INDEPENDENT WORDS, INDEPENDENT VECTORS

* A (word) type is an element of a vocabulary. A (word) token is an instance of a type in context.

$$v_{Tea} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix} \qquad v_{coffee} = \begin{bmatrix} \vdots \\ 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

* If we use one hot vectors, we achive to show different words are different, but we encode no meaningful notion of similarity or other relationship. This is because if we take dot product as notion of similarity

$$v_{Tea}^{T} v_{coffee} = 0 \quad \Rightarrow \text{ all words are equally dissimilar from each other.}$$

REPRESENTING WORDS BY THEIR CONTEXT

Distributional Semantics : A word's meaning is given by the words that frequently appear close-by

When a word appears in a text, its context is the set of words that appear nearby (within a fixed sized window)

WORD VECTORS

$$banking = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

$\Rightarrow$ Dense vector for each word.

NOTE: Word vectors are also called word embeddings or (neural) word representations

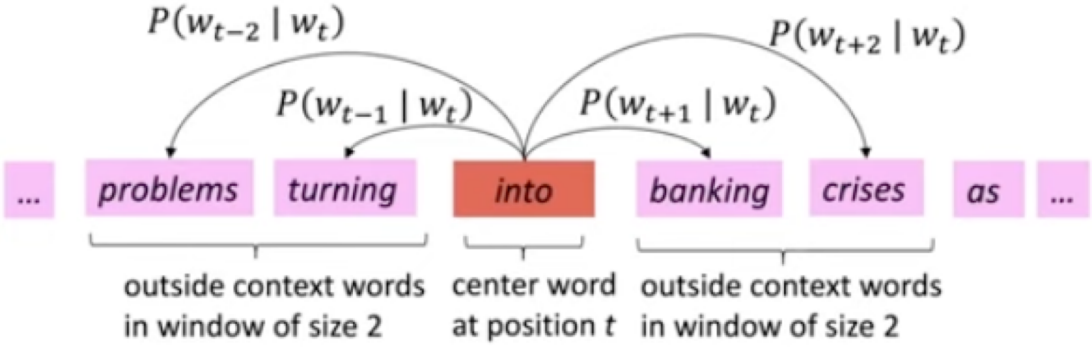
Word2 Vec

* Framework for learning word vectors

1. Have a large corpus of text
2. Every word in a fixed vocab represented by a vector
3. Go through each position t in the text, which has a center word c and context ("outside") words o
4. Use similarity of the word vectors for c and o to calculate the probability of o given c
5. Keep adjusting the word vectors to maximize this probability

* The word2vec model represents each word in a fixed vocab as a low-dimensional (much smaller than vocab size) vectors

Example windows and process for computing $P(w_{t+j} \mid w_t)$



OBJECTIVE FUNCTION

$t = 1, \dots, T$: positions

m : window size

w_j : center word

probability of a word occurring in the context

objective function = cost function = loss function

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

minimizes objective function

minimizing objective func → maximizing predictive accuracy

How to calculate $\log P(w_{t+j} \mid w_t; \theta)$?

* we will use 2 vectors per word w:
  1. v_w ⇒ when w is a center word
  2. u_w ⇒ when w is a context word

For a center word c and context word o:

② Exponentiation makes anything positive

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

① Dot product compares similarity of o and c.
$u^T v = u \cdot v = \sum_{i=1}^{n} u_i v_i$
Larger dot product = larger probability

③ Normalize over entire vocabulary to give probability distribution

$P(w_{t+j} \mid w_t; \theta)$

$$\frac{\partial}{\partial v_c} \log P(o|c) = \frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

NOTE: Each word type has two word representation

1. as a center word
2. as a context word

v_c : center word vector
u_o : context word vector

$$= \underbrace{\frac{\partial}{\partial v_c} \log \exp(u_o^T v_c)}_{1} - \underbrace{\frac{\partial}{\partial v_c} \log \sum_{w=1}^{V} \exp(u_w^T v_c)}_{2}$$

(1) $$\frac{\partial}{\partial v_c} \log \exp(u_o^T v_c) = \frac{\partial}{\partial v_c} u_o^T v_c = u_o$$

log & exp: Inverses

NOTE: $\frac{\partial}{\partial v_c} u_o^T v_c$ we can do this one variable at a time

$$\forall j \; \frac{\partial}{\partial (v_c)_j} u_o^T v_c = \frac{\partial}{\partial (v_c)_j} \sum_{i=1}^{d} (u_o)_i (v_c)_i = (u_o)_j$$

↳ Each term is zero except when $i = j$.

(2) $$\frac{\partial}{\partial v_c} \underbrace{\log}_{f} \underbrace{\sum_{w=1}^{V} \exp(u_w^T v_c)}_{z = g(v_c)}$$

solve with chain rule.

NOTE: Chain Rule

if $y = f(u)$ where $u = g(x)$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

(2) $$= \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \cdot \frac{\partial}{\partial v_c} \sum_{x=1}^{V} \exp(u_x^T v_c)$$

use chain rule again

$$= \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \cdot \left(\sum_{x=1}^{V} \frac{\partial}{\partial v_c} \exp(u_x^T v_c) \right)$$

$\frac{\partial}{\partial v_c}$: f ; $\exp(u_x^T v_c)$: $z = g(v_c)$

$$\left(\sum_{x=1}^{V} \exp(u_x^T v_c) \frac{\partial}{\partial v_c} u_x^T v_c \right)$$

We found this above

$$\left(\sum_{x=1}^{V} \exp(u_x^T v_c)\, u_x \right)$$

Sum up

$$\frac{\partial}{\partial v_c} \log(P(o|c)) = u_o - \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \left(\sum_{x=1}^{V} \exp(u_x^T v_c)\, u_x \right)$$

$$= u_o - \sum_{x=1}^{V} \frac{\exp(u_x^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)} u_x$$

Distribute terms across sum

$$= u_o - \sum_{x=1}^{V} P(x|c)\, u_x$$

This is an expectation. Average over all context vectors weighted by their probability

= observed - expected

To train a model we gradually adjust parameters to minimize loss. Optimize params with walking down the gradient.
Compute all vector gradients.