Machine Learning

Lecture 9: Application - Word Embedding

单词嵌入

Fall 2022

Instructor: Xiaodong Gu

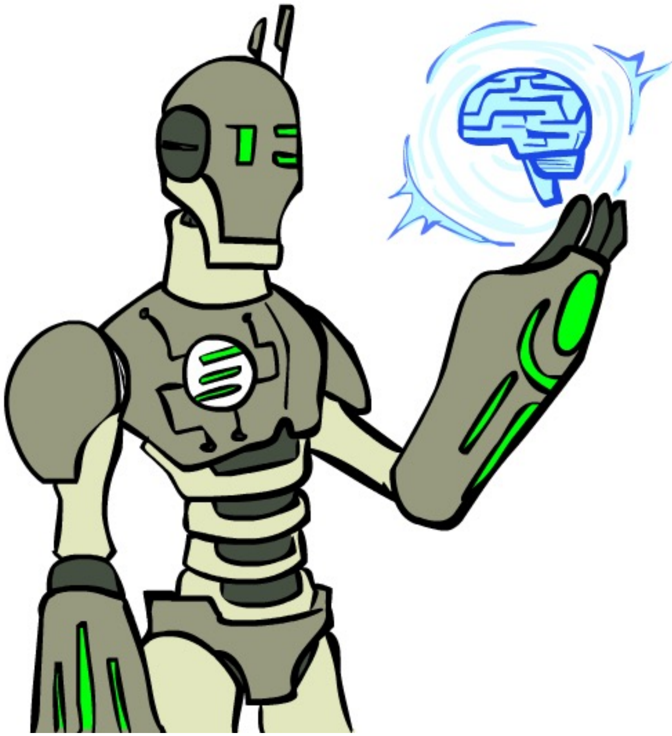
Today

- Word Embedding (word2vec)

  an application of **neural networks** for distributed representations of words

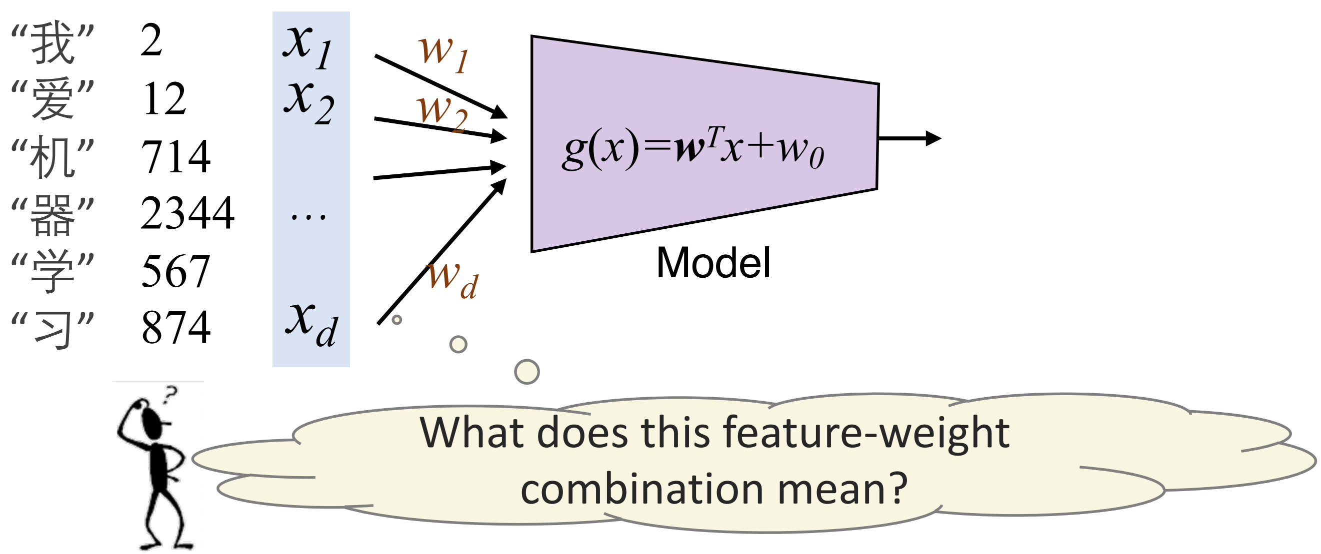
- Language Model 语言模型

  a key concept in NLP that is important in the follow-up lectures.



Before learning word embedding…

- How to represent words in computers?
  - discrete-symbols (ASCII, id, etc.)
  - e.g., "hotel", "conference", "motel", 1203, …
  - a localist representation
- How to represent words in machine learning models?



Words as Vectors

- Words can be represented by **one-hot** vectors:

Means one 1, the rest 0s

One-Hot Encoding

hotel = [1 0 0 0 0 0 0 0 0]

flower = [0 1 0 0 0 0 0 0 0]

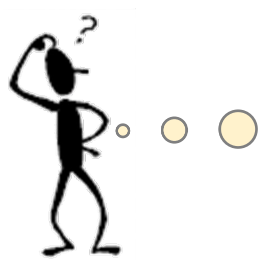
tree = [0 0 0 1 0 0 0 0 0]

motel = [0 0 0 0 0 0 0 1 0]

elephant = [0 0 0 0 0 0 0 0 1]

Vector dimensionality = number of words in vocabulary (e.g., **500,000**)

直接使用one-hot vector表示word的缺点：
1.过多的0，向量过于稀疏
2.只含有1与0，不好获取单词之间的关联性以及相似性(因为两两正交)



How can we capture relationships (similarity) between words?

Problems of One-hot Encoding

Example: in web search, if user searches for "Minhang motel", we would like to match documents containing "Minhang hotel".

But

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

orthogonal.

There is no natural notion of **similarity** for one-hot vectors!

Solution:

- Use **WordNet**'s list of synonyms?
- But it is well-known to fail badly: incompleteness, etc.

Instead: learn to encode similarity in the vectors themselves

Distributed Word Representation

- Represent words as low-dimensional **dense vectors** that can reflect their semantic similarities. dense：密集的，稠密的

One-Hot Encoding

hotel = [1 0 0 0 0]

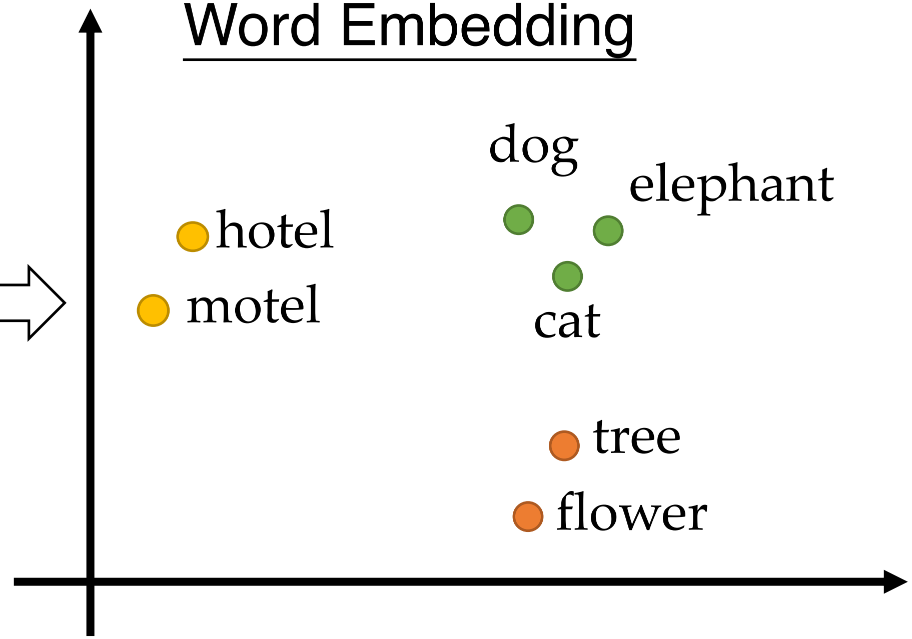
flower = [0 1 0 0 0].

business = [0 0 1 0 0]

motel = [0 0 0 1 0]

elephant = [0 0 0 0 1]

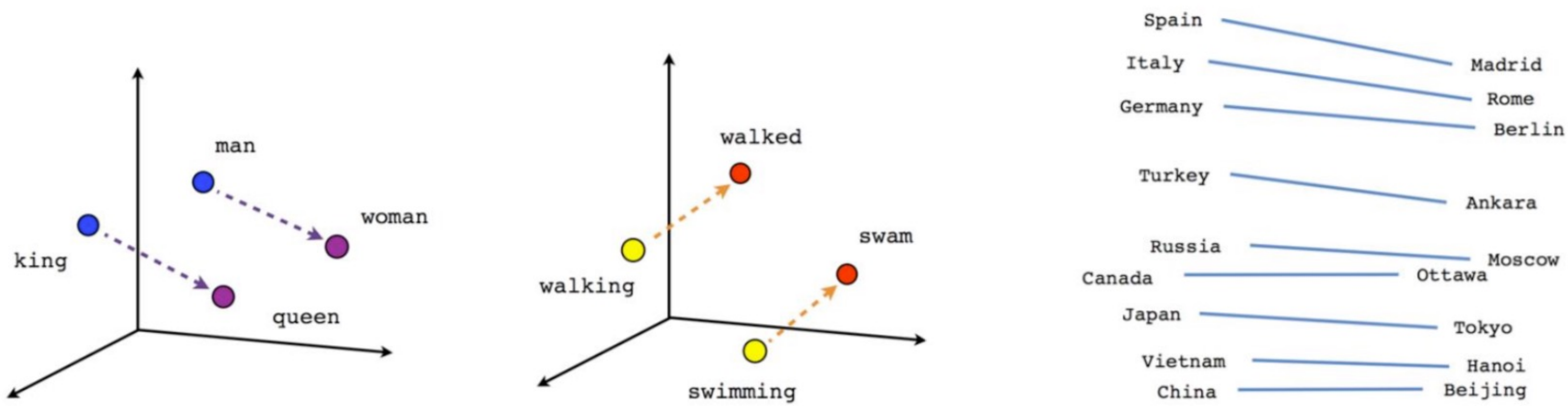
Word Embedding



Note: word vectors are sometimes called word embeddings or word representations. They are distributed representations.

Why Word Embeddings?

- Can capture the rich relational structure of the lexicon.



Male-Female | Verb tense | Country-Capital

How to obtain word embeddings?

How to obtain word embeddings?

- A word can be understood by its context

"A bottle of ***tesgüino*** is on the table"
"Everybody likes ***tesgüino***"
"***Tesgüino*** makes you drunk"
"We make ***tesgüino*** out of corn"

- From context words we can guess **tesgüino** means an alcoholic beverage such as beer.

"You shall know a word by the company it keeps" (J. R. Firth 1957)

- Intuition for an algorithm:

Two words are similar if they have similar word contexts

idea1：根据单词的上下文来判断单次可能的语义，并根据上下文来判断单词的相似性。

How to Exploit the Context?

Counting-based Approach

two words have similar vectors if they frequently co-occur in the same context (sentence, document, etc.).

e.g. Glove:

Prediction-based Approach

train **neural networks** to predict a word (vector) given its context words (vector).

e.g., word2vec

Counting based: the vector space model

(向量空间模型)

Vector Space Model

- The cornerstone technology in information retrieval.
- **Term-Document Matrix**

  Each cell is the count of word t in document d

	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]

(看的是列向量)

Vector Space Model

- **Term-Document Matrix**

Each cell is the count of word t in document d

	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 the word "sakit" = [4, 6, 0, 4, 25]

- Two words are similar if they have similar vector!

pusing = [4, 5, 1, 3, 30]
sakit = [4, 6, 0, 4, 25]

(看的是行向量)

Vector Space Model

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

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

log项是一个惩罚项，包含单次w的文档数量越多，同一个文档对于单词w的贡献越小，这样的话更容易体现出单词的独特性，即只出现在少数文档中的单词应该有更大的count。

$\text{tf}_{w,d}$ = frequency of w in d
df_w = number of documents containing w
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]

$$\left.\begin{array}{l}\text{DF(sakit)} = 4 \\ N = 5\end{array}\right\} \quad \text{IDF(sakit)} = \log(5/4)$$

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

Prediction based: word2vec

Word2Vec: Overview

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

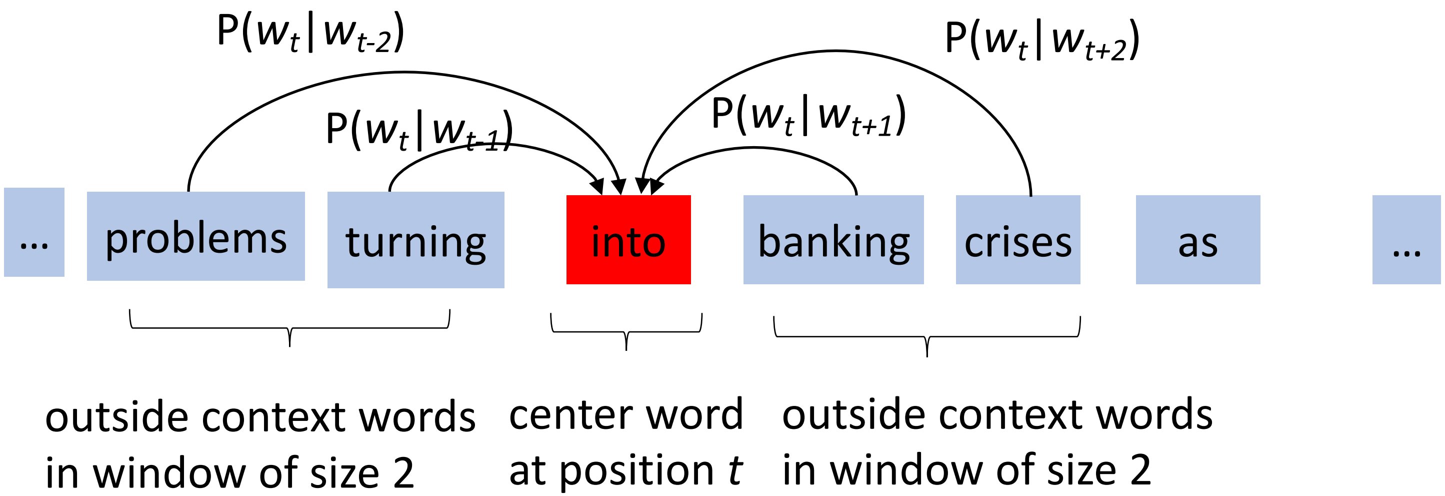
Idea:

- we have a large corpus of text.
- every word in a fixed vocabulary is represented by a **learnable vector**;
- go through each position t in the text, which has a center word c and context ("outside") words o ;
- 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.

Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. NIPS 2013.

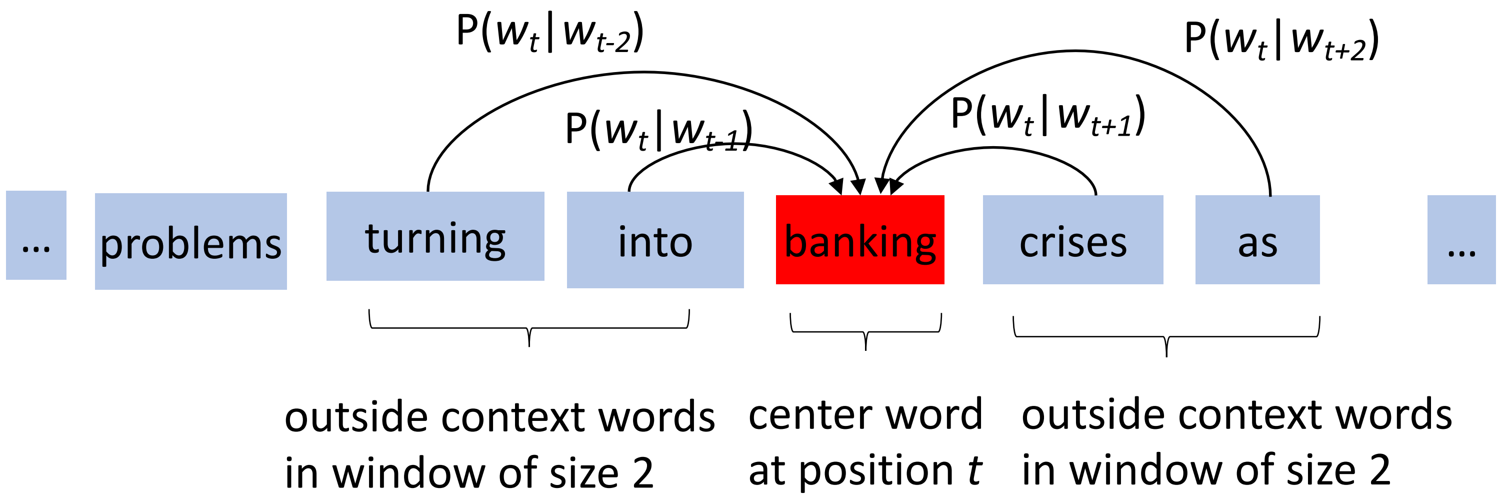
Word2Vec: Overview

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



Word2Vec: Overview

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



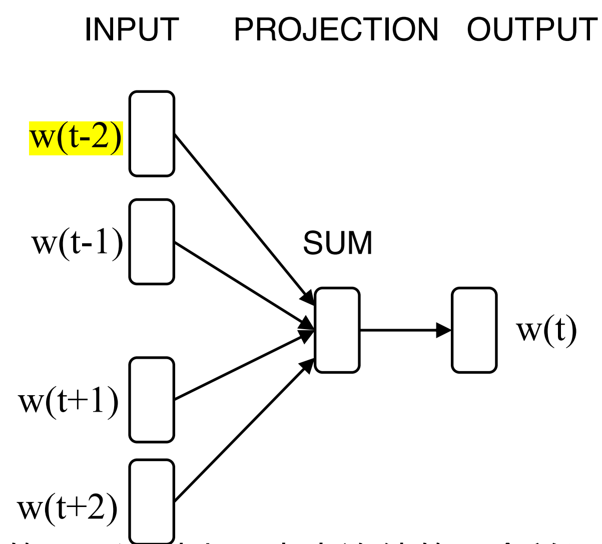
Mikolov's CBOW

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

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

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

- An input layer which converts each word (one-hot) into a dense vector.
- A projection layer which combines the vectors of input words.
- An output layer which predicts the target word w_t given the combined context vector.



CBOW = Continuous-Bag-of-Words

continuous指的是是通过上下文中连续的几个单词来判断中心单词的语义。

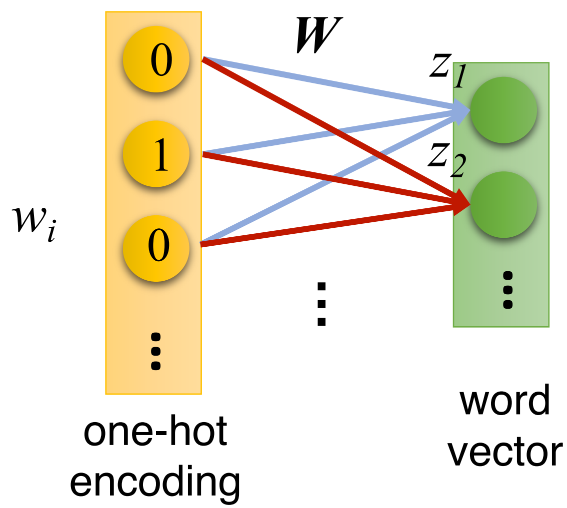
the order of words in the context does not influence the projection.

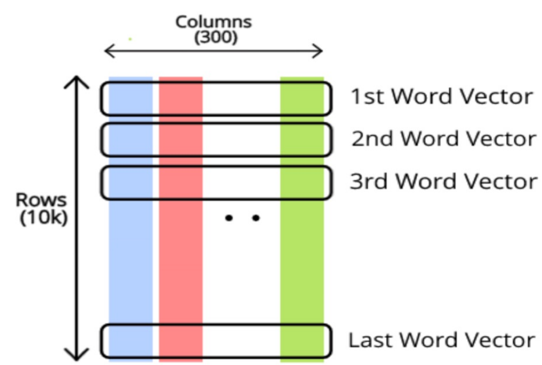
Model Architecture

- Input Layer: represent any word into a vector.

$$z_i = \boldsymbol{W}x_i = \boldsymbol{W}_i$$

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

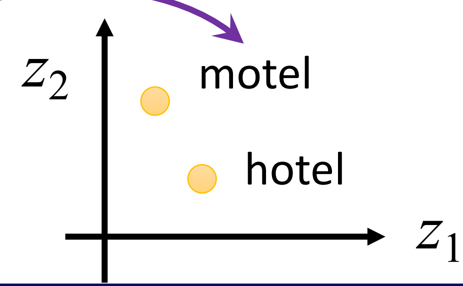




Example

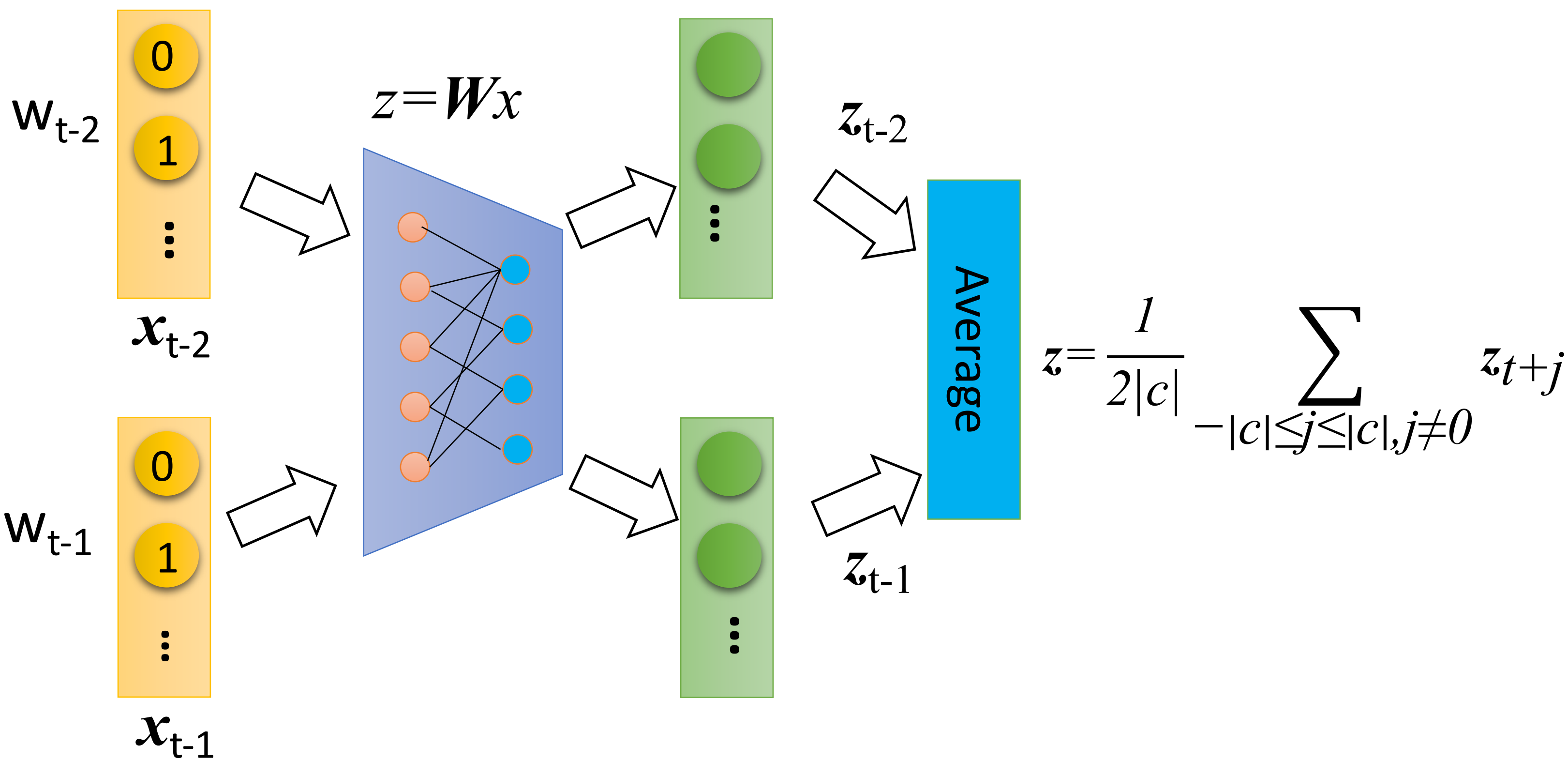
$$[0 \quad 0 \quad 0 \quad \boxed{1} \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ \boxed{10 \quad 12 \quad 19} \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

"motel"



Architecture

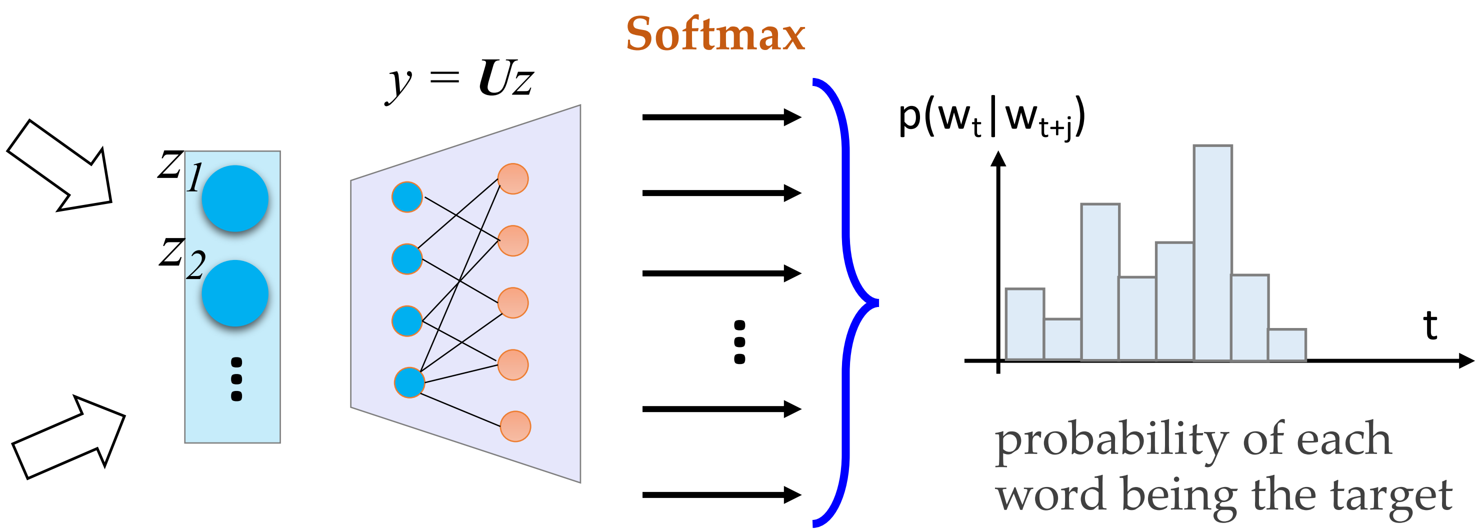
- Projection Layer: combining context vectors into one vector.



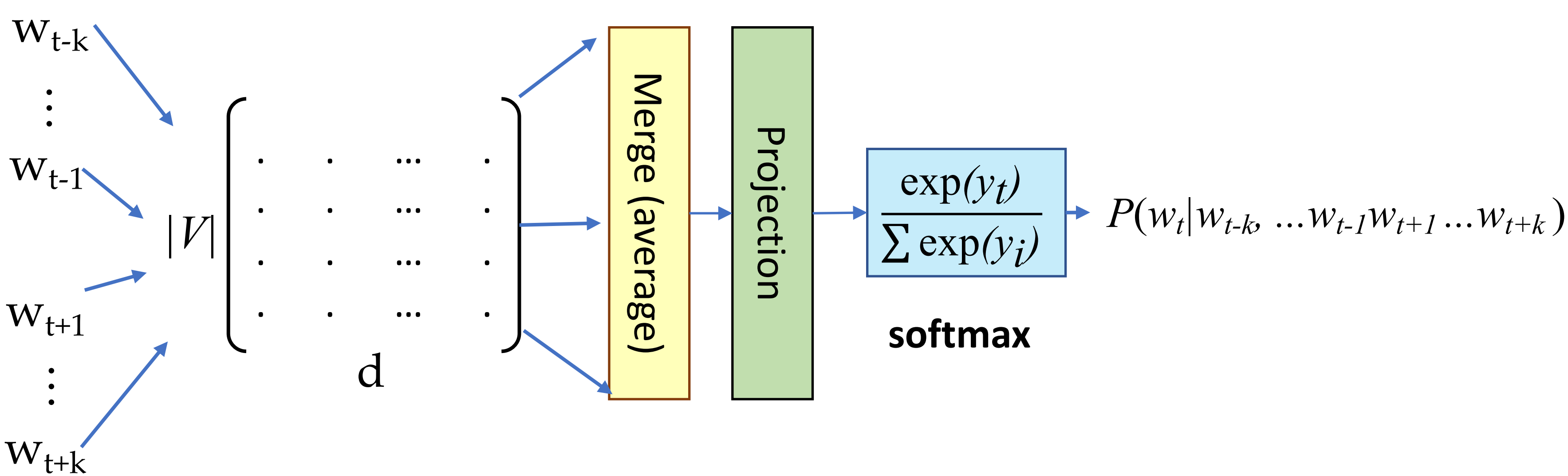
Architecture

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

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



Architecture: the big picture



Training

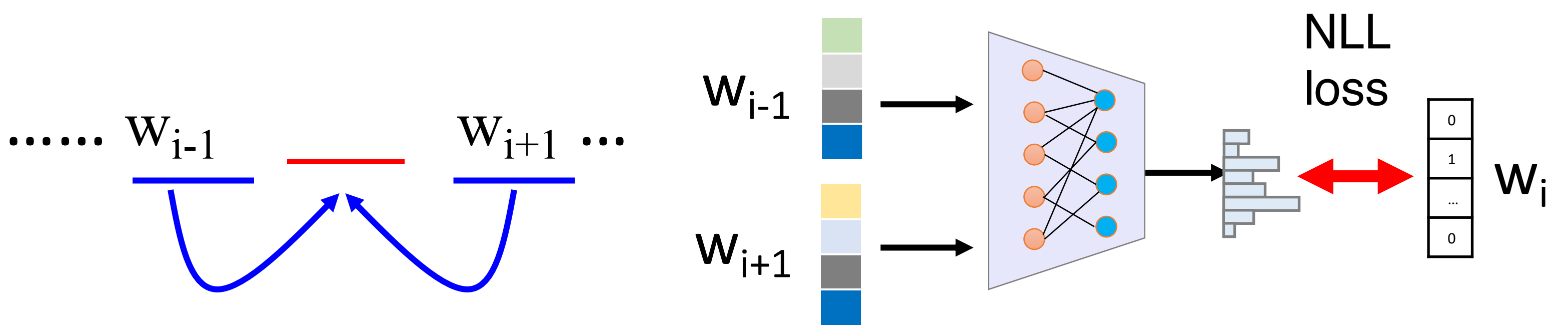
- Given $D = \{w_1, w_2, \ldots, w_N\}$, minimize the **negative log likelihood (NLL)** loss function:

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

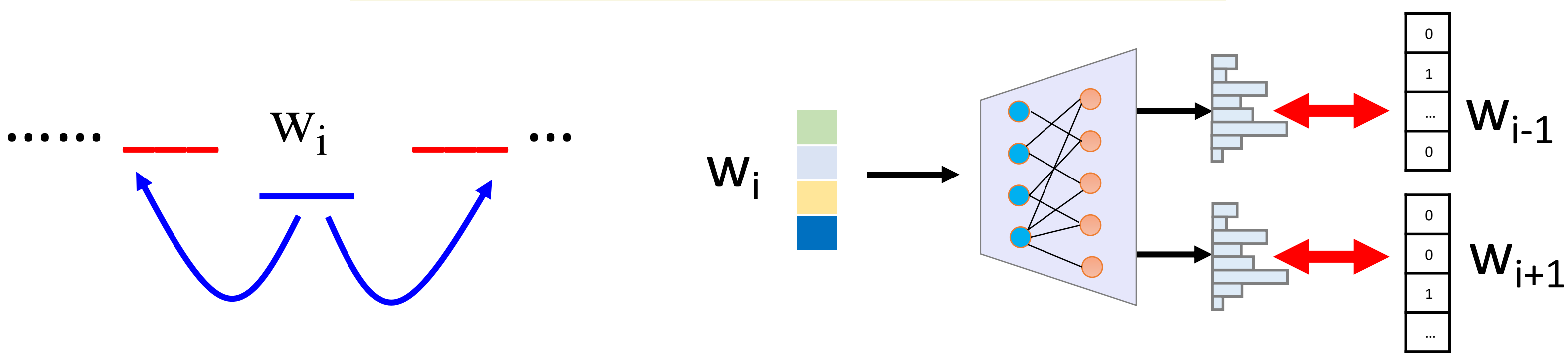
using gradient descend.

Other Models

- **CBOW** predict the word given its context



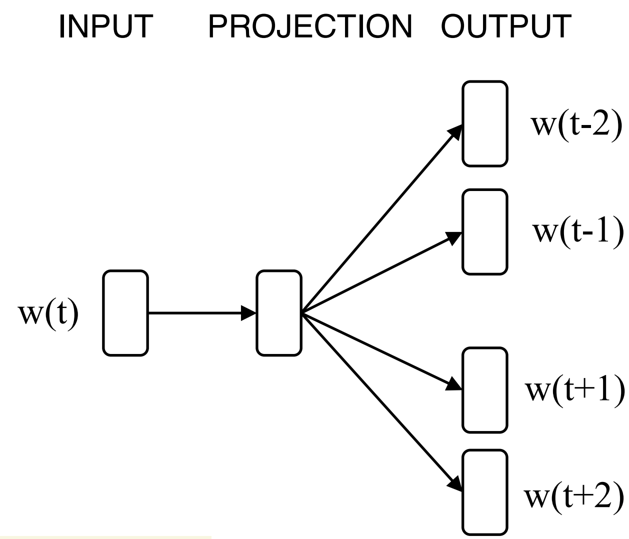
- **Skip-gram** predict the context given a word



The Skip-Gram Model

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

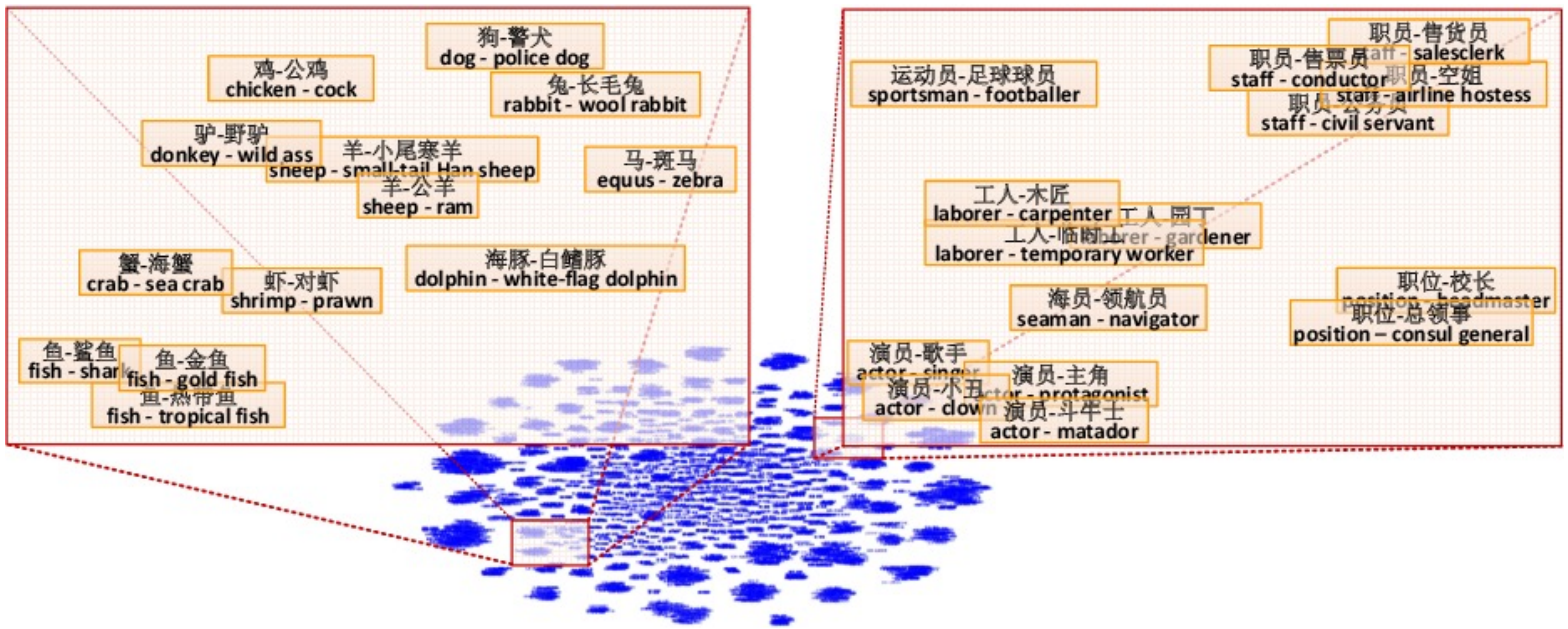
$$P(w_{t+j}|w_t) = \frac{\exp(y_{t+j})}{\sum_{i=1}^{|V|}\exp(y_i)}$$
$$y = \boldsymbol{U}z$$
$$z = \boldsymbol{W}x$$



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

Example

- Word2vec maximizes an objective function by putting similar words nearby in space.



Fu, Ruiji, et al. "Learning semantic hierarchies via word embeddings."*ACL* 2014.

The Word Analogy Task analogy：类比

- Word Analogy:

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__?

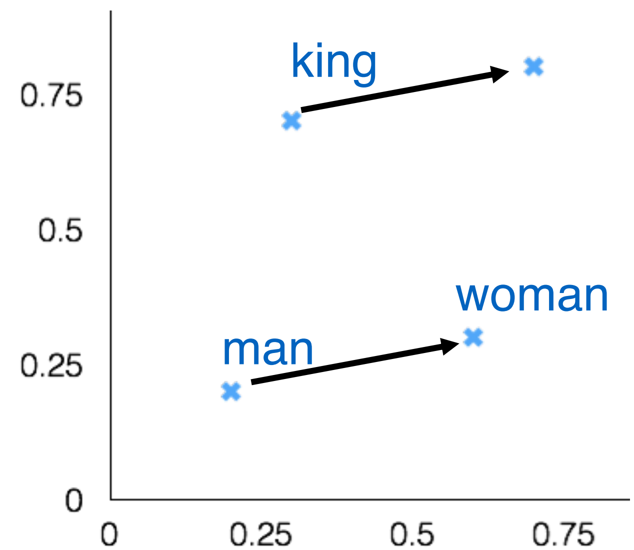
How to find d ?

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

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

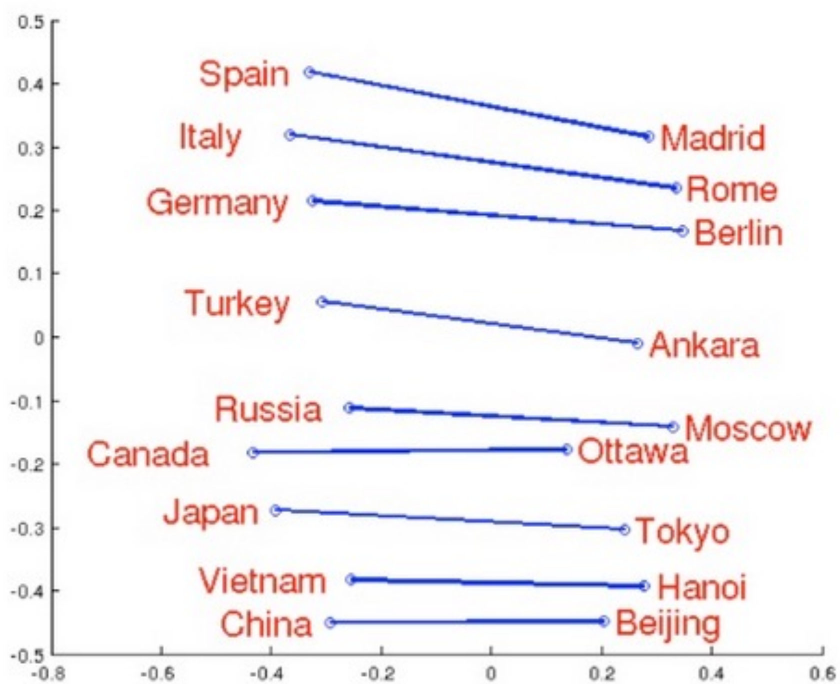
$V_{king} - V_{man} + V_{woman} = V_{queen}$

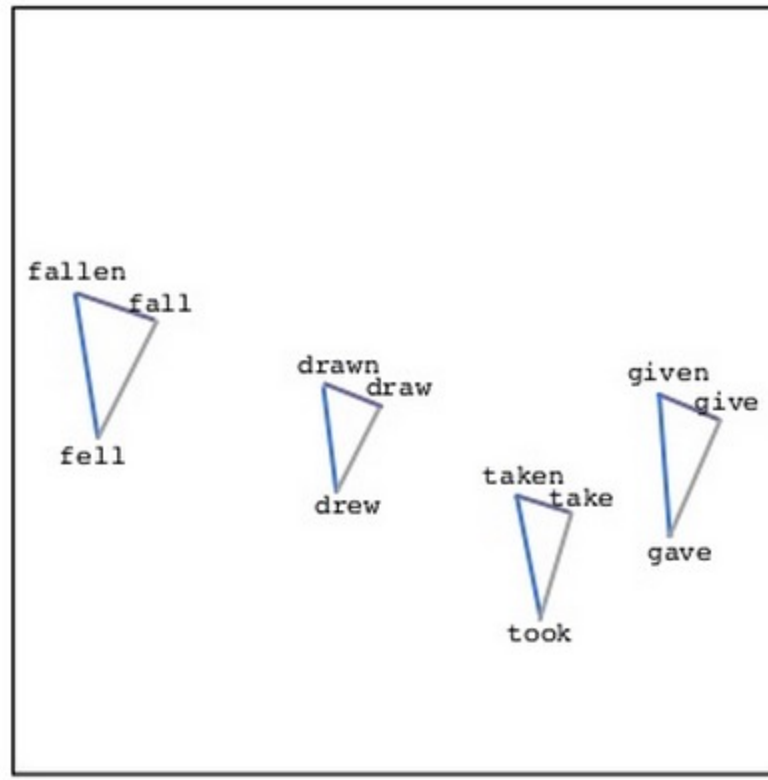
平行且同向时显然最大



Mikolov, T., et al. Distributed Representations of Words and Phrases and Their Compositionality. NIPS 2013.

Example



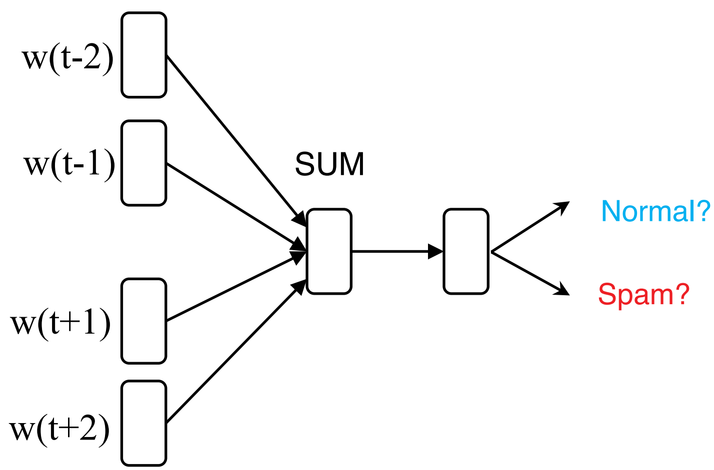


Source: http://www.slideshare.net/hustwj/cikm-keynotenov2014

Thinking

总体而言是一个非监督学习，但是其中间过程是需要softmax的，是一个分类模型，所以其中间过程也包括一部分的监督学习

- What kind of machine learning is word2vec?
- What limitations does word2vec have if we use it for text classification ?



Beyond Bag-of-Words

Order Matters!

- To understand the meaning of a sentence, the **order** of the words can not be ignored.

The food was good, not bad at all.

exactly the same bag-of-words

The food was bad, not good at all.

different meaning

Paragraph Vector: Le, Quoc, and Tomas Mikolov. "Distributed Representations of Sentences and Documents.“ ICML, 2014
Seq2seq Auto-encoder: Li, Jiwei, Minh-Thang Luong, and Dan Jurafsky. "A hierarchical neural autoencoder for paragraphs and documents." arXiv preprint, 2015
Skip Thought: Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, Sanja Fidler, “Skip-Thought Vectors” arXiv preprint, 2015.

Beyond Bag-of-Words

- **Language Models**
  Recurrent Neural Networks, Transformers,…

- **Pretrained Language Models**
  BERT, GPT-2, …

To appear in future lectures..

Language Models

- A probabilistic model of how likely a given string appear in a given "language".
- For any sequence $x = (w_1, w_2, \ldots, w_N)$, a **language model** can be defined as:

由于这个语言模型是判断一整个string出现在language的概率，所以其研究的应该是各个string中单次同时出现的概率，所以使用的是联合概率分布。

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

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;…

我爱机器学__?__

Language Model

- 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)

- 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)

Bigram Language Model

So that's what we get for n=2:

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

$1/18 \times 1/8 \times 1/120 \times 1/4 \times 1/420 \times 1/2$

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

Q: How to estimate these probabilities?

A: Straightforward counting.

Q: But remember in word embedding, counting discrete words have many drawbacks?

A: Don't worry, we have neural networks as language models.

What’s Next?

Recurrent Neural Networks

- A deep neural network for sequence (language) modeling.