Distributional Semantics

Ling571
Deep Processing Techniques for NLP
February 25, 2015

Roadmap

- Distributional models
  - Context
  - Features
  - Weighting
  - Compression
  - Integration
- Thesaurus-based similarity models
  - Distance & Similarity in a Thesaurus

Distributional Similarity

- Represent ‘company’ of word such that similar words will have similar representations
  - ‘Company’ = context

Distributional Similarity

- Represent ‘company’ of word such that similar words will have similar representations
  - ‘Company’ = context
- Word represented by context feature vector
  - Many alternatives for vector

Distributional Similarity

- Represent 'company' of word such that similar words will have similar representations
  - 'Company' = context
- Word represented by context feature vector
  - Many alternatives for vector
- Initial representation:
  - 'Bag of words' binary feature vector

Distributional Similarity

- Represent 'company' of word such that similar words will have similar representations
  - 'Company' = context
- Word represented by context feature vector
  - Many alternatives for vector
- Initial representation:
  - 'Bag of words' binary feature vector
  - Feature vector length N, where N is size of vocabulary
    - f_i= 1 if word_i within window of *w*, 0 o.w.

Binary Feature Vector

	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

Distributional Similarity Questions

Distributional Similarity Questions

- What is the right neighborhood?
  - What is the context?

Distributional Similarity Questions

- What is the right neighborhood?
  - What is the context?

- How should we weight the features?

Distributional Similarity Questions

- What is the right neighborhood?
  - What is the context?
- How should we weight the features?
- How can we compute similarity between vectors?

Feature Vector Design

- Window size:
  - How many words in the neighborhood?
    - Tradeoff:

Feature Vector Design

- Window size:
  - How many words in the neighborhood?
    - Tradeoff:
      - +/- 500 words

Feature Vector Design

- Window size:
  - How many words in the neighborhood?
    - Tradeoff:
      - +/- 500 words: ‘topical context’

Feature Vector Design

- Window size:
  - How many words in the neighborhood?
    - Tradeoff:
      - +/- 500 words: ‘topical context’
      - +/- 1 or 2 words:

Feature Vector Design

- Window size:
  - How many words in the neighborhood?
    - Tradeoff:
      - +/- 500 words: ‘topical context’
      - +/- 1 or 2 words: collocations, predicate-argument

Feature Vector Design

- Window size:
  - How many words in the neighborhood?
    - Tradeoff:
      - +/- 500 words: ‘topical context’
      - +/- 1 or 2 words: collocations, predicate-argument
      - Only words in some grammatical relation
        - Parse text (dependency)
        - Include subj-verb; verb-obj; adj-mod
          - NxR vector: word x relation

Context Windows

- Same corpus, different windows
  - BNC
  - Nearest neighbors of “dog”

- 2-word window:
  - Cat, horse, fox, pet, rabbit, pig, animal, mongrel, sheep, pigeon
- 30-word window:
  - Kennel, puppy, pet, terrier, Rottweiler, canine, cat, to bark, Alsatian

Example Lin Relation Vector

	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

Document Context

- All models so far:
  - Term x term (or term x relation)
- Alternatively:
  - Term x document
    - Vectors of occurrences (association) in "document"
      - Document can be:
        - Typically: article, essay, etc
        - Also, utterance, dialog act
- Well-known term x document model:
  - Latent Semantic Analysis (LSA)

LSA Document Contexts

- (Deerwester et al, 1990)
- Titles of scientific articles

Example of text data: Titles of Some Technical Memos

c1:	*Human* machine *interface* for ABC *computer* applications
c2:	A *survey* of *user* opinion of *computer system response time*
c3:	The *EPS user interface* management *system*
c4:	*System* and *human system* engineering testing of *EPS*
c5:	Relation of *user* perceived *response time* to error measurement
m1:	The generation of random, binary, ordered *trees*
m2:	The intersection *graph* of paths in *trees*
m3:	*Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4:	*Graph minors*: A *survey*

Document Context Representation

- Term x document:

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Document Context Representation

- Term x document:
  - Corr(human,user) = -0.38; corr(human,minors)=-0.29

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Improved Representation

- Reduced dimension projection:
  - Corr(human,user) = 0.98; corr(human,minors)=-0.83

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

Weighting Features

- Baseline: Binary (0/1)

Weighting Features

- Baseline: Binary (0/1)
  - Minimally informative
  - Can't capture intuition that frequent features informative

Weighting Features

- Baseline: Binary (0/1)
  - Minimally informative
  - Can't capture intuition that frequent features informative
- Frequency or Probability:

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

Weighting Features

- Baseline: Binary (0/1)
  - Minimally informative
  - Can't capture intuition that frequent features informative
- Frequency or Probability:

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

  - Better but,

Weighting Features

- Baseline: Binary (0/1)
  - Minimally informative
  - Can't capture intuition that frequent features informative
- Frequency or Probability:

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

  - Better but,
  - Can overweight a priori frequent features
    - Chance cooccurrence

Pointwise Mutual Information

$$assoc_{PMI}(w,f) = \log_2 \frac{P(w,f)}{P(w)P(f)}$$

PMI:

- Contrasts observed cooccurrence
- With that expected by chance (if independent)

Pointwise Mutual Information

$$assoc_{PMI}(w, f) = \log_2 \frac{P(w, f)}{P(w)P(f)}$$

PMI:

- Contrasts observed cooccurrence
- With that expected by chance (if independent)

- Generally only use positive values
  - Negatives inaccurate unless corpus huge

Lin Association

- Recall:
  - Lin's vectors include:
    - r: dependency relation
    - w': other word in dependency relation
- Decomposes weights on that basis:

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

Vector Similarity

- Euclidean or Manhattan distances:

Vector Similarity

- Euclidean or Manhattan distances:
  - Too sensitive to extreme values

Vector Similarity

- Euclidean or Manhattan distances:
  - Too sensitive to extreme values

- Dot product: $sim_{dot-product}(\vec{v},\vec{w}) = \vec{v} \bullet \vec{w} = \sum_{i=1}^{N} v_i \times w_i$

Vector Similarity

- Euclidean or Manhattan distances:
  - Too sensitive to extreme values
- Dot product: $sim_{dot-product}(\vec{v},\vec{w}) = \vec{v} \bullet \vec{w} = \sum_{i=1}^{N} v_i \times w_i$
  - Favors long vectors:
    - More features or higher values

Vector Similarity

- Euclidean or Manhattan distances:
  - Too sensitive to extreme values
- Dot product: $sim_{dot-product}(\vec{v},\vec{w}) = \vec{v} \bullet \vec{w} = \sum_{i=1}^{N} v_i \times w_i$
  - Favors long vectors:
    - More features or higher values
- Cosine: $sim_{cosine}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$

Alternative Weighting Schemes

- Models have used alternate weights of computing similarity based on weighted overlap

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

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

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

Results

- Based on Lin_{assoc}
  - Hope (N): optimism, chance, expectation, prospect, dream, desire, fear
  - Hope (V): would like, wish, plan, say, believe, think

  - Brief (N): legal brief, affidavit, filing, petition, document, argument, letter
  - Brief (A): lengthy, hour-long, short, extended, frequent, recent, short-lived, prolonged, week-long

Curse of Dimensionality

- Vector representations:
  - Sparse
  - Very high dimensional:
    - # words in vocabulary
    - # relations x # words, etc

Curse of Dimensionality

- Vector representations:
  - Sparse
  - Very high dimensional:
    - # words in vocabulary
    - # relations x # words, etc
- Google1T5 corpus:
  - 1M x 1M matrix

Curse of Dimensionality

- Vector representations:
  - Sparse
  - Very high dimensional:
    - # words in vocabulary
    - # relations x # words, etc
- Google1T5 corpus:
  - 1M x 1M matrix: < 0.05% non-zero values
- Computationally hard to manage
  - Lots of zeroes
  - Can miss underlying relations

Reducing Dimensionality

- Feature selection:
  - Desirable traits:

Reducing Dimensionality

- Feature selection:
  - Desirable traits:
    - High frequency
    - High variance

Reducing Dimensionality

- Feature selection:
  - Desirable traits:
    - High frequency
    - High variance
- Filtering:
  - Can exclude terms with too few occurrences
  - Can include only top X most frequent terms
  - Chi-squared selection

Reducing Dimensionality

- Feature selection:
  - Desirable traits:
    - High frequency
    - High variance
- Filtering:
  - Can exclude terms with too few occurrences
  - Can include only top X most frequent terms
  - Chi-squared selection
- Cautions:
  - Feature correlations
  - Joint feature selection complex, expensive

Reducing Dimensionality

- Projection into lower dimensional space:
  - Principal Components Analysis (PCA), Locality Preserving Projections (LPP), Singular Value Decomposition, etc
- Create new lower dimensional space that
  - Preserves distances between data points
    - Keep like with like
  - Approaches differ on exactly what is preserved.

SVD

- Enables creation of reduced dimension model
  - Low rank approximation of original matrix
    - Best-fit at that rank (in least-squares sense)

SVD

- Enables creation of reduced dimension model
  - Low rank approximation of original matrix
    - Best-fit at that rank (in least-squares sense)

- Motivation:
  - Original matrix: high dimensional, sparse
    - Similarities missed due to word choice, etc
  - Create new projected space
    - More compact, better captures important variation
  - Landauer et al argue identifies underlying "concepts"
    - Across words with related meanings

Diverse Applications

- Unsupervised POS tagging
- Word Sense Disambiguation
- Essay Scoring
- Document Retrieval
- Unsupervised Thesaurus Induction
- Ontology/Taxonomy Expansion
- Analogy tests, word tests
- Topic Segmentation

Distributional Similarity for Word Sense Disambiguation

Schutze's Word Space

- Build a co-occurrence matrix

Schutze's Word Space

- Build a co-occurrence matrix
  - Restrict Vocabulary to 4 letter sequences

Schutze' s Word Space

- Build a co-occurrence matrix
  - Restrict Vocabulary to 4 letter sequences
    - Similar effect to stemming
    - Exclude Very Frequent - Articles, Affixes

Schutze's Word Space

- Build a co-occurrence matrix
  - Restrict Vocabulary to 4 letter sequences
    - Similar effect to stemming
    - Exclude Very Frequent - Articles, Affixes
  - Entries in 5000-5000 Matrix
    - Apply Singular Value Decomposition (SVD)
    - Reduce to 97 dimensions

Schutze’s Word Space

- Build a co-occurrence matrix
  - Restrict Vocabulary to 4 letter sequences
    - Similar effect to stemming
    - Exclude Very Frequent - Articles, Affixes
  - Entries in 5000-5000 Matrix
    - Apply Singular Value Decomposition (SVD)
    - Reduce to 97 dimensions
- Word Context
  - 4grams within 1001 Characters

Word Representation

- 2nd order representation:
  - Identify words in context of *w*
  - For each x in context of w
    - Compute x's vector representation
  - Compute centroid of those x vector representations

Computing Word Senses

- Compute context vector for each occurrence of word in corpus
- Cluster these context vectors
  - # of clusters = # number of senses
- Cluster centroid represents word sense
- Link to specific sense?

Computing Word Senses

- Compute context vector for each occurrence of word in corpus
- Cluster these context vectors
  - # of clusters = # number of senses
- Cluster centroid represents word sense
- Link to specific sense?
  - Pure unsupervised: no sense tag, just i^{th} sense
  - Some supervision: hand label clusters, or tag training

Disambiguating Instances

- To disambiguate an instance t of w:
  - Compute context vector for the instance
  - Retrieve all senses of w
  - Assign w sense with closest centroid to t

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"

Example Sense Selection for Plant Data

- Build a Context Vector

Example Sense Selection for Plant Data

- Build a Context Vector
  - 1,001 character window - Whole Article

Example Sense Selection for Plant Data

- Build a Context Vector
  - 1,001 character window - Whole Article
- Compare Vector Distances to Sense Clusters

Example Sense Selection for Plant Data

- Build a Context Vector
  - 1,001 character window - Whole Article
- Compare Vector Distances to Sense Clusters
  - Only 3 Content Words in Common
  - Distant Context Vectors
  - Clusters - Build Automatically, Label Manually

Example Sense Selection for Plant Data

- Build a Context Vector
  - 1,001 character window - Whole Article
- Compare Vector Distances to Sense Clusters
  - Only 3 Content Words in Common
  - Distant Context Vectors
  - Clusters - Build Automatically, Label Manually
- Result: 2 Different, Correct Senses
  - 92% on Pair-wise tasks

Odd Cluster Examples

- The “Ste.” Cluster:
  - Dry Oyster Whisky Hot Float Ice

Odd Cluster Examples

- The “Ste.” Cluster:
  - Dry Oyster Whisky Hot Float Ice
  - Why? – River name

Odd Cluster Examples

- The “Ste.” Cluster:
  - Dry Oyster Whisky Hot Float Ice
  - Why? – River name
    - Learning the Corpus, not the Sense

- Keeping cluster:
  - Bring Hoping Wiping Could Should Some Them Rest

Odd Cluster Examples

- The “Ste.” Cluster:
  - Dry Oyster Whisky Hot Float Ice
  - Why? – River name
    - Learning the Corpus, not the Sense
- Keeping cluster:
  - Bring Hoping Wiping Could Should Some Them Rest
    - Uninformative: Wide context misses verb sense

Distributional Models

- Upsurge in distributional compositional models
  - Neural network embeddings:
    - Discriminatively trained, low dimensional reps
    - E.g. word2vec
      - Skipgrams etc over large corpora
  - Composition:
    - Methods for combining word vector models
      - Capture phrasal, sentential meanings

Thesaurus-Based Similarity

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

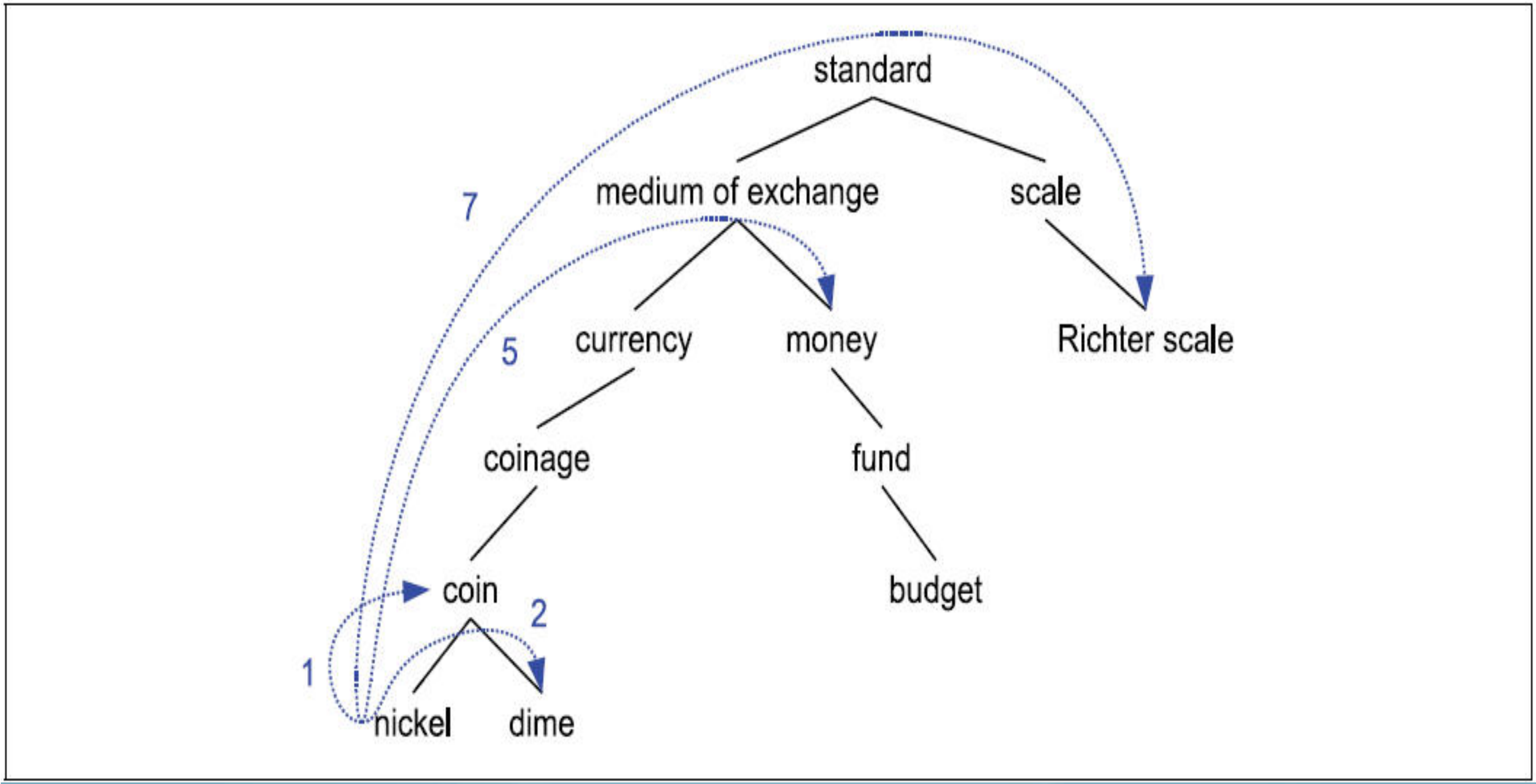
- Key idea:
  - Shorter path length in thesaurus, smaller semantic dist.
    - Words similar to parents, siblings in tree
      - Further away, less similar
- Pathlength=# edges in shortest route in graph b/t nodes
  - Sim_{path}= -log pathlen(c_1 ,c_2) [Leacock & Chodorow]
- Problem 1:
  - Rarely know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

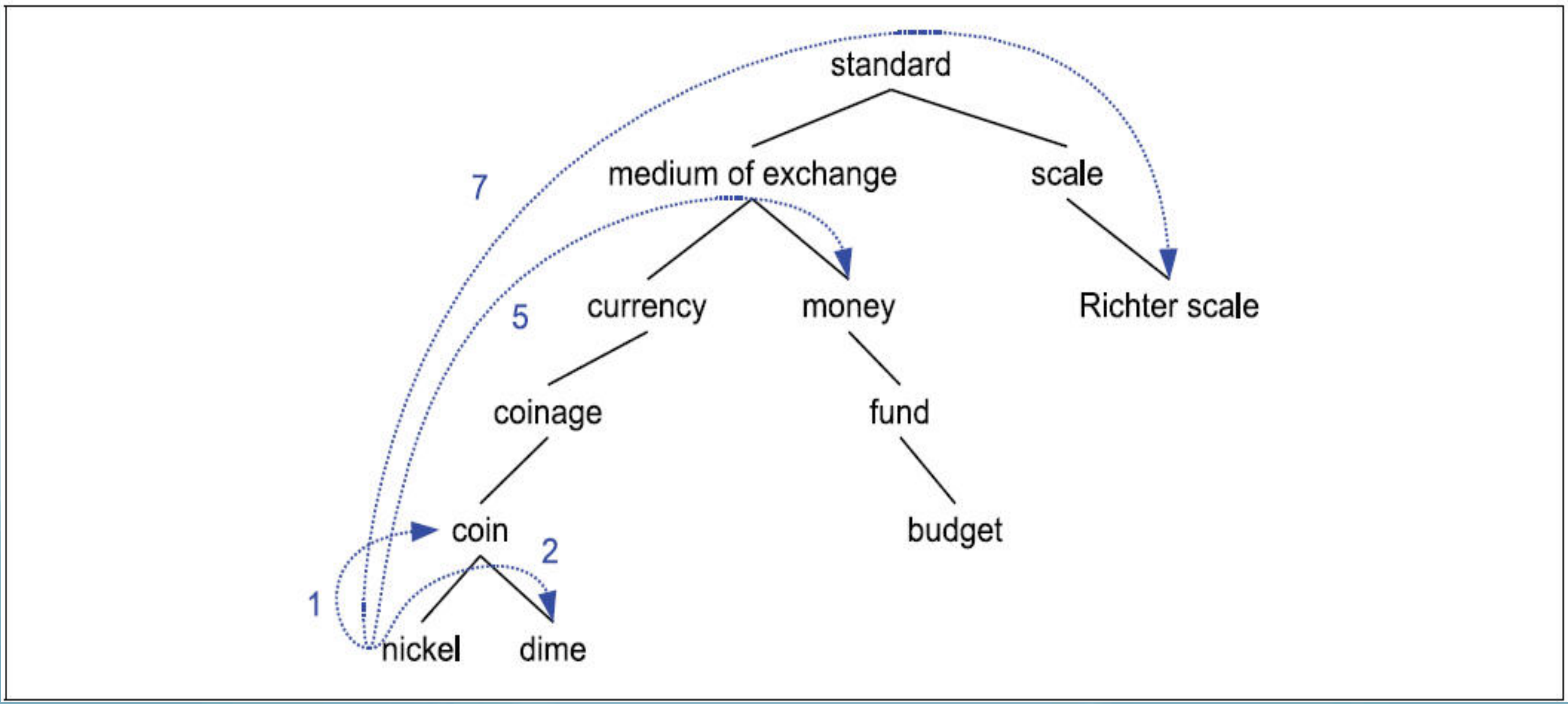
Path Length

- 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

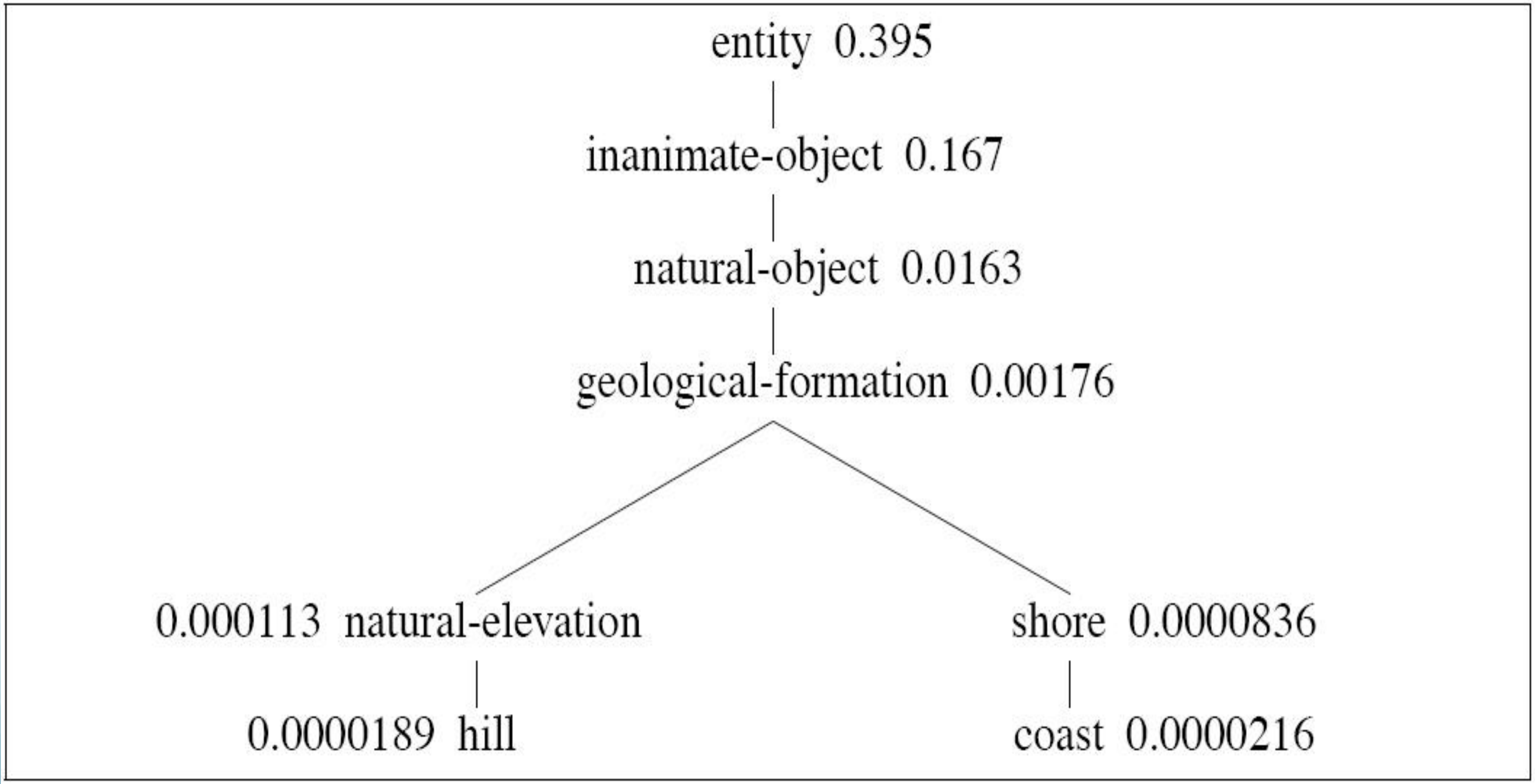
Resnik's Similarity Measure

- 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:

IC Example


entity 0.395
inanimate-object 0.167
natural-object 0.0163
geological-formation 0.00176
0.000113 natural-elevation
shore 0.0000836
0.0000189 hill
coast 0.0000216

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

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

Application to WSD

- Calculate Informativeness
  - For Each Node in WordNet:

Application to WSD

- Calculate Informativeness
  - For Each Node in WordNet:
    - Sum occurrences of concept and all children
    - Compute IC

Application to WSD

- Calculate Informativeness
  - For Each Node in WordNet:
    - Sum occurrences of concept and all children
    - Compute IC
- Disambiguate with WordNet

Application to WSD

- Calculate Informativeness
  - For Each Node in WordNet:
    - Sum occurrences of concept and all children
    - Compute IC
- Disambiguate with WordNet
  - Assume set of words in context
    - E.g. {plants, animals, rainforest, species} from article

Application to WSD

- Calculate Informativeness
  - For Each Node in WordNet:
    - Sum occurrences of concept and all children
    - Compute IC
- Disambiguate with WordNet
  - Assume set of words in context
    - E.g. {plants, animals, rainforest, species} from article
  - Find Most Informative Subsumer for each pair, I
    - Find LCS for each pair of senses, pick highest similarity

Application to WSD

- Calculate Informativeness
  - For Each Node in WordNet:
    - Sum occurrences of concept and all children
    - Compute IC
- Disambiguate with WordNet
  - Assume set of words in context
    - E.g. {plants, animals, rainforest, species} from article
  - Find Most Informative Subsumer for each pair, I
    - Find LCS for each pair of senses, pick highest similarity
  - For each subsumed sense, Vote += I

Application to WSD

- Calculate Informativeness
  - For Each Node in WordNet:
    - Sum occurrences of concept and all children
    - Compute IC
- Disambiguate with WordNet
  - Assume set of words in context
    - E.g. {plants, animals, rainforest, species} from article
  - Find Most Informative Subsumer for each pair, I
    - Find LCS for each pair of senses, pick highest similarity
  - For each subsumed sense, Vote += I
  - Select Sense with Highest Vote

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 and many others. We use reagent injection in molten metal for the…

Industrial Example

Label the First Use of "Plant"

Sense Labeling Under WordNet

- Use Local Content Words as Clusters
  - Biology: Plants, Animals, Rainforests, species…
  - Industry: Company, Products, Range, Systems…

Sense Labeling Under WordNet

- Use Local Content Words as Clusters
  - Biology: Plants, Animals, Rainforests, species…
  - Industry: Company, Products, Range, Systems…
- Find Common Ancestors in WordNet

Sense Labeling Under WordNet

- Use Local Content Words as Clusters
  - Biology: Plants, Animals, Rainforests, species…
  - Industry: Company, Products, Range, Systems…
- Find Common Ancestors in WordNet
  - Biology: Plants & Animals isa Living Thing

Sense Labeling Under WordNet

- Use Local Content Words as Clusters
  - Biology: Plants, Animals, Rainforests, species…
  - Industry: Company, Products, Range, Systems…
- Find Common Ancestors in WordNet
  - Biology: Plants & Animals isa Living Thing
  - Industry: Product & Plant isa Artifact isa Entity

Sense Labeling Under WordNet

- Use Local Content Words as Clusters
  - Biology: Plants, Animals, Rainforests, species…
  - Industry: Company, Products, Range, Systems…
- Find Common Ancestors in WordNet
  - Biology: Plants & Animals isa Living Thing
  - Industry: Product & Plant isa Artifact isa Entity
  - Use Most Informative
- Result: Correct Selection

Thesaurus Similarity Issues

Thesaurus Similarity Issues

- Coverage:
  - Few languages have large thesauri

Thesaurus Similarity Issues

- Coverage:
  - Few languages have large thesauri
  - Few languages have large sense tagged corpora

Thesaurus Similarity Issues

- Coverage:
  - Few languages have large thesauri
  - Few languages have large sense tagged corpora
- Thesaurus design:
  - Works well for noun IS-A hierarchy

Thesaurus Similarity Issues

- Coverage:
  - Few languages have large thesauri
  - Few languages have large sense tagged corpora
- Thesaurus design:
  - Works well for noun IS-A hierarchy
  - Verb hierarchy shallow, bushy, less informative

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

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

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

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

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

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

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 $P(f_j|s)$

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

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

Training NB Classifier

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

- Estimate P(s):
  - Prior

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

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