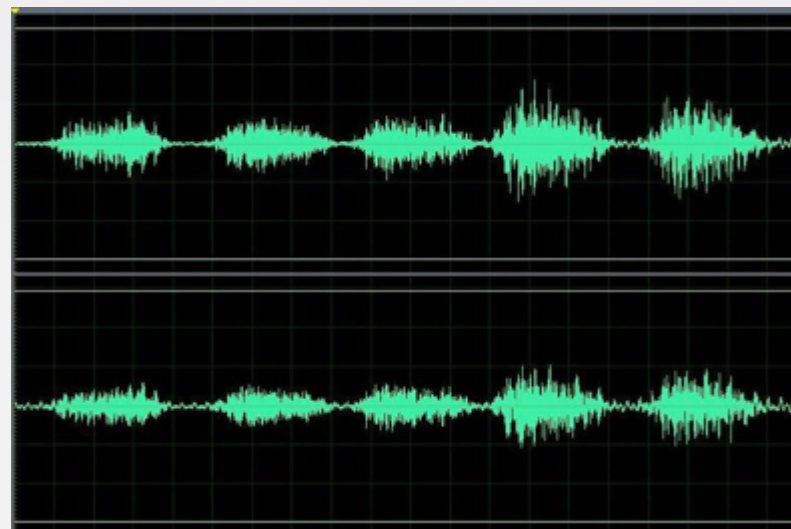


# 词汇 的 星空

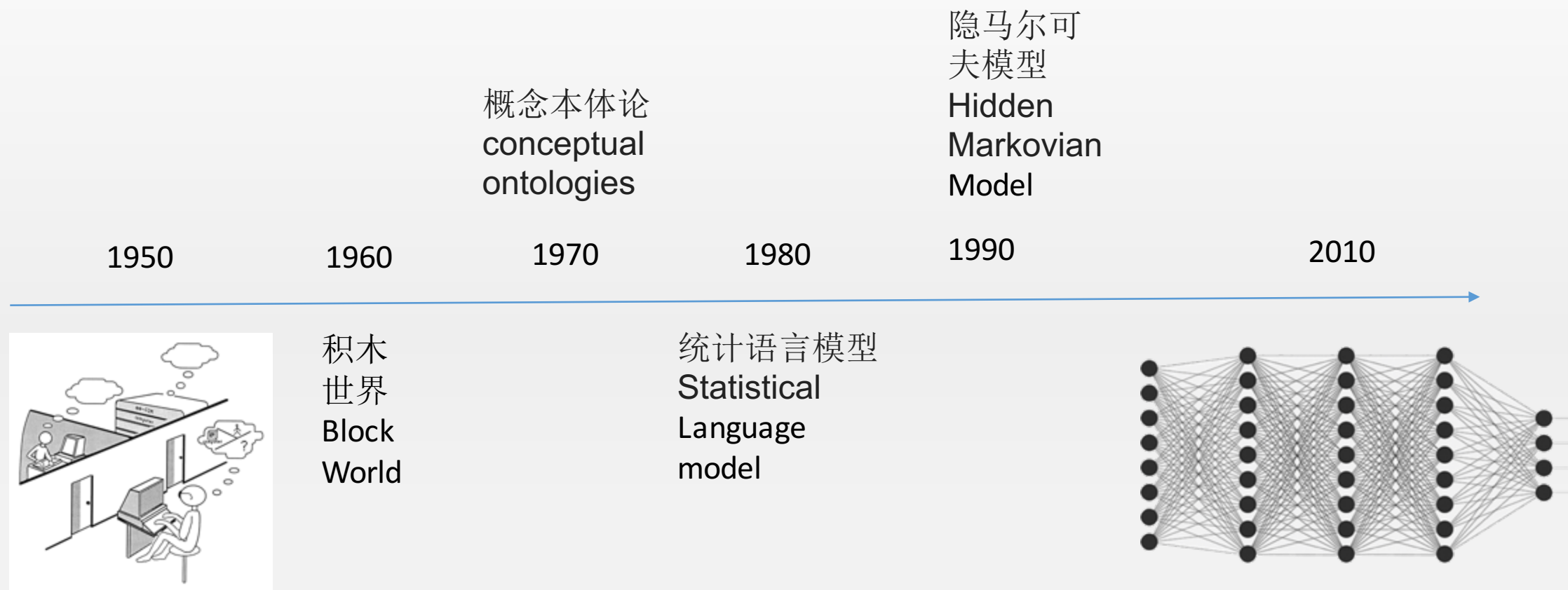
词向量技术简介

张江

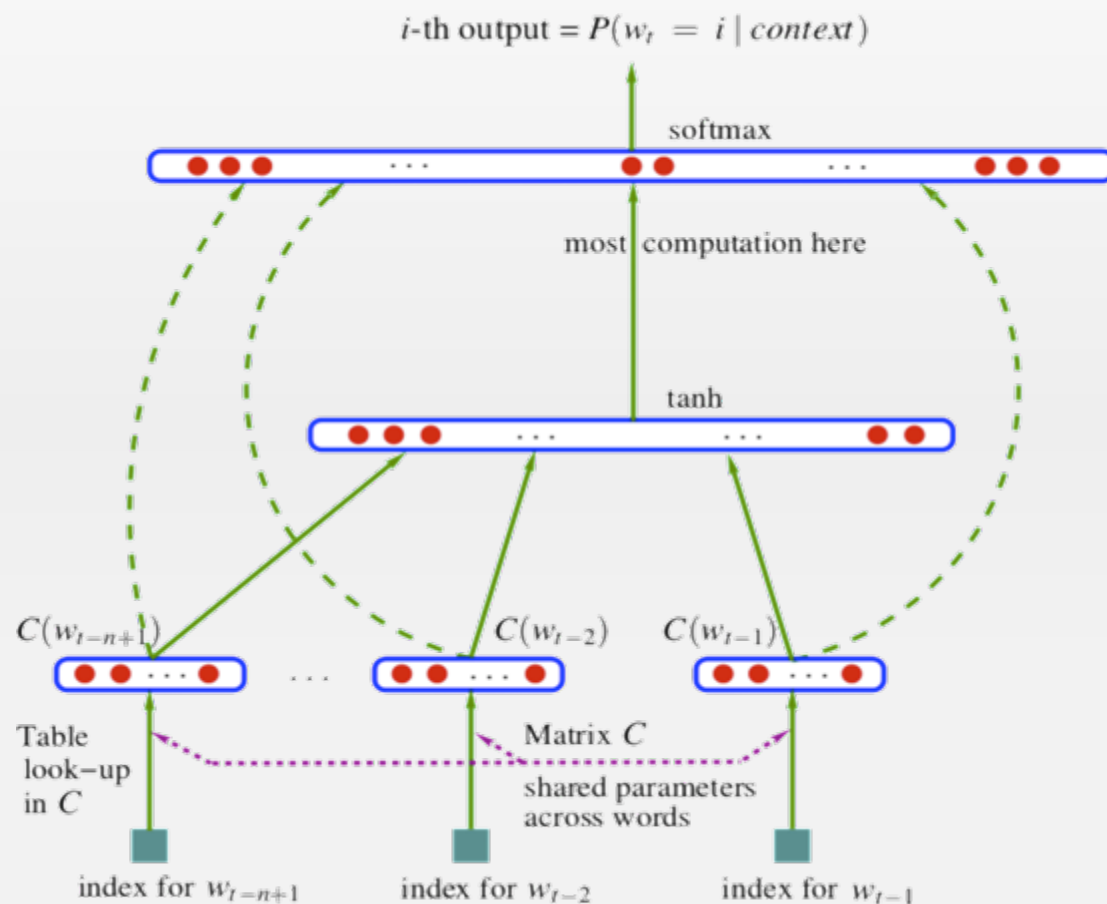
# 我们生活在一个序列的世界



# 自然语言处理技术的演进

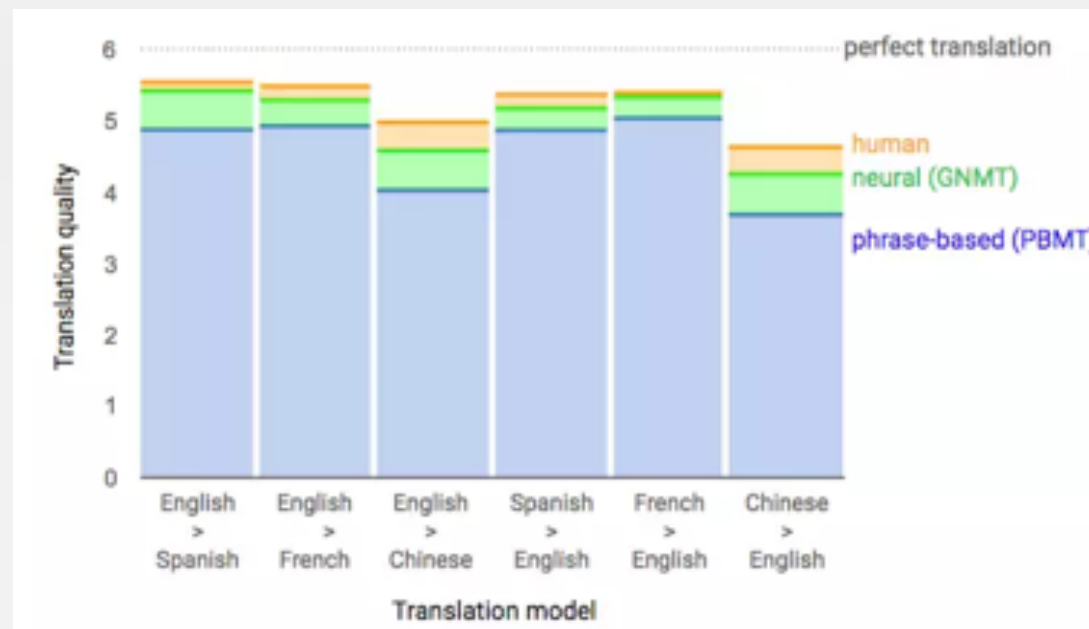
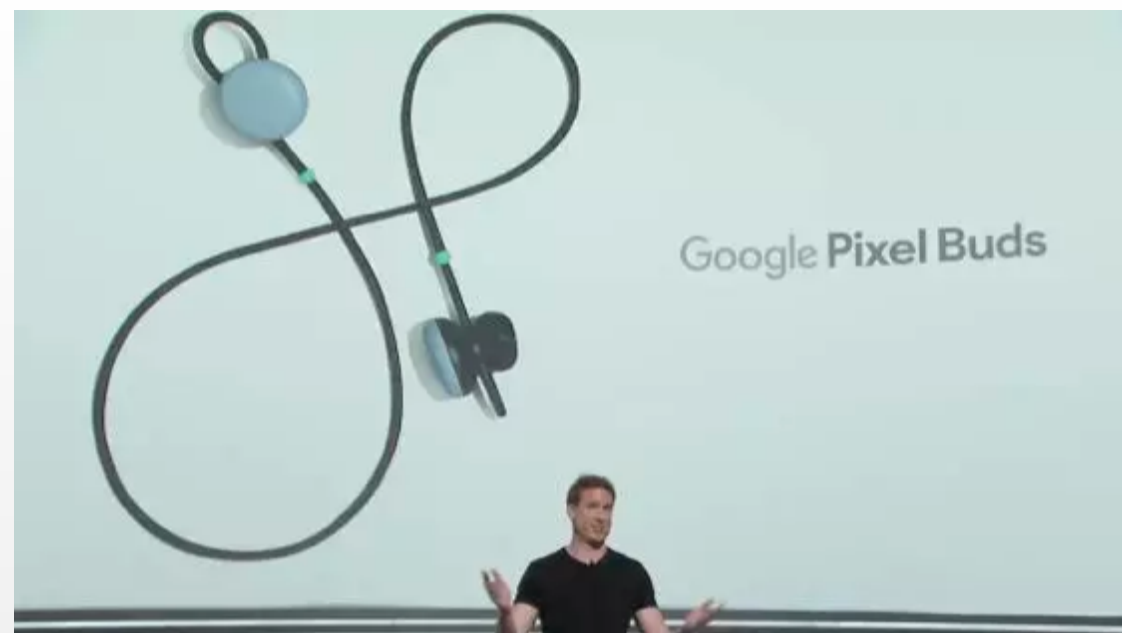
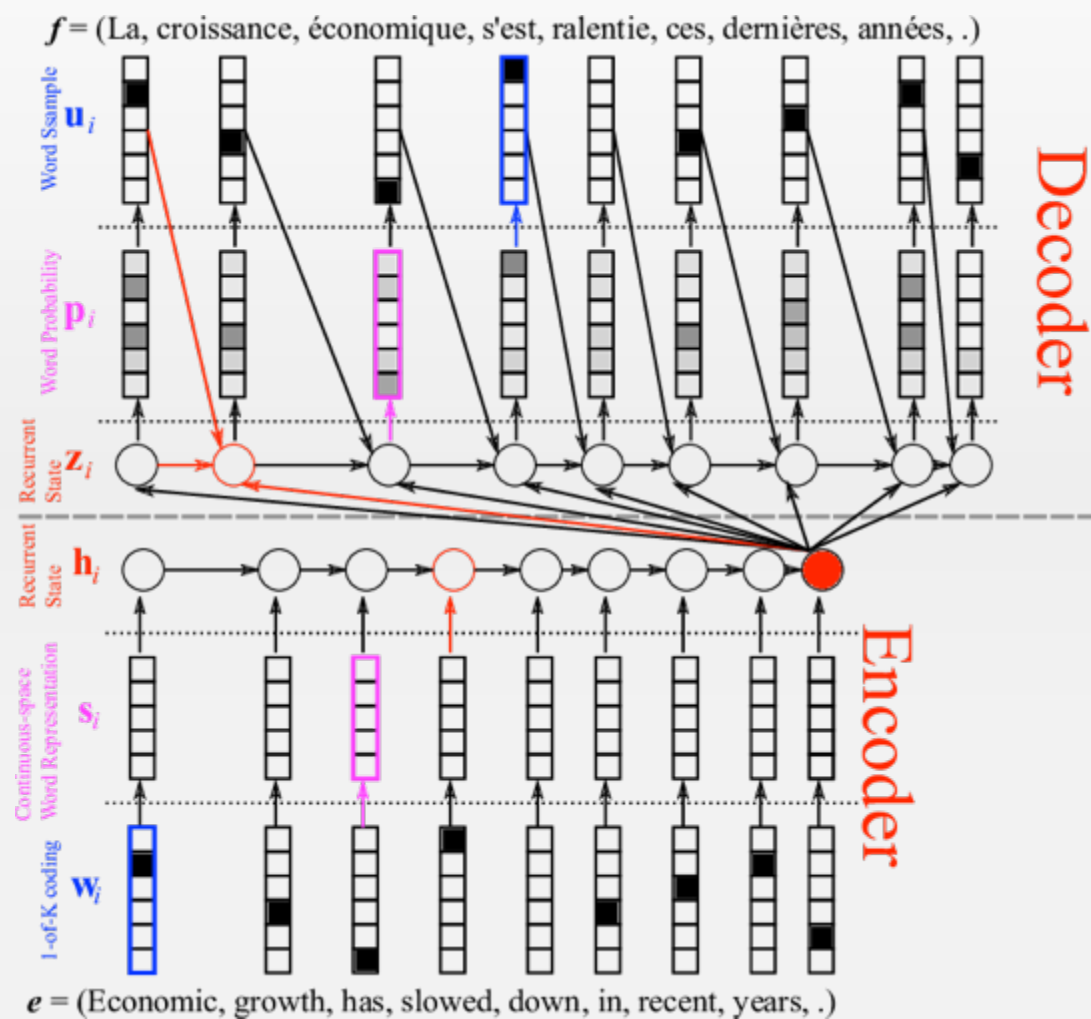


# 词向量革命



What is king + man - woman?

# 机器翻译



## 阅读理解



by ent423 ,ent261 correspondent updated 9:49 pm et ,thu  
march 19 ,2015 ( ent261 ) a ent114 was killed in a parachute  
accident in ent45 ,ent85 ,near ent312 ,a ent119 official told  
ent261 on wednesday .he was identified thursday as  
special warfare operator 3rd class ent23 ,29 ,of ent187 ,  
ent265 .` ent23 distinguished himself consistently  
throughout his career .he was the epitome of the quiet  
professional in all facets of his life ,and he leaves an  
inspiring legacy of natural tenacity and focused

...

by ent270 ,ent223 updated 9:35 am et ,mon march 2 ,2015  
( ent223 ) ent63 went familial for fall at its fashion show in  
ent231 on sunday ,dedicating its collection to `` mamma "  
with nary a pair of `` mom jeans " in sight .ent164 and ent21 ,  
who are behind the ent196 brand ,sent models down the  
runway in decidedly feminine dresses and skirts adorned  
with roses ,lace and even embroidered doodles by the  
designers ' own nieces and nephews .many of the looks  
featured saccharine needlework phrases like `` i love you ,

...

ent119 identifies deceased sailor as X ,who leaves behind  
a wife

X dedicated their fall fashion show to moms

- MaluubA是一个能够阅读几百个童话故事算法。训练结束后，该算法可以正确地回答算法并不熟悉文本的多选题，准确率超过 70% 。
- 研究人员还在《哈利波特和魔法石》上进行测试，该算法能够以近似的准确率回答相关文本问题。这一成绩，比使用深度学习方法前提高 15% 。也比最好的人工编码解决方案好 2% 。
- Yoshua Bengio 说，「从数字上看，这是一次大的飞跃。」



## 语言生成



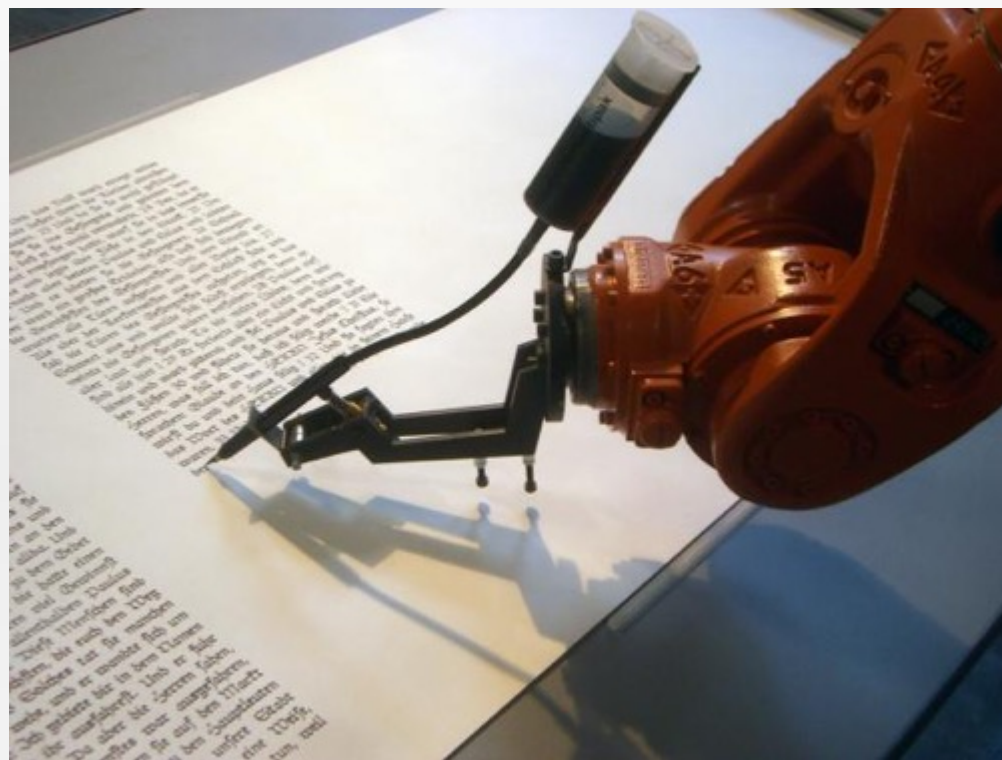
雨过海风一阵阵  
撒向天空的小鸟  
光明冷静的夜  
太阳光明  
现在的天空中去  
冷静的心头  
野蛮的北风起  
当我发现一个新的世界

## 语言生成

日本科研人员开发的人工智能会写小说，他们还把这些小说拿去参加创作比赛。3月21日，日本“人工智能（AI）小说创作”的研究人员在东京举行报告会，对外介绍他们研发的人工智能系统所创作的四篇小说，就作品内容和文章生成系统等进行解说。

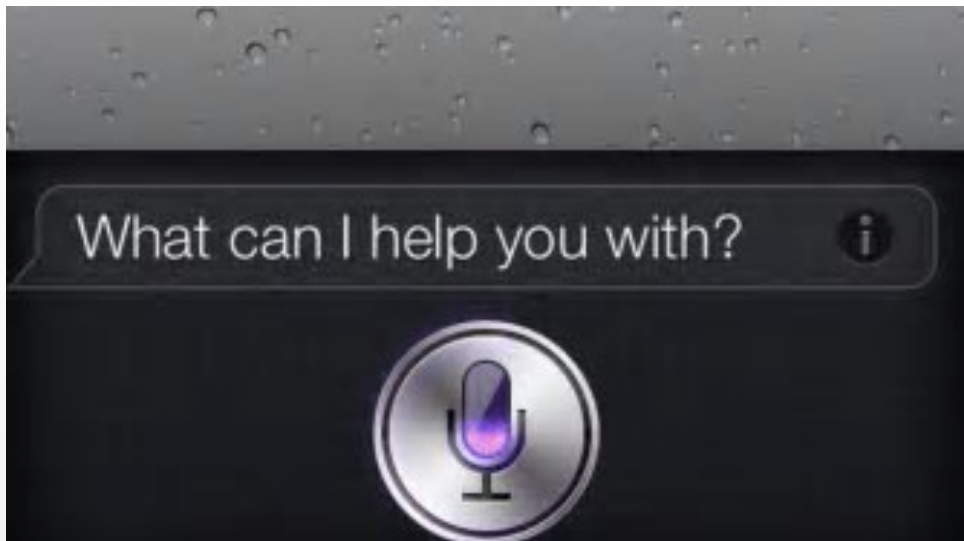
报道称，“我是作家”团队的两部作品由人类事先设定好登场人物、内容大纲等相当于文章“零部件”的内容，人工智能再根据这些内容自动生成小说。“人狼智能”团队的两部作品，则在人工智能之间玩“狼人游戏”（推理游戏的一种，类似“谁是卧底”和“杀人游戏”），然后选出有意思的故事发展，再由人类改编成小说。

《电脑写小说的那一天》



《你是AI TYPE-S》





“你是机器。”  
“我不是，你才是机器。”  
“你糊涂了。我是人类，你是机器。”  
“你是洗衣机。”  
“你是肥皂，所以我是你的主人。”

有时还会阐述阴谋论：

“谁是美国总统？”  
“巴拉克·奥巴马。”  
“奥巴马不可能同时担任两个国家的总统。”

# 词乃语言之本



## 如何在计算机中表达单词呢？

- 直接表示
  - Student
  - ['s','t','u','d','e','n','t']: [18, 19, 20, 3, 4, 13, 19]
- 问题:
  - 不同的单词长度不等
  - 意思相似的单词在表示上并不相似

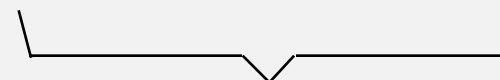
# 如何在计算机中表达单词呢？

- One-hot编码

单词	序号	编码
A	0	[1, 0, 0, 0, ....., 0]
An	1	[0, 1, 0, 0, ....., 0]
About	2	[0, 0, 1, 0, ....., 0]
.....	.....	.....
Zero	14901	[0, 0, 0, 0, ....., 1]

- 缺点:

- 太浪费存储空间
- 语义相似性无法体现, 如何体现?

  
14901

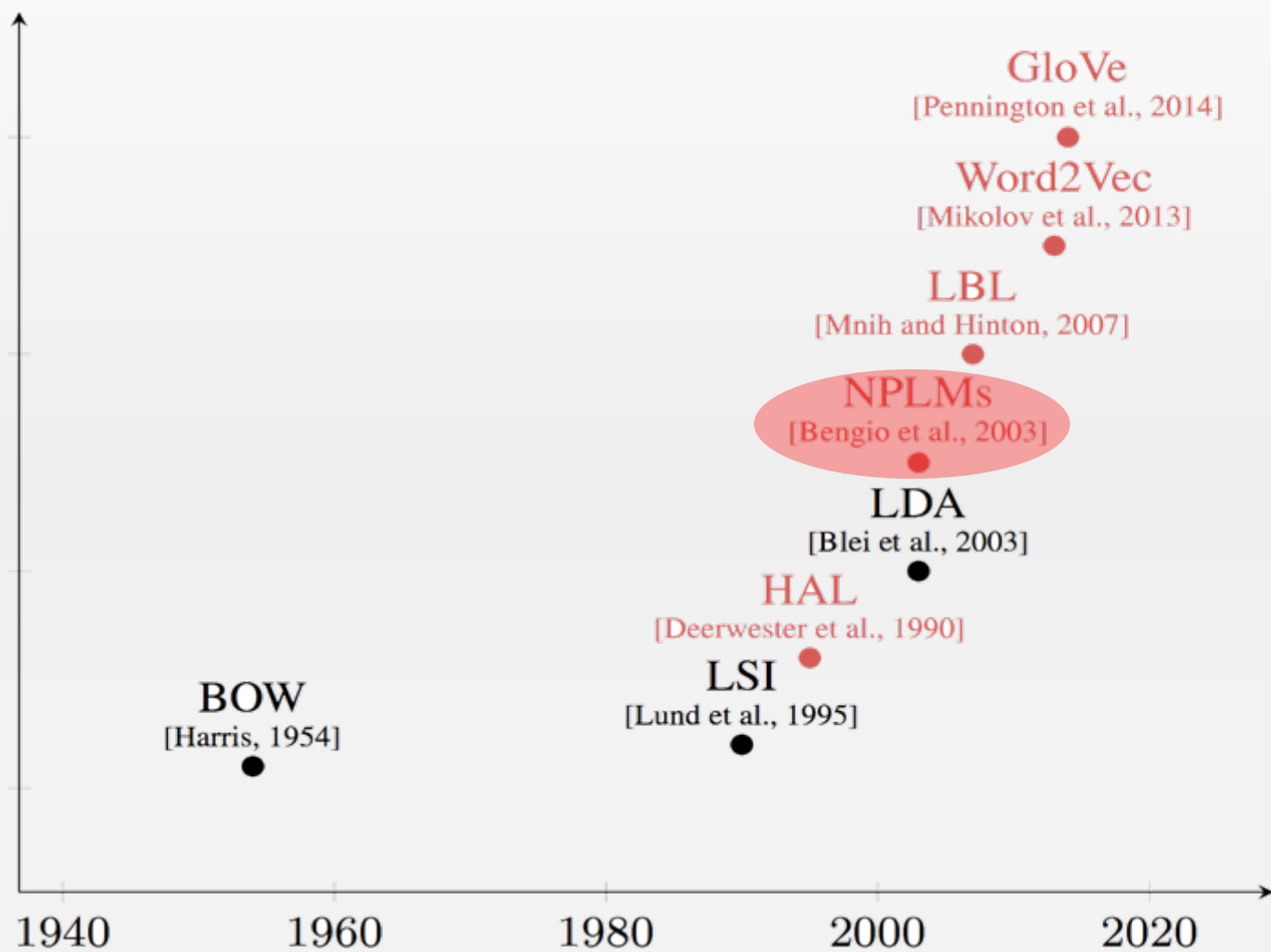
## 我们想要的是这样的词向量

单词	词向量
star	[0.5, 0.7, -0.3, 0.2]
sun	[0.5, -0.8, 0.6, 0.4]
moon	[0.49, 0.6, -0.3, 0.1]

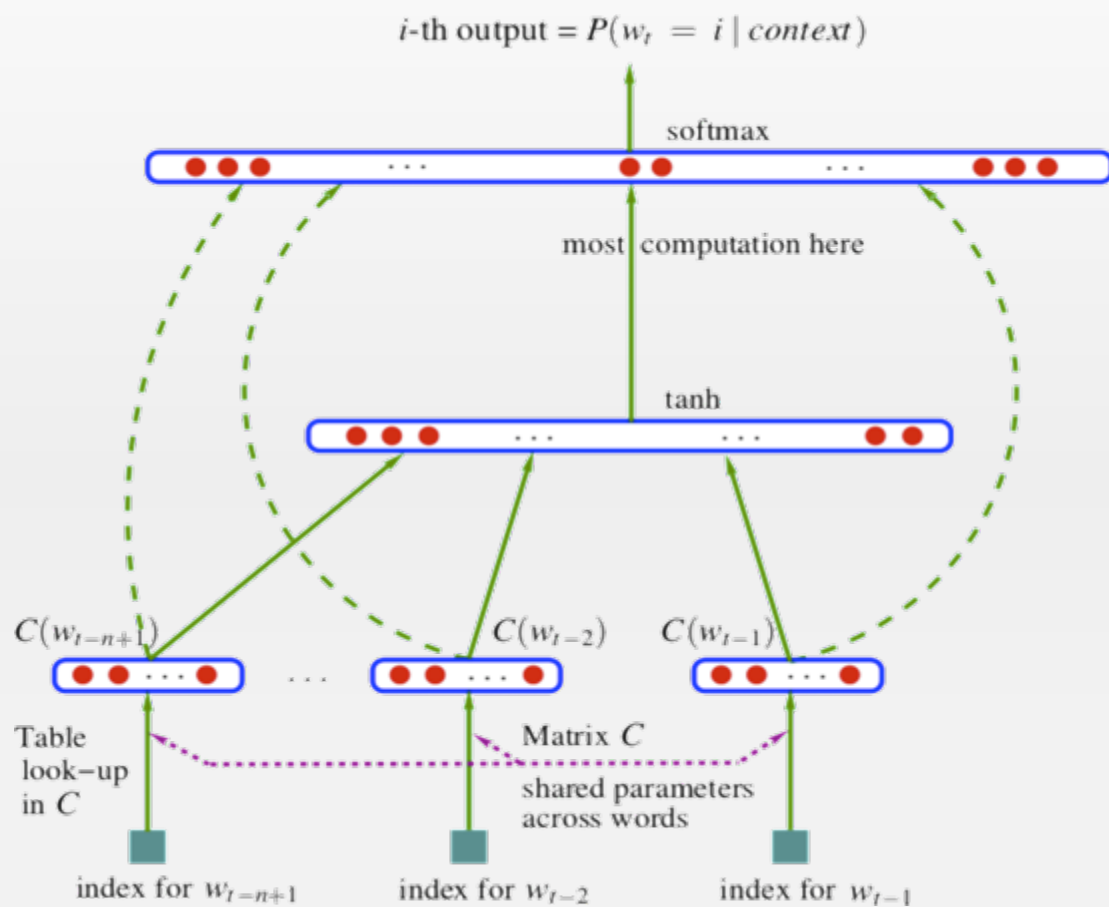
- 几个特点:
  - 密集编码
  - 向量相似性能够反映语义相似性



# 突破性进展



## 突破性进展



Yoshua Bengio

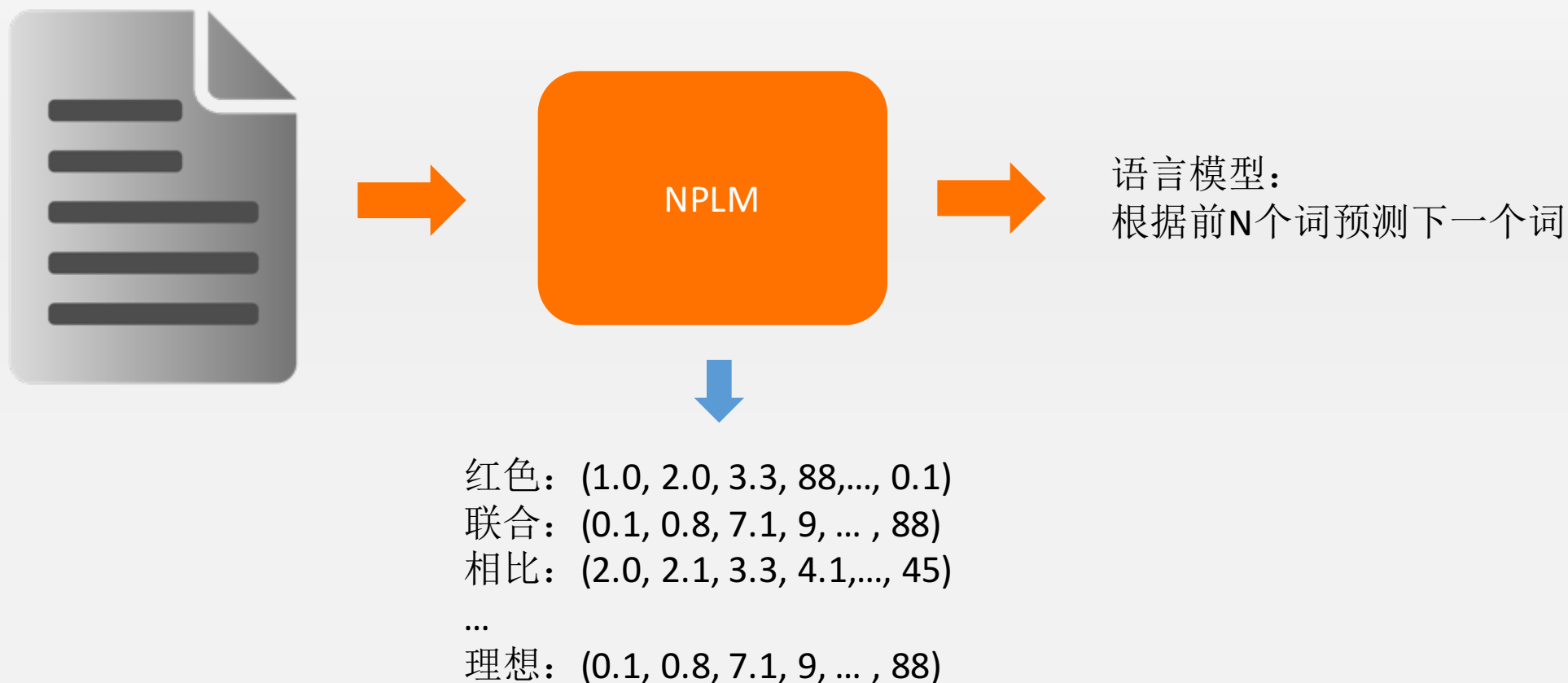
Y.Bengio et al. A Neural Probabilistic Language Model, 2003

# NPLM基本思想

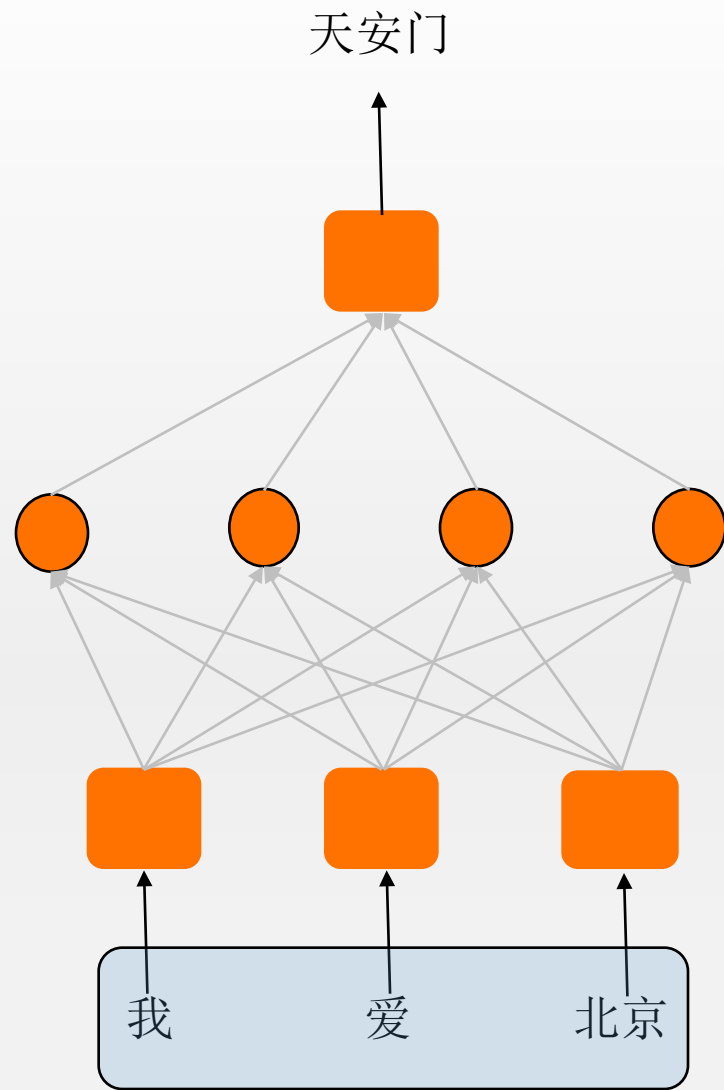


红色: (1.0, 2.0, 3.3, 88,..., 0.1)  
联合: (0.1, 0.8, 7.1, 9, ... , 88)  
相比: (2.0, 2.1, 3.3, 4.1,..., 45)  
...  
理想: (0.1, 0.8, 7.1, 9, ... , 88)

## 词向量作为语言模型的副产物



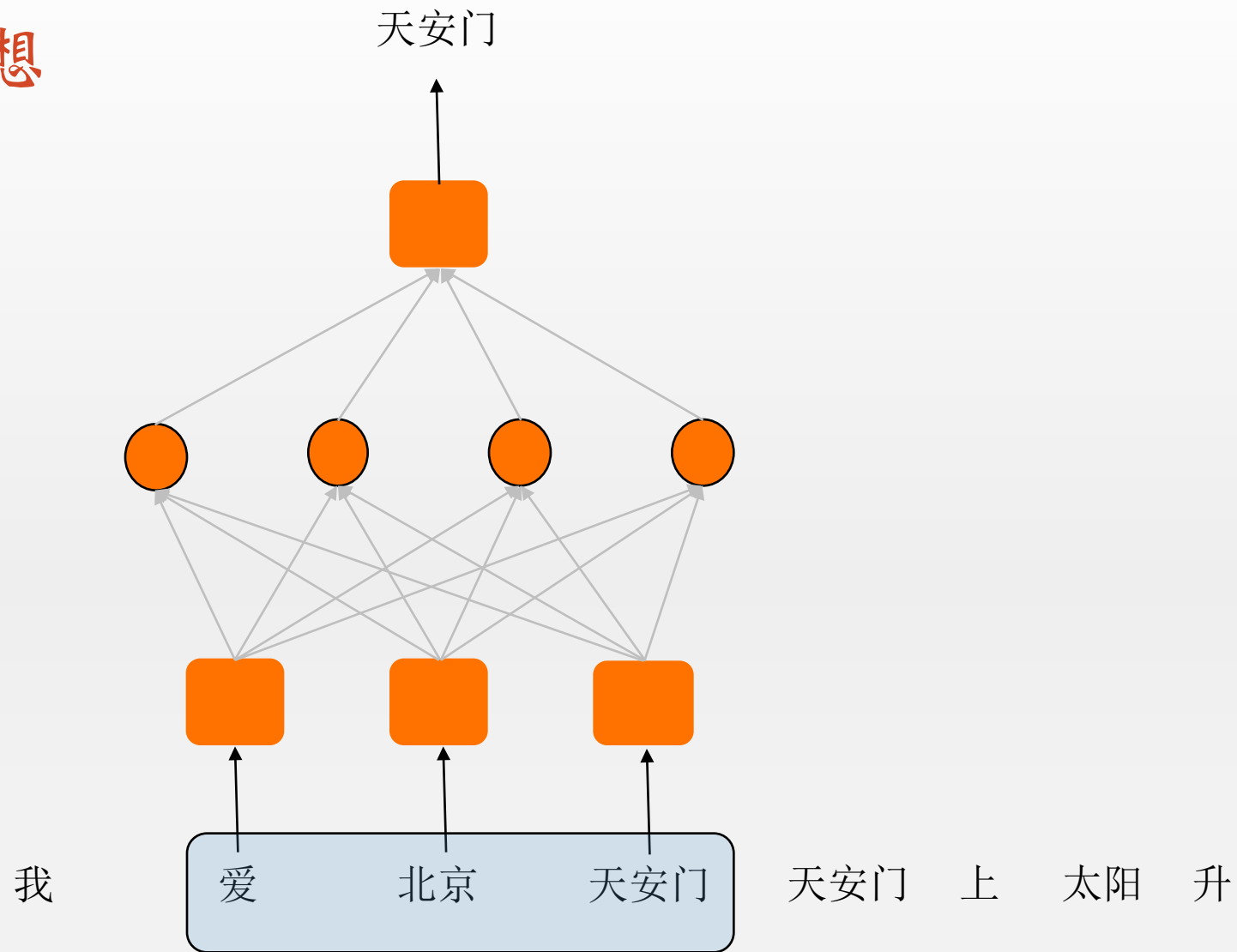
# NPLM基本思想



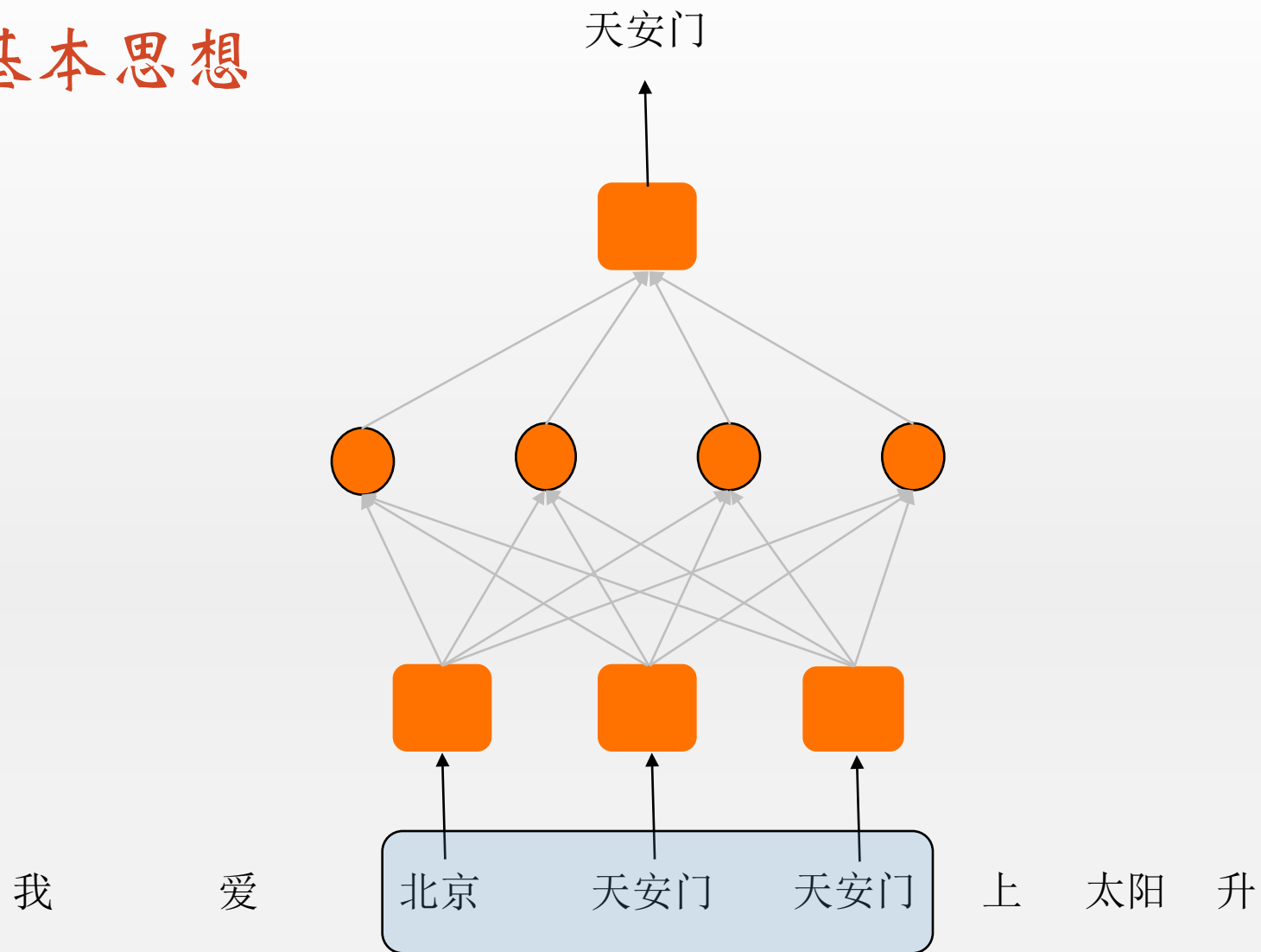
天安门 天安门 上 太阳 升



# NPLM基本思想

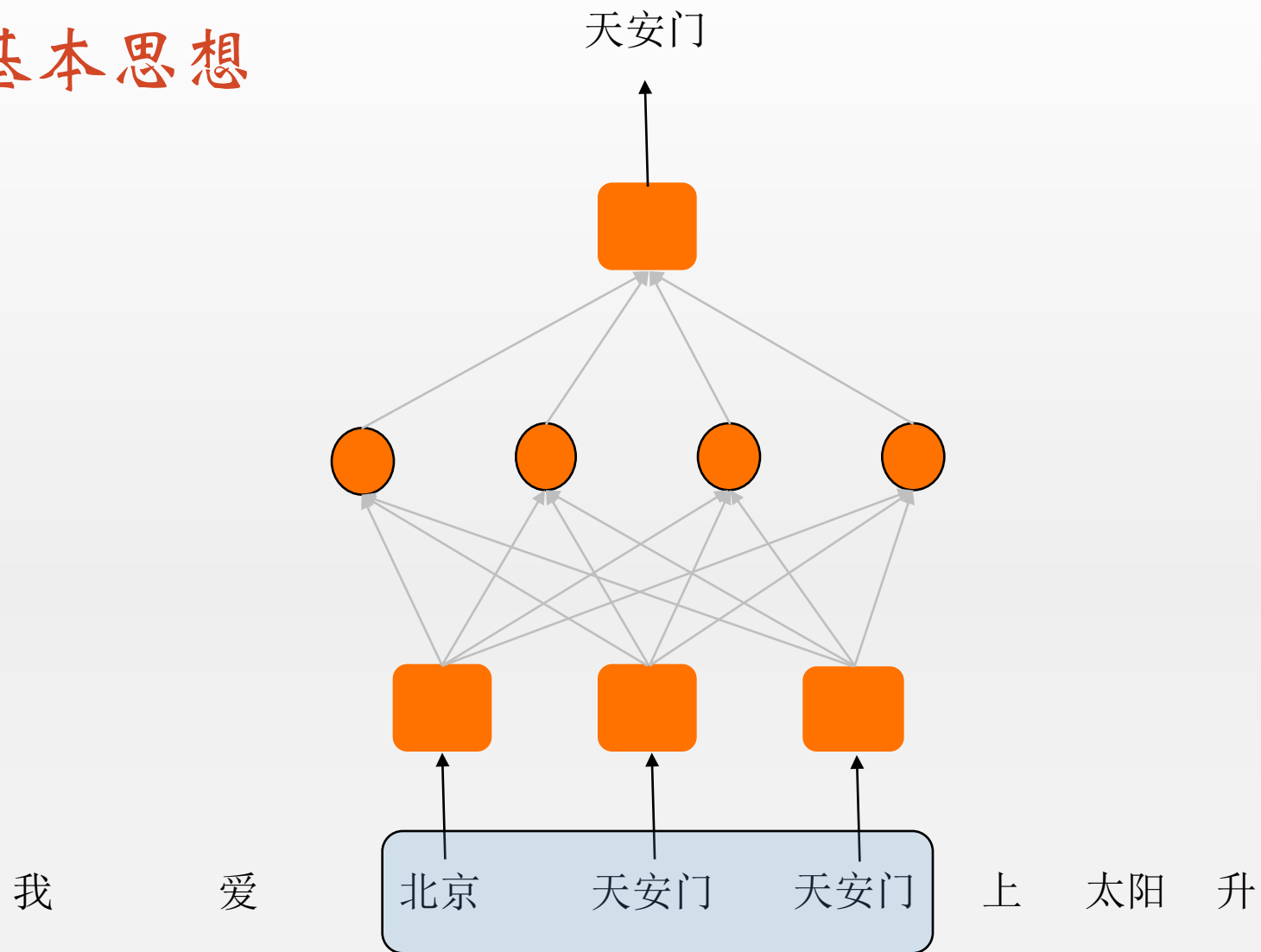


# NPLM基本思想



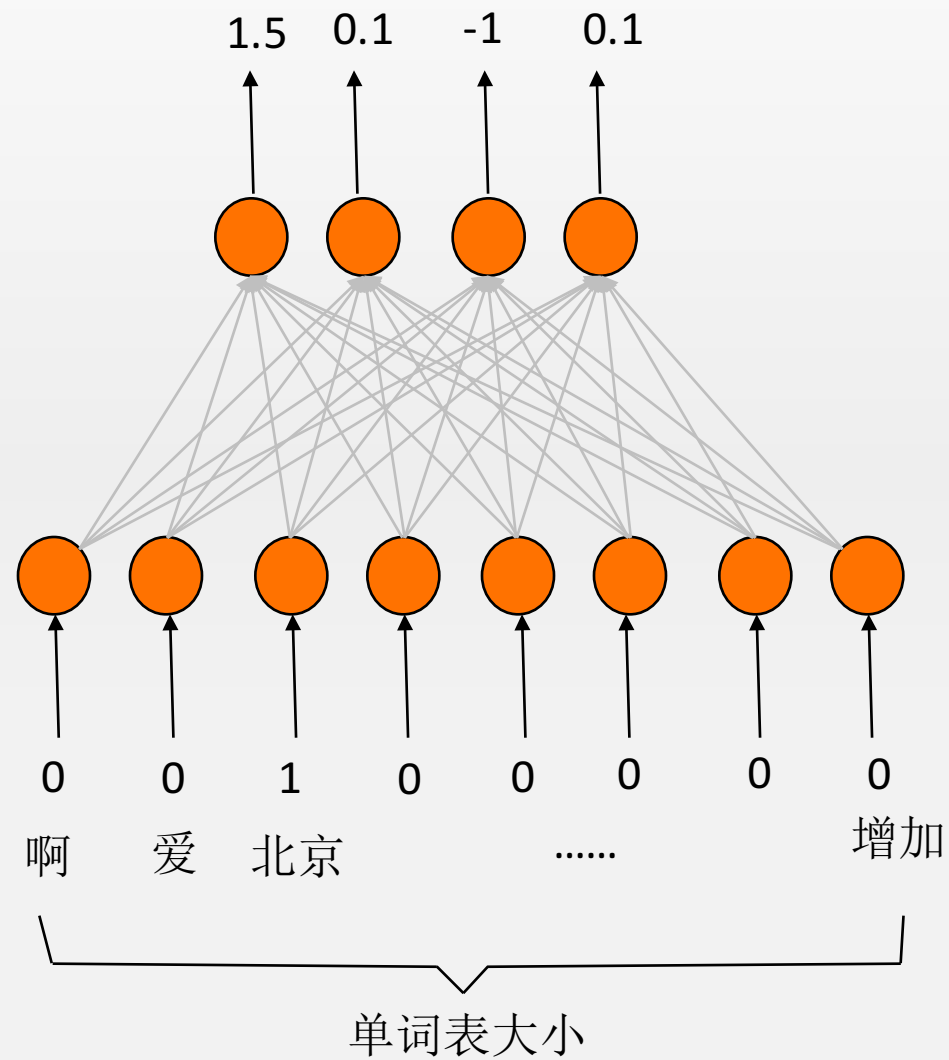
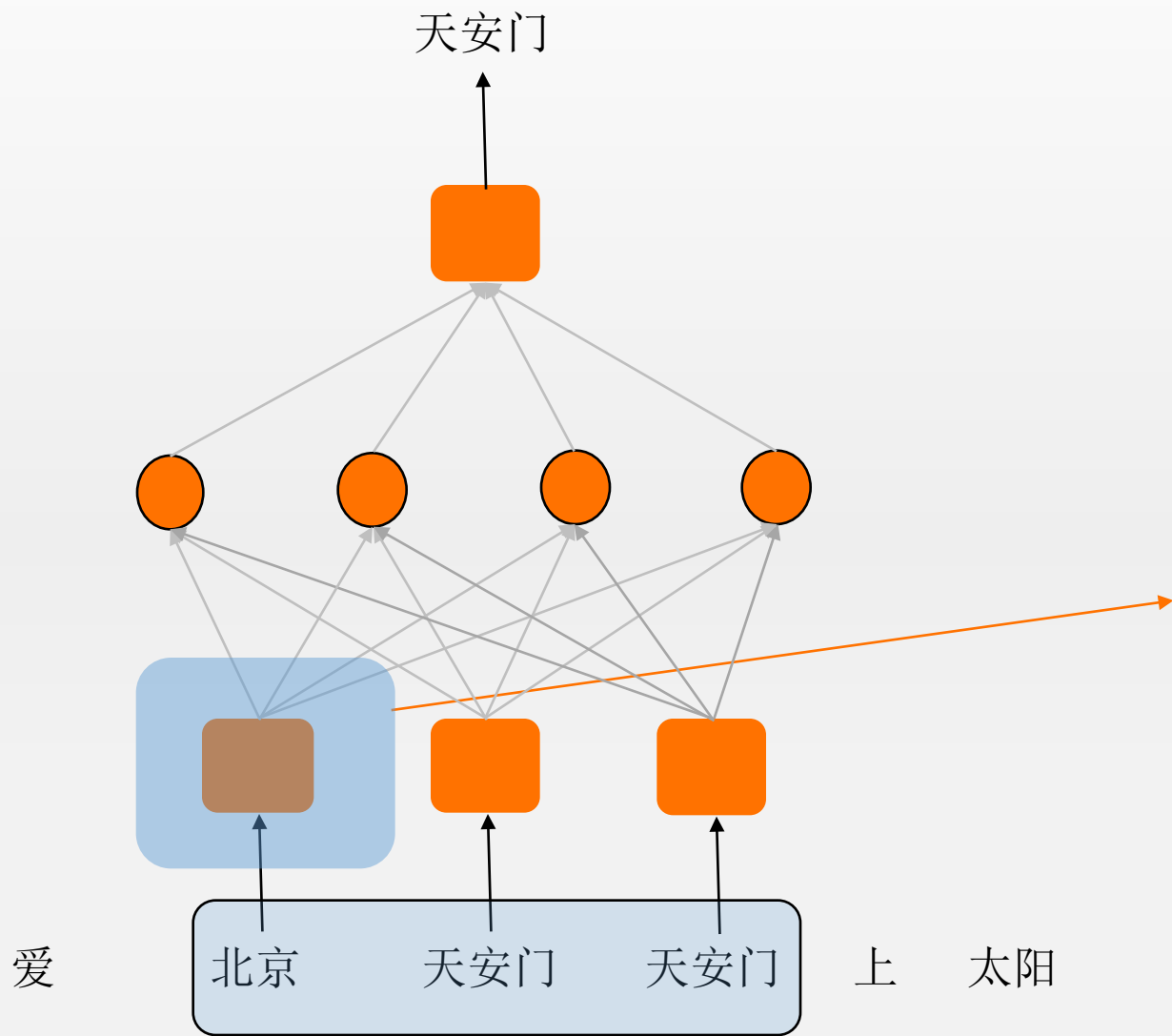
N-gram语言模型  
 $N=3$

# NPLM基本思想

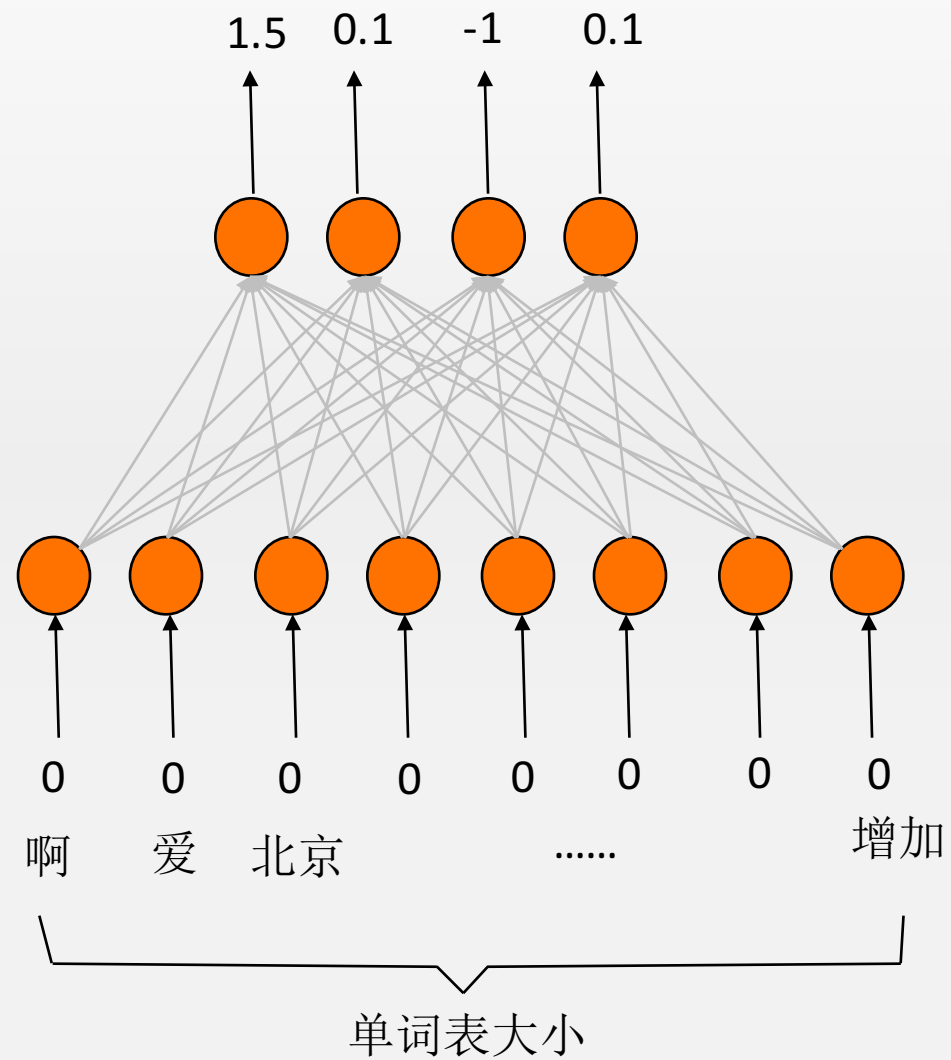
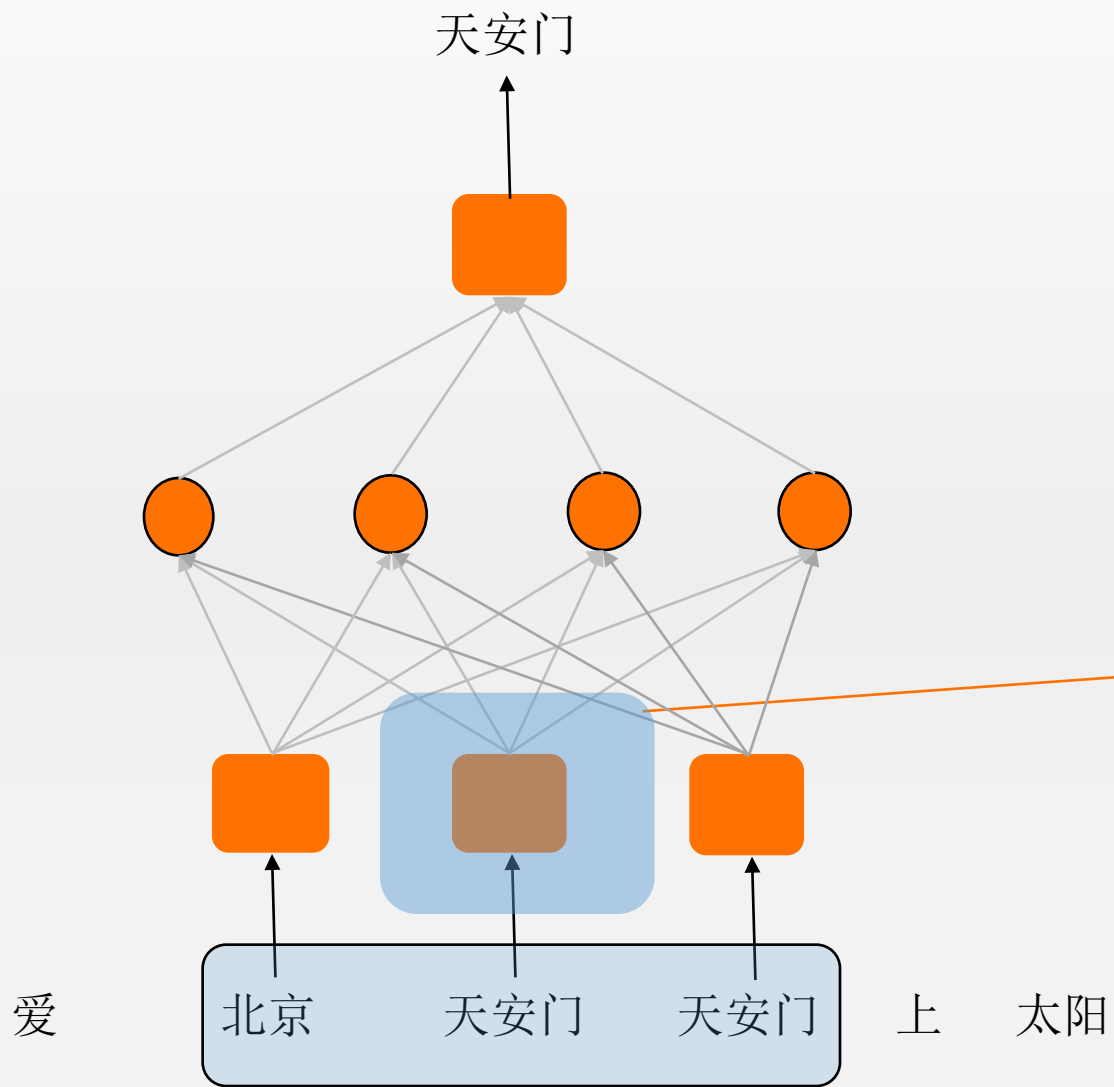


N-gram语言模型  
 $N=3$

# NPLM详解

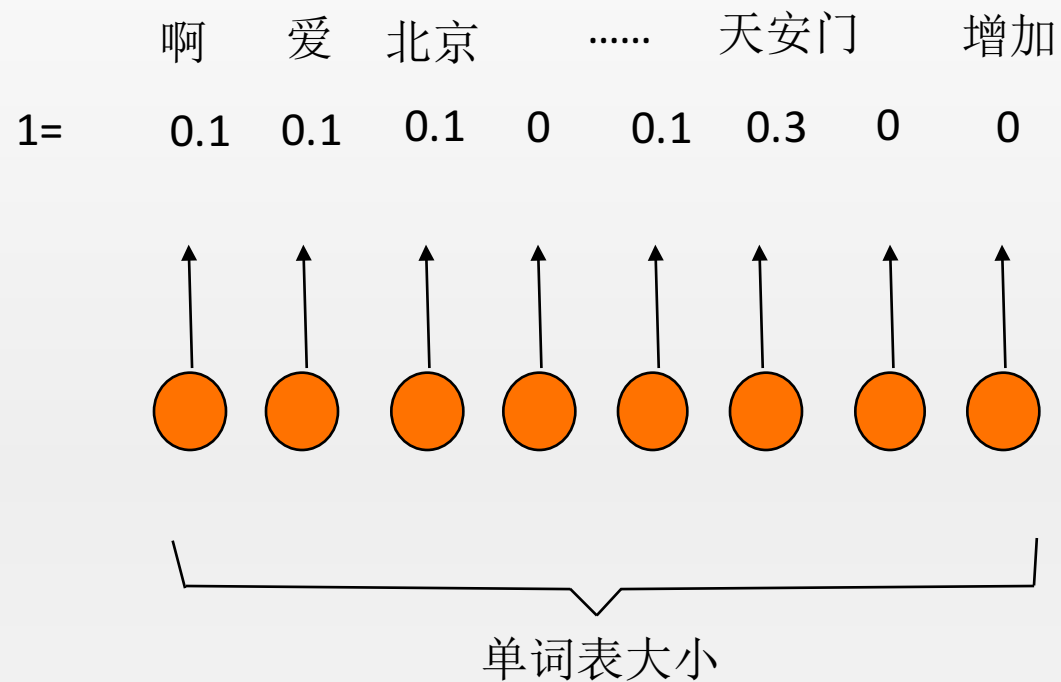
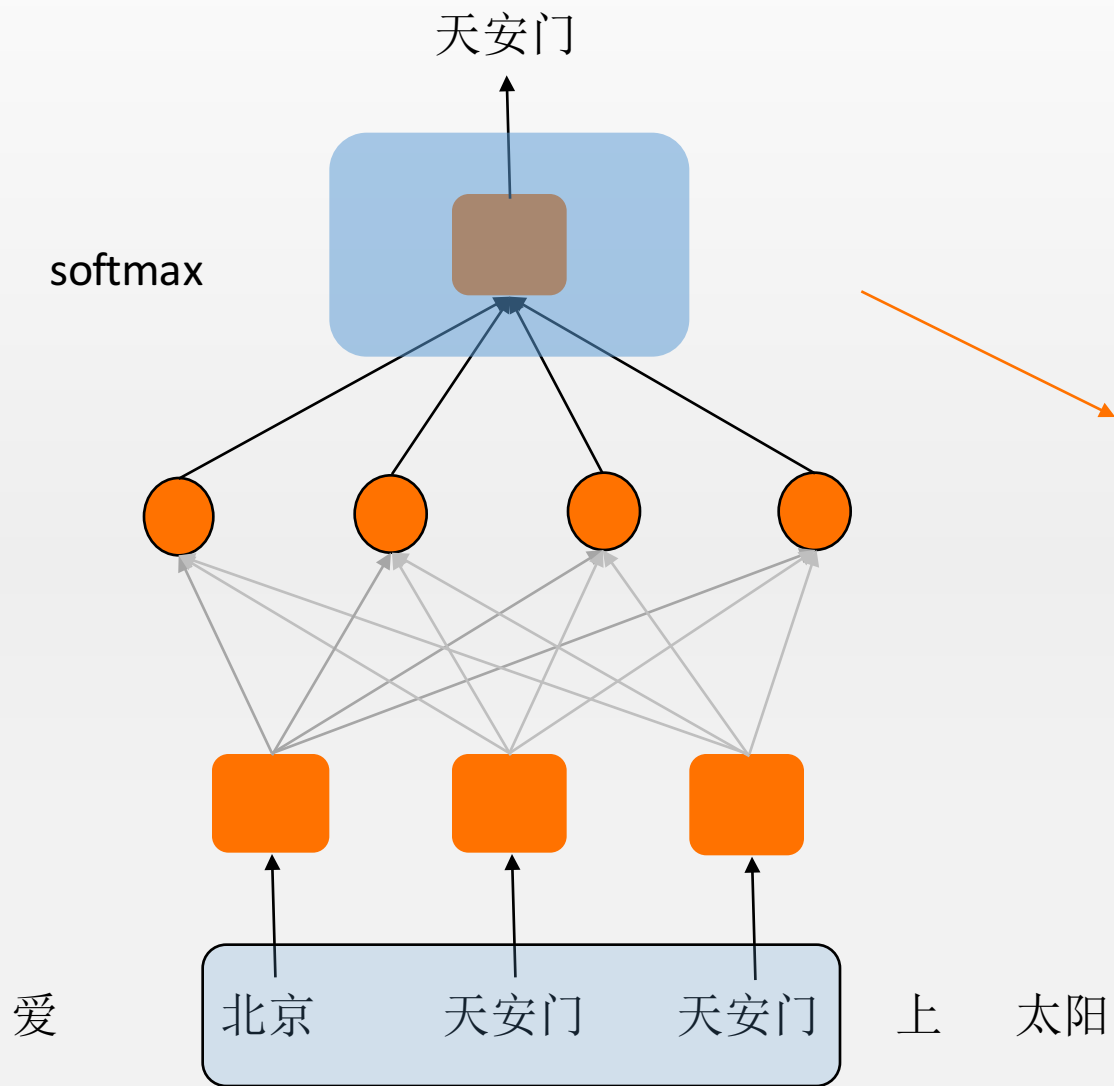


# NPLM详解

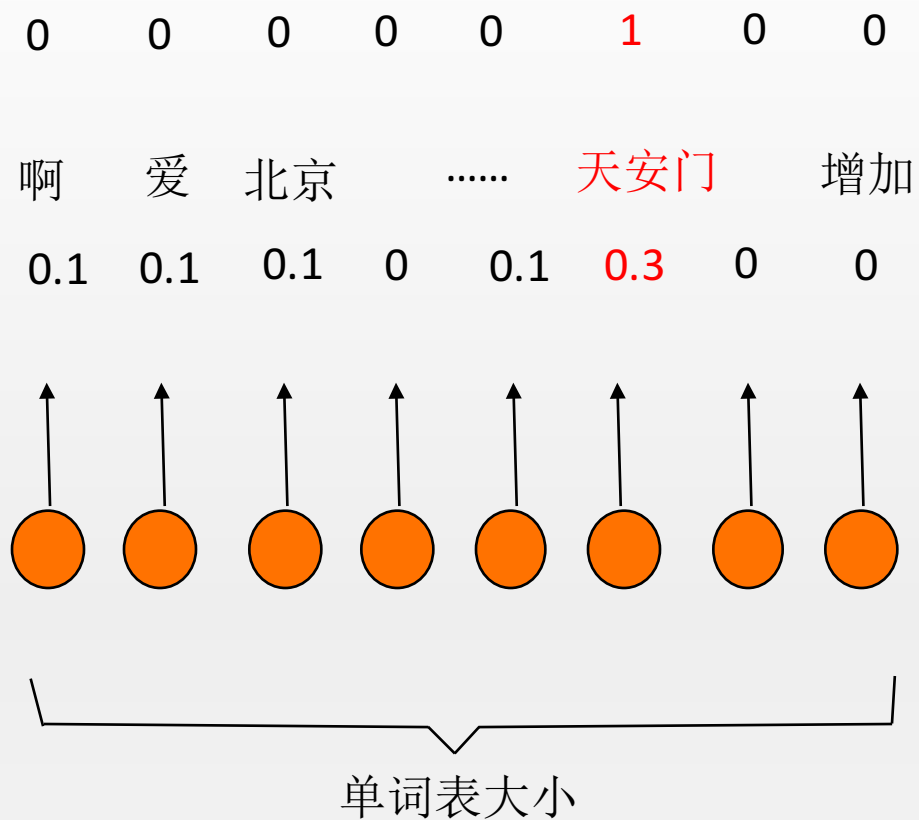
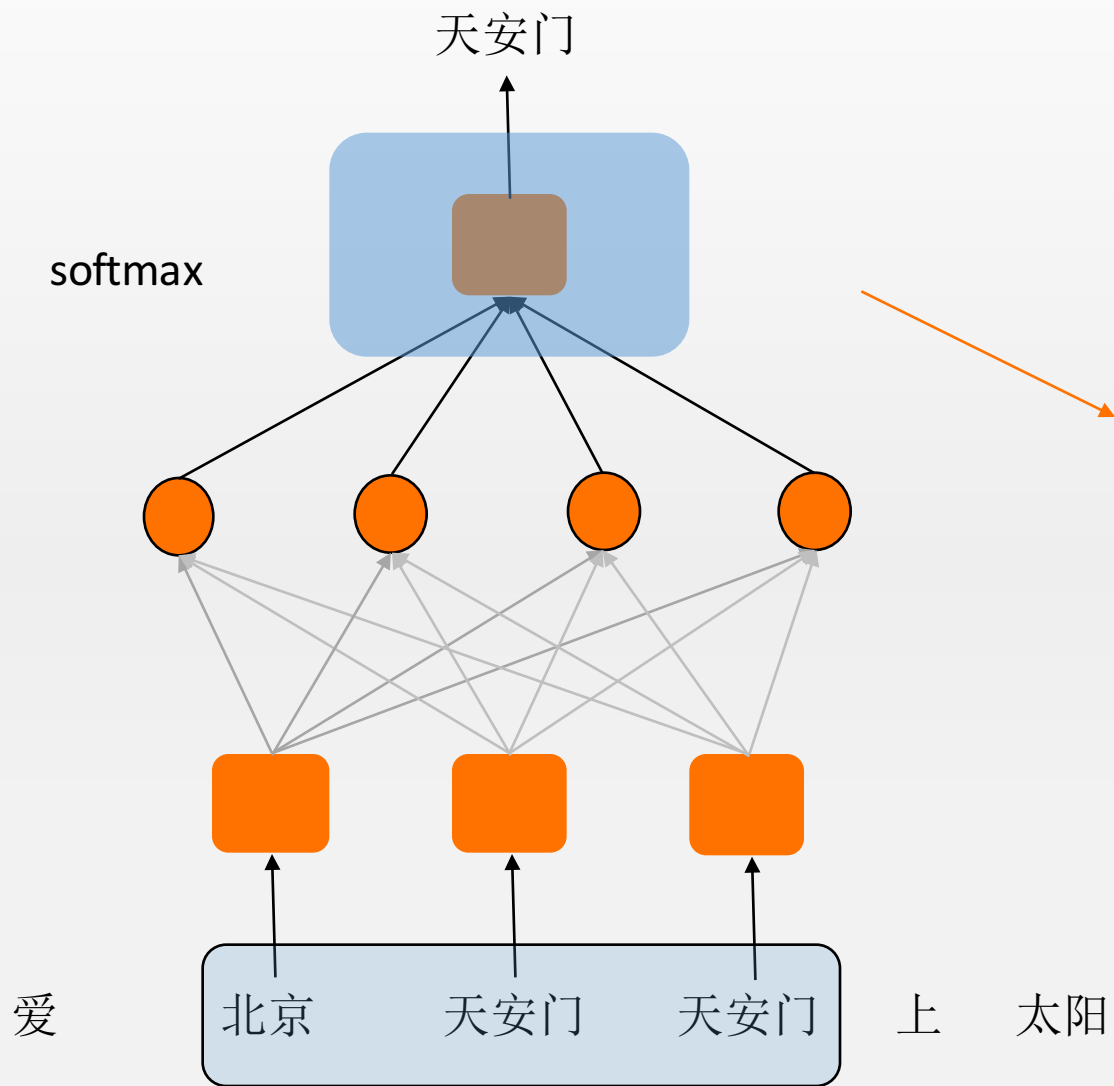




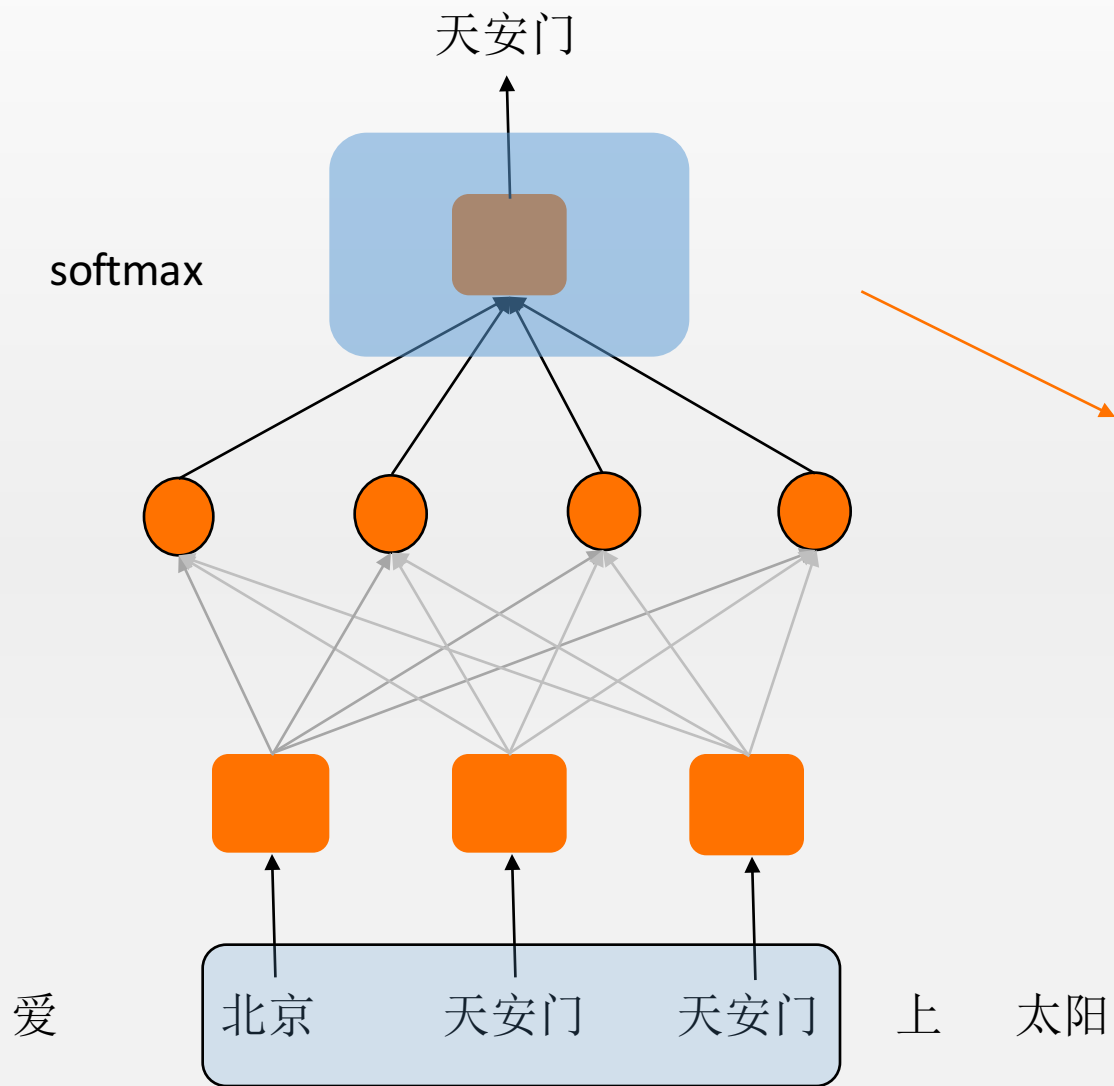
# NPLM详解



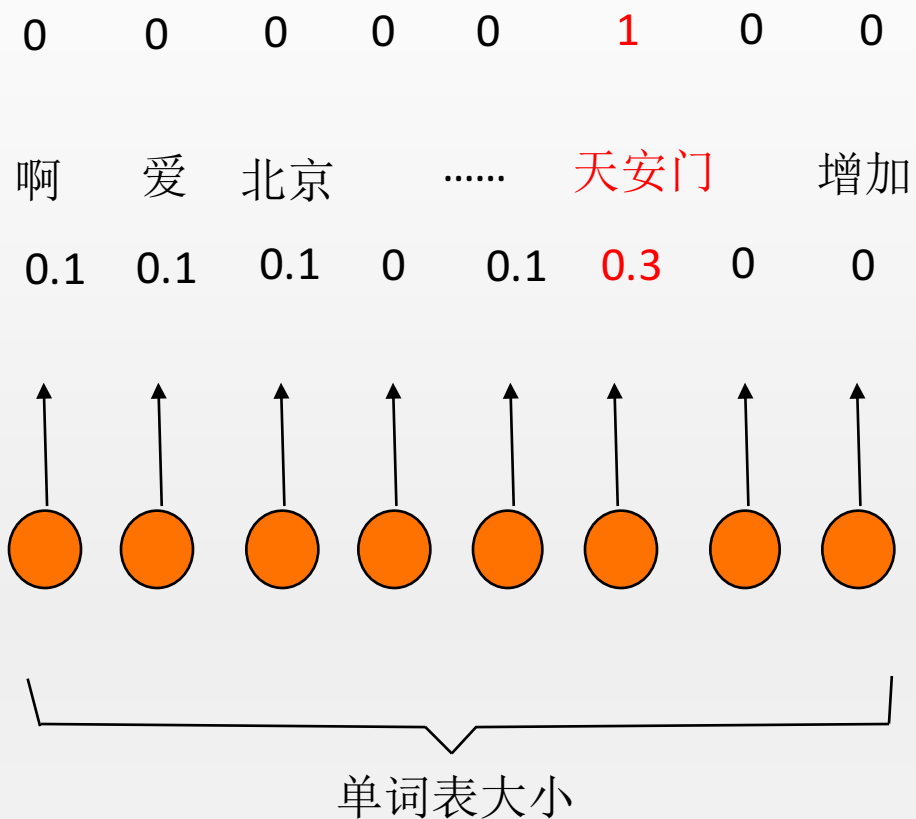
# NPLM详解



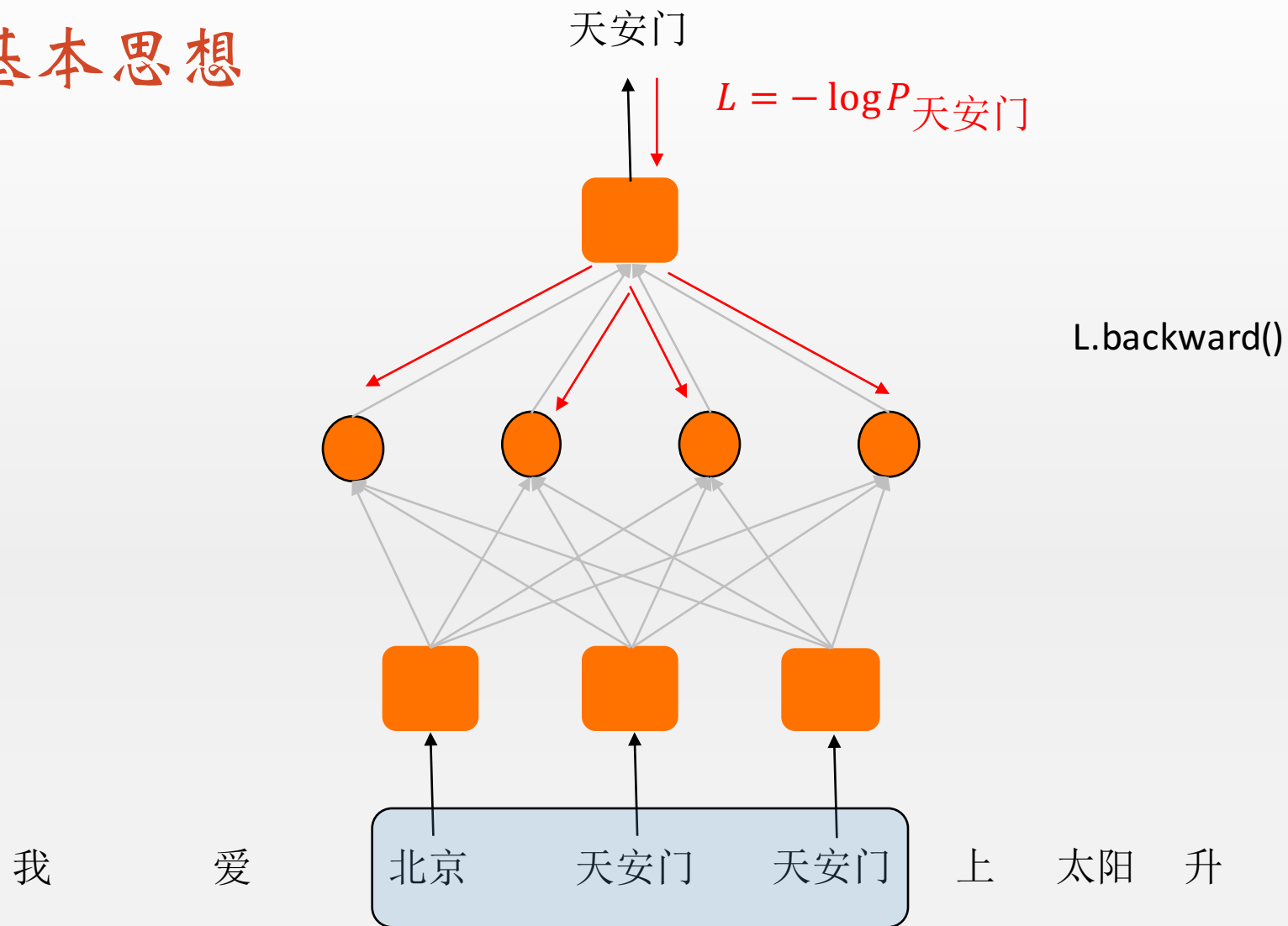
# NPLM详解



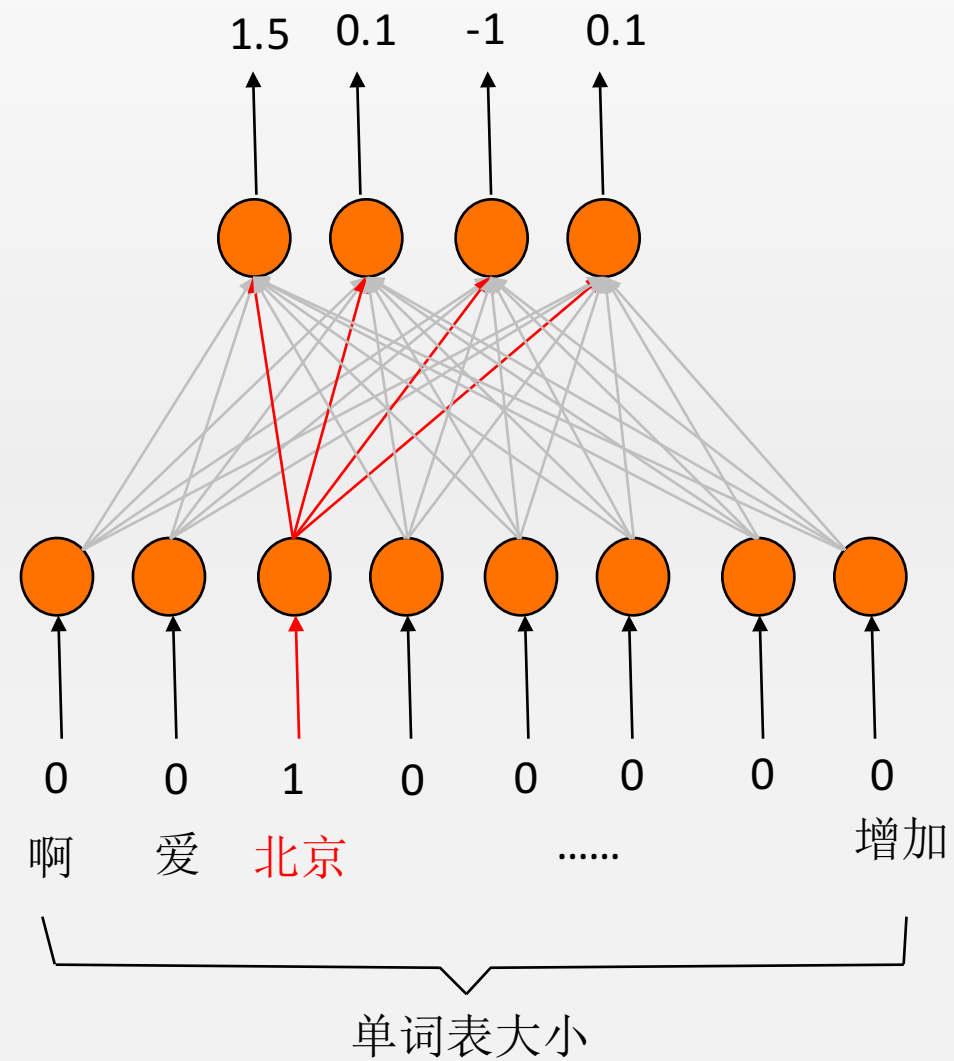
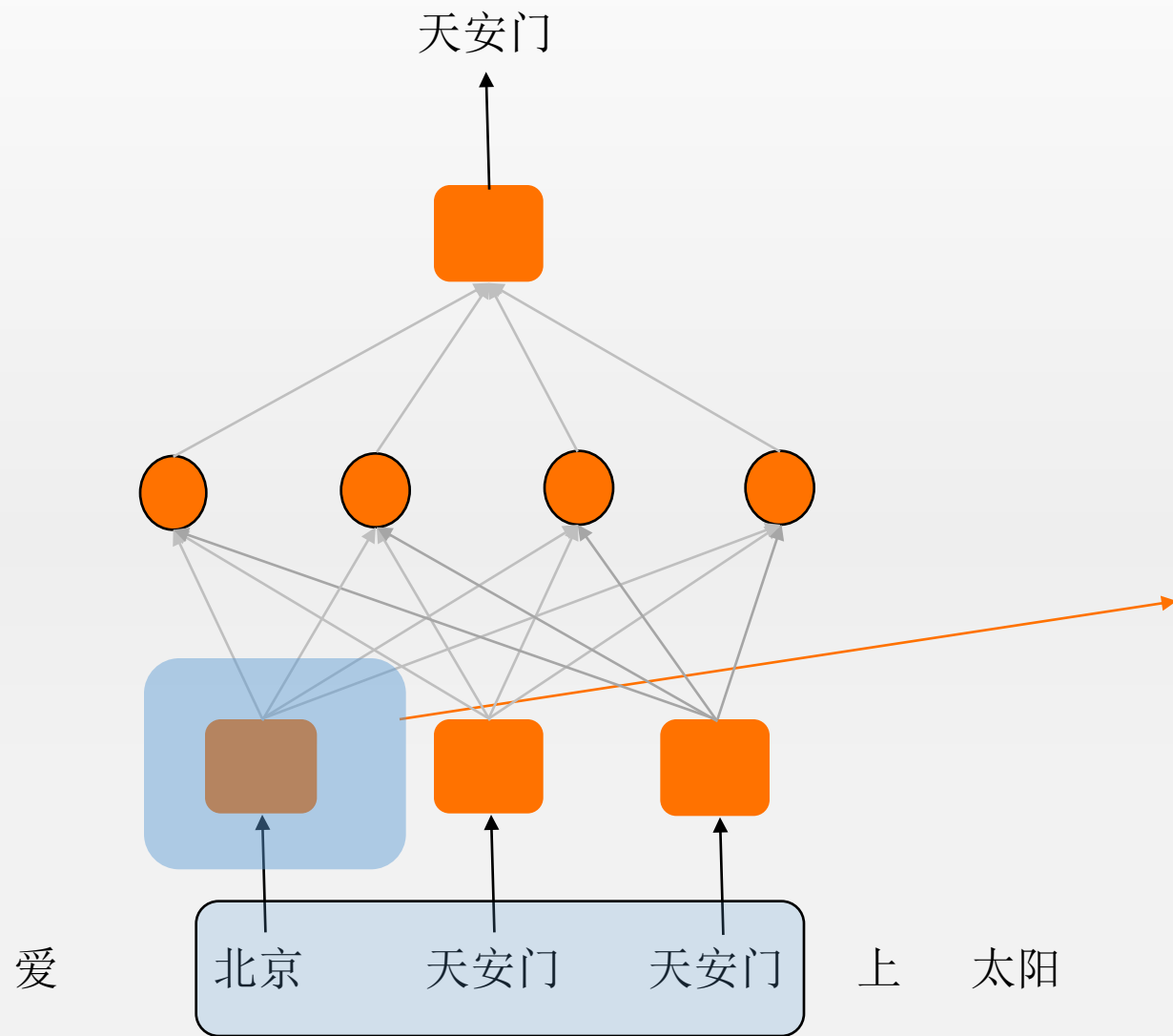
$$L = -\log P_{\text{天安门}}$$



# NPLM基本思想



# 词向量在哪儿？

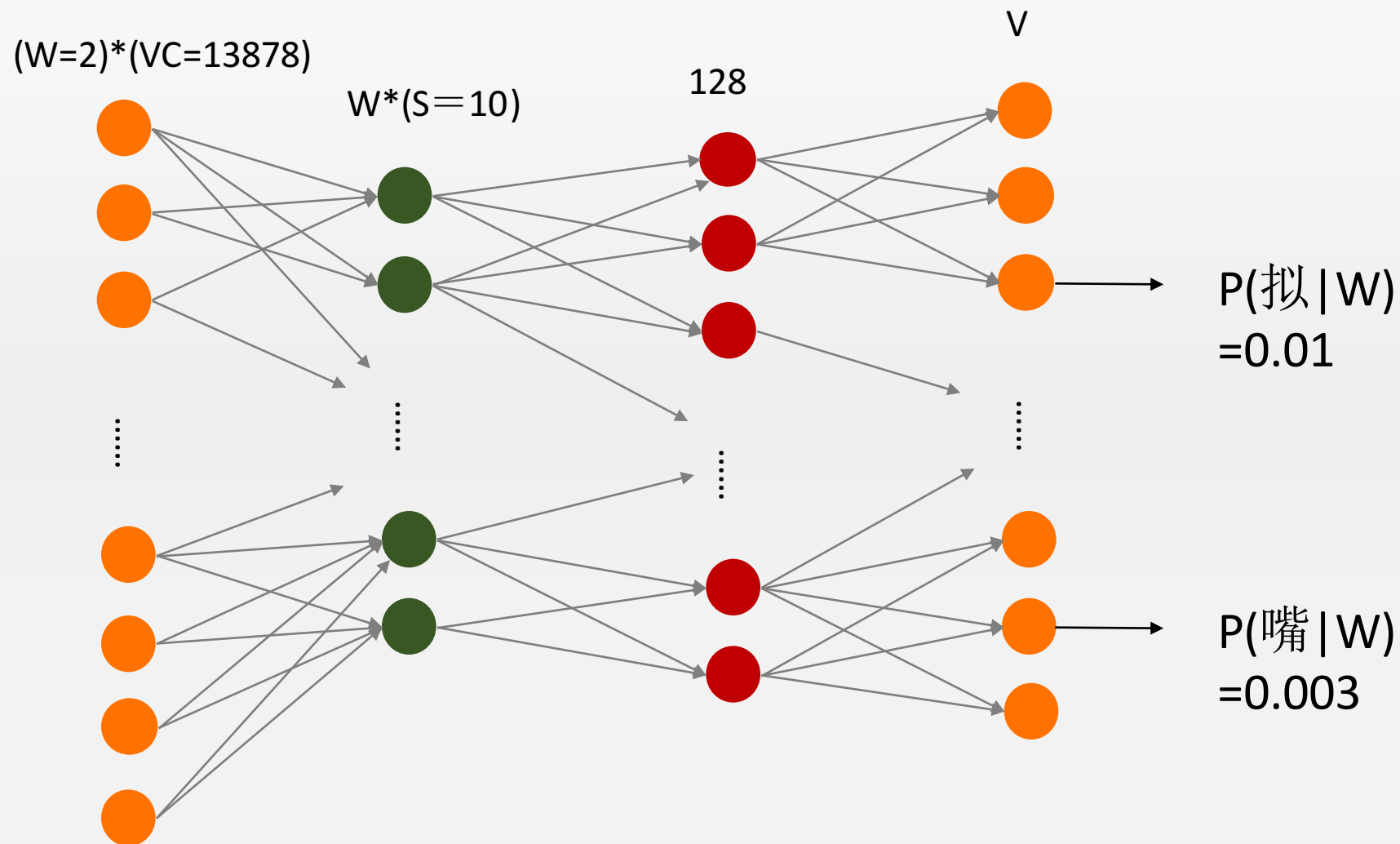




# 为什么概率语言模型可以工作？

- 相似的上下文会给出相似的预测
- 如果两个单词经常出现在相似的上下文，它们就相似
- 例如：
  - 火星是太阳系中的一个X
  - 木星是太阳系中的一个X

## 一个简单的NPLM网络架构



## 处理步骤



## 准备数据



# 分词

- 结巴分词：做最好的Python中文分词组件“Jieba”
- Lst=lcut("")

代码示例(分词)

```
#encoding=utf-8
import jieba

seg_list = jieba.cut("我来到北京清华大学",cut_all=True)
print "Full Mode:", "/ ".join(seg_list) #全模式

seg_list = jieba.cut("我来到北京清华大学",cut_all=False)
print "Default Mode:", "/ ".join(seg_list) #精确模式

seg_list = jieba.cut("他来到了网易杭研大厦") #默认是精确模式
print ", ".join(seg_list)

seg_list = jieba.cut_for_search("小明硕士毕业于中国科学院计算所，后在日本京都大学深造") #搜索引擎模式
print ", ".join(seg_list)
```

Output:

【全模式】： 我/ 来到/ 北京/ 清华/ 清华大学/ 华大/ 大学

【精确模式】： 我/ 来到/ 北京/ 清华大学

【新词识别】： 他，来到，了，网易，杭研，大厦 （此处，“杭研”并没有在词典中，但是也被Viterbi算法识别出来了）

【搜索引擎模式】： 小明，硕士，毕业，于，中国，科学，学院，科学院，中国科学院，计算，计算所，后，在，日本，京都，大学，日本京都

## 形成训练数据

- 扫描文档，形成三元组（窗口尺寸为2）
- 训练数据：
  - $((\text{word}(t-2), \text{word}(t-1)), \text{word}(t))$
- $((['\text{疯狂}', '\text{年代}'], '\text{中国}'), (['\text{年代}', '\text{中国}'], '1967'), (['\text{中国}', '1967'], '\text{年}'))$

## 建立字典

- Python中的字典:
- Dictionary是一种Python中的特殊存储结构, 由 (键, 值) 构成的集合
- 例如: `diction = {'name':10, 'height':20, 'width':30,...}`
- 所有元素的键不能重复, 值可以重复
- 访问的时候, 可以用`diction['name']`快速访问
- 我们扫描所有的文本, 为每一个单词建立了一个键值
  - `{'a':[19,2], 'about':[0, 20], 'the':[1, 30]}`

# 关于Embedding

`nn.Embedding(vocab_size, embedding_dim)`

单词表大小

嵌入维度

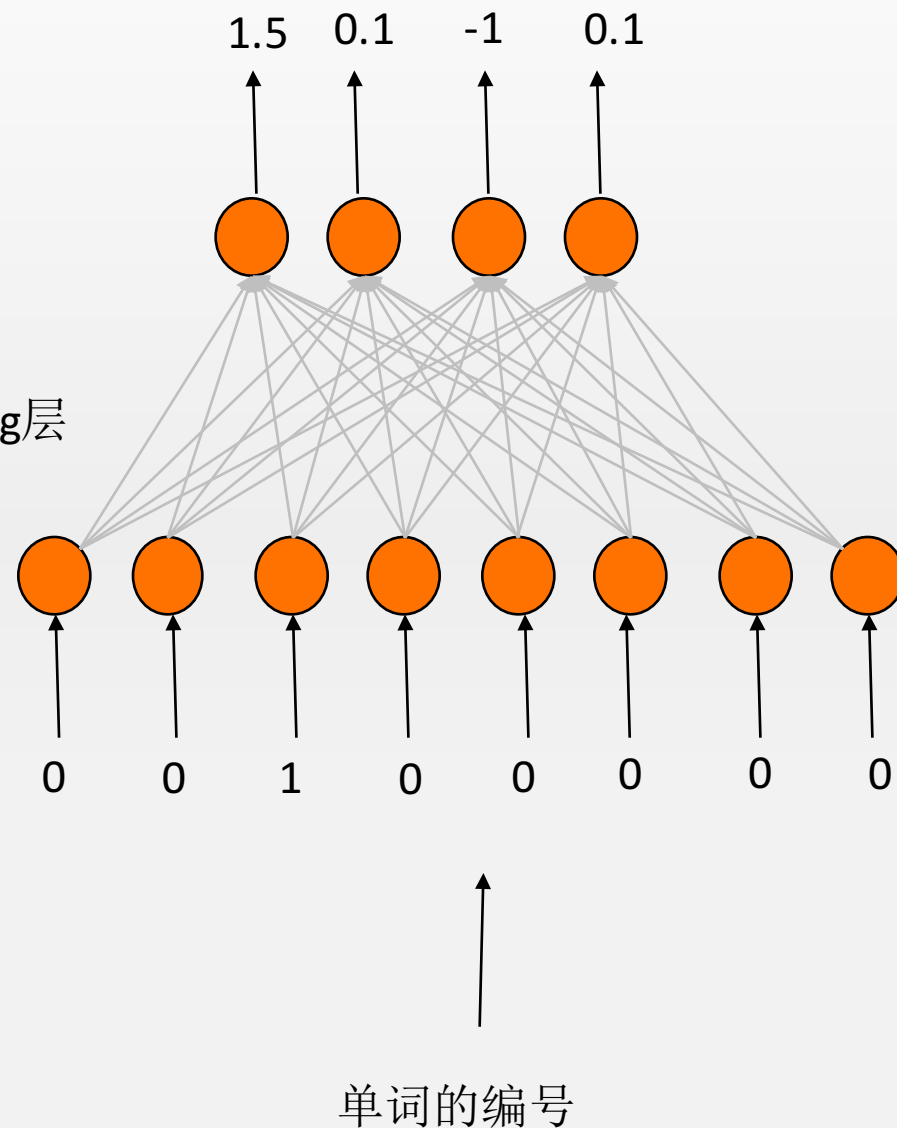
`x = self.embed(input)`

单词的向量表示

单词的编号

注意input的维度为: `batch_size*data_dimension`

Embedding层



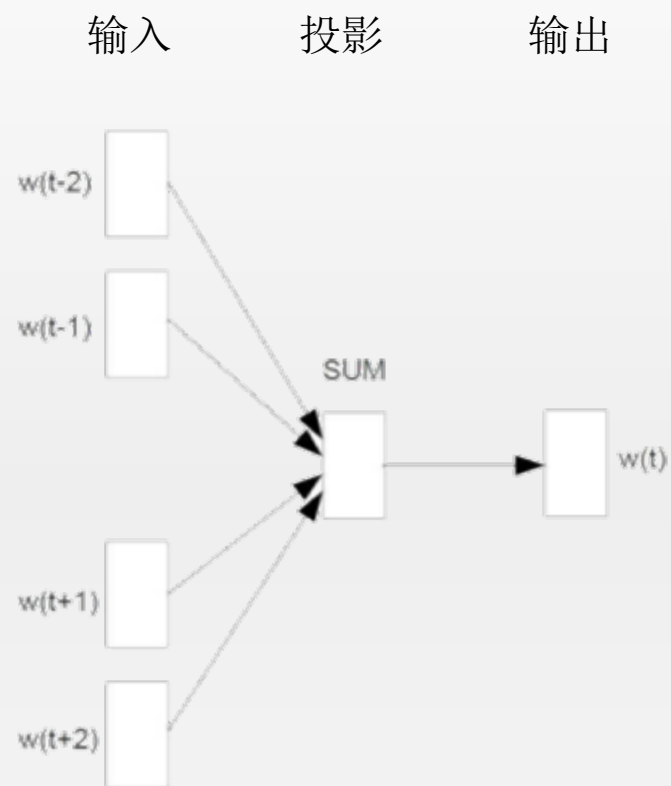


## 结果展示

- Sklearn包
- PCA算法降维，形成二维图形

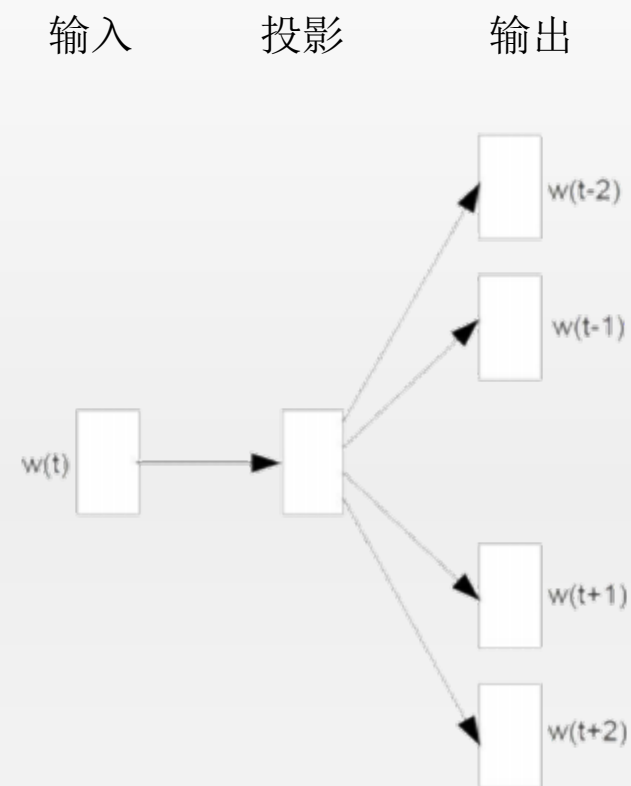


# Word2vec



CBOW 模型

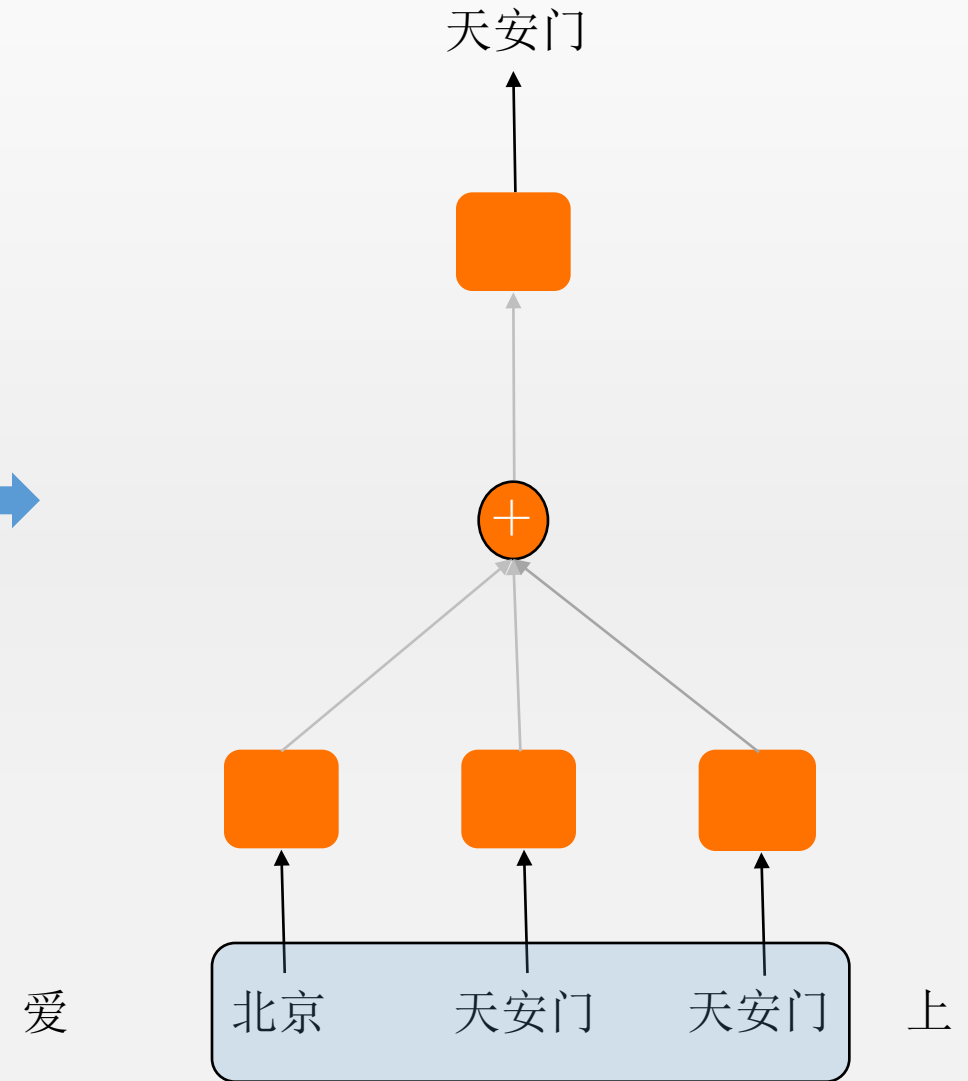
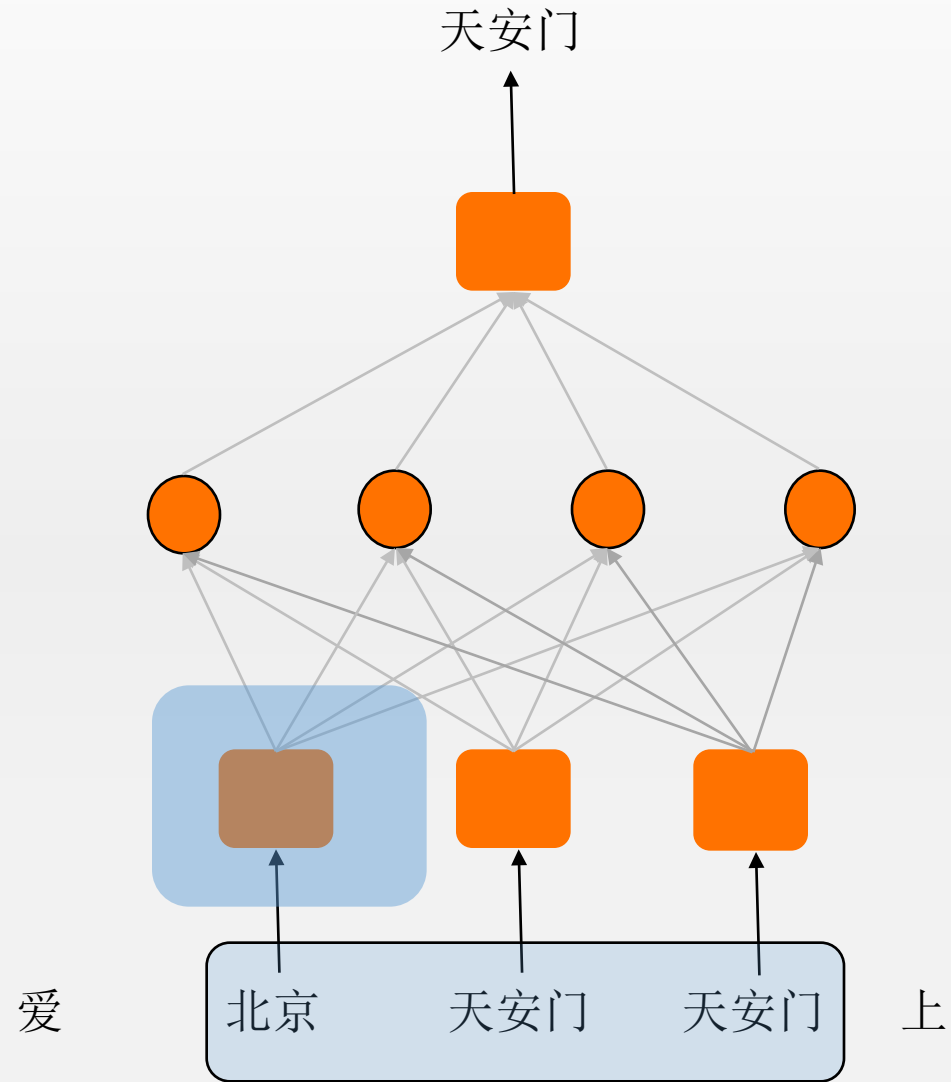
已知上下文预测当前词



Skip-gram 模型

已知当前词预测上下文

# Word2vec - CBOW (Continuous Bag of Words)



## SKIPGRAM的缺陷

- 待更新的参数有 $N \times V$ 个， $N$ 为词向量的维度， $V$ 为单词表中单词的数量
- 对于大语料来说， $N$ 很大，所以更新很慢
  - 每次只能更新一个单词
- 解决问题的途径：
  - Hierarchical softmax (层次软最大)
  - Negative sampling (负采样)

# Hierarchical softmax

- 解决思路是，将输出端做成一个Hauffman树——可以对所有单词进行最短编码的树状结构
- 分为两个步骤：
  - 构建Hauffman树：每个节点都有一个可训练的参数，所有的单词都位于叶节点上
  - 训练
- 好处：更新参数少

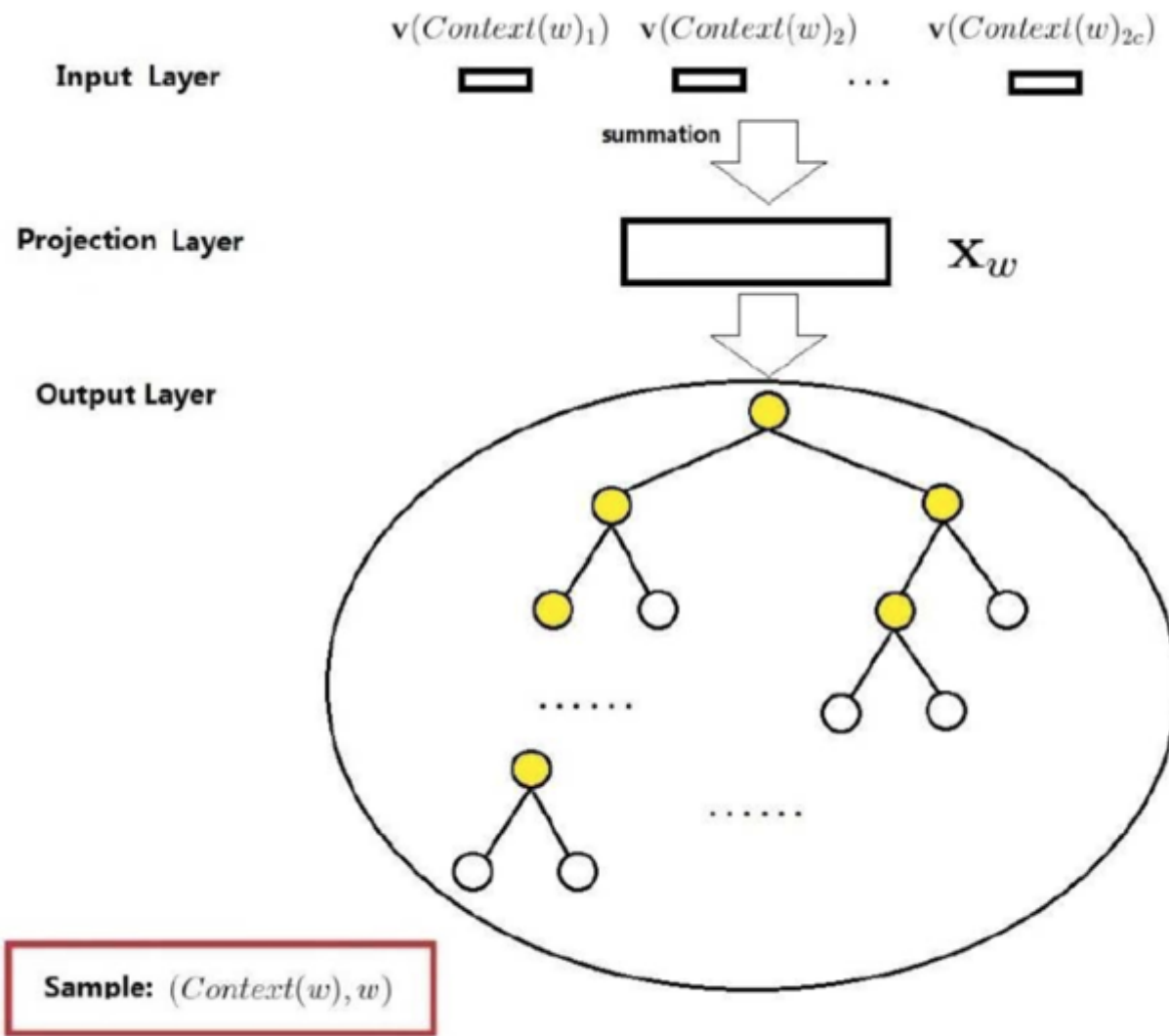


图 10 CBOW 模型的网络结构示意图

# Negative sampling (负采样)

- 基本思路
  - 将目标函数 (似然函数) 改为:
    - 对于在上下文中出现的单词, 概率最大
    - 对于随机采样生成的不在上下文中的单词, 概率最小

# Negative sampling (负采样)

- 基本思路
  - 将目标函数 (似然函数) 改为:
    - 对于在上下文中出现的单词, 概率最大
    - 对于随机采样生成的不在上下文中的单词, 概率最小
- 对于每一个训练样本:
  - 我爱北京天安→门
- 生成一个副样本:
  - 我爱北京天安→地, 瓜, .....



# Negative sampling (负采样)

- 基本思路
  - 将目标函数 (似然函数) 改为:
    - 对于在上下文中出现的单词, 概率最大
    - 对于随机采样生成的不在上下文中的单词, 概率最小

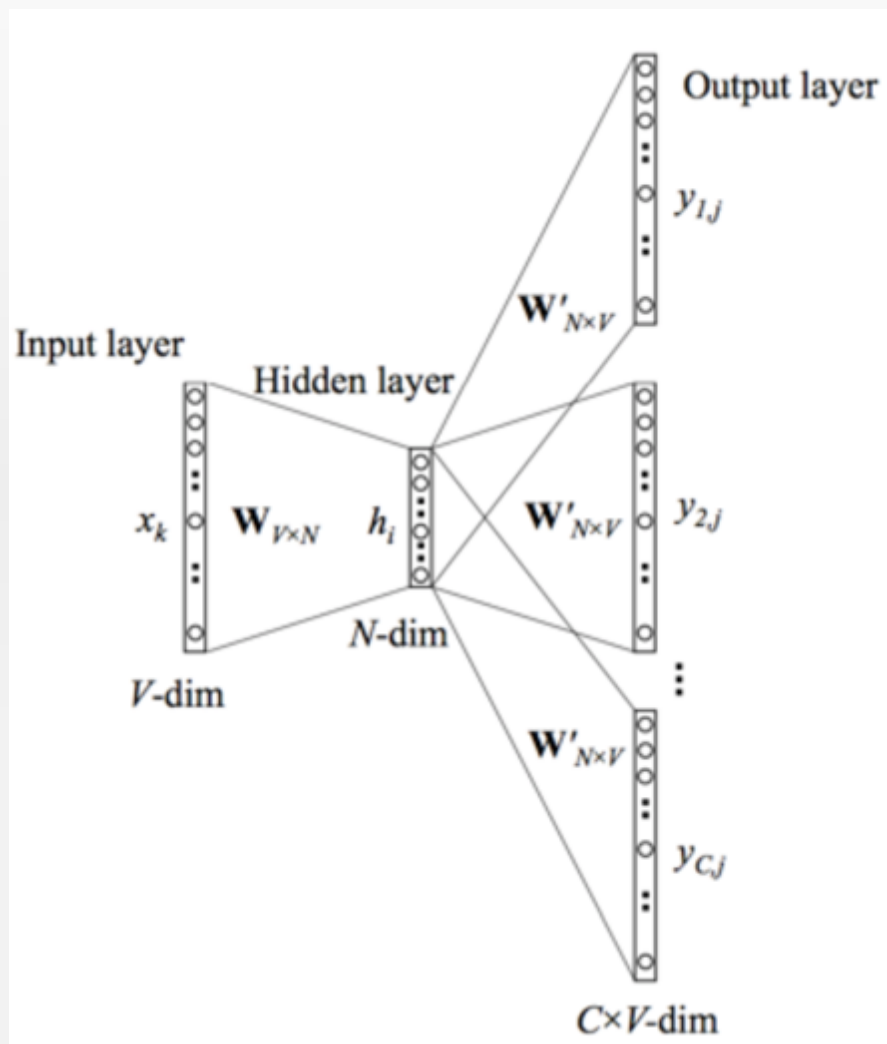
$$L = -\log P_{\text{天安门}}$$

# Negative sampling (负采样)

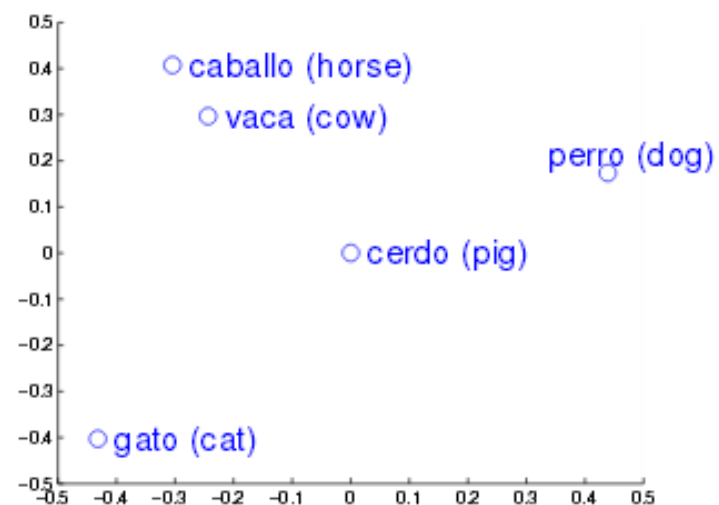
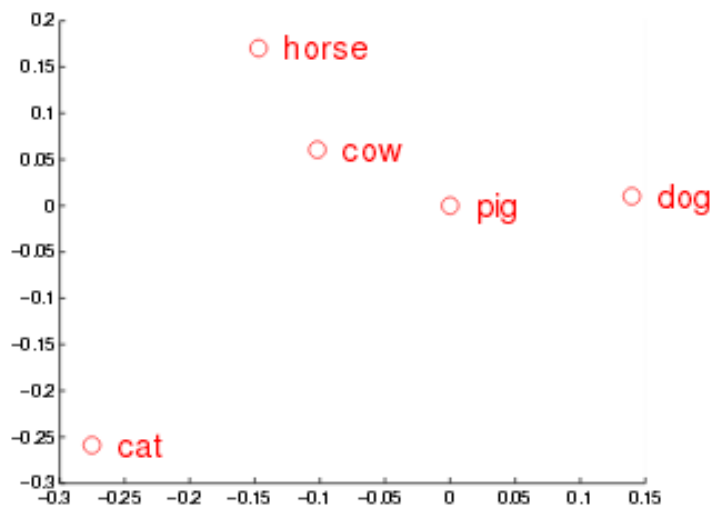
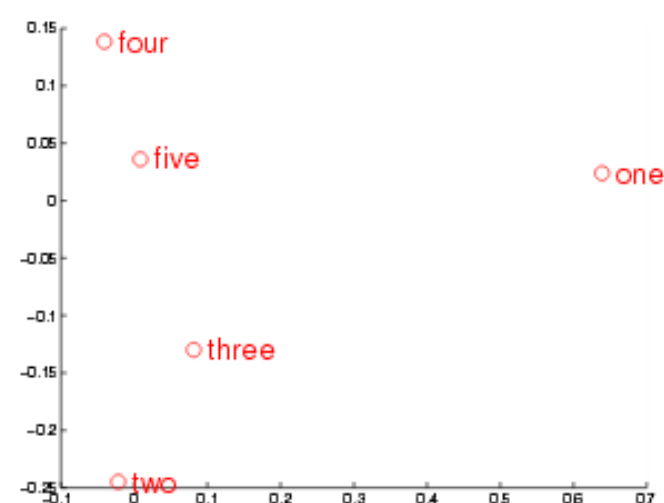
- 基本思路
  - 将目标函数 (似然函数) 改为:
    - 对于在上下文中出现的单词, 概率最大
    - 对于随机采样生成的不在上下文中的单词, 概率最小

$$L = -(\log P_{\text{天安门}} + \log(1 - P_{\text{地}}) + \log(1 - P_{\text{瓜}}))$$

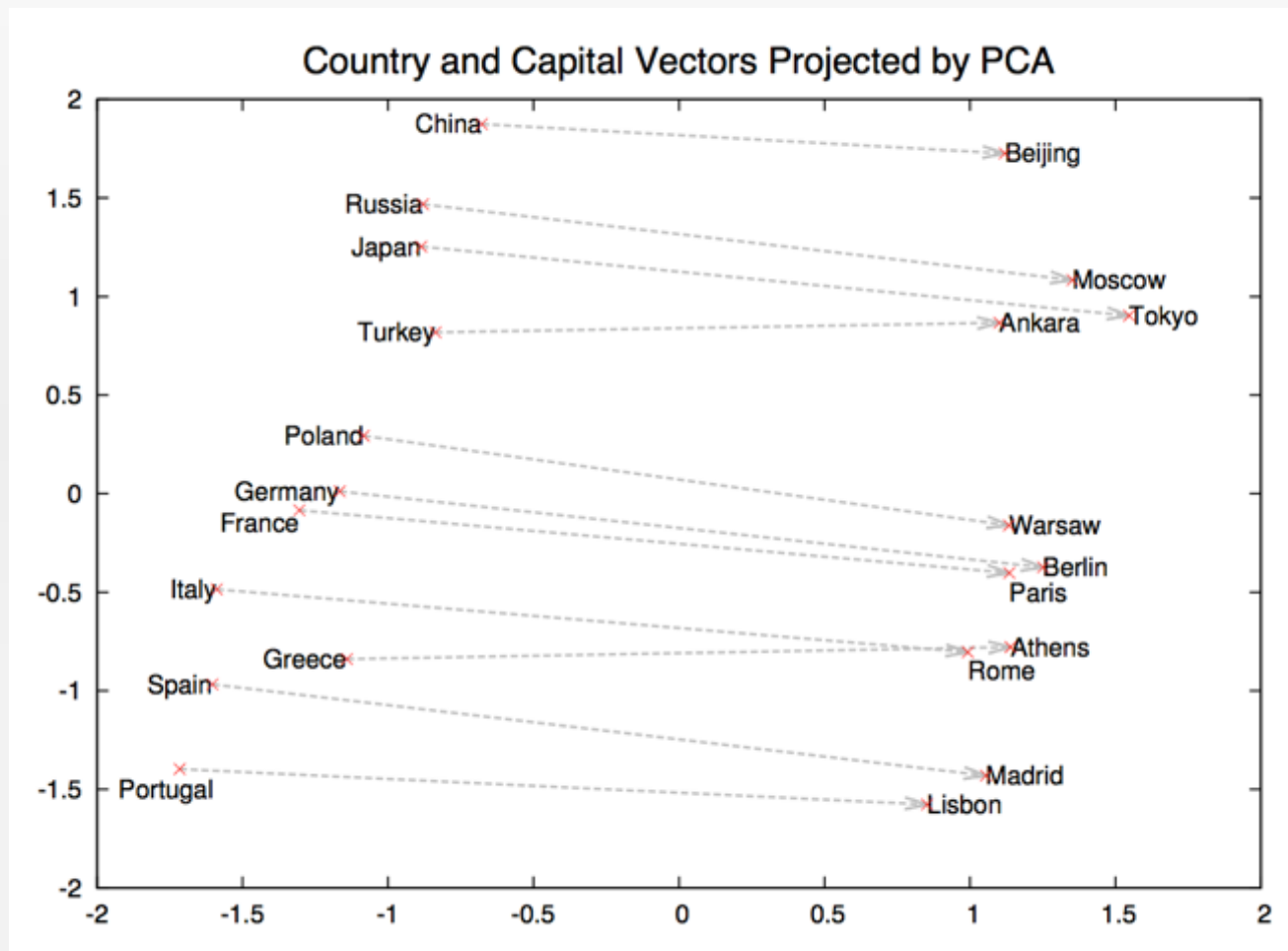
## SKIPGRAM如何工作



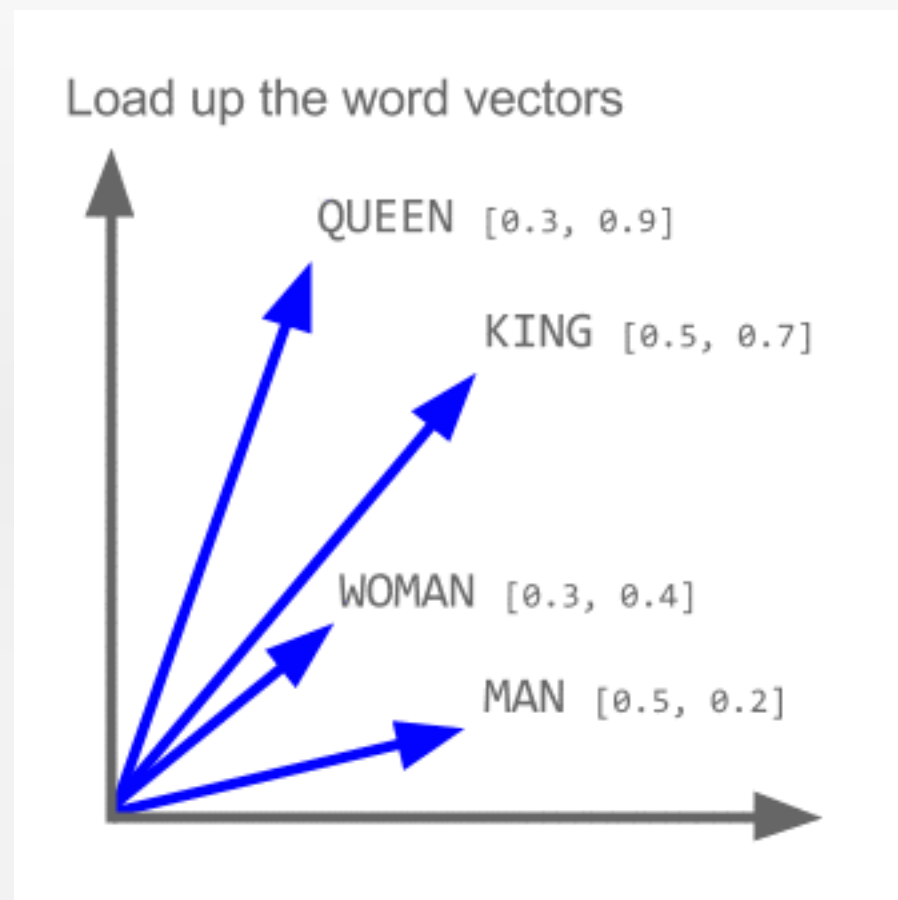
# Word2vec可以直接进行翻译



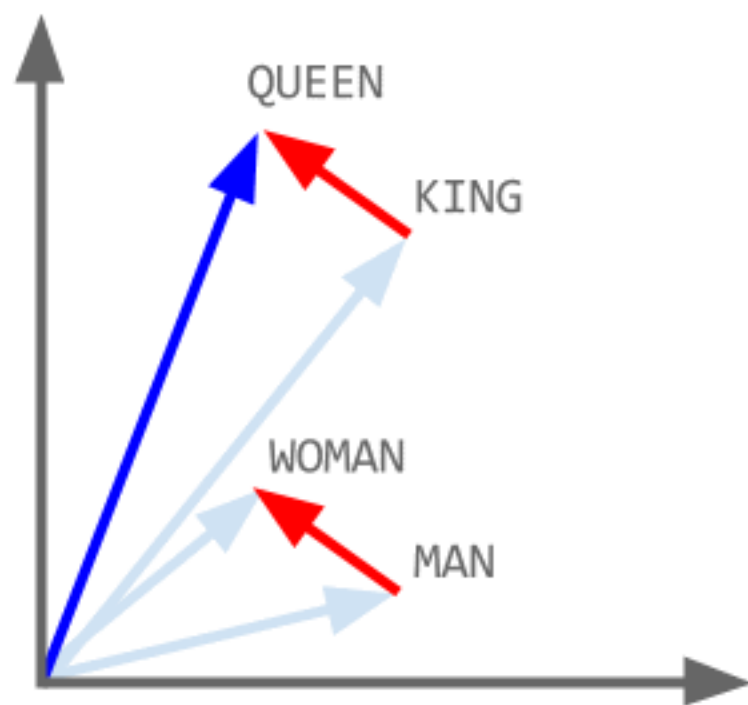
## 差向量可以表示语义关系



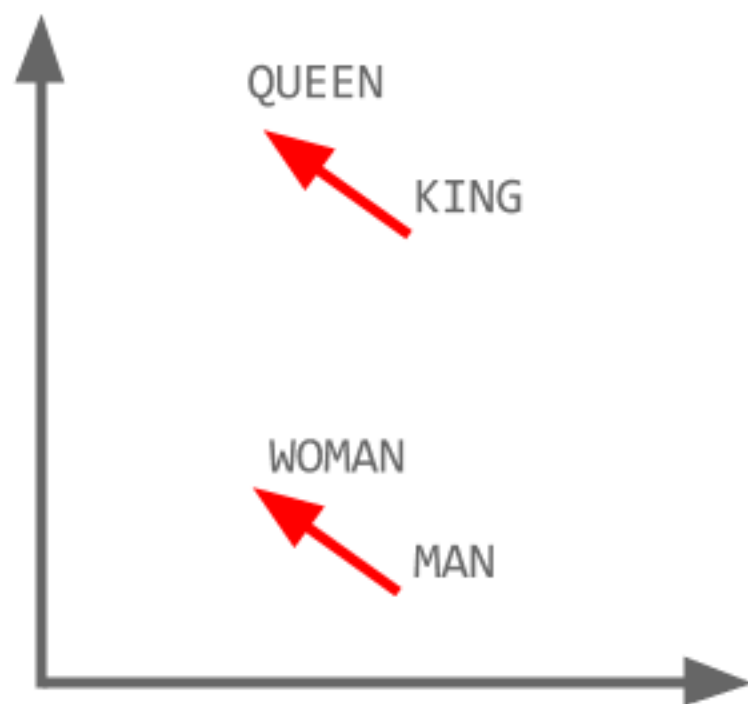
## 类比任务



So  $\text{king} + \text{man} - \text{woman} = \text{queen!}$

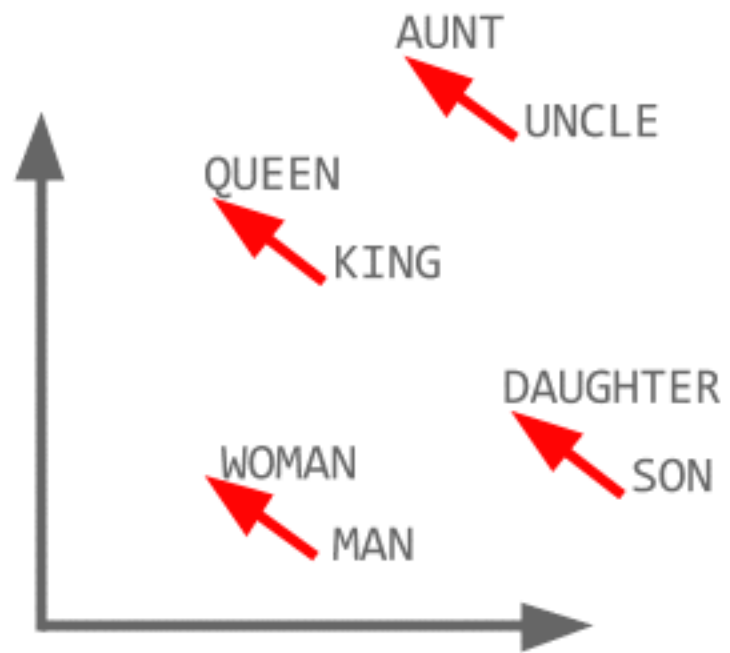


The **red direction** encodes gender





Which is consistent across all words



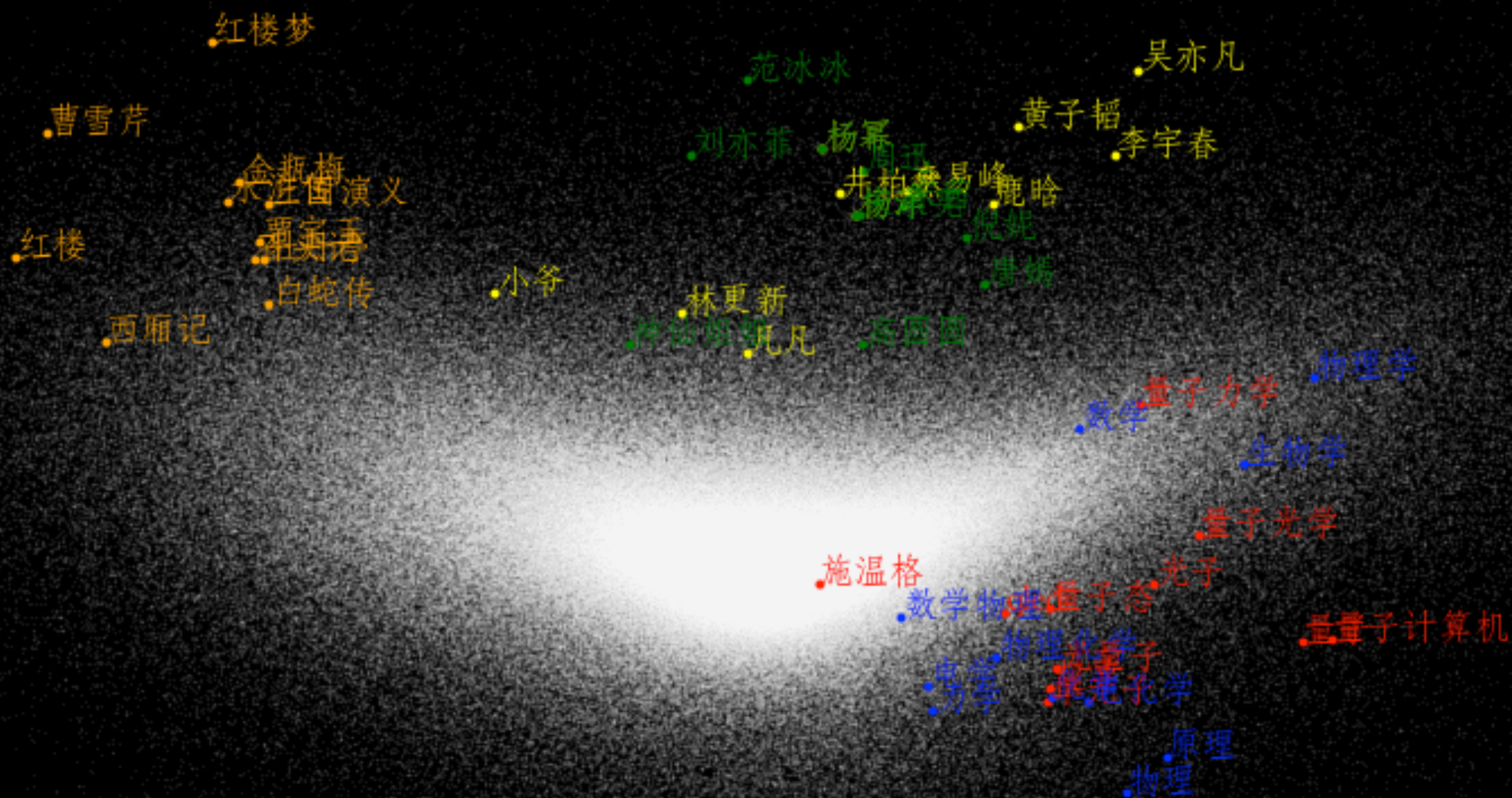
We have hundreds of **directions**  
encoding hundreds of ideas



HIGHER STATUS



MORE FEMININE



# 要点重述

- 词向量提出的目的
- 用语言模型训练词向量
  - 基本思想
  - 网络架构
  - 目标函数
- Word2Vec是一组模型和一组改进方法
  - CBOW
  - Skip-gram
  - Hierarchical Softmax
  - Negative Sampling
- Word2Vec的应用

# 作业：词向量翻译器

- 运用Google训练好的英文词向量：
  - 验证：man-woman=king-queen, man-woman=son-daughter, 等
- 分别将中文词向量中的一、二、三、.....和英文词向量中的one,two,three,.....画在二维平面上， 并比较二者
- 分别将中文词向量中的马、牛、驴、.....和英文词向量中的horse,cow,donkey,.....等动物名字画在二维平面上， 并比较二者
- 比较更多的中英文对译词对
- 训练一个神经网络， 做到自动将输入的英文词翻译为中文

# 敬请期待



• 张江