Distributed word representations

Christopher Potts

Stanford Linguistics

CS 224U: Natural language understanding
April 8 and 13
Plan

1. High-level goals and guiding hypotheses
2. Matrix designs
3. Vector comparison
4. Basic reweighting
5. Subword information
6. Visualization
7. Dimensionality reduction
   a. Latent Semantic Analysis
   b. Autoencoders
   c. GloVe
   d. word2vec
8. Retrofitting
### Meaning latent in co-occurrence patterns

<table>
<thead>
<tr>
<th>against</th>
<th>age</th>
<th>agent</th>
<th>ages</th>
<th>ago</th>
<th>agree</th>
<th>ahead</th>
<th>ain’t</th>
<th>air</th>
<th>aka</th>
<th>al</th>
</tr>
</thead>
<tbody>
<tr>
<td>against</td>
<td>2003</td>
<td>90</td>
<td>39</td>
<td>20</td>
<td>88</td>
<td>57</td>
<td>33</td>
<td>15</td>
<td>58</td>
<td>22</td>
</tr>
<tr>
<td>age</td>
<td>90</td>
<td>1492</td>
<td>14</td>
<td>39</td>
<td>71</td>
<td>38</td>
<td>12</td>
<td>4</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>agent</td>
<td>39</td>
<td>14</td>
<td>507</td>
<td>2</td>
<td>21</td>
<td>5</td>
<td>10</td>
<td>3</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>ages</td>
<td>20</td>
<td>39</td>
<td>2</td>
<td>290</td>
<td>32</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>ago</td>
<td>88</td>
<td>71</td>
<td>21</td>
<td>32</td>
<td>1164</td>
<td>37</td>
<td>25</td>
<td>11</td>
<td>34</td>
<td>11</td>
</tr>
<tr>
<td>agree</td>
<td>57</td>
<td>38</td>
<td>5</td>
<td>5</td>
<td>37</td>
<td>627</td>
<td>12</td>
<td>2</td>
<td>16</td>
<td>19</td>
</tr>
<tr>
<td>ahead</td>
<td>33</td>
<td>12</td>
<td>10</td>
<td>4</td>
<td>25</td>
<td>12</td>
<td>429</td>
<td>4</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>ain’t</td>
<td>15</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>11</td>
<td>2</td>
<td>4</td>
<td>166</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>air</td>
<td>58</td>
<td>18</td>
<td>9</td>
<td>6</td>
<td>34</td>
<td>16</td>
<td>12</td>
<td>0</td>
<td>746</td>
<td>5</td>
</tr>
<tr>
<td>aka</td>
<td>22</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>11</td>
<td>19</td>
<td>10</td>
<td>3</td>
<td>5</td>
<td>261</td>
</tr>
<tr>
<td>al</td>
<td>24</td>
<td>39</td>
<td>25</td>
<td>6</td>
<td>38</td>
<td>14</td>
<td>7</td>
<td>3</td>
<td>11</td>
<td>9</td>
</tr>
</tbody>
</table>
### Meaning latent in co-occurrence patterns

<table>
<thead>
<tr>
<th>Class</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>awful</td>
</tr>
<tr>
<td>0</td>
<td>terrible</td>
</tr>
<tr>
<td>0</td>
<td>lame</td>
</tr>
<tr>
<td>0</td>
<td>worst</td>
</tr>
<tr>
<td>0</td>
<td>disappointing</td>
</tr>
<tr>
<td>1</td>
<td>nice</td>
</tr>
<tr>
<td>1</td>
<td>amazing</td>
</tr>
<tr>
<td>1</td>
<td>wonderful</td>
</tr>
<tr>
<td>1</td>
<td>good</td>
</tr>
<tr>
<td>1</td>
<td>awesome</td>
</tr>
</tbody>
</table>

**A hopeless learning scenario**
### Meaning latent in co-occurrence patterns

<table>
<thead>
<tr>
<th>Class</th>
<th>Word</th>
<th>Pr(Class = 1)</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>awful</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>terrible</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>lame</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>worst</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>disappointing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>nice</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>amazing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>wonderful</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>good</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>awesome</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>w₁</td>
</tr>
<tr>
<td>w₂</td>
</tr>
<tr>
<td>w₃</td>
</tr>
<tr>
<td>w₄</td>
</tr>
</tbody>
</table>

**A hopeless learning scenario**
### Meaning latent in co-occurrence patterns

<table>
<thead>
<tr>
<th>Class Word</th>
<th>excellent</th>
<th>terrible</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 awful</td>
<td>-0.69</td>
<td>1.13</td>
</tr>
<tr>
<td>0 terrible</td>
<td>-0.13</td>
<td>3.09</td>
</tr>
<tr>
<td>0 lame</td>
<td>-1.00</td>
<td>0.69</td>
</tr>
<tr>
<td>0 worst</td>
<td>-0.94</td>
<td>1.04</td>
</tr>
<tr>
<td>0 disappointing</td>
<td>0.19</td>
<td>0.09</td>
</tr>
<tr>
<td>1 nice</td>
<td>0.08</td>
<td>-0.07</td>
</tr>
<tr>
<td>1 amazing</td>
<td>0.71</td>
<td>-0.06</td>
</tr>
<tr>
<td>1 wonderful</td>
<td>0.66</td>
<td>-0.76</td>
</tr>
<tr>
<td>1 good</td>
<td>0.21</td>
<td>0.11</td>
</tr>
<tr>
<td>1 awesome</td>
<td>0.67</td>
<td>0.26</td>
</tr>
</tbody>
</table>

**A promising learning scenario**
Meaning latent in co-occurrence patterns

<table>
<thead>
<tr>
<th>Class Word</th>
<th>excellent</th>
<th>terrible</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 awful</td>
<td>−0.69</td>
<td>1.13</td>
</tr>
<tr>
<td>0 terrible</td>
<td>−0.13</td>
<td>3.09</td>
</tr>
<tr>
<td>0 lame</td>
<td>−1.00</td>
<td>0.69</td>
</tr>
<tr>
<td>0 worst</td>
<td>−0.94</td>
<td>1.04</td>
</tr>
<tr>
<td>0 disappointing</td>
<td>0.19</td>
<td>0.09</td>
</tr>
<tr>
<td>1 nice</td>
<td>0.08</td>
<td>−0.07</td>
</tr>
<tr>
<td>1 amazing</td>
<td>0.71</td>
<td>−0.06</td>
</tr>
<tr>
<td>1 wonderful</td>
<td>0.66</td>
<td>−0.76</td>
</tr>
<tr>
<td>1 good</td>
<td>0.21</td>
<td>0.11</td>
</tr>
<tr>
<td>1 awesome</td>
<td>0.67</td>
<td>0.26</td>
</tr>
</tbody>
</table>

A promising learning scenario
High-level goals

1. Begin thinking about how vectors can encode the meanings of linguistic units.

2. Foundational concepts for vector-space model (VSMs).

3. A foundation for deep learning NLU models.

4. In your assignment and projects, you’re likely to use representations like these:
   - to understand and model linguistic and social phenomena; and/or
   - as inputs to other machine learning models.
Associated materials

1. Code
   a. vsm.py
   b. vsm_01_distributional.ipynb
   c. vsm_02_dimreduce.ipynb
   d. vsm_03_retrofitting.ipynb

2. Homework 1 and bake-off 1: hw_wordsim.ipynb

3. Screencasts:
   a. Overview [link]
   b. Vector comparison [link]
   c. Reweighting [link]
   d. Dimensionality reduction [link]

4. Core readings: Turney and Pantel 2010; Smith 2019; Pennington et al. 2014; Faruqui et al. 2015
Guiding hypotheses

Firth (1957)
“You shall know a word by the company it keeps.”

Firth (1957)
“the complete meaning of a word is always contextual, and no study of meaning apart from context can be taken seriously.”

Wittgenstein (1953)
“the meaning of a word is its use in the language”

Harris (1954)
“distributional statements can cover all of the material of a language without requiring support from other types of information.”

Turney and Pantel (2010)
“If units of text have similar vectors in a text frequency matrix, then they tend to have similar meanings.”
Great power, a great many design choices

tokenization
annotation
tagging
parsing
feature selection

.cluster texts by date/author/discourse context/

Matrix design
- word × document
- word × word
- word × search proximity
- adj. × modified noun
- word × dependency rel.

Reweighting
- probabilities
- length norm.
- TF-IDF
- PMI
- Positive PMI

Dimensionality reduction
- LSA
- PLSA
- LDA
- PCA
- NNMF

Vector comparison
- Euclidean
- Cosine
- Dice
- Jaccard
- KL

(Nearly the full cross-product to explore; only a handful of the combinations are ruled out mathematically. Models like GloVe and word2vec offer packaged solutions to design/weighting/reduction and reduce the importance of the choice of comparison method.)
Designs

1. High-level goals and guiding hypotheses
2. Matrix designs
3. Vector comparison
4. Basic reweighting
5. Subword information
6. Visualization
7. Dimensionality reduction
   a. Latent Semantic Analysis
   b. Autoencoders
   c. GloVe
   d. word2vec
8. Retrofitting
<table>
<thead>
<tr>
<th></th>
<th>against</th>
<th>age</th>
<th>agent</th>
<th>ages</th>
<th>ago</th>
<th>agree</th>
<th>ahead</th>
<th>ain’t</th>
<th>air</th>
<th>aka</th>
<th>al</th>
</tr>
</thead>
<tbody>
<tr>
<td>against</td>
<td>2003</td>
<td>90</td>
<td>39</td>
<td>20</td>
<td>88</td>
<td>57</td>
<td>33</td>
<td>15</td>
<td>58</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td>age</td>
<td>90</td>
<td>1492</td>
<td>14</td>
<td>39</td>
<td>71</td>
<td>38</td>
<td>12</td>
<td>4</td>
<td>18</td>
<td>4</td>
<td>39</td>
</tr>
<tr>
<td>agent</td>
<td>39</td>
<td>14</td>
<td>507</td>
<td>2</td>
<td>21</td>
<td>5</td>
<td>10</td>
<td>3</td>
<td>9</td>
<td>8</td>
<td>25</td>
</tr>
<tr>
<td>ages</td>
<td>20</td>
<td>39</td>
<td>2</td>
<td>290</td>
<td>32</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>ago</td>
<td>88</td>
<td>71</td>
<td>21</td>
<td>32</td>
<td>1164</td>
<td>37</td>
<td>25</td>
<td>11</td>
<td>34</td>
<td>11</td>
<td>38</td>
</tr>
<tr>
<td>agree</td>
<td>57</td>
<td>38</td>
<td>5</td>
<td>5</td>
<td>37</td>
<td>627</td>
<td>12</td>
<td>2</td>
<td>16</td>
<td>19</td>
<td>14</td>
</tr>
<tr>
<td>ahead</td>
<td>33</td>
<td>12</td>
<td>10</td>
<td>4</td>
<td>25</td>
<td>12</td>
<td>429</td>
<td>4</td>
<td>12</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>ain’t</td>
<td>15</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>11</td>
<td>2</td>
<td>4</td>
<td>166</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>air</td>
<td>58</td>
<td>18</td>
<td>9</td>
<td>6</td>
<td>34</td>
<td>16</td>
<td>12</td>
<td>0</td>
<td>746</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>aka</td>
<td>22</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>11</td>
<td>19</td>
<td>10</td>
<td>3</td>
<td>5</td>
<td>261</td>
<td>9</td>
</tr>
<tr>
<td>al</td>
<td>24</td>
<td>39</td>
<td>25</td>
<td>6</td>
<td>38</td>
<td>14</td>
<td>7</td>
<td>3</td>
<td>11</td>
<td>9</td>
<td>861</td>
</tr>
<tr>
<td></td>
<td>d1</td>
<td>d2</td>
<td>d3</td>
<td>d4</td>
<td>d5</td>
<td>d6</td>
<td>d7</td>
<td>d8</td>
<td>d9</td>
<td>d10</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>against</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>age</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>agent</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ages</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ago</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>agree</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ahead</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ain’t</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>air</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>aka</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

word x document
### word x discourse context

Upper left corner of an interjection x dialog-act tag matrix derived from the Switchboard Dialog Act Corpus:

<table>
<thead>
<tr>
<th></th>
<th>%</th>
<th>+</th>
<th>^2</th>
<th>^g</th>
<th>^h</th>
<th>^q</th>
<th>aa</th>
</tr>
</thead>
<tbody>
<tr>
<td>absolutely</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>95</td>
</tr>
<tr>
<td>actually</td>
<td>17</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>anyway</td>
<td>23</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>boy</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>bye</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bye-bye</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>dear</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>definitely</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>56</td>
</tr>
<tr>
<td>exactly</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>294</td>
</tr>
<tr>
<td>gee</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>goodness</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
phonological segment $\times$ feature values

Derived from [http://www.linguistics.ucla.edu/people/hayes/120a/](http://www.linguistics.ucla.edu/people/hayes/120a/).

Dimensions: $(141 \times 28)$.

<table>
<thead>
<tr>
<th></th>
<th>syllabic</th>
<th>stress</th>
<th>long</th>
<th>consonantal</th>
<th>sonorant</th>
<th>continuant</th>
<th>delayed.release</th>
<th>approximant</th>
<th>tap</th>
<th>trill</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>æ</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>œ</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>Æ</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>œ</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>Æ</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>æ</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>œ</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>ë</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</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>
</tr>
</tbody>
</table>
phonological segment × feature values

Derived from http://www.linguistics.ucla.edu/people/hayes/120a/.
Dimensions: (141 × 28).
Feature representations of data

- *the movie was horrible* becomes \([4, 0, 1/4]\).

- The complex, real-world response of an experimental subject to a particular example becomes \([0, 1]\) or \([118, 1]\).

- A human is modeled as a vector \([24, 140, 5, 12]\).

- A continuous, noisy speech stream is reduced to a restricted set of acoustic features.
Other designs

- word × dependency rel.
- word × syntactic context
- adj. × modified noun
- word × search query
- person × product
- word × person
- word × word × pattern
- verb × subject × object

·
Windows and scaling: What is a co-occurrence?
Windows and scaling: What is a co-occurrence?

from swerve of shore to bend of bay, brings

4 3 2 1 0 1 2 3 4 5
Windows and scaling: What is a co-occurrence?

from swerve of shore to bend of bay, brings

Window: 3
Scaling: flat

from swerve of shore to bend of bay, brings

• Larger, flatter windows capture more semantic information.
• Small, more scaled windows capture more syntactic (collocational) information.
• Textual boundaries can be separately controlled; core unit as the sentence/paragraph/document will have major consequences.
Windows and scaling: What is a co-occurrence?

from swerve of shore to bend of bay, brings

Window: 3
Scaling: flat
Scaling: $\frac{1}{n}$
Windows and scaling: What is a co-occurrence?

from swerve of shore to bend of bay, brings

4 3 2 1 0 1 2 3 4 5

- Larger, flatter windows capture more semantic information.
- Small, more scaled windows capture more syntactic (collocational) information.
- Textual boundaries can be separately controlled; core unit as the sentence/paragraph/document will have major consequences.
import os
import pandas as pd

DATA_HOME = os.path.join('data', 'vsmdata')

# IMDB: Window size = 5; scaling = 1/n
imdb5 = pd.read_csv(
    os.path.join(DATA_HOME, 'imdb_window5-scaled.csv.gz'), index_col=0)

# IMDB: Window size = 20; scaling = flat
imdb20 = pd.read_csv(
    os.path.join(DATA_HOME, 'imdb_window20-flat.csv.gz'), index_col=0)

# Gigaword: Window size = 5; scaling = 1/n
giga5 = pd.read_csv(
    os.path.join(DATA_HOME, 'giga_window5-scaled.csv.gz'), index_col=0)

# Gigaword: Window size = 20; scaling = flat
giga20 = pd.read_csv(
    os.path.join(DATA_HOME, 'giga_window20-flat.csv.gz'), index_col=0)
Vector comparison

1. High-level goals and guiding hypotheses
2. Matrix designs
3. Vector comparison
4. Basic reweighting
5. Subword information
6. Visualization
7. Dimensionality reduction
   a. Latent Semantic Analysis
   b. Autoencoders
   c. GloVe
   d. word2vec
8. Retrofitting
Running example

- Focus on distance measures
- Illustrations with row vectors
Euclidean

Between vectors $u$ and $v$ of dimension $n$:

$$\text{euclidean}(u, v) = \sqrt{\sum_{i=1}^{n} |u_i - v_i|^2}$$
Length normalization

Given a vector $u$ of dimension $n$, the L2-length of $u$ is

$$\|u\|_2 = \sqrt{\sum_{i=1}^{n} u_i^2}$$

and the length normalization of $u$ is

$$\left[ \frac{u_1}{\|u\|_2}, \frac{u_2}{\|u\|_2}, \cdots, \frac{u_n}{\|u\|_2} \right]$$
Length normalization

<table>
<thead>
<tr>
<th></th>
<th>$d_x$</th>
<th>$d_y$</th>
<th>$|u|_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>4</td>
<td>4.47</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>15</td>
<td>18.03</td>
</tr>
<tr>
<td>C</td>
<td>14</td>
<td>10</td>
<td>17.20</td>
</tr>
</tbody>
</table>

row L2 norm

<table>
<thead>
<tr>
<th></th>
<th>$d_x$</th>
<th>$d_y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>C</td>
<td>14</td>
<td>10</td>
</tr>
</tbody>
</table>

0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

(10,15) B
(14,10) C
(2,4) A

(0.45,0.89) A
(0.55,0.83) B
(0.81,0.58) C

(0.55,0.83) B
(0.45,0.89) A
(0.81,0.58) C
Cosine distance

Between vectors $u$ and $v$ of dimension $n$:

$$\text{cosine}(u, v) = 1 - \frac{\sum_{i=1}^{n} u_i \times v_i}{||u||_2 \times ||v||_2}$$
Cosine distance

Between vectors $u$ and $v$ of dimension $n$:

$$\text{cosine}(u, v) = 1 - \frac{\sum_{i=1}^{n} u_i \times v_i}{\|u\|_2 \times \|v\|_2}$$

<table>
<thead>
<tr>
<th>$d_x$</th>
<th>$d_y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>14</td>
</tr>
</tbody>
</table>

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

$\|10, 15\| \times \|14, 10\| = 0.065$

$1 - \frac{(2 \times 10 + 4 \times 15)}{\|2, 4\| \times \|10, 15\|} = 0.008$

$1 - \frac{(10 \times 14 + 15 \times 10)}{\|10, 15\| \times \|14, 10\|} = 0.065$
Cosine distance

Between vectors $u$ and $v$ of dimension $n$:

$$\text{cosine}(u, v) = 1 - \frac{\sum_{i=1}^{n} u_i \times v_i}{\|u\|_2 \times \|v\|_2}$$
Matching-based methods

Matching coefficient

\[
\text{matching}(u, v) = \sum_{i=1}^{n} \min(u_i, v_i)
\]

Jaccard distance

\[
\text{jaccard}(u, v) = 1 - \frac{\text{matching}(u, v)}{\sum_{i=1}^{n} \max(u_i, v_i)}
\]

Dice distance

\[
\text{dice}(u, v) = 1 - \frac{2 \times \text{matching}(u, v)}{\sum_{i=1}^{n} u_i + v_i}
\]

Overlap

\[
\text{overlap}(u, v) = 1 - \frac{\text{matching}(u, v)}{\min(\sum_{i=1}^{n} u_i, \sum_{i=1}^{n} v_i)}
\]
KL divergence

Between probability distributions $p$ and $q$:

$$D(p \parallel q) = \sum_{i=1}^{n} p_i \log \left( \frac{p_i}{q_i} \right)$$

$p$ is the reference distribution. Before calculation, smooth by adding $\epsilon$. 
 KL divergence

\[
\begin{array}{c|c|c}
\text{d}_x & \text{d}_y \\
\hline
A & 2 & 4 \\
B & 10 & 15 \\
C & 14 & 10 \\
\end{array}
\]

Normalize the rows

\[
\begin{array}{c|c|c}
\text{d}_x & \text{d}_y \\
\hline
A & 0.33 & 0.67 \\
B & 0.40 & 0.60 \\
C & 0.58 & 0.42 \\
\end{array}
\]

\[
(0.33 \times \log(0.33/0.4)) + (0.67 \times \log(0.67/0.6)) = 0.01
\]

\[
(0.4 \times \log(0.4/0.58)) + (0.6 \times \log(0.6/0.42)) = 0.13
\]
KL variants

Symmetric KL

\[ D(p \parallel q) + D(q \parallel p) \]

KL-divergence with skew

\[ D(p \parallel \alpha q + (1 - \alpha)p) \quad 0 \leq \alpha \leq 1 \]

\begin{align*}
\alpha = 1 &; \text{SkewKL} = 1.17 \\
\alpha = 0.8 &; \text{SkewKL} = 0.63 \\
\alpha = 0.5 &; \text{SkewKL} = 0.25 \\
\alpha = 0.2 &; \text{SkewKL} = 0.05 \\
\alpha = 0 &; \text{SkewKL} = 0
\end{align*}

Jensen–Shannon distance

\[ \sqrt{\frac{1}{2} D\left( p \parallel \frac{p + q}{2} \right) + \frac{1}{2} D\left( q \parallel \frac{p + q}{2} \right) } \]
Relationships and generalizations

1. Euclidean, Jaccard, and Dice with raw count vectors will tend to favor raw frequency over distributional patterns.

2. Euclidean with L2-normed vectors is equivalent to cosine w.r.t. ranking (Manning and Schütze 1999:301).

3. Jaccard and Dice are equivalent w.r.t. ranking.

4. Both L2-norms and probability distributions can obscure differences in the amount/strength of evidence, which can in turn have an effect on the reliability of cosine, normed-euclidean, and KL divergence. These shortcomings might be addressed through weighting schemes.
Proper distance metric?

To qualify as a distance metric, a vector comparison method \(d\) has to be symmetric (\(d(x, y) = d(y, x)\)), assign 0 to identical vectors (\(d(x, x) = 0\)), and satisfy the **triangle inequality**:

\[
d(x, z) \leq d(x, y) + d(y, z)
\]

Cosine distance as I defined it doesn’t satisfy this:

**Distance metric?**

**Yes**: Euclidean, Jaccard for binary vectors, Jensen–Shannon, cosine as

\[
\cos^{-1}\left(\frac{\sum_{i=1}^{n} u_i \times v_i}{||u||_2 \times ||v||_2}\right)
\]

**No**: Matching, Jaccard, Dice, Overlap, KL divergence, Symmetric KL, KL with skew
Comparing the two versions of cosine

Random sample of 100 vectors from our giga20 count matrix. Correlation is 99.8.
## Code snippets

```python
In [1]: import os
       import pandas as pd
       import vsm

In [2]: ABC = pd.DataFrame([
                            [2.0, 4.0],
                            [10.0, 15.0],
                            [14.0, 10.0]], index=['A', 'B', 'C'], columns=['x', 'y'])

In [3]: vsm.euclidean(ABC.loc['A'], ABC.loc['B'])
Out[3]: 13.601470508735444

In [4]: vsm.vector_length(ABC.loc['A'])
Out[4]: 4.47213595499958

In [5]: vsm.length_norm(ABC.loc['A']).values
Out[5]: array([0.4472136 , 0.89442719])

In [6]: vsm.cosine(ABC.loc['A'], ABC.loc['B'])
Out[6]: 0.007722123286332261

In [7]: vsm.matching(ABC.loc['A'], ABC.loc['B'])
Out[7]: 6.0

In [8]: vsm.jaccard(ABC.loc['A'], ABC.loc['B'])
Out[8]: 0.76
```
Code snippets

In [9]: DATA_HOME = os.path.join('data', 'vsmdata')

    imdb5 = pd.read_csv(
        os.path.join(DATA_HOME, 'imdb_window5-scaled.csv.gz'), index_col=0)

In [10]: vsm.cosine(imdb5.loc['good'], imdb5.loc['excellent'])
Out[10]: 0.9644382411451131

In [11]: vsm.cosine(imdb5.loc['good'], imdb5.loc['bad'])
Out[11]: 0.9480014759326252

In [12]: vsm.neighbors('bad', imdb5).head()
Out[12]:  
   bad  0.000000
   guys  0.823744
   .  0.844851
  taste  0.893747
  guy   0.896312
dtype: float64

In [13]: vsm.neighbors('bad', imdb5, distfunc=vsm.jaccard).head(3)
Out[13]:  
   bad  0.000000
   think 0.783744
  better 0.788782
dtype: float64
Basic reweighting

1. High-level goals and guiding hypotheses
2. Matrix designs
3. Vector comparison
4. Basic reweighting
5. Subword information
6. Visualization
7. Dimensionality reduction
   a. Latent Semantic Analysis
   b. Autoencoders
   c. GloVe
   d. word2vec
8. Retrofitting
Goals of reweighting

- Amplify the important, the trustworthy, the unusual; deemphasize the mundane and the quirky.
- Absent a defined objective function, this will remain fuzzy.
- The intuition behind moving away from raw counts is that frequency is a poor proxy for the above values.
- So we should ask of each weighting scheme:
  - How does it compare to the raw count values?
  - How does it compare to the word frequencies?
  - What overall distribution of values does it deliver?
- We hope to do no feature selection based on counts, stopword dictionaries, etc. Rather, we want our methods to reveal what’s important without these ad hoc interventions.
Normalization

L2 norming (repeated from earlier)

Given a vector $u$ of dimension $n$, the L2-length of $u$ is

$$||u||_2 = \sqrt{\sum_{i=1}^{n} u_i^2}$$

and the length normalization of $u$ is

$$\left[ \frac{u_1}{||u||_2}, \frac{u_2}{||u||_2}, \ldots, \frac{u_n}{||u||_2} \right]$$

Probability distribution

Given a vector $u$ of dimension $n$ containing all positive values, let

$$\text{sum}(u) = \sum_{i=1}^{n} u_i$$

and then the probability distribution of $u$ is

$$\left[ \frac{u_1}{\text{sum}(u)}, \frac{u_2}{\text{sum}(u)}, \ldots, \frac{u_n}{\text{sum}(u)} \right]$$
Observed/Expected

\[
\text{rowsum}(X, i) = \sum_{j=1}^{n} X_{ij} \quad \text{colsum}(X, j) = \sum_{i=1}^{m} X_{ij} \quad \text{sum}(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij}
\]

\[
\text{expected}(X, i, j) = \frac{\text{rowsum}(X, i) \cdot \text{colsum}(X, j)}{\text{sum}(X)}
\]

\[
\text{oe}(X, i, j) = \frac{X_{ij}}{\text{expected}(X, i, j)}
\]
Observed/Expected

\[
\text{rowsum}(X, i) = \sum_{j=1}^{n} X_{ij} \quad \text{colsum}(X, j) = \sum_{i=1}^{m} X_{ij} \quad \text{sum}(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij}
\]

\[
\text{expected}(X, i, j) = \frac{\text{rowsum}(X, i) \cdot \text{colsum}(X, j)}{\text{sum}(X)}
\]

\[
\text{oe}(X, i, j) = \frac{X_{ij}}{\text{expected}(X, i, j)}
\]

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>rowsum</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>34</td>
<td>11</td>
<td>45</td>
</tr>
<tr>
<td>y</td>
<td>47</td>
<td>7</td>
<td>54</td>
</tr>
<tr>
<td>colsum</td>
<td>81</td>
<td>18</td>
<td>99</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c|cc}
\text{oe} & a & b \\
\hline
\text{x} & \frac{34}{45} & \frac{11}{45} \\
\text{y} & \frac{47}{54} & \frac{7}{54}
\end{array}
\]
Observed/Expected

\[
\text{rowsum}(X, i) = \sum_{j=1}^{n} X_{ij} \quad \text{colsum}(X, j) = \sum_{i=1}^{m} X_{ij} \quad \text{sum}(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij}
\]

\[
\text{expected}(X, i, j) = \frac{\text{rowsum}(X, i) \cdot \text{colsum}(X, j)}{\text{sum}(X)}
\]

\[
\text{oe}(X, i, j) = \frac{X_{ij}}{\text{expected}(X, i, j)}
\]

<table>
<thead>
<tr>
<th></th>
<th>Observed</th>
<th></th>
<th></th>
<th>Expected</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tabs</td>
<td>reading</td>
<td>birds</td>
<td>tabs</td>
<td>reading</td>
<td>birds</td>
<td></td>
</tr>
<tr>
<td>keep</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>60\frac{21}{101}</td>
<td>60\frac{40}{101}</td>
<td>60\frac{40}{101}</td>
<td></td>
</tr>
<tr>
<td>enjoy</td>
<td>1</td>
<td>20</td>
<td>20</td>
<td>41\frac{21}{101}</td>
<td>41\frac{40}{101}</td>
<td>41\frac{40}{101}</td>
<td></td>
</tr>
</tbody>
</table>

*keep* and *tabs* co-occur more than expected given their frequencies, *enjoy* and *tabs* less than expected
Pointwise Mutual Information (PMI)

PMI is observed/expected in log-space (with log(0) = 0):

\[
\text{pmi}(X, i, j) = \log \left( \frac{X_{ij}}{\text{expected}(X, i, j)} \right) = \log \left( \frac{P(X_{ij})}{P(X_{i*}) \cdot P(X_{*j})} \right)
\]

<table>
<thead>
<tr>
<th></th>
<th>(d_1)</th>
<th>(d_2)</th>
<th>(d_3)</th>
<th>(d_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\Rightarrow P(w, d) = \begin{bmatrix} 0.11 & 0.11 & 0.11 & 0.11 \ 0.11 & 0.11 & 0.00 & 0.00 \ 0.11 & 0.11 & 0.00 & 0.00 \ 0.00 & 0.00 & 0.00 & 0.01 \end{bmatrix}, \quad P(w) = \begin{bmatrix} 0.44 \ 0.33 \ 0.22 \ 0.01 \end{bmatrix}
\]

\[
P(d) = \begin{bmatrix} 0.33 \ 0.33 \ 0.22 \ 0.12 \end{bmatrix}
\]

PMI

<table>
<thead>
<tr>
<th></th>
<th>(d_1)</th>
<th>(d_2)</th>
<th>(d_3)</th>
<th>(d_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-0.28</td>
<td>-0.28</td>
<td>0.13</td>
<td>0.73</td>
</tr>
<tr>
<td>B</td>
<td>0.01</td>
<td>0.01</td>
<td>0.42</td>
<td>0.00</td>
</tr>
<tr>
<td>C</td>
<td>0.42</td>
<td>0.42</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>D</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>2.11</td>
</tr>
</tbody>
</table>
Selected PMI values

P(word, context) = 0.25

P(word)
P(context)
P(word, context) = 
1.02
0
-0.67
-1.18
0.51
0.17
-0.08
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
0.25
Positive PMI

The issue
PMI is actually undefined when $X_{ij} = 0$. The usual response is the one given above: set PMI to 0 in such cases. However, this is arguably not coherent (Levy and Goldberg 2014):

- Larger than expected count $\Rightarrow$ large PMI
- Smaller than expected count $\Rightarrow$ small PMI
- 0 count $\Rightarrow$ placed right in the middle!?
TF-IDF

For a corpus of documents $D$:

- Term frequency (TF): $P(w|d)$

- Inverse document frequency (IDF): $\log\left(\frac{|D|}{|\{d\in D: w\in d\}|}\right)$ ($\log(0) = 0$)

- TF-IDF: $TF \times IDF$

<table>
<thead>
<tr>
<th></th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$B$</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>$C$</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$D$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
IDF values

The graph shows the inverse document frequency (IDF) values calculated as $\text{IDF} = \log\left(\frac{10}{\text{docCount}}\right)$. The x-axis represents the number of documents (docCount), and the y-axis represents the IDF values. The corpus size is indicated as the value when docCount reaches 10.
Selected TF-IDF values

TF vs docCount scatter plot with selected values:
- TF = 0.01, docCount = 0.23
- TF = 0.02, docCount = 0.07
- TF = 0.05
- TF = 0.11
- TF = 0.07, docCount = 0.02
- TF = 0.14
- TF = 0.35
- TF = 0.69
- TF = 0.23, docCount = 1.15
- TF = 0.46
- TF = 2.3
Other weighting/normalization schemes

- t-test: \[ \frac{P(w,d) - P(w)P(d)}{\sqrt{P(w)P(d)}} \]

- TF-IDF variants that seek to be sensitive to the empirical distribution of words (For discussion and references, Manning and Schütze 1999:553.)

- Pairwise distance matrices:

<table>
<thead>
<tr>
<th></th>
<th>( d_x )</th>
<th>( d_y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>( B )</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>( C )</td>
<td>14</td>
<td>10</td>
</tr>
</tbody>
</table>

\[ \text{cosine} \Rightarrow \]

<table>
<thead>
<tr>
<th></th>
<th>( A )</th>
<th>( B )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>0</td>
<td>0.008</td>
<td>0.116</td>
</tr>
<tr>
<td>( B )</td>
<td>0.008</td>
<td>0</td>
<td>0.065</td>
</tr>
<tr>
<td>( C )</td>
<td>0.116</td>
<td>0.065</td>
<td>0</td>
</tr>
</tbody>
</table>
Weighting scheme cell-value distributions

Uses the giga5 matrix loaded earlier. Others look similar.
Weighting scheme relationships to counts

Uses the giga5 matrix loaded earlier. Others look similar.
Relationships and generalizations

• The theme running through nearly all these schemes is that we want to weight a cell value $X_{ij}$ relative to the value we expect given $X_{i*}$ and $X_{*j}$.

• Many weighting schemes end up favoring rare events that may not be trustworthy.

• The magnitude of counts can be important; [1, 10] and [1000, 10000] might represent very different situations; creating probability distributions or length normalizing will obscure this.

• PMI and its variants will amplify the values of counts that are tiny relative to their rows and columns. Unfortunately, with language data, these are often noise.

• TF-IDF severely punishes words that appear in many documents – it behaves oddly for dense matrices, which can include word $\times$ word matrices.
In [1]: import os
   ...: import pandas as pd
   ...: import vsm
   ...:
   ...:
   ...: DATA_HOME = os.path.join('data', 'vsmdata')
   ...:
   ...:
   ...: imdb5 = pd.read_csv(
   ...:     os.path.join(DATA_HOME, 'imdb_window5-scaled.csv.gz'), index_col=0)
   ...:
   ...:
   ...: imdb5_oe = vsm.observed_over_expected(imdb5)
   ...:
   ...:
   ...: imdb5_norm = imdb5.apply(vsm.length_norm, axis=1)
   ...:
   ...:
   ...: imdb5_ppmi = vsm.pmi(imdb5)
   ...:
   ...:
   ...: imdb5_pmi = vsm.pmi(imdb5, positive=False)
   ...:
   ...:
   ...: imdb5_tfidf = vsm.tfidf(imdb5)
Code snippets

In [2]: vsm.neighbors('bad', imdb5).head()

Out[2]:
bad  0.000000
  guys  0.823744
  .  0.844851
  taste  0.893747
  guy  0.896312
dtype: float64

In [3]: vsm.neighbors('bad', imdb5_ppmi).head()

Out[3]:
bad  0.000000
  good  0.701241
  awful  0.757309
  terrible  0.763324
  horrible  0.763637
dtype: float64
Subword information

1. High-level goals and guiding hypotheses
2. Matrix designs
3. Vector comparison
4. Basic reweighting
5. **Subword information**
6. Visualization
7. Dimensionality reduction
   a. Latent Semantic Analysis
   b. Autoencoders
   c. GloVe
   d. word2vec
8. Retrofitting
Motivation

1. Schütze (1993) pioneered subword modeling to improve representations by reducing sparsity, thereby increasing the density of connections in a VSM.

2. Subword modeling will also
   a. Pull morphological variants closer together
   b. Facilitate modeling out-of-vocabulary items
   c. Reduce the importance of any particular tokenization scheme
Bojanowski et al. (2016) (the fastText team) motivate a straightforward approach:

1. Given a word-level VSM, the vector for a character-level $n$-gram $x$ is the sum of all the vectors of words containing $x$.

2. Represent each word $w$ as the sum of its character-level $n$-grams.

3. Add in the representation of $w$ if available

A linguistically richer variant might use sequences of morphemes rather than characters.

Example with 4-grams

`superbly` becomes

`[<w>sup, supe, uper, perb, erbl, rbly, bly</w>]`
Code snippets

```
In [1]: import os
   import pandas as pd
   import vsm

   DATA_HOME = os.path.join('data', 'vsmdata')

   imdb5 = pd.read_csv(os.path.join(DATA_HOME, 'imdb_window5-scaled.csv.gz'), index_col=0)

In [2]: imdb5_ngrams = vsm.ngram_vsm(imdb5, n=4)

In [3]: imdb5_ngrams.loc['<w>sup'].values

Out[3]: array([3.41545000e+03, 3.70000000e+01, 4.95458333e+04, ...,
     2.23950000e+02, 4.64833333e+01, 3.12166667e+01])

In [4]: imdb5_ngrams.shape

Out[4]: (9806, 5000)

In [5]: vsm.get_character_ngrams("superbly", n=4)

Out[5]: ['<w>sup', 'supe', 'uper', 'perb', 'erbl', 'rbly', 'bly<w>']

In [6]: def character_level_rep(word, cf, n=4):
   ngrams = vsm.get_character_ngrams(word, n)
   ngrams = [n for n in ngrams if n in cf.index]
   reps = cf.loc[ngrams].values
   return reps.sum(axis=0)

In [7]: superbly = character_level_rep("superbly", imdb5_ngrams)

In [8]: superbly.shape

Out[8]: (5000,)
```
Word-piece tokenizing

Later in the term, we’ll encounter tokenizers that break some words into subword chunks:

Encode this sentence.

```
En
##code
this
sentence.
```

Bert knows Snuffleupagus

```
Bert
knows
S
##nu
##ffle
##up
##agu
##s
```

https://github.com/google/sentencepiece
Visualization

1. High-level goals and guiding hypotheses
2. Matrix designs
3. Vector comparison
4. Basic reweighting
5. Subword information

6. Visualization

7. Dimensionality reduction
   a. Latent Semantic Analysis
   b. Autoencoders
   c. GloVe
   d. word2vec

8. Retrofitting
Techniques

- Our goal is to visualize very high-dimensional spaces in two or three dimensions. **This will inevitably involve compromises.**

- Still, visualization can give you a feel for what is in your VSM, especially if you pair it with other kinds of qualitative exploration (e.g., using `vsm.neighbors`).

- There are many visualization techniques implemented in `sklearn.manifold`; see **this user guide** for an overview and discussion of trade-offs.
t-SNE on the giga20 PPMI VSM
t-SNE on the giga20 PPMI VSM

cooking

conflict
t-SNE on the imdb20 PPMI VSM
t-SNE on the imdb20 PPMI VSM


positivity

negativity
**Code snippets**

In [1]: %matplotlib inline
   
   from nltk.corpus import opinion_lexicon
   import os
   import pandas as pd
   import vsm

In [2]: DATA_HOME = os.path.join('data', 'vsmdata')
   
   imdb5 = pd.read_csv(os.path.join(DATA_HOME, 'imdb_window5-scaled.csv.gz'), index_col=0)

In [3]: imdb5_ppmi = vsm.pmi(imdb5)

In [4]: # Supply a str filename to write the output to a file:
   
   vsm.tsne_viz(imdb5_ppmi, output_filename=None)

In [5]: # To display words in different colors based on external criteria:
   
   positive = set(opinion_lexicon.positive())
   negative = set(opinion_lexicon.negative())
   
   colors = []
   for w in imdb5_ppmi.index:
       if w in positive:
           color = 'red'
       elif w in negative:
           color = 'blue'
       else:
           color = 'gray'
       colors.append(color)
   
   vsm.tsne_viz(imdb5_ppmi, colors=colors)
Latent Semantic Analysis (LSA)

1. High-level goals and guiding hypotheses
2. Matrix designs
3. Vector comparison
4. Basic reweighting
5. Subword information
6. Visualization
7. Dimensionality reduction
   a. Latent Semantic Analysis
   b. Autoencoders
   c. GloVe
   d. word2vec
8. Retrofitting
Overview

- Due to Deerwester et al. 1990.

- One of the oldest and most widely used dimensionality reduction techniques.

- Also known as Truncated Singular Value Decomposition (Truncated SVD).

- Standard baseline, often very tough to beat.
Guiding intuitions for LSA
The LSA method

Singular value decomposition
For any matrix of real numbers $A$ of dimension $(m \times n)$ there exists a factorization into matrices $T$, $S$, $D$ such that

$$A_{m \times n} = T_{m \times m} S_{m \times m} D_{n \times m}^T$$

$$
\begin{pmatrix}
\vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots \\
\end{pmatrix}
= 
\begin{pmatrix}
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\end{pmatrix}
\begin{pmatrix}
\cdot & \cdot & \cdot & \cdot & \cdot \\
\end{pmatrix}
\begin{pmatrix}
\vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots \\
\end{pmatrix}
^T$$

$A_{3 \times 4} = T_{3 \times 3} S_{3 \times 3} D_{4 \times 3}^T$
Idealized LSA example

\[ \begin{array}{cccccc}
\text{d1} & \text{d2} & \text{d3} & \text{d4} & \text{d5} & \text{d6} \\
\hline
\text{gnarly} & 1 & 0 & 1 & 0 & 0 \\
\text{wicked} & 0 & 1 & 0 & 1 & 0 \\
\text{awesome} & 1 & 1 & 1 & 1 & 0 \\
\text{lame} & 0 & 0 & 0 & 0 & 1 \\
\text{terrible} & 0 & 0 & 0 & 0 & 1 \\
\end{array} \]

Distance from gnarly
1. gnarly
2. awesome
3. terrible
4. wicked
5. lame

\[\begin{bmatrix}
gnarly & 0.41 & 0.00 & 0.71 & 0.00 & -0.58 \\
wicked & 0.41 & 0.00 & -0.71 & 0.00 & -0.58 \\
awesome & 0.82 & -0.00 & -0.00 & -0.00 & 0.58 \\
lame & 0.00 & 0.85 & 0.00 & -0.53 & 0.00 \\
terrible & 0.00 & 0.53 & 0.00 & 0.85 & 0.00 \\
\end{bmatrix} \times
\begin{bmatrix}
2.45 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 1.62 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 1.41 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.62 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & -0.00 \\
\end{bmatrix}
\times
\begin{bmatrix}
d1 & 0.50 & -0.00 & 0.50 & 0.00 & -0.71 \\
d2 & 0.50 & 0.00 & -0.50 & 0.00 & 0.00 \\
d3 & 0.50 & -0.00 & 0.50 & 0.00 & 0.71 \\
d4 & 0.50 & -0.00 & -0.50 & -0.00 & 0.00 \\
d5 & -0.00 & 0.53 & 0.00 & -0.85 & 0.00 \\
d6 & 0.00 & 0.85 & 0.00 & 0.53 & 0.00 \\
\end{bmatrix}
\]

Distance from gnarly
1. gnarly
2. wicked
3. awesome
4. terrible
5. lame
Cell-value comparisons ($k = 100$)
Choosing the LSA dimensionality

**Dream scenario for choosing k**

- Singular value rank vs. Singular value
- PMI+LSA

- Singular value (log-scale)

```
Dream scenario for choosing k

Singular value rank
0  10  20  30  40
0.3  0.4  0.5  0.6  0.7

PMI+LSA

Singular value rank
0  1000  2000  3000  4000  5000
5.0  2.5  0.0  2.5  5.0  7.5
```
Related dimensionality reduction techniques

- Principal Components Analysis (PCA)
- Non-negative Matrix Factorization (NMF)
- Probabilistic LSA (PLSA; Hofmann 1999)
- Latent Dirichlet Allocation (LDA; Blei et al. 2003)
- t-SNE (van der Maaten and Hinton 2008)

See sklearn.decomposition and sklearn.manifold
Code snippets

In [1]: import os
   import pandas as pd
   import vsm

In [2]: DATA_HOME = os.path.join('data', 'vsmdata')

   giga5 = pd.read_csv(
       os.path.join(DATA_HOME, 'giga_window5-scaled.csv.gz'), index_col=0)

In [3]: giga5.shape

Out[3]: (5000, 5000)

In [4]: giga5_lsa100 = vsm.lsa(giga5, k=100)

In [5]: giga5_lsa100.shape

Out[5]: (5000, 100)
Autoencoders

1. High-level goals and guiding hypotheses
2. Matrix designs
3. Vector comparison
4. Basic reweighting
5. Subword information
6. Visualization
7. Dimensionality reduction
   a. Latent Semantic Analysis
   b. Autoencoders
   c. GloVe
   d. word2vec
8. Retrofitting
Overview

- Autoencoders are a flexible class of deep learning architectures for learning reduced dimensional representations.

- Chapter 14 of Goodfellow et al. (2016) is an excellent discussion.
The basic autoencoder model

Assume $f = \tanh$ and so $f'(z) = 1.0 - z^2$. Per example error is $\sum_{i} 0.5 * (x_{hat_i} - x_i)^2$

Seeks to predict its own input.

High-dimensional inputs are fed through a narrow hidden layer (or multiple hidden layers). This is the representation of interest – akin to LSA output.

This might be preceded by a separate dimensionality reduction step (e.g., LSA)
Autoencoder code snippets

```python
In [1]: from np_autoencoder import Autoencoder
   : import os
   : import pandas as pd
   : from torch_autoencoder import TorchAutoencoder
   : import vsm

In [2]: DATA_HOME = os.path.join('data', 'vsmdata')

   giga5 = pd.read_csv(
       os.path.join(DATA_HOME, 'giga_window5-scaled.csv.gz'), index_col=0)

In [3]: # You'll likely need a larger network, trained longer, for good results.
   : ae = Autoencoder(max_iter=10, hidden_dim=50)

In [4]: # Scaling the values first will help the network learn:
   : giga5_l2 = giga5.apply(vsm.length_norm, axis=1)

In [5]: # The `fit` method returns the hidden reps:
   : giga5_ae = ae.fit(giga5_l2)

Finished epoch 10 of 10; error is 0.4883386066987744

In [6]: torch_ae = TorchAutoencoder(max_iter=10, hidden_dim=50)

In [7]: # A potentially interesting pipeline:
   : giga5_ppmi_lsa100 = vsm.lsa(vsm.pmi(giga5), k=100)

In [8]: giga5_ppmi_lsa100_ae = torch_ae.fit(giga5_ppmi_lsa100)

Finished epoch 10 of 10; error is 1.2230274677276611
```
Autoencoder code snippets

```python
In [9]: vsm.neighbors("finance", giga5).head()
Out[9]:
finance 0.000000
minister 0.870300
. 0.880074
</p> minister 0.896013
ministry 0.897051
dtype: float64

In [10]: vsm.neighbors("finance", giga5_ae).head()
Out[10]:
finance 0.000000
article 0.504076
style 0.526473
domain 0.538920
investigators 0.548903
dtype: float64

In [11]: vsm.neighbors("finance", giga5_ppmi_lsa100_ae).head()
Out[11]:
finance 0.000000
affairs 0.232635
management 0.248080
commerce 0.255099
banking 0.256428
dtype: float64
```
Global Vectors (GloVe)

1. High-level goals and guiding hypotheses
2. Matrix designs
3. Vector comparison
4. Basic reweighting
5. Subword information
6. Visualization
7. Dimensionality reduction
   a. Latent Semantic Analysis
   b. Autoencoders
   c. GloVe
   d. word2vec
8. Retrofitting
Overview

• Pennington et al. (2014)

• Roughly speaking, the objective is to learn vectors for words such that their dot product is proportional to their probability of co-occurrence.

• We’ll use the implementation in the mittens package (Dingwall and Potts 2018). There is a reference implementation in vsm.py. For really big vocabularies, the GloVe team’s C implementation is probably the best choice.

• We’ll make use of the GloVe team’s pretrained representations throughout this course.
The GloVe objective

\[ w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik}) \]

Equation (6):

\[ w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i) \]

Allowing different rows and columns:

\[ w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i \cdot X_{*k}) \]

That’s PMI!

\[ \text{pmi}(X, i, j) = \log \left( \frac{X_{ij}}{\text{expected}(X, i, j)} \right) = \log \left( \frac{P(X_{ij})}{P(X_i \cdot P(X_{*j})} \right) \]

By the equivalence \( \log(x) = \log(x) - \log(y) \)
The weighted GloVe objective

Original

\[ w_i^T \cdot \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik}) \]

Weighted

\[
\sum_{i,j=1}^{\mid V \mid} f(X_{ij}) \left( w_i^T \cdot \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij} \right)^2
\]

where \( V \) is the vocabulary and \( f \) is

\[
f(x) \begin{cases} 
(x/x_{\text{max}})^{\alpha} & \text{if } x < x_{\text{max}} \\
1 & \text{otherwise}
\end{cases}
\]

Typically, \( \alpha \) is set to 0.75 and \( x_{\text{max}} \) to 100.
GloVe hyperparameters

- Learned representation dimensionality.
- $x_{\text{max}}$, which flattens out all high counts.
- $\alpha$, which scales the values as $(x/x_{\text{max}})^\alpha$.

$$f(x) = \begin{cases} 
(x/x_{\text{max}})^\alpha & \text{if } x < x_{\text{max}} \\
1 & \text{otherwise}
\end{cases}$$

$$f([100 \ 99 \ 75 \ 10 \ 1]) = 
\begin{bmatrix}
1.00 \\
0.99 \\
0.81 \\
0.18 \\
0.03
\end{bmatrix}$$
GloVe learning

The loss calculations
\[ f(X_{ij}) (w_i^T \bar{w}_j - \log X_{ij}) \]
show how *gnarly* and *wicked* are pulled toward *awesome*. Bias terms left out for simplicity. *Gnarly* and *wicked* deliberately far apart in \( w_0 \) and \( \bar{w}_0 \).

<table>
<thead>
<tr>
<th>Counts</th>
<th>gnarly</th>
<th>wicked</th>
<th>awesome</th>
<th>terrible</th>
</tr>
</thead>
<tbody>
<tr>
<td>gnarly</td>
<td>10</td>
<td>0</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>wicked</td>
<td>0</td>
<td>10</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>awesome</td>
<td>9</td>
<td>10</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>terrible</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weights ((x_{\max} = 10, \alpha = 0.75))</th>
<th>gnarly</th>
<th>wicked</th>
<th>awesome</th>
<th>terrible</th>
</tr>
</thead>
<tbody>
<tr>
<td>gnarly</td>
<td>1.00</td>
<td>0.00</td>
<td>0.92</td>
<td>0.18</td>
</tr>
<tr>
<td>wicked</td>
<td>0.00</td>
<td>1.00</td>
<td>0.92</td>
<td>0.18</td>
</tr>
<tr>
<td>awesome</td>
<td>0.92</td>
<td>0.92</td>
<td>1.00</td>
<td>0.18</td>
</tr>
<tr>
<td>terrible</td>
<td>0.18</td>
<td>0.18</td>
<td>0.18</td>
<td>0.41</td>
</tr>
</tbody>
</table>

\[
\begin{bmatrix}
0.27 \\
-0.27
\end{bmatrix}^T \begin{bmatrix}
0.03 & 0.20
\end{bmatrix} - \log(9) = -2.06
\]

\[
\begin{bmatrix}
-0.27 & 0.27
\end{bmatrix}^T \begin{bmatrix}
0.03 & 0.20
\end{bmatrix} - \log(9) = -1.98
\]

\[
\begin{bmatrix}
0.99 & -0.85
\end{bmatrix}^T \begin{bmatrix}
0.34 & -0.25
\end{bmatrix} - \log(9) = -1.51
\]

\[
\begin{bmatrix}
0.74 & -0.54
\end{bmatrix}^T \begin{bmatrix}
0.34 & -0.25
\end{bmatrix} - \log(9) = -1.66
\]
GloVe cell-value comparisons \((n = 50)\)
GloVe code snippets

```
In [1]: from mittens import GloVe
   : import numpy as np
   : import os
   : import pandas as pd
   : import vsm

In [2]: DATA_HOME = os.path.join('data', 'vsmdata')

   : imdb5 = pd.read_csv(
   :       os.path.join(DATA_HOME, 'imdb_window5-scaled.csv.gz'), index_col=0)

   : imdb20 = pd.read_csv(
   :       os.path.join(DATA_HOME, 'imdb_window20-flat.csv.gz'), index_col=0)

In [3]: # What percentage of the non-zero values are being mapped to 1 by f?
   : def percentage_nonzero_vals_above(df, n=100):
   :     v = df.values.reshape(1, -1).squeeze()
   :     v = v[v > 0]
   :     above = v[v > n]
   :     return len(above) / len(v)

In [4]: percentage_nonzero_vals_above(imdb5)
   : Out[4]: 0.017534398942316464

In [5]: percentage_nonzero_vals_above(imdb20)
   : Out[5]: 0.1519065095882084
```
## GloVe code snippets

```python
In [6]: glv = GloVe(max_iter=100, n=50)
In [7]: imdb5_glv = glv.fit(imdb5)
Iteration 100: loss: 536157.755
In [8]: glv.sess.close()
In [9]: imdb20_glv = glv.fit(imdb20)
Iteration 100: loss: 1043351.625
In [10]: # Restore the original `pd.DataFrame` structure:
   imdb20_glv = pd.DataFrame(imdb20_glv, index=imdb20.index)
In [11]: # To what a degree is the GloVe objective achieved?
   def correlation_test(true, pred):
      mask = true > 0
      M = pred.dot(pred.T)
      with np.errstate(divide='ignore'):
         log_cooccur = np.log(true)
         log_cooccur[np.isinf(log_cooccur)] = 0.0
         row_prob = np.log(true.sum(axis=1))
         row_log_prob = np.outer(row_prob, np.ones(true.shape[1]))
         prob = log_cooccur - row_log_prob
         return np.corrcoef(prob[mask], M[mask])[0, 1]
In [12]: correlation_test(imdb5.values, imdb5_glv)
Out[12]: 0.38032242586515264
In [13]: correlation_test(imdb20.values, imdb20_glv.values)
Out[13]: 0.484126476892789
```
word2vec

1. High-level goals and guiding hypotheses
2. Matrix designs
3. Vector comparison
4. Basic reweighting
5. Subword information
6. Visualization
7. Dimensionality reduction
   a. Latent Semantic Analysis
   b. Autoencoders
   c. GloVe
   d. word2vec
8. Retrofitting
Overview

- Introduced by Mikolov et al. (2013).
- Goldberg and Levy (2014) identify the relationship between word2vec and PMI.
- The TensorFlow tutorial Vector representations of words is very clear and links to code.
- Gensim package has a highly scalable implementation.
word2vec: From corpus to labeled data

it was the best of times, it was the worst of times, …

With window size 2:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>it</td>
<td>was</td>
</tr>
<tr>
<td>it</td>
<td>the</td>
</tr>
<tr>
<td>was</td>
<td>it</td>
</tr>
<tr>
<td>was</td>
<td>the</td>
</tr>
<tr>
<td>was</td>
<td>best</td>
</tr>
<tr>
<td>the</td>
<td>was</td>
</tr>
<tr>
<td>the</td>
<td>it</td>
</tr>
<tr>
<td>the</td>
<td>best</td>
</tr>
<tr>
<td>the</td>
<td>of</td>
</tr>
<tr>
<td>…</td>
<td></td>
</tr>
</tbody>
</table>
word2vec: Basic skip-gram

The basic skip-gram model estimates the probability of an input–output pair \((a, b)\) as

\[
P(b \mid a) = \frac{\exp(x_a w_b)}{\sum_{b' \in V} \exp(x_a w_{b'})}
\]

where \(x_a\) is the row-vector representation of word \(a\) and \(w_b\) is the column vector representation of word \(b\). Minimize:

\[
- \sum_{i=1}^{m} \sum_{k=1}^{|V|} 1\{c_i = k\} \log \frac{\exp(x_i w_k)}{\sum_{j=1}^{|V|} \exp(x_i w_j)}
\]

where \(V\) is the vocabulary and \(c\) is a one-hot encoded vector of the same length as \(V\). This gives rise to a classifier:

\[
C = \text{softmax}(XW + b)
\]

We’re back to our core insight for this unit: word and context matrix, pushing their dot products in a specific direction.
word2vec: Noise contrastive estimation

Training the basic skip-gram model directly is expensive for large vocabularies, because $W$, $b$, and $C$ get so large. Noise contrastive estimation addresses that:

$$\sum_{a,b \in D} - \log \sigma(x_aw_b) + \sum_{a,b \in D'} \log \sigma(x_aw_b)$$

with $\sigma$ the sigmoid activation function $\frac{1}{1+\exp(-x)}$. $D'$ is a sample of pairs that don’t appear in the training data.
Retrofitting

1. High-level goals and guiding hypotheses
2. Matrix designs
3. Vector comparison
4. Basic reweighting
5. Subword information
6. Visualization
7. Dimensionality reduction
   a. Latent Semantic Analysis
   b. Autoencoders
   c. GloVe
   d. word2vec
8. Retrofitting
Central goals

- Distributional representations are powerful and easy to obtain, but they tend to reflect only similarity (synonymy, connotation).

- Structured resources are sparse and hard to obtain, but they support learning rich, diverse semantic distinctions.

- Can we have the best aspects of both? Retrofitting is one way of saying, “Yes”.

- Retrofitting is due to Faruqui et al. (2015).
Purely distributional representations

- High-dimensional
- Meaning from dense linguistic inter-relationships
- Meaning *solely* from \((n\text{-th-order})\) co-occurrence
- No grounding in physical or social contexts
- Not symbolic
Grounding via supervision

Word vectors to maximize unsupervised log-likelihood of words given documents and supervised prediction accuracy:

(Maas et al. 2011)
Hidden representations from a deep classifier
The retrofitting model

\[
\sum_{i \in V} \alpha_i \|q_i - \hat{q}_i\|^2 + \sum_{(i,j,r) \in E} \beta_{ij} \|q_i - q_j\|^2
\]

- **Balances** fidelity to the original vector \(\hat{q}_i\)
- **against looking more like one’s graph neighbors.**
- **Forces are balanced with** \(\alpha = 1\) and \(\beta = \frac{1}{\text{Degree}(i)}\)

Figure 1: Word graph with edges between related words showing the observed (grey) and the inferred (white) word vector representations.
Simple retrofitting examples

\[ \sum_{i \in V} \alpha_i \| q_i - \hat{q}_i \|^2 + \sum_{(i,j,r) \in E} \beta_{ij} \| q_i - q_j \|^2 \]
Simple retrofitting examples

\[ \sum_{i \in V} \alpha_i \| \mathbf{q}_i - \hat{\mathbf{q}}_i \|^2 + \sum_{(i,j,r) \in E} \beta_{ij} \| \mathbf{q}_i - \mathbf{q}_j \|^2 \]
Simple retrofitting examples

\[
\sum_{i \in V} \alpha_i \| q_i - \hat{q}_i \|^2 + \sum_{(i,j,r) \in E} \beta_{ij} \| q_i - q_j \|^2
\]

\[\alpha = 0\]
Extensions

Drop the assumption that every edge means ‘similar’:

- Mrkšić et al. (2016) AntonymRepel, SynonymAttract, and VectorSpacePreservation for different edge types.

- Lengerich et al. (2018): functional retrofitting to learn the semantics of any edge types.

- This work is closely related to graph embedding (learning distributed representations for nodes), for which see Hamilton et al. 2017.
Retrofitting code snippets

```python
In [1]: import pandas as pd
   from retrofitting import Retrofitter

In [2]: Q_hat = pd.DataFrame(
   [[0.0, 0.0],
    [0.0, 0.5],
    [0.5, 0.0]],
   columns=['x', 'y'])

edges = {0: {1, 2}, 1: set(), 2: set()}

In [3]: Q_hat

Out[3]:
   x  y
  0  0.0  0.0
  1  0.0  0.5
  2  0.5  0.0

In [4]: retro = Retrofitter(verbos= True)

In [5]: X_retro = retro.fit(Q_hat, edges)

Converged at iteration 2; change was 0.0000

In [6]: X_retro

Out[6]:
   x  y
  0 0.125 0.125
  1 0.000 0.500
  2 0.500 0.000

In [7]: # For an application to WordNet, see `vsm_03_retrofitting`.
```
References


References II


