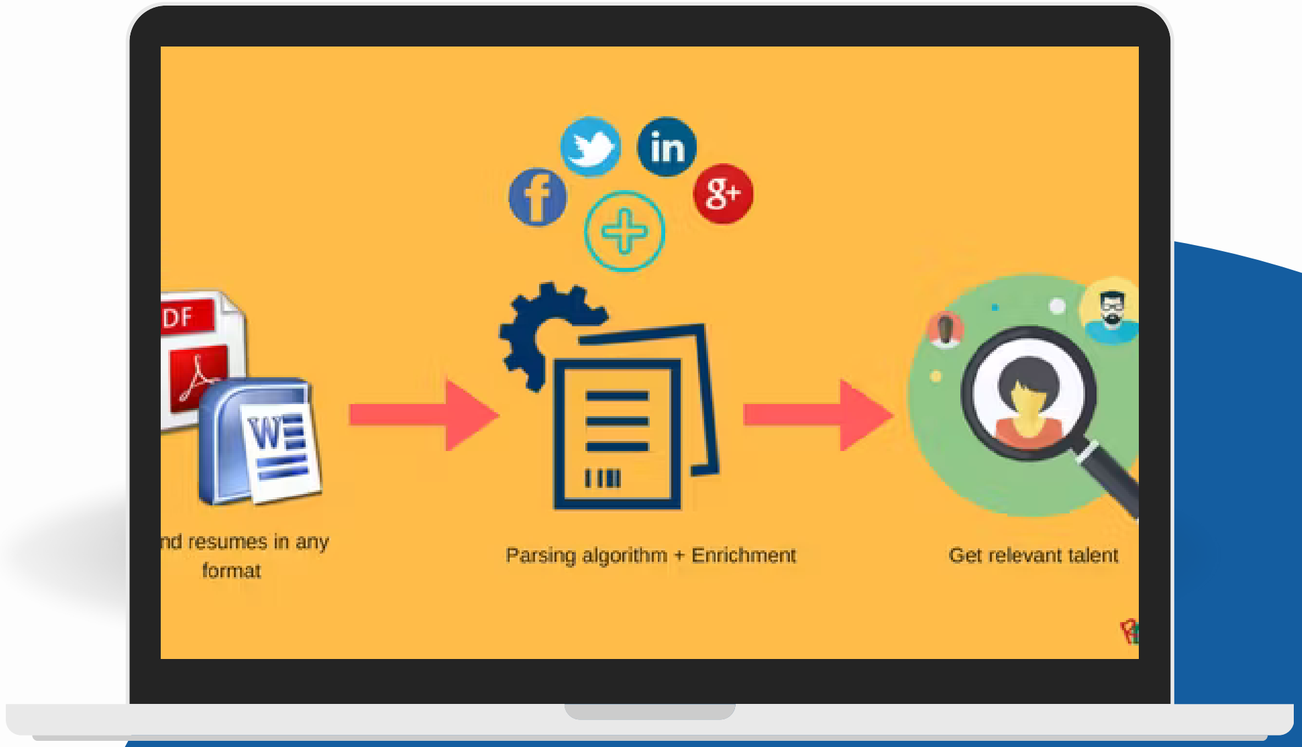
Resume Parsing and Scoring System

Using Natural Language Processing to Extract Key Resume Details

By: Moses Mugambi



Problem Statement

- Recruiters often spend hours manually screening resumes, leading to inefficiencies, inconsistencies, and potential bias in hiring. Traditional methods struggle to process diverse resume formats and extract relevant information accurately.

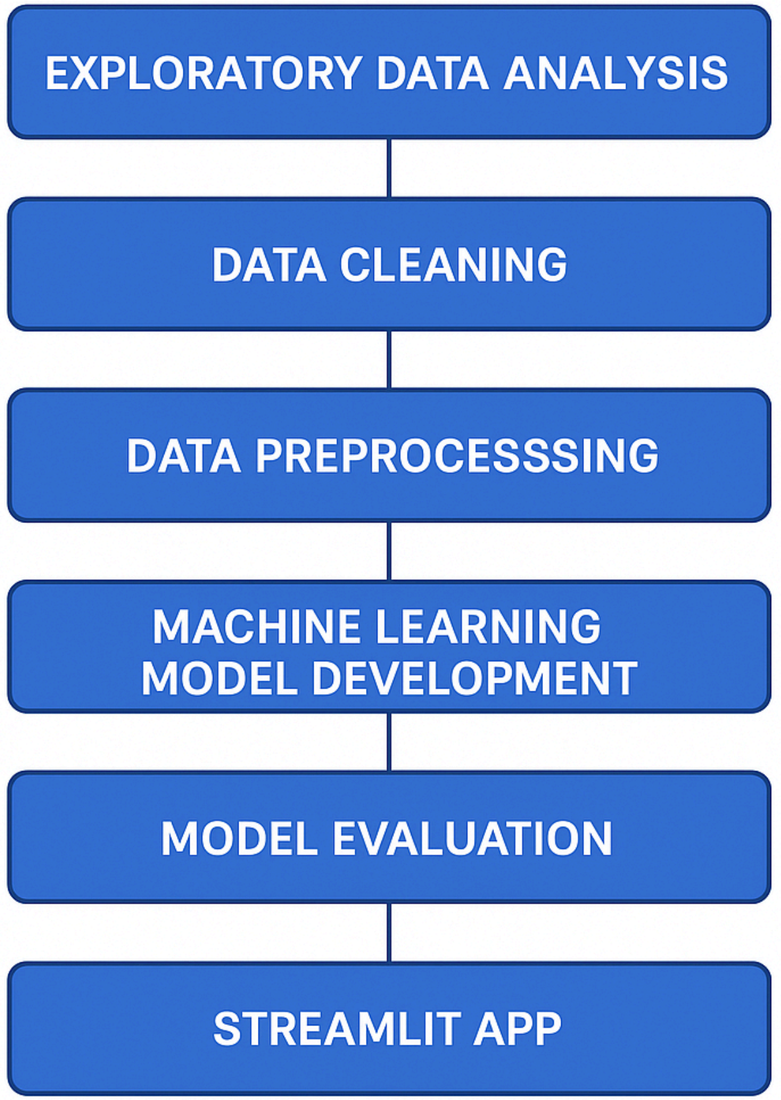
- With the rise of AI and automation, resume parsing has become essential for streamlining recruitment, improving candidate selection, and enhancing HR efficiency.

- Project Goal: Develop robust resume parser system that can accurately extract key details like skills and experience, and score/rank CVs making hiring faster and more efficient.

Project Objectives

- Develop a Named Entity Recognition model to extract key resume details such as Candidates Information (Contacts, Educational Background, Work Experience, Job Role/Title), Skills.
- Score and Rank Resumes based on Job description.
- Batch resume handling.
- Deploy a demonstration web application using Streamlit.

Project Overview



Data Sourcing

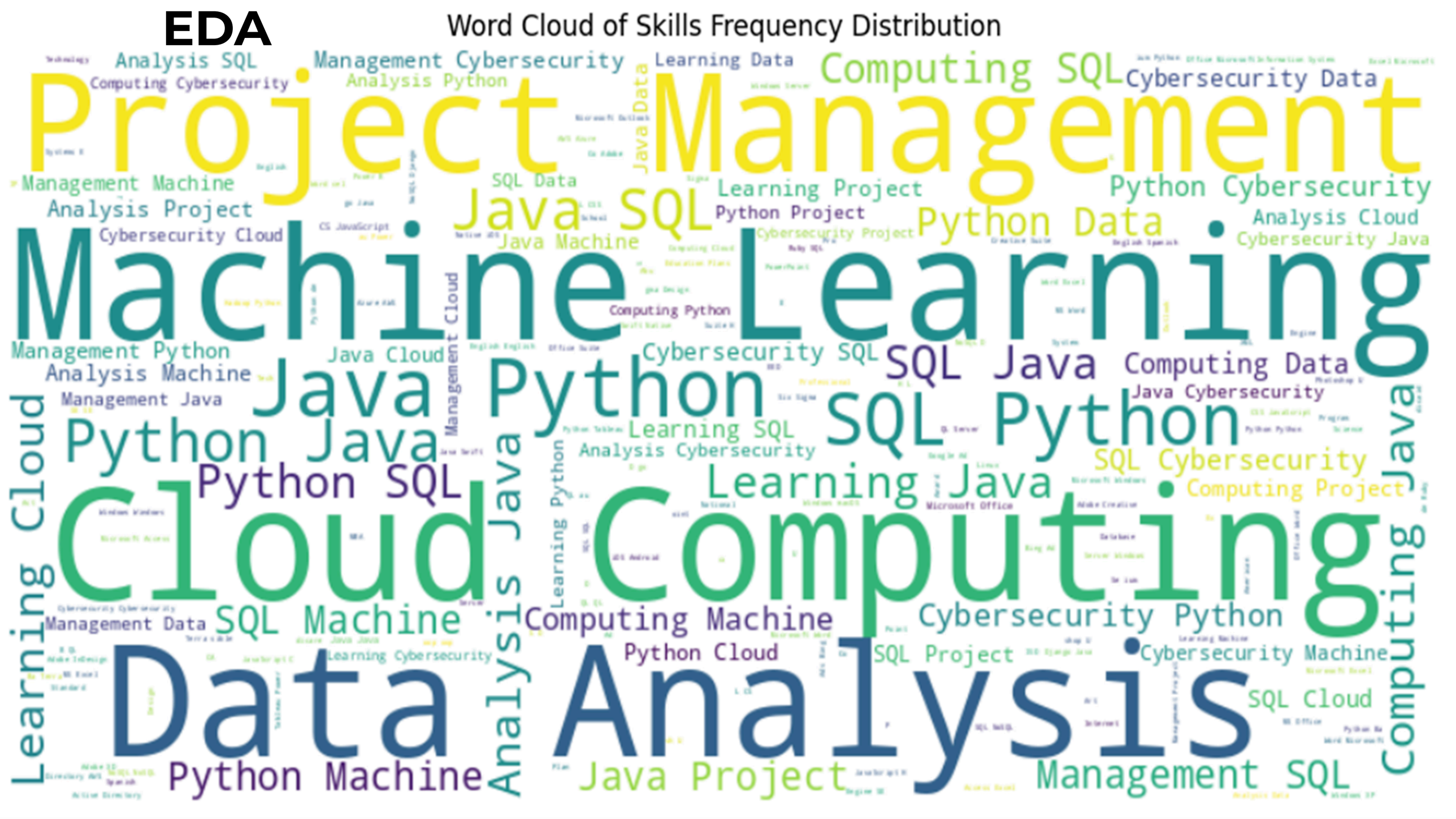
- The original jobs dataset was sourced from Kaggle
- CSV file with Job Roles and extracted texts (1.6 million entries and 23 columns)
- Unnecessary columns were dropped.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37600 entries, 0 to 37599
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Name            37600 non-null  object
 1   Title           37600 non-null  object
 2   Role            37600 non-null  object
 3   Contact         37600 non-null  object
 4   Qualifications  37600 non-null  object
 5   Experience      37600 non-null  object
 6   Skills          37600 non-null  object
 7   Company         37600 non-null  object
dtypes: object(8)
memory usage: 2.3+ MB
```

EDA

Job Role Word Cloud

EDA


Word Cloud of Skills Frequency Distribution
Project Management
Machine Learning
Cloud Computing
Data Analysis
Java Python
Python Java
Python SQL
SQL Python
SQL Java
Learning Cloud
Analysis Java
Computing Java
Learning Python
Management Cloud
Analysis SQL
Management Cybersecurity
Learning Data
Computing SQL
Cybersecurity Data
Computing Cybersecurity
Analysis Python
Java Data
Management Machine
Analysis Project
Cybersecurity Cloud
Java SQL
SQL Data
Learning Project
Python Project
Python Cybersecurity
Python Data
Analysis Cloud
Cybersecurity Java
Cybersecurity Project
Java Machine
Computing Python
Management Python
Java Cloud
Analysis Machine
Management Java
Cybersecurity SQL
SQL Java
Computing Data
Java Cybersecurity
Learning SQL
Analysis Cybersecurity
SQL Cybersecurity
Learning Java
Computing Project
Python SQL
Management Data
SQL Machine
Computing Machine
Cybersecurity Python
Learning Cybersecurity
Python Cloud
SQL Project
Cybersecurity Machine
SQL Cloud
Python Machine
Java Project
Management SQL

Model Selection

- A DistilBERT model was selected for Named Entity Recognition.

- Lightweight & Efficient. DistilBERT is 40% smaller and 60% faster than BERT, making it ideal for real-time parsing systems.
- Good Performance. Retains over 95% of BERT’s accuracy, ensuring robust Named Entity Recognition (NER).
- Pretrained on Rich Text Corpora meaning it can understand contextual meaning in resumes effectively.
- Easily adaptable for classification, entity extraction, and similarity tasks.
- Suitable for deployment in web apps like Streamlit for instant resume analysis.

Data Preprocessing Steps

- Raw Text : Start with the input text.
- Tokenization : Split the text into tokens.
- Token BIO tagging
- Numerical Encoding : Convert tokens to integers.
- Embedding Conversion : Map integers to dense vectors.

Data Preprocessing : Text tokenization

```
[ ] df.head()
```

	Name	Title	Role	Contact	Email	Qualifications	Experience	Skills	Company
0	Christopher Duffy	Back-End Developer	API Developer	922.551.4444	christopherduffy@hotmail.com		3 to 14 Years	RESTful APIs,Cloud Services (AWS/Azure),Script...	State Farm Insurance
1	Stephanie Morris	Back-End Developer	API Developer	806.716.2250x944	smorris@outlook.com	Advanced Certificate	5 to 14 Years	RESTful APIs,Cloud Services (AWS/Azure),Script...	Capital One Financial
2	Anthony Taylor	Back-End Developer	API Developer	(953)310-0075x7268	anthony.taylor@outlook.com	Advanced Diploma	4 to 13 Years	RESTful APIs,Cloud Services (AWS/Azure),Script...	Cummins
3	Jacqueline Anderson	Back-End Developer	API Developer	+1-923-200-8008	janderson@hotmail.com	A-Level	5 to 8 Years	RESTful APIs,Cloud Services (AWS/Azure),Script...	Eastman Chemical
4	Angela Hall	Back-End Developer	API Developer	(246)327-9483	angela@hotmail.com	Associate Degree	3 to 15 Years	RESTful APIs,Cloud Services (AWS/Azure),Script...	Analog Devices

Tokenized Dataset Example:
{'Name': 'Stephanie Morris', 'Title': 'Back-End Developer', 'Role': 'API Developer', 'Contact': '806.716.2250x944', 'Email': 'smorris@outlook.com', 'Qualifications': 'Advanced Certificate', 'Experience': '5 to 14 Years', 'Skills': 'RESTful APIs, Cloud Services (AWS/Azure), Scripting (Python/Bash), Debugging, Version Control (Git), API design and development , 'Company': 'Capital One Financial'}

Data Preprocessing : (Token BIO Tagging)

```
    print(f"Token: {token}, L
```

```
Token: [CLS], Label: O
Token: stephanie, Label: B-Name
Token: morris, Label: I-Name
Token: back, Label: B-Title
Token: -, Label: I-Title
Token: end, Label: I-Title
Token: developer, Label: I-Title
Token: api, Label: B-Role
Token: developer, Label: I-Role
Token: 80, Label: B-Contact
Token: ##6, Label: B-Contact
Token: ., Label: I-Contact
Token: 71, Label: I-Contact
Token: ##6, Label: I-Contact
Token: ., Label: I-Contact
Token: 225, Label: I-Contact
Token: ##0, Label: I-Contact
Token: ##x, Label: I-Contact
Token: ##9, Label: I-Contact
Token: ##44, Label: I-Contact
Token: sm, Label: B-Email
Token: ##or, Label: B-Email
Token: ##ris, Label: B-Email
```

```
Token: ##ris, Label: B-Email
Token: @, Label: B-Email
Token: outlook, Label: B-Email
Token: ., Label: I-Email
Token: com, Label: I-Email
Token: advanced, Label: B-Qualifications
Token: certificate, Label: I-Qualifications
Token: 5, Label: B-Experience
Token: to, Label: I-Experience
Token: 14, Label: I-Experience
Token: years, Label: I-Experience
Token: rest, Label: B-Skills
Token: ##ful, Label: B-Skills
Token: api, Label: I-Skills
Token: ##s, Label: I-Skills
Token: ,, Label: O
Token: cloud, Label: B-Skills
Token: services, Label: I-Skills
Token: (, Label: I-Skills
Token: aw, Label: I-Skills
Token: ##s, Label: I-Skills
Token: /, Label: I-Skills
Token: azure, Label: I-Skills
Token: ), Label: I-Skills
Token: ,, Label: O
Token: script, Label: B-Skills
Token: ##ing, Label: B-Skills
Token: (, Label: I-Skills
Token: python, Label: I-Skills
```

```
Token: (, Label: I-Skills
Token: python, Label: I-Skills
Token: /, Label: I-Skills
Token: bash, Label: I-Skills
Token: ), Label: I-Skills
Token: ,, Label: O
Token: de, Label: B-Skills
Token: ##bu, Label: B-Skills
Token: ##gging, Label: B-Skills
Token: ,, Label: O
Token: version, Label: B-Skills
Token: control, Label: I-Skills
Token: (, Label: I-Skills
Token: gi, Label: I-Skills
Token: ##t, Label: I-Skills
Token: ), Label: I-Skills
Token: ,, Label: O
Token: api, Label: B-Skills
Token: design, Label: I-Skills
Token: and, Label: I-Skills
Token: development, Label: I-Skills
Token: rest, Label: I-Skills
Token: ##ful, Label: I-Skills
Token: api, Label: I-Skills
Token: knowledge, Label: I-Skills
Token: security, Label: I-Skills
Token: protocols, Label: I-Skills
Token: o, Label: I-Skills
```

```
Token: security, Label: I-Skills
Token: protocols, Label: I-Skills
Token: o, Label: I-Skills
Token: ##au, Label: I-Skills
Token: ##th, Label: I-Skills
Token: ,, Label: O
Token: j, Label: B-Skills
Token: ##wt, Label: B-Skills
Token: capital, Label: B-Company
Token: one, Label: I-Company
Token: financial, Label: I-Company
Token: [SEP], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
Token: [PAD], Label: O
```

- B- denotes the beginning of an entity, I- denotes inside an entity, and O denotes no entity.

Data Preprocessing : (Numerical Labelling)

```
inspect_tokenized_output(tokenized_dataset,
```

```
Tokenized Input IDs, Tokens, and Aligned Labels:
Input ID: 101, Token: [CLS], Label: 0
Input ID: 11496, Token: stephanie, Label: 1
Input ID: 6384, Token: morris, Label: 2
Input ID: 2067, Token: back, Label: 3
Input ID: 1011, Token: -, Label: 4
Input ID: 2203, Token: end, Label: 4
Input ID: 9722, Token: developer, Label: 4
Input ID: 17928, Token: api, Label: 5
Input ID: 9722, Token: developer, Label: 6
Input ID: 3770, Token: 80, Label: 7
Input ID: 2575, Token: ##6, Label: 7
Input ID: 1012, Token: ., Label: 8
Input ID: 6390, Token: 71, Label: 8
Input ID: 2575, Token: ##6, Label: 8
Input ID: 1012, Token: ., Label: 8
Input ID: 14993, Token: 225, Label: 8
Input ID: 2692, Token: ##0, Label: 8
Input ID: 2595, Token: ##x, Label: 8
Input ID: 2683, Token: ##9, Label: 8
Input ID: 22932, Token: ##44, Label: 8
Input ID: 15488, Token: sm, Label: 9
Input ID: 2953, Token: ##or, Label: 9
Input ID: 6935, Token: ##ris, Label: 9
Input ID: 1030, Token: @, Label: 9
Input ID: 17680, Token: outlook, Label: 9
Input ID: 1012, Token: ., Label: 10
```

```
Input ID: 4012, Token: com, Label: 10
Input ID: 3935, Token: advanced, Label: 11
Input ID: 8196, Token: certificate, Label: 12
Input ID: 1019, Token: 5, Label: 13
Input ID: 2000, Token: to, Label: 14
Input ID: 2403, Token: 14, Label: 14
Input ID: 2086, Token: years, Label: 14
Input ID: 2717, Token: rest, Label: 15
Input ID: 3993, Token: ##ful, Label: 15
Input ID: 17928, Token: api, Label: 16
Input ID: 2015, Token: ##s, Label: 16
Input ID: 1010, Token: ,, Label: 0
Input ID: 6112, Token: cloud, Label: 15
Input ID: 2578, Token: services, Label: 16
Input ID: 1006, Token: (, Label: 16
Input ID: 22091, Token: aw, Label: 16
Input ID: 2015, Token: ##s, Label: 16
Input ID: 1013, Token: /, Label: 16
Input ID: 24296, Token: azure, Label: 16
Input ID: 1007, Token: ), Label: 16
Input ID: 1010, Token: ,, Label: 0
Input ID: 5896, Token: script, Label: 15
Input ID: 2075, Token: ##ing, Label: 15
Input ID: 1006, Token: (, Label: 16
Input ID: 18750, Token: python, Label: 16
Input ID: 1013, Token: /, Label: 16
Input ID: 24234, Token: bash, Label: 16
Input ID: 1007, Token: ), Label: 16
Input ID: 1010, Token: ,, Label: 0
```

```
Input ID: 12588, Token: ##gging, Label: 15
Input ID: 1010, Token: ,, Label: 0
Input ID: 2544, Token: version, Label: 15
Input ID: 2491, Token: control, Label: 16
Input ID: 1006, Token: (, Label: 16
Input ID: 21025, Token: gi, Label: 16
Input ID: 2102, Token: ##t, Label: 16
Input ID: 1007, Token: ), Label: 16
Input ID: 1010, Token: ,, Label: 0
Input ID: 17928, Token: api, Label: 15
Input ID: 2640, Token: design, Label: 16
Input ID: 1998, Token: and, Label: 16
Input ID: 2458, Token: development, Label: 16
Input ID: 2717, Token: rest, Label: 16
Input ID: 3993, Token: ##ful, Label: 16
Input ID: 17928, Token: api, Label: 16
Input ID: 3716, Token: knowledge, Label: 16
Input ID: 3036, Token: security, Label: 16
Input ID: 16744, Token: protocols, Label: 16
Input ID: 1051, Token: o, Label: 16
Input ID: 4887, Token: ##au, Label: 16
Input ID: 2705, Token: ##th, Label: 16
Input ID: 1010, Token: ,, Label: 0
Input ID: 1046, Token: j, Label: 15
Input ID: 26677, Token: ##wt, Label: 15
Input ID: 3007, Token: capital, Label: 17
Input ID: 2028, Token: one, Label: 18
Input ID: 3361, Token: financial, Label: 18
Input ID: 102, Token: [SEP], Label: 0
```

- Numerical labels are then converted to embeddings (vectors) using a vector map in the embedding layer

Train - Test Split

```
# Split the dataset into train and test sets
train_test_split = tokenized_dataset.train_test_split(test_size=0.2)
train_dataset = train_test_split['train']
test_dataset = train_test_split['test']

# Optionally, split the training set further into train and validation sets
train_val_split = train_dataset.train_test_split(test_size=0.1)
train_dataset = train_val_split['train']
val_dataset = train_val_split['test']

print(f"Training examples: {len(train_dataset)}")
print(f"Validation examples: {len(val_dataset)}")
print(f"Test examples: {len(test_dataset)}")
```

```
Training examples: 27072
Validation examples: 3008
Test examples: 7520
```

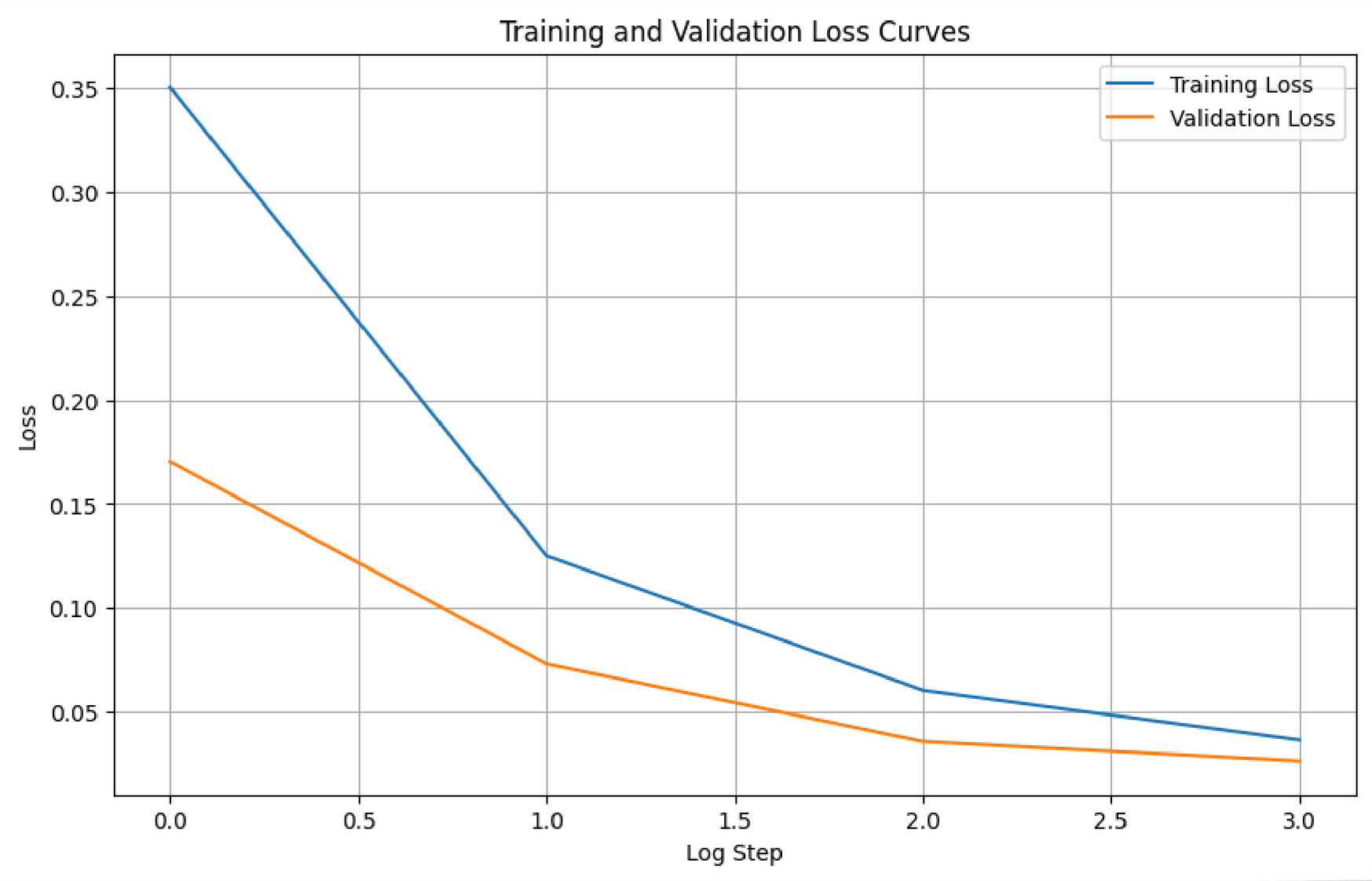
- Only a small fraction of the dataset was used due to computational limitations (37600) entries

Modelling Results and Evaluation

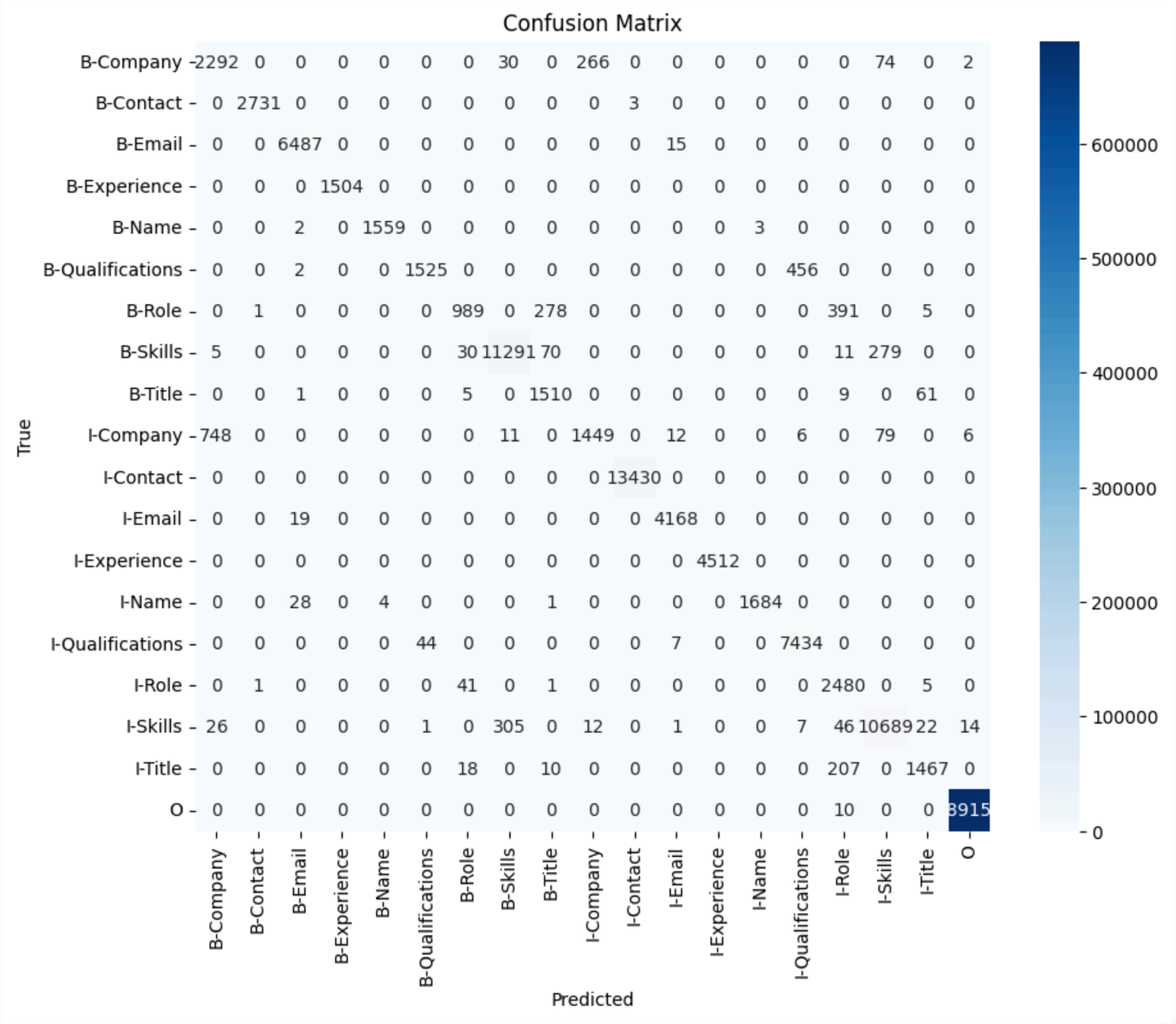
```
# Print the classification report
print(classification_report(true_labels, true_predictions))
```

	precision	recall	f1-score	support
Company	0.55	0.67	0.60	2664
Contact	1.00	1.00	1.00	2734
Email	0.99	1.00	0.99	6502
Experience	1.00	1.00	1.00	1504
Name	0.96	0.98	0.97	1564
Qualifications	0.79	0.67	0.73	1983
Role	0.60	0.63	0.62	1710
Skills	0.92	0.92	0.92	11686
Title	0.63	0.78	0.70	1649
micro avg	0.87	0.89	0.88	31996
macro avg	0.83	0.85	0.84	31996
weighted avg	0.88	0.89	0.88	31996

Overfitting Analysis (Loss Curves)



Confussion Matrix



Streamlit Web Application

Challenges

- Unable to train on entire dataset due to computational and time limitations.
- Model overfitting during training.
- Faced difficulties in streamlit model app deployment

Future Improvements

- Use a large and diverse manually annotated dataset.
- Domain-specific fine-tuning for improved accuracy.
- Support for multilingual resumes.
- Integration with job boards or ATS.
- Highly interactive dashboard for recruiters to manage parsed resumes.

Conclusion

- Automated resume parsing system using NLP to extract entities and score resumes.

- Streamlit interface for ease of use.