Distributional Similarity

- A common method to assess the semantic similarity of words is to compare the contexts in which they occur.

"YOU SHALL KNOW A WORD BY THE COMPANY IT KEEPS!" (Firth 1957)

- **Distributional Similarity** methods compare the contexts that occur around words in a large text collection to determine how similar two words are.

**Distributional Hypothesis** (Harris, 1954)

words that occur in the same contexts tend to have similar meanings

Intuition

- Julie bought a *gok*.
- Mark ordered *gok* for lunch.
- The *gok* seeds fell all over the floor.
- Harry is allergic to *gok*.
- The *gok* wasn't quite ripe yet.
- Gloria ate a sandwich and a *gok*.
- They planted a rose bush and a *gok* tree in their yard.
- The recipe called for beef, *goks*, and curry paste.

Applications

- “entity set” expansion (essentially, semantic classes)
- word sense disambiguation
- paraphrasing
- spelling correction
- query expansion (IR)
- textual advertising
- and many others!

Distributional Similarity & Co-occurrence

- Two words will have a high distributional similarity if their surrounding contexts are similar.

- Two words that have a high distributional similarity:
  - may rarely co-occur with each other!
    Example: synonyms
  - but do co-occur with many of the same words in surrounding contexts.
Computing Distributional Similarity

1. Gather all of the contexts around each term.

2. Create a feature vector from the contextual evidence for the term.

3. Compute the similarity of pairs of terms by computing the similarity of their feature vectors.

4. Rank or cluster the vectors to identify the most similar terms.

Context

- Context is the neighborhood around an instance of \( w \).
- The neighborhood around \( w \) is typically defined as a context window of words, phrases, or structures on its left (-) and/or on its right (+).
  - Some tasks use “local” small context windows (e.g., +/- 2 words).
  - Some tasks use “global” large context windows (e.g., +/- 100 words).

Context Features

- The features in a context window can be simple or complex.
  - individual words
  - syntactic chunks (e.g., NPs)
  - syntactic roles
    e.g., SubjectOf(“barked”), PPObj(“near”)
  - syntactic modifiers and structures
    e.g., mod(“furry”), appositive(“vice president”)

Feature Vectors

- A term \( w \) is represented using a vector of features:
  \(<f_1 f_2 f_3 f_4 \ldots f_k>\)
  
  Most commonly, the features are surrounding words and the vector represents co-occurrence with instances of \( w \).

Example (with binary feature values):

<table>
<thead>
<tr>
<th></th>
<th>( f_1 )</th>
<th>( f_2 )</th>
<th>( f_3 )</th>
<th>( f_4 )</th>
<th>( f_5 )</th>
<th>( f_6 )</th>
<th>( f_7 )</th>
<th>( f_8 )</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>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>information</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
## Stopwords

- **Stopwords** are extremely high-frequency words, usually closed-class, that are often removed from consideration as features because they have relatively little semantic content.

- Commonly used stopwords are articles, determiners, prepositions, conjunctions, quantifiers, etc.

- Stopword lists are also used in most information retrieval systems. Removing stopwords can dramatically reduce the space needed to "index" the text collection.

## Simple Feature Weighting

- **binary**: 1/0 indicating the presence/absence of a feature in the context windows.
  \(<0 1 0 0 1 1 1 0 1 0 0 0 1 0 0 ...>\)

- **frequency**: the number of times that the feature occurs in the context windows.
  \(<0 34 0 0 5 22 7 0 98 0 0 0 0 2 0 0 ...>\)

---

### Example

**CORPUS**

- She ate chili for lunch.
- She went to the park.
- She had lunch at a *diner*.
- That *diner* serves chili for lunch.
- She went shopping at the store.
- She had chili at the *diner*.
- For lunch, she went to the *diner*.

**Features**

<table>
<thead>
<tr>
<th></th>
<th>she</th>
<th>ate</th>
<th>chili</th>
<th>lunch</th>
<th>went</th>
<th>park</th>
<th>had</th>
<th>diner</th>
<th>serves</th>
<th>shopping</th>
<th>store</th>
</tr>
</thead>
<tbody>
<tr>
<td>she</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>diner (bin)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>diner (freq)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

### Example

**CORPUS**

- She ate chili for lunch.
- She went to the park.
- She had lunch at a *diner*.
- That *diner* serves chili for lunch.
- She went shopping at the store.
- She had chili at the *diner*.
- For lunch, she went to the *diner*.

**Features**

<table>
<thead>
<tr>
<th></th>
<th>she</th>
<th>ate</th>
<th>chili</th>
<th>lunch</th>
<th>went</th>
<th>park</th>
<th>had</th>
<th>diner</th>
<th>serves</th>
<th>shopping</th>
<th>store</th>
</tr>
</thead>
<tbody>
<tr>
<td>she</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>diner (bin)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>diner (freq)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>diner (prob)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---
Better Feature Weighting

- **Conditional probabilities**: \( P(f \mid w) \)
  - \( \text{count}(w) = \text{# times } w \text{ occurs} \)
  - \( \text{count}(f,w) = \text{# times } f \text{ occurs "near" } w \)

- **Pointwise Mutual Information (PMI)**:
  \[
  \text{PMI}(f,w) = \log_2 \left( \frac{P(f,w)}{P(w) \cdot P(f)} \right)
  \]

Other options include tf-idf (an IR measure) and statistical significance tests, such as t-test or chi-squared.

Computing Semantic Similarity

- The **semantic similarity** of two words is the similarity of their context vectors.

- Many metrics can be used to assess vector similarity. The simplest is **Manhattan distance**:
  \[
  \text{ManhattanDistance}(\vec{X}, \vec{Y}) = \sum_{i=1}^{N} |x_i - y_i|
  \]
  But this metric doesn't work well for semantic similarity…

Jaccard Similarity

The Jaccard similarity metric assesses the amount of weighted overlap between features.

\[
\text{Jaccard}(\vec{X}, \vec{Y}) = \frac{\sum_{i=1}^{N} \min(x_i, y_i)}{\sum_{i=1}^{N} \max(x_i, y_i)}
\]

Cosine Similarity

A consistently good similarity metric is the cosine between the vectors. The numerator is the dot product (from linear algebra), and the denominator normalizes by the length of each vector.

\[
\text{Cosine}(\vec{X}, \vec{Y}) = \frac{\sum_{i=1}^{N} (x_i \cdot y_i)}{\sqrt{\sum_{i=1}^{N} x_i^2} \cdot \sqrt{\sum_{i=1}^{N} y_i^2}}
\]
Some final thoughts

• Distributional similarity measures have achieved good success for a wide variety of problems.

• But .. it typically provides only a coarse sense of semantic similarity. Often a wide variety of words will be judged to be similar!
  – Example: antonyms, strongly associated items (e.g., many different things associated with basketball will likely cluster).

• Computing pairwise distributional similarities across a large text collection is computationally expensive. Tricks are necessary to be practical in that setting.