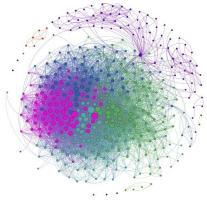


Web Content and Usage Analysis: Clustering

Miłosz Kadziński
Institute of Computing Science
Poznan University of Technology, Poland
www.cs.put.poznan.pl/mkadzinski/wpi

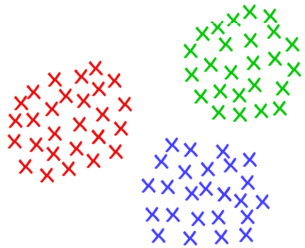
[1] Wykład będzie dotyczył klasycznego zagadnienia analizy danych, jaką jest analiza skupień lub grupowanie. Zagadnienie zostanie omówienie w kontekście analizy użytkowania i zawartości sieci. Rozpoczniemy od przedstawienia przydatności takich algorytmów oraz ich kilku przykładowych zastosowań. Następnie omówimy cztery najsławniejsze i najczęściej używane w praktyce algorytmy analizy skupień. Będą one reprezentowały różne grupy podejścia do tego zagadnienia, w tym algorytmy iteracyjno-optymalizacyjne, których są oparte na rozdziale obiektów między grupy i systematycznym poprawianiu takiego rozdziału, metody hierarchiczne oraz podejścia bazujące na analizie gęstości danych. Jako mały dodatek - przydatny przy omawianiu jednego z podejść - przedstawimy zasadę obliczania odległości Levenshteina, którą można wykorzystać m.in. do poprawiania literówek.

What is Clustering in Data Mining?

Clustering is a process of partitioning a set of data (or objects) in a set of meaningful sub-classes, called clusters

Aim: *help users understand the natural structure in a data set*

Cluster is collection of data objects:
- similar to one another (hence, can be treated collectively as one group)
- as a collection, they are sufficiently different from other groups

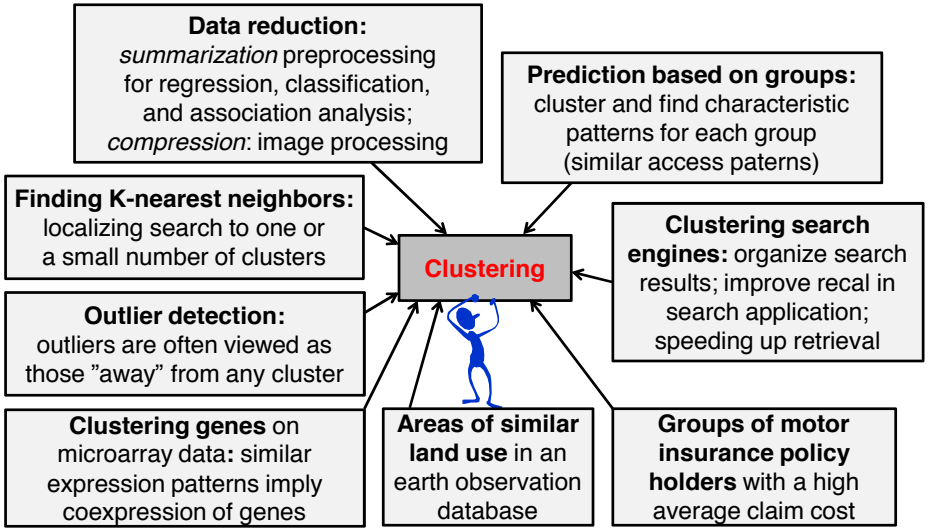


intra- vs. inter-cluster similarity

Clustering can be seen as unsupervised classification (no pre-defined classes)

[2] Odkrywanie skupień jest procesem podziału zbiory danych lub instancji na zbiór interpretowalnych grup zwanych klastrami. Celem takiego podziału jest zrozumienie struktury, prawidłowości i zależności występujących danych. Z jednej strony, klaster (grupa) należy postrzegać jako zbiór obiektów, które są do siebie podobne, co uprawnia nas do traktowania ich jako logiczną całość. Z drugiej strony, aby uzasadnić istnienie wielu klastrów, obiekty do nich przynależące muszą się od siebie istotnie różnić. Wyróżniamy więc podobieństwo wewnątrzgrupowe, które powinno być wysokie oraz międzygrupowe, które winno być niskie. Z poprzedniego wykładu rozumiecie już, na czym polega klasyfikacja. W tym kontekście, warto zaznaczyć, że odkrywanie skupień można postrzegać jako klasyfikację nienadzorowaną. Zbiór klas (grup) nie jest tu więc określony z góry, a musi dopiero zostać odkryty.

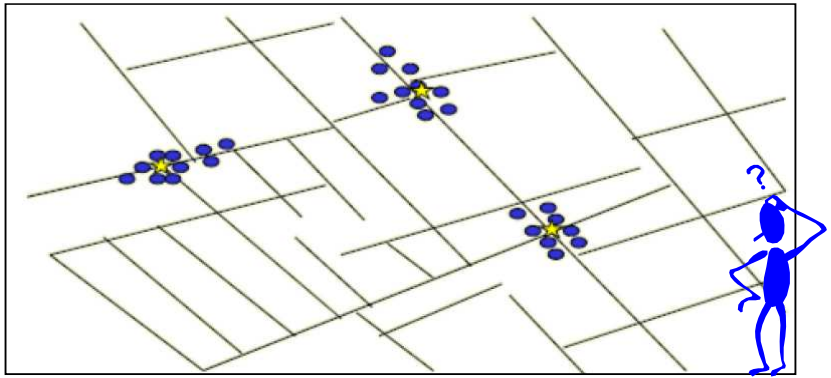
Clustering Task



[3] Aby uzasadnić potrzebę istnienia algorytmów klastrowania, rozpocznijmy od omówienia ich kilku przykładowych zastosowań w zakresie przetwarzania informacji oraz eksploracji zasobów Internetu. Pierwsze polega na tym, by dokonać redukcji liczności danych. Grup jest bowiem znacznie mniej niż oryginalnych obiektów, co ma znaczenie przy podsumowywaniu danych, ich raportowaniu, kompresji, ale także zastosowaniach takich jak klasyfikacja czy regresja. Drugie istotne zastosowanie to odkrywanie wzorców charakterystycznych dla grup; choćby dla użytkowników moglibyśmy odkryć pewne wzorce dostępu do zasobów albo grupy charakteryzujące się podobnymi preferencjami/gustami. Trzeci obszar to organizacja wyników, np. wyszukiwania. W grupach można też znaleźć obiekty podobne do danego, co jest przydatne choćby do wykrycia najbliższych sąsiadów (pojęcie to znacie już z poprzedniego wykładu). Wreszcie analiza taka może też prowadzić do identyfikacji obserwacji, które nie pasują do żadnej grupy, tj przypadków odstających. Oczywiście algorytmy grupowania znajdują też zastosowanie w wielu innych dziedzinach. Na slajdzie podano przykłady dotyczące grupowania genów, obszarów lub terenów w systemach informacji geograficznej czy wniosków badanych przez ubezpieczyciela.

Historic Application of Clustering

- John Snow, a London physician plotted the location of cholera deaths on a map during an outbreak in the 1850s
- The locations indicated that cases were clustered around certain intersections where they were polluted wells – thus expoising both the problem and the solution



[4] Historycznie pierwsze zastosowanie analizy skupień miało miejsce w połowie XIX wieku w Wielkiej Brytanii. Zastosowanie to jest na czasie, bo dotyczy analizy przeprowadzonej przez londyńskiego lekarza, Johna Snow, w czasach epidemii cholery. Jego celem było odkrycie jej przyczyn. Z pomocą przyszło pogrupowanie przypadków śmierci z wykorzystaniem mapy miasta. Okazało się, że były one zorientowane wokół różnych ujęć wody, co pozwoliło też na znalezienie rozwiązania problemu. Implikowało bowiem fundamentalne zmiany w systemie gospodarki wodnej nie tylko w Londynie, ale wielu innych europejskich miastach.

Clustering Application: Usage Profiles

Discovering Aggregate Usage Profiles

- **Goal**: to effectively capture "user segments" based on their common usage patterns from potentially anonymous click-stream data
- **Method**: cluster user transactions to obtain user segments automatically, then represent each cluster by its centroid

user-pageview transaction matrix

	A	B	C	D	E	F
U0	1	1	0	0	0	1
U1	0	0	1	1	0	0
U2	1	0	0	1	1	0
U3	1	1	0	0	0	1
U4	0	0	1	1	0	0
U5	1	0	0	1	1	0
U6	1	1	0	0	0	1
U7	0	0	1	1	0	0
U8	1	0	1	1	1	0
U9	0	1	1	0	0	1

clustering

		A	B	C	D	E	F
cluster 1	U1	0	0	1	1	0	0
	U4	0	0	1	1	0	0
	U7	0	0	1	1	0	0
cluster 2	U0	1	1	0	0	0	1
	U3	1	1	0	0	0	1
	U6	1	1	0	0	0	1
	U9	0	1	1	0	0	1
cluster 3	U2	1	0	0	1	1	0
	U5	1	0	0	1	1	0
	U8	1	0	1	1	1	0

[5] Przejdźmy teraz do kilku przykładowych dotyczących analizy użytkowania i zawartości. Pierwszy będzie dotyczył profili użytkowania, a więc pewnych wzorców korzystania z serwisu lub sklepu internetowego. Punktem wyjścia do przeprowadzenia analizy jest tu zbiór sesji użytkowników. W naszym przykładzie dysponujemy macierzą transakcyjną o charakterze binarnym i zanonimizowanym, dotyczącą 10 użytkowników, którzy odwiedzali 6 stron. Dane takie grupuje się z wykorzystaniem wybranego algorytmu analizy skupień (pomijamy tu na razie wybór konkretnego algorytmu). Załóżmy, że uzyskamy trzy grupy. Przykładowo, w grupie 1, znajdują się użytkownicy U1, U4 i U7. Rzeczywiście ich dostęp do serwisu był podobny, bo wszyscy odwiedzili strony C oraz D.

Usage Profile Aggegation

- Profiles are represented as weighted collections of items (pages, products, etc.)
- Weights represent the significance of the item within each cluster
- **Usage profiles are obtained from each centroid after sorting by weight and filtering out low-weight items in each centroid**
- Profiles are overlapping, so they capture common interests among different groups/types of users (e.g., customer segments)

		A	B	C	D	E	F
cluster 1	U1	0	0	1	1	0	0
	U4	0	0	1	1	0	0
	U7	0	0	1	1	0	0
cluster 2	U0	1	1	0	0	0	1
	U3	1	1	0	0	0	1
	U6	1	1	0	0	0	1
	U9	0	1	1	0	0	1
cluster 3	U2	1	0	0	1	1	0
	U5	1	0	0	1	1	0
	U8	1	0	1	1	1	0

```
PROFILE 1
  1.00   C
  1.00   D

PROFILE 2
  1.00   B
  1.00   F
  0.75   A
  0.25   C

PROFILE 3
  1.00   A
  1.00   D
  1.00   E
  0.33   C
```

Possible use in recommenation

Given an active session A ➔ B,
the best matching profile is Profile 2

This may result in
a recommendation for page F,
since it appears with high weight
Profile 2

develop targeted marketing programs

[6] Grupy powstałe z analizy sesji lub transakcji użytkowników stanowią bazę do uzyskania profili użytkownika. Profile takie są kolekcjami obiektów, jak strony czy produkty, z którymi mogą skojarzone być wagi. Wagi wskazują na istotność obiektu w danej grupie. Profil użytkowania uzyskuje się jako centroid danej grupy (czyli średnia reprezentacja użytkowników), w którym poszczególne obiekty (w naszym wypadku strony) sortuje się i ewentualnie odfiltrowuje te, których wagi są niskie. Oznacza to, że nie reprezentowały one cech decydujących o znalezieniu się użytkowników w tej samej grupie. Przykładowo, profil użytkowania dla grupy 1 to strony C i D z wagą 1, bo wszyscy użytkownicy w tej grupie je odwiedzili. Innych stron tu nie ma, bo ich waga byłaby zerowa. Profil użytkowania dla grupy 2 to strona B z wagą 1, F z wagą 1, A z wagą 0.75 (bo 3 z 4 użytkowników z tej grupy na niej było) i C z wagą 0.25. Zwróć uwagę na posortowanie oraz fakt, że stronę C można by odfiltrować, bo jej waga jest bardzo niska (tylko 1 z 4 użytkowników tej grupy na niej był). Profile przecinają się, co odzwierciedla wspólne zainteresowania członków różnych grupy (np. grup klientów). Porównaj np. profile dla grupy 2 i 3. Profile takie są bardzo przydatne dla zrozumienia charakterystyki użytkowników, co może prowadzić do opracowania długoterminowych programów marketingów, dedykowanych dla różnych grup. W bardziej dynamicznym zastosowaniu, mogą posłużyć do rekomendacji stron, które powinien odwiedzić użytkownik z aktywną sesją. Przykładowo, jeśli był on już na stronach A i B, to profil 2 wskazuje, że można mu też polecić stronę F (ze względu na jej wysoką wagę).

Clustering Application: Content Profiles

Discovering Aggregate Content Profiles

- **Goal**: automatically group together documents which partially deal with similar concepts
- **Method**: identify concepts by clustering features (keywords) based on their common occurences among documents, then represent each cluster by its centroid

page-feature matrix

	A	B	C	D	E	F
business	1	1	0	0	0	1
data	0	1	1	1	0	0
ecommerce	0	1	1	0	0	1
information	1	1	1	1	1	0
intelligence	1	1	0	0	1	1
marketing	1	1	1	0	1	1
mining	0	1	1	1	0	0
retrieval	1	0	1	1	1	0
search	1	0	0	0	1	0
web	0	0	1	1	0	0

clustering

		A	B	C	D	E	F
cluster 1	web	0	0	1	1	0	0
	data	0	1	1	1	0	0
	mining	0	1	1	1	0	0
cluster 2	business	1	1	0	0	0	1
	intelligence	1	1	0	0	1	1
	marketing	1	1	1	0	1	1
	ecommerce	0	1	1	0	0	1
cluster 3	search	1	0	0	0	1	0
	information	1	1	1	1	1	0
	retrieval	1	0	1	1	1	0

[7] W przypadku profili użytkowania, grupowaliśmy użytkowników opisanych względem stron, które odwiedzali. Dla profili zawartości - analogicznie grupować będziemy termy ze względu na to, na jakich stronach się znajdują. Celem będzie odkrycie dokumentów, które dotyczą podobnej tematyki. Krokiem pośrednim jest realizacja grupowania słów kluczowych lub termów, na podstawie ich współwystępowania w dokumentach. Na slajdzie zaprezentowano macierzy termy na dokumenty, którą świetnie znacie z wykładów Irminy Masłowskiej. Stosujemy algorytm grupowania, co daje nam - powiedzmy - trzy grup. W każdej znajdują się termy, które znajdowały się na podobnych stronach. Przykładowo, w grupie 1 odnaleźć można web, data oraz mining, bo występowały one często razem, w dokumentach B, C i D.

Content Profile Aggegation

- Profiles are represented as weighted collections of items (terms, keywords, etc.)
- **Usage profiles are obtained from each centroid after sorting by weight and filtering out low-weight items in each centroid**
- Cluster centroids represent documents in which features in the cluster appear frequently

		A	B	C	D	E	F
cluster 1	web	0	0	1	1	0	0
	data	0	1	1	1	0	0
	mining	0	1	1	1	0	0
cluster 2	business	1	1	0	0	0	1
	intelligence	1	1	0	0	1	1
	marketing	1	1	1	0	1	1
	ecommerce	0	1	1	0	0	1
cluster 3	search	1	0	0	0	1	0
	information	1	1	1	1	1	0
	retrieval	1	0	1	1	1	0

PROFILE 0
1.00 C (web, data, mining)
1.00 D (web, data, mining)
0.67 B (data, mining)

PROFILE 1
1.00 B (business, intelligence, marketing, ecommerce)
1.00 F (business, intelligence, marketing, ecommerce)
0.75 A (business, intelligence, marketing)
0.50 C (marketing, ecommerce)
0.50 E (intelligence, marketing)

PROFILE 2
1.00 A (search, information, retrieval)
1.00 E (search, information, retrieval)
0.67 C (information, retrieval)
0.67 D (information, retireval)

B filtered out, because of too low weight

[8] Profil ponownie będzie ważoną kolekcją obiektów, która de facto jest centroidem każdej grupy. Centroidem uporządkowanym przez posortowanie obiektów zgodnie z malejącymi wagami oraz odsianie obiektów o bardzo niskich wagach. Profile zawartości to zbiory dokumentów, w których cechy interpretowane jako termy, powtarzają się bardzo często. Przykładowo, na podstawie analizy grupy 1 powstał profil ze stronami C, D i B. Dla dwóch pierwszych waga to 1, bo występują na nich wszystkie trzy termy typowe dla tej grupy, a dla B waga to 2/3, bo można na niej odnaleźć 2 z 3 termów charakterystycznych dla tego klastra. Zwróćcie uwagę, że w profilu dla ostatniej grupy, strona B została odsiana, bo jej waga była bardzo niska.

Clustering and Collaborative Filtering

- Provides a model-based (scalable) version of user-based collaborative filtering, compared to k-nearest-neighbor
- Each centroid represented the average rating (in that cluster of users) for each item

centroids for 4 clusters

Cluster	Full	Cl1	Cl2	Cl3	Cl4	NU
Size	20	4	7	4	5	
M1	2.83	4.21	1.81	2.83	3.17	3
M2	3.86	4.21	3.84	2.96	4.31	
M3	2.33	2.50	2.71	2.17	1.80	3
M4	2.25	2.56	2.64	1.63	1.95	
M5	2.77	2.19	2.73	2.69	3.35	4
M6	2.82	2.16	3.49	2.20	2.89	3
M7	2.42	1.50	2.04	2.81	3.37	
M8	3.76	2.44	4.36	3.00	4.60	4

Correlation (similarity) with NU

0.63	-0.41	0.50	0.65	**0.74**

new user

- New user has highest similarity to Cl4 centroid
- The whole cluster could be used as the neighborhood for new user

Original user-item matrix (scale 1-5)

	M1	M2	M3	M4	M5	M6	M7	M8
U1	1	5		3			3	5
U2	5	4			3	2	1	
U3	3		1	2	2			5
U4		3			4	1		3
U5	2	4	3			2	2	
U6	5			3	1		3	1
U7	1	4	5	5	2			4
U8	2	1			4	5	1	
U9			3	2	2			5
U10	3	5	1				4	4
U11			2	1		2		3
U12	4	4		2		1	1	4
U13			2		4		4	5
U14		5	3	3	2		1	1
U15		2			3	3		2
U16		3	2	1	1		4	4
U17	1	5	1	2		4		4
U18	5		4		3	3	4	5
U19		4		2		5	1	5
U20	2	5	1	1	5	3		4

[9] Podczas ostatniego wykładu mówiliśmy o rekomendacji. Wspomnieliśmy też, że wykorzystanie algorytmu user-based collaborative filtering w połączeniu z metodą najbliższych sąsiadów k-NN pociąga za sobą problemy ze skalowalnością. Wykorzystanie analizy skupień może tym problemom zapobiec. Załóżmy, że dana jest macierz 20 użytkowników oceniających 8 filmów. Chcąc polecić film nowemu użytkownikowi NU, musielibyśmy porównać go z wszystkimi 20 użytkownikami w bazie. Zamiast tego moglibyśmy ich pogrupować, np. w 4 klastry zaprezentowane po lewej stronie slajdu, a następnie porównywać użytkownika z centroidami tych grup. Na slajdzie przedstawiono współczynnik korelacji Pearsona jako miarę takiego podobieństwa. Widzimy, że ma on wysokie podobieństwo do grupy Cl4. Tą z kolei można by wykorzystać jako sąsiedztwo w predykcji oceny dla użytkownika NU.

Clustering Based on Ratings: *movielens*

movielens

Non-commercial, personalized movie recommendations

"MovieLens helps you find movies you will like.
Rate movies to build a custom taste profile,
then MovieLens recommends other movies for you to watch."

Groups of users named after animals

You are a member of the Eagle Group (**what's this?**)

About this group: Eagles have powerful eyesight, so they tend to sit in the back of the theater. They like classic movies.

The Eagle Group thinks these movies are cool.

Title	Average rating
12 Angry Men (1957)	★★★★½
It's a Wonderful Life (1946)	★★★★½
Mr. Smith Goes to Washington (1939)	★★★★½
Roman Holiday (1953)	★★★★½
Cinema Paradiso (Nuovo cinema Paradiso) (1989)	★★★★½

These movies have high ratings from the Eagle Group and low ratings from other groups.

Title	Average Rating
Manon of the Spring (Manon des sources) (1986)	★★★★½
Jean de Florette (1986)	★★★★½
Witness for the Prosecution (1957)	★★★★½
Dial M for Murder (1954)	★★★★½
Charade (1963)	★★★★½

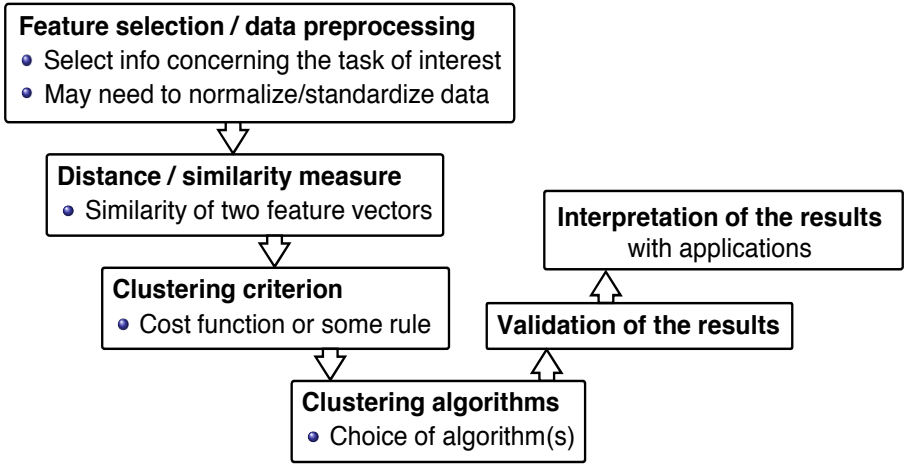
[10] Najlepszym potwierdzeniem przydatności wykorzystania analizy skupień w rekomendacji jest system movielens, który dotyczy oceny filmów. System ten tworzy profil użytkownika na podstawie wystawionych przez niego ocen, a następnie poleca filmy, które mogą mu się podobać. W szczególności klasyfikuje go do pewnej grupy (na slajdzie Eagle Group) i na tej bazie rekomenduje filmy wysoko ocenione przez członków tej grupy, ale też filmy, które członkowie tej grupy postrzegają dobrze, podczas gdy inne grupy oceniają je nisko.

Clustered Searched Results

- Clustering search engines: Grouper, **Carrot2**, Vivisimo, SnakeT, Yippy
- Perform clustering and labeling on the results of a search engine
- Help users to find a quick overview of the search results

[11] Ostatnie przykładowe zastosowanie, które ma przed sobą bardzo dużą przyszłość, to organizacja wyników wyszukiwania. Zamiast listy dokumentów istotnych dla naszego zapytania (którą trzeba niekiedy przewijać w poszukiwaniu czegoś istotnego), wyszukiwarka zwraca je w postaci grup. Istniejące systemy, które działają w tym duchu to np. Grouper, Vivisimo czy rozwijany pierwotnie na Politechnice Poznańskiej przez Dawida Weissa - Carrot2. Niekiedy grupom w automatyczny sposób nadawane są nazwy. Taka organizacja pozwala nie tylko na szybsze zapoznanie się z wynikami i typami zwróconych dokumentów, ale też zwiększa miarę recall i może przyspieszyć działanie wyszukiwarki.

Clustering Task: Basic Steps



[12] Analiza skupień realizowana jest w kilku krokach. W pierwszym etapie wybiera się cechy, które posłużą do opisu grupowanych obiektów. W omówionych przykładach były to strony dla termów lub użytkowników albo oceny wystawione przez użytkowników. Etap ten może wymagać normalizacji lub standaryzacji cech tak, by uniknąć nadmiernego., nieuprawniego wpływu jednej z nich na ostateczne wyniki tylko ze względy na skalę, na której ta cecha operuje. W drugim kroku obiekty reprezentowane jako wektory liczb muszą być ze sobą porównane. Proces ten angażuje pewną miarę podobieństwa lub odległości. Następnie powinniśmy określić kryterium, które będzie optymalizowane przy grupowaniu lub regułę, zgodnie z którą obiekty będą łączone w grupy albo tworzone będą grupy obiektów. Wreszcie stosujemy właściwy algorytm, co zwykle wymaga określenia dla niego wartości pewnych parametrów. Uzyskane wyniki powinny zostać poddane walidacji oraz interpretacji w kontekście konkretnego przypadku użycia.

Popular Similarity Measures for Clustering

Feature vectors: • $X = < x_1, x_2, \ldots, x_n >$ • $Y = < y_1, y_2, \ldots, y_n >$

***Euclidean distance*(x,y)** = $\sqrt{(x_1 - y_1)^2 + \ldots. + (x_n - y_n)^2}$

***Manhattan distance*(x,y)** = $|x_1 - y_1| + \ldots. + |x_n - y_n|$

***cosine similarity*(x,y)** = $\frac{\sum_j x_j \cdot y_j}{\sqrt{\sum_j x_j^2} \cdot \sqrt{\sum_j y_j^2}}$

***simple matching distance*(x,y)** = $\sum_j x_j \cdot y_j$

***Jaccard's coefficient*(x,y)** = $\frac{\sum_j x_j \cdot y_j}{\sum_j x_j^2 + \sum_j y_j^2 - \sum_j x_j \cdot y_j}$

***Dice's coefficient*(x,y)** = $\frac{2 \cdot \sum_j x_j \cdot y_j}{\sum_j x_j^2 + \sum_j y_j^2}$

sim: *Sorensen, Czekanowski*
dist: *Hellinger, Bray-Curtis*

***distance*(x,y) = 1 - *similarity*(x,y)**

[13] Przypomnijmy podstawowe miary odległości lub podobieństwa, które wykorzystuje się w kontekście analizy skupień dla obiektów reprezentowanych jako wektory liczb. Podstawowe metryki odległości to tzw. L-normy. Dla L=1, mówimy o odległości taksówkowej (manhattańskiej), a dla L=2, o euklidesowej. W przypadku odległości, im mniejsza jej wartość, tym bardziej podobne porównywane obiekty. Dla miar podobieństwa jest przeciwnie. W tym zakresie, podstawowe miary to podobieństwo kosinusowe i współczynnik Jaccarda, które znacie już z wcześniejszych wykładów. Miary te były w przeszłości wielokrotnie modyfikowane, dając początek innym współczynnikiem, takim jak miara prostego dopasowania (tylko licznik miary kosinusowej) albo miara Dice'a. Warto podkreślić, że często dokonuje się przekształceń między miarami odległości i podobieństwa poprzez obliczenie odwrotności, zanegowanie wartości lub odjęcie od 1. Pozwala ta na zachowanie implementacji algorytmu, który np. operuje na miarach podobieństwa, podczas gdy do porównania obiektów wykorzystano jakąś odległość.

What is Good Clustering?

A **good clustering** method will produce high quality clusters

- high intra-class similarity: cohesive within clusters
- low inter-class similarity: distinctive between clusters

The quality of a clustering method depends on the similarity measure used, its implementation, and its ability to discover some or all of the hidden patterns

Partitioning approach

- Constructs various partitions and evaluates them by some criterion
- K-means, K-medoids, CLARANS

Hierarchical approach

- Hierachical decomposition of the set of data (objects) using some criterion
- Diana, Agnes, BIRCH, CAMELEON

Density-based approach

- Based on connectivity and density functions
- DBSCAN, OPTICS, DenClue

Model-based approach

- A model is hypothetised for each cluster and the best fit of that model is searched
- EM, SOM, COBWEB

More: grid-based (STING, CLIQUE), frequent pattern-based (pCluster), user-guided or constrained-based (COD, constrained clustering)

[14] Przechodzimy do omówienia algorytmów analizy skupień. W tym celu, musimy zrozumieć, jakie są pożądane wyniki takiej analizy. Z jednej strony, chcielibyśmy, aby obiekty w tej samej grupie było do siebie podobne. Z drugie, pożądane jest, by obiekty należące do różnych grup różniły się od siebie znacząco. Istnieje wiele grup algorytmów grupowania. Do najsławniejszych należą algorytmy iteracyjno-opytmalizacyjne bazujące na rozdziale; podejścia hierarchiczne, które opierają się na łączeniu lub rozdziale zgodnie ze ściśle określonym kryterium oraz metody bazujące na głębokości, które analizują połączenia między obiektami i gęstość danych. Z biegiem wykładu, poznamy reprezentantów tych trzech grup. Warto jednak mieć świadomość, że istnieją też inne grupy algorytmów analizy skupień, w których centralną rolę pełni model, analiza zbiorów częstych lub wymagań zdefiniowanych przez użytkownika.

Partitioning Approaches

The notion of comparing item similarities can be extended to clusters themselves, by focusing on a representative vector for each cluster

- **cluster representatives** can be actual items in the cluster or other “virtual” representatives such as the centroid
- reduces the number of similarity computations in clustering
- clusters are **revised successively until a stopping condition is satisfied**, or until no more changes to clusters can be made

Reallocation-Based Partitioning Methods

- Start with an initial assignment of items to clusters and then move items from cluster to cluster to obtain an improved partitioning
- Most common algorithm: ***k-means***

[15] Rozpoczniemy od algorytmów bazujących na rozdziale. Ich centralne założenie jest następujące: podobieństwo między obiektami może zostać uogólnione do podobieństwa obiektów z klastrami, a ściślej rzecz biorąc z ich reprezentantami. Takim reprezentantem może być centroid. Pozwala to na znaczącą redukcję liczby porównań, które musi zrealizować algorytm. Metody w tym duchu mają charakter iteracyjno-optymalizacyjny. Oznacza to, że rozpoczynają swoje działanie od jakiegoś początkowego przydziału obiektów do grup i sukcesywnie go zmieniają w myśl pewnej reguły. Zmiana ta polega na realokacji pewnych obiektów i jest kontynuowana aż do spełnienia warunku stopu. Najsławniejszym reprezentantem tej grupy jest algorytm K-średnich (po angielsku K-means), który zostanie omówiony na kolejnych slajdach.

K-Means Clustering Method - Example (1)

Given the number of desired clusters *K*:

- Randomly assign objects to create ***K*** nonempty initial partitions (clusters)
- Compute the centroids of the clusters of the current partitioning (the centroid is the center, i.e., mean point, of the cluster)
- Assign each object to the cluster with the nearest centroid (**reallocation**)
- Repeat the steps (2 and 3) until the assignment does not change

Initial (arbitrary) assignment: C1={D4}, C2={D6}, C3={D7}

	T1	T2	T3	T4	T5
D1	0	3	2	0	2
D2	2	1	0	1	0
D3	0	3	0	0	2
D4	1	2	0	2	1
D5	0	1	3	0	1
D6	2	0	1	1	2
D7	1	0	2	2	0
D8	3	1	0	0	2

Compute the similarity of each item to each cluster (dot product as the similarity measure):

	D1	D2	D3	D4	D5	D6	D7	D8
C1	**8**	**6**	**8**	**10**	3	6	5	7
C2	6	5	4	6	5	**10**	6	**10**
C3	4	4	0	5	**6**	6	**9**	3

Allocate each document to the cluster to which it has the highest similarity (shown in red in the above table)

C1={D1, D2, D3, D4}, C2={D6, D8}, C3={D5, D7}

End of the first iteration

[16] Algorytm K-średnich zakłada, ze liczba grup, które mają powstać w wyniku jego działania musi być podana przez użytkownika. Jest ona oznacza właśnie przez K. Jego działanie rozpoczyna się najczęściej od losowego przydziału obiektów do K grup (w ten sposób, by żadna z nich nie była pusta). Następnie obliczamy centroid dla każdej z nich i badamy podobieństwo obiektów do centroidów. Każdy obiekt przypisujemy do tej grupy, do której centroidu jest najbardziej podobny. Proces obliczenia centroidu oraz realokacji obiektów kontynuujemy przez z góry zdefiniowaną liczbę iteracji lub do momentu, gdy w dwóch kolejnych iteracjach przydział obiektów do grup nie ulegnie zmianie. W przykładzie na slajdzie chcemy dokonać podziału 8 dokumentów na podstawie 5 termów z wykorzystaniem algorytmu 3-means, czyli K-means, gdzie K=3 jest liczbą grup, które chcemy uzyskać. Wykorzystamy przy tym prostą miarę podobieństwa określoną na jednym z poprzednich slajdów jako simple matching similarity (jest to suma iloczynów odpowiadających sobie współrzędnych dla dwóch wektorów). Rozpoczynamy od losowego przydziału, w którym D4, D6 i D7 trafiły do trzech grup i stanowią ich początkowe centroidy. Następnie badamy podobieństwo wszystkich 8 dokumentów do tych centroidów. Wartości tych podobieństw zaprezentowano w tabeli. Każdy dokument trafia do tej grupy, do której centroidu jest najbardziej podobny, a więc D1 idzie do grupy reprezentowanej przez C1, D2 do C1, ..., D8 do C2. Na końcu iteracji dysponujemy rozdziałem dokumentów na trzy grupy. Przykładowo, w grupie C3 znajdują się D5 i D7.

K-Means Clustering Method - Example (2)

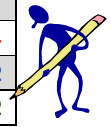
We repeat the process for another reallocation…

… starting from: C1={D1, D2, D3, D4}, C2={D6, D8}, C3={D5, D7}

	T1	T2	T3	T4	T5
D1	0	3	2	0	2
D2	2	1	0	1	0
D3	0	3	0	0	2
D4	1	2	0	2	1
D5	0	1	3	0	1
D6	2	0	1	1	2
D7	1	0	2	2	0
D8	3	1	0	0	2

Compute new cluster centroids using the original document-term matrix

	T1	T2	T3	T4	T5
C1	**3/4**	**9/4**	**2/4**	**3/4**	**5/4**
C2	**5/2**	**1/2**	**1/2**	**1/2**	**4/2**
C3	**1/2**	**1/2**	**5/2**	**2/2**	**1/2**



Compute a new centroid-doc similarity matrix:

	D1	D2	D3	D4	D5	D6	D7	D8
C1	**10.25**	4.5	**9.25**	**8**	5	5.25	3.25	7
C2	6.5	**6**	5.5	6.5	4	**10**	4.5	**12**
C3	7.5	2.5	2.5	4	**8.5**	5.5	**7.5**	3

Reallocate the items to clusters with the highest similarity: C1={D1, D3, D4}, C2={**D2**, D6, D8}, C3={D5, D7}

End of the second iteration

[17] Wynik poprzedniej iteracji stanowi wejście dla iteracji następnej. Obliczamy więc nowe centroidy. Przykładowo centroid C3 powstał z uśrednienia reprezentacji D5 i D7 na pięciu termach T1-T5. Centroidy są teraz wirtualnymi reprezentantami poszczególnych grup - wirtualnymi, bo nie odpowiadają żadnym istniejącym dokumentom. Następnie badamy podobieństwo wszystkich dokumentów do nowo-obliczonych centroidów i dokonujemy przydziału. Przykładowo, D1 zostaje w C1, a D8 w C2, ale już D2 zmienia swoją grupą z C1 na C2. W związku z tym, że w stosunku do poprzedniej iteracji podział uległ zmianie, musimy kontynuować.

K-Means Clustering Method - Example (3)

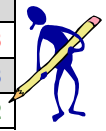
We repeat the process for another reallocation…

… starting from: C1={D1, D3, D4}, C2={D2, D6, D8}, C3={D5, D7}

	T1	T2	T3	T4	T5
D1	0	3	2	0	2
D2	2	1	0	1	0
D3	0	3	0	0	2
D4	1	2	0	2	1
D5	0	1	3	0	1
D6	2	0	1	1	2
D7	1	0	2	2	0
D8	3	1	0	0	2

Compute new cluster centroids using the original document-term matrix

	T1	T2	T3	T4	T5
C1	1/3	8/3	2/3	2/3	5/3
C2	7/3	2/3	1/3	2/3	4/3
C3	1/2	1/2	5/2	2/2	1/2



Compute a new centroid-doc similarity matrix:

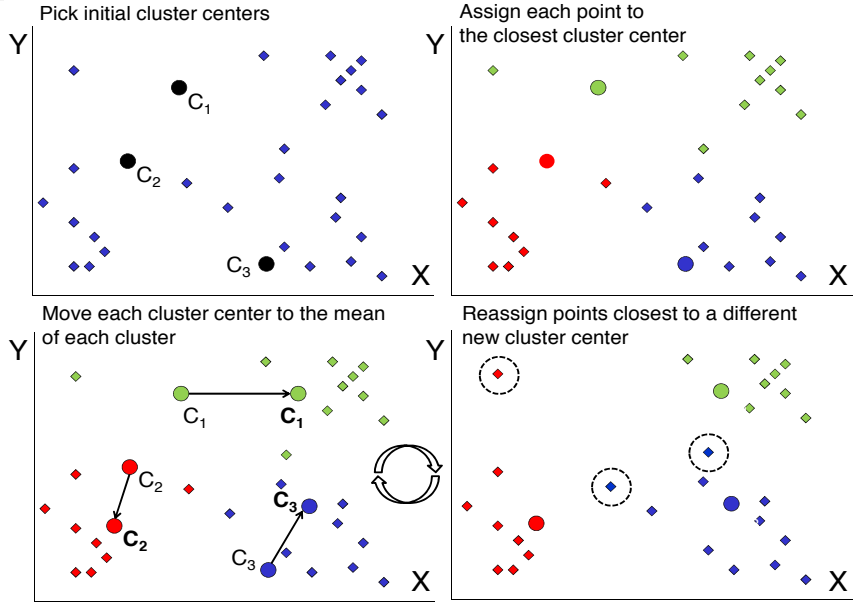
	D1	D2	D3	D4	D5	D6	D7	D8
C1	**12.67**	4	**11.33**	**8.67**	6.33	5.33	3	7
C2	5.33	**6**	4.67	6.33	3	**8.33**	4.33	**10.33**
C3	7.5	2.5	2.5	4	**8.5**	5.5	**7.5**	3

Reallocate the items to clusters with the highest similarity: C1={D1, D3, D4}, C2={D2, D6, D8}, C3={D5, D7}

No change to the clusters ⇨ *terminate the algorithm*

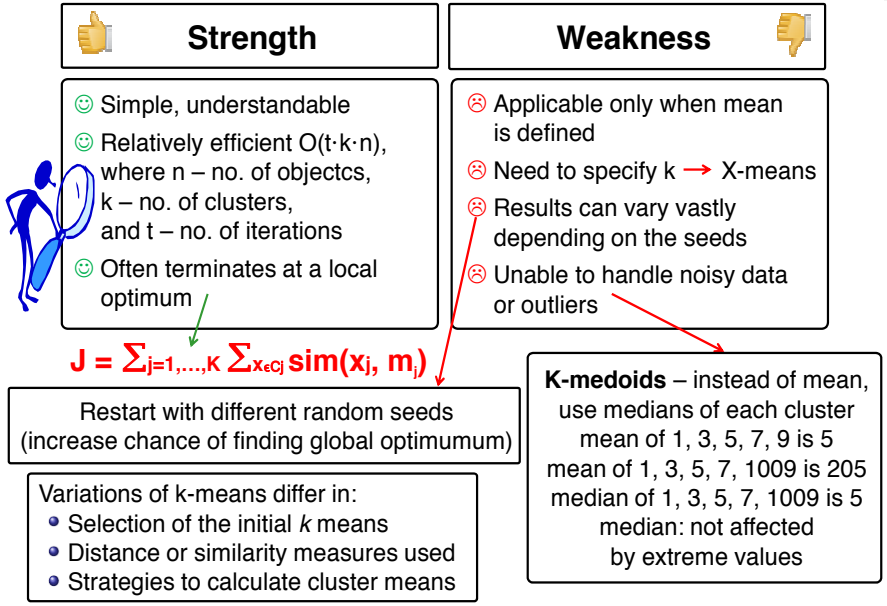
[18] Wyjście poprzedniej iteracji stanowi wejście dla iteracji kolejnej. Zmiana w grupach pociąga za sobą zmianę centroidów (patrz C1 i C2, ale już nie C3, bo tej grupy zmiany nie dotyczyły). Znów badamy podobieństwo dokumentów do nowych centroidów. Okazuje się, że mimo iż podobieństwa liczbowe uległy zmianie, to sam przydział już się nie zmienił. D1 pozostało w grupie C1, D2 w C2, itd. A skoro tak, to centroidy by się dalej nie zmieniły i żadna realokacja do końca wszechświata nie miałaby miejsca. Możemy więc zakończyć działanie algorytmu z podziałem na trzy grupy przedstawione na dole slajdu.

K-Means Clustering Method



[19] Aby powtórzyć kroki działania algorytmu K-średnich, przedstawmy je raz jeszcze, ale tym razem w postaci graficznej. Będzie interesował nas widok z góry, dla kolekcji złożonej z kilkudziesięciu obiektów reprezentowanych jako punkty w przestrzeni dwuwymiarowej. Rozpoczynamy od losowego wyboru centroidów (lewy górny rysunek). Przypisujemy obiekty do grup, do których centroidów są najbardziej podobne (prawy górny rysunek - kolory zielony, czerwony i niebieski determinują podział na trzy grupy). Obliczamy nowe centroidy (lewy dolny rysunek) i dokonujemy realokacji (prawy dolny rysunek). Niektóre z obiektów zmieniły grupę, do której przynależą. W związku z tym, iteracyjno-optymalizacyjny proces ulepszania naszych skupień należy kontynuować.

K-Means Clustering Method - Summary



[20] Omawianie algorytmu K-średnich zakończmy, podsumowując jego zalety i wady, a także wybrane rozszerzenia. Niewątpliwymi zaletami są jego prostota i stosunkowo niska złożoność obliczeniowa, która bierze się z uniknięcia porównania wszystkich obiektów parami dzięki wykorzystaniu centroidów. Pewną zaletą jest też gwarancją znalezienie stosunkowo dobrego rozwiązania, a więc optimum lokalnego względem tzw. miary J. Miara ta bada, na ile podobne są obiekty wewnątrz każdej grupy do jej centroidu i sumuje takie podobieństwa po wszystkich grupach, promując wysokie podobieństwo wewnątrz grupy, a niskie między grupami. Wadą K-średnich jest fakt, że algorytm ten jest stosowalny w kontekście cech liczbowych, które pozwalają na obliczenie średniej. Niewątpliwą niedogodnością jest też konieczność określenia liczby grup K, które chcemy uzyskać. Problem ten adresuje algorytm X-means. Poza tym wyniki mogą się znacząco różnić w zależności od początkowych centroidów, które determinuje ostateczny rezultat. Jest to jednocześnie szansa na wyskoczenie z optimum lokalnego. Można bowiem algorytm zapuścić wielokrotnie dla różnych początkowych centroidów, zwiększając szanse na znalezienie optimum globalnego. Wreszcie metoda K-średnich jest też podatna na obserwacje odstające. Wystarczy przeanalizować przykład w prawym dolnym rogu, żeby uświadomić sobie, jak bardzo średnia ulega zaburzenia, gdy jedna wartość bardzo odstaje od pozostałych. Z tą wadę walczy algorytm K-medoids, który zamiast średniej rozważa medianę, mniej podatną na wartości odstające. Oczywiście modyfikacji algortymu K-średnich istnieją tysiące. Do często ulepszanych aspektów należą sposób wybory początkowych centroidów, wykorzystywane miary podobieństwa i odległości, a także strategie obliczania reprezentantów grup, którymi wcale nie muszą być przecież centroidy.

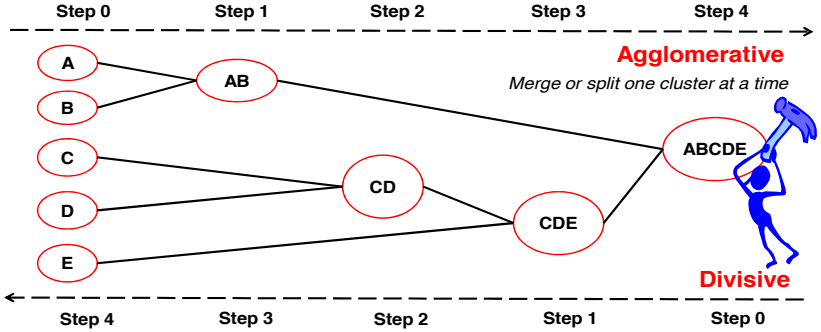
Hierarchical Clustering Approaches

Two main types of hierarchical clustering

Agglomerative
- Start with the points as individual clusters
- At each step, merge the closest pair of clusters until a stopping criterion (e.g., one cluster left)

Divisive
- Start with one, all-inclusive cluster
- At each step, split a cluster until a stopping criterion is met (e.g., each cluster contains a point)

Traditional hierarchical algorithms use a similarity or distance matrix

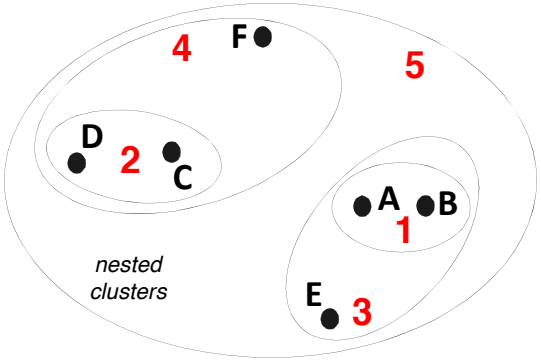


[21] Druga grupa metod grupowania, którą omówimy, to podejścia hierarchiczne. Można wśród nich wyróżnić metody aglomeracyjne (zlepkowe) oraz dzielące (rozłamowe). Te pierwsze rozpoczynają od grup, składających się z pojedynczych obiektów. W każdym kroku łączą zaś dwie grupy najbardziej do siebie podobne aż do spełnienia warunku stopu (np. utworzenia jednej, wielkiej grupy). W przykładzie na slajdzie, w pierwszej kolejności połączono obiekty A i B, w drugiej C i D, w trzecie grupę składającą się z C i D (a więc CD) z E, a w ostatniej grupy AB i CDE. Dla odmiany w algorytmie rozdziałowym rozpoczynamy od grupy zawierającej wszystkie obiekty, a w każdym kroku dokonujemy podziału jednej grupy, który przynosi nam największą korzyść. Metodę kontynuujemy do spełnienia warunku stopu (np. aż każdy obiekt będzie w osobnej grupie). Przykładowo, w przykładzie na slajdzie w pierwszej kolejności dzielimy grupę ABCDE na dwie grupy AB oraz CDE, potem dzielimy CDE na CD i E, itd. Warto zwrócić uwagę, że na rysunku nie ma pełnej odpowiedniości między krokami dla algorytmów aglomeracyjnego i dzielącego (dla tego drugiego trzeba by dokonać przesunięć grup w prawo w graficznej reprezentacji).

Hierarchical Agglomerative Clustering

Basic procedure

- Place each of N items into a cluster of its own
- Compute all pairwise item-item similarity coefficients
- Form a new cluster by **combining the most similar pair** of current clusters C_i and C_i
  - Update similarity matrix by deleting rows/columns corresponding to C_i and C_j
  - Calculate the entries in the row corresponding to the new cluster C_{i+j}
- Repeat step 3 (forming a new cluster) until a stopping criterion is met



Methods for computing **similarity between clusters**:

- single-link
- complete link
- group average
- centroid method

[22] Bardziej intuicyjne i łatwiejsze w implementacji, bo mnie złożone, jest podejście aglomeracyjne. W związku z tym, to na nim się skupimy. Jego angielsku nazwa to Hierachical Agglomerative Clustering, w skrócie HAC. Podsumujmy kroki tego podejścia. Inicjalizując algorytm, umieszczamy każdy obiekt w innej grupie. Potem badamy podobieństwo między wszystkimi grupami (początkowo są to oryginalne obiekty). Kluczowy krok polega na połączeniu w nową grupę dwóch klastrów, które są do siebie najbardziej podobne. Po takim połączeniu należy uaktualnić macierz podobieństwa (powstała przecież nowa grupa). Kroki te, tj. łączenie najbardziej podobnych grup i uaktualnienie macierzy podobieństw, powtarza się aż do napotkania warunku stopu (o tym później). W przykładzie na slajdzie, w pierwszej kolejności łączone są obiekty A i B, potem C i D, następnie AB z E, itd. Warto jednak zwrócić uwagę na fakt, że aby dokonywać łączenia najbardziej podobnych grup, musimy mieć możliwość obliczania podobieństw między grupami (a nie tylko pojedynczymi obiektami). Istnieją 4 podstawowe metody, które pozwalają na realizację tego cely. Opierają się one na analizie pojedynczego połączenia, wszystkich połączeń, średniego połączenia lub połączeń między reprezentantami grup. Omówimy je na kolejnych slajdach.

HAC - Distance Between Two Clusters (1)

Single-link distance between clusters $\mathbf{C_i}$ and $\mathbf{C_j}$ is the ***minimum distance*** between any object in $\mathbf{C_i}$ and any object in $\mathbf{C_j}$

The distance is defined by the two **closest** objects (data points):

$$\boldsymbol{dist}(\mathbf{C_i},\mathbf{C_j}) = \min_{x,y}\{\text{dist}(x,y\} : x \in C_i,\ y \in C_j\}$$

Single-link similarity between clusters $\mathbf{C_i}$ and $\mathbf{C_j}$ is the ***maximum similarity*** between any object in $\mathbf{C_i}$ and any object in $\mathbf{C_j}$

The similarity defined by the two **most similar** objects:

$$\boldsymbol{sim}(\mathbf{C_i},\mathbf{C_j}) = \max_{x,y}\{\text{sim}(x,y\} : x \in C_i,\ y \in C_j\}$$

It can find arbitrarily shaped clusters, but may cause the undesirable "chain effect" due to noisy points

[23] W podejściu bazującym na pojedynczym połączeniu (ang. single link) odległość między grupami jest zdefiniowana jako najmniejsza odległość między obiektami znajdującymi się w tych grupach. Analizujemy więc odległość między wszystkimi parami z dwóch grup i bierzemy z tego wartość minimalną, a więc najbardziej dla grup korzystną. Analogicznie można zdefiniować podobieństwo dla tego podejścia, ale w związku z różnicą interpretacyjną, tu wzięlibyśmy podobieństwo największe dla jakiejkolwiek pary obiektów, należących do analizowanych grup. Zaletą wykorzystanie tej miary jest możliwość znalezienia grup o arbitralnych kształtach, a wadą fakt, że może ona prowadzić do utworzenia niepożądanych łańcuchów obiektów. Wystarczy być podobnym do jednego obiektu, by załapać się do danej grupy, co jest też kłopotliwe, gdy występują obserwacje odstające.

HAC - Example Incorporating Single-Link Similarity

Similarity matrix

	D1	**D2**	D3	D4	D5
D1	1	**0.9**	0.1	0.65	0.2
D2	**0.9**	1	0.7	0.6	0.5
D3	0.1	0.7	1	0.4	0.3
D4	0.65	0.6	0.4	1	0.8
D5	0.2	0.5	0.3	0.8	1

sim(D12,D3) = **max**{sim(D1,D3), sim(D2,D3)} = **max**{0.1, 0.7} = 0.7

	D12	D3	D4	D5
D12	1	0.7	0.65	0.5
D3	0.7	1	0.4	0.3
D4	0.65	0.4	1	**0.8**
D5	0.5	0.3	**0.8**	1

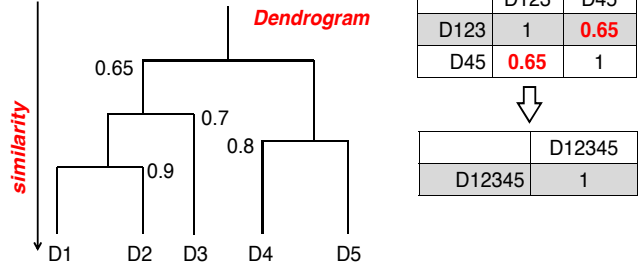
	D12	D3	D45
D12	1	**0.7**	0.65
D3	**0.7**	1	0.4
D45	0.65	0.4	1

sim(D12,D45) = **max**{sim(D12,D3), sim(D12,D4)} = **max**{0.65, 0.5} = 0.65

	D123	D45
D123	1	**0.65**
D45	**0.65**	1

	D12345
D12345	1

Possible stopping criteria:
- number of clusters
- similarity thresholds (do not combine clusters which are not similar)



[24] Przykład dotyczy grupowania 5 dokumentów, dla których dysponujemy macierzą podobieństwa (pomijamy, jak została ona obliczona). Wykorzystamy miarę podobieństwa między grupami bazującą na podejściu single link. Ogromnie ważne jest to, że niezależnie od tego, jak to podobieństwo jest zdefiniowane, w HAC zawsze łączymy grupy najbardziej podobne. W naszym wypadku, w pierwszej iteracji są to dokumenty D1 i D2. Musimy uaktualnić macierz podobieństwa dla porównania grupy D12 z pozostałymi dokumentami (grupami). Zwróćcie uwagę, że na głównej przekątnej takiej macierzy zawsze są 1. Dalej, gdy zestawić D12 z D3, to analizujemy podobieństwo między D1 i D3 oraz D2 i D3, a więc 0.1 i 0.7, i bierzemy z tego maxa, czyli 0.7. W kolejnym kroku najbardziej podobne są do siebie grupy D4 i D5. Analogicznie uaktualniamy macierz podobieństwa. Na slajdzie przedstawiono obliczenia dla porównania D12 i D45. Dalej łączymy D12 z D3, a w ostatnim kroku D123 z D45 w wielką grupę, zawierającą wszystkie obiekty. Proces grupowania dokumentów można przedstawić w postaci graficznej z wykorzystaniem dendrogramu. Każdy dokument jest tu reprezentowany jako osobny punkt, z którego wychodzi pionowa linia. Są one łączone poziomą linią tym niżej, im szybciej nastąpiło połączenie obiektów w grupę. Dodatkowo dendrogram można opisać podobieństwami, z którymi następowało łączenie (patrz np. 0.9 dla połączenia D1 i D2 albo 0.65 dla łączenia na najwyższym poziomie). Analiza takiego dendrogramu nie dość, że pozwala zrozumieć,

HAC - Distance Between Two Clusters (2)

Complete-link distance between clusters $\mathbf{C_i}$ and $\mathbf{C_j}$ is the ***maximum distance*** between any object in $\mathbf{C_i}$ and any object in $\mathbf{C_j}$

The distance is defined by the two **furthest** objects (data points):

$$dist(C_i,C_j) = \max_{x,y}\{dist(x,y) : x \in C_i,\ y \in C_j\}$$

Complete-link similarity between clusters $\mathbf{C_i}$ and $\mathbf{C_j}$ is the ***minimum similarity*** between any object in $\mathbf{C_i}$ and any object in $\mathbf{C_j}$

The similarity defined by the two **least similar** objects:

$$sim(C_i,C_j) = \min_{x,y}\{sim(x,y) : x \in C_i,\ y \in C_j\}$$

It is sensitive to outliers because they are far away from each other

[25] W podejściu bazującym na wszystkich połączeniach (ang. complete link) odległość między grupami jest zdefiniowana jako największa odległość między obiektami znajdującymi się w tych grupach. Analizujemy więc odległość między wszystkimi parami z dwóch grup i bierzemy z tego wartość maksymalną, a więc najmniej korzystną. Analogicznie można zdefiniować podobieństwo dla tego podejścia, ale w związku z różnicą interpretacyjną, tu wzięlibyśmy podobieństwo najmniejsze dla jakiejkolwiek pary obiektów, należących do analizowanych grup. Grupowanie hierarchiczne wykorzystujące tę miarę tworzy bardzo zwarte grupy, bo żeby się do jakiejś grupy załapać trzeba być podobnym do wszystkich obiektów, do niej należących. Wadą zaś jest tu podatność na obserwacje odstające.

HAC - Example Incorporating Complete-Link Similarity

Similarity matrix

	D1	**D2**	D3	D4	D5
D1	1	**0.9**	0.1	0.65	0.2
D2	**0.9**	1	0.7	0.6	0.5
D3	0.1	0.7	1	0.4	0.3
D4	0.65	0.6	0.4	1	0.8
D5	0.2	0.5	0.3	0.8	1

sim(D12,D3) = **min**{sim(D1,D3), sim(D2,D3)} = **min**{0.1, 0.7} = 0.1

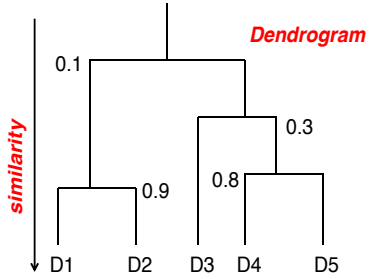
	D12	D3	D4	D5
D12	1	0.1	0.6	0.2
D3	0.1	1	0.4	0.3
D4	0.6	0.4	1	**0.8**
D5	0.2	0.3	**0.8**	1

	D12	D3	D45
D12	1	0.1	0.2
D3	0.1	1	**0.3**
D45	0.2	**0.3**	1

sim(D12,D45) = **min**{sim(D12,D3), sim(D12,D4)} = **min**{0.6, 0.2} = 0.2

	D12	D345
D12	1	**0.1**
D345	**0.1**	1

	D12345
D12345	1



[26] Rozważmy ten sam przykład co 2 slajdy temu, ale z miarą podobieństwa bazującą na complete link. W pierwszym kroku łączymy D1 i D2. Uaktualniając macierz podobieństwa, tym razem analizując podobieństwo dla wszystkich par obiektów należących do dwóch grup, bierzemy z nich mina. Przykładowo, dla porównania D12 z D3, mamy D1 i D3 oraz D2 i D3, czyli 0.1 i 0.7, i bierzemy 0.1. Warto ponownie podkreślić, że niezależnie od tego, jak zdefiniowana jest miara podobieństwa, w każdym kroku łączymy grupy najbardziej podobne. Zaznaczam to ponownie w tym momencie, bo niektórzy błędnie rozumują, że skoro wykorzystuje się complete link, to trzeba łączyć obiekty najmniej podobne. Nic bardziej mylnego - zawsze w HAC łączymy grupy najbardziej podobne. W kolejnych etapach są to: D4 i D5, D3 i D45, D12 i D345. Widać więc różnicę w przebiegu grupowania w stosunku do wykorzystania miary single link.

HAC - Distance Between Two Clusters (3)

Average-link distance between clusters C_i and C_j is the average distance of all pair-wise distances between the data points in two clusters

$$dist(C_i,C_j) = \text{average}_{x,y}\{\text{dist}(x,y) : x \in C_i,\ y \in C_j\}$$

A compromise between:

- the sensitivity of complete-link clustering to outliers
- the tendency of single-link clustering to form long chains that do not correspond to the intuitive notion of clusters as compact, spherical objects

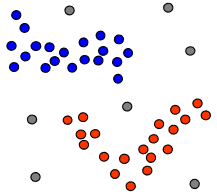
Centroid method: the distance between two clusters is the distance between their centroids

[27] Oprócz miar opartych na pojedynczym połączeniu lub wszystkich połączeniach, do porównania grup można wykorzystać średnią odległość/podobieństwo lub odległość/podobieństwo między reprezentantami grup. W tym pierwszym, analizowane są wszystkie pary i odległości/podobieństwa między nimi są uśredniane. Stanowi to pewnego rodzaju kompromis między wrażliwością podejścia bazującego na wszystkich połączeniach oraz tendencją procedury opierających się na pojedynczym połączeniu do tworzenie długich łańcuchów obiektów. Warto podkreślić, że o ile w single link czy complete link, każdą kolejną macierz podobieństwa można uzyskać z macierzy z poprzedniej iteracji, biorąc mina lub maxa z odpowiednich wartości, to w tym wypadku trzeba odwołać się zawsze do macierzy oryginalnej podobieństw między dokumentami. W drugim podejściu, najczęściej wykorzystywanymi reprezentantami są centroidy. Zakłada się więc, że grupy są na tyle podobne na ile podobne do siebie są ich centroidy.

Density-based Clustering Approaches

Why density-based clustering methods?

- Discover clusters of **arbitrary shape** in spatial databases with noise
- Clusters – **dense regions of objects** separated by regions of low density



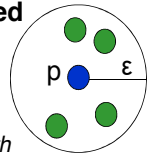
DBSCAN: Density Based Spatial Clustering of Applications with Noise

- For any point in a cluster, its local density has to exceed some threshold
- **The set of points from one cluster is spatially connected**
- Local point density at ***p*** defined by two parameters

  ε: *radius for the neighborhood of point* **p**

  $N_\varepsilon(\mathbf{p}) = \{\mathbf{q}$ in data set $\mathbf{D}$ | $dist(\mathbf{p},\mathbf{q}) \leq \varepsilon\}$

  M_{pts}: minimum number of points in $N_\varepsilon(\mathbf{p})$

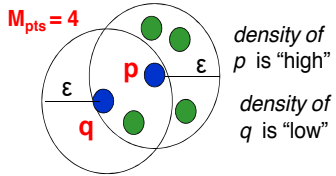


point p *with* 5 *points in* $N_\varepsilon(\mathbf{p})$

[28] Inna grupa algorytmów analiza skupień opiera się na analizie gęstości danych. Podejścia takie pozwalają na odkrywanie grup o arbitralnych kształtach w wielowymiarowej przestrzeni danych przy braku wrażliwości na obserwacje odstające. Grupy są tu rozumiane jako gęste obszary danych (obiektów), które oddzielone są obszarami o mniejszej gęstości. Omówimy zasadę działania algorytmu DBSCAN. Kluczowe dla jego zrozumienia jest pojęcie gęstości. Z punktu widzenia pojedynczej obserwacji (obiektu), ta gęstość definiowana jest przez dwa parametry. Pierwszym jest epsilon, który definiuje sąsiedztwo, a dokładniej maksymalną odległość, przy której możemy uznać jakiś inny obiekt za sąsiada. Liczba takich sąsiadów dla punktu p oznacza jest przez N od p. Przykładowo dla punktu na slajdzie rozmiar jego sąsiedztwa to 5, bo 5 punktów (włączając w to sam analizowany punkt) jest oddalone nie więcej niż o epsilon. Drugi parametr, to minimalny próg M liczby sąsiadów, która pozwala uznać sąsiedztwo za wystarczająco duże, a tym samym gęstość w danym obszarze za wysoką.

DBSCAN - Categorization of Objects

- ε-Neighborhood: objects within a radius of ε from an object
- High density: ε-Neighborhood of an object contains at least M_{pts} of objects



Given ε and M_{pts}, categorize the objects into three exclusive groups

CORE

A **core point** has at least M_{pts} points within ε; These are points that are at the interior of a cluster

BORDER

A **border point** has fewer than M_{pts} points within ε, but is in the neighborhood of a core point

OUTLIER

A noise (outlier) point is any point that is neither a core point nor a border point

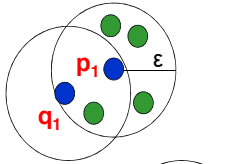
ε = 1unit, $M_{pts} = 4$

[29] Należy więc zapamiętać dwa pojęcia dotyczące sąsiedztwa. Epsilon-sąsiedztwo to obiekty oddalone od danego nie o więcej niż epsilon, a wysoka gęstość to epsilon-sąsiedztwo wystarczająco liczne, tj. spełniające minimalny próg M. Dla punktów na slajdzie: p jest w obszarze o wysokiej gęstości, zaś q w obszarze o niskiej gęstości. Patrząc na zbiór danych globalnie, wszystkie obiekty możemy podzielić na trzy kategorie. Punkt rdzeń (core) to taki, którego sąsiedztwo jest wystarczająco gęste. Punkty takie będą należeć do określonego klastra. Drugi rodzaj to punkt graniczny - jego gęstość jest niska, ale należy on do sąsiedztwa punktu rdzenia; tym samym zostanie pociągnięty przez rdzeń, do klastra do którego ten ostatni należy. Wreszcie obiekt odstający to taki, który nie jest ani rdzeniem ani punktem granicznym.

DBSCAN - Density-reachability and -connectivity

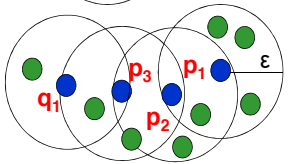
*q is **directly density-reachable** from p*
if p is a core object and q is in p's ε-neighborhood

- density-reachability is asymmetric
- q_1 is directly density-reachable from p_1, but not vice versa



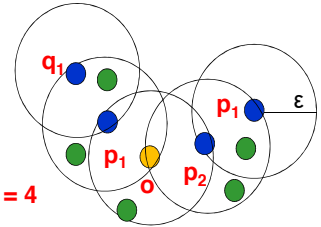
*q is **indirectly density-reachable** from p, if there is a chain of directly-reachable objects that lead from p to q*

- q_1 is directly density-reachable from p_3, p_2 is …
  p_2 is directly density-reachable from p_1



*A pair of points q and p are **density-connected** if they are commonly density-reachable from a point o*

- density-connectivity is symmetric
  - q_1 and p_1 are density-connected



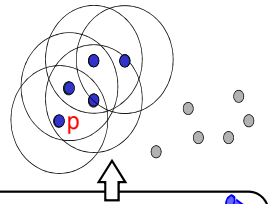
[30] Kluczowe dla działania DBSCAN są pojęcia osiągalności (dostępności) oraz łączliwości ("połączalności") punktów. Definiuje się je dla par obiektów przy określonych parametrach epsilon oraz M. Punkt q jest bezpośrednio gęstościowo osiągalny z punktu p, o ile p jest jądrem, a q znajduje się w jego sąsiedztwie. Jest to własność asymetryczna, co widać na przykładzie górnego rysunku, gdzie q1 jest bezpośrednio gęstościowo osiągalne z p1, ale na odwrót już to nie działa. Dalej, q jest pośrednio gęstościowo osiągalne z p, o ile istnieje łańcuch bezpośrednio gęstościowo osiągalnych obiektów między q i p. Dla rysunku środkowego z p1 do q1 nie można dojść bezpośrednio, ale poprzez rdzenie p2 i p3 już się da. Wreszcie punkty p i q są gęstościowo połączone, jeśli są pośrednio gęstościowo osiągalne z tego samego punktu o. Tak jest na dolnym rysunku dla q1 i p1. Ta ostatnio własność jest symetryczna.

The DBSCAN Clustering Algorithm

Given a data set D, parameter ε and threshold M_{pts}

A cluster C is a subset of objects satisfying two criteria:

- ***Connected***: ∀ p,q∈C: p and q are density-connected
- ***Maximal***: ∀ p,q: if p∈C and **q is density-reachable from p**, then q∈C (avoid redundancy)



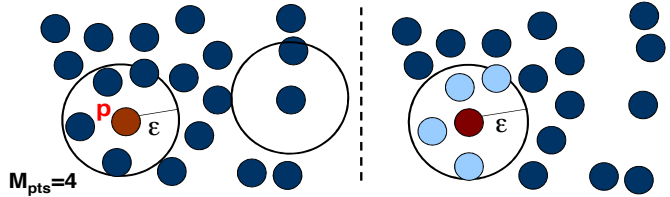
DBSCAN

```
for object p in D //arbitrarily select a point in D
    if p is not yet processed then
        if p is a core object then
            C = retrieve all objects density-reachable from p
            report C as a cluster //mark points in C as processed
        else mark p as outlier //no point density-reachable from p
```

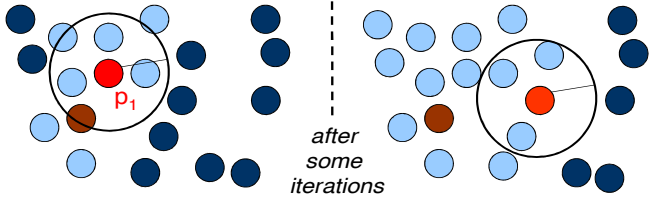
[31] Rozumiejąc pojęcia wprowadzone na poprzednim slajdzie, jesteśmy gotowi, by sformułować algorytm DBSCAN dla zbioru obserwacji D. W myśl jego sposobu działania, grupa (klaster) to zbiór obiektów, które są połączone, tzn. każda para obiektów w grupie jest gęstościowo połączona, oraz maksymalny, tzn. jeśli p należy do grupy, to każdy obiekt gęstościowo osiągalny z p też musi do tej grupy należeć. Algorytm działa w pętli poprzez wybranie losowego punktu ze zbioru D. Jeśli punkt ten nie był jeszcze przetwarzany, to jeśli jest obiektem rdzeniem, to tworzymy grupę C poprzez identyfikację wszystkich punktów gęstościowo osiągalnych z tego obiektu. Dodatkowo wszystkie punkty z C oznaczamy jako przetworzone. Jeśli wylosowany punkt nie był rdzeniem, to jest oznaczany jako obserwacja odstająca. Jeśli w dalszej fazie działania algorytmu pojawi się jakiś rdzeń, z którego można do niego dojść, to załapie się do odpowiedniej grupy. W przeciwnym razie, pozostanie poza utworzonymi grupami. Algorytm kontynuowany jest tak długo aż wszystkie punkty będą przetworzone, tj. znajdą się albo w grupach jako rdzenie lub punkty z nich gęstościowo osiągalne albo poza grupami jako obserwacje odstające.

DBSCAN - Clustering Example

- If **p** has at least $\mathbf{M_{pts}}$ neighbors, put all its neighbors in cluster **C** and mark **p** as processed
- Otherwise, mark **p** as outlier and continue with the next object



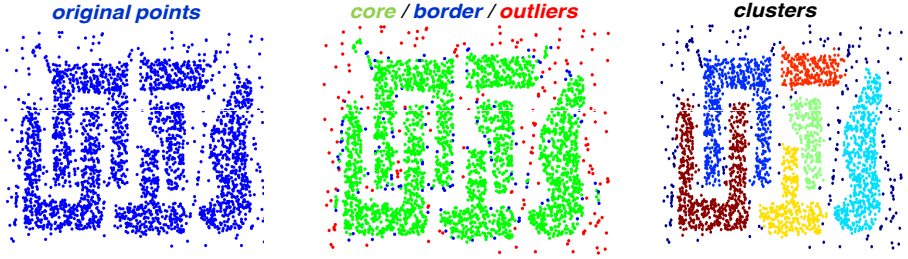
- Check the unprocessed objects in **C**; If no core object, return **C**
- Otherwise, randomly pick up one core object p_1, mark p_1 as processed, and put all unprocessed neighbors of p_1 in cluster **C**



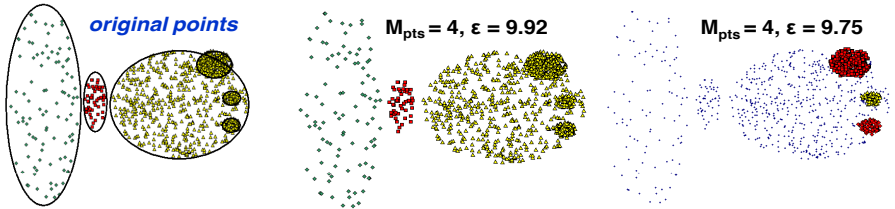
[32] Prześledźmy, jak proces grupowania przebiega dla przykładowego zbioru danych. Dla wylosowanego punktu, jeśli w jego epsilon-sąsiedztwie jest co najmniej M punktów, to jest on rdzeniem. Tworzymy więc grupę C poprzez zagarnięcie wszystkich punktów z niego bezpośrednio lub pośrednio osiągalnych. W praktyce wygląda to tak, że najpierw zagarniamy punkty bezpośrednio osiągalne z wylosowanego rdzenia i sprawdzamy, czy jest wśród nich co najmniej jeden rdzeń. Jeśli tak, to losowo wybieramy jakiś rdzeń i zagarniamy obiekty bezpośrednio osiągalne z niego, itd. W ten sposób identyfikujemy punkty pośrednio osiągalnego z rdzenia wylosowanego na samym początku. Jeśli takiego rdzenia nie ma, to kończymy tworzenie klastra. Jeśli natomiast wylosowany na samym początku punktu rdzeniem nie był, to oznaczamy go jako obserwację odstającą i kontynuujemy procedurę, losując punkt kolejny.

When DBSCAN Works/Does Not Work Well?

- Can handle clusters of different shapes and sizes
- Number of clusters is determined automatically
- Resistant to noise

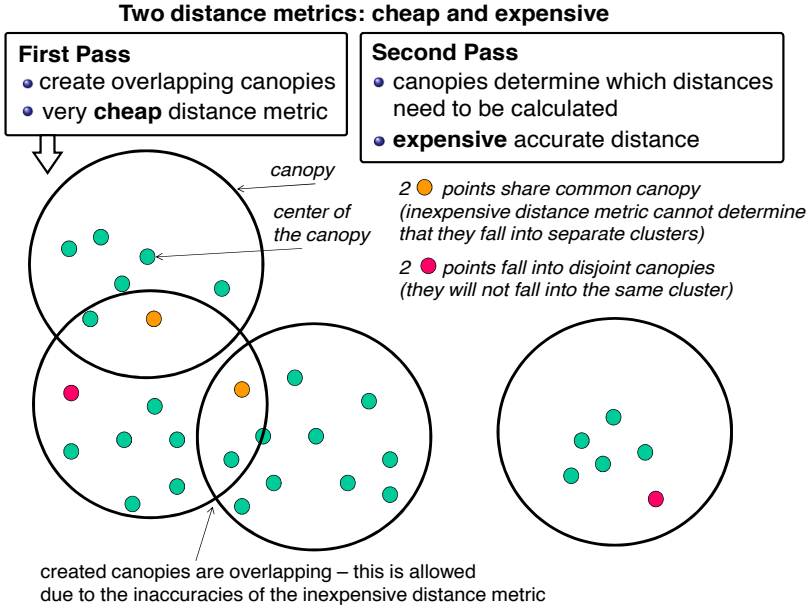


- Cannot handle varying densities
- Sensitive to parameters



[33] Ostatni slajd dotyczący DBSCAN ilustruje przykładowe wyniki jego działania dla dwóch zbiorów danych. Przykład górny uzmysławia, jak liczne są jego zalety. Po pierwsze, tworzone grupy mogą mieć bardzo skomplikowane kształty, ale zawsze są to obszary o wysokiej gęstości przedzielone obszarami o gęstości zdecydowanie niższej. Po drugie, liczba grup jest ustalona automatycznie; nie trzeba jej podawać, wynika ona po prostu z charakterystyki danych. Po trzecie, algorytm jest odporny na obserwacje odstające. Nie wymusza, by wszystkie punkty znalazły się w utworzonych grupach, a obserwacje odstające zostawia na boku tak, jak na to zasługują. Przykład dolny uzmysławia jednak, że nie jest to algorytm doskonały. Z jednej strony, słabo radzi sobie w sytuacjach, gdy w różnych obszarach wielowymiarowej przestrzeni danych ich gęstość jest zróżnicowana. Z drugiej, pokazuje, jak podatny jest DBSCAN na drobne nawet zmiany wartości parametrów (porównaj wartości epsilon na rysunku środkowym i prawym).

Canopy Clustering (1)



[34] Trzecim - obok K-średnich i grupowania hierarchicznego - najczęściej wykorzystywanym algorytmem analizy skupień jest canopy clustering. Metodę tą rzadko wykorzystuje się samodzielnie; najczęściej występuje ona w połączeniu z innymi algorytmami, dostarczając dla nich częściowych wyników. Canopy to po angielsku czasza spadochronu, ale w dalszej części posługiwać się będziemy angielskim pojęciem, a nie jego polskim, zbyt długim odpowiednikiem. Algorytm składa się z dwóch przejść po zbiorze danych, w których potencjalnie można wykorzystać różne miary podobieństwa. W pierwszym przejściu tworzy się canopies, które mogą na siebie nachodzić (jeden obiekt może należeć do wielu canopies). Na tym etapie wykorzystuje się tanią, tj. mało kosztowną w obliczeniach, miarę podobieństwa. Canopy składa się z centrum i obiektów, które są z nim związane, tj. są w niedużej odległości od centrum. W drugim przejściu, zakłada się, że canopies determinują, dla których obiektów trzeba policzyć podobieństwa w sposób bardziej dokładny. Robi się to dla wszystkich par obiektów, które znajdują się w ten samej canopy i w związku z tym, że jest to liczba ograniczona w stosunku do wszystkich par, to można tu wykorzystać drogą, bardziej złożoną miarę podobieństwa. Założenie jest więc takie, że pary w tej samej canopy są uznawane jako potencjalnie grupowalne razem. Natomiast obiekty w różnych canopies nie mają takiej potencjału, tj. na pewno nie znajdą się w tej samej grupie, więc nie ma sensu tracić zasobów na realizację dokładniejszych obliczeń dla takich par.

Canopy Clustering (2)

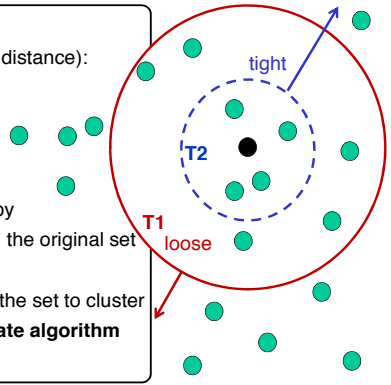
Create canopies

Use two thresholds **T1** (the loose distance) and **T2** (the tight distance):

Begin with the set of data points to be clustered

- Remove a point from the set, beginning a new canopy (*)
- Use inexpensive distance metric
  - For each point left in the set:
    if the distance is less than **T1**, assign it to the new canopy
    if the distance is additionally less than **T2** remove it from the original set
    (so that it cannot become a center of the new canopy)
- Repeat from step (*) until there are no more data points in the set to cluster

Sub-cluster canopies using a more expensive but accurate algorithm
(incorporating expensive distance metric)

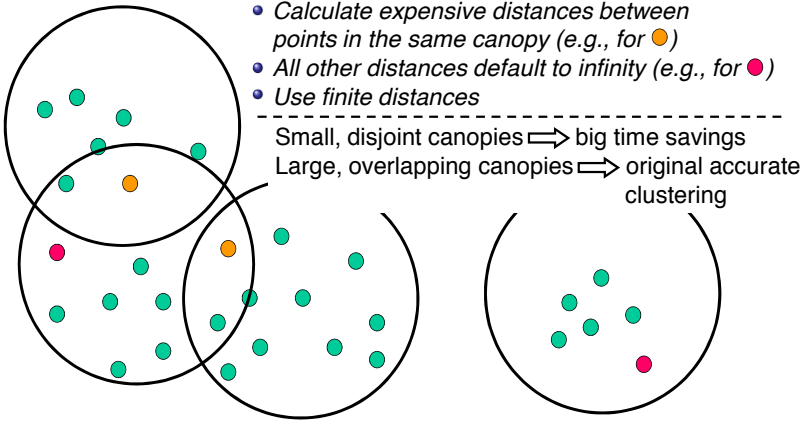


[35] Canopy clustering zakłada wykorzystanie dwóch progów: T1, luźnego progu podobieństwa oraz T2, mniejszego, bardzo bliskiego progu podobieństwa. Zakładamy, że dostępny jest zbiór obiektów do pogrupowania. Wybieramy z niego losowo punkt, który daje początek nowej canopy (na slajdzie jest to punkt czarny). Usuwamy ten punkt z analizowanego zbioru - tym samym centrum jakiejś canopy nie będzie mogło znaleźć się w innej canopy. Następnie wykorzystujemy tanią miarę podobieństwa, aby dla każdego punktu pozostałego w zbiorze obliczyć jego podobieństwo z centrum nowej canopy. Jeśli odległość ta jest mniejsza od T1, to przypisujemy go do tej canopy (na slajdzie punkty wewnątrz czerwonego okręgu). Jeśli dodatkowo odległość ta jest mniejsza od T2, to usuwamy go z analizowanego zbioru. Tym samym, będąc bardzo podobnym do centrum canopy, nie będzie on się mógł stać centrum innej canopy w kolejnych iteracjach (na slajdzie punktu wewnątrz granatowego okręgu). Kroki te (tj. losowanie centrum nowej canopy, porównanie z wszystkimi obiektami nieusuniętymi do tej pory, analizę luźnego i bliskiego powiązania) powtarzamy aż zbiór obiektów do rozważenia będzie pusty. Dopiero w dalszej kolejności poszczególne canopies można pogrupować, korzystając z bardziej dokładnego algorytmu, który może stosować kosztowną obliczeniowo miarę podobieństwa.

Canopy Clustering (3)

Canopy clustering can be combined with other algorithms using more expensive distance metrics:

- K-means – use centers of the canopies as initial centroids and their number to determine K
- Hierachical clustering – iteratively merge the closest



- *Calculate expensive distances between points in the same canopy (e.g., for ●)*
- *All other distances default to infinity (e.g., for ●)*
- *Use finite distances*

Small, disjoint canopies ⇨ big time savings
Large, overlapping canopies ⇨ original accurate clustering

[36] Najczęściej canopy clustering łączone jest z algorytmem K-średnich lub grupowaniem hierarchicznym Dla tej pierwszej kombinacji, centra canopies mogą zostać wykorzystane jako początkowe centroidy, a liczba canopies ustalona jako K. Tym samym algorytm rozwiązuje dwa fundamentalne dla wykorzystania K-średnich problemy. W przypadku HAC, można iteracyjnie łączyć najbardziej podobne obiekty, rozważając jednak już tylko te pary, które znajduję się w tej samej canopy (np. pomarańczowe kropki na slajdzie). To dla nich można obliczyć podobieństwo, korzystając z drogiej miary podobieństwo. Dla obiektów z różnych canopies można to sobie odpuścić (np. różowe kropki na slajdzie). Gdy wynikowe canopies są małe, oszczędność czasu jest ogromna, bo można pominąć liczenie podobieństwa dla ogromnej liczby par. Gdy canopies są duże i przecinają się w wysokim stopniu, zysk jest niewielki, bo właściwy algorytm grupowania musi w drugim etapie zrealizować obliczenia dla niemal wszystkich par.

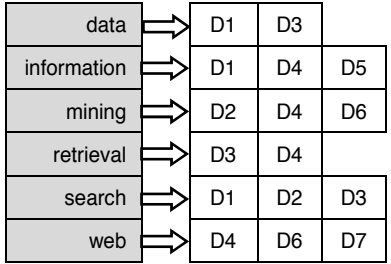
Canopy Clustering - Two-Stage Example

Example problem that can be appropriately dealt with canopies

- 17M biomedical papers in Medline
- Each paper contains ~20 citations
- Clustering 340M citations (which papers cite a given work)
- Remark: *citations have very different formats*
- 10^{17} distance calculations for naïve hierarchical clustering

Solution: first pass

- Word-level matching (TF-IDF)
- Inexpensive using an inv. index
- Two-threshold technique means we can avoid many pair-wise comparisons



Solution: second pass

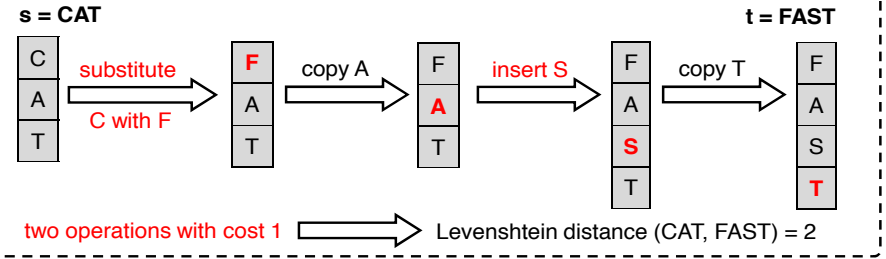
Cluster citations in canopies using edit-distance metric (very expensive)

[37] Przykładowy problem, w którym można by wykorzystać canopy clustering dotyczy grupowania dokumentów. Zrobiono to m.in. dla dokumentów medycznych z bazy Medline. Celem było pogrupowanie dokumentów pod względem prac, które cytują. Każdy artykuł średnio cytował 20 innych prac, co przy 17-milionowej bazie dawało 340 milionów cytowań do analizy. Problem z cytowaniami w świecie nauki jest taki, że ich format nie jest ustandaryzowany. Zdecydowano więc, że wobec ogromnej liczby porównań, którą trzeba by zrealizować dla potrzeb grupowania, w pierwszej kolejności zostanie wykorzystania miara mało kosztowna, operująca na poziomie porównania termów. Taką miarą jest np. podobieństwo bazujące na TF-IDF. Utworzenie z jej wykorzystaniem canopies pozwala w dalszym etapie uniknąć bardzo wielu porównań parami. A skoro tak, to można już pozwolić sobie na wykorzystanie bardziej kosztownej miary podobieństwa, która zmierzy się z problemem porównania różnych formatów cytowań (różne kolejności dat w referencjach, skrótowe lub nie nazwy czasopism, wymienieni wszyscy autorzy lub nie, itd.). Zdecydowano więc, że w drugim etapie do porównania i grupowania cytowani zostanie wykorzystana bardzo kosztowna miara podobieństwa, służąca do porównania odmienności ciągów znaków. Jest nią dystans edycyjny, który zostanie omówiony na kolejnych slajdach.

Edit Distance - Levenshtein Distance

Distance is **shortest sequence of edit commands** that transform *s* to *t*
Simplest set of operations:

- Copy character from *s* over to *t*
- Delete a character in *s* (cost 1)
- Insert a character in *t* (cost 1)
- Substitute one character for another (cost 1 – *can be set to other value*)

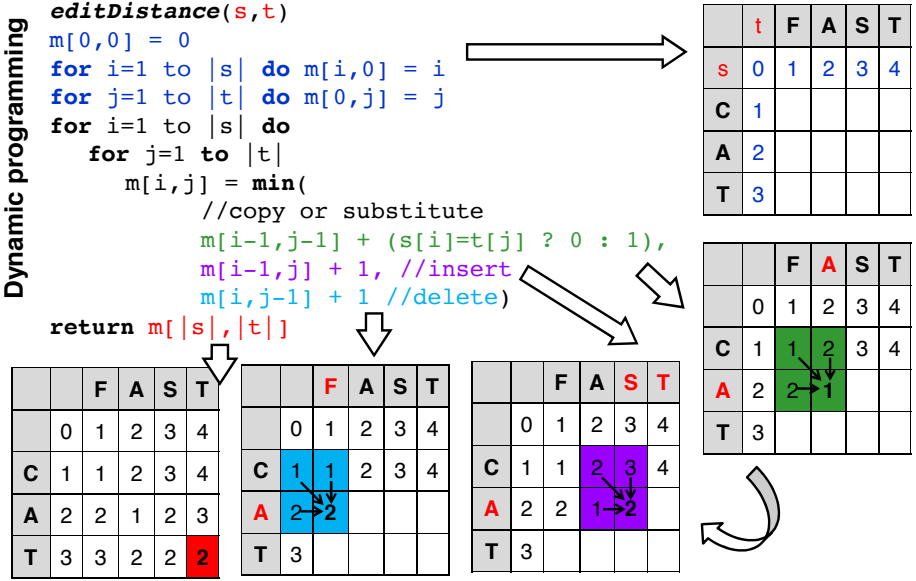


[38] Jako bonus, który można wykorzystać w canopy clustering, omówimy bardzo kosztowną miarę podobieństwa dla termów (lub w ogólności ciągów znaków). Jest to odległość edycyjna lub odległość Leveshteina, która definiujemy jako najmniejsza liczba operacji edycji, które zmieniają jeden ciąg znaków w inny. Operacje edycyjne to kopiowanie znaku z jednego ciągu znaków do drugiego, co nie pociąga za sobą żadnego kosztu oraz trzy operacje, których domyślny koszt jest jednostkowy. Należą do nich: usunięcie znaku, wstawienie znaku lub zastąpienie jednego znaku drugim. Przykładowo, aby porównać ciągi znaków CAT i FAST, dla CAT należałoby najpierw zastąpić C przez F, potem skopiować A, wstawić S i skopiować T. Dwie z tych operacji, a więc zastąpienie C przez F i wstawienie S, kosztują 1, a więc odległość edycyjna między CAT i FAST wynosi 2. Pytanie, jak dokonać takich obliczeń w sposób systematyczny z wykorzystaniem algorytmu.

Levenshtein Distance - Dynamic Programming

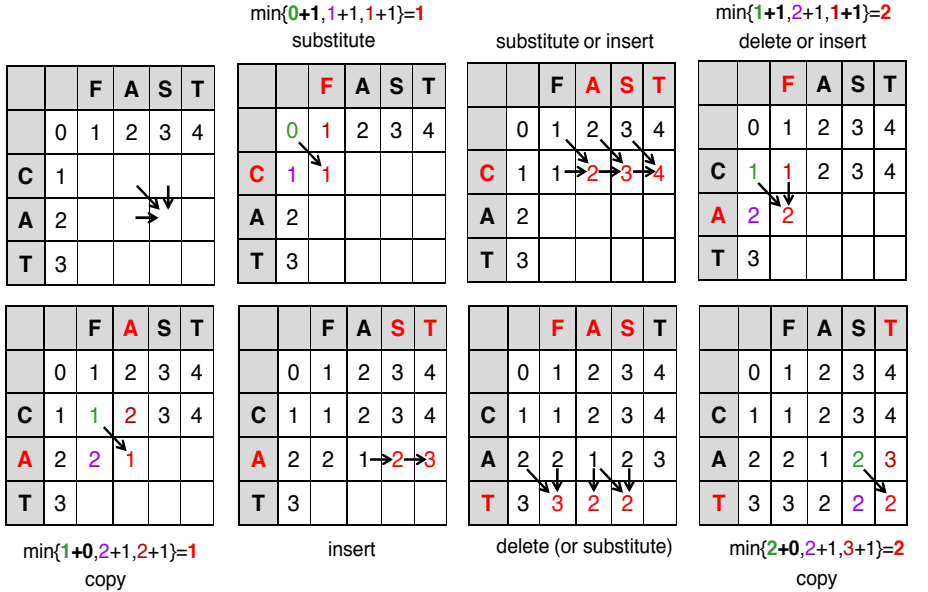
Dynamic programming

```
editDistance(s,t)
m[0,0] = 0
for i=1 to |s| do m[i,0] = i
for j=1 to |t| do m[0,j] = j
for i=1 to |s| do
   for j=1 to |t|
      m[i,j] = min(
            //copy or substitute
            m[i-1,j-1] + (s[i]=t[j] ? 0 : 1),
            m[i-1,j] + 1, //insert
            m[i,j-1] + 1 //delete)
return m[|s|,|t|]
```



[39] Algorytm do obliczenia odległości edycyjnej bazuje na programowaniu dynamicznym, które znacie z przedmiotu Optymalizacja kombinatoryczna, gdzie był wykorzystywany w kontekście rozwiązania problemu plecakowego. Ogólne sformułowanie zakłada, że porównywane są ciągi znaków s i t. W tym celu tworzymy macierz o rozmiarach długość s + 1 na długość t plus 1. Inicjalizując ją, w zerowym wierszu i kolumnie wstawiamy kolejne liczby naturalne, od 0 do długości danego ciągu znaków. Zasada wypełnienia macierzy jest następująca - systematycznie wypełniamy poszczególne komórki, przesuwając się w niej wierszami. Wartość w każdej komórce pochodzi z analizy trzech sąsiadujących już wypełnionych komórek: ze skosu, z góry i z lewej strony. Taka wartość reprezentuje odległość edycyjną między ciągami znaków porównywanymi do tej pory. Komórka ze skosu odpowiada przekopiowaniu lub zastąpienia znaku. Jeśli porównywane aktualnie znaki są takie same, to wystarczy rozważyć wartość ze skosu, bo wtedy możemy znak przekopiować, co nic nie kosztuje. Jeśli te znaki są różne, to wartość ze skosu trzeba zwiększyć o 1, bo wtedy zastępujemy jeden znak innym. Wartość z lewej strony trzeba zawsze zwiększyć o 1, bo odpowiada ona sytuacji, w której jakiś znak musimy wstawić/dodać. Wartość z góry też musimy zawsze zwiększyć o 1, bo modeluje ona scenariusz, gdy jakiś znak musimy usunąć. Z takich roboczych wartość, tj. skos + 0 (znaki te same) lub + 1 (znaki różne), lewa +1 i góra + 1, bierzemy minimum, bo w odległości edycyjnej chodzi o minimalną odległość. Taki sposób postępowania kontynuujemy aż do ostatniej komórki. Wartość, która w niej się znajdzie jest ostatecznym rozwiązaniem Przykład na slajdzie dotyczy porównania termów CAT i FAST. Zaprezentowano na nim m.in. inicjalizację macierzy, scenariusz ze skopiowaniem znaku (porównanie CA i FA), wstawieniem znaku (CA i FAS) i usunięciem znaku (CA i F). Jest też cała wypełniona macierz i w jej prawym dolnym rogu jest ostateczna odległość edycyjna (2).

Levenshtein Distance - Example



[40] Slajd reprezentuje poszczególne kroki wypełnianie macierzy zgodnie z programowaniem dynamicznym. Pamiętaj, że wartość w każdej komórce pochodzi z analizy skosu, góry i prawej. Przykładowo, przy wypełnieniu pierwszej komórki, porównujemy C i F, a więc różne znaki, czyli bierzemy min z (0 + 1, bo skos i różne znaki, 1 + 1, bo lewa zawsze + 1, oraz 1 + 1, bo góra zawsze plus 1). Wychodzi 1, co oznacza, że odległość edycyjna między C i F, to 1 (wynika ona z konieczności zastąpienia jednego znaku drugim). Następnie przesuwamy się w wierszu do kolejnej komórki. W pierwszym wierszu akurat wychodzi tak, że zawsze bierzemy skos + 1, bo C z CAT jest różne od wszystkich znaków w kolumnie. Jeśli spojrzeć na porównanie CA i FA (macierz po lewej na dole), to bierzemy min z (1 + 0, bo skos i te same znaki, 2 + 1, bo lewa zawsze + 1, oraz 2 + 1, bo góra zawsze + 1), czyli 1. Oznacza to, że odległość edycyjna między CA i FA, to 1. W poszczególnych macierzach zaprezentowanych na slajdzie starałem się przedstawić różne sytuacje, które mogą wystąpić przy realizacji algorytmu. Kontynuując wypełnianie wierszami, ostatecznie dochodzimy do końcowego wyniku w prawym dolnym rogu.

Levenshtein Distance with Different Costs

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	3	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19																		
o	91	1	1	3	116	0	0	0																		
p	0	11	1	2	0	6	5	0																		
q	0	0	1	0	0	0	27	0																		
r	0	14	0	30	12	2	2	8																		
s	11	8	27	33	35	4	0	1																		
t	3	4	9	42	7	5	19	5																		
u	20	0	0	0	44	0	0	0																		
v	0	0	7	0	0	3	0	0																		
w	2	2	1	0	1	0	0	2																		
x	0	0	0	2	0	0	0	0																		

- Costs for different substitutions can/should vary

[41] Wspomnieliśmy, że koszty wstawienia, usunięcia i zastąpienia znaku są jednostkowe, a więc takie same. Algorytm będzie jednak działał, gdy koszty te zostaną zróżnicowane. Szczególny potencjał istnieje dla zróżnicowania kosztów zastąpień różnych znaków. W macierzy zaprezentowanej na slajdzie przedstawiono liczby pomyłek znaków dla wybranej kolekcji tekstów. Przykładowo, znaków a i b nie pomylono ze sobą nigdy, ale o z a lub o z e już bardzo często. Sugeruje ta, że koszt takiej zmiany, reprezentującej typową pomyłkę, mógłby być mniejszy niż dla pomyłki, która jest bardzo nietypowa. Poza tym analiza rozkładu liter na klawiaturze także sugeruje możliwość zróżnicowania kosztów pomyłek. Sami wiecie, że szybko pisząc na klawiaturze, bliskie znaki mylą się nam stosunkowo często, a te dalekie - rzadko.

Spelling Correction

Spelling correction: the user typed "graffe"
Which is closest? • graf • graft • grail • giraffe
The problem is appealing…

Misspellings detected by Google for "britney spears" in 3 months in 2003

488941 britney spears	811 brithney spears	220 breatney spears
40134 brittany spears	811 brtiney spears	220 britiany spears
36315 brittney spears	664 birtney spears	199 britnney spears
24342 britany spears	664 brintney spears	163 britnry spears
7331 britny spears	664 briteney spears	147 breatny spears
6633 briteny spears	601 bitney spears	147 brittiney spears
2696 britteny spears	601 brinty spears	147 britty spears
1807 briney spears	544 brittaney spears	147 brotney spears
1635 brittny spears	544 brittnay spears	147 brutney spears
1479 brintey spears	364 britey spears	133 britteney spears
1479 britanny spears	364 brittiny spears	133 briyney spears
1338 britiny spears	329 brtney spears	121 bittany spears
1211 britnet spears	269 bretney spears	121 bridney spears
1096 britiney spears	269 britneys spears	121 britainy spears
991 britaney spears	244 britne spears	121 britmey spears
991 britnay spears	244 brytney spears	109 brietney spears

... and thousands of other variants

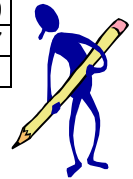
[42] Odległość Levenshteina jest jednym ze środków, który może posłużyć do poprawy literówek. Sensownym podejściem jest bowiem szukanie dla termu nieistniejącego, takiego innego, który byłyby poprawny, ale jednocześnie najbliższy temu, co niepoprawnie napisał użytkownik. A stwierdzenie ?najbliższy' wymaga zastosowania pewnej miary odległości i dystans edycyjny jest tu świetnym kandydatem do zastosowania. Problem poprawy literówek jest bardzo istotny. Jak wielka jest skala problemu, z którym trzeba się zmierzyć, dowodzi przykład ze slajdu. Jest to wykaz literówek dla zapytań dotyczących Britney Spears sprzed kilkunastu lat z okresu 3 miesięcy i to tylko dla błędnych zapisów jej imienia. W całym pliku wariantów tego imienia są tysięcy, a te najczęstsze warianty błędów powtarzają się dziesiątki tysięcy razy.

Summary (1)

Given the TF-IDF representation of five documents and the cosine similarity matrix, use 2-means to group these docs into two clusters:

	D1	D2	D3	D4	D5
T1	0.2	0.5	0.7	0.7	0.8
T2	0.7	0.2	0.6	0.3	0.6

	D1	D2	D3	D4	D5
D1	1	0.61	0.83	0.63	0.79
D2	0.61	1	0.94	1	0.96
D3	0.83	0.94	1	0.95	0.99
D4	0.63	1	0.95	1	0.97
D5	0.79	0.96	0.99	0.97	1



I) Assume the least similar docs are the initial centroids. Which docs would be used?

II) What would be the clustering obtained the first iteration? G1 and G2? Would it be different in case D3 and D4 were used as the centroids?

III) Compute the J measure after the first iteration?

IV) Compute the new centroid after the first iteration (for the case of starting with D1 and D2 as the initial centroids).

	C1	C2
T1	0.2	?
T2	0.7	?

[43] I)
II)
III)
IV)

Summary (2)

Given the cosine similarity matrix for five documents, use agglomerative hierarchical clustering (AHC) to group these docs:

	D1	D2	D3	D4	D5
D1	1	0.61	0.83	0.63	0.79
D2	0.61	1	0.94	1	0.96
D3	0.83	0.94	1	0.95	0.99
D4	0.63	1	0.95	1	0.97
D5	0.79	0.96	0.99	0.97	1

	D1	D24	D3	D5
D1	1	?	0.83	0.79
D24	?	1	?	?
D3	0.83	?	1	0.99
D5	0.79	?	0.99	1

I) Which docs would be clustered together first (irrespective of how the similarity between groups is defined)?

II) Compute the similarity matrix after the first iteration while assuming that the similarity between groups is equal to the maximal/minimal/average similarity of the docs contained in these clusters?

sim(D1,D24) = *max/min/ave*{*sim*(D1,D2),*sim*(D1,D4)} = *max/min/ave*{0.63,0.61}

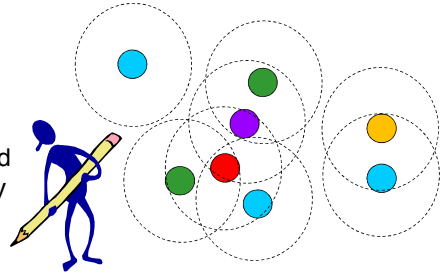
III) Present the process of AHC be means of a dendrogram.

IV) How many groups would be obtained if the similarity threshold for AHC would be set to 0.8?

[44] I)
II)
III)
IV)

Summary (3)

Given the minimal neighborhood threshold M_{pts}=3 and a similarity threshold marked in the figure, use DBSCAN:



I) Is the neighborhood of the red/violet/orange point dense?

II) Mark all points directly/indirectly density reachable from the red/green point?

III) Are the green points density-connected?

IV) Identify the cores/border points and outliers.

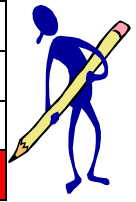
V) Mark the cluster (if any) that would be formed starting from the red/orange point?

[45] I)
II)
III)
IV)
V)

Summary (4)

Compute the Levenshtein distance for ”LEGIA” and ”LECHIA”

		L	E	C	H	I	A
	0	1	2	3	4	5	6
L	1	0					
E	2		0				
G	3			1	2		
I	4					2	
A	5						**2**



editDistance(LEGIA, LECHIA) =