

项目报告

21371036 郝嘉霖

1. 课题背景与意义

在当前的职场环境中，多个行业都呈现出不同程度的招聘需求。信息技术领域的需求持续增加，尤其是软件开发、数据科学和网络安全领域。招聘需求中对于个人能力和需求有着不同的要求，本文的研究主要目的就是详细分析这些能力需求，从未来应聘者的角度看待招聘市场，从人才需求的角度出发，充分利用网络招聘文本信息，研究其中蕴含的对人才的要求。希望为人才培养单位洞悉市场需求现状，并依此改善学科专业建设提供参考。同时有关人才可以根据本文的研究结果，在高校统一的课程安排之下，结合自身特点和职业规划有意识的安排重点学习内容，早做准备。

此外对于不同行业 and 不同职位，我们也对薪资、地点等关键属性进行了分析，了解多个行业的招聘就业需求、薪资水平和地点可以帮助个人做出更明智的职业规划和就业决策。不同的行业和地点提供了多样化的机会，可以根据个人的兴趣和技能来选择最合适的职业路径。

2. 研究方法

2.1 数据采集

在我们的研究中，获取招聘数据的方式是通过网络爬虫，通过python来编写爬虫爬取第三方招聘平台上的招聘数据。

2.1.1 数据来源

在本课题中，获取招聘数据的方式是通过网络爬虫，通过python来编写爬虫爬取第三方招聘平台上的招聘数据，我们主要的数据来源是智联招聘、前程无忧、猎聘等网站，辅以华为，百度等企业的招聘数据。我们从10月-11月持续从对应网站上爬取数据，最终在智联上获取到近5k条数据，前程无忧上获取到1w条左右数据，猎聘上获取到2w条数据，百度、华为等网站综合获取到5k条数据左右，总数据量在近4w条。

我国的第三方招聘平台数量众多，种类多样。目前，我国的第三方招聘平台主要有综合招聘和垂直招聘两种主要模式。最早出现的是综合模式，这类平台覆盖多个行业和地理区域，具有很强的综合竞争力。前程无忧是这类平台的典型代表。而垂直招聘网站则更具针对性，专注于特定地域或领域，因此岗位匹配度较高。此外，一些分类信息网站主要面向蓝领人群，并融入了社交元素。最新的招聘模式将求职培训与岗位推荐相结合，通常规模较小但专业性很强，例如牛客网。

为了获取到更加多元化的招聘数据，我们主要采用了综合招聘网站例如智联招聘、前程无忧和猎聘等，辅以垂直招聘网站特别是一些企业的官方招聘网站例如华为、百度等，这些大型企业的招聘通常比较有针对性。

2.1.2 网络爬虫介绍

概述

爬虫是一种用于获取到网络数据的技术，由于高速发展的互联网使得很多信息都出现在了网络上，所以诞生了爬虫，爬虫可以模拟人类点击网站，访问内容并获取到页面内的全部数据，然后通过进一步的对于页面内的数据进行解析就可以提取到数据，最后对于数据进行处理就可以保存在数据库内。因此整体流程可以被写作如下：

1. 模拟人类点击发送请求

2. 获取整体网页数据
3. 解析数据，获取需要的部分
4. 处理数据，保存进入数据库

而想要爬取的目的网络也已经上面确定了，就是智联、前程无忧、猎聘、华为、百度等招聘网站，我们只需要分析页面结构，按照指定的方法对其中的数据爬取、解析、保存即可。

实现

本研究中使用python的request库和selenium包，前者可以模拟浏览器给对应的网站发送请求，后者则可以模拟人类行为点击和移动界面中的元素，由于现在的网站存在较为复杂的反爬手段，像以前一样单纯的使用request发送请求无法获取到有效的数据，想要获取到目标数据有时候会经过“机器人验证”过程，也就是拖动滑块，这时候就需要selenium来完成目标元素的移动，selenium可以完成反爬验证。

通过selenium和request的有机结合，很快就能通过验证并爬取到对应的数据，最终获取到了约3.7w条数据。



图2-1 招聘网站示例

2.2 数据处理

在我们的研究中，除去获取数据，还需要对数据进行筛选和处理，通过python来完成数据清洗的过程

2.2.1 数据清洗

爬取到的数据中，由于发布方的问题，可能有一些数据中有一些字段不存在或者模糊不清、格式混乱，这样的数据在分析过程中会导致程序无法解析出现错误，所以首先要被剔除。因此数据首先需要被清洗一遍，对于一些无法被解析的字段需要清洗筛选掉。

我们使用了python的pandas包进行数据处理，pandas包是一个强大的数据处理包，对于处理二维表格等有着非常高的效率，通过pandas包中自带的函数如 `dropna()` 等可以很快的清洗掉不合法的数据

爬取到的数据中，由于许多第三方的招聘网站对于格式要求非常不严谨，所以各个招聘信息的格式五花八门，在清洗掉不合格的数据之后，还需要对全部数据进行格式化处理，让他们有着相同的格式，便于后续的工作和分析。例如：在招聘数据中给出的薪酬字段，一些数据会写：“10k/月”，另一些数据会写：“120元一天”。所有的薪酬字段为了便捷最终会被格式化以K/月为单位的字段。

我们仍使用pandas包，对于不同列上的数据进行不同的处理，根据其中的关键词来判断是以日、月、年作为单位，并识别其中的数字字段，按照换算关系换算成统一的单位。

对于地区，我们构建了一个区域-省-市的关系表，然后根据招聘数据给出的具体地区，通过关系表就能很快地找到区域和省、市，获取到这三个字段。

经过数据清洗之后，我们仍保留了近2.7w条招聘数据。

大类专业	大类岗位	岗位名称	薪酬	地点-区域	地点-省	地点-市	学历要求	经验要求	企业名称	企业规模	岗位详情
软件工程	前后端	前端开发主	7.0K/月	华东	江苏省	南京	本科	3-5年	瑞仪光电	1000-9999	任职要求：1. 本科学历，计算机、软件等相关专业，良好的英文能力 2. 熟练使用c#语言，掌握Winform、MVC、WebService开发技术3. 熟悉SQL语言，ORM框架，掌握SQLServer等主流数据库开发技术4. 熟悉前端语言，掌握HTML、CSS、Javascript技术，熟悉jQuery、echarts等常用前端库福利待遇：工资待遇：另加绩效奖金、年终奖金等，如有加班按法律规定进行结算福利政策：提供五险一金，提供食宿休假制度：享有双休，法定假日，婚、丧假及产假，法定带薪年假等教育培训：提供良好的教育训练及在职进修体系
软件工程	前后端	前端开发工	13.0K/月	华东	上海市	上海	本科	3-5年	国以贤智能	100-499人	岗位职责1、负责前端界面构建，各类交互设计实现等;2、完成产品(网页端)和管理平台的前端开发，编写可复用的用户界面组件;3、持续优化前端体验和页面响应速度，并保证兼容性和执行效率;4、和后端工程师紧密配合，保证产品的用户体验及稳定性;5、负责项目的数据交互等业务，独立完成项目并解决遇到的技术问题; 6、具有较强的性能优化意识并发挥积极作用。任职资格：1、本科及以上学历，3年及以上前端开发经验; 2、精通easyui、JavaScript、html、css、js等前端开发技术; 3、熟悉ES6标准、主流的JS库和开发框架，例如:Node.js、Vue、Webpack且有实际项目经验; 4、工作态度积极主动，认真负责岗位职责。

图2-2 处理后的数据示例

2.2.2 数据分词

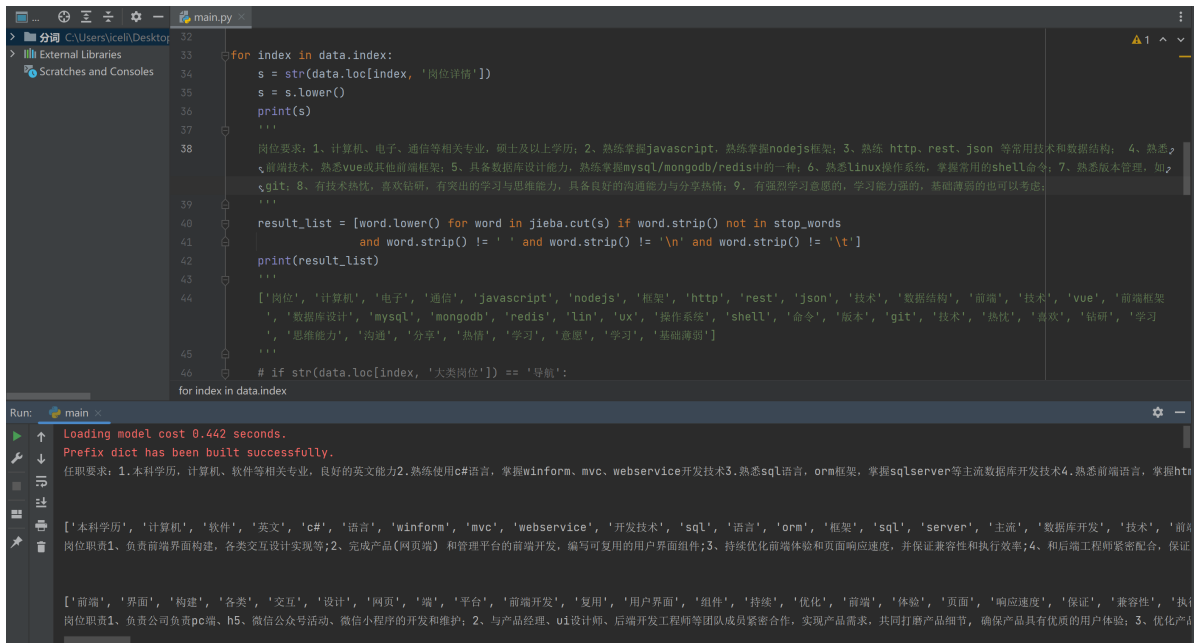
对于数据中“岗位详情”这一个字段，发布方会发布岗位的任职资格、任职资历以及福利等信息，我们的目的是研究具体的任职需求，因此为了删除所有的噪音文字例如“的”、“和”等与需求无关的无意义的字还需要对其进行筛选。此外为了后续的文本向量化，除了筛选无意义的词语还需要把一段文字划分为一个个的单词，这样才能够对这段话进行数据分析，所以这里需要一次分词。

我们使用了jieba包进行分词，由于jieba包虽然自带大型字典，但其中都是常用词，对于招聘信息这样有着大量专业词汇的特殊文本，需要给词典中加入一些专业词汇，否则会出现分词错误的情况，我们通过大语言模型生成了目标行业内主要的专业词汇并构造词典dict.txt，搭配jieba包中的一般词汇一同作为词典进行分词。

```
# 掌握机器学习算法，熟练使用python、计算机相关专业
'''
'''
添加自定义字典之前：['掌握', '机器', '学习', '算法', '熟练', '使用', 'python', '计算机', '相关', '专业']
'''
'''
添加自定义字典之后：['掌握', '机器学习算法', '熟练', '使用', 'python', '计算机相关专业']
'''
```

图2-3 分词示例

此外对于常见的停用词如“的”、“和”等，我们在csdn上找到了常见的中英文停用词，构造了一个停用词字典，在此基础上我们还添加了一些常见的对于需求没有意义的动词和形容词，例如“至少”，“非常”，“强”，“有着”等无意义的词语，旨在尽可能保留和专业需求相关的名词性词语，依次构造了stopwords.txt，在分词的基础上剔除了上述这一类无效的词语。



```
32
33 for index in data.index:
34     s = str(data.loc[index, '岗位详情'])
35     s = s.lower()
36     print(s)
37
38     岗位要求: 1. 计算机、电子、通信等相关专业, 硕士及以上学历; 2. 熟练掌握javascript, 熟练掌握nodejs框架; 3. 熟练 http, rest, json 等常用技术和数据结构; 4. 熟悉
39     前端技术, 熟悉vue或其他前端框架; 5. 具备数据库设计能力, 熟练掌握mysql/mongodb/redis中的一种; 6. 熟悉linux操作系统, 掌握常用的shell命令; 7. 熟悉版本管理, 如
40     git; 8. 有技术热忱, 喜欢钻研, 有突出的学习与思维能力, 具备良好的沟通能力与分享热情; 9. 有强烈学习意愿的, 学习能力强的, 基础薄弱的也可以考虑;
41
42     result_list = [word.lower() for word in jieba.cut(s) if word.strip() not in stop_words
43                    and word.strip() != ' ' and word.strip() != '\n' and word.strip() != '\t']
44     print(result_list)
45
46     ['岗位', '计算机', '电子', '通信', 'javascript', 'nodejs', '框架', 'http', 'rest', 'json', '技术', '数据结构', '前端', '技术', 'vue', '前端框架',
47      '数据库设计', 'mysql', 'mongodb', 'redis', 'lin', 'ux', '操作系统', 'shell', '命令', '版本', 'git', '技术', '热忱', '喜欢', '钻研', '学习',
48      '思维能力', '沟通', '分享', '热情', '学习', '意愿', '学习', '基础薄弱']
49
50     # if str(data.loc[index, '大类岗位']) == '导航':
51     for index in data.index
```

图2-4 分词结果示例

2.3 数据分析

这是研究过程中最重要的一个部分，就是如何对这些招聘数据进行处理，获取到不同类别的不同数据。我们的目标是对同一个专业下的所有招聘数据进行分类，根据全部招聘数据的“岗位详情”描述字段，对招聘主题进行分类，这样就可以获取到该专业下有哪些主要的招聘方向，例如“软件工程”下往往有“后端”，“前端”等分类。分类完成后再分析不同招聘主题下的薪资、地区等数据，得到最终的结果并展示。

2.3.1 LDA模型

概述

Latent Dirichlet Allocation (LDA) 是一种用于文本分析和主题建模的机器学习模型。它的基本思想是：每篇文档都由多个主题组成，每个主题又由一组单词组成。LDA的目标是从给定的文档集合中推断出主题和每个文档与主题的关系。

LDA的工作过程包括以下步骤：

1. 初始化：为每个文档中的每个单词分配一个随机的主题。
2. 迭代更新：在多次迭代中，LDA根据文档中的单词分布和主题分布来更新主题的分配，直到达到稳定状态。
3. 输出结果：最终，LDA输出每个文档的主题分布以及每个主题的单词分布。

LDA的应用包括文本分类、信息检索、主题发现等领域。它有助于理解文本数据中的潜在结构，从而使文本数据更容易分析和利用。通过LDA，我们可以识别文本中的关键主题，为信息检索和知识管理提供有力支持。

确定困惑度

我们使用了python的gensim包里的IdaModel来实现我们的LDA模型，首先我们根据LDA模型对某一个专业下的所有招聘数据的主题数量进行确定，对于所有招聘数据的“岗位详情”字段，由于招聘数据本身的混杂性和数据量较为庞大，人为确定主题数量是不现实的，所以我们通过LDA的困惑度来确定主题数量，这样的结果可以更加准确。

我们首先对主题的范围进行一个确定，让LDA从1-35个主题上，以5为步长进行迭代计算，从而确定主题从1-35之间的最佳区间，结果如下：

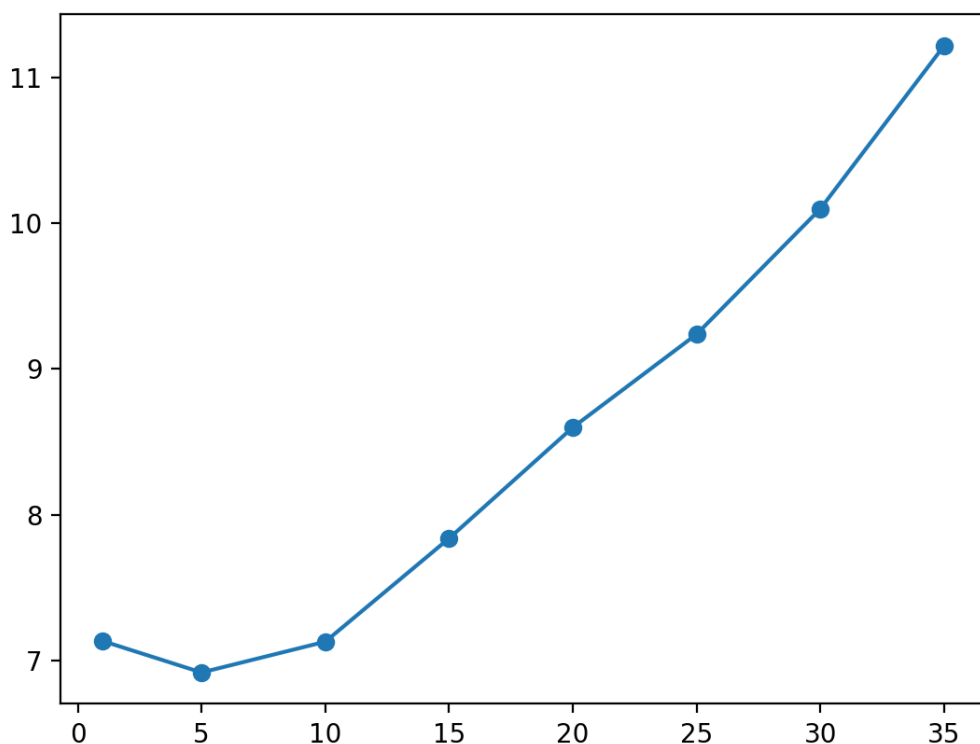


图2-5 困惑度大小示例

可以看到，主题数过多导致模型反而无法集中注意力，困惑度太高，最佳的主题数量在1-10之间，因此我们之后将会遍历1-10，找到最佳的主题数量，并以此来训练LDA模型。

训练LDA模型

在遍历和确定主题数量之后，我们使用LDA模型按照主题数量来获取到具体的主题内容，LDA的迭代次数我们选择了2次，过少的迭代次数会导致欠拟合，而过多的迭代次数会导致过拟合，根据我们自己的数据量，最终选择了这个迭代次数。

接下来，我们分专业进行分析，对于某一个专业下的所有招聘信息，我们把“岗位详情”在分词结果中处理得到的向量通过 `doc2bow` 方法根据词典转化为向量，这样我们就得到了所有招聘信息的向量化结果，第*i*的位置上的元素是*n*就表示第*i*个词在这个招聘文本中出现了*n*次，然后我们把全部向量传递给LDA模型，通过困惑度确定好的主题数量以及设定好的迭代次数进行迭代，最终完成训练。

```
Project C:\Users\ice\I Desktop main.py
LDA模型 C:\Users\ice\I Desktop
main.py
stopwords.txt
External Libraries
Scratches and Consoles

34 # 构建词袋模型
35 for file_path in file_paths:
36     with open(file_path, 'rb') as file:
37         result = pickle.load(file)
38         dictionary = gensim.corpora.Dictionary(result)
39         corpus = [dictionary.doc2bow(doc) for doc in result]
40     # 比较困惑度
41     print(file_path)
42     topic_num = min_perplexity(dictionary, corpus)
43     # 获取LDA模型
44     lda_model = gensim.models.LdaModel(corpus, num_topics=topic_num, id2word=dictionary, passes=pass_num)
45     _topics = lda_model.show_topics(num_words=50) # 获取到主题及其词语分布
46     write_string = ''
47     for topic_id, topic_words in _topics:
48         write_string += f'主题: {topic_id} 词项分布: {topic_words} \n'
49         print(f'主题: {topic_id} 词项分布: {topic_words}')
50     write_path = '../主题结果/' + file_path.lstrip('../').rstrip('.pkl') + '.txt'
51     with open(write_path, 'w', encoding='utf-8') as f:
52         f.write(write_string)

Run: main
../人工智能.pkl
主题数量: 1 困惑度: -6.989112865578985
主题数量: 2 困惑度: -6.911824571525198
主题数量: 3 困惑度: -6.881808361465027
主题数量: 4 困惑度: -6.984498167869595
主题数量: 5 困惑度: -6.885646683033647
主题数量: 6 困惑度: -6.915549340874364
主题数量: 7 困惑度: -6.980856493161604
3
主题: 0 词项分布: 0.015*"模型" + 0.015*"算法" + 0.013*"深度学习" + 0.012*"经验" + 0.011*"技术" + 0.009*"机器学习" + 0.008*"分析" + 0.008*"数据" + 0.007*"沟通" + 0.006*"python" +
主题: 1 词项分布: 0.035*"算法" + 0.022*"深度学习" + 0.019*"经验" + 0.018*"技术" + 0.018*"模型" + 0.011*"设计" + 0.010*"研究" + 0.010*"优化" + 0.010*"ai" + 0.008*"项目" + 0.008*"
主题: 2 词项分布: 0.032*"算法" + 0.026*"经验" + 0.025*"模型" + 0.018*"深度学习" + 0.015*"优化" + 0.012*"神经网络" + 0.011*"机器学习" + 0.009*"项目" + 0.008*"技术" + 0.008*"研究" +
```

图2-6 LDA训练结果示例

训练结束后我们获取到了每个专业下有几个主题以及每个主题的词项分布，接下来按照这些主题进行聚类。

2.3.2 TF-IDF词袋模型

文本向量化仅仅用于LDA模型的训练，对于大规模的招聘数据，除了频率向量化，更好的办法是基于TF-IDF词袋模型的word2vec向量化，在频率的基础上，TF-IDF还添加了逆文档频率，使得向量更加特征化，对于聚类等都有着更好的效果帮助。

我们在这里使用了python中gensim包里的TF-IDF的模型实现word2vec，得到了每个招聘数据最后的向量表示，用于后面的PCA和kmeans处理。

```
# 获取TF-IDF模型
tf_idf_model = gensim.models.TfidfModel(corpus)
tf_idfs = tf_idf_model[corpus]
dict_num = len(topics[0]) # 所有主题的语料库中词语是一样的，选择第一个主题计算即可
vectors = [] # vectors数组是一个普通数组，里面的每一项都是一个文章的tf-idf词项分布
for tf_idf in tf_idfs:
    result_list = [0] * dict_num
    for (index, num) in tf_idf:
        result_list[index] = num
    vectors.append(result_list)
```

图2-7 TF-IDF模型示例

2.3.3 PCA降维

概述

主成分分析（Principal Component Analysis, PCA）是一种常用的数据降维技术，旨在通过线性变换将高维数据集转换为低维数据集，同时最大程度地保留原始数据的信息。PCA的目标是找到一个新的坐标系，其中数据的方差最大，从而减少数据的冗余性。

PCA的基本步骤如下：

1. 数据标准化：首先，将原始数据进行标准化，以确保所有特征具有相同的尺度。这可以通过减去每个特征的平均值并除以标准差来实现。

2. 计算协方差矩阵：接下来，计算标准化后的数据集的协方差矩阵。协方差矩阵描述了不同特征之间的相关性和方差。
3. 计算特征值和特征向量：对协方差矩阵进行特征值分解，得到特征值和对应的特征向量。特征值表示了新坐标系中每个主成分（主特征）的方差大小，而特征向量则是每个主成分的方向。
4. 选择主成分：根据特征值的大小，选择要保留的主成分数量。通常，可以根据特征值的比例来确定要保留的主成分数量，以保留足够的数据信息。
5. 投影数据：将原始数据投影到选定的主成分上，从而将高维数据转换为低维数据。

通过这些步骤，PCA可以用来降维，减少数据的维度，同时保留数据的主要信息，以便更容易进行数据分析、可视化和建模。它在数据压缩、特征选择、图像处理等领域都有广泛的应用。

实现

由于招聘文本通常都比较短，而招聘文本的总数量比较大，所以招聘文本向量化之后是一个稀疏向量，其中包含了很多的0，所以一般的PCA效果并不是太好，我们在这里使用了python中sklearn包里的Sparse PCA降维，它是PCA的一个变种，更适合用于处理稀疏向量的降维，其本质上的计算原理和PCA非常相似，只是额外增加了一些和稀疏向量处理相关的方法。

除此之外，由于招聘文本中总是包含了一些共同的文字特征，例如对于岗位的通用要求“负责”，“经验”，“理解”等词汇，大部分的招聘文本都包含了这些词语，因此如果直接进行聚类将会不可避免地导致聚类结果混乱，因为聚类会按照这些共同的通用词汇进行分类，而不是按照每条招聘信息里特有的需求词汇进行分类。

使用PCA降维的原因是原本的招聘文本向量维度比较大，取决于其词典中词语的数量，通常在6000-10000维，并且包含了大量的0，所以在聚类之前必须要用PCA进行降维之后提取出关键的特征之后在进行聚类，这样的方法可以让聚类结果更准确并且效率更高，如果不降维聚类的运行速度将会非常慢甚至不可行。

另外降维的最终维度也是一个超参数，需要手动设定，使用SPCA中的 `explain_variance_ratio` 能够直接查看到最终的降维结果效果，这个结果给出了降维后的各个维度上的评分，我们只需要选择前n个评分较高的维度即可，对于评分很低的维度可以直接舍弃，因为这种维度对于特征化降维的作用微乎其微。

```
[0.02245145 0.01782119 0.01199621 0.01021026 0.00931065 0.00895806
 0.00863744 0.00835335 0.00797295 0.00741988 0.00709579 0.00689983
 0.00629303 0.00622933 0.00603443 0.00590527 0.00585204 0.00557854
 0.00554798 0.00550415 0.00538363 0.00532848 0.00521559 0.00516259
 0.00503562 0.00490761 0.00484803 0.00474119 0.00472062 0.0046949
 0.0045311 0.00440439 0.00438909 0.00429429 0.00424068 0.00414804
 0.00404866 0.00400788 0.00398648 0.00391229 0.00388553 0.00386005
 0.00384555 0.00377379 0.00372628 0.00362851 0.00357396 0.00351802
 0.00350315 0.00349149 0.00344885 0.00341039 0.00335284 0.00332152
 0.0032823 0.0032683 0.00325664 0.0031988 0.00316569 0.0031006
 0.00308182 0.00303776 0.00300036 0.0029545 ]
```

图2-8 前64维评分示例

最终我们选择了64维作为降维目标维度，使用 `fit` 和 `transform` 方法完成PCA降维。

2.3.4 kmeans聚类

概述

K均值聚类（K-means clustering）是一种常用的无监督学习算法，用于将数据集中的样本划分成K个不同的簇或群组。其基本原理是通过迭代的方式，将数据点分配到K个簇中，使得每个数据点与其所属簇的中心点之间的距离最小化。

K均值聚类的工作流程如下：

- 1. 随机选择K个初始簇中心点。
- 2. 将每个数据点分配到离其最近的簇中心点。
- 3. 计算每个簇的新中心点，即簇内所有数据点的平均值。
- 4. 重复第2和第3步，直到簇中心点不再发生显著变化，或达到预定的迭代次数。

K均值聚类的优点包括简单、高效、易于理解和实现。然而，它对初始簇中心点的选择非常敏感，可能会陷入局部最优解，因此通常需要多次运行算法，以选择最优的聚类结果。此外，K值的选择也需要一定的领域知识或通过一些启发式方法来确定。总之，K均值聚类是一种强大的聚类算法，适用于各种数据分析和模式识别任务。

实现

我们在之前获取了主题的数量以及每个主题的词项分布，并且还通过PCA获取到了每个文章的向量，由于LDA模型虽然能够给出主题，但是不能够给出某一篇文章属于哪个主题，因此我们需要聚类来根据主题的词项分布向量，把招聘信息的向量通过kmeans聚类向其靠拢，这样能够更好的把招聘向量根据主题完成聚类。

我们使用了python中sklearn里的kmeans包完成了kmeans聚类分组，通过降维后的向量完成聚类，最终获取到了每一个招聘信息所在的主题分类，并添加到csv文件里，至此我们就完成了对于招聘数据的分类分析。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	id	大类专业	大类岗位	岗位名称	薪酬	地点-区域	地点-省	地点-市	学历要求	经验要求	企业名称	企业规模	岗位详情	分组
2	1	1 软件工程	前后端	前端开发主7.0K/月	华东	江苏省	南京	本科	3-5年	瑞仪光电	(1000-9999	任职要求：1. 本科学历，计算机、软件等相关专业，良好的英文能力2. 熟练使用c#语言，掌握Winform、MVC、WebService开发技术3. 熟悉SQL语言，ORM框架，掌握SQLServer等主流数据库开发技术4. 熟悉前端语言，掌握HTML、CSS、Javascript技术，熟悉jQuery、echarts等常用前端库福利待遇：工资待遇：另加绩效奖金、年终奖等，如有加班按法律规定进行结算福利政策：提供五险一金，提供食宿休假制度：享有双休，法定假日，婚、丧假及产假，法定带薪年假等教育培训：提供良好的教育训练及在职进修体系	1	
3	2	2 软件工程	前后端	前端开发工13.0K/月	华东	上海市	上海	本科	3-5年	国以贤智能	100-499人	岗位职责：1、负责前端界面构建，各类交互设计实现等；2、完成产品（网页端）和管理平台的前端开发，编写可复用的用户界面组件；3、持续优化前端体验和页面响应速度，并保证兼容性和执行效率；4、和后端工程师紧密配合，保证产品的用户体验及稳定性；5、负责项目的数据交互等业务，独立完成项目并解决遇到的技术问题；6、具有较强的性能优化意识并发挥积极作用。任职资格：1、本科及以上学历，3年及以上前端开发经验；2、精通easyui、JavaScript、html、css、js等前端开发技术；3、熟悉ES6标准、主流的JS库和开发框架，例如:Node.js、Vue、Webpack且有实际项目经验；4、工作态度积极主动，认真负责岗位职责。	2	

图2-9 聚类分组结果示例

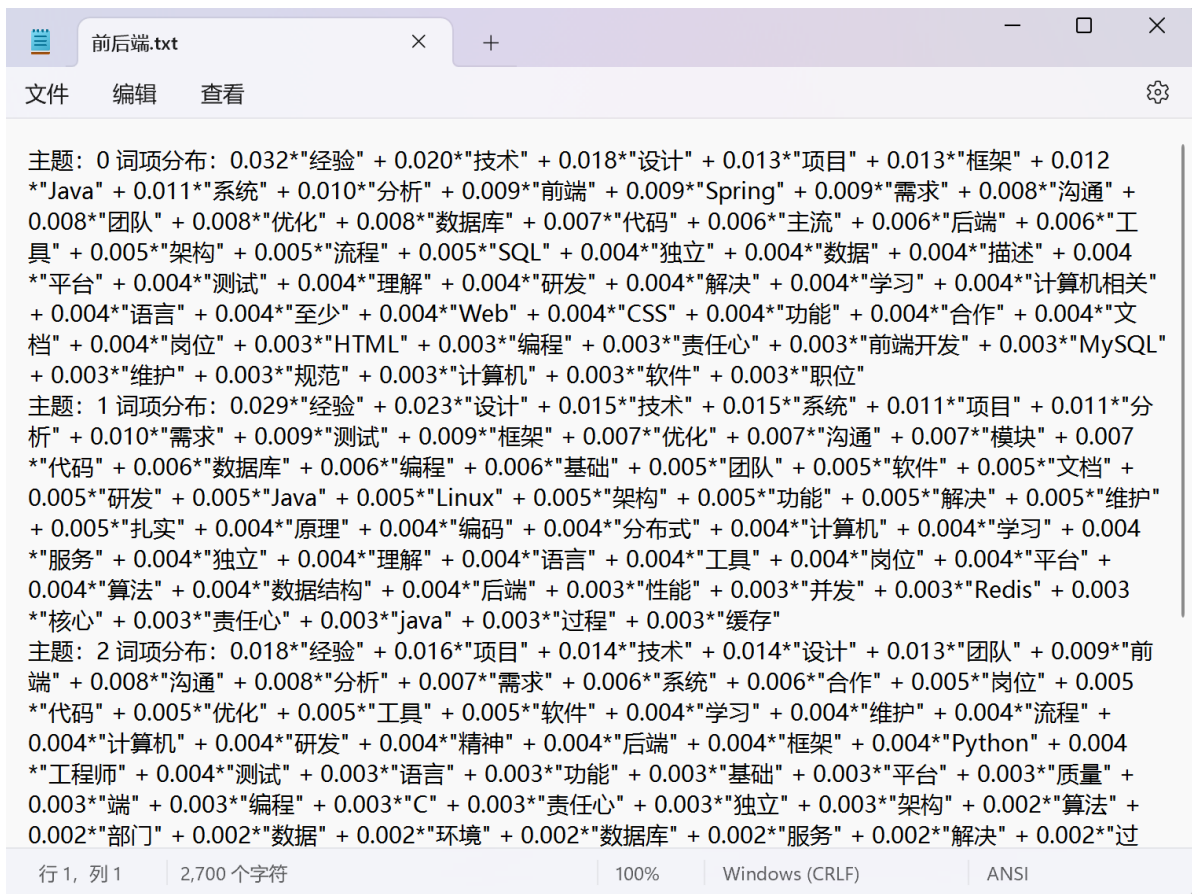


图2-10 主题词项分布示例

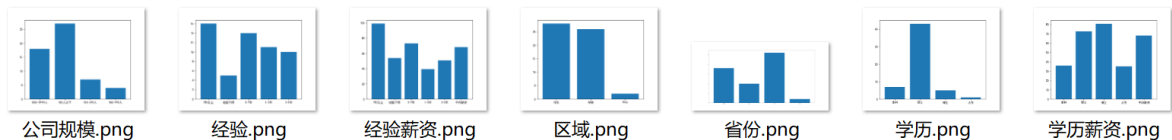
3. 研究结果

每一个专业都会通过LDA和kmeans被聚类称为多个类，我们将会根据每个类的词项分布和对应的所有的招聘信息的岗位名称、岗位详情来给每个类取一个合理的名字。

在这里我们的数据展示图中给出了一个“经验不限”的字段，通常招聘者也会在招聘信息里看到“经验不限”，但其通常不是真的经验不限，只是为了增加应聘者数量，背后有着自己的筛选规则。通过我们的研究和数据展示，可以直观地给招聘者展示“经验不限”一般对应了什么段位的薪资，通过和其他经验的薪资的对比，招聘者可以大致了解到所谓的“经验不限”背后的潜规则。

3.1 数据统计结果

我们把最终的聚类结果按照类别来计算薪酬、学历、经验、地区和企业规模并统计对应的数量，最后使用了python的pyplot构建并生成了png图像并保存在文件夹内，文件夹内包含了对应的png图像。



每一个专业下的每一个聚类都建立了对应的文件夹并保存了我们得到的统计量，生成了柱状图来更直观的反应我们的研究结果，通过对比词项分布和对应的柱状图就可以对比不同专业的区别以及每一个专业内部之间的区别。

3.1.1 统计结果示例

下面举一些具体的例子来展示成果：

前后端通过LDA和kmeans被聚成了7类

1. python开发

词项分布： 0.017 经验 / 0.015 数据 / 0.015 python / 0.012 技术 / 0.011 算法 / 0.010 平台 / 0.008 ai / 0.008 模型 / 0.008 云 / 0.007 研发 / 0.007 工具 / 0.007 工程师 / 0.007 团队 / 0.007 运维 / 0.006 系统 / 0.006 机器学习 / 0.005 服务 / 0.005 提升 / 0.005 职位 / 0.005 设计 / 0.005 沟通 / 0.004 分析 / 0.004 研究 / 0.004 优化 / 0.004 项目 / 0.004 建设 / 0.004 语言 / 0.004 人工智能 / 0.004 描述 / 0.004 爬虫 / 0.004 ic / 0.003 方向 / 0.003 计算机 / 0.003 深度学习 / 0.003 golang / 0.003 框架 / 0.003 学习 / 0.003 自动化 / 0.003 需求 / 0.003 效率 / 0.003 容器 / 0.003 企业 / 0.003 解决方案 / 0.003 数据分析 / 0.003 生产 / 0.003 k8s / 0.003 网络 / 0.003 岗位 / 0.002 大数据 / 0.002 广告

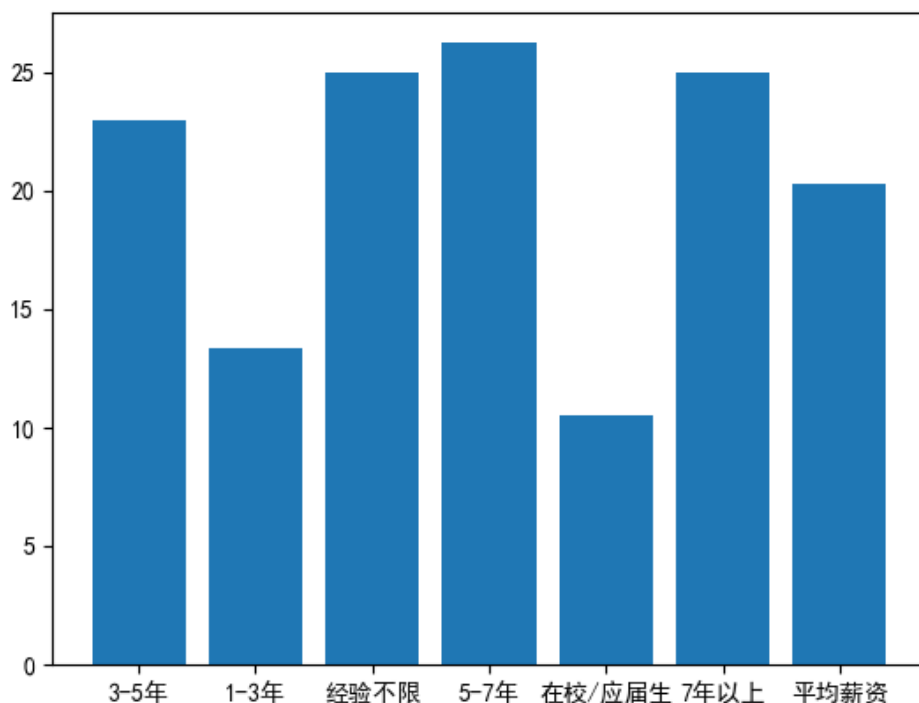


图3-1-1 python开发岗位的经验薪资

2. Java开发

词项分布： 0.034 java / 0.032 经验 / 0.020 项目 / 0.019 设计 / 0.019 技术 / 0.017 spring / 0.015 系统 / 0.014 框架 / 0.013 需求 / 0.013 数据库 / 0.012 工程师 / 0.010 分析 / 0.009 测试 / 0.009 sql / 0.009 沟通 / 0.009 软件 / 0.009 代码 / 0.009 文档 / 0.009 mysql / 0.007 团队 / 0.007 独立 / 0.006 软件开发 / 0.006 模块 / 0.006 mybatis / 0.006 cloud / 0.006 主流 / 0.006 acl / 0.005 架构 / 0.005 编程 / 0.005 维护 / 0.005 优化 / 0.005 语言 / 0.005 解决 / 0.004 springboot / 0.004 功能 / 0.004 计算机 / 0.004 mvc / 0.004 redis / 0.004 过程 / 0.004 学习 / 0.004 合作 / 0.004 责任心 / 0.004 编码 / 0.004 工具 / 0.004 计算机相关 / 0.004 开源 / 0.003 linux / 0.003 规范 / 0.003 研发

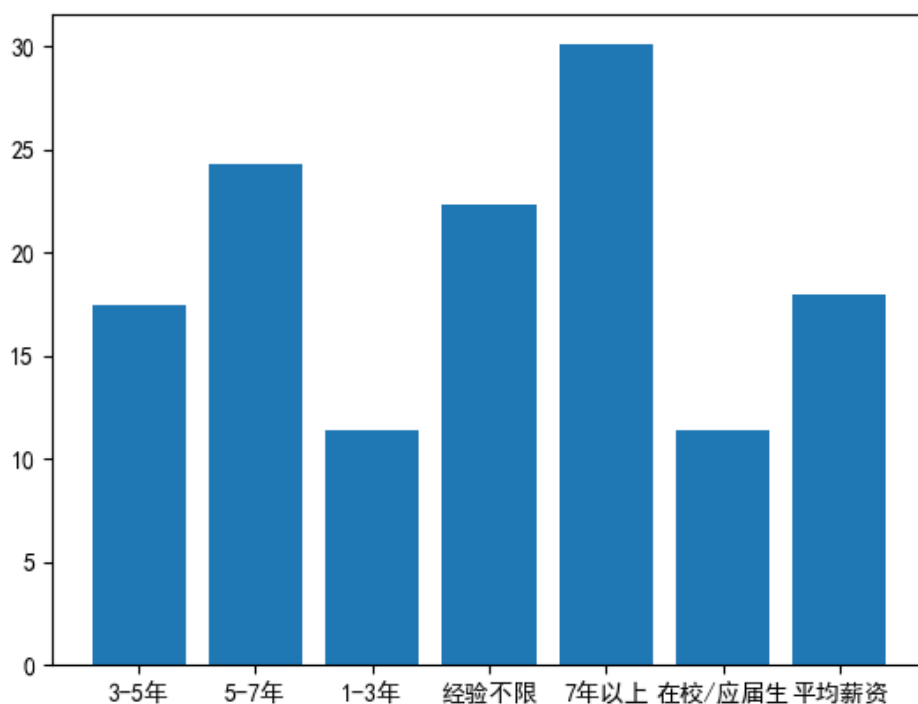


图3-1-2 java开发岗位的经验薪资示例

导航技术通过LDA和Kmeans被聚成了4类：

1. SLAM算法工程师：

词项分布：0.092"slam" + 0.063"算法" + 0.031"经验" + 0.022"优化" + 0.021"传感器" + 0.021"机器人" + 0.017"视觉" + 0.016"激光" + 0.012"工程师" + 0.012"研发" + 0.012"数学" + 0.011"开源" + 0.011"ar" + 0.010"ros" + 0.010"地图" + 0.009"框架" + 0.008"研究" + 0.008"imu" + 0.008"c++" + 0.007"定位" + 0.007"导航" + 0.007"计算机" + 0.006"点云" + 0.006"tographer" + 0.006"场景" + 0.006"激光雷达" + 0.005"电子" + 0.005"项目" + 0.005"环境" + 0.005"规划" + 0.005"路径" + 0.005"相机" + 0.005"编程" + 0.005"构建" + 0.004"技术" + 0.004"产品化" + 0.004"深度" + 0.004"非线性" + 0.004"移植" + 0.004"lin" + 0.004"ux" + 0.004"理论" + 0.004"系统" + 0.004"滤波" + 0.004"功底" + 0.004"gmapping" + 0.004"通信" + 0.004"设计" + 0.004"三维" + 0.004"前沿"

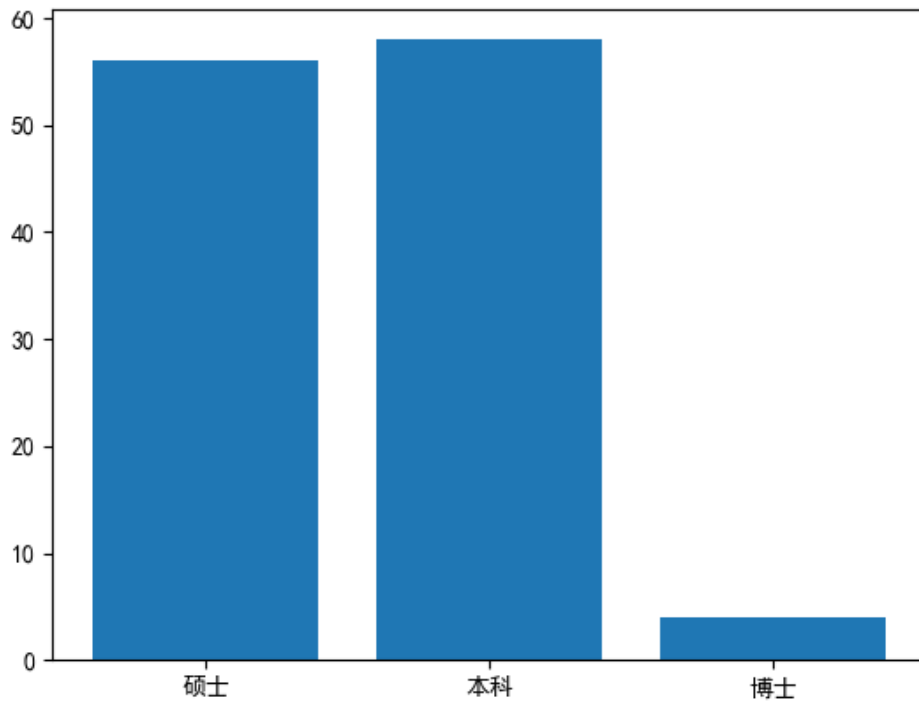


图3-1-3 slam算法工程师对学历的要求

2. 导航定位工程师:

词项分布: 0.045"算法" + 0.029"定位" + 0.023"slam" + 0.022"经验" + 0.014"传感器" + 0.014"技术" + 0.013"系统" + 0.013"导航" + 0.012"设计" + 0.009"工程师" + 0.009"项目" + 0.009"卫星导航" + 0.008"研究" + 0.008"视觉" + 0.007"机器人" + 0.007"优化" + 0.007"gnss" + 0.007"c++" + 0.006"测试" + 0.006"组合" + 0.006"研发" + 0.006"通信" + 0.006"imu" + 0.005"原理" + 0.005"仿真" + 0.005"沟通" + 0.005"理论" + 0.005"地图" + 0.005"分析" + 0.004"高精度" + 0.004"卫星" + 0.004"软件" + 0.004"编程" + 0.004"电子" + 0.004"团队" + 0.004"标定" + 0.004"计算机" + 0.004"调试" + 0.004"激光" + 0.004"数学" + 0.004"自动驾驶" + 0.004"建图" + 0.004"工程" + 0.004"方案" + 0.004"自动化" + 0.003"ros" + 0.003"岗位" + 0.003"rtk" + 0.003"规划" + 0.003"描述"

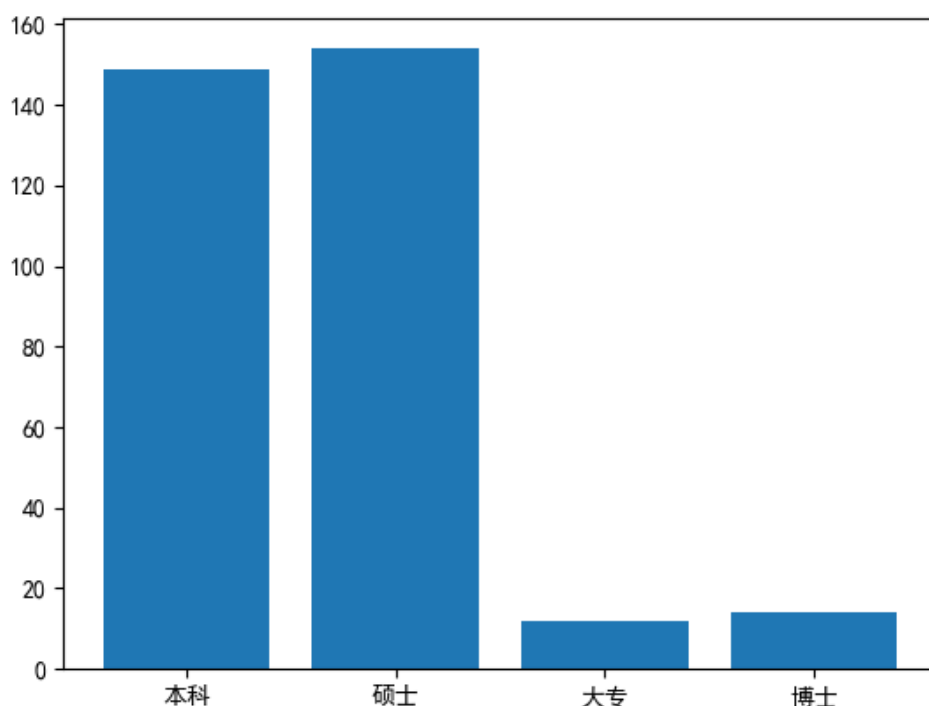


图3-1-4 导航定位算法对学历的要求

3.1.2 统计结果分析

统计结果发现，大部分的工作都分布在华东、华南和华北，部分通信和电气控制等工作在华中地区也有分布。对于计算机、软件和网络等工作主要就业地区在超一线城市。

对于经验的要求，计算机、软件和网络专业通常对1-3年和3-5年的人才需求较多，而通信、集成电路、电子信息等对于3-5年和5-7年的人才需求较多。

学历方面，软件工程主要的要求是本科学历即可，其他专业对于硕士和本科生都有要求，导航定位、slam等特殊的电子信息以及集成电路专业以及高新人工智能行业对于硕博的要求比较高。

3.2 词项分布结果

在我们所有研究的专业里，包括了软件工程、电子科学与技术、电子信息工程、计算机科学、通信工程、网络工程、集成电路、数据分析这几类专业。

我们在研究得到词项分布的基础上绘制了关键词的节点图，过程中剔去了所有英文专业能力词项，这样保留下来的部分更加接近于非专业能力词项。我们使用了python内的networkx包来构建关键词节点分布图，类似于**共现网络**，关键词节点图可以更好地呈现专业能力词项和非专业能力词项，并且方便在专业内部进行对比。接下来我们选择几个比较典型的示例进行展示：

3.2.1 词项分布结果展现

1. 软件工程：

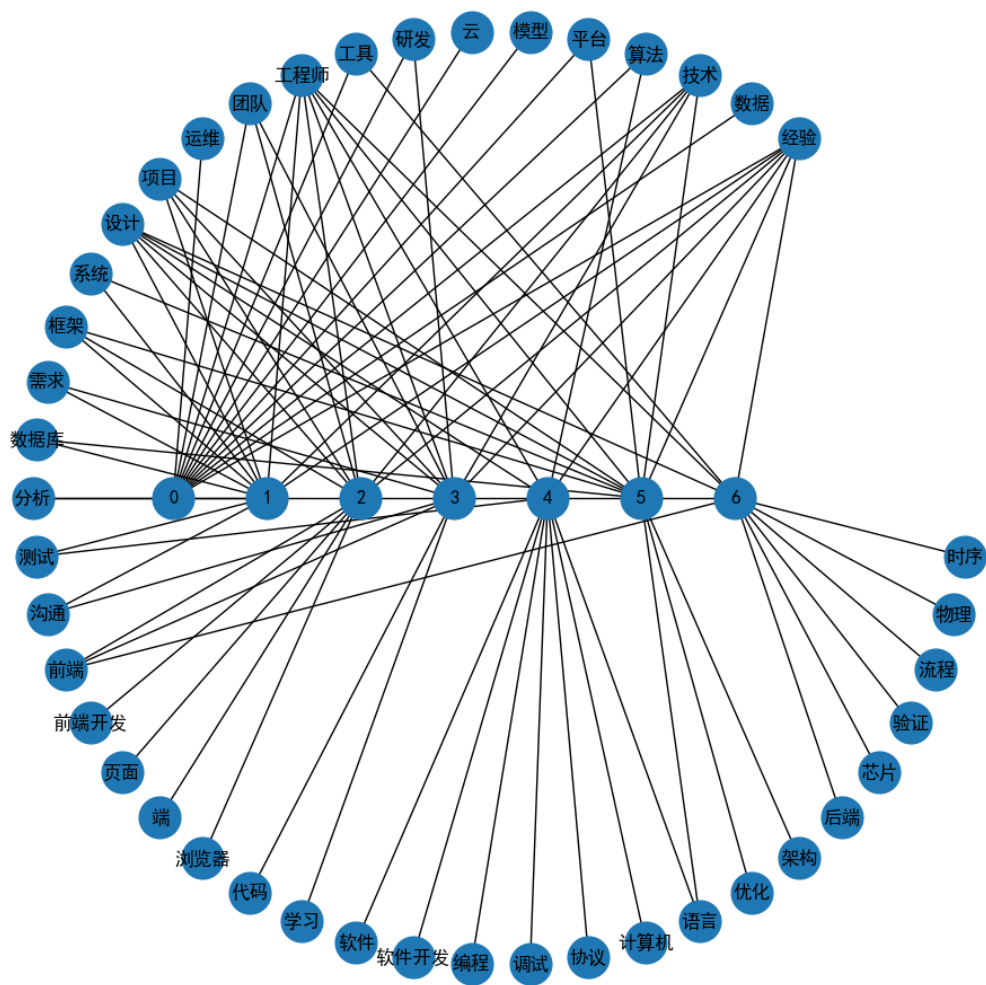


图3-2-1 软件工程的词项分布

其中0-6对应了：python开发、java开发、前端开发、领域应用开发、c/c++开发、golang开发、数字前后端开发

2. 计算机科学：

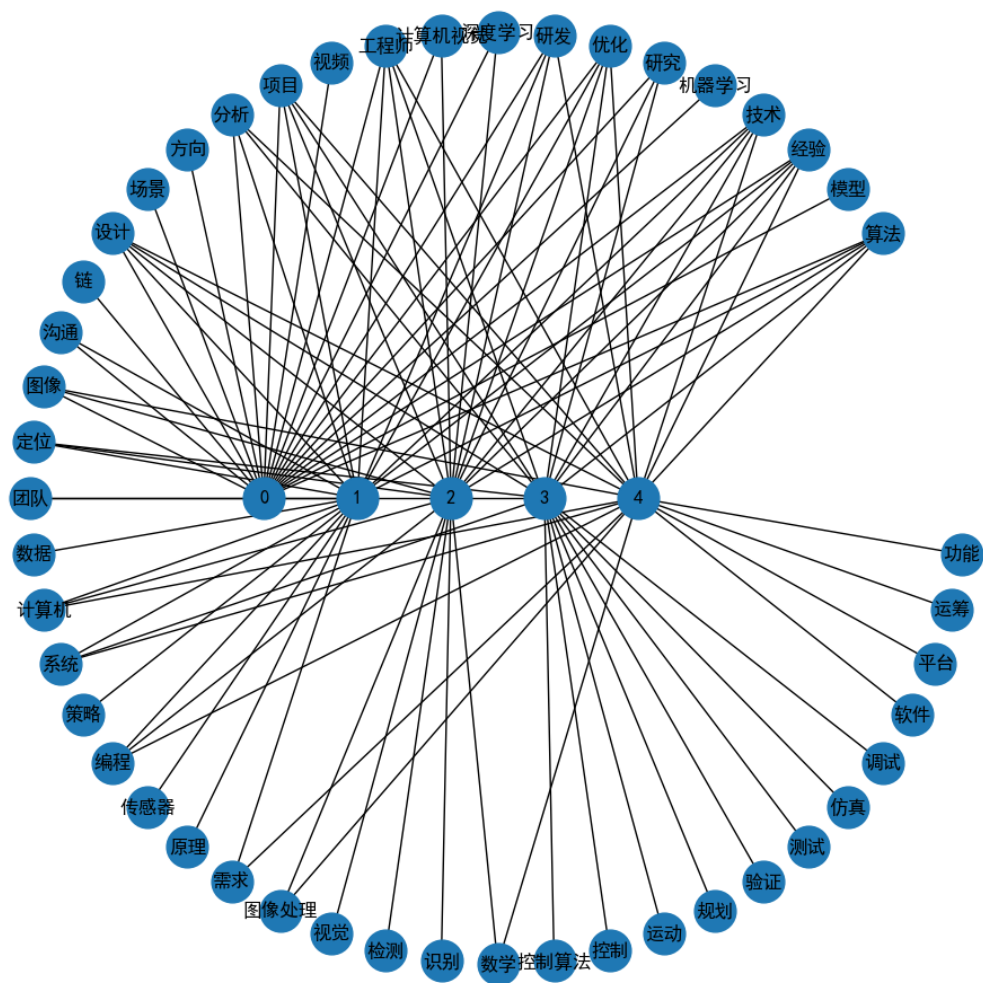


图3-2-2 算法的词项分布

其中0-4对应了：ai算法开发、定位算法开发、cv算法开发、控制算法开发、优化算法开发

3. 电子信息工程：

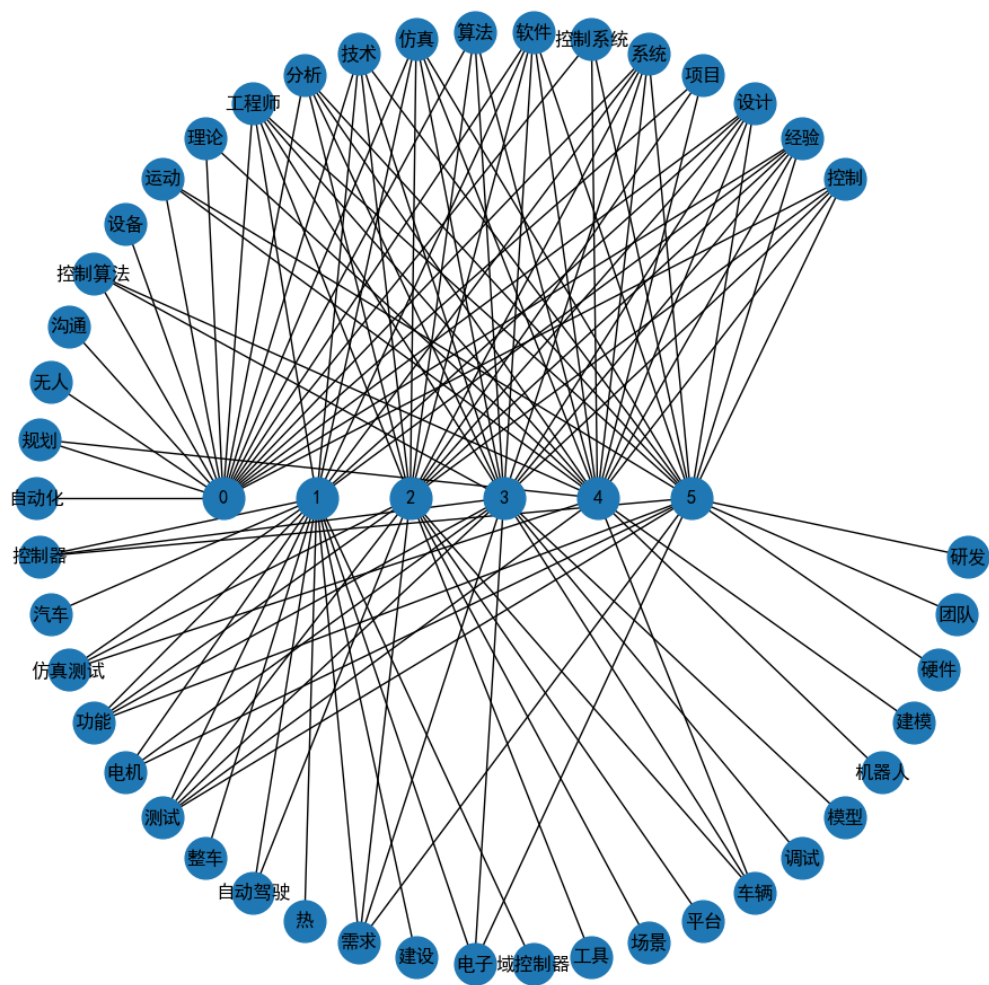
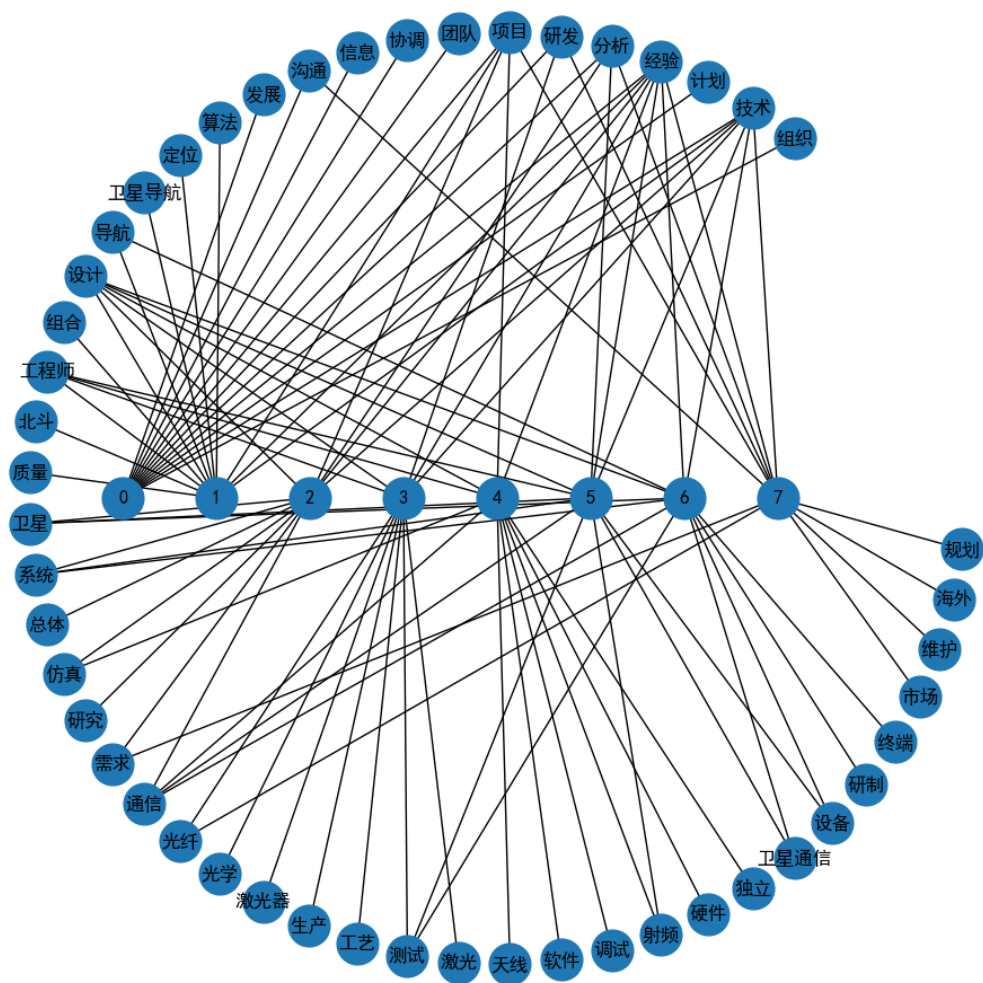


图3-2-3 电机控制的词项分布

其中0-5对应了：仿真控制开发、整车控制开发、智能驾驶开发、电机控制开发、机械控制开发、硬件开发

4. 通信工程：



0-7对应了：采购专员、定位算法工程师、卫星结构设计师、光纤工程师、天线工程师、DSP工程师、卫星通信工程师、销售专员

图3-2-4 通信技术的词项分布

5. 数据管理：

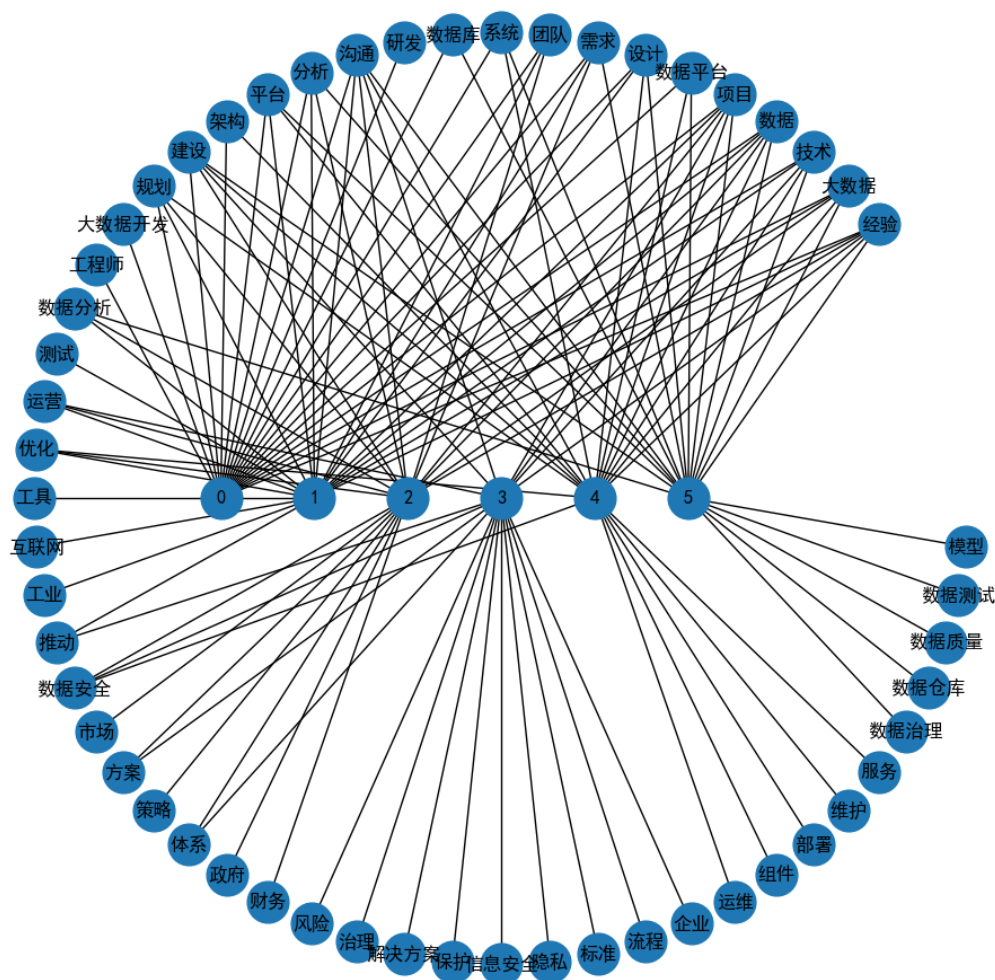


图3-2-5 数据分析的词项分布

其中0-5对应了：大数据开发、商品数据分析、数据产品方案、数据安全、数据系统运维、数据测试工程师

3.2.2 词项分布结果分析

在上面的词项分布图中已经可以很好的看到同一个专业内部的对于专业能力和非专业能力的词汇描述差异了，可以看到大部分的专业都包含了“项目”，“团队”，“经验”等词项，这说明团队协作完成项目在任何一一个相关领域内都有着至关重要的意义，并且对于应聘者就业经验的考究是一个非常重要的指标，经验在招聘中的占比非常大。

除此之外，专业之间也可以通过这个词项分布图找到差异，例如偏向于软件和计算机方面的更多需要在沟通中进行设计和开发，而偏向于硬件的行业则更需要深入研究、钻研。

相似度分析

仅仅从词项分布上并不一定能够准确的获取到相似度，但是可以大致分析出公共点和差异，在此基础上，我们又采用了相似度对词项分布的结果进行解析。我们再次使用了TF-IDF的词袋模型，把已经得到的各个专业下的各个聚类主题的词汇作为原始词汇构建词典，对于每一个专业，把里面的所有聚类主题合并成一个词汇主题，作为这个专业的词项描述。选择使用聚类后的主题词项分布作为原始数据分析相关性而不是用招聘信息的内容作为原始数据分析相关度是因为招聘信息中的噪音太多，很容易会导致错误的结果出现，而主题则是已经浓缩了招聘信息中关键信息的精炼部分，所以使用主题进行相似度分析效果反而会更好。

首先把其中一些偏向于专业词汇描述的部分剔除，保留较多的非专业能力词项，这样相似度的比较可以得到**职业能力**的相似度。然后对所有专业的词汇主题进行TF-IDF词袋模型的构造，这样添加的逆文档频率可以更有效的反映出更明显的特征，基于这个结果，我们使用向量的余弦夹角来计算相似度，对于两个词袋向量，其相似度可以表示为：

$$\cos\theta = \frac{\langle a, b \rangle}{||a|| \cdot ||b||} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

我们使用了python的numpy包完成余弦的计算，结果如表1所示：

表1：不同技术难度要求岗位需要的核心职业能力列表

	人工智能	光电	前后端	导航	嵌入式	数字电子	数据分析	数据开发	电气控制	算法	网络安全	软件测试	通信技术	集成电路
人工智能	100	5.9	3.4	17.3	1.8	1.7	10.1	3.4	2.1	32.2	4.0	3.0	2.2	0.6
光电		100	3.9	4.4	5.7	7.6	3.8	1.6	9.7	9.3	10.7	5.6	14.5	6.1
前后端			100	5.4	8.7	8.5	6.4	6.3	4.9	9.4	6.9	6.0	3.1	7.2
导航				100	7.9	2.3	4.6	0.6	8.4	24.5	1.8	3.8	15.1	1.5
嵌入式					100	20.6	5.2	2.2	20.5	6.6	5.5	1.9	3.5	4.0
数字电子						100	12.8	3.9	8.4	6.4	4.8	7.2	6.0	31.6
数据分析							100	33.4	6.0	8.2	20.0	6.0	3.7	0.5
数据开发								100	2.6	4.3	7.6	1.1	0.2	0.4
电气控制									100	16.2	6.6	9.7	6.7	3.3
算法										100	5.3	8.8	8.4	2.0

	人工智能	光电	前后端	导航	嵌入式	数字电子	数据分析	数据开发	电气控制	算法	网络安全	软件测试	通信技术	集成电路
网络安全											100	2.7	6.3	2.0
软件测试												100	6.4	0.9
通信技术													100	8.9
集成电路														100

得到了专业之间两两的相似度，这里因为 $\cos\theta$ 的值总是在0-1之间，为了方便观察即乘以了100之后保留了1位小数，这样更方便查看，对角线上全是100是因为自己对自己的夹角的 $\cos\theta$ 值永远是1，乘以100之后得到了100，由于表格是对称的，所以只保留了一半的数据。

在这个表格里可以看到一些格子里的数据明显较大，这时就可以认为是两个专业相关，为了反应这个特点我们选择了9.5作为标准，在这个标准之上被称为有强相关性，按照这个规律可以得到：

- 人工智能和导航、数据分析、算法都有着强相关性，其中和算法有着非常强的相关性
- 光电和网络工程、通信技术有着强相关性
- 前后端和算法有强相关性
- 导航和算法、通信技术有着强相关性
- 嵌入式和数字电子、电气控制、软件测试有着强相关性
- 数字电子和数据分析、集成电路有着强相关性
- 数据分析和数据开发、网络安全有着强相关性
- 电气控制和算法、软件测试有着强相关性

我们在前面已经对词项分布进行了剔除工作，剔除了一部分专业特征非常明显的词项，所以剩下的部分是包括了专业词项和非专业词项的，因此这里的原始数据向量是根据了部分非主要专业词项和大部分非专业词项构成的，这样的原始数据更偏向于综合能力的反应，我们称之为**职业能力**，而不仅仅是专业能力。对于两个专业，如果在表中出现了强相关则说明了这两个专业的普适综合能力要求相对接近。

我们可以把专业的招聘需求分成“功底”和“专业技能”两个部分，专业技能就是类似于“C”，“C++”等对于某一技术的详细要求，而功底则是更模糊、更普适的一些专业能力需求和非专业能力需求，例如“计算机”，“团队”，“负责”等。我们剔除了所有的“专业技能”，把有强烈专业相关性的词汇都剔除掉了，例如把算法中剔除掉了“C”，“C++”等词汇，而电气控制里剔除掉了“matlab”等，这样就算保留下来的专业词项也是较为普适的，例如“计算机”，“维护”等在多行业内都需要具备的词项，因此保留下的词项就更接近于“功底”的要求。

例如我们剔除专业词汇后的两个主题如下：

电气控制：['控制', '算法', '自动', '电机', '团队', '项目', '研发', '深入', '仿真', '解决', '负责']
 算法：['算法', '团队', '解决', '负责', '思维', '深入', '优化', '模型', '技术']

我们可以看到二者的相似度主要归咎于"算法", "解决", "团队", "深入"等这几个词项, 其中解决、团队、深入等词汇多是描绘非专业能力的词项, 而算法又是描述了一个比较普适的专业能力的词项, 因此这样比较的结果可以得到一个更普适的相似度。因此, 我们得到的相似度很大程度的反映了两个专业上有哪些普适共有的一些技能, 从而反映出了有哪些专业的基本技能要求更相似。

例如算法和数据分析有着很大的相似程度, 观察二者的词项分布, 大概能得到其相似词汇:

"算法","项目","分析","设计","需求","团队","负责","理解","计算机","沟通","服务","数据"

这就说明了, 应聘者如果具有一定的团队沟通协同开发项目的能力, 并且具备一定的计算机基础、数据处理能力以及算法思维, 能够理解需求并负责任的参与到工作当中, 那么应聘者在这两个专业下的“功底”就是基本相似的, 只需要学习对应专业要求的技术例如C/C++或matlab等就可以在这两个专业之间任意选择就业, 同时应聘者想要从一个专业转到另一个专业方向时, 只需提升技术能力就可以迅速完成职业转变。

接下来我们用相同的方法来计算专业能力的相似度, 这一次剔除了一些非专业能力词项, 例如“团队”, “项目”等, 得到了表2中的结果:

表2：不同技术难度要求岗位需要的核心专业能力列表

	人工智能	光电	前后端	导航	嵌入式	数字电子	数据分析	数据开发	电气控制	算法	网络安全	软件测试	通信技术	集成电路
人工智能	100.0	4.8	3.1	9.2	1.6	2.7	5.5	5.1	2.5	21.7	0.4	1.4	1.7	0.6
光电		100.0	2.9	4.1	2.4	8.5	2.4	2.8	4.7	5.1	1.8	2.4	8.9	5.3
前后端			100.0	3.4	5.3	5.6	5.2	7.4	1.9	9.4	2.3	9.1	3.7	6.9
导航				100.0	2.8	1.9	2.3	1.3	6.8	8.3	1.5	2.9	17.5	1.7
嵌入式					100.0	19.4	6.3	1.1	10.0	4.2	7.8	5.6	5.2	7.1
数字电子						100.0	7.8	1.5	7.2	5.4	6.3	8.2	8.4	14.4
数据分析							100.0	25.3	1.7	6.1	8.0	3.7	3.8	1.2
数据开发								100.0	2.4	3.9	1.6	3.7	1.0	0.1
电气控制									100.0	8.8	1.0	4.5	4.8	4.1
算法										100.0	4.3	8.6	7.1	2.6
网络安全											100.0	9.0	3.3	1.5

	人工智能	光电	前后端	导航	嵌入式	数字电子	数据分析	数据开发	电气控制	算法	网络安全	软件测试	通信技术	集成电路
软件测试												100.0	3.4	2.8
通信技术													100	7.9
集成电路														100

在表格中我们可以看到，以同样的算法计算出来的相似度出现了明显的下降，这正是因为各专业在专业能力上的差异显然更大，因此我们的强相关标准也需要下调，我们选择了8.5这个数字作为标准，得到了如下结果：

- 人工智能和导航、算法有着强相关性
- 光电和通信技术有着强相关性
- 前后端和算法、软件测试有着强相关性
- 导航和通信技术有着强相关性
- 嵌入式和数字电子、电气控制有着强相关性
- 数字电子和集成电路有着强相关性
- 数据分析和数据开发有着强相关性
- 电气控制和算法有着强相关性
- 算法和软件测试有着强相关性
- 网络安全和软件测试有着强相关性

例如，前后端专业经过职业能力关键词的剔除之后保留的专业能力有：

```
前后端: ['sql', 'css', 'springboot', 'ic', 'ds', '架构', 'windows', '操作系统', '服务', '浏览器', 'c', '性能优化', 'ip', 'react', '前端开发', 'java', '代码', '分布式', 'python', 'webpack', '后端', '前端', 'js', 'mvc', 'linux', 'es', 'jquery', 'tcp', 'javascript', '前端框架', 'ajax', 'vue', '多线程', '组件', 'angular', 'spring', 'web', 'redis', 'mybatis', '后端开发', 'mysql', 'app', 'cloud', 'ui', '框架', '架构设计']
```

这个结果和常识上认识到的专业能力相关性较为吻合。

能力值分析

接下来为了更加直观的呈现出各个专业对于职业能力和专业能力的要求，我们根据各个主体的词项分布按照以下的加权词频来计算需求的能力值，这个能力值反映了该专业下对于应聘者的能力的需求等级。

对于专业能力来讲，由于专业能力的针对性和特殊性，每一项专业能力都有着自身无可比拟的作用，不能因为一个专业能力频繁的在多个主题中出现而认为这项专业能力更重要、技术难度含量更高。恰恰相反，普遍出现的专业能力由于要求更广，所以相对而言技术难度反而并没有很高的难度。而那些仅在某些主题下出现的比较有针对性的专业能力反而有着更高的技术难度，因为具有针对性所以需要领域专精。例如：很多专业都要求学会python，也正是因为python的入门难度低并且适用性广，换句话说，适用性广的专业能力往往难度会稍微低一些。而FPGA则是仅在电子信息中出现了要求，因为需要针对性专精人才投入到研究中，很多学生选择了FPGA往往会投入一生的时间进入研究。

因此我们在对每个主题的专业能力的难度进行评分的时候，通过TF-IDF把专业能力词项分布向量转化为TF-IDF词袋向量，并将其中每个维度的评分进行累加获取到这个主题的专业能力技术难度能力值。这是因为TF-IDF模型中除了词频以外还考虑到了逆文档频率，即如果一个词出现的越多的主题下其重要性反而会下降，恰好符合我们的计算思路，因此我们通过累加各个TF-IDF向量中各个维度的值得到这个能力值，它反映了某个主题的专业能力有多“难学”。

$$TFIDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

图3-2-1 TF-IDF的计算公式

对于职业能力来讲，其需求往往更加普适而不需要用技术难度来衡量，而是通过重要性来衡量，显然出现频率越高的词汇越为重要，并且要求的词汇越多越重要。因此我们使用非专业能力词项分布向量计算了每个单词的词频，然后用词频和非专业能力词项分布向量相乘，这样就得到的向量是经过加权之后的非专业能力词项分布向量。我们最终使用加权向量的范数来衡量其综合重要性，这样既包含了频率又包含了数量，囊括了所有的影响因素。最后为了使职业能力值和专业能力值更方便阅读并在数量级上相等，我们给最终结果乘了一个系数。这个最终的数值反映了某个主题对于职业能力的要求有多“多”。

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}$$

图3-2-2 范数的计算公式

最后，表2列出了所有不同技术难度要求的岗位需要的核心职业能力列表及重要程度。

表3：不同技术难度要求岗位需要的核心职业能力列表

岗位类别	专业能力	职业能力	能力总和
通信技术	8.54	10.01	18.55
数据分析	9.19	9.09	18.28
网络安全	8.29	9.86	18.15
电气控制	7.86	9.99	17.85
算法	7.78	7.76	15.54
嵌入式	7.06	8.16	15.22
前后端	7.18	7.93	15.11
软件测试	6.46	8.42	14.88
数字电子	6.43	8.35	14.78
光电	6.64	7.01	13.65

岗位类别	专业能力	职业能力	能力总和
集成电路	6.16	7.28	13.44
导航	6.34	6.93	13.27
人工智能	7.29	5.36	12.65
数据开发	5.1	7.08	12.18

我们可以看到，对于数据分析、通信技术为主题，其专业能力要求也较高，主要是因为这种主题内包含了较多的研究方向，因此如果想要在主体范围内自由选择就业的话，就需要精通多个方面的专业技能。例如对通信技术而言，里面的各个专业有些需要“FPGA”，有些需要“算法”，有些需要“SLAM”，还有些需要“硬件”。数据分析则也是包括了“算法”，“python”，“c”，“市场”，“规划”等多个方面的专业能力需求。因此其专业能力要求较高

我们可以看到，对于通信技术、电气控制等主题，其职业能力要求较高，主要是因为这类专业的工作性质的特殊性导致的，因为这样的工作不仅仅包括了独立研究，同时也包括了团队开发，甚至有些还要去一线进行实践、去参与到采购和销售等工作上，所以其职业能力要求比较广泛，例如这两个主题既包含了“团队”，“沟通”，“项目”，“负责”等职业能力要求，还包含了“深入”，“研发”，“独立”等职业能力要求，因此其职业能力要求较高。

具有强相关性的两个主题的能力值并不一定非常接近，这是因为我们在计算这个能力值的时候并不是简单的累加，对于专业能力我们添加了逆文档频率，放大了尖端技术的影响力。而对于职业能力，我们还考虑了词频和维度带来的影响，因此二者并没有采用同一个标准进行计算，所以结论不能进行比较，但是可以各自反应不同的结论。

4. 研究总结

在现代科技领域中，各种不同的专业领域扮演着重要的角色，推动着科技的不断发展和创新。这些领域之间的相关性体现在它们在解决现实世界问题时常常需要跨界合作和整合各种技术，并且它们之间也存在着一些共同的技术栈。同时它们也各自具有独特的特点和重要性，了解这些专业的差异性有助于学生和从业者更好地选择适合自己兴趣和职业目标的领域。为了研究在软件工程、计算机科学、网络安全、通信工程、电子信息技术、集成电路、数据管理和电子科学与技术等专业上各自在专业能力和职业能力的共同性和差异性，我们着手开始了研究。

首先为了获取到足够多的样本数量，我们对第三方网站进行了筛选，然后通过在秋招旺季的持续长达2个月的时间内使用网络爬虫对其中的数据进行了爬取，得到了近4w条高质量秋招数据，保存在了数据库中。

然后为了分析数据，我们还需要对原始数据进行处理和筛选以及格式化，这里我们用了python的pandas对这4w条数据对原始数据进行清洗，按照每一的字的特性处理原始数据并进行格式化，得到了更加统一、标准的数据，最终保留了近3w条合法且标准化的数据。

接下来，想要获取到具体专业方向的聚类，还需要把所有原始数据转化为向量，我们通过jieba包，并构造了含有10000+个专业词语的词典以及约3000个停用词的停用词典，以此为标准进行了分词，从而得到了每一条原始招聘信息数据被拆解成的词项。

在最关键的一步中，我们通过TF-IDF词袋模型把原始数据词项转化成了一个一个的向量，并采用LDA对主题的困惑度进行了分析，在分析结果中采用了困惑度最低的主题数量作为聚类主题数，然后通过LDA模型确定了聚类主题及其词项分布，这一过程为我们确定了具体的专业方向。接下来我们需要按照已经得到的专业方向，把所有的招聘信息进行聚类分析，由于原始向量的稀疏性，为了更好地获取到聚类结果我们使用了PCA对原始向量进行了降维，获取到了彼此之间差异最大的维度，依此进行了Kmeans聚类，

这样可以让聚类结果更加准确。最终在这一步中我们获取到了都有哪些主题以及每个主题的词项分布，以及每一条招聘数据属于那个主题。

最后，我们对结果数据进行了分析，根据主题的词项分布以及招聘数据的特点，我们为每个主题给出了一个专业的名称，然后根据不同的主题使用了pyplot绘制了其数据分析柱状图。为了研究某个专业下的不同主题的词项差异，我们根据不同主题的词项分布使用networkx绘制了其词项分布共现网络，这样可以更好地对比分析不同的主题之间的词项差异。为了研究专业之间的相似度，我们又对不同专业的主题构造了TF-IDF词袋模型，使用numpy通过余弦夹角来分析两个不同专业下的主题之间的相似度。

我们根据绘制得到的图片以及最后的相似度表格，可以得到不同专业和不同主题的薪资、学历、地区、企业规模、经验要求等信息，并且能够得到主题之间的词项分布差异，这反映了专业内不同主题之间对应聘者能力需求的差异。根据相似度我们还能获得不同专业之间普适能力相似度。

我们发现，算法和人工智能、导航、电气控制等专业有着较高的相似度，因为后者的工作显然脱离不开前者。光电和通信技术有着较高的相似度，数字电子和集成电路有着较高的相似度，他们之间有一些和光学通信、微电子相关的技术栈是共享的，并且工作性质也相近，所以无论是职业能力还是专业能力都存在一定的相似度。电气控制和软件测试存在较高的相似度，是因为二者在电机、机械和整车控制软件上存在着共同部分。

我们还建立了专业技术难度判别标准，并根据标准给出了每个专业的能力值评分，得到了对于能力要求从高到低的专业排序：通信技术、数据分析、网络安全、电气控制、算法、嵌入式、前后端、软件测试、数字电子、光电、集成电路、导航、人工智能、数据开发。

最后，我们希望这些数据可以被客观看待，并希望我们的研究结果可以帮助学生和从业者在选择和专业知识培养上做出更加客观的选择，能够根据自身和要求有机结合，选择适合自己的就业方向。