简历处理目标：

1. 将不同格式、不同风格的简历转换为一个标准格式，储存在数据库里，以供之后的分析使用。
2. 此标准格式应该初步将简历内容进行分类（姓名，联系方式，工作经验，教育水平，兴趣爱好，等等），以方便比对和查询；分类不可太多详细，因为太详细的转换器难以 generalize，也会影响之后分析的准确度。
3. 将处理完的简历进行 topic extraction，提取出简历各类别内的细分 topic。（比如工作经验内可依据职位、行业、公司规模、项目类型进行细分）
4. 简单的基于特征的分类：教育水平，成绩，学校，工作经验（几年），掌握的技能，等等
5. 加入新的特征：用一个 score 代表求职者各方面的能力，在推荐求职者时作为一个 ranking 的参数
6. 推荐系统：通过求职者的信息与他们工作过的公司的信息或职位要求做一个 hybrid recommender，为每个公司的职位找出最适合的求职者。

简历处理流程：

1. 根据不同格式读取文件：
   - txt: 直接读取
   - html：beautiful soup 抓取
   - pdf, word: third party libraries like “pypdf” and “pywin32”
2. 关键字眼抓取，分割文本：因为我们无法通过字体大小、下划线、粗体等等其它办法来分辨标题，只能通过寻找关键字眼来分辨标题。
   - 可以使用 wordnet 里的 synsets 来找到类似标题的近义词。比如找到类似“工作经验”的词
   - 使用 regular expression 的 chunk/tokenizer method 来找出两边都有空格的词语，此类词语有大概率是标题
   - 找出标题后，将其它内容依次按照标题分割，各自形成一个分类

- 人工比对 output 和原简历，修正错误，形成一个可适用于大多数文件的方法，写成一个 pipeline
- 处理完的简历写入数据库或存在本地

3. Text processing
   - 分句(sentence segmentation)
   - 分词（tokenization，中文分词方法）
   - 加上 tag
   - name entity recognition: 标注名词
4. Topic extraction
   - 根据第三步的结果找出每个句子/类别的关键字眼（比如工作经验：工作岗位、公司名称、工作时间、项目内容，等等）
   - 这一步过后应该可以完成目标(4)：基于简单特征的分类
5. 加入新的特征
   - 通过对简历更详细的分析，建立更能体现求职者能力的特征，比如在大公司工作的经验、工作总年数、编程比赛获奖名次，形成一个新的数值表（可以按照职位要求在 ranking 的时候给这些数值加上偏重）
6. 推荐系统
   - 使用已有信息：使用简历里求职者工作过的公司、职位、求职者信息（包括基本信息和以上步骤形成的附加信息），以及网上能找到的公司信息（类型、产业、规模），作为 recommender system 的 input，输出结果为适合每个公司每个职位的最适合应聘者。