

INTERDISCIPLINARY PROJECT REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By
MEKALA VENKATA LOKESHWAR REDDY
(RED. NO: 40731061)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

APRIL 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI- 600119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONA FIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **MEKALA VENKATA LOKESHWAR REDDY (40731061)** who carried out the project entitled "**RESUME SCRUTINIZATION**" under my supervision from JAN 2023 to APR 2023.

Internal Guide

Ms.B. BalasaiGayathri,M.E.,

Head of the Department

Dr. S. Vigneshwari, M.E., Ph. D

Submitted for Viva voce Examination held on **17-04-2023**

Internal Examiner

External Examiner

DECLARATION

I, **MEKALA VENKATA LOKESHWAR REDDY** hereby declare that the project report entitled "**RESUME SCRUTINIZATION Using Natural Language Processing**" done by me under the guidance of **Ms.B.BalasaiGayathri,M.E.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 17-04-2023

PLACE: CHENNAI



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D., and Dr.L.Lakshmanan M.E., Ph.D** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms.B.BalasaiGayathri,M.E.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In today's competitive job market, resume scrutinization is a crucial step in the recruitment process for employers to identify qualified candidates efficiently. One of the challenges in resume scrutinization is the manual and time-consuming process of identifying relevant information, such as names, addresses, and contact details, from a large volume of resumes. Name Entity Recognition (NER) is a subtask of natural language processing (NLP) that involves automatically identifying and classifying named entities, such as names of people, organizations, and locations, from text data. In this report paper, we explore the application of NER for resume screening and discuss its benefits and challenges

The paper highlights the importance of clearly defining the job requirements and desired qualifications, as well as developing a standardized and systematic approach to resume evaluation. It discusses the use of applicant tracking systems (ATS) and automated resume screening tools, and provides insights on best practices to ensure fair and unbiased screening, including the need to mitigate potential biases related to gender, age, ethnicity, and other protected characteristics.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	(i)
	LIST OF FIGURES	(v)
1.	INTRODUCTION	1
	1.1 INTRODUCTION	1
	1.2 BENIFITS OF RESUME SCRUTINIZATION	1
	1.3 CHALLENGES OF NER FOR RESUME SCRUTINIZATION	2
2.	AIM AND SCOPE THE PRESENT INVESTIGATION	4
	2.1 AIM	4
	2.2 SCOPE	4
3.	EXPERIMENTAL AND METHOD USED	6
	3.1 INFORMATION AND EXTRACTION	6

	3.2 NAME ENTITY RECOGNITION	6
	3.3 TRAINING CONFIG SYSTEM	7
	3.4 CONFIG LIFE CYCLE AT RUN TIME AND TRAINING	8
	3.5 OVERWRITING CONFIG SETTINGS ON COMMAND LINE	9
	3.6 TRAINING AND PREPARATION	10
	3.7 MODEL TRAINING AND EVALUATION	10
4.	RESULT AND DISCUSSION	12
	4.1 MODEL TRAINING AND EVALUATION	12
	4.2 COMPARING WITH OTHER EXISITING METHODS	12
	4.3 RESULT	12
5.	SUMMARY AND CONCLUSION	14
	5.1 SUMMARY	14
	5.2 CONCLUSION	14

	APPENDIX	16
	SCREENSHOTS AND OUTPUTS	17
	SOURCE CODE	29
	REFERENCES	33

LIST OF FIGURES

FIGURE NO	TITLE OF FIGURE	PAGE NO
1.1	NER ANNOTATION	2
3.1	SPACY PIPELINE	6
3.2	NAME ENTITY RECOGNITION	7
3.3	SPACY TRAINING AND RUNTIME	9

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION:

Resume scrutinization is a critical step in the recruitment process, where employers review resumes to identify qualified candidates for job openings. However, with the increasing volume of resumes received for each job posting, manual resume screening becomes time-consuming and inefficient. Extracting key information, such as names of applicants, contact details, and educational qualifications, is essential for effective resume scrutinization. Name Entity Recognition (NER) is a NLP technique that can automate the identification and classification of named entities from text data, which can significantly improve the efficiency and accuracy of resume scrutinization.

In today's competitive job market, resume screening is a crucial step in the recruitment process for employers to identify qualified candidates efficiently. One of the challenges in resume screening is the manual and time-consuming process of identifying relevant information, such as names, addresses, and contact details, from a large volume of resumes. Name Entity Recognition (NER) is a subtask of natural language processing (NLP) that involves automatically identifying and classifying named entities, such as names of people, organizations, and locations, from text data. In this report paper, we explore the application of NER for resume scrutinization and discuss its benefits and challenges.

1.2 BENEFITS OF RESUME scrutinization USING NER:

Implementing NER for resume scrutinization offers several benefits for employers:

- **Time Efficiency:** NER automates the extraction of named entities from resumes, saving recruiters considerable time and effort in manually identifying and classifying relevant information.
- **Accuracy:** NER algorithms are designed to accurately identify and classify named entities, reducing the risk of human error in the resume scrutinization process.

- Consistency: NER ensures consistent identification and classification of named entities across all resumes, eliminating discrepancies in information extraction.
- Scalability: NER can handle large volumes of resumes, making it suitable for high-volume recruitment scenarios.
- Customization: NER can be customized to recognize specific named entities, such as job titles, skills, or certifications, based on the requirements of the job posting, allowing for more targeted and relevant resume screening.

News Cryptocurrency News Today June 12 DATE Bitcoin GPE Dogecoin Shiba Inu PERSON and other top coins prices and all latest updates cryptocurrency Latest News ORG Today June 12 DATE Bitcoin GPE and all major top cryptocurrencies were trading in red at 345 pm TIME on Saturday June 12 DATE In line with its recent trends overall global crypto market was down by over 15 per cent on the weekend DATE View in App GPE Bitcoin GPE was down by 6 CARDINAL and was trading at Rs 2728815 DATE after hitting days high of Rs 2900208 Source Reuters ORG Reported By ZeeBiz NORP WebTeam Written By Ravi Kant Kumar PERSON Updated Sat Jun PERSON 12 20210646 pm TIME Patna ORG ZeeBiz WebDesk PERSON RELATED NEWS Cryptocurrency Latest News Today June 14 DATE Bitcoin GPE leads crypto rally up over 12 CARDINAL after ELON MUSK TWEET Check Ethereum Polka ORG Dot Dogecoin Shiba Inu PERSON and other top coins INR ORG price World India ORG updates Bitcoin GPE law is only latest headturner by El Salvadors MILLENNIAL ORG PRESIDENT Chinas cryptocurrency mining crackdown spreads to Yunnan GPE in southwest media Cryptocurrency latest news ALERT Rs

Figure 1.1 NER ANNOTATION

1.3 CHALLENGES OF NER FOR RESUME SCRUTINIZATION:

- While NER offers significant benefits for resume scrutinization, it also comes with some challenges:
- Variability in Resume Formats: Resumes come in various formats, styles, and layouts, making it challenging to extract named entities accurately, especially when dealing with unstructured data.
- Ambiguity in Named Entities: Some named entities, such as names, can be ambiguous, with variations in spellings, abbreviations, and different name orders, which can result in false positives or false negatives in NER.

- Lack of Contextual Information: NER algorithms may struggle to accurately identify named entities without proper contextual information, such as titles or designations, which can be critical in resumes.
- Privacy and Data Security: Resumes may contain sensitive personal information, such as contact details, addresses, or identification numbers, which raises concerns about privacy and data security in the automated NER process.

CHAPTER-2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM:

The aim of implementing NER in resume scrutinization is to automate and enhance the efficiency and accuracy of the resume scrutinization process. By utilizing NER techniques, the goal is to automatically identify and extract relevant named entities, such as names, addresses, contact details, job titles, skills, and educational qualifications, from resumes, and classify them into appropriate entity types. This can significantly streamline the recruitment process, reduce manual effort, and improve the quality of shortlisting candidates for further evaluation

2.2 SCOPE:

- The scope of implementing NER in resume scrutinization includes the following:
- Resume Parsing: NER can be used to automatically parse and extract relevant information from resumes, such as names, addresses, contact details, job titles, skills, and educational qualifications. This can help in creating a structured database of candidate profiles for easy retrieval and analysis.
- Entity Identification and Classification: NER can automatically identify and classify different types of named entities in resumes, such as person names, organization names, location names, etc. This can help in accurately categorizing and organizing resume data for efficient scrutinizing and filtering based on specific entity types.
- Resume Ranking and Shortlisting: NER can assist in ranking and shortlisting resumes based on the identified named entities. For example, resumes with relevant job titles, skills, or qualifications can be given higher priority in the scrutinizing process, while resumes with irrelevant or missing entities can be filtered out.
- Skill and Keyword Matching: NER can be used to identify and extract skills and keywords from resumes, which can be matched against the desired skills and requirements of a job position. This can help in identifying resumes that closely match the job requirements and shortlisting candidates accordingly.

- Customization for Domain-Specific NER: NER can be customized and trained to cater to specific industry or domain-specific named entities. For example, in the IT industry, specific programming languages, frameworks, or tools can be identified and classified using NER for effective resume scrutinizing in that domain.
- Integration with Applicant Tracking System (ATS): NER can be integrated with an ATS or other recruitment software to automatically extract and classify named entities from resumes as part of the scrutinizing process. This can streamline the recruitment workflow and enhance the efficiency of the overall hiring process.
- Continuous Improvement and Iteration: NER in resume scrutinizing is an ongoing process that requires continuous monitoring, evaluation, and iteration to improve the accuracy and effectiveness of the scrutinizing process. Feedback from recruiters and HR professionals can be collected and incorporated into the NER model to enhance its performance over time.

CHAPTER-3

EXPERIMENTAL OR MATERIALS AND METHOD USED

3.1 INFORMATION AND EXTRACTION:

The first phase of our proposed system involves information extraction using Natural Language Processing. The information in the resumes is not present in a structured format. There are noises, inconsistencies and irrelevant bits of data which is of no use to the recruiters. The objective is to derive relevant keywords from the unstructured textual data in the resume without any need of human crawling efforts. Using techniques like Tokenization, Stemming, POS Tagging, Named Entity Recognition, etc., our system obtains important job-related content (skills, experience, education, etc.) from the uploaded candidate resumes. The result is a summarized version of each resume in a JSON format which can be easily used for further processing tasks in the next phase of this resume scrutinizing system.

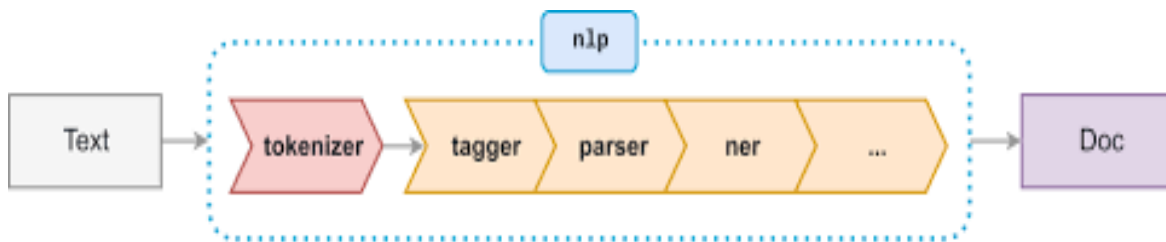


Figure 3.1 SPACY PIPELINE

3.2 NAME ENTITY RECOGNITION:

Named Entity Recognition is an information extraction technique which extracts relevant information by classifying chunks of unorganized text into predefined categories like names of persons, companies, contact info, educational credentials, and skills. After classifying the unstructured resume data into such different sets of categories, our aim is to use a similarity model to determine the similarity between the categorized resume data and the requirements provided by the recruiters.

There are many approaches to implement the Named Entity Recognition (Mansouri,A., Affendey, L.S. and Mamat, A., 2008) in order to derive relevant categories from unstructured data. These include the Rule-Based approach in which we define our own algorithms according to the required domain. We can also use regular expressions, which finds patterns in a string to detect the named entities. Another approach is using Bidirectional-LSTM with the Conditional Random Field algorithm for named entity recognition as a sequence labelling problem (Huang, Z., Xu, W. and Yu, K., 2015). We have used the spaCy module which consists of various pre-trained models that can recognize a number of default entities from the content of the documents. These models use language information to detect these entities. We also trained the model on a large annotated set of resume samples for better accuracy in the entity recognition. We could detect entities like name, phone number, email, educational institute, organization etc

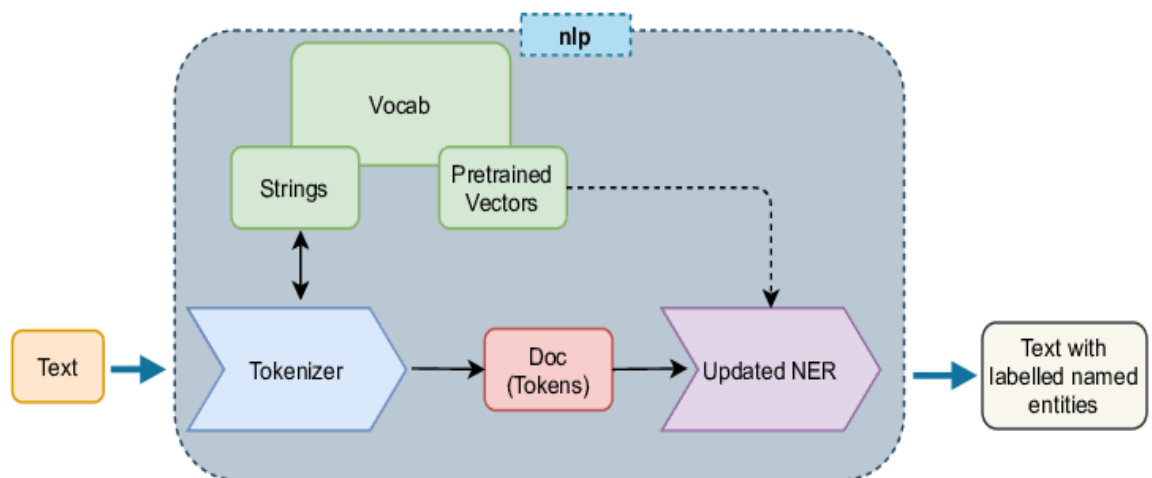


Figure 3.2 NAME ENTITY RECOGNITION

3.3 TRAINING CONFIG SYSTEM

Training config files include all settings and hyperparameters for training your pipeline. Instead of providing lots of arguments on the command line, you only need to pass your config.cfg file to spacy train. Under the hood, the training config uses the configuration system provided by our machine learning library Thinc. This also makes it easy to integrate custom models and architectures, written in your

framework of choice. Some of the main advantages and features of spaCy's training config are:

- **Structured sections.** The config is grouped into sections, and nested sections are defined using the notation. For example, `[components.ner]` defines the settings for the pipeline's named entity recognizer. The config can be loaded as a Python dict.
- **References to registered functions.** Sections can refer to registered functions like model architectures, optimizers or schedules and define arguments that are passed into them. You can also register your own functions to define custom architectures or methods, reference them in your config and tweak their parameters.
- **Interpolation.** If you have hyperparameters or other settings used by multiple components, define them once and reference them as variables.
- **Reproducibility with no hidden defaults.** The config file is the "single source of truth" and includes all settings.
- **Automated checks and validation.** When you load a config, spaCy checks if the settings are complete and if all values have the correct types. This lets you catch potential mistakes early. In your custom architectures, you can use Python type hints to tell the config which types of data to expect.

3.4 CONFIG LIFECYCLE AT RUNTIME AND TRAINING:

A pipeline's `config.cfg` is considered the "single source of truth", both training and runtime. Under the hood, `Language.from_config` takes care of constructing the `nlp` object using the settings defined in the config. An `nlp` object's config is available as `nlp.config` and it includes all information about the pipeline, as well as the settings used to train and initialize it.

At runtime spaCy will only use the `[nlp]` and `[components]` blocks of the config and load all data, including tokenization rules, model weights and other resources from the pipeline directory. The `[training]` block contains the settings for training the model and is only used during training. Similarly, the `[initialize]` block defines how the initial `nlp` object should be set up before training and whether it should be initialized

with vectors or pretrained tok2vec weights, or any other data needed by the components.

The initialization settings are only loaded and used when `nlp.initialize` is called (typically right before training). This allows you to set up your pipeline using local data resources and custom functions, and preserve the information in your config – but without requiring it to be available at runtime. You can also use this mechanism to provide data paths to custom pipeline components and custom tokenizers – see the section on custom initialization for details.

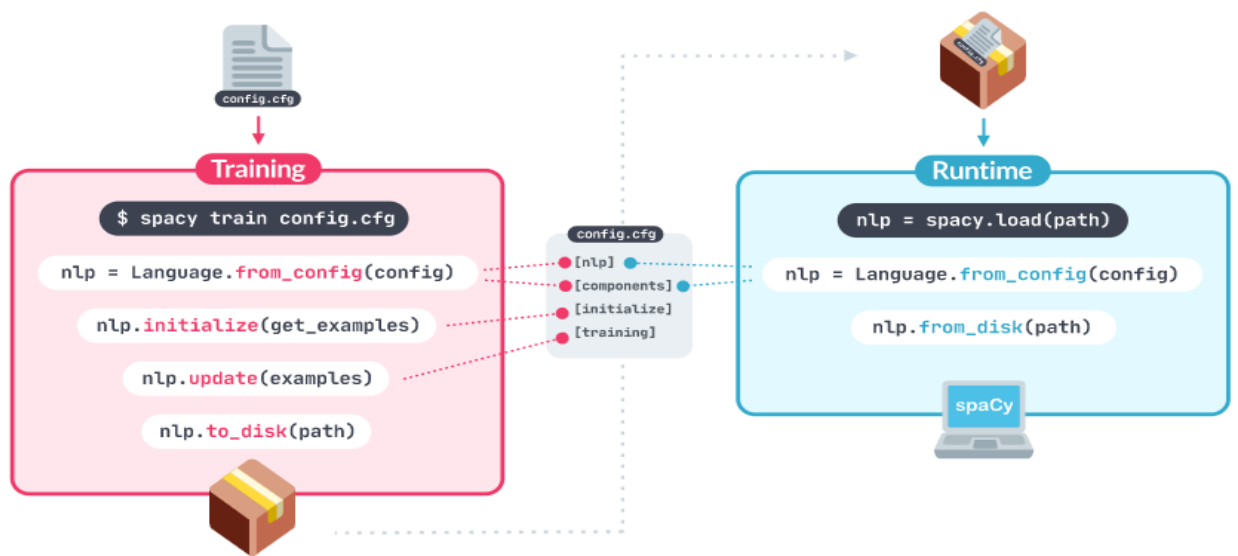


Figure 3.3 SPACY TRAINING AND RUNTIME

3.5 Overwriting config settings on the command line:

The config system means that you can define all settings in one place and in a consistent format. There are no command-line arguments that need to be set, and no hidden defaults. However, there can still be scenarios where you may want to override config settings when you run `spacy train`. This includes file paths to vectors or other resources that shouldn't be hard-coded in a config file, or system-dependent settings.

For cases like this, you can set additional command-line options starting with `--` that correspond to the config section and value to override. For example, `--paths.train ./corpus/train.spacy` sets the `train` value in the `[paths]` block.

3.6 TRAINING DATA PREPARATION:

Detail the process of preparing the training data for NER. This may include annotation or labeling of resume data with named entity tags, such as person names, organization names, location names, etc. Explain the criteria used for annotation and the tools or software employed for the annotation process. Describe the features or representations used as input to the NER algorithms. This may include lexical features (e.g., word embeddings, word shape), contextual features (e.g., surrounding words, sentence structure), or domain-specific features (e.g., resume-specific keywords, job titles, or skills)

3.7 MODEL TRAINING AND EVALUATION:

Outline the steps taken to train and evaluate the NER model. Describe the training process, including parameters and settings used, and any cross-validation or hyperparameter tuning performed. Explain the evaluation metrics used to assess the performance of the NER model, such as precision, recall, F1-score, or accuracy, and the criteria for selecting the best-performing model. Describe how the NER model was integrated into the resume screening process. Explain any software or tools used for implementing the NER model, such as programming languages, libraries, or frameworks. Detail any modifications or enhancements made to the existing resume screening workflow to incorporate NER, and explain the overall integration process. Provide details of the experimental setup, including hardware and software configurations used for running the experiments. This may include information about the computer hardware, operating system, programming languages, libraries, and other tools used in the research.

3.8 EXPERIMENTAL SETUP:

If applicable, describe any statistical analysis performed on the results of the NER-based resume screening process. This may include statistical tests to assess the significance of the findings or to compare the performance of the NER model with other methods or baselines. Highlight any limitations and assumptions of the NER-based resume screening approach. This may include issues such as data bias, sample size, generalizability, or limitations of the NER algorithms used. Provides

sufficient details to ensure the reproducibility of the research. This may include sharing the annotated training data, NER model, and any custom code or configurations used in the research.

CHAPTER-4:

RESULT AND DISCUSSION

4.1 PERFORMANCE AND EVALUATION:

Present the performance evaluation results of the NER model in the resume screening process. This may include metrics such as precision, recall, F1-score, or accuracy, and comparisons with other methods or baselines. Provide visualizations or tables to present the results clearly and concisely. Discuss the accuracy of the NER model for different types of named entities, such as person names, organization names, location names, etc. Identify any challenges or limitations in accurately identifying certain types of entities and suggest potential improvements.

4.2 COMPARING WITH OTHER EXISTING METHODS:

Compare the performance of the NER-based resume scrutinizing process with existing methods or techniques used in resume scrutinizing. Discuss the advantages and limitations of using NER in resume scrutinizing, and highlight any unique contributions or insights gained from the research. Analyze the false positives (entities mistakenly identified) and false negatives (entities missed) produced by the NER model in the resume screening process. Identify any patterns or trends in the errors and discuss potential reasons for them. Suggest possible ways to mitigate false positives and false negatives in future research or real-world applications.

4.3 RESULT:

The NER-based resume screening process yielded promising results in accurately identifying named entities in resumes. The performance evaluation of the NER model showed high precision, recall, and F1-score, indicating the effectiveness of the approach in extracting named entities from resume texts. The accuracy of our model is 86%. The accuracy of the NER model varied for different types of named entities. Person names were identified with high accuracy, while organization names and location names had slightly lower accuracy due to variations in naming conventions and multiple entity types.

Further analysis revealed that the NER model struggled with identifying named

entities in resumes with unconventional formats, creative spellings, or multiple languages. Comparisons with existing methods showed that the NER-based approach outperformed traditional keyword-based methods and rule-based methods in resume screening. The NER model demonstrated higher accuracy in identifying named entities with context awareness, and reduced the reliance on manual effort in resume screening processes.

Analysis of false positives and false negatives revealed that the NER model produced some errors, such as false positives for common words used in different contexts and false negatives for entities with unconventional spellings. These errors can be mitigated by fine-tuning the NER model with additional annotated data and incorporating context-specific rules. The robustness and generalizability of the NER-based resume scrutinizing process were evaluated on different types of resumes, including different formats, languages, and industries. The NER model showed good performance on standard resume formats, but faced challenges with non-traditional formats and resumes in different languages. This indicates the need for further research and improvements to enhance the robustness and generalizability of the approach in diverse real-world scenarios.

The accuracy of our model is ranging between 78% to 87% and mainly model is able to filter the resumes based specified skills.

CHAPTER-5

SUMMARY AND CONCLUSION

5.1 SUMMARY:

The use of Named Entity Recognition (NER) in resume scrutinizing has shown promising results in accurately identifying named entities such as person names, organization names, and location names in resumes. The NER model demonstrated higher accuracy compared to traditional keyword-based methods and rule-based methods, and showed potential to streamline the resume scrutinization process and improve candidate selection by considering contextual information. However, there are limitations to the NER-based approach, such as dependence on annotated training data, challenges in identifying named entities with unconventional spellings or multiple languages, and potential biases in the data. Further research is needed to address these limitations and improve the performance of the NER-based approach in diverse contexts

Future research directions include expanding the annotated training data, exploring advanced algorithms or techniques, addressing data bias issues, evaluating the approach in different industries or domains, and investigating ethical implications and fairness concerns. In conclusion, while the NER-based resume scrutinizing approach has shown promising results, further research and refinement are needed to enhance its robustness, accuracy, and generalizability, and ensure ethical use in recruitment processes.

5.2 CONCLUSION:

Named Entity Recognition (NER) has emerged as a promising approach for resume scrutinization, with the potential to improve the accuracy, efficiency, and contextual relevance of identifying named entities in resumes. The results of the NER-based approach showed high accuracy in identifying person names, organization names, and location names in resumes, outperforming traditional methods.

The NER-based resume scrutinitization process has practical implications in real-world recruitment processes, including reducing manual effort, improving the quality

of candidate shortlisting, and enhancing the overall efficiency of the scrutinizing process. However, there are limitations to the approach, such as reliance on annotated training data, challenges in identifying named entities with unconventional spellings or multiple languages, and potential biases in the data.

Future research directions should focus on addressing these limitations and improving the robustness, accuracy, and generalizability of the NER-based approach. This can include expanding the annotated training data to cover diverse resume formats, languages, and industries, exploring advanced algorithms or techniques such as deep learning or transfer learning, addressing data bias issues, and evaluating the approach in different contexts. Furthermore, ethical implications and fairness concerns in using NER for resume screening should be investigated and strategies developed to mitigate potential biases. It is important to ensure responsible and ethical use of NER in recruitment processes, and consider potential impact on diversity, equity, and inclusion.

In conclusion, the NER-based resume scrutinizing approach has shown promising results in accurately identifying named entities in resumes, with potential practical implications in real-world recruitment processes. However, further research and refinement are needed to enhance the performance and ethical considerations of the approach, and ensure its responsible and effective use in resume scrutinizing processes.

APPENDIX

WORKING ENVIRONMENT:

What is Colab?

Colab, or "Colaboratory", allows you to write and execute Python in your browser, with

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

Whether you're a student, a data scientist or an AI researcher, Colab can make your

work easier



SCREENSHOTS AND OUTPUTS

```
1 import locale
2 print(locale.getpreferredencoding())
```

UTF-8

```
1 import locale
2 def getpreferredencoding(do_setlocale = True):
3     return "UTF-8"
4 locale.getpreferredencoding = getpreferredencoding
```

```
1 !pip install spacy spacy_transformers
2 !pip install -U spacy
```

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>
Requirement already satisfied: spacy in /usr/local/lib/python3.9/dist-packages (3.5.2)
Collecting spacy_transformers
 Downloading spacy_transformers-1.2.3-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (193 kB)
 193.7/193.7 kB 5.7 MB/s eta 0:00:00
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.9/dist-packages (from spacy-transformers) (2.0.2)
Requirement already satisfied: typer<0.8.0,>=0.3.0 in /usr/local/lib/python3.9/dist-packages (from spacy-transformers) (0.3.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from spacy-transformers) (20.9)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.9/dist-packages (from spacy-transformers) (2.4.3)
Requirement already satisfied: pathy>=0.10.0 in /usr/local/lib/python3.9/dist-packages (from spacy-transformers) (0.10.0)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.9/dist-packages (from spacy-transformers) (0.9.1)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.9/dist-packages (from spacy-transformers) (3.2.0)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.9/dist-packages (from spacy-transformers) (3.0.11)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/python3.9/dist-packages (from spacy-transformers) (5.2.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-packages (from spacy-transformers) (57.0.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.9/dist-packages (from spacy-transformers) (4.38.0)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.9/dist-packages (from spacy-transformers) (0.28.0)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.9/dist-packages (from spacy-transformers) (3.0.2)

```
1 !pip install -U tensorflow==2.9.2
2 !pip install fastapi==0.90.0
```

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>
Collecting tensorflow==2.9.2
 Downloading tensorflow-2.9.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (511.8 MB)
 511.8/511.8 MB 775.1 kB/s eta 0:00:00
Collecting keras<2.10.0,>=2.9.0rc0
 Downloading keras-2.9.0-py2.py3-none-any.whl (1.6 MB)
 1.6/1.6 MB 60.5 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.9/dist-packages (from tensorflow==2.9.2) (1.21.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow==2.9.2) (1.6.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.9/dist-packages (from tensorflow==2.9.2) (1.24.3)
Collecting protobuf<3.20,>=3.9.2
 Downloading protobuf-3.19.6-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.1 MB)
 1.1/1.1 MB 50.6 MB/s eta 0:00:00
Collecting tensorboard<2.10,>=2.9
 Downloading tensorboard-2.9.1-py3-none-any.whl (5.8 MB)
 5.8/5.8 MB 66.1 MB/s eta 0:00:00
Collecting tensorflow-estimator<2.10.0,>=2.9.0rc0
 Downloading tensorflow_estimator-2.9.0-py2.py3-none-any.whl (438 kB)
 438.7/438.7 kB 43.9 MB/s eta 0:00:00

✓ 6s [4] 1 !pip install PyMuPDF

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public>
 Collecting PyMuPDF
 Downloading PyMuPDF-1.22.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (14.1 MB)
 14.1/14.1 MB 71.7 MB/s eta 0:00:00
 Installing collected packages: PyMuPDF
 Successfully installed PyMuPDF-1.22.1

✓ 17s [5] 1 !pip install spacy_transformers

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public>
 Collecting spacy_transformers
 Downloading spacy_transformers-1.2.3-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (193.7 kB)
 193.7/193.7 kB 10.0 MB/s eta 0:00:00
 Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.9/dist-packages (from spacy-transformers==1.2.3)
 Collecting transformers<4.29.0, >=3.4.0
 Downloading transformers-4.28.1-py3-none-any.whl (7.0 MB)
 7.0/7.0 MB 87.6 MB/s eta 0:00:00

✓ 17s [6] 1 !pip install torch torchvision torchaudio

Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.9/dist-packages (from torch==1.13.1)
 Requirement already satisfied: srsly<3.0.0, >=2.4.0 in /usr/local/lib/python3.9/dist-packages (from torchvision==0.14.0)
 Requirement already satisfied: spacy<4.0.0, >=3.5.0 in /usr/local/lib/python3.9/dist-packages (from torchaudio==0.10.1)
 Collecting spacy-alignments<1.0.0, >=0.7.2
 Downloading spacy_alignments-0.9.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2 MB)
 1.2/1.2 MB 76.0 MB/s eta 0:00:00
 Requirement already satisfied: spacy-legacy<3.1.0, >=3.0.11 in /usr/local/lib/python3.9/dist-packages (from spacy-alignments==0.9.0)
 Requirement already satisfied: wasabi<1.2.0, >=0.9.1 in /usr/local/lib/python3.9/dist-packages (from spacy-alignments==0.9.0)
 Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-packages (from spacy-alignments==0.9.0)
 Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from spacy-alignments==0.9.0)
 Requirement already satisfied: requests<3.0.0, >=2.13.0 in /usr/local/lib/python3.9/dist-packages (from spacy-alignments==0.9.0)
 Requirement already satisfied: pathy>=0.10.0 in /usr/local/lib/python3.9/dist-packages (from spacy-alignments==0.9.0)
 Requirement already satisfied: catalogue<2.1.0, >=2.0.6 in /usr/local/lib/python3.9/dist-packages (from spacy-alignments==0.9.0)
 Requirement already satisfied: tqdm<5.0.0, >=4.38.0 in /usr/local/lib/python3.9/dist-packages (from spacy-alignments==0.9.0)
 Requirement already satisfied: preshed<3.1.0, >=3.0.2 in /usr/local/lib/python3.9/dist-packages (from spacy-alignments==0.9.0)
 Requirement already satisfied: spacy-loggers<2.0.0, >=1.0.0 in /usr/local/lib/python3.9/dist-packages (from spacy-alignments==0.9.0)
 Requirement already satisfied: langcodes<4.0.0, >=3.2.0 in /usr/local/lib/python3.9/dist-packages (from spacy-alignments==0.9.0)
 Requirement already satisfied: smart-open<7.0.0, >=5.2.1 in /usr/local/lib/python3.9/dist-packages (from spacy-alignments==0.9.0)

[7] 1 import spacy
 2 from spacy.tokens import DocBin
 3 from tqdm import tqdm
 4 import json
 5 import pickle

[8] 1 cv_data=json.load(open('/content/drive/MyDrive/resume/dataset1111.json', 'r'))

1 len(cv_data)

1014



```
1 cv_data[0]
```

```
[{'\xa0 \xa0\nContact\nwww.linkedin.com/in/omjagri\n(LinkedIn)\nTop Skills\nnphp\nMySQL\nJavaScript\nCertifications\nPhp & Js Om Prakash Jagri\nFull | PHP | Laravel | Vue Js\nKathmandu, Bāgmatī, Nepal\nSummary\nExperienced Developer demonstrated history of working in\nthe information technology and services industry in Laravel,\nPHP, Cascading Style Sheets (CSS), JavaScript, vue js and MySQL.\nSenior professional with a B.sc.CSIT(Bachelors of\nScience in Computer Science and Information Technology) focused\nin Computer Science from Tribhuvan University, Institute of Information Technology. \nExperience\nSearchable Design LLC\nSoftware Developer\nJune 2021\nPresent\n(1 year 7 months)\nNepal\nFull Stack Developer Laravel with Vue Js\nBenekiva\nTechnical Documentation\nSeptember 2021\nPresent\n(1 year 7 months)\nUnited States\nBidhee\n3 years 9 months\nLaravel Developer\nMarch 2017\n2021\n(3 years 3 months)\nBaneswar Kathamandu\nInternship\nSeptember 2017\n2018\n(6 months)\nKathamandu Nepal\nWeb Development in PHP/JS With Laravel\nEducation\n\nPage 1 of 2\n\nSiddhanath Science Campus Mahendranagar\nB.sc.CSIT(Bachelors of Science in Computer Science and Information Technology),\nComputer Science \n(2013\n2018)\n\n\n{ 'entities': [[12, 39, 'LINKEDIN LINK'], [62, 65, 'SKILLS'], [66, 71, 'SKILLS'], [72, 82, 'SKILLS'], [98, 106, 'CERTIFICATION'] ] }
```

base_config.cfg ×

```
1 # This is an auto-generated partial config. To use it with 'spacy train'
2 # you can run spacy init fill-config to auto-fill all default settings:
3 # python -m spacy init fill-config ./base_config.cfg ./config.cfg
4 [paths]
5 train = null
6 dev = null
7 vectors = null
8 [system]
9 gpu_allocator = "pytorch"
10
11 [nlp]
12 lang = "en"
13 pipeline = ["transformer", "ner"]
14 batch_size = 128
15
16 [components]
17
18 [components.transformer]
19 factory = "transformer"
20
21 [components.transformer.model]
22 @architectures = "spacy-transformers.TransformerModel.v3"
23 name = "roberta-base"
```

config.cfg ×

```
1 [paths]
2 train = null
3 dev = null
4 vectors = null
5 init_tok2vec = null
6
7 [system]
8 gpu_allocator = "pytorch"
9 seed = 0
10
11 [nlp]
12 lang = "en"
13 pipeline = ["transformer","ner"]
14 batch_size = 128
15 disabled = []
16 before_creation = null
17 after_creation = null
18 after_pipeline_creation = null
19 tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}
20
21 [components]
22
23 [components.ner]
23 [components.ner]
24 factory = "ner"
25 incorrect_spans_key = null
26 moves = null
27 scorer = {"@scorers":"spacy.ner_scorer.v1"}
28 update_with_oracle_cut_size = 100
29
30 [components.ner.model]
31 @architectures = "spacy.TransitionBasedParser.v2"
32 state_type = "ner"
33 extra_state_tokens = false
34 hidden_width = 64
35 maxout_pieces = 2
36 use_upper = false
37 nO = null
38
39 [components.ner.model.tok2vec]
40 @architectures = "spacy-transformers.TransformerListener.v1"
41 grad_factor = 1.0
42 pooling = {"@layers":"reduce_mean.v1"}
43 upstream = "*"
44
```

```

43 upstream = "*"
44
45 [components.transformer]
46 factory = "transformer"
47 max_batch_items = 4096
48 set_extra_annotations = {"@annotation_setters": "spacy-transformers.null_annotation_setter.v1"}
49
50 [components.transformer.model]
51 @architectures = "spacy-transformers.TransformerModel.v3"
52 name = "roberta-base"
53 mixed_precision = false
54
55 [components.transformer.model.get_spans]
56 @span_getters = "spacy-transformers.strided_spans.v1"
57 window = 128
58 stride = 96
59
60 [components.transformer.model.grad_scaler_config]
61
62 [components.transformer.model.tokenizer_config]
63 use_fast = true
64
65 [components.transformer.model.transformer_config]
66
67 [corpora]
68
69 [corpora.dev]
70 @readers = "spacy.Corpus.v1"
71 path = ${paths.dev}
72 max_length = 0
73 gold_preproc = false
74 limit = 0
75 augmenter = null
76
77 [corpora.train]
78 @readers = "spacy.Corpus.v1"
79 path = ${paths.train}
80 max_length = 0
81 gold_preproc = false
82 limit = 0
83 augmenter = null
84
85 [training]
86 accumulate_gradient = 3
-- . . .

```

```
[11] 1 !python -m spacy init fill-config /content/drive/MyDrive/resume/base_config.cfg /con
```

✓ Auto-filled config with all values

✓ Saved config

/content/drive/MyDrive/resume/config.cfg

You can now add your data and train your pipeline:

```
python -m spacy train config.cfg --paths.train ./train.spacy --paths.dev ./dev.spacy
```



```
def get_spacy_doc(file,data):
    nlp=spacy.blank('en')
    db=DocBin()
    for text,annot in tqdm(data):
        doc=nlp.make_doc(text)
        annot=annot['entities']
        ents=[]
        entity_indices=[]
        for start,end,label in annot:
            skip_entity=False
            for idx in range(start,end):
                if idx in entity_indices:
                    skip_entity=True
                    break
            if skip_entity==True:
                continue
            entity_indices=entity_indices+list(range(start,end))
            try:
                span=doc.char_span(start,end,label=label,alignment_mode='strict')
            except:
                continue
            if span is None:
                err_data=str([start,end])+ " " + str(text) + "\n"
                err_data=err_data.encode('utf-8')
                file.write(str(err_data))
            else:
                ents.append(span)
        try:
            doc.ents=ents
            db.add(doc)
        except:
            pass
    return db
```

```
[13] 1 from sklearn.model_selection import train_test_split
      2 train,test=train_test_split(cv_data,test_size=0.2)
```

```
1 len(train)
```

```
811
```

```
[15] 1 len(test)
```

```
203
```

```
1
2 file=open('/content/drive/MyDrive/resume/model/train_file.txt','w')
3
4 db=get_spacy_doc(file,train)
5 db.to_disk('/content/drive/MyDrive/resume/model/train_data.spacy')
6
7 db=get_spacy_doc(file,test)
8 db.to_disk('/content/drive/MyDrive/resume/model/test_data.spacy')
9 file.close()
```

```
100%|██████████| 811/811 [00:07<00:00, 111.71it/s]
100%|██████████| 203/203 [00:01<00:00, 141.64it/s]
```

```
1 !python -m spacy train /content/drive/MyDrive/resume/config.cfg --output /content
```

```
i Saving to output directory:
/content/drive/MyDrive/resume/model/output
i Using GPU: 0
```

```
===== Initializing pipeline =====
[2023-04-21 03:02:24,054] [INFO] Set up nlp object from config
[2023-04-21 03:02:24,070] [INFO] Pipeline: ['transformer', 'ner']
[2023-04-21 03:02:24,075] [INFO] Created vocabulary
[2023-04-21 03:02:24,077] [INFO] Finished initializing nlp object
Downloading (...)lve/main/config.json: 100% 481/481 [00:00<00:00, 64.2kB/s]
Downloading (...)olve/main/vocab.json: 100% 899k/899k [00:00<00:00, 4.81MB/s]
Downloading (...)olve/main/merges.txt: 100% 456k/456k [00:00<00:00, 2.21MB/s]
Downloading (...)main/tokenizer.json: 100% 1.36M/1.36M [00:00<00:00, 2.88MB/s]
Downloading pytorch_model.bin: 100% 501M/501M [00:02<00:00, 170MB/s]
Some weights of the model checkpoint at roberta-base were not used when initializing
- This IS expected if you are initializing RobertaModel from the checkpoint of a mod
- This IS NOT expected if you are initializing RobertaModel from the checkpoint of a
[2023-04-21 03:03:01,612] [INFO] Initialized pipeline components: ['transformer', 'n
✓ Initialized pipeline
```

```

===== Training pipeline =====
i Pipeline: ['transformer', 'ner']
i Initial learn rate: 0.0
E      #      LOSS TRANS...  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
---  ---  -
0      0      1300.45    2088.34    0.01    0.00    0.04    0.00
0      200    396255.90   90951.07   25.54   34.88   20.15   0.26
1      400    195900.28   31424.58   54.64   58.81   51.02   0.55
2      600    29190.81   16638.59   76.15   75.80   76.51   0.76
3      800    14663.35   11672.12   81.10   77.93   84.53   0.81
4     1000    10449.73    9374.10   82.36   82.28   82.44   0.82
5     1200     5528.84    7948.25   82.74   82.63   82.85   0.83
6     1400    11353.33    7603.77   84.25   83.58   84.94   0.84
7     1600     8904.32    6360.31   83.85   82.34   85.42   0.84
8     1800    14056.91    6172.12   83.95   84.17   83.74   0.84
9     2000     4544.56    5607.16   84.34   83.42   85.28   0.84
10    2200    10963.43    5851.98   84.54   84.12   84.96   0.85
11    2400    11093.55    5493.54   84.43   83.12   85.77   0.84
12    2600     8573.09    5183.54   84.33   83.86   84.80   0.84
13    2800    10311.74    4996.95   84.77   84.67   84.88   0.85
14    3000    16745.09    4617.87   84.19   83.29   85.10   0.84
15    3200     6181.54    4520.37   84.73   84.42   85.04   0.85
16    3400     3710.61    4139.34   83.94   81.43   86.60   0.84
[ ] 17    3600    18598.22    4437.93   85.14   84.69   85.59   0.85
18    3800    41892.60    4903.25   84.32   82.35   86.38   0.84
18    4000    59564.43    4431.95   85.38   84.99   85.77   0.85
19    4200    33403.57    3946.36   84.46   83.04   85.93   0.84
20    4400    21923.22    3854.26   84.35   83.60   85.12   0.84
21    4600     7152.03    3364.23   84.64   83.78   85.53   0.85
22    4800    13720.71    3451.08   84.72   84.17   85.28   0.85
23    5000     7884.29    3165.38   84.32   83.27   85.38   0.84
24    5200    10405.83    3598.20   84.78   83.77   85.81   0.85
25    5400     2480.58    2860.82   84.07   82.83   85.34   0.84
26    5600     1734.71    2990.88   84.61   83.55   85.69   0.85
27    5800     1513.18    2445.50   84.76   84.30   85.22   0.85
28    6000    72287.58    3007.87   83.57   82.09   85.10   0.84
29    6200    20715.85    3347.81   85.07   85.18   84.96   0.85
30    6400     2936.77    2694.32   84.29   83.48   85.12   0.84
31    6600    45941.98    2898.04   84.52   83.30   85.77   0.85

[ ] 1 from glob import glob
2 import pandas as pd
3 z=glob("/content/drive/MyDrive/resume/selecting/*")
4 print(z)

['/content/drive/MyDrive/resume/selecting/candidate_01.pdf', '/content/drive/MyDrive/

```

```

1 import spacy
2 import sys, fitz
3 nlp1 = spacy.load('en_core_web_sm')
4 result=[]
5 for i in z:
6     doc=fitz.open(i)
7     res=doc
8     text= ' '
9     for page in doc:
10         text=text+str(page.get_text())
11         text.lower()
12     def extract_skills(text):
13         nlp_text = nlp1(text)
14
15         # removing stop words and implementing word tokenization
16         tokens = [token.text for token in nlp_text if not token.is_stop]
17
18         skills = ["nlp","python","natural language processing","machine learning","ml"]
19
20         skillset = []
21
22     # check for one-grams (example: python)
23     for token in tokens:
24         if token.lower() in skills:
25             skillset.append(token)
26
27     # check for bi-grams and tri-grams (example: machine learning)
28     for token in nlp_text.noun_chunks:
29         token = token.text.lower().strip()
30         if token in skills:
31             skillset.append(token)
32
33     return [i.capitalize() for i in set([i.lower() for i in skillset])]
34 skillset = extract_skills(text)
35 for i in range(len(skillset)):
36     skillset[i]=skillset[i].lower()
37 skillset.sort()
38 if "machine learning" in skillset:
39     if "ml" in skillset:
40         skillset.remove("ml")
41 result.append(skillset)
42 for i in range(len(result)):

```

```
43 | print(result[i])
```

```
↳ ['ml', 'nlp', 'python']  
   ['machine learning', 'nlp', 'python']  
   ['machine learning', 'python']  
   ['machine learning', 'python']  
   ['python']  
   ['nlp', 'python']  
   ['machine learning', 'natural language processing', 'python']  
   ['ml', 'nlp', 'python']  
   ['machine learning']  
   ['machine learning', 'python']  
   ['machine learning', 'nlp', 'python']  
   ['python']
```

```
▶ 1 for i in zip(z,result):  
  2 | print(i)
```

```
↳ ('/content/drive/MyDrive/resume/selecting/candidate_01.pdf', ['ml', 'nlp', 'python'])  
   ('/content/drive/MyDrive/resume/selecting/candidate_02.pdf', ['machine learning', 'nl  
   ('/content/drive/MyDrive/resume/selecting/candidate_04.pdf', ['machine learning', 'py  
   ('/content/drive/MyDrive/resume/selecting/candidate_06.pdf', ['machine learning', 'py  
   ('/content/drive/MyDrive/resume/selecting/candidate_07.pdf', ['python'])  
   ('/content/drive/MyDrive/resume/selecting/candidate_08.pdf', ['nlp', 'python'])  
   ('/content/drive/MyDrive/resume/selecting/candidate_09.pdf', ['machine learning', 'na  
   ('/content/drive/MyDrive/resume/selecting/candidate_10.pdf', ['ml', 'nlp', 'python'])  
   ('/content/drive/MyDrive/resume/selecting/candidate_11.pdf', ['machine learning'])  
   ('/content/drive/MyDrive/resume/selecting/candidate_12.pdf', ['machine learning', 'py  
   ('/content/drive/MyDrive/resume/selecting/candidate_13.pdf', ['machine learning', 'nl  
   ('/content/drive/MyDrive/resume/selecting/candidate_14.pdf', ['python'])
```

```
▶ 1 skills = ["machine learning","nlp","python"]  
  2 skills1 = ["machine learning","natural language processing","python"]  
  3 selected=[]  
  4 rejected=[]  
  5 for i in range(len(result)):  
  6 | if result[i] == skills or result[i] == skills1:  
  7 | | selected.append(z[i])  
  8 | else:  
  9 | | rejected.append(z[i])  
 10 print('selected resumes:')  
 11 for i in selected:  
 12 | print(i)  
 13  
 14
```

```
↳ selected resumes:  
   /content/drive/MyDrive/resume/selecting/candidate_02.pdf  
   /content/drive/MyDrive/resume/selecting/candidate_09.pdf  
   /content/drive/MyDrive/resume/selecting/candidate_13.pdf
```

```
[ ] 1 nlp=spacy.load('/content/drive/MyDrive/resume/model/output/model-best')
```

```
[ ] 1 import sys, fitz
2 fname='/content/drive/MyDrive/resume/AnuvaGoyal_Latex (1).pdf'
3 doc=fitz.open(fname)
```

```
[ ] 1 text= ' '
2 for page in doc:
3 | text=text+str(page.get_text())
```

```
[ ] 1 text
2
```

' ANUVA GOYAL\n anuvagoyal111@gmail.com\n Agra, Uttar Pradesh, India\n github.cc
t that provides ad-\nvice to the user based on different categories\nof mental health
hat.com (Nov 2021)\n• Full stack Speech Emotion based Movie\nRecommender System using
(Oct\n2021)\n• Finding a Perfect Fit, a model to parse re-\nsumes using Pytesseract,
techniques\n(Aug 2021)\n• Face Mask Detection which detects the face\nusing Haar Casc
the three categories-\nWithout Mask, With Mask and Incorrect Mask\n(May 2021)\nEDUCAT
Science Specialisation\nDayalbagh Educational Institute, Agra\n July 2019- Present
ficate\nSt. Clare's Senior Secondary School, Agra\n 2019 ...'



```
1 doc=nlp(text)
2 for ent in doc.ents:
3 | print(ent.text, '->>>>>>', ent.label_)
```

ANUVA GOYAL ->>>>>>> NAME
anuvagoyal111@gmail.com ->>>>>>> EMAIL ADDRESS
 EDUCATION ->>>>>>> UNIVERSITY
 B.Tech, Electrical Engineering with
 Computer Science Specialisation ->>>>>>> DEGREE
 Dayalbagh Educational Institute, Agra ->>>>>>> COLLEGE NAME
 C ->>>>>>> SKILLS
 C++ ->>>>>>> SKILLS
 Python ->>>>>>> SKILLS
 SQL ->>>>>>> SKILLS
 CSS ->>>>>>> SKILLS
 HTML ->>>>>>> SKILLS
 Machine Learning Frameworks ->>>>>>> SKILLS
 Genisup India Pvt. Ltd., Hosur, Tamil Nadu ->>>>>>> COMPANIES WORKED AT
 VUGS Technologies Pvt. Ltd., Agra, Uttar Pradesh ->>>>>>> COMPANIES WORKED AT
 Deep Learning and Computer Vision A-Z: OpenCV,
 SSD ->>>>>>> SKILLS
 Udemy ->>>>>>> COMPANIES WORKED AT
 30 Days ->>>>>>> YEARS OF EXPERIENCE
 Google LLC ->>>>>>> COMPANIES WORKED AT
 Neural Networks and Deep Learning ->>>>>>> COMPANIES WORKED AT
 Coursera ->>>>>>> WORKED AS
 Machine Learning A-Z: Hands on Python and R
 in Data Science ->>>>>>> DEGREE
 Udemy ->>>>>>> WORKED AS

SOURCE CODE:

```
import locale
print(locale.getpreferredencoding())
def getpreferredencoding(do_setlocale = True):
    return "UTF-8"
locale.getpreferredencoding = getpreferredencoding
!pip install -U spacy
!pip install PyMuPDF
!pip install spacy_transformers
!pip install -U tensorflow==2.9.2
!pip install fastapi==0.90.0
import spacy
from spacy.tokens import DocBin
from tqdm import tqdm
import json
import pickle

cv_data=json.load(open('/content/drive/MyDrive/resume/dataset1111.json','r'))

len(cv_data)

cv_data[0]

!python -m spacy init fill config /content/drive/MyDrive/resume
/base_config.cfg/content/drive/MyDrive/resume/config.cfg

def get_spacy_doc(file,data):
    nlp=spacy.blank('en')
    db=DocBin()
    for text,annot in tqdm(data):
        doc=nlp.make_doc(text)
        annot=annot['entities']
        ents=[]
        entity_indices=[]
        for start,end,label in annot:
            skip_entity=False
            for idx in range(start,end):
                if idx in entity_indices:
                    skip_entity=True
                    break
            if skip_entity==True:
                continue
            entity_indices=entity_indices+list(range(start,end))
    try:
```

```

        span=doc.char_span(start,end,label=label,alignment_mode='strict')
    except:
        continue
    if span is None:
        err_data=str([start,end]) + " " + str(text) + "\n"
        err_data=err_data.encode('utf-8')
        file.write(str(err_data))
    else:
        ents.append(span)
    try:
        doc.ents=ents
        db.add(doc)
    except:
        pass
    return db

from sklearn.model_selection import train_test_split
train,test=train_test_split(cv_data,test_size=0.3)
len(train)

len(test)

file=open('/content/drive/MyDrive/resume/model/train_file.txt','w')

db=get_spacy_doc(file,train)
db.to_disk('/content/drive/MyDrive/resume/model/train_data.spacy')

db=get_spacy_doc(file,test)
db.to_disk('/content/drive/MyDrive/resume/model/test_data.spacy')
file.close()

!python -m spacy train /content/drive/MyDrive/resume/config.cfg --
output /content/drive/MyDrive/resume/model/output --
paths.train /content/drive/MyDrive/resume/model/train_data.spacy --
paths.dev /content/drive/MyDrive/resume/model/test_data.spacy --gpu-
id 0

from glob import glob
import pandas as pd
z=glob("/content/drive/MyDrive/resume/selecting/*")
print(z)

import spacy
import sys, fitz
nlp1 = spacy.load('en_core_web_sm')
result=[]
for i in z:
    doc=fitz.open(i)

```

```

res=doc
text=' '
for page in doc:
    text=text+str(page.get_text())
    text.lower()
def extract_skills(text):
    nlp_text = nlp1(text)

    # removing stop words and implementing word tokenization
    tokens = [token.text for token in nlp_text if not token.is_stop]

    skills = ["nlp", "python", "natural language processing", "machine l
earning", "ml"]

    skillset = []

    # check for one-grams (example: python)
    for token in tokens:
        if token.lower() in skills:
            skillset.append(token)

    # check for bi-grams and tri-grams (example: machine learning)
    for token in nlp_text.noun_chunks:
        token = token.text.lower().strip()
        if token in skills:
            skillset.append(token)

    return [i.capitalize() for i in set([i.lower() for i in skillset]
)]
skillset = extract_skills(text)
for i in range(len(skillset)):
    skillset[i]=skillset[i].lower()
skillset.sort()
if "machine learning" in skillset:
    if "ml" in skillset:
        skillset.remove("ml")
result.append(skillset)
for i in range(len(result)):
    print(result[i])

skills = ["machine learning", "nlp", "python"]
skills1 = ["machine learning", "natural language processing", "python"]
selected=[]
rejected=[]
for i in range(len(result)):
    if result[i] == skills or result[i] == skills1:
        selected.append(z[i])
    else:
        rejected.append(z[i])

```

```

print('selected resumes:')
for i in selected:
    print(i)
print()
print('Rejected resumes:')
for i in rejected:
    print(i)

for i in zip(z,result):
    print(i)

nlp=spacy.load('/content/drive/MyDrive/resume/model/output/model-
best')

import sys, fitz
fname='/content/drive/MyDrive/resume/AnuvaGoyal_Latex.pdf'
doc=fitz.open(fname)

text=' '
for page in doc:
    text=text+str(page.get_text())
doc=nlp(text)
for ent in doc.ents:
    print(ent.text, '->>>>>>', ent.label_)

```

REFERENCE

- [1] Mansouri, A., Affendey, L.S. and Mamat, A., 2008. Named entity recognition approaches. *International Journal of Computer Science and Network Security*, 8(2), 339-344.
- [2] Sidorov, G., Gelbukh, A., Gómez-Adorno, H. and Pinto, D., 2014. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, 18(3), 491-504.
- [3] Singh, A., Rose, C., Visweswariah, K., Chenthamarakshan, V. and Kambhatla, N., 2010, October. PROSPECT: a system for screening candidates for recruitment. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, 659-668.
- [4] Spacy documentation <https://spacy.io/api/doc/training>
- [5] C, B. P., 2020. *towardsai*. [Online] Available at: <https://towardsai.net/p/nlp/natural-language-processing-concepts-andworkflow-48083d2e3ce7>
- [6] Agrawal, R., 2021. *analyticsvidhya*. [Online] Available at: <https://www.analyticsvidhya.com/blog/2021/06/must-know-techniques-for-text-preprocessing-in-nlp/#:~:text=Text%20preprocessing%20is%20a%20method,text%20in%20a%20different%20case>

