



南京航空航天大学
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

自然语言处理

计算机科学与技术学院

智周万物·道济天下

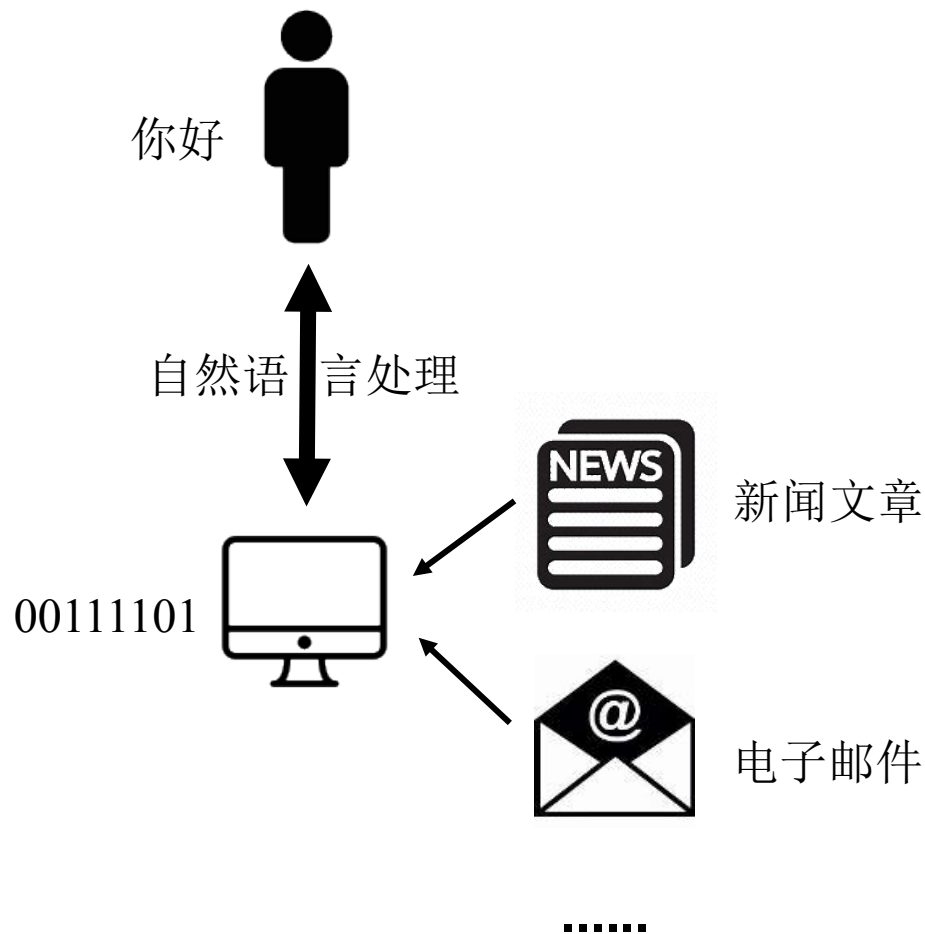
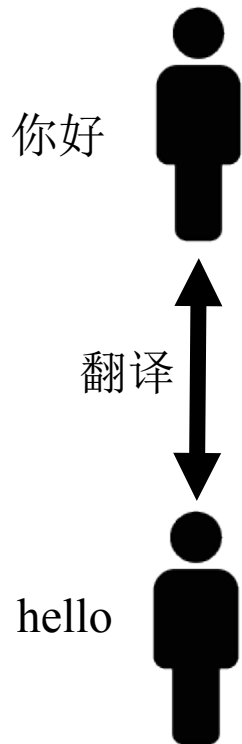
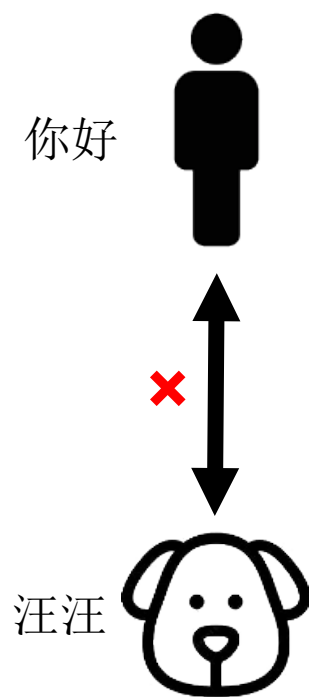


- 自然语言处理概述
- 词嵌入
 - 独热向量
 - word2vec
 - 跳元模型
 - 连续词袋模型
- 循环神经网络 (RNN)
- 长短期记忆网络 (LSTM)
- 门控循环单元 (GRU)

自然语言处理概述



南京航空航天大学
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS



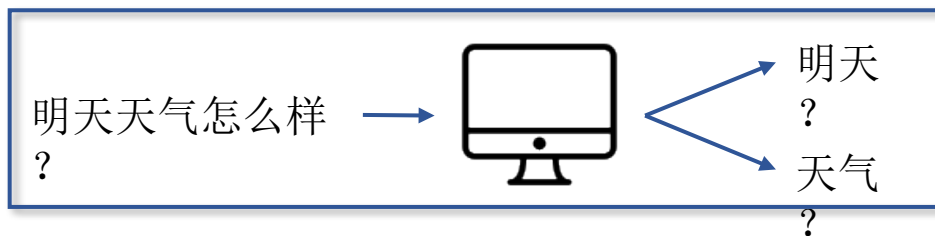
自然语言处理使计算机能够解读、处理和理解人类语言，成为人类和计算机之间沟通的桥梁

自然语言处理概述——基本任务



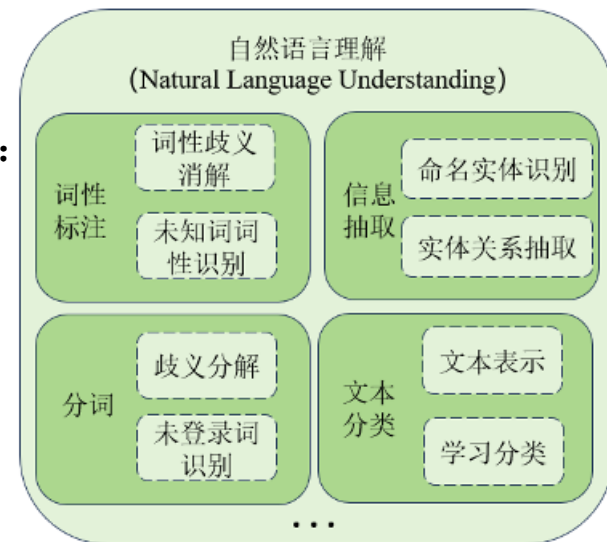
自然语言理解

人与计算机交流的第一步就是让计算机理解人类输入给它的信息。这类任务的研究目的是使计算机能够理解自然语言，从自然语言中提取有用的信息输出或用于下游任务



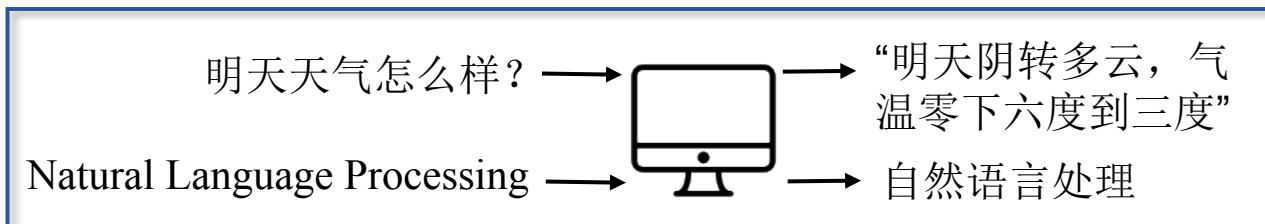
自然语言理解类任务包括:

- 词性标注
- 分词
- 文本分类
- 信息抽取



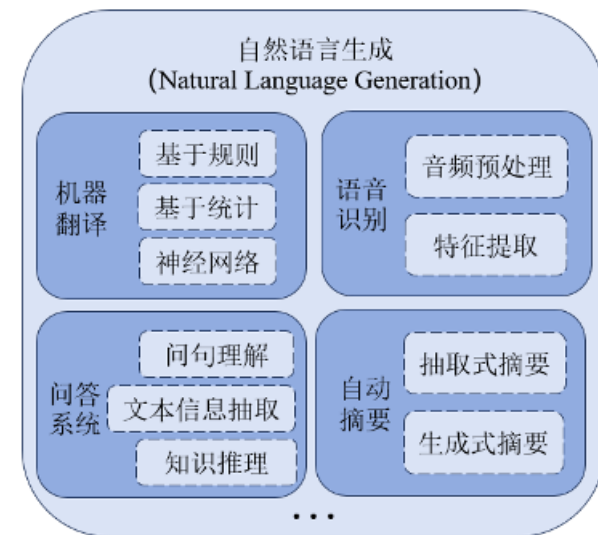
自然语言生成

计算机理解人类的输入后，我们还希望计算机能够生成满足人类目的的、可以理解的**自然语言形式的输出**，从而实现真正的交流。



自然语言生成类任务包括:

- 机器翻译
- 问答系统
- 自动摘要
- 语音识别



自然语言处理概述——发展历程



传统理论

深度学习兴起

大模型时代

70年代以后主要采用**基于统计**的方法。这种方法通常依靠大量的语言数据来学习，得到数据中词、短语、句子的概率分布，从而实现对语言的处理和分析。

自然语言处理领域**神经网络时代**，也逐渐开始，循环神经网络、卷积神经网络开始被广泛应用到自然语言处理领域

Bahdanau等人的工作使用注意力机制在机器翻译任务上将翻译和对齐同时进行，是第一个将**注意力机制**应用到**NLP领域**的科研工作。

BERT、GPT等大规模预训练语言模型出现，**大模型时代**逐渐到来

20世纪
50年代

70年代

2000

2013

2015

2017

BERT、GPT
2018年之后

20世纪50年代到70年代主要采用**基于规则**的方法。这种方法依赖于语言学家和开发者预先定义的规则系统，以便解析和理解语言。

Bengio等人提出**第一个神经语言模型**。这个模型将某词语之前出现的n个词语作为输入，预测下一个单词输出。模型一共三层，第一层是映射层，将n个单词映射为对应的词嵌入；第二层是隐藏层；第三层是输出层，使用softmax 输出单词的概率分布，是一个多分类器。

Mikolov等人提出了**word2vec**，大规模词向量的训练成为可能

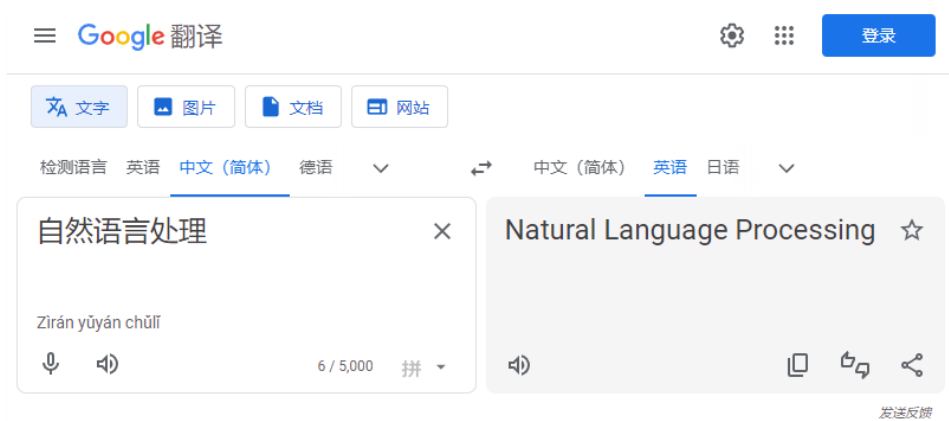
Transformer提出，它创造性地用非序列模型来处理序列化的数据，并且大获成功。

自然语言处理概述——应用领域



南京航空航天大学
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

1. 翻译软件



2. 聊天机器人



3. 语音助手



4. 搜索引擎



目录



? 计算机是无法直接读懂非数值的自然语言，
只有将其**转化为数值形式**才能被计算机处理

词嵌入

? 完成各种下游任务 → 神经网络模型

- 循环神经网络 (RNN)
- 长短期记忆网络 (LSTM)
- 门控循环单元 (GRU)

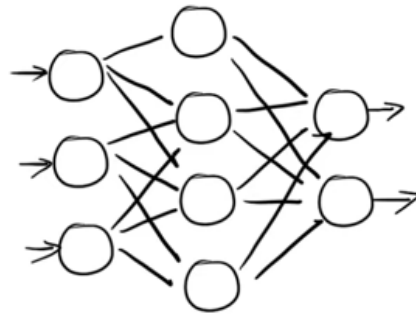
What is Natural Language Processing?

? 词嵌入技术!

What is Natural Language Processing



词向量



神经网络模型

RNN, LSTM, GRU, Transformer, ...

- 自然语言处理概述
- 词嵌入
 - 独热向量
 - word2vec
 - 跳元模型
 - 连续词袋模型
- 循环神经网络 (RNN)
- 长短期记忆网络 (LSTM)
- 门控循环单元 (GRU)

词嵌入——独热向量(One-hot Encoding)



文本 $\xrightarrow{?}$ 数值 最简单的方法就是用独热向量表示每个单词

□ 独热向量是指使用 N 位 0 或 1 对 N 个单词进行编码，其分量和类别数一样多，**类别对应的分量设置为 1**（即 one-hot），其余分量设置为 0。

例如，编码 apple、bag、cat、dog、elephant 五个单词，用 5 位向量进行编码：

apple = [1 0 0 0 0]

bag = [0 1 0 0 0]

cat = [0 0 1 0 0]

dog = [0 0 0 1 0]

elephant = [0 0 0 0 1]

✓ 优点：

- 独热向量容易构建

$$\cos(x, y) = \frac{x^T y}{\|x\| \|y\|} = 0$$

但任意两词之间余弦相似度为 0!

独热向量的维度等于词汇表大小，在词汇表较大时会变得非常长

✗ 缺点：

- 独热向量不能编码词之间的相似性
- 特征矩阵非常稀疏，占用空间很大

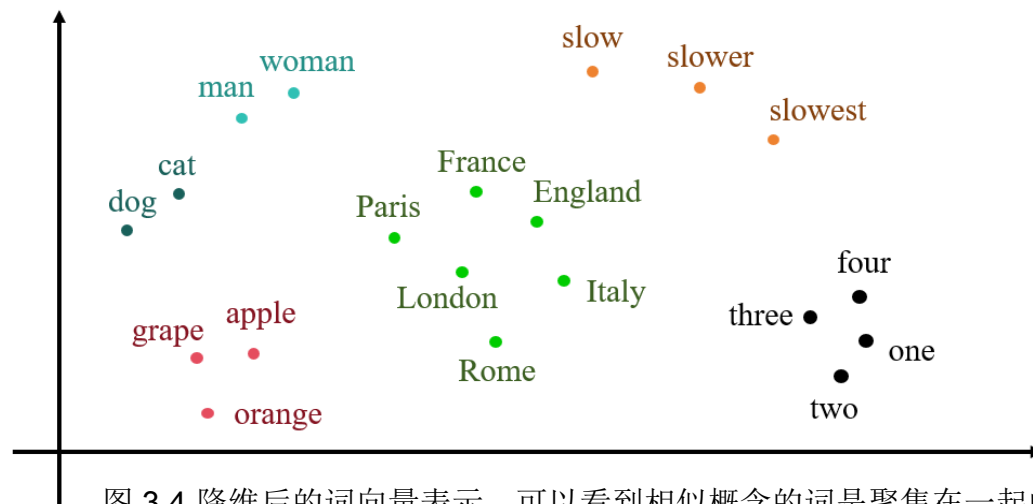
词嵌入——word2vec



我希望 向量

- 携带上下文信息，即词与词之间的联系能在词的向量表示中体现。
- 词的表示是稠密的，能用**更少的空间、更低的维数**表示更多的信息。

我希望 的效果



实现 word2vec!

图 3.4 降维后的词向量表示，可以看到相似概念的词是聚集在一起的

□ word2vec是一种词嵌入技术，也可被看作是一个神经网络模型，其参数是词向量，通过**预测上下文**来学习好的词向量。

和独热向量相比，word2vec生成的词向量具有以下优点：

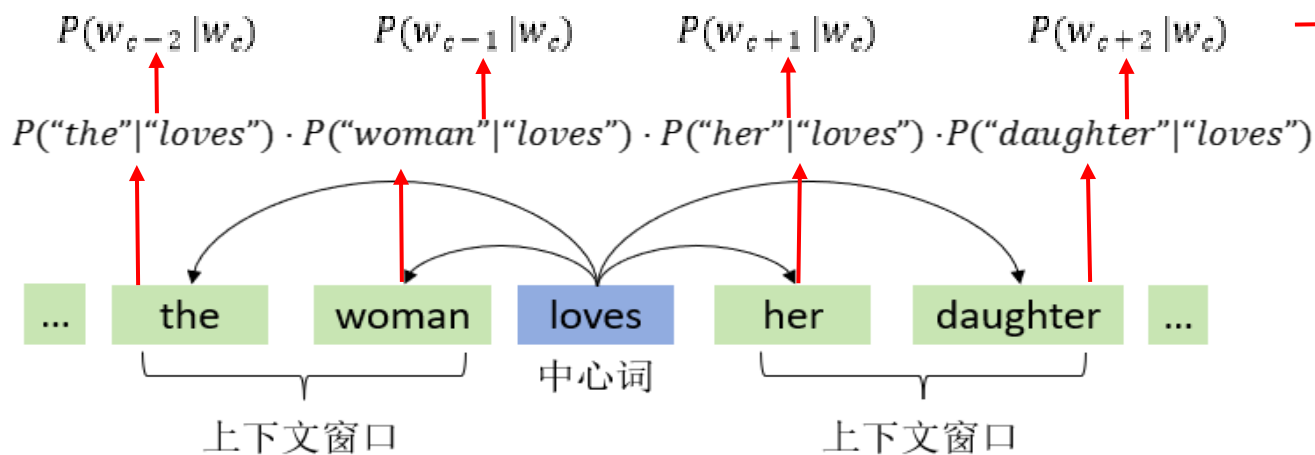
- 训练时利用上下文信息，词向量包含词的语义信息和词与词之间的联系。
- 维度更少，所以占用空间更少、计算成本更低。
- 通用性强，可用于各种下游 NLP 任务。

训练word2vec的常用方法有两种：跳元模型（Skip-Gram）和连续词袋（Continuous Bagsof Words: CBOW）

词嵌入——跳元模型



□ 根据中心词预测上下文词

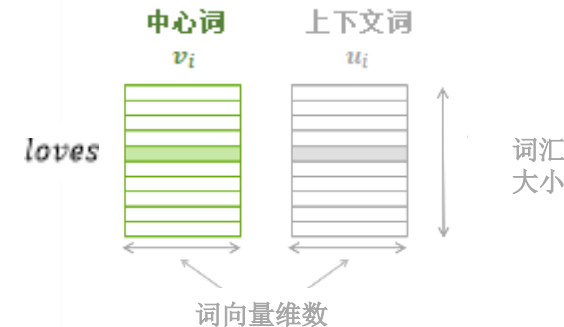


如何计算？

$$P(w_o|w_c) = \frac{\exp(u_o^T v_c)}{\sum_{i \in V} \exp(u_i^T v_c)} \quad \text{就是 softmax!}$$

对于每个单词 w_i 我们有两个向量：

- 当它是一个中心词时，用 v_i 表示该词
- 当它是一个上下文词时，用 u_i 表示该词



□ 目标函数（损失函数）

- 给定长度为 T 的文本序列，时刻 t 处的词表示为 $w^{(t)}$
- 上下文窗口大小为 m

则，似然函数为在给定任何中心词的情况下生成所有上下文的概率：

$$\text{Likelihood} = \prod_{t=1}^T \prod_{-m \leq j \leq m, j \neq 0} P(w^{(t+j)} | w^{(t)})$$

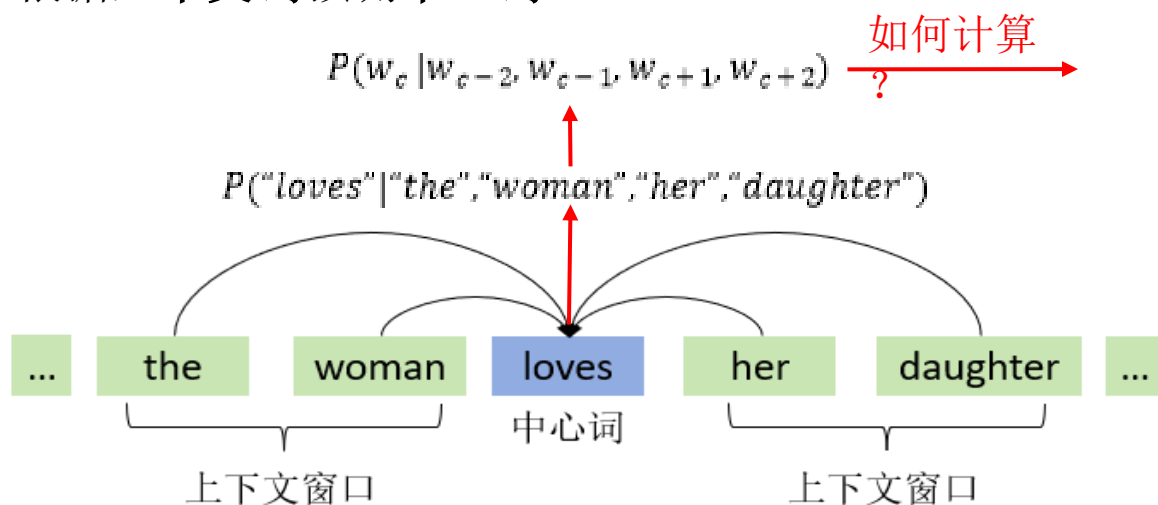
- 目标是最大化该似然函数，即最小化损失函数：

$$\text{Loss} = - \sum_{t=1}^T \sum_{-m \leq j \leq m, j \neq 0} P(w^{(t+j)} | w^{(t)})$$

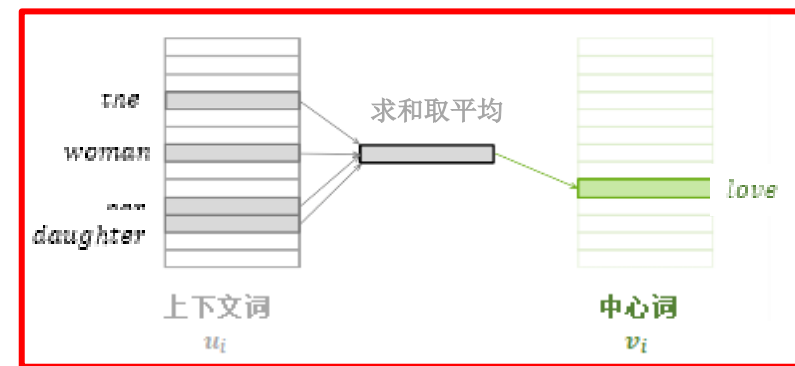
词嵌入——连续词袋模型



- 根据上下文词预测中心词



$$P(w_c | w_{o_1}, \dots, w_{o_{2m}}) = \frac{\exp(\frac{1}{2m} u_c^T (v_{o_1} + \dots + v_{o_{2m}}))}{\sum_{i \in V} \exp(\frac{1}{2m} u_i^T (v_{o_1} + \dots + v_{o_{2m}}))}$$



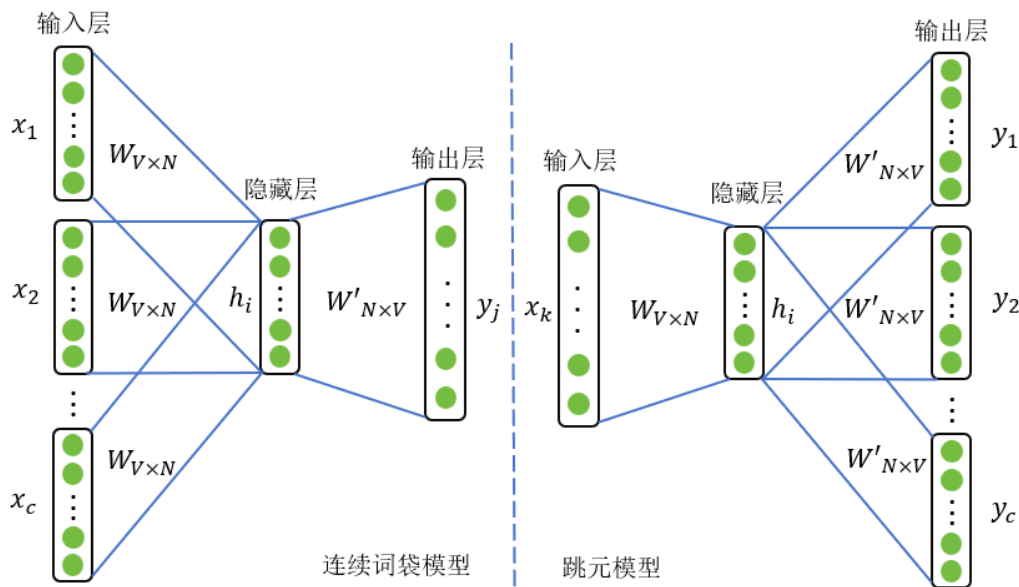
- 目标函数（损失函数）

- 给定长度为 T 的文本序列，时刻 t 处的词表示为 $w^{(t)}$
- 上下文窗口大小为 m

则，似然函数为在给定上下文词的情况下生成所有中心词的概率：
$$\text{Likelihood} = \prod_{t=1}^T P(w^{(t)} | w^{(t-m)}, \dots, w^{(t-1)}, w^{(t+1)}, \dots, w^{(t+m)})$$

- 目标是最大化该似然函数，即最小化损失函数：
$$\text{Loss} = - \sum_{t=1}^T P(w^{(t)} | w^{(t-m)}, \dots, w^{(t-1)}, w^{(t+1)}, \dots, w^{(t+m)})$$

词嵌入——连续词袋模型举例



使用单词的独热编码作为输入：

- the = [1 0 0 0 0]
- woman = [0 1 0 0 0]
- loves = [0 0 1 0 0]
- her = [0 0 0 1 0]
- daughter = [0 0 0 0 1]

假设训练得到的权重矩阵 $W_{V \times N}$ 为：

$$\begin{bmatrix} 1 & 2 & 3 & 0 & 1 \\ 1 & 2 & 1 & 2 & 2 \\ -1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

其中， $N=5$ 表示输入层单词的维数， $V=3$ 表示希望得到的词向量维数

现在将“the”输入，即与权重矩阵相乘：

$$\begin{bmatrix} 1 & 2 & 3 & 0 & 1 \\ 1 & 2 & 1 & 2 & 2 \\ -1 & 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

“the”的词向量

同理，可以得到每个单词的词向量为： $woman = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$, $her = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$, $daughter = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$

词嵌入——连续词袋模型举例



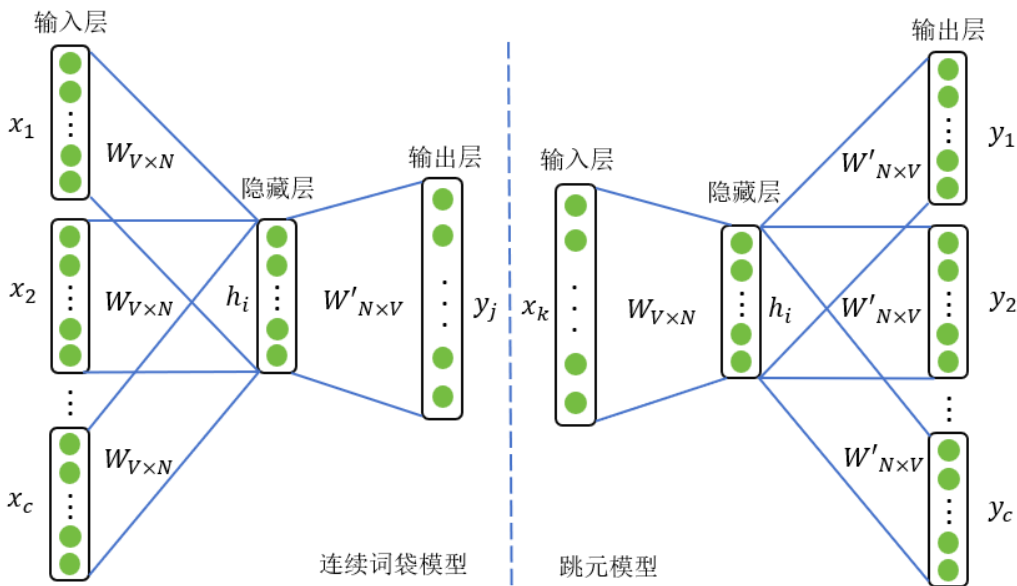
将得到的4个向量相加求平均作为输出层的输入：

$$\frac{1}{4} \left(\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 1.00 \\ 1.75 \\ 0.25 \end{bmatrix}$$

将该向量与权重矩阵 $W'_{N \times V}$ （也是网络训练的结果）相乘得到输出向量，最后将 **softmax** 作用在输出向量上，得到每个词的概率分布：

$$\text{softmax} \left(\begin{bmatrix} 1 & 2 & -1 \\ -1 & 2 & -1 \\ 1 & 2 & 2 \\ 0 & 2 & 0 \\ 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1.00 \\ 1.75 \\ 0.25 \end{bmatrix} \right) = \text{softmax} \left(\begin{bmatrix} 4.250 \\ 2.250 \\ 5.000 \\ 3.000 \\ -0.250 \end{bmatrix} \right) = \begin{bmatrix} 0.268 \\ 0.036 \\ 0.567 \\ 0.126 \\ 0.003 \end{bmatrix}$$

最后计算损失函数，反向传播，更新网络参数。



! 预测目标单词是训练网络的方式，获得网络的中间产物——权重矩阵 $W_{V \times N}$ 才是期望得到的，因为任意一个单词的独热向量乘该矩阵就能得到自己的词向量



- 自然语言处理概述
- 词嵌入
 - 独热向量
 - word2vec
 - 跳元模型
 - 连续词袋模型
- 循环神经网络 (RNN)
- 长短期记忆网络 (LSTM)
- 门控循环单元 (GRU)

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：
<https://d.book118.com/746100215234011004>