Unstructured Data Mining

Unstructured Data Mining

- Unstructured data mining is the practice of looking at relatively unstructured data and trying to get more refined data sets out of it.
- It often consists of extracting data from sources not traditionally used for data mining activities.

Data Mining and Text Mining

BASE FOR COMPARISION	Data Mining	Text Mining
Concept	Data mining is a spectrum of different approaches, which searches for patterns and relationships of data.	Text mining is a process required to turn unstructured text document into valuable structured information.
Retrieval of data	With standard data mining techniques reveals business patterns in numerical data.	With standard text mining methods discovers a lexical & syntactic feature in the text.
Type of Data	Discovery of knowledge from structured data, which are homogeneous and easy to access.	Discovery of text from unstructured data which are heterogeneous, more diverse.

Text Mining

- **Text mining**, also referred to as **text data mining**, roughly equivalent to **text analytics**, is the process of deriving high-quality information from text.
- Typical text mining tasks include
  - Text categorization,
  - Text clustering,
  - Concept/entity extraction,
  - Production of granular taxonomies,
  - Sentiment analysis,
  - Document summarization, and
  - Entity relation modeling (i.e., learning relations between named entities).

--- Wikipedia

Text mining

- Text mining is the process of exploring and analyzing large amounts of **unstructured text data** aided by software that can identify concepts, patterns, topics, keywords and other attributes in the data.
- It's also known as **text analytics**.
- Text mining has become more practical for data scientists and other users due to the development of **big data platforms** and **deep learning algorithms** that can analyze massive sets of unstructured data.

Text mining

Text mining applications today draw on a wide range of techniques and serve many purposes in information management and business intelligence.

TM techniques can be organized into four categories:

1) Classification techniques

3) Association analysis

5) Information extraction techniques

7) Clustering techniques

The Document Collection and the Document

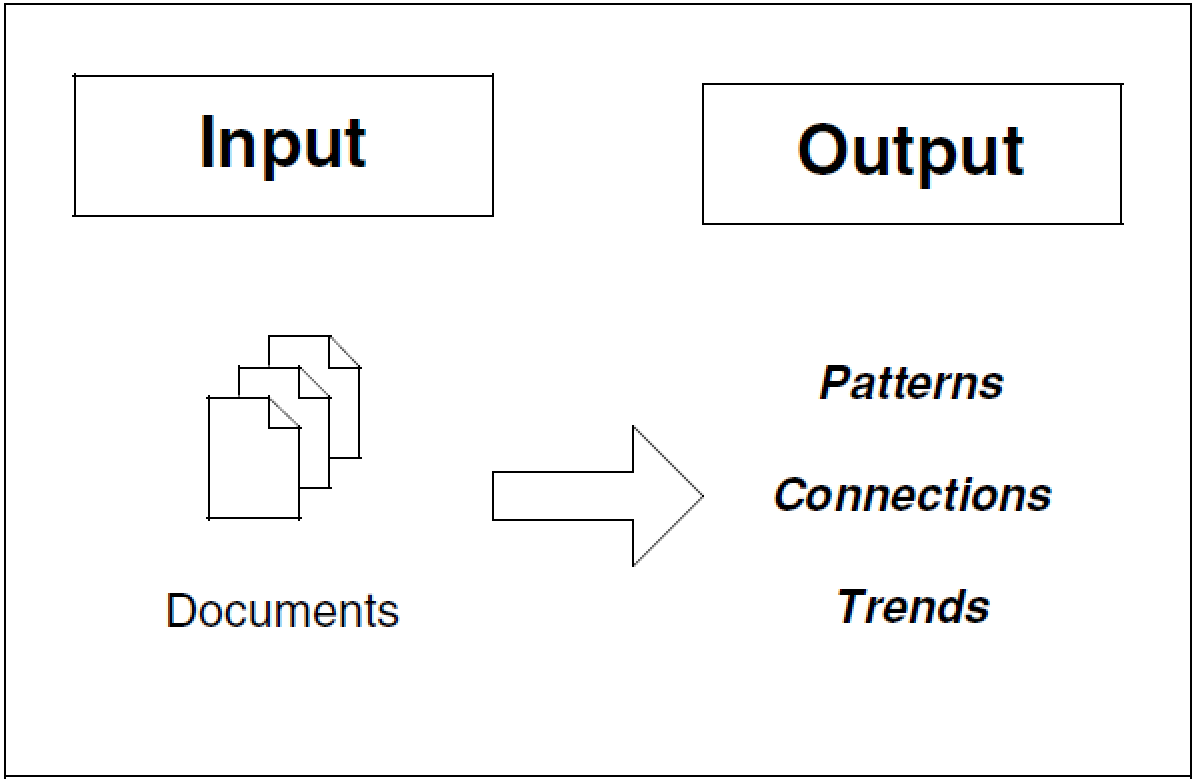
- A key element of text mining is its focus on the *document collection*.
- At its simplest, a document collection can be any grouping of text-based documents.
- Another basic element in text mining is the document.
- Document is a unit of discrete textual data within a collection that usually, but not necessarily, correlates with some real-world document such as a business report, legal memorandum, e-mail, research paper, manuscript, article, press release, or news story.

Document Features

- **Commonly Used Document Features:**
  - Characters,
  - Words,
  - Terms, and
  - Concepts

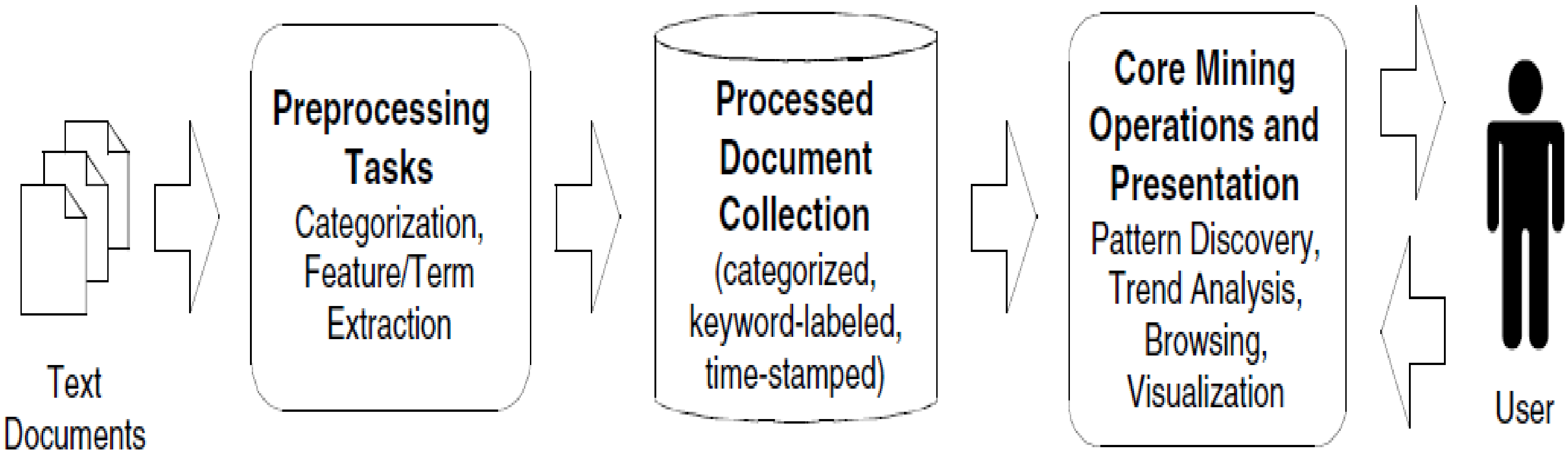
- **Set of features as the representational model of a document.**

Functional Architecture



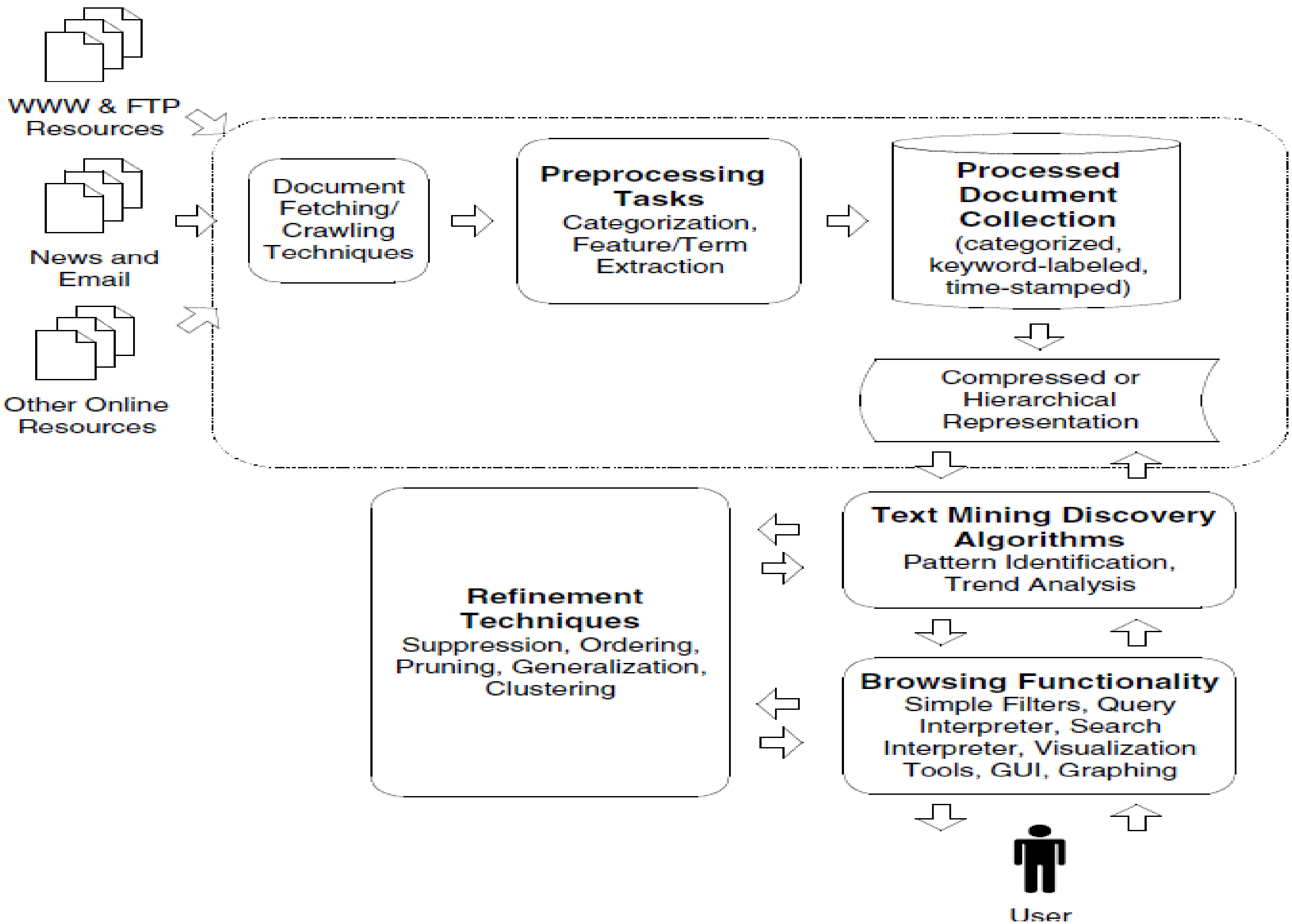
Simple input–output model for text mining.

High-level text mining functional architecture.



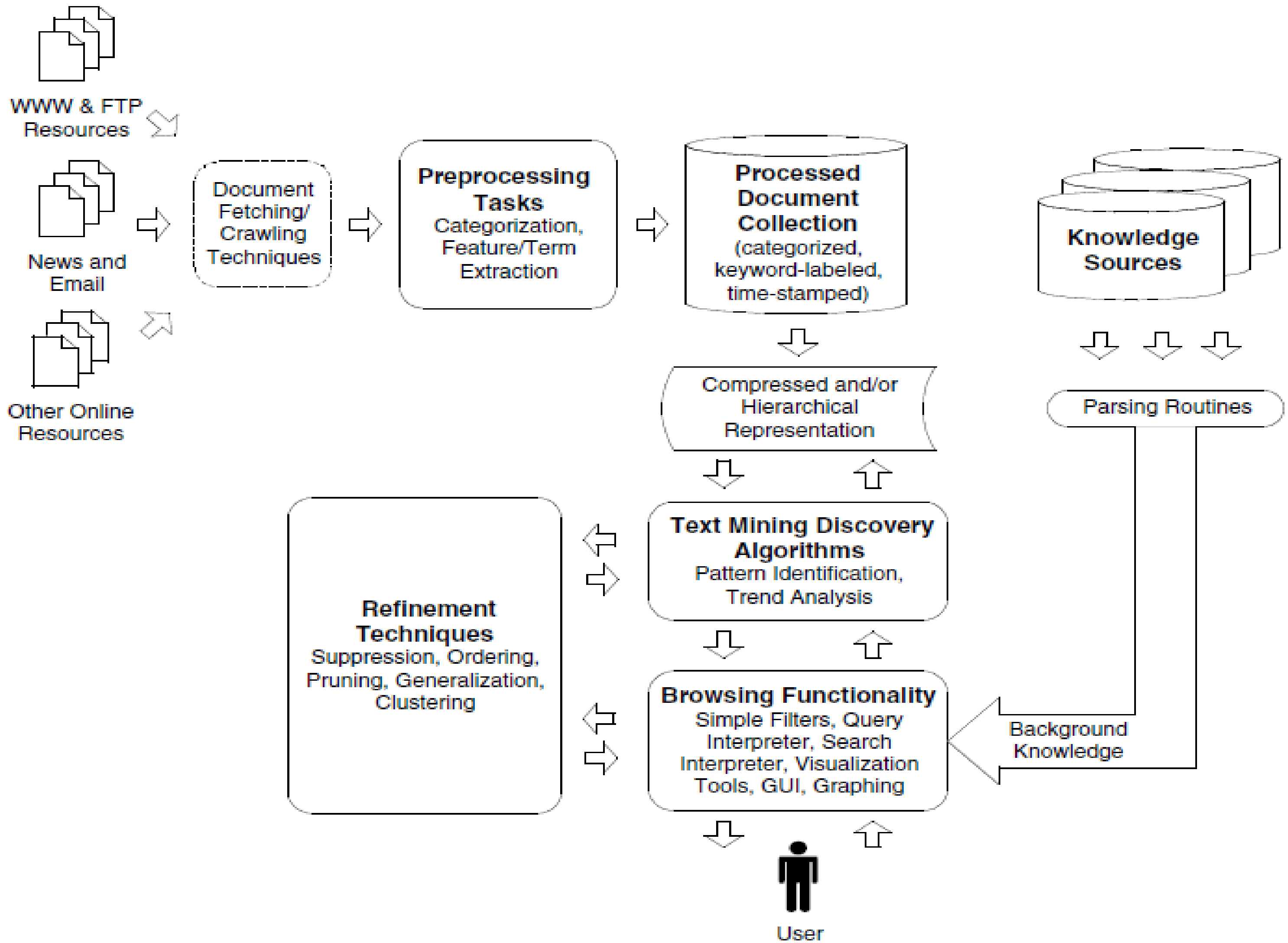
High-level text mining functional architecture.

System architecture for generic text mining system



System architecture for generic text mining system.

System architecture for an advanced or domain-oriented text mining system



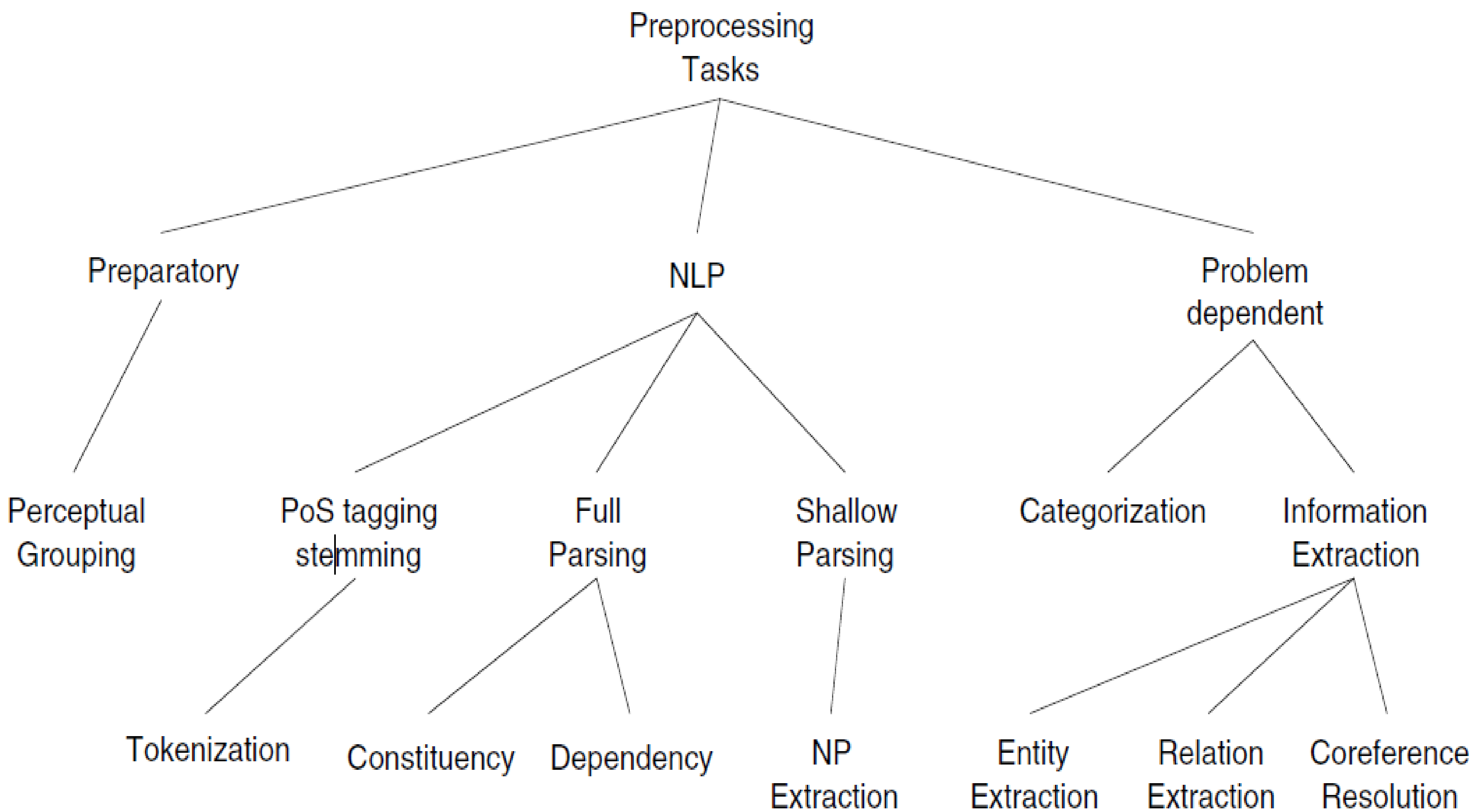
System architecture for an advanced or domain-oriented text mining system.

Text Mining

Text mining systems follow the general model divisible into four main areas:

(a) Preprocessing tasks,
(b) Core mining operations,
(c) Presentation layer components and browsing functionality, and
(d) Refinement techniques.

A taxonomy of text preprocessing tasks



A taxonomy of text preprocessing tasks.

CORE TEXT MINING OPERATIONS

CORE TEXT MINING OPERATIONS:

- Distributions
- Frequent and Near Frequent Sets
- Associations
- Isolating Interesting Patterns

Text Mining Tools

Software	Features
RapidMiner	Unified platform for data prep, machine learning, and model deployment.
Gensim	Extracts semantics information from unstructured data and topic modelling on large scale.
Natural language toolkit (NLTK)	Provides programs and libraries for statistical and symbolic natural language processing using Python.
Orange	Provides an add-on of text mining.
AlchemyLanguage	Entity extraction, keyword extraction, emotion and sentiment analysis.
OpenNLP	Processing of natural language.
SAS text miner	Automatic Boolean rule generator, theme discovery, term profiling.
Stanbol	Semantic content management, mainly used in scholarly projects, has a web based text mining environment.
WordStat	faster extraction of themes and trend, efficient analysis of qualitative content.
Stanford CoreNLP	Text Mining Library

Text Categorization

- *Text Categorization* (TC) – given a set of categories (subjects, topics) and a collection of text documents, the process of finding the correct topic (or topics) for each document.

  - **Spam**
    - “spam” / “not spam”
  - **Topics**
    - “finance” / “sports” / “asia”
  - **Author**
    - “Shakespeare” / “Marlowe” / “Ben Jonson”
    - The Federalist papers author
    - Male/female
    - Native language: English/Chinese,...
  - **Opinion**
    - “like” / “hate” / “neutral”
  - **Emotion**
    - “angry”/”sad”/”happy”/”disgusted”/...

Text Categorization Approaches

- Two main approaches to text categorization
  1. Knowledge Engineering Approach
  2. Machine Learning Approach

APPLICATIONS OF TEXT CATEGORIZATION

- Indexing of Texts Using Controlled Vocabulary
- Document Sorting and Text Filtering
- Hierarchical Web Page Categorization

Indexing of Texts Using Controlled Vocabulary

- Early research in the TC field is text indexing.
- Information retrieval (IR) systems - each document is assigned one or more key terms describing its content.
- IR system is able to retrieve the documents according to the user queries
- The key terms all belong to a finite set called *controlled vocabulary*
- The task of assigning keywords from a controlled vocabulary to text documents is called *text indexing*.
- If keywords are viewed as categories, then text indexing is an instance of general *TC* problem
- Typically, each document should receive at least one, and not more than *k*, keywords.
- Automatic indexing can be a part of *automated extraction of metadata*
- Extraction of this metadata can be viewed as a document indexing problem, which can be tackled by TC techniques.

Document Sorting and Text Filtering

- Another common problem - sorting the given collection of documents into “bins.”
- E, g. newspaper, classified ads categorized into “Personal,” “Car Sale,” “Real Estate,” and so on.
- E-mail classified into categories such as “Complaints,” “Deals,” “Job applications,” and others.
- Document sorting features - each document belong to exactly one category.
- Text filtering activity can be seen as document sorting with only two bins – the “relevant” and “irrelevant” documents.
- Examples
  - ✓ A sports related online magazine filter out all nonsport stories it receives from the news feed.
  - ✓ An e-mail client should filter away spam.
  - ✓ A personalized ad filtering system should block any ads that are uninteresting to the particular user.

Hierarchical Web Page Categorization

- A common use of TC is the automatic classification of Web pages under the hierarchical catalogues.
- Useful for direct browsing and for restricting the query-based search to pages belonging to a **particular topic**.
- Can constrain the number of categories to which a document may belong.
- Hierarchical Web page categorization constrains number of documents belonging to a particular category.
- Whenever number of documents in a category exceeds *k*, it should be split into two or more subcategories.
- Supports adding new categories and deleting obsolete ones.
- Hypertextual nature of the documents - contain links, which may be important sources of information for classifier because linked documents often share semantics.

TC - DEFINITION OF THE PROBLEM

- The general text categorization task can be formally defined as the task of approximating an unknown category assignment function $F : D \times C \rightarrow \{0, 1\}$, where D is the set of all possible documents and C is the set of predefined categories.
- The value of F(d, c) is 1 if the document d belongs to the category c and 0 otherwise.
- Function $M : D \times C \rightarrow \{0, 1\}$ is called **a classifier**
- Need to build a classifier that produces results as "close" as possible to the true category assignment function F.

Single-Label versus Multilabel Categorization

- Depending on the properties of F, we can distinguish between single-label and multilabel categorization.
- In multilabel categorization - document may belong to any number of categories.
- In single-label categorization, each document belongs to exactly one category.
- Binary categorization sp. single-label categorization number of categories is two.
- Single label case is a simple generalization of the binary case.
- The multilabel case can be solved by $|C|$ binary classifiers, one for each category.

Document-Pivoted versus Category Pivoted Categorization

- Given a document, the classifier finds all categories to which the document belongs. This is called a document-pivoted categorization.
- Alternatively, find all documents that should be filed under a given category, called a category-pivoted categorization.
- Difference - case in which not all documents or not all categories are immediately available.
- e.g. "online" categorization - only the document-pivoted categorization is possible
- If the categories set is not fixed, and if the documents need to be reclassified with respect to the newly appearing categories, then category-pivoted categorization is appropriate.

Hard versus Soft Categorization

- A fully automated categorization system makes a binary decision on each document category pair. Such a system is said to be doing the *hard* categorization.
- In semiautomated approach decision to assign a document to a category is made by a human and TC system provides a list of categories - soft or ranking categorization
- Many classifiers produce a real value between zero and one for each document category pair. This value is called a **Categorization Status Value (CSV).**
- Binary decision can be done by checking the CSV against a specific threshold.
- Various possible policies exist for setting the threshold.

DOCUMENT REPRESENTATION

- The common classifiers and learning algorithms **cannot directly process the text documents in their original form**.
- Documents are converted into manageable representation.
- Document's are represented by **feature vectors**.
- A **feature** is simply an **entity** without internal structure – **dimension** in the feature space.
- A document is represented as a vector in this space – a sequence of features and their weights.

DOCUMENT REPRESENTATION

- Most common bag-of-words model - uses all words in a document as features.
- Dimension of the feature space is equal to the number of different words in all of the documents.

DOCUMENT REPRESENTATION

- The methods of giving weights to the features may vary.
- The simplest is the binary in which the feature weight is either zero OR one.
- More complex weighting schemes are possible that take into account the frequencies of the word in the document, in the category, and in the whole collection.
- The most common **TF-IDF scheme** gives the word w in the document d the weight

$$TF\text{-}IDF\_Weight\,(w, d) = TermFreq(w, d) \cdot \log\,(N\,/\,DocFreq(w)),$$

where TermFreq(w, d) is the frequency of the word in the document,
N is the number of all documents, and
DocFreq(w) is number of documents containing word w.

DOCUMENT REPRESENTATION

Numerically, term frequency of a word is defined as follows:

> tf(w) = doc.count(w)/total words in corpus

> idf(w) = log(total number of documents/number of documents containing word w)

Toy corpus and desired behavior

Let's take an example of a corpus consisting of following 5 documents:

1. This car got the excellence award
2. Good car gives good mileage
3. This car is very expensive
4. This company is financially good
5. The company is growing with very high production

Tf-idf(car) for D1 = 1/16 * log(5/3) = 0.0625 * 0.2218 = 0.01386 - Normalized TF
Tf-idf(car) for D1 = 1 * log(5/3) = 0.4771 - Non-Normalized TF

Assume total Number of Words in Corpus = 16

DOCUMENT REPRESENTATION

- **Let's another take an example to get a clearer understanding.**

  Sentence 1 : The car is driven on the road.

  Sentence 2 : The truck is driven on the highway.

- In this example, each sentence is a separate document.

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

	The	car	driven	highway	is	on	road	the	truck
0	0.0	0.043004	0.0	0.000000	0.0	0.0	0.043004	0.0	0.000000
1	0.0	0.000000	0.0	0.043004	0.0	0.0	0.000000	0.0	0.043004

Feature Selection

- Number of different words is **large** even in **relatively small documents**.
- In big document collections can be huge.
- Dimension of the bag-of-words feature space can reach hundreds of thousands.
- Document representation vectors are sparse.
- Most of words are irrelevant to categorization task - can be dropped.
- **The preprocessing step that removes the irrelevant words is called feature selection.**
- Most TC systems at least remove the stop words - common words that do not contribute to the semantics of the documents.
- Many systems perform more aggressive filtering, removing 90 to 99 percent of features.

Feature Selection

- To perform the filtering, a measure of the relevance of each feature needs to be defined.
- Probably the simplest such measure is the document frequency DocFreq(w).
- Experimental evidence - top 10% of the most frequent words does not reduce the performance of classifiers.
- May Contradict well-known “law” of IR - terms with low-to-medium document frequency are the most informative.
- No contradiction, large majority of words have a very low document frequency.
- More sophisticated measures of feature relevance - account the relations between features and the categories.

Feature Selection

- The *information gain*

$$IG(w) = \sum_{c \in C \cup \overline{C}} \sum_{f \in \{w, \overline{w}\}} P(f, c) \cdot \log \frac{P(c \mid f)}{P(c)}$$

  - Measures the number of bits of information obtained for the prediction of categories by knowing the presence or absence in a document of the feature f.
  - The probabilities are computed as ratios of frequencies in the training data.

- Another good measure is the chi-square

$$\chi^2_{\max}(f) = \max_{c \in C} \frac{|Tr| \cdot (P(f, c) \cdot P(\bar{f}, \bar{c}) - P(f, \bar{c}) \cdot P(\bar{f}, c))^2}{P(f) \cdot P(\bar{f}) \cdot P(c) \cdot P(\bar{c})},$$

which measures the maximal strength of dependence between the feature and the categories.

Dimensionality Reduction by Feature Extraction

- Another way is to create a new, much smaller set of synthetic features from the original feature set.
- **Rational** - owing to polysemy, homonymy, and synonymy, the words may not be the optimal features.
- **Term clustering** addresses the problem of synonymy by grouping together words with a high degree of semantic relatedness.
- Experiments conducted by several groups of researchers showed a potential in this technique when background information about categories was used for clustering.

Dimensionality Reduction by Feature Extraction

- A more systematic approach is **Latent Semantic Indexing (LSI)**.
- Several LSI representations, one for each category, outperform a single global LSI representation.
- LSI usually performs better than the chi-square filtering scheme.

KNOWLEDGE ENGINEERING APPROACH TO TC

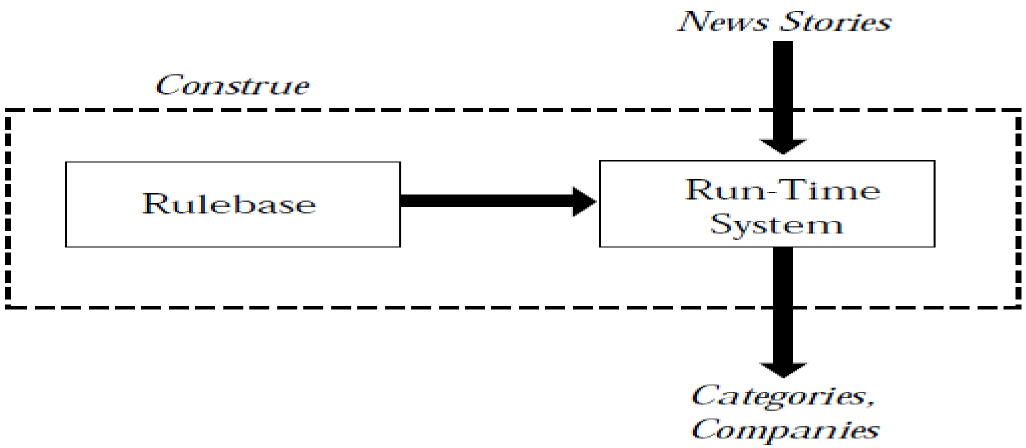
- Focused around manual development of classification rules
- A domain expert defines a set of sufficient conditions for a document to be labelled with a given category.
- Approach to the TC - CONSTRUE system – developed by Carnegie group for Reuters.
- A typical rule in the CONSTRUE system is as follows

  if DNF (disjunction of conjunctive clauses) formula ***then*** category ***else*** ¬category
- Such rule may look like the following:

  If ((wheat & farm) or
  (wheat & commodity) or
  (bushels & export) or
  (wheat & tonnes) or
  (wheat & winter & ¬soft))
  then Wheat
  else ¬Wheat
- 90-percent breakeven between precision and recall on a small subset of the Reuters collection (723 documents).
- Even accuracy is GOOD, more efforts and req. domain experts to generate rules, **ML approach with little less accuracy is better**.

KNOWLEDGE ENGINEERING APPROACH TO TC

CONSTRUE System - news story categorization system



Specific goals :

• Accept Reuters news stories, including economic, financial, and general news

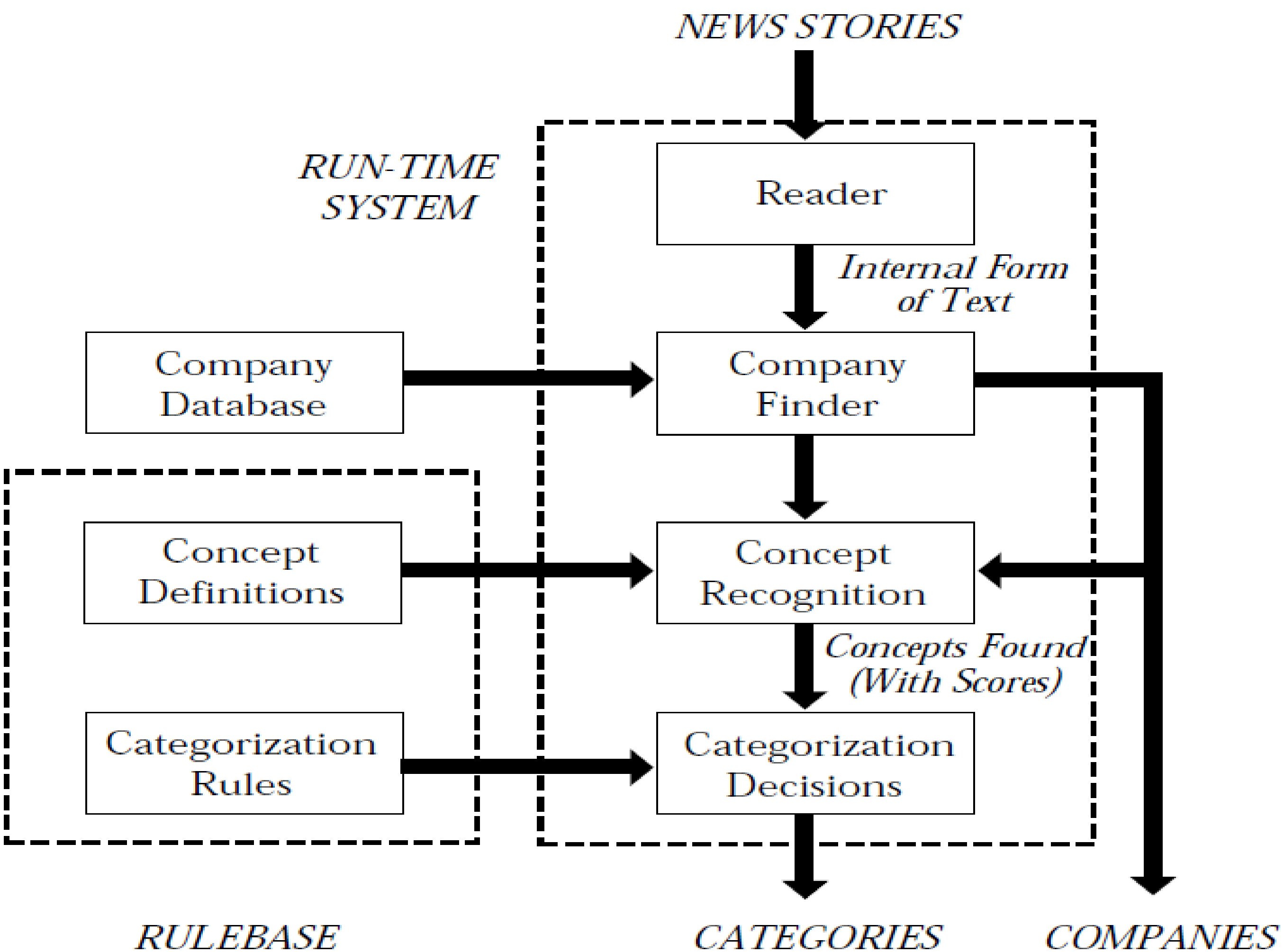
• Categorize each story into zero, one, or several of 150 distinct categories (674 distinct categories were delivered)

• Recognize mentions of companies from a database of 10,000 company names (over 17,000 names were delivered).

• Process stories in an average of five seconds (delivered with an average of 4.36 seconds)

• Achieve an average accuracy level of 85 percent (89-percent accuracy level was delivered)

Construe's Flow of Control.



MACHINE LEARNING APPROACH TO TC

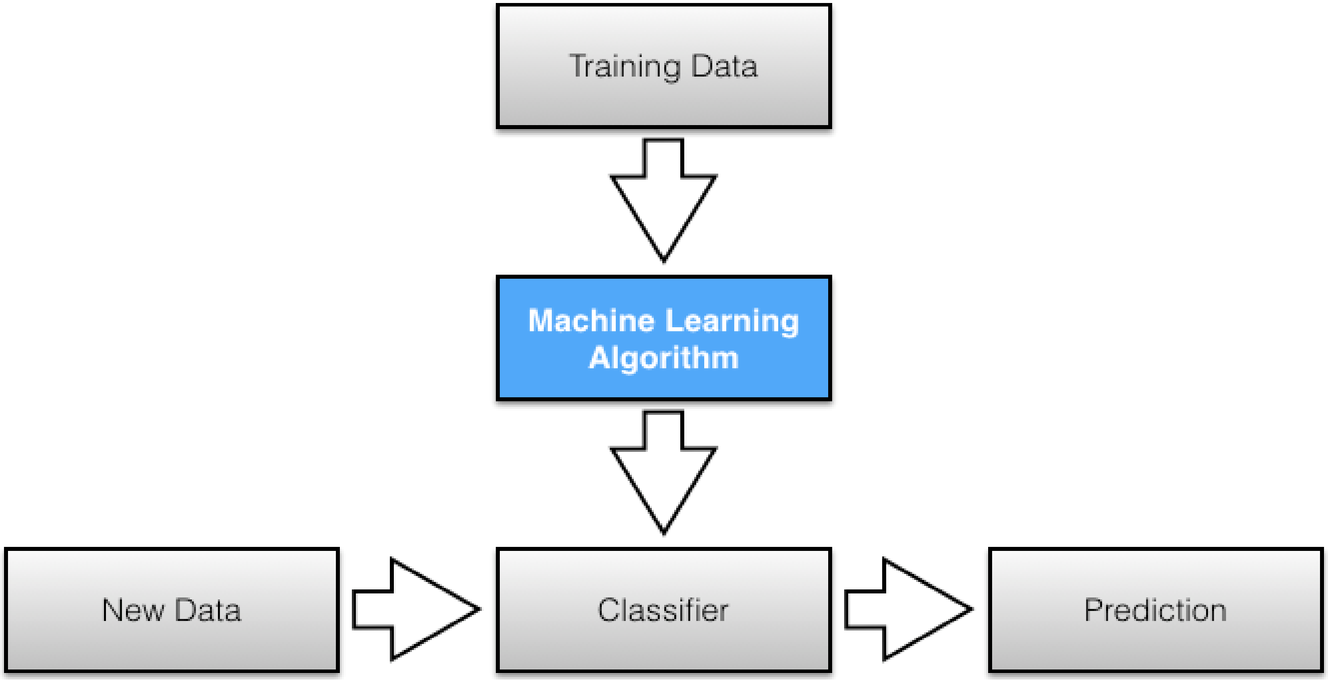
- **Classifier** is built automatically by learning the properties of categories from a set of **pre-classified training documents.**
- Learning process is an instance of supervised learning.
- The unsupervised version of the classification task, called clustering
- **Humans learn from past experiences, machines learn from past instances.**
- **Many approaches to classifier learning:**
  - Variants of general ML algorithms
  - Created specifically for categorization

MACHINE LEARNING APPROACH TO TC

Four **main issues** need to be considered

1. decide on the categories that will be used to classify the instances.
2. Provide a training set for each of the categories (30 examples are needed for each category).
3. Decide on the features that represent each of the instances
4. Decide on the algorithm to be used for the categorization

Probabilistic Classifiers - Naıve Bayes



- Bayes' theorem forms the core of the whole concept of naive Bayes classification.
- Samples are dependent and identically distributed.
  (e.g. first coin flip does not affect outcome of a second coin flip.)

Bayesian Methods -Naıve Bayes

- Use Bayes theorem to build a generative model that approximates how data are produced.
- Use prior probability of each category.
- Produce a posterior probability distribution over the possible categories given a description of an item.

Probabilistic Classifiers - Naıve Bayes

- Probabilistic classifiers view the categorization status value CSV(*d*, *c*) as the probability *P*(*c* | *d*)
- The document *d* belongs to the category *c* and compute this probability by an application of Bayes' theorem:

$$P(c \mid d) = \frac{P(d \mid c) P(c)}{P(d)}.$$

- The marginal probability P(d) need not ever be computed because it is constant for all categories.
- To calculate P(d | c), - assumptions about the structure of the document d.
  - ✓ document representation as a feature vector d = (w_1, w_2, . ..),
  - ✓ all coordinates are independent, and thus

$$P(d \mid c) = \prod_i P(w_i \mid c).$$

Probabilistic Classifiers - Naıve Bayes

- The classifiers resulting from this assumption are called Naıve Bayes (NB) classifiers.
- They are called “naive” because the assumption is never verified and often is quite obviously false.
- Attempts to relax the naive assumption and to use the probabilistic models with dependence so far have **not produced** any significant improvement in performance.

Naıve Bayes – Practical Example

Our training data has 5 sentences:

Text	Tag/Category
“A great game”	Sports
“The election was over”	Not sports
“Very clean match”	Sports
“A clean but forgettable game”	Sports
“It was a close election”	Not sports

- Now, which tag does the sentence ”A very close game” belong to?
- Since Naive Bayes is a probabilistic classifier,
  - ✓ Calculate the probability that the sentence “A very close game” is Sports, and
  - ✓ probability that it’s *Not Sports*.

In our case, we have , so using this theorem

Naıve Bayes – Practical Example

Feature engineering

- We use **word frequencies**
- Ignore word order and sentence construction

- Bayes' Theorem is useful

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

- In our case P(Sports| A very close game)

$$P(sports|a\,very\,close\,game) = \frac{P(a\,very\,close\,game|sports) \times P(sports)}{P(a\,very\,close\,game)}$$

- Can discard divisor and compare P(A very close game | Sports) x P(Sports) and P(A very close game | Not Sports) x P(Not Sports)

Naıve Bayes – Practical Example

- **Problem :** "**A very close game**" doesn't appear in our training data, so this probability is zero.
- Being Naive
  - every word in a sentence is **independent** of the other ones
  - "this was a fun party" is the same as "this party was fun" and "party fun was this"
- We write this as:

  P(A very close game) = P(A) x P(very) x P(close) x P(game)

  P(A very close game | Sports) = P(A | Sports) x P(very | Sports) x P(close | Sports) x P(game | Sports)
- Calculating a probability is just counting in our training data.

P(Sports) = 3/5

P(Not Sports) = 2/5

P(game| Sports) = 2/11

(Calculating means counting how many times the word **"game"** appears in Sports texts (2) divided by total no. of words in sports (11))

Text	Tag/Category
"A great game"	Sports
"The election was over"	Not sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not sports

However, we run into a problem here: "close" doesn't appear in any Sports text!

Naıve Bayes – Practical Example

- Using something called **Laplace smoothing**: we add 1 to every count so it's never zero and divide by number of possible words (14)
- Applying smoothing we get

$$P(game|sports) = \frac{2+1}{11+14}.$$

Text	Tag/Category
"A great game"	Sports
"The election was over"	Not sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not sports

Word	P(word | Sports)	P(word | Not Sports)
a	$\frac{2+1}{11+14}$	$\frac{1+1}{9+14}$
very	$\frac{1+1}{11+14}$	$\frac{0+1}{9+14}$
close	$\frac{0+1}{11+14}$	$\frac{1+1}{9+14}$
game	$\frac{2+1}{11+14}$	$\frac{0+1}{9+14}$

Naıve Bayes – Practical Example

Now we just multiply all the probabilities, and see who is bigger:

$$P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times P(Sports)$$
$$= 2.76 \times 10^{-5}$$
$$= 0.0000276$$

$$\mathrm{P(a} \quad |Not\,Sports) \times P(very|Not\,Sports) \times P(close|Not\,Sports) \times P(game|Not\,Sports) \times P(Not\,Sports)$$
$$= 0.572 \times 10^{-5}$$
$$= 0.00000572$$

Excellent! Our classifier gives "A very close game" the **Sports** tag.

Naıve Bayes – Practical Example - Enhancements

- Removing stop words.
- Lemmatizing words.
- Using n-grams - could count sequences of words
- **Using TF-IDF**.

Logistic Regression

- In statistics, the **logistic model** is a widely used statistical model that, in its basic form, uses a logistic function to model a binary dependent variable.
- Logistic regression was developed by statistician David Cox in 1958.
- In **logistic regression**, the outcome (dependent variable) has only a limited number of possible values.
- Logistic regression is used when the response variable is categorical.

LINEAR REGRESSION AND LOGISTIC REGRESSION

BASIS FOR COMPARISON	LINEAR REGRESSION	LOGISTIC REGRESSION
Basic	The data is modelled using a straight line.	The probability of some obtained event is represented as a linear function of a combination of predictor variables.
Linear relationship between dependent and independent variables	Is required	Not required
The independent variable	Could be correlated with each other. (Specially in multiple linear regression)	Should not be correlated with each other (no multicollinearity exist).

Key Differences Between Linear and Logistic Regression

- The Linear regression models data using continuous numeric value. As against, logistic regression models the data in the binary values.
- Linear regression requires to establish the linear relationship among dependent and independent variable whereas it is not necessary for logistic regression.
- In the linear regression, the independent variable can be correlated with each other. On the contrary, in the logistic regression, the variable must not be correlated with each other.

TC With Logistic Regression challenges

- Documents to be classified are typically represented as **vectors of numeric feature** values derived from
  - words,
  - phrases, or
  - other characteristics of documents
- TC applications treat documents as atomic units converted to feature vectors.
- The dimensionality of these vectors ranges from 10^3 to 10^6 or more.
- Major **challenges** to apply Logistic Regression to TC was
  - avoiding overfitting (make accurate predictions for future input)
  - reducing memory and computational requirements
- **Bayesian approach to logistic regression avoids overfitting.**

Bayesian Logistic Regression Cont…

- It is possible to model the conditional probability $P(c|d)$ directly.
- Bayesian Logistic Regression (BLR) is an old statistical approach that was only recently applied to the TC problem.
- Quickly gaining popularity owing to its apparently very high performance.
- Assuming categorization is binary Logistic Regression model has form

$$P(c\,|\,\mathbf{d}) = \varphi(\boldsymbol{\beta} \cdot \mathbf{d}) = \varphi\left(\sum_i \beta_i d_i\right)$$

where $c = \pm 1$ 1 is used instead of {0, 1})

d = (d1, d2, . . .) - document representation in the feature space

β = (β1, β2, . . .) - model parameters vector

ψ is the ***logistic link* function**

$$\varphi(x) = \frac{\exp(x)}{1+\exp(x)} = \frac{1}{1+\exp(-x)}$$

- **Bayesian approach to logistic regression avoids overfitting.**

Bayesian Logistic Regression Cont…

- The **Bayesian approach** - use a prior distribution for β - assigns a high probability to each βi's
- The simplest is the Gaussian prior with zero mean and variance τ :

$$p(\beta_i \mid \tau) = N(0, \tau) = \frac{1}{\sqrt{2\pi\tau}} \exp\left(\frac{-\beta_i^2}{2\tau}\right)$$

- The alt… seness:
- Model will not be sparse

$$p(\beta_i \mid \lambda) = \frac{\lambda}{2} \exp(-\lambda \mid \beta_i \mid)$$

- Using this kind … ef that a small portion of the input variables has a substantial effect on the outcome, whereas **other** variables are **unimportant**.

Bayesian Logistic Regression Cont…

- Owing to computational cost constraints, however, it is common to use a point estimate of β, of which the posterior mode (any value of β at which the posterior distribution of β takes on its maximal value) is the most common.
- The log-posterior distribution of β is

$$l(\boldsymbol{\beta}) = p(\boldsymbol{\beta} \mid D) = -\left(\sum_{(\mathbf{d},c)\in D} \ln(\exp(-c\,\boldsymbol{\beta} \cdot \mathbf{d}) + 1) \right) + \ln p(\boldsymbol{\beta})$$

where D= {(d1, c1), (d2, c2) . . .} is the set of training documents di and their true category membership values ci = ±1, and p(β) is the chosen prior:

$$\ln p(\boldsymbol{\beta}) = -\left(\sum_i \left(\ln\sqrt{\tau} + \frac{\ln 2\pi}{2} + \frac{\beta_i^2}{\tau} \right) \right), \text{ for Gaussian prior, and}$$

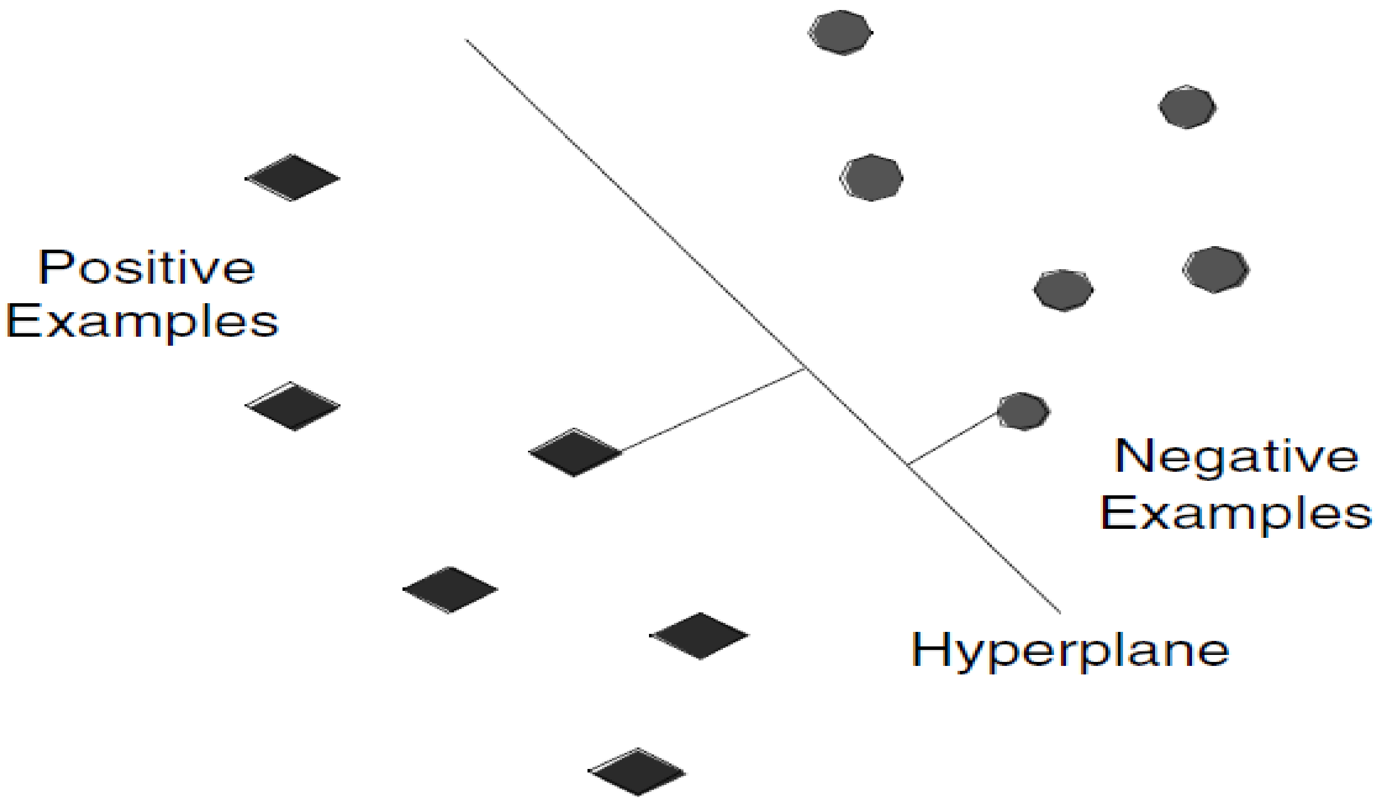
$$\ln p(\boldsymbol{\beta}) = -\left(\sum_i (\ln 2 - \ln\lambda + \lambda \mid \beta_i \mid) \right), \text{ for Laplace prior.}$$

Decision Tree Classifiers

- Many categorization methods share a certain drawback: classifiers cannot be easily understood by humans.
- Decision tree classifiers do not suffer from this problem.
- **Decision Trees** can present us with a graphical representation of how the classifier reaches its decision.
- A decision tree (DT) classifier is a tree in which the internal nodes are labelled by the features,
- Edges leaving a node are labelled by tests on the feature's weight
- The leaves are labelled by categories.
- A DT categorizes a document by starting at the root of the tree and moving successively downward via the branches whose conditions are satisfied by the document until a leaf node is reached.
- The document is then assigned to the category that labels the leaf node.

Decision Tree Classifiers

Tree that corresponds to CONSTRUE rule mentioned may look like



A Decision Tree classifier.

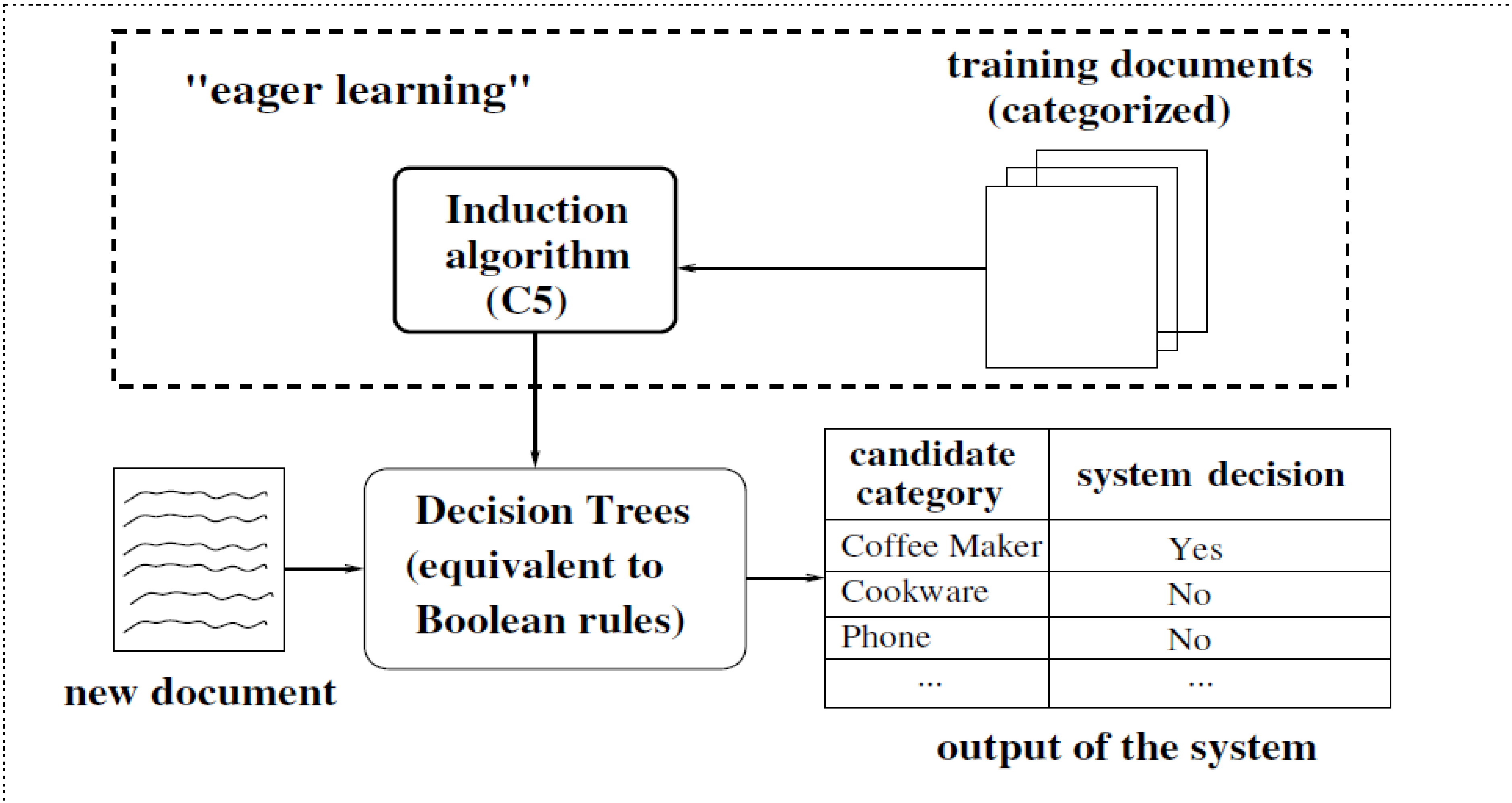
Decision Tree Classifiers

- Most of the DT-based systems use some form of general procedure for a **DT induction such** as ID3, C4.5, and CART (Classification and Regression Trees).
- Typically, the tree is built recursively by picking a feature *f* at each step and dividing the training collection into two sub-collections, one containing *f* and another not containing *f*, until only documents of a single category remain – at which point a leaf node is generated.
- The choice of a feature at each step is made by some information-theoretic measure such as information gain or entropy.

Decision Tree Classifiers

- Trees generated in such a way are prone to overfit the training collection, and so most methods also include *pruning* – that is, removing the too specific branches.
- The performance of a DT classifier is mixed but is inferior to the top-ranking classifiers.
- Thus it is rarely used alone in tasks for which the human understanding of the classifier is not essential.
- DT classifiers, however, are often used as a baseline for comparison with other classifiers and as members of classifier committees.

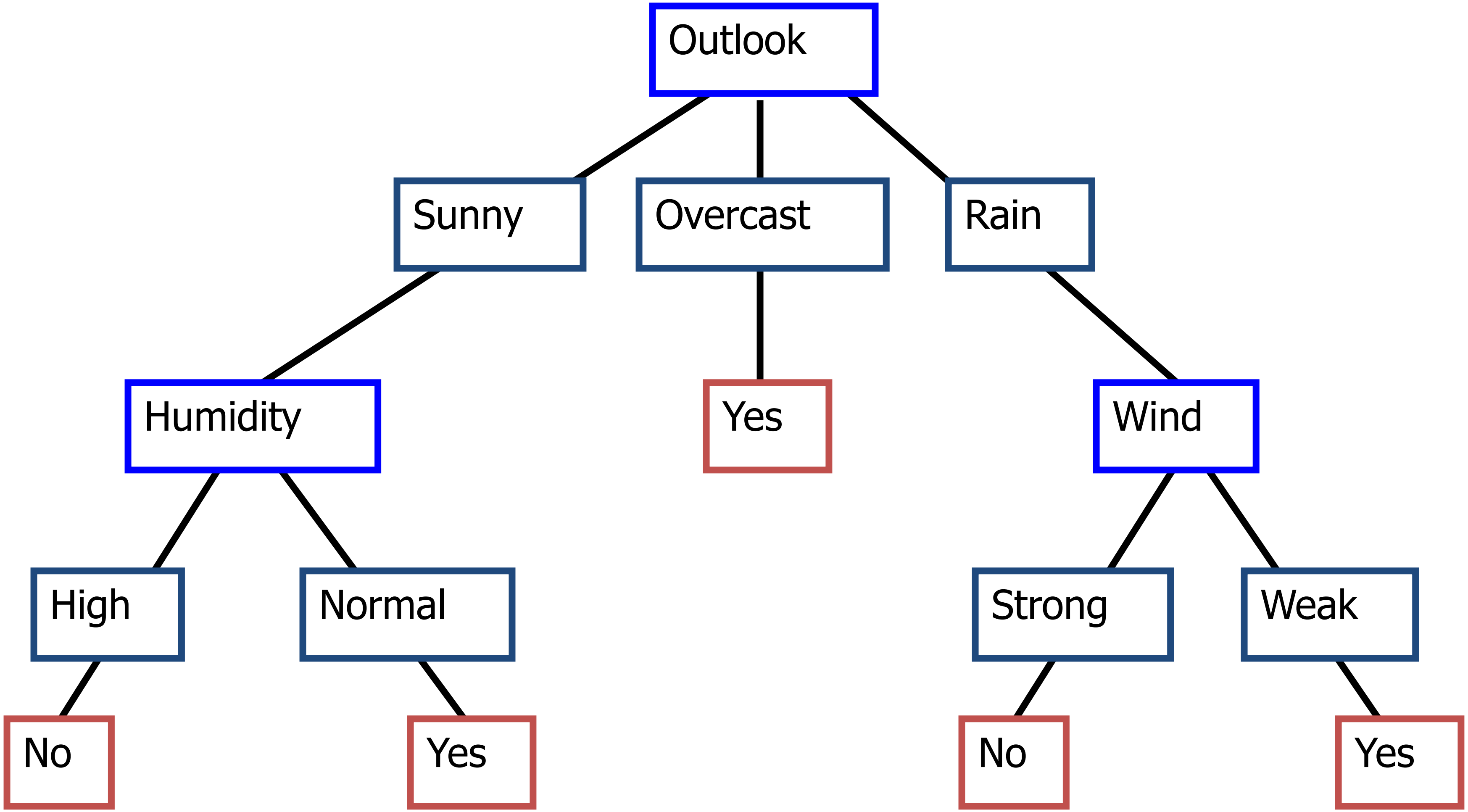
Decision Tree Classifiers



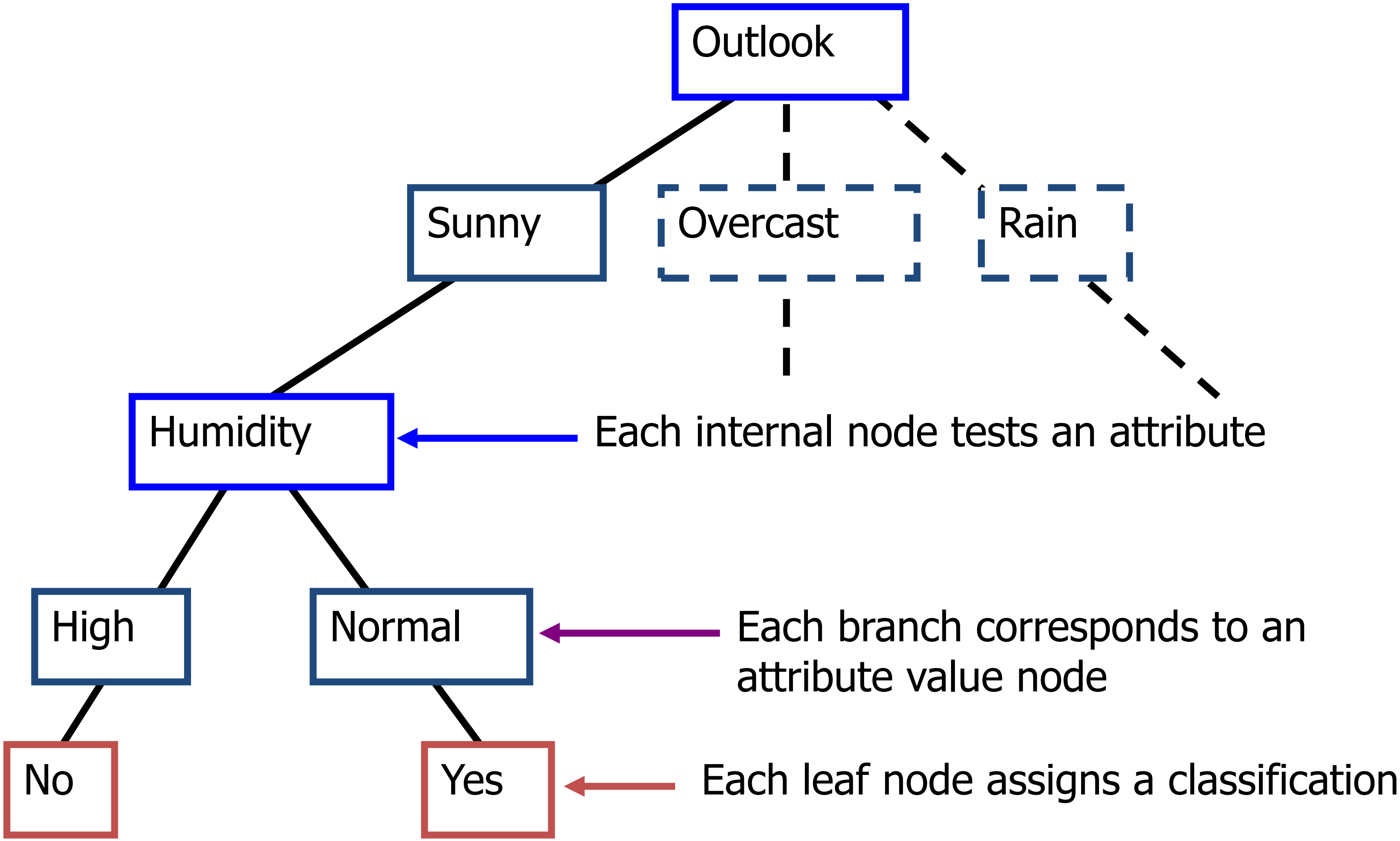
Decision Tree Classifiers

Day	Outlook	Temp.	Humidity	Wind	**Play Tennis**
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Decision Tree Classifiers



Decision Tree Classifiers



Decision Rule Classifiers

- Decision rule (DR) classifiers are also like decision trees.
- The rules look very much like DNF rules of the CONSTRUE - built from the training collection using *inductive rule learning*.
- Rule learning methods selects best rule from the set of all possible covering rules according to some optimality criterion.
- DNF rules are often built in a bottom-up fashion.
- The initial most specific classifier is built from the training set by viewing each training document as a clause.
- Example

  $d1 \wedge d2 \wedge . . . \wedge dn \rightarrow c,$

  where di are features of document and c its category

Decision Rule Classifiers

- Rule learner then applies a series of generalizations for maximizing the compactness of the rules
- At the end of the process, a pruning step is applied
- Rule learners vary widely in their specific methods depending on
  - heuristics, and
  - Optimality criteria.
- One of the prominent algorithms is **RIPPER** (**R**epeated **I**ncremental **P**runing to **P**roduce **E**rror **R**eduction)
- Ripper builds a rule set by first adding new rules until all positive category instances are covered and then adding conditions to the rules until no negative instance is covered.
- One of the attractive features of Ripper is its ability to bias the performance by the setting of the loss ratio parameter.

Regression Methods

- *Regression* is a technique for approximating a real-valued function using the knowledge of its values on a set of points.
- It can be applied to TC, which is the problem of approximating the category assignment function.
- For this method to work, the assignment function must be considered a member of a suitable family of continuous real-valued functions.
- Then the regression techniques can be applied to generate the (real-valued) classifier.

Regression Methods

- One method is the **L**inear **L**east-**S**quare **F**it **(LLSF)**
- Category assignment function is viewed as a $|C| \times |F|$ matrix
- Describes some linear transformation from the feature space to the space of all possible category assignments.
- To build a classifier, we create a matrix that best accounts for the training data.
- The LLSF model computes the matrix by minimizing the error on the training collection according to the formula

$$\boxed{M = \arg\min_M \|MD - O\|_F,}$$

where D is the |F| × |Training Collection| matrix
O is the |C| × |TrainingCollection| matrix of category assignments, and the
$\|\cdot\|_F$ is the Frobenius norm

$$\boxed{\|A\|_F = \sqrt{\sum A_{ij}^2}.}$$

The Rocchio Method

- **Rocchio classifier** categorizes a document by computing its distance to the prototypical examples of the categories.
- A prototypical example for the category c is a vector $(w1, w2, \ldots)$ in the feature space computed by

$$\boxed{w_i = \frac{\alpha}{|\mathrm{POS}(c)|} \sum_{d \in \mathrm{POS}(c)} w_{di} - \frac{\beta}{|\mathrm{NEG}(c)|} \sum_{d \in \mathrm{NEG}(c)} w_{di},}$$

  - where POS(c) and NEG(c) sets of all training documents that belong and do not belong to the category c, respectively,
  - wdi is the weight of ith feature in the document d.
- Usually, positive examples are more important than negative ones, and so **α >>β**.
- If β = 0, then prototypical example for a category is simply **centroid** of all documents belonging to the category.

The Rocchio Method

- The Rocchio method is very easy to implement
- It is computationally cheap.
- Its performance, however, is usually mediocre – especially with the categories that are not linearly separable.

Neural Networks

- Neural network (NN) can be built to perform text categorization.
- Input nodes of the network receive the feature values, the output nodes produce the categorization
- Link weights represent dependence relations.
- For classifying a document, its feature weights are loaded into input nodes
- Activation of the nodes is propagated forward through the network, and the final values on output nodes determine the categorization decisions.

Neural Networks

- The neural networks are trained by back propagation
- If a misclassification occurs, the error is propagated back through the network, modifying the link weights in order to minimize the error.
- Simplest kind of a neural network is Perceptron (two layers), equivalent to a linear classifier
- More complex networks contain one or more hidden layers.

Example-Based Classifiers

➢ Example-based classifiers do not build explicit declarative representations of categories.

➢ Rely on directly computing the similarity between the document to be classified and the training documents.

➢Those methods have thus been called lazy learners.

➢ Defer the decision on how to generalize beyond the training data until each new query instance is encountered.

➢Training" for such classifiers consists of simply storing the representations of the training documents together with their category labels.

Example-Based Classifiers

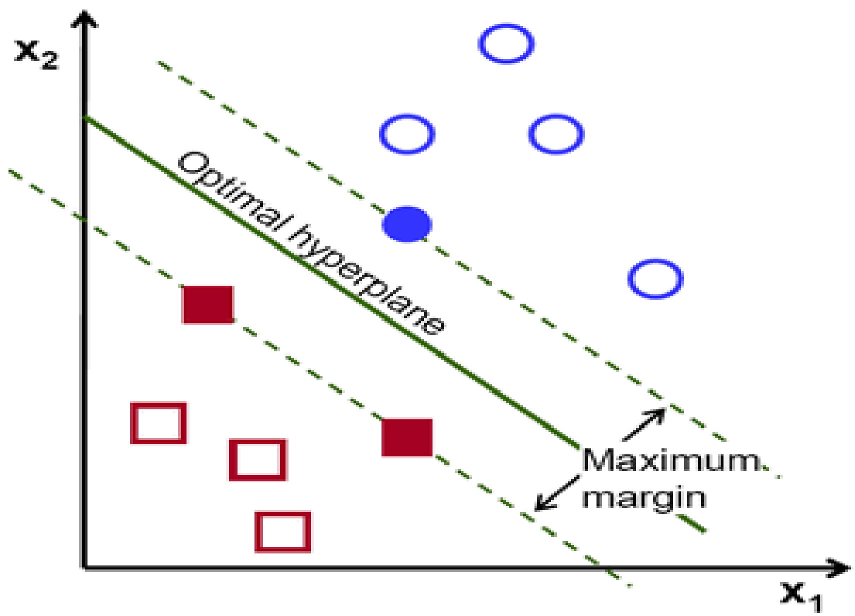
- Most prominent example of an example-based classifier is kNN (k-nearest neighbor).
- To decide whether a document d belongs to the category c, kNN checks whether the k training documents most similar to d belong to c.
- If the answer is positive for a sufficiently large proportion of them, a positive decision is made; otherwise, negative.
- The distance-weighted version of kNN is a variation that weighs the contribution of each neighbor by its similarity to the test document.

Example-Based Classifiers

- Need choose the value of k.
- Can be optimized using a validation set, but it is probable that a good value can be picked a priori.
- Researchers use k = 20, or 30 ≤ k ≤ 45
- Experiments have shown that increasing the value of k does not significantly degrade the performance.
- kNN is one of the best-performing text classifiers.
- It is **robust** - categories **NOT to be** linearly separated.
- **Drawback** - relatively high computational cost of classification

Support Vector Machines

- Support Vector Machine (SVM) is very fast and effective for text classification problems
- In geometrical terms,
  - ✓ a binary SVM classifier can be seen as a hyperplane in the feature space separating the points that represent the positive instances of the category from the points that represent the negative instances.
  - ✓The classifying hyperplane is chosen during training as the unique hyperplane that separates the known positive instances from the known negative instances with the maximal margin.



Support Vector Machines

Example of a maximal margin hyperplane in two dimensions

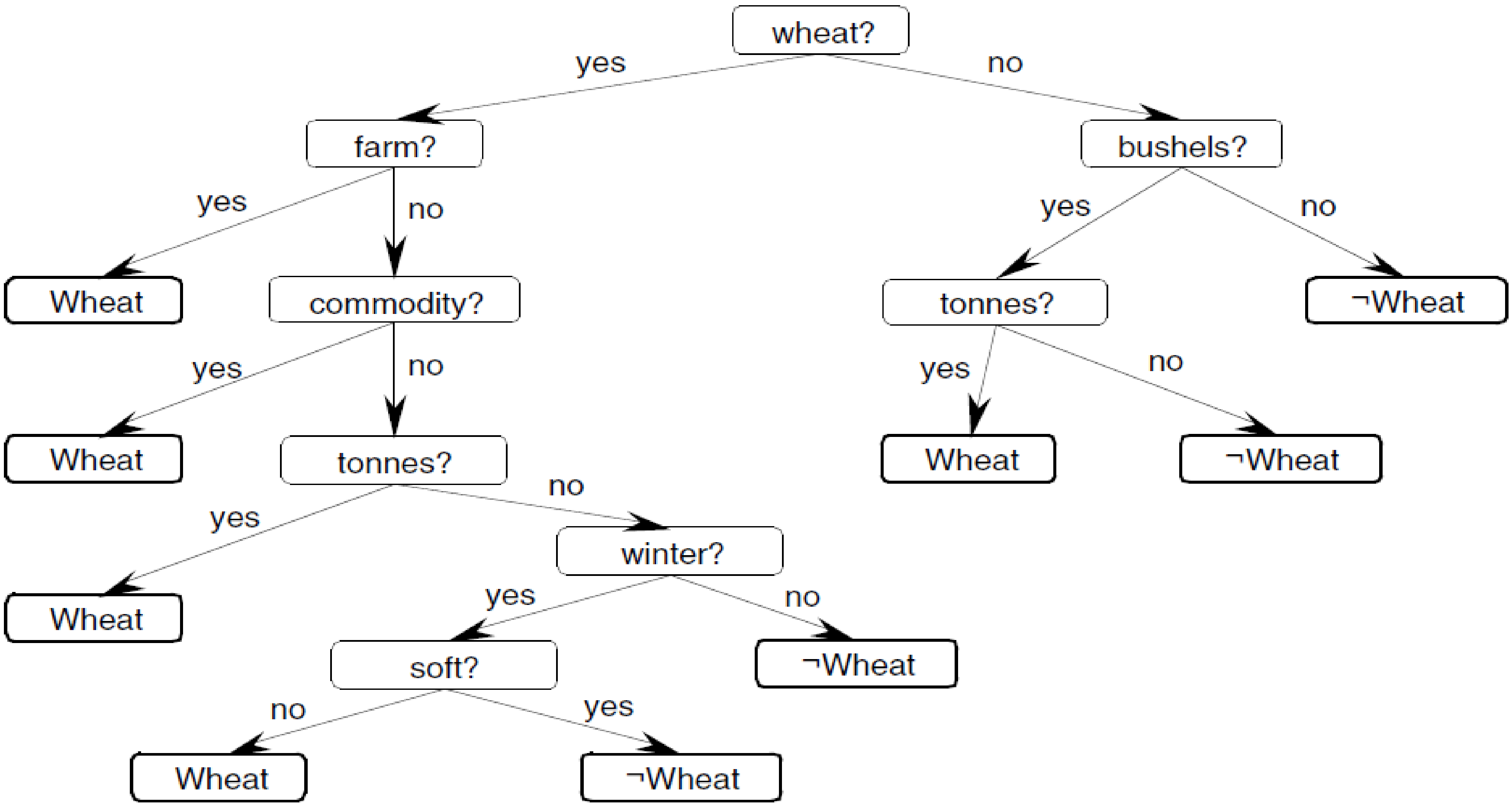


Diagram of a 2-D Linear SVM.

Support Vector Machines

- SVM hyperplanes are determined by small subset of the training instances - **support vectors**.
- The rest of the training data have no influence on the trained classifier.
- SVM algorithm - different categorization algorithms.
- SVM classifier has an important advantage in overfitting problem.
- Needs no parameter adjustment because there is a theoretically motivated “default” choice of parameters.
- Experimentally to provide the best performance for default” choice of parameters.

Classifier Committees: Bagging and Boosting

- Committees of classifiers stems from the intuition a team of experts, may produce better results

Bagging Method –

- Individual classifiers are trained in parallel on same training data
- Assume there are k different classifiers
- must choose the method of combining their results
- The simplest method is the majority vote
  - ✓ category is assigned to a document iff at least (k+1)/2 classifiers decide this way
- Another possibility, suited for continuous output, is the weighted linear combination, whereby the final Categorization Status Value (CSV) is given by a weighted sum of the CSVs of the k classifiers.

Classifier Committees: Bagging and Boosting

Boosting Method –

- Classifiers are trained sequentially
- Before training ith classifier, the training set is reweighed with greater weight given to the documents that were misclassified by the previous classifiers.
- AdaBoost algorithm - best known example of this approach
- Let X be the feature space, and let D= {(d1, c1), (d2, c3), . . .} be training data , ci ∈ {+1,−1}
- A weak learner is some algorithm that is able to produce a weak hypothesis (classifier) h : X→ {±1} given the training data D together with a weight distribution W upon it.
- The "goodness" of a hypothesis is measured by its error

$$\varepsilon(h, W) = \sum_{i : h(d_i) \neq c_i} W(i),$$

(sum of weights of misclassified documents)

AdaBoost algorithm

- Initializes weights distribution $W_1(i) = 1/|D|$ for all i, and
- Repeats for $t = 1, \ldots, k$.

  Train a weak classifier h_t using the current weights W_t.

  Let $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon(h_t, W_t)}{\varepsilon(h_t, W_t)}$.

  Update the weights: $W_{t+1}(i) = Z_t \cdot W_t(i) \cdot$

$$\begin{cases} \exp(-\alpha_t), & \text{if } h_t(d_i) = c_i, \\ \exp(\alpha_t), & \text{otherwise.} \end{cases}$$

  (Z_t is the normalization factor chosen so that $\sum_i W_{t+1}(i) = 1$).
- The final classifier is $H(d) = \operatorname{sign}\left(\sum_{t=1..k} \alpha_t h_t(d)\right)$.

USING UNLABELED DATA TO IMPROVE CLASSIFICATION

- ML classifiers require fairly large training collections.
- Unlabeled documents usually exist in abundance
- improve classifier performance by augmenting a small number of labelled documents with a large number of unlabelled
- The two common ways of incorporating knowledge from unlabeled documents
    1. Expectation maximization (EM)
    2. Cotraining

Expectation Maximization (EM)

EM algorithm performs the optimization in a simple way

- First, the model is trained over the labelled documents.
- Then the following steps are iterated until convergence in a local maximum occurs:

E-step: the unlabeled documents are classified by the current model.

M-step: the model is trained over the combined corpus.

Cotraining

- Cotraining works with the documents, for which two views are available
- Views providing two different document representations - sufficient for classification.
- For example –
  - ✓ Web page may have its content as one view and anchor text appearing in the hyperlinks to the page as another.
  - ✓ In the domain of MedLine papers, abstract may be one view and the whole text another.
- In cotraining unlabeled documents classified by means of one of the views
- Then used for training the classifier using the other view, and vice versa.
- Significant reduction (up to 60%) in amount of labelled training data required

EVALUATION OF TEXT CLASSIFIERS

- TC experiment requires a document collection labelled with a set of categories.
- Divided into two parts: the training and test document sets.
- The training set is used for training classifier, and the test set is the one on which the performance measures are calculated.
- Don’t use the test set in any way during the classifier training and fine-tuning.
- To optimize classifier parameters – the training set divided into two parts – the training set proper and a validation set.
- A commonly used method is the n-fold cross-validation.
- The whole document collection is divided into n equal parts, and then the training-and-testing process is run n times, each time using a different part of the collection as the test set.
- Then the results for n folds are averaged.

EVALUATION OF TEXT CLASSIFIERS

Performance Measures

- The most common performance measures are recall and precision.
- A recall for a category is defined as the percentage of correctly classified documents among all documents belonging to that category,
- Precision is the percentage of correctly classified documents among all documents that were assigned to category by the classifier.
- Another measures the breakeven point, which is the value of recall and precision at the point on the recall-versus-precision curve where they are equal.
- Alternatively, F1 measure, equal to 2/(1/recall + 1/precision), which combines the two measures in an ad hoc way.

Benchmark Collections

- Most known publicly available collection
  1. Reuters set
  2. OHSUMED collection
  3. TREC-AP collection
- This collection accounts for most of the experimental work in TC.
- In order for the results of two experiments to be directly comparable
  - The experiments must be performed on exactly the same collection using the same split between training and test sets.
  - The same performance measure must be chosen
  - If a particular part of a system is compared, all other parts must be exactly the same

Comparison among Classifiers

General conclusions in reference to the question Which classifier is the best?

- According to most researchers, the top performers are SVM, AdaBoost, kNN, and Regression methods.
- Rocchio and Naive Bayes have the worst performance among the ML classifiers-very useful as a member of classifier committees
- There are mixed results regarding the neural networks and decision tree classifiers.
  - ✓ Some of the experiments have demonstrated rather poor performance
  - ✓ whereas in other experiments they performed nearly as well as SVM

Clustering

Clustering

- **Clustering** is an unsupervised process through which objects are classified into groups called **clusters**.
- In case of **clustering**, the problem is to group the given unlabeled collection into meaningful clusters **without any prior information**
- Labels associated with objects are obtained solely from the data.
- Clustering is useful in
  - data mining,
  - document retrival,
  - image segmentation, and
  - pattern classification.

CLUSTERING TASKS IN TEXT ANALYSIS

- One application of clustering is the analysis and navigation of big text collections such as Web pages.

- Cluster hypothesis states that relevant documents tend to be more similar to each other than to nonrelevant ones.

- If this assumption holds, the clustering of documents based on the similarity of their content may help to improve the search effectiveness

CLUSTERING APPLICATIONS

- Improving Search Recall
- Improving Search Precision
- Scatter/Gather
- Query-Specific Clustering

Precision And Recall In Search Engines

- ***Precision*** is the percentage of documents in the result set that are relevant.
- ***Recall*** is the percentage of relevant documents that are returned in the result set.

$$\textbf{\textit{Recall}} = \frac{\text{Number of pages that were retrieved and relevant}}{\text{Total number of relevant pages}}$$

$$\textbf{Precision} = \frac{\text{Number of pages that were retrieved and relevant}}{\text{Total number of retrieved pages}}$$

Improving Search Recall

- Standard search engines and IR systems return lists of documents that match a user query.
- Same concepts are expressed by different terms – e.g. “car” - “automobile,”
- Clustering,, may help improve the recall
- Might significantly degrade precision

Improving Search Precision

- More documents difficult task to browse
- Must know exact search terms in order to find a document of interest.
- or left with tens of thousands of matched documents
- Clustering can group documents into a much smaller number of groups of related documents, ordering them by relevance
- Experience, however, has shown that the user needs to guide the clustering process so that the clustering will be more relevant to the user's specific interest.
- An interactive browsing strategy called scatter/gather is the development of this idea.

Scatter/Gather

- Scatter/gather browsing method uses clustering as a basic organizing operation.
- Purpose - enhance the efficiency of human browsing of a document collection when a specific search query cannot be formulated.
- Similar to searching book by index
- During scatter/gather browsing session, a document collection is scattered into a set of clusters, and the short descriptions of the clusters are presented to the user.
- Based on the descriptions, the user selects one or more of the clusters that appear relevant.
- The selected clusters are then gathered into a new sub collection with which the process may be repeated.

Query-Specific Clustering

- Direct approaches to making the clustering query-specific are also possible.
- The hierarchical clustering is especially appealing
- The most related documents will appear in the small tight clusters.
- Recent experiments show better perform over document collections of realistic size.
- Significant improvement in document retrieval can be obtained by using clustering without the need for relevance information from by the user.

THE GENERAL CLUSTERING PROBLEM

A clustering task may include the following components:

- ❑ Problem representation, including feature extraction, selection, or both,
- ❑ Definition of proximity measure
- ❑ Actual clustering of objects,
- ❑ Data abstraction, and
- ❑ Evaluation.

Problem Representation

- All clustering problems are optimization problems.
- **Goal :** Select the best among all possible groupings of objects according to the given clustering quality function.
- The quality function maps a set of possible groupings of objects into the set of real numbers in such a way that a better clustering would be given a higher value.
- A good clustering should group together similar objects and separate dissimilar ones.
- Clustering quality function is usually specified in terms of a similarity function between objects.

Problem Representation

- **Basic requirement** – Similar objects belong to the same clusters and dissimilar to separate ones
- A similarity function takes a pair of objects and produces a real value that is a measure of the objects' proximity.
- To do so, the function must be able to compare the internal structure of the objects.
- Various features of the objects are used for this purpose.
- The most common vector space model assumes that the objects are vectors in the high-dimensional feature space.

Similarity Measures

- The most popular metric is the usual Euclidean distance

$$D(\mathbf{x_i}, \mathbf{x_j}) = \sqrt{\sum_k (x_{ik} - x_{jk})^2},$$

which is a particular case with p = 2 of Minkowski metric

$$D_p(\mathbf{x_i}, \mathbf{x_j}) = \left(\sum_k (x_{ik} - x_{jk})^p \right)^{1/p}.$$

- For text documents clustering, the Cosine Similarity measure is the most common:

$$Sim(\mathbf{x_i}, \mathbf{x_j}) = (x_i' \cdot x_j') = \sum_k x_{ik}' \cdot x_{jk}',$$

where x' is the normalized vector x = x/|x|.

CLUSTERING ALGORITHMS

- Several different variants of an abstract clustering problem.
- A flat (or partitional) clustering produces a single partition of a set of objects into disjoint groups.
- Whereas a hierarchical clustering results in a nested series of partitions.
- Hard clustering, every object may belong to exactly one cluster.
- Soft clustering, the membership is fuzzy – objects may belong to several clusters
- Clustering optimization problems are computationally very hard
- Clustering of n-element sets into k clusters would need to evaluate $k^n / k!$ possible partitionings - exponential in n

CLUSTERING ALGORITHMS

- Agglomerative algorithms begin with each object in a separate cluster and successively merge clusters until a stopping criterion is satisfied.
- Divisive algorithms begin with a single cluster containing all objects and perform splitting until a stopping criterion is met.
- "Shuffling" algorithms iteratively redistribute objects in clusters.
- The most commonly used algorithms are
  - K-means (hard, flat, shuffling),
  - EM-based mixture resolving (soft, flat, probabilistic),
  - HAC (hierarchical, agglomerative)

K-Means Algorithm

The K-means algorithm partitions a collection of vectors *{x*1, *x*2, . . . *xn}* into a set of clusters *{C*1, *C*2, . . . *Ck}*.

Initialization:

k seeds, either given or selected randomly, form the core of k clusters. Every other vector is assigned to the cluster of the closest seed.

Iteration:

The *centroids* M_i of the current clusters are computed:

$$M_i = |C_i|^{-1} \sum_{x \in c_i} x.$$

Each vector is reassigned to the cluster with the closest centroid.

Stopping condition:

At convergence – when no more changes occur.

The K-means algorithm maximizes the clustering quality function Q:

$$Q(C_1, C_2, \ldots, C_k) = \sum_{C_1} \sum_{x \in C_i} Sim(x - M_i).$$

K-Means Algorithm

- K-means algorithm is popular because of its simplicity and efficiency
- Complexity of each iteration is O(kn) similarity comparisons
- **Major problem** - sensitivity to the initial selection of seeds
- If a bad seeds very much suboptimal clusters
- How to predict seed value k
  - ✓ Make several clustering runs with different random choices of seeds
  - ✓ Allow postprocessing of the resulting clusters – e.g. ISO-DATA algorithm
  - ✓ Buckshot algorithm
  - ✓ Run K-means algorithm with different values of k and choosing the best one according to any clustering quality function

EM-based Probabilistic Clustering Algorithm

- The underlying assumption of mixture-resolving algorithms is that the objects to be clustered are drawn from k distributions.
- **Goal :** To identify the **parameters of each distributions** that would allow the calculation of the probability P(Ci | x) of the given object's belonging to the cluster Ci.
- **Expectation Maximization (EM)** is a general purpose framework for estimating the parameters of distribution in the presence of hidden variables in observable data.

EM-based Probabilistic Clustering Algorithm

Adapting it to the clustering problem produces the following algorithm:

Initialization:

The initial parameters of k distributions are selected either randomly or externally.

Iteration:

E-Step: Compute the $P(C_i \mid x)$ for all objects x by using the current parameters of the distributions. Relabel all objects according to the computed probabilities.

M-Step: Reestimate the parameters of the distributions to maximize the likelihood of the objects' assuming their current labeling.

Stopping condition:

At convergence – when the change in log-likelihood after each iteration becomes small.

Hierarchical Agglomerative Clustering (HAC)

- HAC - begins with each object in separate cluster
- Proceeds to repeatedly merge pairs of clusters that are most similar
- Finishes when everything is merged into a single cluster.

Initialization:

Every object is put into a separate cluster.

Iteration:

Find the pair of most similar clusters and merge them.

Stopping condition:

When everything is merged into a single cluster.

Hierarchical Agglomerative Clustering (HAC)

- Different versions based how the similarity between clusters is calculated
  - single-link - maximum of similarities between pairs of objects
  - complete-link – minimum of similarities of such pairs of objects
  - center of gravity – similarity between centroids of clusters
  - average link - average similarity between pairs of objects
  - group average- average similarity between all pairs of objects in a merged cluster
- Complexity of HAC is $O(n^2s)$

Hierarchical Agglomerative Clustering (HAC)

- Compute group average cluster similarity in constant time
- Makes HAC truly quadratic.
- By definition, the group average similarity between clusters Ci and Cj is

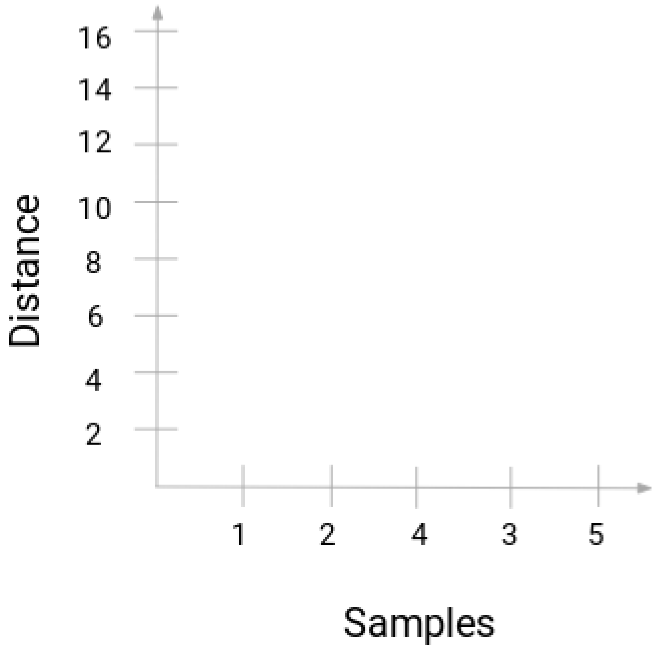
$$Sim(C_i, C_j) = \frac{1}{|C_i \cup C_j|(|C_i \cup C_j| - 1)} \sum_{x,y \in C_i \cup C_j, x \neq y} Sim(x, y).$$

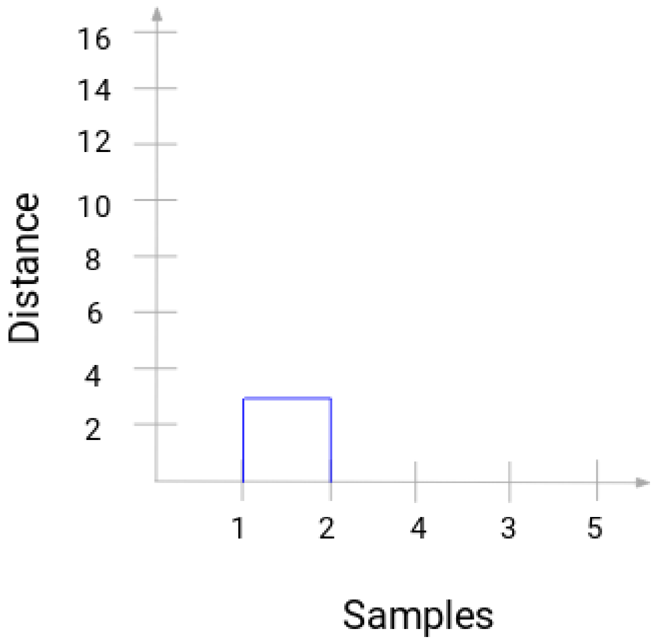
- Assuming that the similarity between individual vector is the cosine similarity, we have

$$Sim(C_i, C_j) = \frac{(S_i + S_j) \cdot (S_i + S_j) - (|C_i| + |C_j|)}{|C_i \cup C_j|(|C_i \cup C_j| - 1)},$$

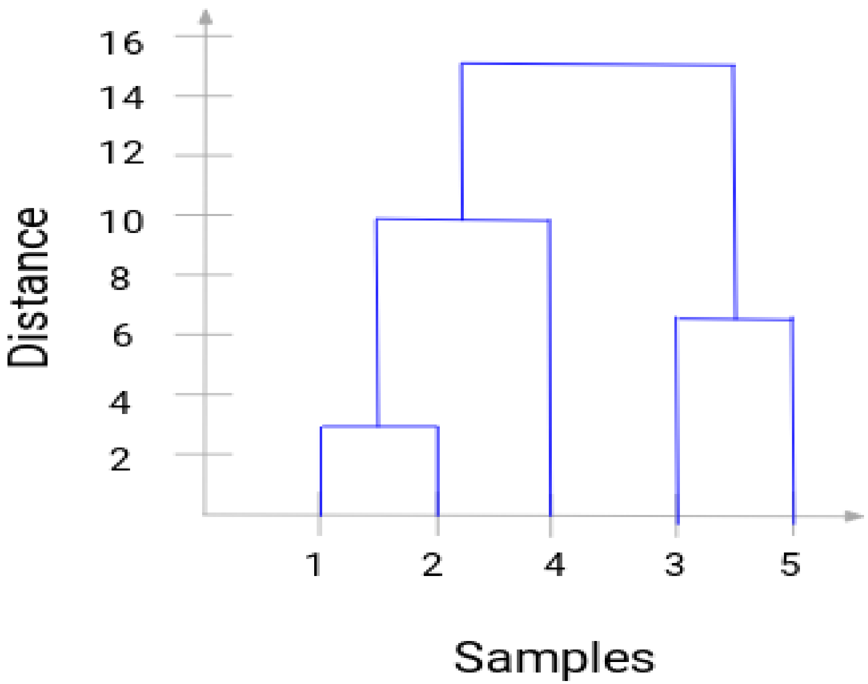
How should we Choose the Number of Clusters in Hierarchical Clustering?

- Makes use of an awesome concept called a **Dendrogram**.
- *A dendrogram is a tree-like diagram that records the sequences of merges or splits.*





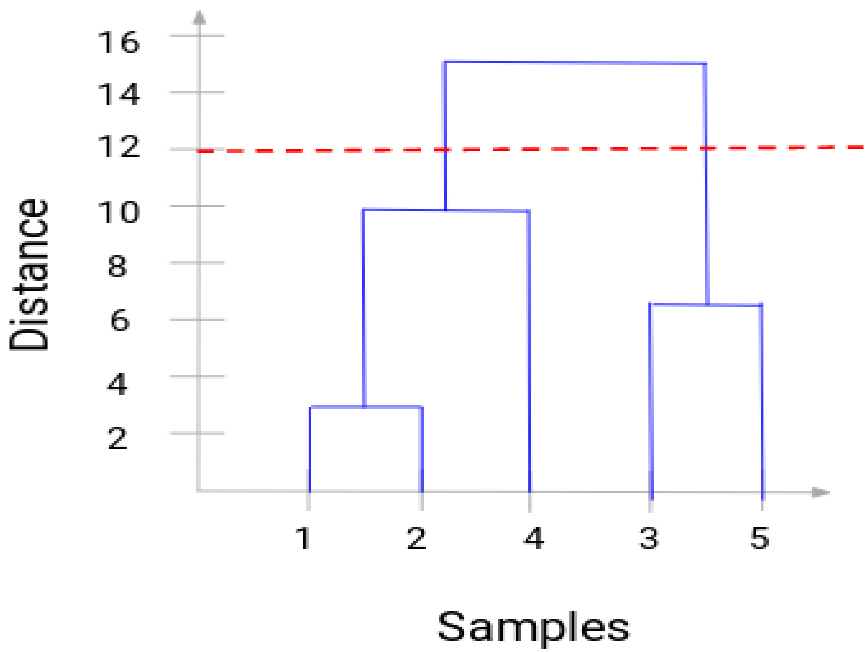
Hierarchical Agglomerative Clustering (HAC)



- We can clearly visualize the steps of hierarchical clustering.
- More the distance of the vertical lines in the dendrogram, more the distance between those clusters.

Hierarchical Agglomerative Clustering (HAC)

- Now, we can set a threshold distance and draw a horizontal line.
- *Generally, set the threshold in such a way that it cuts the tallest vertical line*).
- Let's set this threshold as 12 and draw a horizontal line:



- **The number of clusters will be number of vertical lines which are being intersected by the line drawn using the threshold.**
- Since the red line intersects 2 vertical lines, we will have 2 clusters.
- One cluster will have a sample (1,2,4) and the other will have a sample (3,5).

Other Clustering Algorithms

- Several graph-theoretic clustering algorithms exist.
- Minimal Spanning Tree (MST)
- Nearest Neighbor Clustering
- Buckshot algorithm

CLUSTERING OF TEXTUAL DATA

- The clustering of textual data has several unique features
- These features distinguish it from other clustering problems.
- We will discuss
  - Various issues of representation,
  - Algorithms,
  - Data abstraction, and
  - Evaluation of text data clustering problems.

Representation of Text Clustering Problems

- To cluster the documents must be converted into vectors in the feature space.
- Common way - the bag-of-words document representation
- One very important problem arises for clustering – feature selection
- Dimension of feature space range into the tens and hundreds of thousands
- Two possible ways of reducing the dimensionality
    - Local methods - simply delete “unimportant” components from individual document vectors
    - Global Dimension Reduction – Latent Semantic Indexing (LSI).

      Disadvantage - does not adapt to unique characteristics of document.

      Advantage - preserves ability to compare dissimilar documents

Dimension Reduction with Latent Semantic Indexing

- The Singular Value Decomposition (SVD) of a matrix A is the factorization of A into the product of three matrices $A = UDV^T$
- Singular value decomposition is a method of decomposing a matrix into three other matrices.
- A popular application of SVD is for dimensionality reduction.
- Data with a large number of features, such as more features (columns) than observations (rows) may be reduced to a smaller subset of features that are most relevant to the prediction problem.
- The result is a matrix with a lower rank that is said to approximate the original matrix.
- Leads to a low-dimensional representation of a high-dimensional matrix.

Singular Value Decomposition

- An SVD of a real $m \times n$ *matrix A is a representation of the matrix as a product*

$$A = UDV^T,$$

where *U is a column-orthonormal* $m \times r$ *matrix,*

D is a diagonal $r \times r$ *matrix, and*

V is a column-orthonormal $n \times r$ *matrix*

r denotes the rank of A.

- The term “column-orthonormal” means that the column vectors are normalized and have a zero dot-product; thus,

$$UU^T = V^TV = I.$$

- The diagonal elements of *D are the singular values of A and can all be chosen to* be positive and arranged in a descending order.
- Then the decomposition becomes unique.
- There are many methods of computing the SVD of matrices.

Using SVD for Dimension Reduction

- First, a terms-by-documents rectangular matrix *A is formed.*
- *Its columns are the vector representations of documents.*
- Thus, the matrix element A_{td} *is nonzero when the term t appears in the* document *d.*
- Then, the SVD of the matrix *A is calculated:*

$$A = UDV^T.$$

- Next the dimension reduction takes place
- We keep the *k highest values in the* matrix *D and set others to zero, resulting in the matrix D'.*
- It can be shown that the matrix

$$A' = UD'V^T$$

is the matrix of rank *k that is closest to A.*

Using SVD for Dimension Reduction

- Cosine similarity given by dot product of their corresponding columns in the *A matrix*
- The reduced-dimensional approximation is calculated as the dot product of the columns of *A'.*

$$A'^{T}A = VD'^{T}U^{T}UD'V^{T} = VD'^{T}D'V^{T}$$

- Low-dimensional LSI representation of documents space is given by the rows of the VD'^{T} *matrix*,
- *Dot product can be calculated between those k-dimensional rows.*

Using Naıve Bayes Mixture Models with the EM Clustering Algorithm

- Model has the following parameters:
  - prior cluster probability $P(Ci)$ *and*
  - *probabilities* $P(fi\,|Ci)$ *of* features in the cluster
- Given the model parameters, the probability that a document belongs to a cluster is

$$\boxed{P(C_i|x) = P(C_i)\prod_f P(f|C_i) \Big/ \sum_C P(C)\prod_f P(f|C)}$$

- Assuming current document labeling is $L(x)$, *the maximum* likelihood estimation of the parameters is

$$P(C_i) = |\{x : L(x) = C_i\}|/N,$$
$$P(f|C_i) = |\{x : L(x) = C_i \quad \text{and} \quad f \in x\}|/|\{x : L(x) = C_i\}|,$$

where N *is the number of documents.*

- Can improve *categorization systems in cases of few* labeled documents.
- Labeled documents can be used to *train initial NB models*, then *EM for clustering*.

Data Abstraction in Text Clustering

- Data abstraction in clustering problems entails generating a meaningful and concise description of the cluster.
- Useful for automatic processing or for user consumption
- Machine-usable abstraction - cluster centroids or probabilistic models of clusters.
- Text Clustering – Give the user a meaningful cluster label
- For scatter/gather browsing - good labeling is required
  - very small number of terms precisely distinguishing cluster from others
- Generating Cluster Labels automatically-
  - Title of medoid document or other document titles
  - Several words common to the cluster documents- top 5-10 most frequent terms
  - A distinctive noun phrase, is probably the best label.

Evaluation of Text Clustering

- Measuring the quality of an algorithm is a common problem
- Quality of the results needs human judgment - subjectivity.
- Need a measure of how good clustering is for human consumption or for further processing.
- Given a set of categorized (manually classified) documents.
- Most common measure is *purity.*
- *Assume* $\{L_1, L_2, \ldots, L_n\}$ *are the manually labeled classes*
- *Let* $\{C_1, C_2, \ldots, C_m\}$ *are the clusters returned by the clustering process*

$$\boxed{Purity(C_i) = \max_j |L_j \cap C_i| / |C_i|.}$$

- Other measures include the *entropy of classes in clusters, mutual information*
- Most useful evaluation - utility of the resulting clustering in its intended application

Thank You !!!