Week1. CS 224n : NLP with Deeping Learning

Lecture 1 : Introduction and Word Vectors

In Traditional NLP, words as discrete symbols : a localist representation

Means one 1s, the rest 0s   one-hot vectors

vector dimension is big (500,000+)

But also no similarity for one-hot vectors

* learn to encode similarity into the vectors themselves

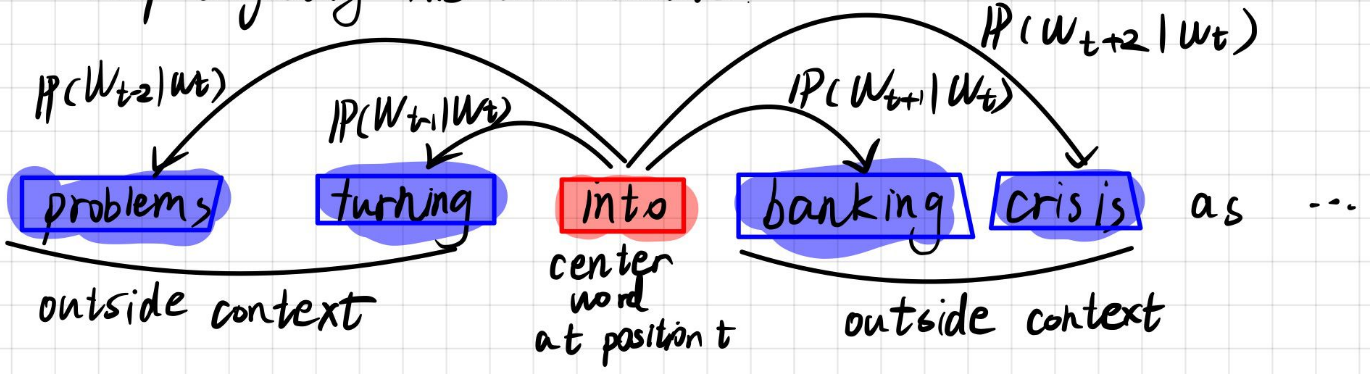
Representing words by their context.

"A word's meaning is given by the words that frequently appear near-by"

Word Vectors : a dense distributed vector

Word2vec:

- a large corpus
- each word by a vector
- has center word "c" and context outside word "o"
- use similarity of the word vectors for "c" and "o" to calculate the probability of "o" given "c".
- keep adjusting the word vectors



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

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

θ is all parameters to be optimized

The objective function $J(\theta)$ is the average Log likelihood :

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

Q : how to calculate $P(W_{t+j}|W_t;\theta)$?

Answer: 2 vectors
- 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:

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

Dot product produces similarity

sum to normalize

- This is an example of softmax function $\mathbb{R}^n \mapsto \mathbb{R}^n$

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

- "max" means amplifies probability of largest x_i
- "soft" still assign some probability to smaller x_i
- Frequent used in Deep Learning

To train the model, Compute all $\boxed{2d}$ vector gradients.

(recall every word has 2 vectors)

Blackboard Derivation

$$\max \ L(\theta) = \prod_{t=1} \prod_{\substack{-m\le j\le m \\ j\neq 0}} \mathbb{P}(w_{t+j} | w_t; \theta)$$

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

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

<useful tips: $\frac{\partial x^T a}{\partial x} = \frac{\partial a^T x}{\partial x} = a$>

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

$$= \frac{\partial}{\partial v_c} u_o^T v_c - \frac{\frac{\partial}{\partial v_c} \sum_{w\in V} \exp(u_w^T v_c)}{\sum_{w\in V} \exp(u_w^T v_c)}$$

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

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

Current context word   expected context word

home derivation

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

$$= v_c - \frac{\frac{\partial}{\partial u_o} \sum_{w \in V} \exp(u_w^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

$$= v_c - \frac{\frac{\partial}{\partial u_o} \exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

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

$$= v_c - P(O \mid C) \cdot v_c$$

Lecture 2: Word Vectors and Word Senses

* $\nabla_\theta J(\theta)$ very sparse in SGD
  — only update the word vectors that actually appear.

Two model variants:

1. Skip-grams (SG)
   predict context ("outside") words (position independent) given center words
2. Continuous Bag of Words [CBOW]
   predict center word from (bag of) context words

Additional efficiency in training: Negative sampling

In hw 2: the skip-gram model with negative sampling

window-based cooccurrence matrix

- I like deep learning
- I like NLP
- I enjoy flying

	I	like	enj	dep	learnig	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enj	1	0	0	0	0	0	1	0
dep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
fly	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

- increase in size
- sparsity
- much storage

Solution: Low dimensional vectors

Method: SVD (hw1) Factor X into $U\Sigma V^T$

Method: Hacks to X
- $\min(X, t)$, $t \approx 100$
- ramped windows, count closer words more

Count-based v.s. direct prediction

- LSA, HAL
- COALS, Hellinger-PCA
- Fast training
- Efficient use of statistic
- primarily used to capture word similarity
- Disproportionate importance given to large corpus

- skip-gram / CBOW
- NNLM, HLBL, RNN
- Scale with corpus size
- Inefficient use of statistics
- Generate improved performance
- capture complex patterns

Evaluation in NLP: intrinsic v.s. extrinsic

Word Sense ambiguity:
- common words
- words that exist for a long time