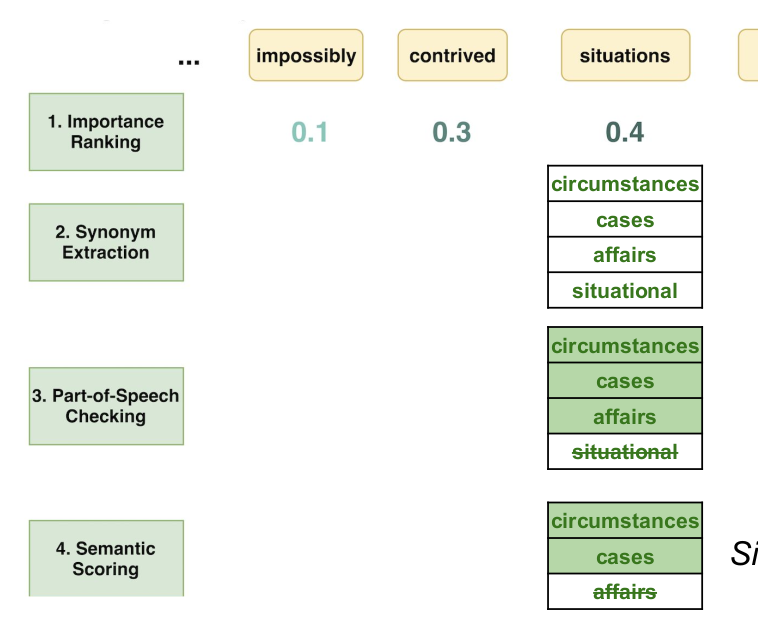
2020.2.14 接着2020.1.13的GAN生成文本和文本分类的学习内容

这个实验采用的是攻击自然语言模型预训练模型BERT使其具有更精确

在这个实验中在先对这个输出的文字做排序，在这个实验中一共有４步，第一步：然后取最重要的那个单词，在上面的例子中最重要的是circumstances,第二步：然后从counter fitting中选取同义词case, affairs, situational, 第３步：然后再检测语句的通顺性;第4步：语法打分。

在对抗的时候用同义词进行输入后，判别器输出的标签应该跟原先相反，但是词语本身的相似读却要跟原来的相似，通过这样的原理可以用来做文本的生成;

评价标准有４种，人工的和自动评估，在黑盒的基础上评价这个自动评估模型是不是准确率比以前高;　人工评价的方法运用在这个论文中的方法对现在的工作不现实，他们找的是标注师来评价。

下面是自然语言的学习笔记：

１.语言模型的介绍

2. Chain Rule以及马尔科夫假设

3.Unigram,Bigram,Ngram

4.估计语言模型的概率

5.评估语言模型模型

6.Add-one平滑，Add-k平滑

7.Interpolation 平滑

**１.语言模型**：

语言模型用来判断：是否一句话从语法上通顺

P(s) = P(w1,w2,w3,w4,w5...wn)

方式：给定一个句子，计算该句子在当前LM下的概率。重点是LM的训练。

### **2.1Chain Rule**

P(A,B,C,D) =P(A)\* P(B|A)\* P(C|A,B)\*P(D|A,B,C) ——chain rule

　　　把包含4个随机变量的联合分布，拆分成3个条件分布和1个P(A)

　　　P(A,B) = P(A|B)\* P(B) = P(B|A)\*P(A)

P(w1,w2,w3,w4,w5...wn) = P(w1)\* P(w2|w1)\* P(w3|w1w2)\* P(w4|w1,w2,w3)\* P(w5|w1,w2,w3,w4)...P(wn|w1,w2...wn-1) ——chain rule

**2.2.马尔科夫假设**

first order markov assumption

　　　p(w1,w2,w3,w4...wn) = p(w1)\* p(w2|w1)\* p(w3|w2)...p(Wn|Wn-1)

secod order

　　　p(w1,w2,w3,w4...wn) = p(w1)\* p(w2|w1)\* p(w3|w2，w1)...p(Wn|Wn-2,Wn-1)

## **3 Unigram, Bigram, N-gram**

### Language Model:Unigram

### P(w1,w2,w3,w4...Wn) = P(w1) \* P(w2)\* P(w3)...P(Wn)

### Language Model:Bigram

first order markov assumption

　　p(w1,w2,w3,w4...wn) = p(w1)\* p(w2|w1)\* p(w3|w2，w1)...p(Wn|Wn-2,Wn-1)

4**.估计语言模型的概率**

### Unigram:Estimating Probability

P(Wn) = 词汇Wn出现次数 / 语料库所有词汇数量V

### Bigram:Estimating Probability

p(w1,w2,w3,w4...wn) = p(w1)\* p(w2|w1)\* p(w3|w2，w1)...p(Wn|Wn-2,Wn-1)

* P(Wn/Wn-2,Wn-1) = 出现Wn的同时也出现词汇Wn-1,Wn-2的次数 / 词汇Wn出现次数

**5.评估语言模型**

**Perplexity = 2^(-x) ,**x : average log likelihood

将训练好的 LM 放入 语料库，计算 likelihood。  
likelihood 的越大，x越大，2^(-x) 越小，Perplexity越小，越小越好！！  
(可以画图像更生动展示变化曲线)

**Perplexity 尤其适用 unsupervised 文本的情况**

#### **Perplexity计算**

Perplexity = 2^(-x) ,  
x : average log likelihood （平均 对数 似然）  
计算X 的过程:

* likelihood : 计算P(Wn)
* log likelihood: 计算logP(Wn)
* average log likelihood: x = (logP(W1)+...+logP(Wn))/n

最后将x带入Perplexity公式。

一般情况下，N-gram LM 效果更好

## 6.Smoothing（平滑处理）

### Add-one Smoothing 方法（Laplace Smoothing 方法）

若不考虑Smoothing,通常计算的是PMLE,

* PMLE(Wi | Wi-1) = c((Wi , Wi-1)) / c(Wi )

但是此方法可能出现P(Wn)为0的情况而导致无法评估，因此引入Smoothing方法。

* PAdd-1(Wi | Wi-1) = c((Wi , Wi-1)) + 1 / c(Wi ) + V
* V是词典的大小（无重复词）
* 在分母中加V，会使得所有P()的总和为1

### Add-K Smoothing(Laplace Smoothing)

* PAdd-1(Wi | Wi-1) = c((Wi , Wi-1)) + k / c(Wi ) + kV
* k=1时， 是Add-one Smoothing

#### 如何选择K

1. K = 1，2，3...n ,一个一个计算对比找到最合适的 k
2. 优化 f(k) ,此时Perplexity = f(k),故 Minperplexity= Minf(k) => k= argmink f(k)

### Interpolation 方法

背景：现在不出现的概率，不代表未来数据更多anyway还不出现

* 核心思路：在计算Trigram概率时同时Unigram,Bigram,Trigram出现的频次
* P(Wn|Wn-1,Wn-2) = λ1P(Wn|Wn-1,Wn-2)+ λ2P(Wn|Wn-1)+ λ3P(Wn) , λ1+λ2+λ3 =1

上面公式 <=> Tri-gram = λ1Trigram+ λ2 Bigram+ λ3Unigram

对各方法加权平均，好的方法权重大些，加权平均的和为1

### Good-Turning Smoothing

背景：在训练数据里没有见过的怎么处理？

* Nc : 出现c次的单词的个数