About “Embeddings”
What’s an “embedding”?  

• A neural network embodies a function $f: X \rightarrow X'$

• $X$ is the input to the net and $X'$ is the set of activations of some layer of the net.

• Colloquially, we refer to $X'$ as an “embedding” of the input to the net.

• Some even call any vector of real numbers an “embedding” (e.g. calling a hand-made one-hot vector an “embedding”)

• Ideally, want an embeddings to have meaningful “groupings”
Common design goals

• We want to provide task-relevant similarity judgements
• We want the ability to compare things we’ve never trained on
Training a Dedicated Embedding Network

Example: VoiceID
Train an embedding net
Move things from the same group closer

\[ \text{Loss} \propto D_f(x_1, x_2) \]
Push things from different groups apart

\[ \text{Loss } \propto M - D_f(x_1, x_2) \]
Train an embedding net
Enroll a voice

△ Enrolled embedding, speaker A
Enroll more voices

△ Enrolled embedding, speaker A
△ Enrolled embedding, speaker B
Find centroids
Use nearest-neighbor search to pick a group

Query embedding, speaker A
Finding neighbors: Cosine similarity

• Often used with embeddings
• A kind of normalized dot product
• Higher values = more similar

\[ S_C(A, B) := \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}, \]
What if we remove the normalization?

- Often used with embeddings
- A kind of normalized dot product
- Higher values = more similar
- How would scale affect things?

\[ S(A, B) = \sum_i A_i B_i \]
Making an embedding for words

- With voices, we know which 2 things should go together.
- How do I decide this for words?
- Is there a way to make training go faster than to train an entire neural network?
Example: GloVE

The big ideas: distance and co-occurrence

- Words that are occur together in text are related (big assumption).

- Measuring co-occurrence of words tells us how related they are.

- Words beyonds some cutoff distance aren’t co-occurring.
  - “Bryan is, in no way even remotely close to a billionaire”
Table 1: Co-occurrence probabilities for target words *ice* and *steam* with selected context words from a 6 billion token corpus. Only in the ratio does noise from non-discriminative words like *water* and *fashion* cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam.

<table>
<thead>
<tr>
<th>Probability and Ratio</th>
<th>$k = solid$</th>
<th>$k = gas$</th>
<th>$k = water$</th>
<th>$k = fashion$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(k</td>
<td>ice)$</td>
<td>$1.9 \times 10^{-4}$</td>
<td>$6.6 \times 10^{-5}$</td>
<td>$3.0 \times 10^{-3}$</td>
</tr>
<tr>
<td>$P(k</td>
<td>steam)$</td>
<td>$2.2 \times 10^{-5}$</td>
<td>$7.8 \times 10^{-4}$</td>
<td>$2.2 \times 10^{-3}$</td>
</tr>
<tr>
<td>$P(k</td>
<td>ice) / P(k</td>
<td>steam)$</td>
<td>8.9</td>
<td>$8.5 \times 10^{-2}$</td>
</tr>
</tbody>
</table>

\[
P_{ik} = P(k | i) = \frac{X_{ik}}{X_i}
\]

$X_{ik}$ is the co-occurrence count between $i$ and $k$

\[
X_i = \sum_j X_{ij}
\]

Sum over all words in dictionary
Lets learn a function that...

- Captures the ratio of word probabilities:

\[ F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \]

- Encodes words as vectors in a vector space (or two):

\[ e.g. \ w_x = [0, 1, -3, 4] \]

  - (The exact values in these vectors is part of what we’ll learn)

- Makes addition and subtraction of word vectors meaningful

That ~ is meaningful, by the way. It means that the context word’s vector is drawn from a different vector space (of the same dimensionality as the other two words vector)
Defining a function to enforce additivity

• If we make $F$ the exponential function (which is the inverse of taking the natural log), then we can do this…

$$F \left( (w_i - w_j)^T \tilde{w}_k \right) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} = \frac{P_{ik}}{P_{jk}}$$

The difference between the word vectors relates to the ratio of their probabilities of co-occurrence
Make inner products relate to probabilities

- Define making the inner product between a word and its context word relate to the LOG of the probability of seeing that word in the context.

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

We’ll learn these word vectors

This is in the data
So what do we need to learn, exactly?

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

- If we create a bias term related to each word i and k, they can stand in for \( \log(X_i) \), resulting in this…
  \[ w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik}) \]

- We’re going to learn everything to the left of the = in equation 7.
So what do we use as our word vectors?

• We end up learning 2 vectors for every word.
• Just sum them up and use those as the embeddings

\[ E = W + \tilde{W} \]
The word analogy task


<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, Astana</td>
<td>Greece, Kwanza</td>
</tr>
<tr>
<td>All capital cities</td>
<td>Angola, Chicago</td>
<td>Illinois, Sister</td>
</tr>
<tr>
<td>Currency</td>
<td>brother</td>
<td>Oslo, Harare</td>
</tr>
<tr>
<td>City-in-state</td>
<td></td>
<td>Iran, Stockton</td>
</tr>
<tr>
<td>Man-Woman</td>
<td></td>
<td>Norway, Zimbabwe</td>
</tr>
<tr>
<td>Adjective to adverb</td>
<td>apparent, possibly</td>
<td>apparently, rapidly</td>
</tr>
<tr>
<td>Opposite</td>
<td>great, easy, think</td>
<td>ethically, tough</td>
</tr>
<tr>
<td>Comparative</td>
<td>Switzerland, walking</td>
<td>easiest, thinking</td>
</tr>
<tr>
<td>Superlative</td>
<td>mouse, work</td>
<td>lucky, read</td>
</tr>
<tr>
<td>Present Participle</td>
<td></td>
<td>Cambodia, swimming</td>
</tr>
<tr>
<td>Nationality adjective</td>
<td></td>
<td>Swiss, swimming</td>
</tr>
<tr>
<td>Past tense</td>
<td></td>
<td>Dollar, speak</td>
</tr>
<tr>
<td>Plural nouns</td>
<td></td>
<td>Dollars, speaks</td>
</tr>
<tr>
<td>Plural verbs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We evaluate the overall accuracy for all question types, and for each question type separately (semantic, syntactic). Question is assumed to be correctly answered only if the closest word to the vector computed using the above method is exactly the same as the correct word in the question; synonyms are thus counted as mistakes. This also means that reaching 100% accuracy is likely to be impossible, as the current models do not have any input information about word morphology.
How big should we make embeddings?

**Word analogies.** The word analogy task consists of questions like, “a is to b as c is to __?”
What is the right context window size?

**Word analogies.** The word analogy task consists of questions like, “a is to b as c is to _?”

*experiment used 300 dimension word vectors.*

![Graph showing accuracy over window size for different tasks: Semantic, Syntactic, and Overall.](image-url)
How does Glove compare to Word2Vec?

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SG</td>
<td>300</td>
<td>1B</td>
<td>61</td>
<td>61</td>
<td>61</td>
</tr>
<tr>
<td>CBOW</td>
<td>300</td>
<td>1.6B</td>
<td>16.1</td>
<td>52.6</td>
<td>36.1</td>
</tr>
<tr>
<td>vLBL</td>
<td>300</td>
<td>1.5B</td>
<td>54.2</td>
<td>64.8</td>
<td>60.0</td>
</tr>
<tr>
<td>ivLBL</td>
<td>300</td>
<td>1.5B</td>
<td>65.2</td>
<td>63.0</td>
<td>64.0</td>
</tr>
<tr>
<td>GloVe</td>
<td>300</td>
<td>1.6B</td>
<td>80.8</td>
<td>61.5</td>
<td>70.3</td>
</tr>
</tbody>
</table>

**Word analogies.** The word analogy task consists of questions like, "a is to b as c is to __?"
The dataset contains 19,544 such questions, di-
What do Transformers use?
NOT GLOVE!

NOT WORD2VEC!

They learn their own!