项目中遇到过的问题。
总之要把写在简历上的项目部分熟悉一遍，技术栈、项目功能、难点都要考虑好

研究方向

云计算方向，具体为云操作系统openstack、docker技术以及分布式系统自动化部署系统的开发设计

分布式云系统的自动部署技术研究

- 讲述下你的项目

  - 我们的项目是对分布式云系统进行自动化部署。这里所说的分布式云系统是有一个中央管理节点和若干个落地云集群构成的一个系统。我们主要是对靠近用户的落地云集群的云环境进行自动部署。我们通过一系列的web页面引导用户填写配置信息，然后对于几台服务器进行自动化部署。部署的内容包括操作系统的重装、云操作系统的自动安装。
  - 为了更加具体的描述我的项目，我可以用一个简单的例子来讲述，结合一些实现的细节。比如说现在我们已经部署好了一个云集群，需要对他进行扩容，简单来说就是添加一台物理主机，我们的系统就能够完成自动扩容的过程。
    - 首先是主机发现，就是让远程主机主动向部署节点汇报自己的ip地址、mac地址和硬件信息，这些信息都有助于后续的安装操作。
    - 然后是系统配置，用户可以选择操作系统的版本，填写一些配置信息，包括操作系统登录的账号密码、网络的配置（设置网段、配置网卡）。这些信息会生成一个模板文件(preseed文件，自动应答文件)，用于操作系统的自动安装。
    - 紧接着是云节点的配置，我们要选择这个服务器在云集群担任的角色，比如计算节点、网络结点还是存储节点，然后配置他的管理网络、外部网络和内部网络。
    - 最终我们利用ansible和docker技术，将openstack的组件封装成一个个容器，服务器安装完软件配置之后会主动拉取部署节点容器仓库中对应的容器镜像，然后在本机启动对应组件的容器，并提供相应的云服务。
  - 在这个项目里具体用到的技术包括微型镜像开发、网络安装、容器技术、ansible远程运维、web-ssh等等。

- 你负责的部分
  我从一开始就负责这个项目。因此参与了前期的技术调研，比如说各个自动化部署工具的对比，各种不同方案的选择，以及后续的技术方案确定，对于项目的需求进行明确的定义，以及整个项目的实施流程设计，并且还进行了数据库设计、功能设计等前期工作。在后期的项目开发中，主要负责的是远程运维和容器部署云操作系统openstack这部分的代码开发工作。

- 云计算是什么

  - 随着计算、存储和处理能力需求的提高，传统的it架构已经远远不能满足要求。因此为了节省成本和提高系统的可拓展性，云计算应运而生。
    云计算是一个很泛的概念，在我看来，云计算其实是通过互联网访问的一些服务。最直观的就是用户需要用到计算能力或者处理能力，但是因为资金或者条件的限制，不能构建这么一个云集群，那么就可以尝试云计算服务这种方式，通过互联网付费按需地去使用运营商提供的云服务。
  - 一般我们按服务类型来分类的话，分为三种，分别是iaas、paas和saas。
    - iaas，基础设施即服务，其实是由互联网提供的基础的计算资源，包括处理能力、存储空间、网络等等。用户可以申请硬件或虚拟硬件，包括裸机、虚拟机，然后再在上面安装自己想要的操作系统或者其他应用程序。
    - paas，平台及服务，将计算环境和开发环境等平台作为一种服务通过互联网提供给用户，一般来说用户会申请到一个已经安装好操作系统以及能够支持应用程序运行所需的运行库等软件的虚拟机或者物理机，然后再在上面安装其他应用程序。但是呢，这时候用户是不能去修改已经预装好的操作系统和运行环境。
    - saas，软件及服务，通过互联网，向用户提供软件及应用程序的一种服务方式，应用程序安装在厂商或者服务提供商那里，用户可以通过租赁的方式来使用这些软件，而不是购买，直观来说，用户可能获得到了一组用户名或者密码。

- 云计算的一些特点

  - 高可靠性 (多副本容错、计算节点同构等措施保障服务可靠性)
  - 实时在线 (服务是通过互联网来提供的，随时随地)
  - 通用性 (可以支持多种应用，使用范围广)
  - 超大规模的计算能力 (跟服务器相关)

- 按需服务 (根据用户的需求提供想要的服务)

- 云计算的一些发展

  - 云计算与大数据技术(大数据根植于云计算，云计算的关键技术中海量数据存储，海量数据管理技术和map reduce编程模型等都是大数据的基础)
  - 云计算与机器学习(提升性能，提供强大的计算能力)
  - 云计算与物联网(通过传感器，把任何物品与互联网相连，实现识别、定位、跟踪、监控等。智能融合，构建智慧地球)
  - 云计算与移动互联网(将计算从终端转到云端，只需要互联网就能实时在线访问服务，不再受限于终端的计算能力)

- 分布式云系统
  我们需要部署的分布式云系统是由一个中央管理节点和若干个落地云集群构成。中央管理节点与各个落地云集群相连接。这样的一个架构分离了数据面、控制面和服务面，为分布式部署建云提供良好的体系。

  - 中央管理节点为整个系统的管理中心和资源中心，集中管理整个云的各种资源，包括镜像资源、应用资源、数据资源等。它可以收集不同落地云集群的资源信息、节点信息以及配置信息，并且将自动配置部署的命令推送到落地云集群的部署节点上。
  - 落地云集群是部署在用户区域的小型数据中心，包含了多个节点，比如部署节点、计算节点、网络结点和存储节点等。管理云集群的云资源并提供云服务，执行来自中央管理节点的命令。

- openstack是什么

  - openstack是对数据中心的计算、存储和网络资源进行统一管理，它针对的是IT基础设施，是iaas这个层次的云操作系统。
  - openstack是iaas层的一个软件系统，包含多个服务和应用程序，这些服务和应用程序被部署到不同的主机上，这些物理主机通过网络相连，形成一个大的分布式系统。openstack云操作系统就是要解决如何自动管理由这些物理主机虚拟出来的虚拟机，包括虚拟机的创建、迁移、关闭、虚拟存储的创建和维护，虚拟网络的管理，还包括监控计费，高可用性，安全等等。
  - 管理三种类型的IT资源：计算（cpu与内存）、存储、网络，通过云计算平台，这三个资源成为了三个池子，当需要虚拟机时，根据虚拟机请求的规格，平台快速从这三个资源池分配相应的资源，部署一台虚拟机，虚拟机使用者不需要去关心运行在哪里，存储空间从哪里来，ip是怎么分配等待。云平台把这些都搞定了。

- openstack各个组件的介绍(创建虚拟机的过程，串起来)
  openstack提供了一个web界面，用户可以进行操作，这里用到的是horizon组件。我们现在要创建一台虚拟机，点击了创建虚拟机，并且填写了虚拟机的配置（包括操作系统选择、网络选择、安全组选择）然后会将这一请求交由openstack后端来完成。
  nova组件负责虚拟机的一系列生命周期管理，要知道操作系统的类型同时下载镜像以供虚拟机启动，这部分工作由glance组件来执行，glance管理的镜像可能在swift组件上，需要与swift交互才能得到所需的镜像文件。
  当然也需要cinder提供块存储的服务，需要neutron组件提供网络服务，使得虚拟机能够有volume可用，并且被分配到ip地址与外界网络与其他虚拟机进行通信。而且，在这之后，虚拟机资源的访问由keystone组件认证之后才可以继续连接。至此，openstack所有的核心组件都参与到了创建虚拟机的这个操作中。
  nova - 计算服务：虚拟机生命周期管理
  neutron - 网络服务：虚拟网络的创建、管理和网络设备接口连接
  glance - 镜像管理：镜像管理、存储
  swift - 对象存储：存储检索对象
  cinder - 块存储：块存储服务
  keystone - 身份验证：身份验证，权限控制
  horizon - 界面：基于django开发，可视化界面

- 分布式的概念
  分布式分为3步：
  将整个软件视为一个系统
  然后将系统分割成一系列的进程，每个进程完成一部分的功能
  在将这些进程分布到不同的主机上，通过若干种通信协议把他们连接起来

- 分布式与集群的区别

1. 分布式是指将不同的业务分布在不同的地方。 而集群指的是将几台服务器集中在一起，实现同一业务。
   分布式中的每一个节点，都可以做集群。 而集群并不一定就是分布式的。
2. 分布式是以缩短单个任务的执行时间来提升效率的，而集群则是通过提高单位时间内执行的任务数来提升效率。

- 虚拟化是什么
  虚拟化是云计算的基础，简单来说，虚拟化使得一台物理的服务器可以跑多台虚拟机，虚拟机共享物理机的cpu、内存、io硬件资源，但是在逻辑上各个虚拟机时分隔的。每个虚拟机有自己的虚拟硬件，并且提供独立的执行环境。
  我们一般更关注于硬件层面上的虚拟化，将物理计算机虚拟化成多台虚拟机，然后通过网络将这些虚拟机互联互通，形成了我们所说的云计算系统。硬件抽象层的虚拟化是指通过虚拟硬件抽象层来实现虚拟机，为客户机操作系统呈现出与物理硬件相同或者相近的硬件抽象层。
  有两种方式:
  方式1：hardware-hypervisor-os，vmm直接运行在硬件上并管理客户操作系统
  方式2：hardware-os-hypervisor-os kvm,virtualbox，vmm运行在一个传统的操作系统上，可以看做是软件第二层客户机os在第三层。

- 举例：hypervisor的例子——kvm
  kvm本身只管理虚拟机调度(cpu调度)和内存管理两个方面
  io的虚拟化就交给linux内核和qemu
  libvirt可以管理kvm，当然也可以管理virtualbox、xen等等。由后台daemon程序 libvirtd、api库以及virsh命令行工具构成。
  基于libvirt的高级工具：virt-manager，这个是一个图形化的工具，可以对kvm进行管理
  virsh是管理kvm的命令行工具

- kvm虚拟化的原理：

  - cpu虚拟化：查看是否支持虚拟化
    ```
    egrep -o '(vmx|svm)' /proc/cpuinfo vmx//说明cpu支持kvm
    ```
    一个kvm的虚拟机其实是一个qemu-kvm进程，与其他linux进程一样被调度。
    虚拟机中的每一个虚拟vcpu对应这个qemu-kvm进程中的一个线程(虚拟机的vcpu的数量可以超过物理cpu的数量)
  - 内存虚拟化
    kvm通过内存虚拟化来共享物理系统内存，动态分配给虚拟机。为了在一台机器上运行多个虚拟机，kvm需要实现VA(虚拟内存)-PA(物理内存)-MA(机器内存)之间的地址转换。虚拟os控制虚拟地址到客户内存物理地址的映射，但是虚拟os不能直接访问实际机器的内存，因此由kvm来负责映射客户物理内存到实际机器内存。
  - 存储虚拟化
    kvm的存储虚拟化是通过存储池和卷来管理的
    storage pool是宿主机能够看到的一片存储空间，可以是多种类型。：例如文件目录类型、lvm类型等等
    volume是storage pool中划分出来的一块空间，宿主机将volume分配给虚拟机，在虚拟机中看volume就是一块硬盘。：就是目录下的文件
  - 网络虚拟化 linux bridge和vlan
    - linux bridge
      应用场景：给虚拟机分配一个vnet0(vnet0是该虚拟网卡在宿主机中的设备名称，设备类型是TAP设备)，通过linux bridge br0将eth0与vnet0相连。
      原理：linux bridge是用来tcp/ip二层协议交换的设备，简单可以理解为一个二层交换机或者hub，多个网络设备可以连接到同一个linux bridge，当某个设备收到数据包时，linux bridge可以将数据转发给其他设备。
    - vlan
      virtual local area net 虚拟本地局域网，一个带vlan功能的switch会将自己的端口划分成多个LAN，每个lan代表一个广播域。一个广播包会被lan中的所有计算机收到。其实vlan是将一个交换机分成了多个交换机，限制了广播的范围，在二层上将计算机隔离到不同的广播域中。二层上的隔离指的是tcp/udp的包，但是在三层上还是可以互通。
      trunk口：允许所有vlan id的数据通过，并且带上自己的vlan id
      access 口：端口被打上vlan的标签，直接与计算机网卡相连
      vlan设备总是以母子关系出现，母子设备之间是一对多的关系 eth0对应eth0.10, eth0.20等等。
      eth0可以看成是trunk口，eth0.10,vnet0,brvlan10可以看成是vlan10的access口
      eth0可以看成是trunk口，eth0.20,vnet0,brvlan20可以看成是vlan20的access口
    - 总结
      linux bridge(交换)+vlan(隔离)在功能层面上完整模拟现实世界中的二层交换机

- 容器是什么
  把docker比喻成集装箱，负责运送软件

容器是直接运行在操作系统内核上的用户空间，容器技术可以让多个独立的用户空间运行在同一个宿主机上。
容器被认为是不安全的，因为它客居在宿主机上
得益于linux内核的cgroup与namespace技术，使得容器与宿主机之间的隔离更加彻底。容器有自己独立的网络和存储栈。还拥有自己管理资源的能力，使得同一台主机上的多个容器可以独立友好地共存。
docker就是能够将应用程序自动部署到容器的开源引擎。
docker客户端和服务器：c/s架构，客户端将请求发送到docker服务端或者守护进程，然后服务端/守护进程完成所有工作并返回结果。
docker镜像：镜像是基于联合(union)文件系统的一种层式的结构，由一系列的指令一步步构建。这些指令会写在dockerfile里面。
registry ：分两种，公有的docker hub与私有的registry
docker容器：容器是基于镜像启动起来，一个容器里可以运行一个或者多个进程

- docker的底层实现原理
  cgroup(资源限额)：control group控制分配的资源, linux操作系统通过cgroup设置进程使用CPU、内存和IO资源的限额。
  namespace(资源隔离)：以彼此隔离的命名空间运行，用命名空间作为权限的隔离控制。每个容器中我们都可以查看文件系统、网卡等资源，这些资源就像是容器自己的一样。实现这种方式的技术是namespace，它管理着host中全局唯一的资源，并可以让每个容器都认为自己是唯一使用者。namespace能够实现资源的隔离。
  一共有6种namespace，对应6种资源——mount namespace、UTS namespace、IPC namespace、PID namespace、network namespace、user namespace

- docker存储
  docker为容器提供了两种存放数据的资源
  由storage driver管理的镜像层和容器层：容器是由一个容器层和若干个镜像层组成，容器的数据是存放在这些层中，最大的特点是copy on write。修改的数据直接存放在最上面的容器层，修改现有的数据会先从镜像层复制到容器层，修改后的数据直接保存在容器层，镜像层不改变，如果多层中有命名相同的文件，用户只会看到最上层的文件
  data volume(可进行数据共享)：bind mount(将host上已存在的目录或文件mount到容器里)和docker managed volume(不用指定mount源)
  datavolume本质上是host文件系统中的目录或者文件，能够直接mount到容器的文件系统中，容器可以读写volume中的数据，volume数据可以被永久地保存，即使容器被销毁

- docker进行网络、cpu资源、存储资源虚拟化
  操作系统级虚拟化。内核通过创建多个虚拟的操作系统实例（内核和库）来隔离不同的进程。

- 容器编排是否了解 为什么选择ansible
  kubernate, docker compose
  项目需求

- 容器与虚拟机的区别
  首先明确一点，容器不是虚拟机，虚拟机是一种模拟系统，就是说在软件层面上通过模拟硬件的输入、输出，让虚拟机的操作系统得以运行在没有物理硬件支持的环境中。他的隔离性比较强。
  基础设施 -> 虚拟机管理系统hypervisor -> 客户机os -> 各种软件依赖-应用
  而容器呢，容器是基于镜像创建的，守护进程/docker服务端取代了hypervisor层，它是运行在操作系统上的后台进程，负责管理docker容器。守护进程直接与操作系统通信，为各个容器分配资源，将容器与操作系统进行隔离
  容器的优势：容器小巧，启动比较快，迁移部署快，运行高效，可以节省大量的磁盘空间和其他系统资源。更擅长隔离不同的应用。
  虚拟机的优势：更擅长隔离不同的运行环境。
  基础设施-> 主操作系统 -> 各种依赖（打包在容器镜像里） -> 应用

- docker的特点
  打包对象是任何软件及其依赖
  硬件依赖：容器无需修改便可运行在几乎所有的平台上-虚拟机、物理机、公有云、私有云
  隔离性：资源、网络和库都是隔离的，不会出现依赖问题
  自动化：听过run/start/stop等标准化操作，非常适合自动化
  高效性：轻量级，能够快速启动和迁移
  职责分工：开发人员考虑怎么写代码，运维人员考虑怎么配置基础环境

- 项目的难点是什么

远程运维部分，用容器来部署openstack部分，流程设计部分

- 说一下微型镜像
  首先通过 PXE 下载 microkernel ，然后直接在内存中执行，启动网卡，运行 agent 并向服务器汇报信息，并接收来自服务器的命令。基本的技术原理都是 PXE + linux initramfs ，根据不同的需要向 initramfs 中加硬件驱动。
  debootstrap + busybox 工具
  整个小系统在不安装额外的软件和内核模块的情况下，为 100 M 左右，并可加入 busybox 后裁减到 40-50 M（包含完整的基础库）
  使用 busybox 替换基本命令并裁减
  一个简化版的ubuntu系统，我们删除了一些不必要的软件、文件(busybox)，最终得到一个大约为60MB大小的操作系统，安装的时间大约为一分钟。

- linux启动过程
  BIOS自检-->从BIOS中读取启动顺序-->读取MBR中的bootloader-->加载内核-->读取伪根-->读取根文件中的init

- 用到了哪些网络协议
  dhcp获取ip地址 动态分配IP地址（dhcp-discover, dhcp-offer, dhcp-request, dhcp-ack）
  xmlrpc协议 底层是用http协议，发送硬件信息到部署节点。xml词汇表作为消息有效负载，用于两台主机之间的通信
  tftp协议：TFTP（Trivial File Transfer Protocol,简单文件传输协议）是TCP/IP协议族中的一个用来在客户机与服务器之间进行简单文件传输的协议，提供不复杂、开销不大的文件传输服务。端口号为69。简单，占用资源小，适合传递小文件。基于udp实现的。
  ssh协议：是一种网络协议，用于计算机之间的加密登录。
  用户发送登录请求-远程主机发送公钥-用户用公钥加密登录密码并发送-远程主机用私钥解密并判断密码是否正确，回复用户
  攻击：中间人攻击，假扮远程主机发送公钥，然后欺骗用户发送密码，再去登录远程主机。预防措施是口令登录，首次登录的时候提示公户只知道公钥指纹，是否继续连接。然后如果链接的话会存储到known_hosts中。

- 远程运维的过程、设计的技术 - web-ssh 介绍 原理
  web-ssh，我用到了一个开源的工具gateone。它是一个开源的项目，利用html5开发的终端模拟器，相当于一个ssh客户端。我的工作就是将它嵌入到web应用中，让我们能够在网页上点击一个链接就能远程通过ssh连接到远程主机。
  解决的问题：1. 公钥的注入（在填写操作系统配置信息的时候将公钥写入，生成preseed文件时将公钥写入客户机对应的文件中.ssh/authorized_key）2. 配置gateone服务器，gateone这个开源项目里还是有一些bug的，需要自己去修改，然后要将这个js文件放入到web应用中，然后用div组件来引入这个控件 3. 动态修改我们点击的链接，传入客户机的ip地址组成新的url，传入到gateone服务器上。4. 实现一个免密码的登录。将id_rsa放入到default_ids中

- 为什么选用容器来部署

1. 减少人工操作，解决人工部署易出错，上线时间长，后期运维人员工作量比较大，技术门槛高
2. 版本可以升级回退，只需要停止旧的容器，运行新的容器
3. 集群扩容的时间大大缩短，docker秒级启动
4. 通过编排，提高openstack的耦合度
5. 隔离不同的应用，开销比虚拟机小

- flask-web框架
  用python写的轻量级web应用框架。一个强健的核心，通过拓展的方式加入其他组件

- cobbler 自动装机， 用到的原理
  用于快速建立网络安装环境，继承了PXE,DHCP,DNS,KISKSTART服务管理和yum仓库管理工具，可以上传镜像，注册system，各个组件包括distro,profile,system,repository,image等待。

- 网络启动 pxe启动
  工作于c/s的网络模式，允许工作站通过网络从远端服务器下载映像，并由此通过网络启动操作系统，在启动过程中要求服务端分配IP地址，再用tftp协议下载一个启动软件包到本机内存中执行，由这个软件启动包完成客户机的基本软件设置，从而引导预先安装在服务器的终端操作系统。

跨领域情感文本分类

- 项目简述。什么是跨领域文本情感分类
  对于自然语言处理的情感挖掘是有重要的意义：可以挖掘大众观点，并且用于衡量大众的偏好和需求。我们研究的内容主要是研究读者对于新闻、微博等短文本信息的情感反馈，比如说看到一篇短文本，预测他可能是愤怒的、开心的或者是伤心的。简单来说就是通过一段文字来预测用户可能带有的情绪。
  什么是跨领域的文本，也就是说我们现在有两个领域的文本信息，时间和话题会导致不同领域的文本出现。比如说金融方面的，和计算机方面的。然后呢，我们在金融方面搜集到的数据是比较多的，而在计算机方面搜集到的数据是比较少的。但是我们希望能够预测计算机领域上的文本情感。我们的想法就是通过学习金融领域上的文本情感，从而映射到计算机领域上。

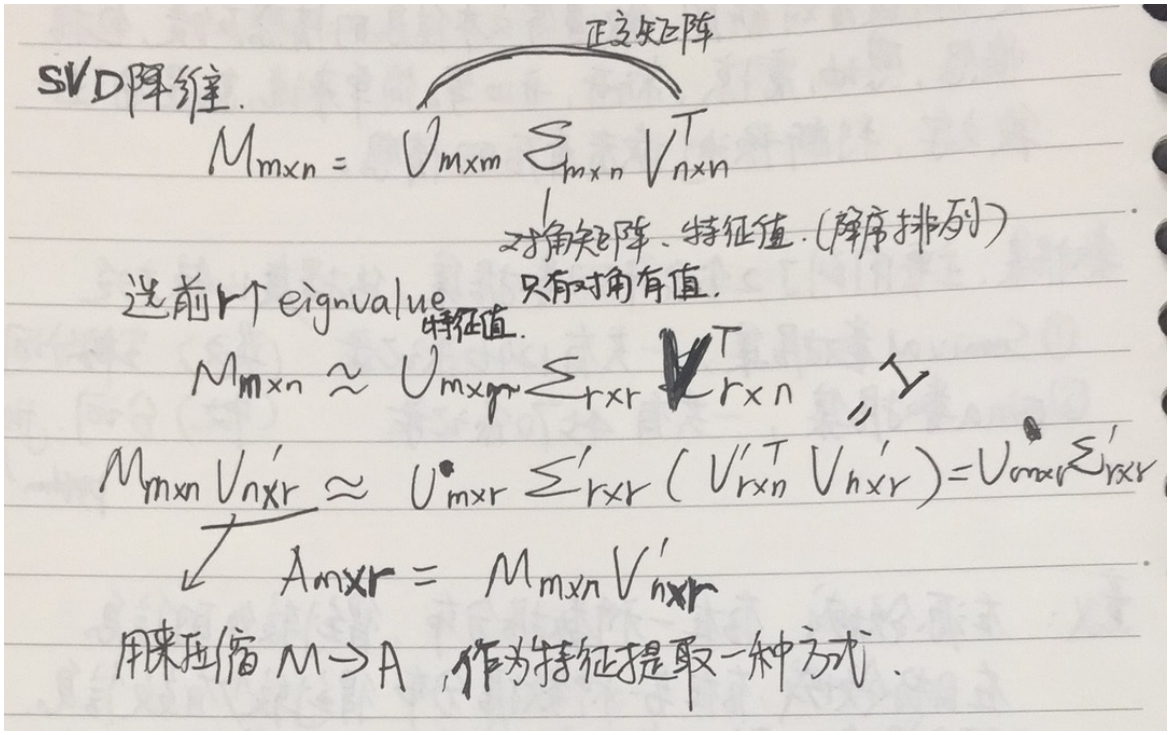
- 数学原理
  我们在源领域因为有比较多的数据信息，能够学习到一种数据分布，也可以在源领域进行情感预测；然后在目标领域，我们希望找到一个映射关系，建立一个学习模型，减少两个数据分布的差异，将源领域上的数据分布映射到目标领域的数据分布。

- 简单的过程

  - 针对源领域&少量目标领域：
  i. 将一个文本进行分词（中文，pandas库的分词）（英文）
  ii. 用预处理的方法处理输入的信息，得到文本向量（一个向量的一维可能表示一个单词，也可能表示一类词）
  iii. 将文本向量输入到学习模型中，这里选择了bp神经网络来进行学习
  iv. 通过训练得到学习参数，得到学习好参数的模型，预测情感（我们的输出有6维，然后每一维表示一个情感的分布，最终取rank1作为最终预测的情感）
  - 针对目标领域:
    将这样的学习模型用在目标领域上，用于预测

- 预处理
  主要目的是用于降低输入向量的维度，减少维度灾难，提高预测的准确度。另外我们也可能在源领域有overfitting的情况出现，从而导致在目标领域的预测准确度下降，因此需要控制正则化项。

  - 对数据用矩阵来表示（输入、输出定义）
    输入的矩阵X是：m*n*，*m表示行数，也是文本编号，n表示特征数量 输出的矩阵Y是：m*k，m表示文本编号，k表示情感类别，我们在k个标签值进行归一化后取rank1值，得到它对应的情感标签，最终判定这个文本就是对应这个标签。
  - 如何确定文本向量
    数据进行归一化：词频
    A.产生不重复的全部词列表
    B.将训练文本与测试文本转为向量
    C.对上述的向量进行归一化
  - 如何进行特征提取
  - 词共现筛选词
    方法1: 找只出现在共同领域的词，只用源领域进行训练
    方法2：找出现在共同领域的+目标领域的词，只用源领域进行训练
    方法3：用部分目标领域的文本+所有源领域文本一起训练，目标领域文本已经标记上情感，所以词来源于共同领域的词+已知的目标领域的词
  - svd降维
    将一个m*n维的矩阵压缩成一个m*r的矩阵，作为一种特征提取的方式



- k-means聚类
  是一种数据挖掘的方法，无监督算法，不需要标签，选定k，选定k个中心点，然后根据距离公式计算向量之间的距离（余弦距离、曼哈顿距离、欧式距离等距离公式）多次迭代之后，直到中心点不再改变

- 训练
  我们对训练集的数据进行有监督学习，调整学习步长、正则化参数和学习次数，观察收敛的曲线，得到一堆theta值用于预测。预测的方式就是将目标领域中的文本向量输入到已经调整好参数的学习模型中，得到对应的一个情感标签。

BP神经网络
输入层节点 - 隐藏层节点 - 输出层节点
参数初始化：不能相同，如果相同的话，层与层之间出现相同的权重，虽然拓扑结构看起来很复杂，隐藏层节点数增加，但是神经网络的表达能力却没有得到提高，效果相当于单个隐藏层节点
也不能设置为全0，会影响后面的学习。

学习率过高，会出现震荡，学习率过低，需要更多的一个迭代次数，可以根据cost function的值来自动调参。

伪代码：

```
void main(){
    读取样本点 readfile();
    初始化BP 神经网络 initialization(){
        数据归一化
        wij[]、wjk[]随机赋值
    }
    BP 神经网络训练 trainNetwork(){
        while(1){
        for(i<样本容量data_num){
        计算第i 个样本输入，产生BP 神经网络输出 compute(i); [forward]
        累计误差
        反馈调节BP 神经网络的神经元，完成第i 个样本的学习 backUpdate(i); [backward]
        }
        if(达到训练次数 或者 符合误差要求)
            break
    }
    最终的神经网络确定 writeNeuron();
    用一些数据来测试 testnetwork();
    }
```

- 数据集
  semeval数据集的246篇训练文本和1000篇测试文本在词或主题等特征的分布上截然不同；此外，新浪数据集的2342篇训练文本发布于2012年1月至2月，而2228篇测试文本发布于2012年3月至4月。我们都知道，不同时间段发布的新闻可能属于非常不同的领域或主题，导致在训练集上得到的分类模型不能很好地用在测试集上。我们的目标是缓解训练数据和测试数据的分布不一致问题，进而提高情感分类的准确度

云晖实习

- 负责的工作
  参与了phoenix项目的二次开发工作，与一个正式员工一起承担了两个子功能的开发。包括身份认证功能以及数据同步功能。

- 身份认证功能模块
  本校师生用户使用一卡通号和密码登录，使用一卡通系统进行认证；非本校师生使用自定义账户和密码登录，使用云桌面系统进行认证；不同身份不会觉察到认证方式不同。[用到了外部接口]
  根据身份认证的不同，然后限定这个用户的权限，以及他可以看到的一些界面。
  项目的用户分为管理员、教师、学生这三大类。其中管理员可细分为超级管理员、课程管理员和技术管理员，超级管理员拥有系统最高权限；技术管理员主要负责云桌面技术问题，继承自超级管理员；课程管理员主要定制课程，负责课程信息的维护。根据不同的角色，对应的菜单栏以及功能属性有所改变。
  视图切换：
  云桌面系统识别用户的身份，对于既是老师又是管理员的用户，云桌面系统提供视图切换入口。

发送了一个什么东西过去那个接口
发送一卡通号、密码到服务器端，然后得到认证信息

一卡通认证是通过调用一卡通认证系统接口的方式实现，进行一卡通账号认证。需要实现一卡通认证系统的外部接口，后台调用封装好的一卡通认证实现的jar包，并向远程服务器发送身份信息。管理系统后台根据返回的结果对用户身份的合法性进行判断，再进行下一步操作。

- 视图切换
  查询用户角色关联表user_role_relation，若返回记录数目等于2，则说明为多角色用户，主页显示视图切换按钮。根据user_role_relation表返回记录的两个role_id,，跟查询用户表得到的role_id，即当前用户所在视图角色，进行对比，得到需切换到的角色id。
  增：

  - 单一角色：
    修改User表中的role_id为新增的角色id，并且在USER_ROLE_RELATED表中新增一条数据。
    多角色（教师+管理员）：
    修改User表中的role_id为最高权限的role_id，并且在USER_ROLE_RELATED表中新增两条数据，分别对应管理员角色以及教师角色。
  - 权限控制：
    role_id，根据不同的role_id对应不同的模板
    角色表：roles
    数据库使用用户表users及角色表roles保存用户数据。使用users.role_id及role.id关联。确定permissions，
  - 底层
    数据库：使用 mysql （或 postgresql），数据库接口使用 pymysql
    ORM 框架：sqlalchemy

- 数据同步模块

  - 做什么的
    需要跟江西财大的教务系统进行数据同步，同步数据包括课程、学生、老师等信息
    1）创建课程时，根据课程号在本科教务系统中获取学生名单，或通过excel表格获取学生名单；
    2）创建课程时，根据课表信息创建教师（获取工号、姓名、一卡通号，镜像数量根据系统默认值进行设置）；
    3）每晚同步本科教务系统数据库，只增加新增学生/教师，不删除原有学生/教师。冲突数据以教务系统为准进行修改（修改学生/教师姓名）
    4）每晚根据执行课表主键更新在云桌面系统中课程信息，数据冲突以执行课表为准进行修改，如果云桌面系统课程比执

行课表，不用删除

- 如何实现同步
  管理系统根据系统参数确定同步时间，定时同步课程及与课程相关的教师、学生信息。先同步课程信息，判断课程内容是否发生变化，更新课程表，然后将课程中新增的教师同步到教师列表，最后查询教务系统，将学生信息同步。同步时不做课程增量同步，即不在云桌面管理系统数据库的课程不做同步，对课程内容，课程教师，及学生发生变化的进行更新。
  python使用pymssql模块连接sql server数据库，查询得到数据进行判断后，插入或修改云桌面管理系统mysql数据库，部署管理系统时需安装pymssql。
  通过pymssql连接sql server 数据库方法如下：

```
import pymssql
# server    数据库服务器名称或IP, user用户名,password  密码,database  数据库名称
conn = pymssql.connect(server, user, password, database)
cursor = conn.cursor()
```

数据源：使用江西财大提供的数据库视图获取数据
（1） 课表基本信息来自于实践教学中心sqlserver数据库
（2） 课表学生花名册来自于本科教务系统sqlserver数据库
同步设定在每天晚上某一时间进行，定时任务调度celery

celery后台任务
每天固定某一时间进行与远程数据库进行数据同步，更新本地数据库信息
方法：
使用python contrab实现定时任务调度。在manage.py的deploy()函数中定义执行任务的具体时间。并将设置的信息存入DatabaseSchedulerEntry()中(此表用于celery的调度)。
定时任务def sync_courseInfo()

- 任务添加修饰器@celery.task(name=' sync_courseInfo ')，代码位置在celery_tasks.py中
- 通过python mssql，连接sqlserver，查询远程数据库的内容，包括课程基本信息以及课表学生花名册，并且与本地的数据库进行比对。如果内容有更新，那么更新本地的mysql数据库。
- 根据系统参数，函数每天在固定的时间执行一次

- 遇到的难点
  由于教师视图和管理员视图不同，能给使用的操作不同，而同一一卡通号可能对应两种身份。在管理员视图中，提供切换教师视图按钮，而教师视图中，提供管理员视图按钮。
  每次登录时，默认显示上一次用户使用的视图。也即上一次用户登录系统时操作的时教师视图，则第二次登录仍然为教师视图。

- 实习的收获
  OpenStack的增值服务——云桌面系统，参与相关的业务开发
  熟悉w eb-flask框架

华为实习

- 实习的工作是什么
  Mateline是OWS(Operation Web Service)的一款集合信息聚合、社交协同、服务自编排的App，结合全球150+MS交付项目需求，发布的一款互联网产品提供Tele Operation领域的系列云服务的App界面。

  - mateline的简介：
    1.查看任务清单：提供任务列表页面，显示任务的概要信息：任务标题、最晚完成时间、任务状态。
    2.信息聚合：连接OWS应用市场，提供各种业务人员开发的应用，例如备件管理应用、告警查询应用、FME考勤管理应用、巡检管理应用等。
    3.社交化协同，实时上报任务完成情况 维护工程师根据任务进度及时处理任务，实时与调度员在线沟通，同时终端将任务状态、处理时间和所处位置同步到服务端。
    4.自编排服务：通过OWS强大的自编排服务，可以随时随地定制开发自己的应用。
    5.暂存离线数据：当手机网络不可用时，终端系统处于离线状态，系统自动将期间的任务处理、位置、照片等任务反馈信

息暂存到终端中。当手机网络恢复时，系统自动将离线操作数据信息提交到服务端，服务端接受处理后，通知手机终端已完成提交处理。

- 两个功能都是在项目APP mateLine上进行开发，第一个功能是历史版本说明，简单来说，就是通过界面上的图文形式来展示每次版本升级解决的问题以及新特性。用户能够快速地了解到版本更新前与版本更新后的差别，也能对于我们的产品有更多的补充了解。
  我的第二个功能是实现朋友圈功能。概况来说就是分享自有信息和他人分享的信息，比如文字、图片等，并且对于信息进行赞或者评论的互动交流。
- 我的所有功能并不是一开始就全部做好的，而是在原有的基础上不断进行迭代开发。首先是数据模型的应用，在本地与云端都建立了数据模型，紧接着，添加第一种类型的朋友圈，纯文字朋友圈，这时候的朋友圈列表只有单一的文字数据，然后对于单张图片的朋友圈进行实现，能够实现图片上传与下载的功能，再根据实际需求，我将原来的功能替换成多张图片朋友圈分享，这里就会涉及到图片压缩、列表显示等问题。最后就是再实现评论与点赞功能。
  这个过程中，我其实对于初始设想的交互设计、模型设计、方案选择都没有做太大的改动，在原来的版本上不断实现新的功能。我认为这样的设计与开发是很有效的，一旦前面的设计是正确的，能够大大减少后期开发的工作。

- OWS
  OWS是基于微服务架构的云化运维平台，以开源、可编程的模式来实现覆盖ICT网络的实时、按需的运维管理。OWS主要包括ICT网络端到端的监控处理、大数据分析对接、基于DevOps模式的架构设计以及全球开发者生态的建立。同时，OWS全球运维云平台持续演进，使能运维转型和适配未来ICT混合网络、NFV/SDN虚拟网络及未来5G、IoT等新技术新业务场景，致力于实现业界领先的自动化、智能化运维。

- 实习的最大收获
  有三点比较深刻体会：
  1.将大目标拆分成小目标，比如阶段性的目标、每周的目标，再到每日的目标，踏踏实实地一步一个脚印。
  2.多总结分享 学习就是一个不断发现问题再到解决问题的过程，很多问题的背后其实是有共性的，总结经验，下次会做的更好。
  3.沟通与交流 清晰地表达自己的想法 主动沟通，对于双方来说都是一次获益。

- 未来计划
  关于学习 多看书，理解更深层次的技术，将自己的技术运用到工作中
  关于项目 做项目养成好的开发习惯，谦虚地跟着老员工学习，深入地理解项目，并且能够独当一面
  关于发展 静下心来做技术开发，找准自己的定位，通过技术获得他人的认可

建议

了解这些概念：

- 微服务
  微服务可以理解为细粒度的SOA（面向服务的架构），在这样的一个架构里，每一个小的服务开发成单一应用的形式，运行在单一的进程中，并且使用http这样轻量级的API；服务能够满足使用者的需求，并且可以用自动化部署工具进行独立发布。微服务被封装成轻量级、可移植、自给自足的容器，这些容器可以使用标准操作来处理，并可以在几乎任何硬件平台上一致运行。
- devops
  DevOps一词的来自于Development和Operations的组合，突出重视软件开发人员和运维人员的沟通合作，通过自动化流程来使得软件构建、测试、发布更加快捷、频繁和可靠。
  DevOps希望做到的是软件产品交付过程中IT工具链的打通，使得各个团队减少时间损耗，更加高效地协同工作。专家们总结出了下面这个DevOps能力图，良好的闭环可以大大增加整体的产出.
  快速部署同时提高IT稳定性