



CAPSTONE PROJECT REPORT

RESUME PARSING USING NATURAL LANGUAGE PROCESSING (SpaCy NER)

PREPARED BY

Siddhartha Patra





CONTENTS

Introduction	3
Project Background	4
Current State of Affairs: ATS System.....	4
Business Problem.....	6
What is Named Entity Recognition (NER)?.....	7
Why SpaCy Framework?	8
Data Pipeline.....	10
Data Collection and Preprocessing	11
Machine Learning Pipeline.....	13
Cloud-Based Model Deployment.....	16
Way Forward.....	19
Conclusion.....	21
References.....	22



INTRODUCTION

In the fast-paced and ever-evolving job market of today, the recruitment process has become increasingly intricate and demanding for recruiters and HR professionals. Countless resumes flood the recruitment pipeline, creating a tedious and time-consuming task to identify the most qualified candidates for a position. To combat this challenge, we embarked on a mission to create a resume parser application that harnesses the power of Natural Language Processing (NLP) and, more specifically, Spacy's Named Entity Recognition (NER) technology to autonomously extract essential entities such as name, location, experience, and skills from a resume.

This project report aims to take you through the intricate and elaborate development process of our innovative resume parser application, starting from the preliminary data preparation to the final model deployment. We delve into the highly technical aspects of implementing Spacy's NER technology and share our in-depth experiences of overcoming various challenges along the way.



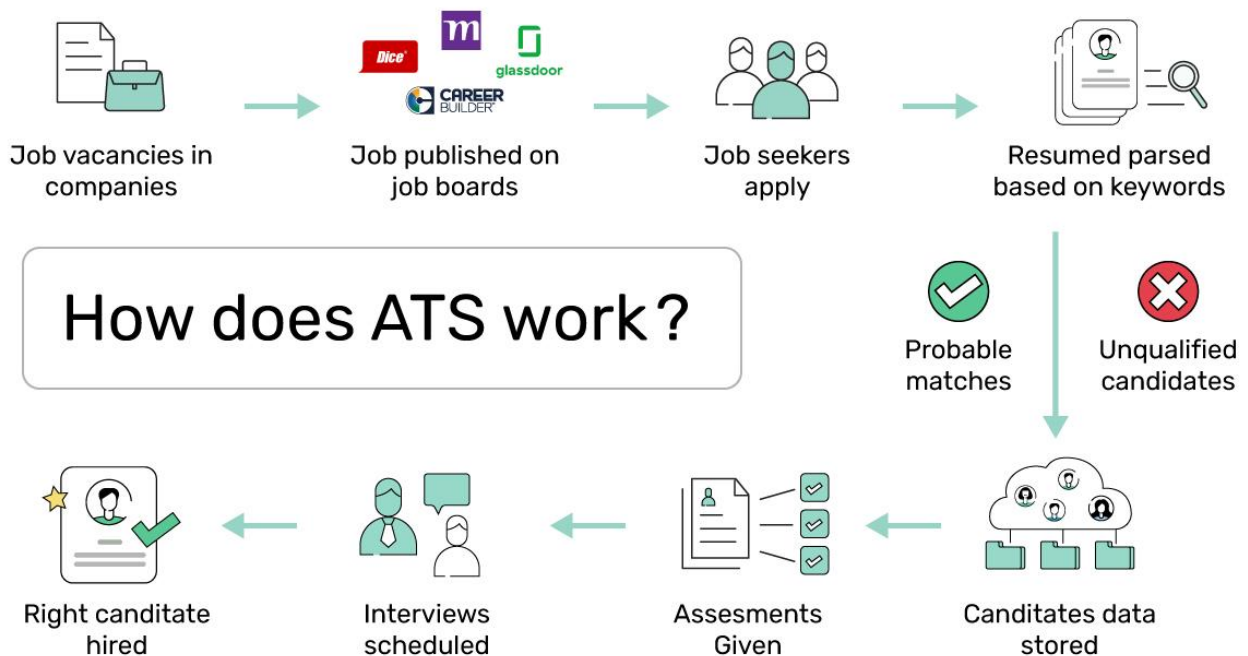
PROJECT BACKGROUND

Current State of Affairs: ATS System

ATS (Applicant Tracking System) is a widely used technology in the recruitment industry that helps automate and streamline the hiring process. It allows recruiters to manage job postings, receive and store resumes, and track candidate progress throughout the recruitment process. However, the current state of ATS systems has both advantages and limitations. Applicant Tracking System (ATS) typically works as follows:

1. **Resume Submission:** Job applicants typically submit their resumes and application materials through the company's career site or a job board. The ATS then parses the submitted documents to extract relevant information, such as the candidate's name, contact information, work history, and education.
2. **Resume Storage:** The ATS stores the candidate's resume and application materials in a searchable database. This allows recruiters and hiring managers to easily search and filter through candidate resumes based on specific keywords, skills, and qualifications.

3. **Candidate Screening:** The ATS helps to automate the initial screening of candidates by comparing their qualifications and experience with the job requirements. It can also score and rank candidates based on their fit for the job, allowing recruiters to focus on the most qualified candidates.
4. **Interview Scheduling:** The ATS can help to streamline the interview scheduling process by allowing recruiters to view candidate availability and schedule interviews directly from the system. Some ATS also offer tools to manage communication with candidates, such as automated email templates and candidate messaging.
5. **Candidate Management:** The ATS tracks the progress of candidates throughout the hiring process, from initial screening to final offer. It allows recruiters to easily view candidate profiles, notes, and feedback from interviews, and to collaborate with hiring managers to make informed hiring decisions.



Business Problem

ATS systems have their limitations. Many ATS systems rely heavily on keyword-based resume screening, which can result in false positives and negatives, and lack of advanced features like natural language processing, leading to potential bias and overlooking qualified candidates. They may also have limitations in customization and integration with other HR tools. We aim to improve the efficiency of the ATS system by using NLP to identify entities in a resume, which can help improve the ranking of candidate resumes beyond just keyword matching. This is due to the following reasons:



TIME & RESOURCE INTENSIVE: The recruitment process can be time-consuming and resource-intensive, particularly when it comes to manually screening and evaluating large numbers of resumes, potentially delaying time-to-hire and increasing cost of hiring.

HUMAN ERROR: Manual extraction of information from resumes, such as the candidate's education, experience, skills, etc. can be prone to human errors & biases, potentially leading to missed opportunities to identify top talent.



NEED FOR AUTOMATION: Through this project we aim to automate the process of resume screening and shortlisting using NLP techniques to extract relevant information from resumes and categorize them based on job requirements, with the goal of helping recruiters identify suitable candidates more quickly, cost-effectively, and free from human error.

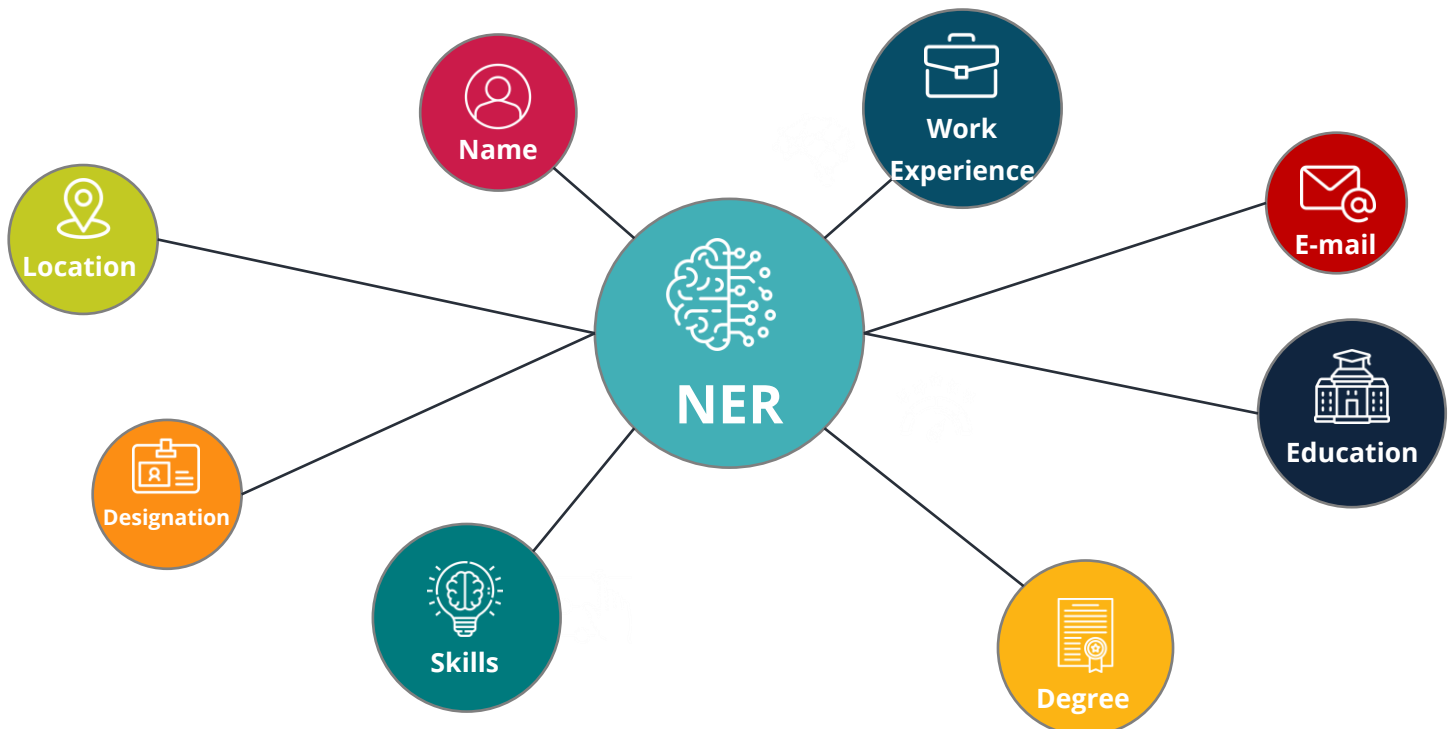
What is Named Entity Recognition (NER)?

Named Entity Recognition (NER) is a technique in natural language processing (NLP) that involves identifying and extracting named entities from unstructured or semi-structured text. Named entities refer to real-world objects or concepts that have a name, such as people, organizations, locations, dates, times, currencies etc.



In the context of resume parsing, NER can be used to automatically extract key information from resumes, such as the candidate's name, location, experience, skills, and more. This can save recruiters and hiring managers valuable time and improve the efficiency of the recruitment process.

Python's SpaCy NER module is a popular and powerful tool for implementing NER in NLP applications. The module can be trained on specific domains or datasets to improve its accuracy and performance. For our project I chose to train our entity parser on the following entities:



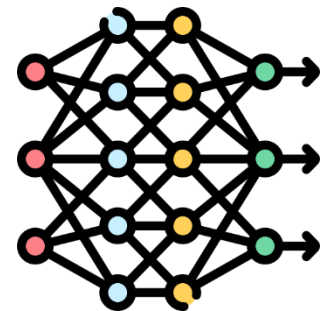
Why SpaCy Framework?

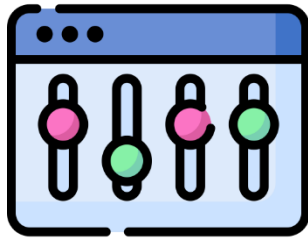
When it comes to building a resume parser application using NLP, Spacy is the go-to framework for many developers and data scientists. Some other popular NLP based framework such as, Stanford Core NLP and NLTK. Here are some reasons why we specifically chose Spacy:



1. **Speed and efficiency:** Spacy is optimized for speed and efficiency and is known to be faster than both Stanford Core NLP and NLTK. This is particularly important for large-scale NLP applications, such as building a resume parser application that needs to process a large number of resumes in a short amount of time.

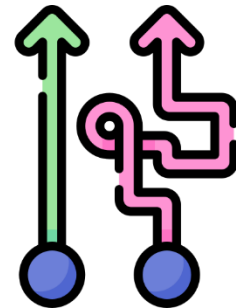
2. **Pre-trained models:** Spacy is better than Stanford Core NLP and NLTK because it comes with pre-trained models for a variety of languages, including English. These models are trained on large and diverse datasets and have high accuracy and performance out of the box. Developers can use these pre-trained models to perform various NLP tasks such as part-of-speech tagging, named entity recognition, and dependency parsing without having to train their own models from scratch, saving time and resources. In contrast, Stanford Core NLP and NLTK require additional training data to achieve good accuracy, which can be time-consuming and resource-intensive. Furthermore, Spacy's pre-trained models are regularly updated and improved, ensuring that developers have access to the latest advances in NLP technology.





3. **Customizability:** Spacy is more customizable than Stanford Core NLP and NLTK because it uses a pipeline architecture that allows developers to create custom processing components, or "pipes," and arrange them in any order they want. This makes it easier to add or remove specific processing steps, fine-tune the models for specific use cases, and experiment with different combinations of pipes. In contrast, Stanford Core NLP and NLTK have a more rigid architecture that limits customization options and requires developers to follow a specific processing flow. With Spacy, developers have more flexibility to tailor the parsing process to their needs and achieve better accuracy and performance.

4. **Ease of use:** Spacy has a simpler and more intuitive API than both Stanford Core NLP and NLTK, which makes it easier for developers to use and integrate into their applications. Spacy also provides extensive documentation and tutorials, making it easier for beginners to get started.



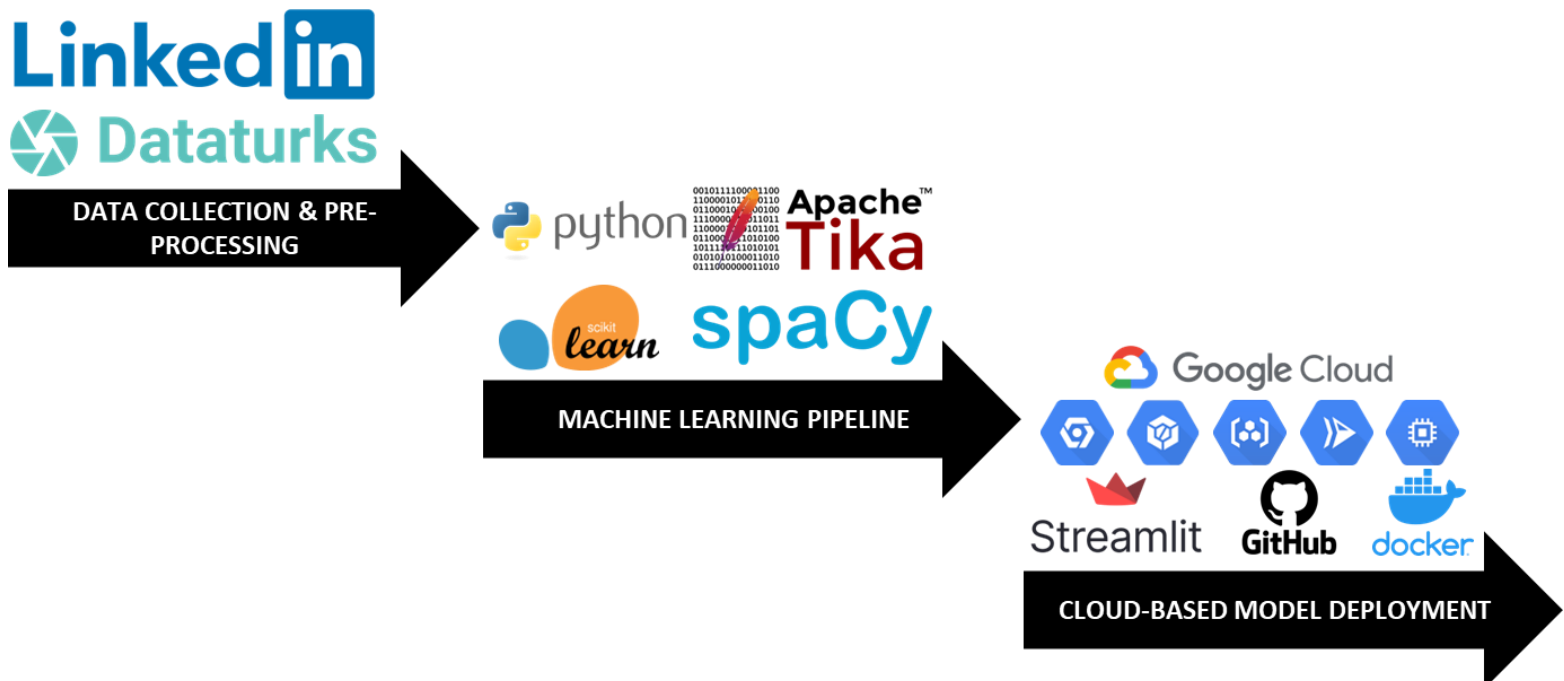
5. **Active community:** Spacy has a larger and more active community of developers and data scientists than both Stanford Core NLP and NLTK, which means there are more resources available for developers to learn from, get support, and find solutions to problems.

Overall, Spacy's speed, pre-trained models, customizability, ease of use, and active community make it a better choice than both Stanford Core NLP and NLTK for building a resume parser application using NLP.



DATA PIPELINE

The overall structure and data flow of our project is as follows:



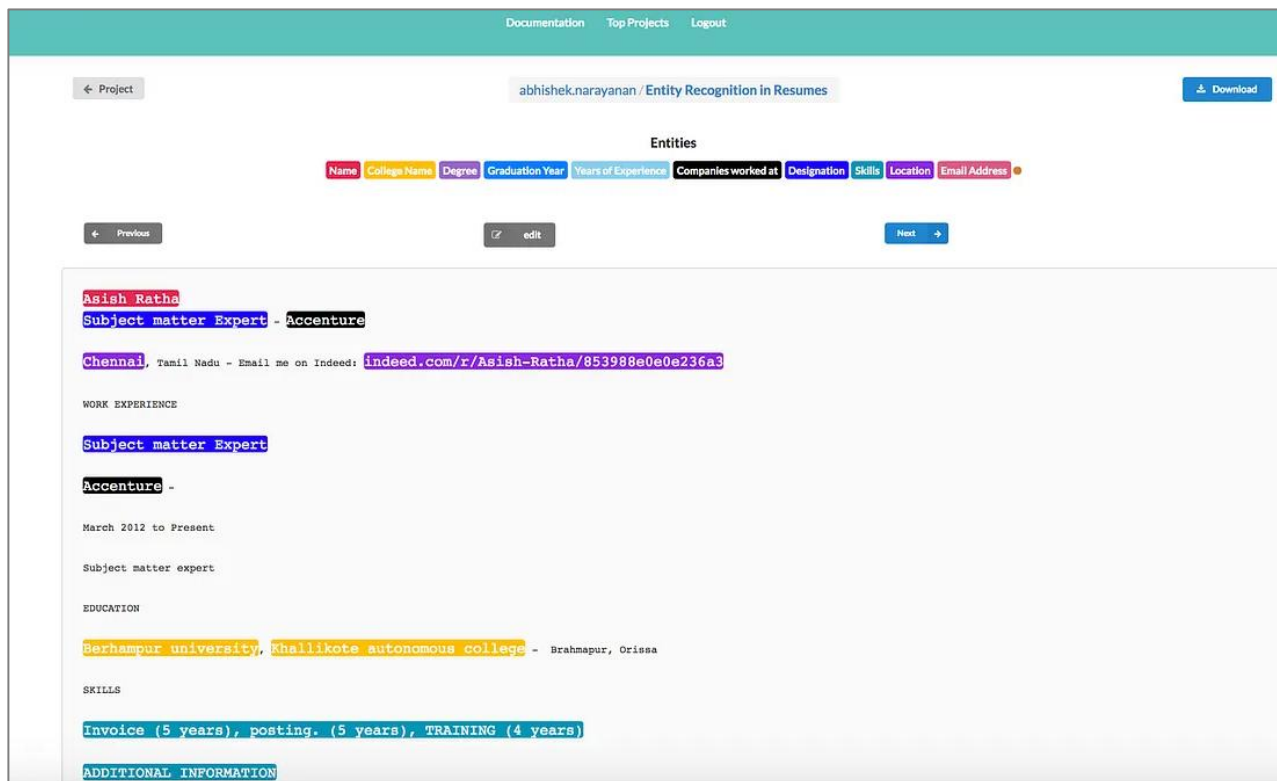
Data Collection and Preprocessing

1. Data Collection: 220+ Resumes from LinkedIn



The project collected 220+ resumes from LinkedIn, which is a widely used platform for job seekers and is considered an open source for resume data. The collected resumes covered various industries and domains, ensuring a diverse set of data for training the machine learning model.

2. Daturks Annotation Tool



The screenshot displays the Daturks Annotation Tool interface. At the top, there's a navigation bar with 'Documentation', 'Top Projects', and 'Logout'. Below this, a breadcrumb trail shows '← Project' followed by 'abhishek.narayanan / Entity Recognition in Resumes'. A 'Download' button is visible on the right. The main area features a horizontal bar with entity types: Name, College Name, Degree, Graduation Year, Years of Experience, Companies worked at, Designation, Skills, Location, and Email Address. Below this, a resume for 'Asish Ratha' is shown with various fields highlighted in different colors corresponding to the entity types. The resume includes sections for 'Subject matter Expert - Accenture', 'Chennai, Tamil Nadu - Email me on Indeed: indeed.com/r/Asish-Ratha/853988e0e0e236a3', 'WORK EXPERIENCE', 'Subject matter Expert', 'Accenture -', 'March 2012 to Present', 'Subject matter expert', 'EDUCATION', 'Berhampur university, Khallikote autonomous college - Brahmapur, Orissa', 'SKILLS', 'Invoice (5 years), posting. (5 years), TRAINING (4 years)', and 'ADDITIONAL INFORMATION'.

The resumes were then uploaded to a Data Annotation Tool, where pre-defined entities were highlighted manually. These entities could include named entities such as names, organizations, locations, skills, etc., which were relevant to the project's objective. After the annotation process, all the resumes were compiled into a JSON

file for further processing. The JSON file contained the resume data along with the annotated entities and their corresponding positions within the resume.

3. Post-Annotation Training Data

```
{
  "content": "Govardhana K\nSenior Software Engineer\n\nBengaluru, Karnataka, Karnataka - Email me on Indeed: in",
  "annotation": [
    {
      "label": "Companies worked at",
      "points": [
        {
          "start": 1749,
          "end": 1754,
          "text": "Oracle"
        }
      ]
    },
    {
      "label": "Companies worked at",
      "points": [
        {
          "start": 1696,
          "end": 1701,
          "text": "Oracle"
        }
      ]
    },
    {
      "label": "Companies worked at",
      "points": [
        {
          "start": 1417,
          "end": 1422,
          "text": "Oracle"
        }
      ]
    },
    {
      "label": "Skills",
      "points": [
        {
          "start": 1356,
          "end": 1792,
          "text": "Languages: Core Java, Gc"
        }
      ]
    }
  ]
}
```

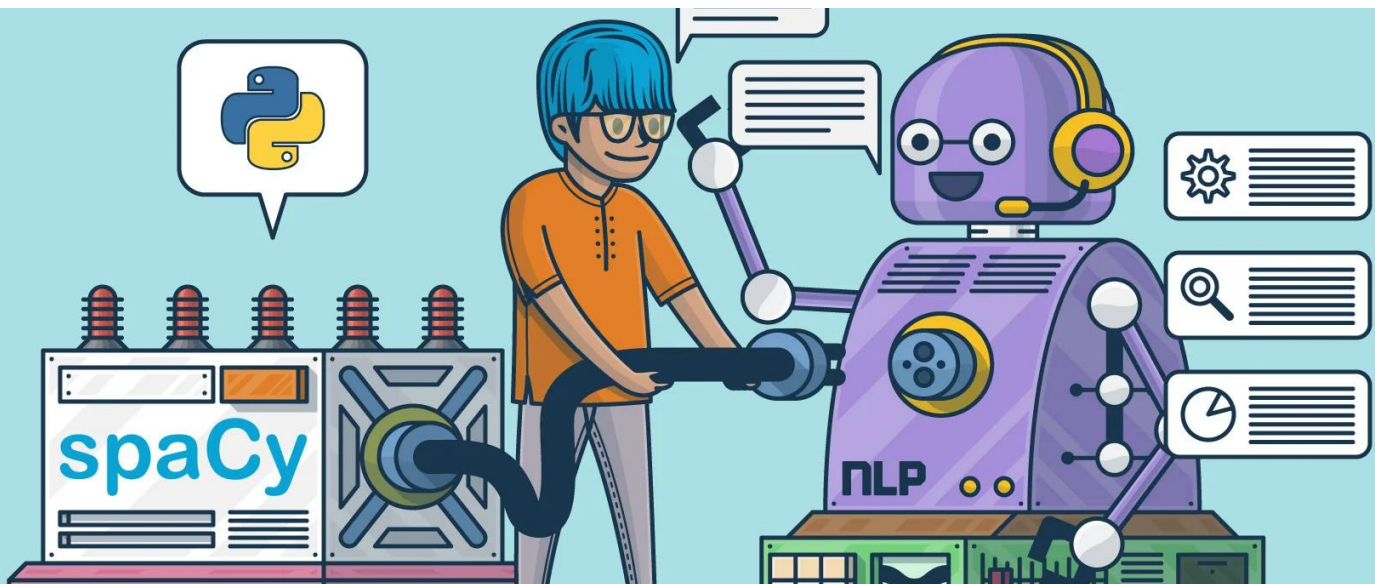
However, the JSON file format was not directly compatible with Spacy, a popular Natural Language Processing (NLP) library. Hence, the JSON file was further preprocessed to convert it into a Spacy-friendly format. The Spacy-friendly format included the text from the resumes followed by the entity positions and entity labels. This format was suitable for training the Spacy NLP model as it provided the necessary information about the entities and their locations in the resume text.

4. SpaCy Input Format

```
1 # SpaCy training data format
2 TRAIN_DATA = [
3     ("Walmart is a leading e-commerce company", {"entities": [(0, 7, "COMPANY")]})
4     ("I reached Chennai yesterday.", {"entities": [(19, 28, "LOCATION")]})
5     ("I recently ordered a book from Amazon", {"entities": [(24, 32, "COMPANY")]})
6     ("I was driving a BMW", {"entities": [(16, 19, "PRODUCT")]})
7     ("I ordered this from ShopClues", {"entities": [(20, 29, "COMPANY")]})
8     ("Fridge can be ordered in Amazon ", {"entities": [(0, 6, "PRODUCT")]})
9     ("I recently ordered from Swiggy", {"entities": [(24, 29, "COMPANY")]}])
10 ]
```

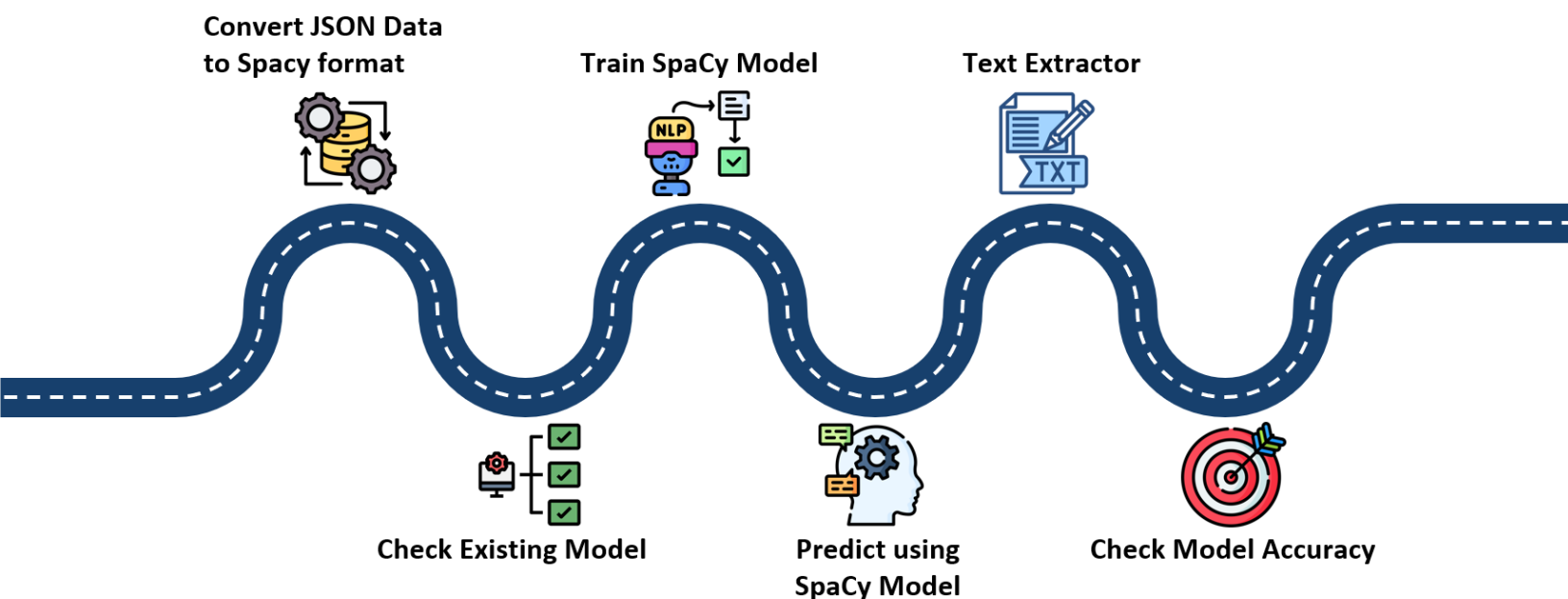
The converted Spacy-friendly data was then utilized for training the machine learning model using Spacy to perform named entity recognition and other Python libraries, such as scikit-learn classification for accuracy metrics and Apache Tika for text extraction using Optical Character Recognition (OCR).

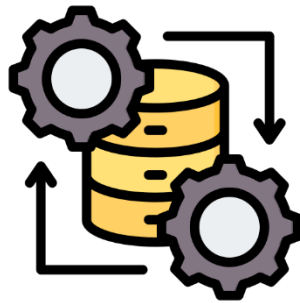
This detailed data collection and preprocessing process ensured that the project had a diverse and annotated dataset to train the Spacy model accurately, allowing for better performance and accuracy in identifying named entities in resumes during the resume screening and shortlisting process.



MACHINE LEARNING PIPELINE

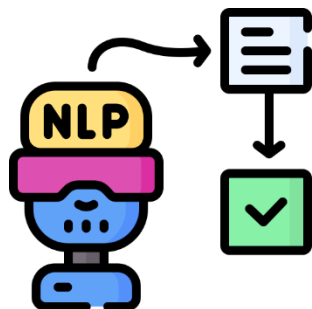
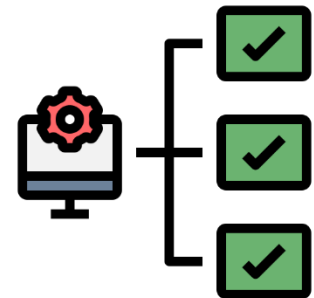
The roadmap for our machine learning pipeline is as follows:





1. **Convert JSON Data to Spacy Format:** The JSON data containing the resumes and their annotated entities were preprocessed to convert it into a Spacy-friendly format. This format included the text from the resumes, followed by the entity positions and entity labels, which were necessary for training the Spacy NLP model.

2. **Check Existing Model:** Before training a new Spacy model, the existing Spacy model was checked to see if it could be utilized for the named entity recognition task. This step involved evaluating the performance and accuracy of the pre-trained Spacy model on the specific resume dataset to determine if it could effectively identify the relevant named entities.



3. **Train SpaCy Model:** If the existing Spacy model did not provide satisfactory results, a new Spacy model was trained using the preprocessed resume data in Spacy format. This involved utilizing Spacy's machine learning capabilities to train the model using various NLP techniques, such as tokenization, part-of-speech tagging, and entity recognition, to identify and classify named entities in the resumes.

4. **Predict using SpaCy Model:** Once the Spacy model was trained, it was used for prediction on new, unseen resumes. The trained model was applied to the resume text to extract the named entities, such as names,



organizations, locations, skills, etc., based on the learned patterns and classifications from the training data.



5. **Text Extractor:** A text extractor module was utilized to extract the text from the resume files, such as the word, pdf etc., which were used as input for the Spacy model. This step involved text processing techniques, such as Optical Character Recognition (OCR) using the Apache Tika, to prepare the resume text for input into the Spacy model.

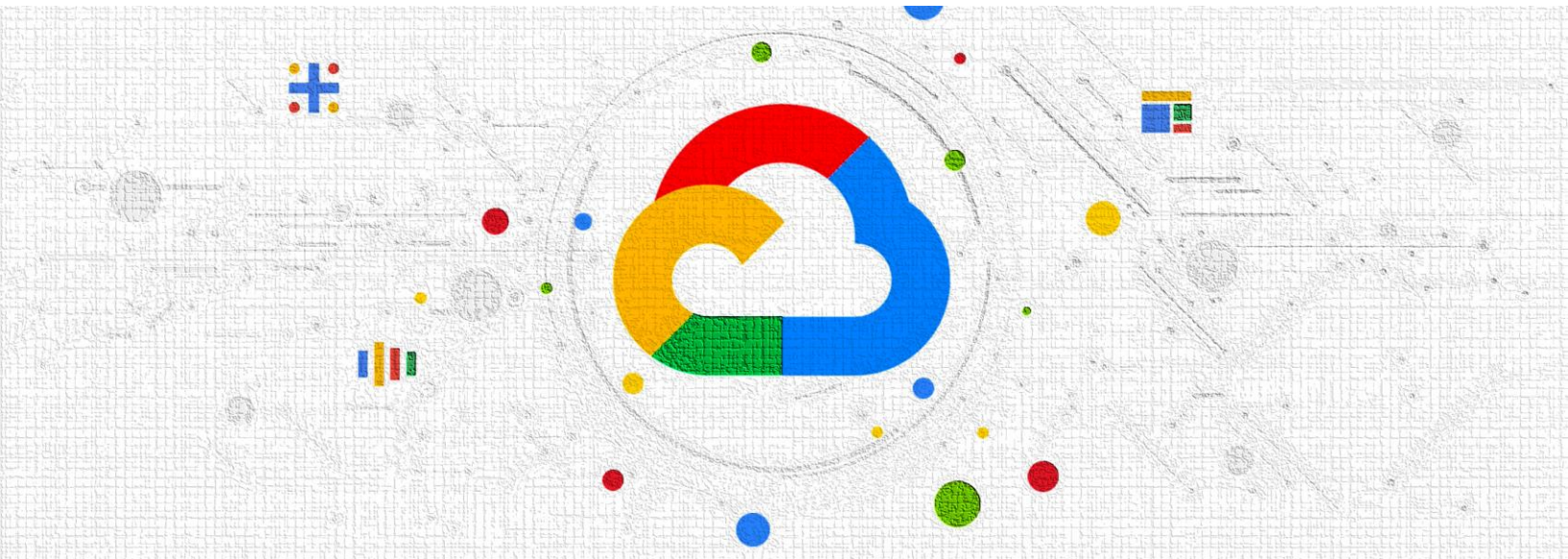
6. **Check Model Accuracy:** After the prediction step, the accuracy of the Spacy model was evaluated by identifying the named entities in the resumes generated in prediction.



This machine learning pipeline ensured that the resume data was effectively processed, trained, and evaluated using Spacy and other text processing techniques to accurately identify and classify the named entities, resulting in an efficient and effective resume screening and shortlisting process. The following sample output on unseen test data:

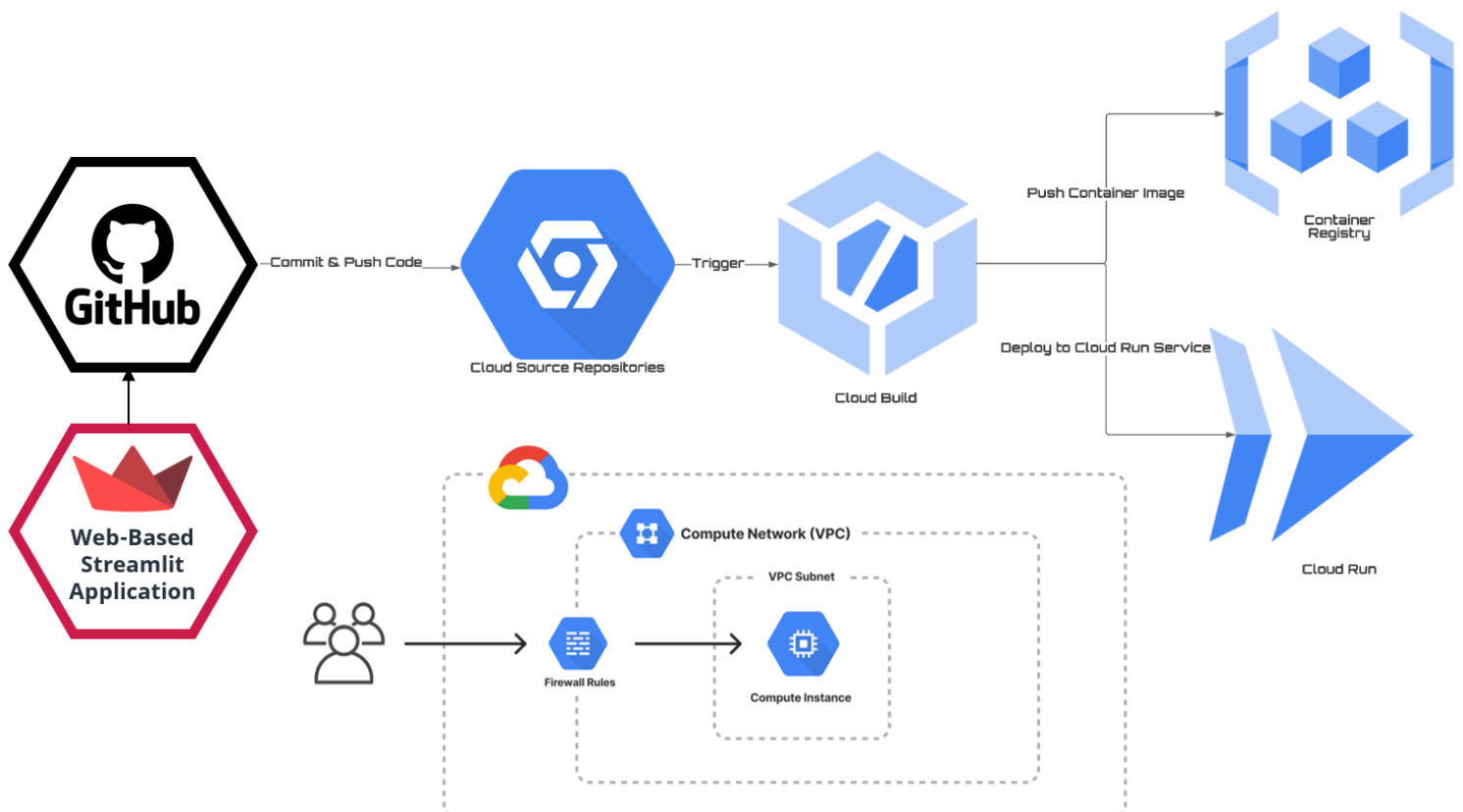
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  Python  +  -  [ ]  [X]  ...  ^  X

['Test Resume.pdf']
Processing F:\SEMESTER 4\CAPSTONE PROJECT\ACTUAL PROJECT\RESUME PARSER\OUTPUT\Test Resume.pdf
NAME                -Alice Clark
LOCATION              -Delhi
DESIGNATION          -Software Engineer
COMPANIES WORKED AT -Microsoft -
DEGREE               -Indian Institute of Technology - Mumbai
SKILLS               -Machine Learning, Natural Language Processing, and Big Data Handling
ON Professional Skills • Excellent analytical, problem solving, communication, knowledge transfer and interpersonal skills
s with ability to interact with individuals at all the levels • Quick learner and maintains cordial relationship with project
manager and team members and good performer both in team and independent job environments • Positive attitude towards superiors
& peers • Supervised junior developers throughout project lifecycle and provided technical assistance
PS F:\SEMESTER 4\CAPSTONE PROJECT\ACTUAL PROJECT\RESUME PARSER> [ ]
```



CLOUD-BASED MODEL DEPLOYMENT

The overall architecture of our Google Cloud based model deployment:



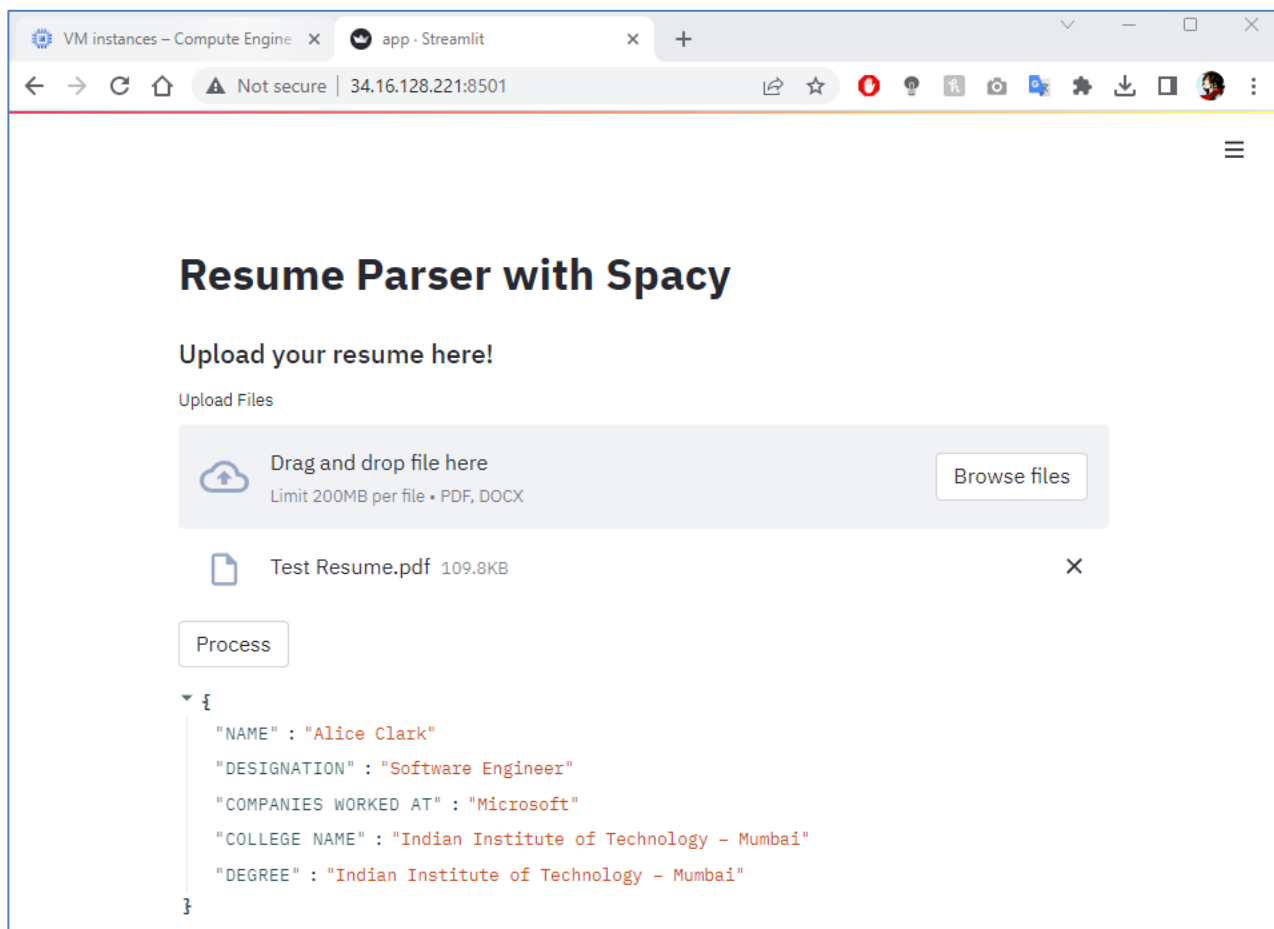
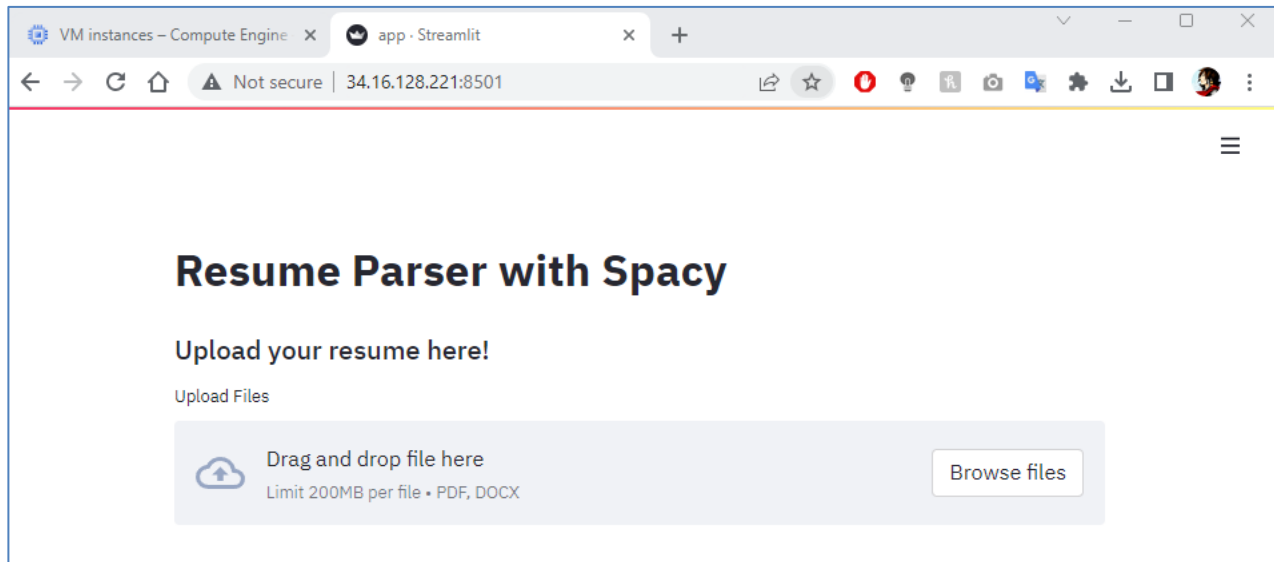
To deploy the project, I followed a cloud-based model deployment approach on Google Cloud Platform(GCP).

I took the following are steps to deploy the project:

1. **Streamlit Application:** I created a web-based Streamlit application that interacts with the backend Python project to provide a user interface for our model.
2. **Github Repository:** I pushed & committed the application code and backend project source code to the Github, so that it could be accessed and deployed from any machine.
3. **Google Cloud repository:** From Github, I pushed the application code and Backend Project source code to the Google Cloud repository.
4. **Cloud Build and Trigger:** Then I created a Cloud Build and defined the Trigger for the build. The trigger set up the environment and dependencies, such as Python, Pandas, SpaCy, Streamlit, and Apache Tika using a docker file.
5. **Container Registry:** On successful run of the trigger, it generated a unique image ID, which was stored in the Container Registry.
6. **Cloud Run:** From Container Registry, using the unique image ID, I deployed the Spacy Generated Project Model to the Cloud Run services using a virtual Machine.
7. **Virtual Machine:** I created a basic virtual machine instance, as I had limited free credit, and set up our own ports to ensure that the generated link did not collide with any other port in any system.

In summary, I used the cloud-based deployment approach to deploy the project, which ensured that our application was accessible from any machine and ran smoothly without any port collisions.

Screenshots of our resume parsing application running on Google Cloud Platform:





WAY FORWARD

Now that we have successfully established a working prediction model and an application that can parse key entities from a resume, these entities could be further utilized in the following ways:



FOR RECRUITERS:

This information can be organized and presented in a structured format that can be easily searched and filtered, making it easier for recruiters to find and shortlist ideal candidates.

With the help of NER, recruiters can quickly identify candidates with the required experience and skill set for a particular role. For example, if a recruiter is looking for a candidate with experience in Java programming, they can use NER to extract

the candidate's experience and skills related to Java programming, making it easier to determine if the candidate is a good fit for the job.

Furthermore, NER technology can also help to reduce unconscious bias in the recruitment process. By using an automated system to extract relevant information, recruiters can avoid being influenced by factors such as a candidate's name or location, and focus solely on their skills and experience.

FOR JOB SEEKERS:

Named Entity Recognition (NER) technology can also be highly beneficial for job seekers in their quest for better job recommendations. By using NER for resume parsing, job seekers can ensure that their resumes are optimized with relevant information and structured in a way that enables an automated system to extract key entities accurately.

When a job seeker submits a resume that has been optimized with NER, it becomes easier for the recruitment system to match their skills and experience with relevant job openings. For instance, if a job seeker has experience in Python programming, NER can accurately extract this information and use it to match the job seeker with job openings that require Python programming skills.

Moreover, NER technology can also help job seekers to identify skills and experiences that are in high demand in the job market. By analyzing the NER-extracted information from multiple job postings, job seekers can gain valuable insights into the skills and experiences that are in demand, enabling them to tailor their resumes to meet those requirements.

Finally, NER technology can also help job seekers to identify potential gaps in their skills and experiences. By analyzing the extracted entities from job postings, job seekers can identify the skills and experiences that they may be lacking and take steps to acquire those skills or gain relevant experience.





CONCLUSION

In conclusion, this project focused on building a resume parser application using Spacy's Named Entity Recognition (NER) capabilities. The goal was to identify key entities such as names, locations, experience, skills, etc. from a given resume. Spacy's pre-trained models and customizable pipeline architecture proved to be advantageous for this project, allowing for quick and accurate processing of resumes. The resulting application has potential use cases in HR departments and recruitment agencies for automating the screening process of job applicants. Further improvements could be made by incorporating additional training data and using a more powerful virtual machine for cloud run. Overall, this project demonstrates the power and potential of NLP in streamlining and improving various HR processes.



REFERENCES

- <https://analyticsindiamag.com/daturks/>
- <https://www.analyticsvidhya.com/blog/2021/06/nlp-application-named-entity-recognition-ner-in-python-with-spacy/>
- <https://www.turing.com/kb/a-comprehensive-guide-to-named-entity-recognition>
- <https://spacy.io/usage/spacy-101>
- <https://www.tutorialspoint.com/tika/index.htm>
- <https://towardsdatascience.com/a-review-of-named-entity-recognition-ner-using-automatic-summarization-of-resumes-5248a75de175>
- <https://github.com/DataTurks-Engg/Entity-Recognition-In-Resumes-SpaCy>
- <https://medium.com/@debanjanmahata85/natural-language-processing-with-spacy-36b90b9afa3d>
- <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04>
- <https://cloud.google.com/docs/get-started>

Special thanks to all the people who made and released these great resources for free:

- Word report template by [Used to Tech](#)
- Photographs by [Unsplash](#)
- Icons by [Flaticon](#)