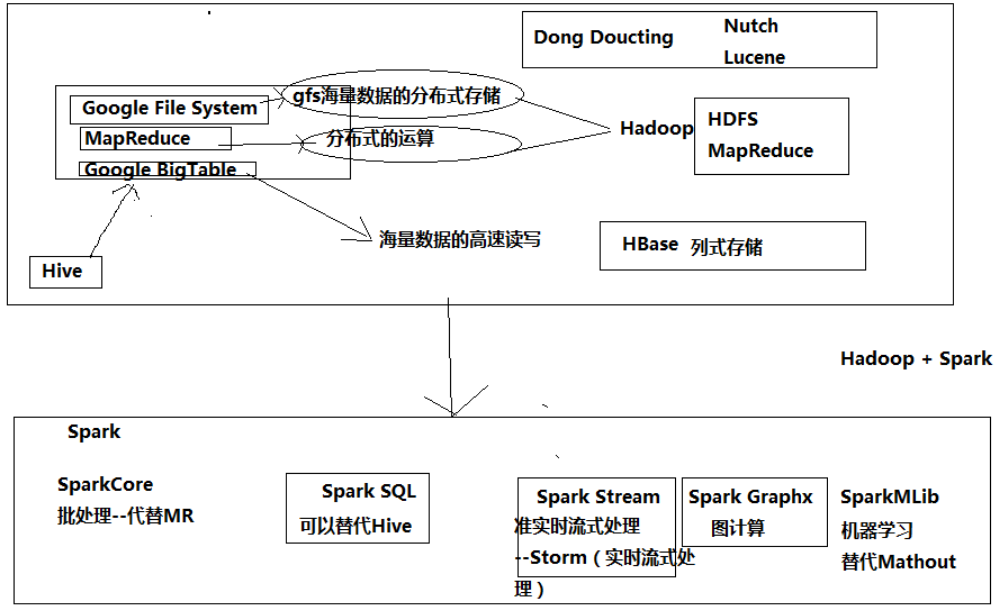
1、自我介绍。  
2、直接简历上项目。  
（1）项目的数据量多大，多少维，数据都是怎么预处理的。  
（2）你应用过哪些数据挖掘算法，针对简历上的算法或者模型，你看过源码或者模型中细节你知道多少，还是仅仅调用API用用而已  
(3）项目中，你遇到过的最大的困难时什么，怎么解决的，从中学到什么。  
（4）项目中hadoop搜索引擎你是怎么设计的，其中的分词是什么。  
（5）数据是存在HDFS中还是Redis中的。  
（6）spark和Hadoop的基本架构，尽量说。



Hadoop是适合于海量数据的分布式存储和分布式计算的框架两大核心：

HDFS(Hadoop Distributed File Sytem hadoop分布式文件系统)：分布式存储

MapReduce:用于分布式计算（批处理 离线计算）

分布式存储：

把一个整体文件拆分成多个模块，存储在不同的主机上，通过对不同主机的上文件块（block）在内存中的映射来对这些文件进行管理。

分布式计算：

把一个比较复杂的计算任务，分解成许多个简单的小的计算任务，在不同的机器上运行，得到结果，最后将这些机器上的结果进行汇总，得出最终结果。

**大数据处理框架是什么？**

　　处理框架和处理引擎负责对数据系统中的数据进行计算。虽然“引擎”和“框架”之间的区别没有什么权威的定义，但大部分时候可以将前者定义为实际负责处理数据操作的组件，后者则可定义为承担类似作用的一系列组件。

　　例如Apache Hadoop可以看作一种以MapReduce作为默认处理引擎的处理框架。引擎和框架通常可以相互替换或同时使用。例如另一个框架Apache Spark可以纳入Hadoop并取代MapReduce。组件之间的这种互操作性是大数据系统灵活性如此之高的原因之一。

　　虽然负责处理生命周期内这一阶段数据的系统通常都很复杂，但从广义层面来看它们的目标是非常一致的：通过对数据执行操作提高理解能力，揭示出数据蕴含的模式，并针对复杂互动获得见解。

　　为了简化这些组件的讨论，我们会通过不同处理框架的设计意图，按照所处理的数据状态对其进行分类。一些系统可以用批处理方式处理数据，一些系统可以用流方式处理连续不断流入系统的数据。此外还有一些系统可以同时处理这两类数据。

　　在深入介绍不同实现的指标和结论之前，首先需要对不同处理类型的概念进行一个简单的介绍。

**批处理系统**

　　批处理在大数据世界有着悠久的历史。批处理主要操作大容量静态数据集，并在计算过程完成后返回结果。

　　批处理模式中使用的数据集通常符合下列特征...

* 有界：批处理数据集代表数据的有限集合
* 持久：数据通常始终存储在某种类型的持久存储位置中
* 大量：批处理操作通常是处理极为海量数据集的唯一方法

　　批处理非常适合需要访问全套记录才能完成的计算工作。例如在计算总数和平均数时，必须将数据集作为一个整体加以处理，而不能将其视作多条记录的集合。这些操作要求在计算进行过程中数据维持自己的状态。

　　需要处理大量数据的任务通常最适合用批处理操作进行处理。无论直接从持久存储设备处理数据集，或首先将数据集载入内存，批处理系统在设计过程中就充分考虑了数据的量，可提供充足的处理资源。由于批处理在应对大量持久数据方面的表现极为出色，因此经常被用于对历史数据进行分析。

　　大量数据的处理需要付出大量时间，因此批处理不适合对处理时间要求较高的场合。

**Apache Hadoop**

　　Apache Hadoop是一种专用于批处理的处理框架。Hadoop是首个在开源社区获得极大关注的大数据框架。基于谷歌有关海量数据处理所发表的多篇论文与经验的Hadoop重新实现了相关算法和组件堆栈，让大规模批处理技术变得更易用。

　　新版Hadoop包含多个组件，即多个层，通过配合使用可处理批数据：

* HDFS：HDFS是一种分布式文件系统层，可对集群节点间的存储和复制进行协调。HDFS确保了无法避免的节点故障发生后数据依然可用，可将其用作数据来源，可用于存储中间态的处理结果，并可存储计算的最终结果。
* YARN：YARN是Yet Another Resource Negotiator（另一个资源管理器）的缩写，可充当Hadoop堆栈的集群协调组件。该组件负责协调并管理底层资源和调度作业的运行。通过充当集群资源的接口，YARN使得用户能在Hadoop集群中使用比以往的迭代方式运行更多类型的工作负载。
* MapReduce：MapReduce是Hadoop的原生批处理引擎。

批处理模式

　　Hadoop的处理功能来自MapReduce引擎。MapReduce的处理技术符合使用键值对的map、shuffle、reduce算法要求。基本处理过程包括：

* 从HDFS文件系统读取数据集
* 将数据集拆分成小块并分配给所有可用节点
* 针对每个节点上的数据子集进行计算（计算的中间态结果会重新写入HDFS）
* 重新分配中间态结果并按照键进行分组
* 通过对每个节点计算的结果进行汇总和组合对每个键的值进行“Reducing”
* 将计算而来的最终结果重新写入 HDFS

优势和局限

　　由于这种方法严重依赖持久存储，每个任务需要多次执行读取和写入操作，因此速度相对较慢。但另一方面由于磁盘空间通常是服务器上最丰富的资源，这意味着MapReduce可以处理非常海量的数据集。同时也意味着相比其他类似技术，Hadoop的MapReduce通常可以在廉价硬件上运行，因为该技术并不需要将一切都存储在内存中。MapReduce具备极高的缩放潜力，生产环境中曾经出现过包含数万个节点的应用。

　　MapReduce的学习曲线较为陡峭，虽然Hadoop生态系统的其他周边技术可以大幅降低这一问题的影响，但通过Hadoop集群快速实现某些应用时依然需要注意这个问题。

　　围绕Hadoop已经形成了辽阔的生态系统，Hadoop集群本身也经常被用作其他软件的组成部件。很多其他处理框架和引擎通过与Hadoop集成也可以使用HDFS和YARN资源管理器。

　　总结

　　Apache Hadoop及其MapReduce处理引擎提供了一套久经考验的批处理模型，最适合处理对时间要求不高的非常大的规模数据集。通过非常低成本的组件即可搭建完整功能的Hadoop集群，使得这一廉价且高效的处理技术可以灵活应用在很多案例中。与其他框架和引擎的兼容与集成能力使得Hadoop可以成为使用不同技术的多种工作负载处理平台的底层基础。

**流处理系统**

　　流处理系统会对随时进入系统的数据进行计算。相比批处理模式，这是一种截然不同的处理方式。流处理方式无需针对整个数据集执行操作，而是对通过系统传输的每个数据项执行操作。

　　流处理中的数据集是“无边界”的，这就产生了几个重要的影响：

* 完整数据集只能代表截至目前已经进入到系统中的数据总量。
* 工作数据集也许更相关，在特定时间只能代表某个单一数据项。
* 处理工作是基于事件的，除非明确停止否则没有“尽头”。处理结果立刻可用，并会随着新数据的抵达继续更新。

　　流处理系统可以处理几乎无限量的数据，但同一时间只能处理一条（真正的流处理）或很少量（微批处理，Micro-batch Processing）数据，不同记录间只维持最少量的状态。虽然大部分系统提供了用于维持某些状态的方法，但流处理主要针对副作用更少，更加功能性的处理（Functional processing）进行优化。

　　功能性操作主要侧重于状态或副作用有限的离散步骤。针对同一个数据执行同一个操作会忽略其他因素产生相同的结果，此类处理非常适合流处理，因为不同项的状态通常是某些困难、限制，以及某些情况下不需要的结果的结合体。因此虽然某些类型的状态管理通常是可行的，但这些框架通常在不具备状态管理机制时更简单也更高效。

　　此类处理非常适合某些类型的工作负载。有近实时处理需求的任务很适合使用流处理模式。分析、服务器或应用程序错误日志，以及其他基于时间的衡量指标是最适合的类型，因为对这些领域的数据变化做出响应对于业务职能来说是极为关键的。流处理很适合用来处理必须对变动或峰值做出响应，并且关注一段时间内变化趋势的数据。

**混合处理系统：批处理和流处理**

　　一些处理框架可同时处理批处理和流处理工作负载。这些框架可以用相同或相关的组件和API处理两种类型的数据，借此让不同的处理需求得以简化。

　　如你所见，这一特性主要是由Spark和Flink实现的，下文将介绍这两种框架。实现这样的功能重点在于两种不同处理模式如何进行统一，以及要对固定和不固定数据集之间的关系进行何种假设。

　　虽然侧重于某一种处理类型的项目会更好地满足具体用例的要求，但混合框架意在提供一种数据处理的通用解决方案。这种框架不仅可以提供处理数据所需的方法，而且提供了自己的集成项、库、工具，可胜任图形分析、机器学习、交互式查询等多种任务。

**Apache Spark**

　　Apache Spark是一种包含流处理能力的下一代批处理框架。与Hadoop的MapReduce引擎基于各种相同原则开发而来的Spark主要侧重于通过完善的内存计算和处理优化机制加快批处理工作负载的运行速度。

　　Spark可作为独立集群部署（需要相应存储层的配合），或可与Hadoop集成并取代MapReduce引擎。

　　批处理模式

　　与MapReduce不同，Spark的数据处理工作全部在内存中进行，只在一开始将数据读入内存，以及将最终结果持久存储时需要与存储层交互。所有中间态的处理结果均存储在内存中。

　　虽然内存中处理方式可大幅改善性能，Spark在处理与磁盘有关的任务时速度也有很大提升，因为通过提前对整个任务集进行分析可以实现更完善的整体式优化。为此Spark可创建代表所需执行的全部操作，需要操作的数据，以及操作和数据之间关系的Directed Acyclic Graph（有向无环图），即DAG，借此处理器可以对任务进行更智能的协调。

　　为了实现内存中批计算，Spark会使用一种名为Resilient Distributed Dataset（弹性分布式数据集），即RDD的模型来处理数据。这是一种代表数据集，只位于内存中，永恒不变的结构。针对RDD执行的操作可生成新的RDD。每个RDD可通过世系（Lineage）回溯至父级RDD，并最终回溯至磁盘上的数据。Spark可通过RDD在无需将每个操作的结果写回磁盘的前提下实现容错。

　　流处理模式

　　流处理能力是由Spark Streaming实现的。Spark本身在设计上主要面向批处理工作负载，为了弥补引擎设计和流处理工作负载特征方面的差异，Spark实现了一种叫做微批（Micro-batch）\*的概念。在具体策略方面该技术可以将数据流视作一系列非常小的“批”，借此即可通过批处理引擎的原生语义进行处理。

　　Spark Streaming会以亚秒级增量对流进行缓冲，随后这些缓冲会作为小规模的固定数据集进行批处理。这种方式的实际效果非常好，但相比真正的流处理框架在性能方面依然存在不足。

　　优势和局限

　　使用Spark而非Hadoop MapReduce的主要原因是速度。在内存计算策略和先进的DAG调度等机制的帮助下，Spark可以用更快速度处理相同的数据集。

　　Spark的另一个重要优势在于多样性。该产品可作为独立集群部署，或与现有Hadoop集群集成。该产品可运行批处理和流处理，运行一个集群即可处理不同类型的任务。

　　除了引擎自身的能力外，围绕Spark还建立了包含各种库的生态系统，可为机器学习、交互式查询等任务提供更好的支持。相比MapReduce，Spark任务更是“众所周知”地易于编写，因此可大幅提高生产力。

　　为流处理系统采用批处理的方法，需要对进入系统的数据进行缓冲。缓冲机制使得该技术可以处理非常大量的传入数据，提高整体吞吐率，但等待缓冲区清空也会导致延迟增高。这意味着Spark Streaming可能不适合处理对延迟有较高要求的工作负载。

　　由于内存通常比磁盘空间更贵，因此相比基于磁盘的系统，Spark成本更高。然而处理速度的提升意味着可以更快速完成任务，在需要按照小时数为资源付费的环境中，这一特性通常可以抵消增加的成本。

　　Spark内存计算这一设计的另一个后果是，如果部署在共享的集群中可能会遇到资源不足的问题。相比Hadoop MapReduce，Spark的资源消耗更大，可能会对需要在同一时间使用集群的其他任务产生影响。从本质来看，Spark更不适合与Hadoop堆栈的其他组件共存一处。

　　总结

　　Spark是多样化工作负载处理任务的最佳选择。Spark批处理能力以更高内存占用为代价提供了无与伦比的速度优势。对于重视吞吐率而非延迟的工作负载，则比较适合使用Spark Streaming作为流处理解决方案。

**3、数据挖掘算法。**  
（1）逻辑回归和线性回归区别，逻辑回归应用场景，其中的核函数有什么作用

（2）什么是梯度下降，梯度下降优化，你知道哪些  
（3）神经网络中，你觉得NN最大的优势在哪里，其中的更新函数是什么   
4、Java  
（1）你知道JVM优化  
（2）简单介绍下多线程  
（3）谈谈你熟悉的设计模型  
（4）垃圾回收机制，有几种  
（5）TreeSet 、HashSet、Hashtable区别

**面试官问的面试题：**

1. 自我介绍；  
2. 为什么觉得自己适合这个岗位；  
3. 为什么选我不选别人；  
4. 目前工作都做什么；  
5. 对电商的指标有什么了解；

腾讯：

（ 6 ）常见面试题（由于有的面试时间较久，主要靠记忆写下来）

1 ）几种模型（ svm ， lr ， gbdt ， em ）的原理以及公式推导；

2 ） rf ， gbdt 的区别； gbdt ， xgboost 的区别（烂大街的问题最好从底层原理去分析回答）；

3 ）决策树处理连续值的方法；

4 ）特征选择的方法；

5 ）过拟合的解决方法；

6 ） kmeans 的原理，优缺点以及改进；

7 ）常见分类模型（ svm ，决策树，[贝叶斯](http://company.zheyibu.com/1078743/" \t "_blank)等）的优缺点，适用场景以及如何选型；

8 ） svm 为啥要引入拉格朗日的优化方法；

9 ）假设面试官什么都不懂，详细解释 CNN 的原理；

10 ）海量的 item 算文本相似度的优化方法；

11 ）梯度下降的优缺点；

12 ） em 与 kmeans 的关系；

13 ） L1 与 L2 的区别以及如何解决 L1 求导困难；

14 ）如何用尽可能少的样本训练模型同时又保证模型的性能；

15 ）解释 word2vec 的原理以及哈夫曼树的改进；

16 ）对推荐算法的未来看法；

17 ）在模型的训练迭代中，怎么评估效果；

18 ）有几个 G 的文本，每行记录了访问 ip 的 log ，如何快速统计 ip 出现次数最高的 10 个 ip ；如果只用 linux 指令又该怎么解决；

19 ）一个绳子烧完需要 1 个小时，假设所有绳子的材质都不一样，也不均匀，怎么取出 1 小时加 15 分钟；

20 ）假设有个 M\*N 的方格，从最左下方开始往最右上方走，每次只能往右或者往上，问有多少种走法，假设中间有若干个格子不能走，又有多少种走法；

21 ）实现 hmm 的状态转移代码；

22 ）最短路径代码；

23 ）拼车软件是如何定价的以及如何优化；

24 ） 100 张牌，每次只能抽一张，抽过的牌会丢掉，怎么选出最大的牌；

25 ）怎么预测降雨量；

26 ） kmeans 代码；

27 ） mr 方案解决[矩阵](http://company.zheyibu.com/44119/" \t "_blank)相乘的代码；

28 ） sql 语句的一些优化技巧；

29 ）关于集群调度的一些经验 trick 掌握多少；

30 ）设计一个系统可以实时统计任意 ip 在过去一个小时的访问量；

31 ）设计 LRU 系统；

1、推公式

LR,SVM,XGBOOST

2、用C语言写内存拷贝

3、hadoop,spark,storm下面的产品，原理，适用场景

阿里面试：

1、把一个完整的数据挖掘流程讲一下，从预处理，特征工程，到模型融合。

2、介绍常用的算法，gbdt和xgboost区别，具体怎么做预处理，特征工程，模型融合常用方式，融合一定会提升吗？

3、如何在海量数据中查找给定部分数据最相似的top200向量，向量的维度也很高。

因为之前了解过其他面蚂蚁金服的朋友，也有问到这个题目的所以反应比较快，直接就说可以用KD树，聚类，hash。

4、怎么衡量两个商品的性价比

京东面试：

1、机器学习算法的东西

2、为什么LR需要归一化或者取对数，为什么LR把特征离散化后效果更好？为什么把特征组合之后还能提升？反正这些基本都是增强了特征的表达能力，或者说更容易线性可分吧。

7 总结

经常会问到的问题，经典算法推导(加分项)，原理，各个损失函数之间区别，使用场景，如何并行化，有哪些关键参数。

 比如LR,SVM,RF,KNN，EM，Adaboost，PageRank，GBDT，Xgboost，HMM，DNN，推荐算法，聚类算法，等等机器学习领域的算法，这些基本都会被问到哪些优化方法，随机梯度下降，牛顿拟牛顿原理，生成模型，判别模型。

  线性分类和非线性分类各有哪些模型。SVM核技巧原理，如何选择核函数

  特征选择方法有哪些(能说出来10种以上加分)。

  常见融合框架原理，优缺点，bagging，stacking，boosting，为什么融合能提升效果？

  信息熵和基尼指数的关系(信息熵在x=1处一阶泰勒展开就是基尼指数)。

  如何克服过拟合，欠拟合？

  L0，L1，L2正则化(如果能推导绝对是加分项，一般人最多能画个等高线，L0是NP问题)。

其实上面的这些问题基本都能在《李航：统计学习方法》《周志华：机器学习》里面找到，能翻个4，5遍基本就无压力了。

另外可以报一下小象学院的机器学习班，等到开团的时候报也就300左右，讲的挺不错的，有算法推导和代码实现，我也是看了之后才明白很多算法的原理。

还会问一些常见的数据结构和算法，写代码的题基本都是出自《剑指offer》和《微软面试100题》，只要刷几遍，代码题这里搞定也是加分项。

其余的基本就是问问比赛，项目，遇到的问题，如何解决，在团队中的角色，评价自己。

面试官问你还有问题要问没，最好别瞎问，多看看别人的面经吧。比如问点现在这个部门做的业务，遇到过的问题，部门发展的一个规划。

最好别叫面试官评价你，要是评价不好会影响心情，自己的面试表现自己应该清楚，所以不问最好。

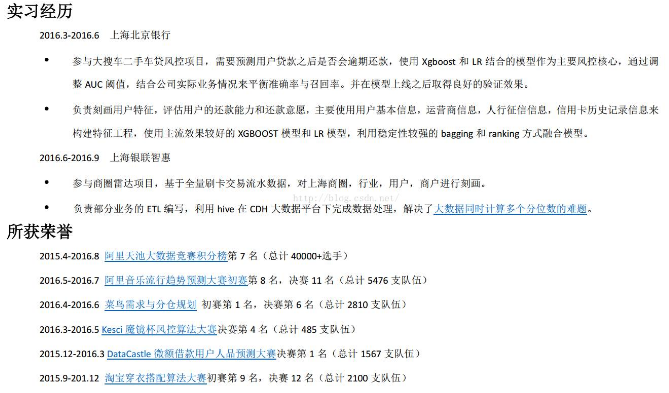
可以在自己面试的时候做记录，经常复习，先从小公司面起，积累经验。

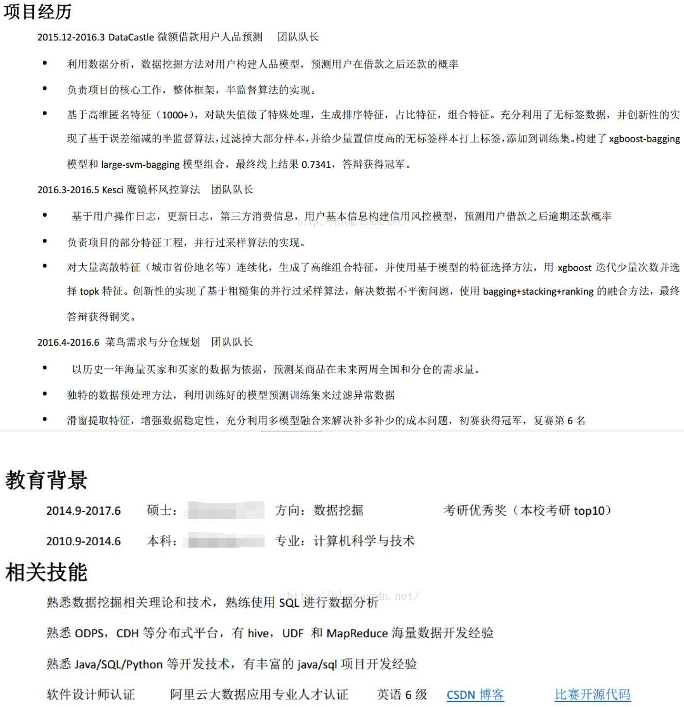
另外要定位准确，自己的水平自己清楚，不要死磕大公司，浪费了大量时间还经常一面被刷。

没事审视下自己的简历，不要把自己不熟悉的东西写上去，像什么精通之类的建议改成了解吧……不然会被问哭的。不要装逼写2页，我很多项目比赛都没写进去，只写了几个名次靠前的比赛，能吹一点的项目，其他没写进去的可以找机会主动说出来。

项目即使很水，也要吹的很难很厉害的样子。比赛和项目尽量突出里面的难点，自己做的贡献。

我的简历也是改了又改，后来找阿里星小江要了他的简历，模仿着改了一份，简单粗暴……





多刷题也比较重要，我自己的感受是内推问的问题会比通过了笔试问的要难一些，深一些。

算法的推导同样重要，之前我也是看到那些公式就蛋疼，后面没办法，看书看视频，多手推几遍，这样面试的时候更有底气。个人觉得在面试的时候能手推算法的应该比较少，所以优势是很大的。

实习生面试跪掉的原因就是没有复习，对算法的原理了解不够深入，所以在校招的时候，9月开始用了将近一个月的时间来复习。

建议多交一些牛逼的朋友，我们有个群，6个人，有4个百度offer，有3个腾讯offer（包括一个sp），还有个估计是下一个阿里星吧，还有其他公司杂七杂八的（为啥没阿里的，估计阿里不招人吧）我们经常一起打比赛，在群里讨论学术，装逼扯淡什么的，今年只要参加了的比赛，基本都拿奖了吧……

另外感觉大公司也并没有学校歧视，我的学校211都不是，问都没问过学校的问题。这篇文章好像被推到CSDN首页了，大家有兴趣可以去看看我，我本科是个学渣，读研开始写代码，所以两年，可以改变很多事情。

下面是我准备面试整理的一些资料，里面有很多算法以及面试问题：

大量算法题参考资料

http://download.csdn.net/detail/bryan\_\_/9640531

http://blog.csdn.net/v\_july\_v/article/details/6543438

机器学习公开课

http://www.julyedu.com/video/play/18/10

面试算法资料

http://www.cnblogs.com/tornadomeet/p/3395593.html

机器学习知识点总结

http://m.blog.csdn.net/article/details?id=50244695

面试编程题十大经典算法

http://mp.weixin.qq.com/s\_\_biz=MjM5Nzk2MDU5NA==&mid=2652545706&idx=3&sn=87e964154fa54999719c34c4b67dfff7&scene=0#wechat\_redirect

本文经Bryan授权转载。

原文链接：http://m.blog.csdn.net/article/details?id=52672912

Bryan的面试攻略

- DataCastle -