Web信息处理与应用

第六节

网页排序（上）

徐童　2024.10.17

上节回顾

- **更一般的检索方式**

- 采用**排序方式**代替严格匹配模式
  - 在排序检索中，系统根据文档与查询的**相关性排序**返回文档集合中的合适文档，而不是简单匹配查询条件
  - 自由文本查询：用户查询条件是**自然语言描述**，而不是由查询词项构造的表达式。
- 当系统给出的是**有序**的查询结果时，结果数目将不再是个问题
  - 着眼于给出Top N结果，而不是完整结果

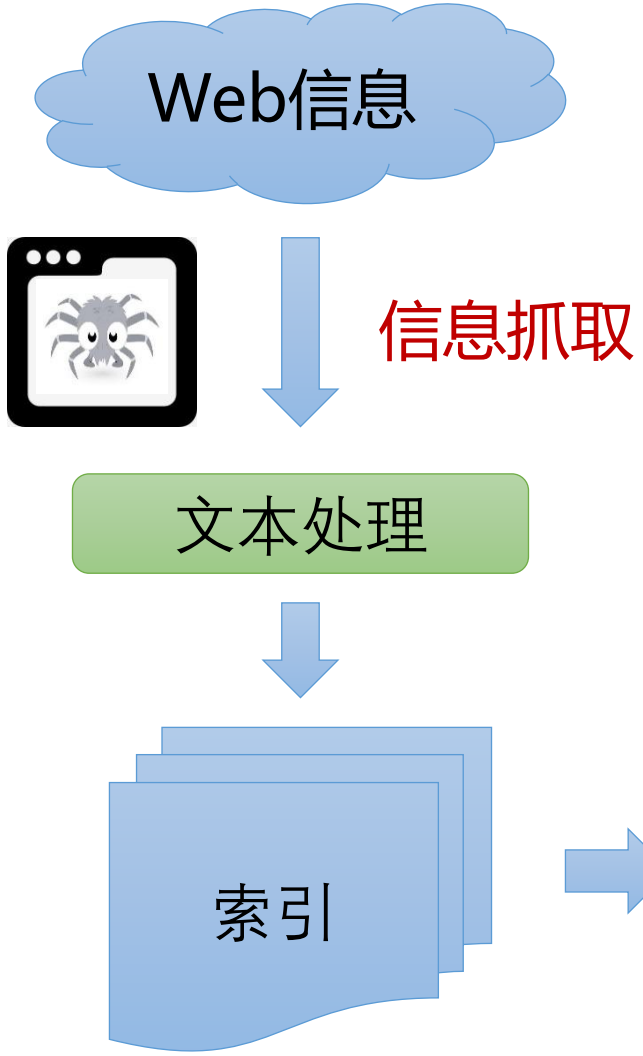
上节回顾

- **相关性反馈与查询优化**

- 用户在查询后标记相关/不相关，然后迭代更新查询，以获得更好的结果
- 相关性反馈的动机
  - 你也许无法表达想要找的内容，但是你至少能够判断所看到的内容
    - “为我提供更多*相似的文档*……”
  - **基于相关性反馈的迭代过程，本质上是网页排序不断优化的过程。**

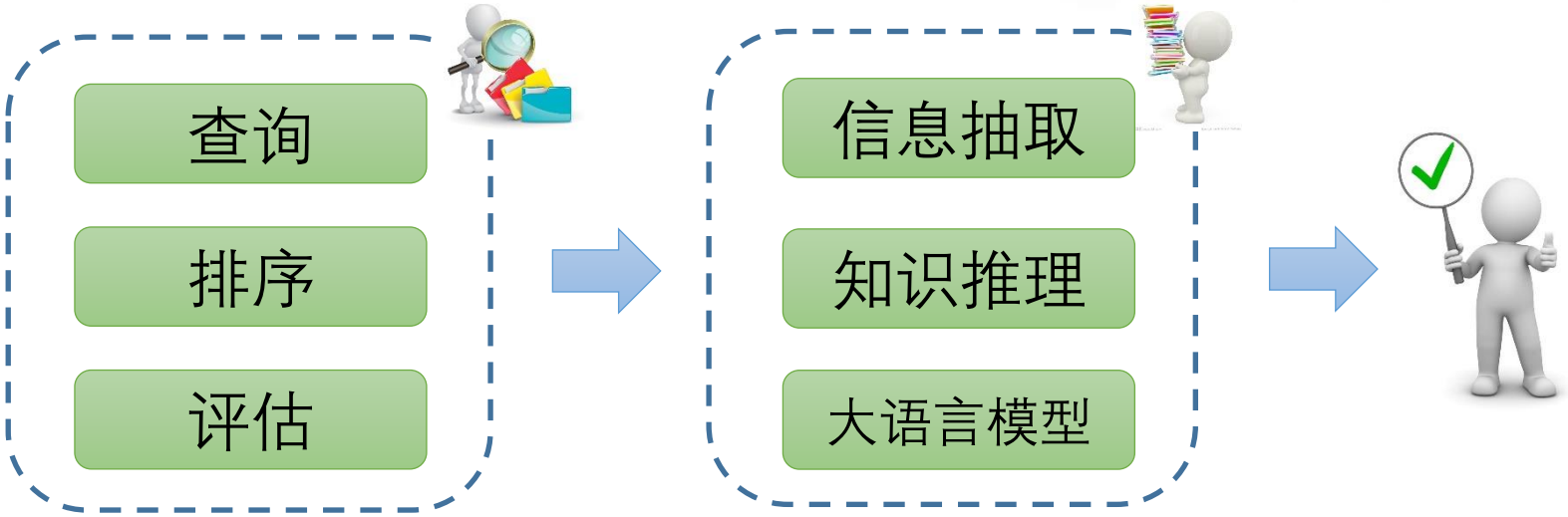
本节主题

- 本课程所要解决的问题



第六个问题：

如何对网页进行排序，使得用户能够最快获得所需信息？



问题背景

- **一个好的排序应该是怎样?**

- 如果得了牙疼，应该去看哪个医生呢?
  - A医生，既治眼病，又治胃病
  - B医生，既治牙病，又治眼病，还治胃病
  - C医生，专治牙病
- 按照布尔检索的思路，B和C都满足需求。
- 如果从医生专长的考虑，C是更好的选择。

- **一个好的排序应该是怎样？（续）**

- 如果我们有更丰富的信息？
  - B医生，从医二十年，经验丰富
  - C医生，只有五年从医经验
- 从择医的角度考虑，需要同时考虑专长与医术
- 对于**网页排序**而言，也是如此
  - 网页内容匹配程度 → 医生的专长
  - 网页内容的质量 → 医生的经验与水平

本节目录

- **信息检索与相关度计算**
  - **向量空间模型**
  - 文档匹配与迭代优化
- 文档表征技术初阶
- 预训练模型初阶

信息检索模型概述

- 信息检索模型是用来描述文档与查询的表示形式与相关性的框架
  - 信息检索的实质是对文档基于相关性进行排序
  - 好的信息检索模型，可以在理解用户的基础之上，产生近似用户决策的结果，从而在顶部返回最相关的信息
- 信息检索模型的形式化表述： [D, Q, F, R(Di, q)]
  - D：文档表达
  - Q：查询表达
  - F：查询与文档间的匹配框架
  - R：查询与文档间的相关性度量函数（Di与q分别表示特定文档与查询）

基本信息检索模型：布尔模型

- 最早的信息检索模型
  - 1957年，布尔逻辑是否可能应用于计算机信息检索已引起了讨论
- 以布尔模型为例，对应介绍信息检索模型的形式化表述
  - D：文档表达——词项的组合（注意是词项不是单词！）
  - Q：查询表达——布尔表达式（词项 + 布尔运算符）
  - F：完全匹配（二值匹配）
  - R：满足布尔表达式，相关性为1，否则为0（即使部分满足）
- 既然布尔检索的二值化匹配不合适，我们就需要为每个【查询-文档】赋予[0,1]之间的一个分值，来衡量并排序其匹配程度

• 量化方法（1）：Jaccard系数

- 布尔模型的一个假设：所有的查询条件都必须满足
  - 假如打破这一约束，能够实现相关性的[0,1]量化?
- 1901年，Paul Jaccard提出计算两个集合重合度的方法
  - 如果A与B为两个集合，那么有：

$$\mathrm{JACCARD}(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Paul Jaccard (1868-1944)

  - 显然，我们有： $\mathrm{JACCARD}(A,A)=1$， $\mathrm{JACCARD}(A,B)=0$ （如果$A \cap B = \emptyset$ ）

- **量化方法（1）：Jaccard系数**

- 由此，我们得到了一个描述集合重合度的，处于[0,1]之间的值
  - 注意前提：“集合”，即仍将文档视作词项的集合
- 通过计算查询与文档之间的重合度，可以初步估算文档的相关性

查询 **“ides of March”**

文档 **“Caesar died in March”**

$$\mathrm{JACCARD}(q,d) = 1/6$$

- **量化方法（1）：Jaccard系数**

- 然而，Jaccard系数显然仍过于简单、粗糙
  - 从完全匹配到部分匹配是一种进步，但查询词本身未做重要性区分
  - 没有仔细考虑文档的长度因素
    - 凭什么我词多我就要吃亏（分母大）
  - 不考虑词项频率，即词项在文档中的出现次数
  - 罕见词比高频词的信息量更大， Jaccard系数没有考虑这个信息

- **量化方法（2）：词项频率TF**

- 比Jaccard更进一步，如何通过词项区分文档权重?
- 一个直观的想法：查询词在文档中出现得越多，该文档越相关
  - 词项频率TF(t,d)，指词项 t 在文档 d 中出现的次数（Term Frequency）
  - 利用原始的TF值，可以粗略计算文档的相关性
    - 但存在一定问题，相关性与频率并不线性相关
      - 某个词在文档A中出现100次，在文档B中出现10次，A比B相关10倍?

- **量化方法（2）：词项频率TF**

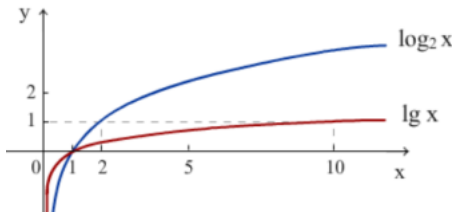
- 原始TF基础之上的改进：引入对数词频

$$wf_{t,d} = \begin{cases} 1 + \log_{10} tf_{t,d} & tf_{t,d} > 0 \\ 0 & otherwise \end{cases}$$

  - 通过这种方式，数量级的差异性所造成的影响变得更为缓和

- 由此，可以将文档与词项的匹配得分定义为所有同时出现在查询与文档中的词项其对数词频之和，即$\sum_{t \in q \cap d}(1 + wf_{t,d})$

  - 如果没有公共词项，显然得分为0



- **量化方法（2）：词项频率TF**

- 上上节课的回顾：基础检索中的关联矩阵
  - 给文档建立索引（Index），出现即为0，未出现即为1

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

- **量化方法（2）：词项频率TF**

- 基于词项频率的改进版本实例
  - 此时，不同文档的相关性出现了区分

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

- **量化方法（3）：文档频率DF**

- 如何再近一步?
  - 有些词在单个文档中出现的多，是因为这个词本身就很常用。
- 回顾：停用词的概念，为什么要去除停用词?
  - 数字、副词等与语义关系不大的词常作为停用词被处理。
  - 对最后结果的排序没什么贡献，反而可能产生干扰。
  - 特定领域中有其专用的停用词!
    - 如URL中的WWW，Wikipedia中的Wiki

- **量化方法（3）：文档频率DF**

- 一个什么样的词真正有区分度?
  - 香农（Shannon）有话说:

香农（C. E. Shannon）信息论

信息量是指从N个相等可能事件中选出一个事件所需要的信息度量或含量，也就是在辩识N个事件中特定的一个事件的过程中所需要提问“是或否”的最少次数. 香农信息论应用概率来描述不确定性。信息是用不确定性的量度定义的。
一个消息的可能性愈小，其信息量愈多；而消息的可能性愈大，则其信息愈少。事件出现的概率小，不确定性越多，信息量就大，反之则少。

- 总结起来：罕见词的信息量更为丰富，而频繁词的信息量相对较少
  - 相应的，如果查询中包含某个罕见词，则包含这个词的文档可能很相关

- **量化方法（3）：文档频率DF**

- 基于上述思想，引入新的度量机制：文档频率（Document Frequency）
  - df_t，指出现词项 t 的文档数量
    - df_t是与词项 t 的成反比的一个值
  - 相应的，我们一般采用逆文档频率（Inverse DF）来衡量

$$idf_t = log_{10}\frac{N}{df_t}$$

    - IDF由剑桥大学的Spock Jones于1972年提出，并由Salton推广
    - 需要注意，这里同样引入对数计算方式来抑制DF中数量级的影响

- **量化方法（3）：文档频率DF**

- 简单的 idf 计算实例（当 N = 1,000,000时）
  - 其中，过于频繁的词（如the）的作用被完全抹掉了

词项	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

- **量化方法（4）：集大成的tf-idf**

- TF与DF，从两个角度为衡量文档相关提供了量化依据
  - 如何将两者整合起来，实现更完整的描述?
  - 乘起来就完事了~

$$W_{t,d} = \left(1 + \log tf_{t,d}\right) \cdot \log \frac{N}{df_t}$$

  - 同时，解答了一个问题：什么样的词适合衡量文档相关性?
    - 在少数文档内多次出现的词
    - 指标随词项频率增大而增大，随词项罕见度增大而增大

- **量化方法（4）：集大成的tf-idf**

- 基于 tf-idf 的再次改进版本实例
  - 此时，不同文档的向量化表达已经呼之欲出

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

- **量化方法（5）：千呼万唤始出来的VSM**

- 从TF到DF再到tf-idf，我们对于文档中的词项有了愈加合理的描述
- 与此同时，相应的文档向量也具有了更丰富的信息量
- 由此，我们终于可以引出向量空间模型（Vector Space Model）

- 回顾的回顾：Salton在SMART系统中提出了著名的**向量空间模型**
  - 1960年代，康奈尔大学的Gerard Salton研发了SMART系统，被视作信息检索的鼻祖

Gerard Salton (1927-1995)

- **量化方法（5）：千呼万唤始出来的VSM**

- 向量空间模型（Vector Space Model，VSM）
  - 每个文档和查询视作一个词项权重构成的向量
  - 查询时通过比较向量之间相似性来进行匹配
- 如果用开头提到的形式化表述加以概括?
  - D：文档表达，每个文档可视作一个向量，其中每一维对应词项的tf-idf值
  - Q：查询表达，可视作一个向量，其中每一维对应词项的tf-idf值
  - F：非完全匹配方式
  - R：使用两个向量之间的相似度来度量文档与查询之间的相关性

- **量化方法（5）：千呼万唤始出来的VSM**

- 文档表示示例

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

- 所以，D1对应的向量为(5.25, 1.21, 8.59, 0, 2.85, 1.51, 1.37)

- **量化方法（5）：千呼万唤始出来的VSM**

- 向量空间模型的总结
  - 首先，将文档与查询表示成词项的tf-idf权重向量（也可采用其他方法）
  - 其次，计算两个向量之间的某种相似度（如余弦相似度）
  - 最后，按相似度大小进行排序，将Top-K的文档返回给用户
- 优点：简洁直观，可以支持多种不同度量或权重方式，实用效果不错
- 缺点：缺乏语义层面的理解和匹配，同时依赖tf-idf值也可能造成干扰
  - 用户无法描述词项之间的关系，词项之间的独立性假设实际上不成立
    - 例如：贝利 + 足球，郎平 + 排球

本节目录

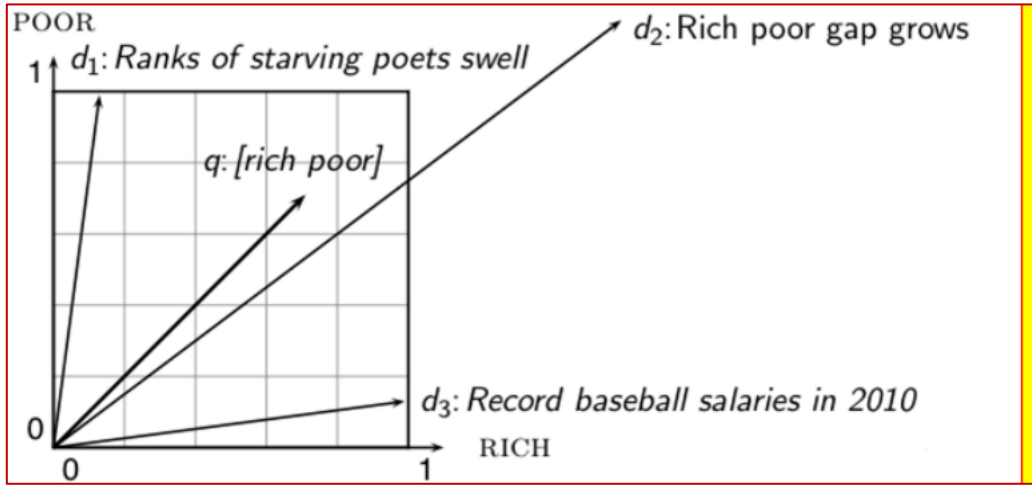
- **信息检索与相关度计算**
  - 向量空间模型
  - **文档匹配与迭代优化**
- 文档表征技术初阶
- 预训练模型初阶

- **查询与文档的匹配**

- 如何计算查询与文档之间的相似度?
  - 最基本的方法：欧氏距离

$$dist(\vec{x},\vec{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$$

  - 然而，欧氏距离并不是一个好的选择，它对于向量长度（文档长度）非常敏感



尽管查询*q*和文档d_2的词项分布非常相似，但是采用欧氏距离计算它们对应向量之间的距离非常大

- **查询与文档的匹配**

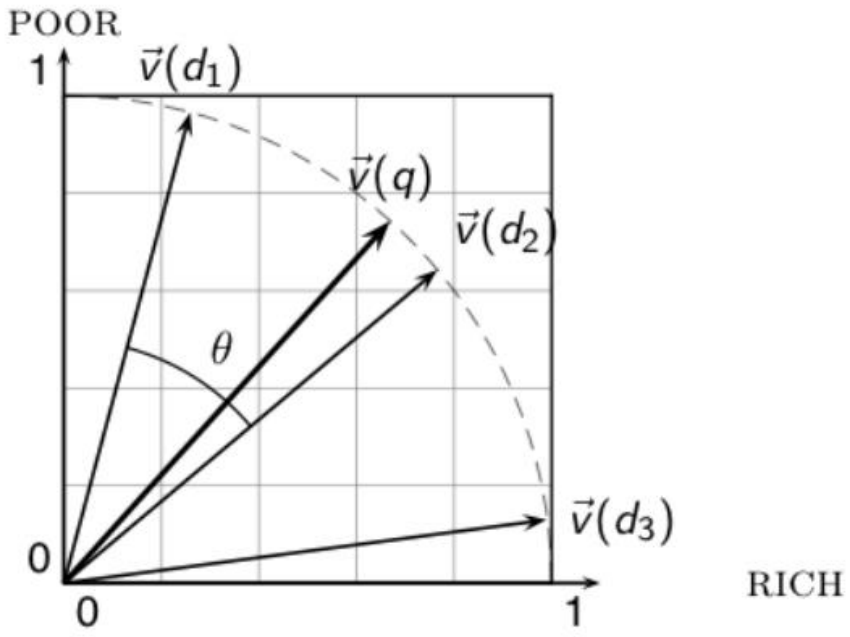
- 事实上，类似欧氏距离缺陷这样的问题，在许多应用场景都会出现
  - 例如，豆瓣电影评分中某用户打分均值或方差的影响
  - 假如有某部电影，三个用户的打分分别如下：
    - A：10， 8， 9
    - B：4， 2， 3
    - C：8，10，9
    - 哪两个用户的偏好更为一致?

- **查询与文档的匹配**

- 更为合适的方案：余弦相似度
  - 按照文档向量与查询向量的夹角大小来计算

$$\cos\left(\vec{q},\vec{d}\right)=\frac{\sum_{i=1}^{|V|}q_i d_i}{\sqrt{\sum_{i=1}^{|V|}q_i^2}\sqrt{\sum_{i=1}^{|V|}d_i^2}}$$

  - 显然，向量越一致，夹角越小，cosine值越高
    - 相应的，它们之间的相似度也就越高
    - 即使这种情况，它们的欧氏距离可能很大



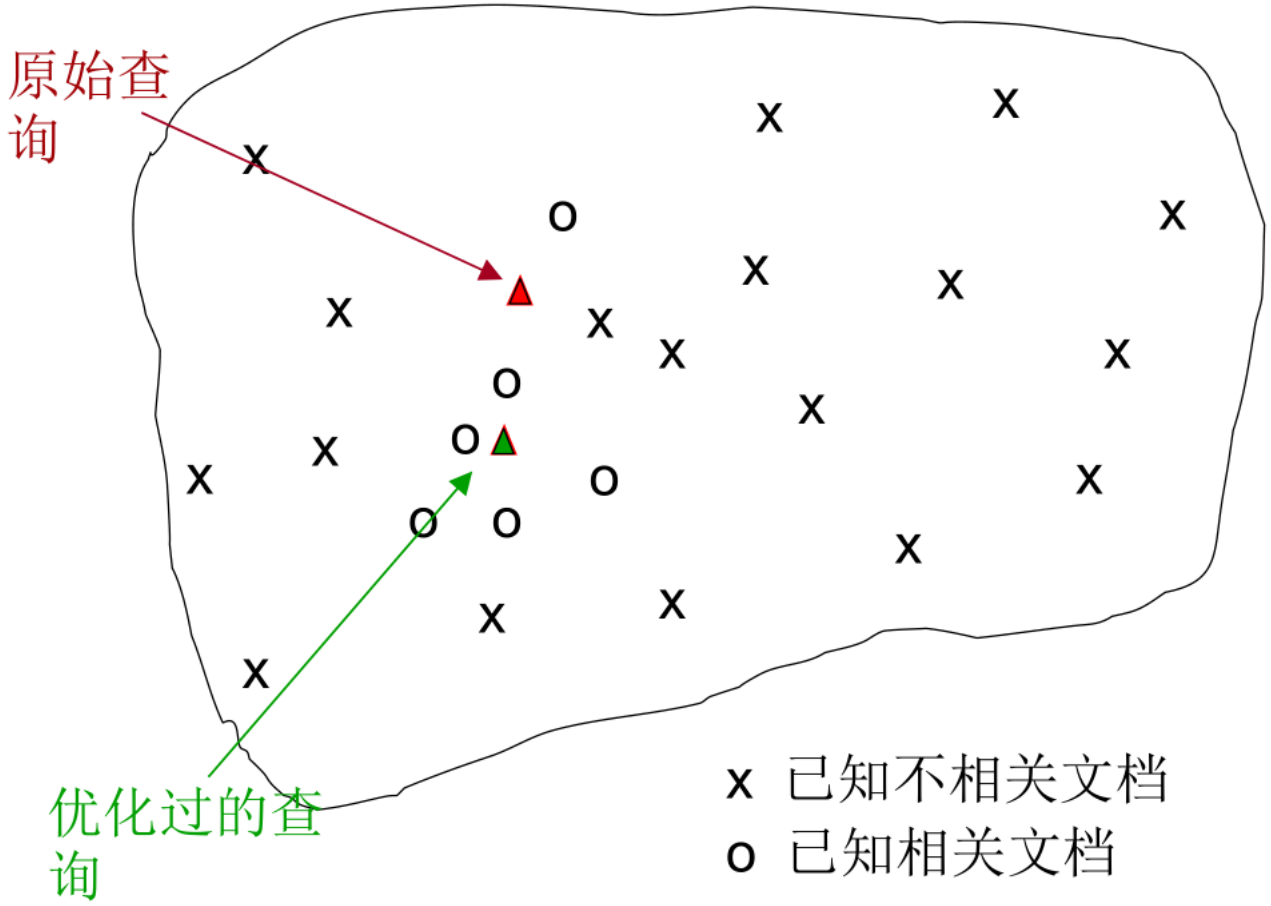
- **查询与文档的匹配**

- 余弦相似度的计算实例
  - 假如有两个文档和一个查询
    - D1 = (0.5, 0.8, 0.3)
    - D2 = (0.9, 0.4, 0.2)
    - Q = (1.5, 1.0, 0)

$$
\begin{aligned}
Cosine(D_1, Q) &= \frac{(0.5 \times 1.5) + (0.8 \times 1.0)}{\sqrt{(0.5^2 + 0.8^2 + 0.3^2)(1.5^2 + 1.0^2)}} \\
&= \frac{1.55}{\sqrt{(0.98 \times 3.25)}} = 0.87 \\
Cosine(D_2, Q) &= \frac{(0.9 \times 1.5) + (0.4 \times 1.0)}{\sqrt{(0.9^2 + 0.4^2 + 0.2^2)(1.5^2 + 1.0^2)}} \\
&= \frac{1.75}{\sqrt{(1.01 \times 3.25)}} = 0.97
\end{aligned}
$$

- **基于迭代的查询意图更新**

- 上节课我们提到，用户的查询意图可能无法一蹴而就，而需要通过相关性反馈实现逐步更新
- 在本质上，这一过程是使查询意图的表达逐步逼近用户目标文档的过程

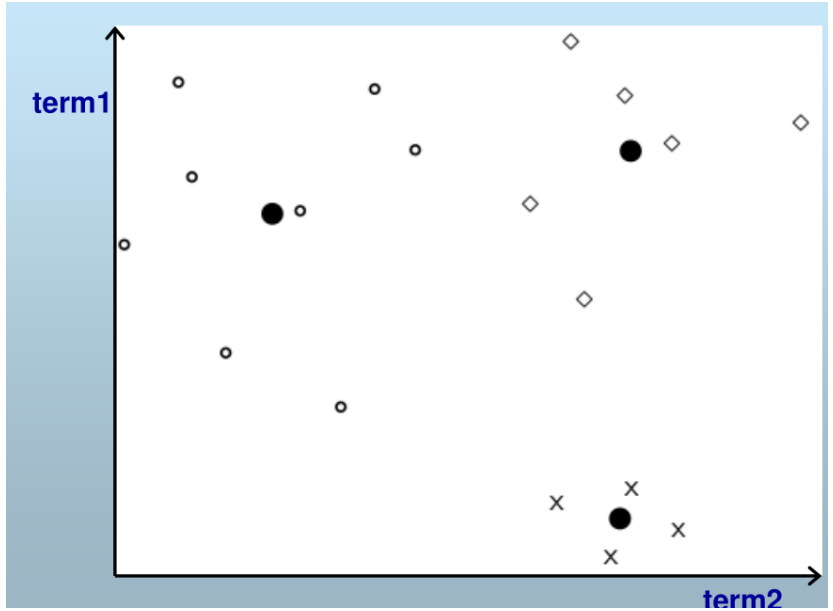


- **基于迭代的查询意图更新**

- 如何接近目标文档？可以从目标文档的质心为出发点
  - 某种意义上说，由向量表示的文档，可以视作高维空间中的一个点
  - 由此，“质心”就是一系列点（文档）的重心
    - 我们可以用如下公式来计算一类文档的质心

$$\vec{\mu}(C) = \frac{1}{|C|}\sum_{d \in C}\vec{d}$$

    - 其中，C是文档的集合



- **基于迭代的查询意图更新**

- 1970年前后，罗基奥（Rocchio）提出了相关性反馈技术，并应用于Salton领导研制的SMART系统中。
- Rocchio算法提供了一种将相关反馈信息融入到向量空间模型的办法
  - 其思想在于试图找到一个完美的最优查询向量，以满足以下目标

$$\vec{q}_{opt} = \arg\max_{\vec{q}} [\cos(\vec{q}, \vec{\mu}(C_r)) - \cos(\vec{q}, \vec{\mu}(C_{nr}))]$$

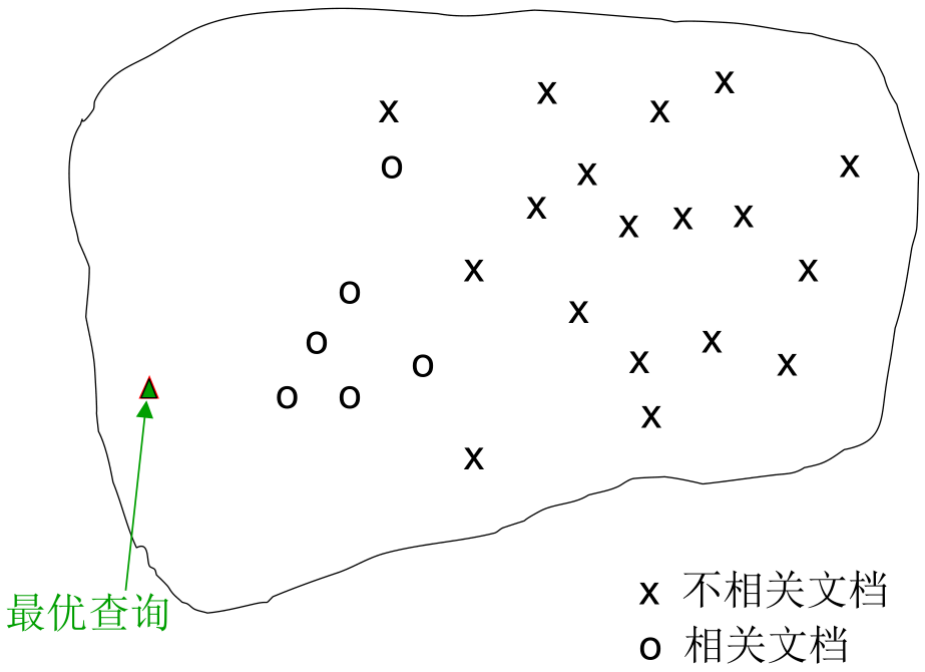
  - 即：使得查询尽可能离与之相关的文档更近，离与之不相关的文档更远。

- **基于迭代的查询意图更新**

- 其中，理想情况是，在可知完整的相关/不相关文档集合的情况下，可以使用以下公式来获得一个完美的查询：

$$\vec{q}_{opt} = \frac{1}{|C_r|}\sum_{\vec{d}_j \in C_r}\vec{d}_j - \frac{1}{|C_{nr}|}\sum_{\vec{d}_j \notin C_r}\vec{d}_j$$

- 问题在于，我们事实上并不可能获得完整的相关/不相关文档集合。



- **基于迭代的查询意图更新**

- Rocchio算法（1971）实际使用的近似方法如下：

$$\vec{q}_m = \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}}\vec{d}_j$$

  - 其中，D_r为已知相关文档的向量集合，D_{nr}为已知不相关文档的向量集合
  - q_0为初始查询向量。α、β、γ为权重，根据手工调节或经验设定
  - 由此，新的查询向量将逐渐向相关文档向量移动，远离不相关文档向量

- **基于迭代的查询意图更新**

- Rocchio算法示例

Rocchio 算法示例

$$\text{query vector} = \alpha \cdot \text{original query vector} + \beta \cdot \text{positive feedback vector} - \gamma \cdot \text{negative feedback vector}$$

Typically, $\gamma < \beta$

Original query	0	4	0	8	0	0	$\alpha = 1.0$	0	4	0	8	0	0	
Positive Feedback	2	4	8	0	0	2	$\beta = 0.5$	1	2	4	0	0	1	(+)
Negative feedback	8	0	4	4	0	16	$\gamma = 0.25$	2	0	1	1	0	4	(-)
New query								-1	6	3	7	0	-3	
								0	6	3	7	0	0	

- **基于迭代的查询意图更新**

- 正反馈 vs 负反馈
  - 正反馈的价值往往大于负反馈
    - 用户更关心符合需求的标准答案，而不是错误答案
    - 相应的，可以通过设置 $\beta > \gamma$ 来给予正反馈更大的权重
  - 很多系统甚至只允许正反馈，即 $\gamma = 0$
    - 收集真正的负反馈往往比较困难（上节课有提到过）

本节目录

- 信息检索与相关度计算
  - 向量空间模型
  - 文档匹配与迭代优化
- **文档表征技术初阶**
- 预训练模型初阶

• TF-IDF技术的局限性

- 使用 tf-idf 来表示词项简单快速而且容易理解
- 但是，tf-idf仅以“词频”度量词的重要性，无法体现词项之间的关联关系
  - 描述关联关系最基本的方法：共现矩阵

语料：

He is not intelligent.

He is not lazy.

He is intelligent.

He is Smart.

	He	is	not	lazy	intelligent	smart
He	0	4	2	1	2	1
is	4	0	1	2	2	1
not	2	1	0	1	0	0
lazy	1	2	1	0	0	0
intelligent	2	2	0	0	0	0
smart	1	1	0	0	0	0

• TF-IDF技术的局限性

- 共现矩阵的缺陷是明显的
  - 问题1：数据规模过大，存储以及处理困难
    - 处理方案：通过SVD、PCA等对该矩阵进行降维
    - 但这样会引来新的问题：原有的词项与矩阵一一对应关系被破坏
  - 问题2：无法体现词项的位置信息以及词项与上下文的关系
    - 如何区分以下表达的区别?

“武松打虎” ⟷ “虎打武松”

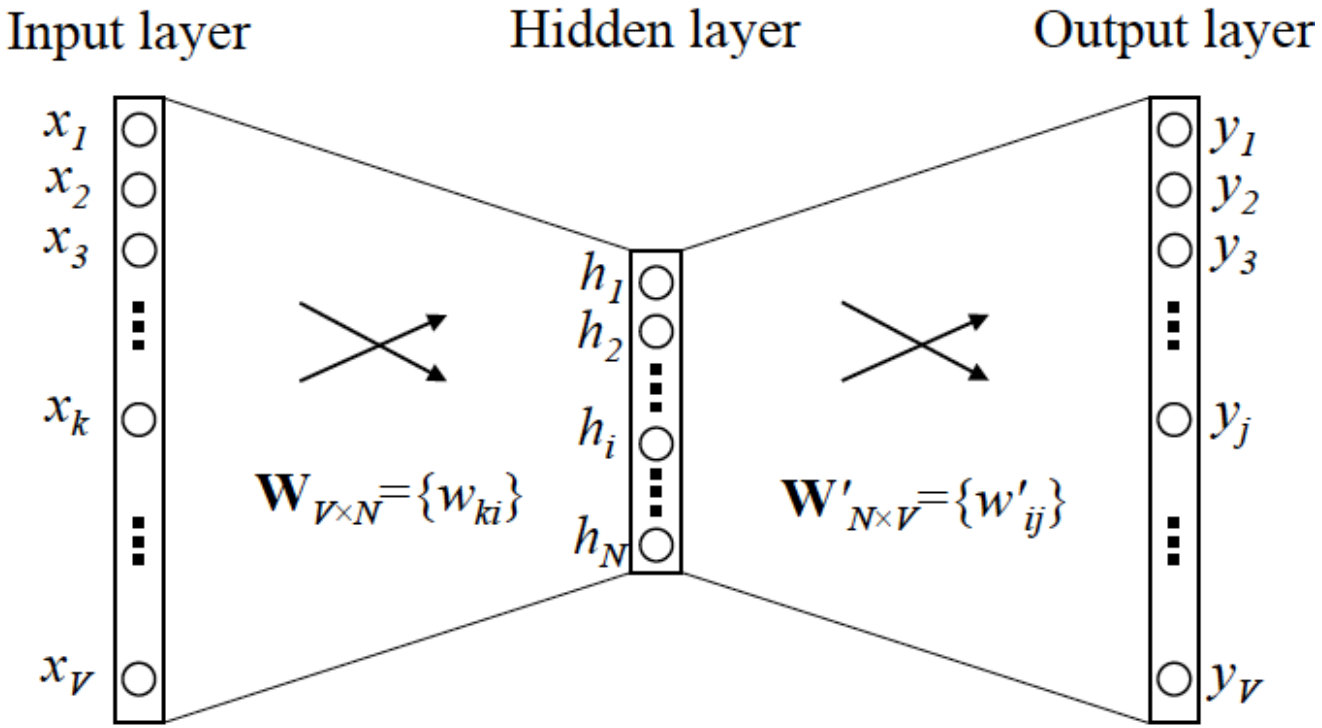
    - 位置关系可以引入扩展倒排表加以解决，但显然过于复杂

- **TF-IDF技术的局限性**

- 使用 tf-idf 来表示词项简单快速而且容易理解
- 但是，tf-idf仅以“词频”度量词的重要性，无法体现词项的位置信息以及词项与上下文的关系：
  - E.g. “武松打虎” ⟷ “虎打武松”
- 因此，我们需要一种能够表示出文本上下文关系的词项表示方法

- **如何描述词项之间的语义关联**

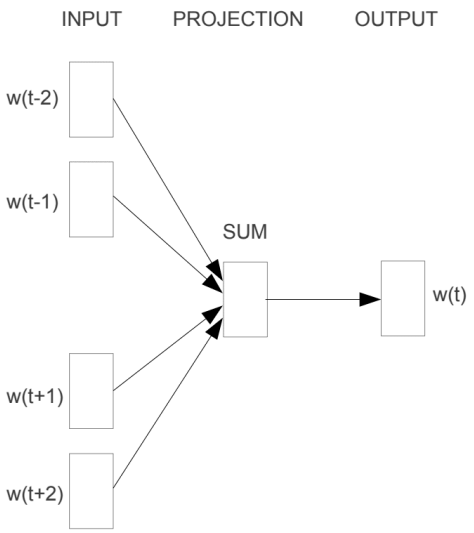
- 因此，我们需要一种能够表示出文本上下文关系的词项表示方法
- Word2vec模型的出现：直接面向文本序列进行建模



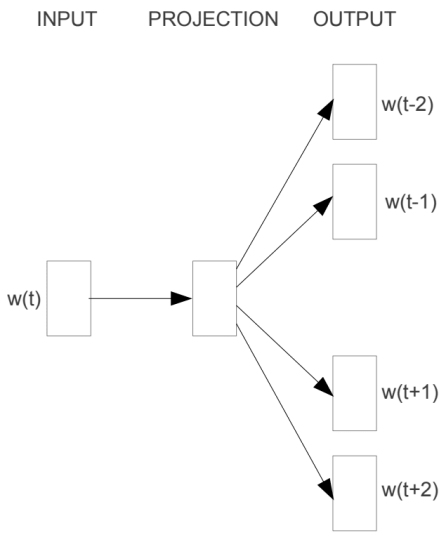
- **如何描述词项之间的语义关联**

- Word2vec模型的两种设计思路
  - 根据上下文预测中心词（CBOW）

  - 根据中心词预测上下文（Skip-gram）



CBOW

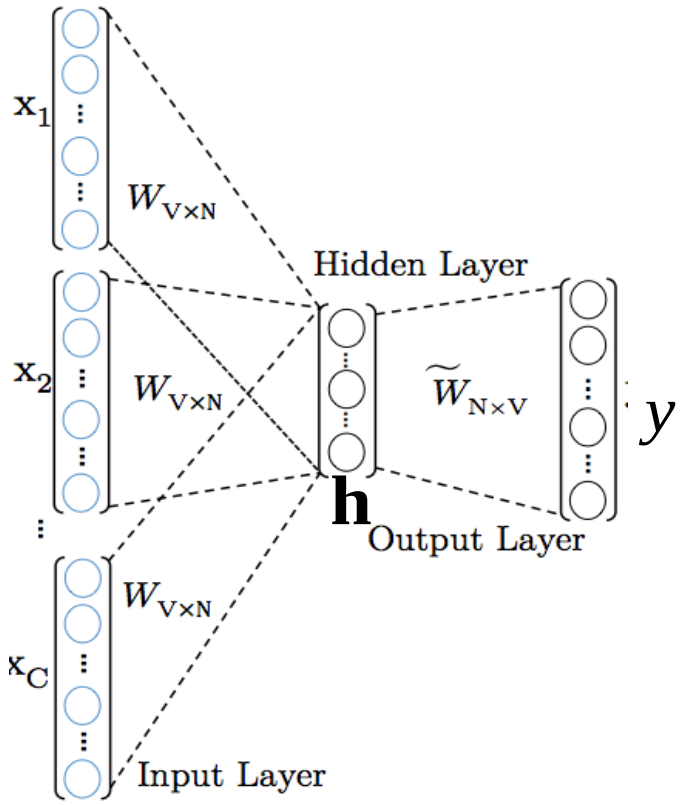


Skip-gram

- **Word2Vec模型的基本流程**

- 我们以CBOW模型为例，介绍其工作原理（Skip-gram模型基本思路类似）
- CBOW本质上是获得上下文的词项→中心词的映射
  - 如何获得这样的映射?
  - 基于深度学习的CBOW算法详细步骤：
    - 使用one-hot（0/1）向量表示词项：x_i
    - 计算隐藏层（深度学习基操）：

$$
\begin{aligned}
\mathbf{h} &= \frac{1}{C}\mathbf{W}^T(\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_C) \\
&= \frac{1}{C}(\mathbf{v}_{w_1} + \mathbf{v}_{w_2} + \cdots + \mathbf{v}_{w_C})^T
\end{aligned}
$$



- **Word2Vec模型的基本流程**

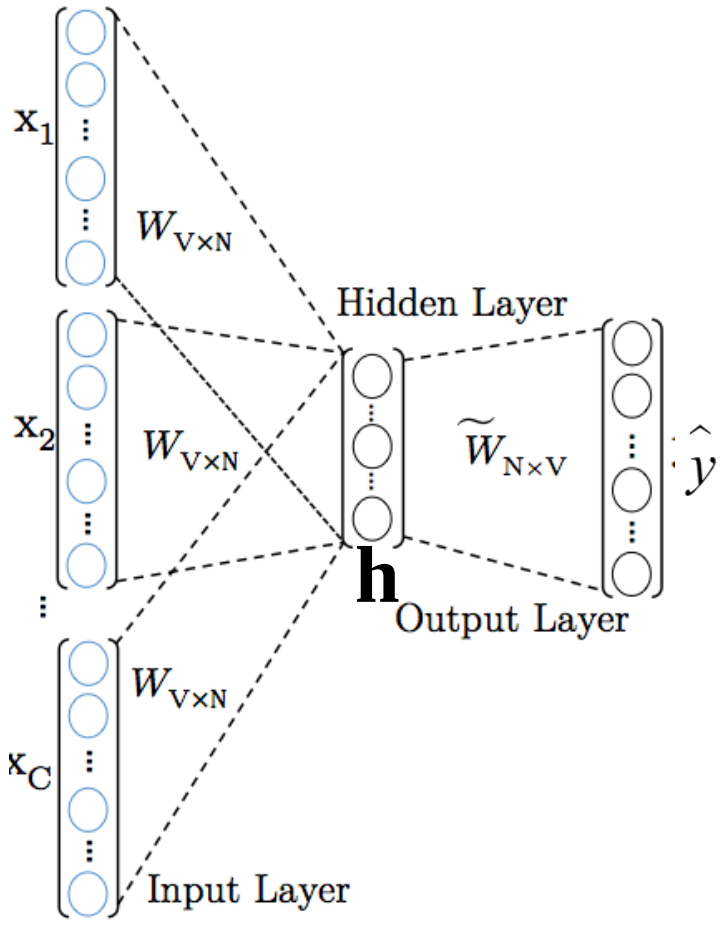
- CBOW算法详细步骤：
  - 计算预测的中心词的概率：

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{h}\widetilde{\mathbf{W}})$$

  - 计算交叉熵损失，反向传播优化模型：

$$L = -\sum y_j \log(\widehat{y_j})$$

  - 基于多个不同滑动窗口进行多次训练
  - 最终得到各个词项的表征：$\mathbf{W}\mathbf{x}_i$

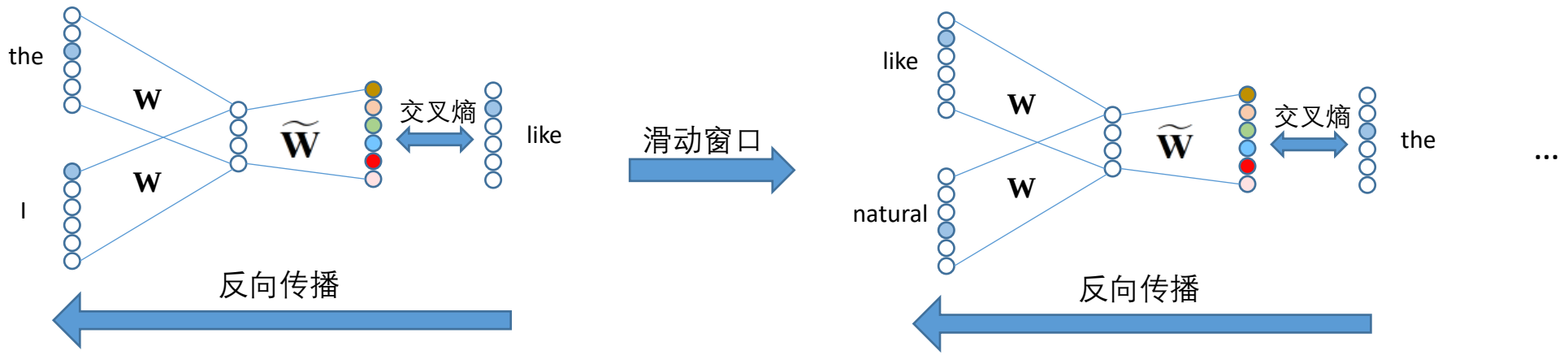


• Word2Vec模型的基本流程

• CBOW算法举例：

I	like	the	natural	language	processing	{like, I}, {like, the}

	I	like	the	natural	language	processing
One-hot vector of “I”	1	0	0	0	0	0
One-hot vector of “like”	0	1	0	0	0	0
One-hot vector of “the”	0	0	1	0	0	0



有关反向传播的详细推导可以参考：https://zhuanlan.zhihu.com/p/652037697

- **CBOW模型与Skip-gram模型的比较**

- Word2vec模型两种设计思路的优劣如何?
- 从**性能**上说
  - CBOW模型仅预测中心词，复杂度约为O(V)，即词表规模
  - 而Skip-gram模型基于中心词预测周边词，复杂度约为O(KV)，即考虑窗口
- 从**效果**上说
  - 在Skip-gram模型中，由于每个词都可以作为中心，都将得到针对性训练
  - 因此，对于生僻词（数据稀疏）的训练而言， Skip-gram模型效果更好

- **Word2Vec的优缺点**

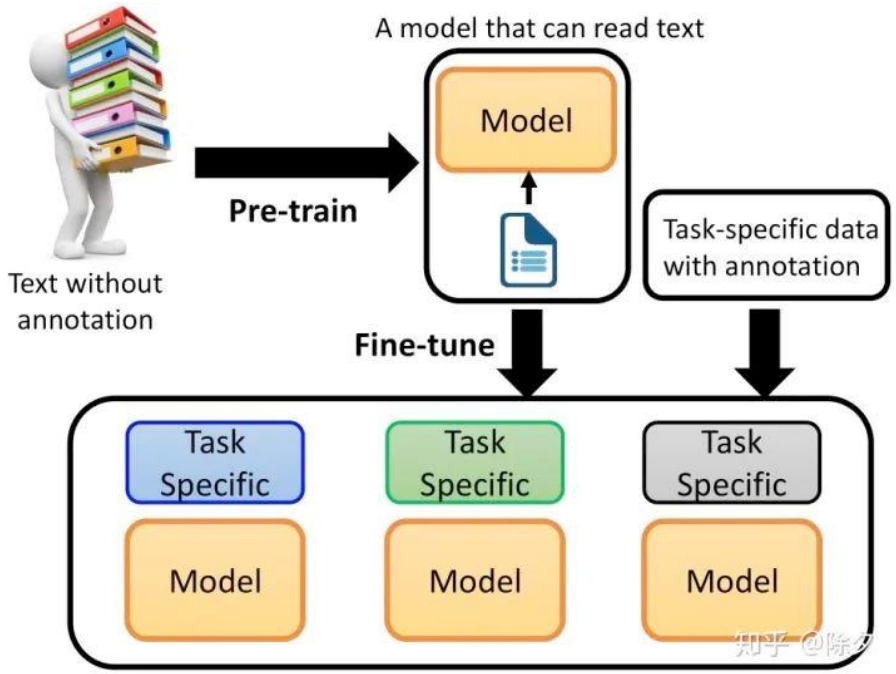
- 优点
  - 有效表征了词项之间的上下文关系
  - 无监督，通用性强，可适用于各种NLP任务
- 缺点
  - 无法解决一词多义的问题，例如play music 和 play football
  - Word2vec 是一种静态的方式，其词项表征一旦训练确定就不会再做更改。因此，虽然通用性强，但是无法针对特定任务进行动态优化

本节目录

- 信息检索与相关度计算
  - 向量空间模型
  - 文档匹配与迭代优化
- 文档表征技术初阶
- **预训练模型初阶**

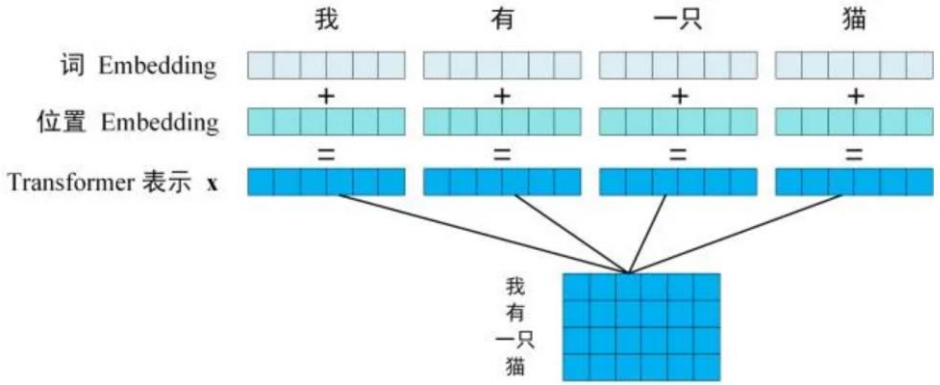
- **从文档表征模型到预训练模型**

- 文档表征技术近年来有了长足进步，但语料的飞速积累，导致大部分人难以充分学习利用
- 预训练技术由此应运而生
  - 什么是预训练模型？通过大规模的数据和模型，学习语料库中的知识，然后将这些知识迁移到某一个具体的任务中，从而获得较好的表现



• 预训练模型基础：Transformer技术

- Transformer赋能大规模神经网络学习
  - 学习词表征的同时学习位置表征
    - 基于位置信息表征，既可以更好地适应较长文本，也可以有效学习词项之间的相对位置关系
  - 核心思想：使用注意力机制捕获序列的全局信息，解决长距离依赖问题



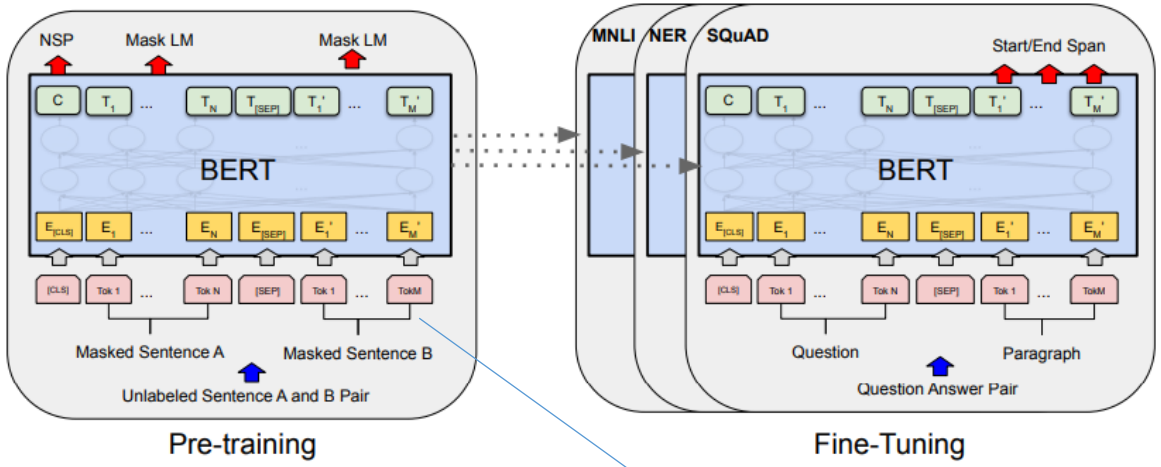
在连德富老师的《深度学习》课程中，有关于Transformer技术的详细原理介绍

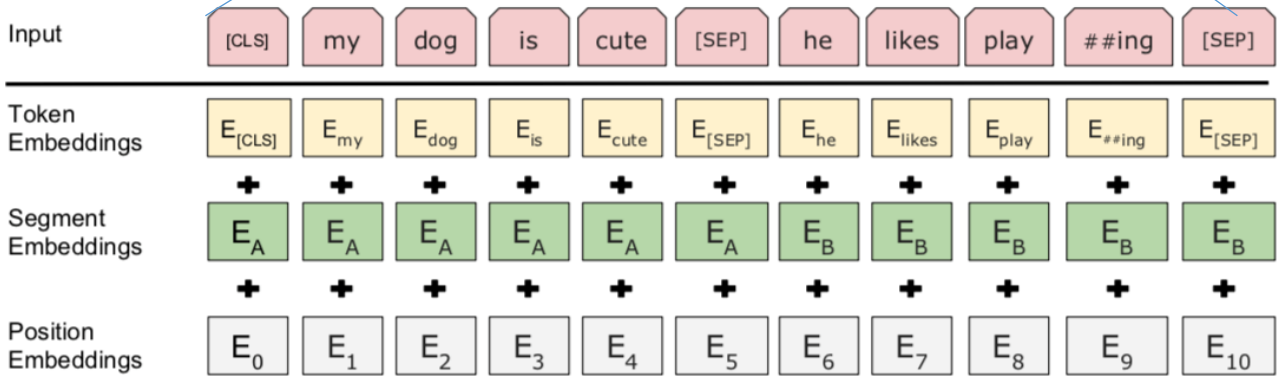
- **预训练模型基础：Transformer技术**

- 如果手痒想实践下Transformer的话……
  - 基础深度学习资料：
    - https://zh.d2l.ai/index.html：动手学深度学习
  - 相关博客：
    - Transformers Explained Visually：https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452
    - Transformer 工作原理：https://towardsdatascience.com/transformers-explained-visually-part-2-how-it-works-step-by-step-b49fa4a64f34
    - 多头注意力机制原理：https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853

• 从Transformer到BERT

• 通过Transformer，实现预训练模型Bert模型架构（双向、多层Transformer）：





正常的词向量，在PyTorch中可以使用`nn.Embedding()`实现

给上句的 token 全 0，下句的 token 全 1，用于区分上下句

待学习的位置参数，用于区分词在不同位置的含义

如何训练BERT模型

- Bert预训练方法（1）与Word2Vec思路类似
  - Masked LM（MLM）：随机的mask或者替换一句话中的任意字词，然后让模型根据上下文预测替换掉的部分，具体的操作如下：
    - 随机把一句话中 15% 的 token（字或词）替换成以下内容：
      - 这些 token 有 80% 的几率被替换成 [MASK]，例如 my dog is hairy→my dog is [MASK]
      - 有 10% 的几率被替换成任意一个其它的 token，例如 my dog is hairy→my dog is apple
      - 有 10% 的几率原封不动，例如 my dog is hairy→my dog is hairy
    - 之后让模型预测和还原被遮盖掉或替换掉的部分，并且计算损失 L_{MLM}
      - 注意，此时损失只与被MASK的部分相关

- **如何训练BERT模型**

- Bert预训练方法（2）从单句内到句与句之间
  - Next sentence prediction（NSP）：判断两个句子是否属于连续的上下文关系，具体操作如下：
    - 从语料库中选择1:1的正负样本（属于上下文和不属于上下文的句子对各占一半）
    - 添加一些特殊的token来分割两句话，如：*[CLS]sentence A[SEP]sentence B [SEP]*
    - 让模型判断两句话是否属于连续的上下文，并且计算NSP任务的损失 L_{NSP}
  - 最终的模型损失： $L = L_{MLM} + L_{NSP}$

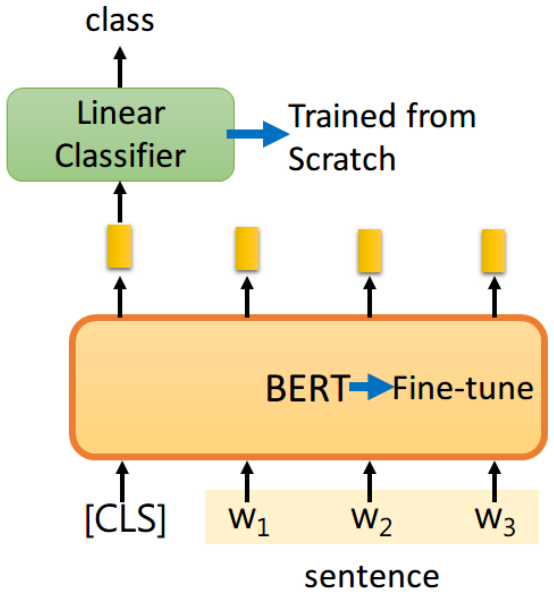
可以根据需要，加上更多的子任务用于训练模型参数！

- **从Pre-train到Fine-tune**

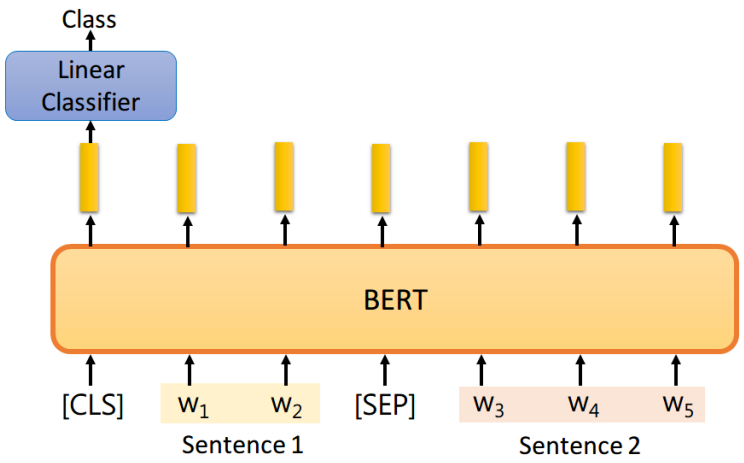
- 利用训练好的BERT模型，我们可以通用地解决许多下游任务
- 然而，与Word2vec类似，BERT同样面临下游任务差异性的挑战
  - 重新训练参数？显然不够经济实惠
  - 站在巨人肩膀上才是坠吼的——基于已有参数的微调（Fine-tune）
- Fine-tune 的核心思路：
  - 通常选择训练好的最后一层——分类层进行替换，冻结先前的部分
  - 利用新的分类层的反向传递优化过程，对已有参数进行微调

- **从Pre-train到Fine-tune**

- Bert Fine Tuning（如何在下游任务中使用Bert）：



文本分类
输入：一段文本
输出：这段话的类别



自然语言推理
输入：一段文本和一个推理
输出：推理是否正确

- **百花齐放的预训练模型**

- 其他常见的预训练模型及其规模

体系架构	模型参数
BERT	12个层，768个隐藏节点，12个heads，110M参数量。在小写英语文本上训练
GPT	12个层，768个隐藏节点，12个heads，110M参数量。OpenAI GPT的英语模型
GPT-2	12个层，768个隐藏节点，12个heads，117M参数量。OpenAI GPT-2的英语模型
Transformer-XL	18个层，1024个隐藏节点，16个heads，257M参数
XLNet	12个层，768个隐藏节点，12个heads，110M参数量。XLNet的英语模型
RoBERTa	12个层，768个隐藏节点，12个heads，125M的参数量。
DistilBERT	6个层，768个隐藏节点，12个heads，66M的参数量。
ALBERT	12个重复的层，embebdding维数128，768个隐藏层，12个heads, 11M参数量。
Bart	12个层，1024个隐藏节点，16个heads，406M的参数量
CTRL	48个层，1280个隐藏节点，16个heads，16亿的参数量。Salesforce的大型CTRL英文模型

- 更多内容：https://huggingface.co/docs/transformers/main/en/index

- **百花齐放的预训练模型**

- 其他常见的预训练模型及其规模

谷歌于2022年4月发布的PaLM模型，参数达5400亿规模，使用780B tokens

训练平台：6144 TPU，总开支达1700万 US刀乐

华为今年三月发布的PanGu-∑，参数达10850亿规模，使用100Btokens

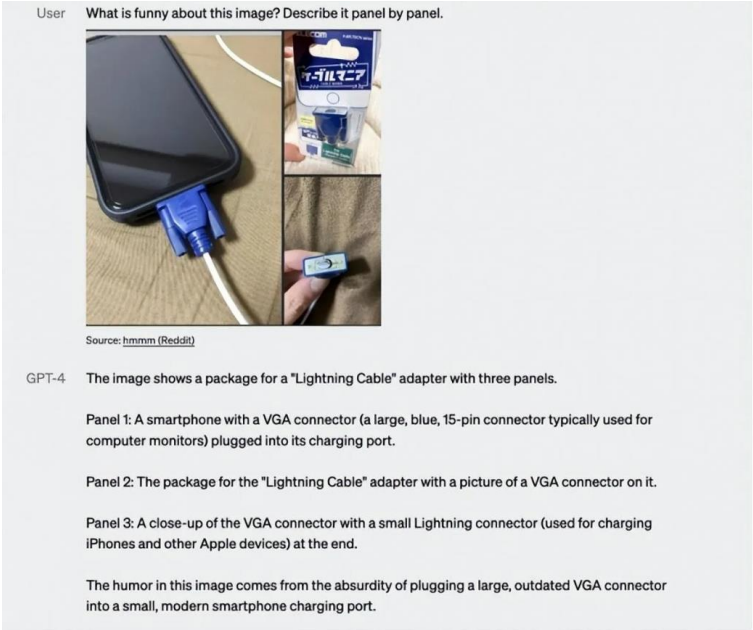
训练平台：512 Ascend910 训练 100 天

DistilBERT	6个层，768个隐藏节点，12个heads，66M的参数量。
ALBERT	12个重复的层，embebdding维数128，768个隐藏层，12个heads, 11M参数量。
Bart	12个层，1024个隐藏节点，16个heads，406M的参数量
CTRL	48个层，1280个隐藏节点，16个heads，16亿的参数量。Salesforce的大型CTRL英文模型

百花齐放的预训练模型

- 预训练模型的多模态时代
  - 将视觉要素引入大模型，使模型同时具备读与看的能力
  - 代表模型： Stable Diffusion，DALLE-3，GPT-4V

an espresso machine that makes coffee from human souls, artstation



User What is funny about this image? Describe it panel by panel.

Source: hmmm (Reddit)

GPT-4 The image shows a package for a "Lightning Cable" adapter with three panels.

Panel 1: A smartphone with a VGA connector (a large, blue, 15-pin connector typically used for computer monitors) plugged into its charging port.

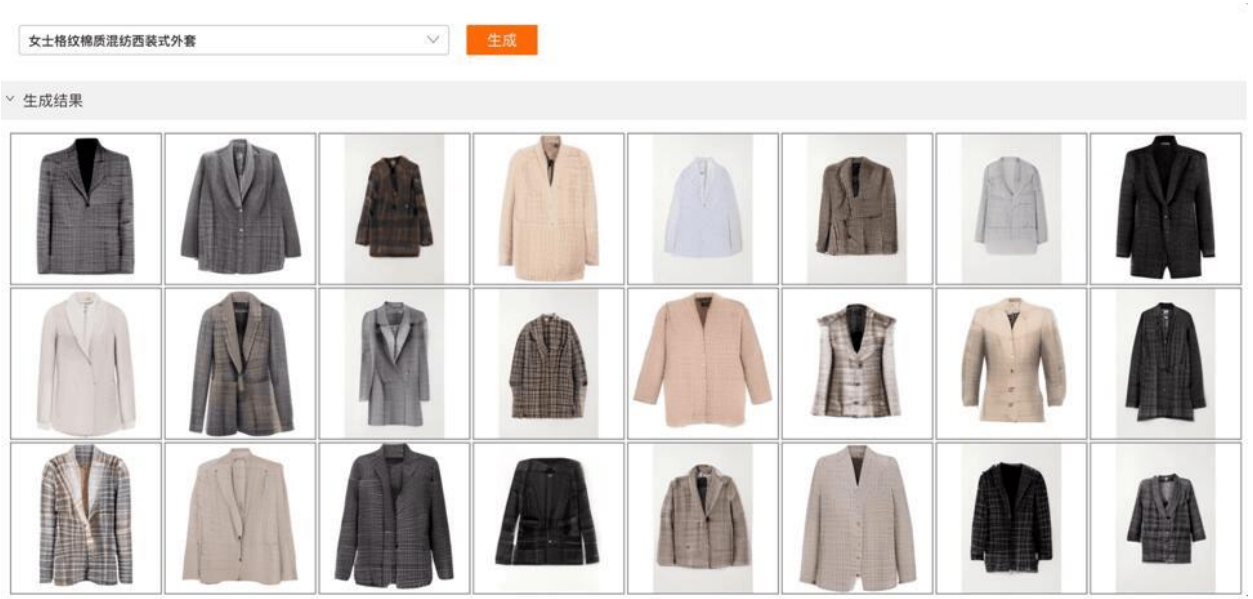
Panel 2: The package for the "Lightning Cable" adapter with a picture of a VGA connector on it.

Panel 3: A close-up of the VGA connector with a small Lightning connector (used for charging iPhones and other Apple devices) at the end.

The humor in this image comes from the absurdity of plugging a large, outdated VGA connector into a small, modern smartphone charging port.

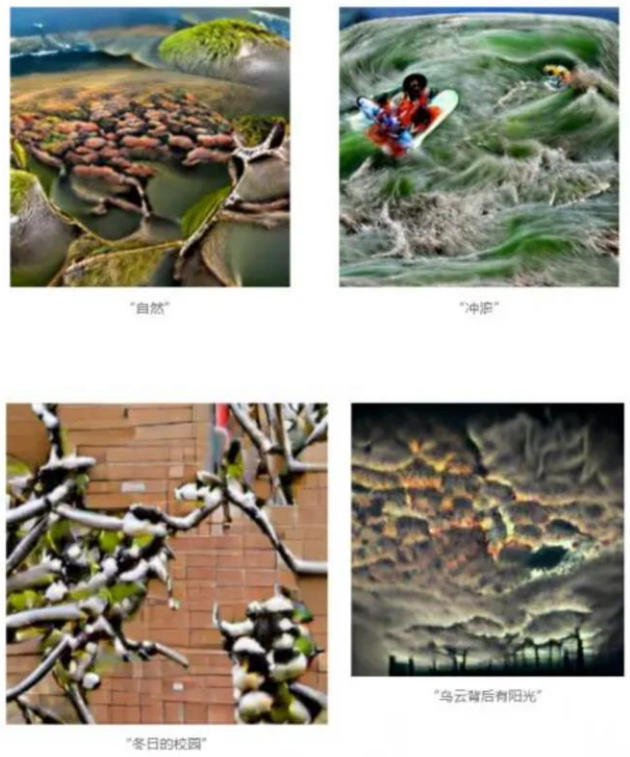
- **百花齐放的预训练模型**

- 国产大模型风起云涌



阿里达摩院 M6 模型

10万亿参数，只需要512张V100即可训练



人大文继荣教授团队 文澜 模型

在抽象概念表达上有其独到之处

https://www.bilibili.com/video/BV1Z54y1G7du

• 预训练模型之殇：能耗问题

• 模型能耗对比

Model	Hardware	Power (W)	Hours	kWh·PUE	CO_2e	Cloud compute cost
$T2T_{base}$	P100x8	1415.78	12	27	26	$41–$140
$T2T_{big}$	P100x8	1515.43	84	201	192	$289–$981
ELMo	P100x3	517.66	336	275	262	$433–$1472
$BERT_{base}$	V100x64	12,041.51	79	1507	1438	$3751–$12,571
$BERT_{base}$	TPUv2x16	—	96	—	—	$2074–$6912
NAS	P100x8	1515.43	274,120	656,347	626,155	$942,973–$3,201,722
NAS	TPUv2x1	—	32,623	—	—	$44,055–$146,848
GPT-2	TPUv3x32	—	168	—	—	$12,902–$43,008

Table 3: Estimated cost of training a model in terms of CO_2 emissions (lbs) and cloud compute cost (USD).[7] Power and carbon footprint are omitted for TPUs due to lack of public information on power draw for this hardware.

人工智能不环保？训练一个神经网络的排炭量竟然比5辆车还多？

发布于2019-11-12 10:20:47　　阅读 314

导读：作为目前最前沿的计研究领域之一，人工智能也受到了环保方面的质疑。

作者：曹培信

来源：大数据文摘（ID：BigDataDigest）

ACL2019新论文，痛批“不计成本追逐丁点提升”的深度学习研究方法

“有些人做神经网络训练，完全不计代价，太不负责任了！”

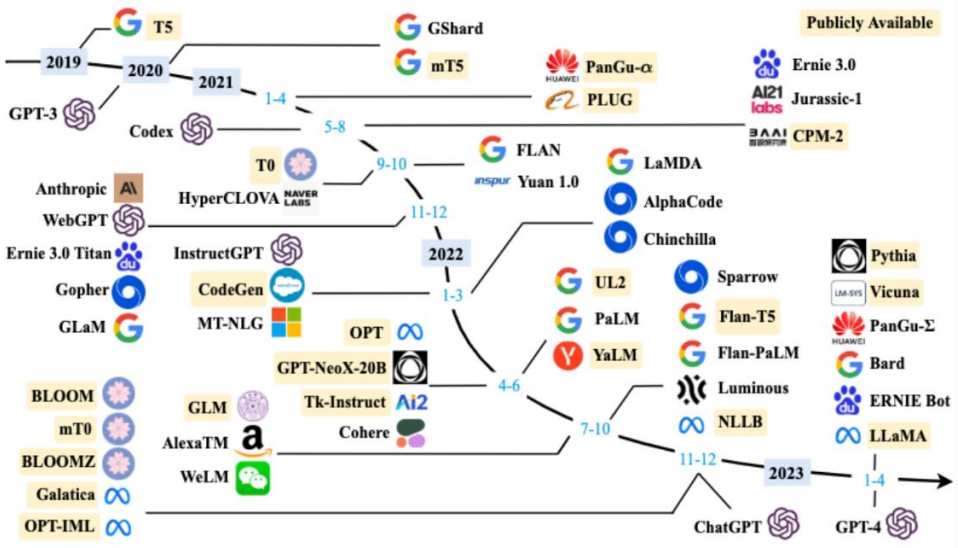
- **预训练模型的优点**

- 预训练模型与word2vec的区别与联系
  - 同样是在语料库中进行训练，但是把任务改成了难度更大的形式，比如：完形填空，句子顺序预测
  - 预训练模型使用上游预训练和下游微调的范式，使模型得以运用于更广泛的场景
- 预训练模型的优点
  - 充分利用了大量的无标签数据
  - 将开放世界的知识应用于下游任务中，从而大幅提高模型效果
  - 针对具体的任务，可以直接使用已经开源预训练好的模型，而且能够得到非常好的结果

预训练模型的缺点

- 预训练模型的缺点：
  - 在一些不那么通用的领域，如医疗，金融等，表现有待提升
  - 预训练模型可能会带来一些安全问题
  - 大规模预训练模型的功耗高，硬件要求高，使得训练一个模型逐渐成为了门槛

讨论：预训练模型与通用人工智能（AGI）



图片来源: https://github.com/RUCAIBox/LLMSurvey

- **预训练模型相关工具**

- 如果实在是想不开想挑战预训练模型……
  - https://huggingface.co/models：3000+NLP预训练模型库
  - https://modelzoo.co/：深度学习各个领域预训练模型库（CV，NLP，生成模型，强化学习，图等）
  - https://huggingface.co/docs/transformers/index：预训练模型工具包
  - https://huggingface.co/datasets：预训练模型数据库
  - https://huggingface.co/course：transformer API 使用教程
  - https://fairseq.readthedocs.io/：NLP预训练模型工具包

下节预告

- **下节预告**

- 本节课程介绍的排序问题，主要围绕用户意图（Query）与文档的匹配度排序
- 如何充分利用用户的点击/排序反馈，对网页进行排序?
  - 排序学习（**Learning to Rank**）类型与常用技术介绍
- 如何在确保主题契合的同时，判断高质量的网页文档?
  - **PageRank**技术及其衍生模型

本章小结

网页排序（上）

- 信息检索与相关度计算
  - 向量空间模型
  - 文档匹配与迭代优化
- 文档表征技术
- 预训练模型初阶