Introduction to Text Mining

Part VII: Text Mining using Clustering

Henning Wachsmuth

https://cs.upb.de/css

Text Mining using Clustering: Learning Objectives

Concepts

- Get to know how to employ clustering within text mining.
- Learn about the role of similarity measures in clustering.
- Understand the pros and cons of different clustering types.

Text analysis techniques

- Learn how to compute the similarity of text spans in various ways.
- Learn how to partition a set of texts into groups with flat clustering.
- Learn how to create soft clusters using topic modeling.
- Learn how to order texts by similarity with hierarchical clustering.

Covered text analyses

- Authorship attribution
- Topic detection
- Discourse pattern recognition

Outline of the Course

I. Overview

II. Basics of Linguistics

III. Text Mining using Rules

IV. Basics of Empirical Research

V. Text Mining using Grammars

VI. Basics of Machine Learning

VII. Text Mining using Clustering

- What Is Text Mining using Clustering?
- Similarity Measures
- Hard and Soft Flat Clustering
- Hierarchical Clustering

VIII. Text Mining using Classification and Regression

IX. Practical Issues

X. Text Mining using Sequence Labeling

What Is Text Mining using Clustering?

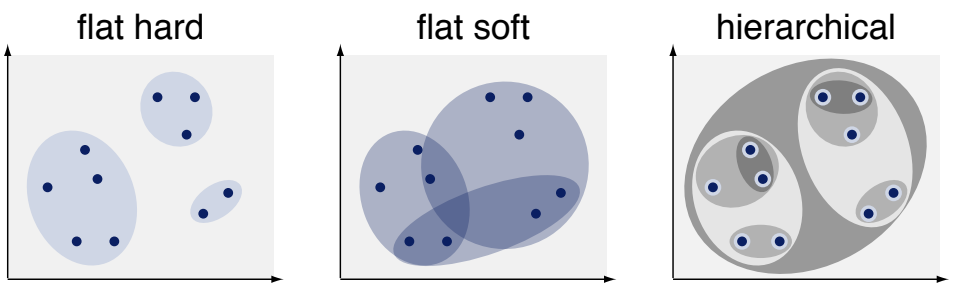
Clustering

What is clustering (aka cluster analysis)?

- The grouping of a set of instances into some number $k \geq 1$ of classes.
  k is possibly, but not necessarily predefined.
- Each resulting group is called a *cluster*.
- The meaning of the classes/clusters is usually unknown beforehand.

Types of clusterings

- Flat vs. hierarchical
- Hard vs. soft



Clustering vs. cluster labeling

- Clustering does not assign labels to the created clusters.
- *Cluster labeling* is all but trivial; it requires to infer the hidden concept connecting the instances in a group.
  Cluster labeling is beyond the scope of this course.

Clustering

Unsupervised Learning

Unsupervised (machine) learning

- Aims to find patterns in unannotated data (without ground truth) that reveals the organization and association of the data.
- A model y is derived from a set of instances $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$ only.

Properties of unsupervised learning

- The learning process is self-organized.
- There is no external “teacher”.
- The optimization criterion is task and domain-independent.

  In supervised learning, it is defined by the target function in a given task or domain.

Clustering as unsupervised learning

- Clustering is mostly approached as an unsupervised learning problem.

  We will see a somewhat supervised variant of clustering below, though.
- In fact, it is the most common unsupervised learning technique.

Clustering

Clustering using Unsupervised Learning

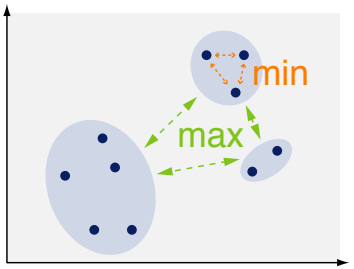
Unsupervised clustering

- Patterns in the instances are learned based on similarity measures.
- The resulting instance clusters correspond to classes.
- The resulting model can assign arbitrary instances to the clusters.

Objective of unsupervised clustering

- Minimize the distance within all clusters.
- Maximize the distance between the clusters.

  Analog: Maximize *similarity* within, mininimize across.



Evaluation of unsupervised clustering

- Rank alternative clusterings wrt. their quality on some test set.
- Determine the ideal number k of clusters.
- Relate found structures to externally provided class information.

Clustering
Similarity Measures

Similarity measures in clustering

- A similarity measure quantifies how similar instances of a concept are.
- Clustering computes similarities to identify instances to be merged.

From similarity to cluster similarity

- To merge clusters, the measures are also computed for clusters.
- Different ways to define cluster similarity exist (details below).

Similarity vs. distance

- Similarity can be seen as the inverse of distance.
- With normalized values, deriving one from the other is straightforward.

Similarity vs. relatedness

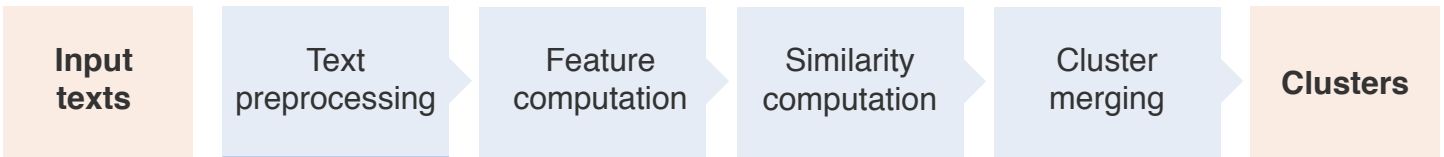
- Similar. Concepts with similar meaning, e.g., “car” and “bike”.
- Related. Dissimilar concepts may still be related, e.g., “car” and “gas”.

  Sometimes, related concepts are accepted as being similar, though.

Text Mining using Clustering

Clustering in text mining

- Input. Usually plain texts or text spans.
- Output. A set of clusters, and a model that maps from texts to clusters.



(similarity computation and cluster merging are mostly done iteratively)

Why clustering in text mining?

- Particularly targets situations where the set of classes is unknown.
- The main goal is often to find out what classes exist.

  The inference of class *labels* is done manually in many cases, though (see above).

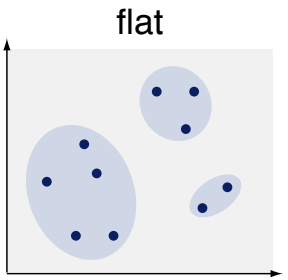
Selected applications in text mining

- Topic detection. What are the topics covered by a corpus of texts?
- Text retrieval. Detection of texts with similar properties.

  For example, in terms of author, structure, genre, or similar.

Text Mining using Clustering

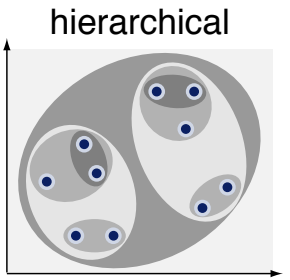
Flat vs. Hierarchical Clustering



Flat and hierarchical clustering

- Flat. Group a set of instances into a set of clusters.

  $\{1, 2, 3, 4\} \quad \rightarrow \quad \{1, 3, 4\}, \{2\}$



- Hierarchical. Create a binary tree over all instances where each node represents a cluster of a certain size.

  $\{1, 2, 3, 4\} \quad \rightarrow \quad \{ \ \{ \{\{1\}, \{3\}\}, \{4\} \}, \{6\} \ \}$

What type to use in text mining?

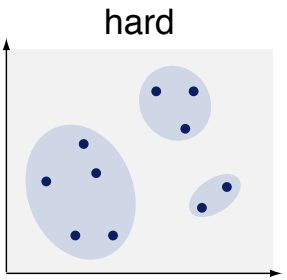
- In many settings, the final goal is to obtain a flat clustering.
- Flat clusterings can also be obtained through cuts in a hierarchy tree.
- The choice between the two types is rather an implementation decision, related to the effectiveness and efficiency of clustering.
- This is different if the hierarchical information is really required.

  For instance, when a taxonomy of related concepts shall be created.

Text Mining using Clustering

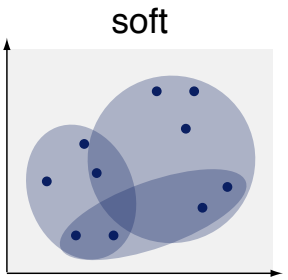
Hard vs. Soft Clustering

Hard and soft clustering



- Hard. The clustering creates a partition, such that each instance belongs to a single cluster.

  {1, 2, 3, 4} $\rightarrow$ C_1 = {1, 3, 4}, C_2 = {2}



- Soft. The clustering creates overlapping clusters, such that each instance o_i belongs to each cluster C_j with some weight $\theta_{i,j} \in [0, 1]$, $\sum_j \theta_{i,j} = 1$.

  {1, 2, 3, 4} $\rightarrow$ C_1 = (1, 0.6, 0.8, 0), C_2 = (0, 0.4, 0.2, 1)

What type to use in text mining?

- Hard clustering is used to identify a set of classes.
- Soft clustering can be understood as defining weighted concepts based on the classes, which is preferred where overlap is assumed.

  A variant of soft clustering is *topic modeling*, which finds overlapping topics.

Similarity Measures

Similarity Measures

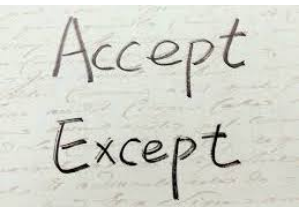
What is a similarity measure?

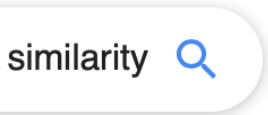
- A similarity measure is a real-valued function that quantifies how similar two instances of the same concept are.
- Usually, possible values range between 0 (no similarity) and 1 (identity).
- In text mining, instances are (the representations of) input text spans.

Various use cases in text mining

- Clustering
- Spelling correction
- Retrieval of relevant web pages
- Detection of related documents
- Paraphrase recognition
- (Near-) Duplicate or plagiarism detection
- Identification of counterarguments

... and many more





Similarity Measures

Text Similarity

Similarity in text mining

- Similarity between the *form* of two texts or text spans.
- Similarity between the *meaning* of two texts or text spans.

Similar form, different meaning: "This is shit." vs. "This is *the* shit."

Other way round: "Obama visited the capital of France." vs. "Barack Obama was in Paris."

- Ultimately, similarity measures aim to capture the latter.
- But the former is often used as a proxy.

Text similarity measures

- Vector-based measures. Mainly, for similarities between feature vectors.
- Edit distance. For spelling similarities.
- Thesaurus methods. For synonymy-related similarities.
- Distributional similarity. For similarities in the contextual usage.

  Clustering is mostly based on the first, but the others may still be used internally.

Vector-based Similarity Measures

Vector-based similarity measures

- Given a collection of input texts or text spans, the goal is to compare any two instances o_1, o_2 from them.
- Comparison is done on feature-based representations, i.e., o_1 and o_2 are mapped to feature vectors $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, respectively.

Feature-based representation (recap)

- A feature vector is an ordered set of values of the form $\mathbf{x} = (x_1, \ldots, x_m)$, where each feature x_i denotes a measurable property of an input, $m \geq 1$.

  We consider only real-valued features here.

- Each instance o_j is mapped to a vector $\mathbf{x}^{(j)} = (x_1^{(j)}, \ldots, x_m^{(j)})$ where $x_i^{(j)}$ denotes the value of feature x_i.

  We consider only values normalized to the range $[0, 1]$ here.

Similarity measures and clustering

- Clustering mostly relies on vector-based similarity measures.

Vector-based Similarity Measures

Concept

Measuring similarity between vectors

- Compare two vectors of the same representation with each other.

  $(1.0, 0.0, 0.3)$ vs. $(0.0, 0.0, 0.7)$ for $\mathbf{x} = (\text{red, green, blue})$

- The difference of each vector dimension is computed individually.

  1.0 vs. 0.0 0.0 vs. 0.0 0.3 vs. 0.7

- The similarity results from an aggregation of all differences.

  For example: $\frac{1.0+0.0+0.4}{3} \approx 0.467$

Concrete similarity measures

- Numerous vector-based measures are found in the literature. (Cha, 2007)
- We focus on four of the most common measures here: *Cosine similarity*, *Jaccard similarity*, *Euclidean distance*, and *Manhattan distance*.

  As mentioned above, distance can be seen as the inverse of similarity.

Vector-based Similarity Measures

Distance Functions

Properties of a distance function (aka metric)

- Non-negativity. $d(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) \geq 0$
- Identity. $d(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}) = 0$
- Symmetry. $d(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = d(\mathbf{x}^{(2)}, \mathbf{x}^{(1)})$
- Subadditivity. $d(\mathbf{x}^{(1)}, \mathbf{x}^{(3)}) \leq d(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) + d(\mathbf{x}^{(2)}, \mathbf{x}^{(3)})$

  Clustering actually does not necessarily require subadditivity.

Distance computation in clustering

- Internally, clustering algorithms compute distances between instances.

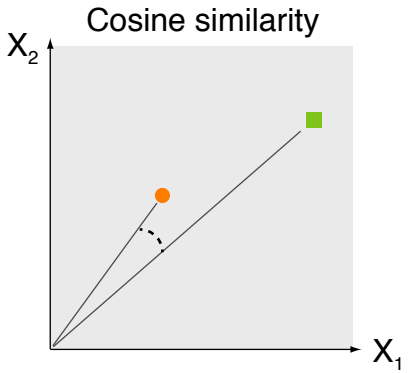
Instance	x_1	x_2	...	x_m
$\mathbf{x}^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$	...	$x_m^{(1)}$
$\mathbf{x}^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$	...	$x_m^{(2)}$
$\vdots$				
$\mathbf{x}^{(n)}$	$x_1^{(n)}$	$x_2^{(n)}$	...	$x_m^{(n)}$

Instance	$\mathbf{x}^{(1)}$	$\mathbf{x}^{(2)}$	...	$\mathbf{x}^{(n)}$
$\mathbf{x}^{(1)}$	0	$d(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$	...	$d(\mathbf{x}^{(1)}, \mathbf{x}^{(n)})$
$\mathbf{x}^{(2)}$	-	0	...	$d(\mathbf{x}^{(2)}, \mathbf{x}^{(n)})$
$\vdots$				
$\mathbf{x}^{(n)}$	-	-	...	0

Vector-based Similarity Measures

Cosine Similarity

Cosine similarity (aka cosine score)



- Cosine similarity captures the cosine of the angle between two feature vectors.
- The smaller the angle, the more similar the vectors.

  This works because cosine is maximal for 0°.

$$sim_{Cosine}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \frac{\mathbf{x}^{(1)} \cdot \mathbf{x}^{(2)}}{||\mathbf{x}^{(1)}|| \cdot ||\mathbf{x}^{(2)}||} = \frac{\sum_{i=1}^{m} x_i^{(1)} \cdot x_i^{(2)}}{\sqrt{\sum_{i=1}^{m} x_i^{(1)^2}} \cdot \sqrt{\sum_{i=1}^{m} x_i^{(2)^2}}}$$

Notice

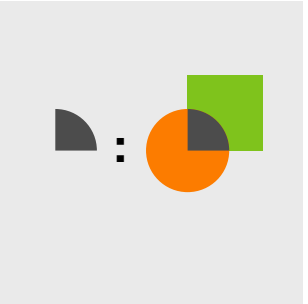
- The cosine similarity abstracts from the length of the vectors.
- Angle computation works for any number of dimensions.
- Cosine similarity is the most common similarity measure.

Vector-based Similarity Measures

Jaccard Similarity

Jaccard similarity coefficient (aka Jaccard index)

Jaccard similarity



- The Jaccard coefficient captures how large the intersection of two sets is compared to their union.
- With respect to vector representations, this makes at least sense for boolean features.

  For others, if there is a reasonable way of thresholding.

$$sim_{Jaccard}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \frac{|\mathbf{x}^{(1)} \cap \mathbf{x}^{(2)}|}{|\mathbf{x}^{(1)} \cup \mathbf{x}^{(2)}|} = \frac{|\mathbf{x}^{(1)} \cap \mathbf{x}^{(2)}|}{|\mathbf{x}^{(1)}| + |\mathbf{x}^{(2)}| - |\mathbf{x}^{(1)} \cap \mathbf{x}^{(2)}|}$$

$$= \frac{\sum_{x_i^{(1)}=x_i^{(2)}} 1}{m + m - \sum_{x_i^{(1)}=x_i^{(2)}} 1}$$

Notice

- The Jaccard similarity does *not* consider the size of the difference between feature values.

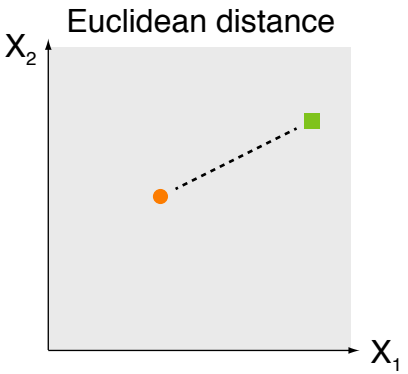
Vector-based Similarity Measures

Euclidean Similarity

Euclidean distance

- The Euclidean distance captures the absolute straight-line distance between two feature vectors.

$$d_{Euclidean}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \sqrt{\sum_{i=1}^{m} |x_i^{(1)} - x_i^{(2)}|^2}$$



Euclidean similarity

- If all feature values are normalized to $[0, 1]$, the Euclidean similarity is:

$$sim_{Euclidean}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = 1 - \frac{d_{Euclidean}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})}{\sqrt{m}}$$

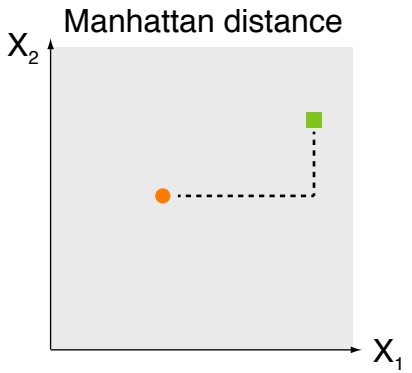
Notice

- Euclidean spaces generalize to any number of dimensions $m \geq 1$.
- Here, this means to any number of features.

Vector-based Similarity Measures

Manhattan Similarity

Manhattan distance (aka city block distance)



- The Manhattan distance is the sum of all absolute differences between two feature vectors.

$$d_{Manhattan}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \sum_{i=1}^{m} |x_i^{(1)} - x_i^{(2)}|$$

Manhattan similarity

- If all feature values are normalized to $[0, 1]$, the Manhattan similarity is:

$$sim_{Manhattan}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = 1 - \frac{d_{Manhattan}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})}{m}$$

Notice

- Manhattan distance and Euclidean distance are both special cases of the *Minkowski distance*.

$$d_{Minkowski}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \sqrt[p]{\sum_{i=1}^{m} |\mathbf{x}_i^{(1)} - \mathbf{x}_i^{(2)}|^p} \quad \text{for any } p \in \mathbb{N}^+$$

Vector-based Similarity Measures

When to Use What Measure?

Comparison of the measures

- Cosine similarity. Puts the focus on those properties that occur. Targets situations where a vector's direction matters rather than its length.
  A prominent use case is matching queries with documents in web search.
- Jaccard similarity. Seems less precise than cosine similarity, but this also makes it more robust (it "overfits" less).
- Euclidean and Manhattan. Target situations where a value of 0 does not mean the absence of a property.
- Euclidean or Manhattan. Depends on whether sensitivity to outliers in certain dimensions is preferred or not.

Similarity as an optimization hyperparameter

- In general, it is not always clear what measure will prove best.
- One way to deal with this is to simply evaluate different measures.
- In some applications, all measures can be used simultaneously.

Similarity between Strings

Limitation of vector-based measures in text mining

- Similarity is defined based on corresponding feature values, $x_j^{(1)}, x_j^{(2)}$.
- Most features in text mining are derived directly from text spans.
- Similarity between different forms with similar meaning is missed...

  "traveling" vs. "travelling" "woodchuck" vs. "groundhog" "Trump" vs. "The President"

- ... unless such differences are accounted for.

Similar strings

- May contain differences in writing, due to spelling errors, language variations, or additional words.
- May contain different words that refer to similar concepts.
- May contain different concepts that are related in a way that should be seen as similar in a given application.

  ... and similar

Similarity between Strings

Edit Distance

What is (minimum) edit distance?

- The minimum number (or cost) of editing operations needed to transform one string to another.
- Editing operations. Insertion, deletion, substitution.
- Weighted edit distance. Different edits vary in costs.

```
I N T E * N T I O N
| | | | | | | | | |
d s s   i s
| | | | | | | | | |
* E X E C U T I O N
```

How to compute edit distance?

- Sequence alignment using dynamic programming.
- Equals shortest path search in a weighted graph.

	E	X	E
I	s(I,E)	i(*,X)	
N	d(N,*)	s(N,X)	
T			

Selected applications

- Spelling correction, e.g., in search engines.

  “wreckonize speach” → Did you mean “recognize speech”?

- Gene comparison in computational biology (kind of language problem).

Similarity between Strings

Thesaurus Methods

What are synonyms?

- Words (or terms) that have the same meaning in some or all contexts.

  "couch" vs. "sofa" "big" vs. "large" "water" vs. "H_20" "vomit" vs. "throw up"

- There are hardly any perfectly synonymous terms.

  Even seemingly identical terms usually differ in terms of politeness, slang, genre, etc.

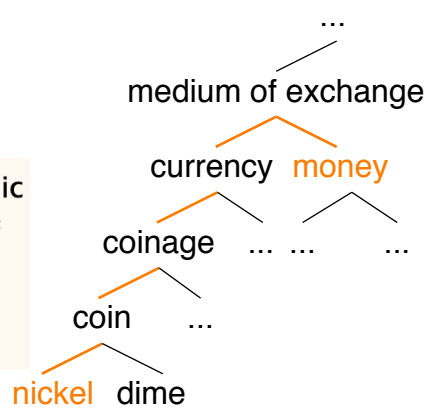
- Synonymy is a relation between senses rather than words.

  "big" vs. "large" → "Max became kind of a <insert> brother to Linda."

How to identify related senses?

- Compute distance in thesauri, such as *WordNet*.
  wordnetweb.princeton.edu/perl/webn

S: (n) **nickel**, Ni, atomic number 28 (a hard malleable ductile silvery metallic element that is resistant to corrosion; used in alloys; occurs in pentlandite and smaltite and garnierite and millerite)
S: (n) **nickel** (a United States coin worth one twentieth of a dollar)
- ***direct hypernym*** / *inherited hypernym* / *sister term*
  - S: (n) coin (a flat metal piece (usually a disc) used as money)



- Several libraries for such measures freely available.

Similarity between Strings

Distributional Similarity

Limitation of thesaurus methods

- Many words are missing as well as virtually all phrases, and also some sense connections.
- Verbs and adjectives are not as hierarchically structured as nouns.
- Thesauri are not available for all languages.

"You shall know a word by the company it keeps!" (Firth, 1957)

Idea of distributional similarity

- If A and B have almost identical environments, they are synonyms.
- Two words are similar if they have similar word contexts, i.e., if they have similar words around them.

"Everybody likes tesgüino." "A bottle of tesgüino is on the table."
"Tesgüino makes you drunk." "We make tesgüino out of corn."

$\rightarrow$ An alcoholic beverage like beer.

Similarity between Strings

Pointwise Mutual Information

Word-context matrix

- Cooccurrences of words in a corpus within a window of some number of words (say, 20).

	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

Pointwise mutual information (PMI) for words

- Do two words w_i and w_j cooccur more than if they were independent?

$$PMI(w_i, w_j) = log_2 \frac{P(w_i, w_j)}{P(w_i) \cdot P(w_j)}$$

- Positive PMI (PPMI). Replace all values < 0 with 0.
- Extensions. Avoid bias towards infrequent words, consider syntax, ...

PPMI approximated based on a word-context matrix

P("information", "data") $= \frac{6}{19} = 0.32$ P("information") $= \frac{11}{19} = 0.58$ P("data") $= \frac{7}{19} = 0.37$

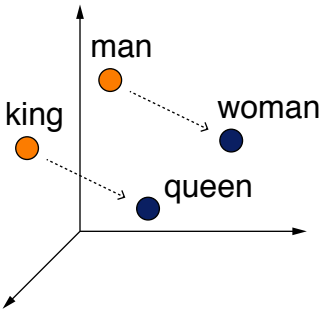
$\rightarrow$ PMI("information", "data") $= log_2 \frac{0.32}{0.37 \cdot 0.58} = 0.58$

Similarity between Strings

Word Embeddings

Extension of the distributional idea

- Representation of a word by the context it occurs in.
- To do so, words are mapped into an *embedding space* where contextually related words are similar.



Word embedding (aka word vector)

- A high-dimensional real-valued vector that represents the *distributional semantics* of a particular word in the embedding space.

$$\text{“king”} \quad \rightarrow \quad v_{king} = (0.13, 0.02, 0.1, 0.4, \ldots, 0.22)$$

- The more dimensions, the more variance is kept (typical: 100–500).

Some properties of embedding spaces

- Similar context results in similar embeddings. projector.tensorflow.org
- Analogies are arithmetically represented. turbomaze.github.io/word2vecjson

$$v_{king} - v_{man} + v_{woman} \approx v_{queen} \qquad v_{france} - v_{paris} + v_{berlin} \approx v_{germany}$$

Similarity between Strings

Embedding Models

Word embedding models

- A word embedding model maps each known word to its embedding.
- The mapping is created unsupervised based on a (usually huge) corpus, capturing the likelihood of words occurring in sequence.

  The technical details are beyond the scope of this course.

Several software libraries and pre-trained models exist

- Libraries. Glove, word2vec, Fasttext, Flair, Bert, ...
- Models. GoogleNews-vectors, ConceptNet Numberbatch, ...

From word embeddings to text embeddings

- Simple. Average the embeddings of each word in a text.
- More sophisticated. Learn embeddings for sentences or similar.
- In general, the longer the text, the harder it is to capture its semantics in an embedding.

Similarity between Strings

From Strings back to Texts

Encoding similarities in feature vectors

- String similarities can be used in diverse ways within features.

  Frequency of ~~"money"~~ the sense "the most common medium of exchange"

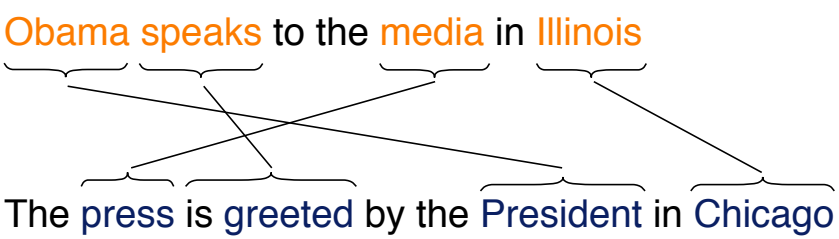
  Frequency of all writings of "traveling"

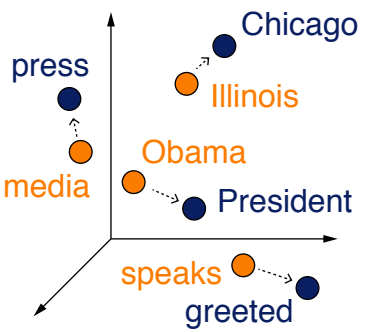
- Where reasonable, embeddings can simply be used as feature vectors.

  "nickel" $\rightarrow$ (0.14, 0.03, 0.44, . . . , 0.22) "money" $\rightarrow$ (0.18, 0.06, 0.49, . . . , 0.01)

Word Mover's Distance (Kusner et al., 2015)

- The distance of the optimal alignment of two texts.





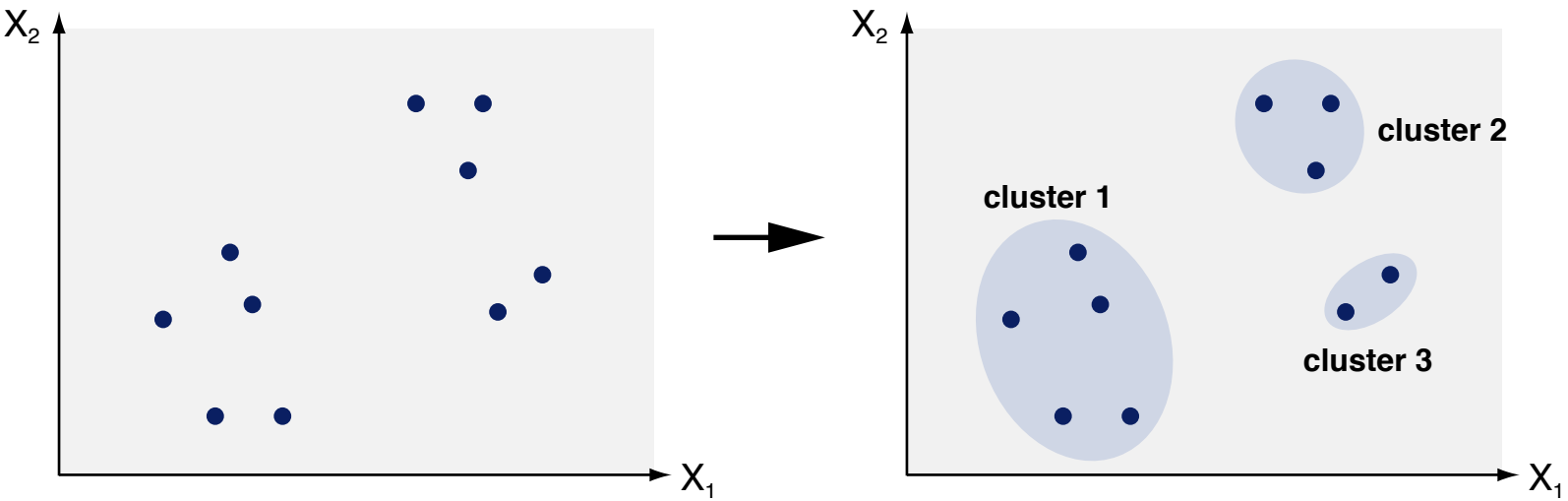
- Represents texts by sequences of word embeddings.

Hard and Soft Flat Clustering

Hard Flat Clustering

What is hard flat clustering?

- Hard flat clustering partitions a set of instances into disjunct clusters.
- Input. A set of instances $X = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$ without class labels.
- Output. A set of clusters $C = \{c_1, \ldots, c_k\}$ and a mapping $X \to C$.



Number of clusters k

- Some clustering algorithms have k as a hyperparameter.
- Others determine k automatically.

Hard Flat Clustering

Two Main Types of Algorithms

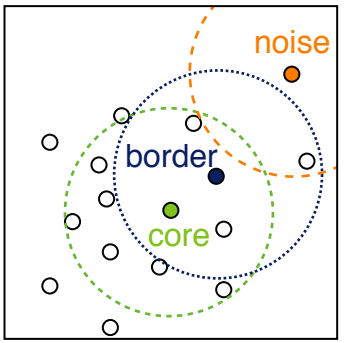
Iterative algorithms

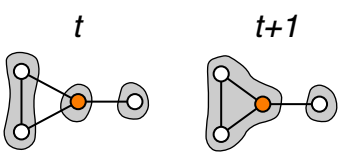
- Iterative clustering and re-assignment of instances to clusters.
- Exemplar-based (e.g., k-means). Instances are considered in isolation when adding them to clusters.

  We focus on this type here.
- Exchange-based (e.g., Kerninghan-Lin). Instances are exchanged between pairs of clusters.

Density-based algorithms

- Clustering of instances into regions of similar density.
- Point density (e.g., DBSCAN). Distinction of instances in the *core* of a region, at the *border*, or *noise*.
- Attraction-based (e.g., MajorClust). Instances in a cluster combine "forces" to "attract" further instances.

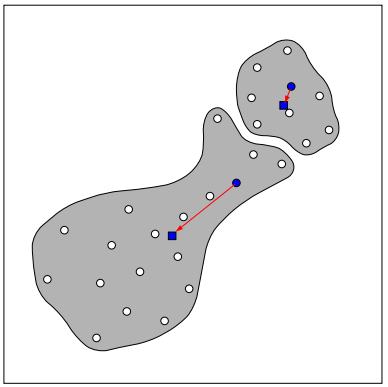




Flat Clustering with k-means

What is k-means?

- A simple hard clustering approach that creates $k \geq 1$ clusters.
- Iterative and exemplar-based.
- k is a hyperparameter chosen based on domain knowledge or based on evaluation measures (see below).



k-means in a nutshell

- Iteratively compute centroids of candidate clusters.
- Re-cluster based on similarity to centroids.
- Repeat until convergence.

Variations

- Some versions of k-means includes a maximum number of iterations.
- Medoid. A generalization of the centroid, which can be computed in different ways. Respective algorithms are not called k-means anymore.

Flat Clustering with k-means

Pseudocode

Signature

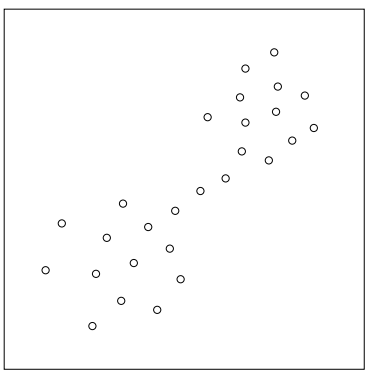
- Input. A set of instances X, a number of clusters k.
- Output. A clustering C, i.e., a set of clusters.

```
kMeansClustering(Set<Instance> X, int k)
 1.   Set<Instance> [] clusters ← ∅
 2.   Instance [] centroids ← chooseRandomInstances(X, k)
 3.   repeat
 4.      Instance [] prevCentroids ← centroids
 5.      for int i ← 1 to k do clusters[i] ← ∅
 6.      for each x ∈ X do // create clusters
 7.         int z ← 1
 8.         for int j ← 2 to k do // find nearest centroid
 9.            if sim(x,centroids[j]) > sim(x,centroids[z]) then z←j
10.         clusters[z] ← clusters[z] ∪ {x}
11.      for int i ← 1 to k do // update centroids
12.         centroids[i] ← computeMeans(clusters[i])
13.   until prevCentroids = centroids // convergence
14.   return clusters
```

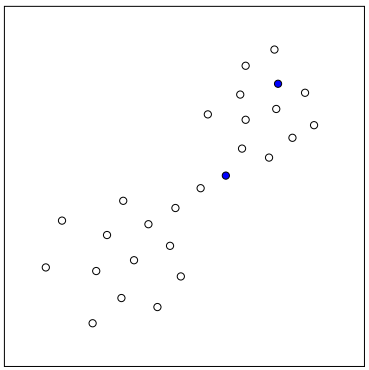
Flat Clustering with k-means

Example for $k = 2$ (recap)

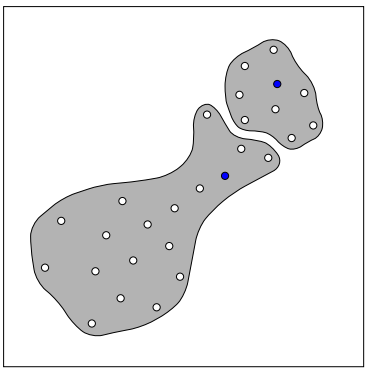
Input instances



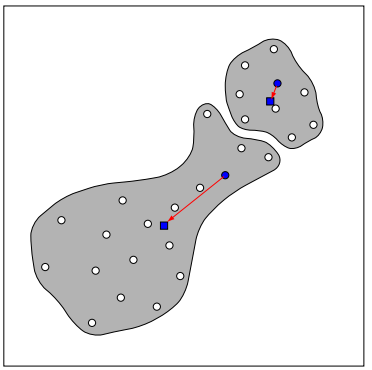
k random centroids



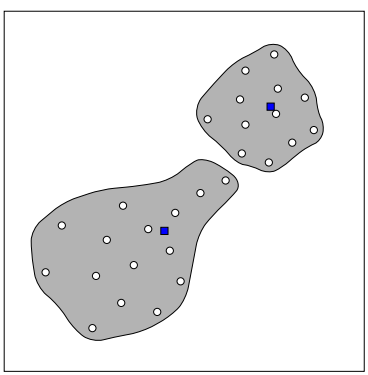
Cluster by similarity



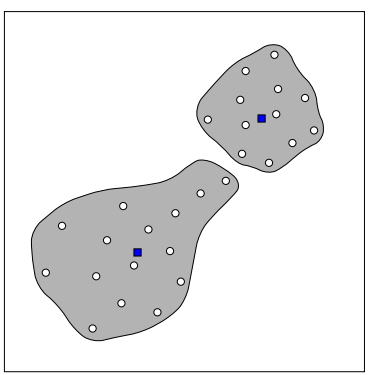
Get cluster centroids



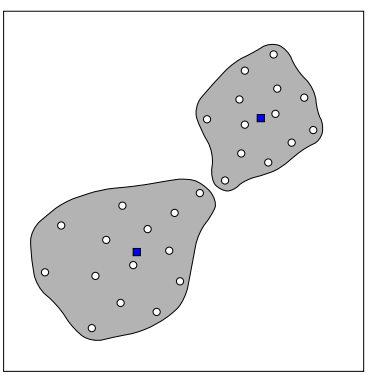
Cluster by similarity



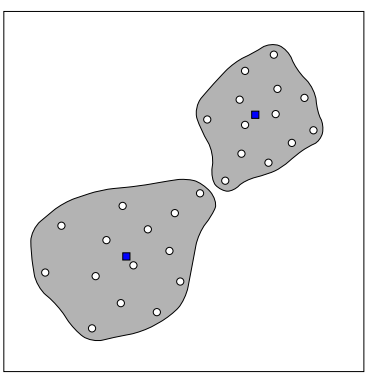
Get cluster centroids



Cluster by similarity



Convergence

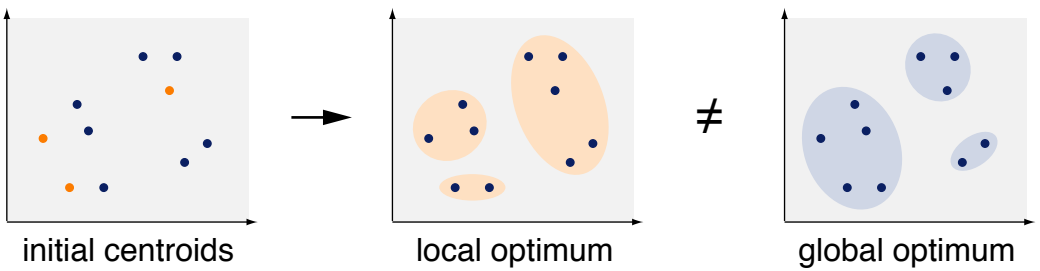


Flat Clustering with k-means

Random Initialization

Problem

- Let's assume we are given an intrinsic or extrinsic cost function J, i.e., there is an optimal clustering.
- k-means converges when it has found a local mininum.
- Due to the random choice of centroids, it may not find the global one.



Approach

- To account for this, k-means is often repeated several (e.g., 100) times.
- The best found local optimum is chosen then.
- An alternative is to pick good initial centroids using expert knowledge.

Flat Clustering with k-means

Number of Clusters

Choice of the number of clusters

- Unless decided by expert knowledge, k needs to be evaluated against some intrinsic or extrinsic cost function.
- However, most cost functions grow (or fall) with the number of clusters.

Example cost functions

- Intrinsic. Squared distances of instances to centroid. $\rightarrow$ 0.0 for $k = |X|$
- Intrinsic. Maximum cluster size. $\rightarrow$ highest for $k = 1$
- Intrinsic. Maximum cluster distance. $\rightarrow$ highest for $k = |X|$
- Extrinsic. Purity of clusters. $\rightarrow$ 1.0 for $k = |X|$
- Extrinsic. Macro/Micro F_1-score. $\rightarrow$ 1.0 for $k = |X|$

Approaches

- Elbow criterion. Find the k that maximizes cost reduction.
- Silhouette analysis. Measure sizes of and distances between clusters.

  Both approaches have a visual intuition, but work mathematically.

Flat Clustering with k-means

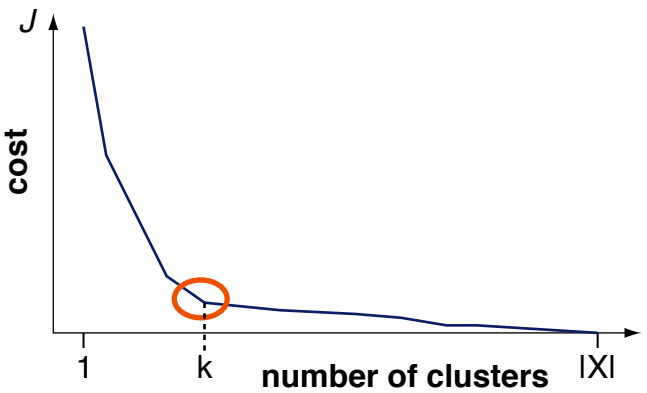
Elbow Criterion

What is the elbow criterion?

- A method to find the best value of a hyperparameter, e.g., k in k-means.
  Other algorithms also have hyperparameters, e.g., DBSCAN has a neighborhood size.
- Relies on an intrinsic or extrinsic cost function J.

Input

- A *set* of clusterings $\mathbf{C} = \{C_1, \ldots, C_p\}$ for hyperparameter values $k_1, \ldots, k_p$.
- A cost $J(C_i)$ for each clustering C_i.



Approach

- Find k that maximizes cost reduction with regard to its successor $k + 1$.

Flat Clustering with k-means

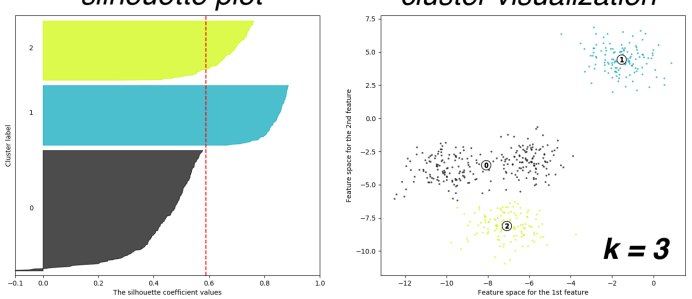
Silhouette Analysis

What is silhouette analysis?

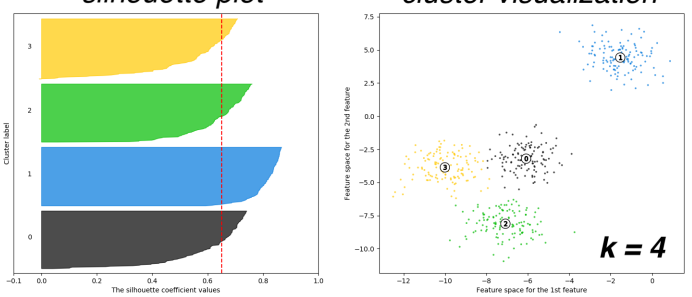
- A method to find the best value of a hyperparameter, e.g., k in k-means.
- Computes an average silhouette score in $[-1, 1]$ that captures how close each instance is to instances in other clusters.

  ~ 1: Far away *~ 0:* At the boundary to other clusters *< 0:* Possibly in wrong cluster

silhouette plot *cluster visualization*



silhouette plot *cluster visualization*



Interpretation based on the average silhouette score

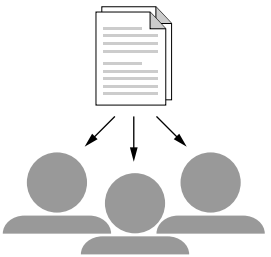
- $k = 3$ is bad. Some clusters have scores below the average, and the size of clusters (thickness of the plots) varies strongly.
- $k = 4$ is better. All scores above average, thickness balanced.
- The k with the highest average score (vertical line) is best in general.

Authorship Attribution

What is authorship attribution?

- The text analysis that reveals the authors of texts.
- Tackled in text mining as a downstream task.

  Related tasks: Authorship verification, plagiarism detection, ...



Settings

- Supervised. Given a set of n training texts with p known authors, learn a mapping from texts to authors.
- Unsupervised. Given a set of n training texts (usually assumed to be single-authored), group them by their author.

Observations

- Unlike in most tasks, computers tend to be better than humans here.
- Features that capture style are mostly in the focus.
- Some successful features capture subconscious language use.

  "The happening of some of the cases given: the clearance of approval by the ..."

Authorship Attribution

CLEF 2016 Shared Task on Author Clustering

Shared task

- Participants develop competing approaches for the same task and data.

Task definition "Author Clustering"

- Given a corpus with up to 100 texts, identify the number k of authors and assign each text to the cluster representing its author.
- Training sets are given; results are averaged over unseen test sets.

18 training sets and test sets

- Six sources. Opinion articles and reviews in Dutch, English, and Greek.
- Three datasets per source. Differ in terms of the number of authors.
- Most texts range between 400 and 800 words.

Eight participating teams

- Two participants used k-means, including an estimation of the best k.
- The others identified authors based on different criteria first.

Authorship Attribution

k-means Approaches in the Shared Task

Mansoorizadeh et al.

- Features. Word and POS unigrams and bigrams, sentence lengths, punctuation n-grams with $n \geq 2$.

  Texts lower-cased, no features discarded, feature values normalized.
- Similarity. Cosine score.
- Choosing k. Creation of a similarity graph with similarity threshold 0.5. The number of subgraphs defines the k used for k-means.

Sari and Stevenson

- Features. TF-IDF on the 5000 top character n-grams with $n \in \{3, \ldots, 8\}$, average word embeddings.

  Embeddings: GoogleNews-vector (English), self-trained (Dutch), none (Greek).
- Similarity. Cosine score.
- Choosing k. Silhouette analysis based on k-means. The k with the highest Silhouette score ist taken.

Authorship Attribution

Shared Task Results Averaged over All Test Sets

Effectiveness and efficiency results

Approach	B^3 precision	B^3 recall	B^3 F_1-score	Run-time
Kocher	**0.982**	0.722	**0.822**	00:01:51
Bagnall	0.977	0.726	**0.822**	63:03:59
Sari and Stevenson	0.893	0.733	0.795	00:07:48
Zmiycharov et al.	0.852	0.716	0.768	01:22:56
Gobeill	0.737	0.767	0.706	00:00:39
Kuttichira	0.512	0.720	0.588	00:00:42
Mansoorizadeh et al.	0.280	0.822	0.401	**00:00:17**
Vartapetiance and Gillam	0.195	**0.935**	0.234	03:03:13

B^3 precision and recall of a text d

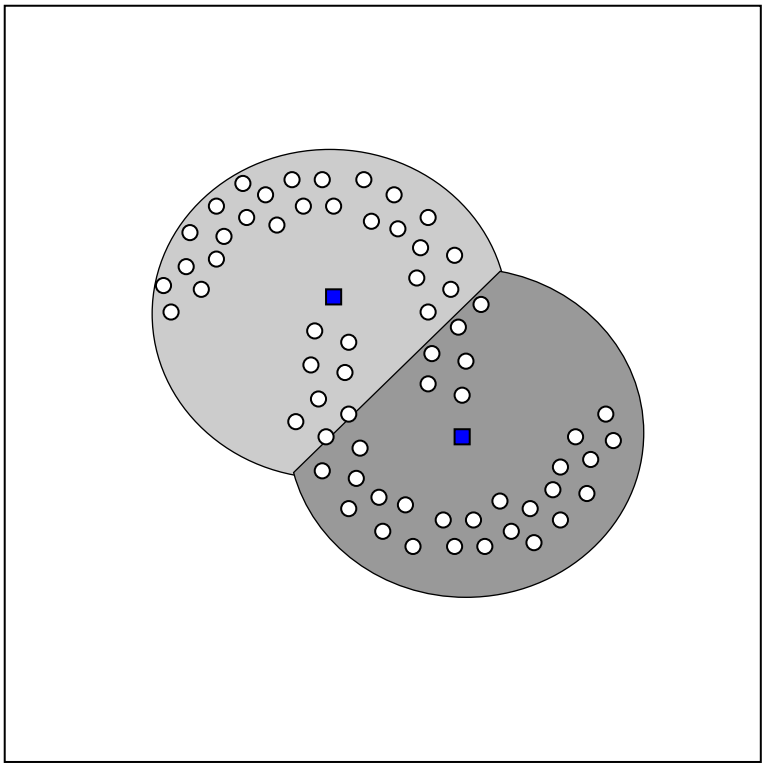
- B^3 precision. Proportion of texts in the cluster of d by the author of d.
- B^3 recall. Proportion of texts by the author of d found in the cluster of d.

The values are averaged over all texts. F_1-score as usual.

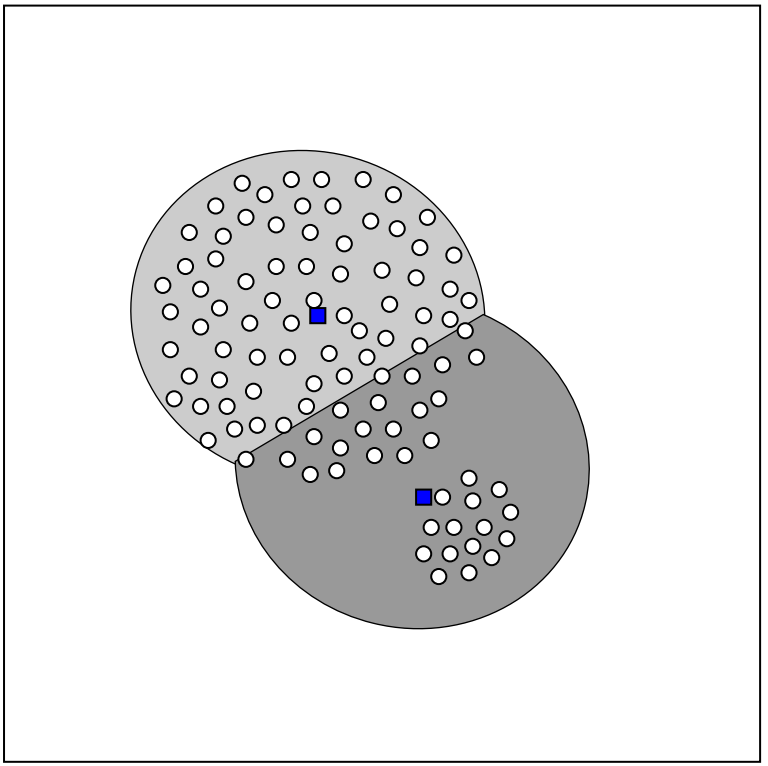
Hard Flat Clustering

Issues with Iterative, Exemplar-based Clustering Algorithms

Algorithms such as k-means fail to detect nested clusters.



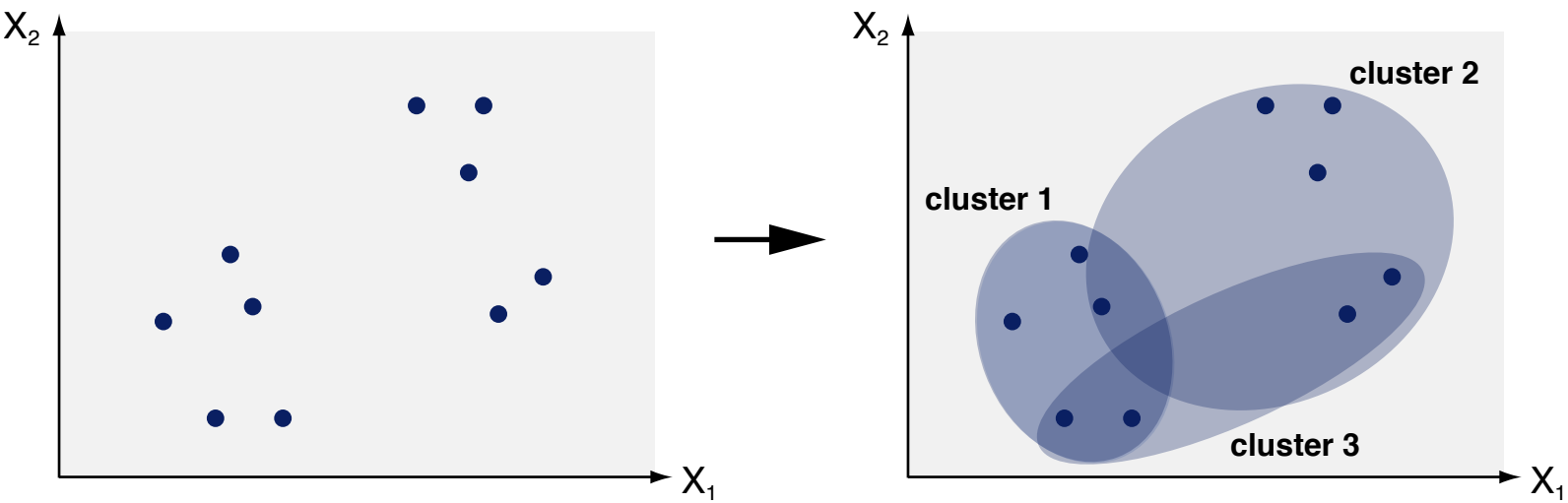
Similarly, they fail to detect clusters with large difference in size.



Soft Flat Clustering

What is soft flat clustering?

- Soft flat clustering maps instances to a set of overlapping clusters.
- Input. A set of instances $X = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$ without class labels.
- Output. A set of clusters $C = \{c_1, \ldots, c_k\}$ and a weighted mapping $X \to \{(c, \theta_c) \mid c \in C, \theta \in [0, 1]\}$, such that $\forall \mathbf{x}_i \in X : \sum_{c \in C} \theta_{i,c} = 1$.



Number of clusters k

- As for flat clustering, k may be a hyperparameter.

Soft Flat Clustering

Idea and Algorithms

Idea of soft clustering

- Given the following five sentences:

“Max likes to eat broccoli and bananas.”	→ 1.0 topic A
“Tim had a banana and spinach smoothie for breakfast.”	→ 1.0 topic A
“Chinchillas and kittens are cute.”	→ 1.0 topic B
“Linda adopted a kitten yesterday.”	→ 1.0 topic B
“The cute hamster munches on a piece of broccoli.”	→ 0.6 topic A, 0.4 topic B

- A clustering algorithm might identify two soft clusters:

  Topic A representing food Topic B representing cute animals

- Each sentence can then be assigned a weight for each cluster.

Selected algorithms used for soft clustering

- Fuzzy k-means clustering
- Gaussian mixture models
- Latent Dirichlet allocation

Topic Modeling

What is topic modeling?

- Topic modeling extracts topics from a text corpus based on patterns in the use of words.
- A topic is modeled as a list of words that cooccur in a statistically meaningful way.

BIRDS NEST TREE
BRANCH LEAVES

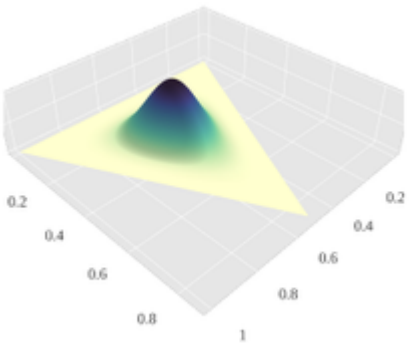
Why topic modeling?

- Finds low-dimensional representations of high-dimensional text.
- Attempts to inject semantic meaning into vocabulary.
- Enables concise summaries of texts and to capture their similarity.

Latent Dirichlet allocation (LDA)

- The most popular topic modeling technique.
- The terms *topic modeling* and *LDA* are often used synonymously.

  In principle, LDA can also be used for data other than text.



Topic Modeling

Latent Dirichlet Allocation (LDA)

What is LDA?

- A probabilistic technique to automatically discover topics in a corpus.
- Learns the relative importance of topics in texts and words in topics.
- Based on the bag-of-words idea.

General LDA process

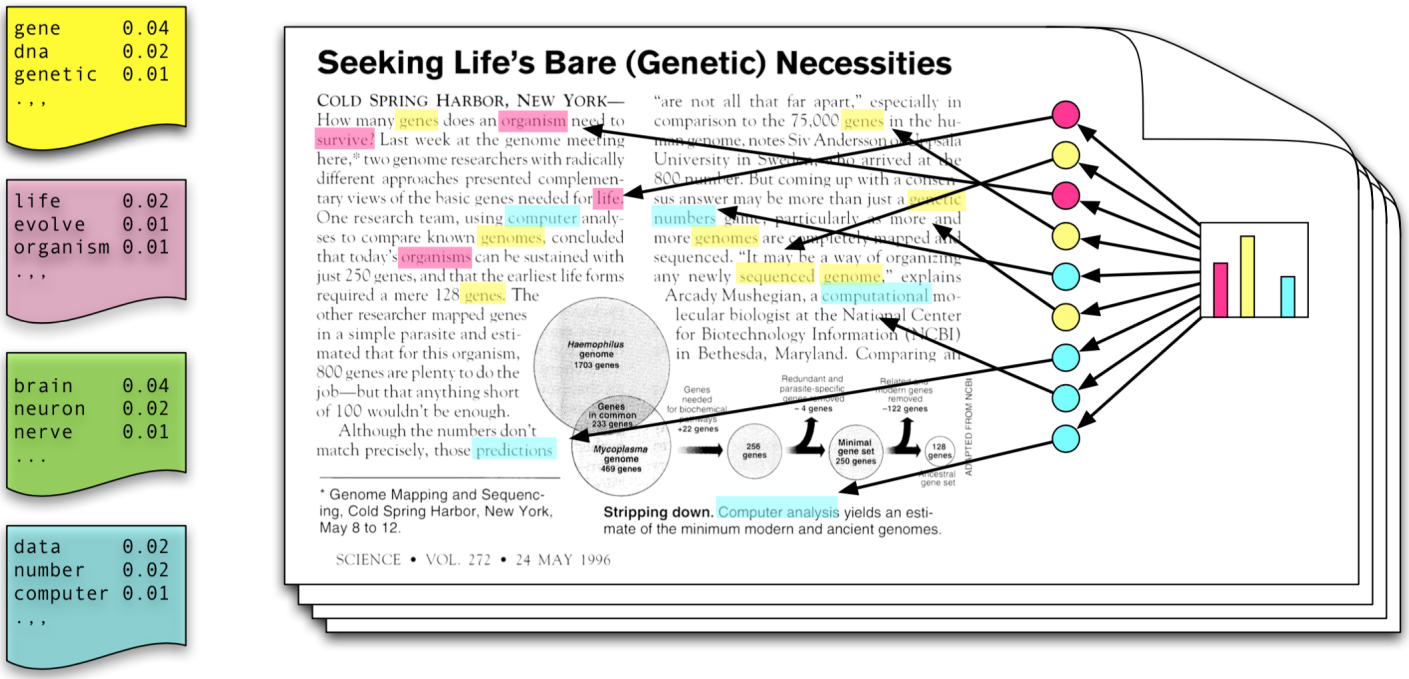
- Assumes a text to be composed of words from word lists called *topics*.
- Decomposes a text into the topics from which the words probably came.
- Repeats this process multiple times to obtain the most likely distribution of words over topics.

Notice

- Machine learning toolkits such as *scikit-learn* include LDA.
- Technically, the process is often implemented using *Gibbs sampling*.
  The mathematical details are beyond the scope of this course.

Topic Modeling

Assumptions behind LDA

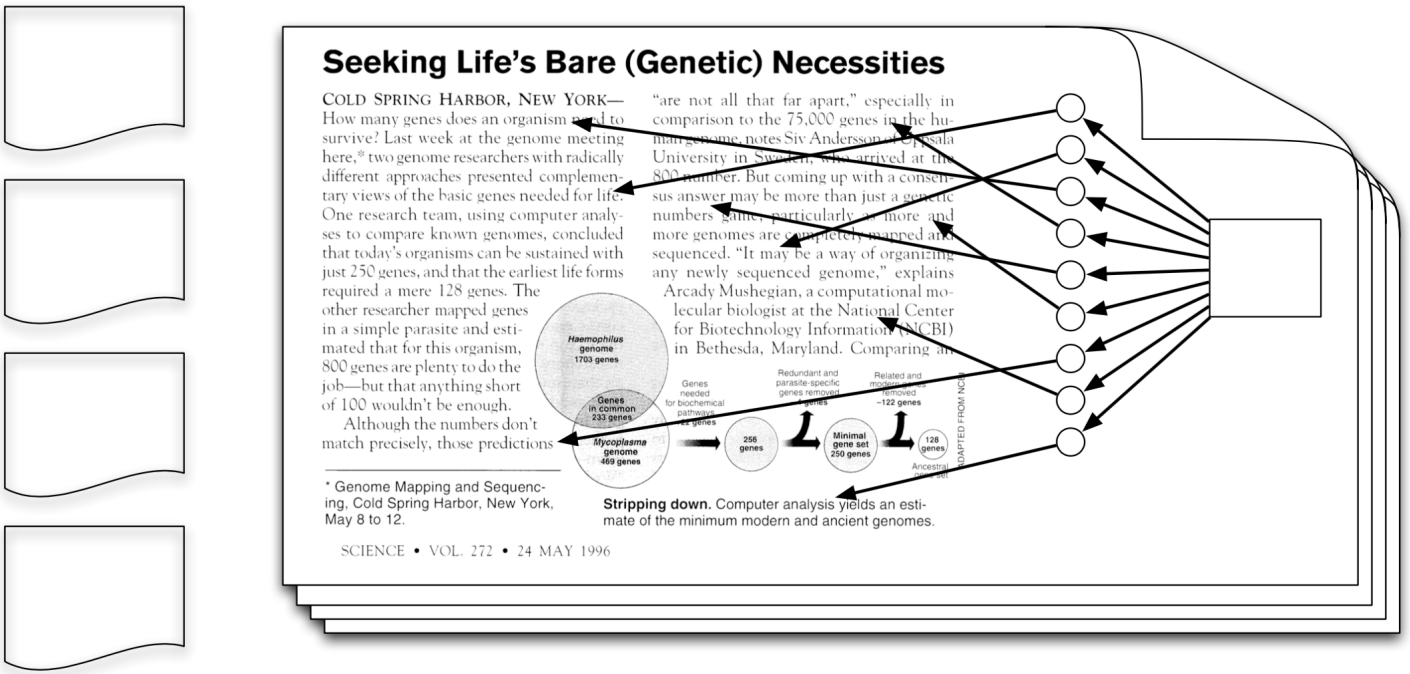


Assumptions

- Each topic is a distribution over words.
- Each text is a mixture of corpus-wide topics.
- Each word is drawn from one of those topics.

Topic Modeling

Setting of LDA



Setting

- In reality, we observe only texts and words, not topics.
- The aim of LDA is to infer the latent (say, hidden) topic structure.

Topic Modeling

LDA Pseudocode Sketch

Signature

- Input. A set of n texts, a number k of topics to be found, and a number m of words to represent each topic with.
- Output. A topic weighting of each text, a set of words for each topic.

Pseudocode sketch

```
1.  repeat
2.      Randomly assign each word w in each text d to one topic t
3.      for each text d, word w in d, topic t do
4.          Reassign w to topic t with probability p(t|d) · p(w|t)
            // p(t|d): fraction of words in d currently assigned to t
            // p(w|t): overall fraction of assignments to t from w
5.  until probabilities stable (or until some max iterations)
6.  Get topic weighting (θ_1,...,θ_k)_d of each text d
    // Fraction of words from each topic within d
7.  Get words (w_1,...,w_m)_t for each topic t
    // Words most often assigned to t over all topics
8.  return all (θ_1,...,θ_k)_d and (w_1,...,w_m)_t
```

Topic Modeling

Example Topic Models

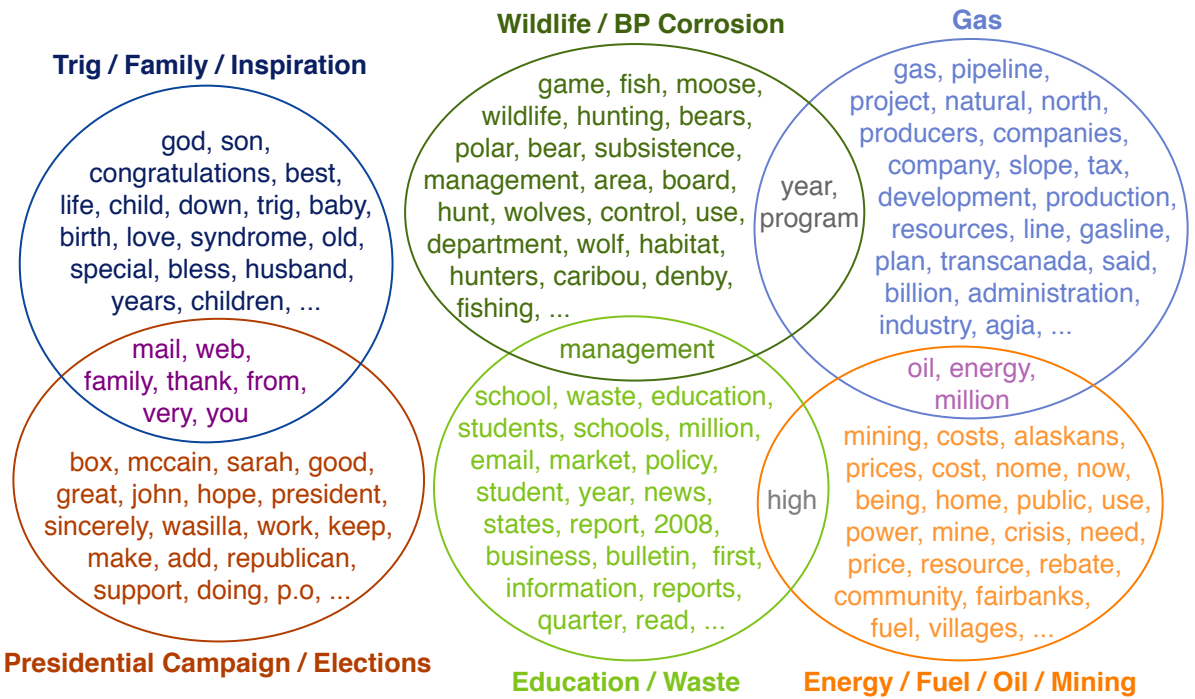
Topic modeling case study

Taken from http://blog.echen.me/2011/06/27/topic-modeling-the-sarah-palin-emails/

- Data. Several thousands of e-mails from Sarah Palin's inbox that were released in 2011.
- Goal. Find the main topics covered in the e-mails.

Found topics

(labeled manually)



Topic Modeling

Example Texts with Highlighted Topic Words

99% Trig / Family / Inspiration

> Hello Governor Palin, Our **family** wanted to congratulate **you** and your **family** on the **birth** of your **son**, **Trig**. Our fourth **child**, Daniel, was **born** with **Down Syndrome**, and we can't imagine our **family** without him. Recently, I met a mom with a 34-year-old **daughter** with DS and she said it best: "Don't **you** feel like you've been chosen to be a member of a **very special** club?" **God** bless your **family**, what a **beautiful** example of **love** you are to all who see you! the Paul & Tricia Pietig **family**, Des Moines, Iowa

90% Wildlife / BP Corrosion, 10% Presidential Campaign / Election

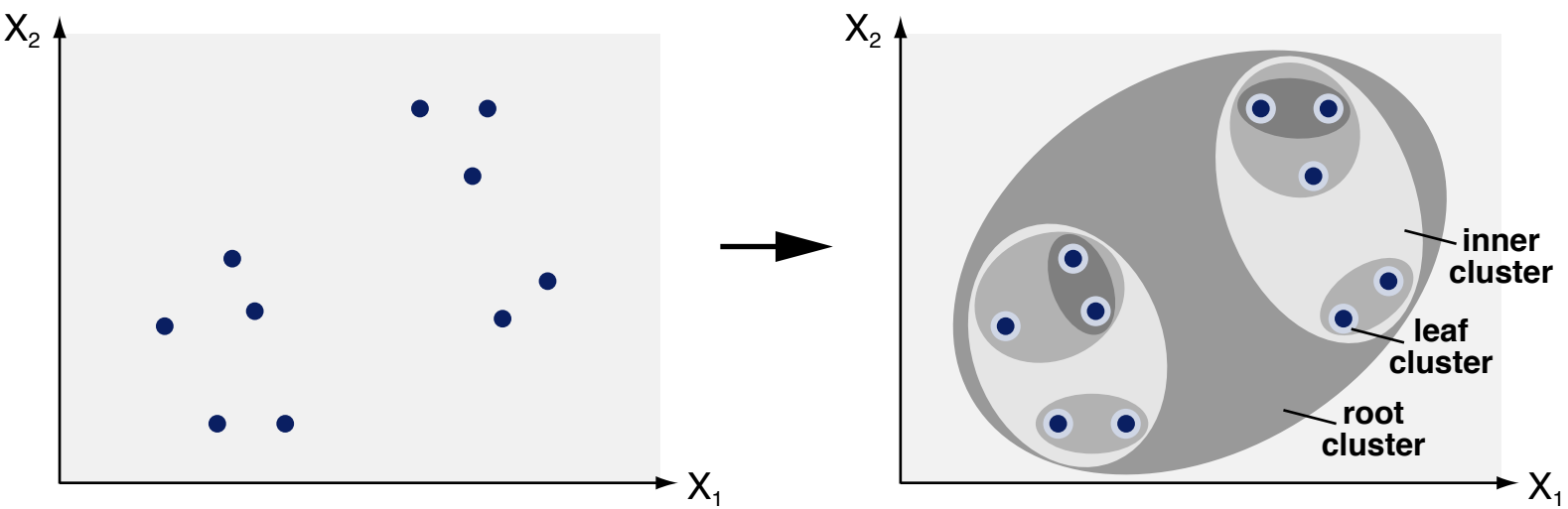
> We understand that you have been discussed as a possible choice for the **Vice Presidency**.
>
> As **people** who **support** the democratic process and care about protecting our **wildlife** for future generations, we want **you** to know that we don't believe **people** in our states would vote for **you** for any office if they knew your record on these issues.
>
> It is troubling that **you** are **now** working to deny more than 50,000 Alaskans a vote on **aerial** killing of **wolves** and **bears** with legislation now **being** considered in the Alaska legislature.

Hierarchical Clustering

Hierarchical Clustering

What is hierarchical clustering?

- Hierarchical clustering creates a binary tree over a set of instances, which represents the stepwise merging of the instances into clusters.
- Input. A set of instances $X = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$ without class labels.
- Output. A tree $\langle V, E \rangle$ where each $v \in V$ represents a cluster of some size, and each $(v_1, v_2) \in E$ indicates that v_2 has been merged into v_1.



Notice

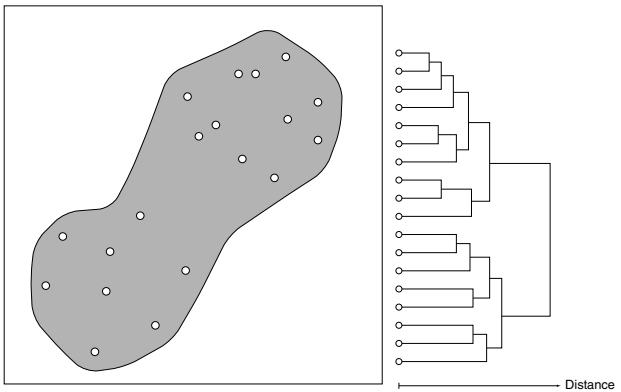
- A flat clustering can be derived via cuts in the hierarchy tree.

Hierarchical Clustering

Two Main Types of Algorithms

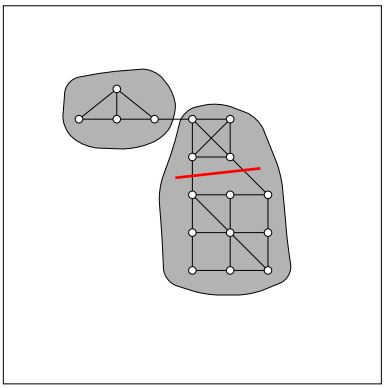
Agglomerative hierarchical clustering

- Incrementally create tree bottom-up, beginning with the single instances.
- Merge clusters based on the distances between the instances they contain.



Divise hierarchical clustering

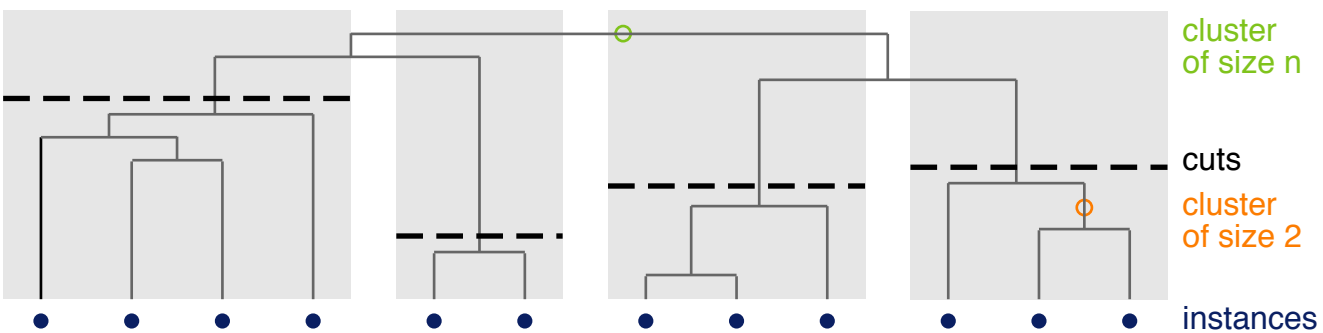
- Incrementally divides a set of instances into smaller clusters (top-down).
- The most widely used algorithm is MinCut; it models the set of instances as a weighted graph.
- MinCut repeatedly splits clusters by finding the minimum cut in a subgraph.



Agglomerative Hierarchical Clustering

Agglomerative hierarchical clustering in a nutshell

- Merge closest pair of clusters.
- Represent clusters and how they are merged as a dendrogram.
- Repeat until only one cluster remains.



Dendrogram

- A dendrogram is a diagram representing a tree.
- In agglomerative hierarchical clustering, dendograms are used to visualize what clusters are merged and when they are merged.
- Also, the cuts done to obtain a flat clustering may be illustrated in them.

Agglomerative Hierarchical Clustering

Pseudocode

Signature

- Input. A set of instances X.
- Output. A binary tree $\langle V, E \rangle$ containing all clusters.

```
agglomerativeHierarchicalClustering(Set<Instance> X)
1.  Set<Set<Instance>> clusters ← {{x^(i)} | x^(i) ∈ X} // curr. clusters
2.  Set<Set<Instance>> V ← clusters // tree nodes
3.  Set<Set<Instance>[]> E ← ∅ // tree edges
4.  while |clusters| > 1 do
5.      double [][] similarities ← updateSimilarities(clusters)
6.      Set<Instance> [] pair ← getClosest(clusters, similarities)
7.      Set<Instance> merged ← pair[0] ∪ pair[1]
8.      clusters ← (clusters \ {pair[0], pair[1]}) ∪ {merged}
9.      V ← V ∪ {merged}
10.     E ← E ∪ {(merged, pair[0]), (merged, pair[1])}
11. return ⟨V, E⟩
```

Agglomerative Hierarchical Clustering

Similarity Re-Computation after each Merging Step

$t = 0$

	C_1	C_2	...	C_n
C_1	0	$sim(C_1, C_2)$	...	$sim(C_1, C_n)$
C_2	-	0	...	$sim(C_2, C_n)$
$\vdots$				
C_n	-	-	...	0

$\equiv$

	$\mathbf{x}^{(1)}$	$\mathbf{x}^{(2)}$	...	$\mathbf{x}^{(n)}$
$\mathbf{x}^{(1)}$	0	$sim(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$	...	$sim(\mathbf{x}^{(1)}, \mathbf{x}^{(n)})$
$\mathbf{x}^{(2)}$	-	0	...	$sim(\mathbf{x}^{(2)}, \mathbf{x}^{(n)})$
$\vdots$				
$\mathbf{x}^{(n)}$	-	-	...	0

$\downarrow$

$t = i$

	C_{i_1}	C_{i_2}	...	$C_{i_{n-i}}$
C_{i_1}	0	$sim(C_1, C_{i_2})$	...	$sim(C_{i_1}, C_{i_{n-i}})$
C_{i_2}	-	0	...	$sim(C_{i_2}, C_{i_{n-i}})$
$\vdots$				
$C_{i_{n-i}}$	-	-	...	0

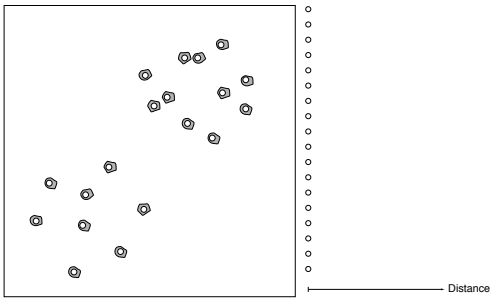
$\downarrow$

$t = n - 1$ C_{n_1}

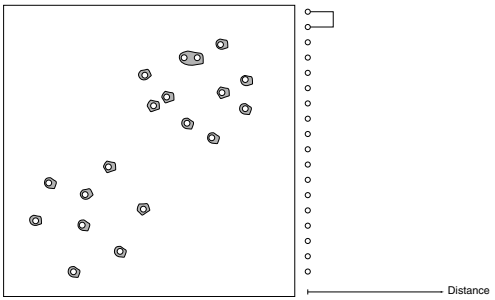
Agglomerative Hierarchical Clustering

Example (recap)

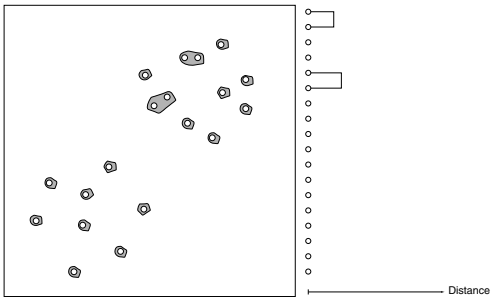
One cluster per instance



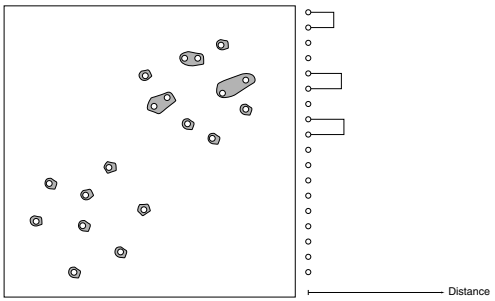
Merge closest cluster pair



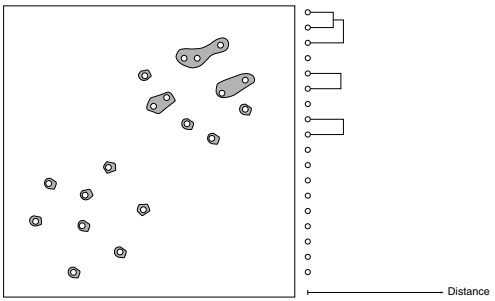
Repeat cluster merging



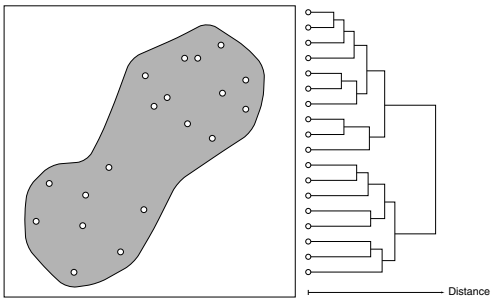
Repeat cluster merging



Repeat cluster merging ...



Terminate with one cluster



Agglomerative Hierarchical Clustering

Cluster Similarity Measures in Agglomerative Hierarchical Clustering

What similarity measure to choose?

- The choice of a measure is the key decision in hierarchical clustering.
- Different measures may result in fully different clusterings.

Common cluster similarity measures

- Single link. Using the nearest neighbors across two clusters C, C'.

$$sim(C, C') = max_{\mathbf{x} \in C, \mathbf{x}' \in C'} sim(\mathbf{x}, \mathbf{x}')$$

- Complete link. Using the furthest neighbors across two clusters C, C'.

$$sim(C, C') = min_{\mathbf{x} \in C, \mathbf{x}' \in C'} sim(\mathbf{x}, \mathbf{x}')$$

- Group-average link. Averaging over all similarities of two clusters C, C'.

$$sim(C, C') = \frac{1}{|C| \cdot |C'|} \sum_{\mathbf{x} \in C, \mathbf{x}' \in C'} sim(\mathbf{x}, \mathbf{x}')$$

Some other relevant measures exist but are omitted here, such as the *Ward criterion*.

Agglomerative Hierarchical Clustering

Characteristics of Cluster Similarity Measures

Overview of characteristics

	single link	**complete link**	**group-average link**
characteristic	contractive:	dilating:	conservative:
cluster number	low	high	medium
cluster form	extended	small	compact
chaining tendency	strong	low	low
outlier-detecting	very good	poor	medium
noisy data	susceptible	susceptible	unaffected
monotonicity	✓	✓	✓

Remarks

- Single link and complete link are the most popular similarity measures.
- Single link can be computed efficiently with a minimum spanning tree.
- Group-average link prefers spherical cluster forms, but it will also be able to detect “potato-shaped” clusters.
- A measure that is not monotonous is, e.g., the *median distance*.

Review Sentiment Analysis

What is sentiment analysis?

- The text analysis that assesses whether a text or a span of text conveys sentiment.
- One of the most tackled downstream tasks in text mining.
  HIgh industrial importance, e.g., in reputation management.
- Usually tackled with supervised classification.

Sentiment: Polarity vs. scores

- Polarity. *Positive* or *negative*, sometimes also *neutral* or similar.
- Scores. Scores from a numeric scale, e.g., $\{1, \ldots, 5\}$ or $[0, 1]$.
- Related. Subjectivity, emotion, stance, and similar.

Reviews

- Consumer judgments of products, services, and works of arts.
  For example, reviews of electronic devices, books, hotels, movies, etc.
- Reviews often comprise several “local” sentiments on different aspects.

Review Sentiment Analysis

Reviews across Topical Domains

Product review
from Amazon

Bought this based on previous reviews and is generally a good player. Setting it up seemed relatively straight forward and I've managed to record several times onto the hard drive without any problems. The picture quality is also very good and the main reason I bought it was the upscaling to match my TV - very impressive. Downsides are that if you have built-in freeview on your TV, it does get confused sometimes and will refuse to allow you to watch it through either TV or HDD player - I had to mess around with the settings several times to make it stop doing this. (Why did I buy it if I had freeview already? It was cheaper than to get one without) It is also very noisy and performs random updates in the night, which can be annoying. But in terms of function and ease of use it's very good.

Global sentiment:
neutral (3 out of 5)

Hotel review
from TripAdvisor

We stayed overnight at the Castle Inn in San Francisco in November. It was a fairly convenient to Alcatraz Island and California Academy of Science in Golden Gate Park. We were looking for a reasonably priced convenient location in SF that we did not have to pay for parking. Very basic motel with comfortable beds, mini refrig and basic continental breakfast. It was within walking distance to quite a few restaurants (Miller's East Coast Deli-yummy!) I did find that the clerk at the desk was rather unfriendly, though helpful. The free parking spaces were extremely tight for our mini van. The noise was not too bad, being only 1 block from Van Ness Ave. If you are looking for a no frills, comfortable place to stay, Castle Inn was a good choice.

Global sentiment:
neutral (3 out of 5)

Movie review
from Rotten Tomatoes

[...] The film was intense and pulsating when it zoomed in on Heather's travails, but lost something when it brought unnecessary action into play, such as a child kidnapping and the problem of drugs being sold in school. There was no place to go in developing Heather's character by adding these major societal problems to Heather's story [...]. Solondz knows his subject well, [...] and the result is an unusual movie that focuses in on a subject very few filmmakers have chosen to do. It was unfortunate that Heather never evolved, so the cruelty we observed in the beginning of the film was also the way she was observed when the film ended; nevertheless, an honest effort was put forth by the filmmaker to see how school age children cope with their unique problems they have.

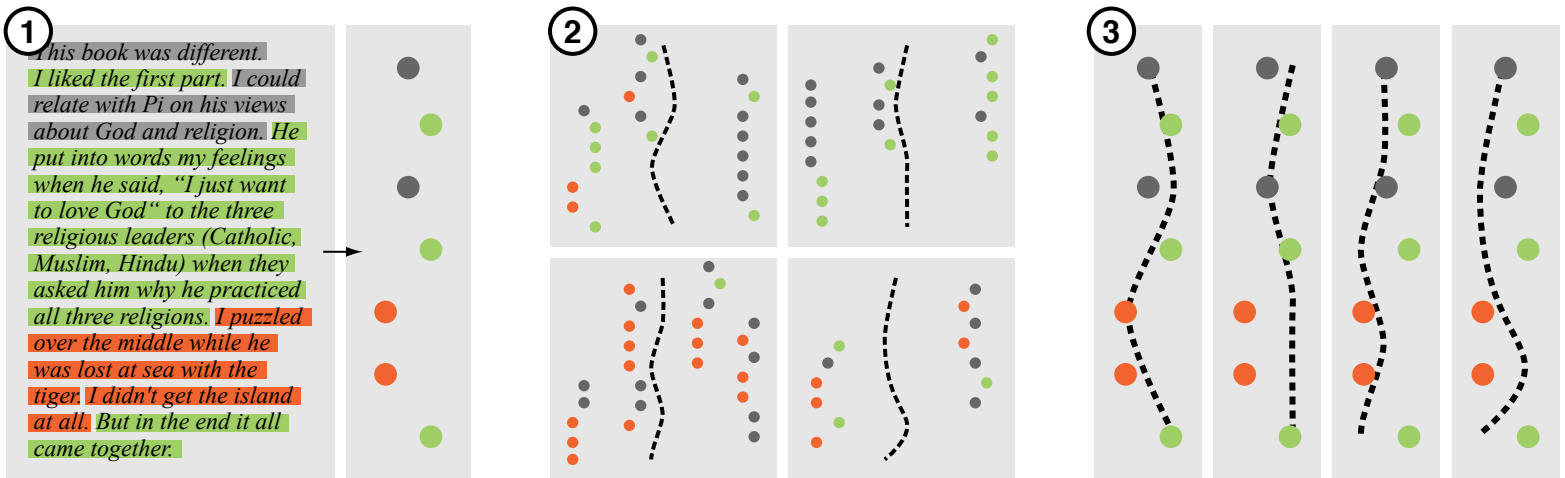
Global sentiment:
neutral (2 out of 3)

Review Sentiment Analysis

Discourse Pattern Recognition

Idea for cross-domain review sentiment analysis

- Model review discourse by the local sentiment flow in the review.
- Hypothesis. Similar flows occur across review domains.



Approach

1. Represent a review by its flow of local sentiment.
2. Cluster known training flows to identify a set of *discourse patterns*.
3. Analyze unknown flow based on its similarity to each pattern.

Review Sentiment Analysis

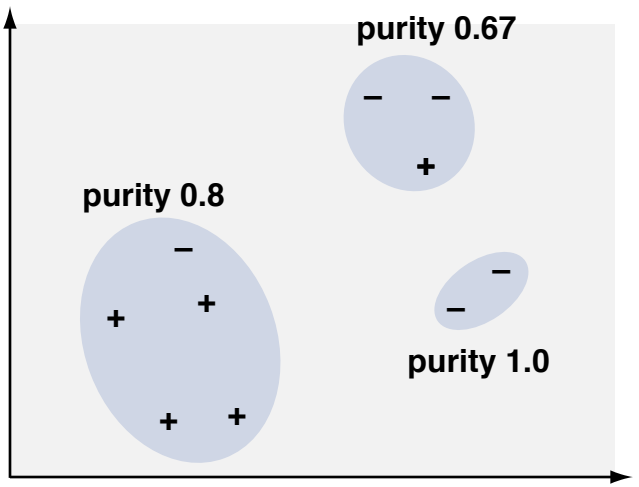
Discourse Patterns as Features

Idea of the discourse patterns

- Shall be used as complex features in review sentiment analysis.
- Should indicate global sentiment and have high “commonness”.

Supervised clustering (recap)

- Cluster instances with known classes.
- Clusters can be evaluated in terms of their *purity*, i.e., the fraction of instances whose class equals the majority class.
- The goal is to ensure that all clusters have a certain minimum purity.



Requirements for flow clustering

- The clusters should have a high purity.
- Their mean size should be high, i.e., the number of clusters is small.

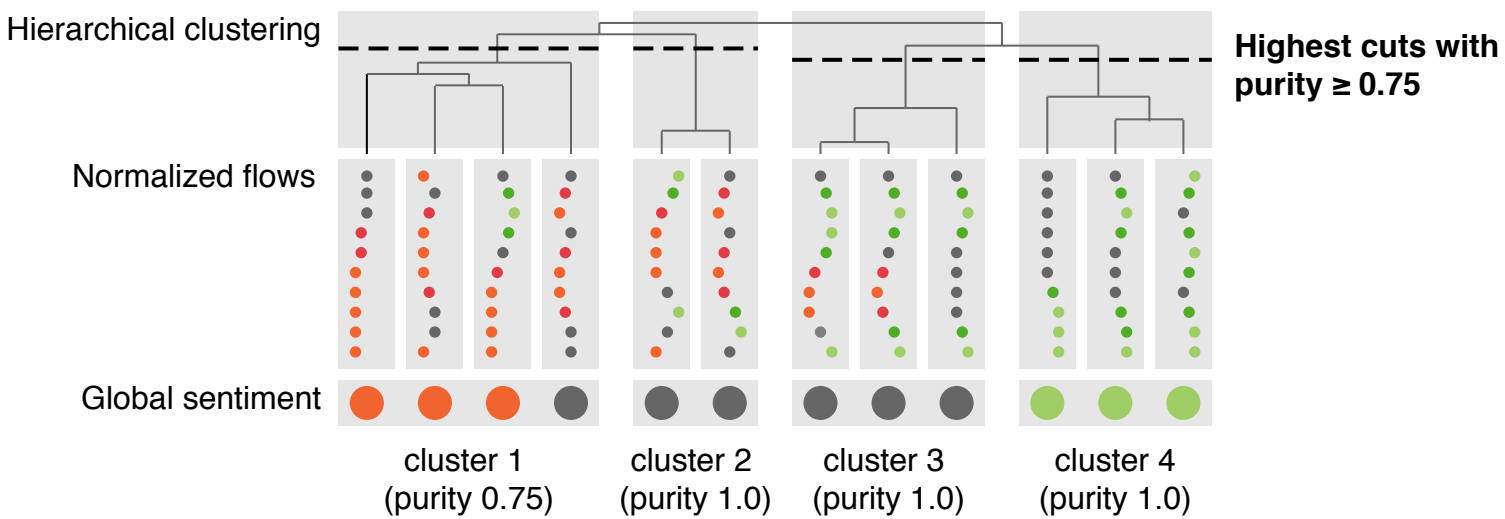
Review Sentiment Analysis

Supervised Clustering of Flows

Supervised clustering of flows

1. Length-normalize all sentiment flows from a training set.
2. Hierarchically cluster the normalized flows to obtain a binary tree.
3. Obtain the minimum number of flat clusters, by finding the cuts closest to the tree's root that create clusters with some defined minimum purity.

Example for a minimum purity of 0.75



Review Sentiment Analysis

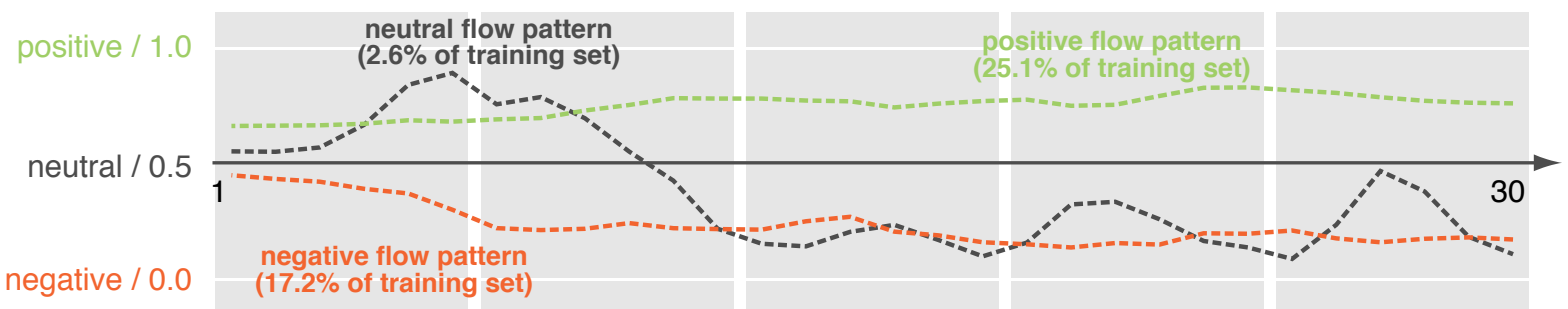
From Clusters to Discourse Patterns

Definition of discourse patterns

- The centroid of each cluster defines a discourse pattern.
  Small clusters might be discarded before, e.g., those of size 1.
- Local sentiment can be interpreted as real-valued between 0 and 1.
  Negative: 0.0, neutral 0.5, positive 1.0.
- The mean of all flows in a cluster may then define a discourse pattern.

Example discourse patterns

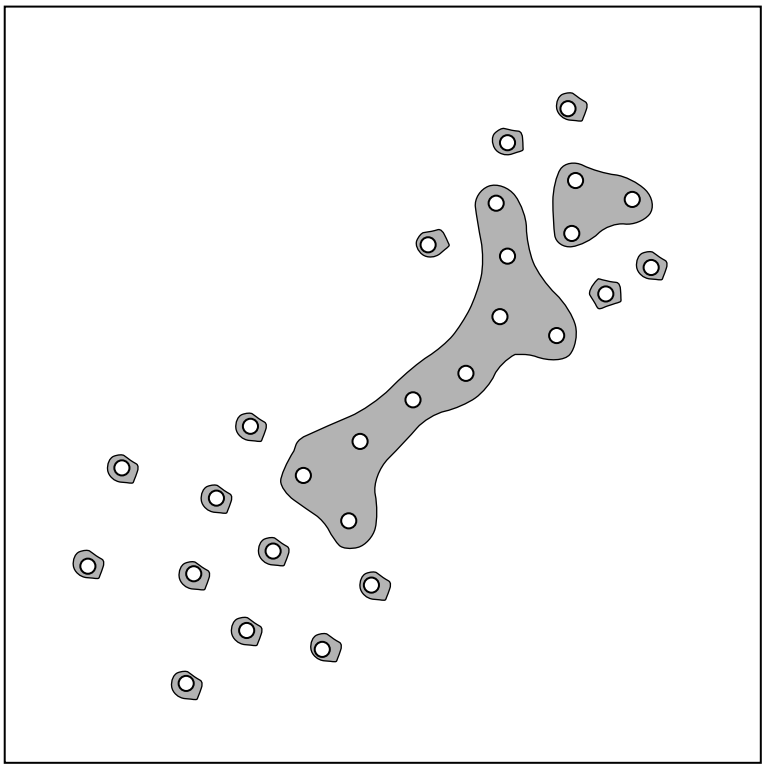
- The three most common patterns in 900 TripAdvisor reviews.



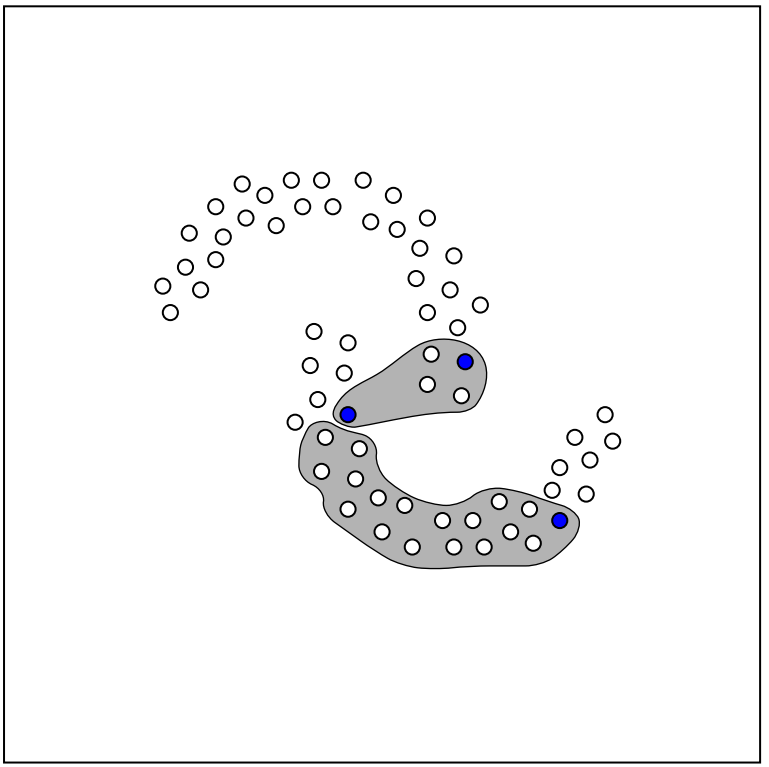
Hierarchical Clustering

Issues with Hierarchical Clustering Algorithms

Chaining problem of clustering using single-link similarity



Nesting problem of clustering using complete-link similarity

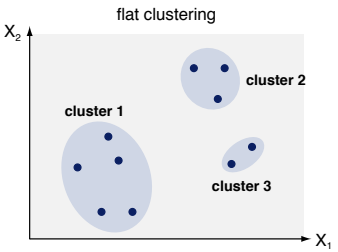


Conclusion

Summary

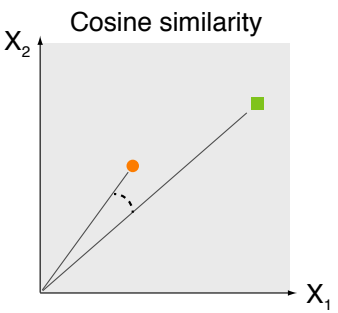
Text Mining using clustering

- Mostly-unsupervised grouping of texts and text spans.
- Targets situations where classes are unknown.
- Relies on similarities between problem instances.



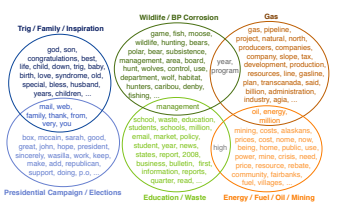
Similarity measures

- Vector-based measures to compare feature vectors.
- Synonyms and embeddings generalize beyond words.
- Clustering often uses cosine similarity or comparable.



Clustering techniques

- Hard clustering identifies disjunct classes of instances.
- Soft clustering (incl. topic modeling) models overlaps.
- Hierarchical clustering stepwise organizes instances.



References

Some content and examples taken from

- David J. Blei (2012). Probabilistic Topic Models. Tutorial at the 29th International Conference on Machine Learning (ICML 2012). http://www.cs.columbia.edu/~blei/talks/Blei_ICML_2012.pdf
- Sung-Hyuk Cha (2007). Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. International Journal of Mathematical Models and Methods in Applied Sciences, 1(4):300–307.
- Daniel Jurafsky and Christopher D. Manning (2016). Natural Language Processing. Lecture slides from the Stanford Coursera course. https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html.
- Efstathios Stamatatos, Michael Tschuggnall, Ben Verhoeven, Walter Daelemans, Günther Specht, Benno Stein, and Martin Potthast (2016). Clustering by Authorship Within and Across Documents. In Working Notes of the CLEF 2016 Evaluation Labs.
- Benno Stein and Theodor Lettmann (2010). Data Mining. Lecture Slides. https://webis.de/lecturenotes/slides.html#data-mining
- Henning Wachsmuth (2015): Text Analysis Pipelines — Towards Ad-hoc Large-scale Text Mining. LNCS 9383, Springer.
- Henning Wachsmuth and Benno Stein (2017). A Universal Model of Discourse-Level Argumentation Analysis. Special Section of the ACM Transactions on Internet Technology: Argumentation in Social Media, 17(3):28:1–28:24.