Project Proposal: from Web Scraping to Job Advice using the Job Platform "De.Indeed.com"

Hertie School

Prof. Dr. Hannah Béchara

Spring Semester 2019

By Aline Bocamino, Andrea Giuliani,

Mariana Saldarriaga, Toma Pavlov

1 Abstract

"Jobs!" This is the key word on the mind of every student soon to graduate from the Hertie School. Does developing data-analysis skills by taking statistics and programming courses help students landing well-paying jobs in Berlin? What is the data job scene like in Berlin and where can Hertie students position themselves? These are intriguing questions for which none of us has a good answer. Therefore, to try to make some clarity, we are embarking on a Python journey to explore and assess how many data jobs are available in Berlin considering the time-frame of the past 6 to 12 months. We analyze their key requirements, and how they match with the skills of Hertie students. Our final project aims at collecting and analyzing data-related job ads in Berlin, as well as creating a search system tailored to the individual user. The project idea consists of three key steps: (1) scraping the data; (2) analyzing possible trends; (3) creating a service which allows every student to find jobs related to his/her skills. The source of data is arguably one of the biggest job websites in Germany – de.indeed.com – and it is particularly popular among job-seekers in Berlin.

2 Scraping the Data

Python is well-known for its ability to scrape data from the web. There are numerous resources available, detailing ways to do this. Our preliminary research led us to useful introductory guides on web scraping in Python, as well as a blog post that talks about specifically scraping job postings from de.indeed.com.[1]. The essential steps we plan to undertake as part of the first stage of the project are: (1) defining the URL, (2) inspecting the page, (3) defining the parameters (search terms, city); (3) writing the code; (4) running the code and scraping the data; (5) storing the data in a data frame and csv format.

In order to scrape the job postings data, we are going to rely on BeautifulSoup – a Python library for pulling data out of HTML and XML files. It works with a parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. Based on our preliminary research and on the experience of other users, we think that BeautifulSoup is the most suitable library for the purpose of our project. Furthermore, the extensive documentation available on the BeautifulSoup library is also something we considered when choosing it.

While we found Python codes for scraping job ads from indeed.com, the script is for the U.S. version of the job platform and it is also rather outdated since it was created two years ago. Running the codes for the U.S. version yielded mixed results with most fields in the final csv output being empty. Therefore, we plan to utilize this existing source along with other available codes on web scraping with BeautifulSoup. This step, however, is not going to merely be a "copy & paste" exercise, given the specific characteristics of our project. In fact, one of the main challenges that we foresee during this first stage of the project is to identify the exact tags we need within the long HTML page of de.indeed.com.

1. The guide that we use as our main source is available here, while the blog entry can be accessed here

3 Analyzing Trends

Once we 'scrape' the data, we will have it in a textual format, which will be highly unstructured. Therefore, we will need to follow a method called "Text Analytics" or "Text Mining". The purpose is to derive all meaningful information from the natural language text. In order to do this, we are going to use the NLTK library that is the natural language toolkit for building Python programs to work with text data. First, we will need to pre-process the basic "text of text" data. There are a number of steps that we are going to take in order to clean the text and obtain a better and more effective analysis:

1. Lower case: we will transform every description of job posting into lower case to avoid multiple copies of the same word

2. Removal of punctuation: this will help us to reduce the size of the text data

3. Removal of stop words: there are some words that should be removed from the text. For this purpose, we will try to use a predefined library. If this doesn't work properly, we will create a list of stop words to remove

4. Removal of common words: besides a general cleaning, we need to remove the most recurrent words that are not relevant for our classification of our text data

5. Removal of rare words: in order to have true associations between some words, we need to remove the 'noise' created by rare words

6. Tokenization: we have to break a complex sentence into words. With this step, the text will be split into tokens (words, comma, punctuations). We are going to use the textblob library to first transform our job postings into a blob and then convert them into a series of words

7. Stemming and/or lemmatization: we also need to relate the words that have the same base (root) form. This means that we have to normalize the words into their base

form. For this step, we are using the method Lancaster Stemming, that will remove suffices like “ing”, “ly”, “s”, etc. However, we are also going to apply lemmatization, which converts the words into their root word (rather than just eliminate the suffices), and see which option is more effective

After completing the steps listed above, our data will be clean and ready for the analysis. To move on the to the data examination phase, we are going to realize text classification, and we will use packages such as pandas. Our goal is to have a text database similar to the one in the example below:

Figure 1: Example of Text Database

	city	job_qry	job_title	company_name	location	link	date	Ratng	full_text
pj_fb35711a	Berlin	data+analyst	Data Analyst - Convenien	Zalando SE	Berlin	/pagead/clk?	Vor mehr als 30	1	As a Data Analyst in our Con
pj_d4c4ff57e	Berlin	data+analyst	Data Analyst Customer E	idealo internet GmbH	Berlin	/pagead/clk?	vor 22 Tagen	1	Als Teil unseres Teams kanns
pj_ca661c44	Berlin	data+analyst	Process Analyst	Zalando SE	Berlin	/pagead/clk?	vor 7 Tagen	2	As a Process Analyst will be f
pj_06ac2afd	Berlin	data+analyst	Senior Data Analyst - Inv	idealo internet GmbH	Berlin	/pagead/clk?	vor 22 Tagen	4	idealo has direct business rel

We will also use the methodology of “part of speech tagging (POS)”, which is employed to assign parts of speech to each word of a text based on its definition and its context. The previous tokenization step is essential to get here. Also, we will realize named entity recognition and chunking (grouping of words into categories). One challenge we foresee at the second stage of the project is the coexistence of German and English job ads. The best way to extract features from a text is to use the Bag-of-words model (BoW). We will create a Document-Term Matrix (DTM) - a matrix of document and words by counting the occurrence of words. Besides, we will also apply term frequency that is simply the ratio of the times a word is present in a sentence and the length of the sentence.

Finally, for this second part of our project, we will do a visualization to represent the content of the job postings we ‘scraped’ and we will also use matplotlib, Plotly’s Python

graphing library and Bokeh visualization library. We will plot the distribution of our variables (see previous table) and create a comprehensive description of the job market in Berlin.

4 Job Recommendation Tool

The third part of our project will be a job recommendation tool. The goal of this tool is to collect information from the candidates and, based on this information, recommend a list of jobs which better matches their profile and preferences. This recommendation tool will use a reduced database which contains only job ads posted in the last month. In terms of programming, the tool will be done in two main steps:

4.1 User interface: collecting information from the job seeker

At this step, a user interface will be created to collect information from the job seeker, who will select his skills and preferences in a dropdown menu. All the following information will be voluntary, while the "skills" will be mandatory:

1. Skills (Python, Java, R Studio, Statistics, Project Management, etc.)
2. Job dedication (part-time, full-time or both)
3. Type of job (Internship, working student or both)
4. Sector (consulting, government, third sector, all of them) – Feasibility to be confirmed during Step 1 of the project (Data scraping)
5. Language of the posting (German, English or both) – Feasibility to be discussed and confirmed during Step 2 of the project (Data analysis)

For this first step, we plan on using the app named Streamlit, which is an open-source app framework that uses Python to create user interfaces and dashboards. The developers recommend using the Streamlit app for the tool, because, compared to a traditional GUI,

the interface is more appealing for the user as well as easier to build and deploy. There is plenty of codes and tutorials on the internet about how to use Streamlit, which will be employed as a reference for the development of our project.[2]

4.2 Recommendation engine: sort the job ads according to the inputs and recommend a list

Based on the parameters selected by the candidates in the previous step, an algorithm will filter the job ads database. The main criteria to filter the database are the skills that will be used to construct the three categories visualized in Figure 2.

1. Category 1: job ads with a perfect match: all the skills listed in the job ads are contained in the information entered by the job seeker

2. Category 2: job ads with a partial match: Jobs ads which require at least some of the skills listed by the job seeker. Within this list, we will sort information according to a skill fulfillment index (a percentage index which denotes how many skills of the job ads are mastered by the candidate)

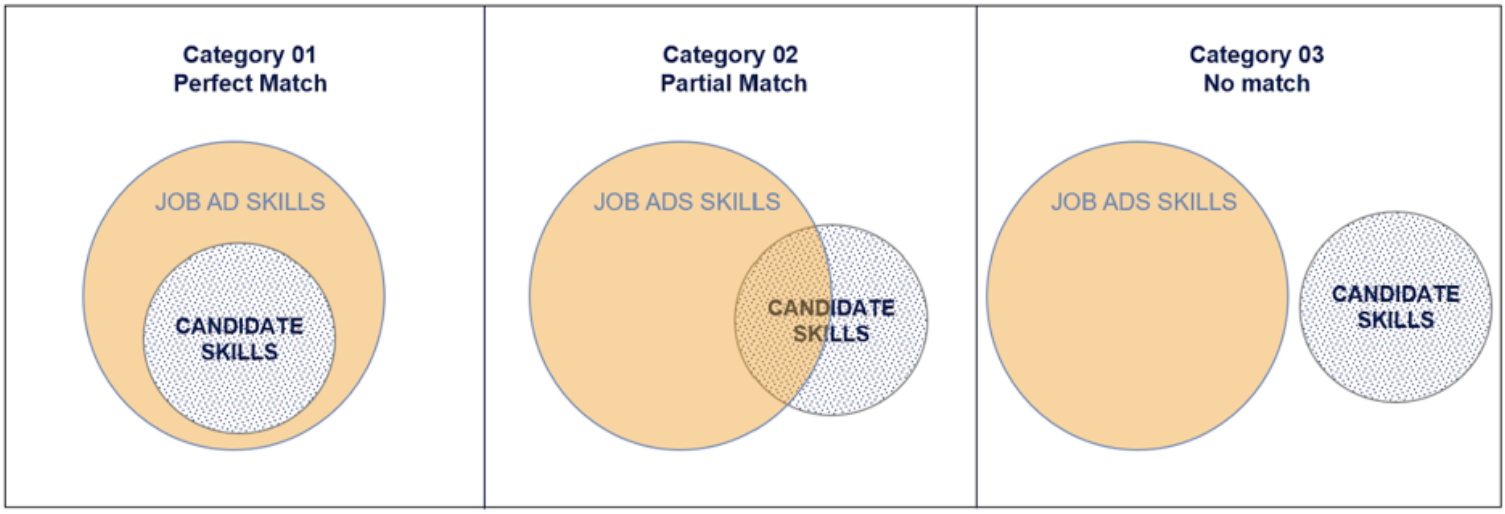
3. Category 3: No match: job ads which require skills that are different from the ones entered by the job seeker

Besides skills, the database will also be filtered in terms of job dedication, type of job, and language. Within these categories, we will also sort the results by the date of the job posting, employer review, and salary. (These two last categories are still to be confirmed).

Finally, after filtering and sorting the database, a list of 20 job ads will be displayed in Streamlit. First, we will display the "Category 1", followed by "Category 2". In case there are no job ads in the first and second category, it will be displayed an informative message explaining that no job ads available matched the profile of the candidate. The interface, as

2. An example of the resources available online can be accessed here

Figure 2: Visualization of the Three Categories for the Job Ads Filter



well as the dropdown menu options, will be displayed in English, however, the algorithm will access a dictionary, with which will be possible to work also with job ads in German. In other words, if a candidate selects the skill "Leadership", the engine will use the dictionary to filter the jobs ads written in German and that contain the skill "Leitung".

5 Timeline

In order to complete the project within the time-frame we were given, we plan to follow the timeline below:

1. **March 31st**: step 1 "scraping the data" and demo of the service completed
2. **April 14th**: step 2 "analyzing trends" completed
3. **April 28th**: step 3 "job recommendation tool" completed

6 Questions

1. **Question 1**: how do you recommend to create categorical variables from the original dataset?

2. **Question 2**: besides the job categories listed in this proposal, is there any other specific category that you would like to see?