WSD
(Word Sense Disambiguation)

Sense Ambiguity:

Definition:

- Many words have several meanings or senses.
- Example:
  - striped **bass** in lake were too skinny
  - An electric guitar and **bass** player stand off
- The meaning of bass depends on context
- 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 fromm WordNet.

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 $\boldsymbol{w_i}$.

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

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

Limitations of (Lesk) 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's algorithm

- Original algorithm is sensitive towards exact words in the definition.
- Extension includes glosses of semantically related senses from WordNet (e.g. hypernyms, hyponyms, etc.).
- The scoring function becomes:

$$score_{ext}(S) = \sum_{s' \in rel(s) \text{ or } s \equiv s'} | context(w) \bigcap gloss(s') |$$

where,

- gloss(S) is the gloss of sense S from the lexical resource
- Context(W) is the gloss of each sense of each context word.
- rel(s) gives the senses related to s in WordNet under some relations.

The (extended) Lesk Algorithm

- A thesaurus-based measure that looks at **glosses**
- 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.
  - Paper and specially prepared for $1^2 + 2^2 = 5$
  - Compute overlap also for other relations
    - Glosses of hypernyms and hyponyms

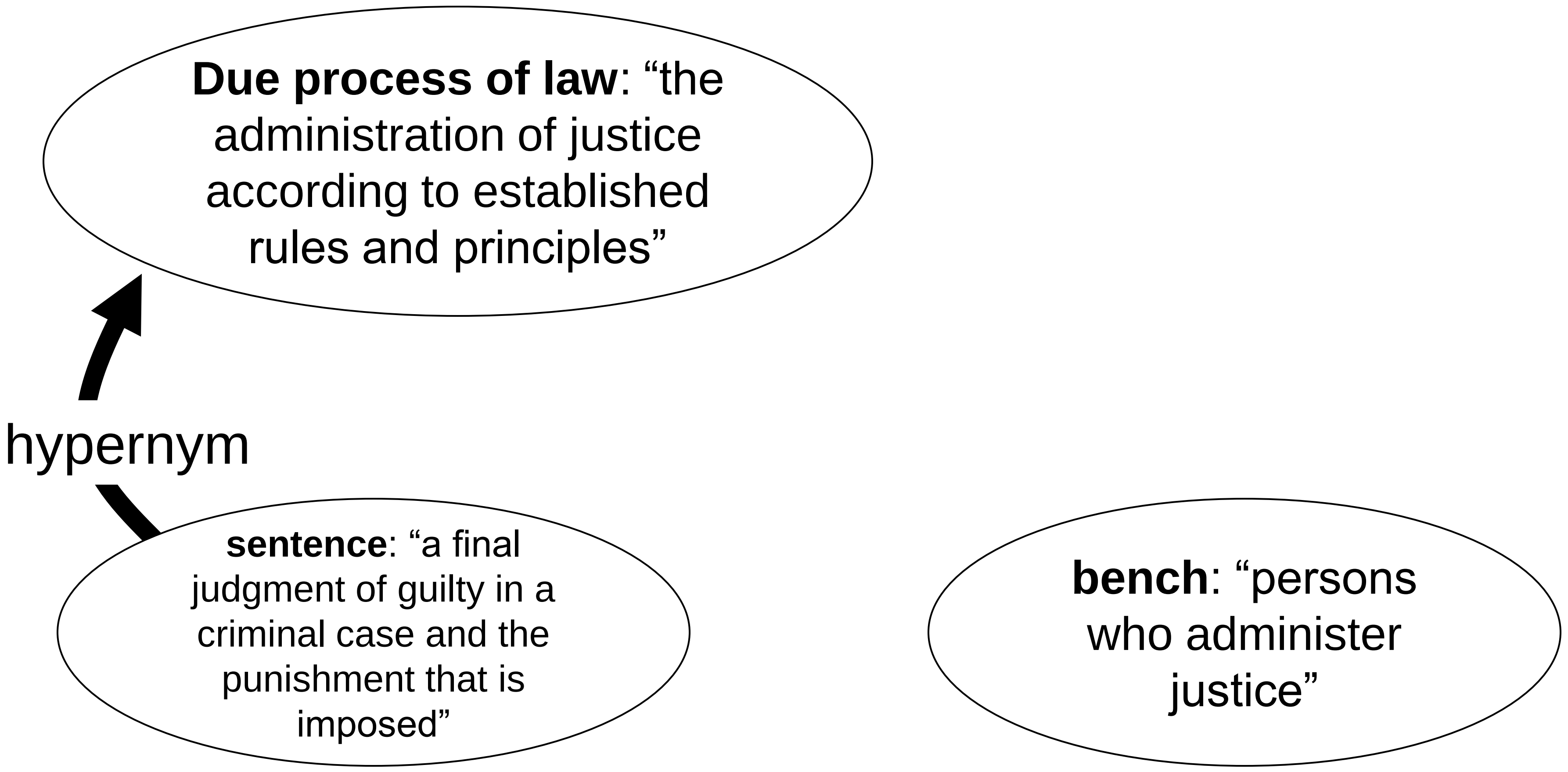
Extending a Gloss

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



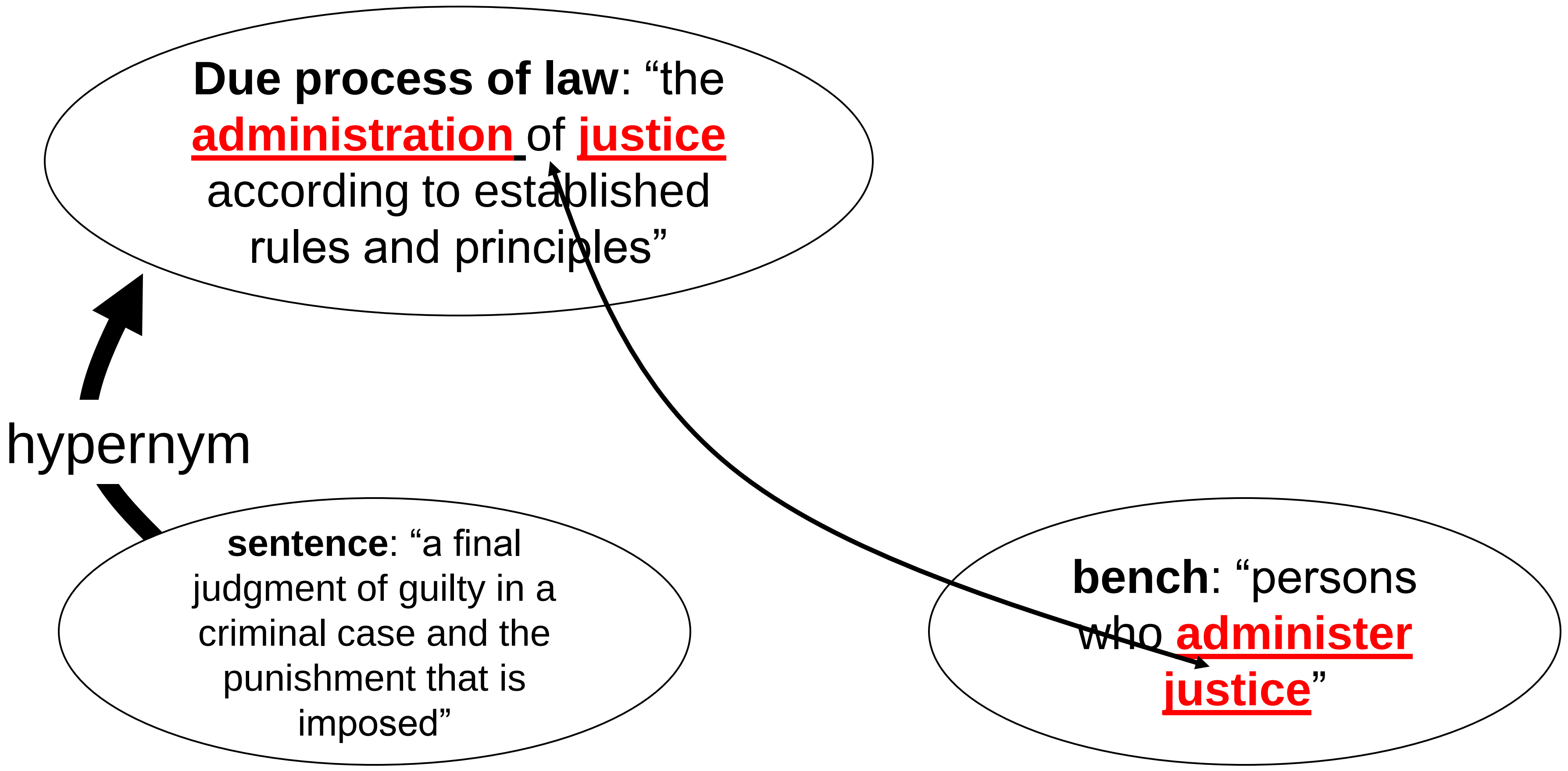
Due process of law: “the administration of justice according to established rules and principles”

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



Due process of law: "the **administration** of **justice** according to established rules and principles"

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

bench: "persons who **administer** **justice**"

# overlapped words = 2

Walkers Algorithm

- A thesaurus based approach

Step1: for **each sense** of the **target word** find the thesaurus category to which that sense belongs

Step2: calculate the score for each sense by using the context words

Note: a context word will add 1 to the score of the sentence if the thesaurus category of the word matches that of the sense

For example: Consider the sentence:

The money in the bank fetches an interest of 8% per annum

Target word: bank

Clue words from the context: money, fetch, interest, annum

	Sense1: Finance	**Sense2: location**
Money	+1	0
Interest	+1	0
Fetch	0	0
Annum	+1	0
Total	3	0

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

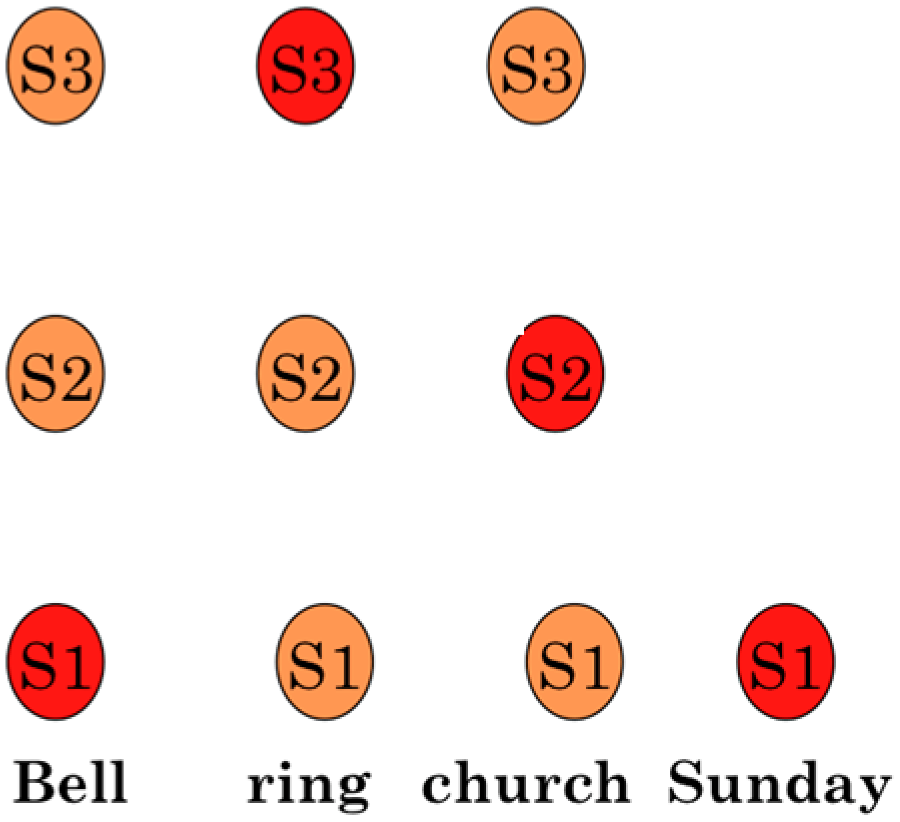
WSD USING RANDOM WALK ALGORITHM (Page Rank) (Sinha and Mihalcea, 2007)

WSD USING RANDOM WALK ALGORITHM (Page Rank)

Example: The Church bells no longer ring on Sundays

Consider 4 words: **Church, bells, ring, Sunday**

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

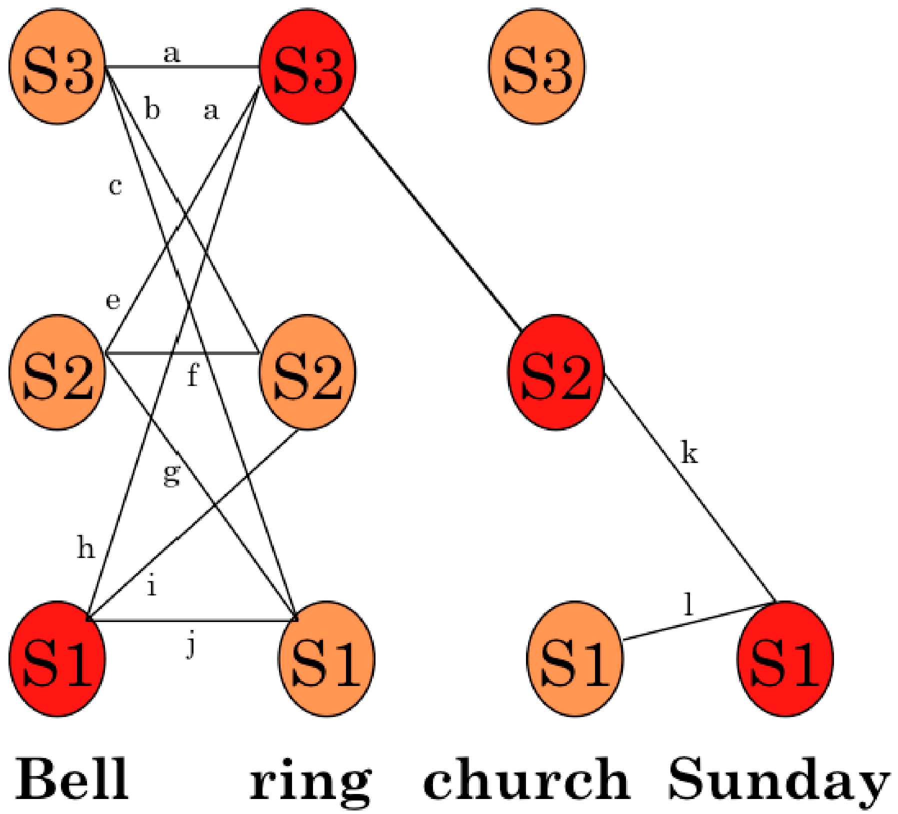


WSD USING RANDOM WALK ALGORITHM (Page Rank)

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
- 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.

Supervised Approach for WSD

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

Naïve Bayes

- The Algorithm find the winner sense using

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

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

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

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

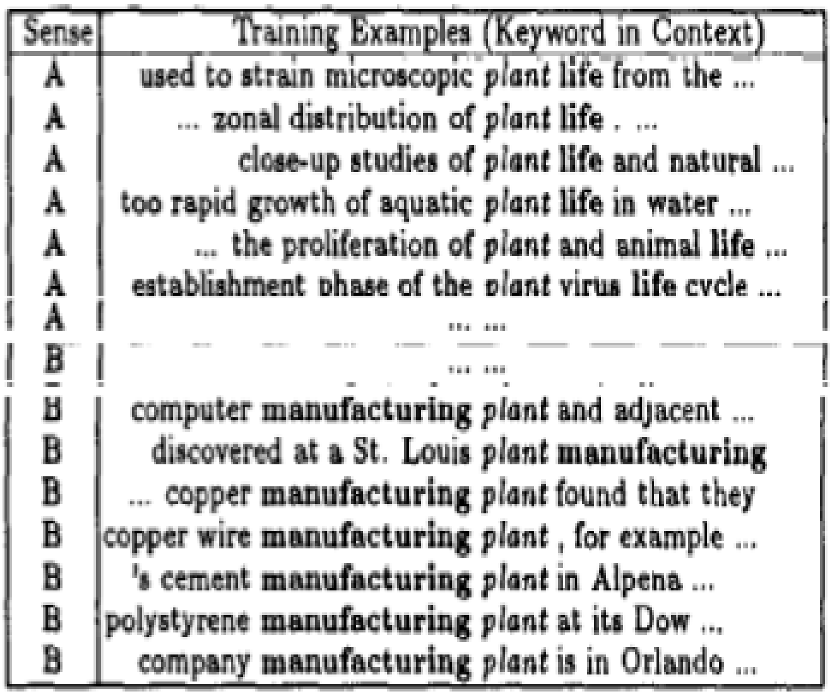
- Based on ‘One sense per collocation’ property
  - Nearby words provide strong and consistent clues as to the sense of a target word
- Collect a large set of collocations for the ambiguous word
- Calculate word-sense probability distributions for all such collocations
- Calculate the log-likelihood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

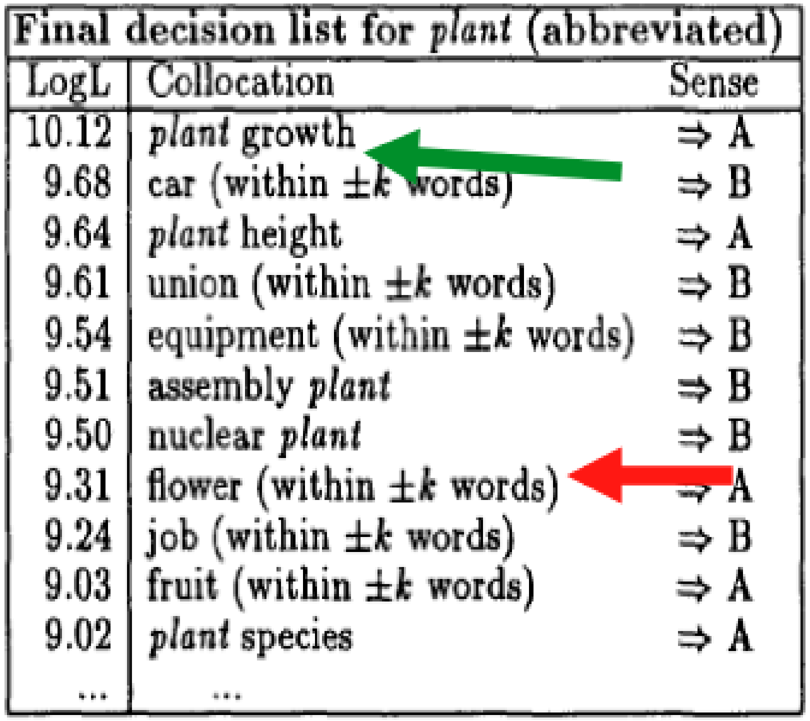
Decision List Algorithm

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 $\pm k$ words)	⇒ B
9.64	*plant* height	⇒ A
9.61	union (within $\pm k$ words)	⇒ B
9.54	equipment (within $\pm k$ words)	⇒ B
9.51	assembly *plant*	⇒ B
9.50	nuclear *plant*	⇒ B
9.31	flower (within $\pm k$ words)	⇒ A
9.24	job (within $\pm k$ words)	⇒ B
9.03	fruit (within $\pm k$ words)	⇒ A
9.02	*plant* species	⇒ A
...	...	

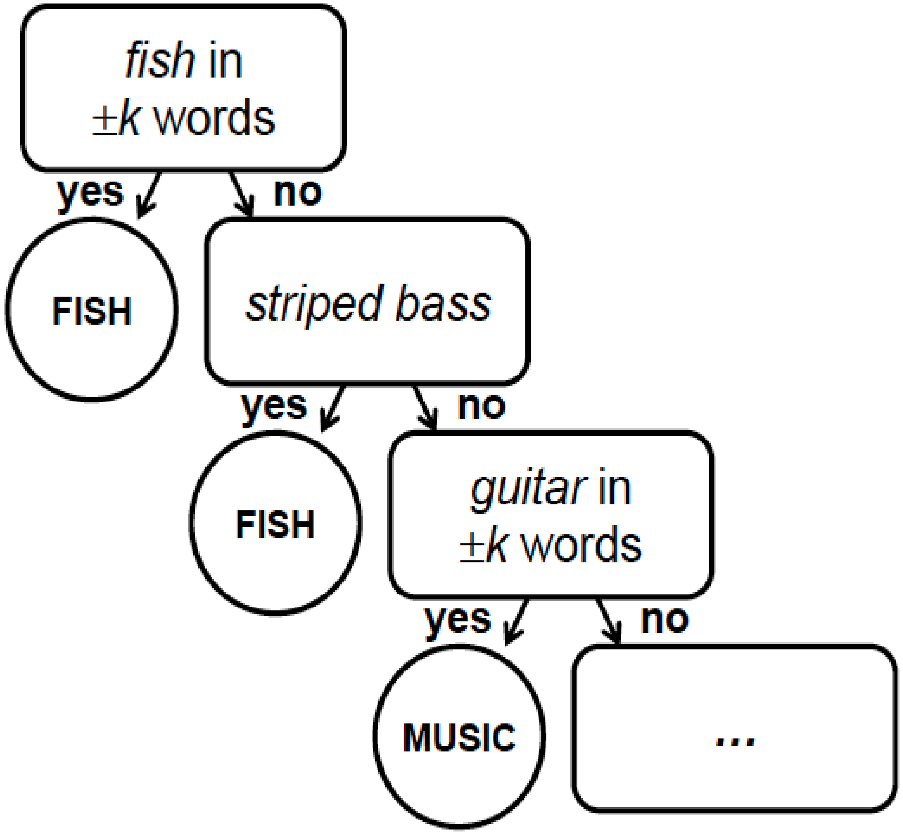
Sense-A: plant life, Sense-B: manufacturing plant

- 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


fish in ±*k* words
yes
no
FISH
striped bass
yes
no
FISH
guitar in ±*k* words
yes
no
MUSIC
...