Utilising NLP Techniques for Job Search and Career Guidance: Intelligent Matching of CVs with Job Titles

Introduction to Text Mining and Natural Language Processing

Angelo Di Gianvito, Oliver Gatland, Sebastian Boxho

March 18, 2024

Introduction

Job recommender systems are helpful tools frequently used by job seekers when searching for new professional roles, however, the technical process behind these systems has not kept pace with the current advancement in machine learning techniques. Although there are various job search tools and portals online, the process of finding appropriate roles for job seekers is rarely streamlined, and there are few automated tools which can help with this process. Searches on job market websites often rely on keyword matching and simplistic information retrieval methods, which may frequently return irrelevant roles for prospective candidates. Alongside this, current job search tools rely on candidates themselves to identify their most suited role before searching. This may work well in efficiently clearing the worker to employer market if candidates have perfect knowledge of the market they are entering, however, this is often unlikely to be the case.

To this end, we aim to design a model which reads a resume and automatically suggests the job title that would be most appropriate for the candidate. This can be considered as a typical multi-label classification problem, where each resume is labelled with an ideal job match, and a model is trained to try and predict this match. After determining the most effective model at matching resumes to job descriptions, we design an interactive tool which allows BSE students to upload their resumes, and outputs the job title which suits them best. The tool also returns a list of required skills for the most appropriate job title, and identifies skills the candidate is missing for their most suited role.

The rest of the report is structured as follows: we first review the existing literature on resume to job matching, before explaining the datasets we use for the project. We then outline the technical methodology used to match resumes to job descriptions, followed by an outline of the performance of the models we used. Our preferred model is then used to create the interactive tool for BSE students, and an outline of how to use the tool is included in the appendix.

Literature Review

Broadly, there are a few areas of literature related to the parsing of resumes and matching them with appropriate job descriptions. Resumes can be considered as semi-structured data as they often follow a similar format, so some research focuses on techniques for parsing semi-structured data into databases, mainly for the use of organisations filtering through prospective candidates. A commonly used approach is an ontology-based method for processing resumes, which uses ontology to assist with the information extraction process. Celik & Elci [1] use a system to process resumes by converting them into plain text, segmenting the sentences and applying an ontology knowledge base to find concepts in each sentence. Here an ontology defines classes, properties, and relationships, helping to define a way of

storing unstructured data in a structured manner. Although this approach is interested, we are mainly focusing on the use of NLP tools, so we will not consider an ontology based approach.

The main area of literature that is related to our project is that which focuses on the use of basic NLP tools to match resumes with job descriptions. This is often considered from one of two sides; either an employer, who needs to filter applications to find appropriate candidates to hire, or a job recommender system, which attempts to filter job descriptions based on a given resume to find an appropriate match for a candidate. The majority of literature uses basic text pre-processing and vectorization to calculate similarity metrics between a set of resumes and job descriptions. The matching performance is often viewed as a typical classification problem, where the resumes are labelled with a most suited job, and a model is trained to predict the class of the most suited job for unseen resumes. Motta [2] tries to improve on this baseline by optimizing the vector based encoding methods of TF-IDF and LDA. They find that optimizing the parameters of the encoding technique leads to better accuracy when using cosine similarity to match resumes to job descriptions. Cosine similarity is commonly used in the literature to match resumes to job descriptions. Kandji and Ndiaye [3] also use cosine similarity after finding matching based on key phrases is an ineffective way of classifying resumes. They also aid their classifications using spaCy's Named Entity Recognition (NER) function, which can identify and classify named entities within sentences, to identify skills within resumes.

Some other research uses more advanced embedding techniques. Alsaif et al. [4] uses traditional NLP techniques to process their resume data, before applying Word2Vec to create embeddings. The key idea behind Word2Vec is to represent text in such a way that words with similar meanings are close to each other in vector space. Using this method allows the context of the words to be considered in the matching process, and the research finds this type of embedding combined with cosine similarity matching performs the best, with an overall accuracy of 0.8. Maheshwary and Misra [5] use more advanced machine learning algorithms to match the resume and job description. They try to improve on the current state of the art models by using a siamese network consisting of a pair of identical convolutional nerual networks. Here, they use Doc2Vec to form the vectors before feeding them into the input layer of the networks. They compare the results of this with cosine similarity and find improvements on all accuracy metrics, achieving a F1 score of 0.8 for their best model.

The majority of literature focuses on testing models with existing resumes, rather than actually implementing a tool which can be used for matching. However, Guo et al. [6] focused on distilling these methods into a usable tool which scrapes data from various job market websites to match a resume to live job postings. They use a combination of an ontology of skills with machine learning techniques to measure the distance between these skills. Combining various features they extract from the text, they compute distances between job descriptions and resumes as weighted averages, and achieve an accuracy score of around 0.8

based on some resumes they labelled manually.

Although the task of matching resumes with job descriptions is often viewed as a typical multi-label classification task, it is notable that many common classification machine learning models, such as logistic regression and random forest, are not considered in previous research. In this project, we will build on previous research by examining the effectiveness of these models at classifying resumes.

Data

We required two main sources of data for our project: Resumes and Job Descriptions. Below we outline the sources and data in detail, before displaying some findings from our exploratory data analysis.

Resumes

We retrieved resume data from Kaggle [7]. The dataset contains resumes sorted into 25 different "categories". We subset the resumes to what we judged to be the 12 most technical categories for two reasons; firstly, the aim of this project is to assign candidates into technical roles, so it makes sense to focus specifically on these types of jobs. Secondly, the roles we are matching over should be somewhat interchangeable in nature. If the roles we are matching over are very disparate, a candidate would have little use for our tool, as they themselves would be able to distinguish which job category they are most suited to (for example a candidate is likely to know if they are more suited to work as an Economist compared with Human Resources, but the decision may be less clear if they are choosing between a Data Scientist and a Python Developer). These categories, which from now on we will refer to as "job titles", will act as ground truths for the most appropriate job for this candidate when we test the effectiveness of the CV matching tool. The 12 job titles we will focus on are[1]:

- Data Science
- Mechanical Engineer
- Sales
- Java Developer
- Business Analyst
- Operations Manager

[1] In the data Blockchain Engineer was initially "Blockchain", and Data Engineer was initially "Database". We changed these categories so they better matched realistic job titles.

- Python Developer
- DevOps Engineer
- Network Security Engineer
- Data Engineer
- Blockchain Developer
- ETL Developer

Job descriptions

To train our matching tool, we required data on job descriptions for the 12 roles outlined above. To retrieve this, we used Selenium to scrape job descriptions based in the US from Indeed.com, a popular job market website. As Indeed.com has anti-bot defences, you are only able to scrape the job descriptions for the first page of results. For this reason, we identify 10 cities in the US, and for every city we scrape the first page of results for each job title. For each job posting we scraped the following information:

- Job title
- Company name
- Description
- Location

In total, we scraped 2,160 job descriptions across the 12 job titles. For the ground truth labels of the job title for each job description, we use the job title searched for (one of the 12 from the resume data), rather than the actual job title on the job posting. We do this for the following reasons:

- The specific job titles often vary even for very similar roles (for example, Digital Data Analyst, Business Performance Data Analyst, etc). If using the unique job titles scraped from Indeed.com we would have 1,004 different job titles in total. By using the job title searched for based on the resume data we achieve 12 consistent roles to match the resumes against.
- Using job titles that match the resume data allows us to easily test the performance of our matching tool. The job titles in the resume data can essentially be considered as the ground truth job title, so then when matching based on the content of the resume, we are easily able to produce evaluation metrics as you would in a typical classification task. This is one advantage this approach has over that used in the literature, as we have retrieved labelled data without the need of human annotators.

One limitation of using this approach is the possibility that Indeed returns jobs which are not directly related to the job title which was searched. This mainly occurs when the job search on Indeed.com is in a location with less jobs matching the search, so Indeed returns related jobs which may not directly pertain to the search. However, since we focus on 10 large cities in the US, this is less likely to impact how we have assigned the job titles.

When exploring the data, we noticed there were some job descriptions which were much longer than others. To prevent bias in the models and improve computational efficiency, we remove any job descriptions with a length of more than 1.5 standard deviations above the mean.

Exploratory Data Analysis

In Figure 1 we display the distribution of job titles in the scraped job description data. We have a good balance of the 12 job titles, with each job making up approximately 8% of the data. This means we do not have any concerns about class imbalance when it comes to running the classification models.

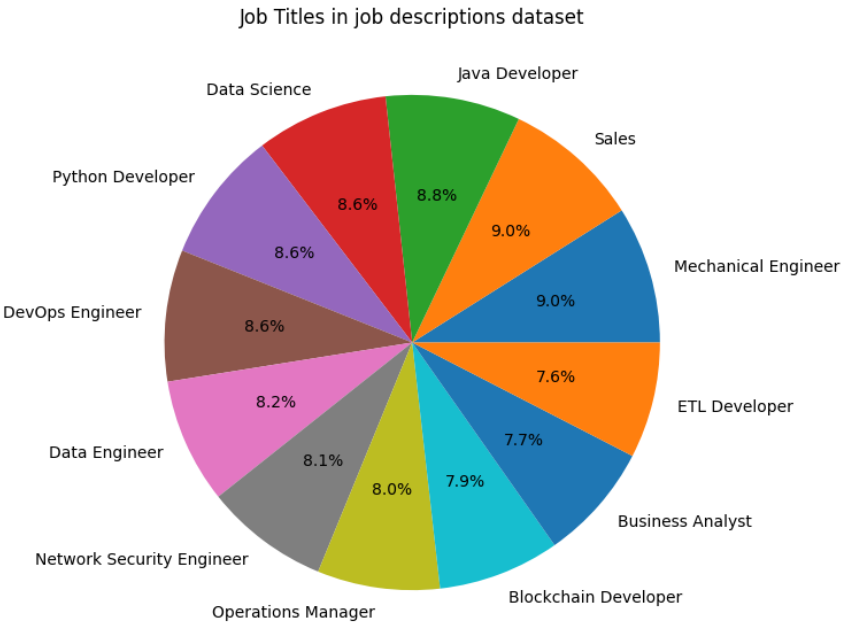


Figure 1: Distribution of job titles in the job descriptions

Figure 2 shows the distribution of job titles in the resume data. Here we have a slight class imbalance; Java Developer is the most common job title making up 16.4% of the data, while Business Analyst makes of the least at 5.5%. The class imbalance in the resume data poses no issues for our models as the resume data is only used for testing, therefore cannot introduce bias into the training of the models.

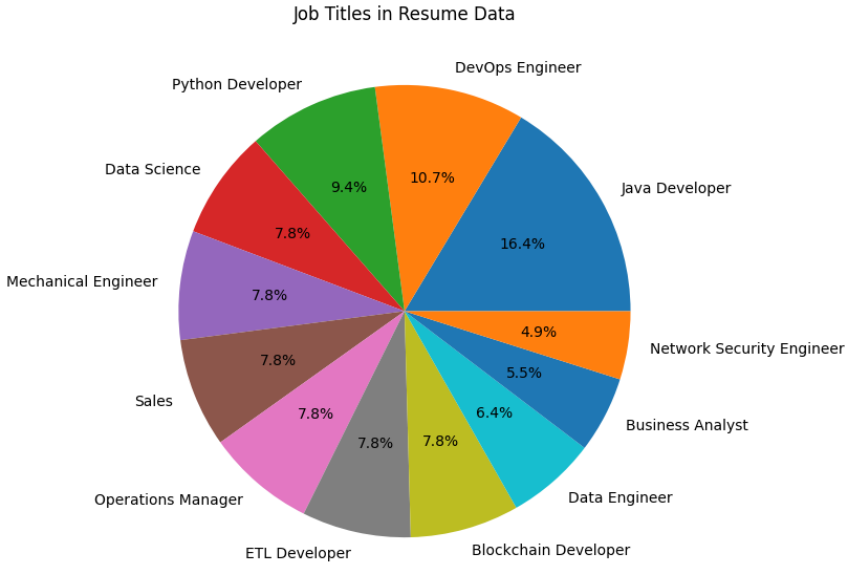


Figure 2: Distribution of job titles in the resumes

To understand the content of each dataset better, Figure 3 and 4 show the 15 most common words in the job descriptions and resumes respectively. As would be expected, the most common words are skills which an employer or prospective employee would need to display to be hired in a role. There is a lot of overlap between the common words of the two datasets, with, java, python, and sales all appearing in the top words for both. This indicates that we should be able to achieve successful matching between the resumes and the job descriptions, if a model can pick up on these common terms.

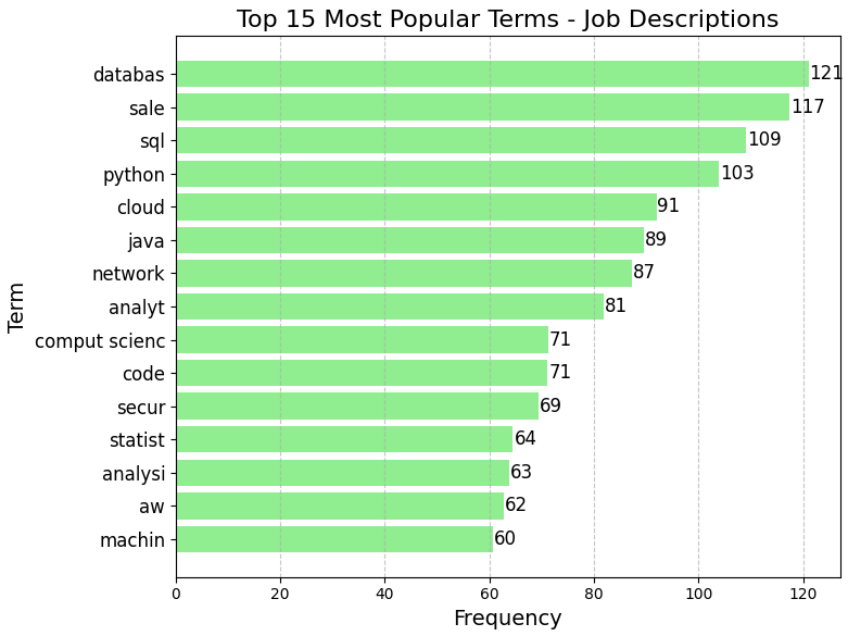


Figure 3: Most popular words in the job descriptions

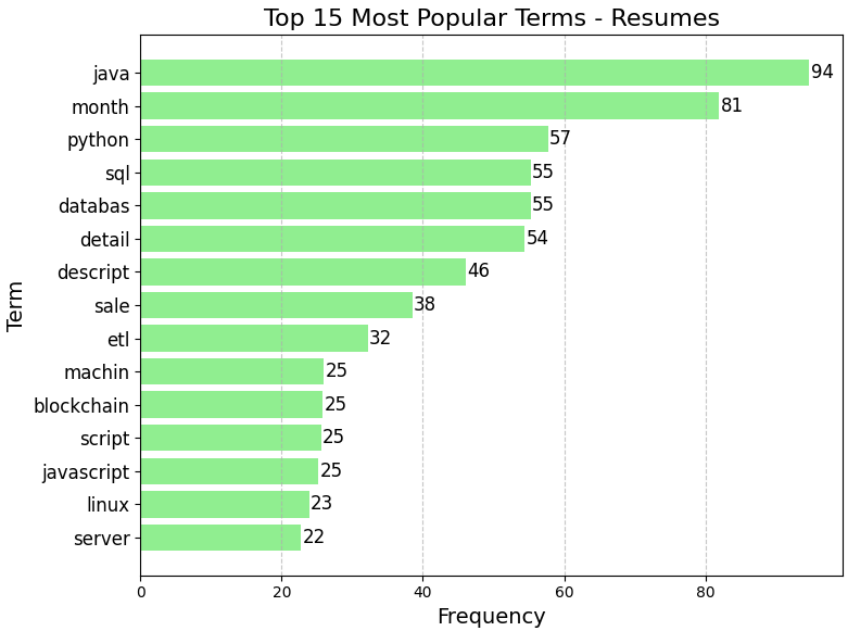


Figure 4: Most popular words in the resumes

Matching methodology

Our general approach follows that of previous literature; we apply standard pre-processing steps to our data before embedding the text into vectors so cosine similarity can be used to match resumes and job descriptions. We also add extra weight for hard and soft skills identified in the text using spaCy's NER model, as we believe skills are the most important aspect for matching a candidate to a professional role. We then test a variety of models beyond just cosine similarity to understand which machine learning approach is best suited to our data. Below we outline all of these steps in detail before reporting the results of our models.

Data preprocessing and skills labelling

Before comparing the resumes and job descriptions we conducted basic pre-processing on all our data, including lowercasing, stopword removal, special character removal, and stemming. These steps ensure we are distilling the text into its core information, aiding our models in focusing on the important information for the decision making process.

A key aspect of our model was identifying skills in both resumes and job descriptions. We believe that the skills in a resume or job description are one of the main components we should be focusing on when matching candidates to a prospective role, so we wanted to give identified skills a higher weighting in the vectors forming the Document Term Matrix (DTM). For this process we followed previous literature by employing spaCy's NER model,

which can identify and classify named entities within sentences. To train the NER model we needed resumes and job descriptions manually labelled with the location of hard and soft skills in the text. To assist us with this process we used an online NER text annotator [8], where we labelled 20 resumes and 75 job descriptions with their soft and hard skills. After training, the model is then able to identify most of the soft and hard skills in the resumes and job descriptions. To give extra weight to the skills in the decision making process, we extracted the skills from every resume and job description and concatenated them three times back into the main text before forming the DTM, essentially giving the skills three times the weight of all other aspects of the text.

To form the DTM (and thus, the vectors we will use for classification), we employed TF-IDF vectorization. TF-IDF accounts for both the frequency of words within a document, as well as how commonly a word appears in the entire corpus of documents. It gives relatively more weight to unique words which are likely to be specific to certain documents. Using TF-IDF, each resume and job description can be represented by a TF-IDF vector, which when all stored together in a matrix form the DTM. When creating the DTM we also set the minimum document frequency of 7.5% and maximum document of 40%. This means that any word that appears in either more than 40% of the douments, or less than 7.5% of the documents will be excluded from the vectors. This process helps us to filter out words that are either too common or too rare to provide useful information to the model.

Modelling

To match the resumes to the job descriptions we tested four different methodologies; two versions of cosine similarity matching, logistic regression, and random forest. Initially the models' performance will be tested running them on the job posting descriptions attempting to predict the job title. Subsequently they will perform the matching process between resumes and job descriptions following different methods. These methodologies are explained in more detail below.

Cosine Similarity Mode

Cosine similarity is a measure used to determine how similar two vectors are in a high-dimensional space. It measures the cosine of the angle between the two vectors, which ranges from -1 to 1. A value of 1 indicates that the vectors are identical, 0 indicates orthogonality (no similarity), and -1 indicates complete dissimilarity. To compute cosine similarity, you take the dot product of the two vectors and divide it by the product of their magnitudes (Euclidean norms). This is displayed in the following formula:

$$\text{Cosine Similarity}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$$

For every resume, we calculated the cosine similarity between the its vectorized text and all job description vectors in our data. We then take the top 10 most similar job descriptions to the resume, and assign the most common job title across these 10 job descriptions as the most suitable job title for the resume.

Cosine Similarity Average

Here, we utilise cosine similarity in a slightly different way. Similarly to above, we compute the cosine similarity between a resume and all job descriptions in our data. Then, for each of the 12 job titles, we compute the average cosine similarity distance across all job descriptions that have that particular job title. The matched job title is then the one with the highest average cosine similarity score across all job descriptions.

Logistic Regression

Logistic regression is a statistical method commonly used for binary classification tasks, where the goal is to predict the probability that an instance belongs to one of two classes. Here, we have 12 different classes, so we utilise the "One-vs-Rest" (OvR) classification strategy, where multiple binary logistic regression classifiers are trained, each one corresponding to a single class, and then the results are combined to make multi-label predictions. The input to the logisitc regression model are the TF-IDF vectors from the job description DTM, as these are the numerical representation of the text. Then the model is used to predict the job title for the resume vectors. When implementing the logisitc regression, we grid search over multiple parameters to optimise the model.

Random Forest

Random Forest consists of a collection of decision trees, where each tree is trained on a random subset of the training data and a random subset of features. For multi-label classification, each decision tree predicts a subset of the possible class labels for a given instance, and the final prediction is determined by aggregating the predictions from all trees determining the majority class. As with logisitc regression, the TF-IDF vectors from the job description DTM are used as the input to the model, then the model predicts the job title for the resume vectors. To optimise the model, we also gird search over a variety of parameters.

Skills matching

For the aforementioned BSE resume matcher tool we have designed, we implement a feature which informs candidates what skills they are missing for the role they have been matched to. For this, we need a set of the most common skills for each of the 12 job titles. We took the top 20 most frequently occurring skills identified by spaCy's NER model for each job title. We then also use the spaCy NER model to extract skills from the resume fed into the tool. With the list of skills from a resume and a list of the most common skills for each job title, we can then inform the candidate of the skills they are missing for their best matched role.

Results

To measure the performance of our models, we report accuracy, recall, precision and F1 scores. Since this is a multi-label classification problem each metric is macro-averaged, where metrics are calculated independently for each class and then averaged across all classes. As previously mentioned, to calculate these metrics the models are trained using the job description data, then prediction accuracy is measured against how well the models label the resume data.

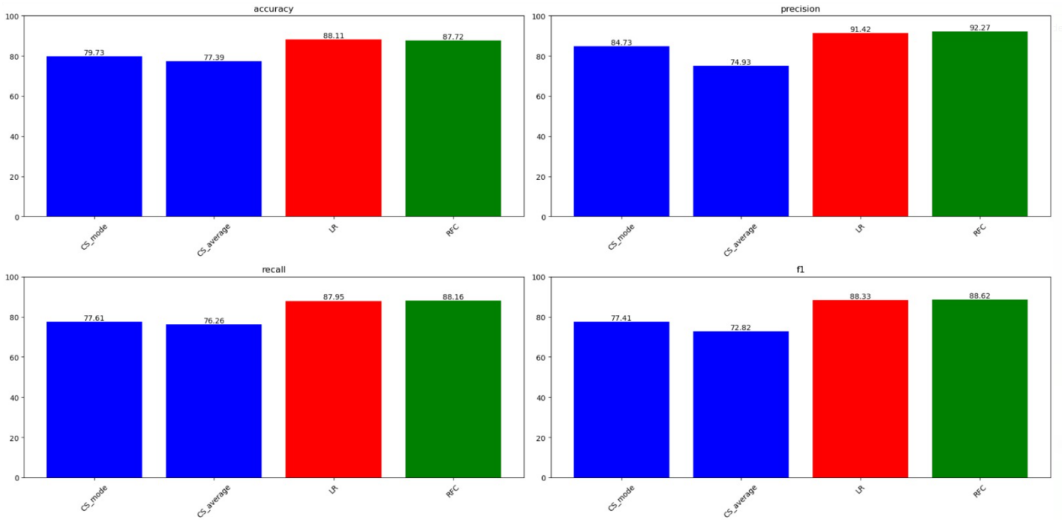


Figure 5: Resume matching accuracy metrics

Figure 5 reports the metrics for all four models. Focusing on the cosine similarity models, both models perform well with the "mode" model achieving an F1 of 77.41, while the "average" model gets a slightly lower 72.82. These results line up with previous literature which often achieved an F1 of around 0.8 when using cosine similarity matching. We find our more advanced machine learning models, logisitc regression and random forest, both improve upon the performance of the cosine similarity models, attaining very high F1 scores

of 88.33 and 88.62 respectively.

To understand how the classifiers perform at different classification thresholds, Figure 6 shows the Receiver Operating Characteristic - Area Under the Curve (ROC-AUC), which can give a more comprehensive understanding of the performance of the different classifiers. Here, the larger the area underneath the curve, the better the models ability to distinguish between positive and negative instances across all possible decision thresholds. We find similar trends as to the F1 metrics; logistic regression and random forest perform better than the more commonly used cosine similarity matching methods. These results show how leveraging more complex models can lead to improvements in matching resumes and job descriptions.

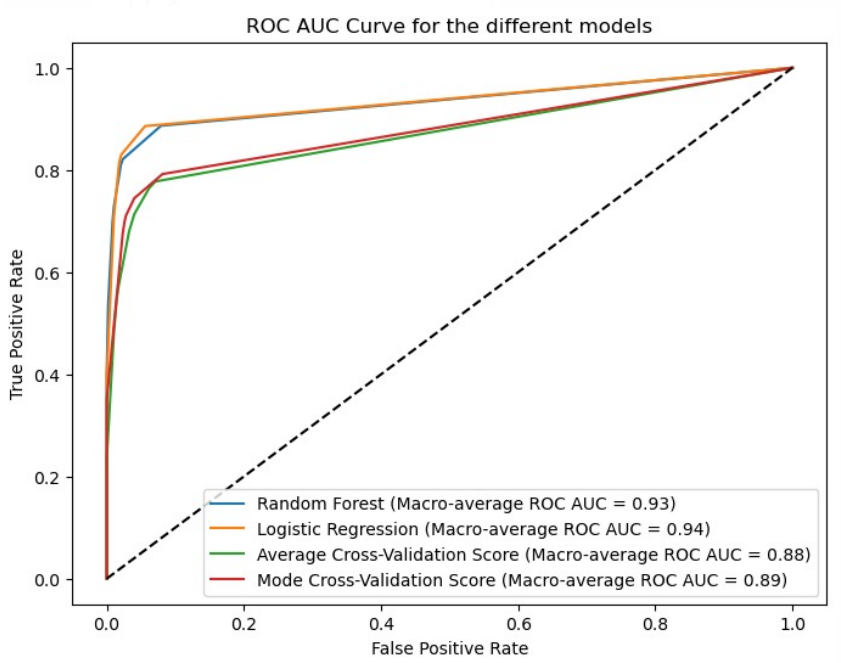


Figure 6: ROC-AUC curve for the different models

Conclusion

Overall, we are successfully able to leverage complex machine learning models to match resumes and job descriptions. We find that cosine similarity matching, which is commonly used in the literature, is an effective method at matching resumes to job descriptions. However, embedding the text into vectors and feeding these into more complex algorithms leads to better predictive performance with our data. We are then able to integrate these better performing models into our interactive tool for BSE students (see the appendix for a guide on the tool).

We should note that the classification task we are dealing with is simpler than that which

is often covered in the literature. We have the same 12 job titles in the resumes and job descriptions by design, which is a simplification of the real world task of matching candidates with prospective jobs. This may explain why we are able to achieve high performance metrics in this project. Nonetheless, we find our results promising, and using these types of methodologies for resume to job matching should be considered more in future research.

Appendix

BSE resume matching tool

The models developed were embedded in an interactive tool to allow job predictions for BSE students using the BSE resume template. The tool can be found in the code notebook submitted with this paper. The tool will parse the resume extracting the meaningful sections and output a prediction for the closest job title (available in the database) with an accessible user interface. In addition, the prediction will be followed by a section with the required skills for the predicted job, along with any skills the student is missing for their predicted job. Current limitations of the model are the limited amount of jobs in the dataset, especially for those graduates aiming to enter economics based roles. The tool offers the possibility to try the three different trained models by modifying the variable "model" with the models mentioned in the report. Below is a brief guide to use the tool:

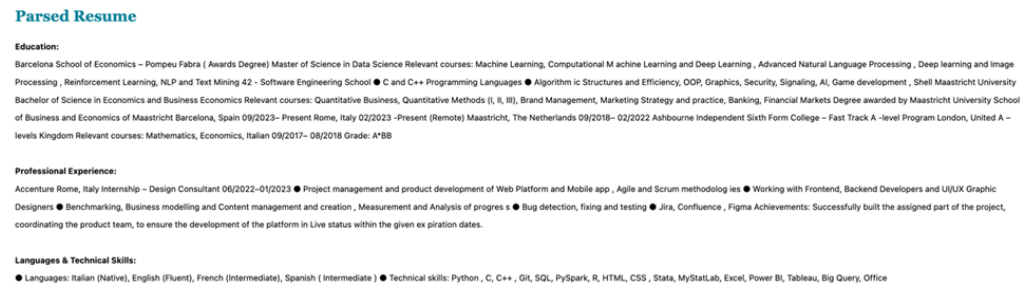
1. Upload the resume

   The resume should be uploaded using the BSE template for the CV book. The file should be a PDF file. To upload it to the tool, click the "Upload CV" button, browse local files and upload it. Just by uploading, the tool will automatically run the models and output predictions.

Figure 7: Example of BSE resume upload

2. Checking results

   In the first section "Parsed Resume", the parsed resume of the candidate will appear showing the single resume sections: "Education", "Professional Experience" and "Language and Technical skills" respectively.



Parsed Resume

Education:

Barcelona School of Economics – Pompeu Fabra (Awards Degree) Master of Science in Data Science Relevant courses: Machine Learning, Computational M achine Learning and Deep Learning , Advanced Natural Language Processing , Deep learning and Image Processing , Reinforcement Learning, NLP and Text Mining 42 - Software Engineering School ● C and C++ Programming Languages ● Algorithm ic Structures and Efficiency, OOP, Graphics, Security, Signaling, AI, Game development , Shell Maastricht University Bachelor of Science in Economics and Business Economics Relevant courses: Quantitative Business, Quantitative Methods (I, II, III), Brand Management, Marketing Strategy and practice, Banking, Financial Markets Degree awarded by Maastricht University School of Business and Economics of Maastricht Barcelona, Spain 09/2023– Present Rome, Italy 02/2023 -Present (Remote) Maastricht, The Netherlands 09/2018– 02/2022 Ashbourne Independent Sixth Form College – Fast Track A -level Program London, United A – levels Kingdom Relevant courses: Mathematics, Economics, Italian 09/2017– 08/2018 Grade: A*BB

Professional Experience:

Accenture Rome, Italy Internship – Design Consultant 06/2022–01/2023 ● Project management and product development of Web Platform and Mobile app , Agile and Scrum methodolog ies ● Working with Frontend, Backend Developers and UI/UX Graphic Designers ● Benchmarking, Business modelling and Content management and creation , Measurement and Analysis of progres s ● Bug detection, fixing and testing ● Jira, Confluence , Figma Achievements: Successfully built the assigned part of the project, coordinating the product team, to ensure the development of the platform in Live status within the given ex piration dates.

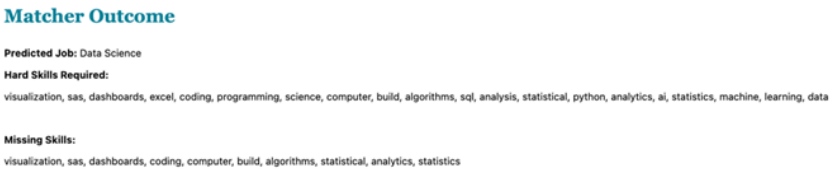
Languages & Technical Skills:

● Languages: Italian (Native), English (Fluent), French (Intermediate), Spanish (Intermediate) ● Technical skills: Python , C, C++ , Git, SQL, PySpark, R, HTML, CSS , Stata, MyStatLab, Excel, Power BI, Tableau, Big Query, Office

Figure 8: Example of Parsed Resume

3. Job title prediction

   In the second section "Matcher Outcome", the model will output the job title predicted for the particular candidate based on their resume information. In the same section there will be two small subsections "Hard skills required" and "Missing Skills". In the first one, the most common skills for the predicted job title will appear; in the following one "Missing skills", any skills not present on the resume which are required for the predicted role are listed.



Matcher Outcome

Predicted Job: Data Science

Hard Skills Required:

visualization, sas, dashboards, excel, coding, programming, science, computer, build, algorithms, sql, analysis, statistical, python, analytics, ai, statistics, machine, learning, data

Missing Skills:

visualization, sas, dashboards, coding, computer, build, algorithms, statistical, analytics, statistics

Figure 9: Example of tool outcome

This tool can be very useful for University graduates for several reasons. Firstly, students can use the tool to gain a clearer idea of the job market and how appealing their profile is for some specific job positions. Secondly, this model can help BSE students to understand what particular skills to focus on when preparing for the job search for the position they have in mind. Finally, the tool can be very interesting in the process of building a robust CV screening process that use similar methodologies to select candidates.

Further improvements can be made by increasing the amount of data retrieved to train the matching models. More specifically for BSE students, utilizing resumes of past students which are currently employed would help to train a model which will be more accurate for the typical jobs BSE studets enter after graduating. The model is currently showing some bias towards predicting some positions more often than others, this can be attenuated with larger amounts of accurately labelled data points. Other improvements can be done on the preprocessing of both resumes and job postings. Utilizing Named Entity Recognition models it would be possible to extract the level of experience of a candidate, which can be important for the level of the role which a candidate is applying for. This can be applied also on job

postings to extract more structured data, aligning with the resume format, and helping the matching process by making job postings and resumes more comparable. Furthermore, subsequent matching patterns can be applied to parse the resumes and the job postings to retrieve solely the relevant information in the text.

References

[1] Duygu Çelik, Askỳn Karakas, Gülsen Bal, Cem Gültunca, Atilla Elçi, Basak Buluz, and Murat Can Alevli. Towards an information extraction system based on ontology to match resumes and jobs. In *2013 IEEE 37th annual computer software and applications conference workshops*, pages 333–338. IEEE, 2013.

[2] Andre Lustosa Motta. Optimizing text encoding model to improve matching between resumes and job postings. Technical report, Working Paper, North Carolina State University.

[3] Abdou Karim Kandji and Samba Ndiaye. Design and realization of an nlp application for the massive processing of large volumes of resumes. In *2022 IEEE Multi-conference on Natural and Engineering Sciences for Sahel's Sustainable Development (MNE3SD)*, pages 1–5. IEEE, 2022.

[4] Suleiman Ali Alsaif, Minyar Sassi Hidri, Imen Ferjani, Hassan Ahmed Eleraky, and Adel Hidri. Nlp-based bi-directional recommendation system: Towards recommending jobs to job seekers and resumes to recruiters. *Big Data and Cognitive Computing*, 6(4):147, 2022.

[5] Saket Maheshwary and Hemant Misra. Matching resumes to jobs via deep siamese network. In *Companion Proceedings of the The Web Conference 2018*, pages 87–88, 2018.

[6] Shiqiang Guo, Folami Alamudun, and Tracy Hammond. Résumatcher: A personalized résumé-job matching system. *Expert Systems with Applications*, 60:169–182, 2016.

[7] Gaurav Dutta. Kaggle - resume dataset. Available online at https://www.kaggle.com/datasets/gauravduttakiit/resume-dataset, 2021.

[8] Ner text annotator. Available online at https://tecoholic.github.io/ner-annotator/, 2024.