# NLP 大家一起复习鸭~！

博百家之所长，集群英之荟萃

大家最好是复习完一个章节之后，再来分享并查缺补漏喔

## Lecture 1: Introduction

### 1.1 nlp 的概念、挑战、常见应用

**概念**：自然语言处理（Natural language processing，NLP）是**语言学**、**计算机科学**和**人工智能**的一个分支领域，研究计算机与人类语言之间的关系、相互作用，特别是如何编程计算机来**处理**和**分析自然语言数据**。目标是使计算机能够理解文本的内容。

**挑战**：

1.歧义：

词汇歧义：门把手弄坏了；

词性歧义：Time flies like an arrow；

结构歧义：关于鲁迅的文章

2.一些新的语法现象：

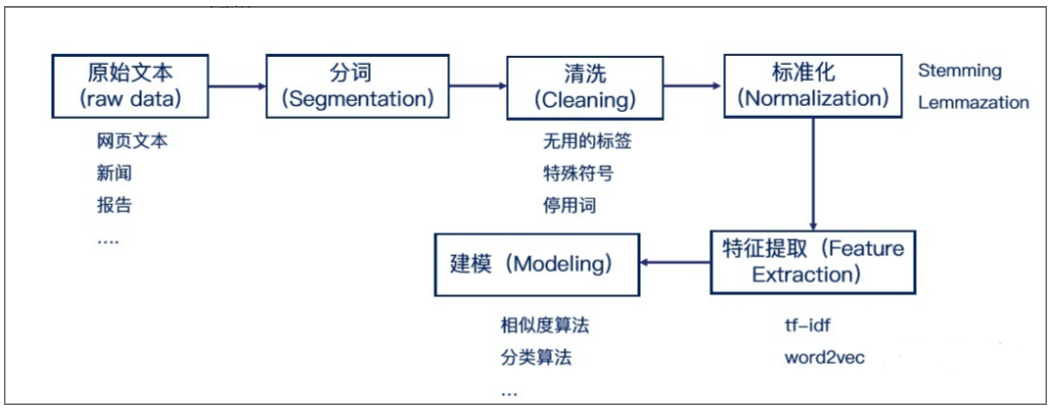
新的术语、人名、地名；

旧词新义，如苹果，凡尔赛；

新的语法，如百度一下。

**常见应用**：信息检索；文本生成；机器翻译；问答系统；知识图谱；情感分析……

### 1.2 文本预处理的步骤和方法



1. 去除数据中非文本部分，比如说html标签、非英文字符和标点符号
2. 分词；
3. 去除停用词
4. 词干化 **Stemming：**
5. 词形还原（Lemmatization）
6. 转为小写
7. 特征处理
8. 规范化 U.S.A. or USA uhhuh or uh-huh
9. 分句
10. …

## 1.3 自然语言处理和人工智能及机器学习的关系（了解）

与机器学习和深度学习一样，自然语言处理是人工智能的**一个分支**，因为其处理自然语言，所以它实际上是人工智能和语言学的交叉。

## 1.4 自然语言处理的不同发展阶段（了解）

第一阶段萌芽期(1956年以前)。在人工智能诞生之前，描述语言的自动机、上下文无关语法等等

第二阶段快速发展期(1957-1970)。这一时期分为两个流派，一个是基于规则方法的符号派(symbolic)，另一个是采用概率方法的随机派(stochastic)。

第三阶段低谷时期(1971 -1993)。基于隐马尔可夫模型的统计方法在语音识别、话语分析

第四阶段复苏融合时期(1994年至今)。里程碑如下：

2001年 - 神经语言模型

2008年 - 多任务学习

2013年 - Word嵌入

2013年 - NLP的神经网络

2014年 - 序列到序列模型

2015年 - 注意力机制

2015年 - 基于记忆的神经网络

2018年 - 预训练语言模型

# Lecture2 : N gram Language Model

## 2.1 语言模型 language model 的概念和目的

语言模型，简单来说就是用于判断一个句子是否通顺，我们的目标是要给一个句子赋予一个概率。

$$P(S)=P\left(W_1,W_2,\ldots,W_k\right)=p\left(W_1\right)P\left(W_2\mid W_1\right)\ldots P\left(W_k\mid W_1,W_2,\ldots,W_{k-1}\right)$$

语言模型的发展先后经历了文法规则语言模型、统计语言模型、神经网络语言模型。

## 2.2 基于 n gram 和极大似然估计的语言模型计算方法；

Markov Assumption（马尔可夫假设）：

$$P\left(w_i\mid w_1w_2\cdots w_{i-1}\right)\approx P\left(w_i\mid w_{i-k}\cdots w_{i-1}\right)$$

Unigram model：

$$P\left(w_1w_2\cdots w_n\right)=\prod_i P\left(w_i\right)$$

Bigram model：

$$P\left(w_i\mid w_1w_2\cdots w_{i-1}\right)\approx P\left(w_i\mid w_{i-1}\right)$$

...

极大似然法：根据大数定理，当样本数量足够大时，我们可以近似地**用频率来代替概率**。

计算流程非常简单（Bigram的例子）

1. 数出每个单词在语料库中的出现次数
2. 建表，计算所有P(word2 | word1)的概率，表示word1之后出现word2的概率
3. 计算某个句子的概率（避免下溢可以转换到log空间计算）

```
P(<s> I want english food </s>) =
  P(I|<s>)
     × P(want|I)
     × P(english|want)
     × P(food|english)
     × P(</s>|food)
   = 0.000031
```

## 2.3 语言模型的评估（了解）

一般方法是：

1. 在训练集(training set)中训练我们的模型参数
2. 使用区别于训练集的数据集作为测试集(test set)
3. 评估指标告诉我们模型在测试集上的表现如何

**Extrinsic evaluation：**

将模型运用到外部的任务上，间接的评估

比较两个模型A和B的最佳评估方法是：

1. 将每个模型放入一个任务中，比如拼写校正器、语音识别器、机器翻译系统
2. 运行任务，获得A和B的精度，包括
    - 正确纠正了多少拼写错误的单词
    - 有多少单词翻译正确
3. 比较A和B的精确度（可以使用困惑度来评价）

**intrinsic evaluation: perplexity**

Shannon Game：香农游戏。即给出前面n个单词，猜出第n+1个单词。

**困惑度**（Perplexity）。PPL是用在自然语言处理领域（NLP）中，衡量语言模型好坏的指标。

基本思想：给测试集的句子赋予较高概率值的语言模型较好

- Chain rule: $\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$
- For bigrams: $\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$

## 2.4 语言模型的平滑（了解）

拉普拉斯平滑，又称加1平滑。

解决语言模型的零概率问题

(只要在所有的计数中加上一个即可)

MLE estimate: $P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$

Add-1 estimate: $P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)+1}{c(w_{i-1})+V}$

# Lecture 3: Text Classification

## 3.1 文本分类任务的定义和意义

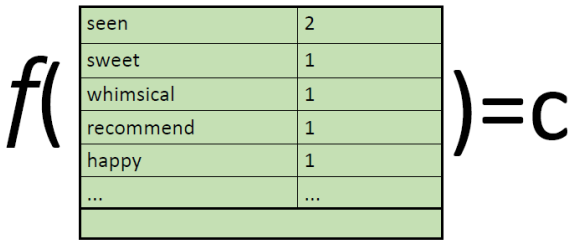
**文本分类任务**：根据已经定义好的类别标签对现有的一段文本进行标注的任务

- 邮件过滤：比如鉴定一封邮件是否是骚扰或者广告营销等垃圾邮件
- 给杂志、论文贴上标签
- 情感分析：划出句子中的正面、负面的词汇
- 舆情分析
- 新闻主题分类（文章分类）：根据文章内容（或者结合标题）给新闻等其他文章一个类别
- 作为其他自然语言处理系统的一部分，比如问答系统中对问句进行主题分类。

## 3.2 朴素贝叶斯分类器的原理与训练方法

朴素贝叶斯分类器是基于贝叶斯规则的。依赖于一个简单的文档表示方法——Bag of words。

Bag of words中包含每个单词的出现频率，我们可以通过它来计算、学习出一个分类器。




对于一个文档d和一个类c有：

$$P(c|d) = \frac{P(dc)}{P(d)} = \frac{P(d|c)P(c)}{P(d)}$$

$$c_{map} = argmax_{c \in C} P(c|d)$$

$$= argmax_{c \in C} \frac{P(d|c)P(c)}{P(d)} \quad \text{贝叶斯公式}$$

$$= argmax_{c \in C} P(d|c)P(c) \quad \text{最大化后验概率P(d)可以看成一个常数}$$

$$= argmax_{c \in C} P(x_1, x_2, \ldots, x_n|c)P(c)$$

$$c_{NB} = argmax_{c \in C} P(c) \prod_{x_i \in X} P(x_i|c) \quad \text{独立假设}$$

$$= argmax_{c \in C} [logP(c) + \sum_{i \in positions} logP(x_i|c)] \quad \text{避免下溢，转换到对数空间}$$

上面的公式中：

- map指的是maximum a posteriori最大后验概率，即我们要找出一个最大可能的类；
- 由于所有类的算出的的分母都是P(d)，所以可以省去
- P(c)是先验概率，即某一个类的出现概率
- 如何估算P(x1,x2...xn | c)，我们可以假设单词的位置、顺序是无关紧要的，且假设之间相互独立，则我们可以得到最后一条式子。
- positions指的是所有单词在文本中的位置
- 考虑小数精度问题、计算的方便性，我们可以取log

在上面的式子中，主要求两个值：P(c)和P(xi|c)

我们可以单纯的使用单词出现的频率来表示这两个值：

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}$$

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

考虑有的单词没有出现在训练集中，可能出现P等于0的情况，所以需要进行拉普拉斯平滑：

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j) + 1}{\sum_{w \in V} count(w, c_j) + |V|}$$

出现在测试集但不在训练集的词语：忽略

## 3.3 基于朴素贝叶斯实现情感分析(sentiment analysis) （了解）

| | Cat | Documents |
|---|---|---|
| Training | - | just plain boring |
| | - | entirely predictable and lacks energy |
| | - | no surprises and very few laughs |
| | + | very powerful |
| | + | the most fun film of the summer |
| Test | ? | predictable with no fun |

1. Prior from training:

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}$$

P(-) = 3/5

P(+) = 2/5

2. Drop "with"

3. Likelihoods from training:

$$p(w_i|c) = \frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|}$$

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \quad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \quad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \quad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

4. Scoring the test set:

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

对于情绪分析，似乎单词的出现次数并不能告诉我们更多信息，比如说fantastic出现五次和出现一次没有太大区别。所以我们可以将其优化成Binary multinominal Naïve Bayes 或者 binary Naïve Bayes——将所有的单词计数记为1。

1. 计算先验部分
2. 丢掉 stop words（可选的）
3. 计算条件概率
4. 测试样本的条件概率

# Lecture 4: Vector Semantics & Embeddings (I)

## 4.1 词向量的动机与意义；

词语的含义可根据上下文确定。不管是机器学习还是深度学习本质上都是对数字的数字，Word Embedding(词嵌入) 做的事情就是将单词映射到向量空间里，并用向量来表示

意义

1. 词语意思由数字表达
2. 获取词语特征，特征是一个向量
3. 可总结未见过的词
4. 计算词语的相似度
5. 代表文本特征

**动机：**

我们能否建立一种理论来表示单词的意义，至少能解释部分需求？注意到，“单词由它们的使用环境（单词周围的单词）所定义”。

想法1：通过语言分布来定义意义。通过词语在语言使用中的分布来定义它的意义，即使用相邻的单词或语法环境。

想法2：作为多维空间中分布的一个点，确定意义。如1个单词的3种影响维度包含警觉度、唤醒度、支配度，在这种情况下，1个单词由三维空间中的一个向量所表示。

在这种表示方法下，每个单词由一个向量表示，相似的单词在语义空间中的距离很近，通过查看文本中相邻的单词，就可以自动创造出**语义空间**。由此，将单词进行向量化表示的方法叫做“embedding”，即单词被“嵌入”到了语义空间中。词嵌入即NLP中一种标准的语义表示方法，现代NLP算法多使用这种方法进行词语的语义表示。同时，词嵌入也是一种代表相似意义的细粒度模型。

**意义：**

将单词向量化后，能帮助我们做什么呢？

考虑对句子进行语义分析。如果使用单词，那么一个特征（feature）就是一个单独的单词，如”Feature 5: ‘The previous word was “terrible”’ “，Feature 5需要“terrible”这一个单词进行训练和测试。而如果使用词嵌入，那么特征就是一个向量，类似于[35, 22, 17, …]，在测试文本中，我们可能会见到一个相似的向量[34, 21, 14, …]，虽然两个向量不完全相同，但是我们可以得知，测试集中的该向量与给定的特征向量表达的意义相同，即“we can generalize to similar but unseen words”，总结未见过的词。

1. 使用words，需要完全相同的单词来进行训练和测试
2. 使用embeddings，可以泛化到相似但是没见过的词

## 4.2 用 TF IDF 获取词向量和文本向量的方法；

TF-IDF（term frequency–inverse document frequency）是一种用于信息检索与数据挖掘的常用**加权**技术。TF是**词频**(Term Frequency)，IDF是**逆文本频率指数**(Inverse Document Frequency)。

对于单词t在文本d中的**ti-idf值**，如下：

$$w_{t,d} = tf_{t,d} \times idf_i$$

其中，像“the”或“it”的**idf值**将会非常小

**tf的计算方法**

$$tf_{t,d} = count(t, d)$$

通常不使用原始数据，我们将其压缩一下：

$$tf_{t,d} = log_{10}(count(t, d) + 1)$$

**idf的计算方法**

注意df不是指该单词在所有文档中出现的总次数(collection Frequency)，而是指该单词出现在了多少个文档中(Document Frequency)

| | Collection Frequency | Document Frequency |
|---|---|---|
| Romeo | 113 | 1 |
| action | 113 | 31 |

$idf_t = log_{10}(\frac{N}{df_t})$ 其中 $N$ 是总的文本数

可以注意到，加入一个单词在所有训练的文本中都有出现，那么它的idf就会是0。而出现的频率越低，它的idf就越大。

Raw counts:

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| battle | 1 | 0 | 7 | 13 |
| good | 114 | 80 | 62 | 89 |
| fool | 36 | 58 | 1 | 4 |
| wit | 20 | 15 | 2 | 3 |

tf-idf: $w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| battle | 0.074 | 0 | 0.22 | 0.28 |
| good | 0 | 0 | 0 | 0 |
| fool | 0.019 | 0.021 | 0.0036 | 0.0083 |
| wit | 0.049 | 0.044 | 0.018 | 0.022 |

TF-IDF：

$\text{tf}_{t,d} = \text{count}(t, d)$

$\text{tf}_{t,d} = \log_{10}(\text{count}(t, d) + 1)$

$idf$: 词语在很多文档出现 重要性降低

$\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right)$ N是集合中的文档总数。

$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$

得到矩阵

横着看是词向量

竖着看是文本向量

## 4.3 基于 TF IDF 计算词语 文本相似度的方法；

余弦距离

$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

## 4.4 词义 word meaning（了解）

1. 一词多义（lexical semantics）
2. 同义词（Synonymy）
3. 相似度（Similarity）
4. Word relatedness（词相关性）
5. Semantic field（语义域）
6. Antonymy（反义）
7. Connotation（sentiment，隐含意义）：一个词语可以在 3 种维度上都有一个分数，这样这个词就可以用向量表示。

## 4.5 基于互信息 mutual information 获取词向量的方法；（了解）

经常会用到PMI（Pointwise Mutual Information，逐点互信息）这个指标来衡量两个事物之间的相关性（比如两个词）：

$$PMI(w_1,w_2) = log\frac{p(w_1,w_2)}{p(w_1)p(w_2)}$$

虽然PMI的取值是负无穷到正无穷，但是取负值是有问题的。所以我们用0取代负的PMI，得到PPMI：

$$PPMI(w_1,w_2) = max(log\frac{p(w_1,w_2)}{p(w_1)p(w_2)},0)$$

Original word-word matrix

| | computer | data | result | pie | sugar | count(w) |
|---|---|---|---|---|---|---|
| **cherry** | 2 | 8 | 9 | 442 | 25 | 486 |
| **strawberry** | 0 | 0 | 1 | 60 | 19 | 80 |
| **digital** | 1670 | 1683 | 85 | 5 | 4 | 3447 |
| **information** | 3325 | 3982 | 378 | 5 | 13 | 7703 |
| **count(context)** | 4997 | 5673 | 473 | 512 | 61 | 11716 |

Resulting PPMI matrix (negatives replaced by 0)

| | computer | data | result | pie | sugar |
|---|---|---|---|---|---|
| cherry | 0 | 0 | 0 | 4.38 | 3.30 |
| strawberry | 0 | 0 | 0 | 4.10 | 5.51 |
| digital | 0.18 | 0.01 | 0 | 0 | 0 |
| information | 0.02 | 0.09 | 0.28 | 0 | 0 |

# Lecture 5: Vector Semantics & Embeddings (II)

## 5.1 Word2Vec 模型计算词向量的原理

- 基于**二进制预测任务**来训练一个**分类器**：w会喜欢出现在c的附近吗？(true/false)
- 寻找到一个中心词target，在设定一个窗口大小（如下图为+/-2）下，如果c出现在target周围，就设为T，否则为F。
- 我们并不关心这个分类器本身，而是将其权重作为词向量。
- **自监督**的学习方式。标签不是人为设置的，而是根据单词是否出现在窗口中自己形成的。c和w之间是非常弱的标签关系(它们可能在某个地方邻近，在其它可能又没有)。

> 一个简单的例子：
>
> “她们 夸 吴彦祖 帅 到 没朋友”
>
> “她们 夸 我 帅 到 没朋友”
>
> 由于“吴彦祖”和“我”使用邻近的词语表示的词向量是非常近似的，所以“我”约等于“吴彦祖”

## 5.2 Word2Vec 模型的训练方法

1. V个随机初始化的d维向量作为初始的词嵌入；
2. 基于向量的相似度来对分类器进行训练；（如果两个向量是相似的，那么它们的点积是会非常大的。所以我们可以直接使用点积来表示相似性。）
3. 基于一个语料库，将同时出现的词作为**正例**，不同时出现的词作为**反例**；

   最大化:target word, context word 的相似性，最小化:negative examples 单词对的相似性
4. 通过**随机梯度下降**的方式来训练分类器，最终提高分类器区分这些词向量的性能；
5. 扔掉分类器，仅仅保留词向量。

## 5.3 Word2Vec 中负采样 negative sampling 的原理和目的

~~！！！不太清楚~~

哦哦哦，重点原来不在“负”，而在“采样”

**原理：**

在神经网络分类器中有大量的权重矩阵，若词表中有V个词，词向量的维数为300，则每一次更新参数需要更新V*300个参数，在V非常大时，梯度下降的计算量非常大，导致模型训练速度慢。

而负采样是在正确的词之外随机采样若干个错误的词作为样本，（在论文中，作者指出对于小规模数据集，选择**5-20个**negative words会比较好，对于大规模数据集可以仅选择**2-5个**negative words）比如采样5个负样本，那么更新参数时就可以只更新1个正样本加5个负样本的权重，一共只更新6*300个参数，使得计算效率大幅提高。

**目的：**

负采样是用来提高训练速度并且改善所得到词向量的质量的一种方法。不同于原本每个训练样本更新所有的权重，**负采样每次让一个训练样本仅仅更新一小部分的权重**，这样就会降低梯度下降过程中的计算量。

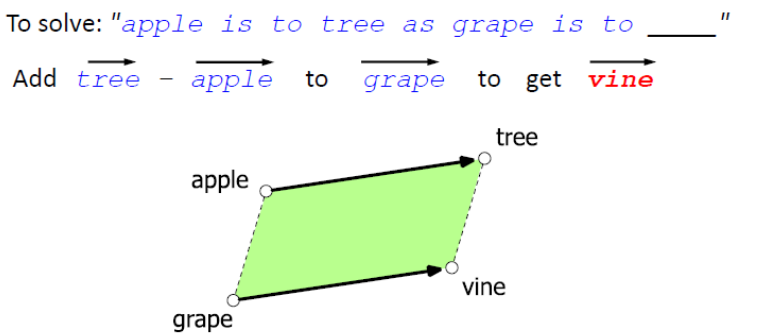
## 5.4 词向量的属性 properties （了解）

**(1)窗口大小：滑动窗口决定词的语境**

- 小窗口(C=+/-2)，邻近的单词是同一分类法下语法相似的词。比如Hogwarts和其他小说里的学院
- 大窗口(C=+/-5)，邻近的单词是同一语义领域下的。比如Hogwarts和Dumbledore

**(2) 类比关系(analogical relations)**

使用向量的平行关系来进行类比推理




对于稀疏的词向量或稠密的词向量都可以使用这种方法：

$$\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} \text{ is close to } \overrightarrow{queen}$$

$$\overrightarrow{Paris} - \overrightarrow{France} + \overrightarrow{Italy} \text{ is close to } \overrightarrow{Rome}$$

## 5.5 Word2Vec 中 skip gram 的原理（了解）

简单理解：给定input word，预测context words (输入中心词并预测上下文中的单词)

skip-gram是Word2Vec分类器的一种形式：给出一个目标单词w和窗口L中的上下文单词c_{1:L}，基于w和c_{1:L}之间的相似度（转换成词向量，再做点乘），估计w出现在窗口中的概率：

$$P(+|w,c) = \frac{1}{1+\exp(-c \cdot w)}$$

（正例的计算方法，负例用1减去这个）。这需要得到所有单词的词向量。

对于每个单词，初始化目标词向量和上下文词向量。然后开始训练，移动窗口，求出窗口中正例的概率值，再选取k个负例，求出概率值。这样，给定正例和负例训练样本和初始化的词向量，目标就是调整词向量，使得对正例的概率值最大化，对负例的概率值最小化。

方法：基于loss损失函数，使用随机梯度下降。

最终词向量收敛后，将目标词向量和上下文词向量简单点加，就得到结果词向量。

# Lecture 6: Neural Language modeling

## 6.1 如何通过神经网络计算语言模型

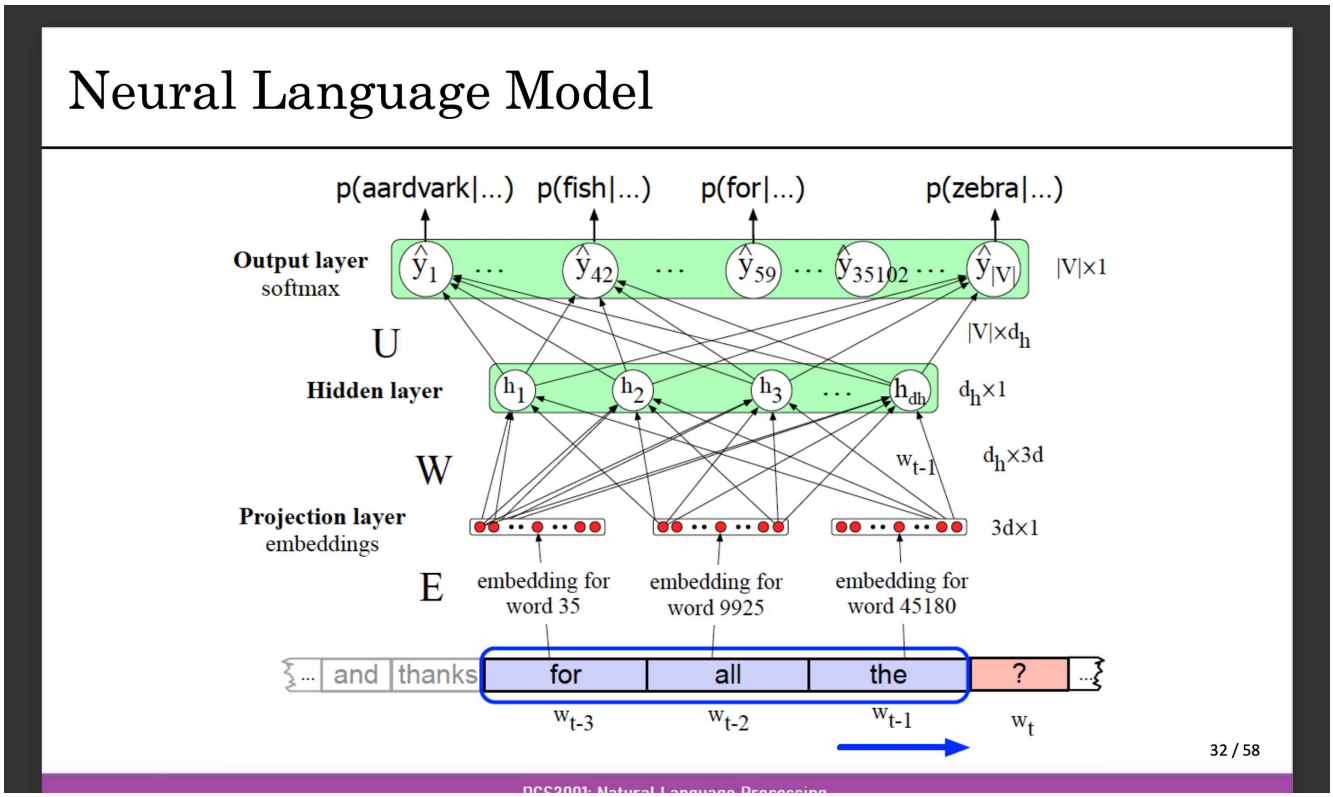
利用 embedding 映射词语，通过神经网络，最后 softmax

语言模型：Language Modeling，给定单词序列，计算下一个单词的出现的是什么的概率。之前已经看过n-gram语言模型，但是神经语言模型比n-gram表现更好。

**前馈神经语言模型**

任务：给定 $w_{t-1}, w_{t-2}, w_{t-3}$ ...，预测下一个单词 $w_t$

将单词 $w_{t-3}, w_{t-2}, w_{t-1}$ 的词向量表示输入到神经网络中，利用前馈神经网络，输出所有可能的预测词对应的概率 $P(w_t|w_{t-3}^{t-1})$ ：



## 6.2 相比n-gram，神经语言模型的优势

- 效果更好
- 统计语言模型对训练集要求高 不能获得词语之间的关联

  训练集有 I have to make sure that the cat gets fed. 但是没用 dog gets fed
- 参与计算的是词嵌入 (能够表达词语的相似度)，例如，dog 和 cat 相近的语义关系

(1) 长距离依赖，具有更强的约束性；

(2) 避免了数据稀疏所带来的OOV（out of vocabulary）问题；

(3) 好的词表征能够提高模型泛化能力。

## 6.3 计算图 （了解）

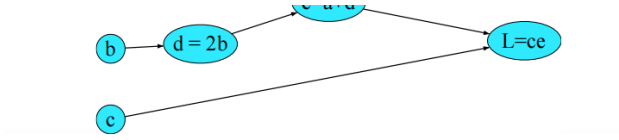
计算图被定义为有向图，其中节点对应于数学运算。

一个计算图表示计算一个数学表达式的过程。

Example: $L(a,b,c) = c(a+2b)$

Computations:

$$d = 2*b$$
$$e = a+d$$
$$L = c*e$$




## 6.4 反向微分（了解）

就是链式求导。节点的微分值等于路径上的微分值相乘。

反向拓扑顺序遍历节点，计算梯度

# Lecture 7: Part of Speech and Named Entities

## 7.1 什么是词性标注 Part of Speech tagging 及意义；

**概念：**

给句子中的所有词语标注对应的词性标签，是一种最低级别的句法分析。

| John | saw | the | saw | and | decided | to | take | it | to | the | table. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NNP | VBD | DT | NN | CC | VBD | TO | VB | PRP | IN | DT | NN |

**意义：**

· 在 NLP 很多任务中有用：句法分析、机器翻译、文本对话、情感分析

· 计算不同句子的相似度，学习语言学的词语和语义转移的规律

给句子中的所有词语标注对应的词性标签，是一种最低级别的句法分析。

## 7.2 什么是命名实体识别 Named Entity tagging 及意义；

**概念：**

是指识别文本中具有特定意义的实体，主要包括人名、地名、机构名、专有名词（地缘政治实体）等。简单的讲，就是识别自然文本中的实体指称的边界和类别。

标出具有特定含义的词语

Citing high fuel prices, [ORG **United Airlines**] said [TIME **Friday**] it has increased fares by [MONEY **$6**] per round trip on flights to some cities also served by lower-cost carriers. [ORG **American Airlines**], a unit of [ORG **AMR Corp.**], immediately matched the move, spokesman [PER **Tim Wagner**] said. [ORG **United**], a unit of [ORG **UAL Corp.**], said the increase took effect [TIME **Thursday**] and applies to most routes where it competes against discount carriers, such as [LOC **Chicago**] to [LOC **Dallas**] and [LOC **Denver**] to [LOC **San Francisco**].

**意义：**

· 在 NLP 很多任务中有用：情感分析、问答系统、信息抽取

情感分析中：情绪的对象需要识别。例如，消费者的情绪是对特定公司的还是对特定的一个人？

问答系统中：问题里面出现问答实体，回答有关一个实体的问题，需要将实体完整地识别出来

信息提取中：从文本中提取有关实体的信息

## 7.3 如何实现词性标注；（了解）

Standard algorithms for POS taggingStandard algorithms for POS tagging

Hidden Markov Models 隐马尔可夫模型

Conditional Random Fields (CRF)

Neural sequence models (RNNs or Transformers)

Large Language Models (like BERT), finetuned

## 7.4 如何实现命名实体识别；（了解）

Standard algorithms for NER

Hidden Markov Models 隐马尔可夫模型

Conditional Random Fields (CRF)

Neural sequence models (RNNs or Transformers)

Large Language Models (like BERT), finetuned

**BIO Tagging**

B：属于命名实体，边缘位置

I：属于命名实体，中间位置

O：不属于命名实体

- B: token that begins a span
- I: tokens inside a span
- O: tokens outside of any span

# Lecture 8: Constituency Parsing

## 8.1 什么是句法结构 syntax structure

分为两种：成分句法 和 依存句法

## 8.2 什么是成分句法 constituency structure

称成分句法分析、短语句法分析，亦即context-free grammers(CFGs 上下文无关句法)，将语句视为嵌套的短语组合：

1. 短语结构将单词组织成嵌套的成分
2. 最开始的单元是单词，每个单词被赋予一个词性标注

```
the, cuddly, cat, by, the, door
Det, Adj, Noun, Prep, Det, Noun
```

3. 单词结合可以得到带有类别的短语

```
the cuddly cat, by the door
NP → Det Adj Noun, PP → Prep Det Noun
```

4. 短语可以递归地组合成更大的短语

```
the cuddly cat by the door
NP → NP PP
```

**应用场景**

- 语法分析：如果一个句子不能被解析，它可能有语法错误（或者至少很难读）
- 用作下游任务的中间表示
  - 机器翻译
  - 信息提取
  - 问答

## 8.3 什么是上下文无关文法 context free grammars 及作用；

一个最广泛使用的用于建模成分句法的规范系统。在计算机科学中，若一个形式文法G = (N, Σ, R, S)（分别对应（非叶子，叶子，规则，根节点））的产生式规则都取如下的形式：V->*w*，则称为上下文无关文法。其中 V∈N ，*w*∈(N∪Σ)* 。上下文无关文法取名如此的原因就是因为字符 V 总可以被字串 w 自由替换，而无需考虑字符 V 出现的上下文。

上下文无关文法重要的原因在于它们**拥有足够强的表达力，又足够简单**

**上下文无关文法（CFG）**

**□ CFG由一系列规则组成，每条规则给出了语言中的某些符号可以被组织或排列在一起的方式.**

符号被分成两类：

- 终结点(叶子节点)：就是指单词，例如 book；
- 非终结点(内部节点)：句法标签，例如 NP 或者 NN；

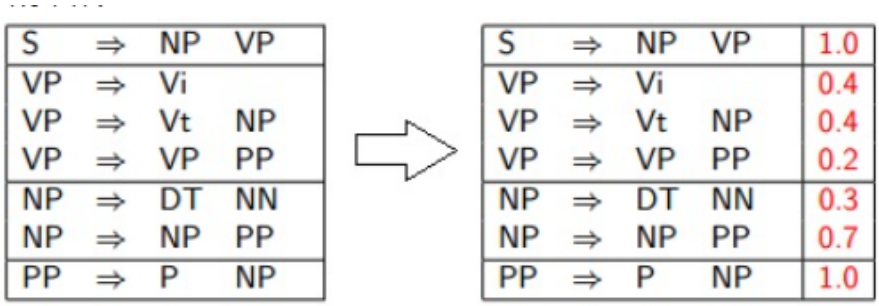
规则是由一个“→”连接的表达式：

- 左侧：只有一个 non-terminal；
- 右侧：是一个由符号组成的序列；

## 8.4 概率上下文无关文法 Probabilistic Context free Grammar 的意义；

CFG 赋予了语言一种层次化的结构。但是根据一个 CFG 构建语法分析树，往往不止一个；对于可能产生多种语法分析结果的问题，我们该如何应对呢？引入概率上下文无关文法（PCFG，Probabilistic context-free grammar）：给每棵树计算一个概率！

PCFGs其实就是在 **Context-Free Grammars**的基础上对每个RULE，也就是R中的内容加以概率限制



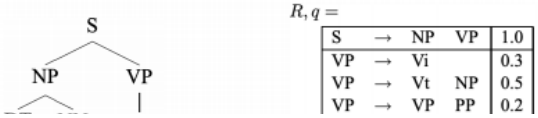
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| S | ⇒ | NP | VP | S | ⇒ | NP | VP | 1.0 |
| VP | ⇒ | Vi | | VP | ⇒ | Vi | | 0.4 |
| VP | ⇒ | Vt | NP | VP | ⇒ | Vt | NP | 0.4 |
| VP | ⇒ | VP | PP | VP | ⇒ | VP | PP | 0.2 |
| NP | ⇒ | DT | NN | NP | ⇒ | DT | NN | 0.3 |
| NP | ⇒ | NP | PP | NP | ⇒ | NP | PP | 0.7 |
| PP | ⇒ | P | NP | PP | ⇒ | P | NP | 1.0 |

## 8.5 如何得到概率上下文无关文法中每条规则 rule 的概率；（了解）

## 8.6 如何计算整棵成分句法树的概率；（了解）

对于树的的规则，累乘概率

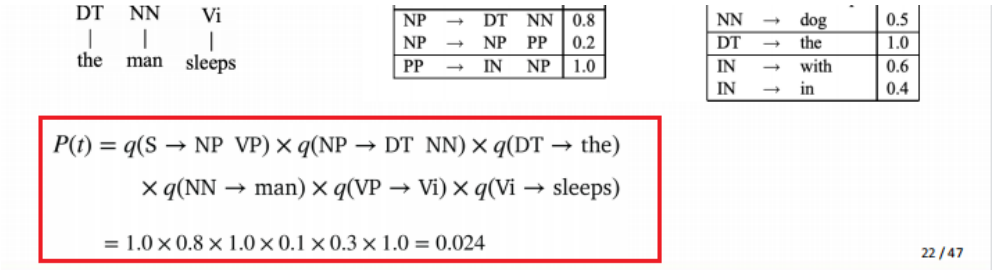
For any derivation (parse tree) containing rules:
$\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_l \rightarrow \beta_l$, the probability of the parse is: $\prod_{i=1}^{l} q(\alpha_i \rightarrow \beta_i)$




$R, q =$

| | | | | |
|---|---|---|---|---|
| S | → | NP | VP | 1.0 |
| VP | → | Vi | | 0.3 |
| VP | → | Vt | NP | 0.5 |
| VP | → | VP | PP | 0.2 |

| | | | |
|---|---|---|---|
| Vi | → | sleeps | 1.0 |
| Vt | → | saw | 1.0 |
| NN | → | man | 0.1 |
| NN | → | woman | 0.1 |
| NN | → | telescope | 0.3 |




## 8.7 CKY 算法和词汇化概率上下文无关文法 Lexicalized PCFGs )（了解）

动态规划!

# Lecture 9: Dependency Parsing

## 9.1 什么是依存句法解析 dependency parsing

- 分析出句子当中所有词汇之间的依存关系，用有向边来表示两个成分之间的相互依存关系，源词汇为**支配词**，终点词汇为**被支配词**。依存语法存在一个共同的基本假设：句法结构本质上包含词和词之间的依存（修饰）关系。一个依存关系连接两个词，分别是核心词（head）和依存词（dependent）。依存关系可以细分为不同的类型，表示两个词之间的具体句法关系
- 依存文法分析考虑的是句子中单词与单词之间的依存关系，而这种依存关系其实就是我们学习的语法，例如主谓、动宾、形容词修饰名词等等。根据这样的关系，我们可以将原来的句子转化成一棵树。

## 9.2 依存句法 (dependency structure 和成分句法 constituency structure 的区别;

1. 成分句法把句子组织成**短语的形式**，而依存句法主要揭示了句子中词的**依赖关系**。
2. 两种句法结构都能够揭示句子中不同的信息，需要使用到句子当中的短语结构就使用成分句法分析，要使用到词与词之间的依赖关系就使用依存句法分析。
3. 依存句法树能够由成分句法树转换而来，但成分句法树不能够从依存句法树转化而来。（成分句法树——>依存句法树，~~依存句法树——>成分句法树~~）

## 9.3 相比成分句法，依存句法有什么优势；

1. 依存关系更加接近于实际语义，有利于对句子的语义方面的理解。
2. 定义比成分句法更加简单，分析高效
3. 能够有效建模长距离依赖关系，依存句法更加适合词序列灵活，自由的语言。
4. 依存关系能够建立词的语义关系和参数之间的联系，有利于执行多种自然语言处理的任务。

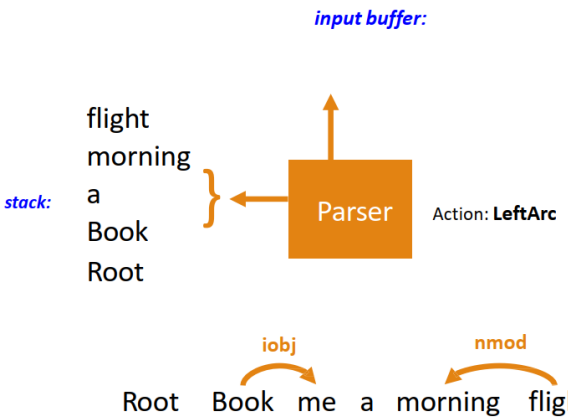
## 9.4 Transition Based 依存句法解析方法；（了解）

这个框架由状态和动作两个部分组成，其中状态用来记录不完整的预测结果，动作则用来控制状态之间的转移。可以看做是state machine

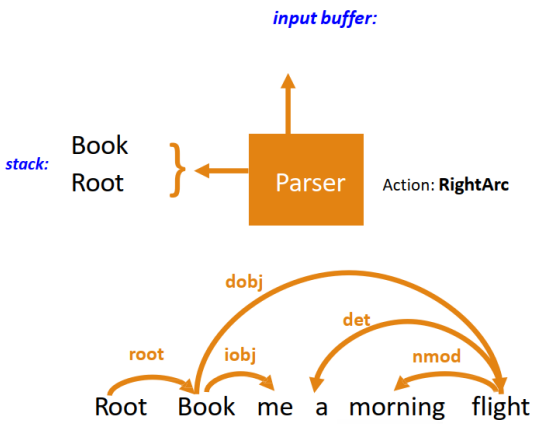
从空状态开始，通过动作转移到下一个状态，逐步生成依存句法树，最后的状态保存了一个完整的依存树。

定义了4种动作

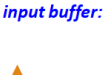
- 移进(shift)：队列首元素出栈，压入栈中成为s0
- 左规约(leftarc)：栈顶两个元素s1,s0规约，s1下沉成为s0的左孩子关系

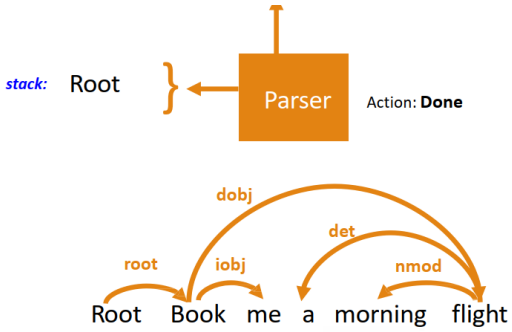



- 右规约(rightarc)：栈顶两个元素s1,s0规约，s0下沉成为s1的右孩子关系




- 根出栈(pop_root)：根节点出栈，分析完成






# Lecture 10: Statistical Machine Translation

## 10.1 机器翻译 Machine Translation（ MT ）的概念；

机器翻译 是一类将某种语言（源语言,source language）的句子 x 翻译成另一种语言（目标语言，target language）y的句子的任务。

## 10.2 统计机器翻译 Statistical Machine Translation （SMT） 的原理；

> 从数据中统计出概率模型。 我们可以将任务描述为给定源句子 $x$ ，找到最有可能是 $x$ 的翻译的目标句子 $y$ ，即
>
> $$\arg\max_y P(y|x) = \arg\max_y P(x|y)P(y)$$
>
> 用贝叶斯公式将其分为两部分，其中 $P(x|y)$ 可以看作翻译模型，保证翻译的正确性； $P(y)$ 则是语言模型，保证翻译结果语言流畅。

P(x|y)翻译模型：

- 分析单词和短语应该如何翻译(逼真)
- 从并行语料库学习 (例如成对的人工翻译的法语/英语句子)

P(y) 语言模型:

- 保证翻译结果的流畅
- 从单语数据中学习

## 10.3 词对齐 word alignment 的概念和计算方法；

源语言词和目标语言词之间的映射

对齐是翻译句子中特定词语之间的对应关系。

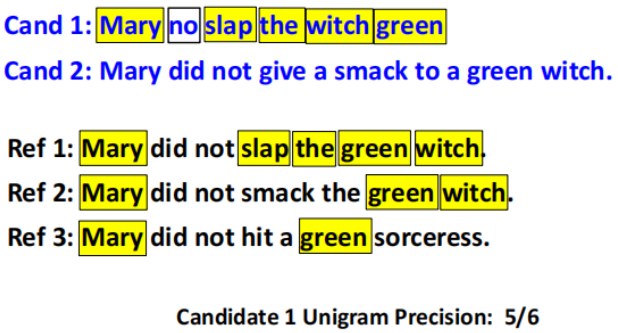
计算：

## 10.4 机器翻译评价指标 (BLEU) 的原理和方法；

主要思想：机器翻译越接近于专业的人类翻译，它就越好。根据一个数值度量来衡量它与一个或多个参考人类翻译的接近程度。一个给定的源句有许多“完美的”翻译。这些翻译可能在单词选择或单词顺序上有所不同，即使它们使用相同的单词。然而，人类可以清楚地区分好的翻译和坏的翻译。匹配项是与位置无关，匹配得越多，候选的翻译就越好

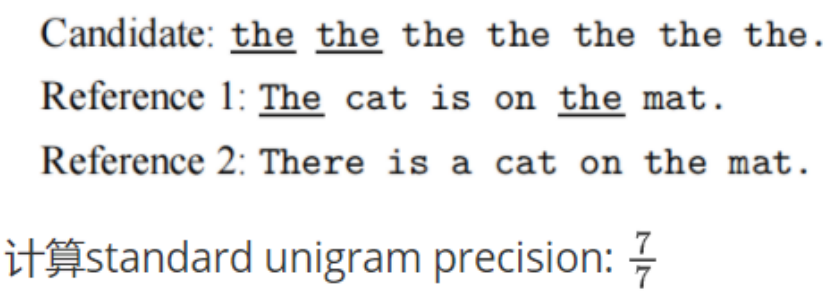
BLEU的主要任务是将候选翻译的n-gram和参考翻译的n-gram 对比，计算匹配的数量。

BLEU-1:它的总体思想就是准确率，假如给定标准译文reference(可能有多个)，神经网络生成的句子是candidate，句子总的单词数n，candidate中单词出现在reference中的个数为m，m/n就是计算公式。




BLEU-1衡量的是单词级别的准确性，更高阶的bleu可以衡量句子的流畅性。

MT系统可能会倾向于过渡生成"合理的单词"，导致不可能但是高准确的翻译。



所以在确定匹配的候选词后，参考词应该被认为耗尽。

对于某个词语出现的次数，在保证不大于candidate中出现的个数的情况下，然后再reference寻找词组出现的最多次，那么需要计算一个单词在任何单一参考翻译中出现的最大次数。

对于多个句子测试集的精度，评价的基本单位是句子。

计算每个句子的n-gram匹配数，然后累加clip的部分。

$$p_n = \frac{\sum_{C \in \{\text{Candidates}\}} \sum_{n-\text{gram} \in C} \text{Count}_{\text{clip}}(n\text{-gram})}{\sum_{C' \in \{\text{Candidates}\}} \sum_{n\text{-gram}' \in C'} \text{Count}(n\text{-gram}')}$$

## 10.5 IBM Model 1 （了解）

IBM Model 1 是最为基本的翻译模型，也是一个最简单的基于词汇的翻译模型(lexical translation model)

$$P(F \mid E) = \sum_A P(F, A \mid E) = \sum_A P(F \mid E, A) \times (A \mid E)$$

- 第一部分 $P(F|E,A)$ 的含义：给定一个Aligment A和英文句子 $E(e_1, e_2, \ldots, e_I)$ ,翻译成外语句子 $F(f_1, f_2, \ldots, f_J)$ 的概率:

$$P(F|E,A) = \prod_{j=1}^{J} t(f_j|e_{A_i})$$

其中 $t(f_j|e_{A_i})$ 表示在alignment A之下， $e_{A_i}$ 翻译到 $f_j$ 的概率，我们以前面图片中提到的句子为例，以此公式计算出的概率为:

$$P(F|E,A) = \prod_{j=1}^{J} t(Maria|Mary) \times t(no|didn't) \times t(dió|slap) \ldots t(bruja|witch) \times t(verde|green)$$

- 第二部分 $P(A|E)$ 的含义：给定长度(单词个数)为 $I$ 的英文句子E，将其翻译成长度为 $J$ 的西班牙语句F，那么对齐位置A的概率:

$$P(A|E) = \frac{\epsilon}{(I+1)^J}$$

IBM Model 1 采用最简单的假设：每个**alignment所发生的概率都相同。**

# Lecture 11 : Neural Machine Translation

🐱 这一章看自己的大作业就够啦！

有补充也可以。。。

## 11.1 循环神经网络 Recurrent Neural Networks, RNN 的概念、原理、优缺点；

**概念**

循环神经网络是一类以序列数据为输入，在序列的演进方向进行递归且所有节点（循环单元）按链式连接的递归神经网络。

**原理**

X 代表序列输入，比如一个句子。 W 一直不变，是每个时间点之间的权重矩阵。 RNN 之所以可以解决序列问题，是因为它可以记住每一时刻的信息，每一时刻的隐状态不仅由该时刻的输入层决定，还由上一时刻的隐状态决定。

**优点**

- 能够处理任意长度的输入
- 可以访问之前时刻的信息的信息
- 对于较长的输入，模型的大小保持不变
- 计算单元只有一个，参数共享

**缺点**

- 循环计算过程慢
- 难以访问距当前时刻较远的信息

## 11.2 如何训练循环神经网络 Recurrent Neural Networks, RNN

- 输入
- 计算每个时刻的输出的概率，计算每个时间步的交叉熵作为损失
- 计算所有时间步的损失的均值

$$J = -\log\left(p\left(\hat{y_1}\right)\right) - \log\left(p\left(\hat{y_2}\right)\right) - \ldots - \log\left(p\left(\hat{y_n}\right)\right) - \log(p([END]))$$
$$= -\frac{1}{T}\sum_{i}^{T}\log\left(p\left(\hat{y_i}\right)\right)$$

- 随机梯度下降更新参数

## 11.3 神经机器翻译 Neural Machine Translation, NMT 的原理；

NMT依赖于Sequence-to-Sequence的模型架构，即通过一个RNN作为encoder将输入的源语言转化为某表征空间中的向量，再通过另一个RNN作为decoder将其转化为目标语言中的句子。

我们可以将decoder看做预测目标句子y 的下一个单词的语言模型，同时其概率依赖于源句子的encoding 。

## 11.4 和统计机器翻译相比，神经机器翻译的优缺点；

优点：

1.端到端的神经网络，整个结构一起进行优化，需要较少人工的特征工程，不需要各种复杂的前后组件，更少的人工干预 。

2.翻译性能上更出色，更流畅，更好地利用上下文。

3.使用了深度学习的方法，可以引入很多语义特征，比如利用文本的相似度，利用文本内隐含的多层次特征，这些都是统计学方法没有的

4.对所有语言对使用相同的方法

缺点：

1. 翻译不忠实问题

以连续表示的方式进行词语表示，一方面给神经机器翻译带来了更好的泛化能力，另一方面也使得神经机器翻译容易产生不忠实的翻译。此处不忠实的翻译是指模型生成的目标语言词语能够保证目标语言语句的流利度，却无法准确地反映出源语言句子的语义信息。

2. 深度学习的黑箱本质

深度学习的不可解释性，难以调试，难以控制

3. 词语表规模受限问题

为了控制模型的时空开销，神经机器翻译通常在源语言端和目标语言端采用规模适当的词语表（词语表规模一般3万至5万）。对于词语表没有覆盖的词语，模型会将其替换为源语言端和目标语言端相应的UNK字符。UNK字符一方面影响模型完整捕获源语言句子语义信息，另一方面也影响用户理解模型所产生的目标语言句子。这一问题对词形丰富的语言（比如德语）显得尤为严重。

4. 源语言翻译覆盖问题

神经机器翻译在解码过程中通过注意力机制的自动调整，选择关注不同的源语言句子片段来产生对应的目标语言单词。由于缺少约束，注意力机制无法保证源语言句子中的词语被“恰到好处”地关注，从而导致“过翻译”“欠翻译”现象的产生。其中“过翻译”指不该多次翻译的源语言词语被多次翻译，“欠翻译”是指应该被翻译的源语言词语没有被翻译 。

## 11.5 神经机器翻译的训练方法；

- online learning： 每次计算一个数据更新参数

**Algorithm 1** A fully online training algorithm

```
procedure ONLINE
    for several epochs of training do
        for each training example in the data do
            Calculate gradients of the loss
            Update the parameters according to this gradient
        end for
    end for
end procedure
```

不需要在执行更新之前完全传递数据，所以online learning通常能更快地找到一个相对较好的解决方案，但是收敛性能不好。由于随机性，通常能从局部最优逃离，并找到更好的全局解决方案。

- batch learning： 它将整个数据集视为一个单元，计算梯度，然后只在完全传递数据后执行更新。

**Algorithm 2** A batch learning algorithm

```
procedure BATCH
    for several epochs of training do
        for each training example in the data do
            Calculate and accumulate gradients of the loss
        end for
        Update the parameters according to the accumulated gradient
    end for
end procedure
```

  在训练结束时，批处理学习算法可以更稳定，因为不会受到最近看到的训练例子的过度影响，但是算法也更容易陷入局部最优状态。

- mini-batch：小批量训练类似于在线训练，但不是一次处理单个训练例子，而是一次处理batch-size个训练例子的梯度。batch-size=1时，相当于online learning，batch-size为语料库大小时，相当于batch learning。

  现代硬件(特别是gpu，还有cpu)具有非常高效的向量处理指令，通过选取多个训练示例并将类似的操作分组在一起同时处理，可以实现计算效率的巨大提高。mini-batch另一个主要优点是，通过使用一些技巧，实际上可以使同时处理n个训练示例比单独处理n个不同的示例要快得多。

计算损失：

梯度下降：

同rnn训练...

## 11.6 Free running 和 teacher forcing 方法的原理和不同；

- **free-running**在训练的过程中，使用上一个状态输出的predicted word来作为下一个状态的输入。由于我们不能判定上一个状态输出的predicted word是否正确，所以该模型收敛速度较慢，模型不稳定。

  在RNN训练的早期，靠前的state中如果出现了极差的结果，那么后面的全部state都会受牵连，以至于最终结果非常不好也很难溯源到发生错误的源头。

- **teacher-forcing**在训练的过程中，则不是将上一个状态的输出结果作为下一个状态的输入结果，而是直接根据语料库使用标准答案ground truth word。测试的过程中，teacher-forcing与free-running使用同样的方法。由于每次经过RNN，都可以确定输入是正确的，所以teacher-forcing模型会显得更加稳定，收敛速度较快。

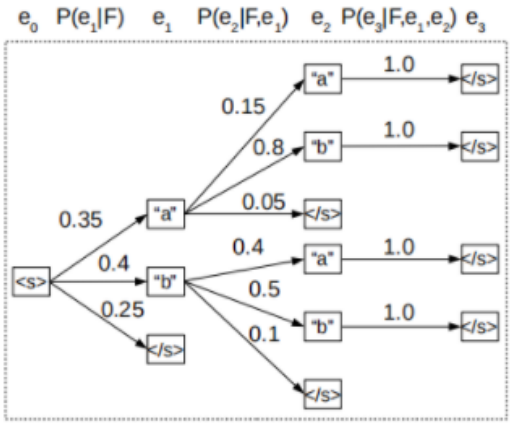
当然，teacher-forcing模式也是有缺点的，虽然在训练过程中，模型会有较好的效果，但是因为测试的时候不能有ground truth的支持，所以它会失去**自动纠错的能力**，默认前面的错误的词语是正确的，之后偏差可能会越来越大。模型会变得脆弱。

## 11.7 神经机器翻译的解码方法：贪婪和束搜索

**贪婪**：找到最大P(E|F)的E,

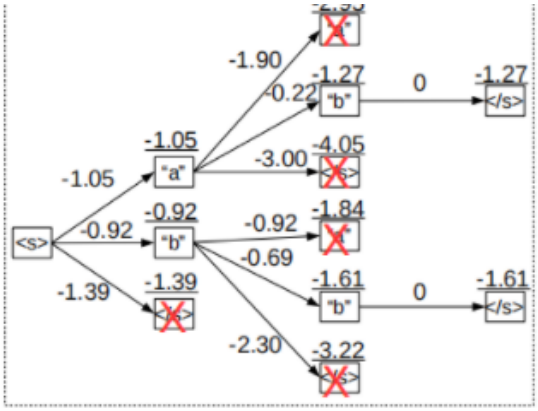
$$\hat{E} = \underset{E}{\operatorname{argmax}} P(E \mid F)$$

计算每个时间步的p_t, 选择概率最高的词语，然后这个词语作为输出序列的下一个词语。贪婪搜索不能保证能找到概率最高的翻译。




**束搜索** 根据P(E|F) 找到概率最高的n个输出

类似于贪婪搜索，但不只考虑一个最佳假设，而是在每个时间步长考虑b个最佳假设，其中b是束的“宽度”（beam-size）。




## 11.8 注意力机制

信息的瓶颈：将输入的所有信息都encode到encoder的最后一个hidden state上。Encoder-Decoder有一个很大的局限性就是 Encoder 和 Decoder 之间只通过一个固定长度的语义向量 C 来唯一联系。

两个弊端：

1. 语义向量 C 可能无法完全表示整个序列的信息
2. 先输入到网络的内容携带的信息会被后输入的信息覆盖掉，输入的序列越长，该现象就越严重

假设我们想要将“我爱你”翻译成"I love you"。在没有attention的情况下，“我爱你”中的每一个元素对于语义编码C的重要程度都是一样的。但实际上，“我”对于"I"起到的作用要比其他元素大得多。在未考虑注意力机制的Encoder-Decoder 模型中，这种不同输入的重要程度并没有体现处理，一般称为 **分心模型。注意力模型就是要从序列中学习到每一个元素的重要程度，然后按重要程度将元素合并。**

简单来说，Attention 机制就是对输入的每个元素考虑不同的权重参数，从而更加关注与输入的元素相似的部分，而抑制其它无用的信息。能够一步到位地考虑到全局联系与局部联系，且能够并行化计算。

- 注意力显著提高了NMT性能
  - 这是非常有用的，让解码器专注于某些部分的源语句
- 注意力解决瓶颈问题
  - 注意力允许解码器直接查看源语句；绕过瓶颈
- 注意力帮助消失梯度问题
  - 提供了通往遥远状态的捷径
- 注意力提供了一些可解释性
  - 通过检查注意力的分布，我们可以看到解码器在关注什么
  - 我们可以免费得到(软)对齐
  - 这很酷，因为我们从来没有明确训练过对齐系统
  - 网络只是自主学习了对齐

**attention**： 允许模型在生成翻译时关注输入句子的不同部分的方法。一种更有效和更直观的方法来表示句子，通常比encoder-decoder更有效。所谓注意力，是指当解码器在生成单个目标语言词时，仅有小部分的源语言词是相关的，绝大多数源语言词都是无关的。

## 11.9 神经机器翻译的实现（了解）

- 神经网络架构称为sequence-to-sequence (又名seq2seq)，它包含两个RNNs
- 一个 encoder 编码器，将模型的输入序列作为输入，然后编码固定大小的“上下文向量”
- 一个 decoder 解码器，使用来自编码器生成的上下文向量作为从其生成输出序列的“种子”。
-