Thursday’s key concepts

Mutual information and pointwise mutual information

Identifying “sticky pairs” of words

Brown clusters:
- class-based bigram model
- minimize MI of adjacent classes
- yields a hierarchical clustering

What we’re going to cover today

Distributional (Vector-space) semantics:
Measuring the semantic similarity of words
- The distributional hypothesis
- Representing word types as vectors

What is tezgüino?

A bottle of tezgüino is on the table.
Everybody likes tezgüino.
Tezgüino makes you drunk.
We make tezgüino out of corn.
(Lin, 1998; Nida, 1975)

Distributional hypothesis:
You shall know a word by the company it keeps.
(Firth 1957)

The contexts in which a word appears tells us a lot about what it means.
Exploiting context for semantics

Distributional similarities (vector-space semantics):
Use the contexts in which words appear to measure their similarity
   Assumption: Words that appear in similar contexts (tea, coffee) have similar meanings.

Word sense disambiguation (on Thursday)
Use the context of a particular occurrence of a word to identify which sense it has.
   Assumption: If a word has multiple distinct senses (e.g. plant: factory or green plant), each sense will appear in different contexts.

Distributional similarities

Distributional similarities use the set of contexts in which words appear to measure their similarity.

They represent each word \( w \) as a vector \( w = (w_1, \ldots, w_N) \in \mathbb{R}^N \) in an N-dimensional vector space.
   - Each dimension corresponds to a particular context \( c_n \)
   - Each element \( w_n \) of \( w \) captures the degree to which the word \( w \) is associated with the context \( c_n \).
   - \( w_n \) depends on the co-occurrence counts of \( w \) and \( c_n \)

The similarity of words \( w \) and \( u \) is given by the similarity of their vectors \( w \) and \( u \)

What is a ‘context'? 

There are many different definitions of context that yield different kinds of similarities:

Contexts defined by nearby words: 
   How often does \( w \) appear near the word \( drink \)?
   Near = “\( drink \) appears within a window of \( \pm k \) words of \( w \)” or “\( drink \) appears in the same sentence as \( w \)”
   This yields fairly broad thematic similarities.

Contexts defined by grammatical relations:
   How often is (the noun) \( w \) used as the subject (object) of the verb \( drink \)? (Requires a parser)
   This gives more fine-grained similarities.
Using nearby words as contexts

- Decide on a fixed vocabulary of N context words \( c_1 \ldots c_N \)
  Context words should occur frequently enough in your corpus that you get reliable co-occurrence counts, but you should ignore words that are too common (‘stop words’: a, the, on, in, and, or, is, have, etc.)

- Define what ‘nearby’ means
  For example: \( w \) appears near \( c \) if \( c \) appears within ±5 words of \( w \)

- Get co-occurrence counts of words \( w \) and contexts \( c \)

- Define how to transform co-occurrence counts of words \( w \) and contexts \( c \) into vector elements \( w_n \)
  For example: compute PMI of words and contexts

- Define how to compute the similarity of word vectors
  For example: use the cosine of their angles.

Using grammatical features

Observation: verbs have ‘selectional preferences’:
E.g. “eat” takes edible things as objects and animate entities as subjects.
Exceptions: metonymy (“The VW honked at me”) and metaphors: “Skype ate my credit”

This allows us to induce noun classes:
Edible things occur as objects of “eat”.
In general, nouns that occur as subjects/objects of specific verbs tend to be similar.

This also allows us to induce verb classes:
Verbs that take the same class of nouns as arguments tend to be similar/related.

Defining and counting co-occurrence

Defining co-occurrences:
  - Within a fixed window: \( v_i \) occurs within ±n words of \( w \)
  - Within the same sentence: requires sentence boundaries
  - By grammatical relations:
    \( v_i \) occurs as a subject/object/modifier/… of verb \( w \)
    (requires parsing - and separate features for each relation)

Counting co-occurrences:
  - \( f_i \) as binary features (1,0): \( w \) does/does not occur with \( v_i \)
  - \( f_i \) as frequencies: \( w \) occurs \( n \) times with \( v_i \)
  - \( f_i \) as probabilities:
    e.g. \( f_i \) is the probability that \( v_i \) is the subject of \( w \).

Examples: binary features

<table>
<thead>
<tr>
<th></th>
<th>arts</th>
<th>boil</th>
<th>data</th>
<th>function</th>
<th>large</th>
<th>sugar</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>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>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>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>0</td>
</tr>
</tbody>
</table>

Some examples from the Brown corpus.
Real co-occurrence vectors are very large, and very sparse:
Vocabulary size: 10K-100K
Example: frequencies of grammatical relations

64M word corpus, parsed with Minipar (Lin, 1998)

<table>
<thead>
<tr>
<th>cell</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>sbj of absorb</td>
<td>1</td>
</tr>
<tr>
<td>sbj of adapt</td>
<td>1</td>
</tr>
<tr>
<td>sbj of behave</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>mod of abnormality</td>
<td>3</td>
</tr>
<tr>
<td>mod of anemia</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>obj of attack</td>
<td>6</td>
</tr>
<tr>
<td>obj of call</td>
<td>11</td>
</tr>
</tbody>
</table>

Measuring association with context

- Every **element** \( f_i \) of the co-occurrence vector corresponds to some **word** \( w' \) (and possibly a relation \( r \)):
  
  e.g. \((r,w')=(\text{obj-of}, \text{attack})\)

- The value of \( f_i \) should indicate the **association strength** between \((r, w')\) and \( w \).

- What **value** should feature \( f_i \) for word \( w \) have?

  **Probability** \( P(f_i | w) \): \( f_i \) will be high for any frequent feature (regardless of \( w \))

Computing PMI of \( w \) and \( c \):

Using a fixed window of \( \pm k \) words

\[
PMI(w, c) = \log \frac{p(w, c)}{p(w)p(c)}
\]

\[N: \text{How many tokens does the corpus contain?}\]
\[f(w) \leq N: \text{How often does } w \text{ occur?}\]
\[f(w, c) \leq f(w): \text{How often does } w \text{ occur with } c \text{ in its window?}\]
\[f(c) = \Sigma_{w} f(w, c) \leq N: \text{How many tokens have } c \text{ in their window?}\]

\[p(w) = \frac{f(w)}{N}\]
\[p(c) = \frac{f(c)}{N}\]
\[p(w, c) = \frac{f(w, c)}{N}\]

Computing PMI of \( w \) and \( c \):

\( w \) and \( c \) in the same sentence

\[
PMI(w, c) = \log \frac{p(w, c)}{p(w)p(c)}
\]

\[N: \text{How many sentences does the corpus contain?}\]
\[f(w) \leq N: \text{How many sentences contain } w?\]
\[f(w, c) \leq f(w): \text{How many sentences contain } w \text{ and } c?\]
\[f(c) \leq N: \text{How many sentences contain } c?\]

\[p(w) = \frac{f(w)}{N}\]
\[p(c) = \frac{f(c)}{N}\]
\[p(w, c) = \frac{f(w, c)}{N}\]
Frequencies vs. PMI

Objects of ‘drink’ (Lin, 1998)

<table>
<thead>
<tr>
<th>Count</th>
<th>PMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>bunch beer</td>
<td>2</td>
</tr>
<tr>
<td>tea</td>
<td>2</td>
</tr>
<tr>
<td>liquid</td>
<td>2</td>
</tr>
<tr>
<td>champagne</td>
<td>4</td>
</tr>
<tr>
<td>anything</td>
<td>3</td>
</tr>
<tr>
<td>it</td>
<td>3</td>
</tr>
</tbody>
</table>

Positive Pointwise Mutual Information

PMI can be negative.

We often just use positive PMI values, and replace all PMI values < 0 with 0.

Positive Pointwise Mutual Information (PPMI):

\[
PPMI(w,c) = \begin{cases} 
PMI(w,c) & \text{if } PMI(w,c) > 0 \\
0 & \text{if } PMI(w,c) \leq 0 
\end{cases}
\]

PMI and smoothing

PMI is biased towards infrequent events:

If \( P(w, c) = P(w) = P(c) \), then \( PMI(w,c) = \log(1/P(w)) \)
So \( PMI(w, c) \) is larger for rare words \( w \) with low \( P(w) \).

Simple remedy: Add-k smoothing of \( P(w, c), P(w), P(c) \)
pushes all PMI values towards zero.
Add-k smoothing affects low-probability events more,
and will therefore reduce the bias of PMI towards
infrequent events.

(Pantel & Turney 2010)

Vector similarity

In distributional models, every word is a point in \( n \)-dimensional space.
How do we measure the similarity between two points/vectors?

In general:

- **Manhattan distance** (Levenshtein distance, L1 norm)

\[
dist_{L1}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} |x_i - y_i|
\]

- **Euclidian distance** (L2 norm)

\[
dist_{L2}(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}
\]
Dot product as similarity

If the vectors consist of simple binary features (0,1), we can use the dot product as similarity metric:

\[ \text{sim}_{\text{dot-prod}}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} x_i \times y_i \]

The dot product is a bad metric if the vector elements are arbitrary features: it prefers long vectors.
- If one \( x_i \) is very large (and \( y \) nonzero), \( \text{sim}(x,y) \) gets very large.
- If the number of nonzero \( x \) and \( y \)'s is very large, \( \text{sim}(x,y) \) gets very large.
- Both can happen with frequent words.

\[ \text{length of } \vec{x} : |\vec{x}| = \sqrt{\sum_{i=1}^{N} x_i^2} \]

Vector similarity: Cosine

One way to define the similarity of two vectors is to use the cosine of their angle.

The cosine of two vectors is their dot product, divided by the product of their lengths:

\[ \text{sim}_{\text{cos}}(\vec{x}, \vec{y}) = \frac{\sum_{i=1}^{N} x_i \times y_i}{\sqrt{\sum_{i=1}^{N} x_i^2} \sqrt{\sum_{i=1}^{N} y_i^2}} = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|} \]

\[ \text{sim}(\vec{w}, \vec{u}) = 1: \vec{w} \text{ and } \vec{u} \text{ point in the same direction} \]
\[ \text{sim}(\vec{w}, \vec{u}) = 0: \vec{w} \text{ and } \vec{u} \text{ are orthogonal} \]
\[ \text{sim}(\vec{w}, \vec{u}) = -1: \vec{w} \text{ and } \vec{u} \text{ point in the opposite direction} \]

Kullback-Leibler divergence

When the vectors \( \vec{x} \) are probabilities, i.e. \( x_i = P(f_i \mid w_x) \), we can measure the distance between the two distributions \( \vec{P} \) and \( \vec{Q} \)

The standard metric is Kullback-Leibler divergence \( D(\vec{P}||\vec{Q}) \)

\[ D(\vec{P}||\vec{Q}) = \sum x \ P(x) \log \frac{P(x)}{Q(x)} \]

But KL divergence is not very good because it is
- Undefined if \( P(x)=0 \) and \( Q(x) \neq 0 \).
- Asymmetric: \( D(\vec{P}||\vec{Q}) \neq D(\vec{Q}||\vec{P}) \)

Jensen/Shannon divergence

Instead, we use the Jensen/Shannon divergence: the distance of each distribution from their average.

- Average of \( \vec{P} \) and \( \vec{Q} \): \( \text{Avg}_{P,Q}(x) = \frac{P(x) + Q(x)}{2} \)
- Jensen/Shannon divergence of \( \vec{P} \) and \( \vec{Q} \):

\[ JS(\vec{P}||\vec{Q}) = D(\vec{P}||\text{Avg}_{P,Q}) + D(\vec{Q}||\text{Avg}_{P,Q}) \]

- As a distance measure between \( \vec{x},\vec{y} \) (with \( x_i = P(f_i \mid w_x) \))

\[ \text{dist}_{JS}(\vec{x},\vec{y}) = \sum x_i \log_2 \left( \frac{x_i}{(x_i + y_i)/2} \right) + y_i \log_2 \left( \frac{y_i}{(x_i + y_i)/2} \right) \]
More recent developments

Neural embeddings

There is a lot of recent work on neural-net based word embeddings:
- word2vec, https://code.google.com/p/word2vec/
- Glove http://nlp.stanford.edu/projects/glove/
eetc.

Using the vectors produced by these word embeddings instead of the raw words themselves can be very beneficial for many tasks.

This is currently a very active area of research.

Analogies

It can be shown that for some of these embeddings, the learned word vectors can capture analogies:

Queen::King = Woman::Man
In the vector representation: queen ≈ king − man + woman

Similar results for e.g. countries and capitals:
Germany::Berlin = France::Paris

“Semantic spaces”? 

Does this mean that these vector spaces represent semantics?

Yes, but only to some extent.
- Different context definitions (or embeddings) give different vector spaces with different similarities
- Often, antonyms (hot/cold, etc.) have very similar vectors.
- Vector spaces are not well-suited to capturing hypernym relations (every dog is an animal)
We will get back to that when we talk more about lexical semantics.

Another open problem: how to get from words to the semantics of sentences
Today’s key concepts

Distributional hypothesis

Distributional similarities:
- word-context matrix
- representing words as vectors
- positive PMI
- computing the similarity of word vectors