11 - Computational Lexical Semantics

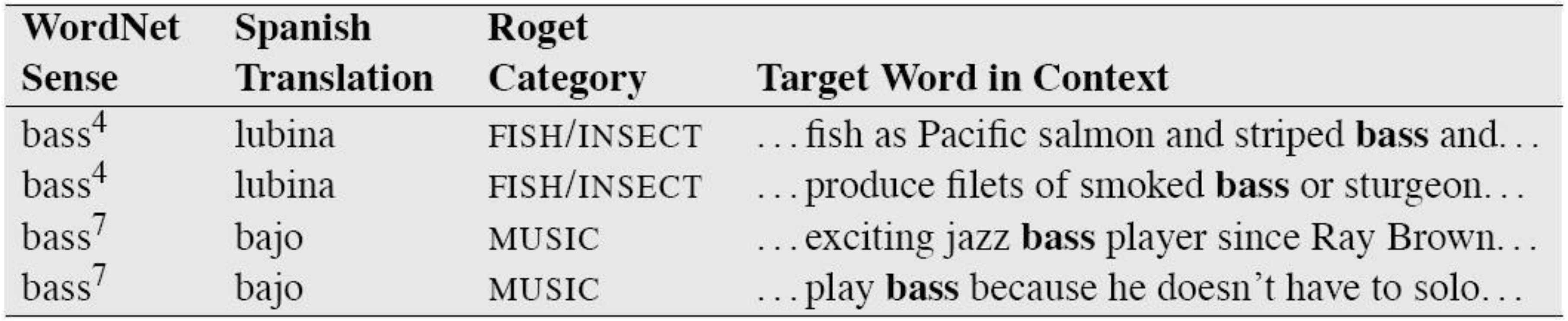
CS 6320

Outline

- Word Sense Disambiguation
- Word Similarity
- Semantic Role Labeling

Word Sense Disambiguation

- WSD is the task of selecting the correct sense for a word
- Applications: machine translation, question answering, information retrieval, text classification
- Baseline: use the most frequently used sense



WordNet Sense	Spanish Translation	Roget Category	Target Word in Context
bass^4	lubina	FISH/INSECT	... fish as Pacific salmon and striped **bass** and...
bass^4	lubina	FISH/INSECT	... produce filets of smoked **bass** or sturgeon...
bass^7	bajo	MUSIC	... exciting jazz **bass** player since Ray Brown...
bass^7	bajo	MUSIC	... play **bass** because he doesn't have to solo...

Supervised WSD

- ML can be applied to WSD
- Features:
    - Collocational features
    - Bag-of-words features

An electric guitar and bass player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps.

Collocational

$$[w_{i-2}, POS_{i-2}, w_{i-1}, POS_{i-1}, w_{i+1}, POS_{i+1}, w_{i+2}, POS_{i+2}]$$

Bag-of-words

[fishing, big, sound, player, fly, rod, pound, double, runs, playing, guitar, band]

[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0]

Naïve Bayes Classifier

- Select the sense of the word that best matches features vector *f*

$$\hat{s} = \underset{s \in S}{\operatorname{argmax}} P\,(s|\vec{f})$$

$$\hat{s} = \underset{s \in S}{\operatorname{argmax}} \frac{P(\vec{f}|s\,)P(s)}{P(\vec{f})}$$

- Assumption: naively assume features are independent of each other

$$P(\vec{f}|s) \approx \prod_{j=1}^{n} P\,(f_j|s)$$

Naïve Bayes Classifier

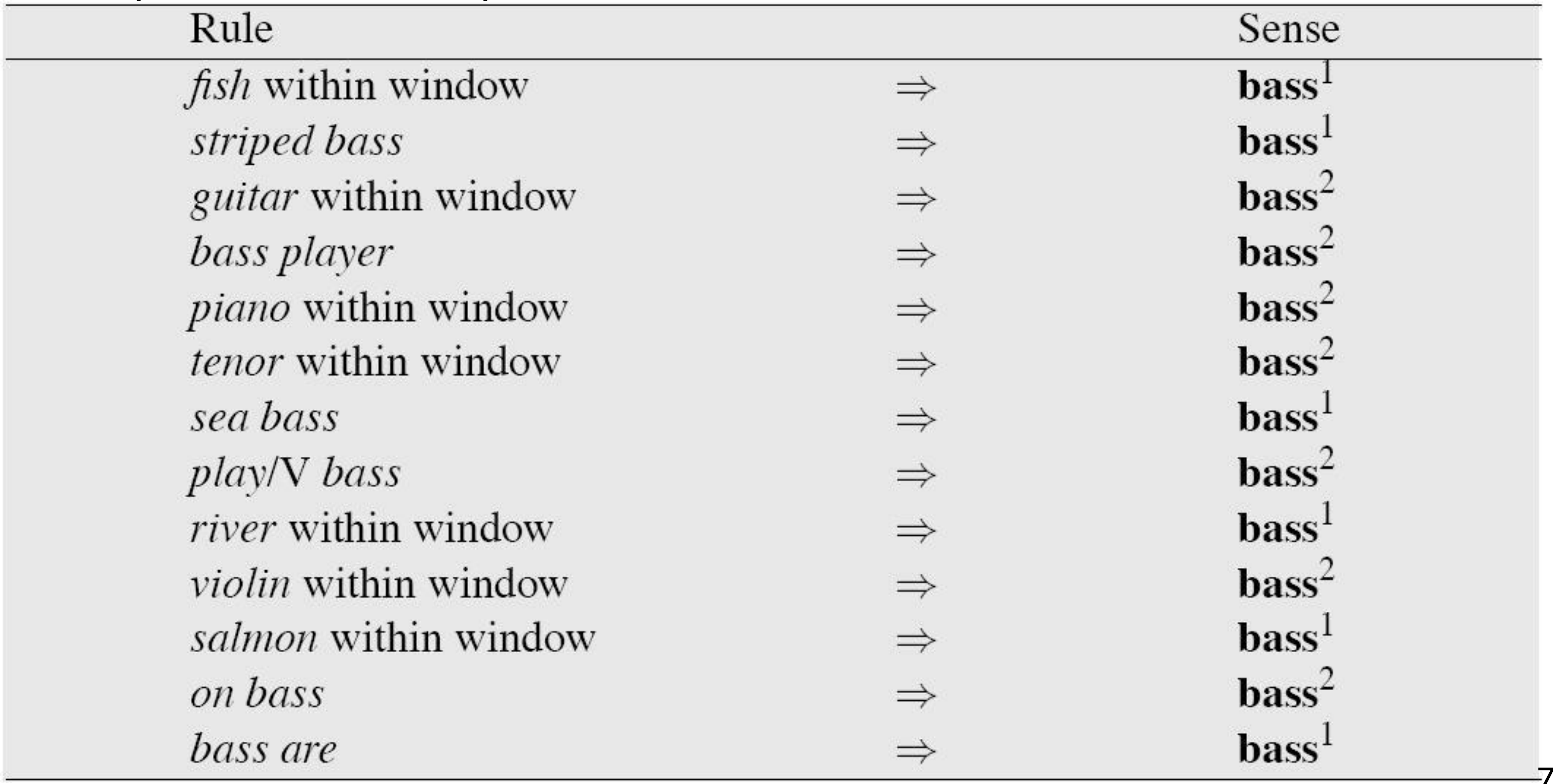
$$\hat{s} = \operatorname*{argmax}_{s \in S} P(s) \prod_{j=1}^{n} P(f_j|s)$$

$$P(s_i) = \frac{count(s_i, w_j)}{count(w_j)}$$

$$P(f_j|s) = \frac{count(f_j, s)}{count(s)}$$

Decision List Classifier

Decision trees are also used and are easier to understand
A sequence of tests are performed.



Rule		Sense
fish within window	$\Rightarrow$	**bass**^{1}
striped bass	$\Rightarrow$	**bass**^{1}
guitar within window	$\Rightarrow$	**bass**^{2}
bass player	$\Rightarrow$	**bass**^{2}
piano within window	$\Rightarrow$	**bass**^{2}
tenor within window	$\Rightarrow$	**bass**^{2}
sea bass	$\Rightarrow$	**bass**^{1}
play/V bass	$\Rightarrow$	**bass**^{2}
river within window	$\Rightarrow$	**bass**^{1}
violin within window	$\Rightarrow$	**bass**^{2}
salmon within window	$\Rightarrow$	**bass**^{1}
on bass	$\Rightarrow$	**bass**^{2}
bass are	$\Rightarrow$	**bass**^{1}

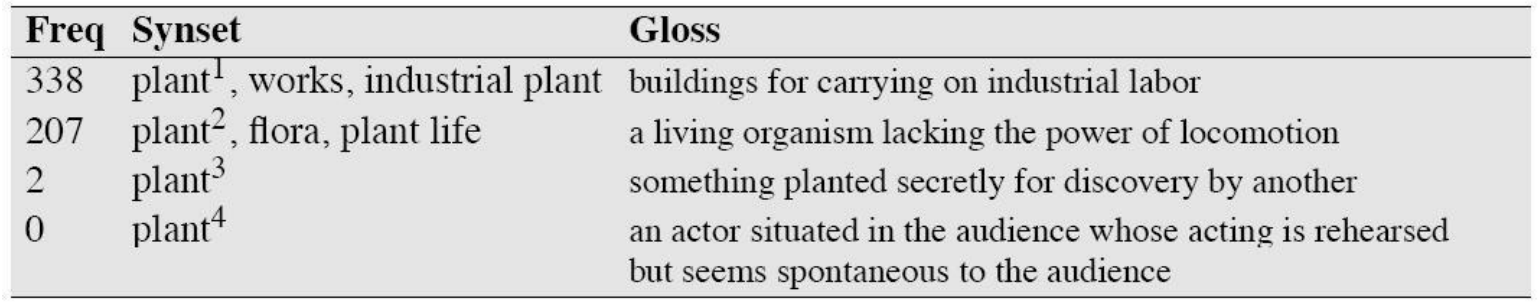
Decision list Classifier

The ratio between the probabilities of the two senses is an indication how discriminative a feature is between senses

$$\left|\log\left(\frac{P(Sense_1|f_i)}{P(Sense_2|f_i)}\right)\right|$$

WSD Evaluation

Baseline most frequently used sense

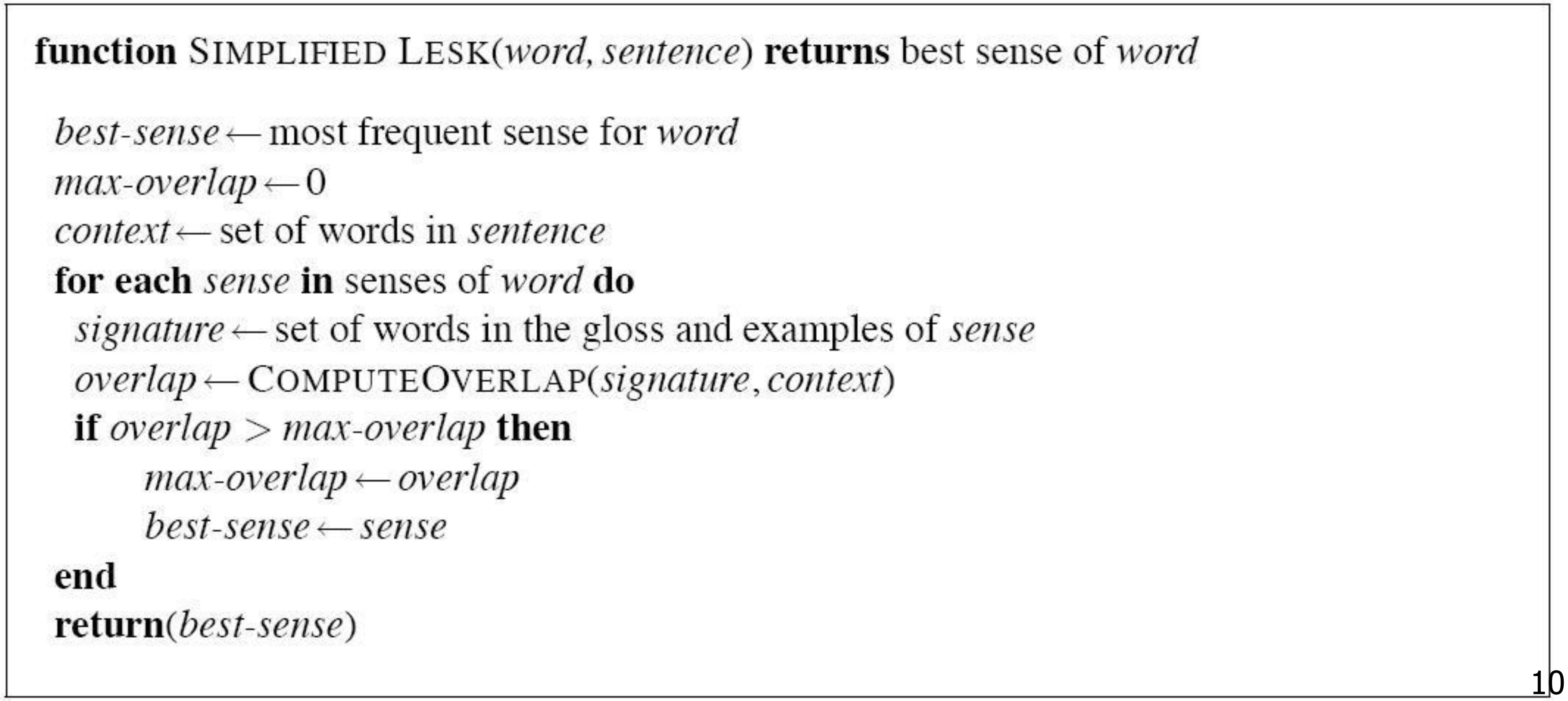


Freq	Synset	Gloss
338	plant^1, works, industrial plant	buildings for carrying on industrial labor
207	plant^2, flora, plant life	a living organism lacking the power of locomotion
2	plant^3	something planted secretly for discovery by another
0	plant^4	an actor situated in the audience whose acting is rehearsed but seems spontaneous to the audience

- Fine grain vs course grain WSD
- Evaluation method: check against humanly annotated data

Lesk Algorithm

- Supervised methods fail for words not in training data
- Use dictionary or thesaurus as indirect kind of supervision. Choose the sense whose gloss shares the most words with target word neighborhood

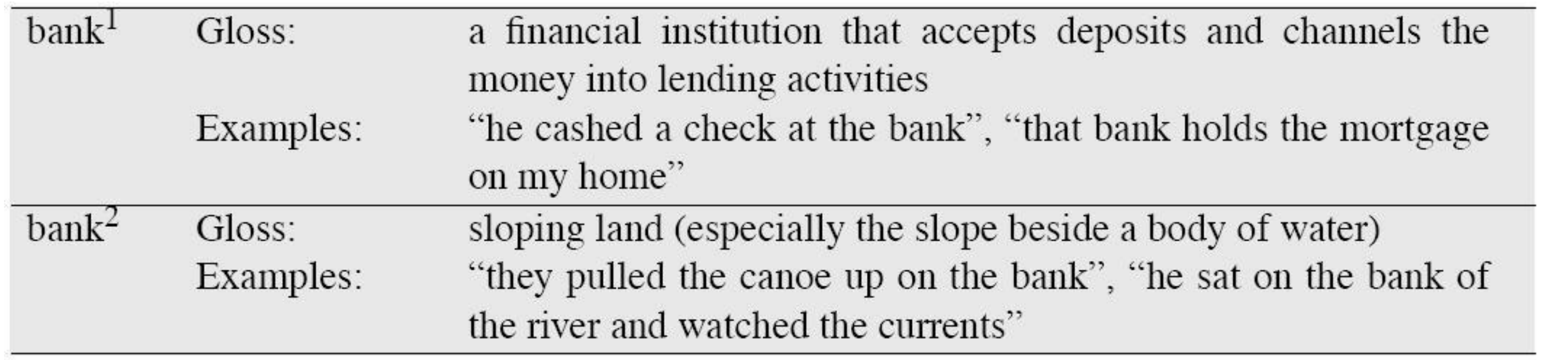


```
function SIMPLIFIED LESK(word, sentence) returns best sense of word

 best-sense ← most frequent sense for word
 max-overlap ← 0
 context ← set of words in sentence
 for each sense in senses of word do
   signature ← set of words in the gloss and examples of sense
   overlap ← COMPUTEOVERLAP(signature, context)
   if overlap > max-overlap then
         max-overlap ← overlap
         best-sense ← sense
 end
 return(best-sense)
```

Lesk Algorithm

The bank can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.



bank[1]	Gloss:	a financial institution that accepts deposits and channels the money into lending activities
	Examples:	"he cashed a check at the bank", "that bank holds the mortgage on my home"
bank[2]	Gloss:	sloping land (especially the slope beside a body of water)
	Examples:	"they pulled the canoe up on the bank", "he sat on the bank of the river and watched the currents"

bank #1 - 2 content words overlap
bank #2 - 0 content words overlap

Pick bank # 1

Selectional Restrictions and Preferences

Improve Lesk Algorithm

- Main problem with Lesk algorithm is the small number of words in gloss definitions

- Possible improvements:

1. Include related words, i.e. hyponyms
2. Apply a weight to each overlapping word

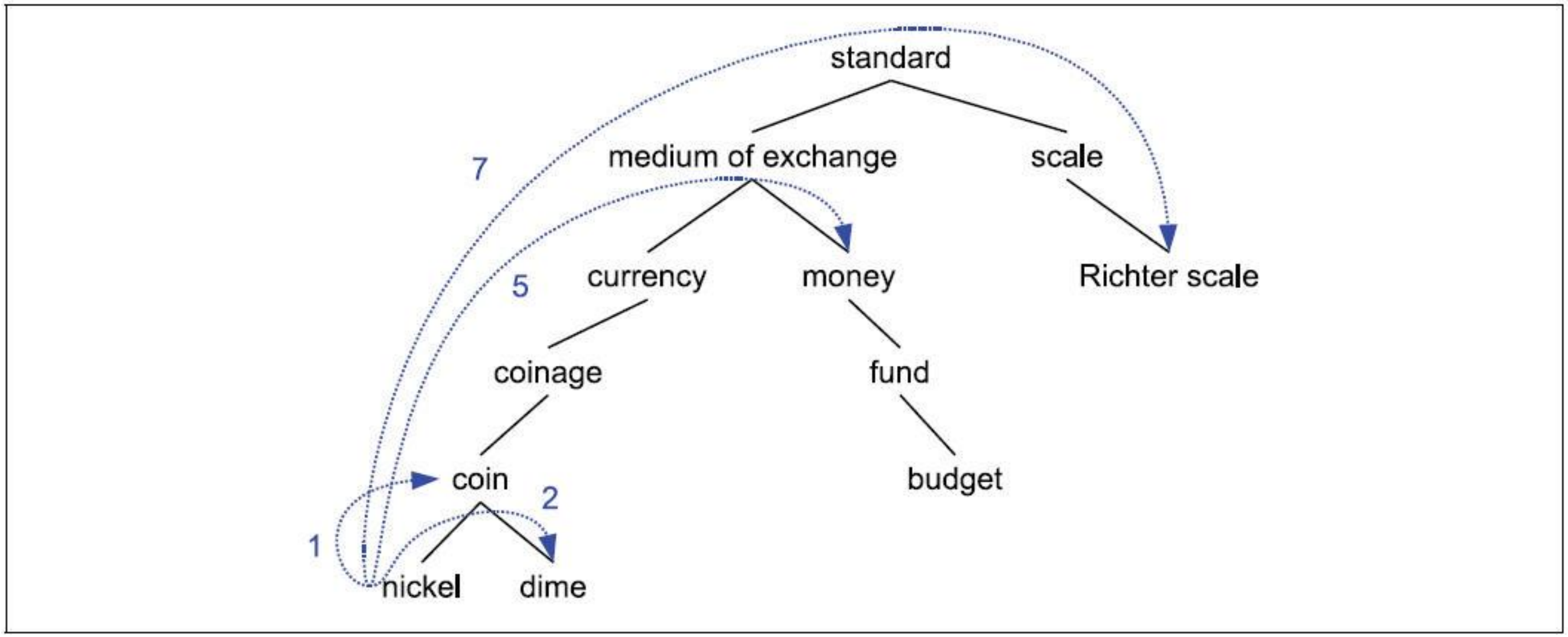
$$\mathrm{idf}_i = \log\left(\frac{Ndoc}{nd_i}\right)$$

where: *Ndoc* is the number of documents in a corpus
ndi is the number of documents in corpus where word *i* occurs

Word Similarity

- Two words are more similar if they share more features of meaning.
- The more similar two words are the less semantic distance between them, the less similar the greater the semantic distance between them.
- Word similarity useful in information retrieval, QA, MT, etc.
- Word similarity vs word relatedness.

Word Similarity on WN



Word Similarity

$$\mathrm{sim}_{\mathrm{path}}(c_1, c_2) = -\log \mathrm{pathlen}(c_1, c_2)$$

pathlen(c_1, c_2) =number of edges the shortest path in thesaurus graph between synsets c_1, c_2

$$wordsim(w_1, w_2) = \max_{\substack{c_1 \in senses(w_1) \\ c_2 \in senses(w_2)}} sim(c_1, c_2)$$

Word Similarity

Define P(c) – the probability that a randomly selected word in a corpus is an instance of concept c

$$P(c) = \frac{\sum_{w \in \text{words}(c)} count(w)}{N}$$

where words(c) set of words in corpus that are present in the thesaurus

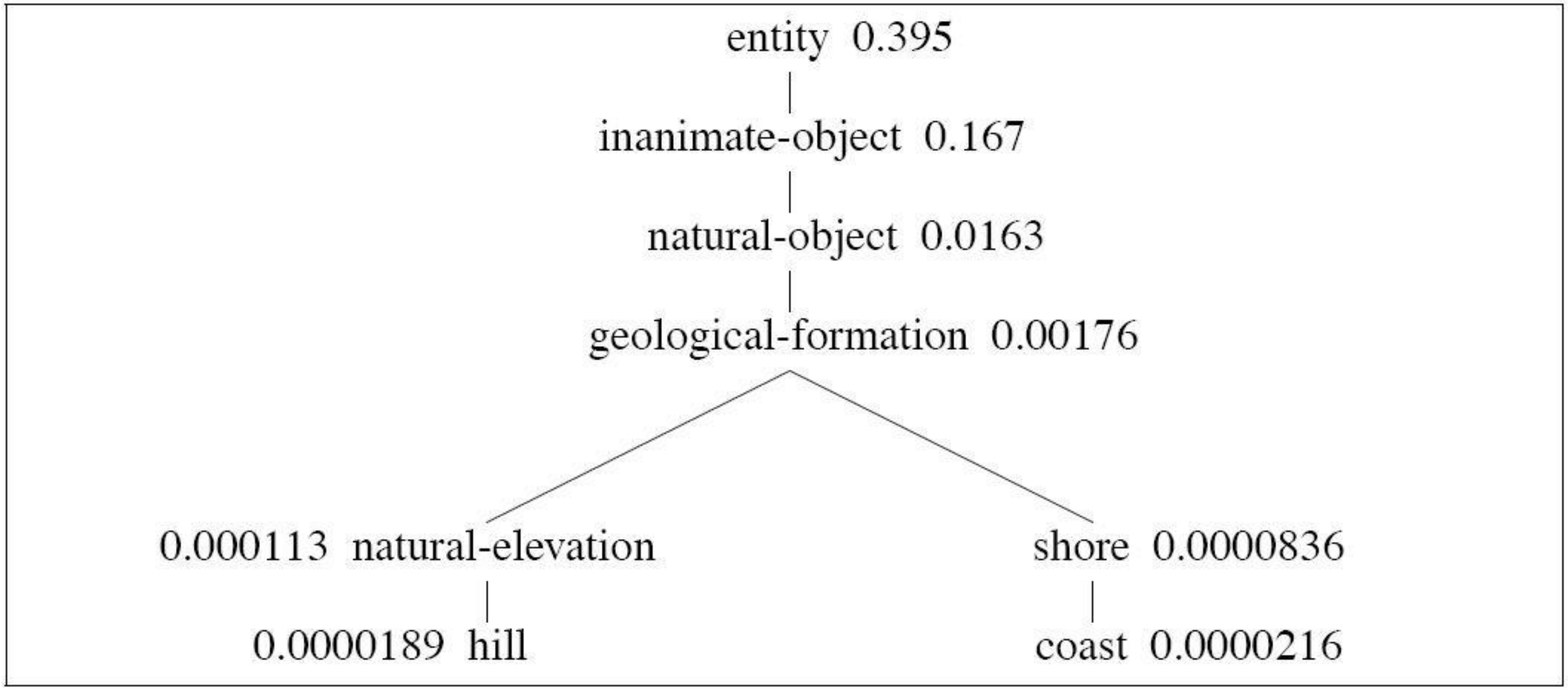
From information theory, use the definition of Information Content IC of concept c

$$\text{IC}(c) = -\log P(c)$$

Then, define LCS – lowest common subsumer of two concepts

LCS (c1, c2) = lowest node in the hierarchy that subsumes both c1 and c2

Word Similarity



Word Similarity

Resnik similarity – think of similarity between words as related to their common information

$$\mathrm{sim}_{\mathrm{resnik}}(c_1, c_2) = -\log P(LCS(c_1, c_2))$$

Lin similarity – measures the commonality and difference between two words A and B

commonality

$$\mathrm{IC}(\mathrm{common}(\mathrm{A}, \mathrm{B}))$$

difference

$$\mathrm{IC}(\mathrm{description}(\mathrm{A}, \mathrm{B})) - \mathrm{IC}(\mathrm{common}(\mathrm{A}, \mathrm{B}))$$

where description(A,B) describes A and B

Word Similarity

$$\mathrm{sim}_{\mathrm{Lin}}(A,B) = \frac{\mathrm{common}(A,B)}{\mathrm{description}(A,B)}$$

The information in common between two concepts is twice the information in their LCS(c1,c2)

$$\mathrm{sim}_{\mathrm{Lin}}(c_1,c_2) = \frac{2 \text{ x} \log P(\mathrm{LCS}(c_1,c_2))}{\log P(c_1) + \log P(c_2)}$$

$$\mathrm{sim}_{\mathrm{Lin}}(c_1,c_2) = \frac{2 \text{ x} \log P(\mathrm{geological\_formation})}{\log P(\mathrm{hill}) + \log P(\mathrm{coast})} = 0.59$$

Jiang-Conrath distance is similar

$$\mathrm{dist}_{\mathrm{JC}}(c_1,c_2) = 2 \text{ x} \log \mathrm{P}\big(\mathrm{LCS}(c_1,c_2)\big) - (\log P(c_1) + \log P\ (c_2))$$

Word Similarity

Lesk method – dictionary based – overlapping words and phrases in glosses

drawing paper – paper that is specially prepared for us in drafting.

decal – the art of transferring designs from specially prepared paper to a wood or _ _ _.

Score: $1^2 + 2^2$ =5

Lesk similarity – gloss overlap plus related glosses overlap

$$sim_{eLesk}(c_1, c_2) = \sum_{r,q \in \mathrm{RELS}} \mathrm{overlap}(\mathrm{gloss}(r(c_1)), \mathrm{gloss}(q(c_2)))$$

Word Similarity - Summary

$$
\begin{aligned}
\mathrm{sim}_{\mathrm{path}}(c_1,c_2) &= -\log \mathrm{pathlen}(c_1,c_2) \\
\mathrm{sim}_{\mathrm{Resnik}}(c_1,c_2) &= -\log P(\mathrm{LCS}(c_1,c_2)) \\
\mathrm{sim}_{\mathrm{Lin}}(c_1,c_2) &= \frac{2\times\log P(LCS(c_1,c_2))}{\log P(c_1)+\log P(c_2)} \\
\mathrm{sim}_{\mathrm{jc}}(c_1,c_2) &= \frac{1}{2\times\log P(\mathrm{LCS}(c_1,c_2))-(\log P(c_1)+\log P(c_2))} \\
\mathrm{sim}_{\mathrm{eLesk}}(c_1,c_2) &= \sum_{r,q\in\mathrm{RELS}} \mathrm{overlap}(\mathrm{gloss}(r(c_1)),\mathrm{gloss}(q(c_2)))
\end{aligned}
$$

Word Similarity: Distributional Methods

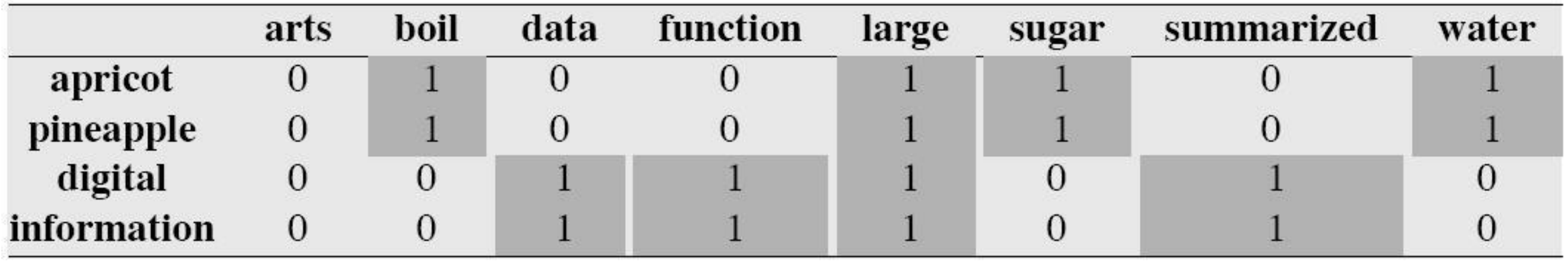
- Problem- Thesauruses with hierarchies do not exist for every language.
- Idea – use corpora to compute concept relatedness.

A bottle of *tezguino* is on the table.
Everybody likes *tezguino* .
Tezguino makes you drunk*.*
We make *tezguino* out of corn.

Word co-occurrence vector

- Represent the meaning of word w as feature vector
- Then use vector distance measures
- Co-occurrence vectors for 4 words

$$\bar{w} = (f_1, f_2, \cdots, f_n)$$



	arts	boil	data	function	large	sugar	summarized	water
apricot	0	1	0	0	1	1	0	1
pineapple	0	1	0	0	1	1	0	1
digital	0	0	1	1	1	0	1	0
information	0	0	1	1	1	0	1	0

Word co-occurrence vector

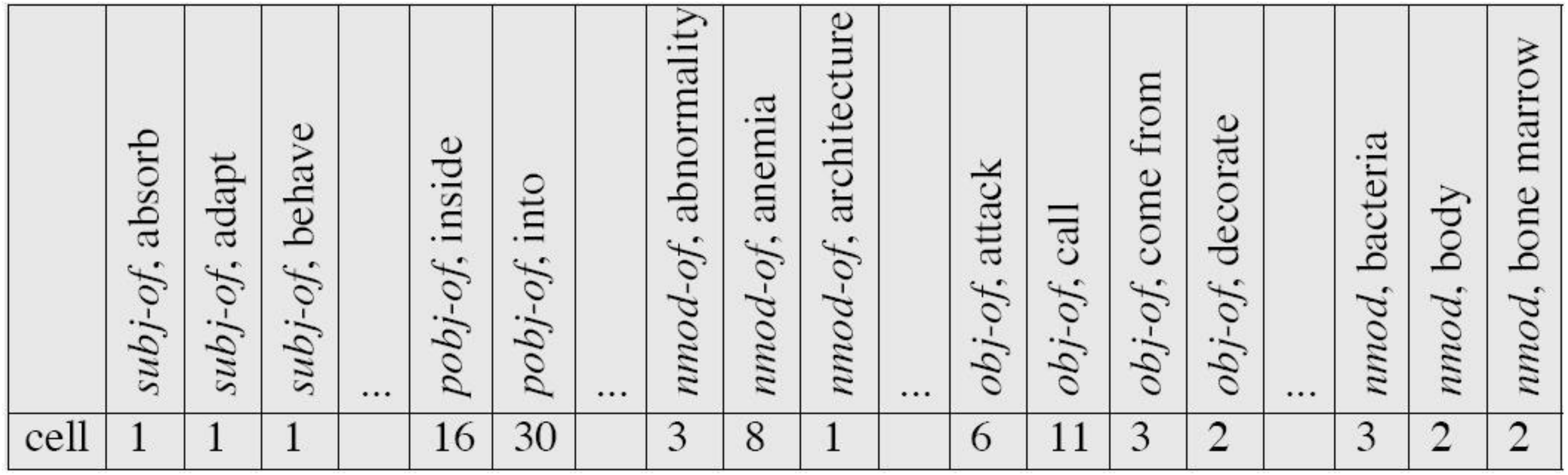
Hindle’s idea: choose words that occur in some grammatical relation to target words.

I discovered dried tangerines:

discover(subject I)	I (subj-of discover)
tangerine (obj-of discover)	tangerine (adj-mod dried)
dried (adj-mod-of tangerine)	

Word co-occurrence vector

- Co-occurrence vector for the word *cell*



	subj-of, absorb	*subj-of*, adapt	*subj-of*, behave	…	*pobj-of*, inside	*pobj-of*, into	…	*nmod-of*, abnormality	*nmod-of*, anemia	*nmod-of*, architecture	…	*obj-of*, attack	*obj-of*, call	*obj-of*, come from	*obj-of*, decorate	…	*nmod*, bacteria	*nmod*, body	*nmod*, bone marrow
cell	1	1	1		16	30		3	8	1		6	11	3	2		3	2	2

Measuring Association with Context

- Assign values or weights to features to better measure the association between a target word *w* and feature *f*.
- Use probabilities to measure association.

$$P(f|w)=\frac{\text{count}(f,w)}{\text{count}(w)}$$

$$P(f,w)=\frac{\text{count}(f,w)}{\sum_w \text{count}(w^j)}$$

$$\text{assoc}_{\text{prob}}(w,f) = P(f|w)$$

Association

- Mutual information between two random variables X and Y.

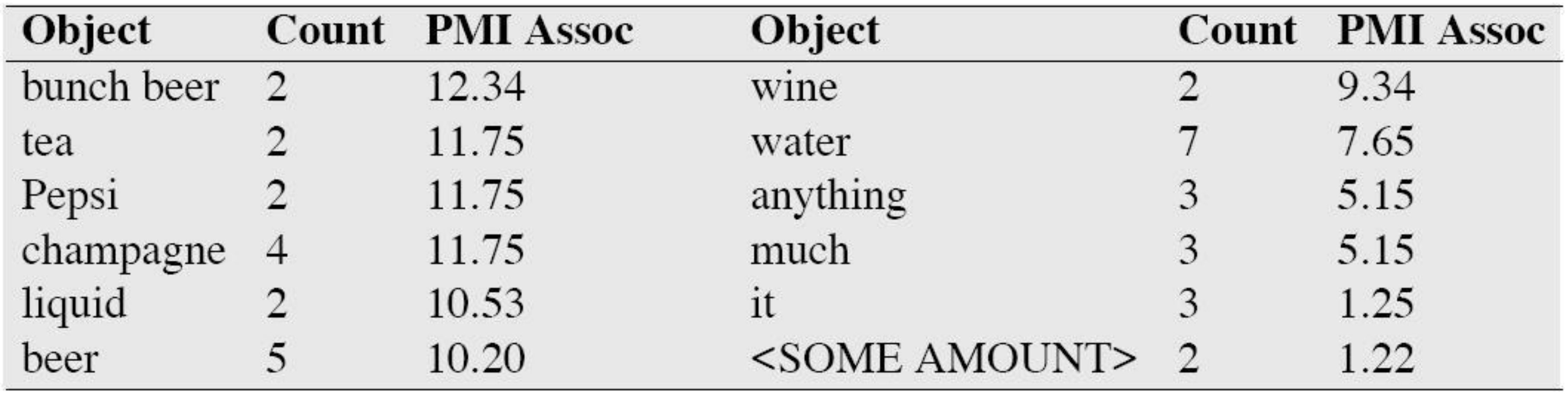
$$I(X,Y) = \sum_x \sum_y P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- Pointwise mutual information – a measure of how often two events x and y occur, compared to what we expect if they were independent.

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

$$\text{assoc}_{\text{PMI}}(w,f) = \log_2 \frac{P(w,f)}{P(w)P(f)}$$

Association



Object	Count	PMI Assoc	Object	Count	PMI Assoc
bunch beer	2	12.34	wine	2	9.34
tea	2	11.75	water	7	7.65
Pepsi	2	11.75	anything	3	5.15
champagne	4	11.75	much	3	5.15
liquid	2	10.53	it	3	1.25
beer	5	10.20	<SOME AMOUNT>	2	1.22

Lin Association – breaks $P(f)$ further down into relation r and word w' – at the other end of relation r.

$$\text{assoc}_{\text{Lin}}(w,f) = \log_2 \frac{P(w,f)}{P(w)P(r|w)P(w'|w)}$$

Association

- t-test - association – measures how much more frequent the association is than chance.
- t-test computes the difference between observed and expected mean normalized by variance.

$$t = \frac{\bar{x} - \mu}{\sqrt{\frac{S^2}{N}}}$$

- Variance approximated by the expected probability product.

$$\text{assoc}_{\text{t-test}}(w, f) = \frac{P(w, f) - P(w)P(f)}{\sqrt{P(f)P(w)}}$$

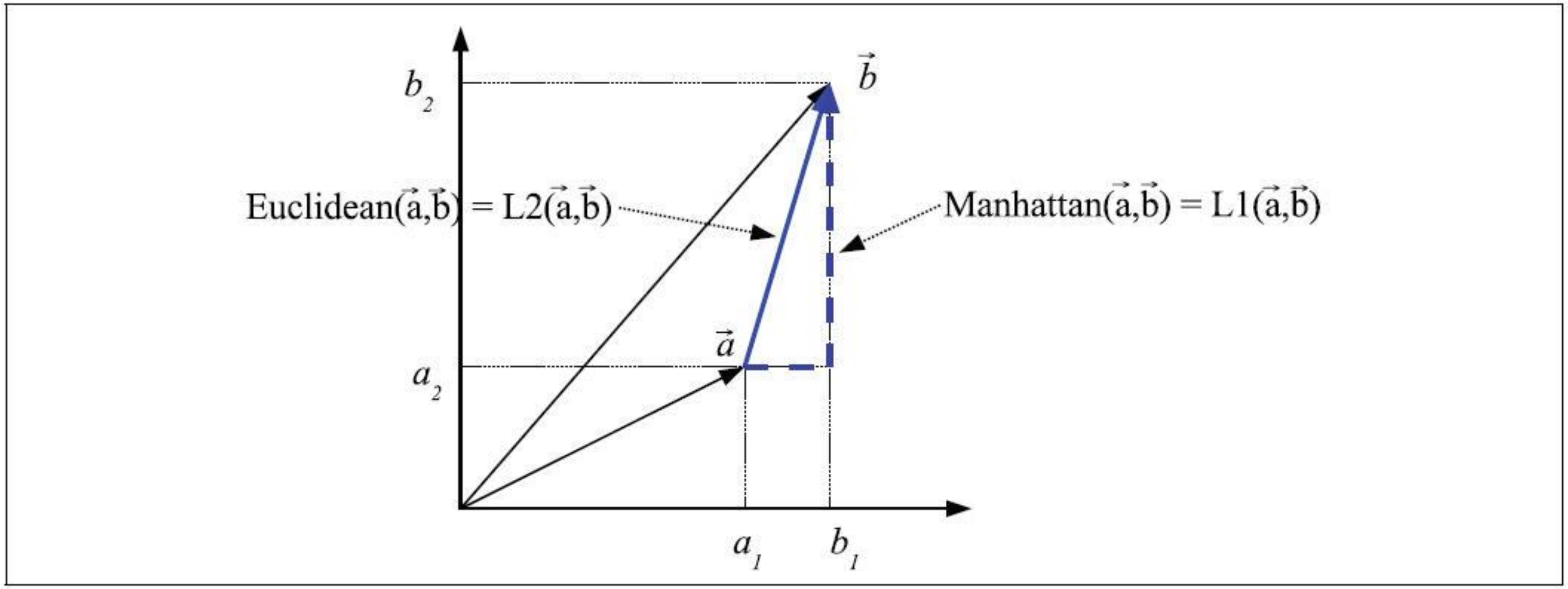
Similarity Between two vectors

- So far we have computed co-occurrence vector for a target word. This gives a distributional definition of the meaning of a target word.

$$\text{distance}_{\text{manhattan}}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} |x_i - y_i|$$

$$\text{distance}_{\text{euclidean}}(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$

Similarity Between two vectors


b_2
$\vec{b}$
$\text{Euclidean}(\vec{a},\vec{b}) = \text{L2}(\vec{a},\vec{b})$
$\text{Manhattan}(\vec{a},\vec{b}) = \text{L1}(\vec{a},\vec{b})$
$\vec{a}$
a_2
a_1
b_1

Information Retrieval Word Similarity

$$\text{sim}_{\text{dot-product}}(\vec{v}, \vec{w}) = \vec{v}, \vec{w} = \sum_{i=1}^{N} v_i \text{ x } w_i$$

- Define a vector for a target word with N features $f_1, \cdots, f_N$.

$$\vec{w} = (\text{assoc}(w, f_1), \text{assoc}(w, f_2), \text{assoc}(w, f_3), \dots, \text{assoc}(w, f_N))$$

- Problem: long vectors are favored. Need to normalize by vector length.

$$|\vec{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

Information Retrieval Word Similarity

$$\mathrm{sim}_{\mathrm{cosine}}(\vec{v},\vec{w}) = \frac{\vec{v},\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \mathrm{x}\, w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

$$\mathrm{sim}_{\mathrm{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i, w_i)}{\sum_{i=1}^{N} \max(v_i, w_i)}$$

$$\mathrm{sim}_{\mathrm{Dice}}(\vec{v},\vec{w}) = \frac{2\,\mathrm{x} \sum_{i=1}^{N} \min(v_i, w_i)}{\sum_{i=1}^{N} (v_i, w_i)}$$

Word Similarity

$$\text{assoc}_{\text{prob}}(w,f) = P(f|w) \qquad (20.35)$$

$$\text{assoc}_{\text{PMI}}(w,f) = \log_2 \frac{P(w,f)}{P(w)P(f)} \qquad (20.38)$$

$$\text{assoc}_{\text{Lin}}(w,f) = \log_2 \frac{P(w,f)}{P(w)P(r|w)P(w'|w)} \qquad (20.39)$$

$$\text{assoc}_{\text{t-test}}(w,f) = \frac{P(w,f)-P(w)P(f)}{\sqrt{P(f)P(w)}} \qquad (20.41)$$

$$\text{sim}_{\text{cosine}}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}} \qquad (20.47)$$

$$\text{sim}_{\text{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} \max(v_i,w_i)} \qquad (20.48)$$

$$\text{sim}_{\text{Dice}}(\vec{v},\vec{w}) = \frac{2 \times \sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} (v_i+w_i)} \qquad (20.49)$$

$$\text{sim}_{\text{JS}}(\vec{v}||\vec{w}) = D(\vec{v}|\tfrac{\vec{v}+\vec{w}}{2}) + D(\vec{w}|\tfrac{\vec{v}+\vec{w}}{2}) \qquad (20.52)$$

Semantic Role Labeling

- SRL – is the task of finding semantic roles for each predicate.

- FrameNet

[You]	can't [blame]	[the program]	[for being unable to identify it]
COGNIZER	TARGET	EVALUEE	REASON

- PropBank

[The San Francisco Examiner]	issued	[a special edition]	[yesterday]
ARG0	TARGET	ARG1	ARGM-TMP

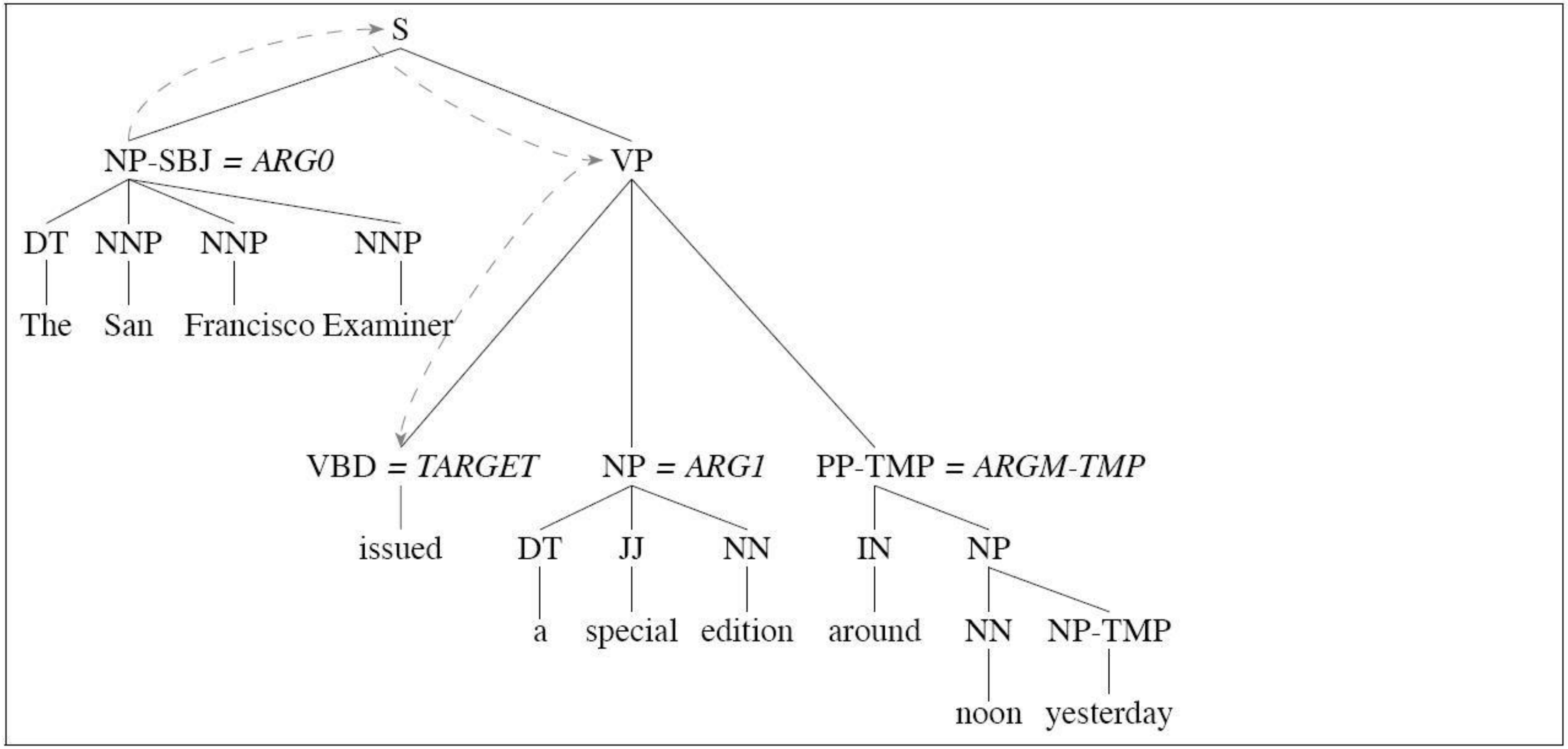
Semantic Role Labeling Algorithm

- Need syntactic parser.
- Extract features.
- Classify node.

```
function SEMANTICROLELABEL(words) returns labeled tree

    parse ← PARSE(words)
    for each predicate in parse do
        for each node in parse do
            featurevector ← EXTRACTFEATURES(node, predicate, parse)
            CLASSIFYNODE(node, featurevector, parse)
```

Semantic Role Labeling



Semantic Role Labeling-Features

- Governing Predicate.
- Phase type of constituent.
- Headword of constituent.
- Path in the parse tree from constituent to the predicate.
- Voice of the clause containing constituent.
- Binary respect to predicate (before or after).
- Sub categorization of predicate.