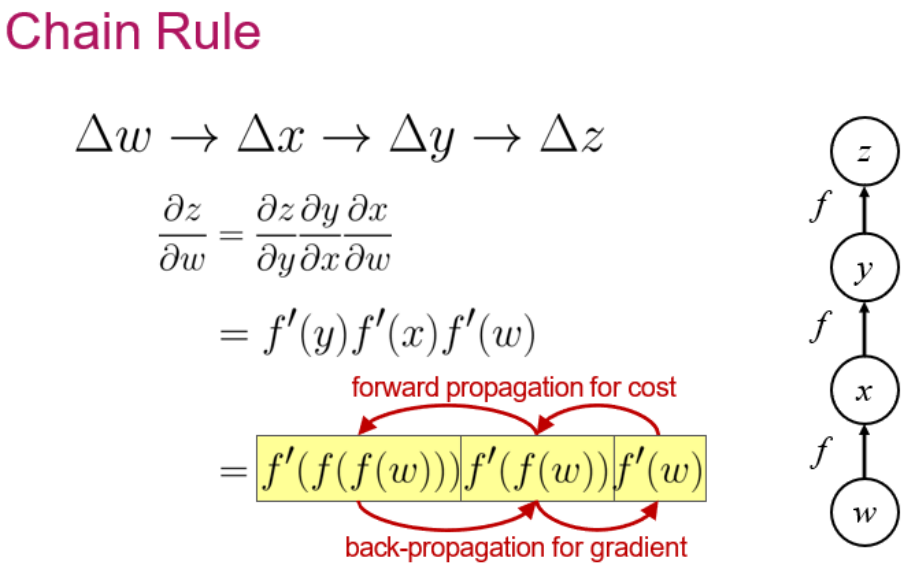
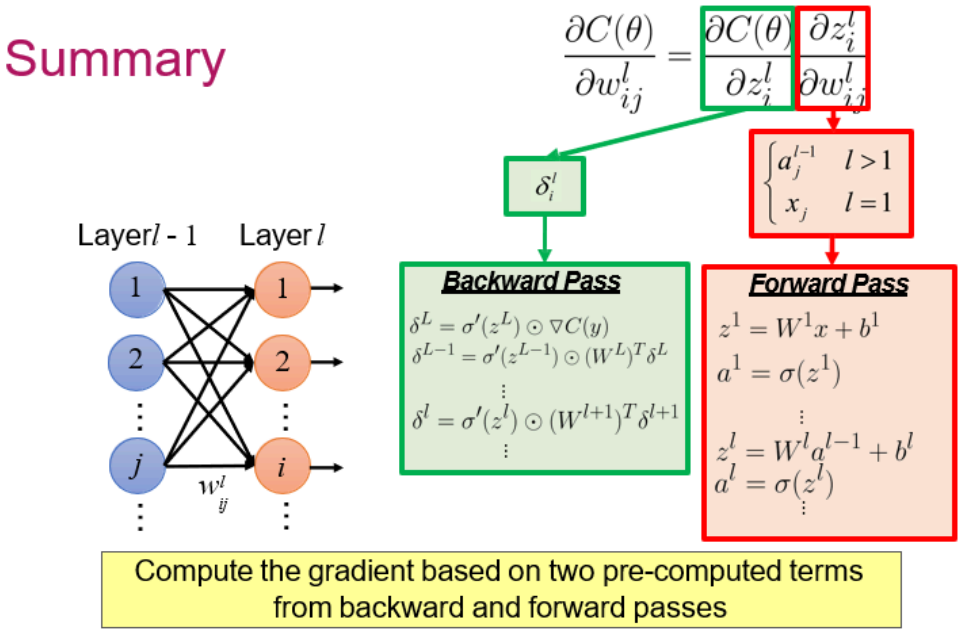
3.Backpropagation & Word Representations

Backpropagation

- 定义：一种梯度的计算机制，用于梯度计算而不是参数更新)
- 链式法则：用于计算复合函数的导数，是反向传播的核心工具。
- 前向传播：从输入层到输出层的计算过程，生成模型的预测输出和计算损失函数的值。
- 反向传播：从输出层到输入层的计算过程，利用链式法则计算损失函数对每个参数的梯度，更新参数。





$C(\theta)$是损失函数，$\odot$：逐元素乘法，$\nabla C(y)$是损失函数C对网络输出的梯度，这里的y是a^L

Word Representations

Meaning Reresentation

Knowledge-Based Representation

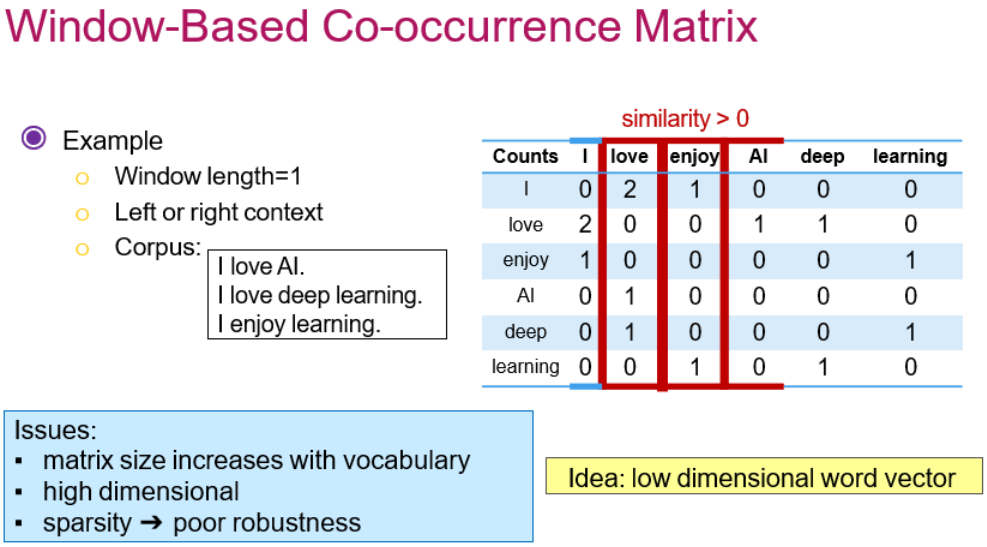
- 最简单的例子就是WordNet,一个大型的英语词汇数据库,通过在同义词集合之间建立关系。
  WordNet应用：
  语义分析：通过分析单词之间的语义关系，WordNet 可以帮助理解文本的含义。
  词义消歧：通过上下文中的语义关系，WordNet 可以帮助确定单词的具体含义。
  信息检索：通过语义关系，WordNet 可以帮助找到与查询词相关的其他词汇。

  超义词（Hypernyms）：表示更一般的概念。例如，“vehicle”是“car”的超义词，因为“car”是一种“vehicle”。
  下义词（Hyponyms）：表示更具体的概念。例如，“car”是“vehicle”的下义词。

Corpus-Based Representation

- 独热编码(one-hot representation)
  独热编码是一种将单词表示为向量的方法。每个单词对应一个唯一的向量，其中只有一个位置是 1，其余位置都是 0。向量长度等于词汇表的大小（例如 10,000 个单词），而 "car" 在词汇表中的位置（假设是第 6 个）被置为 1，其余为 0。

  这种架构导致独热编码是**正交**的，无法计算相似度。
- **改进：Neighbor-based representation**
  基于邻居的表示是一种利用单词在文本中的共现关系来表示单词语义的方法。这种方法通过构建**共现矩阵**（**Co-occurrence Matrix**）来捕捉单词之间的语义关系。共现矩阵可以基于不同的上下文定义来构建，常见的有两种方法：基于全文档（full document）和基于窗口（windows）。
- Window-Based Co-occurrence Matrix
  定义：这种共现矩阵考虑了单词在固定大小的上下文窗口中的出现情况。上下文窗口是指某个单词前后一定数量的单词范围。例如，窗口大小为2意味着考虑当前单词前后各两个单词。
  应用：通过分析单词在上下文窗口中的共现情况，可以捕捉单词的语法（如词性）和语义信息。例如，对于文档 "I like deep learning. I like NLP. I enjoy flying."，可以构建共现矩阵



- Word-Document Co-occurrence Matrix
  定义：这种共现矩阵考虑了单词在全文档中的出现情况，每个单元格表示一个单词在某个文档中出现的次数。这种矩阵可以用于主题建模，如潜在语义分析（Latent Semantic Analysis, LSA）。
  应用：通过分析单词在不同文档中的共现情况，可以提取出文档的主题信息。例如，所有体育术语在文档中会有相似的条目，从而帮助进行主题建模。

Word Embadding

- 定义：Vectors representing words, such that semantically similar words are represented by similar vectors.

Word2Vec

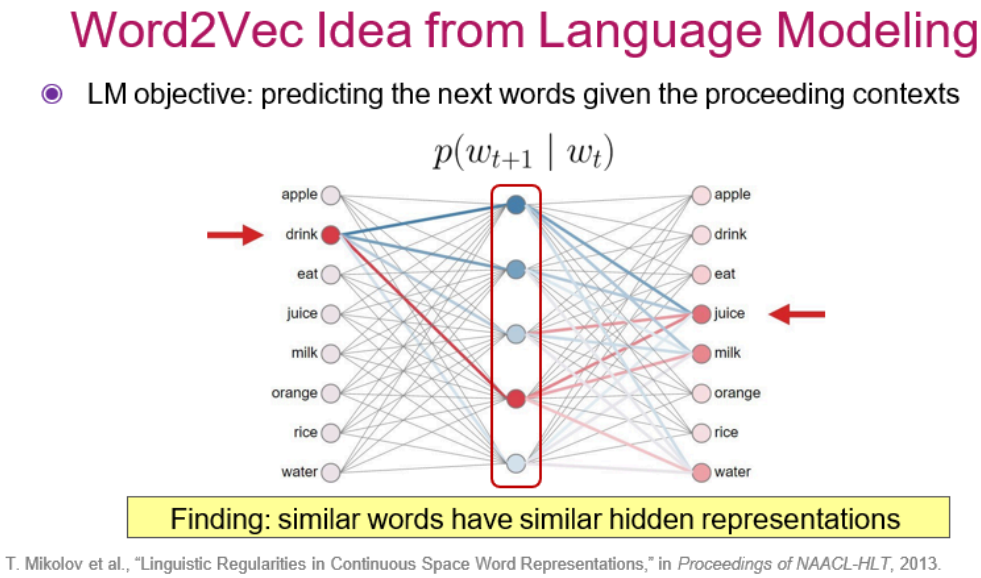
Word2Vec 是一种从大量文本数据中以无监督方式学习语义知识的模型，被广泛应用于自然语言处理（NLP）中。它通过将单词转换为低维密集向量（词向量），捕获单词的语义和语法信息。Word2Vec 的核心思想来源于语言模型，特别是统计语言模型。

Word2Vec Variants:

- **Skip-gram**: Skip-Gram 模型是 Word2Vec 的一种实现方式，其目标是根据中心词预测上下文词。具体来说，给定一个中心词，模型会尝试预测在上下文窗口内的其他词。例如，对于句子 "I love to eat food"，如果中心词是 "love"，那么上下文词可能是 "I"、"to"、"eat" 和 "food"
- **CBOW (continuous bag-of-words)**: 是 Word2Vec 的一种实现方式，其目标是根据上下文单词预测目标单词。具体来说，给定一个上下文窗口内的单词，CBOW 模型会尝试预测中心

词。例如，对于句子 "I love to eat food"，如果上下文窗口内的单词是 "I"、"to"、"eat" 和 "food"，那么模型会尝试预测中心词 "love"

- **LM (Language modeling)**: 基于给出的处理过的内容，预测下一个词



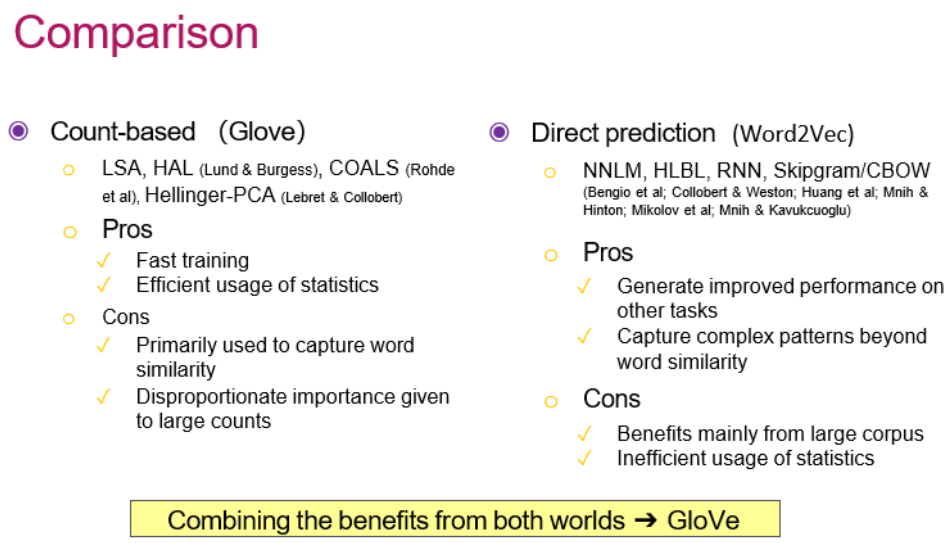
T. Mikolov et al., "Linguistic Regularities in Continuous Space Word Representations," in *Proceedings of NAACL-HLT*, 2013.

GloVe

Global Vectors for Word Representation 是一种流行的词嵌入方法，由斯坦福大学的 Jeffrey Pennington、Richard Socher 和 Christopher D. Manning 在 2014 年提出。它通过结合全局统计信息和局部上下文信息来生成词向量，从而更好地捕捉单词的语义关系。

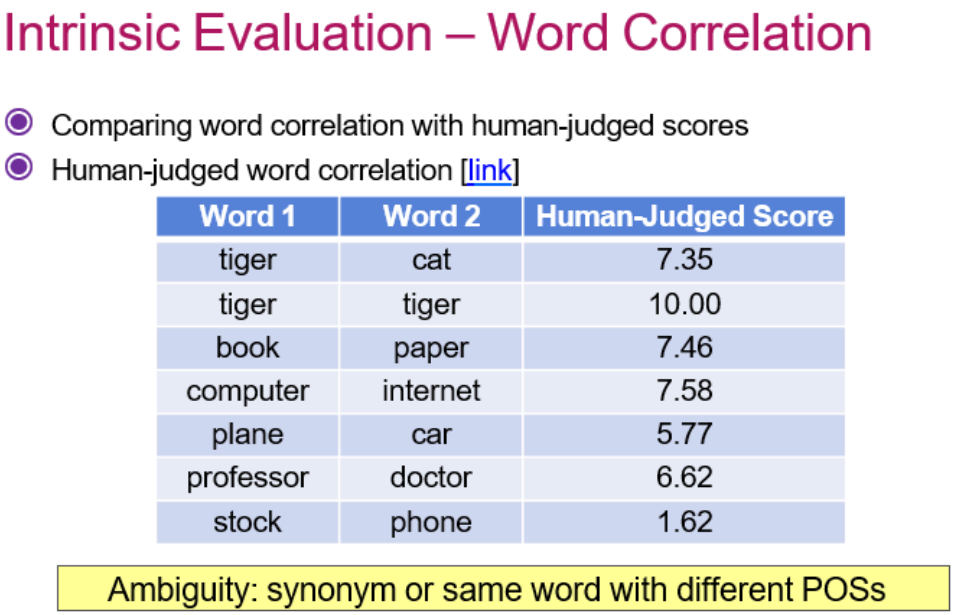
- 核心思想：局部上下文 ➕ 全局统计信息
  1.构建共现矩阵：统计单词在语料库中共现的频率。
  2.优化目标函数：通过最小化预测共现概率与实际共现概率之间的差异来优化词向量。
  3.引入权重：为不同的词对引入权重，以控制它们对训练过程的影响。
- 2. GloVe 的优势
  全局统计信息：GloVe 利用全局统计信息，能够更好地捕捉单词之间的语义关系。
  计算效率高：GloVe 采用矩阵分解技术，训练效率较高，适合大规模语料库。
  灵活的上下文窗口：可以调整上下文窗口的大小，以捕捉局部和全局的上下文信息。

GloVe v.s. Word2Vec



Intrinsic Evaluation – Word Correlation

- 词相关性（Word Correlation）是词嵌入内在评估的一种方法，用于衡量词向量之间的相似性。通过计算词向量之间的余弦相似度，可以评估模型生成的词向量与人类判断的相似性之间的相关性。这种方法可以帮助我们理解词嵌入模型在捕捉词义方面的表现。



NLP Outline

Language Modeling

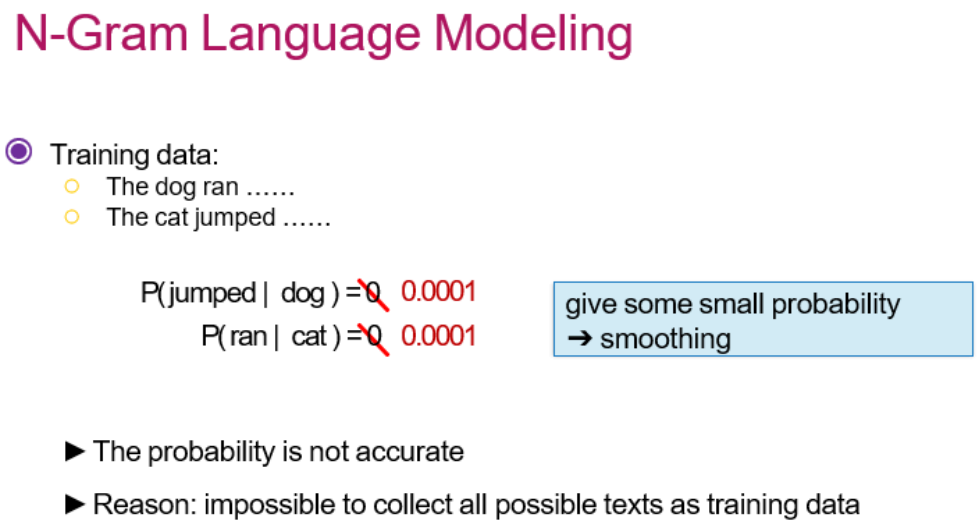
- 语言建模是自然语言处理（NLP）中的一个核心任务，目标是预测文本序列中的下一个词或字符。语言模型通过学习大量文本数据中的模式和结构，能够生成自然语言文本，并评估给定文本序列的概率。

N-Gram Language Modeling

- N-Gram 语言模型是一种简单的统计模型，用于预测文本中下一个词的概率。它通过观察前n−1个词来预测下一个词。这里的 n是一个数字，表示模型考虑的词的数量。比如假设我们有一个简单的句子："I like deep learning."：

```
Unigram 模型（n=1）
•目标：预测每个词的概率，不考虑上下文。
•例子：
•计算每个词出现的次数：
•I: 1, like: 1, deep: 1, learning: 1
•每个词的概率为：
•P(I) = 1/4 P(like) = 1/4 P(deep) = 1/4 P(learning) = 1/4
```

- 在 N-Gram 语言建模中，一个常见的问题是预测的**概率可能不准确**。这主要是因为不可能收集到所有可能的文本作为训练数据。
  - 原因 1：数据稀疏性（Data Sparsity）
    - 解释：在实际应用中，不可能收集到所有可能的文本作为训练数据。这意味着某些 n-gram 的频率可能为零，导致模型无法准确估计这些 n-gram 的概率。
    - 例子：假设我们有一个包含 1000 个单词的语料库，但某些词或词组在语料库中从未出现过。当我们尝试计算这些词或词组的概率时，模型会返回零概率，这显然是不合理的。
  - 原因 2：长距离依赖（Long-Distance Dependencies）
    - 解释：N-Gram 模型只能捕捉局部上下文信息，无法捕捉长距离的依赖关系。这可能导致模型在处理复杂句子时表现不佳。
    - 例子：对于句子 "I like deep learning because it is very interesting."，N-Gram 模型可能无法捕捉 "because" 和 "interesting" 之间的关系，因为它们之间的距离超过了 n-gram 的窗口大小。
  - 原因 3：词汇表大小（Vocabulary Size）
    - 解释：随着词汇表的增大，n-gram 的数量呈指数级增长，这使得数据稀疏性问题更加严重。
    - 例子：假设词汇表中有 10,000 个单词，对于 Trigram 模型，可能的三元组数量为 10,0003=101210,0003=1012。即使语料库非常大，也不可能覆盖所有可能的三元组。



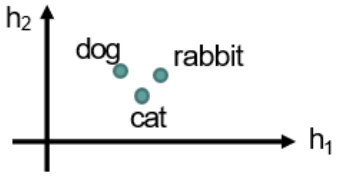
Feed-Forward NN-based LM

- 前馈神经网络语言模型是一种基于神经网络的语言模型，它通过学习输入序列的概率分布来预测下一个词。这种模型通常使用嵌入层、隐藏层和输出层来处理输入序列，并通过最小化交叉熵损失来优化模型参数。

- 结构：
  - 嵌入层（Embedding Layer）：将输入词映射为低维向量。
  - 隐藏层（Hidden Layer）：通常是一个或多个全连接层，用于提取特征。
  - 输出层（Output Layer）：使用 Softmax 函数将隐藏层的输出转换为概率分布。
  - **训练目标**：训练目标是最小化交叉熵损失，即最小化模型预测的概率分布与真实分布之间的差异
- 最小化交叉熵（Minimizing Cross Entropy）
  - 交叉熵是衡量两个概率分布之间差异的指标。在语言建模中，交叉熵用于衡量模型预测的概率分布与真实分布之间的差异。交叉熵越小，模型的预测越接近真实分布。

重要特点：相关词的输入层（或隐藏层）接近，平滑自动完成
在基于前馈神经网络的语言模型中，一个重要的特性是相关词的输入层（或隐藏层）表示通常是接近的。在神经网络语言模型中，词嵌入（word embeddings）是将单词映射到低维向量空间的技术。这些嵌入向量通过训练学习到单词的语义和语法特征。由于嵌入向量是通过学习文本数据中的模式得到的，因此语义或语法相关的单词在嵌入空间中通常是接近的。

The input layer (or hidden layer) of the related words are close



- If P(jump | cat) is large, P(jump | dog) increases accordingly (even there is not "… dog jumps …" in the data)

Smoothing isautomatically done

Issue: fixed context window for conditioning

Recurrent Neural Network Language Model (RNNLM)

- 关键点：condition the neural network on all previous words and tie the weights at each time step（"记忆"，共享权重）

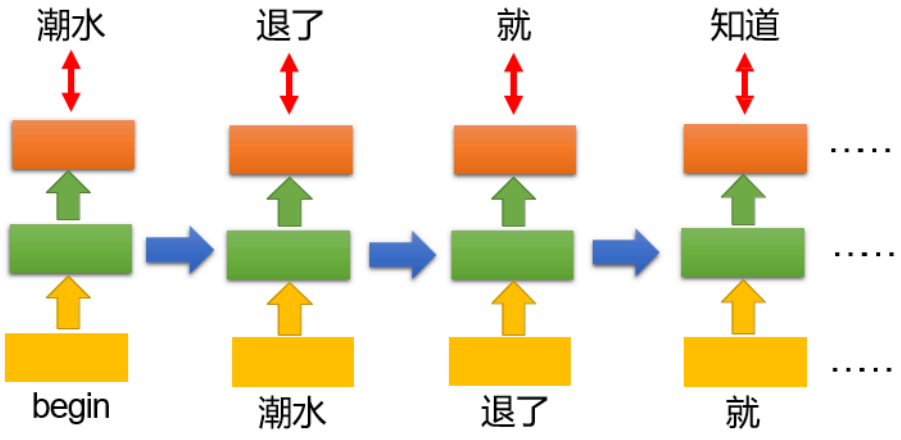
- 也就是RNN 在处理序列数据（例如一个句子）时，会按时间步（time step）逐个处理输入（例如逐个处理单词）。在每个时间步，RNN 都会结合以下两部分信息来更新其状态并生成输出
  当前时间步的输入：也就是"现在正在读的单词"
  之前时间步的隐藏状态：隐藏状态（hidden state）包含了之前所有时间步的信息

- Assumption: temporal information matters：适合时间信息重要的场景

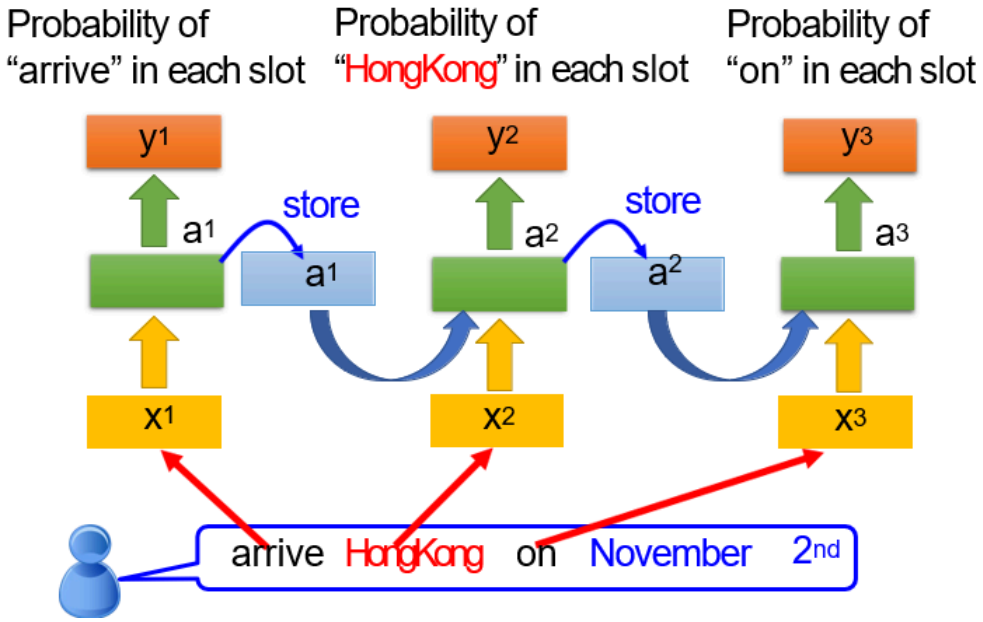
RNN-based LMM
- Training

Collect data:

潮水 退了 就 知道 谁...
楼价高..
.....



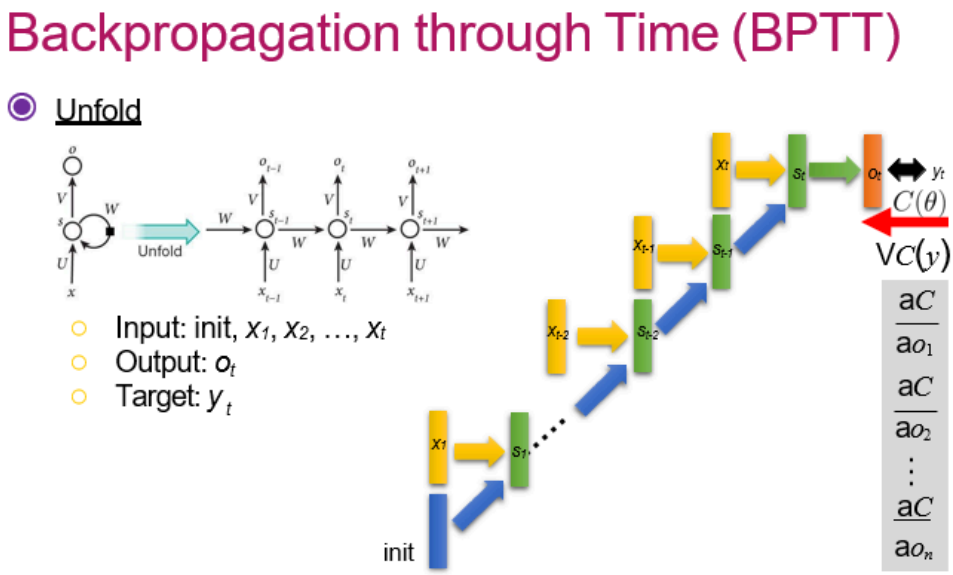
Recurrent Neural Network



- 在循环神经网络（RNN）处理序列数据时，比如句子“arrive HongKong on”，它会逐个读取每个单词，依靠一个共享的神经网络结构来处理这些输入。每次读取一个新单词时，网络会利用之前处理过的所有单词信息（存储在隐藏状态a中）结合当前单词，动态更新隐藏状态，然后根据这个更新后的状态生成对应单词的预测输出。这种机制允许网络记住上下文并在整个序列中保持一致的行为，尤其适用于需要理解时间或顺序依赖的任务。

Backpropagation Through Time, BPTT

- 时间反向传播（BPTT）是反向传播在循环神经网络（RNN）中的应用。RNN 用于处理序列数据，如时间序列或文本。BPTT 通过展开时间序列，将 RNN 的时间步展开为多个独立的层，然后应用标准的反向传播算法。
- **展开时间序列**：将 RNN 的时间步展开为多个独立的层。每个时间步的隐藏状态 h_t 依赖于前一个时间步的隐藏状态 h_{t-1}和当前时间步的输入 x_t。
- **向前传播**：从初始时间步开始，逐步计算每个时间步的隐藏状态和输出。计算最终的损失函数值。
- **反向传播**：从最后一个时间步开始，计算损失函数对每个时间步输出的梯度。使用链式法则将这些梯度传播到前一个时间步。计算每个时间步的参数梯度。
- **参数更新**：使用梯度下降算法更新 RNN 的参数。



Training Issue

- 梯度是雅可比矩阵(Jacobian matrices)的乘积：反向传播中的梯度是与前向计算每个步骤相关的雅可比矩阵的乘积。这意味着梯度依赖于时间步长序列中的变换。
- 矩阵在反向传播中的重复乘法：在***反向传播过程***中，相同的权重矩阵（W^{t+1}）在每个时间步长上反复相乘。这在公式 $\delta^l = \sigma'(z^l) \circ (W^{l+1})^T \delta^{l+1}$中有所体现，其中第 l 层的梯度δ^l 受到下一层权重矩阵转置的影响。
- 导致**梯度爆炸Exploding gradient 或者梯度消失 Vanishing Gradient**：
  如果矩阵的特征值小于1，乘积会呈指数级缩小（梯度消失）。如果特征值大于1，乘积会呈指数级增长（梯度爆炸）。这种不稳定性使得网络难以学习长期依赖关系。
- ***Vanishing Gradient----in RNN--->Slow training***
- ***Exploding gradient----in RNN---->value of NaN or Inf in training***
- *Tips:Saturated activation functions can be one of the causes of the vanishing gradient problem, because when these functions saturate (output values approach 0 or 1), the gradients become very small.*

Solution for Exploding Gradient

Gradient Clipping

（赞美梯度裁剪之神！解决了**梯度爆炸**）

- 在训练深度神经网络时，尤其是在处理长序列数据（如循环神经网络，RNN）时，梯度可能会在反向传播过程中变得非常大。梯度裁剪是一种有效的解决方案，用于防止梯度值过大。

具体来说，梯度裁剪通过设置一个阈值，当梯度的范数（magnitude）超过这个阈值时，将梯度按比例缩放，使其范数不超过该阈值。

- **工作原理**：
  - 设置阈值：选择一个合适的梯度范数阈值，例如 1.0 或 5.0。
  - 计算梯度范数：在每次反向传播时，计算当前梯度的范数。
  - 裁剪梯度：如果梯度的范数超过阈值，则按比例缩放梯度，使其范数等于阈值。

Gating

（赞美门控机制之神！解决了**梯度消失**）

- 消失梯度问题是深度学习中的一个常见问题，特别是在训练深度神经网络时。当网络层数较深时，梯度在反向传播过程中可能会变得非常小，导致模型更新的步长过小，从而使训练过程变得非常缓慢，甚至停滞。
- **工作原理**：
  - 长短期记忆网络（LSTM）：LSTM 是一种特殊的 RNN 架构，通过引入输入门、遗忘门和输出门来控制信息的流动。这些门可以决定哪些信息需要保留、哪些信息需要遗忘，从而有效地解决消失梯度问题。
  - 门控循环单元（GRU）：GRU 是 LSTM 的简化版本，通过引入更新门和重置门来控制信息的流动。GRU 在处理长序列数据时表现出色，能够有效地捕捉长距离依赖。
  - *Tips:Using ReLU activation function also can solve the problem in particular situation.*

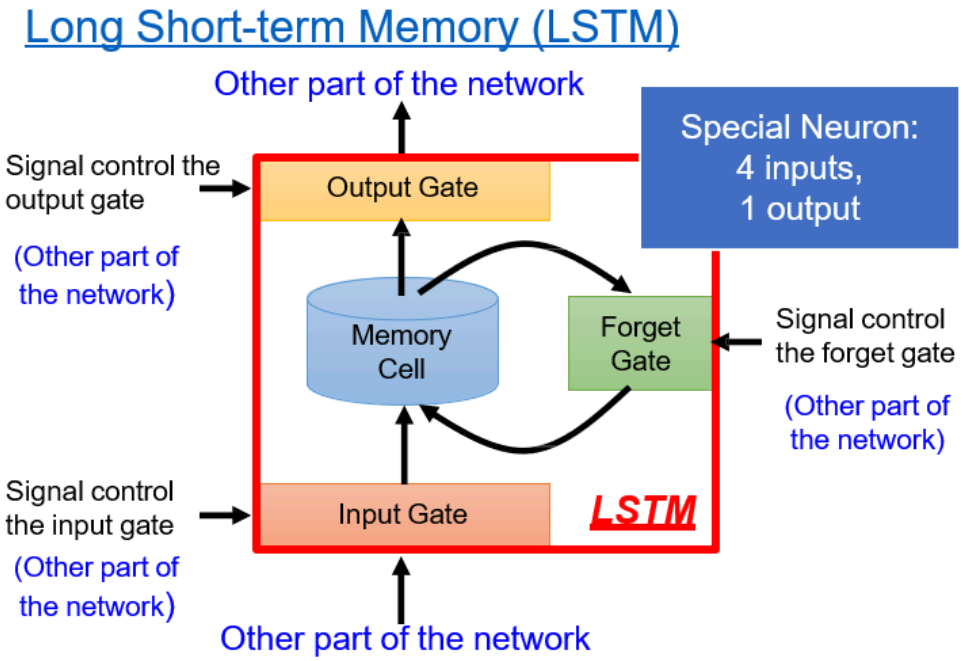
以下是详细解释：
Long Short-term Memory (LSTM)

- 结构：
  输入门（Input Gate）控制新信息的写入。决定输入值是否应该存储在记忆单元中。
  遗忘门（Forget Gate）控制旧信息的遗忘。决定记忆单元中当前的值是否应该保留或遗忘。
  输出门（Output Gate）控制信息的输出。决定记忆单元中的值是否应该被输出。
  细胞状态（Cell State）存储长期信息。

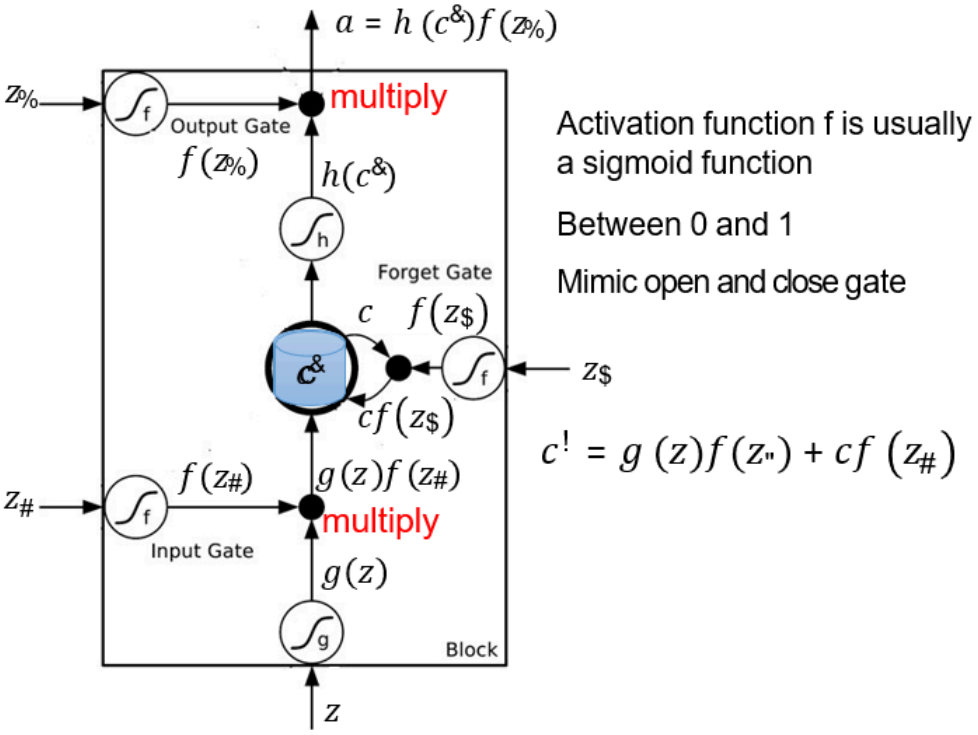
  特殊神经元：4个输入，1个输出
- 4个输入：
  输入值：可能被存储在记忆单元中的值。
  输入门信号：控制输入值是否应该写入记忆单元。
  输出门信号：控制存储在记忆单元中的值是否应该被读出。
  遗忘门信号：控制存储在记忆单元中的值是否应该被遗忘。

- 1个输出：
  输出值：LSTM 神经元的输出，这是由门和记忆单元控制的操作的结果。

  图解如下：



  另外：



  在长短期记忆网络（LSTM）中，**激活函数 f 通常是一个 Sigmoid 函数**。Sigmoid 函数的输出值在 0 和 1 之间，这使得它非常适合模拟门的开启和关闭。在 LSTM 中，Sigmoid 函数通常用于控制门的开启和关闭。具体来说，Sigmoid 函数的输出值可以被解释为“开启”或“关闭”

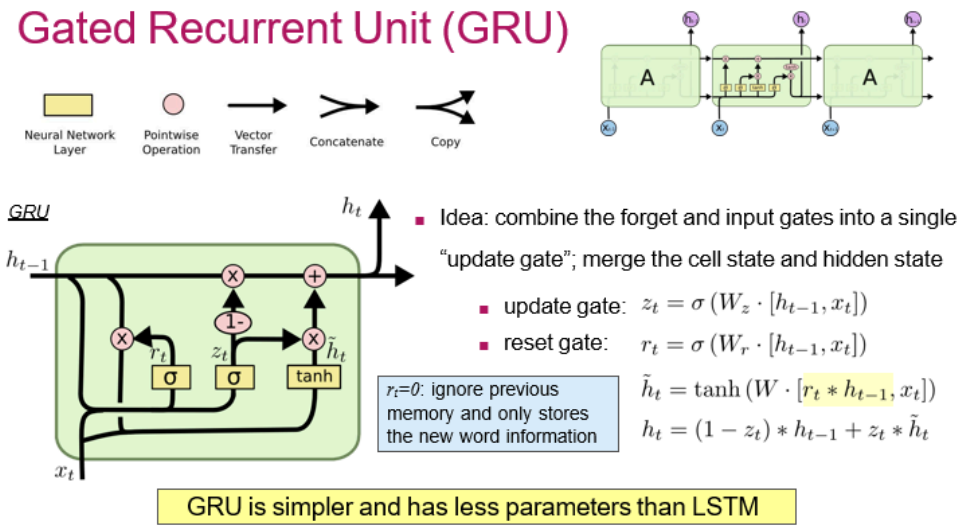
的概率。
示例： 假设我们有一个输入值 x，通过 Sigmoid 函数计算得到的输出值 σ(x) 可以被解释为门的开启概率。如果 σ(x) 接近 1，表示门几乎完全开启；如果 σ(x) 接近 0，表示门几乎完全关闭。

Note: sigmoid activation functions play in LSTM--->Regulate the cell state update

进一步的还有**Multiple-layer LSTM**.堆叠多个 LSTM 层来增强模型的学习能力。单层 LSTM 可以捕捉序列中的短期依赖关系，但多层 LSTM 可以捕捉更复杂的长期依赖关系。通过增加层数，模型可以学习到更丰富的特征表示，从而提高预测的准确性。（别想了学不懂的，靠Pytorch吧就）

Gated Recurrent Unit (GRU)

- 门控循环单元（GRU）是一种简化版的长短期记忆网络（LSTM），它通过引入两个门（更新门和重置门）来控制信息的流动。GRU 的设计比 LSTM 更简单，参数更少，但仍然能够有效地处理长序列数据中的长期依赖问题。
- 结构：
  更新门（Update Gate）：决定新信息的写入程度和旧信息的保留程度。
  重置门（Reset Gate）：决定旧信息的遗忘程度。
- 还有两个状态?
  候选隐藏状态（Candidate Hidden State）：计算新的隐藏状态的候选值。
  最终隐藏状态（Final Hidden State）：结合更新门和候选隐藏状态，更新最终的隐藏状态。
- 优点：
  参数更少：由于结构的简化，GRU 的参数数量比 LSTM 少，这使得训练过程更快，模型更轻量
  性能相当：尽管结构更简单，GRU 在许多任务中的性能与 LSTM 相当，甚至在某些情况下更好。



Summary

- GRU 与 LSTM 的对比
  - **结构复杂度**：
    LSTM：包含输入门、遗忘门和输出门，结构复杂。
    GRU：包含更新门和重置门，结构简单。
  - **参数数量**：
    LSTM：参数较多，训练过程较慢。
    GRU：参数较少，训练过程更快。
  - **性能**：
    LSTM：在某些任务中表现更好，尤其是在需要捕捉非常长的依赖关系时。
    GRU：在许多任务中表现与 LSTM 相当，甚至在某些情况下更好，尤其是在计算资源有限的情况下。
- Gradient clipping只解决梯度爆炸，不解决梯度消失。
- Gating只解决梯度消失，不解决梯度爆炸

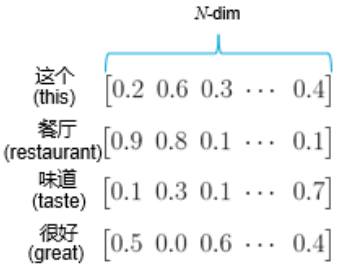
RNN Appliction

Sequential Input

处理具有时间序列的数据，做Sequence-Level Embadding，没啥写的。

Input Domain – Sequence Modeling

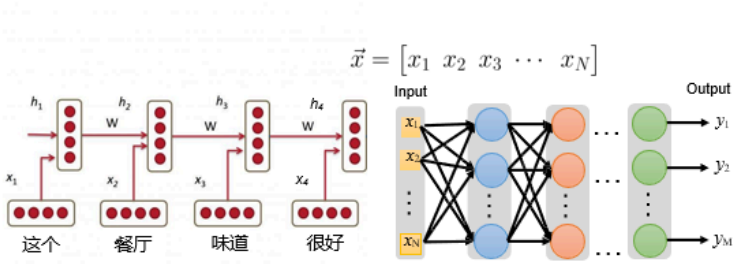
- Idea: aggregate the meaning from all words into a vector
- Method:
  - Basic combination: average, sum
  - Neural combination:
    - ✓ Recursive neural network (RvNN)
    - ✓ Recurrent neural network (RNN)
    - ✓ Convolutional neural network (CNN)
    - ✓ Transformer



How to compute $\vec{x} = [x_1 \ x_2 \ x_3 \ \cdots \ x_N]$

Sentiment Analysis

- Encode the sequential input into a vector using RNN



RNN considers temporal information to learn sentence vectors as classifier's input

---

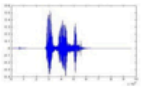
Sequential Output

Output Domain – Sequence Prediction

- POS Tagging

  "推荐我大学附近的餐厅" → 推荐/VV我/PN大学/NR附近/NN的/DEG餐厅/NN

- Speech Recognition



  → "大家好"

- Machine Translation

  "How are you doing today?" → "你好吗?"

The output can be viewed as a sequence of classification

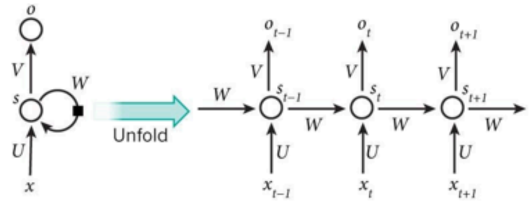
- **Aligned Sequential Pairs (Tagging)**
  以**POS Tagging**为例子，POS 标注（Part-of-Speech Tagging）是自然语言处理（NLP）中的一个任务，目标是为文本中的每个单词分配一个词性标签。词性标签是基于单词在句子中的语法功能和上下文来确定的。POS 标注是许多 NLP 应用的基础，如句法分析、信息提取和机器翻译。
  输入是word sequence,输出是corresponding POS tag sequence
- **Unaligned Sequential Pairs(Seq2Seq/Encoder-Decoder)**
  一般是两个级联的循环神经网络（RNN），一个用于编码，另一个用于解码，是神经机器翻译（NMT）领域的一种经典方法。编码器输入词序列，输出一个固定长度的上下文向量（context vector）或一系列隐藏状态（hidden states）。解码器输入通常是一个特殊的开始符号，以及编码器RNN生成的上下文向量或隐藏状态，输出是下一个词的预测（baseon编码器提供的信息）

  1.编码阶段：源语言的词序列被输入到编码器RNN中，该网络通过处理每个词并更新其内部状态来构建句子的表示。
  2.解码阶段：解码器RNN从编码器接收到的信息开始，逐步生成目标语言的词序列。在每一步中，它都会考虑之前的输出和编码器的信息来预测下一个最可能的词。
- What is the primary purpose of Sequence-to-Sequence (Seq2Seq) models in deep learning?
  Answer：*Handling tasks that involve variable-length sequences, such as machine translation and text summarization*
- What is the primary architectural limitation of traditional Seq2Seq models based on recurrent neural networks (RNNs)?
  Answer：*Lack of parallelism in training and inference*
- In the context of Seq2Seq, what challenge arises when relying solely on the final hidden state of the encoder to represent the entire input sequence?
  Answer：*Loss of sequential information from the beginning of the sequence*

Summary

- Language Modeling
  - RNNLM
- Recurrent Neural Networks
  - Definition

$$s_t = \sigma(W s_{t-1} + U x_t)$$
$$o_t = \mathrm{softmax}(V s_t)$$



  - Backpropagation through Time (BPTT)
  - Vanishing/Exploding Gradient – Clipping & Gating (LSTM & GRU)
  - Extension (Bi-direction RNN and Deep Bi-direction RNN)
- RNN Applications
  - Sequential Input: Sequence-Level Embedding
  - Sequential Output: Tagging / Seq2Seq (Encoder-Decoder)

❀❀❀❀❀❀❀❀❀❀❀❀❀❀❀❀❀❀❀简单的摘要，完结撒花
❀❀❀❀❀❀❀❀❀❀❀❀❀❀❀❀❀❀❀
