一、AI算法基础
1、样本不平衡的解决方法？
2、交叉熵函数系列问题？与最大似然函数的关系和区别？
3、HMM、MEMM vs CRF 对比？
4、SVM和LR的区别与联系？
5、crf的损失函数是什么？lstm+crf怎么理解？
6、GBDT vs Xgboost
7、评估指标f1和auc的区别是哪些?
8、sigmoid用作激活函数时，分类为什么要用交叉熵损失，而不用均方损失？
9、神经网络中的激活函数的对比？
二、NLP高频问题
1、word2vec和tf-idf 相似度计算时的区别？
2、word2vec和NNLM对比有什么区别？（word2vec vs NNLM）
3、 word2vec负采样有什么作用？
4、word2vec和fastText对比有什么区别？（word2vec vs fastText）
5、glove和word2vec、 LSA对比有什么区别？（word2vec vs glove vs LSA）
6、 elmo、GPT、bert三者之间有什么区别？（elmo vs GPT vs bert）
7、LSTM和GRU的区别？
三、其他算法问题
1、怎么进行单个样本的学习？
2、 决策树 bagging boosting adaboost 区别？RF的特征随机目的是什么？
3、transformer各部分怎么用？Q K V怎么计算；Attention怎么用？
4、HMM 假设是什么？CRF解决了什么问题？CRF做过特征工程吗？HMM中的矩阵意义？5、说以一下空洞卷积？膨胀卷积怎么理解？什么是Piece-CNN？
6、怎么解决beam-search局部最优问题？global embedding 怎么做？
7、数学题：什么是半正定矩阵？机器学习中有什么应用？
8、卷积的物理意义是什么？傅里叶变换懂吗？
9、说一下Bert？
10、推导word2vec？
11、怎么理解传统的统计语言模型？现在的神经网络语言模型有什么不同？
12、神经网络优化的难点是什么？这个问题要展开来谈。
13、attention你知道哪些？
14、自动文章摘要抽取时，怎么对一篇文章进行分割？（从序列标注、无监督等角度思考）
15、在做NER任务时，lstm后面可以不用加CRF吗？
16、通过画图描述TextRank？
17、LDA和pLSA有什么区别？
18、Transformer在实际应用中都会有哪些做法？
19、讲出过拟合的解决方案？
20、说一下transforemr、LSTM、CNN间的区别？从多个角度进行讲解？
21、梯度消失的原因和解决办法有哪些？
22、数学题：贝叶斯计算概率？
23、数学题：25只兔子赛跑问题，共5个赛道，最少几次比赛可以选出前5名？
24、数学题：100盏灯问题？

一、AI算法基础

1、样本不平衡的解决方法？

1）上采样和子采样；2）修改权重（修改损失函数）；3）集成方法：bagging，类似随机森林、自助采样；4）多任务联合学习；

2、交叉熵函数系列问题？与最大似然函数的关系和区别？

1）交叉熵损失函数的物理意义：用于描述模型预测值与真实值的差距大小；

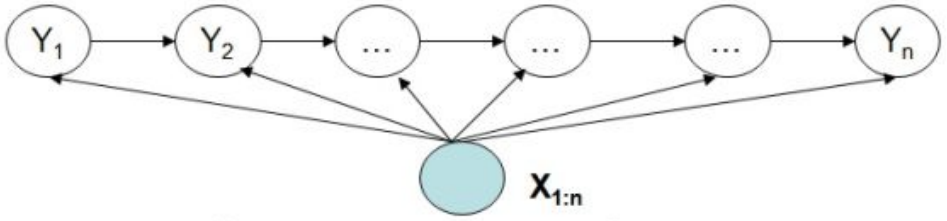
2）最小化交叉熵的本质就是对数似然函数的最大化；

3）对数似然函数的本质就是衡量在某个参数下，整体的估计和真实情况一样的概率，越大代表越相近；而损失函数的本质就是衡量预测值和真实值之间的差距，越大代表越不相近。

3、HMM、MEMM vs CRF 对比？

1）HMM是有向图模型，是生成模型；HMM有两个假设：一阶马尔科夫假设和观测独立性假设；但对于序列标注问题不仅和单个词相关，而且和观察序列的长度，单词的上下文，等等相关。

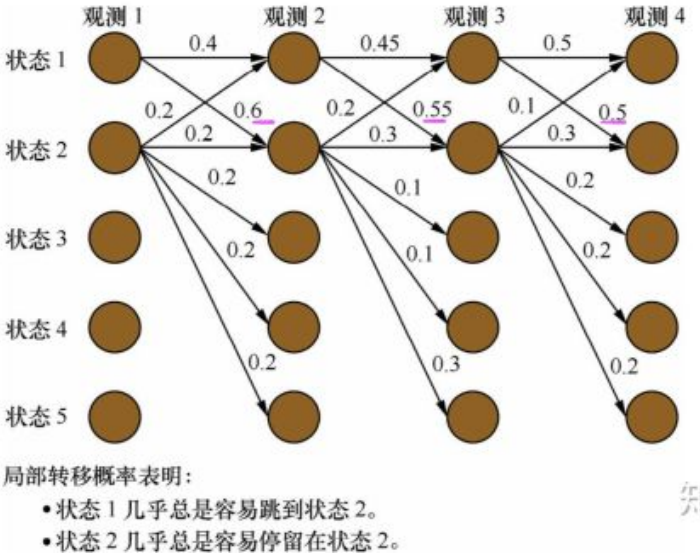
2）MEMM（最大熵马尔科夫模型）是有向图模型，是判别模型；MEMM打破了HMM的观测独立性假设，MEMM考虑到相邻状态之间依赖关系，且考虑整个观察序列，因此MEMM的表达能力更强；但MEMM会带来标注偏置问题：由于局部归一化问题，MEMM倾向于选择拥有更少转移的状态。这就是标记偏置问题。



$$P(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}) = \prod_{i=1}^{n} P(y_i|y_{i-1},\mathbf{x}_{1:n}) = \prod_{i=1}^{n} \frac{\exp(\mathbf{w}^T\mathbf{f}(y_i,y_{i-1},\mathbf{x}_{1:n}))}{Z(y_{i-1},\mathbf{x}_{1:n})}$$

最大熵模型（MEMM）

最大熵马尔可夫模型存在标注偏置问题，如图6.7所示。可以发现，状态1倾向于转移到状态2，状态2倾向于转移到状态2本身。但是实际计算得到的最大概率路径是1->1->1->1，状态1并没有转移到状态2，如图6.8所示。这是因为，从状态2转移出去可能的状态包括1、2、3、4、5，概率在可能的状态上分散了，而状态1转移出去的可能状态仅仅为状态1和2，概率更加集中。由于局部归一化的影响，隐状态会倾向于转移到那些后续状态可能更少的状态上，以提高整体的后验概率。这就是标注偏置问题。



3）CRF模型解决了标注偏置问题，去除了HMM中两个不合理的假设，当然，模型相应得也变复杂了。

HMM、MEMM和CRF的优缺点比较：

a）与HMM比较。CRF没有HMM那样严格的独立性假设条件，因而可以容纳任意的上下文信息。特征设计灵活（与ME一样）

b）与MEMM比较。由于CRF计算全局最优输出节点的条件概率，它还克服了最大熵马尔可夫模型标记偏置（Label-bias）的缺点。

c）与ME比较。CRF是在给定需要标记的观察序列的条件下，计算整个标记序列的联合概率分布，而不是在给定当前状态条件下，定义下一个状态的状态分布.

> 首先，CRF，HMM(隐马模型)，MEMM(最大熵隐马模型)都常用来做序列标注的建模，像分词、词性标注，以及命名实体标注
> 隐马模型一个最大的缺点就是由于其输出独立性假设，导致其不能考虑上下文的特征，限制了特征的选择
> 最大熵隐马模型则解决了隐马的问题，可以任意选择特征，但由于其在每一节点都要进行归一化，所以只能找到局部的最优值，同时也带来了标记偏见的问题，即凡是训练语料中未出现的情况全都忽略掉。
> 条件随机场则很好的解决了这一问题，他并不在每一个节点进行归一化，而是所有特征进行全局归一化，因此可以求得全局的最优值。

4、SVM和LR的区别与联系？

对非线性表达上，逻辑回归只能通过人工的特征组合来实现，而SVM可以很容易引入非线性核函数来实现非线性表达，淡然也可以通过特征组合。
逻辑回归产出的是概率值，而SVM只能产出是正类还是负类，不能产出概率。
逻辑回归的损失函数是log loss，而SVM使用的是hinge loss。
SVM主要关注的是“支持向量”，也就是和分类最相关的少数点，即关注局部关键信息；而逻辑回归是在全局进行优化的。这导致SVM天然比逻辑回归有更好的泛化能力，防止过拟合。
损失函数的优化方法不同，逻辑回归是使用梯度下降来求解对数似然函数的最优解；SVM使用SMO方法，来求解条件约束损失函数的对偶形式。
处理的数据规模不同。LR一般用来处理大规模的学习问题。如十亿级别的样本，亿级别的特征。（SVM是二次规划问题，需要计算m阶矩阵）
svm 更多的属于非参数模型，而logistic regression 是参数模型，本质不同。其区别就可以参考参数模型和非参模型的区别。

我们先来看一下SVM 和正则化的逻辑回归它们的**损失函数**：

$$\text{SVM:}\frac{1}{n}\sum_{i=1}^{n}(1-y_i[w_0+\mathbf{x}_i^T\mathbf{w}_1])^+ + \lambda\|\mathbf{w}_1\|/2 \qquad (1)$$

$$\text{Logistic:}\frac{1}{n}\sum_{i=1}^{n}\overbrace{-\log g(y_i[w_0+\mathbf{x}_i^T\mathbf{w}_1])}^{-\log P(y_i|\mathbf{x},\mathbf{W})} + \lambda\|\mathbf{w}_1\|/2 \qquad (2)$$

其中，$g(z)=(1+\exp(-z))^{-1}$.

5、crf的损失函数是什么？lstm+crf怎么理解？

$$s(\mathbf{X},\mathbf{y})=\sum_{i=0}^{n}A_{y_i,y_{i+1}}+\sum_{i=1}^{n}P_{i,y_i}$$

下面引出损失函数（虽然我感觉这不应该称为“损失”），对真实标记序列y的概率取log：

$$\log(p(\mathbf{y}|\mathbf{X}))=s(\mathbf{X},\mathbf{y})-\log\left(\sum_{\tilde{\mathbf{y}}\in\mathbf{Y}_\mathbf{X}}e^{s(\mathbf{X},\tilde{\mathbf{y}})}\right)$$

那么我们的目标就是最大化上式（即真实标记应该对应最大概率值），因为叫损失函数，所以我们也可以对上式取负然后最小化之，这样我们就可以使用梯度下降等优化方法来求解参数。在这个过程中，我们要最大化真实标记序列的概率，也就训练了转移概率矩阵A和BiLSTM中的参数。

6、GBDT vs Xgboost

yuyuqi：ID3、C4.5、CART、随机森林、bagging、boosting、Adaboost
zhuanlan.zhihu.com

那么如何得到优秀的组合树呢？

一种办法是贪心算法，遍历一个节点内的所有特征，按照公式计算出按照每一个特征分割的信息增益，找到信息增益最大的点进行树的分割。增加的新叶子惩罚项对应了树的剪枝，当gain小于某个阈值的时候，我们可以剪掉这个分割。但是这种办法不适用于数据量大的时候，因此，我们需要运用近似算法。

另一种方法：XGBoost在寻找splitpoint的时候，不会枚举所有的特征值，而会对特征值进行聚合统计，按照**特征值的密度分布**，构造直方图计算特征值分布的面积，然后划分分布形成若干个bucket(桶)，每个bucket的面积相同，将**bucket边界上的特征值**作为split point的候选，**遍历所有的候选分裂点**来找到最佳分裂点。

上图近似算法公式的解释：将特征k的特征值进行排序，计算特征值分布，rk（z）表示的是对于特征k而言，其特征值小于z的权重之和占总权重的比例，代表了这些特征值的重要程度，我们按照这个比例计算公式，将特征值分成若干个bucket，每个bucket的比例相同，选取这几类特征值的边界作为划分候选点，构成候选集；选择候选集的条件是要使得相邻的两个候选分裂节点差值小于某个阈值

传统GBDT以CART作为基分类器，xgboost还支持线性分类器，这个时候xgboost相当于带L1和L2正则化项的逻辑斯蒂回归（分类问题）或者线性回归（回归问题）。

传统GBDT在优化时只用到一阶导数信息，xgboost则对代价函数进行了**二阶泰勒展开，同时用到了一阶和二阶导数。**顺便提一下，xgboost工具支持自定义代价函数，只要函数可一阶和二阶求导。
xgboost在**代价函数里加入了正则项，用于控制模型的复杂度。**正则项里包含了树的叶子节点个数、每个叶子节点上输出的score的L2模的平方和。从Bias-variance tradeoff角度来讲，正则项降低了模型的variance，使学习出来的模型更加简单，防止过拟合，这也是xgboost优于传统GBDT的一个特性。
Shrinkage（缩减），相当于学习速率（xgboost中的eta）。每次迭代，增加新的模型，在前面成上一个小于1的系数，降低优化的速度，每次走一小步逐步逼近最优模型比每次走一大步逼近更加容易避免过拟合现象；
列抽样（column subsampling）。xgboost借鉴了随机森林的做法，支持列抽样（即每次的输入特征不是全部特征），不仅能降低过拟合，还能减少计算，这也是xgboost异于传统gbdt的一个特性。
忽略缺失值：在寻找splitpoint的时候，不会对该特征为missing的样本进行遍历统计，只对该列特征值为non-missing的样本上对应的特征值进行遍历，通过这个工程技巧来减少了为稀疏离散特征寻找splitpoint的时间开销
指定缺失值的分隔方向：可以为缺失值或者指定的值指定分支的默认方向，为了保证完备性，会分别处理将missing该特征值的样本分配到左叶子结点和右叶子结点的两种情形，分到那个子节点带来的增益大，默认的方向就是哪个子节点，这能大大提升算法的效率。
并行化处理：在训练之前，预先对每个特征内部进行了排序找出候选切割点，然后保存为block结构，后面的迭代中重复地使用这个结构，大大减小计算量。在进行节点的分裂时，需要计算每个特征的增益，最终选增益最大的那个特征去做分裂，那么各个特征的增益计算就可以开多线程进行，即在不同的特征属性上采用多线程并行方式寻找最佳分割点。

二、NLP高频问题

1、word2vec和tf-idf 相似度计算时的区别？

word2vec 1、稠密的 低维度的 2、表达出相似度； 3、表达能力强；4、泛化能力强；

2、word2vec和NNLM对比有什么区别？（word2vec vs NNLM）

1）其本质都可以看作是语言模型；

2）词向量只不过NNLM一个产物，word2vec虽然其本质也是语言模型，但是其专注于词向量本身，因此做了许多优化来提高计算效率：

与NNLM相比，词向量直接sum，不再拼接，并舍弃隐层；
考虑到sofmax归一化需要遍历整个词汇表，采用hierarchical softmax 和negative sampling进行优化，hierarchical softmax 实质上生成一颗**带权路径最小的哈夫曼**树，让高频词搜索路劲变小；
negative sampling更为直接，实质上**对每一个样本中每一个词都进行负例采样**；

3、 word2vec负采样有什么作用？

负采样这个点引入word2vec非常巧妙，两个作用，1.加速了模型计算，2.保证了模型训练的效果，一个是模型每次只需要更新采样的词的权重，不用更新所有的权重，那样会很慢，第二，中心词其实只跟它周围的词有关系，位置离着很远的词没有关系，也没必要同时训练更新，作者这点非常聪明。

4、word2vec和fastText对比有什么区别？（word2vec vs fastText）

1）都可以无监督学习词向量， fastText训练词向量时会考虑subword；

2）fastText还可以进行有监督学习进行文本分类，其主要特点：

结构与CBOW类似，但学习目标是人工标注的分类结果；
采用hierarchical softmax对输出的分类标签建立哈夫曼树，样本中标签多的类别被分配短的搜寻路径；
引入N-gram，考虑词序特征；
引入subword来处理长词，处理未登陆词问题；

5、glove和word2vec、 LSA对比有什么区别？（word2vec vs glove vs LSA）

1）**glove vs LSA**

LSA（Latent Semantic Analysis）可以基于co-occurance matrix构建词向量，实质上是基于全局语料采用SVD进行矩阵分解，然而SVD计算复杂度高；
glove可看作是对LSA一种优化的高效矩阵分解算法，采用Adagrad对最小平方损失进行优化；

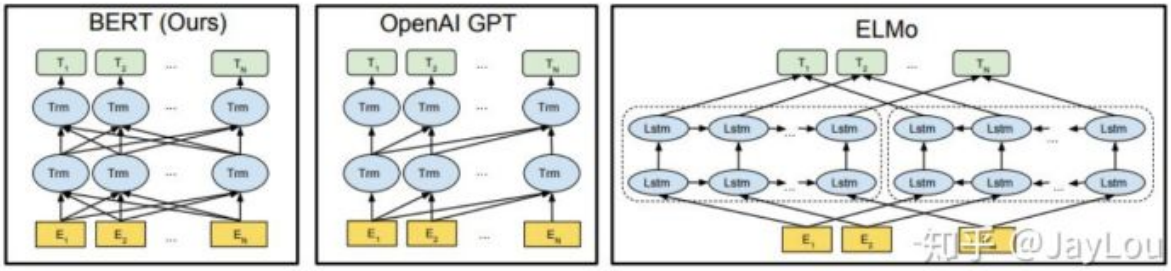
2）word2vec vs LSA

主题模型和词嵌入两类方法最大的不同在于模型本身。

主题模型是一种基于概率图模型的生成式模型。其似然函数可以写为若干条件概率连乘的形式，其中包含需要推测的隐含变量(即主题)
词嵌入模型一般表示为神经网络的形式，似然函数定义在网络的输出之上。需要学习网络的权重来得到单词的稠密向量表示。

3）**word2vec vs glove**

word2vec是局部语料库训练的，其特征提取是基于滑窗的；而glove的滑窗是为了构建co-occurance matrix，是基于全局语料的，可见glove需要事先统计共现概率；因此，word2vec可以进行在线学习，glove则需要统计固定语料信息。
word2vec是无监督学习，同样由于不需要人工标注；glove通常被认为是无监督学习，但实际上glove还是有label的，即共现次数 $log(X_{ij})$ 。
word2vec损失函数实质上是带权重的交叉熵，权重固定；glove的损失函数是最小平方损失函数，权重可以做映射变换。
总体来看，**glove可以被看作是更换了目标函数和权重函数的全局word2vec。**



elmo vs GPT vs bert

6、 elmo、GPT、bert三者之间有什么区别？（elmo vs GPT vs bert）
之前介绍词向量均是静态的词向量，无法解决一次多义等问题。下面介绍三种elmo、GPT、bert词向量，它们都是基于语言模型的动态词向量。下面从几个方面对这三者进行对比：

（1）**特征提取器**：elmo采用LSTM进行提取，GPT和bert则采用Transformer进行提取。很多任务表明Transformer特征提取能力强于LSTM，elmo采用1层静态向量+2层LSTM，多层提取能力有限，而GPT和bert中的Transformer可采用多层，并行计算能力强。

（2）**单/双向语言模型**：

GPT采用单向语言模型，elmo和bert采用双向语言模型。但是elmo实际上是两个单向语言模型（方向相反）的拼接，这种融合特征的能力比bert一体化融合特征方式弱。
GPT和bert都采用Transformer，Transformer是encoder-decoder结构，GPT的单向语言模型采用decoder部分，decoder的部分见到的都是不完整的句子；bert的双向语言模型则采用encoder部分，采用了完整句子。

7、LSTM和GRU的区别？

GRU和LSTM的性能在很多任务上不分伯仲。
GRU 参数更少因此更容易收敛，但是数据集很大的情况下，LSTM表达性能更好。
从结构上来说，GRU只有两个门（update和reset），LSTM有三个门（forget，input，

output），GRU直接将hidden state 传给下一个单元，而LSTM则用memory cell 把hidden state 包装起来。