WSD
(Word Sense Disambiguation)

Sense Ambiguity:

Definition:

- Many words have several meanings or senses.
- In the sentence, what is the intended meaning of the word, is the problem of WSD.

Disambiguation:

- The task of disambiguation is to determine which of the senses of an ambiguous word is invoked in a particular use of the word.

Algorithms for handling WSD problem

1. **Knowledge Based Approaches**
   - Rely on knowledge resources like WordNet, Thesaurus etc.
   - May use grammar rules for disambiguation.
   - May use hand coded rules for disambiguation.
2. **Machine Learning Based Approaches**
   - Rely on corpus evidence
   - Train a model using tagged or untagged corpus.
   - Probabilistic/Statistical models.
3. **Hybrid Approaches**
   - Use corpus evidence as well as semantic relations from WordNet.

Knowledge Based Approaches

Knowledge-Based Approaches:

1. The objective of knowledge-based or dictionary-based WSD is to exploit knowledge resources (such as dictionaries, thesauri, ontologies, collocations, etc) to infer the senses of words in context.
2. These are **overlap based approaches**

OVERLAP BASED APPROACHES

1. Require a **Machine Readable Dictionary (MRD)** like WordNet**.**
2. Find the overlap between the features of different senses of an ambiguous word (sense bag) and the features of the words in its context (context bag).
3. These features could be sense definitions, example sentences, hypernyms, etc.
4. The features could also be given weights.
5. The sense which has the maximum overlap is selected as the contextually appropriate sense.

For example:

Word sequence: w_1, ---, $\mathbf{w_i}$, ---, w_n

- Suppose $\mathbf{w_i}$ is the ambiguous word .
- It has two senses:$\mathbf{w_i}$**'(sense1)** and $\mathbf{w_i}$**''(sense2)**.

Find out whether sense1 or sense2 assigned to w_i.

1. Find sense bag for each sense
2. Construct context bag for all words other than the ambiguous word
3. Find overlap between sense bag and context bag
4. Sense which has maximum overlap would be the most probable sense for the word.

Lesk Algorithm

- (Michael Lesk 1986): Identify senses of words in context using definition overlap

Algorithm:

1. Retrieve from **MRD**(**Machine Readable Dictionary**) all sense definitions of the words to be disambiguated. Construct sense bag and context bag.
2. Determine the definition overlap for all possible sense combinations
3. Choose senses that lead to highest overlap

Example: disambiguate PINE CONE

- **PINE**
  1. kinds of evergreen tree with needle-shaped leaves
  2. waste away through sorrow or illness
- **CONE**
  1. solid body which narrows to a point
  2. something of this shape whether solid or hollow
  3. fruit of certain evergreen trees

Pine#1 ∩ Cone#1 = 0
Pine#1 ∩ Cone#2 = 1
Pine#1 ∩ Cone#3 = 2
Pine#2 ∩ Cone#1 = 0
Pine#2 ∩ Cone#2 = 0
Pine#2 ∩ Cone#3 = 0

LESK'S ALGORITHM

Sense Bag: *contains the words in the definition of a candidate sense of the ambiguous word. { for each sense, sense bag is constructed }*

Context Bag: *contains the words in the definition of each sense of each context word.*

E.g. "On burning ***coal*** we get ***ash***."

Ash

Sense 1
Trees of the olive family with pinnate leaves, thin furrowed bark and gray branches.
Sense 2
The ***solid*** residue left when ***combustible*** material is thoroughly ***burn***ed or oxidized.
Sense 3
To convert into ash

Coal

Sense 1
A piece of glowing carbon or ***burn***t wood.
Sense 2
charcoal.
Sense 3
A black ***solid combustible*** substance formed by the partial decomposition of vegetable matter without free access to air and under the influence of moisture and often increased pressure and temperature that is widely used as a fuel for ***burn***ing

Ash#1 ∩ Coal#1 = 0
Ash#1 ∩ Coal#2 = 0
Ash#1 ∩ Coal#3 = 1
Ash#2 ∩ Coal#1 = 1
Ash#2 ∩ Coal#2 = 0
Ash#2 ∩ Coal#3 = 3
Ash#3 ∩ Coal#1 = 0
Ash#3 ∩ Coal#2 = 0
Ash#3 ∩ Coal#3 = 0

In this case Sense 2 of ash would be the winner sense.

CRITIQUE

- **Proper nouns** in the context of an ambiguous word can act as strong disambiguators.

E.g. “Sachin Tendulkar” will be a strong indicator of the category “sports”.

Sachin Tendulkar plays cricket.

- Proper nouns are **not present** in the thesaurus.
- This approach fails to capture the strong clues provided by proper Nouns.

Limitations of (Lesk's) Gloss Overlaps

- Most glosses are very short.
  - So not enough words to find overlaps with.

- **Solution:** Extended gloss overlaps
  - Add glosses of synsets connected to the input synsets.

Extended Lesk Algorithm

- A thesaurus-based measure that looks at **glosses**
- Extension includes glosses of semantically related senses from WordNet (e.g. hypernyms, hyponyms, etc.).
- Two concepts are similar if their glosses contain similar words
  - ***Drawing paper:*** **paper** that is **specially prepared** for use in drawing
  - ***Decal:*** the art of transferring designs from **specially prepared paper** to a wood or glass or metal surface
- For each *n-word* phrase that's in both glosses
  - Add a score of n^2. [$score_{ext\text{-}lesk}(s) = \sum n^2$]
  - Paper and specially prepared, score is: $1^2 + 2^2 = 5$
  - Compute overlap also for other relations
    - Glosses of hypernyms and hyponyms

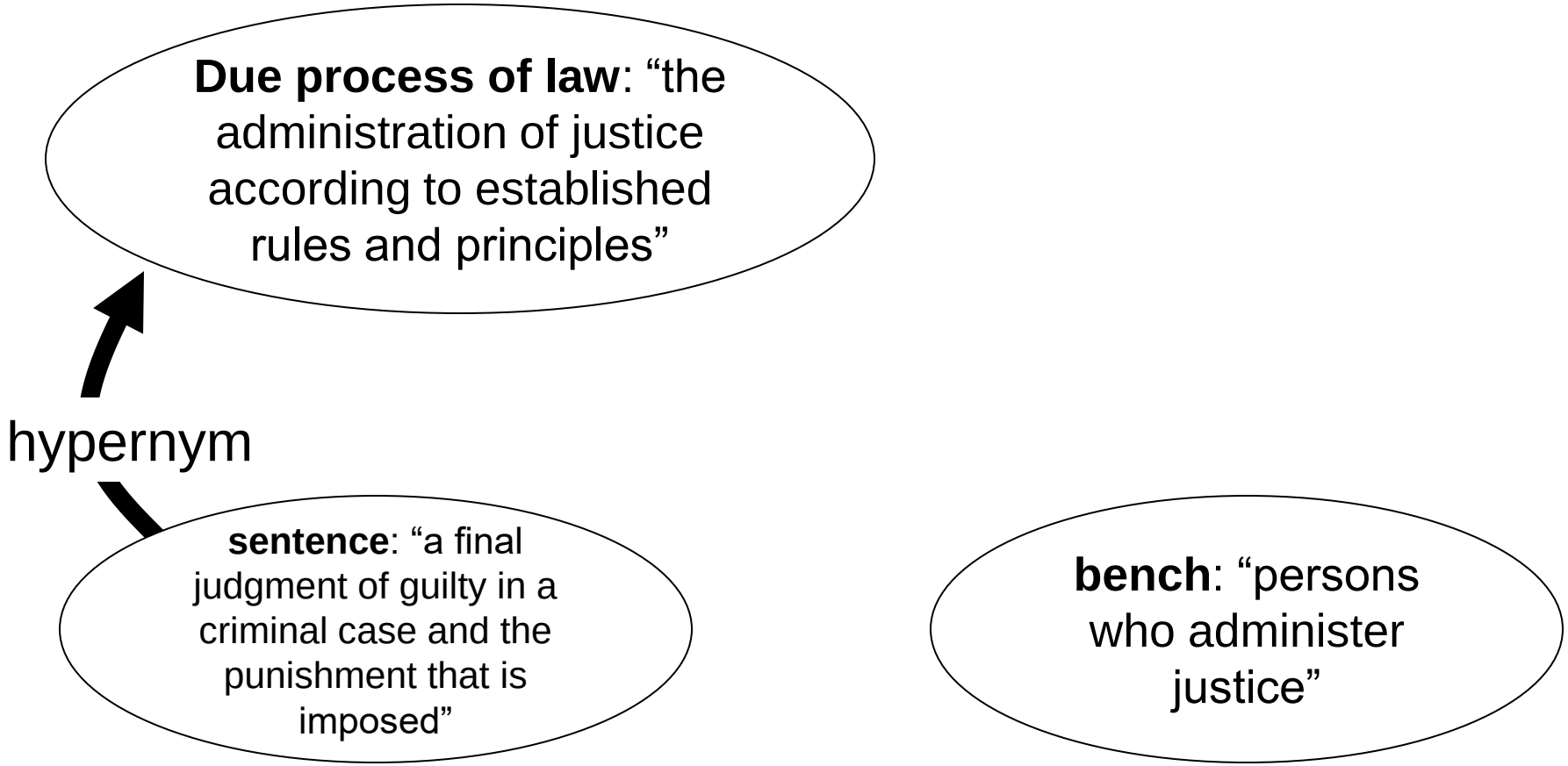
Extending a Gloss

sentence: “a final judgment of guilty in a criminal case and the punishment that is imposed”

bench: “persons who **administer justice**”

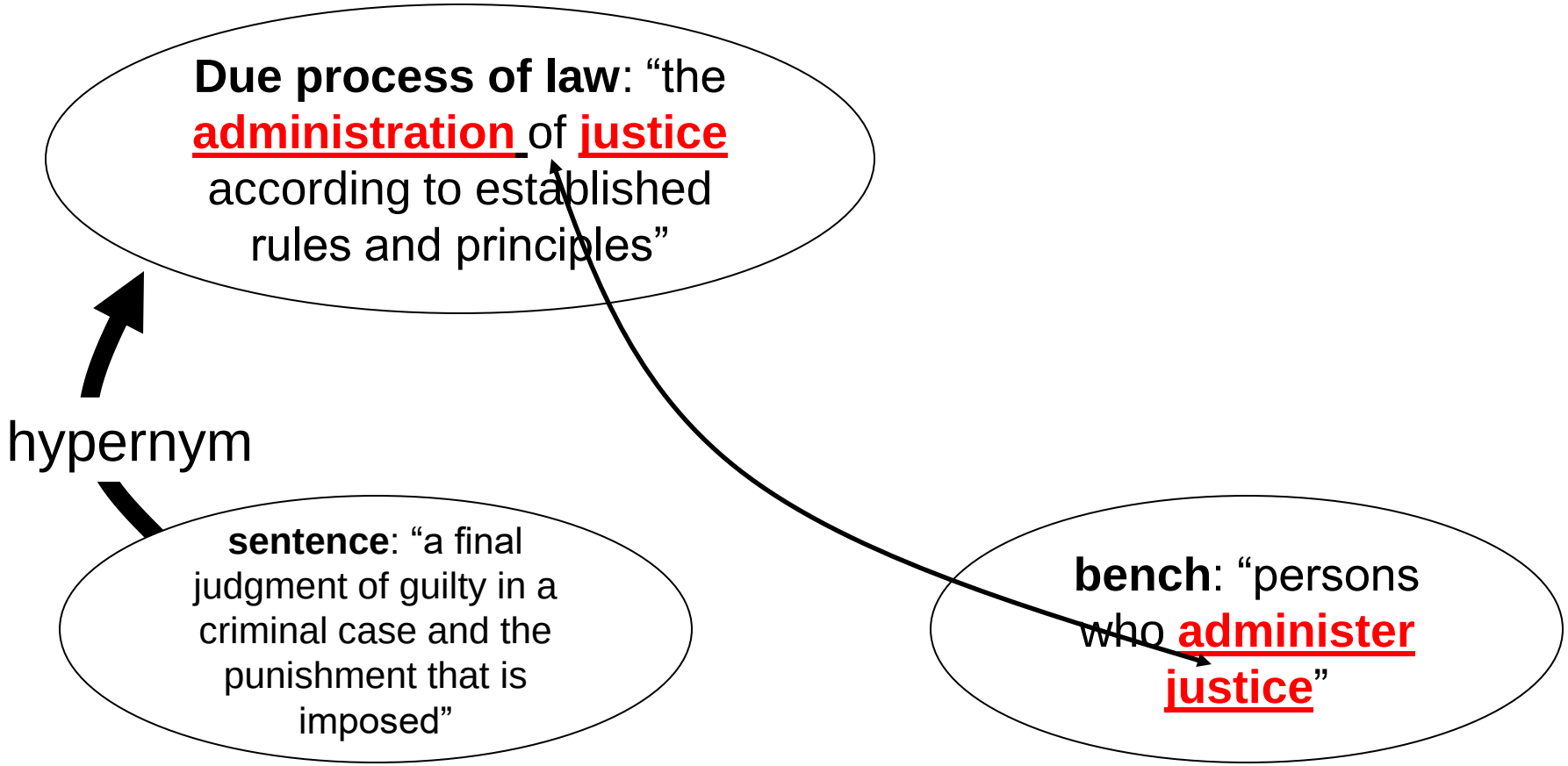
overlapped words = 0

Extending a Gloss



overlapped words = 0

Extending a Gloss



overlapped words = 2

Critique of Extended Lesk

- Larger region of matching in WordNet
  - Increased chance of Matching **BUT**
  - Increased chance of Topic Drift

Problems with thesaurus-based meaning

- We don't have a thesaurus for every language
- Even if we do, they have problems with **recall**
  - Many words are missing
  - Some connections between senses are missing
  - Thesauri work less well for verbs, adjectives
    - Adjectives and verbs have less structured hyponymy relations

Walker's Algorithm

- A thesaurus based approach. Predefined categories for each word of the corpus.

Given a word sequence: w_1, ---, $\mathbf{w_i}$, ---, w_n

- Suppose $\mathbf{w_i}$ is the ambiguous word .
- It has two senses:$\mathbf{w_i'}$**(sense1)** and $\mathbf{w_i''}$**(sense2)**.

Step1: For each sense of the ambiguous word, find the thesaurus category to which that sense belongs.

Step2: Each context word would have a thesaurus category. A context word would add 1 to the score of the sense if the thesaurus category of the context word matches that of the sense of the ambiguous word.

Example:

Sentence: The money in this bank fetches an interest of 10% per annum

Ambiguous word: bank [has two categories: finance, location]

Clue words from the context: money, fetch, interest, annum [each have their own thesaurus categories]

	Sense1: Finance	Sense2: Location
Money	+1	0
fetch	0	0
interest	+1	0
annum	+1	0
Score	3	0

- Since sense1 score than sense2 score, sense of bank is taken a Finance

Drawback

- For disambiguating the sense of an ambiguous word, we are not disambiguating the sense of other context words, we are taking all possible senses in one context bag.

For example, a sentence where a ambiguous word is used in many times in different senses.

I went to the bank located on bank of Yamuna to withdraw money

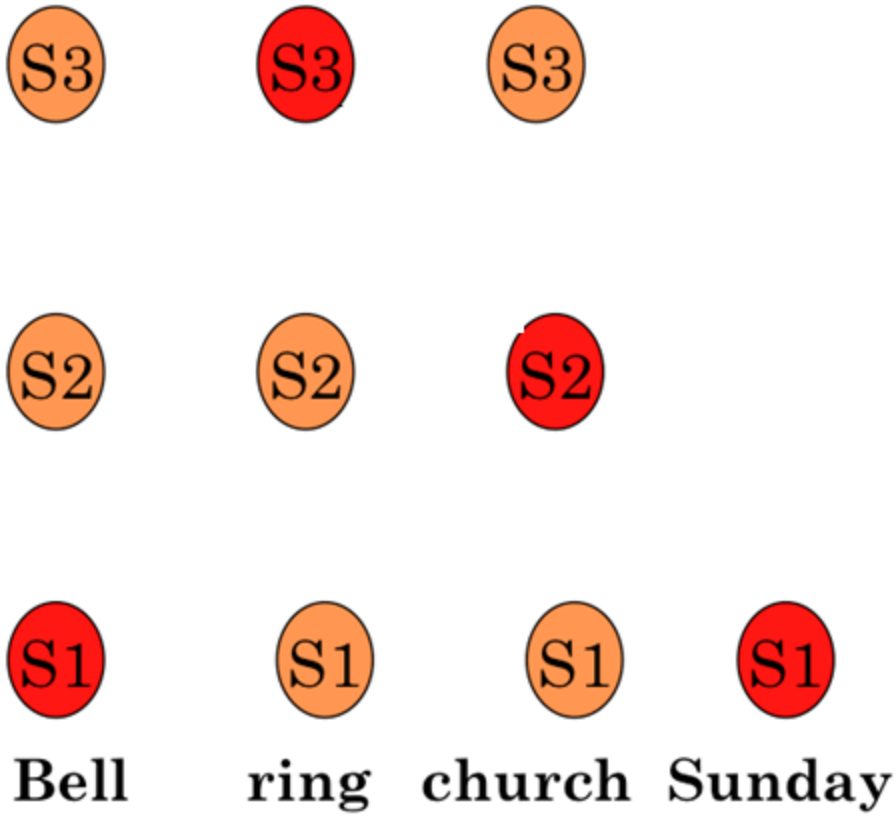
WSD USING RANDOM WALK ALGORITHM (Page Rank) (Sinha andMihalcea, 2007)

WSD USING RANDOM WALK ALGORITHM (Page Rank)

Example: The Church bells no longer rung on Sundays

Consider 4 words:
Church, bells, rung, Sunday

Step1: Add a vertex for each possible sense of each word in the text

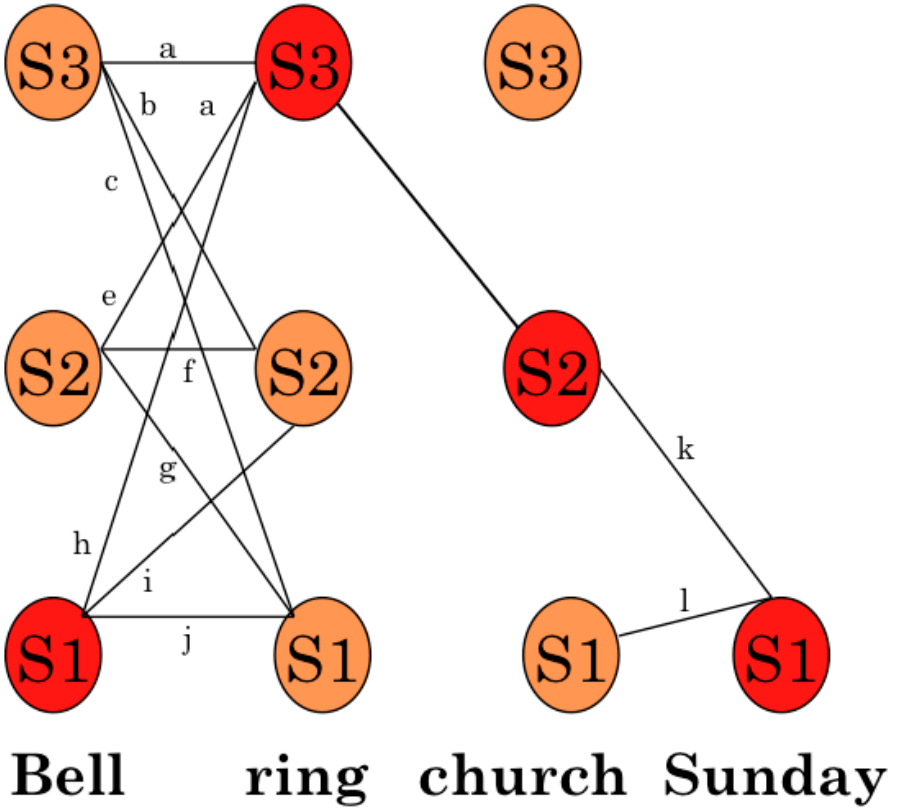


WSD USING RANDOM WALK ALGORITHM (Page Rank)

Step2: Add weighted edges using Lesk based semantic similarity method

Idea : To find that combination of senses such that overall similarity is highest.

Use the PageRank Algorithm:



Idea : If there are appropriate sense definitions that are similar to each other, they would have very high ranking score as they would contribute to each other.

WSD USING RANDOM WALK ALGORITHM (Page Rank)

- We treat it as a problem of finding PR for each vertices of the graph
- PR score is computed in iterative manner

Algorithm for computing PR score:

- Start with some initial random score.
- Use the equation: v=vA –> v denotes PR score
- Start with some initial v0 score and keep on multiplying with A, A^2, A^3,.. Until it converges
- Find PR score for each node. Node with high indegree are given high PR score. This is done for every sense of a word.
- One with max value would be candidate for selection.

Page Rank Algorithm

- PageRank is a probability distribution
- Assume a small universe of four web pages: A, B, C and D.
- The initial approximation of PageRank would be evenly divided between these four pages. Hence, each page would begin with an estimated PageRank of 0.25.

Page Rank Algorithm

- If pages B, C, and D each only link to A, they would each confer 0.25 PageRank to A.
- All links would be pointing to A.

  PR(A)=PR(B)+PR(C)+PR(D)

- PR(A) = 0.75

Page Rank Algorithm

- Suppose that page B has a link to page C as well as to page A, while page D has links to all three pages.
- The value of the link-votes is divided among all the outbound links on a page.
- Thus, page B gives a vote worth 0.125 to page A and a vote worth 0.125 to page C.
- Only one third of D's PageRank is counted for A's PageRank (approximately 0.083).

  PR(A)=PR(B)/2+PR(C)/1+PR(D)/3

- In general,

  PR(U)=$\sum_{V \in B(U)} PR(V)/L(V)$, where B(u) is the set of pages U is linked to, and L(V) is the number of links from V

Supervised Approach for WSD

- NAÏVE BAYES
- Decision List
- K-nn based
- SVM

Naïve Bayes

- The Algorithm find the winner sense using

  $s^* = \text{argmax}_{s \in \text{senses}} \Pr(s|V_w)$

where 'V_w' is a feature vector consisting of:

  - POS of w
  - Semantic & Syntactic features of w
  - Collocation vector (set of words around it) typically consists of next word(+1), next-to-next word(+2), -2, -1 & their POS's
  - Co-occurrence vector (number of times w occurs in bag of words around it)

Applying Bayes Rule and Naive independence assumption

$$s^* = \text{argmax}_{s \in \text{senses}} \Pr(s) \prod_{i=1}^{n} \Pr(V_w^i / s)$$

ESTIMATING PARAMETERS

- Parameters in the probabilistic WSD are:
  - Pr(s)
  - $\Pr(V_w^i / s)$
- Senses are marked with respect to sense repository (WORDNET)
  - Pr(s) = count(s, w) / count(w)
  - $\Pr(V_w^i / s) = \Pr(V_w^i, s) / \Pr(s)$

$$= c(V_w^i, s, w) / c(s, w)$$

Decision List Algorithm

Decision List Algorithm

- Based on 'One sense per collocation' property
  - Nearby words provide strong and consistent clues as to the sense of a target word
- Collect a large set of collocations for the ambiguous word
- Calculate word-sense probability distributions for all such collocations
- Calculate the log-likelihood ratio

$\log(P(\text{Sense-A}/\text{Collocation}_i) / P(\text{Sense-B}/\text{Collocation}_i))$

- Higher log-likelihood $\longrightarrow$ more predictive evidence
- Collocations are ordered in a decision list, with most predictive collocations ranked highest

Decision List Algorithm

Training Data

Sense	Training Examples (Keyword in Context)
A	used to strain microscopic *plant* life from the ...
A	... zonal distribution of *plant* life . ...
A	close-up studies of *plant* life and natural ...
A	too rapid growth of aquatic *plant* life in water ...
A	... the proliferation of *plant* and animal life ...
A	establishment phase of the *plant* virus life cycle ...
A	... ...
B	... ...
B	computer manufacturing *plant* and adjacent ...
B	discovered at a St. Louis *plant* manufacturing
B	... copper manufacturing *plant* found that they
B	copper wire manufacturing *plant* , for example ...
B	's cement manufacturing *plant* in Alpena ...
B	polystyrene manufacturing *plant* at its Dow ...
B	company manufacturing *plant* is in Orlando ...

Resultant Decision List

Final decision list for *plant* (abbreviated)

LogL	Collocation	Sense
10.12	*plant* growth	⇒ A
9.68	car (within ±*k* words)	⇒ B
9.64	*plant* height	⇒ A
9.61	union (within ±*k* words)	⇒ B
9.54	equipment (within ±*k* words)	⇒ B
9.51	assembly *plant*	⇒ B
9.50	nuclear *plant*	⇒ B
9.31	flower (within ±*k* words)	⇒ A
9.24	job (within ±*k* words)	⇒ B
9.03	fruit (within ±*k* words)	⇒ A
9.02	*plant* species	⇒ A
...	...	

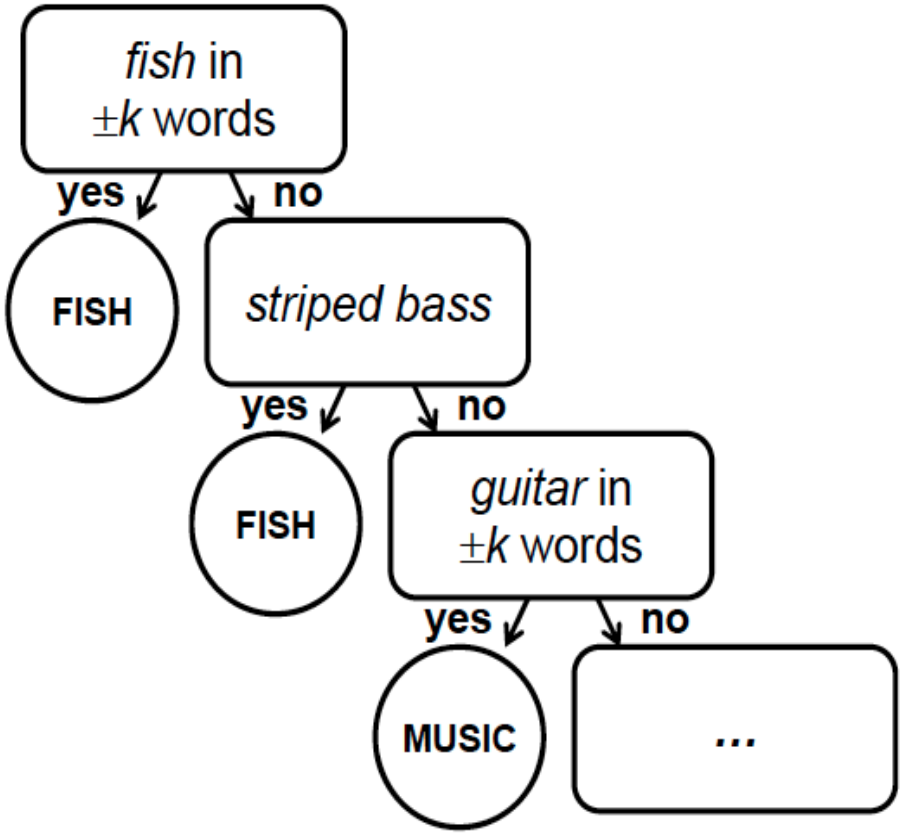
Sense-A: plant life, Sense-B: manufacturing plant

- Classification of a test sentence: **plucking flowers affects plant growth**
- Classification is based on the highest ranking collocation, found in the test sentences.

Decision List: Example

- Example: discriminating between bass (fish) and bass (music):

Context	Sense
fish in ±*k* words	FISH
striped bass	FISH
guitar in ±*k* words	MUSIC
bass player	MUSIC
piano in ±*k* words	MUSIC
sea bass	FISH
play bass	MUSIC
river in ±*k* words	FISH
on bass	MUSIC
bass are	FISH



Applications

- Text Classification
- Text Summarization
- Sentiment Analysis

Text Classification

- Assigning subject categories, topics, etc
- Spam detection
- Authorship identification
- Age/gender identification
- Language identification
- Text Summarization
- Sentiment analysis
- …

Text classification: problem definition

- Input:
  - A document d
  - A fixed set of classes $C = \{ c_1, c_2,\ldots,c_n\}$
- Output:
  - A predicted class $c \in C$

Classification Methods: Hand-coded rules

- Rules based on combinations of words or other features

Example:

Spam Classification

  - black-list-address OR (“dollars” AND “have been selected”)

Hand-coded rules: Pros and Cons

- Accuracy can be high if rules carefully refined by expert.
- But building and maintaining these rules is expensive.

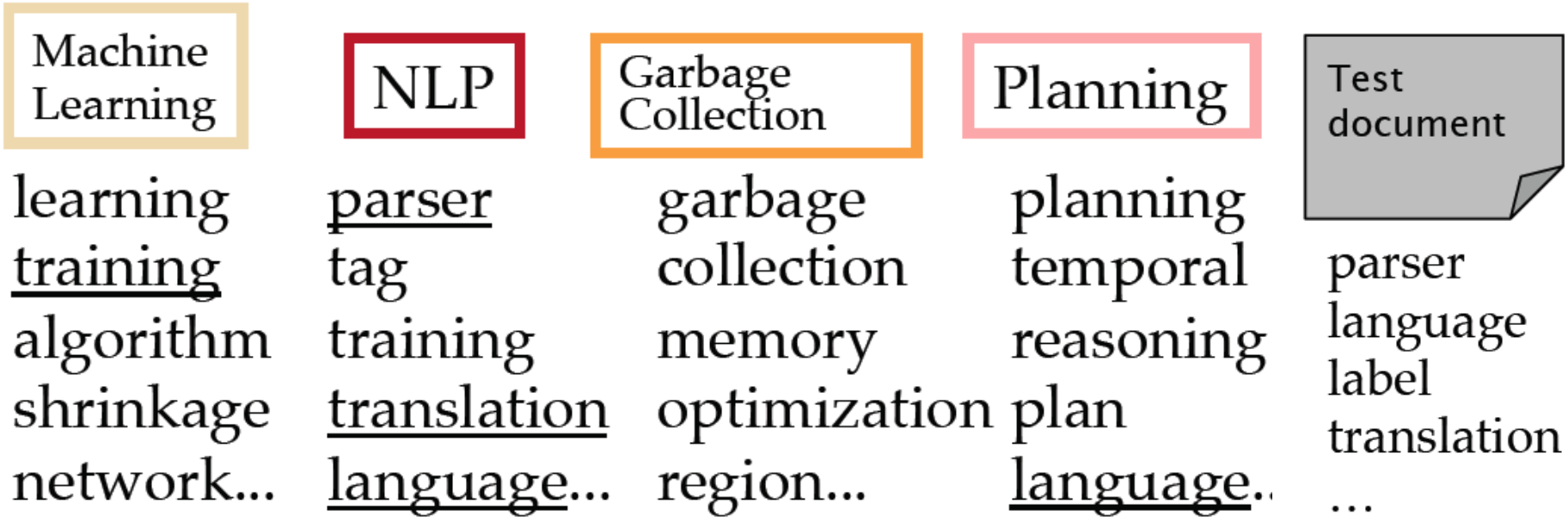
Classification Methods: Supervised Machine Learning

- Naïve Bayes
- Logistic regression
- Support-vector machines
- …

Naïve Bayes Intuition

- Simple classification method based on Bayes' rule
- Relies on very simple representation of document - Bag of words

Bag of words for document classification



Bayes' rule for documents and classes

- For a document ***d*** and a class ***c:***

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

- Naïve bayes Classifier:

$$c_{MAP} = \arg\max_{c \in C} P(c|d)$$

$$= \arg\max_{c \in C} P(d|c)P(c)$$

$$= \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n|c)P(c)$$

Naïve Bayes classification assumptions

$$P(x_1, x_2, \ldots, x_n | c)$$

Bag of words assumption:

- Assume that the position of a word in the document doesn't matter

Conditional Independence:

- Assume the feature probabilities $P(x_i/c_j)$ are independent given the class c_j.

$$P(x_1, x_2, \ldots, x_n | c) = P(x_1 | c) \cdot P(x_2 | c) \ldots P(x_n | c)$$

$$c_{NB} = \arg\max_{c \in C} P(c) \prod_{x \in X} P(x | c)$$

Learning the model parameters

Maximum Likelihood Estimate:

$$P(c_j) = \frac{doc\text{-}count(C = c_j)}{N_{doc}}$$

$$P(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

Learning the model parameters

Problem with MLE:

- Suppose in the training data, we haven't seen the word "fantastic", classified in the topic 'positive'.
  - P(fantastic|positive) = 0

$$c_{NB} = \arg\max_{c \in C} P(c) \prod_{x \in X} P(x|c)$$

Laplace (add-1) smoothing

$$P(w_i|c) = \frac{count(w_i, c) + 1}{(\sum_{w \in V} (count(w, c)) + |V|}$$

Performance Evaluation

- **Confusion Matrix**: True positive, false positive, true negative, false negative
- **Recall:** Fraction of docs in class i classified correctly.
- **Precision**: Fraction of docs assigned class i that are actually about class I
- **F-measure** : Harmonic mean of precision and recall
- **Accuracy** : Fraction of docs classified correctly