Multi-Component Word Sense Disambiguation

Massimiliano Ciaramita and Mark Johnson

Brown University

BLLIP: http://www.cog.brown.edu/Research/nlp
Outline

- **Pattern classification for WSD**
  - Features
  - Flat multiclass averaged perceptron
- **Multi-component WSD**
  - Generating external training data
  - Multi-component perceptron
- **Experiments and results**
Pattern classification for WSD

English lexical sample: 57 test words: 32 verbs, 20 nouns, 5 adjectives. For each word $w$:

1. compile a training set: $S(w) = (x_i, y_i)^n$
   - $x_i \in \mathbb{R}^d$ a vector of features
   - $y_i \in Y(w)$, one of the possible senses of $w$

2. learn a classifier on $S(w)$: $H : \mathbb{R}^d \rightarrow Y(w)$

3. use the classifier to disambiguate the unseen test data
Features

- **Standard feature set for wsd** (derived from (Yoong and Hwee, 2002))
  
  “A-DT newspaper-NN and-CC now-RB a-DT bank-NN have-AUX since-RB taken-VBN over-RB”

- **POS of neighboring words** - \( P_{x,x \in \{-3,-2,-1,0,+1,+2,+3\}} \); e.g., \( P_{-1} = DT \), \( P_0 = NN \), \( P_{+1} = AUX \), ...

- **Surrounding words** - \( WS \); e.g., \( WS = take_v \), \( WS = over_r \), \( WS = newspaper_n \)

- **N-grams**:
  
  - \( NG_{x,x \in \{-2,-1,1,+1,1,2\}} \); e.g., \( NG_{-2} = now \), \( NG_{+1} = have \), \( NG_{+2} = take \)
  
  - \( NG_{x,y:(x,y) \in \{(-2,-1),(-1,+1),(+1,+2)\}} \); e.g., \( NG_{-2,-1} = now_a \), \( NG_{+1,+2} = have\_since \)
Syntactic features (Charniak, 2000)

- **Governing elements under a phrase** - $G_1$; e.g., $G_1 = \text{take}_S$
- **Governed elements under a phrase** - $G_2$; e.g., $G_2 = \text{a}_\text{NP}$, $G_2 = \text{now}_\text{NP}$
- **Coordinates** - $\text{OO}$; e.g., $\text{OO} = \text{newspaper}$
Multiclass Perceptron (Crammer and Singer, 2003)

- **Discriminant function**: \( H(x; V) = \arg\max_k \langle v_k, x \rangle \)
- **Input**: \( V \in \mathbb{R}^{|Y(w)| \times d} \), \( d \approx 200,000 \), initialized as \( V = 0 \)
- **Repeat** \( T \) times - passes over training data or epochs

Multiclass Perceptron\(((x, y)^n, V)\)

1. for \( i = 1 \) to \( i = n \)
2. do \( E = \{ r : \langle v_r, x_i \rangle > \langle v_y, x_i \rangle \} \)
3. if \( |E| > 0 \)
   4. then 1. \( \tau_r = 1 \) for \( r = y \)
      5. 2. \( \tau_r = 0 \) for \( r \in E \cup \{y\} \)
      6. 3. \( \tau_r = -\frac{1}{|E|} \) for \( r \in E \)
4. for \( r = 1 \) to \( r = k \)
5. do \( v_r \leftarrow v_r + \tau_r x_i; \)
Averaged perceptron classifier

- Perceptron’s output: $V^{(0)}, \ldots, V^{(n)}$
- $V^{(i)}$ is the weight matrix after the first $i$ training items
- Final model: $V = V^{(n)}$
- **Averaged perceptron**: (Collins, 2002)
  - final model: $V = \frac{1}{n} \sum_{i=1}^{n} V^{(i)}$
  - reduces the effect of over-training
Outline

- Pattern classification for WSD
  - Features
  - Flat multiclass perceptron
- Multi-component WSD
  - Generating external training data
  - Multi-component perceptron
- Experiments and results
Sparse data problem in WSD

- **Thousands of word senses** - 120,000 in Wordnet 2.0
- **Very specific classes** - 50% of noun synsets contain one noun
- **Problem**: training instances often too few for fine-grained semantic distinctions

**Solution:**

1. use the hierarchy of Wordnet to find similar word senses and generate external training data for these senses
2. integrate task-specific and external data with perceptron

**Intuition** - to classify an instance of the noun *disk* additional knowledge about concepts such as other “audio” or “computer memory” devices could be helpful
Finding neighbor senses

- \(\text{disc}_1 = \text{memory device for information storing}\)
- \(\text{disc}_2 = \text{phonograph record}\)
Finding neighbor senses

- $$\text{neighbors}(\text{disc}_1) = \text{floppy disk}, \text{hard disk}, \ldots$$
- $$\text{neighbors}(\text{disc}_2) = \text{audio recording}, \text{lp}, \text{soundtrack}, \text{audiotape}, \text{talking book}, \text{digital audio tape}, \ldots$$
External training data

- **Find neighbors**: for each sense $y$ of a noun or verb in the task a set $\hat{y}$ of $k = 100$ neighbor senses is generated from the Wordnet hierarchy.

- **Generate new instances**: for each synset in $\hat{y}$ a training instance $(x_i, \hat{y}_i)$ is compiled from the corresponding Wordnet glosses (definitions/example sentences) using the same set of features.

- **Result**: for each noun/verb

  1. task-specific training data $(x_i, y_i)^n$
  2. external training data $(x_i, \hat{y}_i)^m$
Multi-component perceptron

- Simplification of hierarchical perceptron (Ciaramita et al., 2003)
- A weight matrix $V$ is trained on the task-specific data
- A weight matrix $M$ is trained on the external data
- Discriminant function:

$$H(x; V, M) = \arg \max_{y \in \mathcal{Y}(w)} \lambda_y \langle v_y, x \rangle + \lambda_y^\varphi \langle m_{\varphi}, x \rangle$$

$-$ $\lambda_y$ is an adjustable parameter that weights each component’s contribution: $\lambda_y^\varphi = 1 - \lambda_y$
Multi-Component Perceptron

- The algorithm learns $V$ and $M$ independently

\begin{verbatim}
Multi-Component_Perceptron((x_i, y_i)^n, (x_i, \hat{y}_i)^m, V, M)
1  V ← 0
2  M ← 0
3  for t = 1 to i = T
4    do Multiclass_Perceptron((x_i, y_i)^n, V)
5        Multiclass_Perceptron((x_i, y_i)^n, M)
6        Multiclass_Perceptron((x_i, y_i)^m, M)
\end{verbatim}
Outline

• Pattern classification for WSD
  – Features
  – Flat multiclass averaged perceptron

• Multi-component WSD
  – Generating external training data
  – Multi-component perceptron

• Experiments and results
Experiments and results

- One classifier trained for each test word
- **Adjectives**: standard perceptron, only set $T$
- **Verbs/nouns**: multicomponent perceptron, set $T$ and $\lambda_y$
- **Cross-validation** experiments on the training data for each test word:
  1. choose the value for $\lambda_y$; $\lambda_y = 1$ use only the “flat” perceptron, or $\lambda_y = 0.5$ use both component equally weighted
  2. choose the number of iterations $T$
- Average $T$ value = 13.9
- For 37 out of 52 nouns/verbs $\lambda_y = 0.5$; the two-component model is more accurate than the flat perceptron
## English Lexical Sample Results

<table>
<thead>
<tr>
<th>Measure</th>
<th>Precision</th>
<th>Recall</th>
<th>Attempted %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine all POS</td>
<td>71.1</td>
<td>71.1</td>
<td>100</td>
</tr>
<tr>
<td>Coarse all POS</td>
<td>78.1</td>
<td>78.1</td>
<td>100</td>
</tr>
<tr>
<td>Fine verbs</td>
<td>72.5</td>
<td>72.5</td>
<td>100</td>
</tr>
<tr>
<td>Coarse verbs</td>
<td>80.0</td>
<td>80.0</td>
<td>100</td>
</tr>
<tr>
<td>Fine nouns</td>
<td>71.3</td>
<td>71.3</td>
<td>100</td>
</tr>
<tr>
<td>Coarse nouns</td>
<td>77.4</td>
<td>77.4</td>
<td>100</td>
</tr>
<tr>
<td>Fine adjectives</td>
<td>49.7</td>
<td>49.7</td>
<td>100</td>
</tr>
<tr>
<td>Coarse adjectives</td>
<td>63.5</td>
<td>63.5</td>
<td>100</td>
</tr>
</tbody>
</table>
Flat vs. Multi-component: cross validation on train

Ciaramita and Johnson
Conclusion

- **Advantages** of the multi-component perceptron trained on neighbors’ data
  - **Neighbors**: one “supersense” for each sense, same amount of additional data per sense
  - **Simpler model**: smaller variance more homogeneous external data
  - **Efficiency**: fast and efficient training
  - **Architecture**: simple, easy to add any number of (weighted) “components”