STUDENT SESSION

UNDERSTANDING JOB REQUIREMENTS USING NATURAL LANGUAGE PROCESSING

Luka Aničin, (Student)*,
Miloš Stojmenović, (Mentor)

University Singidunum,
Belgrade Serbia

Abstract:

With the rising number of remote jobs and mediums on which companies rely as funnels to attract job candidates, the number of job applications receives increases exponentially over time. Human Resource (HR) departments are becoming bottlenecks for pleasant applicant experiences because of the laboriousness of the application processing task. Increasing the size of HR departments works to a certain point but hiring more HR specialists becomes impossible from a financial and managerial standpoint. We propose a novel approach for candidate filtering based on competence matching between job ads created by companies and submitted resumes. The proposed system achieves 99.5% accuracy and relies on natural language processing techniques to extract information from both candidates' resumes and job ads. It allows companies to create their personalized automated filters using the extracted information.

Keywords:

e-administration, public administration, administrative procedures.

INTRODUCTION

As a result of Covid 19, more and more companies rely on internet-based job boards to find qualified candidates. Each day, we can see more than ten million new job openings posted across the web, which provides fantastic opportunity and great stress for job seekers. LinkedIn, which now has more than 750 million active members worldwide, saw a sudden increase in posted job demand, yet had difficulty adjusting their job search engine with this new traffic. Currently, LinkedIn jobs show a maximum of 10.000 jobs for any given search pattern, whereas they have over 14 million active jobs. This search algorithm limitation is not an isolated problem to LinkedIn, and it results in millions of unreachable jobs for job searchers.

Correspondence:
Luka Aničin

e-mail:
luka.anicin.21@singimail.rs

DOI: 10.15308/Sinteza-2022-457-463

The lack of a personalized search is the direct consequence of outdated technology, which produces a skewed distribution of applicants to the top searched/reached companies and their jobs. On internet job boards, human Resource departments at the most visible companies are buried in hundreds, if not thousands of applicants. Having many applicants for a company's job might sound fantastic for the company, but it is an unnecessary burden for the HR department. Most companies don't have the resources to process all submitted applications. Even if we assume they do, they are unsure how to rank applications to allocate more time to the most suitable candidates.

Artificial Intelligence has changed whole industries in the last couple of years, from creating self-driving cars to supermarkets without human workers. This shift is just beginning for the Human Resource departments and their day-to-day workflow.

Our main contributions are summarised below:

1. We demonstrate an approach to applying state-of-the-art Natural Language Models to job descriptions and create a much faster and more personalised job search and candidate filtering system.
2. We provide an open-source dataset with 100,000 carefully curated job descriptions for technical positions (each job has features extracted, such as - years of experience, degree level, degree area, and skills) and a pre-trained model for extracting relevant information from job descriptions, which will provide a starting point for new researchers in this domain and potentially grounds for downstream commercial applications.

2. OVERVIEW OF NATURAL LANGUAGE PROCESSING EFFORTS IN THE HIRING PROCESS

From the early 2000s, the recruiting procedure became a large playground for companies to explore their AI approaches to improve/automatise certain aspects of this process. The most significant problem in recruitment to this day is bias in decision-making. When a process has humans on both ends, bias is inevitable. This area was the first where companies started investing time and resources to improve this problem and make the hiring process much less stressful. Startups created automated bots for initial screening interviews, algorithms for analysis of telephone screenings, and finally, Resume/CV filtering to remove the urge of deciding with a bias towards certain candidate groups.

The HR community recognized that bias in resume filtering is a long-standing issue and initial attempts at alleviating it was based on simple key-pair matching heuristics [1]. In 2016, SAP researchers tackled this problem using machine learning [2] and showed that they could scale key-based systems by purely relying on data distributions [3].

Building on the first efforts for candidate filtering and matching, a couple of promising research directions relying on machine learning approaches to solve the constraints of key-based systems emerged. Lin et al. [4] created an ensemble model by combining XGBoost [5] with a Convolutional Neural Network (CNN) [6] to create an ensemble model and extract relevant information from job descriptions and match them with job seekers.

The ensemble approach showed promising results but did not understand the context of the given text. We recognized this problem along with an earlier French research team. Their paper [7] demonstrated a candidate matching system with open job positions by utilising the BERT model. They trained their model on a privately held dataset of 106 resumes without tackling the job description extraction part. This paper will go a step further and create our dataset, which allows us to create and train a BERT model with higher retrieval accuracy than the mentioned papers.

According to Wright [8], more than 70% of HR teams showed a positive attitude towards automatization for their jobs. While bias is to this day the first and most important problem to solve for the recruitment process, there are less impactful but still meaningful areas that provide a better quality of life for HR experts. According to the same report from 2019, many startups offer services for auto transcribing calls with candidates to those creating personalized and easy-to-use ATS systems [9]with Resume matching and search functionality built into them, such as TalentLyft [10].

3. DATASET STRUCTURE

Our focus audience for our research, and potentially a product one day, were our peer researchers, Data Scientists, and Machine Learning Engineers. To tackle this problem robustly, we first need a high-quality, up-to-date dataset. Since most of the job titles we are interested in were created and established in the past couple of years, we needed new job descriptions that recently encapsulate those titles and job-related skills created (e.g., TensorFlow, PyTorch, Fast API).

Finding a decent job description dataset is difficult, especially an updated one. We managed to find two relevant datasets ready to be reused/downloaded. The first two datasets we encountered were available for download from Kaggle Datasets [11]. The first was the UK only [12] dataset, and the second was the US only [13], which focused on the Monster website. They were created three/four years ago and did not contain any relevant jobs for our research. Even though these datasets were not used in our further work, they helped us detect a need in the research community and develop a list of features we wanted to add to our version when making it available.

Since we couldn't use any of the existing datasets, we created our system to gather the most relevant jobs around the internet. This project had two challenges that our team had to overcome - creating a reliable method for collecting raw information job descriptions from websites and automatically extracting the information we wanted from the downloaded HTML pages.

3.1. THE JOB DESCRIPTION SCRAPING SYSTEM

Every day, we can expect more than 10 million new job posts across all hiring platforms worldwide [14]. With these numbers constantly increasing, finding relevant jobs for job seekers becomes harder day-in-day-out. To tackle this growing issue, we created a fast and reliable scraping system that gathers Data Science related jobs from the most popular job boards on the internet and saves them to our database.

The first iteration of our system included only Indeed, LinkedIn Jobs, and Glassdoor - three of the biggest job boards out there. However, when Covid 19 started, we pivoted to more remote-friendly options and added TapWage, Remotive, RemoteOK, and a few smaller websites to tackle this newly recognized problem.

While building the system, there was one technology limitation to overcome - anti-scraping systems, which detect non-user behaviour and block those connections to their website. We overcame this by creating stochastic workers who try to mimic a human user while browsing a web page. Those agents make intentional mistakes, navigate randomly, and wait for a random period to create as human-like a session as possible. We used the Selenium testing framework [15] to achieve this effect.

The first version of the infrastructure had N agents in place - one for each website being scraped. Each agent would scrape in parallel and populate the database at its own pace, as demonstrated in Figure 1. This works well with a smaller load of downloaded data, but bottlenecks occur when we increase the workload for more prominent websites.

The first version worked well on a smaller scale, where website scrapes are similar in size (the number of jobs per minute), and the response time from each website to our system is similar. The complexity and first bottlenecks arise with every new website added to the system, and the number of variables that we need to handle increases exponentially.

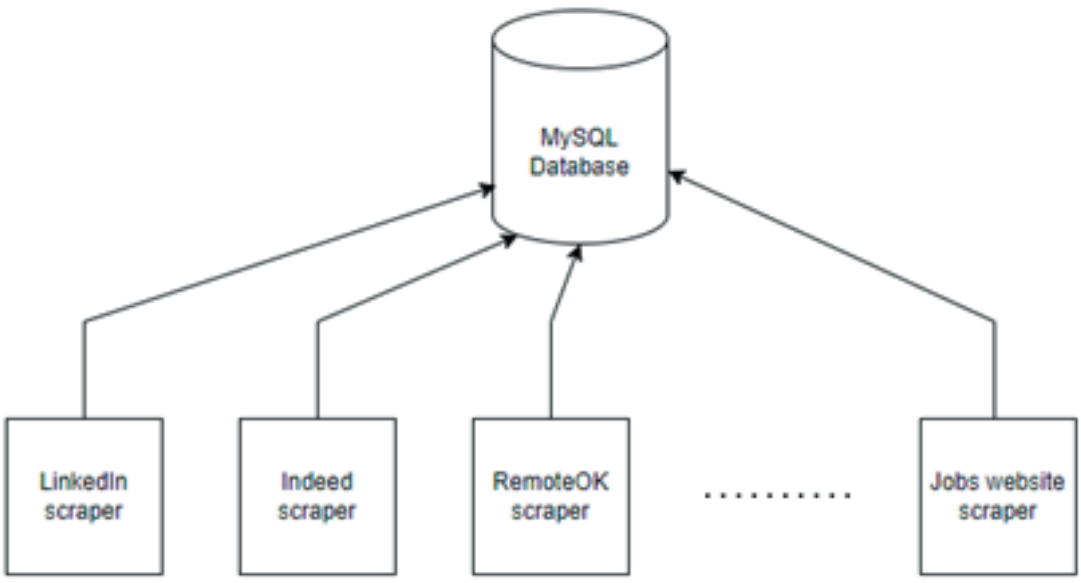

Figure 1 - The first version of the scraping system - without the central controller

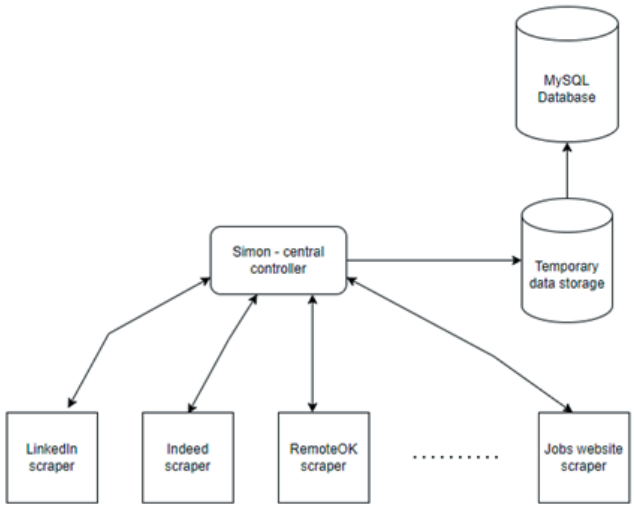

Figure 2 - Scraping system with central control unit for data syncing

To handle the database bottleneck and time variability, we added one more agent to the system called "Simon" from the popular children's game "Simon says, as seen in Figure 2. "This agent will conduct a sync between all other agents and take care of data batching, delays in data gathering between websites, and report errors directly to our Slack channel when such errors occur.

The first version of the system managed to top about 20.000 jobs per day. By adding a central authority agent (Simon), we increased this number to more than 50.000 jobs per day (~ 35 jobs added to the database per minute).

3.2. AUTO-LABELING USING REGEX

When operating at optimal capacity, our system produces about fifty to fifty-five thousand job descriptions per day. If we want to train a machine-learning algorithm on that data, we need to label it first. As our team had only two researchers, labeling all those job descriptions ourselves or paying a third-party company was not an option. We created an automated human-in-the-loop system based on regex rules to solve this challenge.

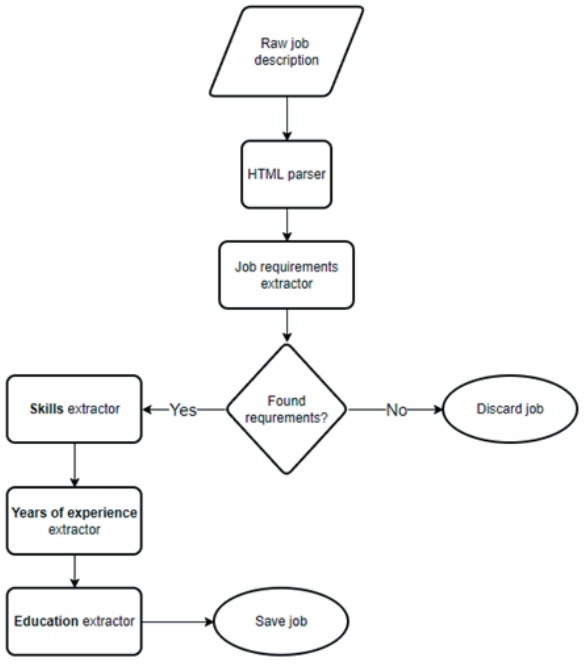

Figure 3 - Job description analyser algorithm flow

Our goal was to extract a set of job requirements for each job downloaded using an end-to-end algorithm. Here is the list of features our system searchers:

- Years of experience
- Education level (Bachelor's, Master's Ph.D.)
- Education area (e.g., Computer Science, Mathematics, Economy)
- Substitution (e.g., experience for education or education for experience)
- Soft skills (e.g., presentation, communication)
- Hard skills (e.g., Python, TensorFlow, C++, R)

We created a set of functions that check HTML structure and put the raw text through Regex [16] filters to handle each of these points, and as the output, the system produces filtered requirements for each of the classes defined in the previous paragraph.

Each filter has the same sequence of checks, with the Regex being the only difference. Here is the breakdown of how each filter processes the raw HTML:

1. Custom formatting checker approves the page or discards it
2. HTML tags filter finds a specific part of the page for job requirements
3. Job requirements are again put through a formatting checker for the second approval
4. Requirements are split into lines
5. Each line goes through a language-based classifier that determines with regex filter to apply to it
6. Regex filter is applied based on the classifier results

The system is very robust in most cases, and after several iterations, it didn't need a human-in-the-loop for additional checks. Regex checkers can capture most of the dataset and automatically label it, but they still miss it from time to time. Even though we can safely rely on the Regex version of the solution, it has its downsides. Besides its instability, it became overly complex and fast, and adding more checks required more time. The upside of the system is its memory requirement and inference speed.

4. CONCLUSION

As described in section 3.2, we built a system based on the Regex rules to extract relevant information from job descriptions. One of its drawbacks is its lack of generalization when confronted with new data points (job descriptions).

One of the main reasons for utilizing machine learning algorithms is as a solution for handling previously unseen categories of new user data, which was the main obstacle for the Regex-based system. For some rules to be detected, they had to be coded manually and added so that future data is parsed correctly. Constantly adding new rules does cover more and more edge-cases that the dataset encounters, but it also increases the complexity of the code. The main generalization issue for our task is finding the years of experience needed in a job post. The Regex system can easily recognize regular cases (e.g., 1-2 years of experience, one to two years of experience) but lacks data understanding, shown in these examples: up to two years of experience and one year in the industry, etc. When detection is context-based, the rules-based system does not perform well.

To handle context-based examples and handle the ever-rising complexity of the code, we turned this problem into a text classification issue called NER (Named Entity Recognition) [17]. In NER, we try to build a model to classify each word (or a sequence of words) into specific objects (classes). In the sentence "*I am living in London and working at DeepMind,*" - a NER model would have to recognize two distinct objects - London as a CITY/PLACE and DeepMind as an ORG (organisation). In the task of extracting relevant requirements from job descriptions, we created custom NER tags suitable for us:

Since our initial dataset was rather small, all our experiments were performed using a novel technique in Natural Language Processing called Transfer learning. We experimented with two different models, BERT [18] and its smaller version DistilBERT [19]. As the accuracy of both models was roughly the same (99.5% for BERT and 99.1% for DistilBERT), to make the final decision, we tested both models on newly scraped data. BERT performed much more accurately and showed better generalization on new data formats and job description syntaxes.

Tag name	Tag description
B-SKILL	The first word of a skill
I-SKILL	Tag for words beyond the first one (if a skill has multiple words)
B-EXPERIENCE	The first word of experience
I-EXPERIENCE	If an experience has multiple words (e.g., two or three)
B-DEGREE	The first word of a degree (e.g., Ph.D.)
B-DEGREE_MAJOR	The first word of a degree major
I-DEGREE_MAJOR	If a degree major has multiple words (e.g., Computer science)

Table 1 - NER Tags for Job Description dataset

With the trained BERT model, we solved the first problem, which is generalization of our rules to unsee data and context-based data. However, the model was still close in the number of lines of code to the original set of rules that relies on Regex only. To handle this downside, we used an open-sourced library called HuggingFace [20] that allows us to load any pre-trained language model in a single line of code. Using HuggingFace, we fine-tune the model to our dataset and deploy it in less than 100 lines of code, which officially handles the second drawback: the complexity of the code and readability.

5. CONCLUSION AND FUTURE WORK

The main objective of this paper is to demonstrate a data collection system for job descriptions and apply a machine learning model to extract relevant information from them, which can be used for better job ad personalization or creating an advanced job search engine.

This paper demonstrates an application of new NLP models called Transformers on information extraction tasks from job descriptions as a data source. We managed to get 99.5% accuracy on the NER application using the BERT model, which is not a trivial task since all categories (classes) being recognized are different. For example, skills such as TensorFlow, PyTorch, Artificial Intelligence are pure text. At the same time, years of experience can be a combination of special characters (e.g., 9 - 10), text (e.g., nine to ten), or a combination of both (e.g., zero - 1).

In our further work, we would like to extend this application to Resumes/CVs and test if we can scale the current model on different data sources with similar data distribution. Furthermore, if the Resume experiment goes well, we would like to extend our research on the matching theory between candidates and open job positions.

6. REFERENCES

[1] "Optimizing resumes for key pair matching systems," [Online]. Available: https://novoresume.com/career-blog/resume-keywords-how-to-use-them. [Accessed 27 March 2022].

[2] T. Zimmermann, L. Kotschenreuther and K. Schmidt, "Data-driven HR - Résumé Analysis Based on Natural Language Processing and Machine Learning," vol. arXiv:1606.05611, 2016.

[3] V. S. Kumaran and A. Sankar, "Towards an automated system for intelligent screening of candidates for recruitment using ontology mapping (EXPERT)," *International Journal of Metadata, Semantics and Ontologies*, pp. 56-64, 2013.

[4] H. L. P. C. A. X. L. Yiou Lin, "Machine Learned Resume-Job Matching Solution," *arXiv*:1607.07657, 2016.

[5] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *arXiv*:1603.02754, 2016.

[6] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[7] Y. Wang, Y. Allouache and C. Joubert, "Analysing CV Corpus for Finding Suitable Candidates using Knowledge Graph and BERT," in DBKDA 2021, *The Thirteenth International Conference on Advances in Databases, Knowledge, and Data Applications*, Valencia, Spain, 2021.

[8] J. Wright, "The impact of artificial intelligence within the recruitment industry: Defining a new way of recruiting," 2019. [Online]. Available: https://www.cfsearch.com/wp-content/uploads/2019/10/James-Wright-The-impact-of-artificial-intelligence-within-the-recruitment-industry-Defining-a-new-way-of-recruiting.pdf . [Accessed 26 March 2022].

[9] "Applicant tracking system - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Applicant_tracking_system. [Accessed 20 March 2022].

[10] "TalentLyft," TalentLyft, [Online]. Available: https://www.talentlyft.com/en. [Accessed 20 March 2022].

[11] "Datasets," kaggle.com, [Online]. Available: https://www.kaggle.com/datasets. [Accessed 21 March 2022].

[12] "Text Analytics Explained- Job Description Data," 2019. [Online]. Available: https://www.kaggle.com/code/chadalee/text-analytics-explained-job-description-data/data. [Accessed 21 March 2022].

[13] "Monster.com USA only job descriptions dataset," 2018. [Online]. Available: https://www.kaggle.com/code/residentmario/exploring-monster-com-job-postings/data. [Accessed 21 March 2022].

[14] J. Marino, "Must-know job websites statistics," [Online]. Available: https://www.huemanrpo.com/resources/blog/must-know-job-website-statistics. [Accessed 23 March 2022].

[15] "The Selenium Browser Automation Project," Selenium, [Online]. Available: https://www.selenium.dev/documentation/. [Accessed 27 March 2022].

[16] "Regex (Regular Expression) search patterns," Microsoft, [Online]. Available: https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-reference. [Accessed 27 March 2022].

[17] J. Li, A. Sun, J. Han and C. Li, "A Survey on Deep Learning for Named Entity Recognition," *arXiv:* 1812.09449, 2018.

[18] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv:1810.04805, 2018.

[19] V. Sanh, L. Debut, J. Chaumond and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv:*1910.01108, 2019.

[20] "Open-sourced deep learning library," HuggingFace, [Online]. Available: huggingface.co. [Accessed 27 March 2022].