ đo tương tự:

Để xác định 2 văn bản có cùng thuộc một nhóm hay không, ta cần một giá trị định lượng thể hiện cho mối quan hệ tương quan giữa chúng. Giá trị này được gọi là độ tương tự giữa 2 văn bản.

Sau khi đã biểu diễn văn bản dưới dạng vector hệ số đặc trưng tf-idf, một phương pháp tính sẽ được áp dụng dựa trên những giá trị này để xác định độ tương tự giữa 2 văn bản. Trong không gian, mối quan hệ giữa 2 vector thường được định lượng bằng khoảng cách hoặc góc giữa 2 vector đó.

1. Khoảng cách

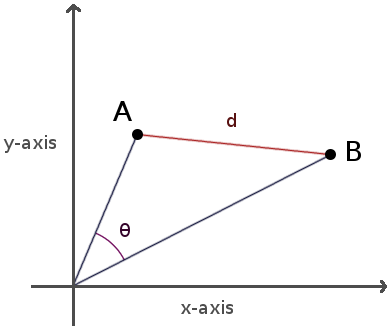
Độ đo thường được sử dụng là độ đo Euclidean. Khoảng cách giữa 2 vector được đo bằng độ dài khoảng cách giữa 2 điểm cuối tương ứng.



Phương pháp này chỉ có hiệu quả khi số chiều không gian không quá lớn. Trong trường hợp gom cụm văn bản, không gian vector thường có số chiều rất lớn, các điểm phân bố thưa thớt và cách xa nhau, dẫn đến khoảng cách giữa chúng cũng rất lớn, cho dù giữa chúng chỉ có sự khác biệt trên một số ít chiều nhất định.

1. Góc

Việc sử dụng khoảng cách không mang lại hiệu quả cao trong gom nhóm văn bản, thay vào đó người ta thường sử dụng góc giữa 2 vector để đo độ tương tự giữa chúng.



Hình 1 – Các độ đo tương tự giữa 2 vector (d–khoảng cách Euclide, ɵ-góc giữa 2 vector).

Trong trường hợp 2 vector có cùng phương (chỉ khác nhau về một thành phần nào đó), khoảng cách Euclidean giữa chúng có thể rất lớn trong khi chúng gần như tương tự nhau. Khi đó, sử dụng độ đo góc (trong trường hợp này có giá trị 0) giúp ta xác định được độ tương tự chính xác hơn hẳn. Độ đo thường được sử dụng là cosine.



Ngoài ra, còn có một số độ đo khác như Jaccard (theo xác suất), Hamming, Tanimoto nhưng ít được sử dụng.

# Các thuật toán gom nhóm:

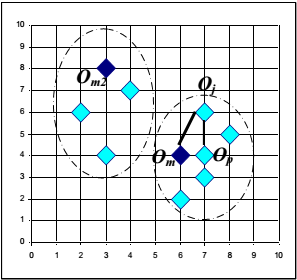
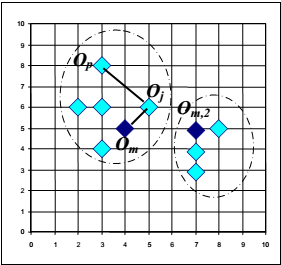
## CLARA (Clustering LARge Application)

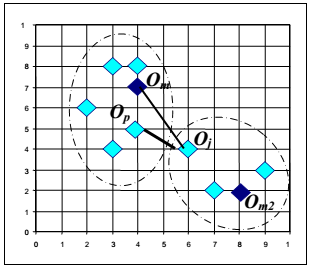
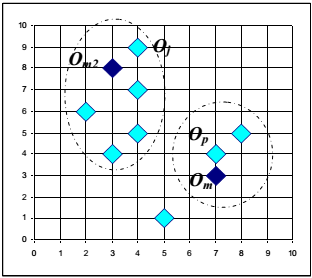
1. **Thuật toán PAM**

Thuật toán PAM (Partioning Around Medoids) được đề xuất năm 1987 bởi Kaufman và Rousseeuw [1] nhằm khắc phục điểm yếu của thuật toán k-means đối với dữ liệu nhiễu hoặc có các phần tử ngoại lai. Bằng cách sử dụng các đối tượng medoid (trung vị) làm đại diện phần tử trung tâm cho nhóm, PAM giảm thiểu được sự ảnh hưởng của các phần tử ngoại lai và nhiễu.

Ban đầu, tập medoid *M* được lựa chọn ngẫu nhiên trong tập dữ liệu. Sau mỗi bước lặp, PAM sẽ cố gắng hoán đổi giữa một đối tượng medoid *Om* với một đối tượng *Op* không phải là medoid, sao cho chất lượng của phân nhóm được cải tiến. Để xác định một hoán đổi như vậy, PAM sử dụng giá trị tổng chi phí hoán chuyển để làm căn cứ quyết định sự thay đổi.

Đặt *Om* là đối tượng medoid cần thay thế, *Op* là ứng viên thay thế, *Oj* là một đối tượng dữ liệu không phải medoid và *Om2* là medoid gần *Oj* nhất nhưng phải khác *Om*. Giá trị tổng chi phí hoán chuyển *Cjmp* được tính theo 4 trường hợp sau:





Hình 3 – Các trường hợp thay thế (a, b, c, d theo thứ tự từ trái sang phải, từ trên xuống dưới).

* *Oj* thuộc nhóm của *Om* và dist(*Oj*, *Op*) ≥ dist(*Oj*, *Om2*) (hình a): *Cjmp­* = dist(*Oj*, *Om2*) - dist(*Oj*, *Om*) > 0 (*Oj* chuyển sang nhóm của *Om2*).
* *Oj* thuộc nhóm của *Om* và dist(*Oj*, *Op*) < dist(*Oj*, *Om2*) (hình b): *Cjmp­* = dist(*Oj*, *Op*) - dist(*Oj*, *Om*). Giá trị *Cjmp­* có thể âm hoặc dương (*Oj* chuyển sang nhóm của *Op*).
* *Oj* thuộc nhóm của *Om2* và dist(*Oj*, *Op*) ≥ dist(*Oj*, *Om2*) (hình c): *Cjmp­* = 0 (do *Oj* không thay đổi nhóm).
* *Oj* thuộc nhóm của *Om2* và dist(*Oj*, *Op*) < dist(*Oj*, *Om2*) (hình d): *Cjmp­* = dist(*Oj*, *Op*) - dist(*Oj*, *Om2*) > 0 (*Oj* chuyển sang nhóm của *Op*).

Tổng *Cjmp*của tất cả các đối tượng trong tập dữ liệu ban đầu chính tổng chi phí hoán chuyển giữa *Om* và *Op*: . *TCmp* mang giá trị âm nghĩa là sự hoán chuyển sẽ cải thiện chất lượng gom nhóm.

**Thuật toán** PAM

**function** PAM(*D*, *k*)

*M* ← *randomSelection*(*D*, *k*)

Gom nhóm đối tượng dựa trên *M*

*change* ← true

**while** *change* **do**

*change* ← **false**

Tìm cặp (*Om*, *Op*) ứng với min*TCmp* nhỏ nhất

**if** min*TCmp* < 0**then**

*change* ← **true**

*M.*change(*Om*,*Op*)

Gom nhóm đối tượng dựa trên *M*

**return** *M*

Để tìm ra cặp (*Om*, *Op*) có chi phí hoán chuyển nhỏ nhất, thuật toán cần duyệt qua *k*(*n – k*) cặp, tính chi phí cho (*n – k*) đối tượng. Do đó, PAM có độ phức tạp trung bình là O(*k*(*n – k*)2). Dễ thấy với dữ liệu có kích thước lớn, PAM tỏ ra kém hiệu quả về mặt thời gian tính toán.

1. **Thuật toán CLARA**

Năm 1990, Kaufman và Rousseeuw [1] đã đề xuất ra thuật toán CLARA [1, 2], giúp khắc phục được điểm yếu của thuật toán PAM đối với dữ liệu kích thước lớn. Thuật toán được xây dựng theo hướng tiếp cận trích mẫu đại diện để xử lý đối với tập dữ liệu lớn. Thay vì tìm kiếm các medoid trên toàn bộ tập dữ liệu lớn, CLARA tiến hành trích một mẫu nhỏ trong tập này và thực hiện thuật toán PAM trên tập mẫu để tìm kiếm các medoid tối ưu.

Họ nhận định rằng, khi tập mẫu được trích một cách ngẫu nhiên thì tập medoid tối ưu cục bộ sẽ xấp xỉ với tập medoid tối ưu của toàn bộ dữ liệu ban đầu. Do đó, khi thực hiện trích mẫu ngẫu nhiên nhiều lần và lựa chọn tập medoid tốt nhất trong các lần thử sẽ cho ta kết quả xấp xỉ ngày càng chính xác.

Mức độ tối ưu giữa các tập medoid được so sánh dựa trên độ phi tương tự trung bình trên tập dữ liệu ban đầu. Độ đo này được tính như sau:



Với *M* là tập các medoid, *O* là tập *n* điểm dữ liệu ban đầu, hàm rep(*M*, *Oi*) trả về medoid gần với *Oi* nhất.

**Thuật toán** CLARA

**function** CLARA(*D*)

*minCost* ← ∞

*i* ← 0

**while** *i* < *q* **do**

Tạo tập *S* gồm *m* phần tử ngẫu nhiên từ *D*

*M* ← PAM(*S*, *k*)

**if** Cost(*M*, *D*) < *minCost* **then**

*minCost* ← Cost(*M*, *D*)

*bestM* ← *M*

*i ← i* + 1

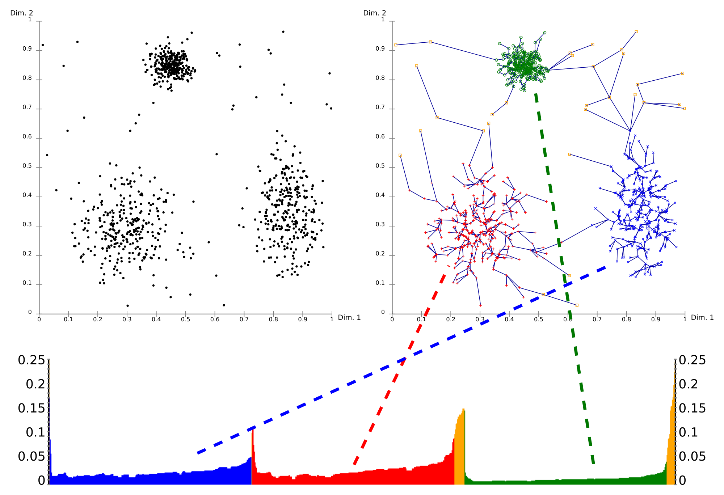
**return** *bestM*

Do đi theo hướng tiếp cận trích mẫu ngẫu nhiên, kết quả của thuật toán CLARA phụ thuộc rất lớn vào kích thước của tập con. Nếu tập con có kích thước lớn thì chất lượng gom nhóm nhìn chung sẽ tốt hơn, nhưng bù lại là chi phí tính toán lớn. Ngược lại với tập con nhỏ, thuật toán sẽ được thực hiện rất nhanh nhưng chất lượng gom nhóm sẽ thấp hơn. Kết quả thực nghiệm [2] cho thấy, thực hiện 5 lần lặp với kích thước tập mẫu là 40 + 2*k* (*k* là số lượng nhóm) sẽ cho kết quả phân lớp tốt.

Độ phức tạp của thuật toán CLARA là O(*k*(40 + *k*)2 + *k*(*n* – *k*)), thấp hơn so với thuật toán PAM là O(*k*(*n – k*)2, do đó CLARA có thể được áp dụng một cách hiệu quả lên tập dữ liệu có kích thước lớn.

## OPTICS:

OPTICS thuộc nhóm phương pháp phân lớp theo mật độ, gom nhóm dữ liệu với hình dạng bất kỳ. Được công bố năm 1999 vởi Mihael Ankerst, Markus M. Breunig, [Hans-Peter Kriegel](http://en.wikipedia.org/wiki/Hans-Peter_Kriegel) and Jörg Sander. Ý tưởng của OPTICS gần giống như DBSCAN, tìm các đối tượng mà số điểm làng giềng lớn hơn một ngưỡng tối thiểu, tìm tất cả các đối tượng mà các điểm láng giềng thuộc về các đối tượng đã tìm ở trên, tập các đối tượng đó là một nhóm, nhưng OPTICS có một mảng chứa các đối tượng được sắp xếp thứ tự theo khoảng cách đến đối tượng lõi (core-object) các cụm tăng dần nhằm tự động phân cụm dữ liệu, xác định bán kính tối thiểu để xác định các điểm láng giềng phù hợp.

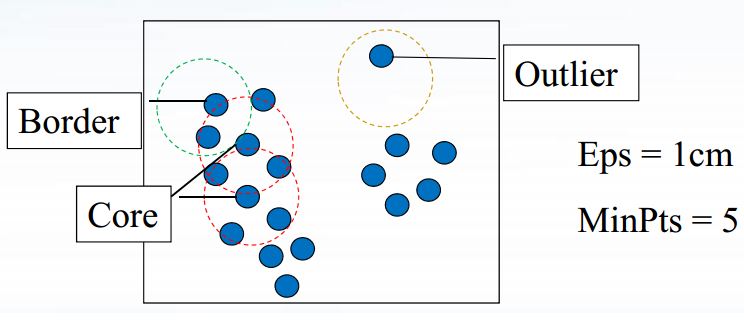


Hình 4 – Minh họa thuật toán OPTICS.

Giống DBSCAN, OPTICS có hai tham số: ε là bán kính vùng lân cận để xét điểm có thuộc cho một cụm; MinPts là số lượng điểm ít nhất để tạo một đối tượng mật độ đủ dầy.

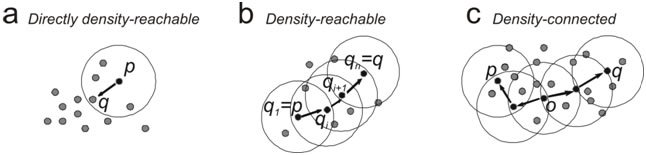
Một vài khái niệm:

* Đối tượng lõi (core-object): là đối tượng thỏa Eps và MinPts.
* Đối tượng biên (border-object): là đối tượng có số điểm lân cận ít hơn MinPts trong Eps nhưng là lân cận của đối tượng lõi.
* Đối tượng nhiễu (noise-object): là bất kì điểm nào không phải là lõi hay biên



Bảng 1 – các biến thể trong tính toán trọng số.

* Đạt được mật độ trực tiếp (directly density-reachable): một điểm p gọi là đạt được mật độ trực tiếp từ q nếu:
  + p nằm trong lân cận Eps của q
  + N Eps(q) phải thỏa MinPts hay |NEps (q)| ≥ MinPts.
* Đạt được mật độ (density-reachable): Một điểm p gọi là đạt được mật độ từ điểm q (thỏa Eps, MinPts) nếu tồn tại một chuỗi các điểm p1, p2,…,pn với p1 là q và pn là p để mà pi+1 là đạt được mật độ trực tiếp từ pi.
* Liên thông mật độ (density-connected): một điểm p gọi là liên thông mật độ đến điểm q (thỏa Eps, MinPts) nếu tồn tại một điểm o (cũng thỏa Eps, MinPts) mà cả hai điểm p và q đều là đạt được mật độ từ o



Hình 5 - fdsafds

Khác với DBSCAN, OPTICS đặt giá trị khoảng cách lõi (core distance) tại mỗi đối tượng lỗi, giá trị được xác định:



*reachability-distance* là khoảng cách từ p đến q:



**Thuật toán** OPTICS

**function** OPTICS(*DB*, *eps*, *MinPts*)

**foreach** *p* **in** *DB*

*p*.reachability-distance ← UNDEFINED

**foreach** *unprocessed\_p* **in** *DB*

*N* ← getNeighbors(*p*, *eps*)

Đánh dấu *p* là processed

Thêm *p* vào danh sách

**if** core-distance(*p*, *eps*, *MinPts*) != UNDEFINED

*seeds* ← toàn bộ hàng đợi

update(*N*, *p*, *seeds*, *eps*, *MinPts*)

**foreach** *q* **in** *seeds*

*N’* ← getNeighbors(*q*, *eps*)

Đánh dấu *q* là processed

Thêm *q* vào danh sách

**if** core-distance(*q*, *eps*, *MinPts*) != UNDEFINED

update(*N’*, *q*, *seeds*, *eps*, *MinPts*)

**function** update(*N*, *p*, *seeds*, *eps*, *MinPts*)

*coredist* = core-distance(*p*, *eps*, *MinPts*)

**foreach** *o* **in** *N*

**if** *o* is not processed

*new-reach-dist* = max(*coredist*, dist(*p*, *o*))

**if** *o*.reachability-distance == UNDEFINED

*o*.reachability-distance ← *new-reach-dist*

*seeds*.insert(*o*, *new-reach-dist*)

**else if** *new-reach-dist* < *o*.reachability-distance

*o*.reachability-distance = *new-reach-dist*

*seeds*.move-up(*o*, *new-reach-dist*)

Thuật toán OPTICS có độ phức tạp là O(*n*log*n*) với n là kích thước của tập dữ liệu. Giá trị Eps (radius) ảnh hưởng đến độ phức tạp thuật tuận, nếu Eps lớn quá (lớn hơn bằng khoảng cách lớn nhất giữa 2 điểm) thì mỗi đối tượng phải duyệt n đôi tượng, chí phí cao.

## CHAMELEON:

Các thuật toán gom cụm đều nhằm hướng đến việc tạo ra các cụm sao cho độ tương tự giữa các đối tượng trong cùng một nhóm là cao nhất và độ tương tự giữa các đối tượng khác nhóm là thấp nhất. Với cách tiếp cận sử dụng mô hình tĩnh, các thuật toán thường phải giả định phân phối của dữ liệu (K-means, PAM, CLARA giả định các cụm dữ liệu được phân phối dạng hình cầu) và bỏ qua (CURE) hoặc đánh giá một cách cứng nhắc (ROCK) về độ tương tự trong nội bộ nhóm.

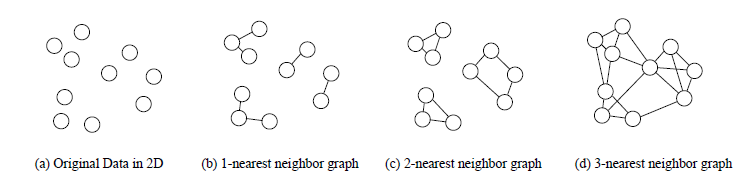
CHAMELEON [3] là thuật toán gom cụm phân cấp thuộc nhóm tích tụ, với hướng tiếp cận sử dụng mô hình động. Không có bất kì một giả định nào về phân phối của dữ liệu, và độ tương tự nội bộ của các nhóm là yếu tố quyết định để kết hợp các nhóm lại với nhau.

Một ma trận kề chứa độ tương tự giữa các đối tượng thường được xây dựng trước để thuận tiện cho việc thực thi thuật toán.

1. **Mô hình hóa dữ liệu**

Với ma trận kề đã xây dựng, CHAMELEON biểu diễn tập dữ liệu theo hướng tiếp cận đồ thị phổ biến: k-nearest neighbor.

Mỗi đỉnh trên đồ thị tương ứng với một điểm dữ liệu. Giữa 2 đỉnh chỉ tồn tại nối khi chúng thuộc cùng một nhóm *k* đỉnh gần nhất. Cạnh nối giữa 2 đỉnh có trọng số là độ tương tự giữa 2 đỉnh đó.



Hình 6 – Minh họa đồ thị theo tham số k.

Việc biểu diễn dữ liệu bằng đồ thị k-nearest neighbor *Gk* mang lại các ưu điểm sau:

* Các điểm xa nhau hoàn toàn không có sự liên hệ trong *Gk*.
* Các cụm láng giềng có kích thước động: đối với các vùng phân bố đặc, cụm láng giềng sẽ có kích thước nhỏ, trong khi đối với các vùng phân bố thưa thì kích thước sẽ lớn hơn.
* Mật độ của vùng được thể hiện ngay ở trọng số của các cạnh: cạnh trong vùng mật độ cao (độ tương tự cao) sẽ có giá trị lớn, cạnh trong vùng mật độ thấp (độ tương tự thấp) sẽ có giá trị nhỏ.
* Thuận tiện cho việc thực thi các thuật toán phân hoạch trên toàn bộ đồ thị.

1. **Mô hình độ tương tự**

Để xác định độ tương tự giữa 2 cụm *Ci* và *Cj*, CHAMELEON xét đến cả hai độ đo *inter-connectivity tương đối RI*(*Ci*, *Ci*) và *closeness tương đối RC*(*Ci*, *Ci*).

1. *Inter-Connectivity tương đối*

Độ đo inter-connectivity tương đối giữa *Ci* và *Cj* được định nghĩa là giá trị độ đo inter-connectivity tuyệt đối được chuẩn hóa bằng giá trị inter-connectivity nội tại của *Ci* và *Cj*. Inter-connectivity tuyệt đối của cặp *Ci* và *Cj* được định nghĩa là tổng trọng số của các cạnh nối các đỉnh từ *Ci* đến *Cj* (nói một cách khác là tổng trọng số các cạnh cắt của một cụm lớn nào đó khi phân hoạch thành *Ci* đến *Cj*), kí hiệu là . Inter-connectivity nội tại của cụm *Ci* đơn giản là kích thước min-cut bisector của cụm đó, kí hiệu là .



Sử dụng độ đo inter-connectivity tương đối giúp CHAMELEON vượt qua được các giới hạn của của mô hình inter-connectivity tĩnh [3].

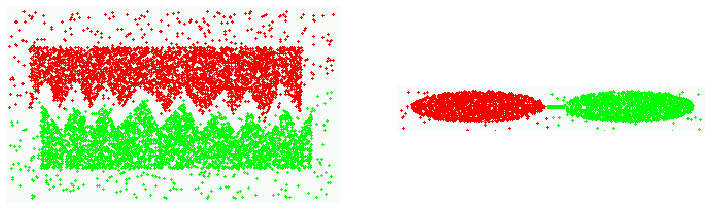
1. *Closeness tương đối*

Độ đo closeness tương đối giữa *Ci* và *Cj* được định nghĩa là giá trị độ đo closeness tuyệt đối được chuẩn hóa bằng giá trị closeness nội tại của *Ci* và *Cj*. Có nhiều cách để xác định giá trị closeness tuyệt đối giữa 2 cụm. Một cách thường thấy là xác định giá trị này dựa trên cặp điểm gần nhất giữa 2 cụm. Cách này có nhược điểm lớn là khi chỉ phụ thuộc vào một cặp điểm duy nhất, mô hình sẽ dễ bị tác động bởi các điểm ngoại lai và nhiễu. CHAMELEON xác định độ đo closeness tuyệt đối bằng cách lấy giá trị trung bình độ tương tự giữa các điểm trên *Ci* được kết nối với các điểm trên *Cj*. Cách làm này vừa phù hợp cho việc đo sự đồng dạng của giữa các phần tử dọc theo biên của 2 cụm, đồng thời giảm thiểu ảnh hưởng của các phần tử ngoại lai và nhiễu.

Tương tự, độ đo closeness nội tại của *Ci* cũng có nhiều cách tính khác nhau. Một cách tiếp cận khả dĩ là lấy giá trị trung bình của tất cả các cạnh nối nội bộ của *Ci*. Ai đó có thể cho rằng đối với kiểu gom nhóm tích tụ, các cạnh được chọn để kết nối trước sẽ mạnh hơn so với các cạnh được chọn sau, do đó việc sử dụng các cạnh phân đôi nội tại của *Ci* và *Cj* sẽ cho giá trị nhỏ hơn. Tuy nhiên CHAMELEON vẫn sử dụng giá trị trung bình của các cạnh nối nội bộ.

, Với kí hiệu biểu thị cho giá trị trung bình.

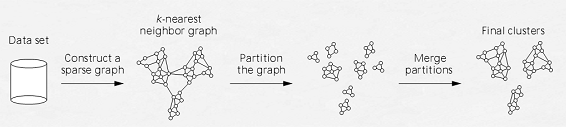
Sử dụng độ đo closeness tương đối giúp CHAMELEON giải quyết được những hạn chế của các thuật toán chỉ dựa vào độ closeness tuyệt đối [3].



1. **Hai giai đoạn của thuật toán gom nhóm**

Mô hình độ tương tự như trên chỉ áp dụng được khi số lượng phần tử trong mỗi cụm con đủ lớn. Để có được các giá trị *inter-connectivity tương đối* và *closeness tương đối*, trước hết CHAMELEON cần phải tính được các giá trị nội tại ở mỗi cụm. Với số lượng phần tử quá nhỏ, việc tính toán các giá trị nội tại này sẽ cho kết quả không thực sự chính xác. Vì lý do này, CHAMELEON được cấu trúc thành 2 giai đoạn riêng biệt:

* Phân hoạch đồ thị thành một số lượng lớn các cụm con có số lượng đỉnh cần thiết cho mô hình động.
* Sử dụng thuật toán phân tầng dựa trên mô hình động nhằm tìm kiếm các cụm hợp lệ bằng cách lặp lại quá trình kết hợp các cụm con nhiều lần.

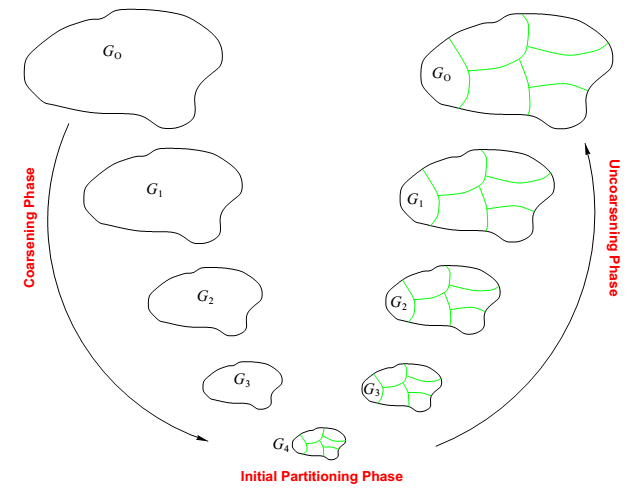


Hình 7 – Minh họa thuật toán CHAMELEON.

1. *Tìm các cụm con cơ sở*

Sau khi biểu diễn các điểm dữ liệu dưới dạng một đồ thị k-nearest neighbor, một thuật toán phân hoạch được thực thi để phân chia dữ liệu vào một số lượng lớn các cụm con, sao cho cạnh cắt là tối tiểu. Điều kiện này phù hợp với một giả định thường thấy trong dữ liệu, rằng khoảng độ liên kết trong nội bộ nhóm luôn mạnh hơn độ liên kết giữa các nhóm với nhau.

CHAMELEON sử dụng một thuật toán phân hoạch đa cấp thuộc thư viện hMETIS [4]. Karypis [4] chỉ ra rằng, các thuật toán phân hoạch phân cấp tỏ ra rất hiệu quả trong việc nắm bắt cấu trúc toàn thể của đồ thị và có khả năng phân hoạch đồ thị với cạnh cắt rất nhỏ. Khả năng này giúp cho thuật toán trở nên rất linh hoạt đối với cả đồ thị đặc lẫn đồ thị thưa, giúp phát hiện ra ranh giới phân chia tự nhiên giữa các vùng.



Hình 8 – Minh họa thuật toán phân hoạch đa cấp.

1. *Sáp nhập các cụm con theo mô hình động*

Ngay sau khi hoàn thành việc phân hoạch các cụm con cơ sở, CHAMELEON tiến hành thực hiện thuật toán tích tụ phân tầng để kết hợp các cụm con này. Ở mỗi lần sáp nhập, CHAMELEON lựa chọn ra các cặp cụm *tương tự nhau nhất* dựa trên các độ đo tương đối và tiến hành kết hợp chúng lại với nhau. Có nhiều cách để đánh giá đồng thời cả hai độ đo trên. Có 2 phương án thường được sử dụng trong CHAMELEON.

Ở cách tiếp cận thứ nhất, các độ đo được đánh giá dựa theo các ngưỡng mà người dùng chỉ định. Hai cụm chỉ kết hợp được với nhau khi thỏa mãn hai điều kiện sau:

*RI*(*Ci*, *Cj*) ≥ *TRI* và *RC*(*Ci*, *Cj*) ≥ *TRC*.

Khi xem xét cụm *Ci*,nếu có nhiều hơn một cụm *Cj* thỏa mãn điều kiện này, cặp (*Ci*, *Cj*) được chọn có giá trị inter-connectivity tuyệt đối lớn nhất. Dễ thấy rằng khác với các thuật toán truyền thống, ở mỗi lần lặp có thể có nhiều hơn một cặp cụm được kết hợp với nhau. Các tham số được chỉ định để điều chỉnh các tính chất mong muốn của cụm. *TRI* cho phép kiểm soát được tính biến thiên của về mức độ liên kết giữa các phần tử trong cụm. *TRC* cho phép kiểm soát được tính thống nhất về độ tương tự giữa các phần tử thuộc cùng một cụm.

Trong cách tiếp cận thứ hai, một hàm số kết hợp giữa *RI* và *RC* được định nghĩa, và cặp cụm được kết hợp sẽ là cặp cho giá trị hàm số lớn nhất. Dễ thấy đây chắc chắn phải là một hàm đồng biến. Một cách tự nhiên, ta có thể định nghĩa nó là tích giữa hai giá trị này. Mặt khác, trong một số trường hợp, mỗi thừa số có thể được quan tâm với một mức độ khác nhau. CHAMELEON sử dụng một hàm số đơn giản sau để đánh giá một cặp cụm:

*RI*(*Ci*, *Cj*) \* *RC*(*Ci*, *Cj*)α.

So với cách tiếp cận thứ nhất, cách tiếp cận này cho phép ta hình dung rõ ràng hơn về tính phân cấp của thuật toán.

**Thuật toán** CHAMELEON

**function** CHAMELEON(*D*)

Xây dựng đồ thị k-nearest neighbour cho tập *D*

Phân hoạch đồ thị bằng thuật toán phân hoạch đa cấp hMETIS

**repeat**

Sáp nhập các cụm thỏa mãn tốt nhất về độ liên kết và mật độ riêng.

**until** không thể sáp nhập được nữa

Độ phức tạp của thuật toán CHAMELEON có thể được đánh giá thông qua độ phức tạp trong từng giai đoạn của thuật toán. Với *n* là kích thước của tập dữ liệu, *m* là kích thước của cụm con cơ sở, độ phức tạp tổng thể là O(*nm* + *n*log*n* + *m2*log*m*) [4].

# Kết quả thực nghiệm

1. **Dữ liệu**

Dữ liệu thực nghiệm gồm 501 bài báo được thu thập từ hai trang tin tức:

<http://www.bbc.com/>

<http://www.pcworld.com/>

Các bài báo thuộc về 5 chủ đề: *Business*, *News*, *Phone\_Tablet*, *Security*, *Sports*. Mỗi chủ đề gồm 100 bài báo, riêng chủ đề *Sports* có 101 bài báo.

Dữ liệu được thu thập bằng công cụ Crawler, được viết dựa trên ngôn ngữ Python [5].

1. **Kết quả thực nghiệm**

# Kết luận

Trong tài liệu này, chúng tôi đã trình bày một cách tổng quan về bài toán gom nhóm văn bản, cùng một số vấn đề có liên quan về xử lý ngôn ngữ tự nhiên, phương pháp biểu diễn văn bản theo mô hình vector space, các phương pháp đo độ tương tự sử dụng giá trị độ dài và số đo góc. Một số phương pháp gom nhóm được giới thiệu là PAM, CLARA, OPTICS và CHAMELEON. Mỗi phương pháp đều có những ưu điểm và nhược điểm riêng, được áp dụng tùy vào ngữ cảnh nhất định của bài toán.

Trong tương lai, với các điều kiện cho phép, chúng tôi sẽ tiếp tục nghiên cứu và mở rộng bài toán gom nhóm văn bản cho ngôn ngữ tiếng Việt.

# Lời cảm ơn

Xin chân thành cảm ơn Huỳnh Nguyễn Phúc Huỳnh đã hỗ trợ chúng tôi trong việc thu thập dữ liệu.

# Tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | L. Kaufmann, P. J. Rousseeuw, "Clustering Large Application (Program CLARA)," in *Finding Groups in Data: An Introduction to Cluster Analysis*, New York, John Wiley & Sons, 1990, pp. 126-163. |
| [2] | C. P. Wei, Y. H. Lee, C. M. Hsu, "Empirical Comparison of Fast Clustering Algorithms for Large Data Sets," in *33rd International Conference on System Sciences*, Hawaii, 2000. |
| [3] | G. Karypis, E. H. Han, V. Kumar, "Chameleon: A Hierarchical Clustering Algorithm Using Dynamic Modeling," University of Minnesota, Minneapolis, USA, 1999. |
| [4] | G. Karypis, V. Kumar, "Multilevel k-way hypergraph partioning," in *Proceeding of the Design and Automation Conference*, New York, USA, 1999. |
| [5] | M. K. Huy. [Online]. Available: https://github.com/maikhaihuy/Crawler\_PcWorld\_by\_Python. |