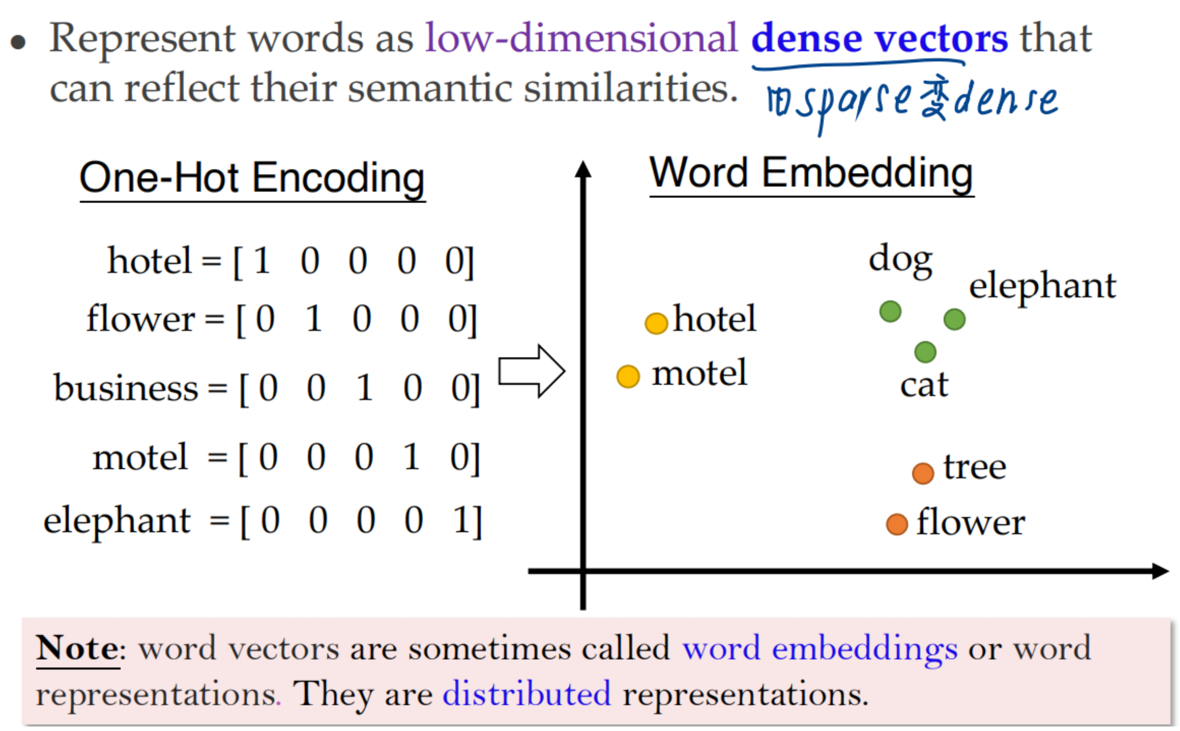
ch9: Word Embedding

Word Embedding

overview(P6)



Why Word Embeddings?

> Can capture the rich relational structure of the lexicon.(抓住词之间的语义关系)

- Two words are similar if they have similar word contexts

Models

Counting based: the vector space model(P10-14)

- The cornerstone technology in information retrieval.

信息检索中的重要算法

- **Term-Document Matrix**

Each cell is the count of word t in document d

在文档d_4中出现的次数

	d_1	d_2	d_3	d_4	d_5
ekonomi	0	1	40	38	1
pusing	4	5	1	3	30
keuangan	1	2	30	25	2
sakit	4	6	0	4	25
Inflasi	8	1	15	14	1

vector of d_3
= [40, 1, 30, 0, 15]

- Two documents are similar if they have similar vector!

d_3 = [40, 1, 30, 0 15]
d_4 = [38, 3, 25, 4, 14]

- **Weighting**: in practice, we usually use weights such as TF-IDF, instead of just using raw counts (only TF).

$$\text{tf-idf}_{w,d} = \text{tf}_{w,d} \times \underbrace{\log(N / \text{df}_w)}_{\text{惩罚项}}$$

$\text{tf}_{w,d}$ = frequency of w in d
df_w = number of documents containing w → 如果在所有文档中都出现，说明重要性一般
N = total number of documents

	d_1	d_2	d_3	d_4	d_5
ekonomi	0	1	40	38	1
pusing	4	5	1	3	30
keuangan	1	2	30	25	2
sakit	4	6	0	4	25
Inflasi	8	1	15	14	1

TF(sakit) = [4, 6, 0, 4, 25]

DF(sakit) = 4
N = 5
} IDF(sakit) = log(5/4)

TF-IDF(sakit) = [……]

- **Limitations of Vector Space Model**

- TF-IDF vectors are
  - long (length |V| = 20,000 to 50,000)
  - sparse (most elements are zero)

  - difficult to use as features in machine learning (more weights to tune)
  - storing explicit counts can be difficult for generalization

在机器学习中很难作为特征使用（更多的权重来调整）
存储显式计数很难泛化

Prediction based: word2vec(P16-26)

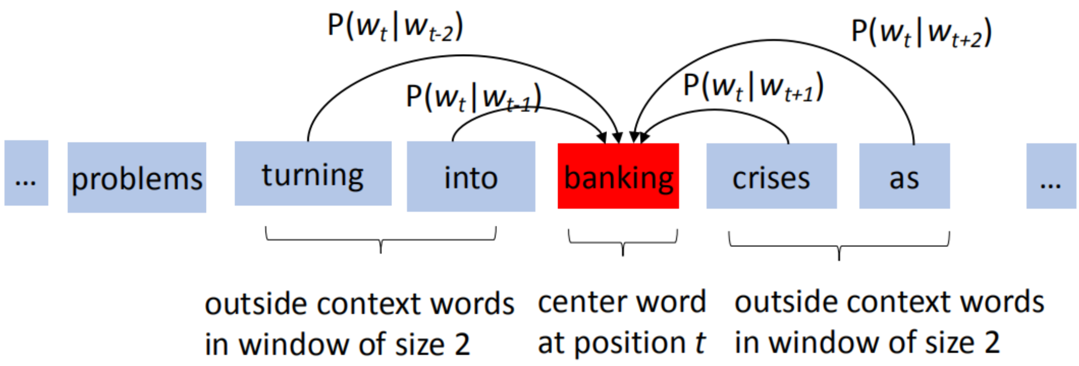
- overview(P17)

- **Word2vec** (Mikolov et al. 2013) is a framework for learning word vectors.

Idea:

- we have a large corpus of text.
- every word in a fixed vocabulary is represented by a vector; learnable，可变的
- go through each position t in the text, which has a center word c and context ("outside") words o ; 保证word和neighbor word共同出现的概率尽可能高
- use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa) ;
- keep adjusting the word vectors to maximize this probability.

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



- **Mikolov's CBOW(P20)**

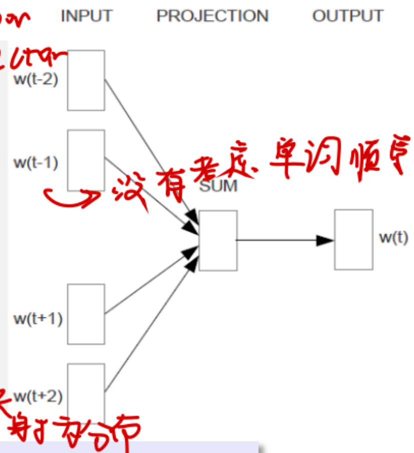
- **CBOW**: the distributed representations of context (or surrounding words) are combined to **predict the word in the middle.**

$$P(w_t|w_{t-k}, \ldots w_{t-1}\ w_{t+1} \ldots w_{t+k}) = \mathrm{NN}\ (V(w_{t-k}) + \ldots + V(w_{t+k}))$$

neighbor vector

This can be represented by a **neural network**:

- An input layer which converts each word (one-hot) into a dense vector. 单词→向量
- A projection layer which combines the vectors of input words. 变成一个向量
- An output layer which predicts the target word w_t given the combined context vector. 把向量映射分布



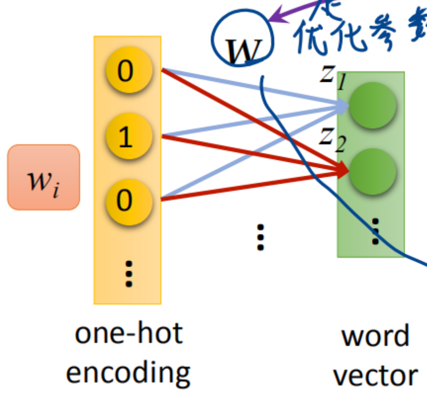
CBOW = Continuous-Bag-of-Words
the order of words in the context does not influence the projection.

- **Architecture(P21-24)**

- **Input Layer:** representing any word into a vector. $z_i = Wx_i = W$

是优化参数，有降维的作用，是线性变换

The weight matrix $W \in R^{|V| \times d}$ is a **lookup table** with each row W_i being the vector for word w_i.



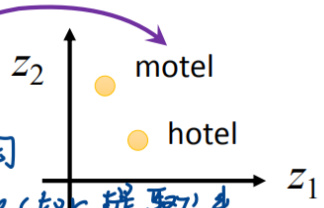
一个查找表

Example

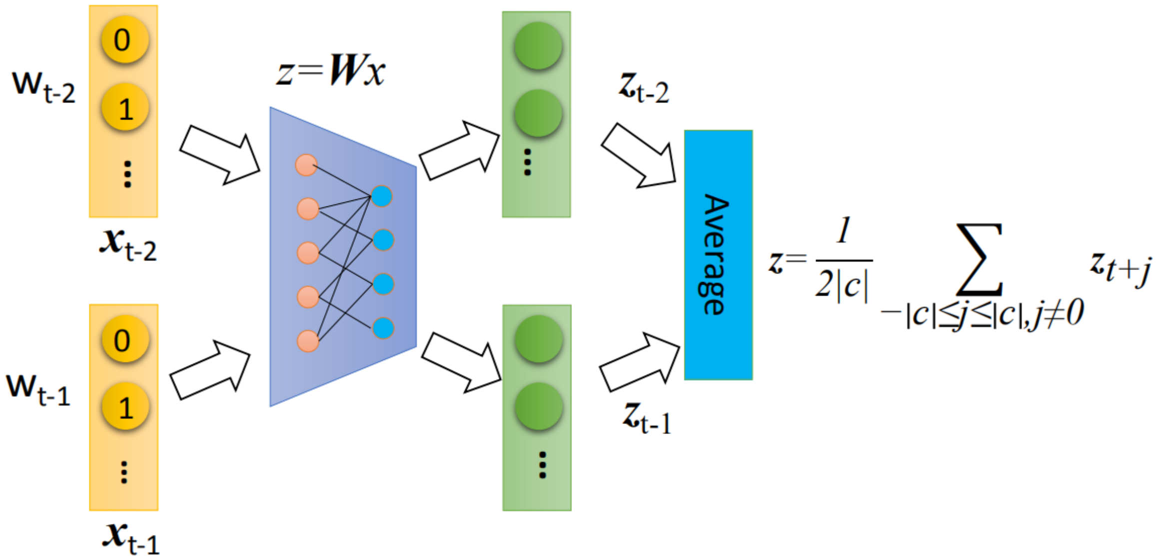
$$[0\ \ 0\ \ 0\ \ \boxed{1}\ \ 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10\ \ 12\ \ 19]$$

"motel"

本质上是把单词对应的vector提取出来

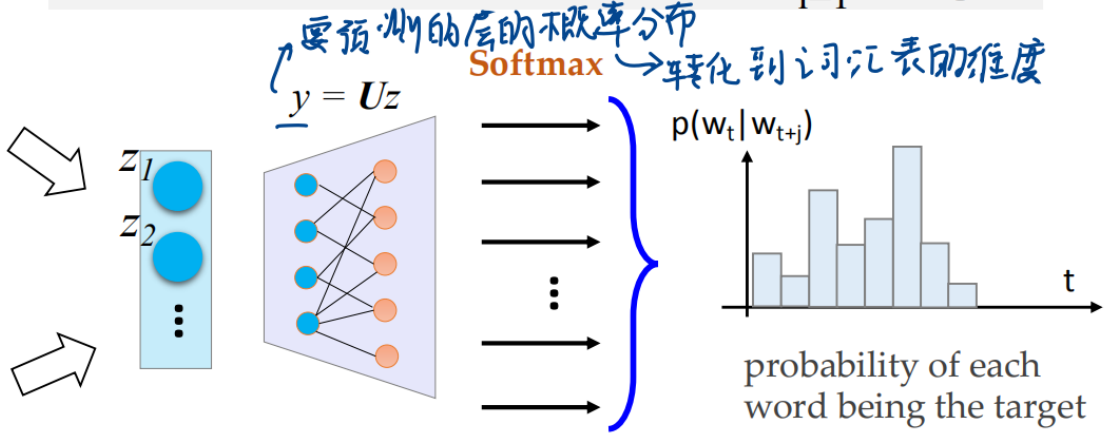


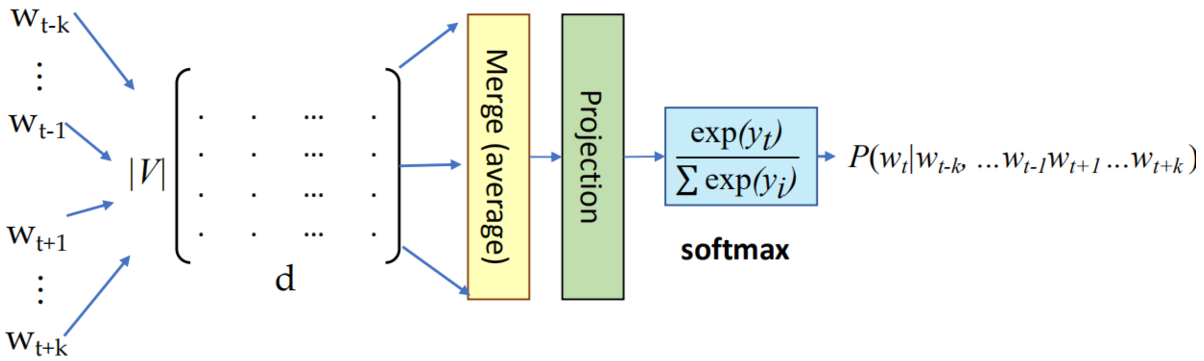
- **Projection Layer**: combining context vectors into one vector.



- **Output Layer**: predicts the probability of the target word.

$$P\left(w_t \middle| w_{t-|c|}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+|c|}\right) = \frac{\exp(y_t)}{\sum_{i=1}^{|V|} \exp(y_i)}$$





- **Training(P25)**

- Given $D = \{w_1, w_2, \ldots, w_N\}$, minimize the **negative log likelihood** (NLL) loss function:

与交叉熵等价

$$L(W, U \mid D) = -\frac{1}{N}\sum_{t=1}^{N} \log p(w_t | w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k})$$

using gradient descend.

The Skip-Gram Model（P27)

- We seek a model for $P(w_{t+j} \mid w_t)$.

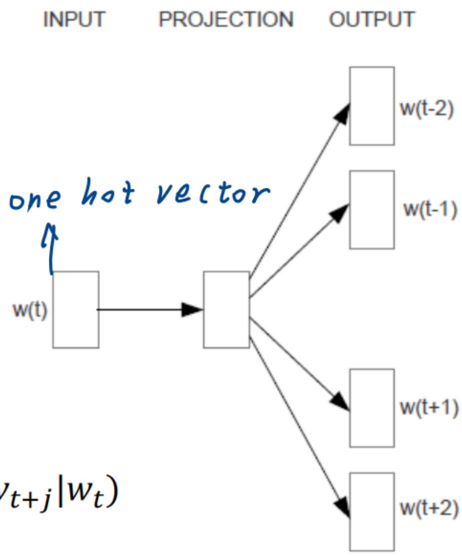
$$P(w_{t+j}|w_t) = \frac{\exp(y_{t+j})}{\sum_{i=1}^{|V|}\exp(y_i)}$$

$$y = Uz$$

$$z = Wx$$

想要得到的

别的词出现的概率



$$L(W, U|\chi) = -\frac{1}{N}\sum_{t=1}^{N}\sum_{-c\le j\le c, j\neq 0} \log p(w_{t+j}|w_t)$$

The Word Analogy Task(P29-30)

- Word Analogy: 单词类推

a:b :: c:?

man:woman :: king:?

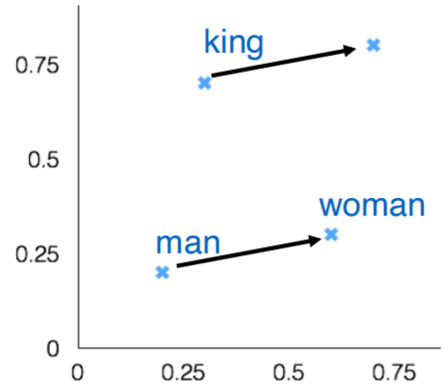
Examples
- Man is to Woman as King is to___?
- Good is to Best as Smart is to___?
- China is to Beijing as America is to__?

How to find d ?

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

- It turns out that word2vec is good for such an analogy task.

$V_{king} - V_{man} + V_{woman} = V_{queen}$



Language Models(P33-35)

- A probabilistic model of how likely a given string appear in a given “language”. 评估一个string在语言中出现的概率
- For a given sequence $x = (w_1, w_2, \ldots, w_N)$. A **language model** can be defined as:

$$p(x) = p(w_1, w_2, \ldots, w_N)$$

概率越高，对语言的掌握程度越高

Example:

P_1=P(“我爱机器学习”)
P_2=P(“我爱学习机器”)
P_3=P(“机器我爱学习”)
P_4=P(“爱我机学习器”)

Chinese: $P_1 > P_2 > P_3 > P_4$

- Applications:

message suggestion; document generation; spelling correction; machine translation; speech recognition;…

- What is the probability of $P(w_1, \ldots, w_N)$?

p(我爱机器学习) = ?

- Chain Rule:

$$p(w_1,\ldots,w_N) = p(w_1)p(w_2|w_1)\ldots,p(w_N|w_1,\ldots,w_{N-1})$$

p(我爱机器学习) = p(我)p(爱|我)p(机|我爱)p(器|我爱机)p(学|我爱机器)p(习|~~我爱~~机器学)

在一个词生成时，只看前面n-1个

- Markov Assumption: (only consider the last *n-1* words)

$$p(w_i|w_1,\ldots,w_{i-1}) = p(w_i|w_{i-n+1},\ldots,w_{i-1})$$

p(习|我爱机器学) ≈ p(习|机器学) ≈ p(习|学)

So that's what we get for n=2: 只看前面一个

$$p(w) = p(w_1)p(w_2|w_1)\ldots,p(w_N|w_{N-1})$$

1/18 × 1/8 × 1/120 × 1/4 × 1/420 × 1/2

p(我爱机器学习) = p(我)p(爱|我)p(机|爱)p(器|机)p(学|器)p(习|学)