Text Data Analysis

CiTIUS Centro Singular de Investigación en Tecnoloxías Intelixentes

Tecnologías de Gestión de Información No Estructurada

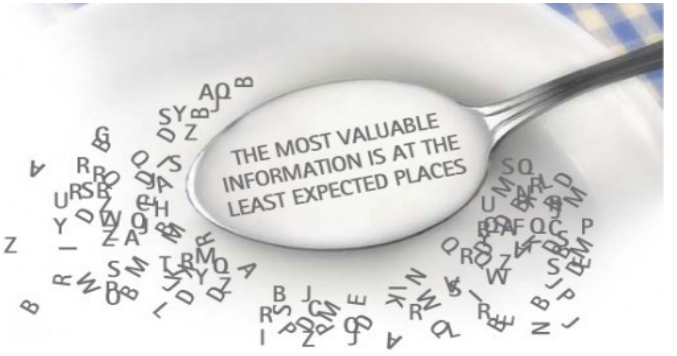
Prof. Dr. David E. Losada

USC Universidade de Santiago de Compostela

Campus Vida

Máster Interuniversitario en Tecnologías de Análisis de Datos Masivos: Big Data

text data analysis



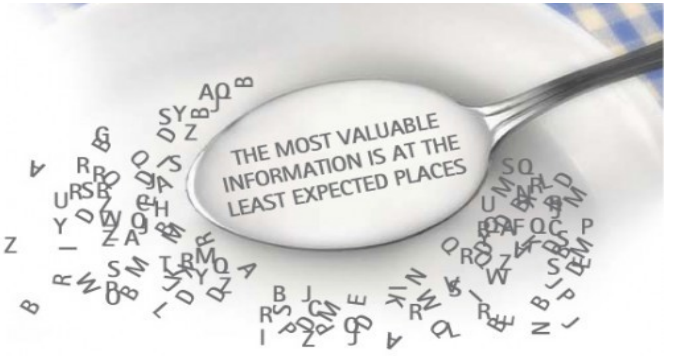
"When you two have finished arguing your opinions, I actually have data!"

extract and discover **useful actionable knowledge** that can be directly used for **decision making** or supporting a user's task

manual processing of text data is **unfeasible**

essential to develop **advanced text analysis tools** to help us **digest** and make use of text data

text data analysis: 2 kinds



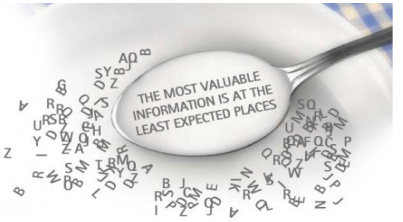
1) replace our current manual labor in **digesting text content;** improve our productivity; do not do anything beyond what we humans can do

e.g, automatic sorting of emails

2) **discover knowledge** that we humans may not be able to do even if we have "sufficient" time to read all the text data.

e.g. an intelligent biomedical literature analyzer may reveal a chain of associations of genes and diseases by synthesizing info scattered in many different research articles

text data analysis: app domains



"Good news, the computer says you can handle 20% more work."

Business Intelligence. e.g., product managers may be interested in hearing customer feedback about their products

Scientific Research. e.g., understand the trends of research topics or learning about discoveries in fields

Disaster response and management. e.g., early discovery of any warning signs of a natural disaster

different types of sensors

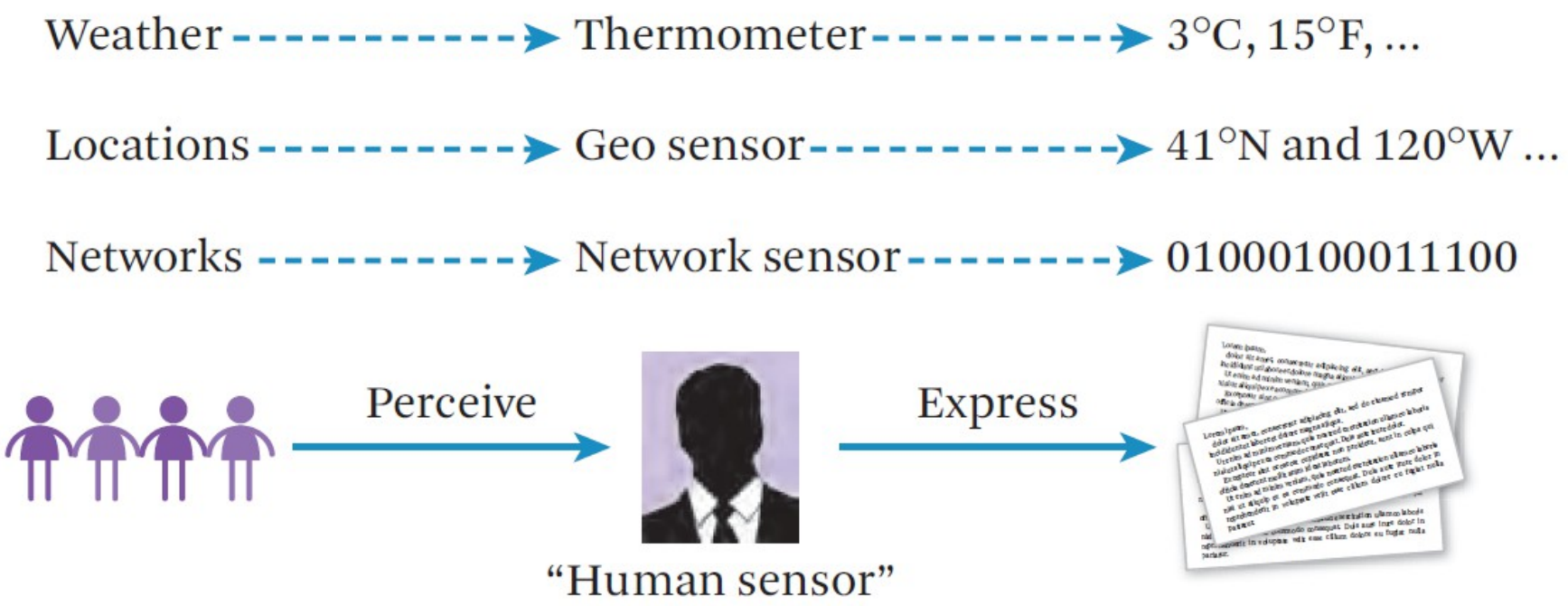


Figure 12.1 Humans as subjective sensors.

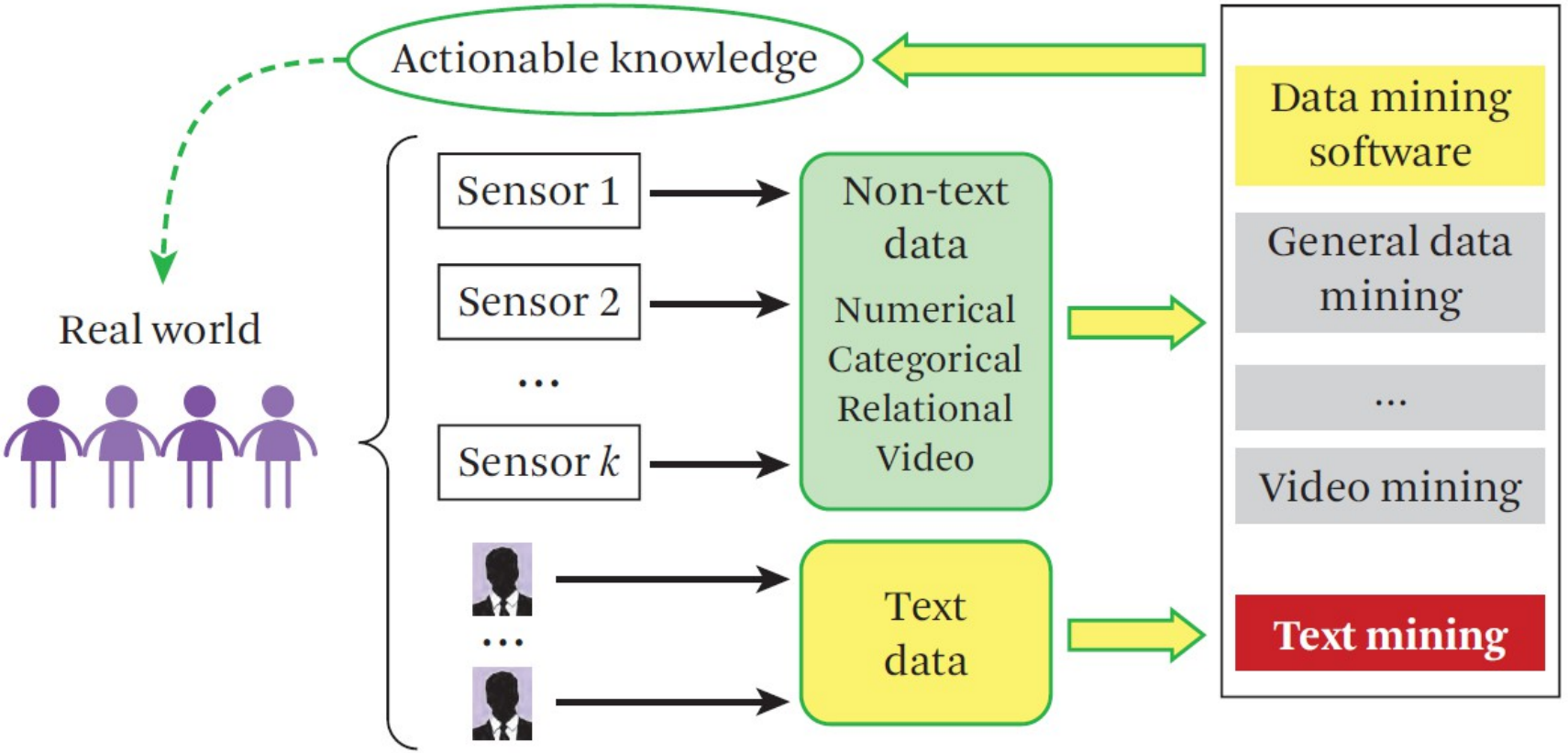


Figure 12.2 The general problem of data mining.

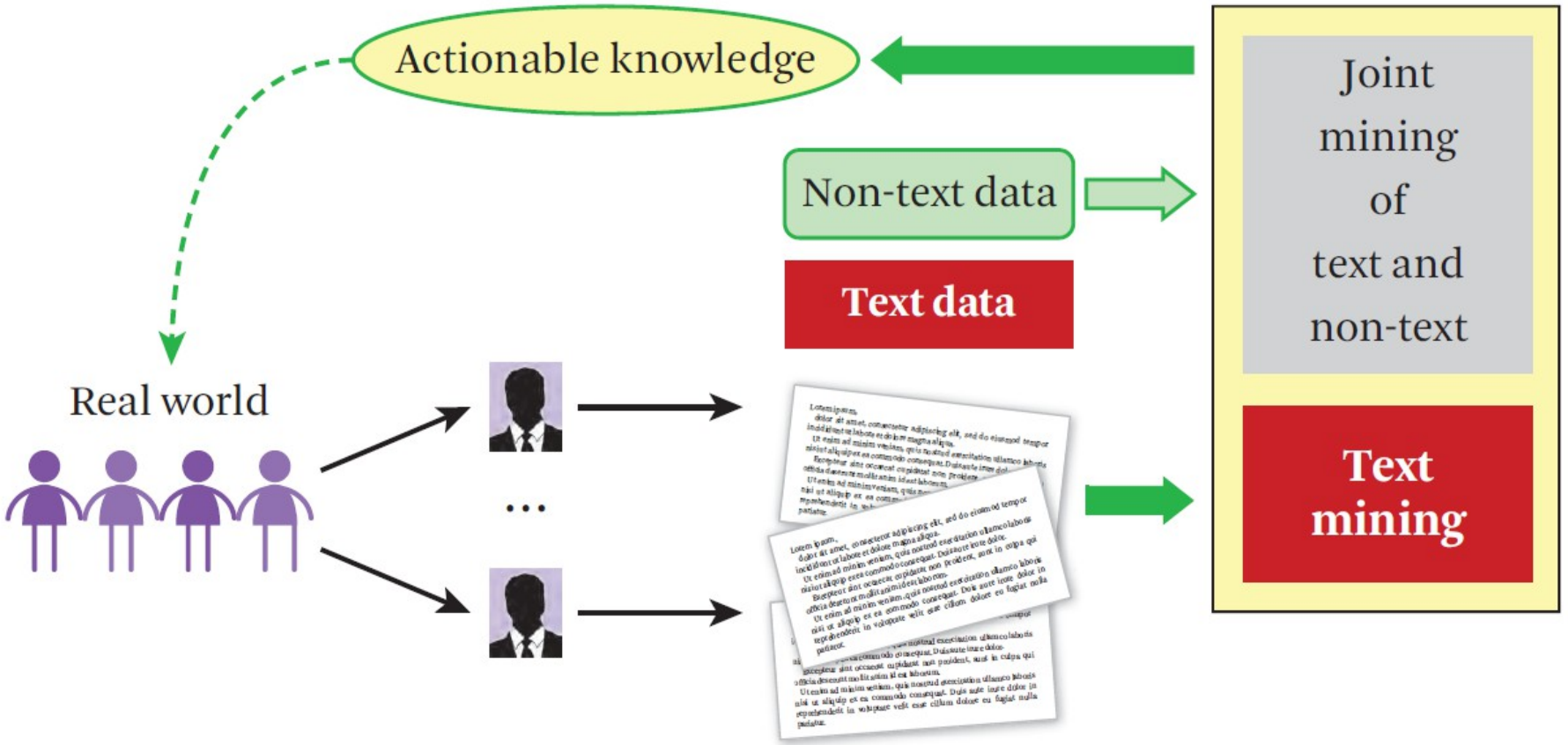


Figure 12.3 Text mining as a special case of data mining.

text mining tasks

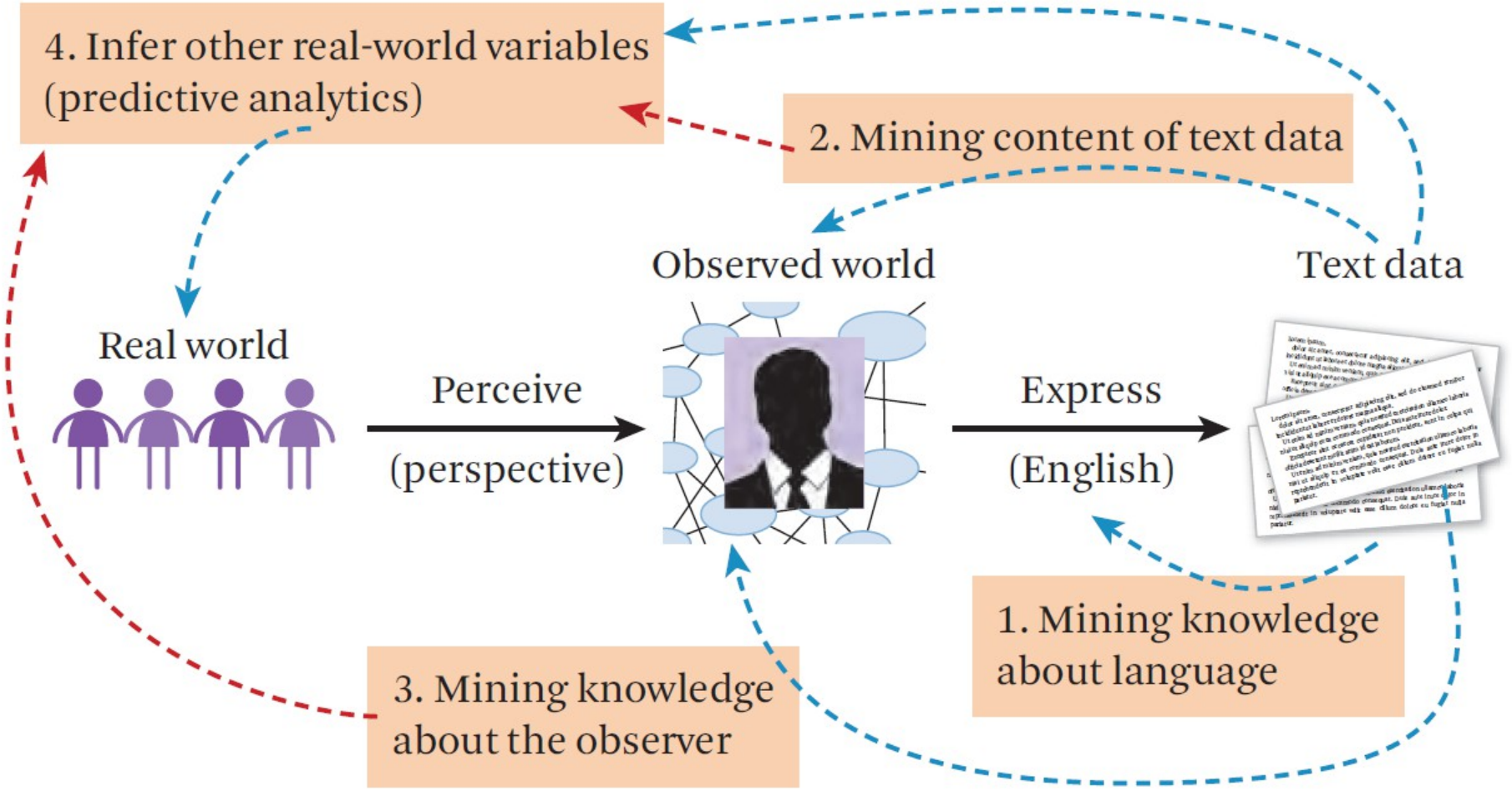


Figure 12.4 Mining different types of knowledge from text data.

text mining tasks

Mining knowledge about natural language. For example, if the text is written in English, we may be able to discover knowledge about English, such as usages, collocations, synonyms, and colloquialisms.

Mining knowledge about the observed world. For example, we can discover everything that has been said about a particular person or a particular entity.

Mining knowledge about the observers (text producers). Infer some properties of the authors that produced the text data, such as the mood or sentiment of the person toward an issue.

Inferring knowledge about properties of the real world. For example, there may be some correlation between the stock price changes on the stock market and the events reported in the news data

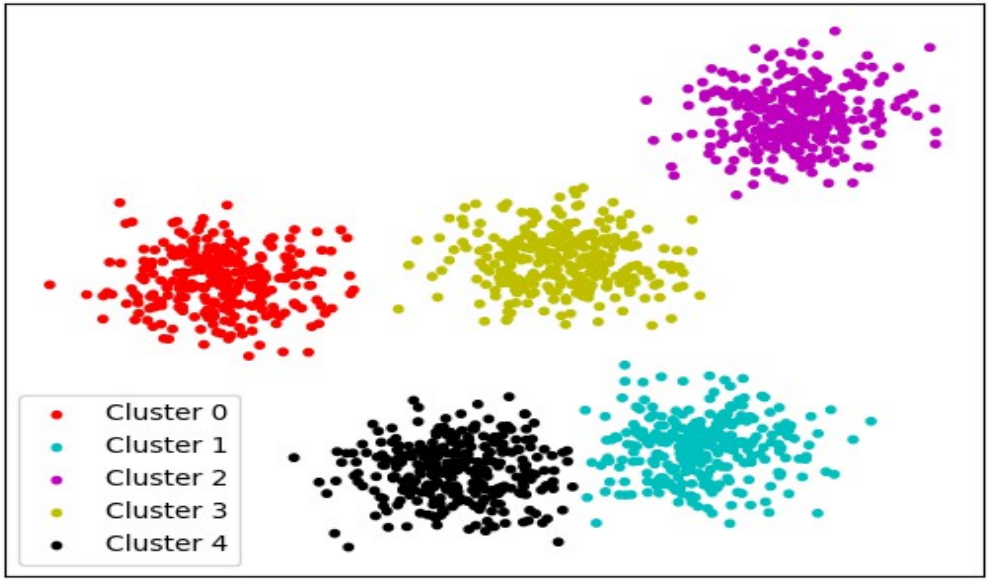
text clustering

discover some **inherent structure**

in our corpus

by **collecting similar objects**

(docs, sentences, words,...)

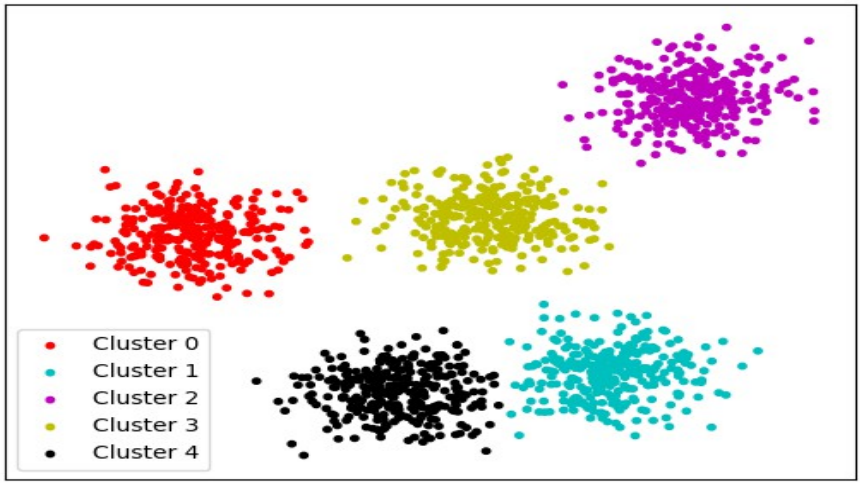


e.g. cluster search engine results, cluster Social Media users

can reveal **natural semantic structures** in the data in the form of multiple clusters

useful for understanding a data set at a high-level before zooming into any specific subset of the data for focused analysis

text clustering



can support **navigation**

into the relevant subsets

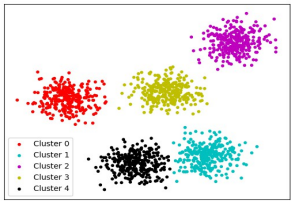
unsupervised learning

term clustering: employed

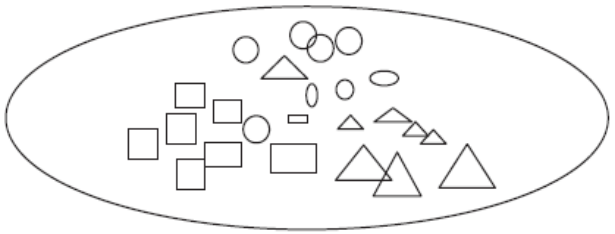
in **query expansion**

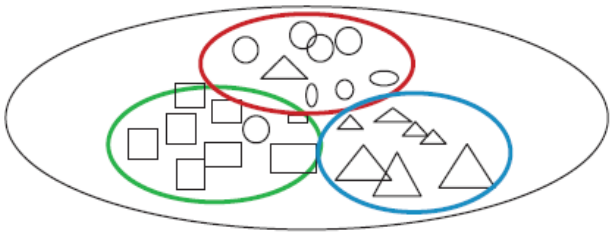
often **does not give labels** for the clusters found

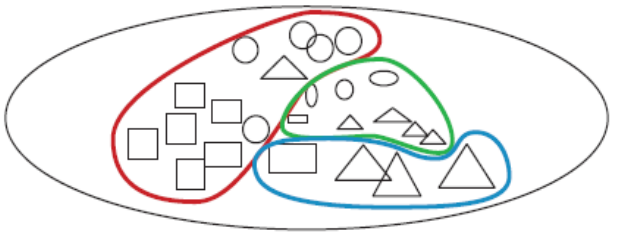
text clustering



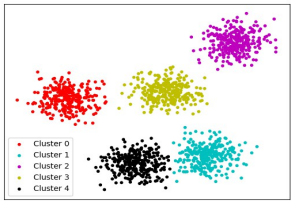
an appropriate definition of **similarity** is **crucial**







document clustering



Similarity-based clustering.

Agglomerative clustering: “bottom-up” a.k.a hierarchical

gradually merges similar objects to generate clusters.

Divisive clustering: “top-down” approach.

gradually divides the whole set of objects into smaller clusters.

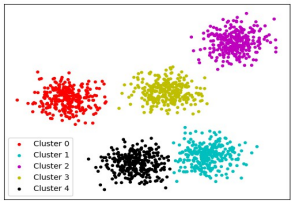
each doc can only belong to one cluster (“hard assignment”)

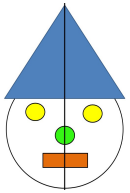
Model-based techniques.

Design a probabilistic model to capture the latent structure of data.

“soft clustering”: one object can be in multiple clusters

sim-based document clustering

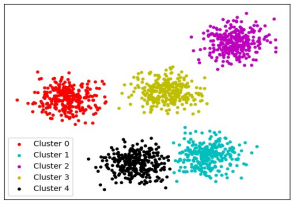




symmetric measure: sim(d1, d2) = sim(d2, d1)

normalized measure (typically in [0,1]) (to fairly compare similarity scores of different pairs of objects)

sim-based document clustering



cosine similarity

$$\text{sim}_{\text{cosine}}(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i (x_i)^2}\sqrt{\sum_i (y_i)^2}}.$$

term weights might be raw counts, tf/idf or BM25 weights

Jaccard similarity

$$\text{sim}_{\text{Jaccard}}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|},$$

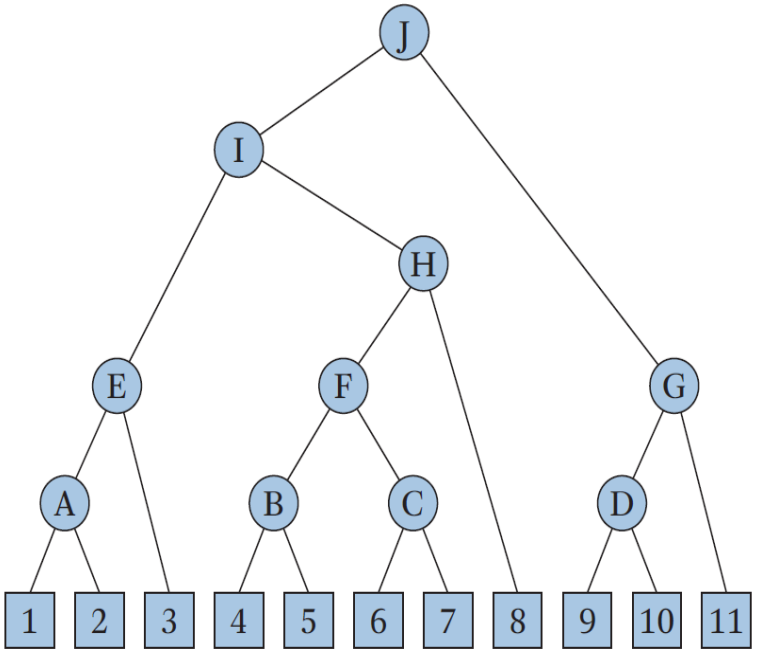
only captures the presence and absence of terms

X and Y represent the set of elements in the two docs

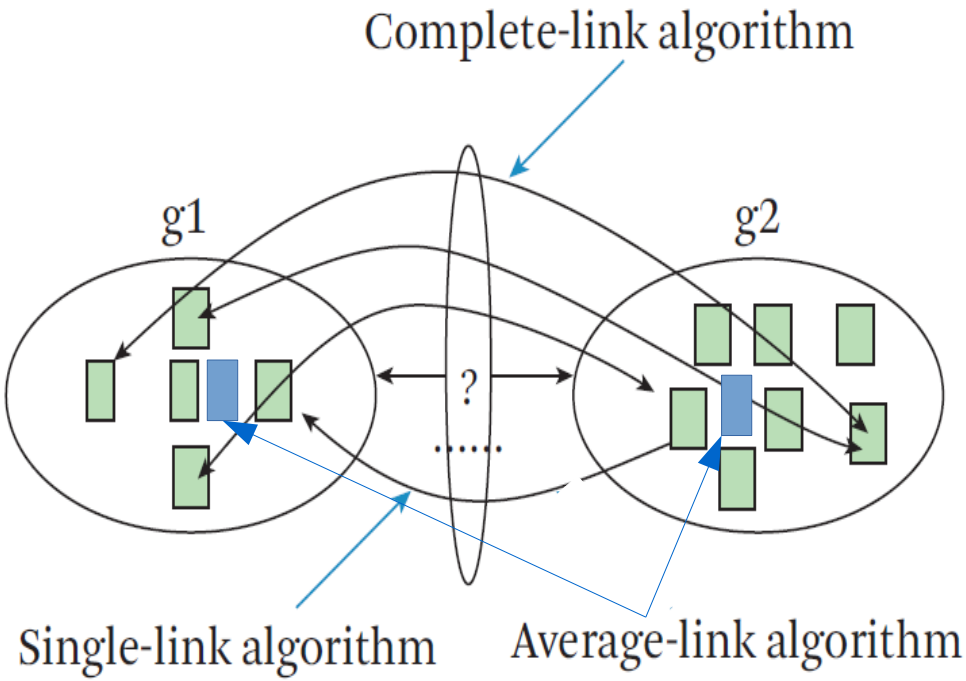
agglomerative hierarchical clustering

bottom-up: progressively constructs clusters to generate a hierarchy of merged groups

gradually groups similar objects (single docs or groups of docs) into larger and larger clusters **until there is only one cluster left**



Hierarchical clustering represented as a dendrogram.



Three different cluster-cluster similarity metrics.

single-link: merges the two clusters with the smallest minimum distance

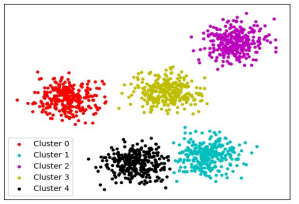
complete-link: merges the two clusters with the smallest maximum distance

average-link takes the smallest average distance between two clusters

single-link and complete-link: sensitive to **outliers** (they rely on only the sim of one pair of documents)

Average-link: group decision, less sensitive to **outliers**

k-means

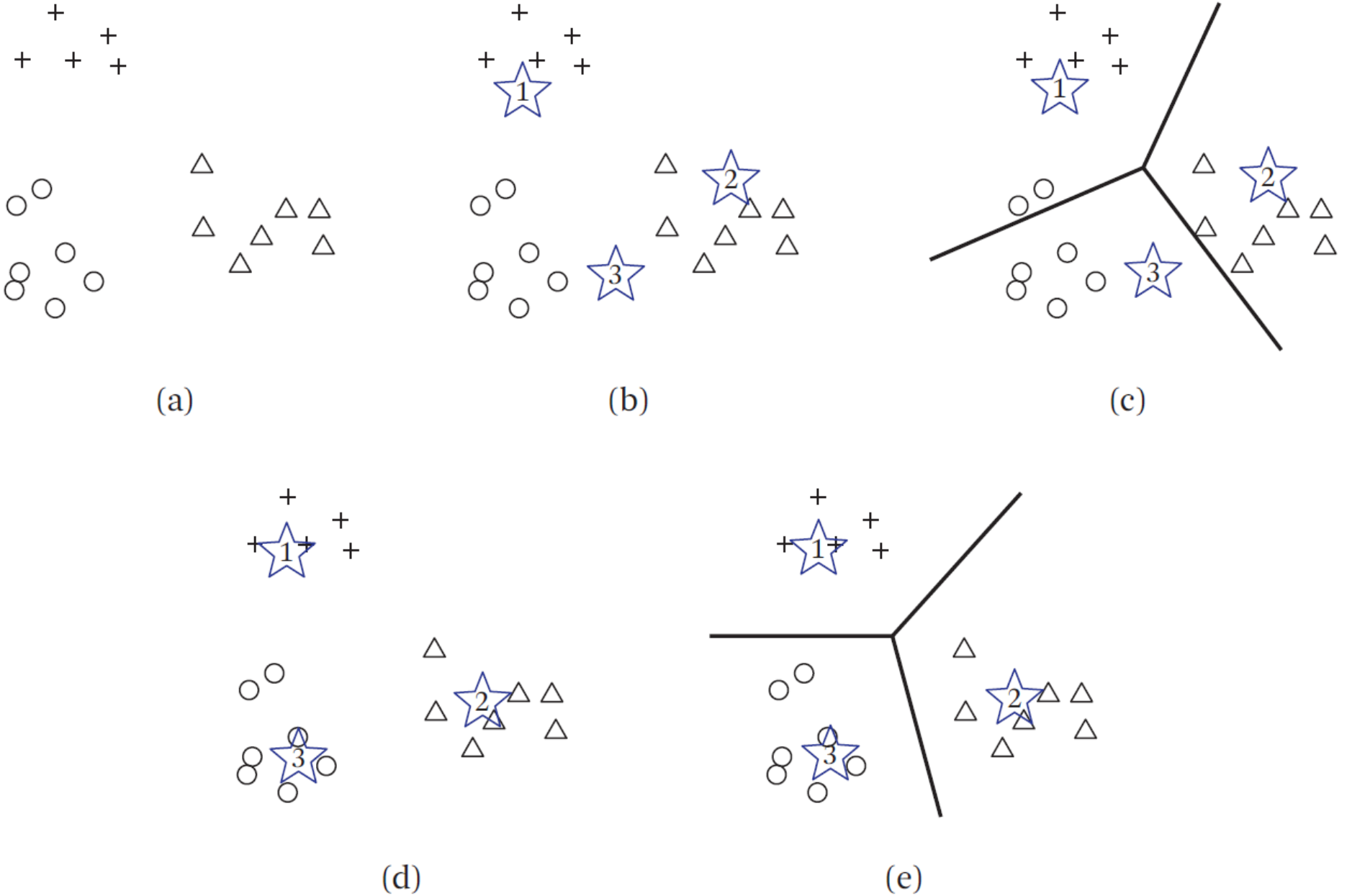


starts with an **initial tentative clustering** and iteratively improves it until we reach some stopping criterion

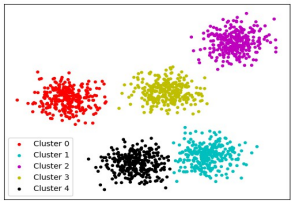
K-means clustering algorithm

Initialize K randomly selected centroids
while not converged **do**
    Assign each document to the cluster whose centroid is closest to it using sim(·) (Ex.)
    Recompute centroids of the new clusters found from previous step (Max.)
end while

k-means



evaluation of clustering



Coherence. How similar are objects in the same cluster?

Separation. How far away are objects in different clusters?

Utility. How useful are the discovered clusters for an application?

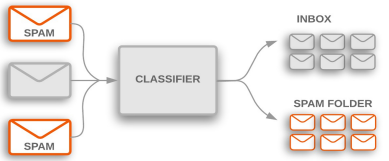
manual evaluation (using humans) or

automatic evaluation (using predefined measures)

coherence/separation can be measured automatically with measures such as **vector similarity**, **purity**, or **mutual information**.

indirect evaluation (use of the clusters to support some tasks)

text categorization



pre-defined categories

(e.g. tech, sports, politics, entertainment, etc)

often **supervised** (training data)

enriches text representation ("annotates" texts with relevant categories)

helps to **infer properties** of the entities (e.g., "Nadal" or "Federer" in Sports News)

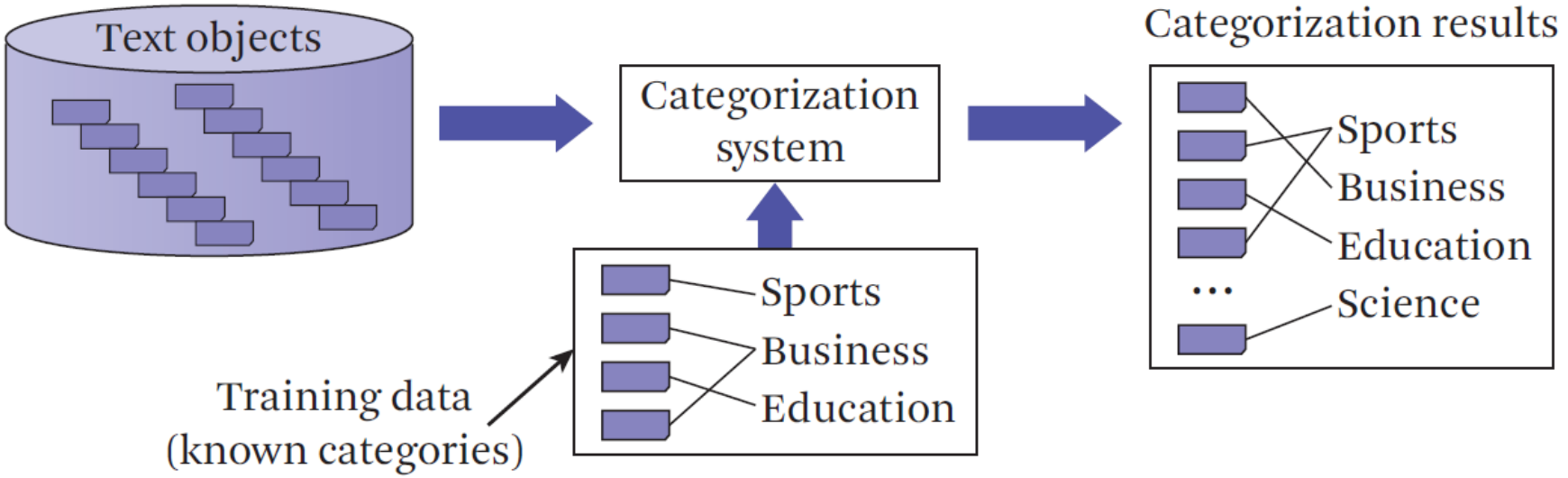
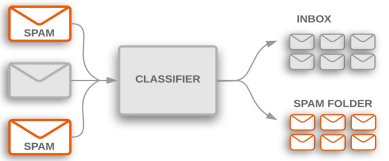


Figure 15.1 The task of text categorization (with training examples available).

text categorization methods



no training data: rule-based

if the word “governor” occurs→assign politics label

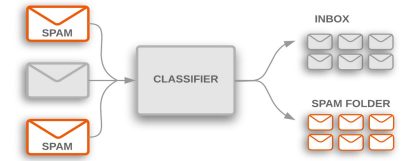
labor-intensive, does not scale up well**, unreliable** rules

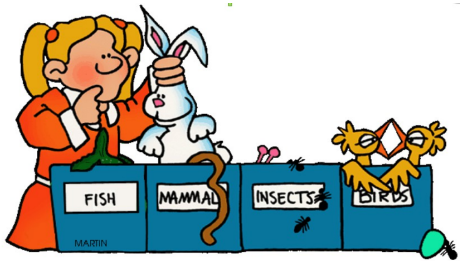
supervised learning: training data.

labeling some examples with the correct categories (i.e., creating training examples)

the machine will **learn from these examples** to somewhat automatically construct rules for categorization

text categorization methods

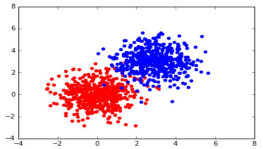




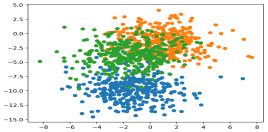
once a classifier (categorizer) is **trained**, it can be used to categorize any unseen text data

categorization methods rely on **discriminative features** of text objects to distinguish categories, and **combine** multiple features in a weighted manner where the weights are automatically learned (i.e., adjusted to minimize errors of categorization on the training data)

different methods tend to vary in their way of measuring the errors on the training data and their way of combining features (e.g., linear vs. non-linear)

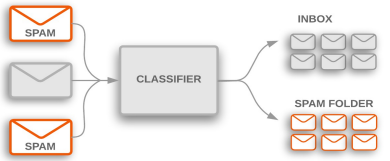


binary classification: only two categories



multiclass classification supports an arbitrary number of labels

text categorization: features

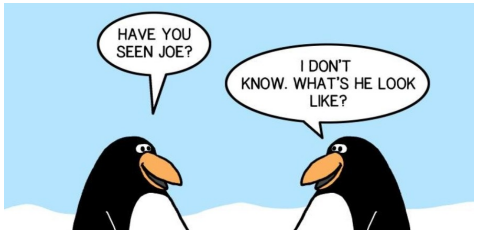


depends on the task, but **typically word-based**

other features might be useful for other tasks (e.g. presence of links for cataloging emails into spam/ham or length of sentences to estimate pass/fail of a student in a test)

feature selection

lazy learners (instance-based classifiers)



do not model the class labels explicitly **compare** the new **instances** with instances seen before

similarity measure

“**lazy**”: lack of explicit generalization or training step; most calculation is performed at testing time.

(e.g. knn)

k-nearest neighbours (knn)

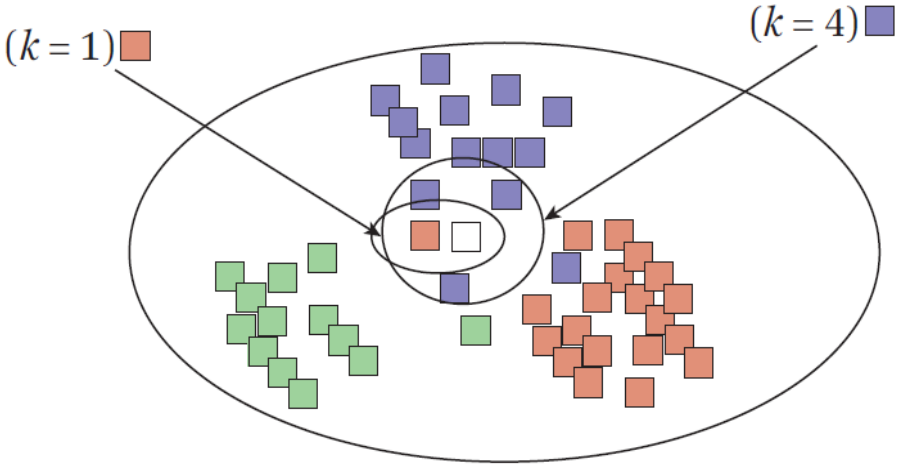
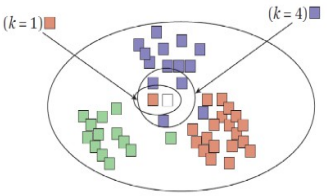


Figure 15.3 An example of k-NN with three classes where $k = 1, 4$. The query is represented as the white square.

k-nearest neighbours (knn)



k-NN Training

Create an inverted index over the training documents

---

k-NN Testing

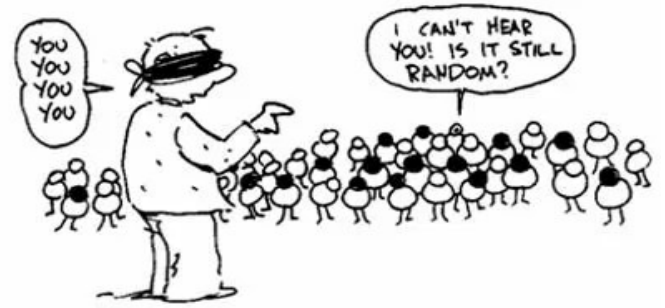
Let R be the results from searching the index with the unseen document as the query
Select the top k results from R
return the label that is most common in the k documents via majority voting

---

generative classifiers

model the data distribution in each category (e.g., unigram language model for each category).

classify an object based on the likelihood that the object would be observed **according to each distribution**



Näive Bayes:

$$\hat{\theta} = \arg\max_\theta \, p(w_1, \ldots, w_n \mid \theta) = \arg\max_\theta \prod_{i=1}^{n} p(w_i \mid \theta).$$

näive bayes

Naive Bayes Training

Calculate $p(y)$ for each class label in the training data

Calculate $p(x_i \mid y)$ for each feature for each class label in the training data

---

Naive Bayes Testing

return the $y \in \mathbf{Y}$ that maximizes $p(y) \cdot \prod_{i=1}^{n} p(x_i \mid y)$

---

discriminative classifiers

compute **features** of a text object that can provide a clue about which category the object should be in, and combine them with parameters to control their **weights**

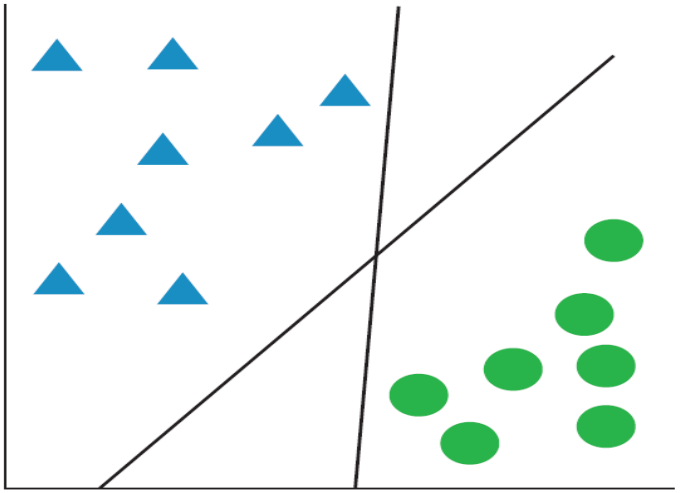
parameters are optimized by **minimizing categorization errors** on training data

discriminative classifiers

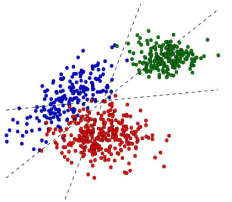
ex. linear Support Vector Machines (SVMs)

<u>linear</u> classifiers: decision is based ona linear combination of feature weights (w and x):

$$f(x) = \begin{cases} +1 & \text{if } w \cdot x > 0 \\ -1 & \text{otherwise} \end{cases}$$

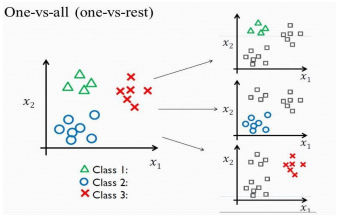


multiclass classification



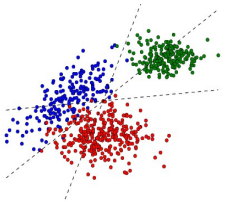
one-vs-all (OVA): trains one classifier per class (for k total classifiers).

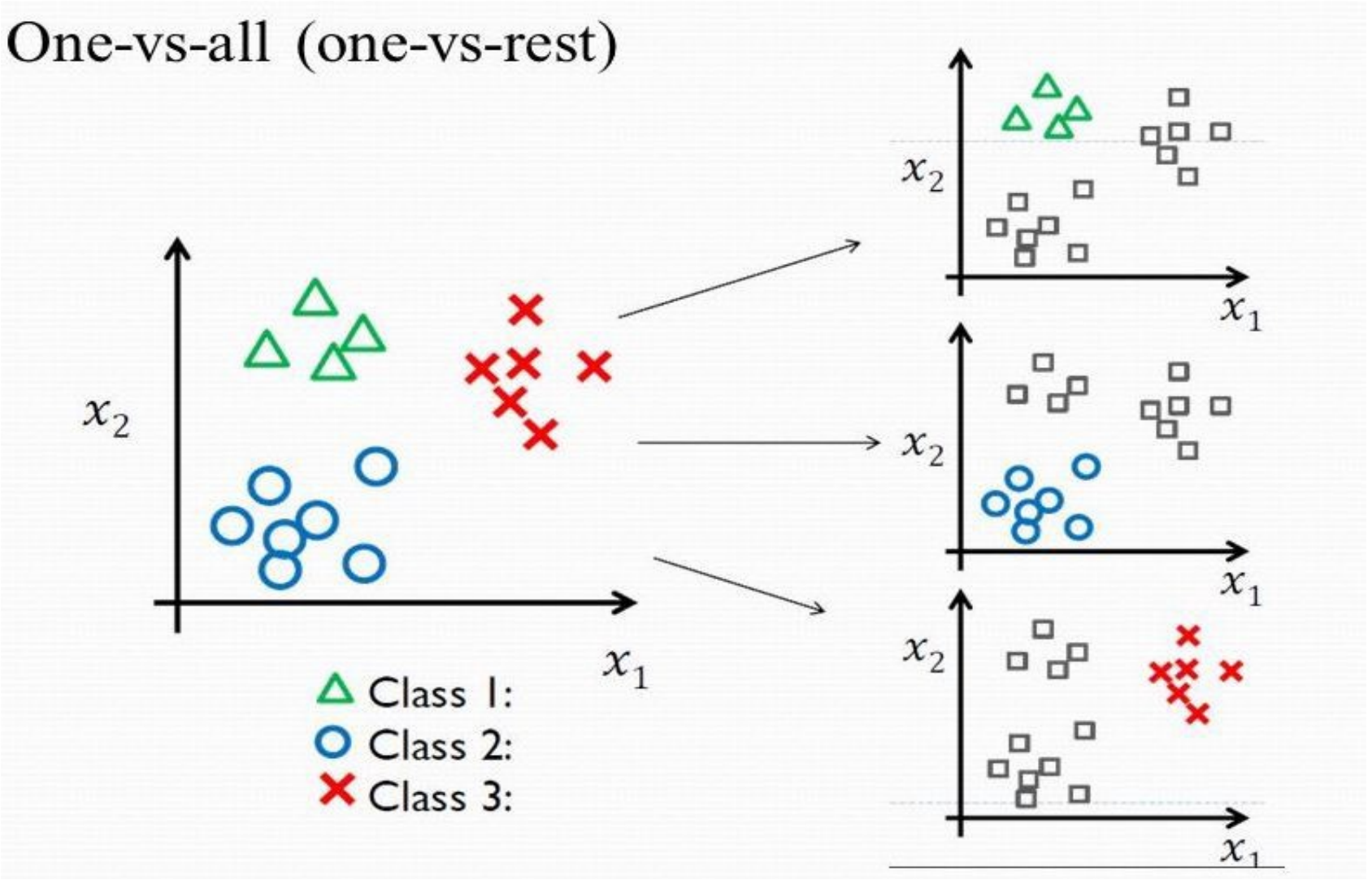
each classifier is trained to predict +1 for its respective class and −1 for all other classes.



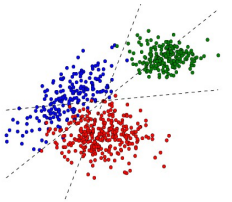
a test example is passed to these k classifiers and it is assigned with the **class whose classifier is more confident**

multiclass classification: OVA

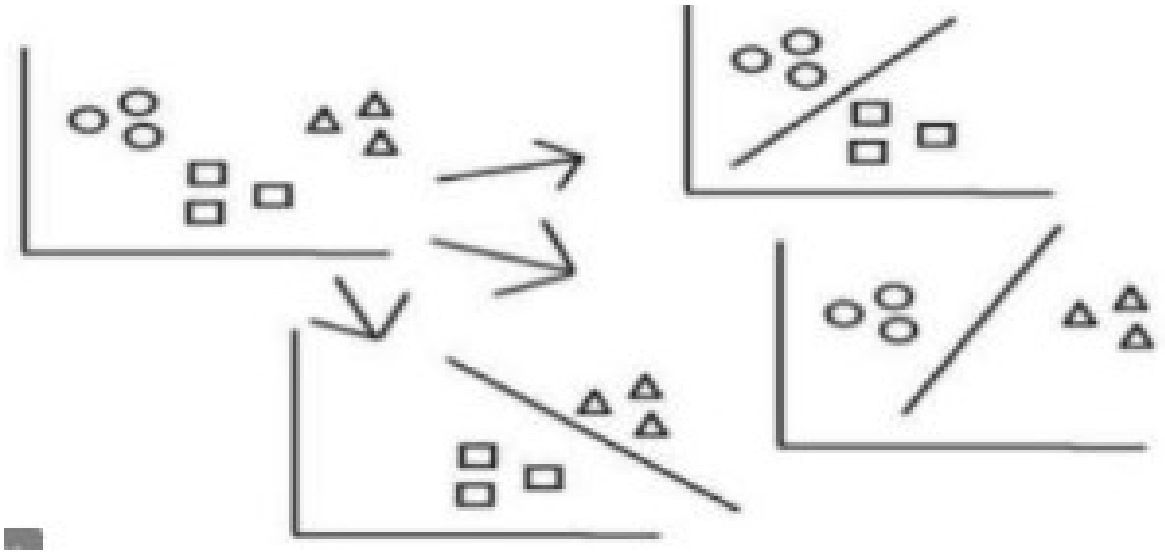




multiclass classification



all-vs-all: trains (k*(k−1))/2 classifiers to distinguish between all pairs of k classes. The class with the most +1 predictions is chosen as the final answer.



text categorization: evaluation

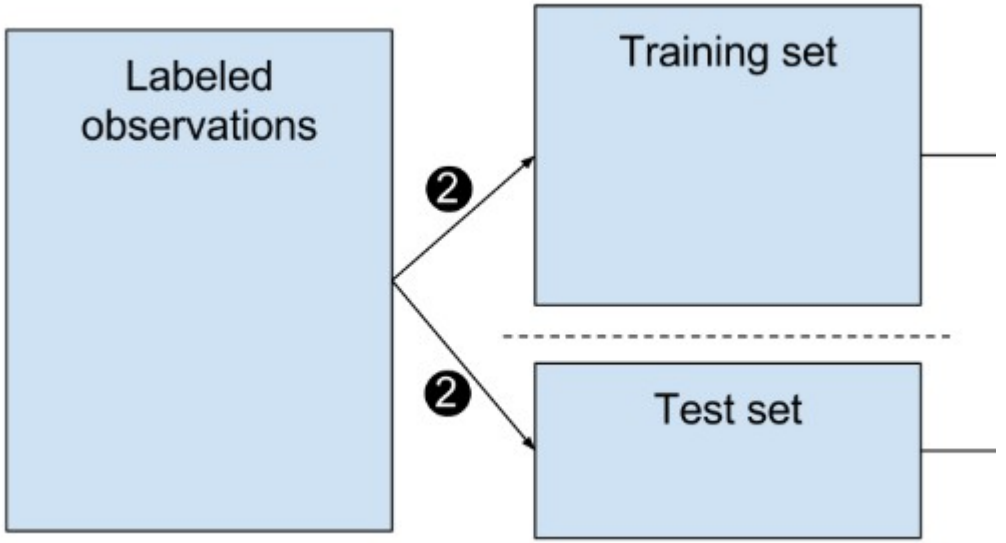
we can use **precision**, **recall**, and **F1** score (considering true positives, false positives, true negatives, and false negatives)

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

accuracy: number of correct predictions divided by the number of total predictions.

text categorization: evaluation

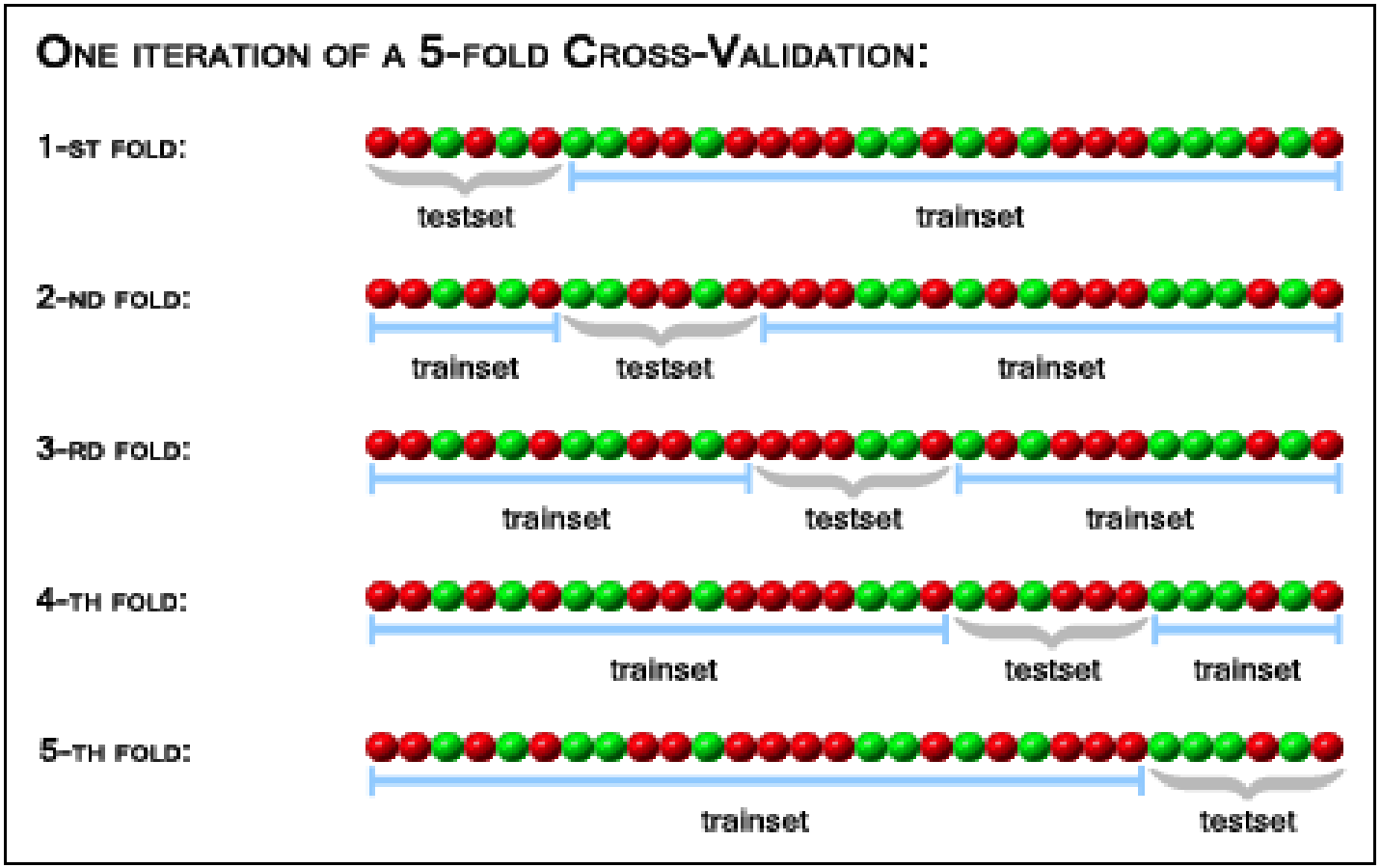
train/test split



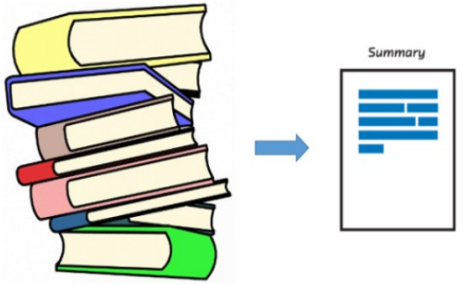
training-development-test

text categorization: evaluation

k-fold cross validation



text summarization



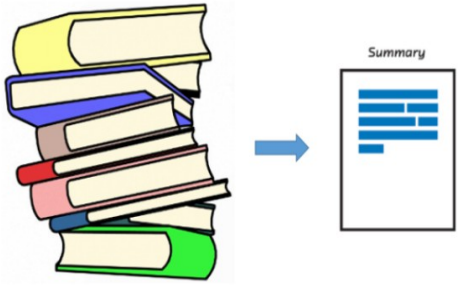
compressing a relatively large amount of text data or a long text article into a more **concise form** for easy digestion

helps users see the main content or points in the text data without having to read all the text

e.g. summarization of search engine results (**snippets**)

difficult: how can we convey the important points in only a few sentences?

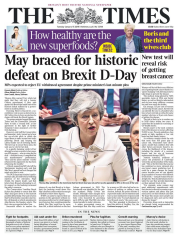
text summarization



ideally a **semantic** compression of text (same info in less space)

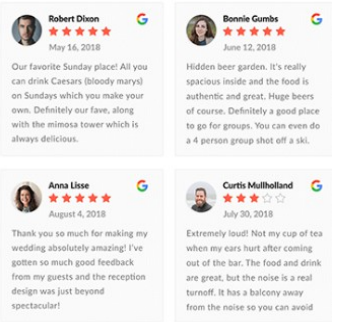
output should be **fluent**, **readable** language

example: a **news summarization** task



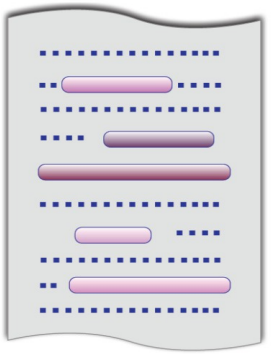
input: a text news article; output: one paragraph explaining what the article talks about

opinion summarization is useful for both businesses and shoppers



selection-based (extractive summarization)

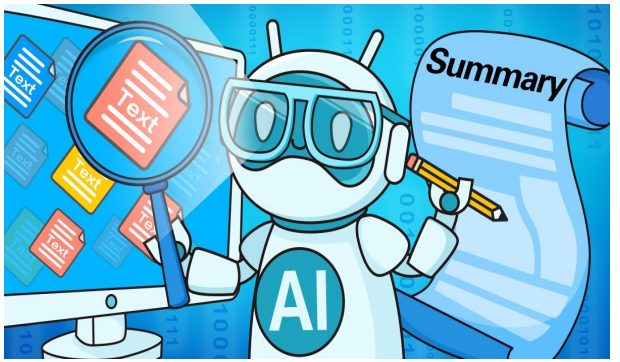
summary: a sequence of sentences **selected the original documents**



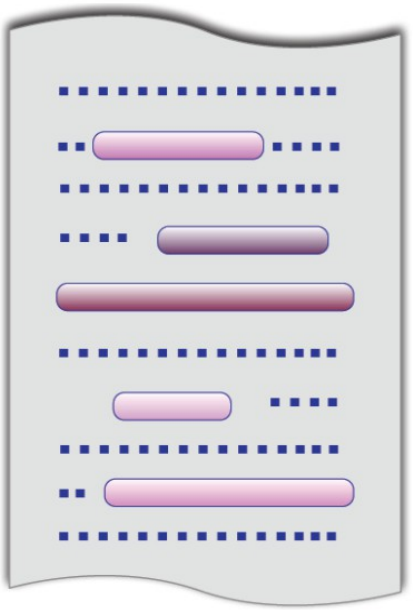
no new sentences are written

generation-based (abstractive summarization)

a summary **may contain new sentences** not in any of the original documents (e.g. using Language Models or Natural Language Generation)



extractive text summarization



information retrieval-based techniques:

sentence vectors & similarity functions

1. **split** the document to be summarized into sections or **passages**

2. for each passage, "**compress**" its sentences into a smaller number of relevant (yet not redundant) sentences

extractive summarization

1) segmentation

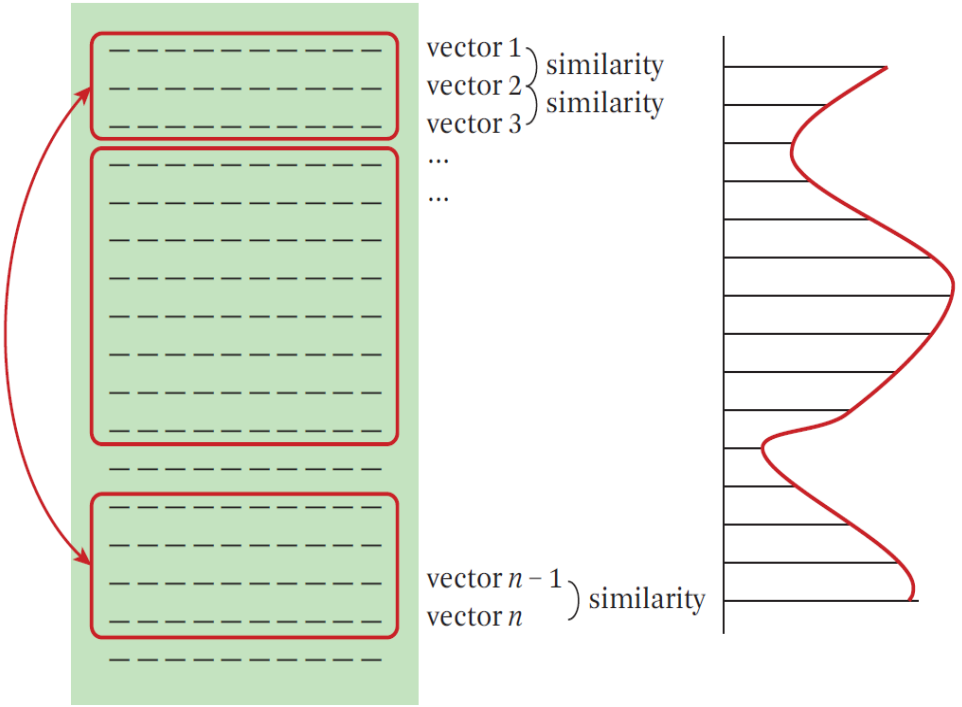


Figure 16.1 Segmenting a document into passages with a similarity-based discourse analysis.

the plot on the right-hand side of the figure shows the **change in similarity between the sentences. segment the document** into passages **when the similarity is low**, i.e., a shift in topic occurs.

alternative approach: use **paragraphs** to segment the docs

sophisticated segmentation: **discourse analysis** (NLP)

extractive summarization

2) summarization

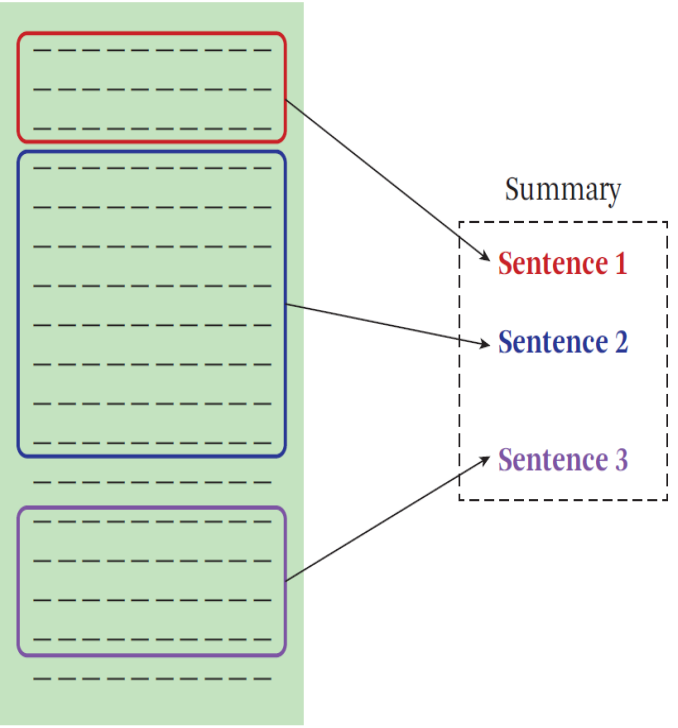


Figure 16.2 Text summarization using maximum marginal relevance to select one sentence from each passage as a summary.

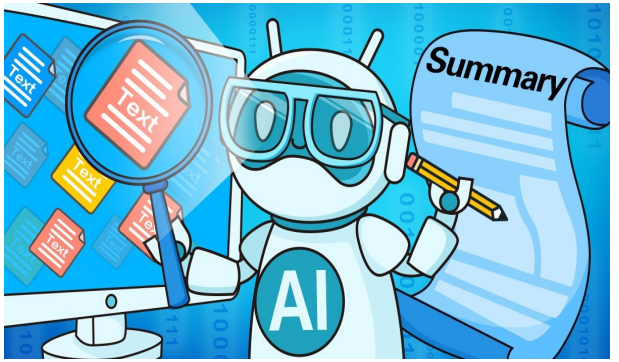
given the passages (or segments), obtain **non-redundant sentences from each passage**

maximal marginal relevance (MMR): selects important sentences that are non-redundant wrt the sentences chosen before

we can include **additional criteria** to select sentences (e.g. **position** -leading sentences preferred- , **centroid** features)

abstractive text summarization

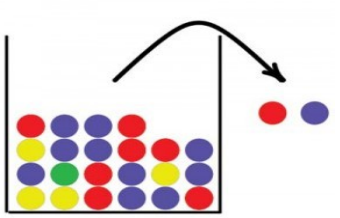
creates sentences that did not exist in the original document(s)



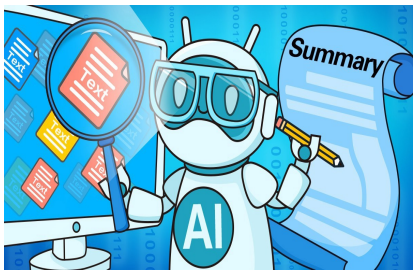
e.g. use a **language model** to represent the original text

gives us a **principled way** in which to generate text

to create our summary, we will **draw words** from this probability distribution



abstractive text summarization



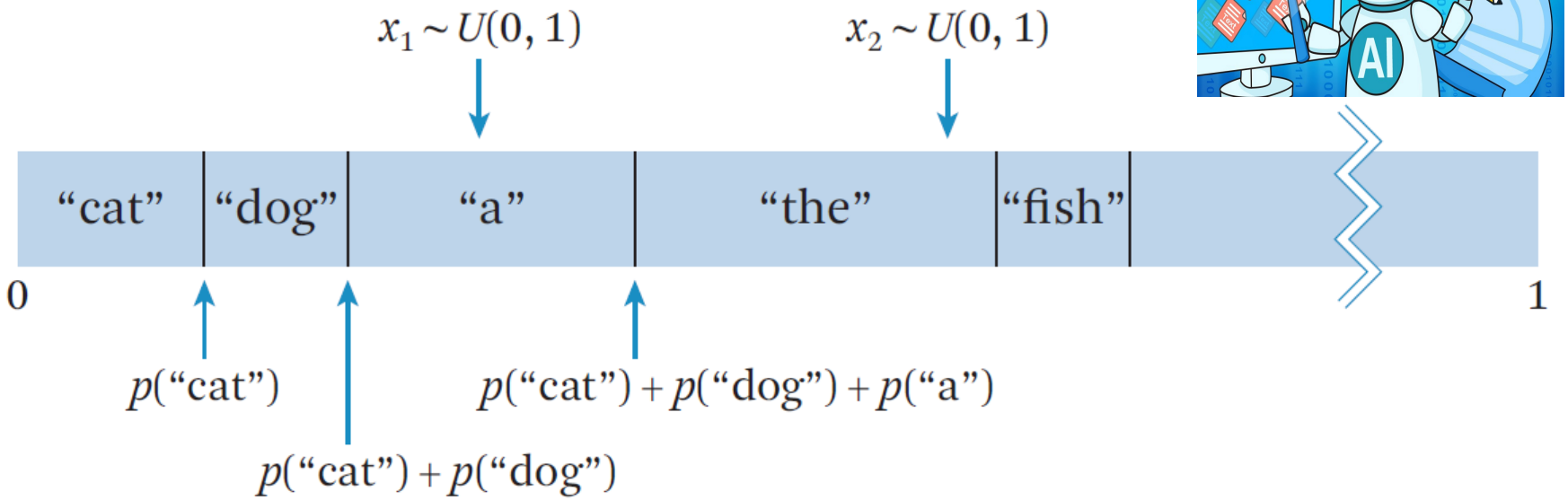
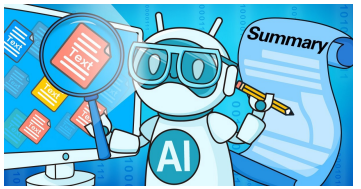


Figure 16.3 Drawing words from a unigram language model.

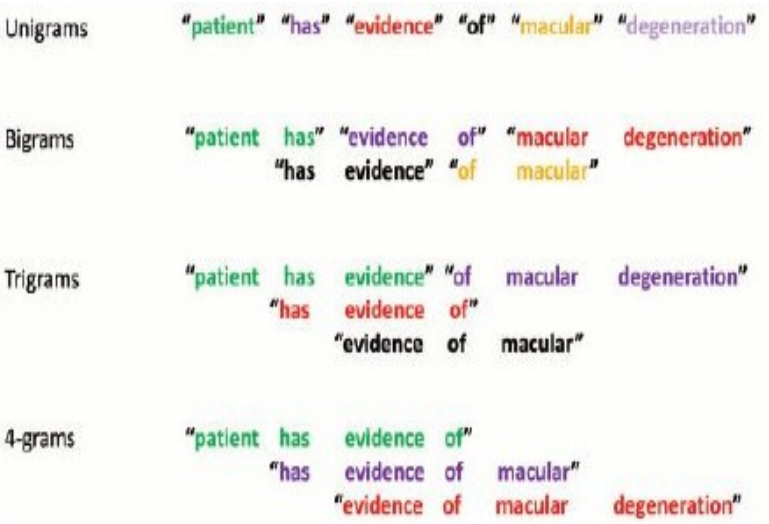
the text we generate will not make any sense: **each word is generated independently without regard to its context**

abstractive text summarization



if more fluent language is required, we can use an **n-gram language model**, where n>1

the new word depends on the previous n−1 words



say we have the word w_i and wish to generate w_{i+1}

with a **bigram language model**:

example:

start with (e.g.) The

then, pick from the distribution p(w | The)

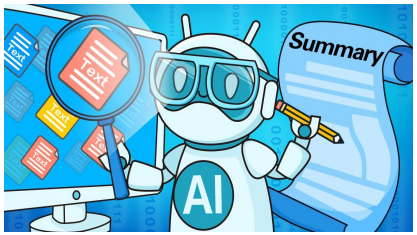
the next selected word could be cat.

then, we use the distribution p(w | cat) to find the next w and so on

we would like to choose an n-gram value that is large enough to produce coherent text

yet small enough to not simply reproduce the corpus

abstractive text summarization



with n-grams, still, there will be **no long-range dependencies** in our generated text

They imposed a gradual smoking ban on virtually all corn seeds planted are hybrids.

all groups of three words make sense, but as a whole the sentence is incomprehensible

evaluation of text summarization

extractive summarization chooses representative sentences

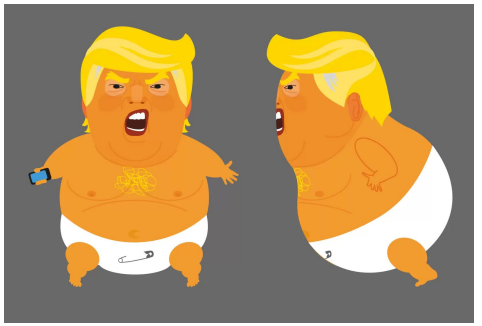
can be **modeled as an information retrieval problem**, and we can evaluate it as such (e.g. search for relevant and non-redudant sentences)

requires **labels** of relevance and redundancy at **sentence level**

evaluation of text summarization

alternatively (and useful for both extractive and abstractive): **human annotators create a gold standard summary**

this “perfect” summary would be compared with the generated one, and some measure (e.g., **ROUGE**) would be used to quantify the difference



evaluation of text summarization

human-produced summary vs automatic summary

we could use the cosine **similarity** between the gold standard and generated summary. This has the downside that fluency is completed ignored

Alternative: **learn an n-gram language model** over the gold standard summary, and then calculate the log-likelihood of the generated summary

ensures a basic level of fluency at the n-gram level, while also producing an interpretable result

evaluation of text summarization

user-centered evaluation: users read a summary and then answer questions about the original text.

was the summary able to **capture the important information** that the evaluator needs?

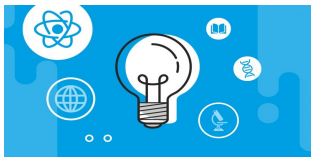
applications of text summarization

news articles, **retrieval** results, and **opinion** summarization

aspect opinion analysis: segments portions of user reviews about a particular topic

e-discovery (electronic discovery): finding relevant information in litigation (lawsuits and court cases).

summarize **research** from a given field



topic analysis

unsupervised text mining : e.g, probabilistic topic models that discover latent topics in text data

a topic is one idea discussed in text data (a theme or subject of a discussion or conversation)

granularities: the topic of a sentence, the topic of an article, the topic of a paragraph, or the topic of all the research articles in a library.

topic analysis: applications

what Twitter users are **talking about today**?

research (e.g. current advances in data mining)

opinion mining: discovering topics in both positive reviews and negative reviews

topic analysis

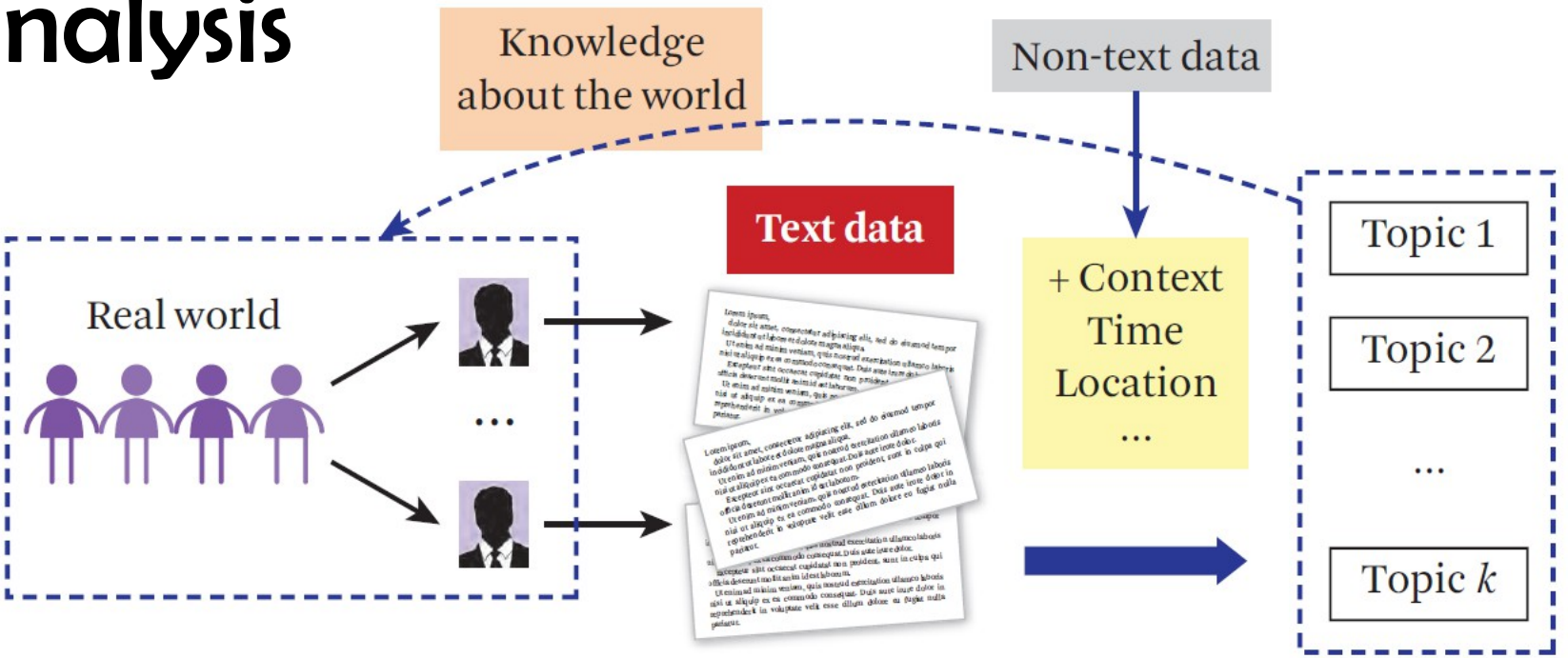


Figure 17.1 Mining topics as knowledge about the world.

looking at topics over **time**, we would be able to discover whether there's a trending topic or some topics might be fading away

looking at topics in different **locations** might help reveal insights about people's opinions indifferent locations

topic analysis

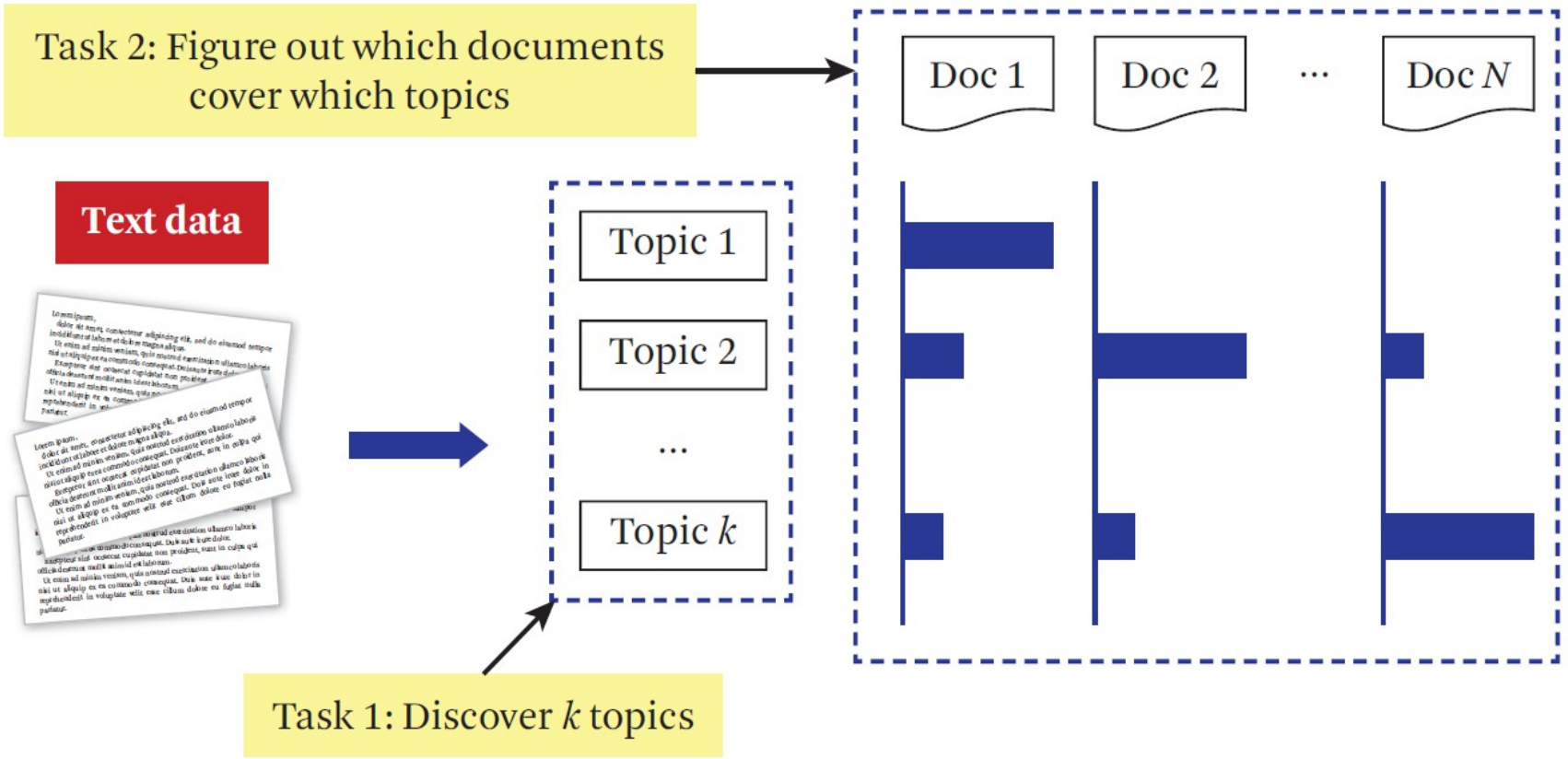


Figure 17.2 The task of topic mining.

we need to have as **input the number of topics**, k, though this number may be potentially set automatically based on data characteristics

topic analysis

- Input
  - A collection of N text documents $C = \{d_1, \ldots, d_N\}$
  - Number of topics: k
- Output
  - k topics: $\{\theta_1, \ldots, \theta_k\}$
  - Coverage of topics in each d_i: $\{\pi_{i1}, \ldots, \pi_{ik}\}$ $\quad \sum_{j=1}^{k} \pi_{ij} = 1$
  - π_{ij} = prob of d_i covering topic θ_j

How to define θ_j?

Figure 17.3 **Formal definition of topic mining tasks**

topic analysis: a term as a topic

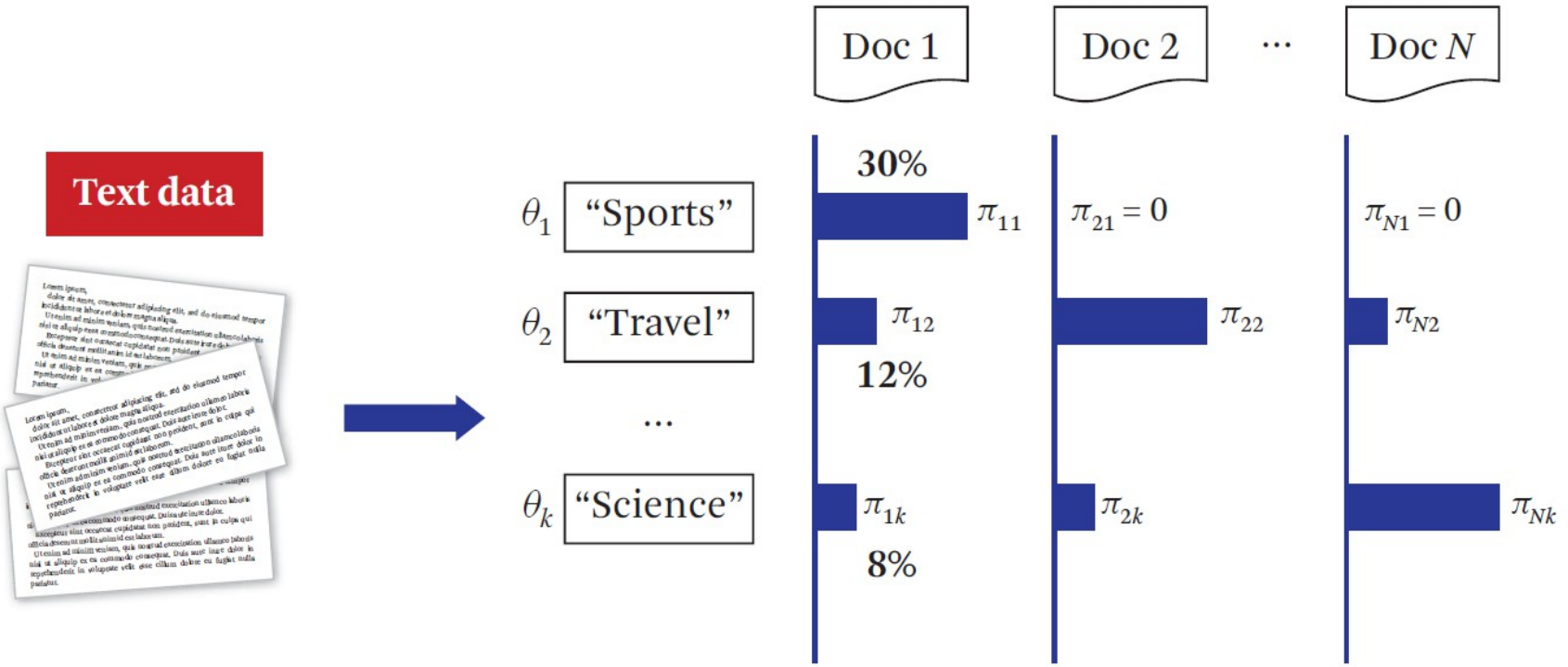


Figure 17.4 A term as a topic.

topic analysis: a term as a topic

we need to **mine k topical terms** from a collection

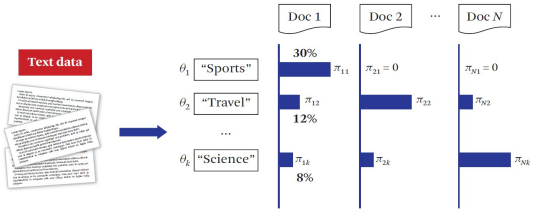


Figure 17.4 A term as a topic.

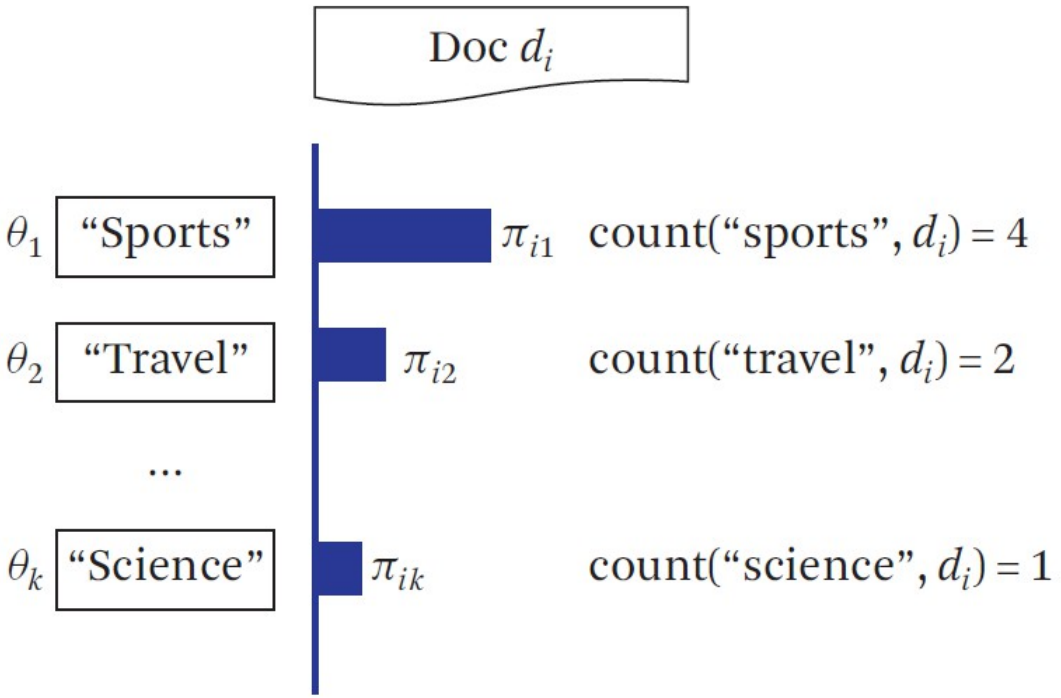
need to design a **scoring function** to quantify how good each term is as a topic

we would like to favor **representative terms** (e.g. using TF/IDF)

and use Maximal Marginal Relevance (MMR) to **avoid the selection of redundant topics** (e.g. synonyms)

how to compute the coverage of each topic in each document, πij ? one solution is to simply count occurrences of each topical term

topic analysis: a term as a topic



$$\pi_{ij} = \frac{\text{count}(\theta_j, d_i)}{\sum_{L=1}^{k} \text{count}(\theta_L, d_i)}$$

Figure 17.5 Computing topic coverage when a topic is a term.

topic analysis: a term as a topic

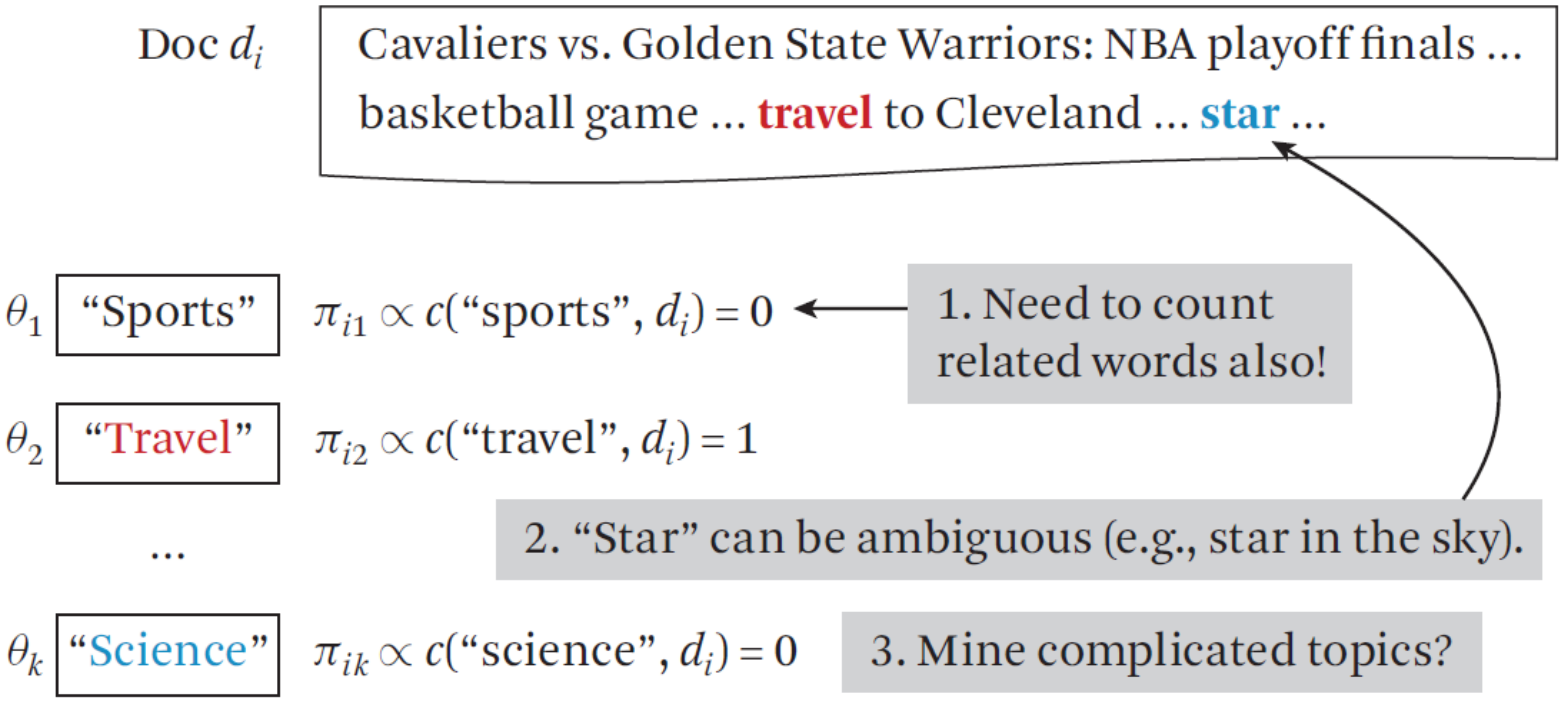


Figure 17.6 Problems in representing a topic as a term.

topic analysis: a term as a topic

there are **many words related to any given topic**, which should presumably also be counted when estimating the coverage of a topic.

cannot describe complicated topics by using just a word or one phrase (e.g. AI)

we need to use more words!

summing up, topics as single terms…

- it **lacks expressive power**
- it's **incomplete** in vocabulary **coverage**
- **ambiguity** of words

topic analysis: topics as word distributions

use **more words** to describe the topic

introduce **weights** on words

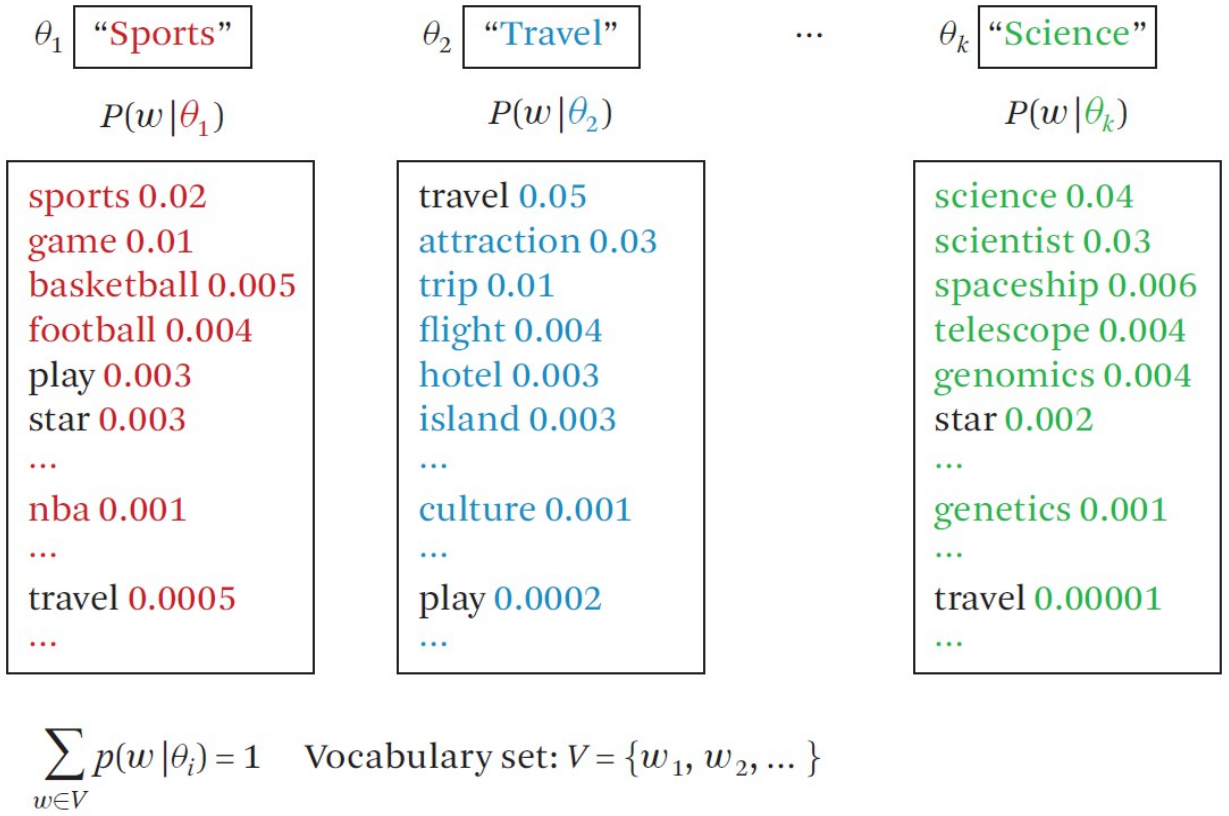


Figure 17.7 Topic as a word distribution.

topic analysis: topics as word distributions

some words that are shared by these topics

(non-zero probabilities for all these topics).

For example, travel occurs in the 3 lists

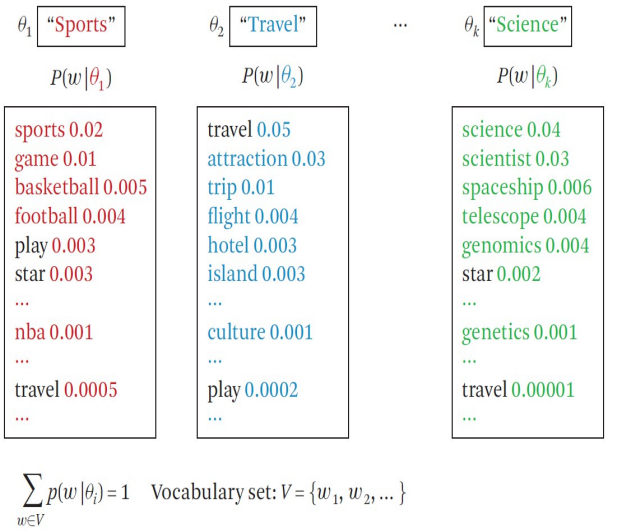


Figure 17.7 Topic as a word distribution.

multiple words to describe a topic

easily brings in **related words** together

accommodates multiple **senses** of a word, addressing the issue of word ambiguity.

topic analysis: topics as word distributions

basic example: first design a generative model for our data,

i.e., a probabilistic model to model how the data are generated

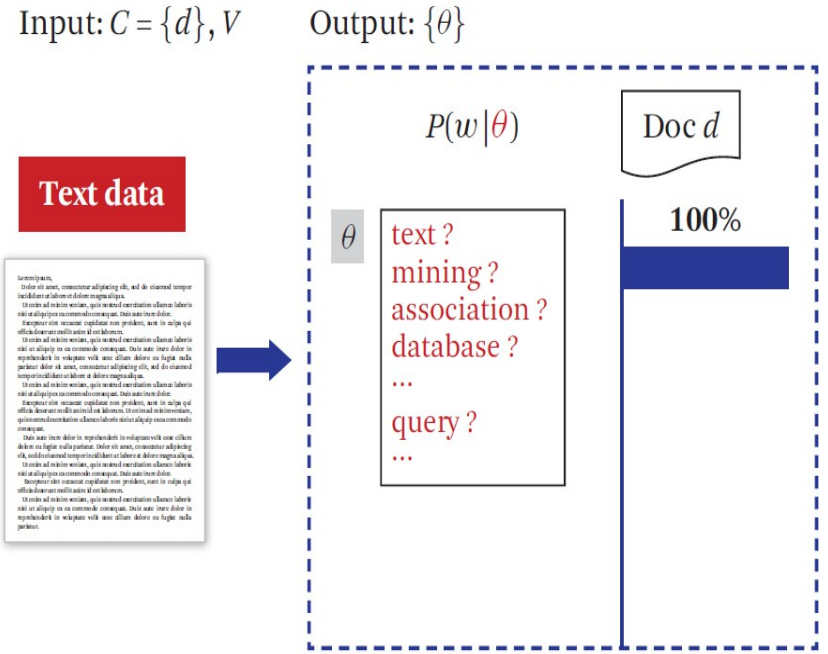
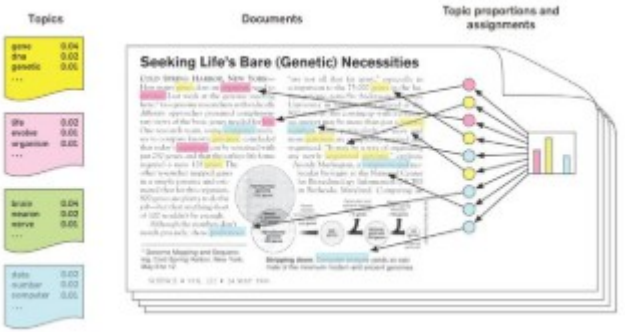


Figure 17.9 The simplest topic model with one topic.

topic analysis: topics as word distributions

more sophisticated methods:

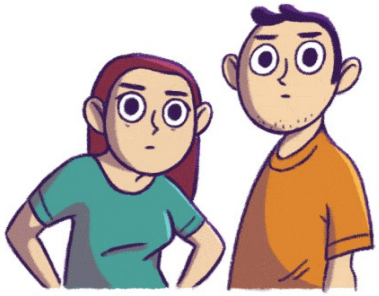
LDA, PLSA, use of **EM** algorithm…



evaluating topic analysis

not one true answer!

evaluation metrics heavily depend on the human issuing **judgements**



log-likelihood and model perplexity are two common evaluation measures (predictive measures):

held-out data is presented to the model and the model is applied to this new information, calculating its likelihood

if the model **generalizes well** to this new data (by assigning it a high likelihood or low perplexity), then the model is assumed to be sufficient.

downstream task improvement: effective (and transparent) evaluation metric. If a different topic analysis variant is shown to statistically significantly improve some task precision, then an argument may be made to prefer the new model. For example, if the topic analysis is meant to produce new features for text categorization, then classification accuracy is the metric we'd wish to improve.