Social Media & Text Analysis
lecture 8 - Vector Semantics

CSE 5539-0010 Ohio State University
Instructor: Alan Ritter
Website: socialmedia-class.org

some slides are adapted from Michael Collins, Dan Jurafsky, Richard Socher,, Chris Manning
Part-of-Speech (POS) Tagging

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cant</td>
<td>MD</td>
</tr>
<tr>
<td>wait</td>
<td>VB</td>
</tr>
<tr>
<td>for</td>
<td>IN</td>
</tr>
<tr>
<td>the</td>
<td>DT</td>
</tr>
<tr>
<td>ravens</td>
<td>NNP</td>
</tr>
<tr>
<td>game</td>
<td>NN</td>
</tr>
<tr>
<td>tomorrow</td>
<td>NN</td>
</tr>
<tr>
<td>...</td>
<td>:</td>
</tr>
<tr>
<td>go</td>
<td>VB</td>
</tr>
<tr>
<td>ray</td>
<td>NNP</td>
</tr>
<tr>
<td>rice</td>
<td>NNP</td>
</tr>
<tr>
<td>!!!!!!!!</td>
<td>.</td>
</tr>
</tbody>
</table>

Cant wait for the ravens game tomorrow....go ray rice!!!!!!!
Named Entity Recognition

Source: Strauss, Toma, Ritter, de Marneffe, Xu
Results of the WNUT16 Named Entity Recognition Shared Task (WNUT@COLING 2016)
BAD LANGUAGE!
...on the INTERNET!!

Boom! Ya ur website suxx bro

...dats why pluto is pluto it can neva be a star

michelle obama great. job. and. whit all my. respect she. look. great. congrats. to. her.

What can we do about it?
Why don't they just write NORMALLY??

Can our software ever ADAPT???

I now h v an iphone

Jacob EISENSTEIN
GEORGIA Institute of TECHnology
How does language go bad?

Illiteracy? No. (Tagliamonte and Denis 2008; Drouin and Davis 2009)

Length limits? (probably not)

Hardware input constraints? (Gouws et al 2011)

Social variables
- Non-standard language does identity work, signaling authenticity, solidarity, etc.
- Social variation is usually inhibited in written language, but social media is less regulated than other written genres.

Source: Jacob Eisenstein
Why is Social Media Text “Bad”? 

- Lack of literacy? no [Drouin and Davis, 2009]
- Length restrictions? not primarily [Eisenstein, 2013]
- Text input method? to some degree, yes [Gouws et al., 2011]
- Mimicking prosodic effects etc. in speech? yeeees [Eisenstein, 2013]
- Social variables/markers of social identity? blood oath! [Eisenstein, 2013]

Source: Jacob Eisenstein & Tim Baldwin
Why is Social Media Text “Bad”?  

- mimicking prosodic effects etc. in speech? yeeees [Eisenstein, 2013]

**HELLA**

Derived from "hell of a lot". Similar to "very, really, a lot," etc.

Used mostly in Northern California though has been heard in other parts of CA and even in the media such as an infamous "hella" South Park episode. (Cartman used it outside of its meaning to annoy Kyle.)

*Before:* There's a hell of a lot of beer in that fridge.

*After:* There's hella beer in that fridge.

*As "very" or "really":*

"That's hella far away!"
Why is Social Media Text “Bad”? 

• Social variables/markers of social identity? blood oath! [Eisenstein, 2013]

“I would like to believe he’s sick rather than just mean and evil.”

“You could’ve been getting down to this sick beat.”

Source: Yang Yi and Jacob Eisenstein (TACL 2017) Overcoming Language Variation in Sentiment Analysis with Social Attention
Text Normalization

• convert non-standard words to standard

Source: Tim Baldwin, Marie de Marneffe, Han Bo, Young-Bum Kim, Alan Ritter, Wei Xu
Shared Tasks of the 2015 Workshop on Noisy User-generated Text: Twitter Lexical Normalization and Named Entity Recognition
An Unsupervised Learning Method:

(1) Brown Clustering

• Input:
  - a (large) text corpus

• Output:
  1. a partition of words into word clusters
  2. or a hierarchical word clustering (generalization of 1)
Brown Clustering

• Example Clusters (from Brown et al. 1992)

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays
June March July April January December October November September August
people guys folks fellows CEOs chaps doubters commies unfortunates blokes
down backwards ashore sideways southward northward overboard aloft downwards adrift
water gas coal liquid acid sand carbon steam shale iron
great big vast sudden mere sheer gigantic lifelong scant colossal
man woman boy girl lawyer doctor guy farmer teacher citizen
American Indian European Japanese German African Catholic Israeli Italian Arab
pressure temperature permeability density porosity stress velocity viscosity gravity tension
mother wife father son husband brother daughter sister boss uncle
machine device controller processor CPU printer spindle subsystem compiler plotter
John George James Bob Robert Paul William Jim David Mike
anyone someone anybody somebody

Source: Miller, Guinness, Zamanian (NAACL 2004)
Name Tagging with Word Clusters and Discriminative Training
Hierarchical Word Clustering

• Each intermediate node is a cluster:
Hierarchical Word Clustering

- Each intermediate node is a cluster:

```
    0
   / \
 00  01
 /     /
000 001 10  11
|     |    |   |
apple pear Apple IBM
```

bit string representation

level:

- 0
- 1
- 2
- 3
- 4
Hierarchical Word Clustering

- Example Clusters (from Miller et al. 2004)

mailman 100000110101110
salesman 100000110110000
bookkeeper 1000001101100010
troubleshooter 1000001101100110
bouncer 1000001101100111
technician 1000001101100100
janitor 1000001101100101
saleswoman 1000001101100110

Nike 1011011100100101011100
Maytag 1011011100100101011100
General 1011011100100101011101
Gap 1011011100100101011110
Harley-Davidson 1011011100100101011110
Enfield 1011011100100101011110
Genus 1011011100100101011111
Microsoft 10110111001001011100
Ventrix 10110111001001011101
Tractebel 10110111001001011110
Synopsys 10110111001001011111
WordPerfect 10110111001001101000

John 101110010000000000
Consuelo 10111001000000001
Jeffrey 10111001000000010
Kenneth 101110010000001100
Phillip 101110010000001101
WILLIAM 1011100100000011011
Timothy 101110010000001110
Hierarchical Word Clustering

- Example Clusters
  (from Miller et al. 2004)

mailman 10000111010111
salesman 10000111010000
bookkeeper 100001110100010
troubleshooter 1000011101000110
bouncer 1000011101000111
technician 100001110100100
janitor 100001110100101
saleswoman 100001110100110

... Nike 10110110010101011100
Maytag 1011011001010101110
General 10110110010101011111
Gap 10110110010101011110
Harley-Davidson 10110110010101011110
Enfield 10110110010101011110
Genus 10110110010101011111
Microsoft 10110110010101100
Ventritex 1011011001010110010
Tractebel 101101100101011010
Synopsys 1011011001010110011
WordPerfect 101101100101011000

... John 101110010000000000
Consuelo 10111001000000000001
Jeffrey 10111001000000000010
Kenneth 101110010000000001100
Phillip 101110010000000011010
WILLIAM 101110010000000011011
Timothy 101110010000000011110

Source: Miller, Guinness, Zamanian (NAACL 2004)
Name Tagging with Word Clusters and Discriminative Training
Hierarchical Word Clustering

- Example Clusters (from Miller et al. 2004)

<table>
<thead>
<tr>
<th>Word</th>
<th>Bit String</th>
</tr>
</thead>
<tbody>
<tr>
<td>mailman</td>
<td>10000111010111</td>
</tr>
<tr>
<td>salesman</td>
<td>10000111011000</td>
</tr>
<tr>
<td>bookkeeper</td>
<td>100001110110010</td>
</tr>
<tr>
<td>troubleshooter</td>
<td>1000011101100110</td>
</tr>
<tr>
<td>bouncer</td>
<td>1000011101100111</td>
</tr>
<tr>
<td>technician</td>
<td>100001110110100</td>
</tr>
<tr>
<td>janitor</td>
<td>100001110110101</td>
</tr>
<tr>
<td>saleswoman</td>
<td>100001110110110</td>
</tr>
<tr>
<td>Nike</td>
<td>1011011100100101011100</td>
</tr>
<tr>
<td>Maytag</td>
<td>101101110010101011010</td>
</tr>
<tr>
<td>Generali</td>
<td>10110111001010111101</td>
</tr>
<tr>
<td>Gap</td>
<td>10110111001010111101</td>
</tr>
<tr>
<td>Harley-Davidson</td>
<td>1011011100100101011110</td>
</tr>
<tr>
<td>Enfield</td>
<td>101101110010101111110</td>
</tr>
<tr>
<td>genus</td>
<td>101101110010101111111</td>
</tr>
<tr>
<td>Microsoft</td>
<td>10110111001010110000</td>
</tr>
<tr>
<td>Ventritex</td>
<td>1011011100101011010010</td>
</tr>
<tr>
<td>Tractebel</td>
<td>1011011100101010110110</td>
</tr>
<tr>
<td>Synopsys</td>
<td>101101110010101100111</td>
</tr>
<tr>
<td>WordPerfect</td>
<td>101101110010101101000</td>
</tr>
<tr>
<td>John</td>
<td>101110010000000000</td>
</tr>
<tr>
<td>Consuelo</td>
<td>101110010000000001</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>101110010000000010</td>
</tr>
<tr>
<td>Kenneth</td>
<td>1011100100000001100</td>
</tr>
<tr>
<td>Phillip</td>
<td>10111001000000011010</td>
</tr>
<tr>
<td>WILLIAM</td>
<td>10111001000000011011</td>
</tr>
<tr>
<td>Timothy</td>
<td>10111001000000011110</td>
</tr>
</tbody>
</table>

word cluster features (bit string prefix)

Source: Miller, Guinness, Zamanian (NAACL 2004)
Name Tagging with Word Clusters and Discriminative Training
Challenges in Twitter

2m 2ma 2mar 2mara 2maro 2morrow 2mor 2mora 2moro 2morow 2morr 2morro 2morrow 2moz 2mr
2mro 2mrrw 2mrw 2mw tmmrw tmo tmoro tmorrow tmoz tmr tmro tmrow tmrrow tmrrw tmrw tmrww tmw
tomaro tomarow tomarro tomarrow tomm
tommarow tommarow tommorrow tommorow
tommorrow tommorow tommrrow tommrow tomo tomolo tomoro
tomorow tomarro tomarrow tomorw tomoz tomrw tomz
## Clusters in Twitter NER

<table>
<thead>
<tr>
<th>System</th>
<th>Fin10Dev</th>
<th>Rit11</th>
<th>Fro14</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoNLL</td>
<td>27.3</td>
<td>27.1</td>
<td>29.5</td>
<td>28.0</td>
</tr>
<tr>
<td>+ Brown</td>
<td>38.4</td>
<td>39.4</td>
<td>42.5</td>
<td>40.1</td>
</tr>
<tr>
<td>+ Vector</td>
<td>40.8</td>
<td>40.4</td>
<td>42.9</td>
<td>41.4</td>
</tr>
<tr>
<td>+ Reps</td>
<td>42.4</td>
<td>42.2</td>
<td>46.2</td>
<td>43.6</td>
</tr>
<tr>
<td>Fin10</td>
<td>36.7</td>
<td>29.0</td>
<td>30.4</td>
<td>32.0</td>
</tr>
<tr>
<td>+ Brown</td>
<td>59.9</td>
<td>53.9</td>
<td>56.3</td>
<td>56.7</td>
</tr>
<tr>
<td>+ Vector</td>
<td>61.5</td>
<td>56.4</td>
<td>58.4</td>
<td>58.8</td>
</tr>
<tr>
<td>+ Reps</td>
<td>64.0</td>
<td>58.5</td>
<td>60.2</td>
<td>60.9</td>
</tr>
<tr>
<td>CoNLL+Fin10</td>
<td>44.7</td>
<td>39.9</td>
<td>44.2</td>
<td>42.9</td>
</tr>
<tr>
<td>+ Brown</td>
<td>54.9</td>
<td>52.9</td>
<td>58.5</td>
<td>55.4</td>
</tr>
<tr>
<td>+ Vector</td>
<td>58.9</td>
<td>55.2</td>
<td>59.9</td>
<td>58.0</td>
</tr>
<tr>
<td>+ Reps</td>
<td>58.9</td>
<td>56.4</td>
<td>61.8</td>
<td>59.0</td>
</tr>
<tr>
<td>+ Weights</td>
<td>64.4</td>
<td>59.6</td>
<td>63.3</td>
<td>62.4</td>
</tr>
</tbody>
</table>

Table 5: Impact of our components on Twitter NER performance, as measured by F1, under 3 data scenarios.

Source: Colin Cherry, Hongyu Guo (NAACL 2015)
Clustering in Twitter NER

Brown clusters, for each \( i \) s.t. \( s \leq i < t \):
\[
\{ [y_j, brn(n, x_i), n] \}_{n \in \{2,4,8,12\}}, \\
\{ [y_j, er_{s,t}(i), brn(n, x_i), n] \}_{n \in \{2,4,8,12\}}
\]

Word vectors, for each \( i \) s.t. \( s \leq i < t \):
\[
\{ [y_j, n] = w2v(n, x_i) \}_{n=1}^{300}, \\
\{ [y_j, er_{s,t}(i), n] = w2v(n, x_i) \}_{n=1}^{300}
\]

Table 2: Word representation features in \( \phi(s, t, y_j, x) \).

\( brn(n, x_i) \) maps a word \( x_i \) to the first \( n \) bits of its Brown cluster bit sequence. \( w2v(n, x_i) \) maps \( x_i \) to the \( n^{th} \) component of its word vector, and \( [str] = v \) stands for a real-valued feature with name \( str \) and value \( v \).

Source: Colin Cherry, Hongyu Guo (NAACL 2015)

The Unreasonable Effectiveness of Word Representations for Twitter Named Entity Recognition
Brown Clustering

• The Intuition:
  - similar words appear in similar contexts
  - more precisely: similar words have similar distributions of words to their immediate left and right
Brown Clustering Algorithm

- An agglomerative clustering algorithm:
  - take the top $m$ most frequent words, put each into its own cluster, $c_1, c_2, \ldots, c_m$
  - repeat for $i = (m+1) \ldots |V|$
    - create a new cluster $c_{m+1}$ for the $i$’th most frequent word
    - choose two clusters from $c_1, c_2, \ldots, c_{m+1}$ to be merged, which give the highest Quality based on a training corpus
Brown Clustering Algorithm

- maximize the **Quality** function that score a given partitioning \( C \):

\[
\text{Quality}(C) = \sum_{i}^{n} \log e(w_{i} \mid C(w_{i}))q(C(w_{i}) \mid C(w_{i-1}))
\]

\[
= \sum_{c=1}^{k} \sum_{c'=1}^{k} p(c, c') \log \frac{p(c, c')}{p(c)p(c')} + G
\]

- \( n(c) \) : count of class \( c \) seen in the corpus

- \( n(c, c') \) : counts of \( c' \) seen following \( c \)

\[
p(c, c') = \frac{n(c, c')}{\sum_{c, c'} n(c, c')}
\]

\[
p(c, c') = \frac{n(c)}{\sum_{c} n(c)}
\]

Learn more: Percy Liang’s phd thesis - Semi-Supervised Learning for Natural Language
Brown Clustering Algorithm

- maximize the Quality function that score a given partitioning \( C \):
  \[
  \text{Quality}(C) = \sum_i^n \log e(w_i \| C(w_i)) q(C(w_i) \| C(w_{i-1}))
  \]
  \[
  = \sum_{c=1}^{k} \sum_{c'=1}^{k} p(c,c') \log \frac{p(c,c')}{p(c)p(c')} + G
  \]

  - \( n(c) \): count of class \( c \) seen in the corpus
  - \( n(c,c') \): counts of \( c' \) seen following \( c \)

\[
\begin{align*}
p(c,c') &= \frac{n(c,c')}{\sum_{c,c'} n(c,c')} \\
p(c,c') &= \frac{n(c)}{\sum_c n(c)}
\end{align*}
\]

Learn more: Percy Liang's phd thesis - Semi-Supervised Learning for Natural Language
Brown Clustering

C++ implementation of the Brown word clustering algorithm.

Implementation of the Brown hierarchical word clustering algorithm.

Percy Liang
Release 1.3
2012.07.24

Input: a sequence of words separated by whitespace (see input.txt for an example).
Output: for each word type, its cluster (see output.txt for an example).
In particular, each line is:
<cluster represented as a bit string> <word> <number of times word occurs in input>
Word Vector Representations
(a.k.a. “word embeddings”)

- 4 kinds of vector semantic models

1. Hard clustering (e.g. Brown clustering)
2. Dimensionality Reduction (e.g. SVD, LSA, LDA)
3. Neural Network inspired models (e.g. skip-grams and CBOW in word2vec)
4. Mutual-information weighted word co-occurrence metrics
In Contrast To

represent word meaning by a taxonomy like WordNet

```
from nltk.corpus import wordnet as wn

panda = wn.synset('panda.n.01')
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

**synonym sets (good):**

S: (adj) full, good
S: (adj) estimable, good, honorable, respectable
S: (adj) beneficial, good
S: (adj) good, just, upright
S: (adj) adept, expert, good, practiced, proficient, skillful
S: (adj) dear, good, near
S: (adj) good, right, ripe
...
S: (adv) well, good
S: (adv) thoroughly, soundly, good
S: (n) good, goodness
S: (n) commodity, trade good, good

Alan Ritter • socialmedia-class.org
In Contrast To

represent word meaning by a taxonomy like WordNet

• problems with this discrete representation:
  - missing new words (impossible to keep up-to-date): *wicked, badass, nifty, crack, ace, wizard, genius, ninja*
  - requires human labor to create and adapt
  - hard to compute accurate word similarity
  - and apparently not enough to handle social media data!
Distributional Intuition

• From context words, human can guess a word’s meaning:

A bottle of **tesgüino** is on the table  
Everybody likes **tesgüino**  
*Tesgüino* makes you drunk  
We make **tesgüino** out of corn.

“*You shall know a word by the company it keeps*”
— J. R. Firth 1957
Distributional Intuition

• From context words, human can guess a word’s meaning:

A bottle of *tesgüino* is on the table
Everybody likes *tesgüino*
*Tesgüino* makes you drunk
We make *tesgüino* out of corn.

• similar words = similar contexts = similar vectors

• word meaning is represented by a vector of numbers
Simple Co-occurrence Vectors

• Option #1: word-document co-occurrence counts

<table>
<thead>
<tr>
<th></th>
<th>As You Like It</th>
<th>Twelfth Night</th>
<th>Julius Caesar</th>
<th>Henry V</th>
</tr>
</thead>
<tbody>
<tr>
<td>battle</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>soldier</td>
<td>2</td>
<td>2</td>
<td>12</td>
<td>36</td>
</tr>
<tr>
<td>fool</td>
<td>37</td>
<td>58</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>clown</td>
<td>6</td>
<td>117</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

this will give general topics (e.g. sports terms will have similar entries), leading to **Latent Semantic Analysis**
Simple Co-occurrence Vectors

- Option #2: use a sliding window over a big corpus of text and count word co-occurrences:

  example corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

  this captures both syntactic (POS) and semantic information
Simple Co-occurrence Vectors

- Problems with this representation of raw counts:
  - increase in size with vocabulary
  - high dimensionality and very sparse!
  - not a great measure of association between words:
    “the” and “of” are very frequent, but maybe not the most discriminative
Lower Dimensional Vectors

- **The Idea**: use dense vectors to store “most” of the important information in a fixed, small number of dimensions

- usually around 25 ~1000 dimensions
Lower Dimensional Vectors

• Word meaning is represented as a dense vector

\[
\text{“linguistic”} = \begin{pmatrix}
0.286 \\
0.792 \\
-0.177 \\
-0.107 \\
0.109 \\
-0.542 \\
0.349 \\
0.271 \\
\end{pmatrix}
\]

How to reduce the dimensionality?
(2) Matrix Factorization

- Singular Value Decomposition (SVD)

\[ X = U S V^T \]

\[ \hat{X} = \hat{U} \hat{S} \hat{V}^T \]

\( \hat{X} \) is the best rank \( k \) approximation to \( X \), in terms of least squares.
SVD Word Vectors

example corpus:
- I like deep learning.
- I like NLP.
- I enjoy flying.

```python
import numpy as np
la = np.linalg
words = ["I", "like", "enjoy", "deep", "learning", "NLP", "flying", "."]
X = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,0,1],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],
              [0,0,1,0,0,0,0,1],
              [0,0,0,0,1,1,1,0]])
U, s, Vh = la.svd(X, full_matrices=False)
```
SVD Word Vectors

- plot first 2 columns of U corresponding to the 2 biggest singular values:

```python
for i in xrange(len(words)):
    plt.text(U[i,0], U[i,1], words[i])
```
Some Hacks

• Problem: function words ("the", "he", "has") are too frequent  ⟷  syntax has too much impact.
  
  - fixes: cap the counts, or ignore them all

• ramped windows that count closer words more

• etc …

Source: Rohde et al. (2005)

An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence
Interesting Syntactic Patterns

Figure 10: Multidimensional scaling of three verb semantic classes.

Figure 11: Multidimensional scaling of present, past, progressive, and past participle forms for eight verb families.

Source: Rohde et al. (2005)
Interesting Semantic Patterns

Source: Rohde et al. (2005)

An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence
SVD Word Vectors

• Still some problems:
  
  - computational cost scales quadratically for $m \times n$ matrix — $O(mn^2)$ when $n<m$
  
  - hard to use large corpus (and vocabulary)
  
  - hard to incorporate new words or documents
(3) Neural Word Embeddings

- **The Idea**: directly learn low-dimensional word vectors

- … can go back to 1980s:
  - Learning Representations by Back-Propagating Errors (Rumelhart et al., 1986)
  - A Neural Probabilistic Language Model (Bengio et al., 2003)
  - NLP from Scratch (Collobert & Weston, 2008)
  - **Word2vec** (Mikolov et al. 2013)
Neural Word Embeddings

• The Basic Idea:
  - We define a model that aims to predict a word given its context words (word vectors), which has a loss function, e.g. $J = 1 - P(\text{context} \mid w_i)$
  - We look at many positions of $t$ in a big text corpus,
  - and keep adjusting the word vectors to minimize this loss.
Word2vec

- simple and efficient

Source: Mikolov et al. (NIPS 2013)
Distributed Representations of Words and Phrases and their Compositionality
Word2vec

- Skip-gram — predicts surrounding “outside” words given the “center” word
Word2vec

- Skip-gram — predicts surrounding “outside” words given the “center” word

Figure 16.5: The skip-gram model viewed as a network (Mikolov et al. 2013, Mikolov et al. 2013a).
Input Layer

- “one-hot” word vectors
  - a vector of dimension $|V|$ (size of vocabulary)
  - all “0”s expect a single “1” in the vector
  - different positions of that “1” represent different words
Hidden (Projection) Layer

- A simple look up — the rows of this weight matrix are actually “input” word vectors
Hidden (Projection) Layer

- A simple look up — the rows of this weight matrix are actually "input" word vectors

\[
\begin{bmatrix}
0 & 0 & 0 & 1 & 0
\end{bmatrix} \times \begin{bmatrix}
17 & 24 & 1 \\
23 & 5 & 7 \\
4 & 6 & 13 \\
10 & 12 & 19 \\
11 & 18 & 25
\end{bmatrix} = \begin{bmatrix}
10 & 12 & 19
\end{bmatrix}
\]
Output Layer

• predicts surrounding “outside” (context) words given the “center” word \( \rightarrow \) A classification problem!

• Softmax Regression = Multi-class Logistic Regression
Softmax Function

• Softmax function is a generalization of logistic function

\[ \text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \]

- Exponentiate to make positive
- Normalized to give probability
Softmax Function

- Softmax function is a generalization of logistic function

$$softmax(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$e^x$</th>
<th>Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.85</td>
<td>0.058</td>
<td>0.016</td>
</tr>
<tr>
<td>0.86</td>
<td>2.36</td>
<td>0.631</td>
</tr>
<tr>
<td>0.28</td>
<td>1.32</td>
<td>0.353</td>
</tr>
</tbody>
</table>
Output Layer

- Intuition

Word vector for “ants”

Output weights for “car”

\[
\frac{e^x}{\sum e^x}
\]

= Probability that “car” shows up near “ants”

Source: Chris McCormick
Output Layer

- Objective function: maximize the log probability of any “outside” (context) word given the “center” word

\[ J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log p(w_{t+j} | w_t) \]
Output Layer

• predicts surrounding “outside” (context) words given the “center” word

\[ p(o|c) = \frac{\exp(u_O^T v_c)}{\sum_{w=1}^{W} \exp(u_w^T v_c)} \]

• so, every word has two vectors!
Gradient Descent

- Cost/Objective function:

\[
J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log p(w_{t+j} | w_t)
\]

- For a “center” word and an “outside” word:

\[
\log p(o|c) = \log \frac{\exp \left( u_o^T v_c \right)}{\sum_{w=1}^{W} \exp \left( u_w^T v_c \right)}
\]
Gradient Descent

• Basics:

\[ \frac{\partial x^T a}{\partial x} = \frac{\partial a^T x}{\partial x} = a \]

\[ \frac{\partial e^x}{\partial x} = e^x \]

\[ \frac{\partial \log x}{\partial x} = \frac{1}{x} \]

• Chain Rule:

\[ \frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x} = \frac{\partial f(g)}{\partial g} \frac{\partial g(x)}{\partial x} \]
Word2vec

• Word2vec is not a single algorithm, but a toolkit
  - which contains two distinct algorithms (Skip-gram & CBOW), two training methods (negative sampling & hierarchical softmax)

• Word2vec is not deep learning, but neural-inspired
  - only one hidden layer followed by softmax, no non-linear activation function

Learn more: Omer Levy's answer on Quora https://www.quora.com/How-does-word2vec-work-Can-someone-walk-through-a-specific-example
Relation between Skip-gram and SVD

• Levy and Goldberg (2014) show that skip-gram is factorizing (a shifted version of ) the traditional word-context PMI matrix:

\[ \text{Opt}(\vec{w} \cdot \vec{c}) = \text{PMI}(w, c) - \log k \]

• So does SVD!

Source: Omer Levy and Yoav Goldberg (NIPS 2014) 
Neural Word Embedding as Implicit Matrix Factorization
Visualization

Male-Female

Verb tense

Country-Capital

Source: tensorflow.org
Figure 1: A 2-dimensional projection of the latent semantic space captured by our algorithm. Notice the semantic trajectory of the word *gay* transitioning meaning in the space.

Source: Kulkarni et al. (WWW 2015)
Statistically Significant Detection of Linguistic Change
Thank You!