Introduction

Ambiguity is a key problem in *lexical semantics*. Two important forms of lexical ambiguity are:

**Polysemy:**
- a word has more than one meaning;
- example: *plant* as industrial complex or as organism.

**Homonymy:**
- two or more unrelated words happen to be spelled the same;
- example: *can* as a verb or noun.

Distinction between polysemy and homonymy not always clear.

Material in this section adapted from Philipp Koehn’s ANLP lectures.
Example for a polysemous word: how many senses does the **interest** have?

(1) a. She pays 3% **interest** on the loan.
   b. He showed a lot of interest in the painting.
   c. Microsoft purchased a controlling **interest** in Google.
   d. It is in the national interest to invade the Bahamas.
   e. I only have your best **interest** in mind.
   f. Playing chess is one of my interests.
   g. Business interests lobbied for the legislation.

Wordnet lists 7 sense for **interest**.
Example for a polysemous word: how many senses does the *interest* have?

(1) a. She pays 3% interest on the loan.
   b. He showed a lot of interest in the painting.
   c. Microsoft purchased a controlling interest in Google.
   d. It is in the national interest to invade the Bahamas.
   e. I only have your best *interest* in mind.
   f. Playing chess is one of my interests.
   g. Business *interests* lobbied for the legislation.

Wordnet lists 7 sense for *interest*.
Word sense disambiguation (WSD):
- for many applications, we would like to disambiguate word senses; for example:
  - we may be only interested in one sense; searching for chemical plant; don’t want documents about chemicals in bananas;
  - task: given a polysemous word, find the correct sense;

WSD as a classification task with features such as:
- context words;
- syntactically related words;
- POS tag of the word and the context words.

Here we focus on unsupervised WSD.

Yarowsky’s (1995) algorithm uses two powerful heuristics for WSD:
- One sense per collocation: nearby words provide clues to the sense of the target word, conditional on distance, order, syntactic relationship.
- One sense per discourse: the sense of a target word is consistent within a given document.

The Yarowsky algorithm is a bootstrapping algorithm, i.e., it requires a small amount of annotated data.

---

**Step 1:** extract all instances of a polysemous word.

**Step 2:** generate a seed set of labeled examples:
- either by manually labeling them (semi-supervised);
- or by using a reliable heuristic (unsupervised).

Example: target word plant: As seed set take all instances of plant life (sense A) and manufacturing plant (sense B).
Step 3a: train classifier on the seed set.

Step 3b: apply classifier to the entire sample set. Add those examples that are classified reliably (probability above a threshold) to the seed set.

Yarowsky uses a decision list classifier:

- rules of the form: collocation $\rightarrow$ sense
- rules are ordered by log-likelihood:

$$\log \frac{P(sense_A|collocation)}{P(sense_B|collocation)}$$

- classification is based on the first rule that applies.

### Classification

**Initial decision list for plant (abbreviated)**

<table>
<thead>
<tr>
<th>LogL</th>
<th>Collocation</th>
<th>Sense</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.10</td>
<td>plant life</td>
<td>$\Rightarrow$ A</td>
</tr>
<tr>
<td>7.58</td>
<td>manufacturing plant</td>
<td>$\Rightarrow$ B</td>
</tr>
<tr>
<td>7.39</td>
<td>life (within ±2-10 words)</td>
<td>$\Rightarrow$ A</td>
</tr>
<tr>
<td>7.20</td>
<td>manufacturing (in ±2-10 words)</td>
<td>$\Rightarrow$ B</td>
</tr>
<tr>
<td>6.27</td>
<td>animal (within ±2-10 words)</td>
<td>$\Rightarrow$ A</td>
</tr>
<tr>
<td>4.70</td>
<td>equipment (within ±2-10 words)</td>
<td>$\Rightarrow$ B</td>
</tr>
<tr>
<td>4.39</td>
<td>employee (within ±2-10 words)</td>
<td>$\Rightarrow$ B</td>
</tr>
<tr>
<td>4.30</td>
<td>assembly plant</td>
<td>$\Rightarrow$ B</td>
</tr>
<tr>
<td>4.10</td>
<td>plant closure</td>
<td>$\Rightarrow$ B</td>
</tr>
<tr>
<td>3.52</td>
<td>plant species</td>
<td>$\Rightarrow$ A</td>
</tr>
<tr>
<td>3.48</td>
<td>automate (within ±2-10 words)</td>
<td>$\Rightarrow$ B</td>
</tr>
<tr>
<td>3.45</td>
<td>microscopic plant</td>
<td>$\Rightarrow$ A</td>
</tr>
</tbody>
</table>

**Step 3c:** use one-sense-per-discourse constraint to filter newly classified examples:

- if several examples have already been annotated as sense A, then extend this to all examples of the word in the discourse;
- this can form a bridge to new collocations, and correct erroneously labeled examples.

**Step 3d:** repeat Steps 3a–d.
Step 4: algorithm converges on a stable residual set (remaining unlabeled instances):

- most training examples will now exhibit multiple collocations indicative of the same sense;
- decision list procedure uses only the most reliable rule, not a combination of rules.

Step 5: the final classifier can now be applied to unseen data.
Discussion

Strengths:
- simple algorithm that uses only minimal features (words in the context of the target word);
- minimal effort required to create seed set;
- does not rely on dictionary or other external knowledge.

Weaknesses:
- uses very simple classifier (but could replace it with a more state-of-the-art one), small number of senses;
- not fully unsupervised: requires seed data;
- does not make use of the structure of the sense inventory.

Alternative: graph-based algorithms exploit the structure of the sense inventory for WSD.

Introduction

Navigli and Lapata’s (2010) algorithm is an example of graph-based WSD.

It exploits the fact that a sense inventory is required for WSD. Most sense inventories have an internal structure, which can be exploited to build a graph.

Example: synsets (senses) of drink in Wordnet:

\[(2) \quad \begin{align*}
    & a. \, \{\text{drink}^1, \text{imbibe}^3\} \\
    & b. \, \{\text{drink}^2, \text{booz}^1, \text{fuddle}^2\} \\
    & c. \, \{\text{toast}^2, \text{drink}^3, \text{pledge}^e, \text{salute}^1, \text{wassail}^2\}
\end{align*}\]

Wordnet encodes a range of lexical and semantic relations:
- lexical: nominalization, antonymy, pertainymy;
- semantic: hyponymy, hypernymy, meronymy;
- other ones can be induced, e.g., gloss.

We can represent Wordnet as a graph whose nodes are synsets and whose edges are relations between synsets.

Note that the edges are not labeled, i.e., the type of relation between the nodes is ignored.
Graph Construction

Disambiguation algorithm:

1. Use the Wordnet graph to construct a graph that incorporates each content word in the sentence to be disambiguated;
2. Rank each node in the sentence graph according to its importance using graph connectivity measures;
3. For each content word, pick the highest ranked sense as the correct sense of the word.

Given a word sequence $\sigma = (w_1, w_2, \ldots, w_n)$, the graph $G$ is constructed as follows:

1. Let $V_\sigma := \bigcup_{i=1}^n \text{Senses}(w_i)$ denote all possible word senses in $\sigma$.
   We set $V := V_\sigma$ and $E := \emptyset$.
2. For each node $v \in V_\sigma$, we perform a depth-first search (DFS) of the Wordnet graph: every time we encounter a node $v' \in V_\sigma$ ($v' \neq v$) along a path $v \rightarrow v_1 \rightarrow \cdots \rightarrow v_k \rightarrow v'$ of length $L$, we add all intermediate nodes and edges on the path from $v$ to $v'$: $V := V \cup \{v_1, \ldots, v_k\}$ and $E := E \cup \{\{v, v_1\}, \ldots, \{v_k, v'\}\}$.

For tractability, we fix the maximum path length at 6.

Example: graph for drink milk.
Example: graph for *drink milk*.

We get $3 \times 2 = 6$ interpretations, i.e., subgraphs obtained when only considering one connected sense of *drink* and *milk*.
Example: graph for `drink milk`.

We get $3 \cdot 2 = 6$ interpretations, i.e., subgraphs obtained when only considering one connected sense of `drink` and `milk`.

Once we have the graph, we pick the most connected node for each word as the correct sense. Two types of connectivity measures:

- **Local measures**: give a connectivity score to an individual node in the graph; use this directly to pick a sense;
- **Global measures**: assign a connectivity score to the graph as a whole; apply the measure to each interpretation and select the highest scoring one.

Navigli and Lapata (2010) discuss a large number of graph connectivity measures; we will focus on the most important ones.

**Degree Centrality**

Assume a graph with nodes $V$ and edges $E$. Then the degree of $v \in V$ is the number of edges terminating in it:

$$\deg(v) = |\{u, v \in E : u \in V\}| \quad (1)$$

**Degree centrality** is the degree of a node normalized by the maximum degree:

$$C_D(v) = \frac{\deg(v)}{|V| - 1} \quad (2)$$

For the previous example, $C_D(drink_1^v) = \frac{3}{14}$, $C_D(drink_2^v) = C_D(drink_5^v) = \frac{2}{14}$, and $C_D(milk_1^v) = C_D(milk_2^v) = \frac{1}{14}$. So we pick `drink_1^v`, while `milk_1^v` is tied.
**Edge Density**

The edge density of a graph is the number of edges compared to a complete graph with \(|V|\) nodes (given by \(\binom{|V|}{2}\)):

\[
ED(G) = \frac{|E(G)|}{\binom{|V|}{2}}
\]  

(3)

The first interpretation of *drink milk* has \(ED(G) = \frac{6}{\binom{5}{2}} = \frac{6}{10} = 0.60\), the second one \(ED(G) = \frac{5}{\binom{3}{2}} = \frac{5}{10} = 0.50\).

### Evaluation on SemCor

<table>
<thead>
<tr>
<th>Measure</th>
<th>WordNet</th>
<th>EnWordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All</td>
<td>Poly</td>
</tr>
<tr>
<td>Degree</td>
<td>50.01</td>
<td>37.80</td>
</tr>
<tr>
<td>PageRank</td>
<td>49.76</td>
<td>37.49</td>
</tr>
<tr>
<td>HITS</td>
<td>44.29</td>
<td>30.69</td>
</tr>
<tr>
<td>KPP</td>
<td>47.89</td>
<td>35.16</td>
</tr>
<tr>
<td>Betweenness</td>
<td>48.72</td>
<td>36.20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measure</th>
<th>WordNet</th>
<th>EnWordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All</td>
<td>Poly</td>
</tr>
<tr>
<td>Compactness</td>
<td>43.53</td>
<td>30.74</td>
</tr>
<tr>
<td>Graph Entropy</td>
<td>42.98</td>
<td>29.06</td>
</tr>
<tr>
<td>Edge Density</td>
<td>43.54</td>
<td>29.76</td>
</tr>
<tr>
<td>First Sense</td>
<td>74.17</td>
<td>68.80</td>
</tr>
</tbody>
</table>

### Discussion

**Strengths:**
- exploits the structure of the sense inventory/dictionary;
- conceptually simple, doesn’t require any training data, not even a seed set;
- achieves good performance for unsupervised system.

**Weaknesses:**
- performance not good enough for real applications (F-score of 53 on Semeval);
- sense inventories take a lot of effort to create (Wordnet has been under development for more than 15 years).
Word-sense disambiguation is the task of assigning the correct sense to an ambiguous word;
the Yarowsky algorithm does WSD using two key heuristics: one sense per collocation, one sense per discourse;
it starts with a small seed set, trains a classifier on it, and then applies it to the whole data set (bootstrapping);
reliable examples are kept, and the classifier is re-trained;
alternative: unsupervised graph-based WSD; exploits the connectivity of the sense inventory;
a graph is constructed that represents the possible interpretations of a sentence; the nodes with the highest connectivity are picked as correct senses;
a range of connectivity measures exists, simple degree is best.
