# Integrated NLP and Visualization System for better Job Market Preparation

Anvita Brahma, Rishidhar Duvvuru, Hriday Harlalka, Sarthak Khare, Yuexi Liao, and Junyuan Quan
*CSE 6242 Team 15, Georgia Institute of Technology, Atlanta, GA 30332*
(Dated: December 4, 2023)

## I. INTRODUCTION

With the expanding diversity of the labor market and the workforce, companies are increasingly adopting techniques like resume screening software and Applicant Tracking System (ATS)[3] to filter and rank job applicants[8]. Given this trend, a job market preparation system can be invaluable to applicants looking to improve their application and increase their chances of success[17]. This project seeks to empower job seekers with a personalized, efficient, and informed approach, transforming the job search process into a more manageable and successful effort.

## II. PROBLEM DEFINITION

Creating a resume is a crucial step in today's job market. We plan to offer an easy-to-use, visual tool to help users understand what to include in their resumes and what skills they should focus on learning with regional insights. Additionally, our tool will recommend users with similar profiles that closely match their search criteria based on contextual text similarity and keyword extractions.

## III. LITERATURE SURVEY

The practice of web scraping has gained significant prominence as a method for acquiring substantial volumes of data from online sources. Previous research[11] presents an extensive examination of web scraping methodologies, encompassing the discussion of BeautifulSoup, a Python package designed for the extraction of data from HTML and XML texts[19].

Other than that, Selenium[30] is a software tool that enables automated web browsing by emulating human browsing behavior. The incorporation of Selenium WebDriver with programming languages such as Python facilitates the execution of intricate automation tasks. One of the most impressive features of Selenium is its automated interaction with the web pages, that users can observe the actions of the code on the website in real-time.

The utilization of application programming interfaces (APIs) for the purpose of data collecting[15] presents itself as a viable alternative to conventional scraping techniques. Assuming the target website has an API service, this methodology has the potential to enhance the stability and structure of data retrieval. Once the data has been collected, data manipulation and storage are carried out using tools such as pandas in the Python programming language. The pandas library can subsequently be used to generate a data frame using the extracted job listing information and store it as a CSV file[9].

While working with job postings data from around the globe, the efficient processing of extensive multilingual job description data is crucial. Our approach focuses on integrating the googletrans library[12] as a language detection tool for data cleaning and context preservation within the domain of job listings. However, it is important to acknowledge the limitations that come with various translation libraries in this field[24]. Some of these libraries come with word limits and usage restrictions, potentially raising concerns about translation accuracy, especially when dealing with specific language pairs. Additionally, the breadth of language support can vary across different libraries. Given the diverse range of languages found our job data, precise language detection is essential for ensuring data quality and its subsequent utility within our model.

Once we have the clean data, it's necessary to extract the information from it. Current methodologies for extracting valuable information from text, such as job descriptions, heavily rely on Natural Language Processing (NLP). NLP transforms unstructured text into structured data and employs keyword extraction techniques to derive pertinent information. One such technique, GloVe[20], which is a word embedding method, has gained prominence in this field, as it enhances our ability to understand keywords in context.

Incorporating downstream processes that leverage contextual embeddings obtained through NLP, Long Short-Term Memory Networks (LSTMs)[14] emerge as a compelling choice. LSTMs excel in modeling sequential data, making them particularly well-suited for tasks like keyword extraction from text or speech. Several pieces of literature have explored the utilization of LSTMs for this purpose[27][10][5], demonstrating their efficacy in capturing the sequential nature of textual/speech data.

Other techniques that involve NLP for keyword extraction are the usage of TF-IDF[21], BERT[26][25][22][28] or its variant in the form of jobBERT[7]. However, despite the wealth of research in the field of keyword extraction[2][4][18][26], there remains a notable gap – the absence of an interactive visual tool tailored for job seekers. Our project seeks to fill this void by creating pre-

cisely such a tool. By presenting information in the form of an interactive visual interface, we aim to significantly enhance the job search experience and address the challenges faced by individuals in today's competitive employment market. This tool will not only streamline the search process but also provide a much-needed solution to meet the demands of the modern job-seeking landscape.

Visualization plays a critical role in our project as it provides insights that are easily digestible. As demonstrated in[29], D3 maps with scatter plots beside original maps offer valuable insights. Although their research was based on data from Facebook and Twitter, we anticipate our broader dataset will deliver more pertinent results. Additionally, word clouds have emerged as a favorite visualization method. Notably, the Word Cloud Explorer by Heimerl, Florian, et al.[13] leverages word clouds to highlight term relations and provide detailed term information. Some other possible visualization methods include combining trend charts with word clouds to illustrate temporal content evolutions in a set of documents, offering potential visualization strategies for our project[16][6].

## IV. PROPOSED METHODOLOGY

**Primary innovations in our approach include:**

Providing a job fit score to help job seekers fit their resume to job postings. We also use this score on the back end to improve job recommendations given resumes and adjustable parameters from users.

By employing a combination of POS pattern matching, tokenization, GloVe embeddings, and LSTM modeling, our methodology delivers a robust framework for keyword extraction that is also scalable. Our intuition is that this will perform better than the basic TF-IDF approach which we test during experiments and evaluation.

Finally, presenting our unique and intuitive dashboards that follow Human-computer interaction (HCI) design principles will make the entire process easy for job seekers.

**Further details on the above process follow next:**

The first step in our methodology involves data preprocessing, which is essential to ensure that the job descriptions are structured and organized for subsequent analysis. We employ part-of-speech (POS) pattern matching to identify and segment relevant textual elements. Specifically, we target four primary patterns: Noun Phrase basic, Noun phrase variation, Verb Phrase, and Noun between commas. These patterns are strategically chosen to capture various forms of keywords within job descriptions. By utilizing these patterns, we create a pool of n-gram phrases that serve as potential candidates for keywords.

We started by gathering data from Kaggle Job listings and LinkedIn API[23]. As the project progressed, we identified a need for a more expansive dataset to achieve our objectives. This led us to use web scraping tools, such as BeautifulSoup and Selenium, to extract additional job listings directly from Glassdoor. This method proved to be a valuable contingency, enabling us to extract up to 900 job postings from Glassdoor, which encompassed diverse global tech profiles, thereby enriching our data set. We finally did not need to augment our data since our model was performing satisfactorily.

Our data cleaning process was multifaceted, involving the resolution of challenges such as job descriptions in various local languages and HTML parsing, missing coordinates, classification of job categories, and the presence of diverse job titles. To address these issues, we employed open-source Python libraries, "bs4", "detect lang" and "googletrans," automating the detection and translation of job descriptions into English and achieving a high data retrieval rate. Furthermore, we tackled the issue of missing coordinates in roughly 20% of the dataset by harnessing location names, open-source GeoJSON data, and the GeoPandas library to accurately retrieve these missing geospatial coordinates.

The process of determining job categories involves a meticulous examination of job descriptions and titles for specific keywords. A role is labeled 'internship' if the term 'internship' or 'intern' is present in either the description or title; otherwise, the default category is 'full time'. Concurrently, a dictionary maps keywords to the appropriate 'entry' or 'senior' experience levels, scanning the amalgamated job description and title text for these indicators. Keywords such as 'entry,' 'graduate,' or 'trainee' suggest an 'entry' level, while 'senior,' 'manager,' or 'lead' denotes a 'senior' level, thereby guiding the classification.

Recognizing the distinct terminology used for similar profiles across different organizations, we devised a dictionary of standardized job titles. Applying the "fuzzywuzzy" library, we calculated token set similarity scores to intelligently categorize job titles into a standardized format. Our threshold for categorization was thoughtfully set at a similarity score exceeding 70, ensuring the uniformity of job titles within our dataset. We will further enhance the profile matching using keywords extracted from our model to get a more accurate representation. The framework that we created for the keyword extraction has been taken from a medium post[1].

Having extracted n-gram phrases, the next stage in our methodology is tokenization and padding. Tokenization involves splitting the n-gram phrases into individual words and phrases, creating a structured and organized representation of the text data. To ensure that the data conforms to a uniform input format for our subsequent deep-learning model, we employ padding to fix the sequence length at 20. This step ensures that the text data is well-prepared for the subsequent analysis by the LSTM model.

Incorporating external knowledge and enhancing the model's understanding of the text is crucial for accurate keyword extraction. To achieve this, we introduce an em-

bedding matrix utilizing pre-trained GloVe (Global Vectors for Word Representation) word vectors. Each unique word in the corpus is linked to a 100-dimensional GloVe vector. This linkage enables the model to comprehend the nuanced semantic relationships between words and phrases within the job descriptions, thereby improving the quality of keyword extraction.

The heart of our methodology lies in the training of a Long Short-Term Memory (LSTM) model. LSTMs are known for their ability to capture long-range dependencies and sequential patterns in data. In this approach, the LSTM is trained on a dataset obtained from an online source. The training process equips the model with the capability to understand the sequential context of words and phrases within job descriptions, enabling it to differentiate between keywords and non-keywords effectively.

To classify and identify keywords within job descriptions, the output of the LSTM model is processed through a dense layer that employs a sigmoid activation function. The sigmoid function assigns a score between 0 and 1 to each word or phrase, with values exceeding a predefined threshold (in our case, set to greater than 0.70) denoting keywords. This step is crucial in distinguishing and labeling relevant skills and qualifications mentioned in the job descriptions.

Finally, we come to the job score calculation. We computed this as a scalar product of keyword vectors, where each keyword is assigned a score as its magnitude, to determine the relevance or similarity between roles and resumes. We ran this model on a dataset of resumes to recommend jobs and got promising results.

The above paragraphs explained our innovative implementations in logic and an effective combination of models. In the next few paragraphs, we want to showcase the unique visualization element in our UI which is better than existing markets since we provide easy and unique data points in our dashboards.

To visualize the data we acquired, and the output we got from keyword extraction, we have designed two interactive dashboards using Tableau. There are 2 main views that we have created namely - Map(Figure 1), and Job Recommendation(Figure 2).

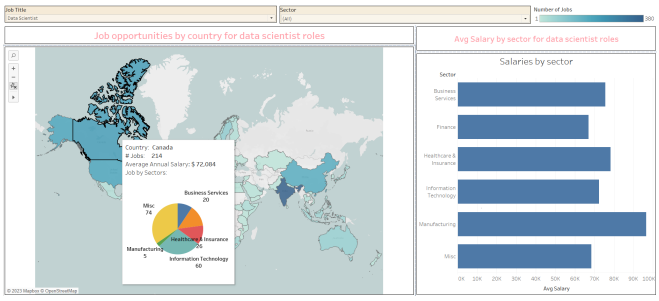


Figure 1. Comparative Analysis Dashboard of various job opportunities and Sectoral Salary Benchmarks

The Map view is an interactive geographic heat map of countries, color-graded on expected salary by region. This visual method allows users to quickly grasp which regions offer higher salaries, effectively communicating geographical salary trends. Additionally, users can hover over each country to view the exact average salary, the number of jobs available, and a pie chart that shows the distribution across sectors in the industry(as seen in Figure 1). Adjacent to the map, a bar chart provides more details about the average salaries by sector for the selected job title. The purpose of showing this bar chart is to compare the sectoral salary based on selected job titles. The longer bars indicate the sector to which the selected job belongs has a higher average salary. For this dashboard, we have job title and sector as the filters to control the display of the visualization. With different selections of the filters as well as the country (or multiple countries), the displayed information will be altered.

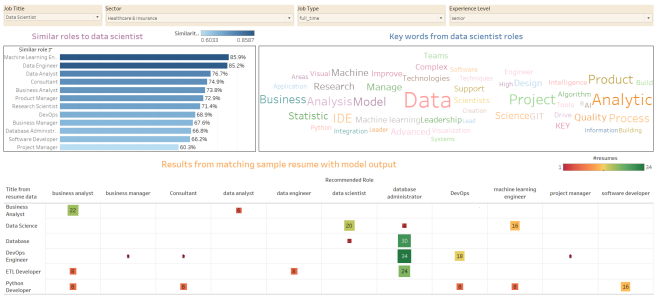


Figure 2. Occupational Analysis Dashboard for jobs and related fields

The Job Recommendation view is an amalgamation of 3 components: Similar Roles, Keyword Cloud, and Resume Based Recommendation.

Similar Roles are ranked based on the Job Match similarity score we develop. The keyword cloud is generated from the LSTM model, which emphasizes the frequency and importance of skills extracted from job descriptions. The prominence of terms visually conveys the demand for these skills. The resume recommendation is a static visualization of the sample resumes. The top row represents the Recommended Role and the leftmost Column corresponds to the applicant's original role(s). Since we could not implement a dynamic web app to include a way to accept resumes, we have provided this view to demonstrate the effectiveness of our model and how it is able to recommend reliably. Complementing these visual tools are interactive filters categorized by "Job Title", "Sector", "Job Type", and "Experience Level" empowering users to customize the data display to fit specific queries and analyses.

The dashboard's layout divides the space into distinct sections for each type of visualization, creating an organized narrative. A consistent color scheme with different shades indicates various data metrics, such as salary ranges. Typography is purposefully utilized to draw attention to the most prevalent skills within the job market. The ability to interact with and customize the data dis-

play elevates the dashboard from a static report to an engaging analytical tool.

## V. EXPERIMENTS/EVALUATION

To evaluate our model performance and make sense of the outputs, we use various statistical metrics. Our LSTM model, trained on a dataset sourced from the internet and split into an 80:20 ratio for training and testing, demonstrated promising results. During training, the model achieved a final accuracy of 76.8%. The progression of accuracy and loss throughout epochs is visually represented in Figure 3.

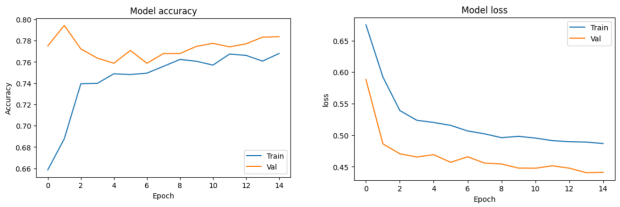


Figure 3. LSTM Model Accuracy and Model Loss

Upon testing, the model maintained a commendable accuracy of 74.1%. The confusion matrix (Figure 4) provides a detailed breakdown of the model's classification performance.

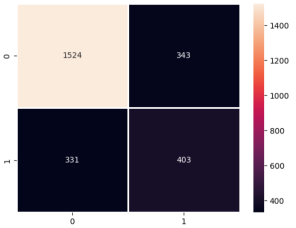


Figure 4. Confusion Matrix of LSTM model

The analysis of the confusion matrix reveals important insights into the performance of the keyword extraction model. With a precision of 54%, the model correctly identifies keywords about 54% of the time. Simultaneously, a recall of 55% indicates that the model captures slightly more than half of the actual keywords present in the text. While this demonstrates a commendable balance between precision and recall, there is noticeable room for improvement in both aspects.
More importantly, the model exhibits a high specificity of 82%, showcasing its effectiveness in correctly identifying non-keywords and avoiding false positives.
The F1 score, standing at 54.5%, reflects the overall performance, and while reasonable, it suggests an opportunity for enhancement, particularly in terms of recall. In summary, the model demonstrates a respectable equilibrium between precision and recall, a good specificity to meet our needs. There exists room for refinement, but it is a fairly comprehensive keyword extraction implementation.

In parallel, we applied the TF-IDF approach to a sample job description, initially yielding singular words. The output included words like: 'learning', 'machine', 'experience', 'bring', 'design', 'problems', 'role', 'algorithms', 'big', and 'complex'. However, this is too simple and naive. The LSTM model on the other hand provides phrases that would provide more context to the end user as depicted in Figure 5. The outputs from the model also provide scores for the phrases that were selected hence allowing for a further quantified view of the keywords.

| | |
|---|---|
| product managers | 0.786768 |
| pytorch background robotics | 0.778590 |
| big data technologies | 0.778454 |
| code reviews | 0.768926 |
| tensorflow pytorch background robotics | 0.765491 |

Figure 5. Sample Output from LSTM

While the output enhances context for end-users, some generated phrases tend to be excessively verbose, including extraneous information.

Example Outputs:

1. "Remote human guidance to create intelligent robots."

2. "Computer science equivalent experience with bonus qualification."

Thus, from these experiments, we conclude that our method performs better than the basic TF-IDF as a threshold, and evaluation metrics reflect its reliability. The model along with our job score functionality is verified by a human-level sanity check by simply observing that the outputs make sense. In a more mathematical sense, the job recommendation on the resume data reflects a matching score for recommended jobs, given the base jobs of a user. The dashboards shown earlier (Figure 1 and Figure 2) confirm the same, visually.

Final observation during our trial was regarding computation time and model performance. Given the fact that our model was returning reliable outputs even within the limited data, along with the hardware and time constraints of running the model for 130k data, we decided to train our data on 30k rows. The computation time for data pre-processing took over 20 hours, and that for training 10k rows of data took 12 hours. Considering the trade-off between computation time and model performance, we decided to train on 30k data points eventually.

## VI. CONCLUSIONS AND DISCUSSIONS

In conclusion, this project signifies a significant advancement in the field of job skill extraction and visualization, leveraging NLP and Deep Learning, alongside powerful visualization tools. Our approach showcases the potential for not only accurately identifying crucial job-related keywords but also visualizing these insights in a comprehensible manner that can give deeper and meaningful suggestions to job seekers. The practical implications of this project are far-reaching, with the ability to streamline the job search process and facilitate more informed decisions. We acknowledge certain limitations of data quality and availability of real-time data which is crucial in the ever-changing landscape of the job market. A future extension would be developing a dynamic web app that can accept user's resumes and give live feedback. Another extension to potentially tackle the limitation could be to have APIs that collect and update real-time job data.

Having said that, our current model stands as a very strong tool that is able to provide sophisticated recommendations. Our pre-processing and model integration methodology is clean and can be reproduced and scaled to positively impact job seekers in the market.

**All team members have contributed a similar amount of effort.**

## REFERENCES


[1] Reme Ajayi. *Job Skills Extraction from Data Science Job Posts*. post retrieved from Medium, `https://medium.com/@Olohireme/job-skills-extraction-from-data-science-job-posts-38fd58b94675`. 2021.

[2] Mathieu Bastian et al. "Linkedin skills: large-scale topic extraction and inference". In: *Proceedings of the 8th ACM Conference on Recommender systems*. 2014, pp. 1–8.

[3] S Bharadwaj et al. "Resume Screening using NLP and LSTM". In: *2022 international conference on inventive computation technologies (ICICT)*. IEEE. 2022, pp. 238–241.

[4] Ricardo Campos et al. "YAKE! Keyword extraction from single documents using multiple local features". In: *Information Sciences* 509 (2020), pp. 257–289.

[5] Guoguo Chen, Carolina Parada, and Tara N Sainath. "Query-by-example keyword spotting using long short-term memory networks". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, pp. 5236–5240.

[6] Weiwei Cui et al. "Context preserving dynamic word cloud visualization". In: *2010 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE. 2010, pp. 121–128.

[7] Jens-Joris Decorte et al. "Jobbert: Understanding job titles through skills". In: *arXiv preprint arXiv:2109.09605* (2021).

[8] Ketki V Deshpande, Shimei Pan, and James R Foulds. "Mitigating demographic Bias in AI-based resume filtering". In: *Adjunct publication of the 28th ACM conference on user modeling, adaptation and personalization*. 2020, pp. 268–275.

[9] Nicholas Diakopoulos. "Algorithmic accountability reporting: On the investigation of black boxes". In: (2014).

[10] Xiaoyu Duan et al. "OILog: An online incremental log keyword extraction approach based on MDP-LSTM neural network". In: *Information Systems* 95 (2021), p. 101618.

[11] Daniel Glez-Pena et al. "Web scraping technologies in an API world". In: *BRIEFINGS IN BIOINFORMATICS* 58.5 (2015), pp. 62–71.

[12] SuHun Han. *googletrans 3.0.0*. data retrieved from PyPI, `https://pypi.org/project/googletrans/`. 2020.

[13] Florian Heimerl et al. "Word cloud explorer: Text analytics based on word clouds". In: *2014 47th Hawaii international conference on system sciences*. IEEE. 2014, pp. 1833–1842.

[14] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[15] V. Krotov and L. Silva. "The Use of APIs for Data Collection on the Internet". In: *The Review of Business Information Systems (RBIS)* 20.1 (2016), pp. 49–56.

[16] Bongshin Lee et al. "Sparkclouds: Visualizing trends in tag clouds". In: *IEEE transactions on visualization and computer graphics* 16.6 (2010), pp. 1182–1189.

[17] Hussain Falih Mahdi et al. "Job descriptions keyword extraction using attention based deep learning models with bert". In: *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*. IEEE. 2021, pp. 1–6.

[18] Rada Mihalcea and Paul Tarau. "Textrank: Bringing order into text". In: *Proceedings of the 2004 conference on empirical methods in natural language processing*. 2004, pp. 404–411.

[19] Ryan Mitchell. *Web scraping with Python: Collecting more data from the modern web.* " O'Reilly Media, Inc.", 2018.

[20] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[21] Shahzad Qaiser and Ramsha Ali. "Text mining: use of TF-IDF to examine the relevance of words to documents". In: *International Journal of Computer Applications* 181.1 (2018), pp. 25–29.

[22] Yili Qian, Chaochao Jia, and Yimei Liu. "BERT-based text keyword extraction". In: *Journal of Physics: Conference Series*. Vol. 1992. 4. IOP Publishing. 2021, p. 042077.

[23] Tom Quirk. *Linkedin API for Python*. data retrieved from GitHub, `https://github.com/tomquirk/linkedin-api`. 2023.

[24] Philipp Schreiber. *Translate long PDF-Reports in Python*. article retrieved from Towards Data Science, `https://towardsdatascience.com/translate-long-pdf-reports-in-python-eab3be08ceb4`. 2022.

[25] Matthew Tang et al. "Progress notes classification and keyword extraction using attention-based deep learning models with BERT". In: *arXiv preprint arXiv:1910.05786* (2019).

[26] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[27] Y Wang and J Zhang. "Keyword extraction from online product reviews based on bi-directional LSTM recurrent neural network". In: *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. IEEE. 2017, pp. 2241–2245.

[28] Ao Xiong et al. "News keyword extraction algorithm based on semantic clustering and word graph model". In: *Tsinghua Science and Technology* 26.6 (2021), pp. 886–893.

[29] Nur Azmina Mohamad Zamani et al. "Visualization of job availability based on text analytics localization approach". In: *Indonesian Journal of Electrical Engineering and Computer Science* 16.2 (2019), pp. 744–751.

[30] A. Zeller. "Automated Web Application Testing with Selenium". In: 16.2 (2012), pp. 72–80.