TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Minor Project Report

On

ResuMate: An Automated Resume Screening System

Submitted By:

Kristina Bhandari (THA077BCT022)

Kristina Ghimire (THA077BCT023)

Pradeepti Dongol (THA077BCT033)

Namita Bhatta (THA077BCT048)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

March, 2024

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Minor Project Report

On

ResuMate: An Automated Resume Screening System

Submitted By:

Kristina Bhandari (THA077BCT022)
Kristina Ghimire (THA077BCT023)
Pradeepti Dongol (THA077BCT033)
Namita Bhatta (THA077BCT048)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in Computer Engineering

Under the Supervision of

Er. Kshetraphal Bohara

March, 2024

DECLARATION

We hereby declare that the report of the project entitled **“ResuMate: An Automated Resume Screening System”** which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Computer Engineering**, is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than those listed here have been used in this work.

Kristina Bhandari (077/BCT/022) __________________________

Kristina Ghimire (077/BCT/023) __________________________

Pradeepti Dongol (077/BCT/033) __________________________

Namita Bhatta (077/BCT/048) __________________________

Date: March, 2024

CERTIFICATION OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a minor project work entitled **“ResuMate: An Automated Resume Screening System”** submitted by **Kristina Bhandari, Kristina Ghimire, Pradeepti Dongol and Namita Bhatta** in partial fulfillment for the award of Bachelor’s Degree in Computer Engineering. The project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor’s degree of Computer Engineering.

____

Project Supervisor

Er. Kshetraphal Bohara

Department of Electronics and Computer Engineering, Thapathali Campus

____

External Examiner

____

Project Coordinator

Er. Umesh Kanta Ghimire

Department of Electronics and Computer Engineering, Thapathali Campus

____

Head of Department

Er. Kiran Chandra Dahal

Department of Electronics and Computer Engineering, Thapathali Campus

March, 2024

COPYRIGHT

ACKNOWLEDGEMENT

We extend our heartfelt gratitude to the Institute of Engineering, Tribhuvan University, for including our minor project in the Bachelor's in Computer Engineering curriculum. This opportunity has significantly enriched our knowledge and skills.

We would sincerely like to thank our supervisor **Er. Kshetraphal Bohara** for his invaluable guidance and feedback on this project.

We express our sincere appreciation to the Department of Electronics and Computer Engineering, Thapathali Campus, for their unwavering support and encouragement. Their commitment to providing a quality learning experience throughout our journey has been invaluable.

Furthermore, we would like to thank the research article authors, classmates, teachers, and everyone else who helped us develop our project, directly or indirectly.

Kristina Bhandari	(THA077BCT022)
Kristina Ghimire	(THA077BCT023)
Pradeepti Dongol	(THA077BCT033)
Namita Bhatta	(THA077BCT048)

ABSTRACT

The process of manual resume screening can prove to be very tedious, inefficient, time-consuming, and biased. This project outlines the development of an Automated Resume Screening System that ranks the applicants based on the similarity between different features like skills, education, experience, and language in their resume and the job description. The features from each resume and job description are extracted by training a NER model using a RoBERTa model on manually annotated data. Recruiters can assign different weights to various features according to the demands of a particular job. The system employs a pre-trained Word Embedding model using CBOW algorithm for effective vectorization of textual data. Additionally, it utilizes cosine similarity for systematic comparison and ranking of resumes. The NER model for resume and job description has performed with an F1 score of 0.96 and 0.68 respectively. The use of machine learning algorithms ensures fairness in the screening process, inclusivity reduction of manual efforts along with increase in efficiency and accuracy.

Keywords: CBOW, Cosine similarity, F1 score, NER, Resume Screening, RoBERTa, Word Embedding

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Adagrad	Adaptive Gradient Optimizer
ADAM	Adaptive Moment Estimation
API	Application Programming InterfaceBiL
AWS	Amazon Web Services
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long Short-Term Memory Network
BioBERT	Bidirectional Encoder Representations from Transformers for Biomedical Text Mining
BoW	Bag of Words
CBOW	Continuous Bag of Words
CLI	Command Line Interface
CNN	Convolutional Neural Network
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
CPU	Central Processing Unit
DFD	Data Flow Diagram
et al.	And Others
FAIR	Facebook's AI Research
GB	Giga Byte
GHz	Gigahertz
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HR	Human Resource
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IT	Information Technology
JPG	Joint Photographic Expert Group
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbors
LSTM	Long Short-term Memory
MAE	Mean Absolute Error
MCQ	Multiple Choice Question

MIME	Multipurpose Internet Mail Extensions
ML	Machine Learning
MLM	Masked Language Model
NER	Named Entity Recognition
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
OCR	Optical Character Recognition
ORM	Object-Relational Mapping
OS	Operating System
PDF	Portable Document Format
POS	Part-of-Speech
RAM	Random Access Memory
RMS	Root Mean Square
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RoBERTa	Robustly Optimized BERT Pretraining Approach
RSME	Root Mean Square Error
SQL	Structured Query Language
TF IDF	Term Frequency – Inverse Document Frequency
WE	Word Embedding
XML	Extensible Markup Language
YAGO	Yet Another Great Ontology

1. INTRODUCTION

The organization receives several resumes during the job hiring process, and the Human Resource Officer (HR) scans all the resumes manually and examines whether the candidate is suitable for the job or not. This manual examination of resumes is a time-consuming process and often most of the resumes remain unchecked. To overcome this problem, a resume screening system can be used. An automated resume screening compares applicants and ranks and sorts them based on constraints provided by HR. This aids in the screening process by increasing accuracy, decreasing the time required, and providing fair judgment.

1.1 Background

From the invention of the resume by Leonardo Da Vinci in 1482 to the resume being a part of the formal job application process in the 1950s to having online resume profiles on LinkedIn in the 2000s, the process of resume screening has been a very labor-intensive manual process prone to biases. With the volume of applicants increasing, where 250 applicants apply for each corporate job offer on average and companies like Microsoft and Meta receiving a large number of applicants per job per day, the process of manual resume screening becomes time-consuming and leads to fatigue. During the same period from 1482 to the 2000s, the world has witnessed a huge shift in the technological field. However, not much has changed in the resume screening process.

1.2 Motivation

Recruiters face a great deal of difficulty when it comes to assessing resumes because of the overwhelming volume of applications received for available positions. The sheer volume of resumes frequently results in time constraints, making it difficult to manually analyze each application in its entirety. For a single hire, resumes are projected to take up to 23 hours. When a job posting generates 250 resumes on average, and 75 percent to 88 percent of them are unqualified, it's no surprise that filtering the right people from such a large applicant pool is the most difficult element of the job. The arbitrary nature of candidate credentials, disparities in information-presenting techniques, and formatting inconsistencies all serve to worsen this issue. With these intricacies, it becomes quite difficult to find the best candidates, and it is frequently possible to miss some qualified applicants. Furthermore, biases (conscious or unconscious) may alter the screening procedure, making it more difficult to evaluate candidates fairly.

Thus, to enhance efficiency in the resume screening and selection process, saving time and enabling recruiters to focus on engaging with the most promising candidates, the Resume Tracking System is imperative.

1.3 Problem Definition

The traditional manual hiring process has long been criticized for its complexity and time-consuming nature. This approach not only necessitates a significant time investment and cost, but it is also prone to human error, potentially leaving out valuable skills and qualifications. Hence, the challenging task of individually analyzing each resume is alleviated by automating the system. The automated process can screen resumes, evaluating candidates based on their skills and qualifications by mapping them with the job description and ranking them based on their abilities to qualify for the job. Resume Scanning System based on TF-IDF, Bag of Words, and KNN methods were found to have encountered several challenges that hinder accuracy and efficiency in matching resumes with the required job descriptions. TF-IDF used as a text mining tool in the past, fails to capture semantic relationships between words. Similarly, Bag-of-words (BoW) ignores the order of words in a document and treats each entity independently, which leads to the loss of sequential information and the loss of context in analysis. Also, BoW matrices can exhibit sparsity, especially with vocabularies which results in demanding substantial memory and computational power. Likewise, in KNN, the computational cost is high as the distance between all the training points is calculated. To address this limitation, Word Embedding can be used by the FastText technique. This approach improves semantic comprehension, leading to a more precise evaluation of candidates.

1.4 Objectives

- To help recruiters by creating an automated resume screening system that matches resumes to job descriptions and ranks top candidates.
- To use word embedding to achieve better semantic matching and increase accuracy.

1.5 Scope and Applications

The resume screening system extracts text from the resume and compares the applicant's resume with the provided job description using NLP to automate the repetitive task of manual resume parsing. Since this process uses semantic search, the candidate filtering is based on the meaning and context of words rather than just keywords which improves the accuracy, efficiency, and relevance of results. The system will adapt according to the changing job market dynamics ensuring adaptability and scalability hence making it applicable for HR departments of companies of various fields.

1.5.1 Scope

Automation of Routine Tasks

The major task of HR is to manually scan every resume collected to select the best candidates. Sometimes, for one position the company receives many resumes for which the selection process becomes time-consuming and expensive. To overcome this manual labor of HR, an automated resume screening system is beneficial. Not only does it reduce time and energy, also every candidate is evaluated in a short period which helps select the best candidate.

Time and Resource Savings

This system saves time and resources by automating resume analysis.

Efficient Candidate Filtering

Since every resume is scanned and filtered using the Natural Language Processing (NLP) method, the process gives more accurate results without any bias. Filtering focuses on important sectors within the resume like experience, education, and skills by cleaning the unwanted information. The use of a ranking algorithm increases the accuracy of the ranking process.

Skill Matching

The system compares the necessary skills specified by the company with each resume. Based on that comparison, ranking of each candidate is done which lists out the best candidates for the particular role. For example, for a frontend developer job, the system ranks the candidates based on their knowledge of the skills needed like HTML, CSS, JavaScript, React, etc.

Customized Ranking Criteria

Recruiters will have the option to alter ranking criteria following the particular specifications of each job posting by assigning weights to various entities, guaranteeing a customized and situation-specific assessment.

However, the major limitation and challenge for the Resume Screening System is to find appropriate and standard dataset. Collecting resumes is difficult as they are private documents. That is why we had to resort to using LinkedIn resumes, as they were the only ones readily available to us. Resumes, in general, do not have a standard format, and there is no industry standard. Unfortunately, we were unable to obtain various resume formats other than those available on LinkedIn. This presents a limitation because our model may not perform as well on different resume styles and structures.

1.5.2 Applications

The major application of our project is in job placement agencies and human resource departments for skill-based hiring.

Corporate recruitment

Make it easier for corporate recruiters to evaluate a large number of resumes by automating and streamlining the initial screening process.

Job Placement Firms

Assist employment agencies in connecting applicants with appropriate positions based on their backgrounds and skill sets.

Educational Institutions

Help academic institutions rank and analyze student resumes for co-ops, internships, and post-graduation jobs.

1.6 Report Organization

The content presented at this project report is organized into eight chapters.

Chapter 1: outlines the background of the project including motivation to choose the project, problem definition, objective, scope and application of the project

Chapter 2: presents short overviews of all the prior work carried out in the field relevant to the topic

Chapter 3: outlines the necessary hardware and software requirements for the project along with its implementation

Chapter 4: explains the detailed steps carried using visual diagrams like Data flow diagram, Class diagram, Sequential diagram, ER diagram and architecture diagrams that illustrates the explanation of how the hardware and software are used to achieve the goal of our project

Chapter 5: describes how the system methodology is implemented by including details of implementation tools.

Chapter 6: showcases the outcomes and the challenges encountered throughout the project, using graphs and pictures of the user interface.

Chapter 7: highlights the action that can be taken in the future to enhance the project

Chapter 8: sums up every step taken for the project

Chapter 9: contains the additional details of the project like budget, project schedule and code snippets.

2. LITERATURE REVIEW

Kamil et al. [1] used NLP for preprocessing, multiple semantic resources, and statistical concept relatedness measures extracted from the Hiring Solved dataset. 500 datasets of resumes and 10 datasets of job descriptions in PDF and docs format were used to train and test the model in a Windows 7 PC with a 2.1 GHz CPU and 4GB RAM. The Natural Language Processing (NLP) model was used for document segmentation, tokenization into unigram, bigram, and trigram, stop words removal, POS tagging, and NER. The Stanford Core NLP package was used for this process. The model made use of Semantic Networks like WordNet ontology and YAGO2 ontology. A missing background knowledge handler was used to bridge the gap between extracted information and semantic networks. The resume and job description were then compared using the Jaro-Winkler Edit Distance Function. The list of candidate concepts was refined with the help of statistical-based concept-relatedness measures.

Tiwari et al. [2] perform text extraction using Optical Character Recognition (OCR) for resumes that convert PDF, JPG, and other digital formats to a text format which is then fed to a Natural Language Processing (NLP) module. The NLP module is the process of analyzing natural languages to make them understandable by computers. It assists in data training as well as text data extraction. NLP performs lexical analysis, syntactic analysis, semantic analysis, and Named Entity Recognition (NER). Lexical analysis generates tokens from text input while syntactic analysis is used to analyze grammar and word arrangement. Semantic analysis checks the correctness of the meaning of the text whereas the NER helps to understand the contextual meaning of words. The Human Resource (HR), through a web portal, defines the constraints and skills required. Based on these constraints, a dashboard containing graphs and pie charts of data fed by LogStash is prepared using Elasticsearch. The resumes are scored and sorted using inbuilt queries of ElasticSearch.

Bhatia et al. [3] perform resume parsing and find candidates using two steps: they first extracted information from resumes and then used BERT sentence pair classification to rank the candidate based on the job description. They converted the resume pdf document into text using two tools: pdftohtml, and Apache Tika. In this method, two types of resume format are considered: LinkedIn format and non-LinkedIn format. The

LinkedIn format is taken as a standard format for a resume so all the non-LinkedIn format resume is first converted to the LinkedIn format. BERT is used to convert non-LinkedIn formatting to LinkedIn format. They used pdftohtml to extract the text and metadata information and displayed output in a JSON or CSV file. Then, Bidirectional Encoder Representations from Transformers (BERT) is used to pair the sentences into different classifications for the ranking process. Using BERT leads to only one output layer to get the required results. The final output is displayed in a web application. They collected 715 LinkedIn resumes and 1000 non-LinkedIn resumes in PDF format. Their system was able to differentiate between LinkedIn and non-LinkedIn format resumes with an accuracy of 100%. Then, they structured extracted text from LinkedIn resumes into different headings and displayed a JSON file with an accuracy of 100%. They also converted a non-LinkedIn to LinkedIn format using BERT with an accuracy of 97%. Sentence pair classification using BERT shows 72.77% accuracy.

Jivtode et al. [4] employ Natural Language Processing (NLP) techniques to parse and rank resumes according to job profiles. The system architecture comprises four modules: Resume Parser, Data Extraction, Resume Ranking Algorithm, and Candidate Suggestions/Feedback. Resume Parsing utilizes NLP, including Morphological, Syntactic, and Semantic Analysis, to structure unstructured resume data for easier analysis. Data Extraction focuses on extracting relevant information (skills, experiences) from resumes using NLP tools like NLTK, with techniques such as Tokenization, Part-of-Speech Tagging, Named Entity Recognition, Semantic Analysis, and Sentiment Analysis. Resume Ranking Algorithm employs machine learning models like Support Vector Machines or K-Nearest Neighbors to rank resumes based on relevance to job profiles, using historical data for training. Candidate Suggestions and Feedback After data extraction, the system provides candidate suggestions matching skills, qualifications, and experience with job requirements. Gathers candidate feedback through surveys, interviews, or online platforms to enhance the recruitment process, ensuring fairness and inclusivity. The system has an average parsing accuracy of 95% and an ML model accuracy of 85%.

Nasr et al. [5] combines techniques of the modified Boyer-Moore Method with verifying string similarity based on Dice metrics. So, instead of strictly organized resume representation, this method provides flexibility in resume writing and does not use keywords, which can increase the relevance of the answers to queries prepared in advance. Firstly, resumes are converted to plain text format using tools like Tika to remove unnecessary elements and retain raw text. Then the raw text is divided into semantic blocks, allowing for a more flexible and error-tolerant representation of information. After identifying keywords in the text and categorizing them based on similarities using the Dice metrics and grouping related ideas, a query answering system is developed to answer queries by comparing detected keywords with predefined criteria, ensuring relevance. After its implementation with the modified Boyer Moore algorithm, the RELIEF method is utilized to estimate and weight specific features where the features are based on different criteria, facilitating the comparison of different resumes based on diverse criteria.

Daryani et al. [6] match each resume to the job description using cosine similarity, vectorization models, and natural language processing to extract information from unstructured resumes. Firstly, tokenization is accomplished by dividing large volumes of unprocessed data into smaller, more relevant pieces known as tokens. Additionally, whitespaces, punctuation marks, empty lines, and stopwords are eliminated. Secondly, using WordNet Lemmatizer, given by the NLTK Python package, words are analyzed morphologically and linguistically to add a base word to its stem or root word. Then, by grouping short phrases, chunking is utilized to add more structure to sentences, and Part of Speech tagging is used to categorize terms such as nouns, verbs, adjectives, adverbs, etc. The outcome is fed into the SpaCy module, which has a pre-trained module to identify entities from the resumes, such as name, phone number, email, college, organization, etc. JSON format is used to extract the result. Following that, vectorization is carried out for feature extraction utilizing TF-IDF (Term Frequency-Inverse Document Frequency). It is a text-mining tool that lets robots decipher sentences and interpret words quantitatively, enabling them to overcome the limits imposed by Bag of Words. Term frequency allows fair comparisons over a range of document lengths by measuring the frequency of words in a document. On the other hand, Inverse Document Frequency highlights uncommon terms and lessens the impact of frequent ones by analyzing a word's relevance across texts. Cosine similarity, which measures how similar two objects are to one another, is used for matching. Based on the similarity score, each resume is ranked. The major limitation of this system is that it does not include semantic information.

Abdul et al. [7] compared the vector space model and the word embedding approach. The resumes were collected and they were preprocessed and cleaned along with the job description using the NLP techniques. The processed text was then fed into the TF IDF mechanism and Word Embedding. For the TF IDF mechanism, a vectorized TF IDF model was used to transform the pre-processed documents into TF IDF vectors. Then their cosine similarity was calculated and hence the ranked resumes were sorted and filtered for a certain threshold. In the second system, already trained Google Word Embeddings were utilized to rank the resumes that used Word2Vec. If no embedding for a given word existed, a random number was generated for that word which was trained later. Then the cosine similarity of the resulting vector was calculated after which the rank was generated. Then according to this rank, the job description was matched with the resume. For the comparison of results, precision and recall curves were plotted which showed that the Word embedding curve covered more area than the Vector Space Model. The experiment was carried out on 169 resumes against 25 to 30 job categories. Hence, this paper concluded that the word embedding approach is more effective than the document vector-based approach.

Guillarme et. al. [8] dealt with the recognition of taxonomic entities using a pre-train-and-finetune approach applied to learn models from an ecological gold standard corpus. SpaCy library is used to create a taxonomic NER system. Two subnetworks, contextual and non-contextual embeddings, are used to describe SpaCy. Non-contextual embeddings use a lookup table learned from unlabeled text to represent a word with a single context-independent vector whereas contextual embeddings associate each input token sequentially in non-contextual representation first, then the encoder is subjected to the aggregation function overcoming polysemy, homonymy and context-dependent nature of non-contextual embedding. TaxoNERD offers two models here. The "en_ner_eco_md" model combines subword features with a CNN-based contextual encoder focusing on speed using spaCy's Tok2Vec layer for contextual embeddings. On the other hand, the "en_ner_eco_biobert" model substitutes BioBERT, a Transformer-based language representation model pretrained on biomedical corpora focused on accuracy. Deep neural models from TaxoNERD fared better than previous methods, especially on ecological corpora. The model based on Transformers regularly has the highest F-score. On corpora with a biomedical focus, dictionary-based approaches

(LINNAEUS and SPECIES) outperformed other approaches. Across all corpora, the baseline dictionary approach (MER) performed poorly.

Najjar et al. [9] work in three main building blocks. The first one is the Training Block where the domain Word Embeddings are trained from a set of resumes using the Skip Gram model. It is done in 4 levels. Firstly, the text is extracted and secondly, its unigram was generated by using NLP for tokenization and POS tagging. Thirdly, the bigram was generated using Natural Language ToolKit (NLTK). The bigrams were scored and the top-scored bigrams were cleaned by NER and POS tagging techniques. Finally, the WE models were trained from the bigrams and unigrams using the Word2Vec technique. The second block is the Matching Block where the text of the job description is matched with the text of each resume. Here the pre-trained Word Embeddings create vectors for all the resumes and job descriptions. The semantic similarity is then calculated using the Cosine Similarity and the top-ranked resumes are the ones with the highest similarity degrees. The top-ranked resumes are then processed further in the Extraction Block where the basic data such as name, phone number, and email are extracted using NLP Gazetteer and pattern-matching approaches. The libraries used in this System are Gensim, SpaCy, and NLTK. The dataset consisted of 101 unstructured resumes in pdf format. For verification, the top 3 candidates among 10 participants for 4 job positions were considered. The performance and accuracy were verified manually by IT specialists. The system was 100% accurate in candidate selection and 92% accurate in ranking the resumes. The limitation of this system is that it only has domain-trained Word Embeddings.

Naseer et al. [10] compares and analyses different tools and algorithms for named entity recognition (NER) models which are SpaCy, StanfordNLP, TensorFlow, and Apache OpenNLP. NER tools were evaluated by F1-score, training data loss, time accuracy, and Prediction probability. NER Spacy tool provides 100% accuracy during training. Stanford gives 99.5%, and Tensorflow and OpenNLP show approximately 99% accuracy in training. The NER TensorFlow method gives loss training 0.0229 value during training and Stanford and OpenNLP loss training 0.00000002137 and 0.00000142. Spacy tool's loss data only 0.00000001029 value which is very less. The Stanford performs 94% F1-Measure, Tensorflow, and OpenNLP show 97% and 96.5%. Spacy NER tool gives a 100% F1- Score. The prediction Probability of the Stanford tool is 90%. Tensorflow and OpenNLP give Prediction Probability accuracy of 96% and 98.3%. The NER Spacy tool performs Prediction Probability is 100%, so the spacy NER method provides the best and highest results in each experiment for NER classification.

Divya, et al. [11] uses pdf as input. Data extracted from the pdf file is stored in a CSV file, cleaned, and then used in the Long Short-Term Memory (LSTM). LSTM is a deep learning model that uses an artificial recurrent neural network (RNN) that can analyze and remember long information. Because of its better-memorizing capacity, it is used in handwriting and speech recognition. Web scraping is employed to find the information about the GitHub profile that influences the result. User data is stored in a database. Then the dataset is cleaned and vectorized for training. VS code and Django framework are used for system development, and the proposed system caters to two types of users.

Tejaswini et al. [12] use content-based filtering for recommendation purposes. It has a candidate screening system where the candidate login to the system and attends an MCQ test with the Face Detection System. It is only after they pass the test that they can submit their resume to the recruiter which is stored in a database in pdf format. Later, the texts from each resume are extracted and pre-processed. The Gensim module is used to summarize the collected text to scale the job description and the resume. Then the TF-IDF vectorization of the job description and the resumes is done. After that, the cosine distance is used to calculate the similarity between the job description and the resume. Finally, this system uses the KNN algorithm using cosine similarity value to identify resumes that closely match the job description and hence ranking is done. The dataset was collected from Kaggle which belonged to Java Developers and Project Manager Roles. It consisted of 50 resumes in doc and docx formats which were converted into pdf format. The average parsing accuracy of this system was found to be 85% and the average ranking accuracy was found to be 92%. The major limitation of this system is the loss of information while performing the summarization.

Racherla et. al. [13] proposed a hashing scheme-based word embedding and BiLSTM-based dynamic resume ranking system. The system consists of two phases: extraction and prediction. The dataset is converted into a hash vector using hashing after which the BiLSTM network is applied to train the model and provide probability prediction to rank the resumes. The system consists of three modules: the hashing module, the process module, and the ranking module. The hashing module converts extracted text to vector using word N-gram scaling while the processing module uses the Adaptive Moment Estimation (ADAM) algorithm over the Adaptive Gradient Descent (Adagrad) optimizer and Root Mean Square Propagation (RMS Prop) optimizer for better optimization to train BiLSTM for probability prediction. The ranking module then utilizes the prediction from the process module to rank the resume. The network performance evaluation was done using R squared (R2), R^2. Root Square Mean Error (RSME) and Mean Absolute Error (MAE) for 8000 instances and was observed that the data first perfectly and the model is well trained for correct prediction. Less than 7% R square error was observed in the system.

Berragan et al. [14] demonstrates a novel method for place name extraction from unstructured text by fine-tuning transformer-based language models (BERT, RoBERTa, DistilBERT) on 200 annotated UK 'place'-classified Wikipedia articles. The articles are initially cleaned using regular expressions and are annotated in CoNLL-03 NER format. Subsequently, the Spacy large pre-trained model was employed for further processing. For multiple tokens, the BIOUL tagging scheme was used using capitalization as an indication of multi-token noun phrases. 10% of labeled tokens were taken for validation and testing.Evaluation against pre-built NER models (SpaCy, Stanza) revealed BERT's superior performance. The prebuilt NER models, SpaCy (small) and SpaCy(large) reported NER F1 scores as 0.84 and 0.85 respectively whereas BERT, DistilBERT, and RoBERTa showed F1 scores of 0.939, 0.924, and 0.923 respectively. This paper demonstrated a best-case scenario where time-frames allow for manual annotation of task-specific data.

3. REQUREMENT ANALYSIS

Requirement analysis includes both hardware and software requirements.

3.1 Hardware Requirements

To develop a resume screening website, we need a computer or laptop with any OS Windows, Linux, or Mac.

3.2 Software Requirements

Software requirements includes different applications, libraries and languages used for the project.

3.2.1 VS Code and Google Colab

Our team trained the model on Google Colab and used Visual Studio Code as the main IDE for coding chores. Graphics Processing Units (GPUs), such as the NVIDIA Tesla K80 and T4 GPUs, are freely accessible through Google Collab for machine learning and deep learning applications. We made use of the T4 GPU, which has a faster clock speed of 1.59GHz and 16GB of GPU memory. Each of the 40 SMs on the Tesla T4 card has a common 6MB L2 cache.

3.2.2 HTML, CSS, and JavaScript

Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript collectively are used to create a simple, user-friendly interface for the user to interact. The primary structure of a web page is created using HTML, a standard markup language. The styling of HTML components is done with CSS. It describes how web pages should be laid together. A scripting language called JavaScript is used to create interactive web pages.

3.2.3 Django

For backend development, we us the Django-python web framework. Through an easy-to-use internet interface, multiple resumes and job descriptions can be uploaded in PDF format. The goal of integrating JavaScript, CSS, and HTML is to enhance the frontend user experience. With Django's built-in capabilities, important elements like file processing, user authentication, and data storage are simply integrated. The robust Object-Relational Mapping (ORM) technology included into Django is utilized by the system and is effortlessly connected with the MySQL database. With a focus on scalability, the backend is designed to manage big datasets and process resumes quickly and effectively.

3.2.4 SpaCy

Spacy is a productive Python open-source toolkit for tasks related to natural language processing. It allows training customized NER model with pretrained transformer models such as RoBERTa, BERT-Cased, BERT-Uncased, XLNet, etc. We used RoBERTa model for training NER.

3.2.5 Beautiful Soup

Beautiful Soup is a Python library that assists in web scraping from and manipulation of HTML and XML (Extensible Markup Language) documents. It works with parsers like lxml, html5lib and html.parser to retrieve information from websites. It allows us to navigate through the HTML and XML documents to locate required elements based on tags and attributes. We have used beautiful Soup to scrape job descriptions from three different websites (Mero Job, Kumari Job and Times Job).

3.2.6 Scikit-learn

Scikit-learn or Sklearn is an open-source Python package for data modelling and machine learning. We used it to it to efficiently partition the data into training and testing sets, using its train_test_split function.

3.2.7 FastText

Fasttext Word Embedding model, an extension of the word2vec model, is an open-source python library word embedding model created by Facebook AI Research (FAIR). It provides pre-trained word vectors in 157 different languages which produces word vectors with a dimension of 300. We used pre-trained word vectors obtained through training with the CBOW model.

3.3 Feasibility Analysis

Feasibility analysis for the project include technical feasibility, operational feasibility, economic feasibility and schedule feasibility.

3.3.1 Technical Feasibility

Our project does not require a high degree of computer specification. Python programming will be used to create the system. We will use the Django framework for the backend and HTML, CSS, and JavaScript, which are all open-source. The system is operational with the least number of technical requirements and hence, it is technically feasible.

3.3.2 Operational Feasibility

The Graphical User Interface (GUI) of the system is user-friendly and interactive, making it easier to upload resumes and job descriptions. It also ensures the security and confidentiality of candidate data.

3.3.3 Economic Feasibility

Since our project is purely software-based, it is extremely feasible in terms of economy. The project will be developed on a personal computer. Furthermore, there will be no requirement for paid resources.

3.3.4 Schedule Feasibility

The project can be completed within the given time frame with rapid progress due to effective team work and proper planning.

3.4 Dataset Analysis

To develop the resume screening system, we require resumes and job descriptions to train our NER system.

3.4.1 Resume Dataset

In addition to that, we have also obtained 1014 annotated resumes to train our NER model through Roman Shilpakar's blog. The data set contains resumes of candidates of the IT field like application development, networking, etc. Using this dataset, we trained and tested our model in 7 0:30, 80:20 and 90:10 ratios.

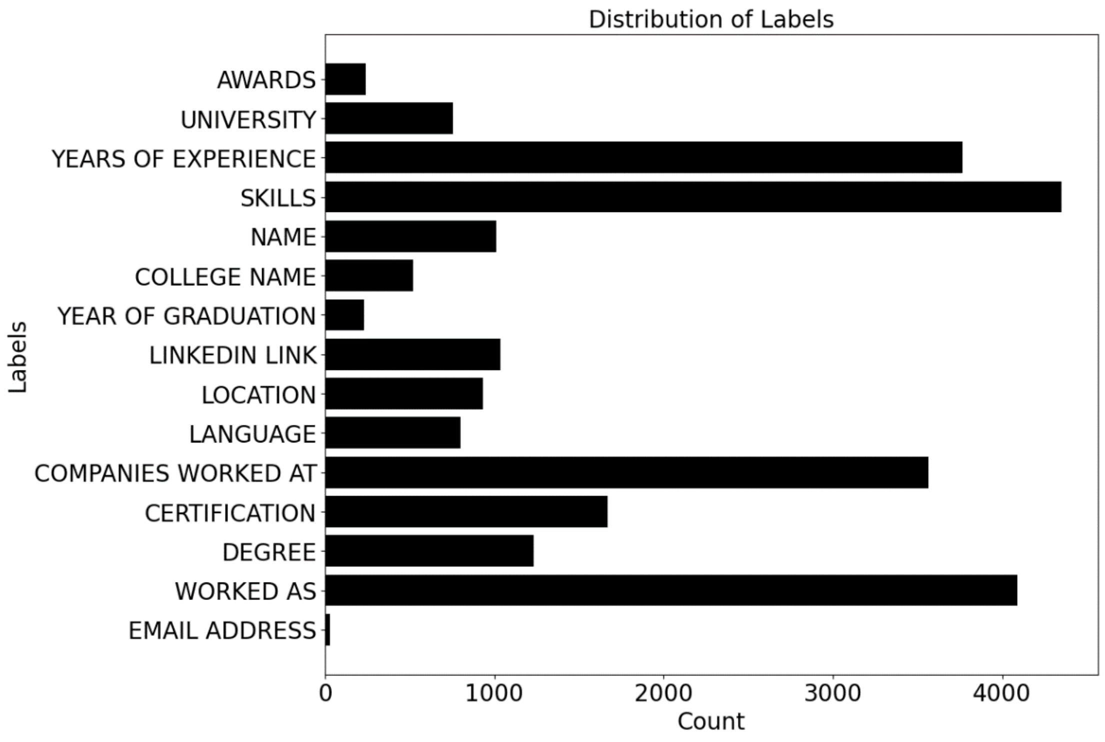


Figure 3-1: Distribution of Labels in Resume NER

3.4.2 Job Description

To obtain job description dataset, we used web scraping to scrape job descriptions from online job portals like Merojob, Kumarijobs, Timesjobs and Dice and compiled 1126 job descriptions. The job description dataset contains vacancies of various job titles of IT field like frontend developer, backend developer, full stack developer, java programmer, python programmer, software engineer, data engineer, etc. Out of those 1126 job descriptions, we annotated 335 job descriptions to train our NER on job descriptions in 70:30, 80:20 and 90:10 ratio.

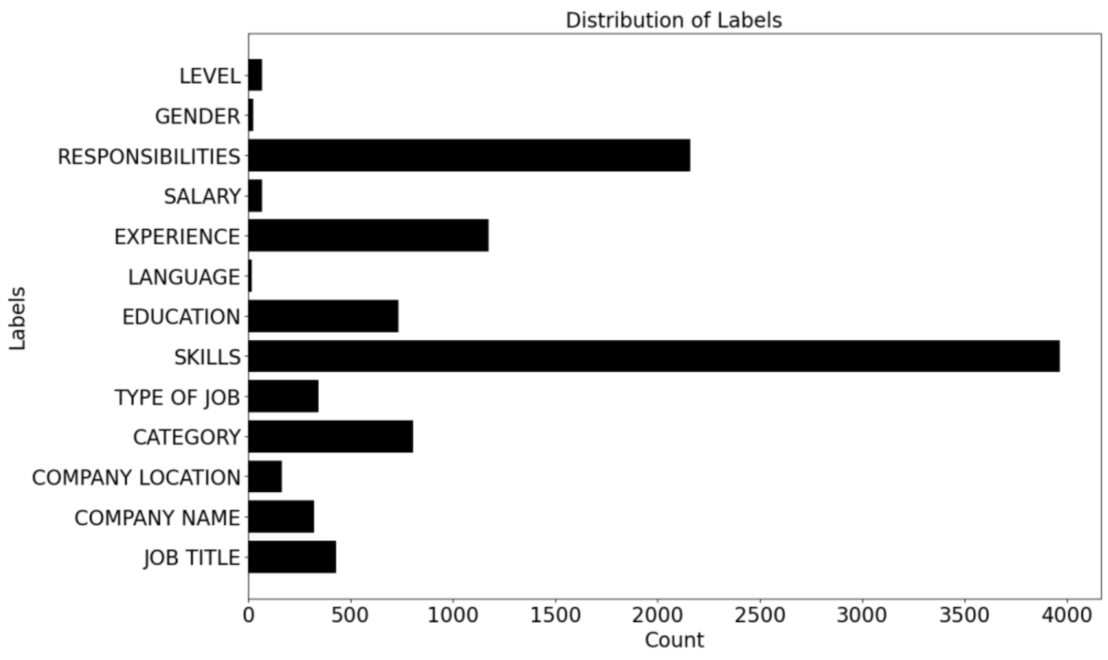


Figure 3-2: Distribution of Labels in Job Description NER

4. SYSTEM ARCHITECTURE AND METHODOLOGY

This section provides the detailed working of the system and different models used.

4.1 System Architecture

This diagram shows the overall working of the system.

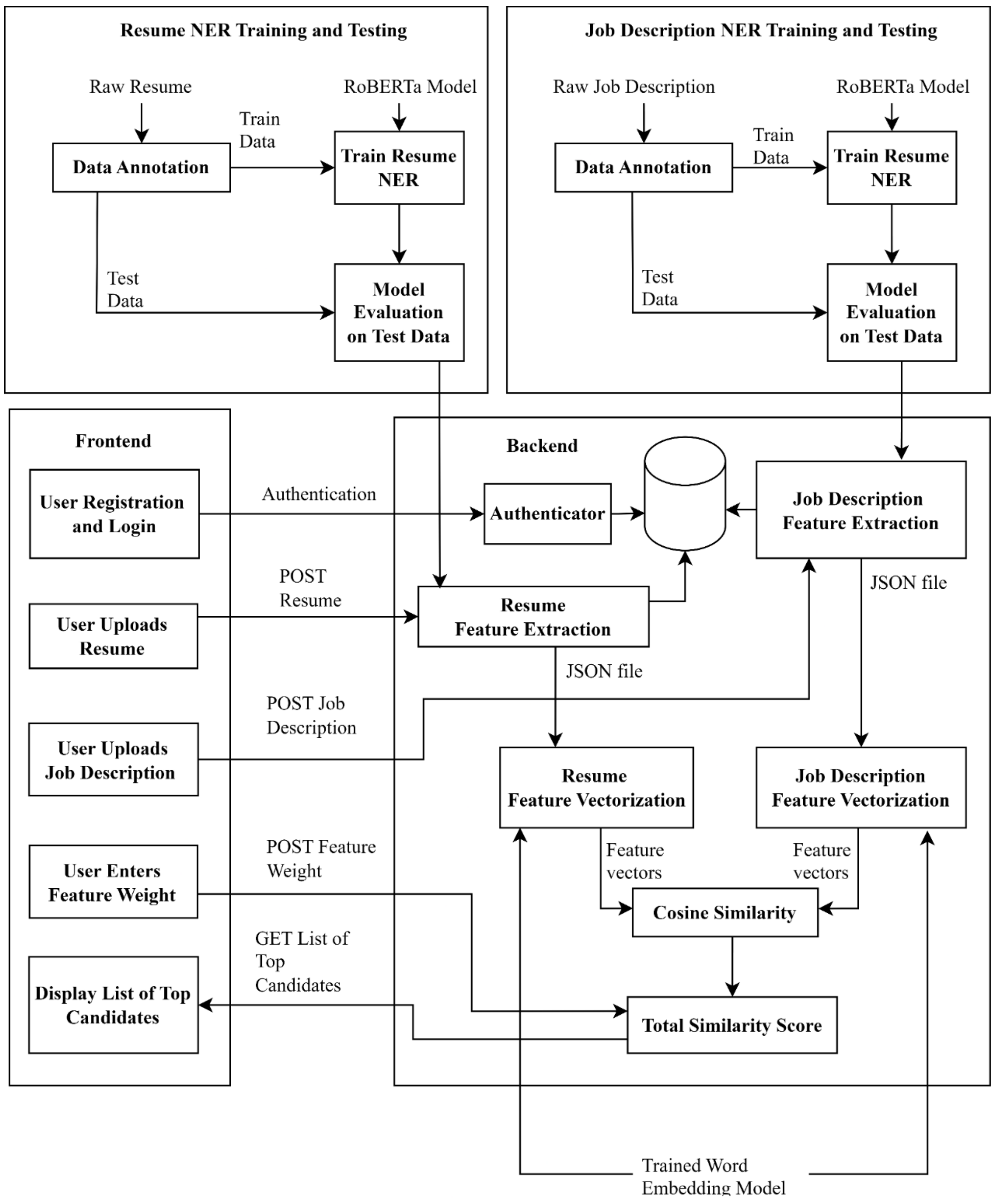

Resume NER Training and Testing
Raw Resume
RoBERTa Model
Data Annotation
Train Data
Train Resume NER
Test Data
Model Evaluation on Test Data
Job Description NER Training and Testing
Raw Job Description
RoBERTa Model
Data Annotation
Train Data
Train Resume NER
Test Data
Model Evaluation on Test Data
Frontend
User Registration and Login
User Uploads Resume
User Uploads Job Description
User Enters Feature Weight
Display List of Top Candidates
Authentication
POST Resume
POST Job Description
POST Feature Weight
GET List of Top Candidates
Backend
Authenticator
Job Description Feature Extraction
Resume Feature Extraction
JSON file
JSON file
Resume Feature Vectorization
Job Description Feature Vectorization
Feature vectors
Feature vectors
Cosine Similarity
Total Similarity Score
Trained Word Embedding Model

Figure 4-1: System Architecture

The company that requires a resume screening service uploads its applicants' resumes along with HR's job description. Text is extracted from the resume. It is then cleaned, encoded, and vectorized for the application of a comparison algorithm which generates output in terms of a score of resumes which is used to rank all the resumes in the web application dashboard.

4.1.1 Frontend

The services provided by frontend are: Registration, Login, Upload job description, feature weight, resume as well as number of shortlisted candidates, displaying list of top n candidates.

The first step in the system is login and registration. Registered user can login to the system and unregistered user can register using credentials like email and password. The password field is to be repeated twice. Once the user presses register, the backend validates email and password. If the passwords match, credentials are stored in the database otherwise an error message is displayed. Then the user can uses the same credentials to login to the system. After logging into the system, the user is presented with the option to upload one job description, feature weight, a number of resumes and assign the number of resumes to be shortlisted. Once these fields are filled, the backend stores the input data into the database. After submitting the form, the backend undergoes certain operations and computations to provide each resume with a similarity score with respect to the job description. Using this score, the resumes are ranked in order and a certain number of resumes given by the user is displayed in a table along with their personal information and features like education, skill, etc.

4.1.2 Backend

Django serves as the strong backbone of our application. To uphold strict privacy demands, users are prompted to log in. When the user enters their credentials (username and password), a POST request is sent to the server. The server checks if the user is authenticated.

After logging in, users are redirected to the upload page, where they can submit resumes and job description files along with feature weights and number of candidates to be shortlisted. Information about job descriptions is intended to be stored in the JobDescription model. This model has fields for the job description file (jobdescription), weights for skills (skills), education requirements (education), language (language), and experience expectations (experience). Every Job Description and Resume is associated with a specific user and has fields for the resume file (resumes). Tables in the database are organized in accordance with the specified models. The text from resume and job description objects are then extracted and our customized NER model is loaded to recognize the required entities. The NER model that is sent to the feature extraction block assists in extraction of features like skills, experience, education, etc. The extracted features of resumes and job description are then separately converted to word vectors using Fasttext, a word embedding model. After that, each feature of the job description is compared with each individual resume's corresponding feature one by one. In this way, similarity score of each feature between job description and each resume using cosine similarity. Based on this, the resumes are ranked from most similar to list similar and only the number of resumes that are to be shortlisted are displayed in a table.

4.1.3 Training NER Model for Resume

This NER Model extracts features from resumes. For this purpose, resumes annotated with required tags like skills, education, experience, age, etc. was fed into the system. The annotated data was then split into test and train data in the ratio 70:30. After training the system was evaluated based on precision, recall and F1 score.

4.1.4 Training NER Model for Job Description

This NER Model extracts features from job descriptions. It follows the same procedure as the NER model trained for resume. The NER is trained using the RoBERTa model.

4.2 Flowchart

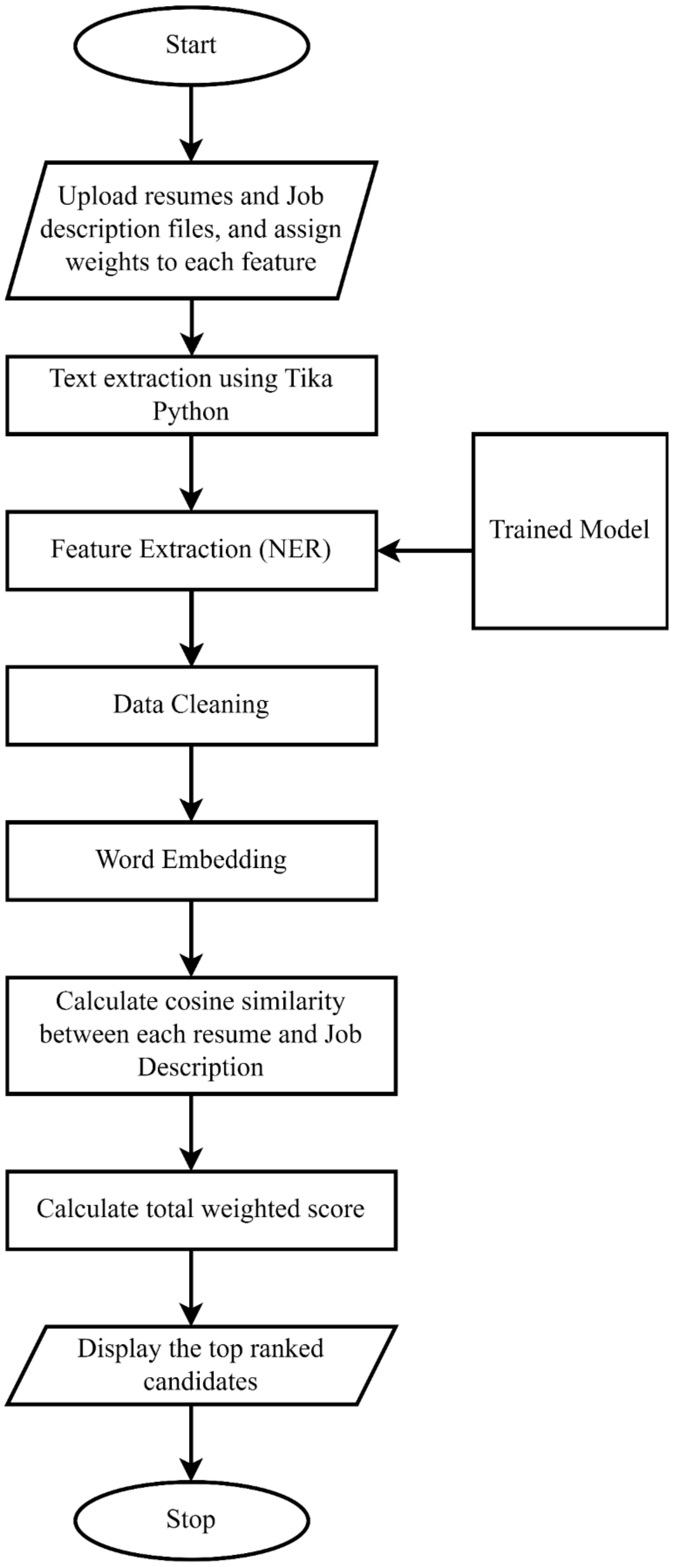

Start
Upload resumes and Job description files, and assign weights to each feature
Text extraction using Tika Python
Feature Extraction (NER)
Trained Model
Data Cleaning
Word Embedding
Calculate cosine similarity between each resume and Job Description
Calculate total weighted score
Display the top ranked candidates
Stop

Figure 4-2: Flowchart

In this system, we collect resumes from an individual or a company for a job. The data is extracted using the Tika library and then preprocessed using an NLP module like NLTK which helps to clean data, tokenize text by word or sentence, filter stop words, and SpaCy for Named Entity Recognition (NER). Then the word is vectorized using word embedding. Each feature has its weights that help in ranking priorities. For ranking resumes, feature vectors of resume and job description are extracted and cosine similarity between the resumes and job description file is calculated. After each resumes have similarity scores, they are ranked in descending order of similarity scores.

4.3 Use Case Diagram

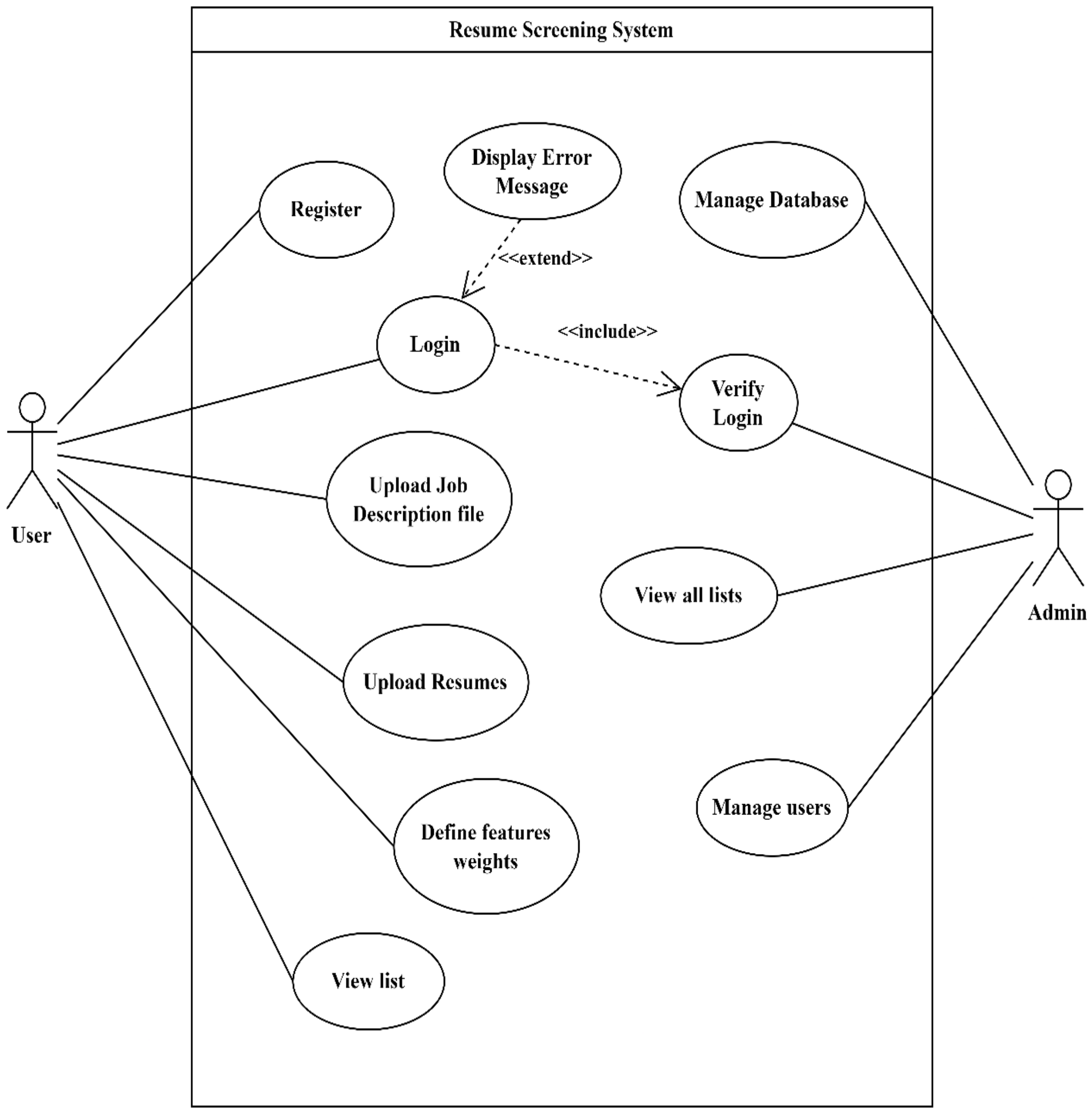


Figure 4-3: Use Case Diagram

Table 4-1: Description of Use Case Diagram

Use Case Name	Resume Screening System
Participating Actors	User and Admin
Preconditions	Users should use the system website.
Flow of events	• The user needs to register before using the system. After registration, the user can log in to the system and upload the files. • The user must upload one or more resumes and one job description file. • The user needs to define weight for different features. • The user then can view the ranked list of each resume based on the job description. • The admin can manage all the users and the files being uploaded.

4.4 Data Flow Diagram (DFD)

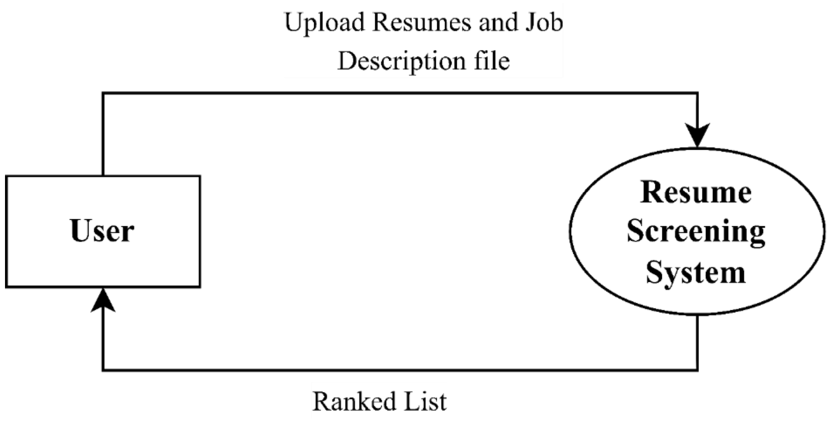


Figure 4-4: DFD Level 0

The above diagram is the conceptual diagram of the Resume Screening System. The Resume Screening System has one external entity: User. The user interacts with the system through a web interface. The user uploads resumes and job description files to the system and after processing, a ranked list is displayed to the user. To upload files, the user needs to log in to the system. Along with the files, the user defines weight for different features as per their priority while ranking.

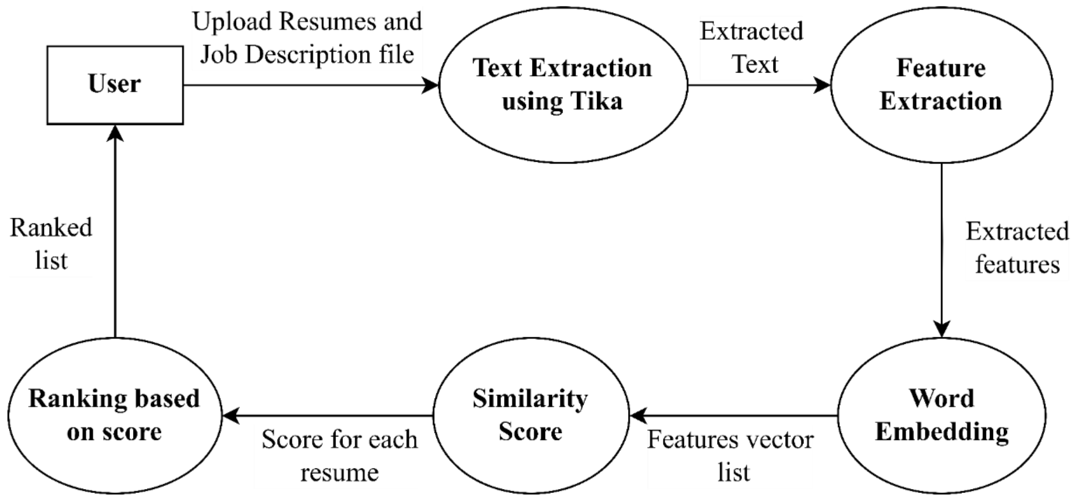


Figure 4-5: DFD Level 1

The Resume Screening System can be further divided into many steps. First, the user uploads resume files and a job description file and defines the weight for each feature. Then, text is extracted from the files uploaded using Tika. The extracted text is then passed into trained model to predict the features. The features words are then converted to vectors using word embedding. The similarity score can be calculated and ranking is done based on score. Finally, a ranked list is displayed to the user. This Level 1 DFD provides a detailed view of different processes involved in the system.

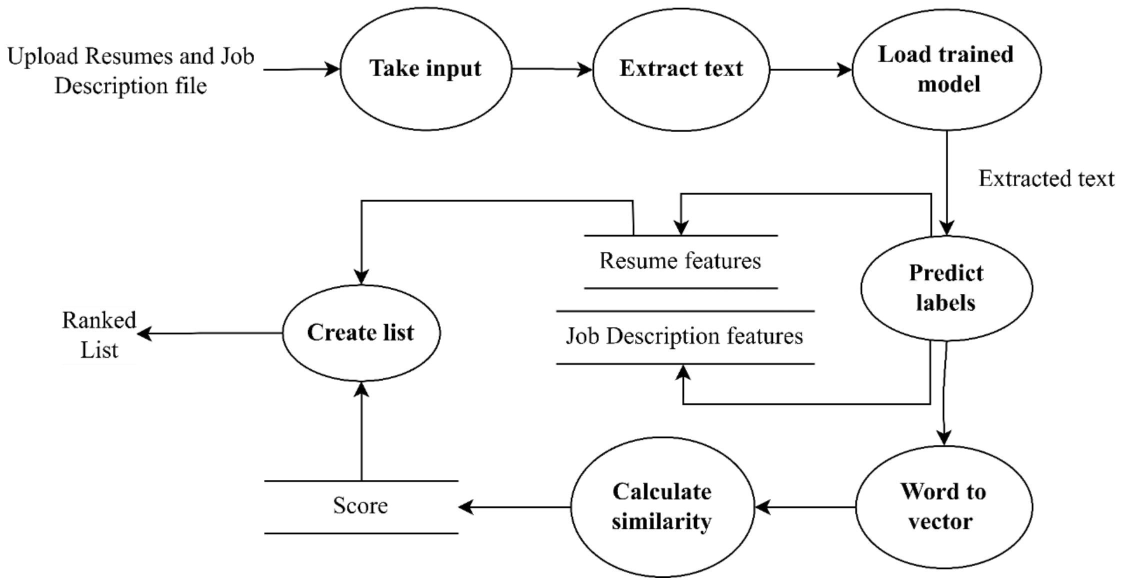


Figure 4-6: DFD Level 2

The main processes explained in Level 1 can be further subdivided into even more detailed sub-processes, in Level 2. The text extraction process can be subdivided. First, the input from the users is gathered and text is extracted using Tika. Then, the trained model is loaded into the system and labels are predicted from the text extraction. The predicted labels for each resume and job description are stored in the database. Each word from the feature list is converted into vectors and the feature vector can be calculated as the average of all words in the list. Then, the similarity score between the feature vectors of each resume and job description is calculated. The total score for a resume is calculated as the weighted sum of all feature vectors. Finally, a ranked list is generated with all the features of resumes including their scores based on scores.

4.5 Class Diagram

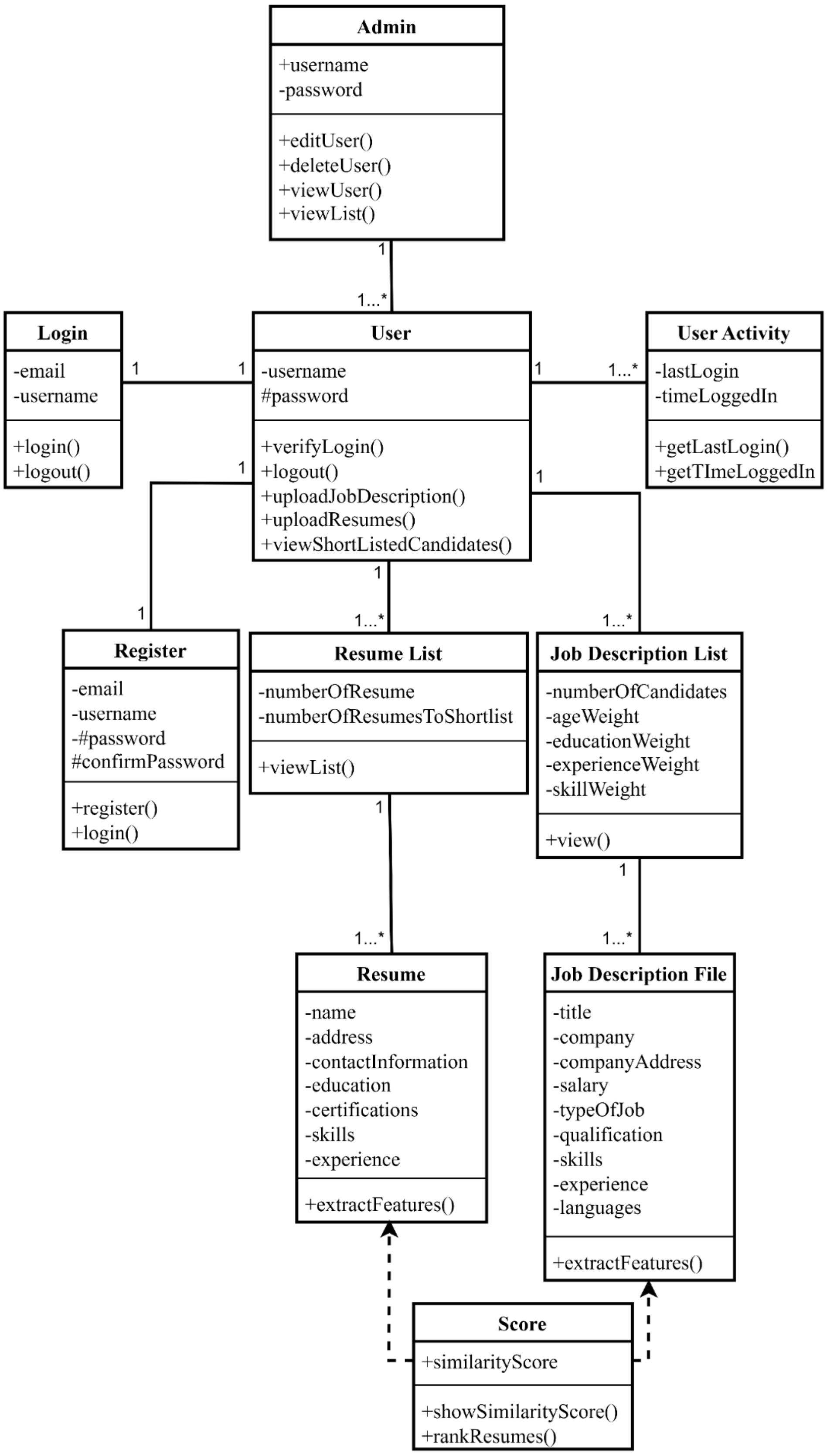


Figure 4-7: Class Diagram

The class diagram shows the structure of the resume screening system along with the relationship between different classes. The admin class monitors the overall functioning of the Resume Screening System. It administers the creation, editing and deletion of users and their information. The user class represents the HR professionals who use the system to upload job description along with the received resumes which is then worked on by the internal ML model to produce a list of shortlisted candidates.

The user activity class monitors the activity of the user such as login activity and time spent in the system. The register class assists in the management of user registration information such as usernames and passwords. The login class manages user login. The resume list class and the job description list class keep track of the job description file, resumes uploaded, shortlist count and weights for (like age, skills, education, experience, etc. The resume contains information of candidates such as personal information, education, skills and experience. The job description file contains the information of the vacant job position such as job title, category, title, company information, responsibilities, etc. The score class represents the similarity score computed based on the similarity of features between job description and each resume.

4.6 Sequential Diagram

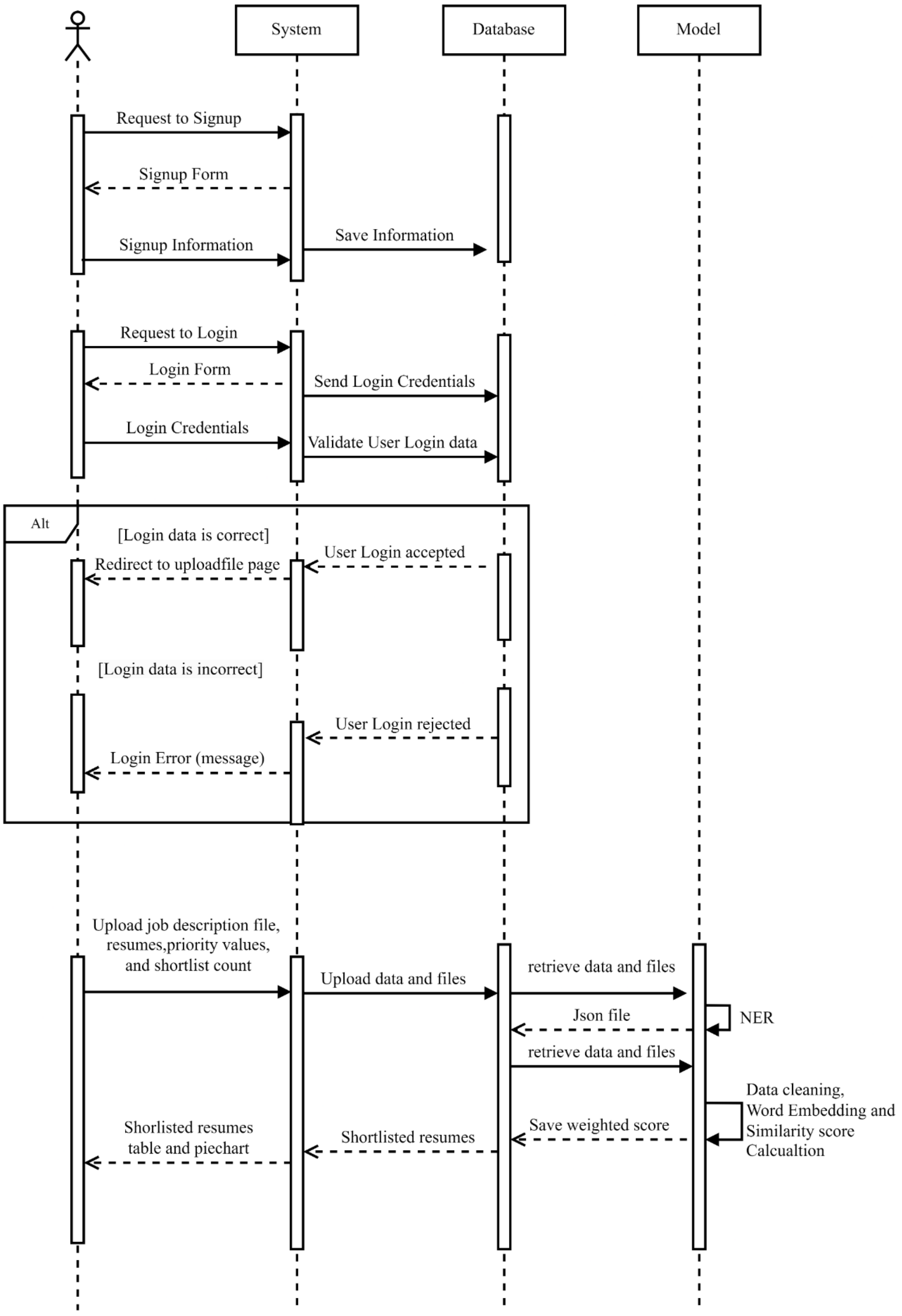


Figure 4-8: Sequential Diagram

The sequence diagram illustrates the dynamic behavior of the Resume Screening System from user signup to the presentation of shortlisted resumes. It showcases the interactions between its key objects; System, Database, Model and an actor; User. The interaction between objects is depicted using messages in sequential order.

Signup

- The sequence commences with the User initiating the signup process by requesting signup from the System.
- The System responds by presenting a signup form to the User, who enters the required information.
- Upon form submission, the System saves the user details into the Database.

Login

- Once user have signed up, the user requests to log in to the System.
- The system presents a login form to the User, who provides login credentials.
- The system validates the login data by querying the database. If correct, the User is accepted and redirected to the upload file page; otherwise, an error message is displayed.

File Upload and Processing:

- Once logged in, the User can upload a job description file, a specified number of resume files, along with priority values and shortlist count.
- Then, the system saves this data to the database.

Model Processing:

- Model retrieves the uploaded data from the Database and performs Named Entity Recognition (NER).
- Extracted data are stored in JSON files, which is subsequently saved in the Database.
- Then, model executes data cleaning, word embedding, and cosine similarity retrieving the saved JSON files from the database.

Resume Shortlisting:

- Model assigns weighted scores to the resumes.
- Based on the shortlist count provided by the User, resumed are shortlisted and saved to the database.
- System displays the shortlisted resumes in tabular and pie chart forms to the User.

4.7 ER Diagram

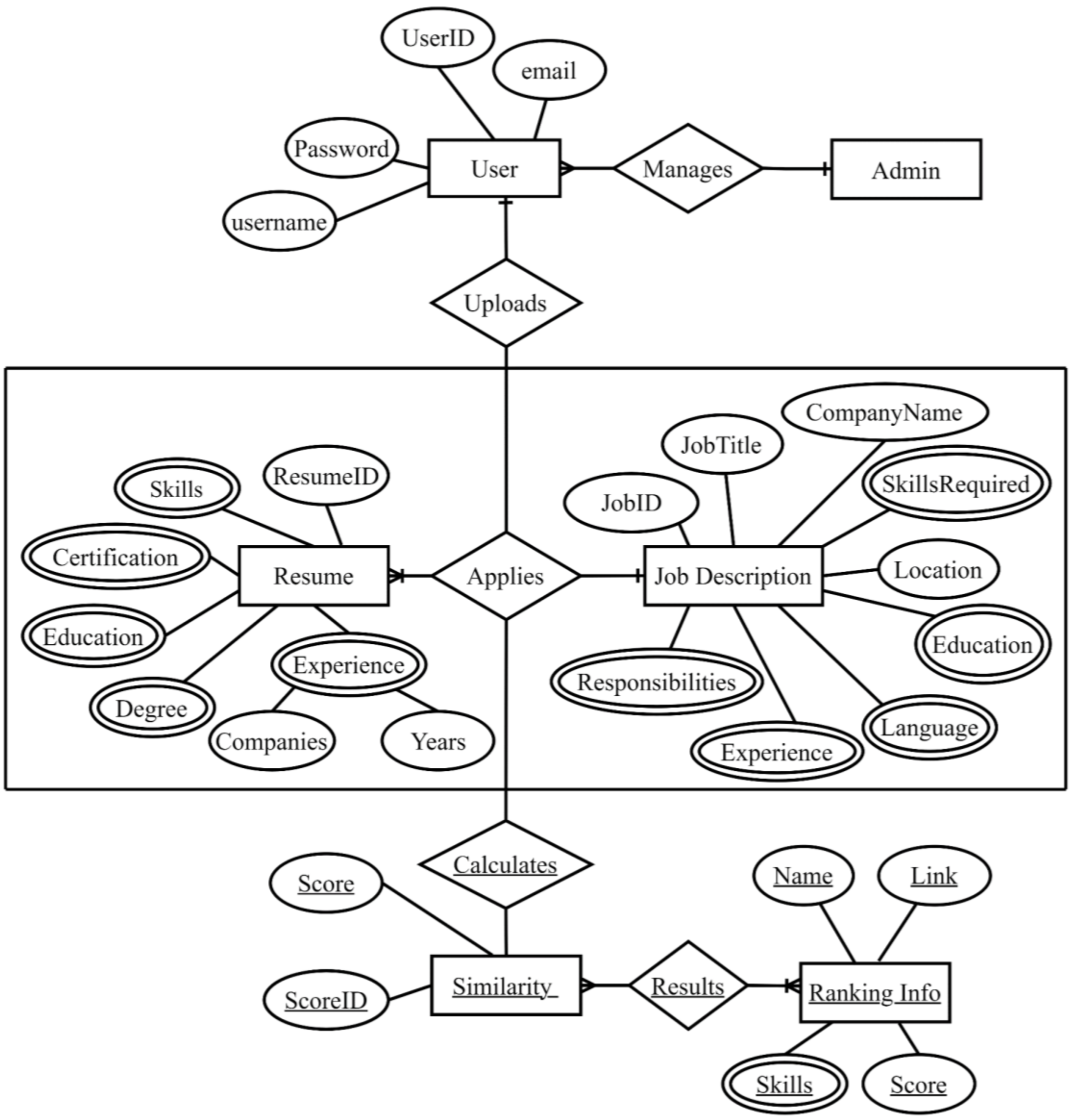


Figure 4-9: ER Diagram

The database schema comprises several entities, including User, HRUser, Resume, JobDescription, SimilarityScore, and RankingInfo.

Resumes contain pertinent information such as ResumeID (primary key), UserID (foreign key linking to User.UserID), Name, Skill, Experience, YearsOfExperience, and Degree. On the other hand, JobDescription entities represent available job positions, featuring attributes like JobID (primary key), JobTitle, SkillsRequired, Responsibilities, and Location. The system calculates the similarity between resumes and job descriptions, resulting in the creation of SimilarityScore entities. Each SimilarityScore entry is identified by a unique ScoreID (primary key) and includes ResumeID and JobID as foreign keys referencing Resume.ResumeID and JobDescription.JobID, respectively. The calculated similarity score is stored as an attribute. To provide users with valuable insights into candidate rankings, the schema introduces the RankingInfo entity. RankingInfo entities are created based on the results of similarity score calculations. Each RankingInfo entry is identified by a unique RankingID (primary key) and includes ResumeID as a foreign key referencing Resume.ResumeID. Additional attributes include Position, Name, Link, Phone, and Skills, offering comprehensive information about the top candidate.

4.8 BERT Model Architecture

BERT – Bidirectional Encoder Representations from Transformers is a simple and powerful model that is designed to capture the information of unlabeled text from both left and right directions within a sequence. This bidirectional approach helps to capture more information and relationships between unlabeled texts.

The BERT model is the first fine-tuning-based model. There are two steps in the BERT model: Pre-training and Fine-tuning. During pre-training a model i.e. training a model on a large dataset from scratch, it has no specific task allocated. The BERT is pre-trained using two unsupervised tasks: Masked LM and Next Sentence Prediction. The next sentence prediction is an important factor to create a relationship between two sentences. During the fine-tuning process, the model starts getting trained for specific tasks without forgetting what it learned during the pre-training process. It only specializes the model to a specific task by updating the parameters on labeled data without the need to train the model from scratch. The fine-tuning in BERT is simple and inexpensive as the self-attention mechanism in the transformer allows to train for a specific task.

BERT uses a Masked Language Model (MLM) strategy. This Masked Language Model randomly masks certain words while pre-training and then the model is trained to predict the masked words. BERT has minimal differences between the pre-trained architecture and the fine-tuned architecture.

BERT architecture is built upon the Transformer model. [15] This transformer model is a neural network architecture that relies on self-attention mechanisms. BERT's model architecture contains multiple layers of bidirectional Transformer encoders each capturing information bi-directionally.

The transformer model is based on self-attention mechanisms i.e. it focuses on different parts of the input sequence to produce each part of the output sequence. The transformer model uses 6 stacked layers in both the encoder and decoder. The encoder contains two sub-layers: multi-headed self-attention and feed-forward. Decoder contains three sub-layers: masked multi-head attention, multi-headed attention, and feed-forward. The Encoder processes the input sequence while the decoder generates the output sequence of dimension 512 from encoded information. The layer normalization is used after each sub-layer that normalizes its output. Multi-headed attention helps to get the information from different representations. The use of attention helps to take into account the output from the encoder as well as the previous decoder layer, allowing the decoder parts to be generated based on all the encoders. The self-attention layer in the encoder takes in the output from the previous layer of the encoder. Similarly, in the decoder, the self-attention layer takes the output from all previous layers of the decoder including itself. The feed-forward network is used in each layer that helps to process the input into complex, non-linear patterns. As the hidden layers use vectors to represent each input and output, the transformer model uses learned embeddings to convert input and output tokens into vectors. To represent the outputs in probabilities, it uses softmax.

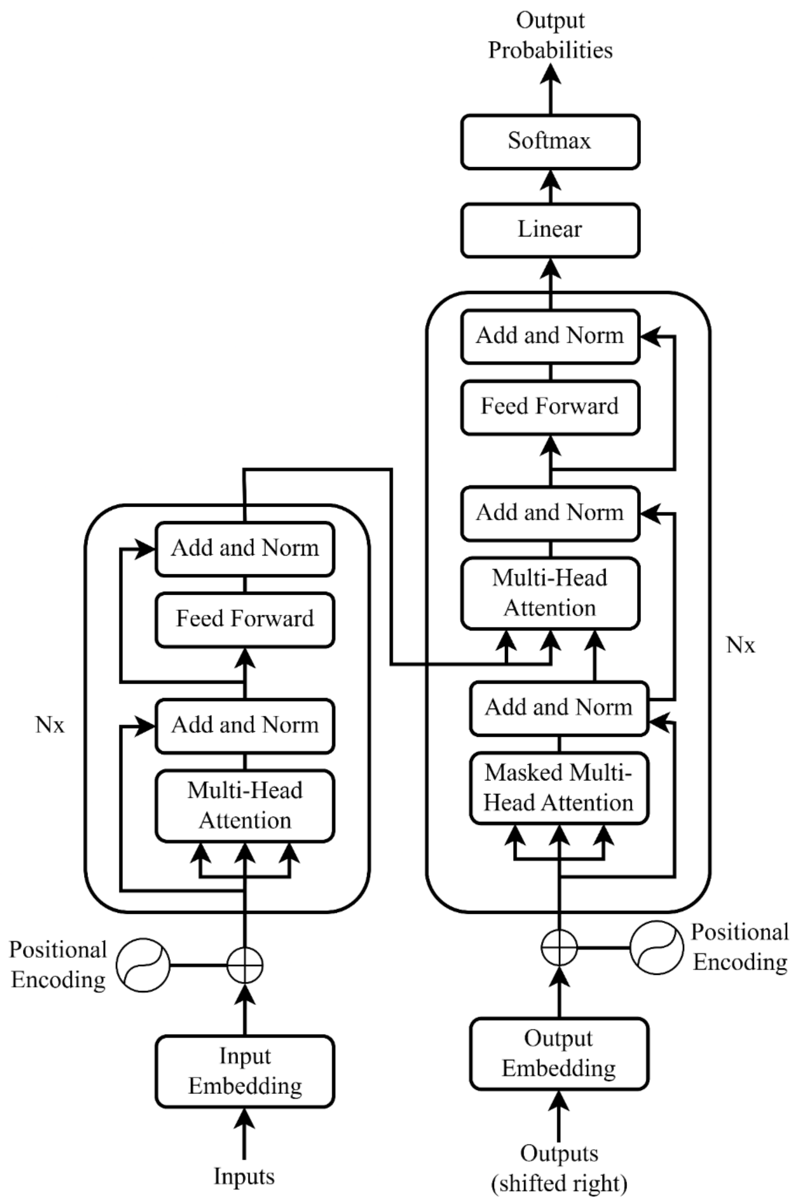


Figure 4-10: Transformer Architecture

4.9 Tika Architecture

The architecture of Tika provides four modules: Language detection mechanism, MIME detection mechanism, Parser interface, and Tika Facade class. The language detection method detects the language in which the document is written which is then added to the metadata of the document while the MIME detection mechanism detects the document type. The Parser interface is used for the extraction of text and metadata from the provided document to summarize it for plugins. Since Tika is a Java library, Tika Façade Class is a simple and direct method of calling Tika from Java. It uses an

N-gram algorithm for language detection. An N-gram is a sequence of adjacent items in a given document. An N-gram language model is used to predict the occurrence of any n-gram in a sequence of words.

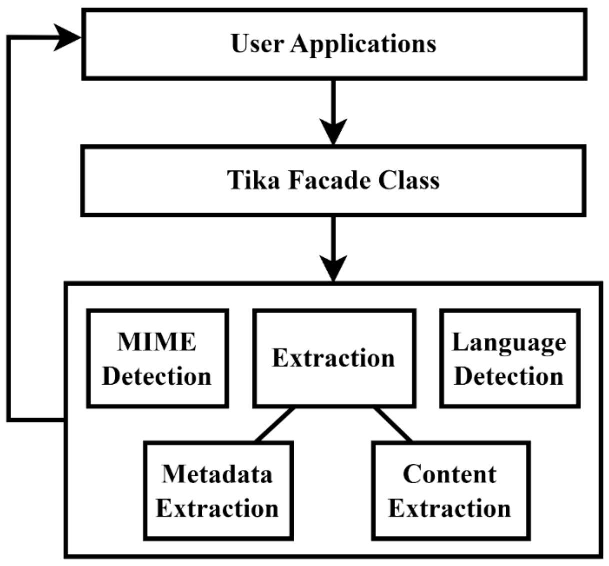


Figure 4-11: Tika Architecture

4.10 FastText Model

FastText supports two models: a continuous bag of words (CBOW) that predicts the targeted word based on surrounding words and a Skip-gram model that predicts the words surrounding the target word. Even though it supports both models, FastText primarily uses the continuous bag of words (CBOW) with Negative Sampling model for word embedding.

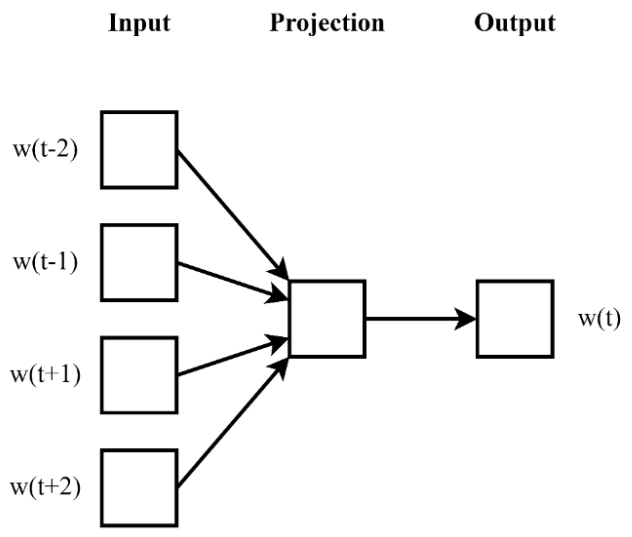


Figure 4-12: CBOW Model of Word Embedding

The main goal of using Word Embedding is to get the appropriate weights of the neural network layers. The output of the training is the creation of vectors for all inputted resumes as well as the job description. Each resume is then represented as the average of the word vectors of its constituent words. This is done by taking the mean of the vectors for all words in the document.

5. IMPLEMENTATION DETAILS

5.1 Tika

Apache Tika is an open-source toolkit used for the detection and extraction of relevant content like metadata, text as well as structured data from different document formats such as pdf, docs, HTML, spreadsheets, images, etc. Tika-Python, a Python binding to Apache Tika, allows Tika to be used as a Python library and provides an efficient document processing capability using an existing document parser repository for the detection and extraction of data. However, to run Tika-python, Java must be installed in the system as Tika acts as a Python interface for the Apache Tika library which is based on Java. The Steps involved in Tika library is given as:

File upload

Using the requests library in Python, an HTTP PUT request is sent to the Tika server with file content as its payload in the background.

Content Type Detection

Then the Apache Tika server analyzes the content of the received file. The server uses a content type detector to identify the file type like a PDF file. It is important as different file has different mechanisms to extract text. To detect the type of content, it checks the content of magic bytes, file signatures, common features of a format and sometimes even file extensions. Magic bytes are special byte sequences that are available at the beginning of a file to identify its format uniquely. As for PDF files, it might include characters like %PDF. File signatures are patterns in a file that are often referred to as the file header. To determine the content type of a file, a function detector.from_file() is used that returns a string like ‘application/pdf’.

Parser selection

After the content type is detected, the appropriate parser is used. In the case of a PDF file, the server uses a PDF parser that understands the internal structure of the PDF to extract its text content and metadata with other information.

Content Parsing

First, the structure of the PDF file is examined to check its fonts, images, pages, and content organization. Then, it extracts the text content and metadata with other information from tables, links, annotations, and images, if needed.

Response Generation

After text extraction in JSON format, the Tika server generates a response for the HTTP request from Python.

Python Interface

The Tika library then parses the JSON response and returns a Python dictionary that includes keys like content and metadata. To extract the text content and metadata, it uses a parser.from_file() function.

5.2 Natural Language Preprocessing (NLP)

This stage involves using Natural Language Processing (NLP) to perform NER and process the data to remove stop words. These processes are performed using the SpaCy library.

5.2.1 Named Entity Relationship (NER)

Named Entity Relationship (NER) classifies named entities into predefined categories such as person names, experience, education, skills, qualification, and so on. NER helps to standardize the format of extracted data across different resumes which makes it easier to match the candidate's resumes with job descriptions.

For this, the SpaCy transformer library was used. The transformer provides an API that assists to download and train state-of-art pretrained models easily. The RoBERTa model was used to train the NER model using a set of annotated resumes to extract features like name, address, link, skills, language, certifications, awards, companies worked at, worked as, years of experience, university, degree, etc. This NER model uses semi-supervised learning approaches use a small amount of labeled data (are called seed) then combine it with a large amount of unlabeled data or corpus. The semi-supervised learning “bootstrapping” method that is used for named entity recognition (NER).

5.2.2 Data Cleaning

Once NER is applied to resumes, the words and phrases are classified into various named entities such as skills, education, experience, etc. However, there is a possibility that those phrases may contain stop words which hinders the word embedding process. Therefore, stop words are removed using a function of SpaCy library.

5.3 Word Embedding

Word Embedding is used for detecting similarities between words. It is a numerical representation of individual words as vectors that can have tens or hundreds of dimensions. It can be achieved using Word2vec, Doc2vec, FastText, and BERT.

For our project, we used pretrained FastText model due to its ability to find semantic similarities and the availability of vectors of technical domain.

5.4 RoBERTa Model in Spacy Transformer

5.4.1 RoBERTa Model

RoBERTa – Robustly Optimized BERT Pre-Training Approach is an improved procedure for training BERT models. The study of the BERT model later revealed that the model was undertrained which resulted in the RoBERTa model. The RoBERTa model was presented by researchers at Facebook and Washington University. The main goal of the RoBERTa model was to optimize the training of BERT architecture to minimize the time during pre-training. This model has a similar architecture as BERT with some additional changes to improve the pre-training process.

This model results in better performance on different tasks like Stanford Question Answering Dataset (SQuAD), General Language Understanding Evaluation (GLUE), and ReAding Comprehension from Examinations (RACE). This model was evaluated by training on a large dataset (CC-News), to evaluate whether larger training dataset results in better performance.

Modifications to BERT

- This model is trained on larger data with bigger batches and longer training time.
- This model had applied a dynamically changing masking pattern to the training data.
- This model was trained on longer sequences.
- This model removes the predictions of the next sentences.

Dynamic Masking

The BERT model performed the same mask for all duplicates of an input sequence known as static masking. Whereas Dynamic masking is a process where different masking patterns are generated every time a sequence is fed i.e. duplicate sequences have different masks. So, dynamic masking in RoBERTa results in slightly better than static masking in BERT.

Model Input Format and Next Sentence Prediction

BERT's training includes the Next Sentence Prediction (NSP) loss to understand the relationships between the two sentences. However, RoBERTa experiments showed that no use of NSP loss did not have a significant impact on the model performance. Instead, it used the Masked Language Model (MLM). The RoBERTa experiments showed Doc-Sentences performed better than Full-Sentences but due to other considerations, RoBERTa uses Full-Sentences instead of Sentence-Pair.

Training with larger batches

RoBERTa experiments showed that training with larger batches improved the predicting capacity for Masked Language Modeling and also improved the accuracy of the downstream tasks (the tasks the model is being trained for after pre-training).

Training on longer sequences

The BERT model used a character-level Byte-Pair Encoding (BPE) vocabulary. While the RoBERTa also used additional sub-word units along with the character-level BPE vocabulary and there were slight differences between these encodings.

5.4.2 Spacy Transformer

Space transformer is used to train a customized Named Entity Recognition (NER) for our system. Spacy transform lets you use all the transformer models that are available in the HuggingFace Transformer library. It connects the components to the transformer using the TransformerListener layer. The component assigns the output from the transformer to the Doc's extensions. Here, the Thinc model is integrated into Spacy to define complex models and combine deep learning components into Spacy pipelines. We used RoBERTa model in spacy transformer.

5.4.3 Training in Spacy

The Spacy allows for training and updating components on customized data and integration of custom models. The training process compares the model's prediction with the annotated text. To train the model in customized data, each data needs to be first converted to spacy format. In spacy format, a text has all predicted labels in the following format.

["Position: <u>Laravel Developer</u> Company info: <u>Bibaabo Service Pvt. Ltd.</u> is a versatile player in the FinTech industry, operating successfully since 2022. ", {"entities": [[11, 27, "JOB TITLE"], [43,67, "COMPANY NAME"]]}]

Each entity has a start index, end index, and label. Here, two labels are annotated from the text: Job Title and Company Name.

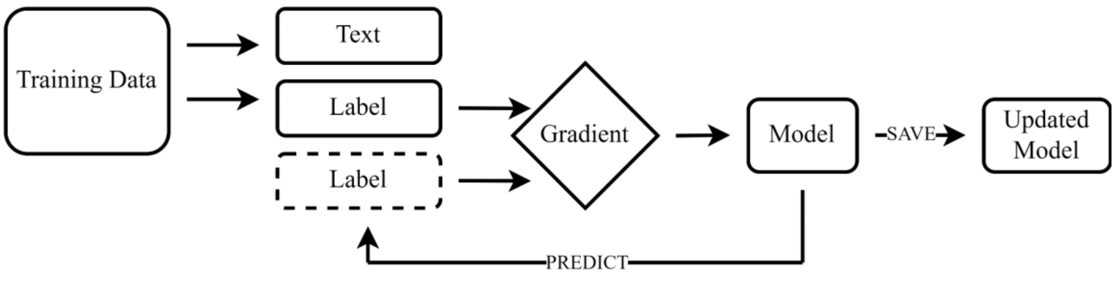


Figure 5-1: Training Process in SpaCy

During the training process, the annotated text with the list of entities is passed. Firstly, the prediction of the entities for the text are predicted and compared with the entities passed, based on these the NER loss is evaluated and the best model is saved. Here, the dataset is split into training and validation data. Our model was evaluated in different train dev ratios of 70/30, 80/20, and 90/10. The model is first trained on the train data and fine-tuned based on dev data.

5.4.4 Spacy Pipeline

Spacy pipeline is a sequence of components and each component is an actual model that forms some text processing tasks. Spacy supports several multi-task learning workflows due to which one transformer is sufficient for other components like ner, textcat, etc. Here, the transformer serves as a shared Embedding layer.

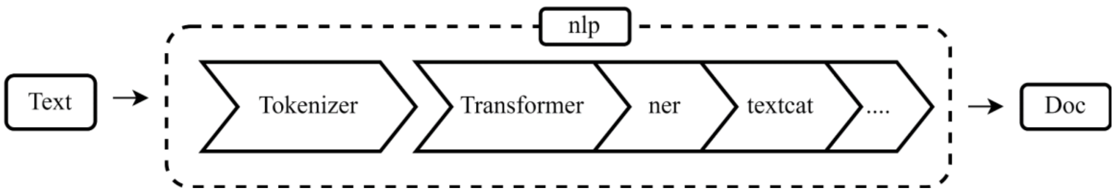


Figure 5-2: SpaCy Pipeline

Each component is connected to the transformer through a listener layer within their model. When each component processes documents, it will forward its predictions to the listeners, allowing it to update the model using the predictions. The transformer component will also save its output so that the data can be accessed in the future.

5.4.5 Training Config

Spacy v3 supports training pipelines via command on the command line. All the settings and hyperparameters for the transformer model can be included in one single configuration file config.cfg. This reduces all the multiple steps used to define the hyperparameters to train the model. This process simplifies the training process that is provided by Thinc library. The config file is used for both training and runtime.

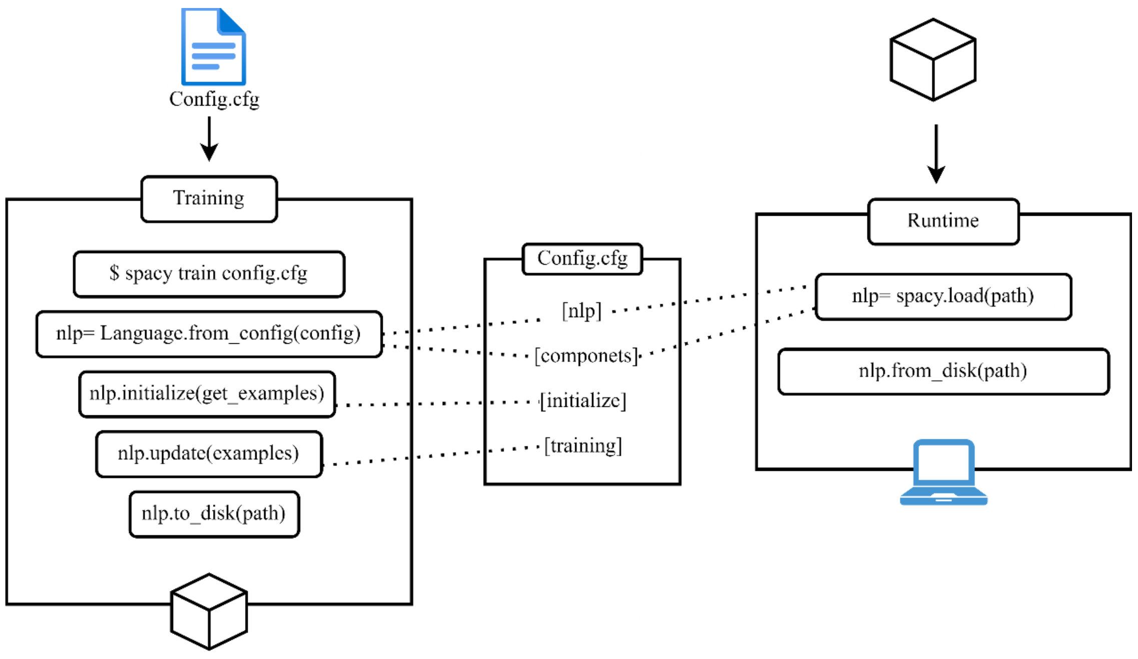


Figure 5-3: Training SpaCy Model

During training, NLP, components, initialize, and training blocks from the config file are used. At runtime, only NLP and component blocks of config is used to load the model and use it to predict NER for unlabeled text.

NER model

The NER model uses spaCy's "spacy.TransitionBasedParser.v2" architecture. The hidden width for NER model is set to 64 indicating 64 units of hidden layer. The maxout pieces is set to 2, which indicates the number of units inside the hidden layer.

Transformer model

The maximum number of batch items is set to 4096. This represents the number of tokens or samples per batch during training. The pre-trained transformer model used is "Roberta Base". Span is a sequence of tokens or words extracted. Here, the window is set to 128 meaning 128 tokens for a span will be extracted. The stride is set to 64. It determines the steps by which the window moves forward between consecutive spans.

Training Parameters

Table 5-1 Parameters for Training Resume NER Model

Parameters	**Training Values**	**Fine Tuned Values**
Accumulate gradient	3	3
Dropout	0.1	0.1
Patience	2000	2000
Max steps	6000	6000
Evaluation frequency	100	200
Batch size	512	2000
Buffer	256	256
F1 score	0.92	0.96

The initial parameters to train the resume NER model produced an F1 score of 0.92 so the parameters like evaluation frequency and batch size were changed which resulted in an increase of F1 score to 0.96.

Table 5-2: Parameters for Training Job Description SpaCy Model

Parameters	**Training Values**	**Fine Tuned Values**
Accumulate gradient	3	2
Dropout	0.1	0.1
Patience	1000	1000
Max steps	5000	5000
Evaluation frequency	200	25
Batch size	128	360
Buffer	48	256
F1 score	0.60	0.68

Similar to resume NER model, the job description NER model also required modification of parameters like accumulate gradient, evaluation frequency, batch size and buffer to increase its F1score from 0.60 to 0.68.

Adam Optimizer Parameters

Table 5-3: ADAM Parameters for Training Resume and Job Description NER

Parameters	**Value**
Beta1	0.9
Beta2	0.999
L2 Regularization	0.01
Gradient Clipping	1.0
Epsilon (ep)	0.00000001
Initial learning rate	0.00005
Total Training steps	10,000
Warm-up Steps	300

The default ADAM optimizer parameters were used to train the resume and job description NER model.

5.5 FastText for Word Embedding

FastText Word Embedding model is an open-source python library word embedding model created by Facebook AI Research (FAIR). The fastText word embedding model provides pretrained word vectors in 157 different languages that are trained on Common Crawl and Wikipedia. The model was trained using CBOW model with position-weights to take context of words into account, with n-grams of length 5, window size 5 and 10 negatives. The resulting vectors had a dimension of 300 in vector space.

The fastText model uses n-gram technique to train the model and offers character level word embeddings where each word is represented with its Gaussian mixture density calculated by taking the mean of the sum of n-grams. The size of n-gram can be determined by using minimum and maximum flags that represent the minimum and maximum number of characters in an n-gram.

The Gaussian density function associated is given by

$$f(x) = \sum_{i=1}^{K} P_{w,i} \, N(x; \underset{\mu_{w,i}}{\longrightarrow}, \Sigma_{w,i}) \qquad 6.1$$

where,

$\{\mu_{w,i}\}_{k=1}^{K}$ are mean vectors

$\{\Sigma_{w,i}\}$ are the covariance matrices

$\{p_{w,i}\}_{k=1}^{K}$ are the component probabilities which sum to 1

The mean vector μ_w for word w over n-gram vectors is

$$\mu_w = \frac{1}{|NG|+1} (v_w + \sum_{g \in NG_w} z_g) \qquad 6.2$$

where,

z_g is a vector associated with an n-gram

v_w is the dictionary representation of word w

NG_w is a set of n-gram of word w

The similarity metric is represented by generalized dot product in Hilbert space called expected likelihood kernel. The energy is given by

$$E(f, g) = \log \sum_{j=1}^{K} \sum_{i=1}^{K} p_i q_j e^{\varepsilon_{i,j}} \qquad 6.3$$

where,

$$f(x) = \sum_{i=1}^{K} p_i \, N(x; \underset{\mu_{f,i}}{\longrightarrow}, \Sigma_{f,i}) \qquad 6.4$$

$$g(x) = \sum_{j=1}^{K} q_j \, N(x; \underset{\mu_{g,i}}{\longrightarrow}, \Sigma_{g,i}) \qquad 6.5$$

$\varepsilon_{i,j}$ is the partial energy which corresponds to the similarity between component i of the first word f and component j of the second word g.

This model uses Huffman algorithm to build tress where the depth is inversely proportional to the frequency of words. The use of binary tree minimizes time of search.

In resume screening system, the FastText follows NER. For the features of resumes that need to be compared with features of job description, each object of the feature is converted into its corresponding word vector using word embedding. Cosine similarity is then applied to these word vectors to compute similarity between features.

Table 5-4: Comparison between Pretrained and Trained FastText Model

Words	**Pretrained**	**Trained Model**
OOP and CPP	0.21765877	0.4667172
OOP and Python	0.3921288	0.14867683
OOP and OOP	1.0	1.0
Frontend and HTML	0.32181436	0.10461625
Frontend and Django	0.25624192	0.124352336
Backend and Django	0.28597176	0.03708393
Frontend and CSS	0.31264055	-0.013093519
Apple and Programming	0.20136516	0.040716965
HTML and CSS	0.64761007	0.020212071

5.6 Cosine Similarity

Now we can calculate the similarity using the Cosine Similarity metric. Cosine Similarity is a metric that measures the similarity of two vectors. Introducing it into a resume tracking system helps assess how closely resumes match job descriptions and can then be used to rank the resume document. Cosine similarity checks the cosine of the angle between two vectors. The documents are similar if the angle is close to zero. It does not consider word frequency and order of the words.

Mathematically:

$$\cos\theta = \frac{\underset{a}{\rightarrow} \cdot \underset{b}{\rightarrow}}{\left|\underset{a}{\rightarrow}\right|\left|\underset{b}{\rightarrow}\right|} \tag{5.1}$$

where,

$\underset{a}{\rightarrow}$ represents word vector of each element in a feature list in job description

$\underset{b}{\rightarrow}$ represents word vector of each element in a feature list in resume

For n-dimensional, we calculate for each word. The cosine similarity value ranges from -1 to 1 where -1 represents the most dissimilar vectors and 1 most similar vector. To calculate cosine similarity in Python, the sklearn package has a cosine similarity function.

The resumes are evaluated to extract the following features: Skills, Education, Experience and Language.

For different skills, different weights are given by the user (HR) while uploading the job description. This overall score for each resume based on the job description is:

$$re = \sum weight(features) * cosine(features) \tag{5.2}$$

6. RESULT AND ANALYSIS

6.1 Comparison of different models for training dataset

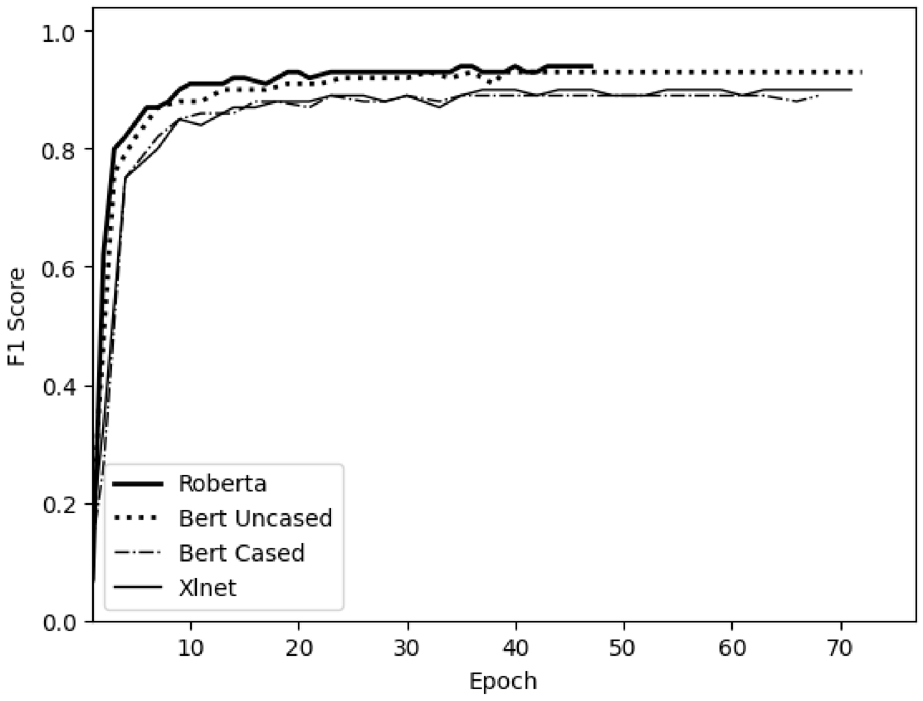


Figure 6-1: F1 score of Roberta, Bert Uncased, Bert Cased, and Xlnet models

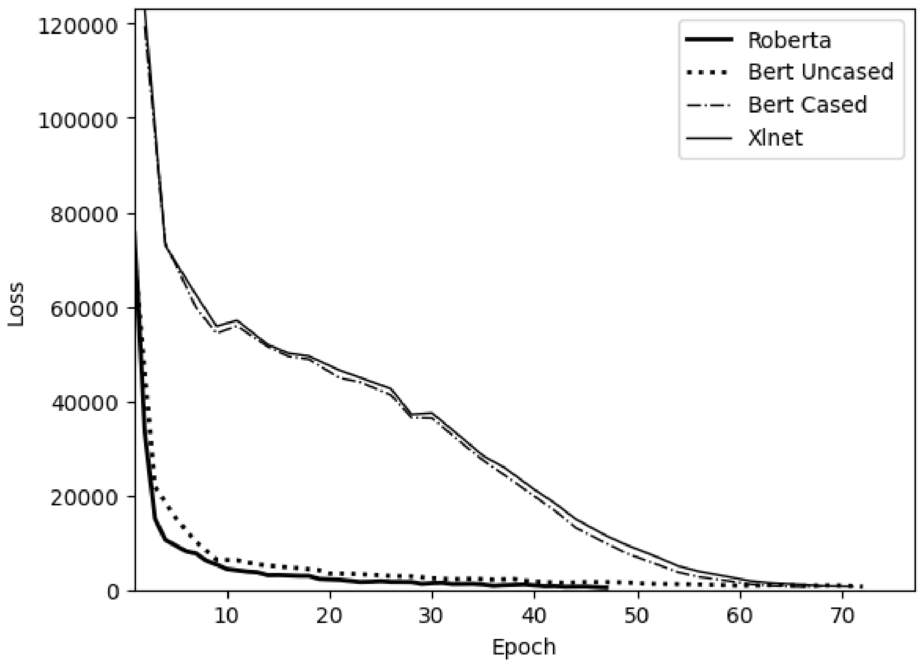


Figure 6-2: Loss of Roberta, Bert Uncased, Bert Cased, and Xlnet models

Observing the above F1 score VS Epoch and Loss VS Epoch, the Roberta model performs better than other models. The Roberta model provides a better F1 score while also providing lower loss on the same dataset and with fewer epochs used.

6.2 Resume NER using RoBERTa Model

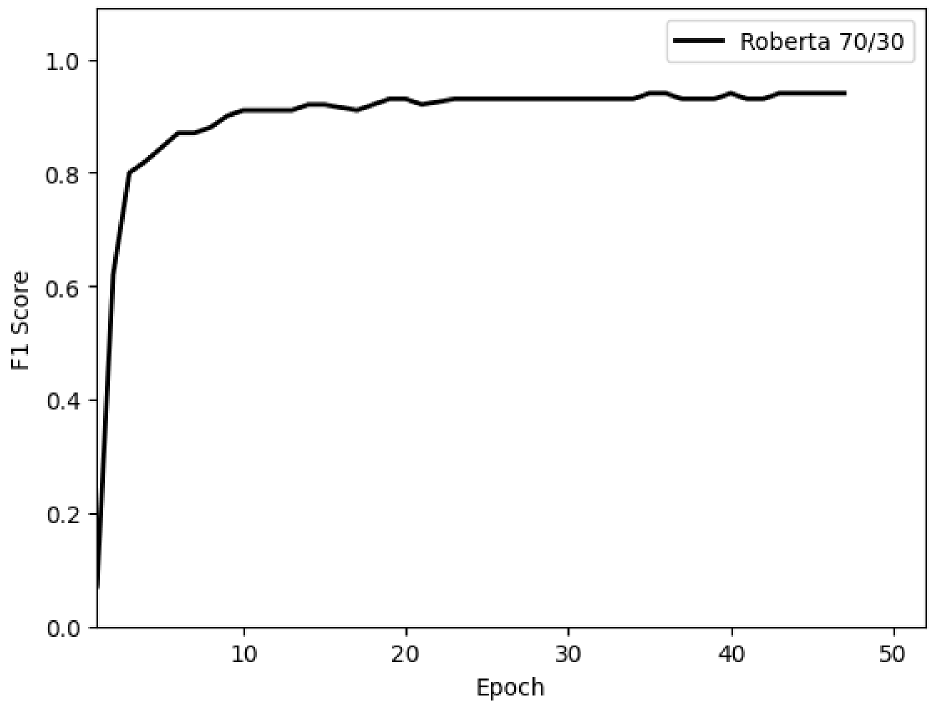


Figure 6-3: F1 score of Resume NER using RoBERTa Model

The graph shows F1 score vs epoch of Resume NER training where the F1 score rapidly increases during the first few epochs after which it stabilizes and delivers the highest F1 score of 0.94.

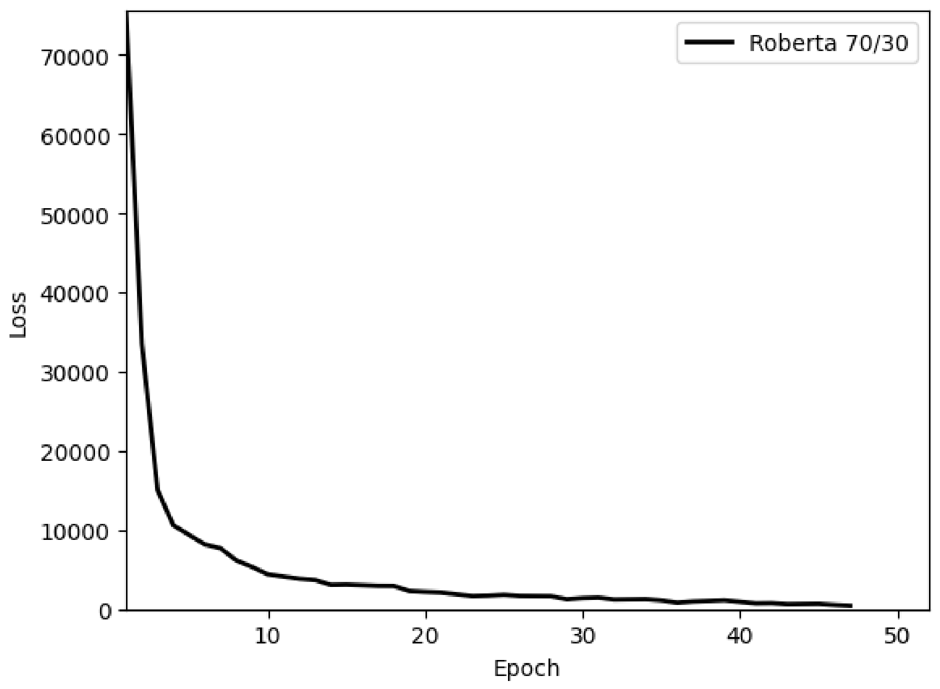


Figure 6-4: Loss of Resume NER using RoBERTa Model

The loss vs epoch graph illustrates rapid reduction of loss in the first few epoch resulting in the minimum loss 210.57.

6.3 Job Description NER using RoBERTa Model

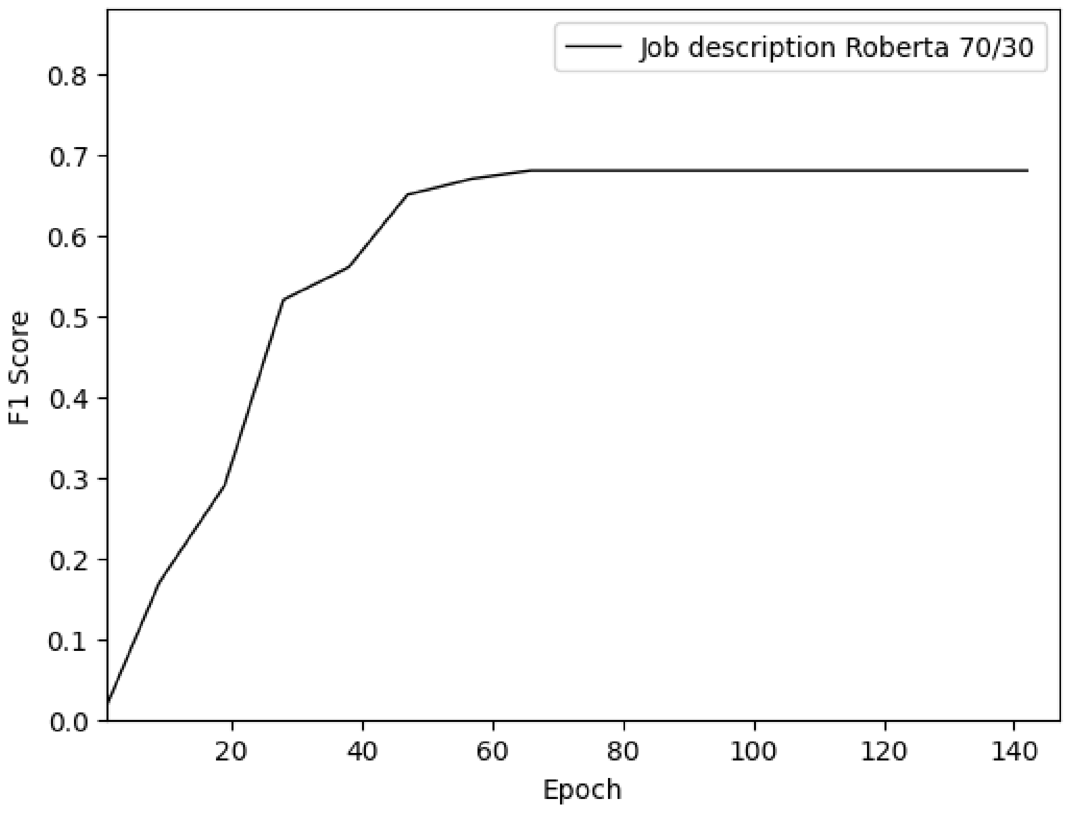


Figure 6-5: F1 score of Job Description NER using RoBERTa Model

The F1 score vs epoch graph of Job Description NER model can be seen in the graph which shows a steady increase in the F1 score over 40 epoch after which it stabilizes and produces a maximum F1 score of 0.91.

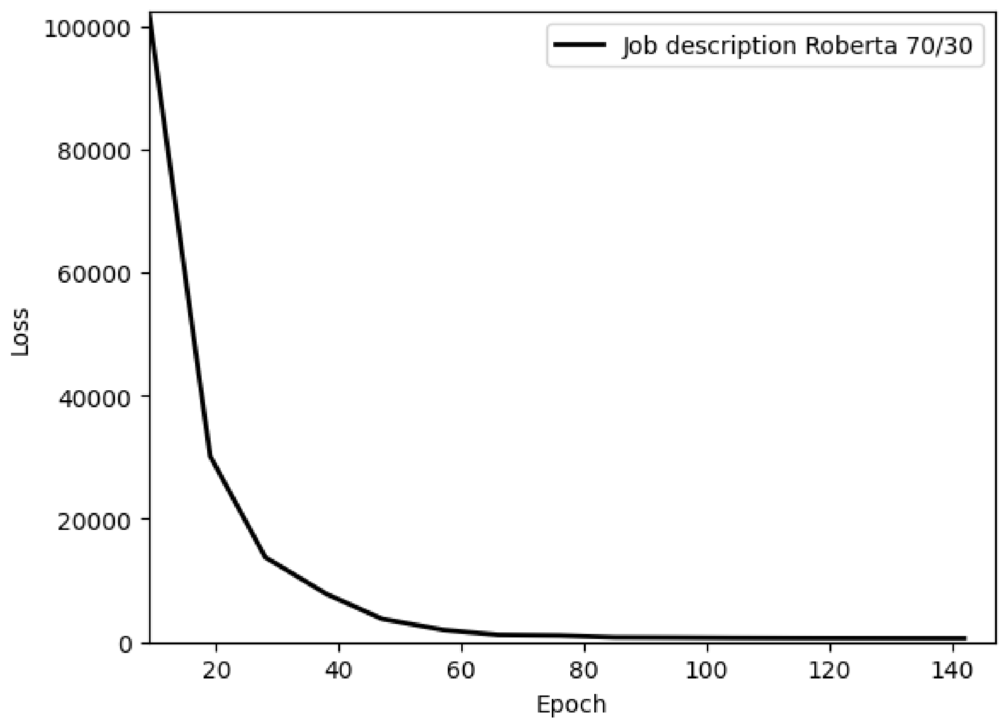


Figure 6-6: Loss of Job Description NER using RoBERTa Model

The graph represents loss vs epoch of Job Description NER which depicts the gradual reduction of loss over the first 4o iterations after which the graphs smooths out to give minimum loss of 0.11.

6.4 Web Interface

The web interface of our project is crafted with Django to serve as the strong backbone of our application. If the user is not registered, when registering, users give their username, email address, and password.

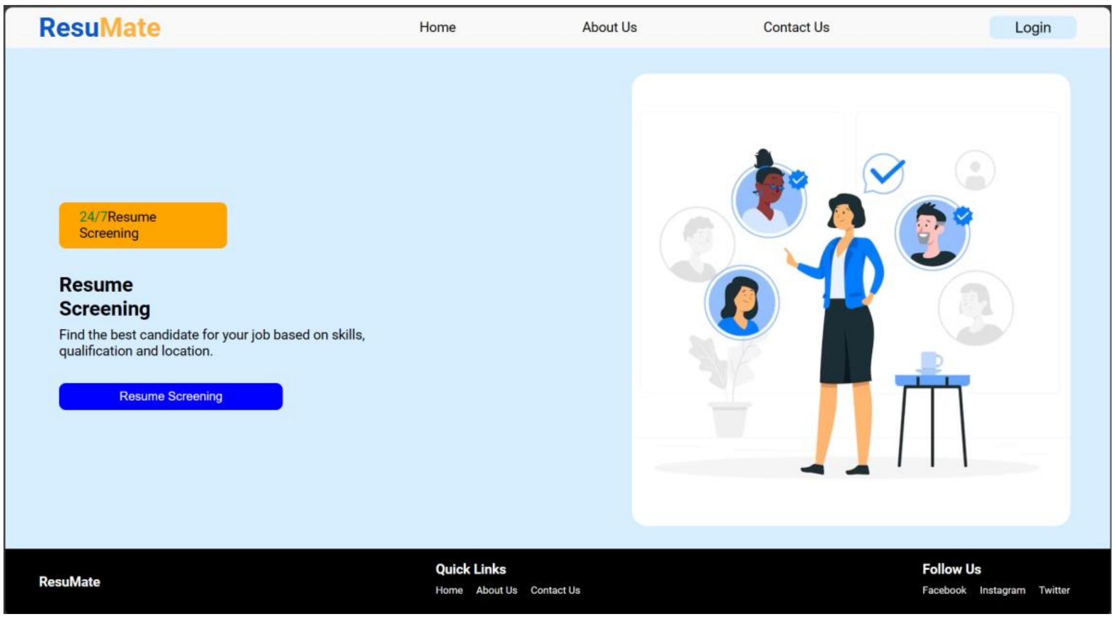


Figure 6-7: Homepage UI

This is the homepage of the system. The user can log in to the system using the top right button present in navigation bar.

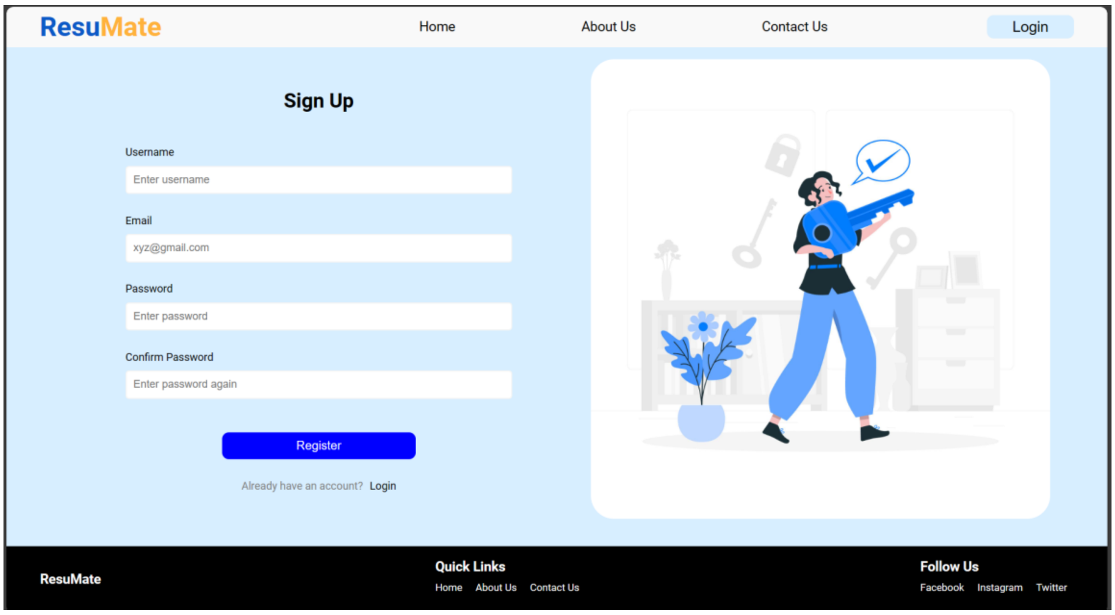


Figure 6-8: Signup UI

If the user is not registered, when registering, users give their username, email address, and password. The server side validates the password, and the passwords are automatically hashed by before being stored in the database. A new user is created upon successful validation.

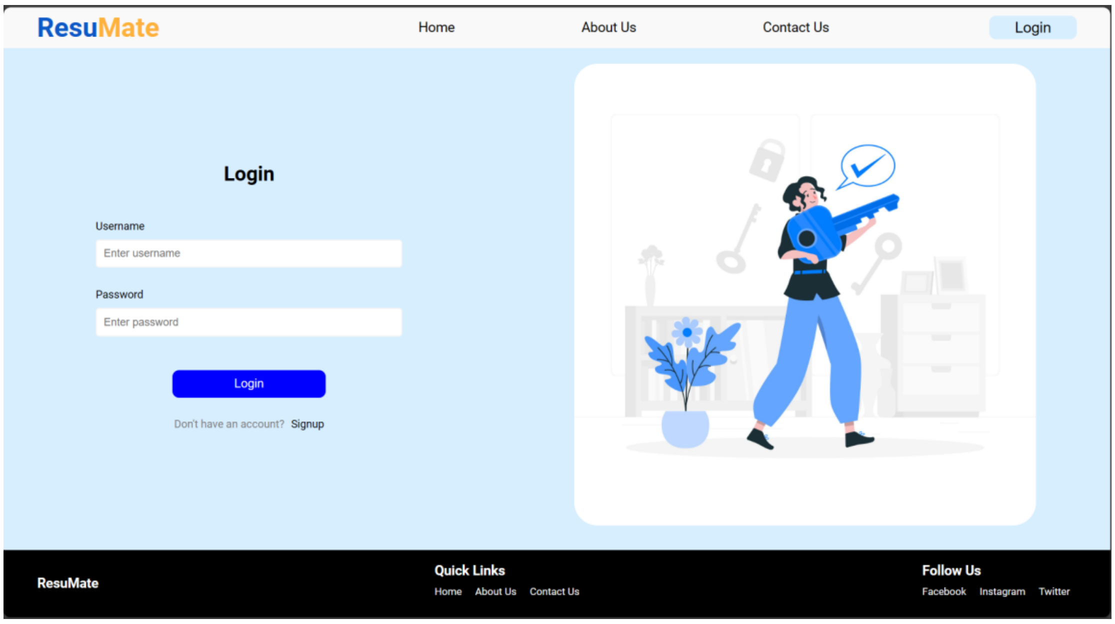
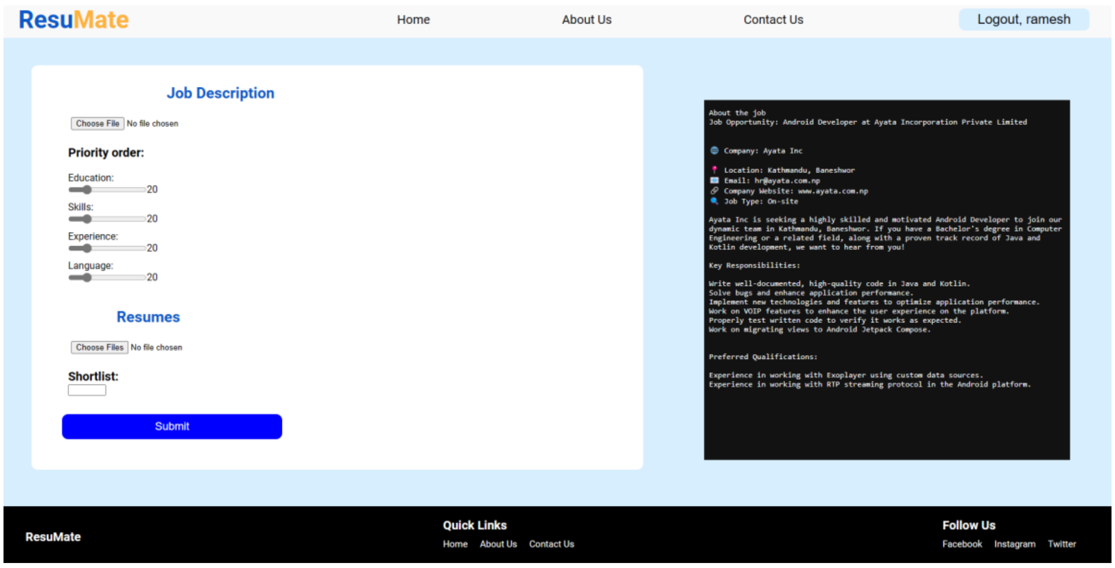


Figure 6-9: Login UI

After logging in, users will be taken to the upload page, where they can submit both their resumes and job description files.

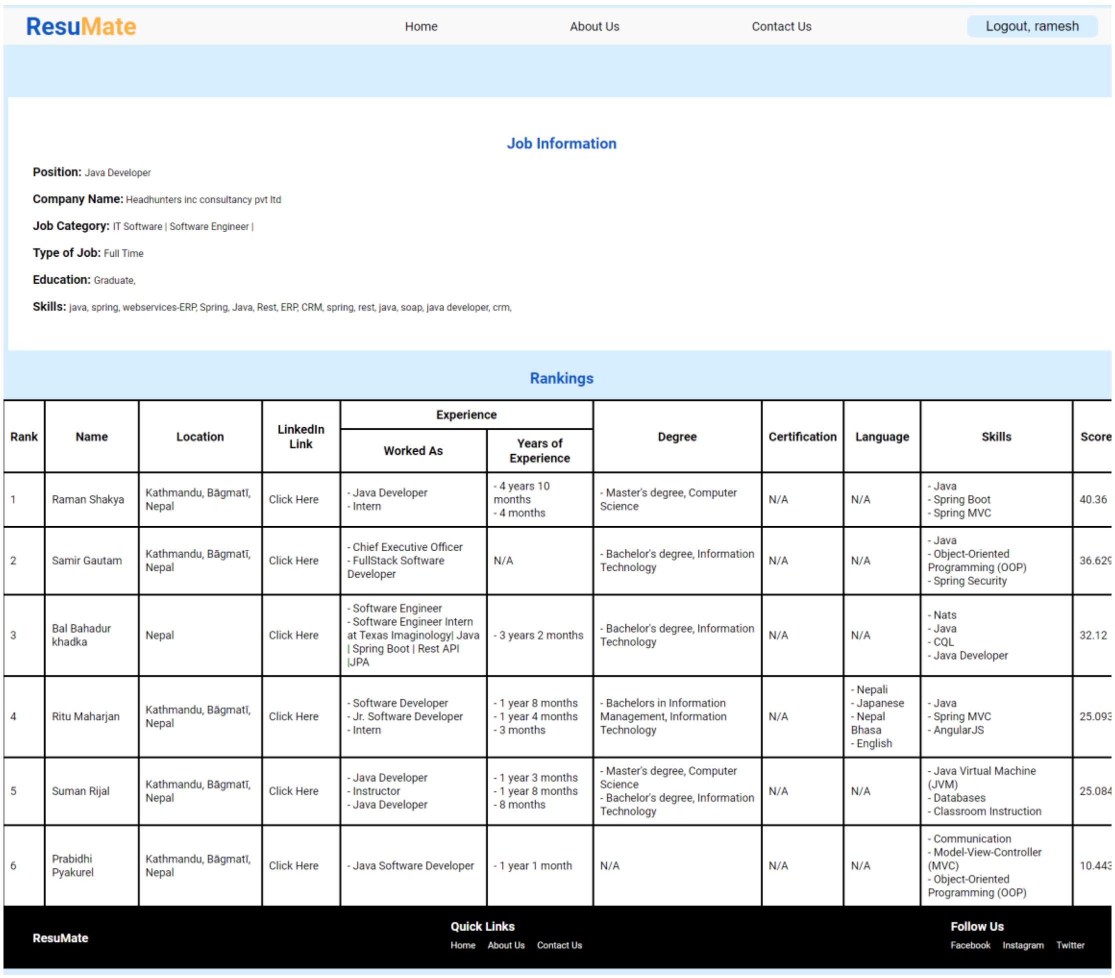


```
About the job
Job Opportunity: Android Developer at Ayata Incorporation Private Limited


🌐 Company: Ayata Inc

📍 Location: Kathmandu, Baneshwor
📧 Email: hr@ayata.com.np
🔗 Company Website: www.ayata.com.np
🔍 Job Type: On-site

Ayata Inc is seeking a highly skilled and motivated Android Developer to join our
dynamic team in Kathmandu, Baneshwor. If you have a Bachelor's degree in Computer
Engineering or a related field, along with a proven track record of Java and
Kotlin development, we want to hear from you!

Key Responsibilities:

Write well-documented, high-quality code in Java and Kotlin.
Solve bugs and enhance application performance.
Implement new technologies and features to optimize application performance.
Work on VOIP features to enhance the user experience on the platform.
Properly test written code to verify it works as expected.
Work on migrating views to Android Jetpack Compose.


Preferred Qualifications:

Experience in working with Exoplayer using custom data sources.
Experience in working with RTP streaming protocol in the Android platform.
```

Figure 6-10: Upload File UI

Information about job descriptions is intended to be stored in the Job description model. This model has fields for the job description file (job description), weights for skills (skills), education requirements (education), language criteria (language), and experience expectations (experience). Each instance of this model is linked to a particular user.

The resume files are stored by the Resume model. Every Resume instance, like the Job Description model, is associated with a specific user and has fields for the resume file (resumes). Tables in the database are organized in accordance with the specified models. The models and the underlying relational database are guaranteed to work together seamlessly. The text from resume and job description objects are then extracted and our customized NER model is loaded to recognize the required entities.



Job Information

Position: Java Developer

Company Name: Headhunters inc consultancy pvt ltd

Job Category: IT Software | Software Engineer |

Type of Job: Full Time

Education: Graduate,

Skills: java, spring, webservices-ERP, Spring, Java, Rest, ERP, CRM, spring, rest, java, soap, java developer, crm,

Rankings

Rank	Name	Location	LinkedIn Link	Experience		Degree	Certification	Language	Skills	Score
				Worked As	Years of Experience					
1	Raman Shakya	Kathmandu, Bāgmatī, Nepal	Click Here	- Java Developer - Intern	- 4 years 10 months - 4 months	- Master's degree, Computer Science	N/A	N/A	- Java - Spring Boot - Spring MVC	40.36
2	Samir Gautam	Kathmandu, Bāgmatī, Nepal	Click Here	- Chief Executive Officer - FullStack Software Developer	N/A	- Bachelor's degree, Information Technology	N/A	N/A	- Java - Object-Oriented Programming (OOP) - Spring Security	36.629
3	Bal Bahadur khadka	Nepal	Click Here	- Software Engineer - Software Engineer Intern at Texas Imaginology| Java | Spring Boot | Rest API |JPA	- 3 years 2 months	- Bachelor's degree, Information Technology	N/A	N/A	- Nats - Java - CQL - Java Developer	32.12
4	Ritu Maharjan	Kathmandu, Bāgmatī, Nepal	Click Here	- Software Developer - Jr. Software Developer - Intern	- 1 year 8 months - 1 year 4 months - 3 months	- Bachelors in Information Management, Information Technology	- Nepali - Japanese - Nepal Bhasa - English	N/A	- Java - Spring MVC - AngularJS	25.093
5	Suman Rijal	Kathmandu, Bāgmatī, Nepal	Click Here	- Java Developer - Instructor - Java Developer	- 1 year 3 months - 1 year 8 months - 8 months	- Master's degree, Computer Science - Bachelor's degree, Information Technology	N/A	N/A	- Java Virtual Machine (JVM) - Databases - Classroom Instruction	25.084
6	Prabidhi Pyakurel	Kathmandu, Bāgmatī, Nepal	Click Here	- Java Software Developer	- 1 year 1 month	N/A	N/A	N/A	- Communication - Model-View-Controller (MVC) - Object-Oriented Programming (OOP)	10.443

Figure 6-11: Result Table of Ranked List

Once the resume files are compared with the job description, each resume receives a score according to which the resumes are ranked. Other features like skills, experience, education, location link to their linkedin profile, certifications and language is also displayed in the table.

Table 6-1: Comparison between Resumate Ranked list and Human Ranked list

Top Ranks	**ResuMate Result**	**Human Result**
Rank 1	Profile 2	Profile 2
Rank 2	Profile 1	Profile 3
Rank 3	Profile 3	Profile 1
Rank 4	Profile 5	Profile 5
Rank 5	Profile 4	Profile 4

In our experiment, we collected 5 resumes for a job position. We had our system rank the resumes according to its similarity with the job description. Then we had a specialist in that filed rank the same resumes for the same job description. When we compared both the results, our system proved to be reliable in rank order. The error is due to the use of pre-trained word embedding that had not been trained specifically on the IT domain.

7. FUTURE ENHANCEMENT

The most significant area for enhancement in this project is its expansion into other domains like healthcare, ecommerce, etc.

The NER model used in this system extracts focuses such as name, skill, location, experience, designation, years of experience, degree, etc. In future, it can be used to extract more fine entities like publications, references, research, etc. which can enhance the information retrieval.

Since a pre-trained word embedding is used, it was not trained on domain specific areas which limits the accuracy of matching. So, a custom word embedding can be trained on a large-corpus of industry-specific texts.

Moreover, our system examines texts in order to determine rankings. Additional modalities, like pictures, graphics, or links to portfolios, can be added to further improve it.

Support for multiple languages can also be extended so that the resume ranking system can be effectively used by businesses with a range of linguistic needs.

8. CONCLUSION

The Resume Ranking system is a streamlined approach toward the recruitment process. The project can be divided into two parts: screening and ranking. The screening process involved file-specific text extraction and recognition of resume-specific named entities. The customized named entity recognition (NER) models trained using RoBERTa model proved to be very effective with an f1 score of 0.96 for resume and 0.68 for job description.

The ranking process involved converting the features into vectors using FastText for Word Embedding, and comparing their similarity using cosine similarity. Hence, the candidates' resumes were ranked based on their similarity with the job description. The rankings generated by the system yielded commendable results when compared to the HR ranking.

9. APPENDICES

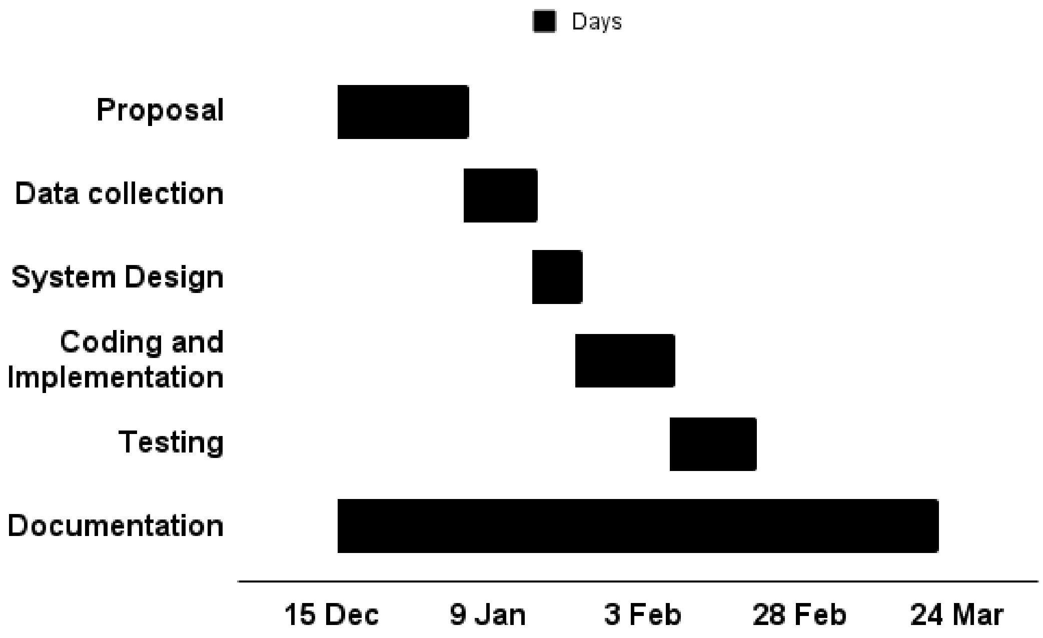
Appendix A: Project Budget

Table 9-1: Project Budget

Item	**Price**
Printing	Rs 2,000
Domain	Rs 500 per year
Hosting	Rs 2,000 per year
Grand Total	Rs. 4,500

Appendix B: Project schedule

Table 9-2: Gantt Chart



Appendix C: Code Snippets

Web Scraping

requests.get(URL): Sends an HTTP GET request to the provided URL and returns a response.

BeautifulSoup(html, 'lxml'): Parses HTML content using lxml parser and creates a BeautifulSoup object, facilitating easy navigation and extraction of data.

soup.find_all(tag, class_): Finds all HTML elements with a specified tag and class in the BeautifulSoup object.

soup.find(tag,class_): Searches for the first occurrence of an HTML element with the specified tag class in the BeautifulSoup object.

urljoin(base, link): Combines a base URL and a relative link to form an absolute URL.

Tika

parser.from_file(pdf_path): Uses Tika's parser to extract text content from a PDF file.

parsed_pdf.get('content', ''): Retrieves the content of the parsed PDF, with an empty string as a default if the content is not available.

Training customized NER in SpaCy

python -m spacy train config.cfg --output ./output --paths.train ./train.spacy --paths.dev ./dev.spacy: Trains a spaCy model using the configuration specified in "config.cfg," with training data from "train.spacy" and validation data from "dev.spacy".

References

[1] A. Kamil, M. Maree, M. Belkhatir and S. Alhashmi, "An Automatic Online Recruitment System Based on Exploiting Multiple Semantic Resources and Concept-Relatedness Measures," 2015.

[2] A. Tiwari, S. Vaghela and R. Nagar, "Applicant Tracking and Scoring System," *International Research Journal of Engineering and Technology (2019),* pp. 320-324, 2019.

[3] V. Bhatia, P. Rawat and A. Kumar, "End-to-end resume parsing and finding candidates for a job description using bert.," *arXiv preprint,* 2019.

[4] A. Jivtode, K. Jadhav and D. Kandhare, "Resume Analysis Using Machine Learning And Natural Language Processing".

[5] S. Nasr and O. German, "Resume searching to decide best candidate based on RELIEF method.," *Open Science Journal,* vol. 5, 2020.

[6] C. Daryani, G. S. Chhabra and H. Patel, "An Automated Resume Screening System Using Natural Language Processing And Similarity," pp. 99-103, 2020.

[7] A. A. Noor, M. Bakhtyar and B. A. Chandio, "Automatic cv ranking using document vector and word embedding.," *Pakistan Journal of Emerging Science and Technologies 2,* 2021.

[8] N. L. Guillarme and W. Thuiller, "TaxoNERD: deep nerural models for the recognition of taxonomic entites in the ecological and evolutionary literature," 2021.

[9] A. Najjar, B. Amro and M. Macedo, "An Intelligent Decision Support System For Recruitment: Resumes Screening and Applicants Ranking," *Informatica,* 2021.

[10] S. Naseer, M. Ghafoor, Sohaib, S. Khalid Alvi, A. Kiran, S. U. Rehman and G. a. C. J. a. J. P. Murtaza, "Named Entity Recognition (NER) in NLP Techniques, Tools Accuracy and Performance," 2022.

[11] D. Mule, S. Doke and S. Navale, "Resume Screening Using LSTM," *International Research Journal of Engineering and Technology (IRJET),* 2022.

[12] Tejaswini, Umadevi and S. M. Kadiwal, "Design and development of machine learning based resume ranking system," 2022.

[13] S. Racherla and P. Sripathi, "An Innovative Hashing Scheme and BiLSTM-based Dynamic Resume Ranking System," vol. 13, 2023.

[14] C. Berragan, A. Singleton, A. Calafiore and J. Morley, "Transformer based named entity recognition for place name extraction from unstructured text," *International Journal of Geographical Information Science,* vol. 37, 2023.

[15] A. Vaswani, A. N. Gomez, J. Uszkoreit, I. Polosukhin, L. Jones, L. Kaiser, N. Parmer and N. Shazeer, "Attention is All you Need," 2017.

[16] Y. Liu, M. Ott, N. Goyal and J. Du, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," 2019.

[17] J. D. Toutanova, M.-W. Chang, K. Lee and Kristina, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2019.