Vector Space Models of Meaning
Word Embeddings
Lexical Semantics
- The meanings of words
- The relationships between words
- WordNet and other semantic lexicons

Cons of large semantic lexicons (also referred to as thesauruses)
- Need to be constructed
- Require humans to make decision
- Only represent a small set of possible relations between words
  - This is fixable, but required more humans
- Aren’t usually kept up to date
  - Again because of humans
What we Would Like in an Alternative

- Something that doesn’t involve humans
  - Data-driven from a corpus
- Is fast to build
- Can represent different types of relationships easily
  - Semantic and syntactic
- Is easily updatable
The solution?

- Use vectors to represent words
  
  \[
  \text{dog} = [\ 0.9 \ 0.1 \ 0.2 \ 0.03 \ 0.23 \ 0.54 \ 0.2 \ 0.1]\n  \]

- Use operations between vectors to represent relations
  
  \[
  \text{human} - \text{child} + \text{dog} \approx \text{puppy}
  \]
“You shall know a word by the company it keeps”

What does Firth mean?

- Mastiff mix and a dog actor best known for his
- And Georgette Magritte With Their Dog After The War.
- such as hormonal imbalances.
- Dog aggression is a common dog
- no one had heard of dog agility in the United States,
- or the speed required in dog agility. Prey drive can
- death of David's previous dog Alfred (originally owned by
- but if someone finds the dog and delivers it, the
- Marissa Nadler, Dr. Dog and The Dead Milkmen)
- *' ' Time Between Dog and Wolf ' ' (restricts every family to one dog as a maximum. Dogs
- , you could think of dog as a type and your
The simplest model

- Just record a count of every word in every length $w$ context
  - In this example, our $w$ is 5, on each side
  - Sort of like a 5-gram language model
  - But we have to keep the entire matrix now!

<table>
<thead>
<tr>
<th></th>
<th>...</th>
<th>Mastiff</th>
<th>mix</th>
<th>)</th>
<th>and</th>
<th>a</th>
<th>actor</th>
<th>best</th>
<th>known</th>
<th>for</th>
<th>his</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</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>
<td></td>
</tr>
</tbody>
</table>
The simplest model

- Repeat for every word in the corpus

, you could think of **dog** as a type and your

<table>
<thead>
<tr>
<th></th>
<th>Mastiff</th>
<th>mix</th>
<th>)</th>
<th>and</th>
<th>a</th>
<th>actor</th>
<th>best</th>
<th>known</th>
<th>for</th>
<th>his</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
We have meaning representations which are
- Learned directly from data
- Can be updated
- Are relatively fast to computer
  - Hours/Days vs. Years/Decades
- We can calculate relationships between vectors using arithmetic

As a bonus
- Have a representation that can be fed to any machine learning algorithm for downstream tasks such as text classification, part of speech tagging, sentiment analysis, etc.
We have a matrix where each row represents a word (co-occurrence matrix)

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>love</th>
<th>feed</th>
<th>may</th>
<th>hair</th>
<th>vet</th>
<th>hug</th>
<th>play</th>
<th>sleep</th>
<th>bark</th>
<th>meow</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>20</td>
<td>23</td>
<td>5</td>
<td>30</td>
<td>29</td>
<td>8</td>
<td>4</td>
<td>15</td>
<td>4</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>cat</td>
<td>14</td>
<td>25</td>
<td>5</td>
<td>32</td>
<td>40</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>10</td>
<td>3</td>
<td>53</td>
</tr>
<tr>
<td>broom</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>29</td>
<td>75</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

To calculate the similarity between two words, we calculate their distance in vector space
The Euclidean distance is the length of a line between any two points in vector space. Intuitively, two words are more similar if the distance between them is smaller. To formalize this and make Euclidean distance into a similarity metric you can just negate the distance:

\[ \text{sim}(x,y) = -\text{distance}(x,y) \]

\[ \text{distance}(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} \]
Euclidean Distance Example

\[ \text{distance}(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} \]

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>love</th>
<th>feed</th>
<th>may</th>
<th>hair</th>
<th>vet</th>
<th>hug</th>
<th>play</th>
<th>sleep</th>
<th>bark</th>
<th>meow</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>20</td>
<td>23</td>
<td>5</td>
<td>30</td>
<td>29</td>
<td>8</td>
<td>4</td>
<td>15</td>
<td>4</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>cat</td>
<td>14</td>
<td>25</td>
<td>5</td>
<td>32</td>
<td>40</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>10</td>
<td>3</td>
<td>53</td>
</tr>
<tr>
<td>broom</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>29</td>
<td>75</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>d-c</td>
<td>6</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>-11</td>
<td>4</td>
<td>-1</td>
<td>9</td>
<td>-6</td>
<td>47</td>
<td>-51</td>
</tr>
<tr>
<td>d-b</td>
<td>18</td>
<td>19</td>
<td>3</td>
<td>1</td>
<td>-46</td>
<td>7</td>
<td>4</td>
<td>14</td>
<td>1</td>
<td>49</td>
<td>-2</td>
</tr>
<tr>
<td>(d-c)^2</td>
<td>36</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>121</td>
<td>16</td>
<td>1</td>
<td>81</td>
<td>36</td>
<td>2209</td>
<td>2601</td>
</tr>
<tr>
<td>SUM</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>(d-b)^2</td>
<td>324</td>
<td>361</td>
<td>9</td>
<td>1</td>
<td>2116</td>
<td>49</td>
<td>16</td>
<td>196</td>
<td>1</td>
<td>2401</td>
<td>4</td>
</tr>
</tbody>
</table>

\[ \sum 5109 \quad \text{SQRT} 71.47 \]

\[ \sum 5478 \quad 74.01 \]
The cosine of the angle between two vectors is
\[ \cos(x, y) = \frac{x \cdot y}{|x||y|} = \frac{\sum_{i=1}^{d} x_i \times y_i}{\sqrt{\sum_{i=1}^{d} x_i^2} \sqrt{\sum_{i=1}^{d} y_i^2}} \]

Because the values in our vectors are always positive, this value will be between 0 and 1.

Additionally
- Cosine similarity normalizes by the vector length, removing artifacts from frequent words.
- Can be calculated very quickly when done correctly!
Cosine Distance Example

- Trick #1: Normalize the vectors once

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

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>love</th>
<th>feed</th>
<th>may</th>
<th>hair</th>
<th>vet</th>
<th>hug</th>
<th>play</th>
<th>sleep</th>
<th>bark</th>
<th>meow</th>
<th>²SUM</th>
<th>SQRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>20</td>
<td>23</td>
<td>5</td>
<td>30</td>
<td>29</td>
<td>8</td>
<td>4</td>
<td>15</td>
<td>4</td>
<td>50</td>
<td>2</td>
<td>5520</td>
<td>74.29</td>
</tr>
<tr>
<td>cat</td>
<td>14</td>
<td>25</td>
<td>5</td>
<td>32</td>
<td>40</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>10</td>
<td>3</td>
<td>53</td>
<td>6465</td>
<td>80.40</td>
</tr>
<tr>
<td>broom</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>29</td>
<td>75</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>6518</td>
<td>80.73</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>love</th>
<th>feed</th>
<th>may</th>
<th>hair</th>
<th>vet</th>
<th>hug</th>
<th>play</th>
<th>sleep</th>
<th>bark</th>
<th>meow</th>
<th>²SUM</th>
<th>SQRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>.27</td>
<td>.31</td>
<td>.07</td>
<td>.40</td>
<td>.39</td>
<td>.10</td>
<td>.05</td>
<td>.20</td>
<td>.05</td>
<td>.67</td>
<td>.027</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>cat</td>
<td>.17</td>
<td>.31</td>
<td>.06</td>
<td>.40</td>
<td>.50</td>
<td>.05</td>
<td>.06</td>
<td>.07</td>
<td>.12</td>
<td>.04</td>
<td>.66</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>broom</td>
<td>.02</td>
<td>.05</td>
<td>.02</td>
<td>.35</td>
<td>.93</td>
<td>.01</td>
<td>.00</td>
<td>.01</td>
<td>.04</td>
<td>.01</td>
<td>.05</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Trick #1: Normalize the vectors once

Cosine Distance Example

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

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>love</th>
<th>feed</th>
<th>may</th>
<th>hair</th>
<th>vet</th>
<th>hug</th>
<th>play</th>
<th>sleep</th>
<th>bark</th>
<th>meow</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>.27</td>
<td>.31</td>
<td>.07</td>
<td>.40</td>
<td>.39</td>
<td>.10</td>
<td>.05</td>
<td>.20</td>
<td>.05</td>
<td>.67</td>
<td>.027</td>
</tr>
<tr>
<td>cat</td>
<td>.17</td>
<td>.31</td>
<td>.06</td>
<td>.40</td>
<td>.50</td>
<td>.05</td>
<td>.06</td>
<td>.07</td>
<td>.12</td>
<td>.04</td>
<td>.66</td>
</tr>
<tr>
<td>broom</td>
<td>.02</td>
<td>.05</td>
<td>.02</td>
<td>.35</td>
<td>.93</td>
<td>.01</td>
<td>0.0</td>
<td>.01</td>
<td>.04</td>
<td>.01</td>
<td>.05</td>
</tr>
<tr>
<td>d \times c</td>
<td>.05</td>
<td>.10</td>
<td>.004</td>
<td>.16</td>
<td>.195</td>
<td>.005</td>
<td>.003</td>
<td>.01</td>
<td>.006</td>
<td>.026</td>
<td>.017</td>
</tr>
<tr>
<td>d \times b</td>
<td>.005</td>
<td>.02</td>
<td>.001</td>
<td>.14</td>
<td>.363</td>
<td>.001</td>
<td>0.0</td>
<td>.002</td>
<td>.002</td>
<td>.006</td>
<td>.001</td>
</tr>
</tbody>
</table>

Note: It is **NOT** always the case that the cosine similarity will produce the same most similar word as when using Euclidean.
Things to Improve Upon

- These matrices will be $N \times N$, so will become extremely large and slow to do calculations over
  - Solution: Perform dimensionality reduction
    - Con: Makes it harder to update

- Currently all word-context pairs are weighted equally
  - Solution: Use PPMI (Next slides) to determine which word-context pairs occur together more than probability suggests they should
    - Con: Makes it harder to update

- All words in a window are currently counted the same
  - Solution: Use a weighting function dependent on the distance from the word
A matrix of $w \times w$ is too large to do computations over

Use a standard dimensionality reduction method to reduce the size of the matrix to $w \times d$, where $d$ is some number $< 1000$

A popular method is SVD

- Start with our co-occurrence matrix $X$
- SVD is a matrix factorization method

$$X = U\Sigma V^T$$

- $X$ is a $w \times c$ matrix, $U$ is an $w \times m$ matrix, $\Sigma$ is an $m \times m$ matrix, and $V$ is an $c \times m$ matrix
- Take the first $d$ dimensions of the $U$ matrix to arrive at a $w \times d$ vector space
$X = U \Sigma V^T$

$X' = w \times d$
Pointwise Mutual Information

- Mutual Information is a measure from information theory that quantifies the dependency between two distributions.
- Pointwise Mutual Information measures a similar idea between two events:
  - How often two events co-occur vs how much they should co-occur if they were independent.
  - Events for us are words. How often does San and Fransico occur together compared to how much we would expect if they were independent? A lot.

\[
\text{PMI}(w, c) = \log \frac{P(w,c)}{P(w)P(c)}
\]
Pointwise Mutual Information

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

\[ = \log \frac{\frac{C(w, c)}{N}}{\frac{C(w)}{N} \frac{C(c)}{N}} \]

\[ = \log \frac{\frac{C(w, c)}{N}}{\frac{C(w)C(c)}{N^2}} \]

\[ = \log \frac{\frac{C(w, c)}{N}}{\frac{C(w)C(c)}{N^2}} \times \frac{N^2}{C(w)C(c)} \]

\[ = \log \frac{C(w, c) \times N}{C(w)C(c)} \]
### Pointwise Mutual Information Example

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>love</th>
<th>feed</th>
<th>may</th>
<th>hair</th>
<th>vet</th>
<th>hug</th>
<th>play</th>
<th>sleep</th>
<th>bark</th>
<th>meow</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>20</td>
<td>23</td>
<td>5</td>
<td>30</td>
<td>29</td>
<td>8</td>
<td>4</td>
<td>15</td>
<td>4</td>
<td>50</td>
<td>2</td>
<td>190</td>
</tr>
<tr>
<td>cat</td>
<td>14</td>
<td>25</td>
<td>5</td>
<td>32</td>
<td>40</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>10</td>
<td>3</td>
<td>53</td>
<td>197</td>
</tr>
<tr>
<td>broom</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>29</td>
<td>75</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>122</td>
</tr>
<tr>
<td>SUM</td>
<td>36</td>
<td>52</td>
<td>12</td>
<td>91</td>
<td>144</td>
<td>13</td>
<td>9</td>
<td>22</td>
<td>17</td>
<td>54</td>
<td>59</td>
<td>-----</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>love</th>
<th>feed</th>
<th>may</th>
<th>hair</th>
<th>vet</th>
<th>hug</th>
<th>play</th>
<th>sleep</th>
<th>bark</th>
<th>meow</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>6840</td>
<td>9880</td>
<td>2280</td>
<td>17290</td>
<td>27360</td>
<td>2470</td>
<td>1710</td>
<td>4180</td>
<td>3230</td>
<td>10260</td>
<td>11210</td>
<td>-----</td>
</tr>
<tr>
<td>cat</td>
<td>7092</td>
<td>10244</td>
<td>2364</td>
<td>17927</td>
<td>28368</td>
<td>2561</td>
<td>1773</td>
<td>4334</td>
<td>3349</td>
<td>10638</td>
<td>11623</td>
<td>-----</td>
</tr>
<tr>
<td>broom</td>
<td>4392</td>
<td>6344</td>
<td>1464</td>
<td>11102</td>
<td>17568</td>
<td>1586</td>
<td>1098</td>
<td>2684</td>
<td>2074</td>
<td>6588</td>
<td>7198</td>
<td>-----</td>
</tr>
</tbody>
</table>
## Pointwise Mutual Information Example

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>love</th>
<th>feed</th>
<th>may</th>
<th>hair</th>
<th>vet</th>
<th>hug</th>
<th>play</th>
<th>sleep</th>
<th>bark</th>
<th>meow</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>10180</td>
<td>11707</td>
<td>2545</td>
<td>15270</td>
<td>14761</td>
<td>4072</td>
<td>2036</td>
<td>7635</td>
<td>2036</td>
<td>25450</td>
<td>1018</td>
</tr>
<tr>
<td>cat</td>
<td>7126</td>
<td>12725</td>
<td>2545</td>
<td>16288</td>
<td>20360</td>
<td>2036</td>
<td>2545</td>
<td>3054</td>
<td>5090</td>
<td>1527</td>
<td>26977</td>
</tr>
<tr>
<td>broom</td>
<td>1018</td>
<td>2036</td>
<td>1018</td>
<td>14761</td>
<td>38175</td>
<td>509</td>
<td>0</td>
<td>509</td>
<td>1527</td>
<td>509</td>
<td>2036</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>love</th>
<th>feed</th>
<th>may</th>
<th>hair</th>
<th>vet</th>
<th>hug</th>
<th>play</th>
<th>sleep</th>
<th>bark</th>
<th>meow</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>1.49</td>
<td>1.18</td>
<td>1.12</td>
<td>0.88</td>
<td>0.53</td>
<td>1.64</td>
<td>1.19</td>
<td>1.82</td>
<td>0.63</td>
<td>2.48</td>
<td>0.09</td>
</tr>
<tr>
<td>cat</td>
<td>1.00</td>
<td>1.24</td>
<td>1.07</td>
<td>0.90</td>
<td>0.72</td>
<td>0.79</td>
<td>1.43</td>
<td>0.70</td>
<td>1.52</td>
<td>0.14</td>
<td>2.32</td>
</tr>
<tr>
<td>broom</td>
<td>0.23</td>
<td>0.32</td>
<td>0.70</td>
<td>1.32</td>
<td>2.17</td>
<td>0.32</td>
<td>0.0</td>
<td>0.18</td>
<td>0.74</td>
<td>0.77</td>
<td>0.28</td>
</tr>
</tbody>
</table>
### Pointwise Mutual Information Example

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>love</th>
<th>feed</th>
<th>may</th>
<th>hair</th>
<th>vet</th>
<th>hug</th>
<th>play</th>
<th>sleep</th>
<th>bark</th>
<th>meow</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dog</strong></td>
<td>0.397</td>
<td>0.167</td>
<td>0.11</td>
<td>-0.124</td>
<td>-0.617</td>
<td>0.49</td>
<td>0.17</td>
<td>0.60</td>
<td>-0.46</td>
<td>0.908</td>
<td>-2.398</td>
</tr>
<tr>
<td><strong>cat</strong></td>
<td>0.004</td>
<td>0.216</td>
<td>0.07</td>
<td>-0.09</td>
<td>-0.33</td>
<td>-0.22</td>
<td>0.36</td>
<td>-0.35</td>
<td>0.41</td>
<td>-1.94</td>
<td>0.841</td>
</tr>
<tr>
<td><strong>broom</strong></td>
<td>-1.46</td>
<td>-1.13</td>
<td>-0.36</td>
<td>0.284</td>
<td>0.77</td>
<td>-1.13</td>
<td>-inf</td>
<td>-1.66</td>
<td>-0.30</td>
<td>-2.56</td>
<td>-1.26</td>
</tr>
</tbody>
</table>

A negative PMI indicates low support. It is common to just set them to 0. This is known as a Positive PMI Matrix (PPM)

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>love</th>
<th>feed</th>
<th>may</th>
<th>hair</th>
<th>vet</th>
<th>hug</th>
<th>play</th>
<th>sleep</th>
<th>bark</th>
<th>meow</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dog</strong></td>
<td>0.397</td>
<td>0.167</td>
<td>0.11</td>
<td>0</td>
<td>0</td>
<td>0.49</td>
<td>0.17</td>
<td>0.60</td>
<td>0</td>
<td>0.908</td>
<td>0</td>
</tr>
<tr>
<td><strong>cat</strong></td>
<td>0.004</td>
<td>0.216</td>
<td>0.07</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.36</td>
<td>0</td>
<td>0.41</td>
<td>0</td>
<td>0.841</td>
</tr>
<tr>
<td><strong>broom</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.284</td>
<td>0.77</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Pointwise Mutual Information Example

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>love</th>
<th>feed</th>
<th>may</th>
<th>hair</th>
<th>vet</th>
<th>hug</th>
<th>play</th>
<th>sleep</th>
<th>bark</th>
<th>meow</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>20</td>
<td>23</td>
<td>5</td>
<td>30</td>
<td>29</td>
<td>8</td>
<td>4</td>
<td>15</td>
<td>4</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>cat</td>
<td>14</td>
<td>25</td>
<td>5</td>
<td>32</td>
<td>40</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>10</td>
<td>3</td>
<td>53</td>
</tr>
<tr>
<td>broom</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>29</td>
<td>75</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>pet</th>
<th>love</th>
<th>feed</th>
<th>may</th>
<th>hair</th>
<th>vet</th>
<th>hug</th>
<th>play</th>
<th>sleep</th>
<th>bark</th>
<th>meow</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>0.397</td>
<td>0.167</td>
<td>0.11</td>
<td>0</td>
<td>0</td>
<td>0.49</td>
<td>0.17</td>
<td>0.60</td>
<td>0</td>
<td>0.908</td>
<td>0</td>
</tr>
<tr>
<td>cat</td>
<td>0.004</td>
<td>0.216</td>
<td>0.07</td>
<td>0</td>
<td>0</td>
<td>0.36</td>
<td>0</td>
<td>0.41</td>
<td>0</td>
<td>0</td>
<td>0.841</td>
</tr>
<tr>
<td>broom</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.284</td>
<td>0.77</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
What happens when we compute cosine similarity on a PPMI matrix?

Original Cosine Similarity
- Dog - Cat .576
- Dog - Broom .541

With PPMI
- Dog - Cat 0.082
- Dog - Broom 0.00
Evaluation on Vector Spaces is usually done either extrinsically
  ○ Use our vectors to perform a task like part of speech tagging or take a test like TOEFL
Or Intrinsically
  ○ Compare to how humans judge the similarity of two words
  ○ Compared using a correlation metric like Pearson’s r or Spearman’s ρ

<table>
<thead>
<tr>
<th>Gold Standard</th>
<th>Vector Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>large</td>
<td>cow</td>
</tr>
<tr>
<td>big</td>
<td>cattle</td>
</tr>
<tr>
<td>cow</td>
<td>machine</td>
</tr>
<tr>
<td>cattle</td>
<td>engine</td>
</tr>
<tr>
<td>machine</td>
<td>large</td>
</tr>
<tr>
<td>engine</td>
<td>big</td>
</tr>
<tr>
<td>bold</td>
<td>bold</td>
</tr>
<tr>
<td>proud</td>
<td>proud</td>
</tr>
<tr>
<td>new</td>
<td>new</td>
</tr>
<tr>
<td>ancient</td>
<td>ancient</td>
</tr>
</tbody>
</table>
### Evaluation

<table>
<thead>
<tr>
<th>Gold Standard</th>
<th>Vector Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>large</td>
<td>cow</td>
</tr>
<tr>
<td>big</td>
<td></td>
</tr>
<tr>
<td>cow</td>
<td>machine</td>
</tr>
<tr>
<td>cattle</td>
<td></td>
</tr>
<tr>
<td>machine</td>
<td>large</td>
</tr>
<tr>
<td>engine</td>
<td></td>
</tr>
<tr>
<td>bold</td>
<td>bold</td>
</tr>
<tr>
<td>proud</td>
<td></td>
</tr>
<tr>
<td>new</td>
<td>new</td>
</tr>
<tr>
<td>ancient</td>
<td></td>
</tr>
</tbody>
</table>

Spearman’s $\rho$ is the correlations between rank values of each word pair.

<table>
<thead>
<tr>
<th>large-big</th>
<th>cow-cattle</th>
<th>machine-engine</th>
<th>bold-proud</th>
<th>new-ancient</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

$\rho = 0.699999$
Rather than count how many times a word appears in a given context, we can use the dependency parse as the context. This can be a little sparse, so it is often useful to encode inverse dependencies too. Can encode different types of relations and similarities.

<table>
<thead>
<tr>
<th></th>
<th>dog_dobj</th>
<th>and_cc</th>
<th>finds_l_dobj</th>
<th>it_dobj</th>
<th>_punct</th>
<th>the_det</th>
<th>if_mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

For more information see Levy & Goldberg, 2014. [Dependency Based Word Embeddings](https://papers.nips.cc/paper/5334-deep-structured-self-attention).
Recap

- Vector space models of meaning are much more flexible in encoding meaning
- We can use PPMI to remove some of the effects of frequency
- We can use matrix factorization to reduce the size of the vector space
- Cons:
  - We need to keep the whole matrix in memory while training
  - Is hard to parallelize
Dense Vectors

● What we really want is a way to get vectors like the ones produced through SVD without having to compute and store the whole matrix
  ○ Solution: Use machine learning methods
● Dense vectors, or word embeddings have become extremely popular over the past 4 years.
● The most widely cited work is Tomas Mikolov’s word2vec methods
  ○ There are a lot of opinions in the community about this style of vector based semantics. For an alternative viewpoint see Ken Church’s column.
The basic idea in word2vec is to capture two transformation matrices such that \( w_1 \times \text{word} \times w_2 = \text{context} \)

- We will get into specifics soon

The input into the word2vec algorithm is word-context pairs

- These pairs can either be window based (the original version) or dependency based (word2vecf modification)

Start with two random matrices, each with size \( N \times d \)

- For each word context pair encoded as a one-hot vectors (each index corresponds to exactly one word), update the weight matrices so that \( w_1 \times \text{word} \times w_2 = \text{context} \) is a little better
The particular version of word2vec we are talking about is known as Skipgram. The other main one is continuous bag of words (CBOW), it is the same idea but we are trying to predict the word given a context.
In word2vec, we take the first weight matrix, $w_1$ as the low-dimensionality embedding of the vectors. The second weight matrix is usually thrown away, but can be useful! (Word substitution task)

```
<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

dog
```

```

| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

bark
```
The particular machine learning used isn’t important for this class but there are several clever tricks that were used

- To really properly train something of this design, every word-pair combination should appear in the training, and the goal is to maximize predicting the correct word-context pair and minimize predicting the ones that don’t exist
  - In reality this would be way too slow
  - Instead we sample words randomly to serve as negative training examples
    - Real: dog - bark
    - Sampled: lollipop - bark, UMBC - bark, flying - bark, etc.

- When constructing the word-context pairs list for training, a random decision can be made dependent on the distance between two to include or exclude this from training
Word2Vec: Magic?

- It turns out that word2vec is almost equivalent to matrix factorization!
- Levy and Goldberg show that the dot product of a word and context vector is an approximation of the PPMI value.
If word2vec is just matrix factorization why all the fuss?

It is much more efficient than building up a whole matrix and then factorizing it
  ○ It can be run across multiple threads
  ○ It can now use much larger vocabularies

Because you have a smaller dimensionality, if you normalize each row of your matrix, the cosine distance between a word and all others is the matrix vector product of that word's vector and the word2vec matrix
  ○ Technically we could have done this with all the matrices, but it is a bad idea!
So I should always use word2vec right?

- Not exactly :(
- It turns out that much of the success of word2vec can be replicated in other vector space models by careful tuning of hyperparameters
  - Levy, Goldberg, and Dagan 2015 show this and release code for tuning the hyperparameters in many different models.
- In addition, word embeddings are hard to interpret, which could be important for some applications
  - This is an open area of research
But vector space models are the way to go at least right?

- It depends
- They are excellent to use in machine learning tasks and are fun
- How to represent certain relationships with them is still an ongoing question
  - It is really hard to differentiate synonymy and hyponymy, for example
- Why not combine them?!
  - A popular method for tuning word embeddings is to train them as normal, and then update them so that words with relationship in a lexical thesaurus are similar. *(Retrofitting)*
  - It is also possible to make a vector for a word using WordNet relationships as the context.