# 自然语言处理

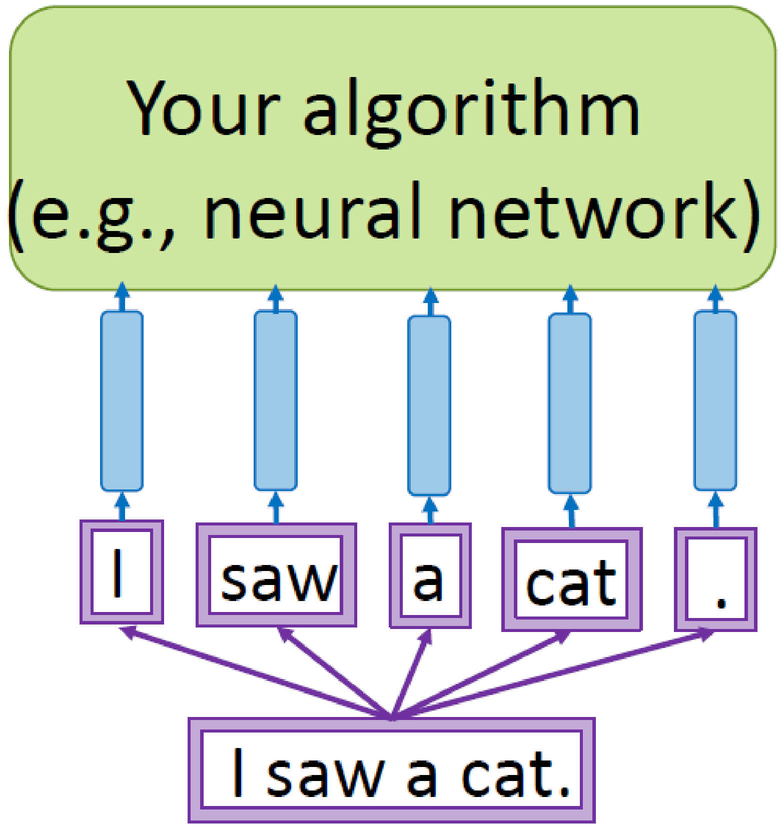
# 词和文档表示与相似度计算

**孙承杰 杨沐昀**
**sunchengjie@hit.edu.cn**
**哈尔滨工业大学计算学部**
**语言技术研究中心**

## 主要内容

- 词的表示
- 文档表示
- 文本相似度计算

# 词的表示方法是NLP的基础



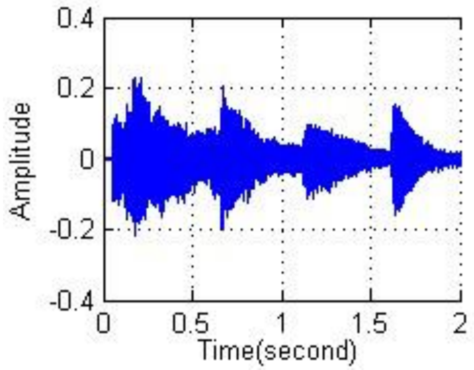
Any algorithm for solving any task

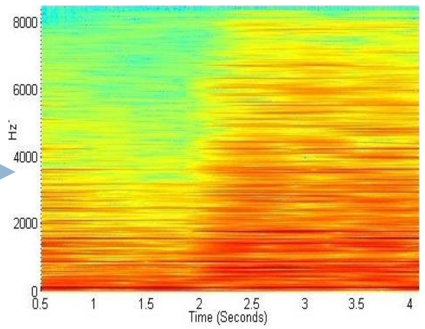
Word representation - vector (word embedding)

Sequence of tokens

Text

# 语言的向量化表示需求





图像和语音可以自然的转化为稠密的实数矩阵

这段文字如何转化成稠密的实数矩阵呢?

词袋模型
潜在语义索引
深度学习
…

语义的连续性能保证么?

计算机
电脑
PC

苹果
手机
香蕉

## 词的表示

- 独热表示 (One-hot representations)
- 词频-逆文档频率 (TF-IDF)
- 分布式表示
  - 潜在语义索引 (Latent Semantic Indexing)
- 词嵌入表示 (Word Embedding)

# 独热表示

- 表示离散概念的简单方法
- 每个词对应一个向量，向量的维度等于词典的大小
- 向量中只有一个元素值为1，其余的元素均为0
- 值为1的元素对应的下标为该词在词典中的位置
- 例:

  V =  (哈尔滨, 冬天, 很, 美, 浪漫)        V表示词典

| | | |
|---|---|---|
| 哈尔滨 | = | [1 0 0 0 0] |
| 冬天 | = | [0 1 0 0 0] |
| 很 | = | [0 0 1 0 0] |
| 美 | = | [0 0 0 1 0] |
| 浪漫 | = | [0 0 0 0 1] |

- 如何区分词的重要性?

# 词频-逆文档频率(TF-IDF)

- 利用词频(Term Frequency)

$f_{ij}$ = frequency of term $t_i$ in document $d_j$

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

- 词频越高越重要吗?
  - 如何降低频率高但是却不重要的词的权重
  - 逆文档频率 (Inverse Document Frequency)

$n_i$ = number of docs that mention term $i$

$N$ = total number of docs

$$IDF_i = \log \frac{N}{n_i}$$

- TF-IDF score

$$W_{ij} = TF_{ij} \times IDF_i$$

## Tips: Karen Spärck Jones

- 26 August 1935 – 4 April 2007
- she introduced IDF in 1972. IDF is used in most search engines today, usually as part of the tf-idf weighting scheme.
- Spärck Jones, Karen (1972). "A statistical interpretation of term specificity and its application in retrieval". *Journal of Documentation* **28** (1): 11–21. doi:10.1108/eb026526.
- the ACL Lifetime Achievement Award (2004)

http://en.wikipedia.org/wiki/Karen_Sp%C3%A4rck_Jones

# 离散表示的问题

- 语义鸿沟 Semantic Gap
  - 无法表示相近语义
  - 例如：“电脑”和“计算机”的独热表示的相似度为0

    电脑 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

    计算机 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
- 维度大
  - 向量维度等于词表的大小
- 数据稀疏
  - 向量中绝大部分元素都是0值

## 分布式表示（Distributional representations）

- You shall know a word by the company it keeps (Firth, J. R. 1957:11)
  - 利用词的邻居（上下文）表示词
  - 统计自然语言处理最成功的思想之一

| | | |
|---|---|---|
| sugar, a sliced lemon, a tablespoonful of | **apricot** | jam, a pinch each of, |
| their enjoyment. Cautiously she sampled her first | **pineapple** | and another fruit whose taste she likened |
| well suited to programming on the digital | **computer**. | In finding the optimal R-stage policy from |
| for the purpose of gathering data and | **information** | necessary for the study authorized in the |

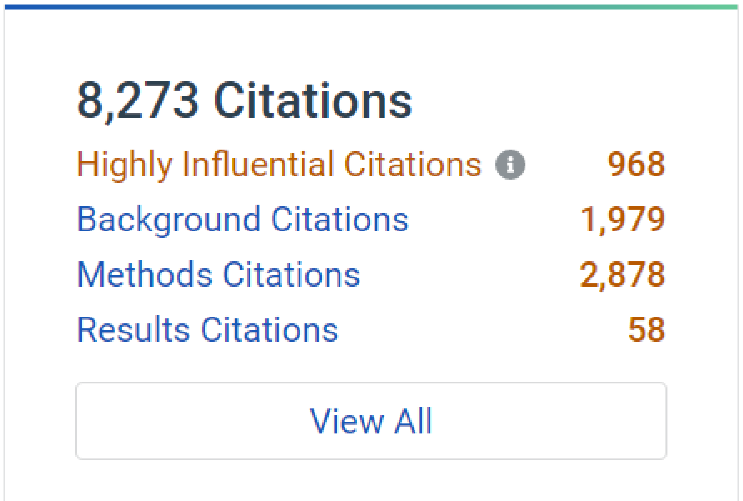
| | aardvark | computer | data | pinch | result | sugar | … |
|---|---|---|---|---|---|---|---|
| apricot | 0 | 0 | 0 | 1 | 0 | 1 | |
| pineapple | 0 | 0 | 0 | 1 | 0 | 1 | |
| digital | 0 | 2 | 1 | 0 | 1 | 0 | |
| information | 0 | 1 | 6 | 0 | 4 | 0 | |

## 分布式表示

- 目标：用固定维度的低维稠密向量来储存“大部分”重要信息
- 维度通常设定为 25 - 1000
- 如何降低维度?

# 潜在语义索引（Latent Semantic Indexing）

- 在一个向量空间（vector space）表示词
- 基于词与文档的共现（co-occurrence）关系学习词的特征向量
  - Deerwester, Scott C., Susan T. Dumais, George W. Furnas, Thomas K. Landauer and Richard A. Harshman. “Indexing by Latent Semantic Analysis.” Journal of the Association for Information Science and Technology, 41 (1990): 391-407.



8,273 Citations

| | |
|---|---|
| Highly Influential Citations | 968 |
| Background Citations | 1,979 |
| Methods Citations | 2,878 |
| Results Citations | 58 |

## Technical Memo Example

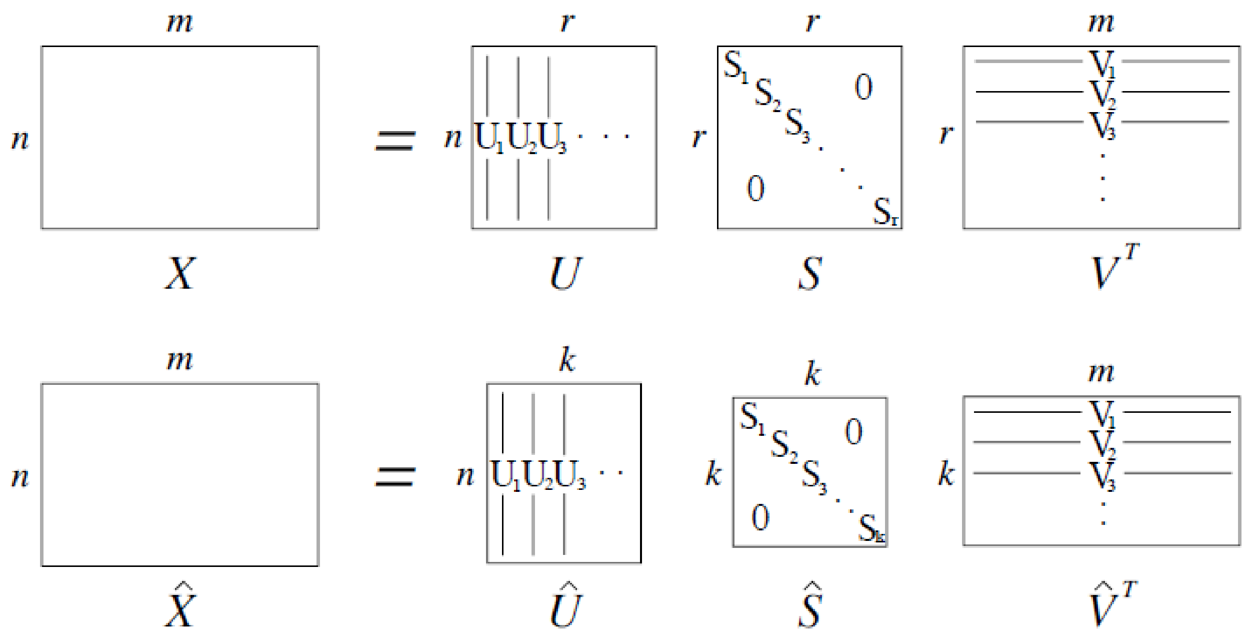
Titles

| | |
|---|---|
| c1: | *Human* machine *interface* for Lab ABC *computer* applications |
| c2: | A *survey of user* opinion of *computer system response time* |
| c3: | The *EPS user interface* management *system* |
| c4: | *System* and *human system* engineering testing of *EPS* |
| c5: | Relation of *user*-perceived *response time* to error measurement |
| m1: | The generation of random, binary, unordered *trees* |
| m2: | The intersection *graph* of paths in *trees* |
| m3: | *Graph minors* IV: Widths of *trees* and well-quasi-ordering |
| m4: | *Graph minors: A survey* |

| Terms | Documents | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | c1 | c2 | c3 | c4 | c5 | m1 | m2 | m3 | m4 |
| *human* | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| *interface* | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| *computer* | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *user* | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| *system* | 0 | 1 | 1 | 2 | 0 | 0 | 0 | 0 | 0 |
| *response* | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| *time* | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| *EPS* | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| *survey* | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| *trees* | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| *graph* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| *minors* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

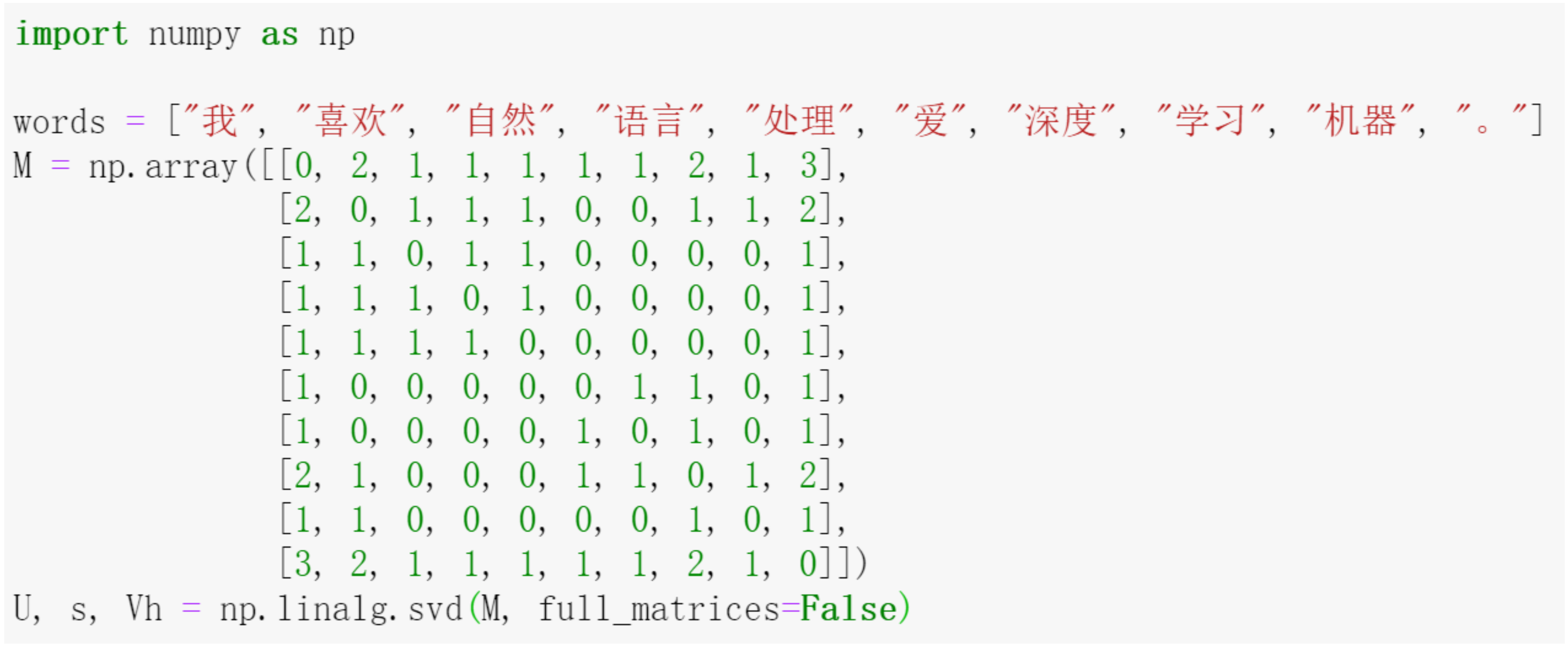
# 潜在语义索引

- Latent semantic indexing (LSI) is an indexing and retrieval method
  - **Uses singular value decomposition (SVD)** to identify patterns in the relationships between the terms and concepts contained in an unstructured collection of text.



# 用Python实现SVD分解的示例

corpus = “我 喜欢 自然 语言 处理 。
我 爱 深度 学习 。
我 喜欢 机器 学习 。"



```python
import numpy as np

words = ["我", "喜欢", "自然", "语言", "处理", "爱", "深度", "学习", "机器", "。"]
M = np.array([[0, 2, 1, 1, 1, 1, 1, 2, 1, 3],
              [2, 0, 1, 1, 1, 0, 0, 1, 1, 2],
              [1, 1, 0, 1, 1, 0, 0, 0, 0, 1],
              [1, 1, 1, 0, 1, 0, 0, 0, 0, 1],
              [1, 1, 1, 1, 0, 0, 0, 0, 0, 1],
              [1, 0, 0, 0, 0, 0, 1, 1, 0, 1],
              [1, 0, 0, 0, 0, 1, 0, 1, 0, 1],
              [2, 1, 0, 0, 0, 1, 1, 0, 1, 2],
              [1, 1, 0, 0, 0, 0, 0, 1, 0, 1],
              [3, 2, 1, 1, 1, 1, 1, 2, 1, 0]])
U, s, Vh = np.linalg.svd(M, full_matrices=False)
```

## 潜在语义索引的缺点

- 对于nxm 的矩阵，需要平方级（ quadratically ）时间复杂度：
  - O($mn^2$) flops (when n<m)
  - 当词或文档数量大（例如超过百万）时，速度会很慢
- 引入新词或新文档代价大

## 词嵌入表示 (Word Embedding)

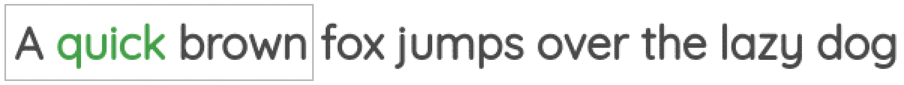
- 直接学习词的低维稠密的向量表示
  - Word2vec
    - Mikolov, Tomas, Kai Chen, Gregory S. Corrado and Jeffrey Dean. “Efficient Estimation of Word Representations in Vector Space.” ICLR (2013).
  - Glove
    - Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. EMNLP
  - Fasttext
    - Bojanowski Piotr, Grave Edouard, Joulin Armand and Mikolov Tomas. Enriching Word Vectors with Subword Information. TACL, , vol. 5, pp. 135–146,2017

# 词向量 Word2Vec

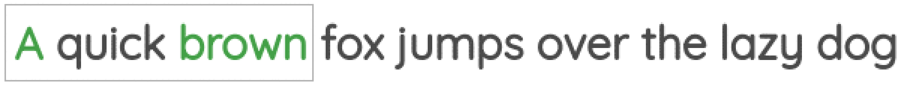
- Mikolov等于2013年提出(Google)
  - Mikolov, Tomas, et al. "Efficient Estimation of Word Representations in Vector Space." abs/1301.3781 (2013).
- 基本思想
  - You shall know a word by the company it keeps - J.R. Firth 1957
  - 直接计算词的表示，而不是通过计数的方法
- 采用自监督的方式
  - A word *c* that occurs near *apricot* in the corpus acts as the gold "correct answer" for supervised learning
  - No need for human labels

## 词向量 Word2Vec

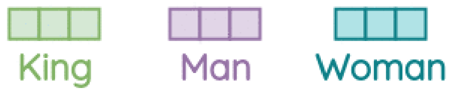
- 两类方法
  - Continuous Bag of Words（CBOW)



  - Skip-gram

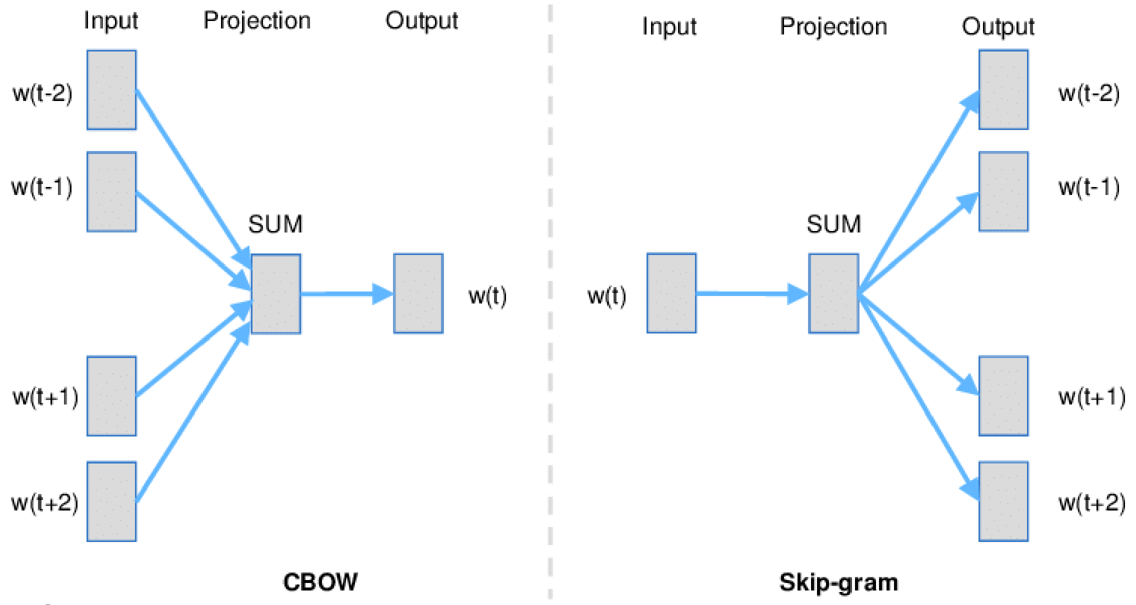


- 效果



## Word2Vec的实现

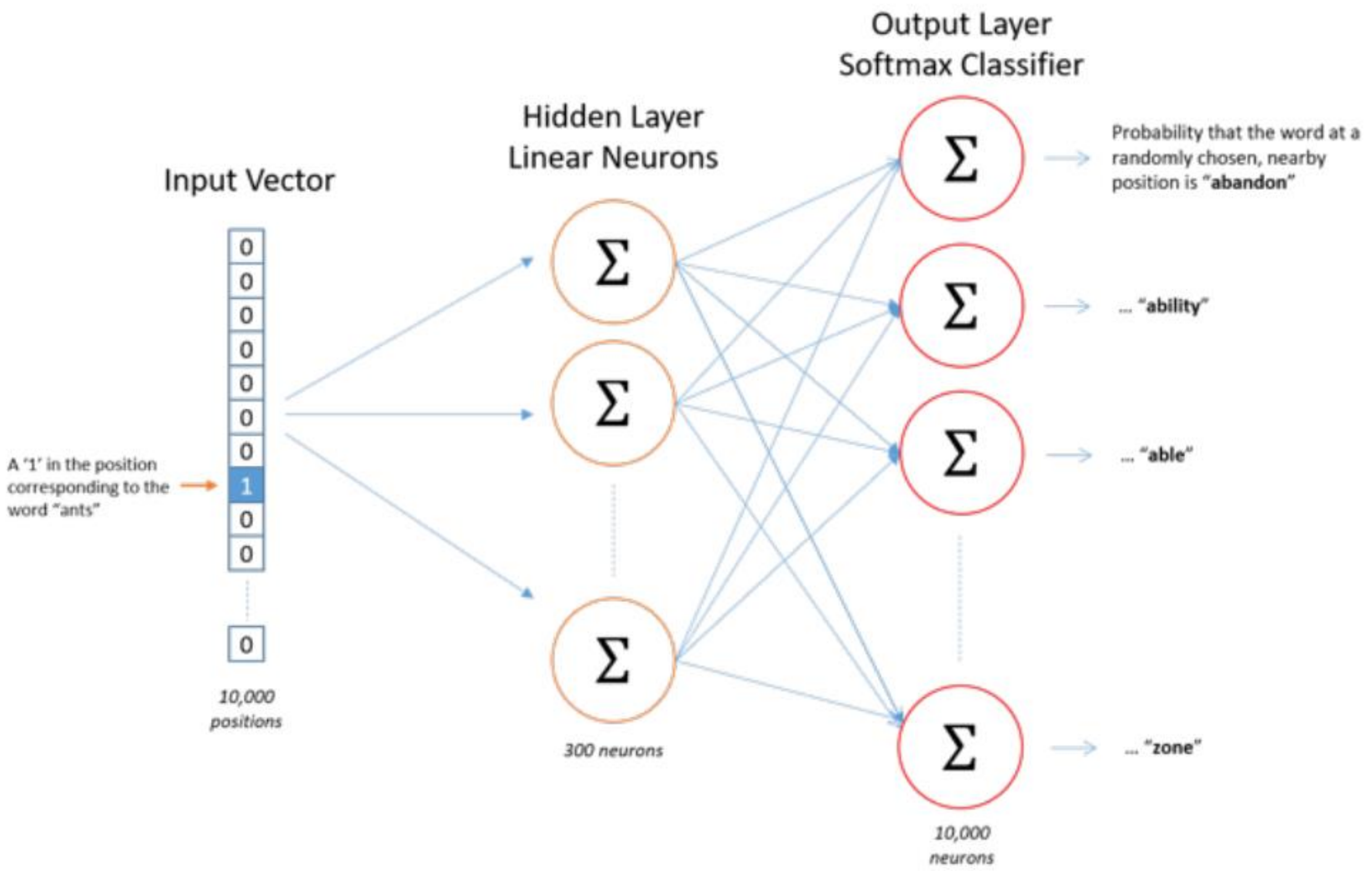
- Word2vec有很多种实现方法
  - CBOW、Skip-gram



  - 优化策略
    - 负采样、层次化softmax
  - 本课程重点介绍 “Skip-gram with Negative Sampling" (SGNS)

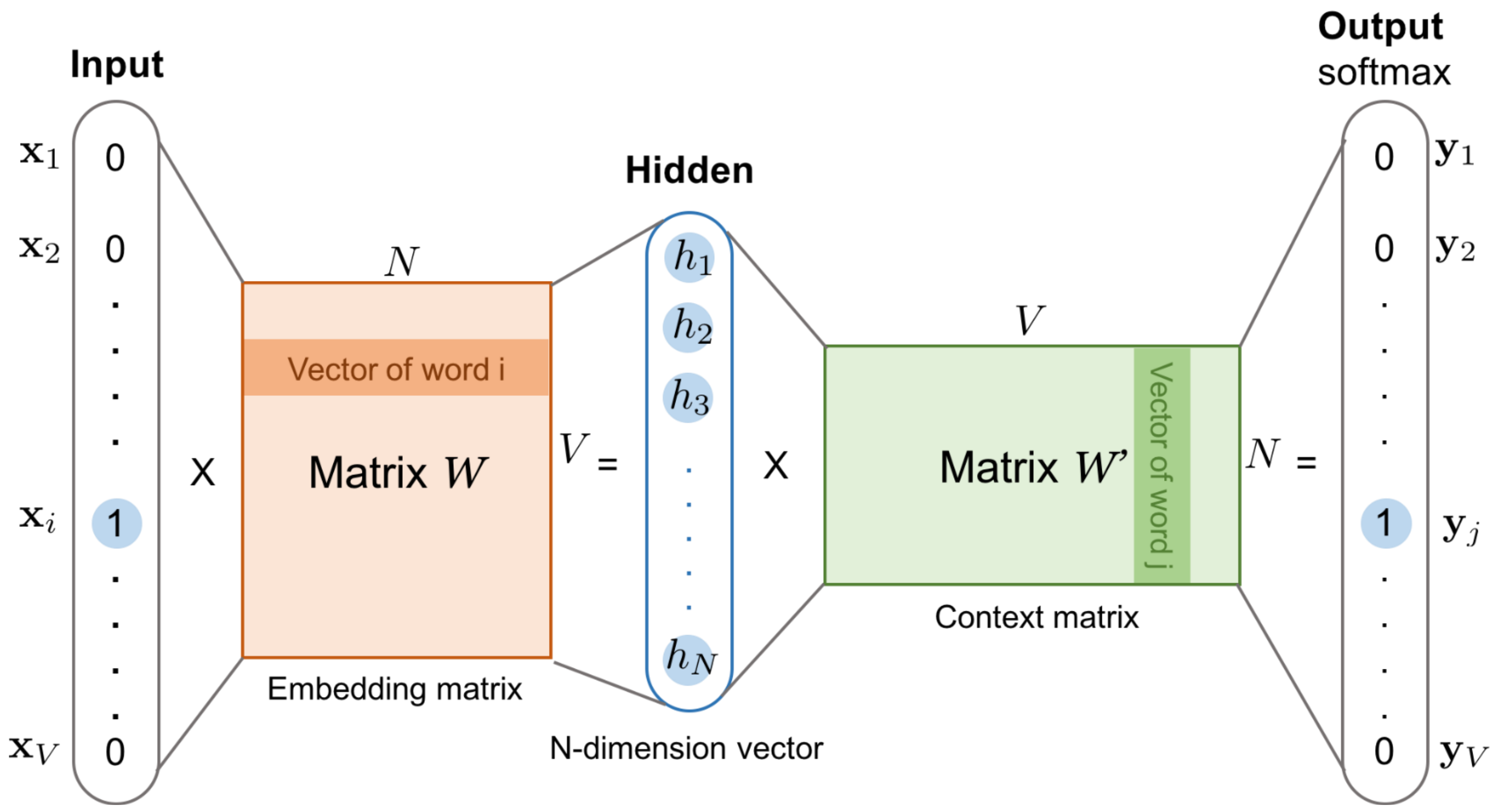
# Word2Vec: Skip-Gram

- 直接计算复杂度高。为什么？如何解决？



# Word2Vec: Skip-Gram

- 直接计算复杂度高。为什么？如何解决？



## Skip-gram with Negative Sampling (SGNS) 算法

- SGNS算法
  1. Treat the target word and a neighboring context word as positive examples.
  2. Randomly sample other words in the lexicon to get negative samples.
  3. Use logistic regression to train a classifier to distinguish those two cases.
  4. Use the weights as the embeddings.

## 如何构造SGNS训练数据中的正例

- Assume context words are those in +/- 2 word window
- Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

c1 c2 target c3 c4

**positive examples +**

| t | c |
|---|---|
| apricot | tablespoon |
| apricot | of |
| apricot | preserves |
| apricot | or |

## 如何构造SGNS训练数据中的负例

- Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

c1 c2 t c3 c4

**positive examples +**

| t | c |
|---|---|
| apricot | tablespoon |
| apricot | of |
| apricot | preserves |
| apricot | or |

- For each positive example, we'll create *k* negative examples.
  - Using *noise* words
    - Any random word that isn't *t*

## 如何构造SGNS训练数据中的负例

- Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

c1 c2 t c3 c4

k=2

**positive examples +**

| t | c |
|---|---|
| apricot | tablespoon |
| apricot | of |
| apricot | preserves |
| apricot | or |

**negative examples -**

| t | c | t | c |
|---|---|---|---|
| apricot | aardvark | apricot | twelve |
| apricot | puddle | apricot | hello |
| apricot | where | apricot | dear |
| apricot | coaxial | apricot | forever |

## 如何选择负例中的词

- Could pick w according to their unigram frequency P(w)
- More common to chosen then according to $p_\alpha(w)$

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_w count(w)^\alpha}$$

- α= ¾ works well because it gives rare noise words slightly higher probability
- To show this, imagine two events p(a)=.99 and p(b) = .01:

$$P_\alpha(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97$$
$$P_\alpha(b) = \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = .03$$

## SGNS目标

- Given a tuple (t,c) = target, context
  - (*apricot*, *jam*)
  - (*apricot*, *aardvark*)

- Return probability that c is a real context word:
  - $P(+|t,c)$
  - $P(-|t,c) = 1 - P(+|t,c)$

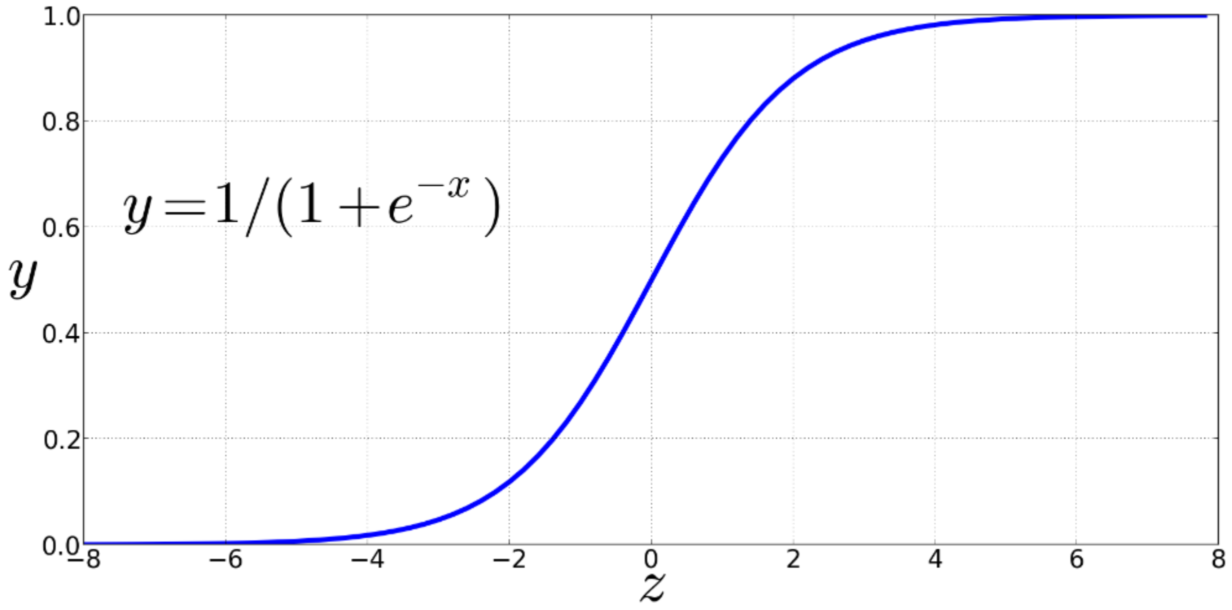
## 如何计算p(+|t,c)?

- Intuition:
  - Words are likely to appear near similar words
  - Model similarity with dot-product!
  - Similarity(t,c) $\propto t \cdot c$
- *Problem:*
  - *Dot-product is not a probability!*
    - *(Neither is cosine)*

## 如何把相似度转变成概率值?

- 使用Sigmoid函数
  - The sigmoid lies between 0 and 1:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



## 利用Sigmoid把相似度转变成概率值

$$P(+|t,c) = \frac{1}{1+e^{-t\cdot c}}$$

$$\begin{aligned} P(-|t,c) &= 1-P(+|t,c) \\ &= \frac{e^{-t\cdot c}}{1+e^{-t\cdot c}} \end{aligned}$$

## SGNS中分类器的设置

- Let's represent words as vectors of some length (say 300), randomly initialized.
  - So we start with 300 * V random parameters
- Over the entire training set, we’d like to adjust those word vectors such that we
  - Maximize the similarity of the target word, context word pairs (t,c) drawn from the positive data
  - Minimize the similarity of the (t,c) pairs drawn from the negative data.

## SGNS中分类器的训练过程

- Iterative process.
- start with 0 or random weights
- Then adjust the word weights to
  - make the positive pairs more likely
  - and the negative pairs less likely
  - over the entire training set

## SGNS中分类器训练的优化目标

- We want to maximize…

$$\sum_{(t,c)\in +} logP(+|t,c) + \sum_{(t,c)\in -} logP(-|t,c)$$

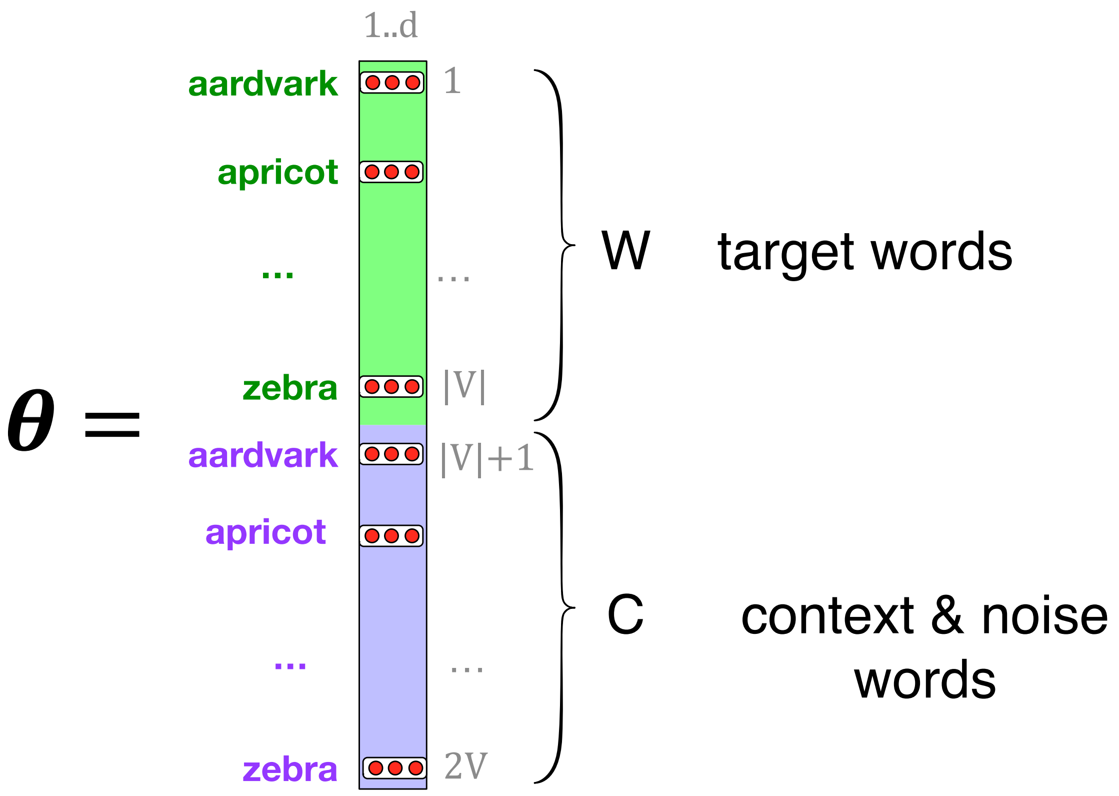
- Maximize the + label for the pairs from the positive training data, and the – label for the pairs sample from the negative data.

## SGNS中分类器训练的优化目标

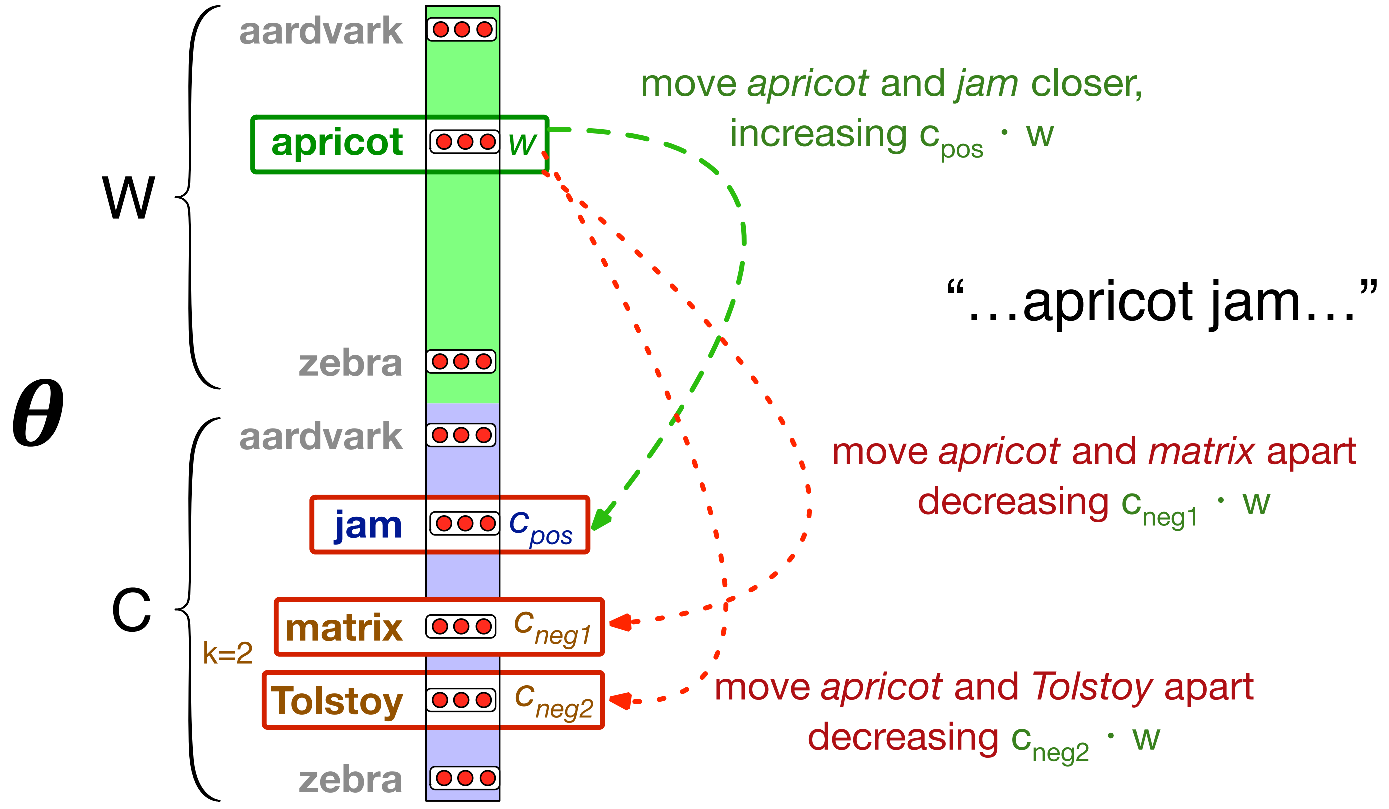
- Focusing on one target word t:

$$
\begin{aligned}
L(\theta) &= \log P(+|t,c) + \sum_{i=1}^{k} \log P(-|t,n_i) \\
&= \log \sigma(c\cdot t) + \sum_{i=1}^{k} \log \sigma(-n_i\cdot t) \\
&= \log \frac{1}{1+e^{-c\cdot t}} + \sum_{i=1}^{k} \log \frac{1}{1+e^{n_i\cdot t}}
\end{aligned}
$$

# SGNS使用梯度下降方法进行训练



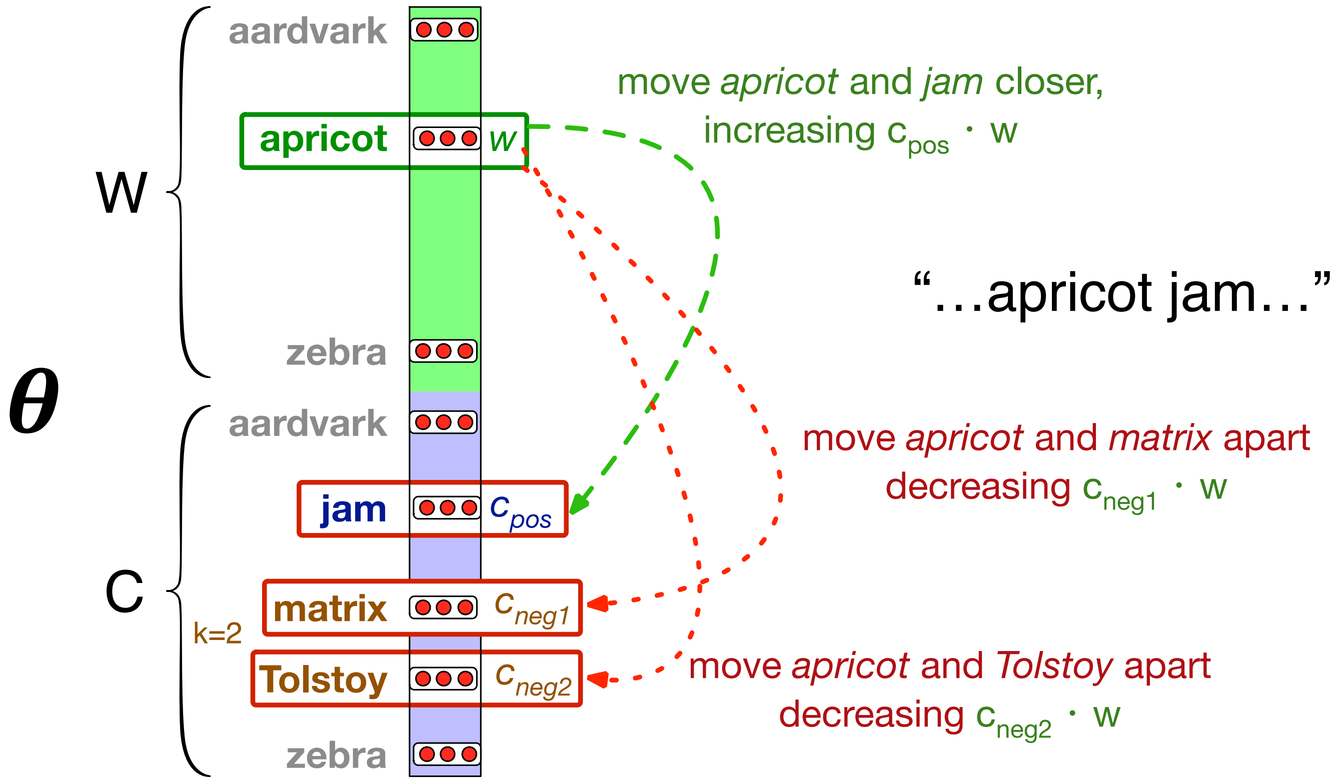
# SGNS使用梯度下降方法进行训练



W
aardvark
apricot
*w*
zebra
$\theta$
C
aardvark
jam
$c_{pos}$
k=2
matrix
$c_{neg1}$
Tolstoy
$c_{neg2}$
zebra
move *apricot* and *jam* closer, increasing $c_{pos} \cdot w$
"…apricot jam…"
move *apricot* and *matrix* apart decreasing $c_{neg1} \cdot w$
move *apricot* and *Tolstoy* apart decreasing $c_{neg2} \cdot w$

# SGNS使用梯度下降方法进行训练

- Intuition of one step of gradient descent

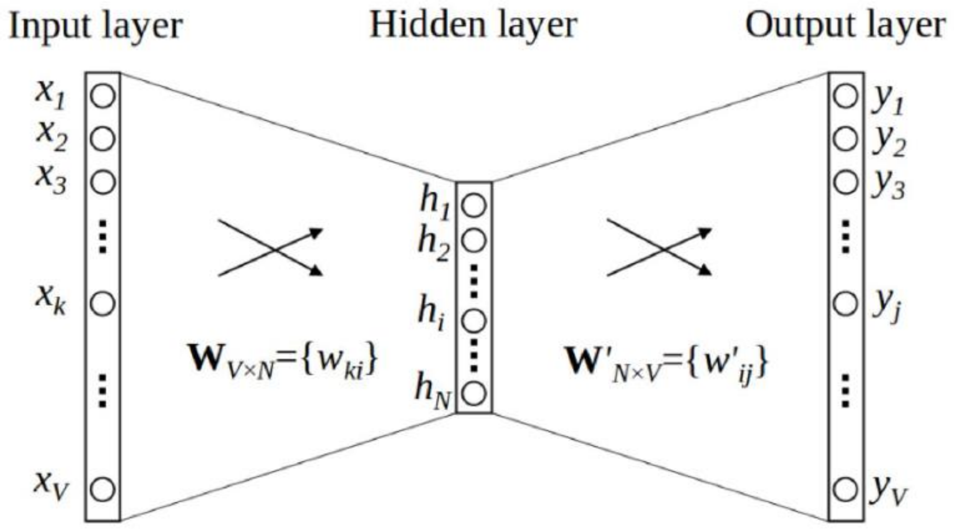


W
aardvark
apricot
$w$
zebra
$\theta$
C
aardvark
jam
$c_{pos}$
k=2
matrix
$c_{neg1}$
Tolstoy
$c_{neg2}$
zebra
move *apricot* and *jam* closer, increasing $c_{pos} \cdot w$
"…apricot jam…"
move *apricot* and *matrix* apart decreasing $c_{neg1} \cdot w$
move *apricot* and *Tolstoy* apart decreasing $c_{neg2} \cdot w$

## SGNS使用梯度下降方法进行训练

- Actually learns two separate embedding matrices W and C
- Can use W and throw away C, or merge them somehow

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



## SGNS总结

- Start with V random 300-dimensional vectors as initial embeddings
- Use logistic regression, the second most basic classifier used in machine learning after naïve bayes
  - Take a corpus and take pairs of words that co-occur as positive examples
  - Take pairs of words that don't co-occur as negative examples
  - Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
  - Throw away the classifier code and keep the embeddings.

## 词向量的评价方法

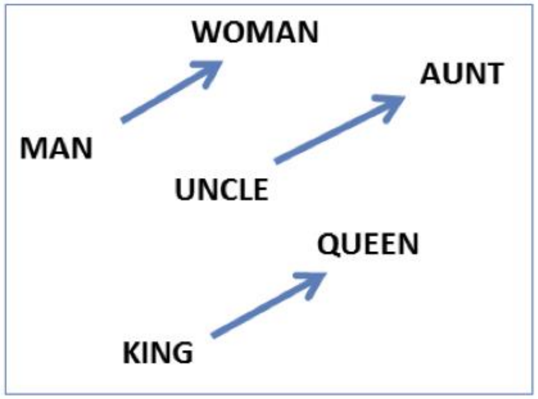
- Compare to human scores on word similarity-type tasks:
  - Data set
    - WordSim-353 (Finkelstein et al., 2002)
    - SimLex-999 (Hill et al., 2015)
    - Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
    - TOEFL dataset: *Levied is closest in meaning to: imposed, believed, requested, correlated*
  - score
    - Spearman’ s rank correlation coefficient

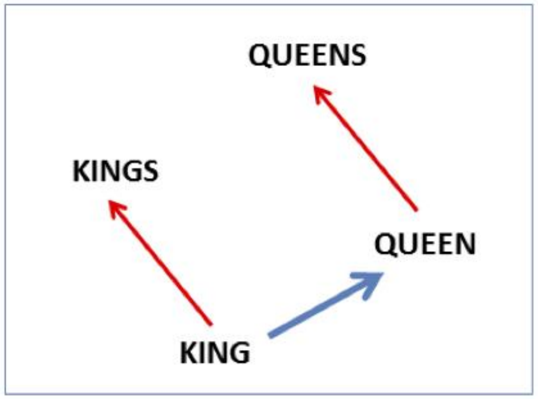
## 词向量的评价方法

- Analogy: Embeddings capture relational meaning!

vector( *'king'* ) - vector( *'man'* ) + vector( *'woman'* ) ≈ vector( 'queen' )

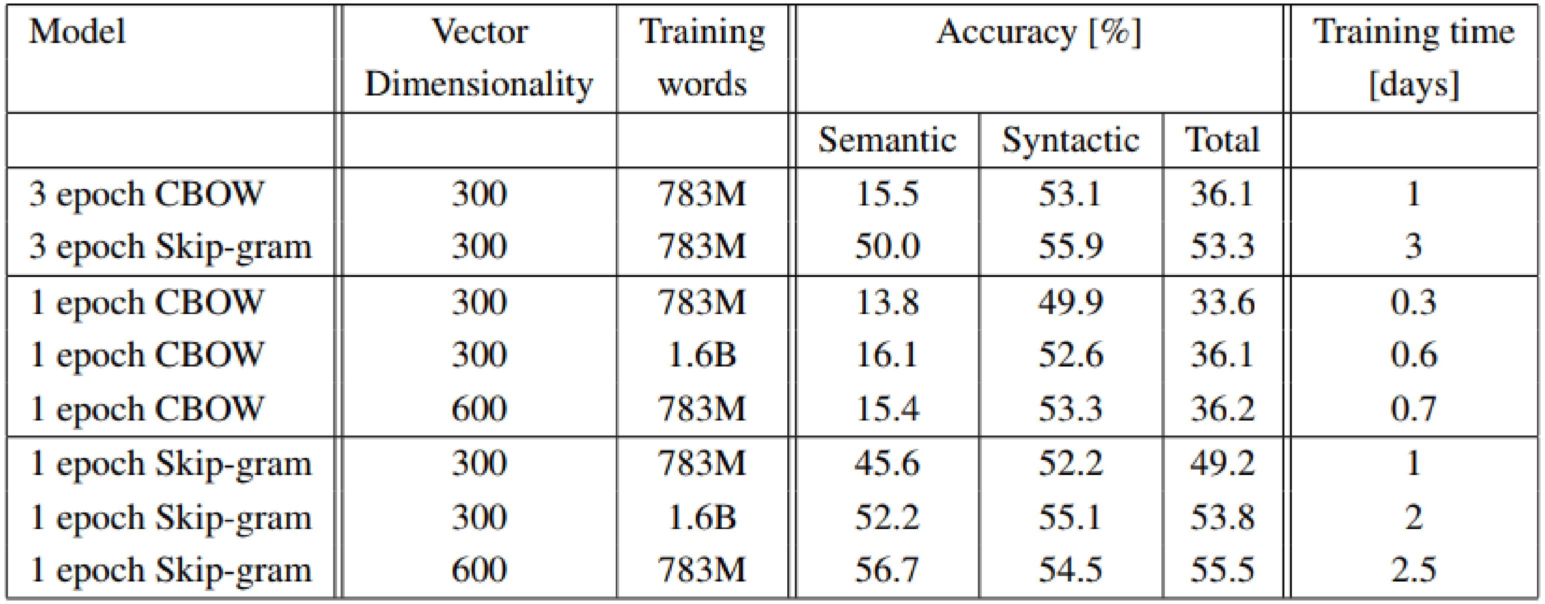
vector( *'Paris'* ) - vector( *'France'* ) + vector( *'Italy'* ) ≈ vector( 'Rome' )





# Word2Vec的评价结果

- Test set contains five types of semantic questions, and nine types of syntactic questions.
- There are 8869 semantic and 10675 syntactic questions.

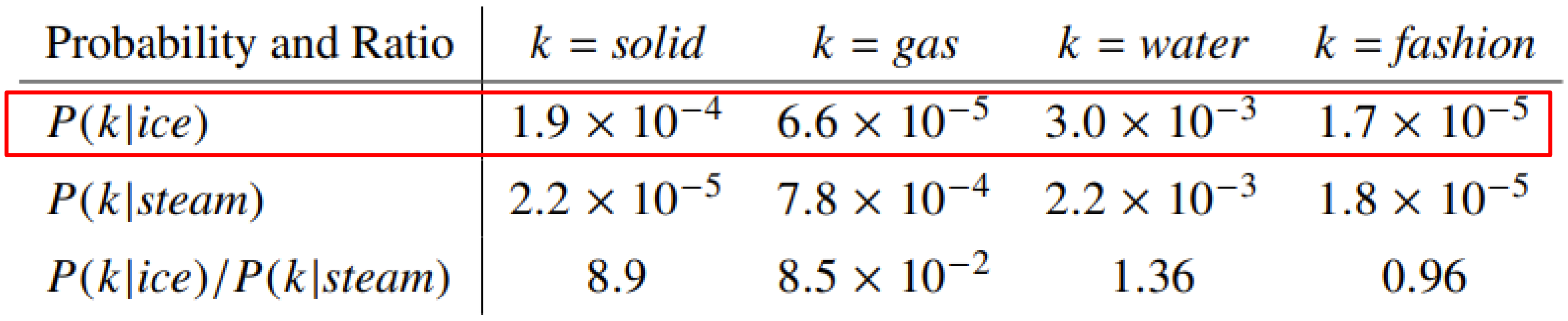


| Model | Vector Dimensionality | Training words | Accuracy [%] | | | Training time [days] |
|---|---|---|---|---|---|---|
| | | | Semantic | Syntactic | Total | |
| 3 epoch CBOW | 300 | 783M | 15.5 | 53.1 | 36.1 | 1 |
| 3 epoch Skip-gram | 300 | 783M | 50.0 | 55.9 | 53.3 | 3 |
| 1 epoch CBOW | 300 | 783M | 13.8 | 49.9 | 33.6 | 0.3 |
| 1 epoch CBOW | 300 | 1.6B | 16.1 | 52.6 | 36.1 | 0.6 |
| 1 epoch CBOW | 600 | 783M | 15.4 | 53.3 | 36.2 | 0.7 |
| 1 epoch Skip-gram | 300 | 783M | 45.6 | 52.2 | 49.2 | 1 |
| 1 epoch Skip-gram | 300 | 1.6B | 52.2 | 55.1 | 53.8 | 2 |
| 1 epoch Skip-gram | 600 | 783M | 56.7 | 54.5 | 55.5 | 2.5 |

Mikolov, Tomas, et al. "Efficient Estimation of Word Representations in Vector Space." abs/1301.3781 (2013).

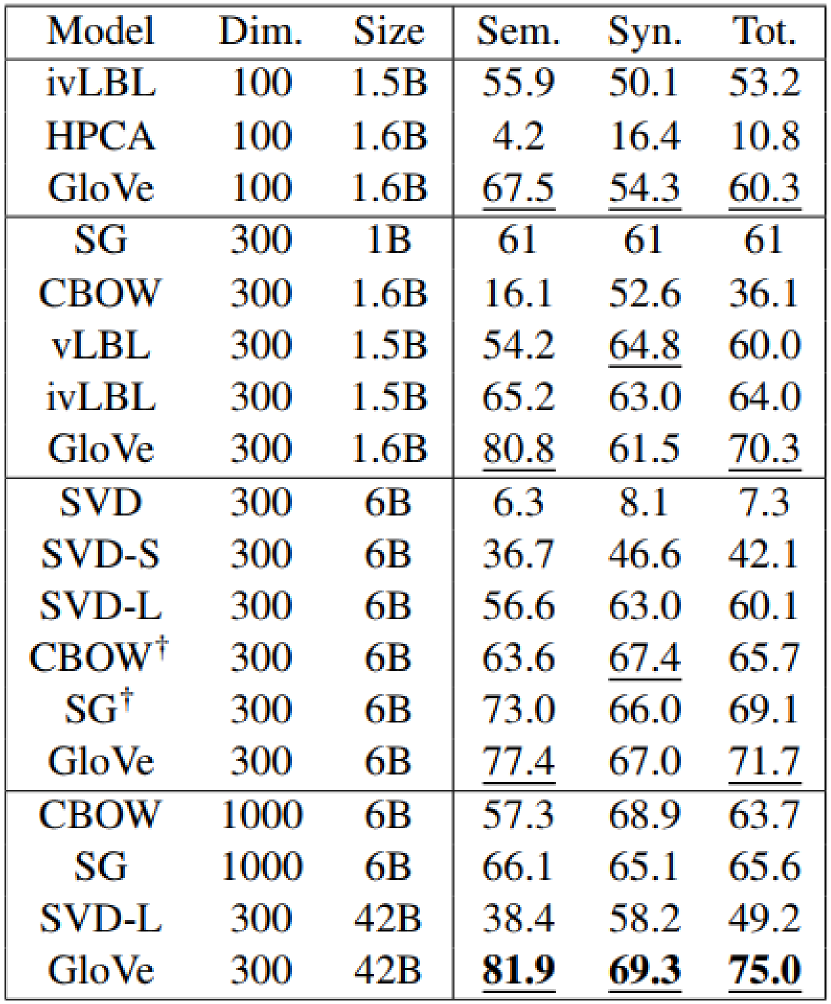
## Glove词向量

- Jeffrey Pennington等2014年提出(Stanford University)
  - Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. EMNLP
- 重要发现：Ratios of co-occurrence probabilities can encode meaning components



| Probability and Ratio | $k = solid$ | $k = gas$ | $k = water$ | $k = fashion$ |
|---|---|---|---|---|
| $P(k\|ice)$ | $1.9 \times 10^{-4}$ | $6.6 \times 10^{-5}$ | $3.0 \times 10^{-3}$ | $1.7 \times 10^{-5}$ |
| $P(k\|steam)$ | $2.2 \times 10^{-5}$ | $7.8 \times 10^{-4}$ | $2.2 \times 10^{-3}$ | $1.8 \times 10^{-5}$ |
| $P(k\|ice)/P(k\|steam)$ | 8.9 | $8.5 \times 10^{-2}$ | 1.36 | 0.96 |

# Glove词向量的评价结果



| Model | Dim. | Size | Sem. | Syn. | Tot. |
|---|---|---|---|---|---|
| ivLBL | 100 | 1.5B | 55.9 | 50.1 | 53.2 |
| HPCA | 100 | 1.6B | 4.2 | 16.4 | 10.8 |
| GloVe | 100 | 1.6B | 67.5 | 54.3 | 60.3 |
| SG | 300 | 1B | 61 | 61 | 61 |
| CBOW | 300 | 1.6B | 16.1 | 52.6 | 36.1 |
| vLBL | 300 | 1.5B | 54.2 | 64.8 | 60.0 |
| ivLBL | 300 | 1.5B | 65.2 | 63.0 | 64.0 |
| GloVe | 300 | 1.6B | 80.8 | 61.5 | 70.3 |
| SVD | 300 | 6B | 6.3 | 8.1 | 7.3 |
| SVD-S | 300 | 6B | 36.7 | 46.6 | 42.1 |
| SVD-L | 300 | 6B | 56.6 | 63.0 | 60.1 |
| CBOW$^\dagger$ | 300 | 6B | 63.6 | 67.4 | 65.7 |
| SG$^\dagger$ | 300 | 6B | 73.0 | 66.0 | 69.1 |
| GloVe | 300 | 6B | 77.4 | 67.0 | 71.7 |
| CBOW | 1000 | 6B | 57.3 | 68.9 | 63.7 |
| SG | 1000 | 6B | 66.1 | 65.1 | 65.6 |
| SVD-L | 300 | 42B | 38.4 | 58.2 | 49.2 |
| GloVe | 300 | 42B | **81.9** | **69.3** | **75.0** |

Results on the word analogy task, given as percent accuracy. Underlined scores are best within groups of similarly-sized models; bold scores are best overall.

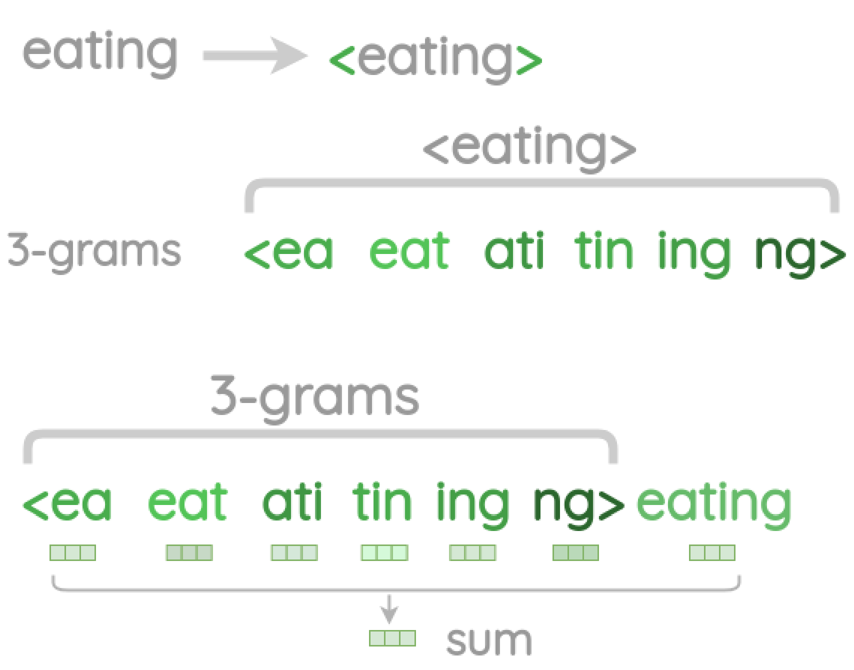
# Fasttext 词向量

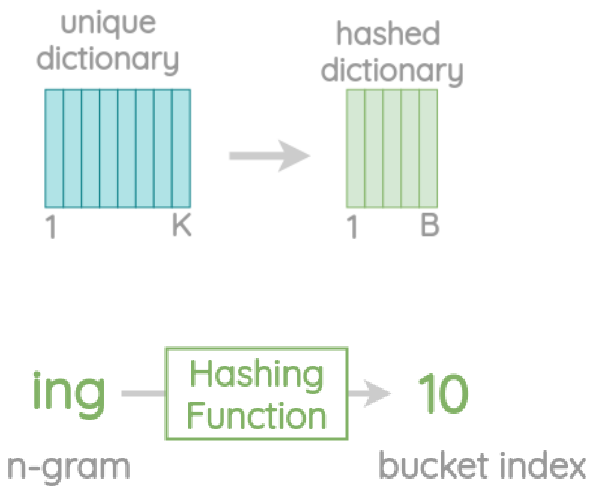
- Bojanowski等2017年提出(Facebook AI Research)
  - Bojanowski Piotr, Grave Edouard, Joulin Armand and Mikolov Tomas. Enriching Word Vectors with Subword Information. TACL, , vol. 5, pp. 135–146,2017
- Fasttext
  - 考虑了单词的形态
    - 利用单词包含的字符的n-gram组合来表示一个词
  - 可以解决未登录词问题
  - 基于Skip gram模型

https://amitness.com/2020/06/fasttext-embeddings/

# Fasttext 词向量

- 利用字符n-gram组合表示单词的例子

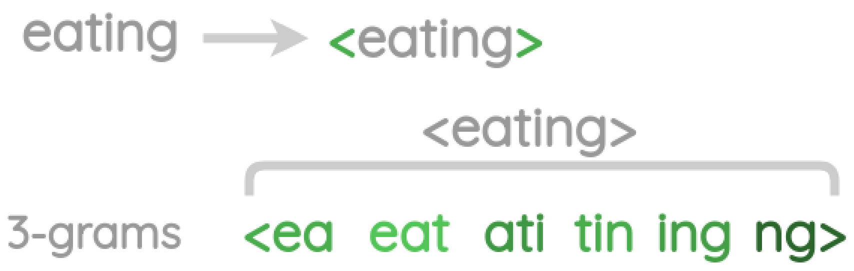


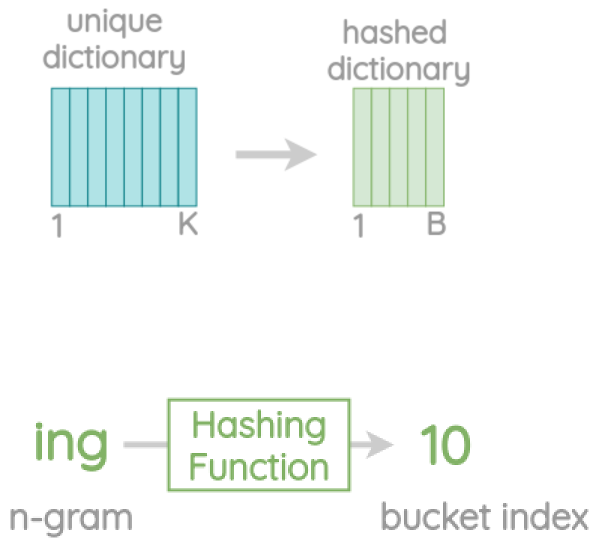


https://amitness.com/2020/06/fasttext-embeddings/

# Fasttext 词向量

- 利用字符n-gram组合表示单词的例子



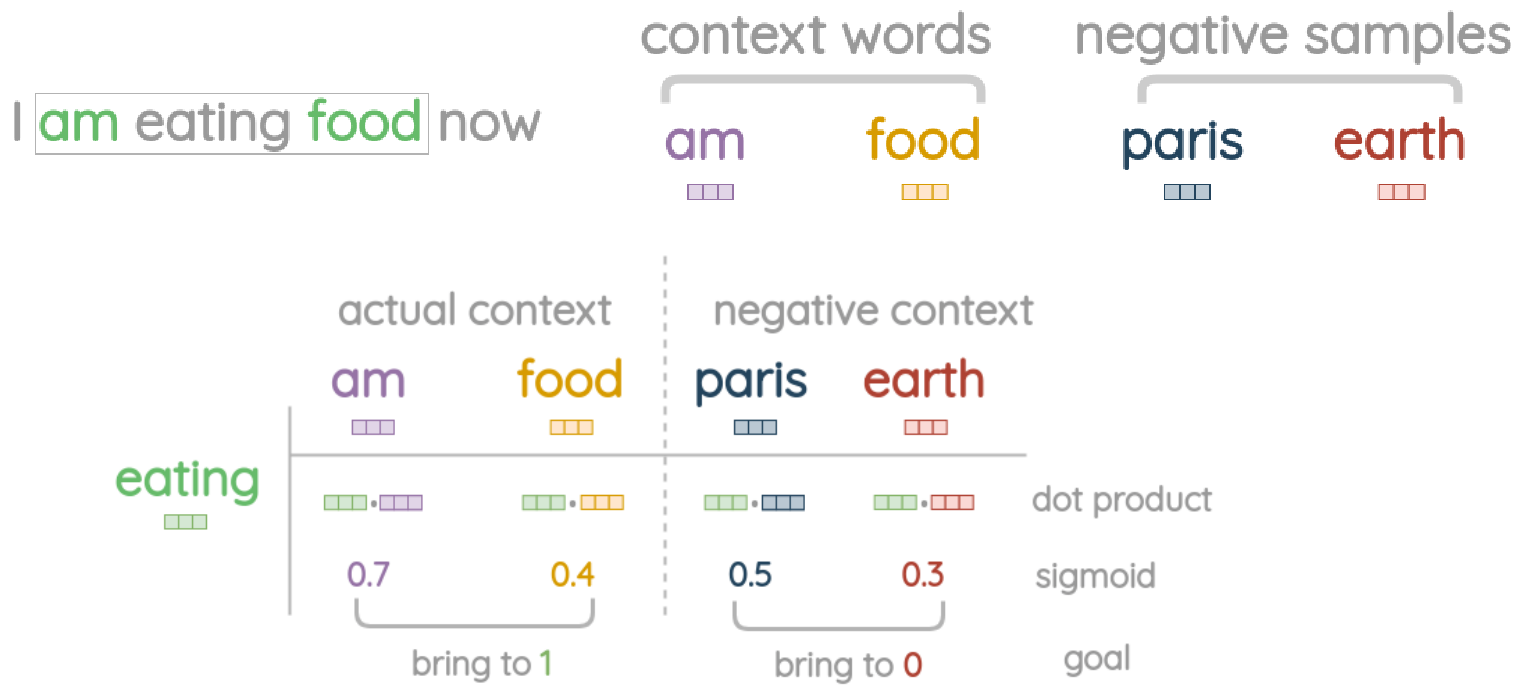


| Word | Length(n) | Character n-grams |
|---|---|---|
| eating | 3 | <ea, eat, ati, tin, ing, ng> |
| eating | 4 | <eat, eati, atin, ting, ing> |
| eating | 5 | <eati, eatin, ating, ting> |
| eating | 6 | <eatin, eating, ating> |

https://amitness.com/2020/06/fasttext-embeddings/

# Fasttext 词向量

## ☐负采样例子



https://amitness.com/2020/06/fasttext-embeddings/

## fasttext词向量的评价结果

- 能够显著提高句法单词类比任务的性能
  - 形态学丰富的语言，如捷克语和德语

| Singular/plural | | | Base/Comparative | | |
|---|---|---|---|---|---|
| cat | → | cats | good | → | better |
| dog | → | ? | rough | → | ? |

| | word2vec-skipgram | word2vec-cbow | fasttext |
|---|---|---|---|
| Czech | 52.8 | 55.0 | 77.8 |
| German | 44.5 | 45.0 | 56.4 |
| English | 70.1 | 69.9 | 74.9 |
| Italian | 51.5 | 51.8 | 62.7 |

https://amitness.com/2020/06/fasttext-embeddings/

# fasttext词向量的评价结果

- 在语义类比任务上性能下降

$$\begin{tikzcd}
\text{man} \arrow[r] & \text{king} \\
\text{woman} \arrow[r] & ?
\end{tikzcd}$$

| | word2vec-skipgram | word2vec-cbow | fasttext |
|---|---|---|---|
| Czech | 25.7 | 27.6 | 27.5 |
| German | 66.5 | 66.8 | 62.3 |
| English | 78.5 | 78.2 | 77.8 |
| Italian | 52.3 | 54.7 | 52.3 |

https://amitness.com/2020/06/fasttext-embeddings/

# fasttext词向量的评价结果

- 在词相似度任务上，具有更好的性能

| | | skipgram | cbow | fasttext(null OOV) | fasttext(char-ngrams for OOV) |
|---|---|---|---|---|---|
| Arabic | WS353 | 51 | 52 | 54 | 55 |
| | GUR350 | 61 | 62 | 64 | 70 |
| German | GUR65 | 78 | 78 | 81 | 81 |
| | ZG222 | 35 | 38 | 41 | 44 |
| English | RW | 43 | 43 | 46 | 47 |
| | WS353 | 72 | 73 | 71 | 71 |
| Spanish | WS353 | 57 | 58 | 58 | 59 |
| French | RG65 | 70 | 69 | 75 | 75 |
| Romanian | WS353 | 48 | 52 | 51 | 54 |
| Russian | HJ | 59 | 60 | 60 | 66 |

https://amitness.com/2020/06/fasttext-embeddings/

## 词向量相关的资源下载

- **Word2vec**
  - https://code.google.com/archive/p/word2vec/
  - http://word2vec.googlecode.com/svn/trunk/
  - http://radimrehurek.com/gensim/
  - https://github.com/NLPchina/Word2VEC_java
- **Glove**
  - http://nlp.stanford.edu/projects/glove/
- **Fasttext**
  - http://www.fasttext.cc/

## 词嵌入方法的问题

- 静态词向量
  - 词向量无法随语境变化
- 不能处理一词多义
  - 多义词无法区分多个含义
- 不能有效区分反义词
  - 反义词的上下文很相似
    - 如：冷和热、good和bad

## 词嵌入方法总结

- These word vectors can be subsequently used as features in many NLP tasks
- As word vectors can be trained on huge text datasets, they provide generalization for systems trained with limited amount of supervised data
- More complex model architectures can be used for obtaining the word vectors

## 主要内容

- 词的表示
- 文档表示
- 文本相似度计算

## 文档表示

- 词袋表示（Bag of words)
- 潜在语义索引 (Latent Semantic Indexing)
- 主题模型 (Topic model)
- 基于深度学习的文档表示

## 词袋表示（Bag of words)

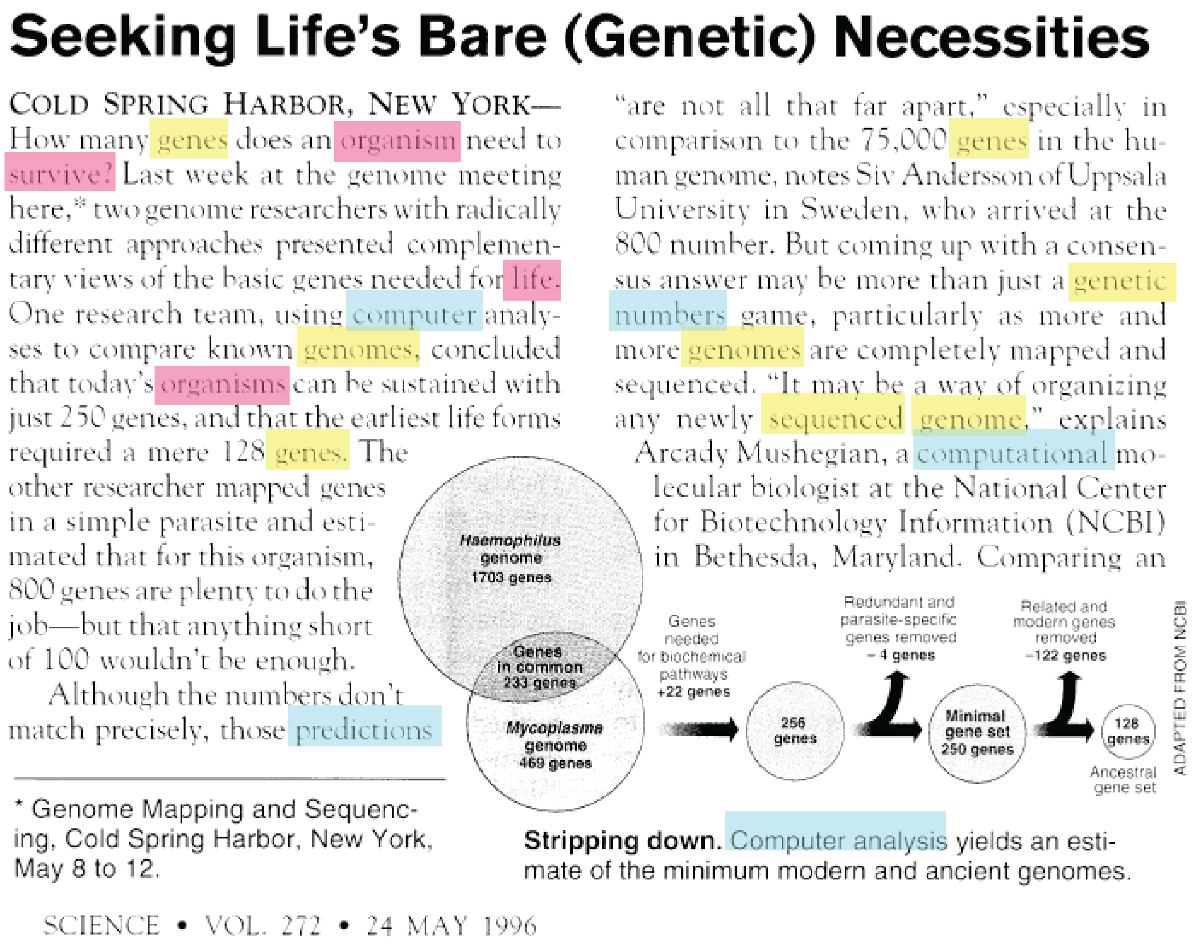
- 用文档中每个词的独热表示的和表示文档
- 忽略了文档中词的顺序
- 例:

```
V =  (哈尔滨, 冬天, 很, 美, 浪漫)
哈尔滨 哈尔滨          = [2 0 0 0 0]
哈尔滨 冬天 很 浪漫 = [1 1 1 0 1]
哈尔滨 冬天 美        = [1 1 0 1 0]
```

## 词袋表示（Bag of words)

- Words are features/index
- Weight of features/index
  - Binary value
    - 1: occurrence      0:otherwise
  - Real value
    - TF
    - TF*IDF (TF: term frequency, IDF: inverse document frequency)
- Vectorization
  - document->vector

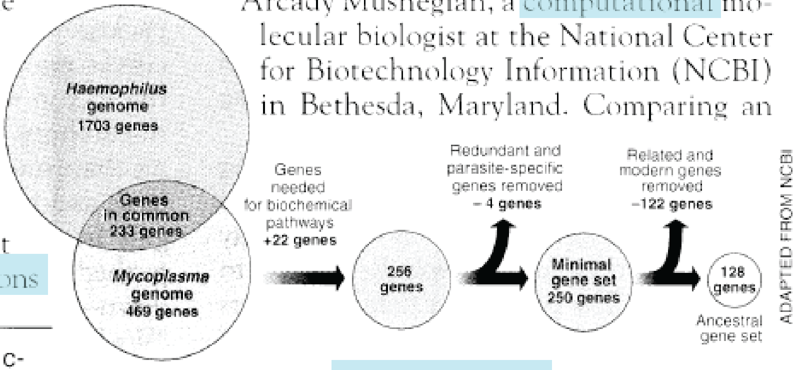
# 主题模型（Topic model)



## Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed –4 genes
Minimal gene set 250 genes
Related and modern genes removed –122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI

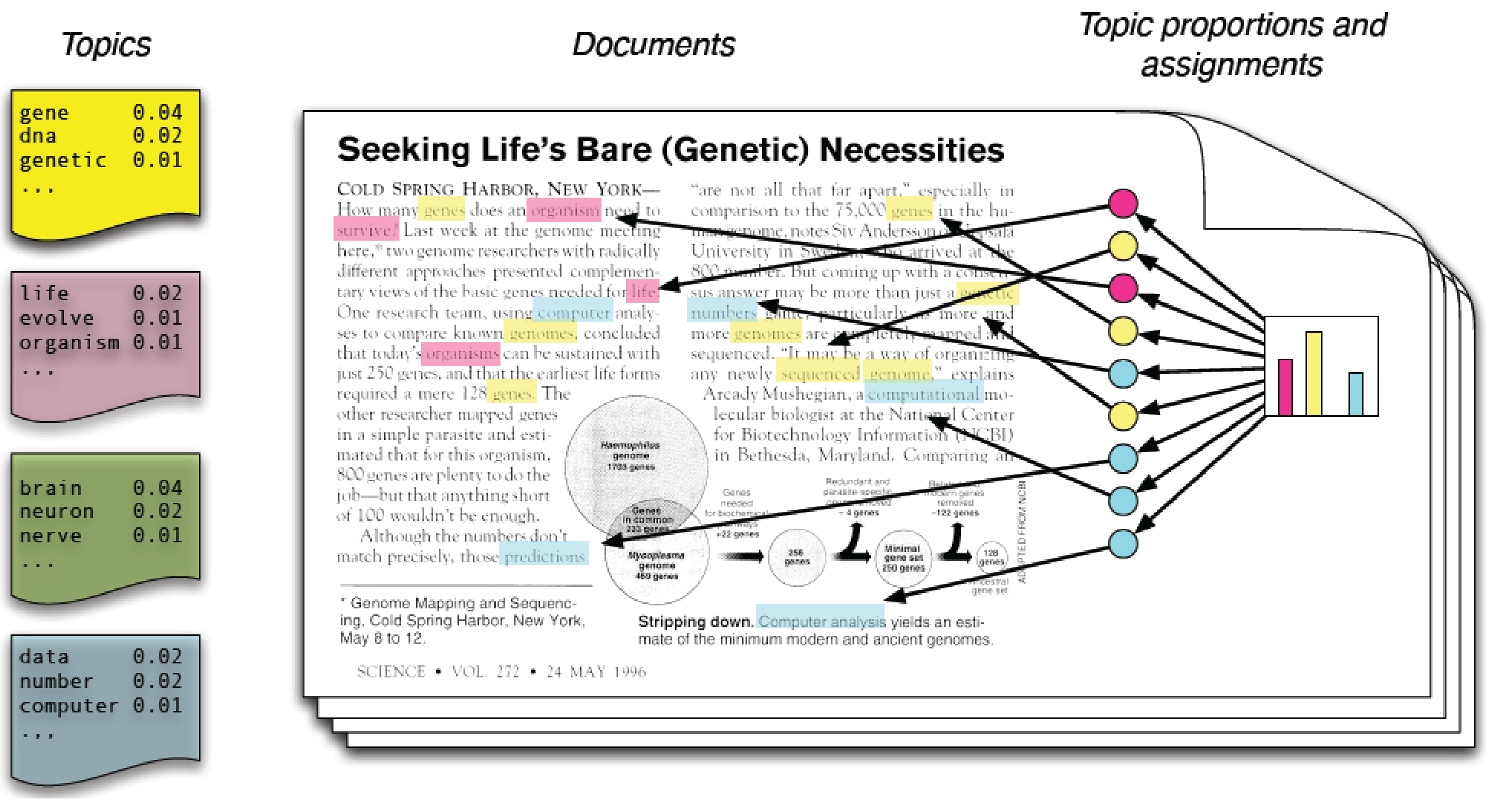
**Stripping down.** Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996

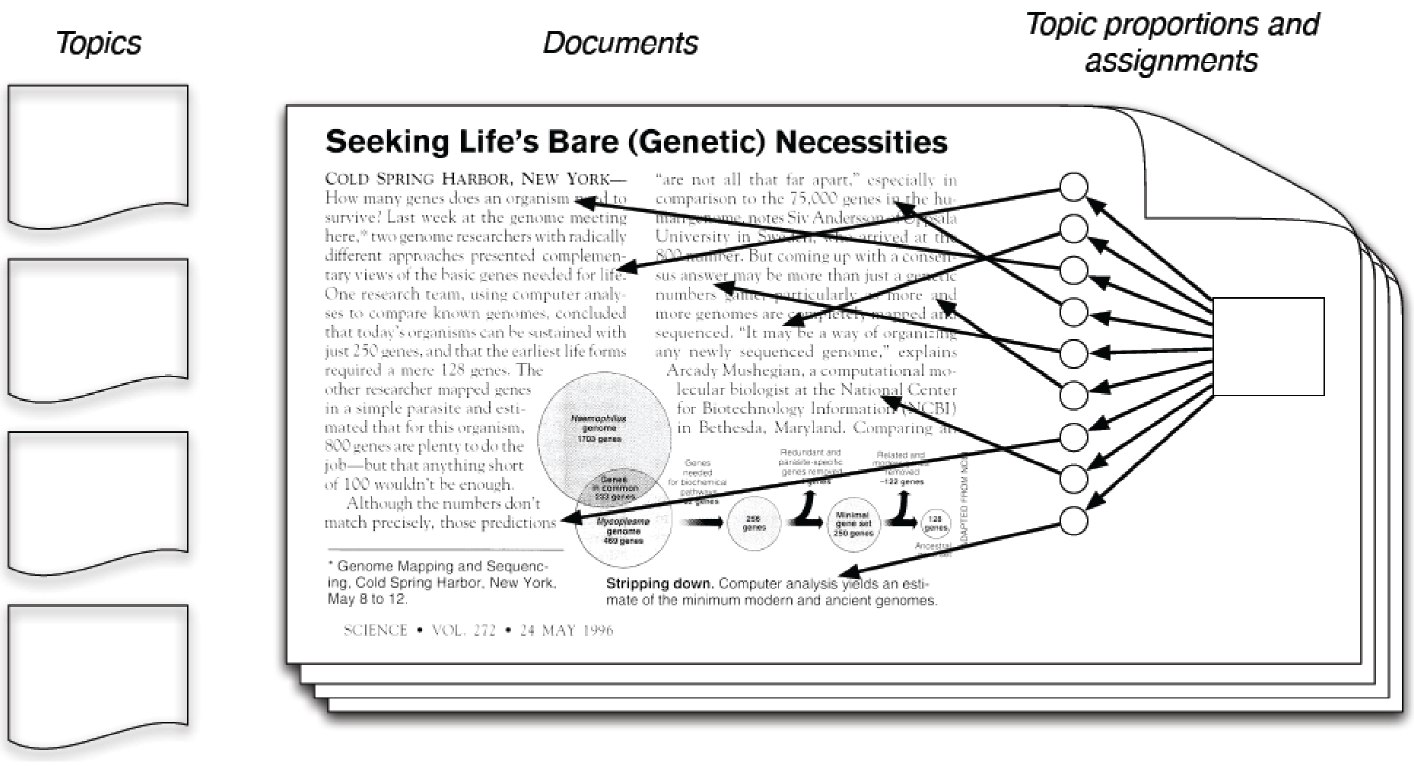
**Simple intuition:** Documents exhibit multiple topics.

# 主题模型



- Each topic is a distribution over words
- Each document is a mixture of corpus-wide topics
- Each word is drawn from one of those topics

# 主题模型


Topics
Documents
Topic proportions and assignments
Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways
256 genes
Redundant and parasite-specific genes removed
Minimal gene set 250 genes
Related and modified genes removed ~122 genes
128 genes
Ancestral
ADAPTED FROM NCBI

**Seeking Life's Bare (Genetic) Necessities**

COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

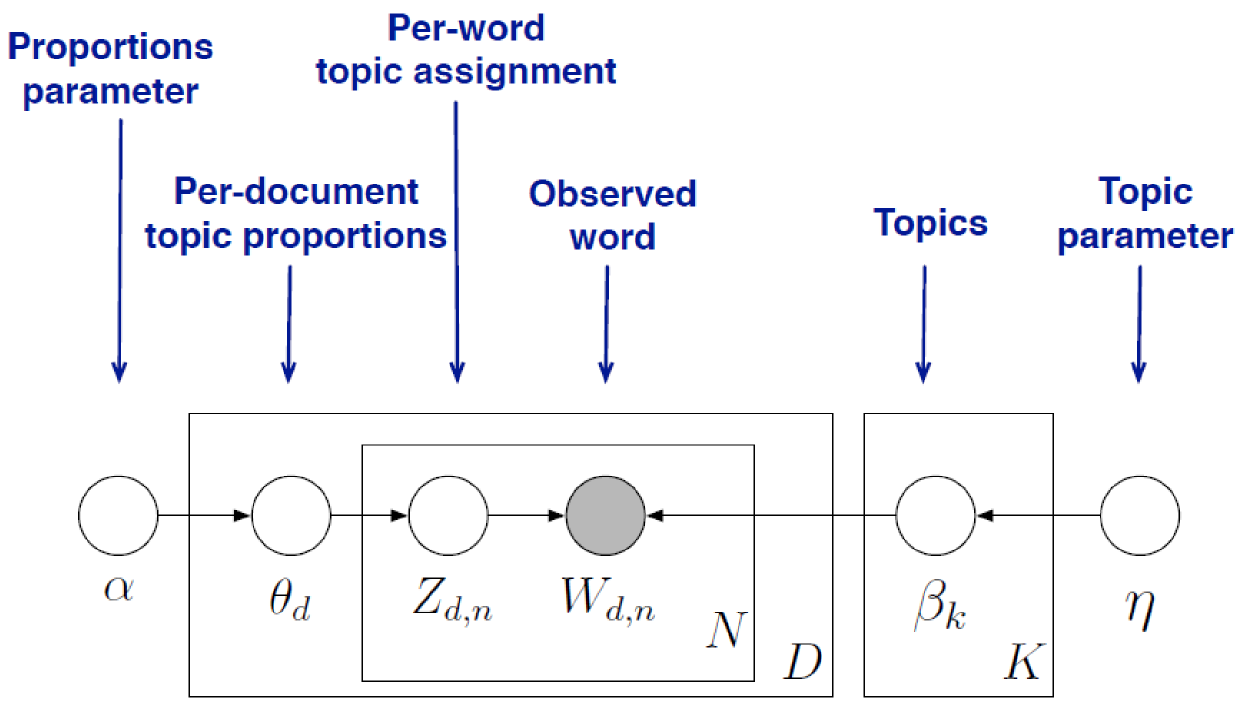
**Stripping down.** Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996

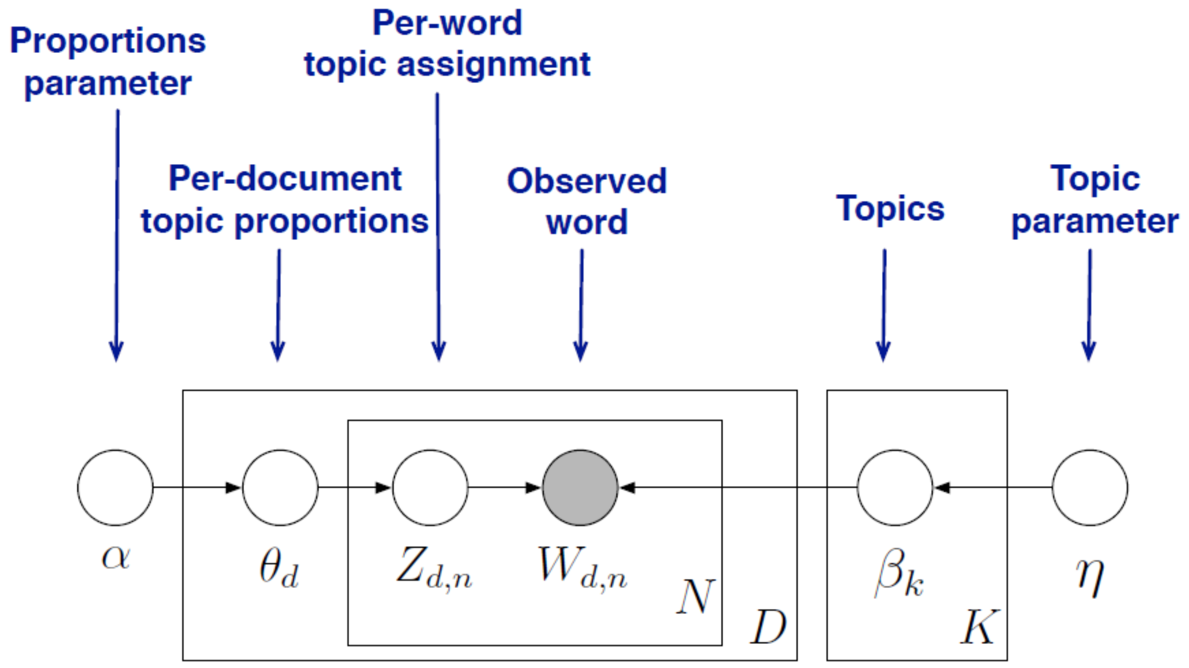
- Our goal is to **infer** the hidden variables
  - compute their distribution conditioned on the documents

# 潜在狄利克雷分布 Latent Dirichlet Allocation (LDA)



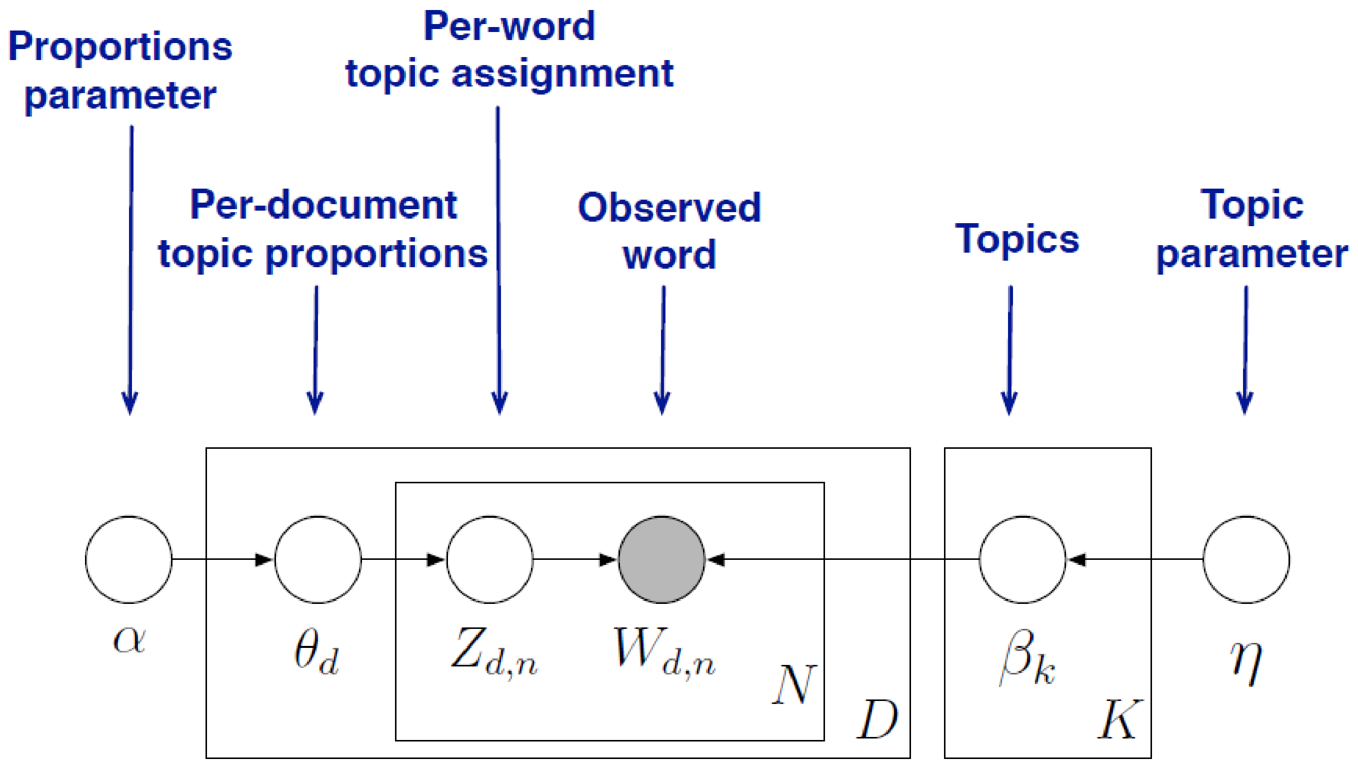
- Encodes our assumptions about the data
- Connects to algorithms for computing with data
- See Pattern Recognition and Machine Learning (Bishop, 2006).

# Latent Dirichlet allocation (LDA)



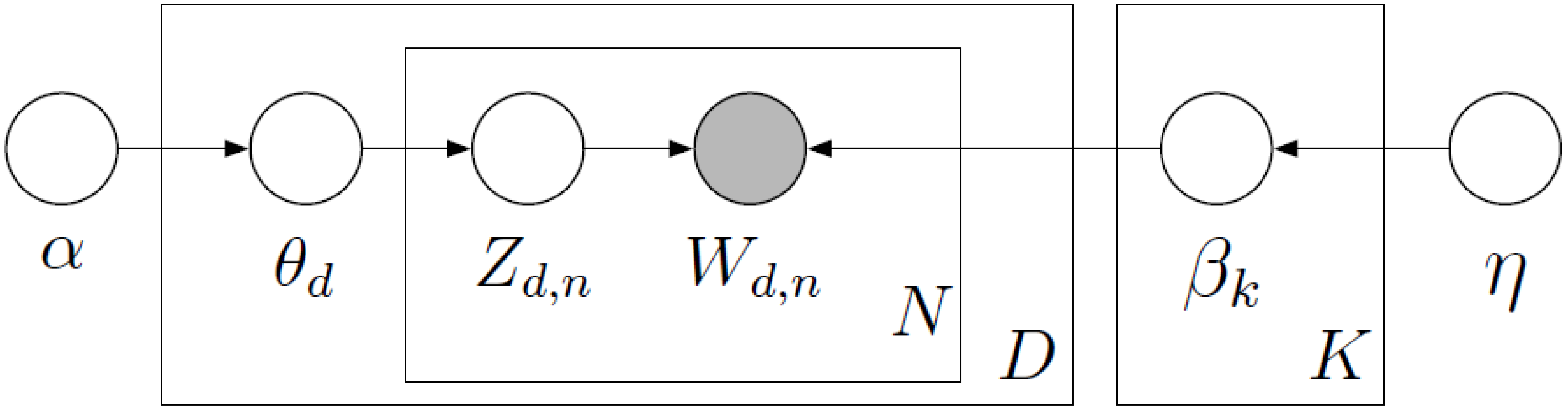
- Nodes are random variables; edges indicate dependence.
- Shaded nodes are observed.
- Plates indicate replicated variables.

# Latent Dirichlet allocation (LDA)



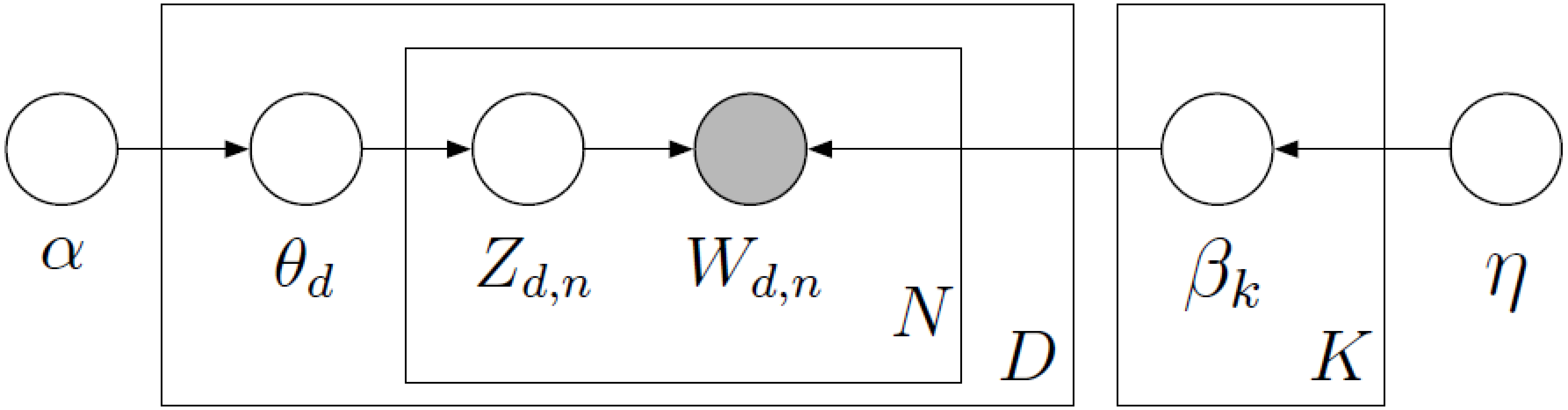
$$\prod_{i=1}^{K} p(\beta_i \mid \eta) \prod_{d=1}^{D} p(\theta_d \mid \alpha) \left( \prod_{n=1}^{N} p(z_{d,n} \mid \theta_d) p(w_{d,n} \mid \beta_{1:K}, z_{d,n}) \right)$$

# Latent Dirichlet allocation (LDA)



- This joint defines a posterior.
- From a collection of documents, infer
  - Per-word topic assignment $z_{d,n}$
  - Per-document topic proportions $\theta_d$
  - Per-corpus topic distributions $\beta_k$
- Then use posterior expectations to perform the task at hand, e.g., information retrieval, document similarity, exploration, ...

# Latent Dirichlet allocation (LDA)



- Approximate posterior inference algorithms
- Mean field variational methods (Blei et al., 2001, 2003)
- Expectation propagation (Minka and Lafferty, 2002)
- Collapsed Gibbs sampling (Griffiths and Steyvers, 2002)
- Collapsed variational inference (Teh et al., 2006)
- Online variational inference (Hoffman et al., 2010)
- Also see Mukherjee and Blei (2009) and Asuncion et al. (2009).

## Implementations of LDA

- There are many available implementations of topic modeling—
  - **LDA-C** *  A C implementation of LDA
  - **HDP** *  A C implementation of the HDP ( "infinite LDA" )
  - **Online LDA** *  A python package for LDA on massive data
  - **LDA in R** *  Package in R for many topic models
  - **LingPipe**  Java toolkit for NLP and computational linguistics
  - **Mallet**  Java toolkit for statistical NLP
  - **TMVE** *  A python package to build browsers from topic models

* available at www.cs.princeton.edu/blei/

## 基于深度学习的文档表示

- 以词向量为基础
- 深度置信网络
- 卷积神经网络
- 循环神经网络
- 预训练语言模型

## 主要内容

- 词的表示
- 文档表示
- 文本相似度计算

## 距离度量（Distance Measures）

- Two major classes of distance measure:

1. *Euclidean*
2. *Non-Euclidean*

## Euclidean Vs. Non-Euclidean

- A *Euclidean space* has some number of real-valued dimensions and points.
  - A *Euclidean distance* is based on the locations of points in such a space.
- A *Non-Euclidean distance* is based on properties of points, but not their "location" in a space.

## Axioms of a Distance Measure

- *d* is a *distance measure* if it is a function from pairs of points to real numbers such that:

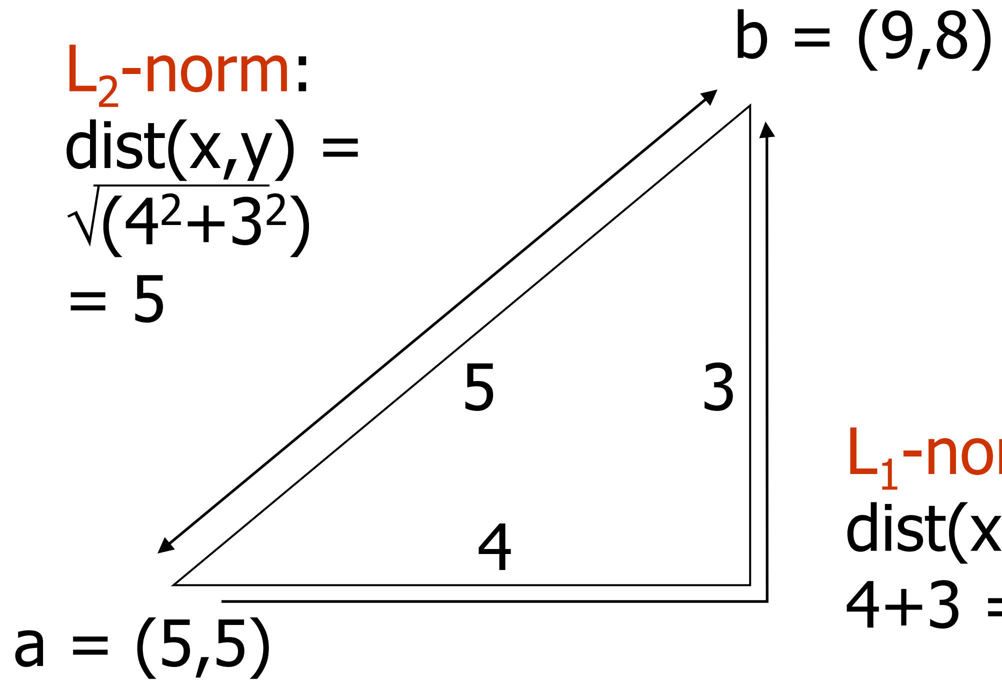
1. $d(x,y) \geq 0$.
2. $d(x,y) = 0$ iff $x = y$.
3. $d(x,y) = d(y,x)$.
4. $d(x,y) \leq d(x,z) + d(z,y)$ (*triangle inequality*).

## Some Euclidean Distances

- *$L_2$ norm* : d(x,y) = square root of the sum of the squares of the differences between *x* and *y* in each dimension.
  - The most common notion of "distance."
- *$L_1$ norm* : sum of the differences in each dimension.
  - *Manhattan distance* = distance if you had to travel along coordinates only.

## Examples of Euclidean Distances

L$_2$-norm:
dist(x,y) =
$\sqrt{(4^2+3^2)}$
= 5



L$_1$-norm:
dist(x,y) =
4+3 = 7

## Another Euclidean Distance

- $L_\infty$ *norm* : d(x,y) = the maximum of the differences between *x* and *y* in any dimension.
- Note: the maximum is the limit as *n* goes to ∞ of the $L_n$ norm: what you get by taking the *n* th power of the differences, summing and taking the *n* th root.

## Non-Euclidean Distances

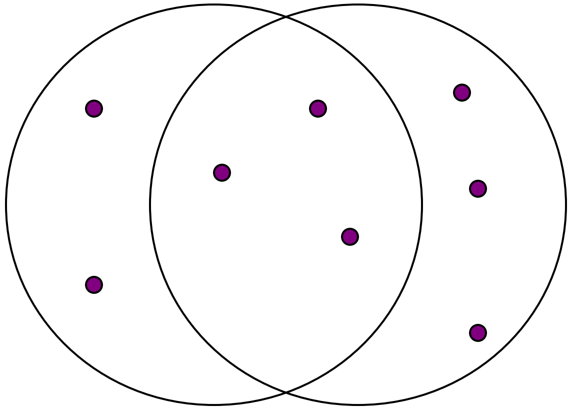
- *Jaccard distance* for sets = 1 minus Jaccard similarity.
- *Cosine distance* = angle between vectors from the origin to the points in question.
- *Edit distance* = number of inserts and deletes to change one string into another.
- *Hamming Distance* = number of positions in which bit vectors differ.

## Jaccard Similarity of Sets

- The *Jaccard similarity* of two sets is the size of their intersection divided by the size of their union.

$$\textit{Sim}\,(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|.$$

## Example: Jaccard Similarity



3 in intersection.
8 in union.
Jaccard similarity
= 3/8

## Cosine Distance

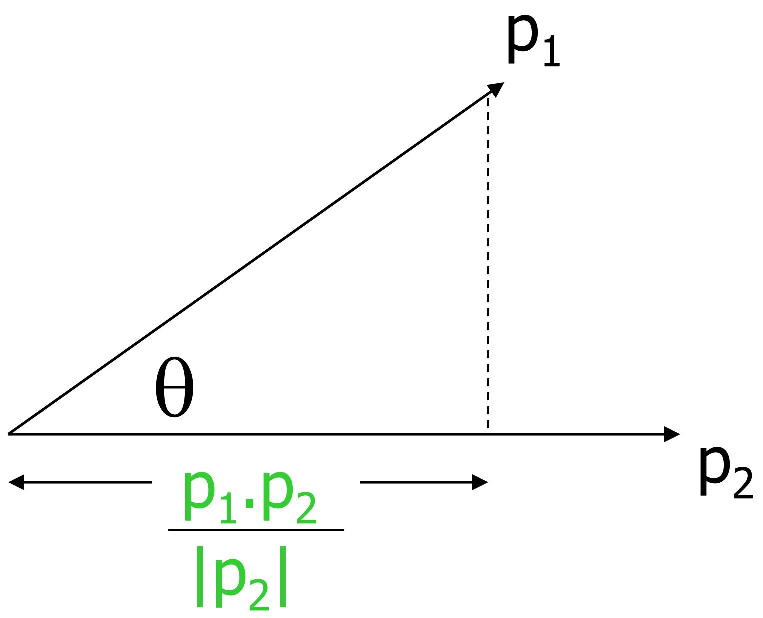
- Think of a point as a vector from the origin (0,0,…,0) to its location.
- Two points' vectors make an angle, whose cosine is the normalized dot-product of the vectors: $p_1.p_2/|p_2||p_1|$.

Example: $p_1$ = 00111; $p_2$ = 10011.

$p_1.p_2 = 2$; $|p_1| = |p_2| = \sqrt{3}$.

$\cos(\theta) = 2/3$; $\theta$ is about 48 degrees.

## Cosine-Measure Diagram



$$d\,(p_1, p_2) = \theta = \arccos(p_1 . p_2 / |p_2||p_1|)$$

## Why C.D. Is a Distance Measure

- d(x,x) = 0 because arccos(1) = 0.
- d(x,y) = d(y,x) by symmetry.
- d(x,y) $\geq$ 0 because angles are chosen to be in the range 0 to 180 degrees.
- Triangle inequality: physical reasoning. If I rotate an angle from *x* to *z* and then from *z* to *y*, I can't rotate less than from *x* to *y*.

## Edit Distance

- The *edit distance* of two strings is the number of inserts and deletes of characters needed to turn one into the other. Equivalently:
- $d(x,y) = |x| + |y| - 2|LCS(x,y)|$.
  - LCS = *longest common subsequence* = any longest string obtained both by deleting from *x* and deleting from *y*.

## Example: LCS

- $x$ = *abcde* ; $y$ = *bcduve*.
- Turn $x$ into $y$ by deleting *a*, then inserting *u* and *v* after *d*.
  - Edit distance = 3.
- Or, LCS(x,y) = *bcde*.
- Note: |x| + |y| - 2|LCS(x,y)| = 5 + 6 –2*4 = 3 = edit distance.

## Why Edit Distance Is a Distance Measure

- d(x,x) = 0 because 0 edits suffice.
- d(x,y) = d(y,x) because insert/delete are inverses of each other.
- d(x,y) $\geq$ 0: no notion of negative edits.
- Triangle inequality: changing *x* to *z* and then to *y* is one way to change *x* to *y*.

## Variant Edit Distances

- Allow insert, delete, and *mutate*.
  - Change one character into another.
- Minimum number of inserts, deletes, and mutates also forms a distance measure.
- Ditto for any set of operations on strings.
  - Example: substring reversal OK for DNA sequences

## Hamming Distance

- *Hamming distance* is the number of positions in which bit-vectors differ.
- Example: $p_1$ = 10101; $p_2$ = 10011.
- $d(p_1, p_2) = 2$ because the bit-vectors differ in the 3rd and 4th positions.

## Why Hamming Distance Is a Distance Measure

- d(x,x) = 0 since no positions differ.
- d(x,y) = d(y,x) by symmetry of "different from."
- d(x,y) $\geq$ 0 since strings cannot differ in a negative number of positions.
- Triangle inequality: changing *x* to *z* and then to *y* is one way to change *x* to *y*.

## Acknowledgements

- 部分slides来自
  - Anand Rajaraman and Jeff Ullman
  - Richard Socher
  - Tomas Mikolov
  - Dan Jurafsky and James Martin