Question Answering

• Question
  What is a good way to remove wine stains?

• Text available to the machine
  Salt is a great way to eliminate wine stains

• What is hard?
  – words may be related in other ways, including similarity and gradation
  – how to know if words have similar meanings?

Can we just use a thesaurus?

Problems:

• May not have a thesaurus in every language

• Even if we do, many words and phrases will be missing

So, let’s try to compute similarity automatically.

Meaning from context(s)

• Consider the example from J&M (quoted from earlier sources):
  a bottle of tezgüino is on the table
  everybody likes tezgüino
  tezgüino makes you drunk
  we make tezgüino out of corn
Distributional hypothesis

• perhaps we can infer meaning just by looking at the contexts a word occurs in
• perhaps meaning IS the contexts a word occurs in (Wittgenstein!)
• either way, similar contexts imply similar meanings:
  – this idea is known as the distributional hypothesis

“Distribution”: a polysemous word

• Probability distribution: a function from outcomes to real numbers
• Linguistic distribution: the set of contexts that a particular item (here, word) occurs in

Distributional semantics: basic idea

• Represent each word $w_i$ as a vector of its contexts
  – distributional semantic models also called vector-space models.
• Ex: each dimension is a context word; = 1 if it co-occurs with $w_i$, otherwise 0.

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>bone</th>
<th>fur</th>
<th>run</th>
<th>brown</th>
<th>screen</th>
<th>mouse</th>
<th>fetch</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$w_2$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$w_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
• Note: real vectors would be far more sparse

Questions to consider

• What defines “context”? (What are the dimensions, what counts as co-occurrence?)
• How to weight the context words (Boolean? counts? other?)
• How to measure similarity between vectors?

Two kinds of co-occurrence between two words:

First-order co-occurrence: (syntagmatic association)
  • Typically nearby each other
    wrote is a first-order associate of book

Second-order co-occurrence: (paradigmatic association)
  • Have similar neighbours
    wrote is a second-order associate of said and remarked
Defining the context

• Usually ignore stopwords (function words and other very frequent/uninformative words)

• Usually use a large window around the target word (e.g., 100 words, maybe even whole document)

• Can use just cooccurrence within window, or may require more (e.g., dependency relation from parser)

• Note: all of these for semantic similarity; for syntactic similarity, use a small window (1-3 words) and track only frequent words.

How to weight the context words

• binary indicators not very informative

• presumably more frequent co-occurrences matter more

• but, is frequency good enough?
  – frequent words are expected to have high counts in the context vector
  – regardless of whether they occur more often with this word than with others

Collocations

• We want to know which words occur unusually often in the context of \( w \): more than we’d expect by chance?

• Put another way, what collocations include \( w \)?

Mutual information

• One way: use pointwise mutual information:

\[
PMI(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)} \leftarrow \text{Actual prob of seeing words } x \text{ and } y \text{ together}
\]

\[
\leftarrow \text{Predicted prob of same, if } x \text{ and } y \text{ are indep.}
\]

• PMI tells us how much more/less likely the cooccurrence is than if the words were independent
A problem with PMI

- In practice, PMI is computed with counts (using MLE).
- Result: it is over-sensitive to the chance co-occurrence of infrequent words
- See next slide: ex. PMIs from bigrams with 1 count in 1st 1000 documents of NY Times corpus

Example PMIs (Manning & Schütze, 1999, p181)

<table>
<thead>
<tr>
<th>$I_{1000}$</th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_1w_2$</th>
<th>Bigram</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.95</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>Schwartz eschews</td>
</tr>
<tr>
<td>15.02</td>
<td>1</td>
<td>19</td>
<td>1</td>
<td>fewest visits</td>
</tr>
<tr>
<td>13.78</td>
<td>5</td>
<td>9</td>
<td>1</td>
<td>FIND GARDEN</td>
</tr>
<tr>
<td>12.00</td>
<td>5</td>
<td>31</td>
<td>1</td>
<td>Indonesian pieces</td>
</tr>
<tr>
<td>9.82</td>
<td>26</td>
<td>27</td>
<td>1</td>
<td>Reds survived</td>
</tr>
<tr>
<td>9.21</td>
<td>13</td>
<td>82</td>
<td>1</td>
<td>marijuana growing</td>
</tr>
<tr>
<td>7.37</td>
<td>24</td>
<td>159</td>
<td>1</td>
<td>doubt whether</td>
</tr>
<tr>
<td>6.68</td>
<td>687</td>
<td>9</td>
<td>1</td>
<td>new converts</td>
</tr>
<tr>
<td>6.00</td>
<td>661</td>
<td>15</td>
<td>1</td>
<td>like offensive</td>
</tr>
<tr>
<td>3.81</td>
<td>159</td>
<td>283</td>
<td>1</td>
<td>must think</td>
</tr>
</tbody>
</table>

Alternatives to PMI for finding collocations

- There are a lot, all ways of measuring statistical (in)dependence.
  - Student $t$-test
  - Pearson’s $\chi^2$ statistic
  - Dice coefficient
  - likelihood ratio test (Dunning, 1993)
  - Lin association measure (Lin, 1998)
  - and many more...
- Of those listed here, Dunning LR test probably most reliable for low counts.
- However, which works best may depend on particular application/evaluation.

Improving PMI

Rather than using a different method, can modify PMI itself to better handle low frequencies.

- Use positive PMI (PPMI): change all negative PMI values to 0.
  - Because for infrequent words, not enough data to accurately determine negative PMI values.
- Introduce smoothing in PMI computation.
  - See J&M or Levy et al. (2015) for a particularly effective method.
How to measure similarity

• So, let’s assume we have context vectors for two words $\vec{v}$ and $\vec{w}$
• Each contains PMI (or PPMI) values for all context words
• One way to think of these vectors: as points in high-dimensional space

Vector space representation

• Ex. in 2-dim space: $\text{cat} = (v_1, v_2)$, $\text{computer} = (w_1, w_2)$

Euclidean distance

• We could measure (dis)similarity using Euclidean distance: $(\sum_i (v_i - w_i)^2)^{1/2}$

Dot product

• Another possibility: take the dot product of $\vec{v}$ and $\vec{w}$:

$$\text{sim}_{DP}(\vec{v}, \vec{w}) = \vec{v} \cdot \vec{w} = \sum_i v_i w_i$$

– Vectors are longer if they have higher values in each dimension.
– So more frequent words have higher dot products.
– But we don’t want a similarity metric that’s sensitive to word frequency.
Normalized dot product

- Some vectors are longer than others (have higher values):

  \[ [5, 2.3, 0, 0.2, 2.1] \] vs. \[ [0.1, 0.3, 1, 0.4, 0.1] \]

  - If vector is context word counts, these will be frequent words
  - If vector is PMI values, these are likely to be infrequent words

- Dot product is generally larger for longer vectors, regardless of similarity

- To correct for this, we normalize: divide by the length of each vector:

  \[
  \text{sim}_{\text{NDP}}(\vec{v}, \vec{w}) = (\vec{v} \cdot \vec{w}) / (|\vec{v}| |\vec{w}|)
  \]

\[
\text{Normalized dot product} = \text{cosine}
\]

- The normalized dot product is just the cosine of the angle between vectors.

\[ \frac{\text{Cosine}}{\text{dog}} \]

\[ \frac{\text{computer}}{\text{cat}} \]

Other similarity measures

- Again, many alternatives
  - Jaccard measure
  - Dice measure
  - Jenson-Shannon divergence
  - etc.

- Again, may depend on particular application/evaluation

Evaluation

- Extrinsic may involve IR, QA, automatic essay marking, ...

- Intrinsic is often a comparison to psycholinguistic data
  - Relatedness judgments
  - Word association
Relatedness judgments

- Participants are asked, e.g.: on a scale of 1-10, how related are the following concepts?
  
  LEMON   FLOWER

- Usually given some examples initially to set the scale, e.g.
  - LEMON-TRUTH = 1
  - LEMON-ORANGE = 10

- But still a funny task, and answers depend a lot on how the question is asked ('related' vs. 'similar' vs. other terms)

Word association

- Participants see/hear a word, say the first word that comes to mind

- Data collected from lots of people provides probabilities of each answer:
  
<table>
<thead>
<tr>
<th>Word</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORANGE</td>
<td>0.16</td>
</tr>
<tr>
<td>SOUR</td>
<td>0.11</td>
</tr>
<tr>
<td>TREE</td>
<td>0.09</td>
</tr>
<tr>
<td>YELLOW</td>
<td>0.08</td>
</tr>
<tr>
<td>TEA</td>
<td>0.07</td>
</tr>
<tr>
<td>JUICE</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Example data from the Edinburgh Associative Thesaurus: http://www.eat.rl.ac.uk/

Comparing to human data

- Human judgments provide a ranked list of related words/associations for each word $w$

- Computer system provides a ranked list of most similar words to $w$

- Compute the Spearman rank correlation between the lists (how well do the rankings match?)

- Often report on several data sets, as their details differ

Learning a more compact space

- So far, our vectors have length $V$, the size of the vocabulary

- Do we really need this many dimensions?

- Can we represent words in a smaller dimensional space that preserves the similarity relationships of the larger space?
Latent Semantic Analysis (LSA)

- One of the earliest methods for reducing dimensions while preserving similarity.
- Uses Singular Value Decomposition, a linear-algebra-based method.
- Converts from sparse vectors with 1000s of dimensions to dense vectors with 10s-100s of dimensions.
- LSA representations actually work better than originals for many tasks.
- More details in optional reading: J&M (3rd ed.) Ch 19.5

Neural network methods

- Recent (and very hyped) new methods for learning reduced-dimensional representations (now often called embeddings).
- Ex: train a NN to predict context words based on input word. Use hidden layer(s) as the input word’s vector representation.
- Deep mathematical similarities to LSA (Levy and Goldberg, 2014), but can be faster to train.
- Appeared to work better than LSA, but likely due to unfair comparisons (Levy et al., 2015).
- More details in optional reading: J&M (3rd ed.) Ch 19.6-19.7

Vector representations in practice

- Very hot topic in NLP
- Embeddings seem to capture both syntactic and semantic information.
- So, used for language modelling and to replace words as ‘observations’ in parsing and other models.
- As noted in Smoothing lecture: this can provide a kind of similarity-based smoothing (models learn to make similar predictions for similar words).

Current work: compositionality

- One definition of collocations: non-compositional phrases
  - White House: not just a house that is white
  - barn raising: involves more than the parts imply
- But a lot of language is compositional
  - red barn: just a barn that is red
  - wooden plank: nothing special here
- Can we capture compositionality in a vector space model?
Compositionality in a vector space

- More formally, compositionality implies some operator $\oplus$ such that
  \[ \text{meaning}(w_1 w_2) = \text{meaning}(w_1) \oplus \text{meaning}(w_2) \]

- Current work investigates possible operators
  - vector addition (doesn’t work very well)
  - tensor product
  - nonlinear operations learned by neural networks

- One problem: words like not—themselves more like operators than points in space.

Summary

- Distributional semantics: represents word meanings as vectors computed from their contexts.
  - Long sparse vectors of counts, PMI values, or others
  - Short dense vectors using LSA, NNets, or others

- Similarity typically measured using cosine distance

- Can work well as input to other systems, but harder to evaluate intrinsically