Word Sense Disambiguation

Ling571
Deep Processing Techniques for NLP
February 23, 2011

Word Sense Disambiguation

- Robust Approaches
  - Supervised Learning Approaches
    - Naïve Bayes
  - Dictionary-based Approaches
  - Bootstrapping Approaches
    - One sense per discourse/collocation
  - Similarity-based approaches
    - Thesaurus-based techniques
    - Unsupervised approaches
  - Why they work, Why they don't

Disambiguation Features

- Key: What are the features?
  - Part of speech
    - Of word and neighbors
  - Morphologically simplified form
  - Words in neighborhood
    - Question: How big a neighborhood?
      - Is there a single optimal size? Why?
  - (Possibly shallow) Syntactic analysis
    - E.g. predicate-argument relations, modification, phrases
  - Collocation vs co-occurrence features
    - Collocation: words in specific relation: p-a, 1 word +/-
    - Co-occurrence: bag of words..

WSD Evaluation

- Ideally, end-to-end evaluation with WSD component
  - Demonstrate real impact of technique in system
  - Difficult, expensive, still application specific
- Typically, intrinsic, sense-based
  - Accuracy, precision, recall
  - SENSEVAL/SEMEVAL: all words, lexical sample
- Baseline:
  - Most frequent sense, Lesk
- Topline:
  - Human inter-rater agreement: 75-80% fine; 90% coarse

Naïve Bayes' Approach

- Supervised learning approach
  - Input: feature vector X label

Naïve Bayes' Approach

- Supervised learning approach
  - Input: feature vector X label
- Best sense = most probable sense given f

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

Naïve Bayes' Approach

- Supervised learning approach
  - Input: feature vector X label
- Best sense = most probable sense given f

$$\hat{s} = \arg\max_{s \in S} P(s \mid \vec{f})$$

$$\hat{s} = \arg\max_{s \in S} \frac{P(\vec{f} \mid s)P(s)}{P(\vec{f})}$$

Naïve Bayes' Approach

- Issue:
  - Data sparseness: full feature vector rarely seen

Naïve Bayes' Approach

- Issue:
  - Data sparseness: full feature vector rarely seen
- "Naïve" assumption:
  - Features independent given sense

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

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

Training NB Classifier

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

- Estimate P(s):
  - Prior

Training NB Classifier

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

- Estimate P(s):
  - Prior

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

Training NB Classifier

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

- Estimate P(s):
  - Prior

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

- Estimate $P(f_j|s)$

Training NB Classifier

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

- Estimate P(s):
  - Prior

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

- Estimate $P(f_j|s)$ $\quad P(f_j \mid s) = \frac{count(f_j, s)}{count(s)}$
- Issues:

Training NB Classifier

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

- Estimate P(s):
  - Prior

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

- Estimate $P(f_j|s)$ $\quad P(f_j \mid s) = \frac{count(f_j, s)}{count(s)}$
- Issues:
  - Underflow => log prob

Training NB Classifier

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

- Estimate P(s):
  - Prior

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

- Estimate $P(f_j|s)$ $\quad P(f_j \mid s) = \frac{count(f_j, s)}{count(s)}$
- Issues:
  - Underflow => log prob
  - Sparseness => smoothing

Dictionary-Based Approach

- (Simplified) Lesk algorithm
  - “How to tell a pine cone from an ice cream cone”

Dictionary-Based Approach

- (Simplified) Lesk algorithm
  - “How to tell a pine cone from an ice cream cone”
  - Compute context ‘signature’ of word to disambiguate
    - Words in surrounding sentence(s)

Dictionary-Based Approach

- (Simplified) Lesk algorithm
  - "How to tell a pine cone from an ice cream cone"
  - Compute context 'signature' of word to disambiguate
    - Words in surrounding sentence(s)
  - Compare overlap w.r.t. dictionary entries for senses

Dictionary-Based Approach

- (Simplified) Lesk algorithm
  - "How to tell a pine cone from an ice cream cone"
  - Compute context 'signature' of word to disambiguate
    - Words in surrounding sentence(s)
  - Compare overlap w.r.t. dictionary entries for senses
  - Select sense with highest (non-stopword) overlap

Applying Lesk

- *The bank can guarantee deposits will eventually cover future tuition costs because it invests in 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"

Applying Lesk

- *The bank can guarantee deposits will eventually cover future tuition costs because it invests in 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

Applying Lesk

- *The bank can guarantee deposits will eventually cover future tuition costs because it invests in 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
- Bank[2]: 0

Improving Lesk

- Overlap score:
  - All words equally weighted (excluding stopwords)

Improving Lesk

- Overlap score:
  - All words equally weighted (excluding stopwords)
- Not all words equally informative

Improving Lesk

- Overlap score:
  - All words equally weighted (excluding stopwords)
- Not all words equally informative
  - Overlap with unusual/specific words – better
  - Overlap with common/non-specific words – less good

Improving Lesk

- Overlap score:
  - All words equally weighted (excluding stopwords)
- Not all words equally informative
  - Overlap with unusual/specific words – better
  - Overlap with common/non-specific words – less good
- Employ corpus weighting:
  - IDF: inverse document frequency
    - $Idf_i = \log (Ndoc/nd_i)$

Minimally Supervised WSD

- Yarowsky's algorithm (1995)
- Bootstrapping approach:
  - Use small labeled seedset to iteratively train

Minimally Supervised WSD

- Yarowsky's algorithm (1995)
- Bootstrapping approach:
  - Use small labeled seedset to iteratively train
- Builds on 2 key insights:
  - One Sense Per Discourse
    - Word appearing multiple times in text has same sense
    - Corpus of 37232 bass instances: always single sense

Minimally Supervised WSD

- Yarowsky's algorithm (1995)

- Bootstrapping approach:
  - Use small labeled seedset to iteratively train

- Builds on 2 key insights:
  - One Sense Per Discourse
    - Word appearing multiple times in text has same sense
    - Corpus of 37232 bass instances: always single sense

  - One Sense Per Collocation
    - Local phrases select single sense
      - Fish -> Bass[1]
      - Play -> Bass[2]

Yarowsky's Algorithm

- Training Decision Lists
  - 1. Pick Seed Instances & Tag

Yarowsky's Algorithm

- Training Decision Lists
  - 1. Pick Seed Instances & Tag
  - 2. Find Collocations: Word Left, Word Right, Word ±K

Yarowsky's Algorithm

- Training Decision Lists
  - 1. Pick Seed Instances & Tag
  - 2. Find Collocations: Word Left, Word Right, Word ±K
    - (A) Calculate Informativeness on Tagged Set,
      - Order: $abs(\log \frac{P(Sense_1 | Collocation)}{P(Sense_2 | Collocation)})$

Yarowsky's Algorithm

- Training Decision Lists
  - 1. Pick Seed Instances & Tag
  - 2. Find Collocations: Word Left, Word Right, Word ±K
    - (A) Calculate Informativeness on Tagged Set,
      - Order: $abs(\log \frac{P(Sense_1 \mid Collocation)}{P(Sense_2 \mid Collocation)})$
    - (B) Tag New Instances with Rules

Yarowsky's Algorithm

- Training Decision Lists
  - 1. Pick Seed Instances & Tag
  - 2. Find Collocations: Word Left, Word Right, Word ±K
    - (A) Calculate Informativeness on Tagged Set,
      - Order: $abs(\log \frac{P(Sense_1 \mid Collocation)}{P(Sense_2 \mid Collocation)})$
    - (B) Tag New Instances with Rules
    - (C) Apply 1 Sense/Discourse
    - (D)

Yarowsky's Algorithm

- Training Decision Lists
  - 1. Pick Seed Instances & Tag
  - 2. Find Collocations: Word Left, Word Right, Word ±K
    - (A) Calculate Informativeness on Tagged Set,
      - Order: $abs(\log \frac{P(Sense_1 \mid Collocation)}{P(Sense_2 \mid Collocation)})$
    - (B) Tag New Instances with Rules
    - (C) Apply 1 Sense/Discourse
    - (D) If Still Unlabeled, Go To 2

Yarowsky's Algorithm

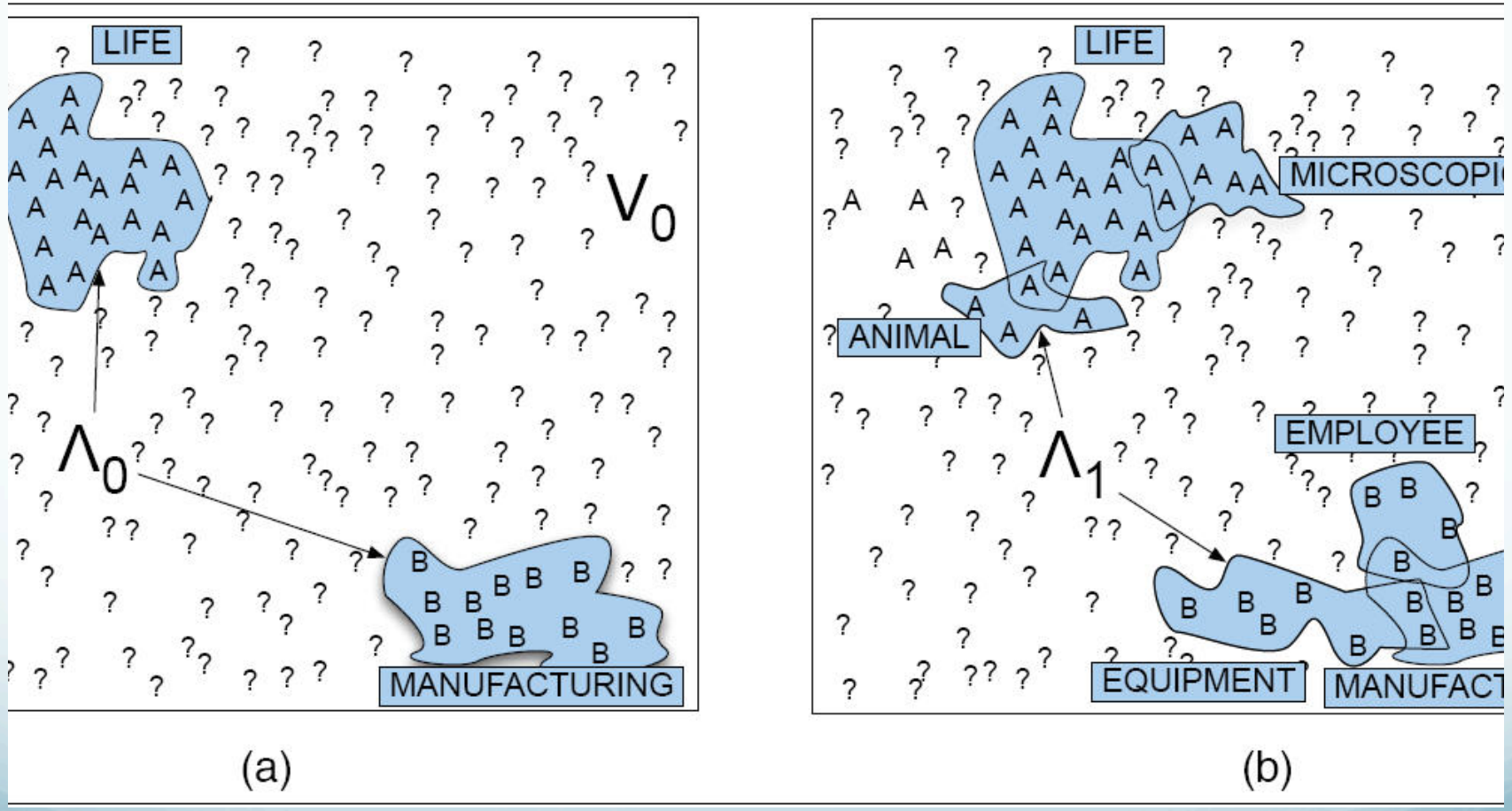
- Training Decision Lists
  - 1. Pick Seed Instances & Tag
  - 2. Find Collocations: Word Left, Word Right, Word ±K
    - (A) Calculate Informativeness on Tagged Set,
      - Order: $abs(\log \frac{P(Sense_1 | Collocation)}{P(Sense_2 | Collocation)})$
    - (B) Tag New Instances with Rules
    - (C) Apply 1 Sense/Discourse
    - (D) If Still Unlabeled, Go To 2
  - 3. Apply 1 Sense/Discourse

Yarowsky's Algorithm

- Training Decision Lists
  - 1. Pick Seed Instances & Tag
  - 2. Find Collocations: Word Left, Word Right, Word ±K
    - (A) Calculate Informativeness on Tagged Set,
      - Order: $$abs(\log \frac{P(Sense_1 \mid Collocation)}{P(Sense_2 \mid Collocation)})$$
    - (B) Tag New Instances with Rules
    - (C) Apply 1 Sense/Discourse
    - (D) If Still Unlabeled, Go To 2
  - 3. Apply 1 Sense/Discourse
- Disambiguation: First Rule Matched

Initial decision list for *plant* (abbreviated)		
LogL	Collocation	Sense
8.10	***plant* life**	⇒ A
7.58	**manufacturing *plant***	⇒ B
7.39	**life (within ±2-10 words)**	⇒ A
7.20	**manufacturing (in ±2-10 words)**	⇒ B
6.27	**animal (within ±2-10 words)**	⇒ A
4.70	**equipment (within ±2-10 words)**	⇒ B
4.39	**employee (within ±2-10 words)**	⇒ B
4.30	**assembly *plant***	⇒ B
4.10	***plant* closure**	⇒ B
3.52	***plant* species**	⇒ A
3.48	**automate (within ±2-10 words)**	⇒ B
3.45	**microscopic *plant***	⇒ A
	...	

Iterative Updating



There are more kinds of plants and animals in the rainforests than anywhere else on Earth. Over half of the millions of known species of plants and animals live in the rainforest. Many are found nowhere else. There are even plants and animals in the rainforest that we have not yet discovered.

Biological Example

The Paulus company was founded in 1938. Since those days the product range has been the subject of constant expansions and is brought up continuously to correspond with the state of the art. We’re engineering, manufacturing and commissioning world-wide ready-to-run plants packed with our comprehensive know-how. Our Product Range includes pneumatic conveying systems for carbon, carbide, sand, lime andmany others. We use reagent injection in molten metal for the…

Industrial Example

Label the First Use of “Plant”

Sense Choice With Collocational Decision Lists

- Create Initial Decision List
  - Rules Ordered by $abs(\log \frac{P(Sense_1 \mid Collocation)}{P(Sense_2 \mid Collocation)})$
- Check nearby Word Groups (Collocations)
  - Biology: "Animal" in ± 2-10 words
  - Industry: "Manufacturing" in ± 2-10 words
- Result: Correct Selection
  - 95% on Pair-wise tasks

Word Similarity

- Synonymy:

Word Similarity

- Synonymy:
  - True propositional substitutability is rare, slippery

Word Similarity

- Synonymy:
  - True propositional substitutability is rare, slippery
- Word similarity (semantic distance):
  - Looser notion, more flexible

Word Similarity

- Synonymy:
  - True propositional substitutability is rare, slippery
- Word similarity (semantic distance):
  - Looser notion, more flexible
  - Appropriate to applications:
    - IR, summarization, MT, essay scoring

Word Similarity

- Synonymy:
  - True propositional substitutability is rare, slippery
- Word similarity (semantic distance):
  - Looser notion, more flexible
  - Appropriate to applications:
    - IR, summarization, MT, essay scoring
      - Don't need binary +/- synonym decision

Word Similarity

- Synonymy:
  - True propositional substitutability is rare, slippery
- Word similarity (semantic distance):
  - Looser notion, more flexible
  - Appropriate to applications:
    - IR, summarization, MT, essay scoring
      - Don't need binary +/- synonym decision
      - Want terms/documents that have high similarity

Word Similarity

- Synonymy:
  - True propositional substitutability is rare, slippery
- Word similarity (semantic distance):
  - Looser notion, more flexible
  - Appropriate to applications:
    - IR, summarization, MT, essay scoring
      - Don't need binary +/- synonym decision
      - Want terms/documents that have high similarity
        - Differ from relatedness

Word Similarity

- Synonymy:
  - True propositional substitutability is rare, slippery
- Word similarity (semantic distance):
  - Looser notion, more flexible
  - Appropriate to applications:
    - IR, summarization, MT, essay scoring
      - Don't need binary +/- synonym decision
      - Want terms/documents that have high similarity
        - Differ from relatedness
- Approaches:
  - Thesaurus-based
  - Distributional

Thesaurus-based Techniques

- Key idea:
  - Shorter path length in thesaurus, smaller semantic dist.

Thesaurus-based Techniques

- Key idea:
  - Shorter path length in thesaurus, smaller semantic dist.
    - Words similar to parents, siblings in tree
      - Further away, less similar

Thesaurus-based Techniques

- Key idea:
  - Shorter path length in thesaurus, smaller semantic dist.
    - Words similar to parents, siblings in tree
      - Further away, less similar
- Pathlength=# edges in shortest route in graph b/t nodes

Thesaurus-based Techniques

- Key idea:
  - Shorter path length in thesaurus, smaller semantic dist.
    - Words similar to parents, siblings in tree
      - Further away, less similar
- Pathlength=# edges in shortest route in graph b/t nodes
  - Sim_{path}= -log pathlen(c_1 ,c_2) [Leacock & Chodorow]

Thesaurus-based Techniques

- Key idea:
  - Shorter path length in thesaurus, smaller semantic dist.
    - Words similar to parents, siblings in tree
      - Further away, less similar
- Pathlength=# edges in shortest route in graph b/t nodes
  - Sim_{path}= -log pathlen(c_1 ,c_2) [Leacock & Chodorow]
- Problem 1:

Thesaurus-based Techniques

- Key idea:
  - Shorter path length in thesaurus, smaller semantic dist.
    - Words similar to parents, siblings in tree
      - Further away, less similar
- Pathlength=# edges in shortest route in graph b/t nodes
  - Sim_{path}= -log pathlen(c_1 ,c_2) [Leacock & Chodorow]
- Problem 1:
  - Rarely know which sense, and thus which node

Thesaurus-based Techniques

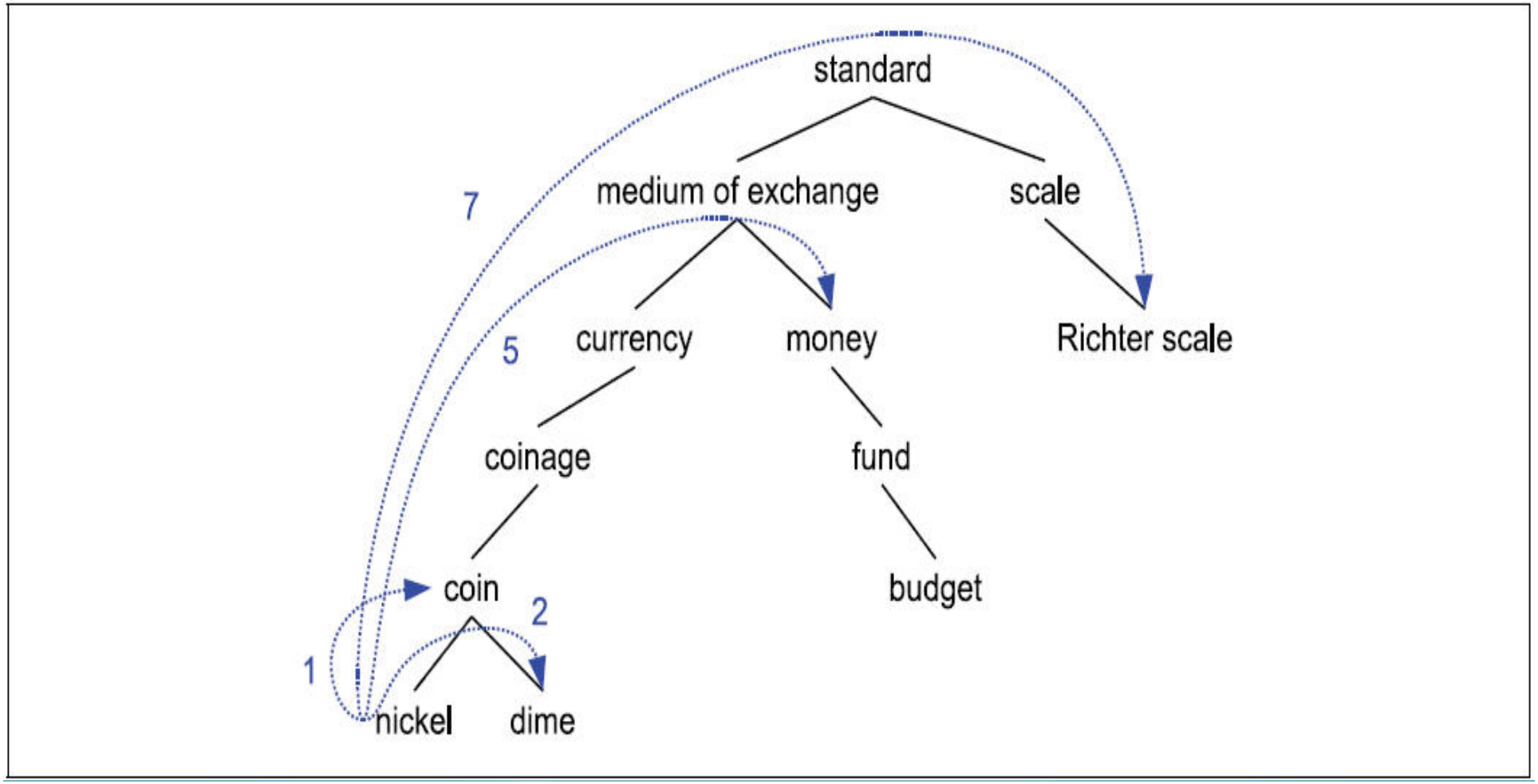
- Key idea:
  - Shorter path length in thesaurus, smaller semantic dist.
    - Words similar to parents, siblings in tree
      - Further away, less similar
- Pathlength=# edges in shortest route in graph b/t nodes
  - Sim_{path}= -log pathlen(c_1 ,c_2) [Leacock & Chodorow]
- Problem 1:
  - Rarely know which sense, and thus which node
- Solution: assume most similar senses estimate

Thesaurus-based Techniques

- Key idea:
  - Shorter path length in thesaurus, smaller semantic dist.
    - Words similar to parents, siblings in tree
      - Further away, less similar
- Pathlength=# edges in shortest route in graph b/t nodes
  - Sim_{path}= -log pathlen(c_1 ,c_2) [Leacock & Chodorow]
- Problem 1:
  - Rarely know which sense, and thus which node
- Solution: assume most similar senses estimate
  - Wordsim(w_1,$w_{2)}$ = max sim(c_1,c_2)

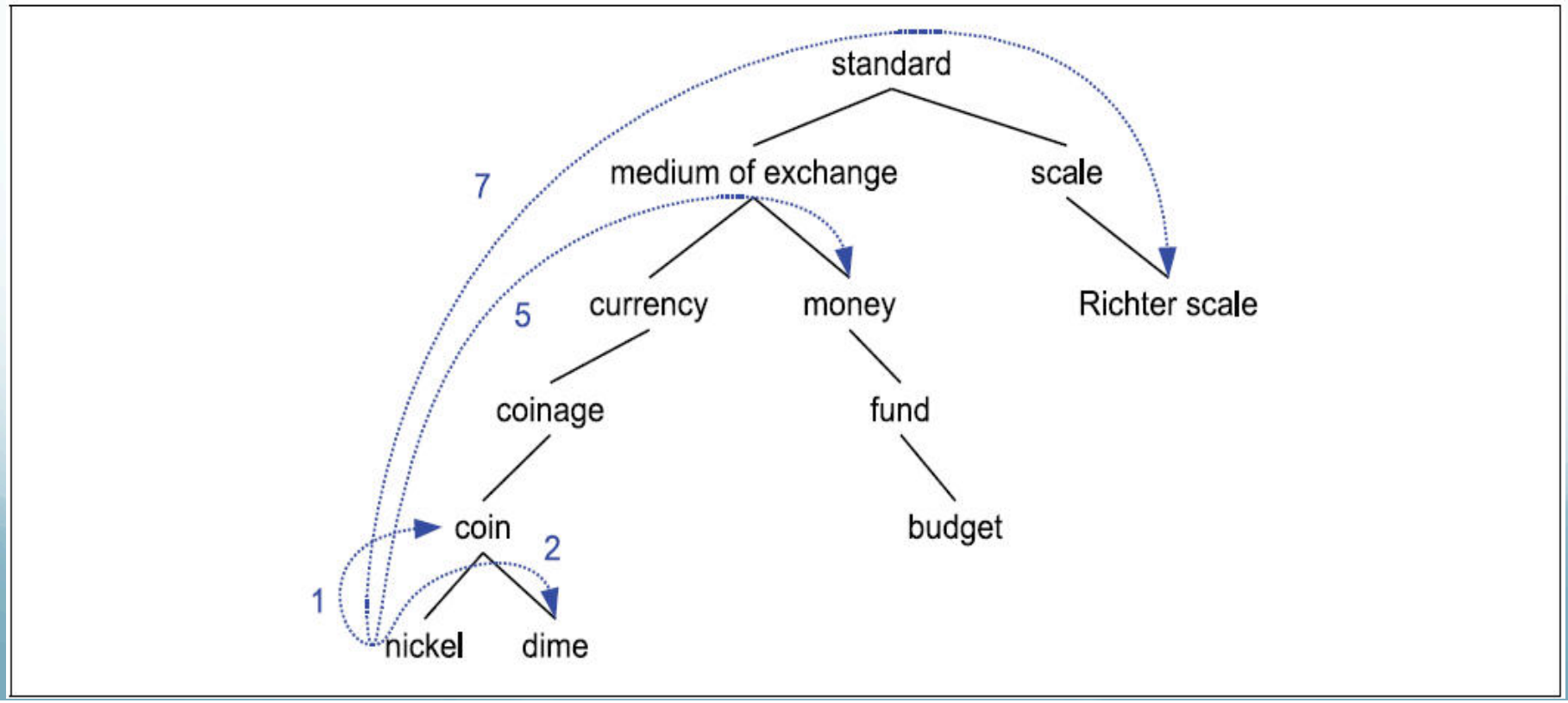
Path Length

- Path length problem:



Path Length

- Path length problem:
  - Links in WordNet not uniform
    - Distance 5: Nickel->Money and Nickel->Standard



Resnik's Similarity Measure

- Solution 1:

Resnik's Similarity Measure

- Solution 1:
  - Build position-specific similarity measure

Resnik's Similarity Measure

- Solution 1:
  - Build position-specific similarity measure
  - Not general
- Solution 2:

Resnik's Similarity Measure

- Solution 1:
  - Build position-specific similarity measure
  - Not general
- Solution 2:
  - Add corpus information: information-content measure
    - P(c) : probability that a word is instance of concept c

Resnik's Similarity Measure

- Solution 1:
  - Build position-specific similarity measure
  - Not general
- Solution 2:
  - Add corpus information: information-content measure
    - P(c) : probability that a word is instance of concept c
      - Words(c) : words subsumed by concept c; N: words in corpus

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

Resnik's Similarity Measure

- Information content of node:
  - IC(c) = -log P(c)

Resnik's Similarity Measure

- Information content of node:
  - IC(c) = -log P(c)
- Least common subsumer (LCS):
  - Lowest node in hierarchy subsuming 2 nodes

Resnik's Similarity Measure

- Information content of node:
  - $IC(c) = -\log P(c)$
- Least common subsumer (LCS):
  - Lowest node in hierarchy subsuming 2 nodes
- Similarity measure:
  - $sim_{RESNIK}(c_1,c_2) = -\log P(LCS(c_1,c_2))$

Resnik's Similarity Measure

- Information content of node:
  - $IC(c) = -\log P(c)$
- Least common subsumer (LCS):
  - Lowest node in hierarchy subsuming 2 nodes
- Similarity measure:
  - $sim_{RESNIK}(c_1,c_2) = -\log P(LCS(c_1,c_2))$
- Issue:

Resnik's Similarity Measure

- Information content of node:
  - $IC(c) = -\log P(c)$
- Least common subsumer (LCS):
  - Lowest node in hierarchy subsuming 2 nodes
- Similarity measure:
  - $sim_{RESNIK}(c_1,c_2) = -\log P(LCS(c_1,c_2))$
- Issue:
  - Not content, but difference between node & LCS

Resnik's Similarity Measure

- Information content of node:
  - IC(c) = -log P(c)
- Least common subsumer (LCS):
  - Lowest node in hierarchy subsuming 2 nodes
- Similarity measure:
  - $sim_{RESNIK}(c_1,c_2) = -\log P(LCS(c_1,c_2))$
- Issue:
  - Not content, but difference between node & LCS

$$sim_{Lin}(c_1,c_2) = \frac{2 \times \log P(LCS(c_1,c_2))}{\log P(c_1) + \log P(c_2)}$$