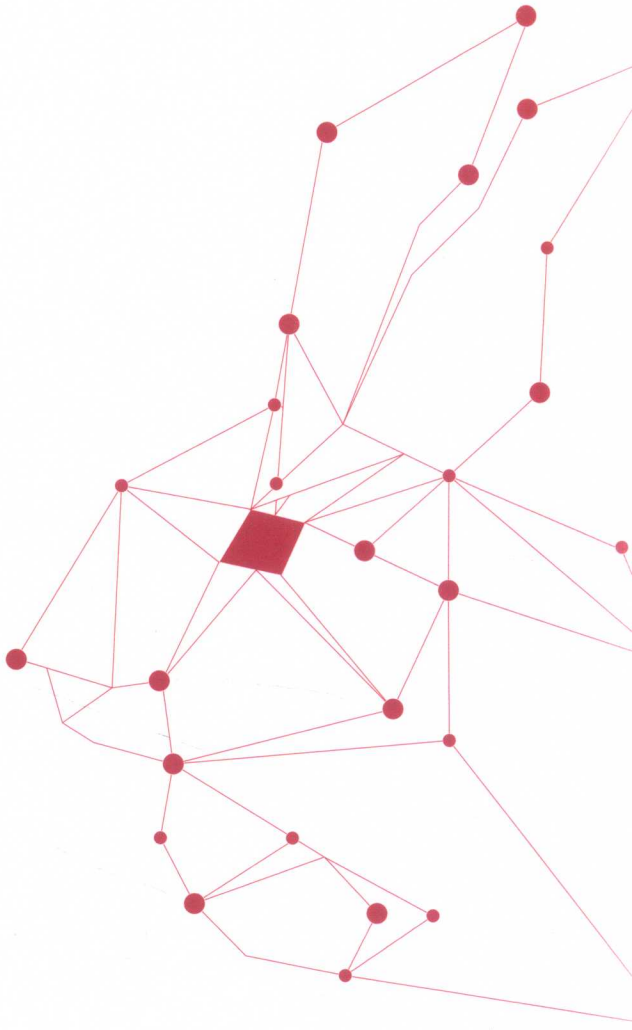
人工智能与大数据系列

# 智能搜索

## 大数据搜索引擎原理及算法解析

沙芸 编著



清華大學出版社

人工智能与大数据系列

# 智能搜索

## 大数据搜索引擎原理及算法解析

沙芸 编著



清华大学出版社
北京

## 内容简介

本书介绍大数据分布式搜索引擎开发原理与技术实现，主要内容包括多种语言的文本处理、分布式算法与代码实现、Elasticsearch 的使用与原理等，通过一个医药领域垂直搜索引擎和电商搜索来说明如何开发实际的大数据智能搜索引擎。全书共分 6 章，第 1 章着重介绍开发智能搜索引擎可以采用的软硬件环境；第 2～5 章着重讨论构建分布式智能搜索引擎可能需要的多种语言文本处理方法，例如 Kaldi 语音识别实现和基于 Raft 共识协议的分布式计算平台实现；第 6 章介绍医药和电商搜索两个应用案例。

本书适合作为高等院校计算机、软件工程专业本科生、研究生的参考用书，对于对人工智能领域感兴趣的人士也有一定的参考价值。



**图书在版编目（CIP）数据**

智能搜索：大数据搜索引擎原理及算法解析/沙芸编著. —北京：清华大学出版社，2019
（人工智能与大数据系列）
ISBN 978-7-302-53550-8

Ⅰ. ①智…　Ⅱ. ①沙…　Ⅲ. ①搜索引擎—程序设计　Ⅳ. ①TP391.3

中国版本图书馆 CIP 数据核字（2019）第 180067 号

**责任编辑：**张　敏
**封面设计：**常雪影
**责任校对：**胡伟民
**责任印制：**杨　艳

**出版发行：**清华大学出版社
网　　址：http：//www.tup.com.cn, http：//www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
**印 装 者：**北京嘉实印刷有限公司
**经　　销：**全国新华书店
**开　　本：**186mm×240mm　　**印　　张：**13　　**字　　数：**320 千字
**版　　次：**2019 年 11 月第 1 版　　**印　　次：**2019 年 11 月第 1 次印刷
**定　　价：**69.80 元

**产品编号：**081028-01

# 前言

PREFACE

智慧生物与机器集群构建的搜索系统已进化成为强大的智能系统。搜索引擎服务早已成为人们生活中不可或缺的一部分。

搜索引擎技术有着悠久的发展历史。1990 年以来，搜索引擎经历了从 Archie 那样的 FTP 文件搜索服务到谷歌网页搜索服务的转变。强化学习、深度学习等技术的发展为搜索引擎技术持续不断地注入新的活力。

本书共分 6 章：第 1 章介绍开发智能搜索引擎可以采用的软件和硬件基础；第 2 章介绍搜索引擎理解文本语义的一些方法；第 3 章介绍通过开发语音识别技术来索引音频信息的一种方法；第 4 章介绍使用 Elasticsearch 实现的大数据分布式搜索引擎；第 5 章介绍分布式计算平台中的共识算法和远程过程调用（RPC）框架；第 6 章介绍医药垂直搜索引擎和电商搜索的案例分析。

本书相关的参考软件和代码在读者 QQ 群（661922108）的附件中可以找到。一些具体的细节也可以在读者 QQ 群讨论。感谢早期合著者、合作伙伴、员工、学员、读者的支持。他们的支持给我们提供了良好的工作基础，就像玻璃容器中的水培植物一样，这是一个持久可用的成长基础。技术的融合与创新无止境，欢迎一起探索。

本书适合需要具体实现搜索引擎的程序员使用，对于信息检索等相关领域的研究人员也有一定的参考价值，同时猎兔搜索技术团队已经开发出以本书为基础的专门培训课程和商业软件。

参与本书编写的还有罗刚、张子宪、石天盈、张继红、刘晓波、叶虎、罗庭亮、柳若边，在此一并表示感谢。

作者

CONTENTS 目录


第 1 章 智能搜索引擎开发 …… 1
1.1 人工智能与智能搜索引擎 …… 1
1.2 Linux 操作系统基础 …… 2
1.2.1 SSH远程登录 …… 2
1.2.2 Micro文本编辑器 …… 4
1.2.3 Linux Shell脚本基础 …… 4
1.2.4 Shell脚本 …… 5
1.2.5 AWK …… 8
1.3 Java 基础 …… 8
1.3.1 使用Ant …… 9
1.3.2 使用Maven …… 11
1.3.3 使用Gradle …… 13
1.3.4 使用Groovy Shell …… 16
1.3.5 使用JShell …… 17
1.4 Python 基础 …… 17

1.4.1 Windows下安装Python ····· 17
1.4.2 Linux下安装Python ····· 17
1.4.3 开发环境 ····· 18
1.5 C#基础 ····· 19
1.6 硬件基础 ····· 21
1.7 本章小结 ····· 22

**第 2 章 搜索引擎理解语义 ····· 23**

2.1 处理文本 ····· 23
2.2 基于文法的语言模型 ····· 24
2.3 正则表达式查找文本 ····· 25
2.4 中文词语切分与词性标注 ····· 27
2.4.1 使用中文分词 ····· 28
2.4.2 正向最大长度匹配法 ····· 30
2.4.3 未登录串识别 ····· 31
2.4.4 基本的*N*元模型 ····· 34
2.5 隐马尔可夫模型 ····· 43
2.5.1 数据基础 ····· 43
2.5.2 维特比算法 ····· 44
2.6 英文文本切分与标注 ····· 48
2.6.1 句子切分 ····· 48
2.6.2 标注词性 ····· 50
2.7 命名实体识别 ····· 52
2.7.1 人名识别 ····· 52
2.7.2 人名识别规则 ····· 53
2.8 文本归一化 ····· 61
2.9 依存树模型 ····· 62
2.10 情感分析 ····· 63
2.11 本章小结 ····· 66

**第 3 章 搜索引擎听懂语音 ····· 67**

3.1 语音识别总体结构 ····· 67

3.2 Kaldi 快速入门……68
3.2.1 安装Kaldi……69
3.2.2 yesno例子……69
3.2.3 数据准备……70
3.2.4 词典准备……71
3.2.5 构建一个简单的ASR……74
3.3 使用 FFmpeg 提取音频……82
3.4 时间序列……82
3.5 动态时间规整……84
3.6 傅里叶变换……86
3.6.1 离散傅里叶变换……86
3.6.2 快速傅里叶变换……89
3.7 MFCC 特征……92
3.8 在线解码……93
3.8.1 使用现成的模型……93
3.8.2 使用Alex-ASR……94
3.9 加权有限状态转换……95
3.9.1 FSA……96
3.9.2 FST……97
3.9.3 WFST……98
3.10 语音识别语料库……99
3.10.1 TIMIT语音库……99
3.10.2 中文语音库……99
3.11 本章小结……100

**第 4 章 Elasticsearch 分布式搜索引擎……101**
4.1 搭建 Elasticsearch 集群……101
4.2 索引数据……103
4.3 实现搜索接口……107
4.4 搜索界面开发……108
4.4.1 使用Spring Boot开发搜索界面……109
4.4.2 使用.NET开发搜索界面……132

4.5 检索模型 ····· 142
4.5.1 使用BM25检索模型 ····· 146
4.5.2 参数调优 ····· 146
4.6 搜索中文优化 ····· 147
4.7 Elasticsearch 源代码分析 ····· 152
4.7.1 导入源代码到Eclipse ····· 152
4.7.2 Guice框架 ····· 152
4.7.3 Netty异步IO框架 ····· 154
4.7.4 分布式设计与实现 ····· 155
4.7.5 使用Lucene ····· 156
4.8 本章小结 ····· 159

**第 5 章 分布式计算平台 ····· 160**

5.1 Atomix 框架 ····· 160
5.1.1 Raft协议 ····· 160
5.1.2 使用Atomix ····· 162
5.2 gRPC 框架 ····· 164
5.3 本章小结 ····· 167

**第 6 章 智能搜索案例分析 ····· 168**

6.1 医药垂直搜索引擎 ····· 168
6.1.1 网络爬虫 ····· 169
6.1.2 抓取PubMed ····· 177
6.1.3 MVC搜索界面开发 ····· 179
6.1.4 构建知识库 ····· 183
6.1.5 自动问答 ····· 185
6.2 电商搜索 ····· 188
6.2.1 电商爬虫 ····· 188
6.2.2 商品搜索 ····· 192
6.2.3 在线客服 ····· 195
6.3 本章小结 ····· 198

**参考文献 ····· 199**

# 第 1 章
# 智能搜索引擎开发

本章首先介绍人工智能的基本知识，然后以商品搜索为例介绍智能搜索引擎。

## 1.1 人工智能与智能搜索引擎

在计算机科学中，人工智能（artificial intelligence，AI），有时也称为机器智能，是机器展示的智能，与人类和其他动物展示的自然智能形成鲜明对比。计算机科学将人工智能研究定义为对“智能代理”的研究：任何能够感知其环境并采取最大化其成功实现目标的机会的设备。更详细的是，Kaplan 和 Haenlein 将人工智能定义为“系统正确解释外部数据，从这些数据中学习，并通过灵活适应实现特定目标和任务的能力”。通俗地说，当机器模仿人类与其他人类思维相关的“认知”功能时，应用“人工智能”这一术语，如“学习”和“解决问题”。

Kaplan 和 Haenlein 将人工智能分为三种不同类型的人工智能系统：分析人工智能、人类启发人工智能和人性化人工智能。分析人工智能只具有与认知智能相一致的特征，从而产生世界的认知表征，并使用基于过去经验的学习来为未来的决策提供信息。除了认知元素之外，人类启发人工智能还具有认知和情感智能，理解并在决策中考虑人类情感。人性化人工智能显示了所有类型能力（即认知、情感和社交智能）的特征，能够在与他人的交互中体现自我意识。

人工智能研究的传统问题（或目标）包括推理、知识表示、计划、学习、自然语言

处理、感知，以及移动和操纵物体的能力。通用智能属于该领域的长期目标。其方法包括统计方法、计算智能和传统的符号人工智能。人工智能中使用了许多工具，包括搜索和数学优化的版本、人工神经网络，以及基于统计学、概率论和经济学的方法。人工智能领域涉及计算机科学、信息工程、数学、心理学、语言学、哲学等。

人工智能领域的基础是人类智能“可以如此精确地描述，可以使机器模拟它”。这提出了关于心灵本质的哲学论证以及创造具有人类智慧的人造生物的伦理。这些问题是自古以来神话、小说和哲学所探讨的问题。有些人认为，如果人工智能进展有增无减，就会对人类构成威胁；也有些人认为，与以往的技术革命不同，人工智能会造成大规模失业的风险。

21 世纪，人工智能技术在计算机能力、大量数据和理论理解的同步发展之后经历了复苏；人工智能技术已经成为技术行业的重要组成部分，有助于解决计算机科学、软件工程和运筹学中的许多挑战性问题。

强化学习（reinforcement learning）这一领域研究人工（和自然）系统如何学习在复杂环境中基于外部和可能延迟的反馈作出决策。强化学习是人工智能的一个重要分支，它也是 DeepMind 公司的围棋软件阿尔法狗这样的智能系统的重要组成部分。

如果把搜索引擎看作智能体（agent），把用户看作环境（environment），则商品的搜索问题可以被视为典型的顺序决策问题（sequential decision making problem）：

（1）用户每次请求访问页面时，智能体作出相应的排序决策，将商品展示给用户。

（2）用户根据智能体的排序结果，给出点击、翻页等反馈信号。

（3）智能体接收反馈信号，在新的页面访问请求时作出新的排序决策。

（4）这样的过程将一直持续下去，直到用户购买商品或者退出搜索。

在以上问题的形式化中，智能体每一次策略的选择可以被看作一次试错。在这种反复不断试错的过程中，智能体将逐步学习到最优的排序策略。而这种在与环境交互的过程中进行试错的学习，正是强化学习的根本思想。

## 1.2 Linux 操作系统基础

很多大数据搜索引擎运行在 Linux 操作系统中。Linux 来源于 UNIX，Linux 是 UNIX 操作系统的开放源代码实现。

### 1.2.1 SSH 远程登录

通过 SSH 客户端软件可以连接到远程的 Linux 服务器。SSH 服务器通常作为大多

数 Linux 发行版上易于安装的软件包提供。用户可以尝试使用 ssh localhost 来测试它是否正在运行。

如果有现成的 Linux 服务器可用，可以使用支持 SSH 协议的终端仿真程序 SecureCRT 连接到远程 Linux 服务器。因为可以保存登录密码，所以比较方便。除了 SecureCRT，还可以使用开源软件 PuTTY(http://www.chiark.greenend.org.uk/~sgtatham/putty)，或者使用可以保存登录密码的 KiTTY(https://www.fosshub.com/KiTTY.html)。如果是用 root 账户登录，则终端提示符是#，否则终端提示符是$。

也可以在 Windows 操作系统下安装 Cygwin，使用它来练习 Linux 常用命令。

使用 VMware，Linux 可以运行在 Windows 系统下。VMware 让 Linux 运行在虚拟机中，而且不会破坏原来的 Windows 操作系统。首先要准备好 VMware，当然仍然需要 Linux 光盘文件。

就好像华山派有剑宗和气宗，Linux 也有很多种版本，例如 RedHat 或者 Ubuntu，以及 SUSE 等。这里介绍 Ubuntu(https://www.ubuntu.com)和 CentOS(http://www.centos.org/)。

操作系统中可能会安装好几个版本的 JDK，在 Linux 中，为了切换 JDK 版本，只需要修改/etc/alternatives 中的符号链接指向。

在 Ubuntu 中，如果需要安装软件，可以下载 DEB 格式的安装包，然后使用 dpkg 命令安装。但一个软件包可能依赖其他软件包。为了安装一个软件可能需要下载其他几个它所依赖的软件包。

为了简化安装操作，可以使用高级包装工具（Advanced Packaging Tool，APT）。APT 会自动计算出程序之间的相互关联性，并且计算出完成软件包的安装需要哪些步骤。这样在安装软件时，不会再被那些关联性问题所困扰。

在/etc/apt/sources.list 文件中指示了包的来源的存储库。包的来源可以是 CD 或 DVD，硬盘上的目录或 HTTP 或 FTP 服务器上的目录。请求的数据包位于服务器（或本地硬盘）上，它将自动下载并安装。APT 主要关注采购包，包的可用版本的比较以及包档案的管理。实际上，可以通过浏览器浏览在 FTP 或 HTTP 上的存储库。

如果需要修改/etc/apt/sources.list 文件，可以先备份这个文件：

```
# sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak
```

如果这一步出现：

```
sudo: unable to resolve host t-000004
```

这样的错误，则可以考虑执行如下的命令修改/etc/hosts 文件的内容：

```
# echo $(hostname -I | cut -d\  -f1) $(hostname) | sudo tee -a /etc/hosts
```

如果安装过程中出现“E: Could not get lock /var/lib/dpkg/lock”这样的错误，则可以尝试使用如下命令修复：

```
# sudo fuser -cuk /var/lib/dpkg/lock
# sudo rm -f /var/lib/dpkg/lock
```

在 CentOS 中，如果需要安装软件，可以下载 RPM 安装包，然后使用 RPM 安装包进行安装。例如，下载 Elasticsearch 软件的安装包 elasticsearch-6.6.0.rpm：

```
# wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.6.0.rpm
```

使用如下命令安装：

```
# rpm -ivh elasticsearch-6.6.0.rpm
```

但有些操作系统对应的 RPM 安装包找起来往往比较麻烦。

为了简化安装操作，可以使用黄狗升级管理器（Yellow dog Updater,Modified），一般简称 YUM。YUM 会自动计算出程序之间的相互关联性，并且计算出完成软件包的安装需要哪些步骤。

YUM 软件包管理器自动从网络下载并安装软件。YUM 有点类似 360 软件管家，但是不会有商业倾向的推销软件。例如安装支持 wget 命令的软件：

```
#yum install wget
```

### 1.2.2 Micro 文本编辑器

为了方便在服务器端开发 Python/Perl 相关应用，可以采用 Micro(https://github.com/zyedidia/micro)这样的终端文本编辑器。

在 Linux 上，可以通过 snap 安装 micro：

```
# snap install micro --classic
```

可以使用它编辑配置文件：

```
#./micro  run.pl
```

输入：

```
die "run.pl: Hello Error";
```

这里的 die 表示终止脚本运行，并显示出 die 后面的双引号里面的内容。

保存文件后，使用 Ctrl+q 组合键退出。

### 1.2.3 Linux Shell 脚本基础

Shell 是用户和 Linux 内核之间的接口程序。用户在命令行提示符下输入的每个命令都由 Shell 先解释再传给 Linux 内核。

Shell 是一个命令语言解释器，拥有自己内建的 Shell 命令集。此外，Shell 也能被系统中其他有效的 Linux 实用程序和应用程序所调用。

Shell 具有如下主要功能。

- 命令解释功能：将用户可读的命令转换成计算机可理解的命令，并控制命令执行。
- 输入输出重定向：操作系统将键盘作为标准输入、显示器作为标准输出。当这些定向不能满足用户需求时，用户可以在命令中用符号“>”或“<”重新定向。
- 管道处理：利用管道将一个命令的输出送入另一个命令，实现多个命令组合完成复杂命令的功能。
- 系统环境设置：用 Shell 命令设置环境变量，维护用户的工作环境。
- 程序设计语言：Shell 命令本身可以作为程序设计语言，将多个 Shell 命令组合起来，编写能实现系统或用户所需功能的程序。

有很多种 Shell，例如 zshell 和 fish。目前一般使用 Bash 脚本。

### 1.2.4 Shell 脚本

在屏幕上输出“Hello”：

```
echo "Hello"
```

将 ABC 分配给 a：

```
a=ABC
```

输出 a 的值：

```
echo $a
```

将 ABC.log 分配给 b：

```
b=$a.log
```

输出 b 的值：

```
# echo $b
ABC.log
```

把文件“ABC.log”的内容写入到 testfile

```
cat $b > testfile
```

指令“--help”会输出帮助信息。

可以把重复执行的 Shell 脚本写入到一个文本文件。在 Linux 中，文件扩展名不作为系统识别文件类型的依据，但是可以作为用户识别文件的依据，可以简单地将脚本文件以.sh 作为扩展名。

在 Linux 下，可以通过 vi 命令创建一个诸如 script.sh 的文件：vi script.sh。创建好脚本文件后就可以在文件内用脚本语言要求的格式编写脚本程序了。

在创建的脚本文件中输入以下代码并保存退出：

```
#! /bin/bash
echo "hello world!"
```

添加脚本文件的可执行运行权限 chmod 777 script.sh 后，运行文件./script.sh 得到结果：

```
hello world!
```

Shell 脚本中用#表示注释，相当于 C 语言的//注释。但如果#位于第一行开头，并且是#!（称为 Shebang）则例外，它表示该脚本使用后面指定的解释器/bin/sh 解释执行。每个脚本程序必须在开头包含这个语句。

使用参数 n 检查语法错误，例如：

```
# bash -n ./test.sh
```

如果 Shell 脚本有语法错误，则会提示错误所在行；否则，不输出任何信息。

if 语句的语法是：

```
if [ condition ] then
      command1
elif   # 和 else if 等价
   then
        command2
   else
     default-command
fi
```

这里的 fi 就是 if 反过来写。

例如，为了判断某个命令是否存在，可以使用如下格式：

```
if which programname >/dev/null; then
   echo exists
else
   echo does not exist
fi
```

判断 yum 是否存在的例子：

```
if which yum >/dev/null; then
   echo "exists"
else
   echo "does not exist"
fi
```

case 语句的语法是：

```
case 字符串 in
   模式 1)
      语句
      ;;
   模式 2)
       语句
```

```
        ;;
    *)
        默认执行的 语句
        ;;
esac
```

这里的 esac 就是 case 反过来写。例如：

```
extension="png"
case "$extension" in
  "jpg"|"jpeg")
    echo "It's image with jpeg extension."
  ;;
  "png")
    echo "It's image with png extension."
  ;;
  "gif")
    echo "Oh, it's a giphy!"
  ;;
  *)
    echo "Woops! It's not image!"
  ;;
esac
```

这里使用“|”把"jpg"和"jpeg"这两个模式连接到了一起。

介绍 4 种模式匹配：

- ${variable#pattern}

从$string 的前面删除$substring 的最短匹配

- ${variable##pattern}

从$string 的前面删除$substring 的最长匹配

- ${variable%pattern}

从$string 的后面删除$substring 的最短匹配

- ${variable%%pattern}

从$string 的后面删除$substring 的最长匹配

使用模式匹配的例子：

```
x=/home/cam/book/long.file.name
echo ${x#/*/}
echo ${x##/*/}
echo ${x%.*}
echo ${x%%.*}
   cam/book/long.file.name
   long.file.name
   /home/cam/book/long.file
   /home/cam/book/long
```

安装 fish Shell：

```
# sudo apt-get install fish
```

### 1.2.5 AWK

典型的 AWK 程序充当过滤器。它从标准输入读取数据，并输出标准输出的过滤数据。它一次读取数据的一个记录。默认情况下，一次读取一行文本。每次读取记录时，AWK 自动将记录分隔到字段中。字段在默认情况下也是由空格分隔的。每个字段被分配给一个变量，该变量有一个数字名称。变量$ 1 是第一个字段，$ 2 是第二个字段，以此类推。$ 0 表示整个记录。此外，还设置了一个名为 NF 的变量，其中包含在记录中检测到的字段的数目。

来试试一个很简单的例子。过滤 ls 命令的输出：

```
# ls -l ./ | awk '{print $0}'
```

显示文本文件 nohup.out 匹配（含有）字符串"sun"的所有行。

```
# awk '/sun/{print}' nohup.out
```

由于显示整个记录（全行）是 awk 的默认动作，因此可以省略 action 项。

例如，得到 Python 的版本号：

```
# python 2>&1 --version | awk '{print $2}'
2.7.5
```

这里 2>&1 的意思是：把标准错误重定向到标准输出。

## 1.3 Java 基础

在 Ubuntu 下安装 JDK 可以参考：

https://github.com/AdamScheller/UbuntuJavaInstaller/blob/master/ujavainstaller.sh

首先从 https://www.oracle.com/technetwork/java/javase/downloads/jdk11-downloads-5066655.html 下载 jdk-11.0.2_linux-x64_bin.tar.gz：

然后设置环境变量：

```
#JDK_VERSION=jdk-11.0.2
#mkdir -p /usr/local/java
#JDK_ARCHIVE=jdk-11.0.2_linux-x64_bin.tar.gz
#JDK_LOCATION=/usr/local/java/$JDK_VERSION
#tar -xf $JDK_ARCHIVE -C /usr/local/java
#cat >> /etc/profile <<EOF
JAVA_HOME=$JDK_LOCATION
JRE_HOME=$JDK_LOCATION/jre
PATH=$PATH:$JDK_LOCATION/bin:$JDK_LOCATION/jre/bin
```