NLP
Deep Learning

Word Embeddings (1/2)
What Is the Feature Vector $x$?

- Typically a vector representation of a single character or word
- Often reflects the *context* in which that word is found
- Could just do counts, but that leads to sparse vectors
- Commonly used techniques: *word2vec* or *GloVe* word embeddings
Embeddings Are Magic, Part 1

\[ \text{vector('king')} - \text{vector('man')} + \text{vector('woman')} \approx \text{vector('queen')} \]
Embeddings Are Magic, Part 2

GloVe vectors for comparative and superlative adjectives

More Examples

Examples from Richard Socher
word2vec

• Popular group of models for word embeddings
• https://code.google.com/p/word2vec/
  – includes the models and pre-trained embeddings
  – Pre-trained is good, because training takes a lot of data
• Gensim: Python library that works with word2vec
  – https://radimrehurek.com/gensim/
• Most notable models:
  – Skipgrams and CBOW
Skip-grams

• Predict each neighboring word
  – in a context window of $2C$ words
  – from the current word.

• So for $C=2$, we are given word $w_t$ and predicting these 4 words:
  $$[w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}]$$
Skip-grams learn 2 embeddings for each \( w \) input embedding \( v \), in the input matrix \( W \):

- Column \( i \) of the input matrix \( W \) is the \( 1 \times d \) embedding \( v_i \) for word \( i \) in the vocabulary.

Output embedding \( v' \), in output matrix \( W' \):

- Row \( i \) of the output matrix \( W' \) is a \( d \times 1 \) vector embedding \( v'_i \) for word \( i \) in the vocabulary.
Setup

• Walking through corpus pointing at word $w(t)$, whose index in the vocabulary is $j$, so we’ll call it $w_j$ $(1 < j < |V|)$.

• Let’s predict $w(t+1)$, whose index in the vocabulary is $k$ $(1 < k < |V|)$. Hence our task is to compute $P(w_k|w_j)$. 
One-hot vectors

• A vector of length $|V|$
• Example:
  - $[0,0,0,0,1,0,0,0,0,0,0,...]$
CBOW and skipgram (Mikolov 2013)

\[ w_i = \sum \{ w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2} \} \]
Skip-gram

Input layer
1-hot input vector

Projection layer
embedding for $w_t$

Output layer
probabilities of context words

$W_{\text{d} \times |V|}$

$W'_{\text{d} \times |V|}$

$y_1$
$y_2$
$\vdots$
$y_k$
$w_{t-1}$
$y_{|V|}$
$\vdots$
$y_{|V|}$
$w_{t+1}$

$y_1$
$y_2$
$\vdots$
$y_k$
$w_{t-1}$
$y_{|V|}$
$\vdots$
$y_{|V|}$
$w_{t+1}$

Slide courtesy of Jurafsky & Martin
Skip-gram

Input layer
1-hot input vector

Projection layer
embedding for $w_t$

Output layer
probabilities of context words

$W$ $|V| \times d$

$h = v_j$

$W'_{d \times |V|}$

$y_1$
$y_2$
$\vdots$
$y_k$
$y_{|V|}$

$W'_{d \times |V|}$

$y_1$
$y_2$
$\vdots$
$y_k$
$y_{|V|}$

$1 \times |V|$
$1 \times d$

$w_t$
$x_1$
$x_2$
$\vdots$
$x_j$
$\vdots$
$x_{|V|}$

$w_{t-1}$
$w_{t+1}$

Slide courtesy of Jurafsky & Martin
Notes

• Sparse vs. dense vectors
  – 100,000 dimensions vs. 300 dimensions
  – <10 non-zero dimensions vs. 300 non-zero dimensions

• Dense vectors
  – Semantic similarity (cf. LSA, Brown clusters)
Similarity Computation

• Similarity is computed using the dot product of the two vectors

• To convert a similarity to a probability, use softmax

\[ p(w_k | w_j) = \frac{\exp(c_k v_j)}{\sum_i \exp(c_i v_j)} \]

• In practice, use negative sampling
  – too many words in the denominator
  – the denominator is only computed for a few words
Softmax

![Diagram showing Softmax output for different inputs. The top chart represents the output for input values 1 to 4, with values ranging from 0 to 1. The bottom chart shows a similar representation for different values, with output values ranging from 0 to 0.6.](image)
Evaluating Embeddings

- Nearest Neighbors
- Analogies
  - (A:B)::(C:?)
## Similarity Data Sets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Word pairs</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>RG</td>
<td>65</td>
<td>Rubenstein and Goodenough (1965)</td>
</tr>
<tr>
<td>MC</td>
<td>30</td>
<td>Miller and Charles (1991)</td>
</tr>
<tr>
<td>WS-353</td>
<td>353</td>
<td>Finkelstein et al. (2002)</td>
</tr>
<tr>
<td>MTurk-287</td>
<td>287</td>
<td>Radinsky et al. (2011)</td>
</tr>
<tr>
<td>MTurk-771</td>
<td>771</td>
<td>Halawi et al. (2012)</td>
</tr>
<tr>
<td>MEN</td>
<td>3000</td>
<td>Bruni et al. (2012)</td>
</tr>
<tr>
<td>RW</td>
<td>2034</td>
<td>Luong et al. (2013)</td>
</tr>
<tr>
<td>Verb</td>
<td>144</td>
<td>Baker et al. (2014)</td>
</tr>
<tr>
<td>SimLex</td>
<td>999</td>
<td>Hill et al. (2014)</td>
</tr>
</tbody>
</table>

[Table from Faruqui et al. 2016]
<table>
<thead>
<tr>
<th>Type of relationship</th>
<th>Word Pair 1</th>
<th>Word Pair 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common capital city</td>
<td>Athens</td>
<td>Greece</td>
</tr>
<tr>
<td>All capital cities</td>
<td>Astana</td>
<td>Kazakhstan</td>
</tr>
<tr>
<td>Currency</td>
<td>Angola</td>
<td>kwanza</td>
</tr>
<tr>
<td>City-in-state</td>
<td>Chicago</td>
<td>Illinois</td>
</tr>
<tr>
<td>Man-Woman</td>
<td>brother</td>
<td>sister</td>
</tr>
<tr>
<td>Adjective to adverb</td>
<td>apparent</td>
<td>apparently</td>
</tr>
<tr>
<td>Opposite</td>
<td>possibly</td>
<td>impossibly</td>
</tr>
<tr>
<td>Comparative</td>
<td>great</td>
<td>greater</td>
</tr>
<tr>
<td>Superlative</td>
<td>easy</td>
<td>easiest</td>
</tr>
<tr>
<td>Present Participle</td>
<td>think</td>
<td>thinking</td>
</tr>
<tr>
<td>Nationality adjective</td>
<td>Switzerland</td>
<td>Swiss</td>
</tr>
<tr>
<td>Past tense</td>
<td>walking</td>
<td>Cambodia</td>
</tr>
<tr>
<td>Plural nouns</td>
<td>mouse</td>
<td>read</td>
</tr>
<tr>
<td>Plural verbs</td>
<td>work</td>
<td>swimming</td>
</tr>
</tbody>
</table>

[Mikolov et al. 2013]
<table>
<thead>
<tr>
<th></th>
<th>Input Vector</th>
<th>Output Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td><img src="image1" alt="Apple Input Vector" /></td>
<td><img src="image2" alt="Apple Output Vector" /></td>
</tr>
<tr>
<td>drink</td>
<td><img src="image3" alt="Drink Input Vector" /></td>
<td><img src="image4" alt="Drink Output Vector" /></td>
</tr>
<tr>
<td>cot</td>
<td><img src="image5" alt="Cot Input Vector" /></td>
<td><img src="image6" alt="Cot Output Vector" /></td>
</tr>
<tr>
<td>juice</td>
<td><img src="image7" alt="Juice Input Vector" /></td>
<td><img src="image8" alt="Juice Output Vector" /></td>
</tr>
<tr>
<td>milk</td>
<td><img src="image9" alt="Milk Input Vector" /></td>
<td><img src="image10" alt="Milk Output Vector" /></td>
</tr>
<tr>
<td>orange</td>
<td><img src="image11" alt="Orange Input Vector" /></td>
<td><img src="image12" alt="Orange Output Vector" /></td>
</tr>
<tr>
<td>rice</td>
<td><img src="image13" alt="Rice Input Vector" /></td>
<td><img src="image14" alt="Rice Output Vector" /></td>
</tr>
<tr>
<td>water</td>
<td><img src="image15" alt="Water Input Vector" /></td>
<td><img src="image16" alt="Water Output Vector" /></td>
</tr>
</tbody>
</table>
Notes

• Word embeddings perform matrix factorization of the co-occurrence matrix
• Word2vec is a simple feed-forward neural network
• Training is done using backpropagation using SGD
• Negative sampling for training
NLP