Windows系统下使用维基百科中文语料训练Word2Vec词向量 - 灰信网（软件开发博客聚合）

Windows系统下使用维基百科中文语料训练Word2Vec词向量

标签： 维基百科 word2vec 词向量

Windows系统下使用维基百科中文语料训练word2vec词向量
By 龙前尘

实验环境：win8、python 2.7

转载请注明地址：
http://blog.csdn.net/svenhuayuncheng/article/details/78751311

笔者按

笔者近期用简单问句模板，搭建了一个面向特定领域的KBQA系统。在做问法泛化的时候，考虑使用问句和同义词集合的交集来处理。
在实验时，笔者发现，单纯使用Jaccard交集来刻画问句与模板的相似度，还是有些粗糙，需要大量的人力统计或制造更可能多的模板，尽量覆盖问法。但是显然，这个做法是没有边界的。
为了解决这个问题，并且体现出问句与模板的语义相似度，笔者想使用词向量为基础，来尝试各种相似度的计算，来找到最相近的模板。
因此，经过一段时间的实验（主要是排雷），完成了词向量训练。笔者使用的是维基百科中文语料，以及特定领域的一些语料进行训练。
这里记录一下全过程，与大家分享与交流。

1. 提取语料

1.1 下载wiki百科的数据

从wiki百科下载中文最新语料。
dump下载地址：https://dumps.wikimedia.org/zhwiki/latest/
本文使用的语料为：zhwiki-latest-pages-articles.xml.bz2，大小大约为1.4GB。

1.2 数据抽取

受到这篇博文获取并处理中文维基百科语料的启发，编写python代码实现。
其中的主要工作，是从下载的压缩包中抽取出wiki百科正文，并通过正则表达式进行初步过滤。首先去掉那些帮助页面和重定向的页面，这些页面的信息都是无用的。其次处理页面的一些特殊的、非文本的标记，将其去掉。最后使用opencc对于文本信息进行繁体到简体的转化。关于如何安装opencc，可以查看博主的文章：手动安装opencc（中文简繁体转换插件） ——解决安装opencc时出现HTTP 403错误的问题。
代码如下：

```
#coding:utf-8
from gensim.corpora.wikicorpus import extract_pages,filter_wiki
import bz2file
import re
import opencc
from tqdm import tqdm
import codecs

wiki = extract_pages(bz2file.open('zhwiki-latest-pages-articles.xml.bz2'))

def wiki_replace(d):
    s = d[1]
    s = re.sub(':*{\|[\s\S]*?\|}', '', s)
    s = re.sub('[\s\S]*?', '', s)
    s = re.sub('(.){{([^{}\n]*?\|[^{}\n]*?)}}', '\\1[[\\2]]', s)
    s = filter_wiki(s)
    s = re.sub('\* *\n|\'{2,}', '', s)
    s = re.sub('\n+', '\n', s)
    s = re.sub('\n[:;]|\n +', '\n', s)
    s = re.sub('\n==', '\n\n==', s)
    #cc = opencc.OpenCC('mix2s')
    #return cc.convert(s).strip()
    return s

i = 0
f = codecs.open('wiki.txt', 'w', encoding='utf-8')
w = tqdm(wiki, desc=u'已获取0篇文章')
for d in w:
    #re.findall返回正则表达式匹配的所有字符串，用于去掉帮助页面和重定向页面;
    if not re.findall('^[a-zA-Z]+:', d[0]) and not re.findall(u'^#', d[1]):
        s = wiki_replace(d)
        f.write(s+'\n\n\n')
        i += 1
        #print(u'已获取%s篇文章'%i)
        if i % 100 == 0:
            w.set_description(u'已获取%s篇文章'%i)

f.close()
```

注意，此处的第21、22行是使用opencc接口进行繁体中文到简体中文的转化代码。但是笔者在跑代码时，发现使用python的opencc接口，程序会一直卡在繁/简转换这里，而不继续运行。这个bug暂时还没有解决。笔者使用了另一种方法来运行opencc进行转换。如果大家可以正常运行代码，那么可以去掉这里的注释，在代码中直接进行繁/简转换。

2. 繁简转化

如前文所述，在python代码中跑opencc会有问题。故笔者使用命令行来进行繁/简转化。
首先在https://bintray.com/package/files/byvoid/opencc/OpenCC下载opencc的windows安装包opencc-1.0.1-win64.7z，并解压放置到自定义的目录下。
然后在该目录下，放入待转换的文件，并在命令行运行该命令来转化，可以得到转化后的文件：

```
opencc -i wiki.txt -o wiki.zh.jian.txt -c t2s.json
     1
```

经过转化的简体中文文件，其编码为是utf16，需要写代码进行文件编码的转换，转换为utf8的编码，转换代码如下：

```
#encoding:utf8
import codecs
from tqdm import tqdm

def transformFile(ipath, opath):
    encoding = 'utf-16-le'
    iFile = codecs.open(ipath, 'r', encoding)
    encoding = 'utf-8'
    oFile = codecs.open(opath, 'w', encoding)
    sentences = iFile.readlines()
    i = 0
    w = tqdm(sentences, desc=u'转换句子')
    for sentence in w:
        oFile.write(sentence)
        i += 1
        if i % 100 == 0:
            w.set_description(u'已转换%s个句子'%i)
    iFile.close()
    oFile.close()

ipath = 'wiki.zh.jian.txt'
opath = 'wiki.zh.jian.utf8.txt'
transformFile(ipath, opath)
```

3. 语料清洗

经过简体化的文档，仍然有很多脏信息。如数字、标点符号、非中文语言字符等，并且文档中的句子是不能用来训练的，需要进行分词处理。
故编写代码，进行非中文字符串的清除，以及分词。

这个博客有整理好的停用词，可以直接拿来做分词用。

分词及文本清洗的代码如下，此处使用jieba分词工具。

```
#encoding:utf8
import jieba
import os
import codecs
from tqdm import tqdm


class MySentences(object):
    def __init__(self, dirname):
        self.dirname = dirname

    def __iter__(self):
        for fname in os.listdir(self.dirname):
            for line in open(os.path.join(self.dirname, fname)):
                if len(line) > 0:
                    yield [segment.strip() for segment in jieba.cut(line.strip(), cut_all=False)
                           if segment not in stoplist and len(segment) > 0]


def is_ustr(instr):
    out_str = ''
    for index in range(len(instr)):
        if is_uchar(instr[index]):
            out_str = out_str + instr[index].strip()
    return out_str


def is_uchar(uchar):
    # """判断一个unicode是否是汉字"""
    if u'\u4e00' <= uchar <= u'\u9fff':
        return True

if __name__ == '__main__':
    dirname = 'zh_simplify'
    # 读取停用词；
    stop_f = codecs.open(u'停用词.txt', 'r', encoding='utf-8')
    stoplist = {}.fromkeys([line.strip() for line in stop_f])
    # 进行jieba分词
    sentences = MySentences(dirname)
    # 分词结果写入文件
    f = codecs.open('wiki_jieba.txt', 'w', encoding='utf-8')
    i = 0
    j = 0
    w = tqdm(sentences, desc=u'分词句子')
    for sentence in w:
        if len(sentence) > 0:
            output = " "
            for d in sentence:
                # 去除停用词；
                if d not in stoplist:
                    output += is_ustr(d).strip() + " "
            f.write(output.strip())
            f.write('\r\n')
            i += 1
            if i % 10000 == 0:
                j += 1
                w.set_description(u'已分词： %s万个句子'%j)
    f.close()
```

经过分词之后的文本，大概是这样的：

```
1 豆乳 晶 加工 豆乳 浓缩 真空 干燥 呈 蜂窝状 固体 速溶 饮料 过程 豆乳 晶 豆乳 特有 风味 克服 豆乳 液体 包装 贮存 弱点 豆乳 晶 加工 工艺
  同豆 乳粉 加工 工艺 主要 区别 干燥 形式 豆乳 晶 采用 真空 干燥 喷雾干燥 真空 干燥 豆乳 晶 生产 关键 工序 真空 干燥 真空 烘箱 中 完成
  先 浓缩 豆浆 砂糖 麦精 维生素 物质 均匀 混合液 盛入 烘 盘中 烘干 时 烘盘 置于 烘箱 多层 蒸汽 加热 板上 烘 盘底 加热 板 大面积 接触
  热量 传递 浓缩 豆乳 真空 系统 作用 减压 蒸发 排除 水分 热 沸腾 发泡 冷却 固形 疏松 多孔 呈 蜂窝状 豆乳 晶 操作 时 烘箱 真空度 控制
  毫米汞柱 温度 干燥 产品 需 粉碎 粒度 筛分 包装 豆乳 晶 呈 淡黄色 溶解度 大于 水分 小于 蛋白质 含量 大于 碳水化合物 大于 不应 超过
2 非洲 猪瘟 非洲 猪瘟 病毒 急性 高度 接触 传染病 病程 短 死亡率 高 全身 各器官 组织 出血 部位 发生 水肿 特征 原发 非洲 现 分布 世界各地
  病毒 分布 病猪 体内 组织 器官 体液 分泌物 排泄物 中 储存 宿主 疣 猪 野猪 表现 无症状 感染 家猪 动物 接触 污染 饲料 饮水 用具 场舍 感染
  发病 猪 虱 隐嘴 蜱 软体 蜱 带毒 传播媒介 感染 天 发热 前 小时 病毒血症 全身性 感染 潜伏期 天 急性 型 病猪 体温 升高 稽留 临床 表现
  症状 热 下降 精神 沉郁 食欲 废绝 后肢 无力 心跳 呼吸 加快 咳嗽 眼 鼻粘膜 浆液 性 粘液性 分泌物 鼻端 耳 腹部 四肢 紫斑 腹泻 粪便 带血
  呕吐 亚急性 病程 延至 天 慢性病 猪 主要 呈 慢性 肺炎 病猪 白蛋白 低 血症 表现 丙种球蛋白 高 血症 病程 数周 数月 病理变化 脾肿大 胸水
  腹水 增量 混 血液 淋巴结 肿大 出血 胆囊 壁 水肿 粘膜 出血 肾 膀胱 粘膜 胃肠道 浆膜 心内膜 心外膜 出血 组织学 脾 淋巴结 淋巴细胞 崩毁
  组织 坏死 肝 肾 实质 变性 症状 病理 组织学 变化 作 印象 诊断 确诊 动物 接种 病毒 分离 应 猪瘟 相 区别 疫区 车 船 飞机 卸下 肉食品 废料
  废水 无害化 杜绝 疫区 进口 猪 产品 进口 猪 产品 应 检疫 猪群 中 发现 可疑 病猪 时 应立即 封锁 诊断 全群 扑杀 销毁 消毒
3 渔业资源 监测 渔业资源 数量 质量 连续 定期 观察 测量 分析 一种 手段 监测 动态 发现 制定 相应措施 控制 资源 波动 幅度 利用 保护 资源
  发展 渔业生产 目的 海里 专属经济区 实施 世界 渔业 开发型 转为 管理型 渔业资源 监测 倍受 重视 沿海 国家 相继 建立 监测 体系 年 中国
  黄海 东海 南海 海区 建立 资源 监测站 体系 资源 监测 主要 两个 数量 监测 渔业资源 一种 更新 生物资源 动态 特性 资源 数量 种群 动态 指标
  包括 种群 数量 数量 世代 数量 补充 量 指标 主要 三种 方式 获取 海上 调查 方式 大面积 拖网 试捕 卵子 幼鱼 数量 调查 声学 资源 评估 航空
```

每一行为一篇文档，每个文档被分为许多词语的组合，且以空格分开。
这里我建立了一个名为‘zh_simplify’的文件夹，里面存放有几个文本文件，代码中迭代地处理这些文件，并将语料都存于最终的一个文档中。
整个清理过程大约40分钟。

4. 词向量训练

经过那么多步骤的预处理，终于得到“干净”的语料，可以开始训练了。
先放训练代码：

```
# -*- coding: utf-8 -*-
import logging
import multiprocessing
import codecs
from tqdm import tqdm

from gensim.models import Word2Vec
from gensim.models.word2vec import LineSentence

if __name__ == '__main__':

    program = "train_word2vec_model.py"
    logger = logging.getLogger(program)

    logging.basicConfig(format='%(asctime)s: %(levelname)s: %(message)s')
    logging.root.setLevel(level=logging.INFO)
    logger.info("running %s" % ' '.join(program))

    infile = "wiki_jieba.txt"
    vec_outfile1 = "wiki.zh.text.model"
    vec_outfile2 = "wiki.zh.text.vector"
    sentences = LineSentence(infile)

    model = Word2Vec(LineSentence(infile), size=300, window=5, min_count=5,
                     workers=multiprocessing.cpu_count(), iter=100)
    # trim unneeded model memory = use(much) less RAM
    # model.init_sims(replace=True)
    model.save(vec_outfile1)
    model.wv.save_word2vec_format(vec_outfile2, binary=False)
```

最重要的训练语句只要一行：

```
model = Word2Vec(LineSentence(infile), size=300, window=5, min_count=5, workers=multiprocessing.cpu_count(), iter=100)
```

这里用了LineSentence函数，文档中这个函数功效为：

```
Simple format: one sentence = one line; words already preprocessed and separated by whitespace.
```

即表示：这里一句话 = 一行（用换行符分割），每一句话都经过预处理，并且用空白符分隔。

另外，训练参数的意义如下：

- size=300 ：词向量的维度大小为300维；size越大，需要更多的训练数据,但是效果会更好. 推荐值为几十到几百
- window=5 ：滑动窗口大小为5，即当前词的上下文为前后各两个词
- min_count=5 ：词频少于5的词被舍弃掉
- workers=multiprocessing.cpu_count() ：多核并行计算，worker参数只有在安装了Cython后才有效. 没有Cython的话, 只能使用单核.
- iter=100 ：训练迭代次数，即将语料训练词向量100次

此外，还有一些参数设置：

- sg=0 ：用于设置训练算法，默认为0，对应CBOW算法；sg=1，则采用skip-gram算法
- alpha=0.05 ：用于设置学习速率
- seed=1 ：用于随机数发生器，初始化词向量
- max_vocab_size=None ：设置词向量构建期间的RAM限制。如果所有独立单词个数超过这个，则就消除掉其中最不频繁的一个。每一千万个单词需要大约1GB的RAM。设置成None则没有限制。
- sample=0.001 ：高频词汇的随机降采样的配置阈值，默认为1e-3，范围是(0,1e-5)
- hs=0 ：如果设置为0（defaut），使用negative sampling；若为1，使用hierarchica softmax技巧
- negative=5 ：如果>0,则会采用negative samping，用于设置多少个noise words（负例）
- cbow_mean=1 ：如果为1（defaut）则对上下文向量求均值进行训练；如果为0，则采用上下文词向量的和；只有使用CBOW的时候才起作用
- hashfxn=< built-in function hash> ： hash函数来初始化权重，默认使用python的hash函数
- trim_rule=None ：用于设置词汇表的整理规则，指定那些单词要留下，哪些要被删除
- sorted_vocab=1 ：如果为1（default），则在分配word index 的时候会先对单词基于频率降序排序
- batch_words=10000 ：表示每一批的传递给线程的单词的数量，默认为10000

所有参数都设置的训练语句如下所示：

```
model = Word2Vec(LineSentence(infile), size=300, alpha=0.5,window=5, min_count=5, max_vocab_size=None, sample=0.001,seed=1,
workers=multiprocessing.cpu_count(),min_alpha=0.0001, sg=0, hs=0, negative=5, cbow_mean=1, hashfxn=<built-in function
hash>,iter=100,null_word=0, trim_rule=None, sorted_vocab=1, batch_words=10000)
     1
```

5. 测试词向量

训练开始：

```
2017-12-08 12:53:16,033: INFO: collected 2736642 word types from a corpus of 176184328 raw words and 16332181 sentences
2017-12-08 12:53:16,033: INFO: Loading a fresh vocabulary
2017-12-08 12:53:20,811: INFO: min_count=5 retains 673090 unique words (24% of original 2736642, drops 2063552)
2017-12-08 12:53:20,811: INFO: min_count=5 leaves 173070010 word corpus (98% of original 176184328, drops 3114318)
2017-12-08 12:53:22,351: INFO: deleting the raw counts dictionary of 2736642 items
2017-12-08 12:53:22,901: INFO: sample=0.001 downsamples 8 most-common words
2017-12-08 12:53:22,901: INFO: downsampling leaves estimated 165796327 word corpus (95.8% of prior 173070010)
2017-12-08 12:53:22,901: INFO: estimated required memory for 673090 words and 400 dimensions: 2490433000 bytes
2017-12-08 12:53:25,255: INFO: resetting layer weights
2017-12-08 12:53:36,502: INFO: training model with 8 workers on 673090 vocabulary and 400 features, using sg=0 hs=0 sample=0.001
2017-12-08 12:53:37,782: INFO: PROGRESS: at 0.00% examples, 51048 words/s, in_qsize 0, out_qsize 0
2017-12-08 12:53:39,019: INFO: PROGRESS: at 0.00% examples, 48507 words/s, in_qsize 0, out_qsize 0
2017-12-08 12:53:40,121: INFO: PROGRESS: at 0.00% examples, 44369 words/s, in_qsize 0, out_qsize 0
2017-12-08 12:53:41,211: INFO: PROGRESS: at 0.00% examples, 52399 words/s, in_qsize 0, out_qsize 0
```

经过将近10个小时，终于完成了词向量的训练。

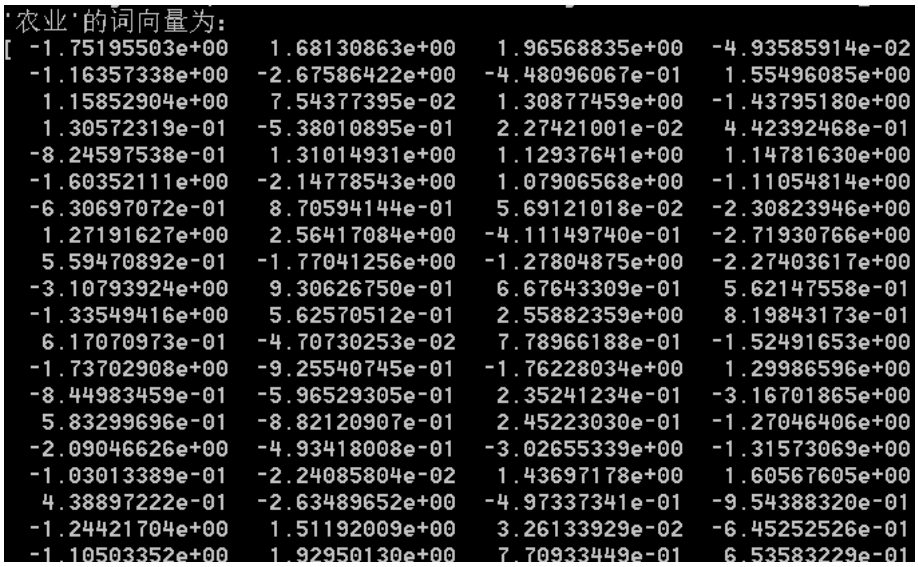
```
2017-12-08 23:39:50,881: INFO: PROGRESS: at 99.99% examples, 427567 words/s, in_qsize 0, out_qsize 0
2017-12-08 23:39:51,881: INFO: PROGRESS: at 100.00% examples, 427568 words/s, in_qsize 0, out_qsize 0
2017-12-08 23:39:52,895: INFO: PROGRESS: at 100.00% examples, 427566 words/s, in_qsize 0, out_qsize 0
2017-12-08 23:39:53,193: INFO: worker thread finished; awaiting finish of 7 more threads
2017-12-08 23:39:53,302: INFO: worker thread finished; awaiting finish of 6 more threads
2017-12-08 23:39:53,302: INFO: worker thread finished; awaiting finish of 5 more threads
2017-12-08 23:39:53,302: INFO: worker thread finished; awaiting finish of 4 more threads
2017-12-08 23:39:53,302: INFO: worker thread finished; awaiting finish of 3 more threads
2017-12-08 23:39:53,302: INFO: worker thread finished; awaiting finish of 2 more threads
2017-12-08 23:39:53,302: INFO: worker thread finished; awaiting finish of 1 more threads
2017-12-08 23:39:53,302: INFO: worker thread finished; awaiting finish of 0 more threads
2017-12-08 23:39:53,302: INFO: training on 17618432800 raw words (16579632329 effective words) took 38776.8s, 427566 effective word
2017-12-08 23:39:53,334: INFO: saving Word2Vec object under wiki.zh.text.model, separately None
2017-12-08 23:39:53,334: INFO: not storing attribute syn0norm
2017-12-08 23:39:53,334: INFO: storing np array 'syn0' to wiki.zh.text.model.wv.syn0.npy
2017-12-08 23:40:04,490: INFO: storing np array 'syn1neg' to wiki.zh.text.model.syn1neg.npy
2017-12-08 23:40:14,959: INFO: not storing attribute cum_table
2017-12-08 23:40:22,849: INFO: saved wiki.zh.text.model
2017-12-08 23:40:22,849: INFO: storing 673090x400 projection weights into wiki.zh.text.vector
```

最终得到两个文件，Gensim中默认格式的词向量文件model，以及原始c版本的词向量文件vector。

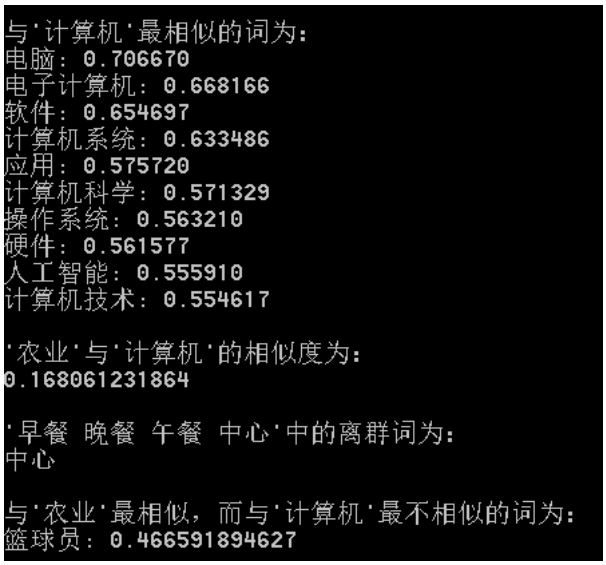
写一个代码测试下效果吧：

```
# encoding:utf8
import gensim

if __name__ == '__main__':

    model = gensim.models.Word2Vec.load('wiki.zh.text.model')
    word1 = u'农业'
    word2 = u'计算机'
    if word1 in model:
        print (u"'%s'的词向量为： " % word1)
        print (model[word1])
    else:
        print (u'单词不在字典中！')

    result = model.most_similar(word2)
    print (u"\n与'%s'最相似的词为： " % word2)
    for e in result:
        print ('%s: %f' % (e[0], e[1]))

    print (u"\n'%s'与'%s'的相似度为： " % (word1, word2))
    print (model.similarity(word1, word2))

    print (u"\n'早餐 晚餐 午餐 中心'中的离群词为： ")
    print (model.doesnt_match(u"早餐 晚餐 午餐 中心".split()))

    print (u"\n与'%s'最相似，而与'%s'最不相似的词为： " % (word1, word2))
    temp = (model.most_similar(positive=[u'篮球'], negative=[u'计算机'], topn=1))
    print ('%s: %s' % (temp[0][0], temp[0][1]))
```

最终结果如下所示：



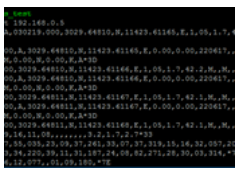
上图为‘农业’的词向量。



上图为一些近义词、离群词等的测试效果。

以上，完成了维基百科中文语料训练词向量的全过程。本文旨在做一个实验记录，也是抛砖引玉，欢迎大家共同探讨！

智能推荐



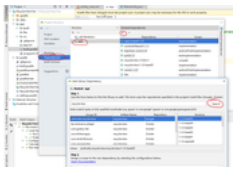
linux下编程epoll实现将GPS定位信息上报到服务器

****

操作系统：CentOS 开发板...

了解wordpress 3.0 数据库表结构（2015年）

数据库 WordPress是使用数据库来实现存储、检索和显示数据功能的。数据库是CMS的最关键的部分，有必要相信，WordPress成功的很大一部分原因就是因为它比较合理的、容易管理的数据库结构。 通过了解数据库的结构，你也可以很容易地解决一些问题。比如，你可以直接通过数据库的操作来实现更改密码、禁用插件、选择主题和做其他事情，而无需访问管理面板。 重要：在进行任何更改之前，请备份您的数据库。 建...



Android sdk 29中导入recyclerview的问题

本人在跟着《第一行代码》（第二版）进行学习当中。进入到学习recyclerview控件时，按照书上的操作，在build.gradle中的dependencies中添加导入语句，如下： 添加compile语句后，提示错误： Version 28 (intended for Android Pie and below) is the last version of the legacy support...

sonarqube最简使用教程-IDE插件版和本地版使用

sonarqube最简使用教程-IDE插件版和本地版使用 简介 一 IDEA插件版 1 下载插件 2 开始使用 4 分析 二 本地搭建版使用 1 在搭建好的sonarqube平台创建账号密码 1 maven设置 2 IDE设置账号密码 3 pom.xml文件配置 3 执行命令![在这里插入图片描述](https://img-blog.csdnimg.cn/20200728163413995.png...

Mongodb的介绍和安装

一：nosql的介绍 二：关系型和非关系型区别 三：mongodb的优势 四：Linux下如何安装 1）使用命令安装 2):源码安装 五：mongodb的启动 1）：服务端启动 2）：客户端启动 MongoDB官方文档：https://docs.mongodb.com/manual/introduction/...