Lecture 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

<table>
<thead>
<tr>
<th>WordNet Sense</th>
<th>Spanish Translation</th>
<th>Roget Category</th>
<th>Target Word in Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>bass⁴</td>
<td>lubina</td>
<td>FISH/INSECT</td>
<td>…fish as Pacific salmon and striped bass and…</td>
</tr>
<tr>
<td>bass⁴</td>
<td>lubina</td>
<td>FISH/INSECT</td>
<td>…produce filets of smoked bass or sturgeon…</td>
</tr>
<tr>
<td>bass⁷</td>
<td>bajo</td>
<td>MUSIC</td>
<td>…exciting jazz bass player since Ray Brown…</td>
</tr>
<tr>
<td>bass⁷</td>
<td>bajo</td>
<td>MUSIC</td>
<td>…play bass because he doesn’t have to solo…</td>
</tr>
</tbody>
</table>
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 $\vec{f}$

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

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

- Assumption: naively assume features are independent of each other

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

2016 Dan I. Moldovan, Human Language Technology Research Institute, The University of Texas at Dallas
Naïve Bayes Classifier

\[ \hat{s} = \arg\max_{s \in S} P(s) \prod_{j=1}^{n} P(f_j | s) \]

\[ P(s_i) = \frac{\text{count}(s_i, w_j)}{\text{count}(w_j)} \]

\[ P(f_j | s) = \frac{\text{count}(f_j, s)}{\text{count}(s)} \]
Decision trees are also used and are easier to understand. A sequence of tests are performed.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Sense</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>fish</em> within window</td>
<td>=&gt;</td>
</tr>
<tr>
<td><em>striped bass</em></td>
<td>=&gt;</td>
</tr>
<tr>
<td><em>guitar</em> within window</td>
<td>=&gt;</td>
</tr>
<tr>
<td><em>bass player</em></td>
<td>=&gt;</td>
</tr>
<tr>
<td><em>piano</em> within window</td>
<td>=&gt;</td>
</tr>
<tr>
<td><em>tenor</em> within window</td>
<td>=&gt;</td>
</tr>
<tr>
<td><em>sea bass</em></td>
<td>=&gt;</td>
</tr>
<tr>
<td><em>play/V bass</em></td>
<td>=&gt;</td>
</tr>
<tr>
<td><em>river</em> within window</td>
<td>=&gt;</td>
</tr>
<tr>
<td><em>violin</em> within window</td>
<td>=&gt;</td>
</tr>
<tr>
<td><em>salmon</em> within window</td>
<td>=&gt;</td>
</tr>
<tr>
<td><em>on bass</em></td>
<td>=&gt;</td>
</tr>
<tr>
<td><em>bass are</em></td>
<td>=&gt;</td>
</tr>
</tbody>
</table>

2016 Dan I. Moldovan, Human Language Technology Research Institute, The University of Texas at Dallas
Decision list Classifier

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

$$\left| \log \left( \frac{P(\text{Sense}_1 | f_i)}{P(\text{Sense}_2 | f_i)} \right) \right|$$
WSD Evaluation

Baseline most frequently used sense

<table>
<thead>
<tr>
<th>Freq</th>
<th>Synset</th>
<th>Gloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>338</td>
<td>plant¹, works, industrial plant</td>
<td>buildings for carrying on industrial labor</td>
</tr>
<tr>
<td>207</td>
<td>plant², flora, plant life</td>
<td>a living organism lacking the power of locomotion</td>
</tr>
<tr>
<td>2</td>
<td>plant³</td>
<td>something planted secretly for discovery by another</td>
</tr>
<tr>
<td>0</td>
<td>plant⁴</td>
<td>an actor situated in the audience whose acting is rehearsed but seems spontaneous to the audience</td>
</tr>
</tbody>
</table>

- 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 ← COMPUTE_OVERLAP(signature, context)
  if overlap > max-overlap then
    max-overlap ← overlap
    best-sense ← sense
  end
return(best-sense)
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, ie hyponyms
  2. Apply a weight to each overlapping word

\[ idf_i = \log \left( \frac{N_{doc}}{n_{di}} \right) \]

where:  \( N_{doc} \) is the number of documents in a corpus  
\( n_{di} \) 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

\[ \text{sim}_{\text{path}}(c_1, c_2) = -\log \text{pathlen}(c_1, c_2) \]

\[ \text{pathlen}(c_1, c_2) = \text{number of edges the shortest path in thesaurus graph between synsets } c_1, c_2 \]

\[ \text{wordsim}(w_1, w_2) = \max_{c_1 \in \text{senses}(w_1)} \max_{c_2 \in \text{senses}(w_2)} \text{sim}(c_1, c_2) \]
Word Similarity

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

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

where $\text{words}(c)$ set of words in corpus that are present in the thesaurus

From information theory, use the definition of Information Content $\text{IC}$ 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

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

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

\[ \text{sim}_{\text{resnik}}(c_1, c_2) = -\log P(LCS(c_1, c_2)) \]

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

commonality

\[ \text{IC}(\text{common}(A, B)) \]

difference

\[ \text{IC}(\text{description}(A, B)) - \text{IC}(\text{common}(A, B)) \]

where description(A,B) describes A and B
The information in common between two concepts is twice the information in their LCS(c1,c2)

\[
\text{sim}_{\text{Lin}}(c_1, c_2) = \frac{2 \times \log P(\text{LCS}(c_1, c_2))}{\log P(c_1) + \log P(c_2)}
\]

\[
\text{sim}_{\text{Lin}}(c_1, c_2) = \frac{2 \times \log P(\text{geological} - \text{formation})}{\log P(\text{hill}) + \log P(\text{coast})} = 0.59
\]

Jiang-Conrath distance is similar

\[
\text{dist}_{\text{JC}}(c_1, c_2) = 2 \times \log P(\text{LCS}(c_1, c_2)) - (\log P(c_1) + \log P(c_2))
\]
Word Similarity

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

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

$$sim_{Lesk}(c_1, c_2) = \sum_{r,q \in \text{RELS}} \text{overlap}(\text{gloss}(r(c_1)), \text{gloss}(q(c_2)))$$
Word Similarity - Summary

\[
\begin{align*}
\text{sim}_{\text{path}}(c_1, c_2) &= -\log \text{pathlen}(c_1, c_2) \\
\text{sim}_{\text{Resnik}}(c_1, c_2) &= -\log P(\text{LCS}(c_1, c_2)) \\
\text{sim}_{\text{Lin}}(c_1, c_2) &= \frac{2 \times \log P(\text{LCS}(c_1, c_2))}{\log P(c_1) + \log P(c_2)} \\
\text{sim}_{\text{jc}}(c_1, c_2) &= \frac{1}{2 \times \log P(\text{LCS}(c_1, c_2)) - (\log P(c_1) + \log P(c_2))} \\
\text{sim}_{\text{eLesk}}(c_1, c_2) &= \sum_{r, q \in \text{RELS}} \text{overlap}(\text{gloss}(r(c_1)), \text{gloss}(q(c_2)))
\end{align*}
\]
Word Similarity: Distributional Methods

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

A bottle of *tezgu·ino* is on the table.
Everybody likes *tezgu·ino*.
*Tezgu·ino* makes you drunk.
We make *tezgu·ino* 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

\[
\bar{w} = (f_1, f_2, \ldots, f_n)
\]

<table>
<thead>
<tr>
<th></th>
<th>arts</th>
<th>boil</th>
<th>data</th>
<th>function</th>
<th>large</th>
<th>sugar</th>
<th>summarized</th>
<th>water</th>
</tr>
</thead>
<tbody>
<tr>
<td>apricot</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>pineapple</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>digital</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>information</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Word co-occurrence vector

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

*I discovered dried tangerines:*

<table>
<thead>
<tr>
<th>Term</th>
<th>Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>discover</td>
<td>subj-of discover</td>
</tr>
<tr>
<td>tangerine</td>
<td>obj-of discover</td>
</tr>
<tr>
<td>dried</td>
<td>adj-mod-of tangerine</td>
</tr>
<tr>
<td>tangerine</td>
<td>adj-mod dried</td>
</tr>
</tbody>
</table>
Word co-occurrence vector

- Co-occurrence vector for the word *cell*

<table>
<thead>
<tr>
<th></th>
<th>subj-of, absorb</th>
<th>subj-of, adapt</th>
<th>subj-of, behave</th>
<th>...</th>
<th>pobj-of, inside</th>
<th>pobj-of, into</th>
<th>...</th>
<th>nmod-of, abnormality</th>
<th>nmod-of, anemia</th>
<th>nmod-of, architecture</th>
<th>obj-of, attack</th>
<th>obj-of, call</th>
<th>obj-of, come from</th>
<th>obj-of, decorate</th>
<th>...</th>
<th>nnmod, bacteria</th>
<th>nnmod, body</th>
<th>nnmod, bone marrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>cell</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>16</td>
<td>30</td>
<td></td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>6</td>
<td>11</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
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.

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

$$ \operatorname{assoc}_{PMI}(w, f) = \log_2 \frac{P(w, f)}{P(w)P(f)} $$
Lin Association – breaks $P(f)$ further down into relation $r$ and word $w'$ – at the other end of relation $r$.

$$assoc_{Lin}(w, f) = \log_2 \frac{P(w, f)}{P(w)P(r|w)P(w'|w)}$$
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.

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

\[
distance_{\text{manhattan}}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} |x_i - y_i|
\]

\[
distance_{\text{euclidean}}(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}
\]
Similarity Between two vectors

\[ \text{Euclidean}(\vec{a}, \vec{b}) = L_2(\vec{a}, \vec{b}) \]

\[ \text{Manhattan}(\vec{a}, \vec{b}) = L_1(\vec{a}, \vec{b}) \]
Information Retrieval Word Similarity

\[ \text{sim}_{\text{dot-product}}(\vec{v}, \vec{w}) = \vec{v} \cdot \vec{w} = \sum_{i=1}^{N} v_i \times w_i \]

- Define a vector for a target word with N features \( f_1, \ldots, f_N \).

\[ \vec{w} = (\text{assoc}(w, f_1), \text{assoc}(w, f_2), \text{assoc}(w, f_3), \ldots, \text{assoc}(w, f_N)) \]

- 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

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

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

\[
\text{sim}_{\text{Dice}}(\vec{v}, \vec{w}) = \frac{2 \times \sum_{i=1}^{N} \min(v_i, w_i)}{\sum_{i=1}^{N} (v_i + w_i)}
\]
### Word Similarity

<table>
<thead>
<tr>
<th>Formula</th>
<th>Description</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{assoc}_{\text{prob}}(w, f) )</td>
<td>Probabilityassoc</td>
<td>( P(f</td>
</tr>
<tr>
<td>( \text{assoc}_{\text{PMI}}(w, f) )</td>
<td>PMI</td>
<td>( \log_2 \frac{P(w,f)}{P(w)P(f)} ) (20.38)</td>
</tr>
<tr>
<td>( \text{assoc}_{\text{Lin}}(w, f) )</td>
<td>Linear assoc</td>
<td>( \log_2 \frac{P(w,f)}{P(w)P(r</td>
</tr>
<tr>
<td>( \text{assoc}_{\text{t-test}}(w, f) )</td>
<td>T-test assoc</td>
<td>( \frac{P(w,f) - P(w)P(f)}{\sqrt{P(f)P(w)}} ) (20.41)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Formula</th>
<th>Description</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{sim}_{\text{cosine}}(\vec{v}, \vec{w}) )</td>
<td>Cosine similarity</td>
<td>( \frac{\vec{v} \cdot \vec{w}}{</td>
</tr>
<tr>
<td>( \text{sim}_{\text{Jaccard}}(\vec{v}, \vec{w}) )</td>
<td>Jaccard similarity</td>
<td>( \frac{\sum_{i=1}^{N} \min(v_i, w_i)}{\sum_{i=1}^{N} \max(v_i, w_i)} ) (20.48)</td>
</tr>
<tr>
<td>( \text{sim}_{\text{Dice}}(\vec{v}, \vec{w}) )</td>
<td>Dice similarity</td>
<td>( \frac{2 \times \sum_{i=1}^{N} \min(v_i, w_i)}{\sum_{i=1}^{N} (v_i + w_i)} ) (20.49)</td>
</tr>
<tr>
<td>( \text{sim}_{\text{JS}}(\vec{v}</td>
<td></td>
<td>\vec{w}) )</td>
</tr>
</tbody>
</table>
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 EVALUTEE 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.

```python
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

NP-SBJ = ARG0

S

VP

NP = ARG1

VBD = TARGET

issued

DT JJ NN IN NP

a special edition around NN NP-TMP

nn noon yesterday

The San Francisco Examiner

DT NNP NNP NNP

2016 Dan I. Moldovan, Human Language Technology Research Institute, The University of Texas at Dallas
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.