# 《智能简历解析系统需求分析与设计》

#### 1. 项目概述

·项目背景：简历分析是招聘流程中的耗时和耗力环节，利用人工智能技术可以实现自动化简历分析和人才匹配。

·项目目标：开发一套智能简历解析系统，实现简历分析、人才画像、人岗匹配等功能。

·用户和用例：主要面向企业HR和招聘管理人员，用于提高招聘效率和质量。用例主要包括简历上传、简历解析、报告生成、人才画像、人岗匹配等。

#### 2. 功能需求

·支持常见简历格式上传：文本、PDF、Word、图片等

·实现简历解析和关键信息抽取：姓名、年龄、学历、工作经历等

·生成简历分析报告：数据统计图表、富文本解析结果展示

·构建人才画像：自定义人才画像维度和标签

·实现人岗智能匹配：根据简历和岗位要求实现匹配和排序

#### 3. 非功能需求

·系统性能：上传、解析和报告生成的响应时间要求

·系统安全：保护用户数据和隐私的安全需求

·系统可靠性：软件和服务的高可用性需求

#### 4. 数据模型设计

·简历表、岗位表、解析报告表、人才画像表等数据库表结构设计



#### 5. 接口设计

·前后端接口：各模块和功能对应的接口地址、参数、返回结果定义

·后端服务接口：简历解析服务、报告生成服务、人岗匹配服务的接口设计

1. 公司API

创建公司: POST /api/companies

获取公司信息: GET /api/companies/{companyId}

更新公司信息: PUT /api/companies/{companyId}

获取公司的所有职位: GET /api/companies/{companyId}/jobs

2. 职位API

创建职位: POST /api/jobs

获取职位信息: GET /api/jobs/{jobId}

更新职位信息: PUT /api/jobs/{jobId}

删除职位: DELETE /api/jobs/{jobId}

3. 简历解析API

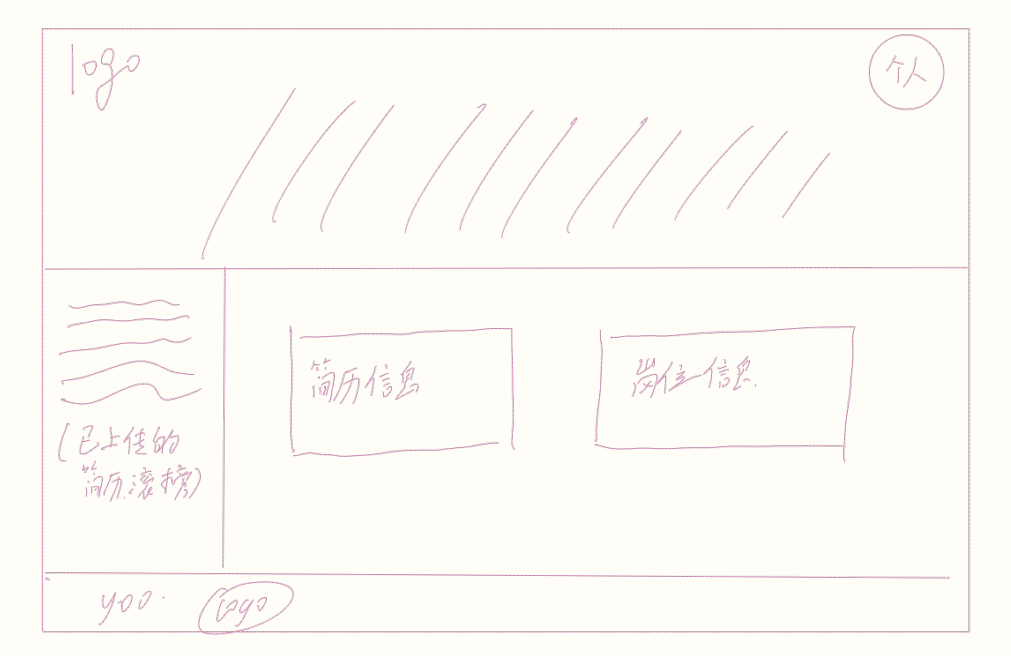
提交简历进行解析: POST /api/resume-parser

获取解析后的简历信息: GET /api/resume-parser/{resumeId}

（暂时主要的接口是这么多）

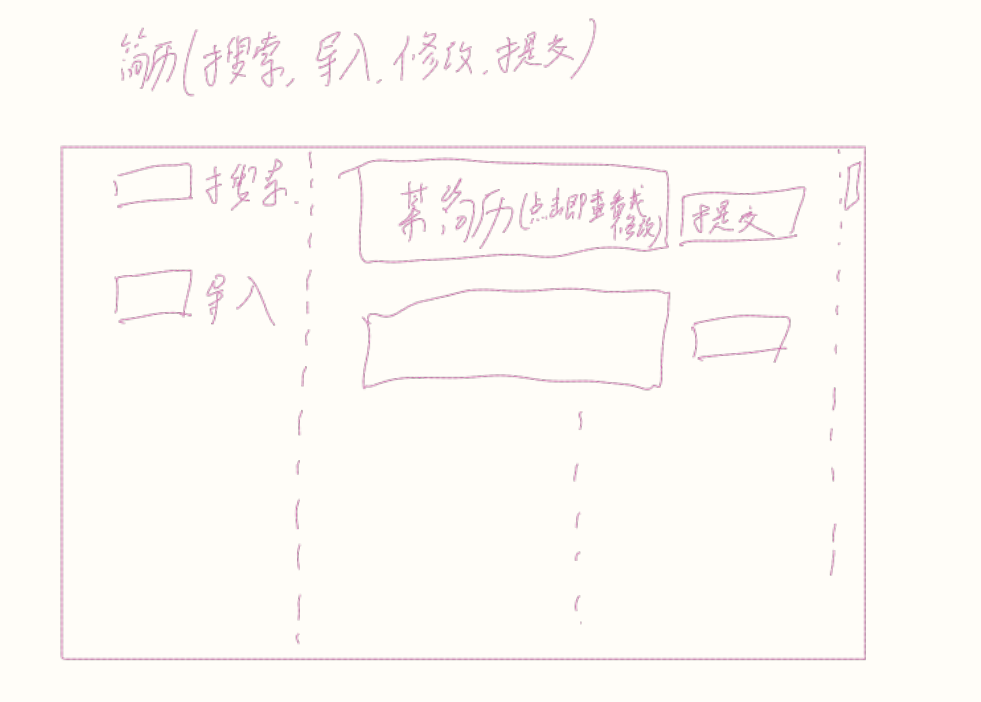
#### 6. 界面原型设计

·简历上传页面、解析结果页面、人才画像页面、人岗匹配页面等界面原型



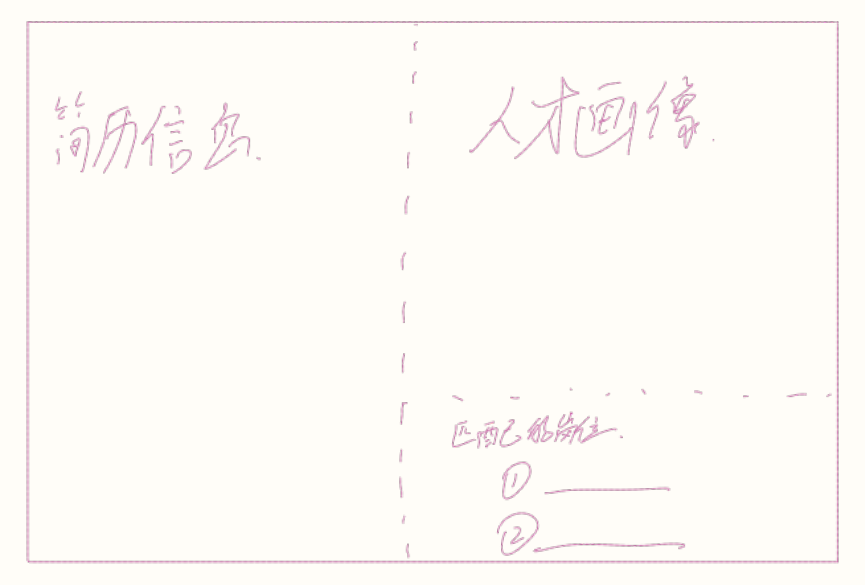
如上，是我们设计的前端主界面，我们大概分为了三段式，比例约为442，页面顶部为滑动式交互界面，可以左右滑动，展示多个内容块。中间部分分为左右两个部分，左边为滚动条，滚动的是我们已经上传的简历名称，动态轮播，右边有两个按钮，分别是“简历”和“岗位”。底部为底栏，包括页面链接和版权信息，仓库地址等。

点击简历：

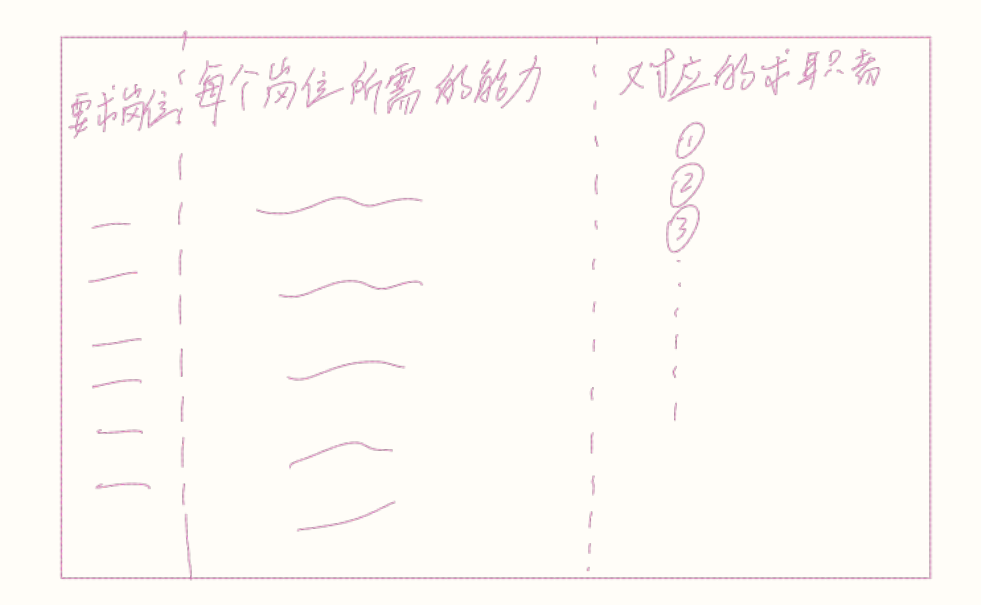


当我们点击简历按钮的时候，我们就需要跳转到简历的相关信息界面，首先我们要先能够对简历自身完成修改和确认提交的功能，在这里，我们分为左右两部分，比例为28或37，左侧完成对简历的搜索查找功能，以及导入简历的功能，右侧就是对简历的大致内容进行展示（比如姓名、年龄、学历、工作经历、应聘岗位等），然后在此处点击简历就可以进入“修改简历及展示人才画像”的界面，以及后边的提交按钮，表示确认此简历信息无误，可以提交到数据库当中。

点击某个简历：

 此处，我们可以左右五五分，左侧显示简历信息，以及如何修改及确认，右侧展示左侧简历所对应的人才画像，以及右侧下边可以有一栏，显示这个人自身对于岗位匹配程度的前两名岗位。

点击“岗位”按钮：



此处，我们的界面大概分为了三部分，比例大概为244，第一部分是企业所需要的岗位，第二部分是每个岗位所需要的能力的要求（可以用文字形式描述，也可以用雷达图之类的画出来要求），右侧就是对应的求职者里，有哪些是符合要求的。

#### 7. 算法设计

·简历解析算法：简历结构识别和关键信息抽取规则

·人才画像算法：根据简历属性生成人才标签的规则

·人岗匹配算法：人才画像和岗位要求匹配模型和匹配规则

简历解析算法：

首先我们要实现简历结构识别，可以用opencv和ocr文字提取相关功能，此处赛题已经要求使用移动云的文字识别，所以我们就要从精确提取每一部分的文字出发，将pdf或者图片的每一部分信息提取出来，使用opencv相关功能，框出每一部分的区域，然后提取这一部分的文字，最后将整张简历，不同部分的信息，组合成一个txt文本，实现提取简历信息的功能（docx的简历可以直接使用docx2txt库函数提取出来相关内容）。

然后就是如果我们想要实现简历解析，需要使用NLP技术来解决，需要使用到的技术有命名实体识别（NER），关键词抽取等。此处，我们可以使用功能较为简单的jieba分词，或按照权重提取关键词，或者是使用BERT、GPT或其他的transformer模型来识别简历中的关键信息，比如说bert的ner（Named entity recognition），首先我么你先提取出相关关键信息即可。

**简历解析（简历可以考虑把文本分块，然后再用BERT+CRF提取里面的实体词）**

1.Detect简历图片中的各个区块,如个人信息区块、教育经历区块、工作经历区块等。对每个区块内的文本进行OCR识别,得到各区块的文本信息。

2.对每个区块的文本进行分词,这里可以使用jieba分词。然后构造输入数据,输入到BERT+CRF模型中。输入数据格式为:[CLS]区块1文本[SEP]区块2文本[SEP]...[SEP]区块n文本[SEP]

3.BERT+CRF模型会识别每个词属于哪个类型的实体,如人名、地名、机构名等。那么对每个区块,我们可以统计实体的种类和数量,得到该区块的主要信息。

4.根据各区块的主要信息,组合生成一个结构化的简历信息。如:

{

"个人信息": {

"姓名": "张三",

"学校": "xx大学"

}, },

"教育经历": {

"学校1": "xx大学",

"专业1": "计算机科学与技术",

...

},

"工作经历": {

"公司1": "xx公司",

"职位1": "软件工程师",

...

}

}

5.完成简历信息的解析和提取。通过统计各区块的实体信息,重新组合成结构化的json格式简历信息

这样通过OCR识别简历图片文本,BERT+CRF模型提取实体信息,最后组合生成结构化简历信息,完成了简历的解析和信息提取。

人才画像算法：

人才画像的生成通常涉及特征工程和标签生成的步骤。特征工程主要是对抽取出来的简历信息进行编码和标准化。标签生成则可能需要使用到机器学习的分类算法，例如决策树、随机森林、支持向量机 (SVM)等。也可以使用深度学习的方法，例如神经网络等。

人才画像的生成主要分为两个步骤：

1. 特征工程：这一步主要对简历信息进行预处理，如：

·简历信息编码：将字符串信息编码为数字，如将学历编码为1本科，2硕士，3博士等。

·信息标准化：对量纲不同的信息进行标准化，如工作年限的范围是1-30年，学历年限是1-8年，需要进行标准化处理。

·特征提取：提取简历中与人才标签相关的关键特征，如技能关键词、行业关键词等。

1. 标签生成：这一步需要利用机器学习算法根据简历特征对人才进行分类和打标，常用的算法有：

·决策树：根据简历特征选择最优的标签分类规则。

·随机森林：集成了多颗决策树，对人才标签进行投票分类。

·SVM：在特征空间中找到最佳的分隔超平面，实现人才标签分类。

·神经网络：使用深度神经网络对简历特征进行自动学习和人才分类。

所以，人才画像算法主要分为简历特征工程和人才分类标签生成两步。特征工程侧重于简历信息的编码、标准化和关键特征提取。分类标签生成则利用机器学习算法根据简历特征对人才进行分类打标。

但事实上述的功能过于复杂且难以实现（对目前的我们来说），所以我们想到了一种更为简便有效的，如下：

此处人才画像的构建，可以依据提取出来的关键信息显示出来的每个人的本领，对于各个能力进行打分，然后依据各方面不同的分数，对每个人才进行画像的构建。

我认为基于关键信息抽取和能力打分的人才画像构建方法是可行的，具体流程可以如下：

1. 关键信息抽取：从简历中抽取关键信息，如技能关键词、工作经历、项目经验、语言水平等。这些信息可以判断一个人的能力和特长。

2. 能力评分：根据关键信息为不同的能力打分。（能力方面有些不太容易打分，可以基于Bert做一下分类。）

如：

·技能：判断关键词中出现的技能词汇，给相应技能打分。如Python、机器学习打分高。

·工作经历：根据工作年限和工作性质给“工作稳定性”、“实际工作能力”等打分。

·项目经验：根据项目关键词，给“项目管理能力”、“团队合作能力”等打分。

·语言能力：根据外语关键词，给“英语水平”、“日语水平”等打分。

3. 人才画像生成：根据不同能力的评分结果，生成人才画像。如：

·技能画像：以评分最高的2-3项技能作为人才标签。如“Python工程师”、“数据分析师”。

·综合画像：考虑多能力的评分，生成例如“技术型人才”、“项目管理型人才”等综合画像。

·细分画像：可以同时生成技能画像与综合画像，进行人才的细分定位。

以上方法根据关键信息评分和打分的思路进行人才画像的构建，可以有效评估一个人的多方面能力，并据此给出较为准确的人才画像。所以，这种方法是可行的，甚至可与机器学习算法相结合，提高人才画像的准确性。

最后，我们可以考虑是否可以通过这些信息，进行语句整合，提交给midjourney，生成一个图片之类的，更为美观的显示出人才画像。

人岗匹配算法：

1. 特征提取

使用BERT等预训练语言模型对简历和岗位描述进行语义编码，得到其语义向量。BERT模型可以捕捉到文本的深层次语义信息，这对于人岗匹配任务是非常有用的。

具体来说，可以使用“transformers”库来加载预训练的BERT模型，然后将简历和岗位描述的文本输入到BERT模型中，得到每个文本的语义向量。这个向量就是该文本的特征向量，可以用来计算两个文本的相似度。

3. 匹配模型训练

在这一步，需要使用大量的标注数据来训练你的匹配模型。你可以选择一些常见的监督学习模型，如逻辑回归、支持向量机、神经网络等。这个模型的任务是，给定一份简历和一份岗位描述的语义向量，预测这份简历和这个岗位是否匹配。

这个模型可以使用各种深度学习框架（如TensorFlow或PyTorch）来实现。训练过程中，可以使用交叉熵损失函数作为优化目标，使用Adam等优化算法进行参数更新。

4. 初筛匹配

在这一步，需要设定一个语义相似度的阈值，然后计算每一份简历和每一个岗位的语义相似度。只有当相似度高于阈值的简历-岗位对才会进入下一步。

计算语义相似度的方法有很多，常见的有余弦相似度、欧氏距离等。可以使用numpy或者scikit-learn等库来进行计算。这里的阈值可能需要通过实验来调整，以使得初筛的结果既不会过于宽松也不会过于严格。

5. 深度匹配与推荐

对初筛通过的简历-岗位对，再次计算其语义相似度，并根据相似度进行排序。最后，选取相似度最高的前N个简历-岗位对作为推荐结果。

这一步可以使用深度学习模型来实现更精细的匹配。例如，可以使用Siamese网络或者Transformers网络来计算简历和岗位描述的深度语义相似度。这种模型可以捕捉到更细粒度的语义信息，从而提高匹配的准确性。

6. 结果解释

为了让用户理解匹配的结果，可以提供一些结果解释的功能。例如，可以显示匹配的关键词，或者显示简历和岗位描述的语义相似度。

对于显示匹配的关键词，可以使用TF-IDF（此处，jieba库里边提取关键信息就包含TF-IDF）或者其他关键词提取方法来找出简历和岗位描述中的关键词，然后找出这些关键词中的共享词。这些共享词就是匹配的关键词。

对于显示语义相似度，你可以直接显示上述计算的语义相似度。为了让用户更好地理解这个数值，你可以将其映射到一个易于理解的范围，如0到100分。

综上，这是一个比较完善的需求分析与设计文档，从需求概述到界面设计，层层递进地描述和定义了整个智能简历解析系统的需求与解决方案。