Lexical Semantics

Regina Barzilay
MIT

October, 5766
Last Time: Vector-Based Similarity Measures

- Euclidean: \( |\vec{x}, \vec{y}| = |\vec{x} - \vec{y}| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \)

- Cosine: \( \cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| |\vec{y}|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}} \)
Last Time: Probabilistic Similarity Measures

(Pointwise) Mutual Information: \( I(x; y) = \log \frac{P(x, y)}{P(x)P(y)} \)

- Mutual Information: \( I(X; Y) = E_{p(x,y)} \log \frac{p(X,Y)}{p(X)p(Y)} \)
Example: Computing MI

<table>
<thead>
<tr>
<th>$I(w_1, w_2)$</th>
<th>$C(w_1)$</th>
<th>$C(w_2)$</th>
<th>$C(w_1, w_2)$</th>
<th>$w_1$</th>
<th>$w_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.31</td>
<td>30</td>
<td>117</td>
<td>20</td>
<td>Agatha</td>
<td>Christie</td>
</tr>
<tr>
<td>15.94</td>
<td>77</td>
<td>59</td>
<td>20</td>
<td>videocassette</td>
<td>recorder</td>
</tr>
<tr>
<td>15.19</td>
<td>24</td>
<td>320</td>
<td>20</td>
<td>unsalted</td>
<td>butter</td>
</tr>
<tr>
<td>1.09</td>
<td>14907</td>
<td>9017</td>
<td>20</td>
<td>first</td>
<td>made</td>
</tr>
<tr>
<td>0.29</td>
<td>15019</td>
<td>15629</td>
<td>20</td>
<td>time</td>
<td>last</td>
</tr>
</tbody>
</table>
Example: Computing MI

<table>
<thead>
<tr>
<th>$I(w_1, w_2)$</th>
<th>$C(w_1)$</th>
<th>$C(w_2)$</th>
<th>$C(w_1, w_2)$</th>
<th>$w_1$</th>
<th>$w_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.02</td>
<td>1</td>
<td>19</td>
<td>1</td>
<td>fewest</td>
<td>visits</td>
</tr>
<tr>
<td>12.00</td>
<td>5</td>
<td>31</td>
<td>1</td>
<td>Indonesian</td>
<td>pieces</td>
</tr>
<tr>
<td>9.21</td>
<td>13</td>
<td>82</td>
<td>20</td>
<td>marijuana</td>
<td>growing</td>
</tr>
</tbody>
</table>
Last Time: Probabilistic Similarity Measures

Kullback Leibler Distance: \( D(p||q) = \sum p(x) \log \frac{p(x)}{q(x)} \)

- Closely related to mutual information

\[
I(X; Y) = D(p(x, y)||p(x)p(y))
\]

- Related measure: Jensen-Shannon divergence:

\[
D_{JS}(p, q) = \frac{1}{2} D(p||\frac{p+q}{2}) + \frac{1}{2} D(q||\frac{p+q}{2})
\]
Beyond Pairwise Similarity

- Clustering is “The art of finding groups in data” (Kaufmann and Rousseeu).
- Clustering algorithms divide a data set into homogeneous groups (clusters), based on their similarity under the given representation.
Hierarchical Clustering

Greedy, bottom-up version:

- Initialization: Create a separate cluster for each object
- Each iteration: Find two most similar clusters and merge them
- Termination: All the objects are in the same cluster
Agglomerative Clustering

A
B
C
D
E

A  0.1  0.2  0.2  0.8
B  0.1  0.1  0.2
C  0.0  0.7
D  0.6

A  B  C  D  E
## Agglomerative Clustering

<table>
<thead>
<tr>
<th></th>
<th>E</th>
<th>D</th>
<th>C</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.8</td>
</tr>
<tr>
<td>B</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0.0</td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The dendrogram shows the clustering process:

- A and D are first clustered together.
- B is then clustered with the A-D group.
- C is clustered with the B-A-D group.
- E is finally clustered with the C-B-A-D group.
Agglomerative Clustering

E  D  C  B
A  0.1  0.2  0.2  0.8
B  0.1  0.1  0.2
C  0.0  0.7
D  0.6
Clustering Function

E  D  C  B
A  0.1 0.2 0.2 0.8
B  0.1 0.1 0.2
C  0.0 0.7
D  0.6
## Clustering Function

<table>
<thead>
<tr>
<th>E</th>
<th>D</th>
<th>C</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>B</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>C</td>
<td>0.0</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0.6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Clustering Diagram]
### Clustering Function

<table>
<thead>
<tr>
<th></th>
<th>E</th>
<th>D</th>
<th>C</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.8</td>
</tr>
<tr>
<td>B</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0.0</td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Diagram showing clustering function with nodes A, B, C, D, E and distances]
Clustering Function

- **Single-link**: Similarity of two most similar members
- **Complete-link**: Similarity of two least similar members
- **Group-average**: Average similarity between members
Single-Link Clustering

- Achieves Local Coherence
- Complexity $O(n^2)$
- Fails when clusters are not well separated
Complete-Link Clustering

- Achieves Global Coherence
- Complexity $O(n^2 \log n)$
- Fails when clusters aren’t spherical, or of uniform size
K-Means Algorithm: Example

Iterative, hard, flat clustering algorithm based on Euclidean distance
K-Means Algorithm

1. Choose $k$ points at random as cluster centers

2. Assign each instance to its closest cluster center

3. Calculate the centroid (mean) for each cluster, use it as a new cluster center

4. Iterate steps 2 and 3 until the cluster centers don’t change anymore
K-Means Algorithm: Hard EM

1. Guess initial parameters
2. Use model to make the best guess of $c_i$ (E-step)
3. Use the new complete data to learn better model (M-step)
4. Iterate (2-3) until convergence
Evaluating Clustering Methods

- Perform task-based evaluation
- Test the resulting clusters intuitively, i.e., inspect them and see if they make sense. Not advisable.
- Have an expert generate clusters manually, and test the automatically generated ones against them.
- Test the clusters against a predefined classification if there is one
Comparing Clustering Methods

(Meila, 2002)

\[ n \quad \text{total \# of points} \]
\[ n_k \quad \text{\# of points in cluster } C_k \]
\[ K \quad \text{\# of nonempty clusters} \]
\[ N_{11} \quad \text{\# of pairs that are in the same cluster under } C \text{ and } C' \]
\[ N_{00} \quad \text{\# of pairs that are in different clusters under } C \text{ and } C' \]
\[ N_{10} \quad \text{\# of pairs that are in the same cluster under } C \text{ but not } C' \]
\[ N_{01} \quad \text{\# of pairs that are in the same cluster under } C' \text{ but not } C \]
Comparing by Counting Pairs

• Wallace criteria

\[ W_1(C, C') = \frac{N_{11}}{\sum_k n_k (n_k - 1)/2} \]

\[ W_2(C, C') = \frac{N_{11}}{\sum_{k'} n_{k'} (n'_{k'} - 1)/2} \]

• Fowles-Mallows criterion

\[ F(C, C') = \sqrt{W_1(C, C') W_2(C, C')} \]

Problems: ?
Comparing Clustering by Set Matching

Contingency table $M$ is a $K \times K$ matrix, whose $kk'$ element is the number of points in the intersection of clusters $C_k$ and $C'_{k'}$.

$$L(C, C') = \frac{1}{K} \sum_k \max_{k'} \frac{2m_{kk'}}{n_k + n'_{k}}$$

Problems: ?
Comparing Clustering by Set Matching

$$L(C, C') = \frac{1}{K} \sum_k \max_{k'} \frac{2m_{kk'}}{n_k + n_k'}$$
Distributional Syntax

Sequences of word clusters and their contexts (Klein, 2005)

<table>
<thead>
<tr>
<th>Tag</th>
<th>Top Context by Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>(IN-NN), (IN-JJ), (IN-NNP), (VB-NN)</td>
</tr>
<tr>
<td>JJ</td>
<td>(DT-NN), (IN-NNS), (IN-NN), (JJ-NN), (DT-NNS)</td>
</tr>
<tr>
<td>MD</td>
<td>(NN-VB), (PRP-VB), (NNS-VB), (NNP-VB), (WDT-VB)</td>
</tr>
<tr>
<td>NN</td>
<td>(DT-IN), (JJ-IN), (DT-NN), (NN-IN), (NN-.)</td>
</tr>
<tr>
<td>VB</td>
<td>(TO-DT), (TO-IN), (MD-DT), (MD-VBN), (TO-JJ)</td>
</tr>
</tbody>
</table>
**Distributional Syntax**

The most similar POS pairs and POS sequence pairs based on $D_{JS}$ of their context

<table>
<thead>
<tr>
<th>Rank</th>
<th>Tag pairs</th>
<th>Sequence Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(VBZ,VBD)</td>
<td>(NNP NNP, NNP NNP NNP NNP)</td>
</tr>
<tr>
<td>2</td>
<td>(DT,PRP$)</td>
<td>(DT JJ NN IN, DT NN IN)</td>
</tr>
<tr>
<td>3</td>
<td>(NN,NNS)</td>
<td>(NNP NNP NNP NNP NNP, NNP NNP NNP NNP NNP)</td>
</tr>
<tr>
<td>4</td>
<td>(WDT,WP)</td>
<td>(DT NNP NNP, DT NNP)</td>
</tr>
<tr>
<td>5</td>
<td>(VBG,VBN)</td>
<td>(IN DT JJ NN, IN DT NN)</td>
</tr>
<tr>
<td>14</td>
<td>(JJS, JJR)</td>
<td>(NN IN DT, NN DT)</td>
</tr>
</tbody>
</table>
Linear vs. Hierarchical Context

The left (right) context of $x$ is the left (right) sibling of the lowest ancestor of $x$.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Linear</th>
<th>Hierarchical</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(NN NNS, JJ NNS)</td>
<td>(NN NNS, JJ NNS)</td>
</tr>
<tr>
<td>2</td>
<td>(IN NN, IN DT NN)</td>
<td>(IN NN, IN DT NN)</td>
</tr>
<tr>
<td>3</td>
<td>(DT JJ NN, DT NN)</td>
<td>(IN DT JJ NN, IN JJ NNS)</td>
</tr>
<tr>
<td>4</td>
<td>(DT JJ NN, DT NN NN)</td>
<td>(VBZ VBN, VBD VBN)</td>
</tr>
<tr>
<td>5</td>
<td>(IN DT JJ NN, IN DT NN)</td>
<td>(NN NNS, JJ NN NNS)</td>
</tr>
</tbody>
</table>
Grammar Induction

- Task: Unsupervised learning of a language’s syntax from a corpus of observed sentences

  The cat stalked the mouse.
  The mouse quivered.
  The cat smiled.

- A tree induction system is not forced to learn all aspects of language (semantics, discourse)
Motivation

• Linguistic motivation:
  – Empirical argument against the poverty of the stimulus (Chomsky, 1965)
  – Empirical investigation of syntax modularity (Fodor, 1983; Jackendoff, 1996)

• Engineering motivation:
  – No need in training data
Evaluation and Baselines

- **Evaluation:**
  - Compare grammars
  - Compare trees

- **Baselines:**
  - Random Trees
  - Left- and Right-Branching Trees
Structure Search Experiment

- Structure search
  - Add production to context free grammar
  - Select HMM topology

- Parameter search
  - Determine parameters for a fixed PCFG
Finding Topology

Stolcke & Omohundro, 1994: Bayesian model merging

- **Data incorporation**: Given a body of data \( X \), build an initial model \( M_0 \) by explicitly accommodating each data point individually such that \( M_0 \) maximizes the likelihood \( P(X|M) \).

- **Generalization**: Build a sequence of new models, obtaining \( M_{i+1} \) from \( M_i \) by applying a *merging* operator \( m \) that coalesces substructures in \( M_i \),

\[
M_{i+1} = m(M_i), \quad i = 0, 1
\]

- **Optimization**: Maximize posterior probability

- **Search strategy**: Greedy or beam search through the space of possible merges
HMM Topology Induction

- **Data incorporation:** For each observed sample create a unique path between the initial and final states by assigning a new state to each symbol token in the sample.

- **Generalization:** Two HMM states are replaced by a single new state, which inherits the union of the transitions and emissions from the old states.
HMM Topology Induction

- **Prior distribution:** Choose uninformative priors for a model $M$ with topology $M_s$ and parameters $\theta_M$.

  $$P(M) = P(M_s)P(\theta_M|M_s)$$

  $$P(M_s) \propto \exp(-l(M_s))$$

  where $l(M_s)$ is the number of bits required to encode $M_s$.

- **Search:** Greedy merging strategy.
Example
PCFG Induction

- **Data Incorporation**: Add a top-level production that covers the sample precisely. Create one nonterminal for each observed terminal.

- **Merging and Chunking**: During *merging*, two nonterminals are replaced by a single new state. *Chunking* takes a given sequence of nonterminals and abbreviates it using a newly created nonterminal.

- **Prior distribution**: Similar to HMM.

- **Search**: Beam search.
Example

Input: \{ab,aabb,aaabbb\}

\[
\begin{align*}
S & \rightarrow A B \\
   & \rightarrow A A B B \\
   & \rightarrow A A A B B B \\
A & \rightarrow a \\
B & \rightarrow b \\
\end{align*}
\]

Chunk(AB)→X
\[
\begin{align*}
S & \rightarrow X \\
   & \rightarrow A X B \\
   & \rightarrow A A X B B \\
X & \rightarrow A B \\
\end{align*}
\]

Chunk(AXB)→Y
\[
\begin{align*}
S & \rightarrow X \\
   & \rightarrow Y \\
   & \rightarrow A Y B \\
X & \rightarrow A B \\
Y & \rightarrow A X B \\
\end{align*}
\]

Merge S,Y
\[
\begin{align*}
S & \rightarrow X \\
   & \rightarrow A S B \\
X & \rightarrow A B \\
\end{align*}
\]

Merge S,X
\[
\begin{align*}
S & \rightarrow A B \\
   & \rightarrow A S B \\
\end{align*}
\]
Results for PCFGS

- Formal language experiments
  - Successfully learned simple grammars

<table>
<thead>
<tr>
<th>Language</th>
<th>Sample no.</th>
<th>Grammar</th>
<th>Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parentheses</td>
<td>8</td>
<td>$S \rightarrow ()</td>
<td>(S)</td>
</tr>
<tr>
<td>$a^{2n}$</td>
<td>5</td>
<td>$S \rightarrow aa</td>
<td>SS$</td>
</tr>
<tr>
<td>$(ab)^n$</td>
<td>5</td>
<td>$S \rightarrow ab</td>
<td>aSb$</td>
</tr>
<tr>
<td>$wcw^R, w \in {a, b}^*$</td>
<td>7</td>
<td>$S \rightarrow c</td>
<td>aaSa</td>
</tr>
<tr>
<td>Addition strings</td>
<td>23</td>
<td>$S \rightarrow a</td>
<td>b</td>
</tr>
</tbody>
</table>

- Natural Language syntax
  - Mixed results (issues related to data sparseness)
### Example of Learned Grammar

<table>
<thead>
<tr>
<th>Target Grammar</th>
<th>Learned Grammar</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S \rightarrow NP \ VP )</td>
<td>( S \rightarrow NP \ VP )</td>
</tr>
<tr>
<td>( VP \rightarrow Verb \ NP )</td>
<td>( VP \rightarrow V \ NP )</td>
</tr>
<tr>
<td>( NP \rightarrow Det \ Noun )</td>
<td>( NP \rightarrow DetN )</td>
</tr>
<tr>
<td>( NP \rightarrow Det \ Noun \ RC )</td>
<td>( NP \rightarrow NP \ RC )</td>
</tr>
<tr>
<td>( RC \rightarrow Rel \ VP )</td>
<td>( RC \rightarrow REL \ VP )</td>
</tr>
<tr>
<td>( Verb \rightarrow saw</td>
<td>heard )</td>
</tr>
<tr>
<td>( Noun \rightarrow cat</td>
<td>dog</td>
</tr>
<tr>
<td>( Det \rightarrow a</td>
<td>the )</td>
</tr>
<tr>
<td>( Rel \rightarrow that )</td>
<td>( Rel \rightarrow that )</td>
</tr>
</tbody>
</table>
Example

Input: \{ab, aabb, aabbbb\}

S \rightarrow A B
\rightarrow A A B B
\rightarrow A A A B B B
A \rightarrow a
B \rightarrow b

Chunk(AB)\rightarrow X
S \rightarrow X
\rightarrow A X B
\rightarrow A A X B B
X \rightarrow A B

Chunk(AXB)\rightarrow Y
S \rightarrow X
\rightarrow Y
\rightarrow A Y B
X \rightarrow A B
Y \rightarrow A X B

Merge S,Y
S \rightarrow X
\rightarrow A S B
X \rightarrow A B

Merge S,X
S \rightarrow A B
\rightarrow A S B
Issue with Chunk/Merge Systems

- Hard to recover from initial choices
- Hard to make local decision which will interact with each other (e.g., group verb preposition and preposition-determiner, both wrong and non consistent)
- Good local heuristics often don’t have well formed objectives that can be evaluated for the target grammar
Learn PCFGs with EM

- (Lari&Young 1990): Learning PCFGs with EM
  - Full binary grammar over $n$ symbols
  - Parse randomly at first
  - Re-estimate rule probabilities of parses
  - Repeat
Grammar Format

- Lari & Young, 1990: Satisfactory grammar learning requires more nonterminals than are theoretically needed to describe a language at hand.

- There is no guarantee that the nonterminals that the algorithm learns will have any resemblance to nonterminals motivated in linguistic analysis.

- Constraints on the grammar format may simplify the reestimation procedure.
  - Carroll & Charniak, 1992: Specify constraints on non-terminals that may appear together on the right-hand side of the rule.
Partially Unsupervised Learning

Pereira & Schabes 1992

- Idea: Encourage the probabilities into a good region of the parameter space
- Implementation: modify Inside-Outside algorithm to consider only parses that do not cross provided bracketing
- Experiments: 15 non terminals over 45 POS tags
  The algorithm uses Treebank bracketing, but ignores the labels
- Evaluation Measure: fraction of nodes in gold trees correctly posited in proposed trees (unlabeled recall)
• Results:
  – Constrained and unconstrained grammars have similar cross-entropy
  – But very different bracketing accuracy: 37% vs. 90%
Current Performance

• Constituency recall:

<table>
<thead>
<tr>
<th>Model</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Baseline</td>
<td>39.4</td>
</tr>
<tr>
<td>Klein’2005</td>
<td>88.0</td>
</tr>
<tr>
<td>Supervised PCFG</td>
<td>92.8</td>
</tr>
</tbody>
</table>

• Why it works?
  – Combination of simple models
  – Representations designed for unsupervised learning