Neural language models

- Skip-grams
- Continuous Bag of Words (CBOW)
- More details can be found at https://cs224d.stanford.edu/lecture_notes/notes1.pdf

Prediction-based models: An alternative way to get dense vectors

- Skip-gram (Mikolov et al. 2013a), CBOW (Mikolov et al. 2013b)
- Learn embeddings as part of the process of word prediction
- Train a neural network to predict neighboring words
- Advantages:
  - Fast, easy to train (much faster than SVD)
  - Available online in the word2vec package
  - Including sets of pretrained embeddings!

Word2vec

- Popular embedding method
- Very fast to train
- Code available on the web
- Idea: predict rather than count

Word2vec

- Given a sentence:
  - lemon, a tablespoon of apricot jam, a pinch ...
- Instead of counting how often each word occurs near “apricot”
- Train a classifier on a binary prediction task:
  - Is word w likely to show up near “apricot”?
- We don’t actually care about this task
  - But we’ll take the learned weights (will be discussed later) as the word embeddings

Brilliant insight: Use running text as implicitly supervised training data!

- A word near apricot
  - Acts as gold ‘correct answer’ to the question
  - “Is word w likely to show up near apricot?”
  - No need for hand-labeled supervision
- The idea comes from neural language modeling
  - Bengio et al. (2003)
  - Collobert et al. (2011)
Word2Vec: **Skip-Gram Task**

- Now we have positive samples.
- Where do the "negative samples" come from?

Word2Vec: **Skip-Gram Task**

- Word2vec provides a variety of options. Let's do
  - "skip-gram with negative sampling" (SGNS)

---

**Skip-gram algorithm**

1. Treat the target word and a neighboring context word as positive examples.
2. Randomly sample other words in the lexicon to get negative samples
3. Use logistic regression (will discuss formulation later) to train a classifier to distinguish those two cases
4. Use the weights as the embeddings

---

**Skip-gram Training Data**

- Training sentence:
  ... lemon, a tablespoon of apricot jam a pinch ...
  \[ c_1 \quad c_2 \quad \text{target} \quad c_3 \quad c_4 \]
  
  Assume context words are those in +/- 2 word window

---

**Skip-gram Goal**

- Given a tuple \((t,c)\) = target, context
  - \((apricot, jam)\)
  - \((apricot, aardvark)\)
- Return probability that \(c\) is a real context word (or not):
  - \(P(+) | t,c\) -> positive
  - \(P(-) | t,c\) = 1\(\rightarrow P(+) | t,c\) -> negative
How to compute $p(\omega | t, c)$?

**Intuition:**
- Words are likely to appear near similar words
- Model similarity with dot product!
- Similarity $(t, c) \propto t \cdot c$

**Problem:**
- Dot product is not a probability!
  - (Neither is cosine)

Turning dot product into a probability

**The sigmoid lies between 0 and 1:**

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

For all the context words:

- Assume all context words are independent

- Training sentence:
  - ... lemon, a tablespoon of apricot jam a pinch ...
  - $c_1 c_2 t c_3 c_4$

- Training data: input/output pairs centering on **apricot**
- Assume a +/- 2 word window

Turning dot product into a probability

$$P(+ | t, c) = \frac{1}{1 + e^{-t \cdot c}}$$

$$P(- | t, c) = 1 - P(+ | t, c)$$

$$= \frac{e^{t \cdot c}}{1 + e^{t \cdot c}}$$

Skip-gram Training Data

- Training sentence:
  - ... lemon, a tablespoon of apricot jam a pinch ...
  - $c_1 c_2 t c_3 c_4$

- Training data: input/output pairs centering on **apricot**
- Assume a +/- 2 word window

Skip-gram Training Data

- Training sentence:
  - ... lemon, a tablespoon of apricot jam a pinch ...
  - $c_1 c_2 t c_3 c_4$

- Training data: input/output pairs centering on **apricot**
- Assume a +/- 2 word window

For the context words $c_1, c_2, c_3, c_4$

$$P(+ | t, c_k) = \prod_{i=1}^{k} \frac{1}{1 + e^{-t \cdot c_i}}$$

$$\log P(+ | t, c_k) = \sum_{i=1}^{k} \log \frac{1}{1 + e^{-t \cdot c_i}}$$

positive examples $+$

<table>
<thead>
<tr>
<th>t</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>apricot</td>
<td>tablespoon</td>
</tr>
</tbody>
</table>
Skip-gram Training Data

- Training sentence: ... lemon, a tablespoon of apricot jam, a pinch ...

positive examples +
- apricot
tablespoon
apricot preserves
apricot or

• For each positive example, we’ll create k negative examples.
• Using noise words
• Any random word that isn’t t

Choosing noise words

- Could pick w according to its unigram frequency P(w)
- More common to chosen then according to p_u(w)

\[ P_u(w) = \frac{\text{count}(w)^\alpha}{\sum_{w'} \text{count}(w')} \]

\( \alpha = 0.75 \) works well because it gives rare noise words slightly higher probability.
- To show this, imagine two events \( p(a) = .99 \) and \( p(b) = .01 \):

\[ P_u(a) = \frac{99^7}{99^7 + 99} = .97 \]
\[ P_u(b) = \frac{0.01^7}{99^7 + 0.01} = .03 \]

Learning the classifier (W and C)

- Iterative process on training data
- Then adjust the word weights to
  - make the positive pairs more likely
  - and the negative pairs less likely

Setup

- Let’s represent words as vectors of some length (say 300), randomly initialized.
- So we start with 300 * V random parameters
- Over the entire training set, we’d like to adjust those word vectors such that we
  - Maximize the similarity of the target word, context word pairs \((t, c)\) drawn from the positive data
  - Minimize the similarity of the \((t, c)\) pairs drawn from the negative data
Formally

- We want to maximize the following objective
  \[ \sum_{(t,c) \in +} \log P(t|c) + \sum_{(t,c) \in -} \log P(-t|c) \]
- Maximize the + label for the pairs from the positive training data, and the - label for the pairs sample from the negative data.

Focusing on one target word t:

\[
L(\theta) = \log P(+|r,c) + \sum_{i=1}^{k} \log P(-|t,n) = \log \sigma(c \cdot t) + \sum_{i=1}^{k} \log \sigma(-n_i \cdot t)
\]

Train using gradient descent (not required)

- Idea: gradually changing W and C
- Finally learns two separate embedding matrices W and C
- Can use W and throw away C, or merge them

Summary: How to learn skip-gram embeddings

- Start with V random 300-dimensional vectors as initial embeddings
- Use logistic regression, the second most basic classifier used in machine learning after naive bayes
- Take a corpus and take pairs of words that co-occur as positive examples
- Take pairs of words that don’t co-occur as negative examples
- Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
- Throw away the classifier code and keep the embeddings.

(Dense) Word embeddings you can download!

- **Word2vec** [Mikolov et al.]
  https://code.google.com/archive/p/word2vec/
- **Fasttext** http://www.fasttext.cc/
- **Glove** [Pennington, Socher, Manning]
  http://nlp.stanford.edu/projects/glove/
Evaluating embeddings

- Compare to human scores on word similarity-type tasks:
  - WordSim-353 (Finkelstein et al., 2002)
  - Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)

- TOEFL dataset:
  - Levied is closest in meaning to:
    - imposed, believed, requested, correlated

Properties of embeddings

- Nearest words to some embeddings (Mikolov et al. 2013)

<table>
<thead>
<tr>
<th>Target</th>
<th>Redmond</th>
<th>Hogwarts</th>
<th>president</th>
<th>velvet</th>
<th>man</th>
<th>woman</th>
<th>London</th>
<th>queen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunnydale</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evernight</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dumbledore</td>
<td>Dumbledore</td>
<td>Halfblood</td>
<td>Halfblood</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Malfoy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>halfblood</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Similarity depends on window size $C$

- $C = 2$: The nearest words to Hogwarts:
  - Sunnydale
  - Evernight
- $C = 5$: The nearest words to Hogwarts:
  - Dumbledore
  - Malfoy
  - halfblood

Analogy: Embeddings capture relational meaning!

$$\text{vector('king')} - \text{vector('man')} + \text{vector('woman')} \approx \text{vector('queen')}$$

$$\text{vector('Paris')} - \text{vector('France')} + \text{vector('Italy')} \approx \text{vector('Rome')}$$
Embeddings can help study word history!

*Train embeddings on old books to study changes in word meaning!!*

**Diachronic word embeddings for studying language change!**

Visualization changes

*Project 300 dimensions down into 2*

The evolution of sentiment words

*Negative words change faster than positive words*

Embeddings reflect cultural bias

*Ask “Paris : France :: Tokyo : x”*
  *x = Japan*
*Ask “father : doctor :: mother : x”*
  *x = nurse*
*Ask “man : computer programmer :: woman : x”*
  *x = homemaker*

Embeddings and bias
Embeddings reflect cultural bias


- Implicit Association test (Greenwald et al 1998):
  - How associated are concepts (flowers, insects) & attributes (pleasantness, unpleasantness)?
  - Studied by measuring timing latencies for categorization.
- Psychological findings on US participants:
  - African-American names are associated with unpleasant words (more than European-American names)
  - Male names associated more with math, female names with arts
  - Old people’s names with unpleasant words, young people with pleasant words.

Embeddings as a window onto history

The cosine similarity of embeddings for decade X for occupations or adjectives (e.g., teacher or smart) to male vs female names
- Find its correlation with the actual percentage of women teachers in decade X

Embeddings reflect ethnic stereotypes over time

- Princeton trilogy experiments
- Attitudes toward ethnic groups (1933, 1951, 1969) scores for adjectives
  - Industrious, superstitious, nationalistic, etc.
- Cosine of Chinese name embeddings with those adjective embeddings correlates with human ratings.

History of biased framings of women

- Embeddings for competence adjectives are biased toward men
  - Smart, wise, brilliant, intelligent, resourceful, thoughtful, logical, etc.
- This bias is slowly decreasing

Change in linguistic framing 1910-1990

Change in association of Chinese names with adjectives framed as ‘othering’ (barbaric, monstrous, biased)
Changes in framing: adjectives associated with Chinese

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Irresponsible</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emirous</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barbaric</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aggressive</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transparent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monstrous</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cruel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greedy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bizarre</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Irresponsible</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emirous</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barbaric</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aggressive</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transparent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monstrous</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cruel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greedy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bizarre</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Directions

- Debiasing algorithms for embeddings
- Use embeddings as a historical tool to study bias