

Word2Vec

WordNet

nlp的瑞士军刀，可以当一个nlp工具库来用

缺点

Problems with resources like WordNet

- Great as a resource but missing nuance
 - e.g. “**proficient**” is listed as a synonym for “**good**”. This is only correct in some contexts.
 - Missing new meanings of words
 - e.g., **wicked, badass, nifty, wizard, genius, ninja, bombest**
 - Impossible to keep up-to-date!
 - Subjective
 - Requires human labor to create and adapt
 - Can't compute accurate word similarity → something like good and marvelous aren't in the same synonymy
- 13 所以，如果好的和奇妙的东西不在同一个同义词集中，

传统NLP的Word表示

用One-hot编码表示单词，但是有两个问题：

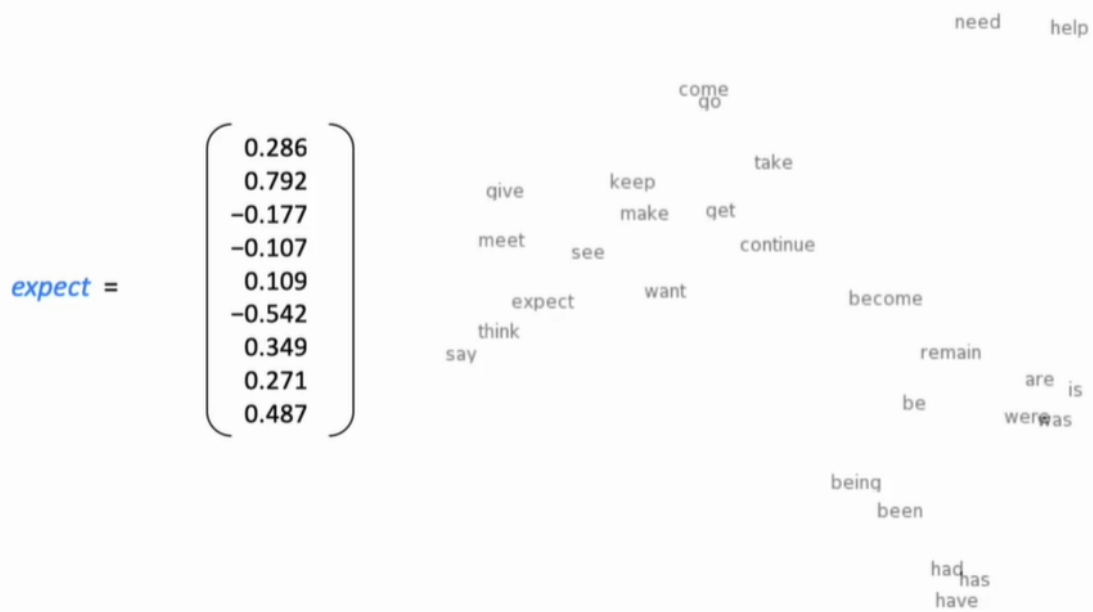
- 单词数量太多了，向量需要上百万维。
- One-hot都是正交的，无法得到word之间的相似性。

Distributed Representation

"如果你知道什么时候该用一个word，什么时候不该用一个word，那么你就已经理解了它的 meaning。"

word vector就是一种distributed representation，即一个多维向量表示一个word。下图是100维投影到2维的可视化图像。

Word meaning as a neural word vector – visualization



18

Um, so, on the one hand, um,

因 一方面 因

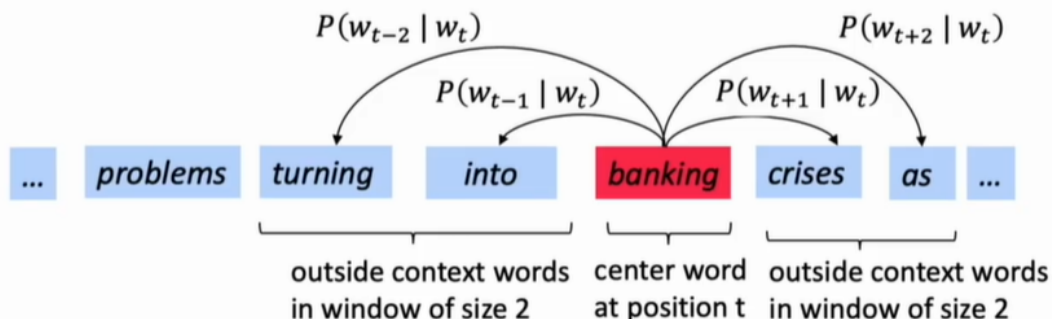
Word2vec算法

要利用word上下文的信息，word在中心称为center，word作为上下文称为context。

用u和v两个向量分别表示一个word在context和在center的向量。

Word2Vec Overview

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



Word2vec: objective function

For each position $t = 1, \dots, T$, predict context words within a window of fixed size m , given center word w_j .

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^T \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} | w_t; \theta)$$

θ is all variables to be optimized

sometimes called *cost* or *loss* function

The **objective function** $J(\theta)$ is the (average) negative log likelihood:

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

Minimizing objective function \Leftrightarrow Maximizing predictive accuracy

上图是最大似然估计(MLE), 取负取log之后变成要minimize的目标函数。

- Question: How to calculate $P(w_{t+j} | w_t; \theta)$?
- Answer: We will use two vectors per word w :
 - v_w when w is a center word
 - u_w when w is a context word
- Then for a center word c and a context word o :

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

23

上图是如何用词向量计算center出现时，context出现的条件概率。

方法是用softmax，其中V是词典里所有单词。想象某个center和context关联性极强，那么 $P(o|c)$ 趋近于1。

To train the model: Compute all vector gradients!

- Recall: θ represents all model parameters, in one long vector
- In our case with d -dimensional vectors and V -many words:

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

- Remember: every word has two vectors
- We optimize these parameters by walking down the gradient

上图是最简单的模型下的 θ 参数，即词典里每个wordvec的每个维度都视作一个参数。

手推求导

推导得到目标函数 $J(\theta)$ 的导数。

$$\max J'(\theta) = \prod_{t=1}^T \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} p(w'_{t+j} | w_t; \theta)$$

$$\underset{\substack{\uparrow \\ \text{change!}}}{\text{minimize}} J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log p(w'_{t+j} | w_t)$$

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

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

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

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

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

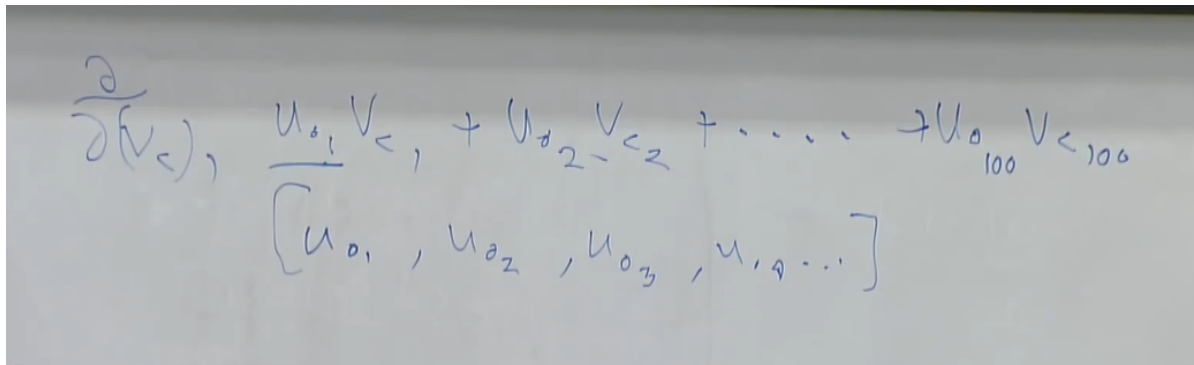
$$\frac{\partial}{\partial v_c} \log p(o|c) = u_o - \sum_{x=1}^V \left[\frac{\exp(u_x^T v_c)}{\sum_{w=1}^V \exp(u_w^T v_c)} \right] \cdot u_x$$

$$p(x|c)$$

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

导数 $u_o - \sum_{x \in V} p(x|c)u_x$ 的直观意义是 actual context word 和 expected context word (各个 context 的加权和) 的差值。这样最后所有 u_i 都变成 $\sum p(x|c)u_x$ 吗? 不是, 这里是对 v_c 求导。 v_c 最后期望达到 actual context word = expected context word

下图证明了为什么 $\frac{\partial(u^T v)}{\partial v} = u$ 。



The image shows a handwritten derivation on a chalkboard. It starts with the partial derivative of a dot product with respect to a vector component v_c :

$$\frac{\partial}{\partial v_c}, \quad u_{o_1} v_c + u_{o_2} v_{c_2} + \dots + u_{o_{100}} v_{c_{100}}$$

Below this, the vector u is written as a row vector in brackets:

$$[u_{o_1}, u_{o_2}, u_{o_3}, u_{o_4}, \dots]$$