232594T Natural Language Processing Project - Coded by Sai Keerthan (232594T)

In this notebook we will go through the following:

1. Text-Preprocessing
2. Text-Vectorisation
3. BERT Model Fine Tuning and Optimisation techniques

Import Libraries:

In [301…

```
## Import libraries
import numpy as np
import pandas as pd
import re
import gensim.downloader
import matplotlib.pyplot as plt

from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score
from sklearn.metrics import classification_report, confusion_matrix
from nltk.tokenize import word_tokenize
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

Text Pre-Processing

The different Text pre-processing steps done is:

1. Change everything to lowercase

Appending Own Resume to the dataset

In [287…

```
# 1. Access the datasst given which consists of Resumes
import pandas as pd

nlp_dataset = pd.read_csv("/Users/saikeerthan/NYP-AI/NLP/Project/SATINI S

nlp_dataset
```

Out[287…

	Category	Resume
0	Blockchain	SKILLS Bitcoin, Ethereum Solidity Hyperledger,...
1	Java Developer	Operating Systems Windows XP, 7, 10. Tools/Pac...
2	Java Developer	SKILLS: - 1) Team leading 2) Self-motivated 3)...
3	Testing	COMPUTER PROFICIENCY â€¢ Basic: MS-Office (Pow...
4	Hadoop	Skill Set: Hadoop, Map Reduce, HDFS, Hive, Sqo...
...	...	...
957	Java Developer	TECHNICAL SKILLS Skills: Java, SQL, PL/SQL, C,...
958	Python Developer	Technical Skills / Responsibilities: â€¢ Hands...
959	Arts	I Other Skills Course/Skill Name Board Year Of...
960	Sales	IT Skills: MS Office. Photoshop. SQL Server.Ed...
961	PMO	Skills Exceptional communication and networkin...

962 rows × 2 columns

In [288…

```
# make a copy of the current dataset

nlp_dataset_copy = nlp_dataset.copy()

nlp_dataset_copy
```

Out[288…

	Category	Resume
0	Blockchain	SKILLS Bitcoin, Ethereum Solidity Hyperledger,...
1	Java Developer	Operating Systems Windows XP, 7, 10. Tools/Pac...
2	Java Developer	SKILLS: - 1) Team leading 2) Self-motivated 3)...
3	Testing	COMPUTER PROFICIENCY â€¢ Basic: MS-Office (Pow...
4	Hadoop	Skill Set: Hadoop, Map Reduce, HDFS, Hive, Sqo...
...	...	...
957	Java Developer	TECHNICAL SKILLS Skills: Java, SQL, PL/SQL, C,...
958	Python Developer	Technical Skills / Responsibilities: â€¢ Hands...
959	Arts	I Other Skills Course/Skill Name Board Year Of...
960	Sales	IT Skills: MS Office. Photoshop. SQL Server.Ed...
961	PMO	Skills Exceptional communication and networkin...

962 rows × 2 columns

In [289…

```
import pandas as pd

# get the input of my resume
category = input("Category: ")
resume_text = input("Resume Text: ")
```

```
# creates a new entry based on my resume
sai_entry = {
    "Category": category,
    "Resume": resume_text
}

# append it to the copied version of the original dataset
new_entry_df = pd.DataFrame([sai_entry])
nlp_dataset_copy = pd.concat([nlp_dataset_copy, new_entry_df], ignore_ind

# save the updated copy
nlp_dataset_copy.to_csv("/Users/saikeerthan/NYP-AI/NLP/Project/SATINI_SAI

print("Resume added successfully to the dataset copy!")
```

```
Resume added successfully to the dataset copy!
```

In [290…
```
# view the last entry(my resume)

nlp_dataset_copy.iloc[-1]
```

Out[290…
```
Category                                            Data Science
Resume      Sai Keerthan Satini  AI & DATA ENGINEER   Choa...
Name: 962, dtype: object
```

Clean the Dataset

- Here are the Data Cleaning steps that will be done in this cell:

    1. Removing Duplicates and Missing Values
    2. Converting everything to lowercase
    3. Remove Punctuation
    4. Remove Stop Words
    5. Lemmatization
    6. Removing any special characters or spaces in the dataset

Removing Duplicates and Missing Values

In [291…
```
# checking for missing values
print(nlp_dataset_copy.isnull().sum())

# no missing values
```

```
Category    0
Resume      0
dtype: int64
```

In [292…
```
# checking for duplicates
print(nlp_dataset_copy.duplicated().sum())
```

```
741
```

In [293…
```
# since there are 741 duplicates, let's view to see if we have to remove

# showing all the duplicates
duplicates = nlp_dataset_copy[nlp_dataset_copy.duplicated()]
print(duplicates)
```

```
            Category                                             Resume
26       Data Science  Skills * Programming Languages: Python (pandas...
30              Sales  KEY SKILLS: â€¢ Planning & Strategizing â€¢ Pr...
31             Hadoop  Areas of expertise â€¢ Big Data Ecosystems: Ha...
32         Blockchain  Hobbies â€¢ Playing Chess â€¢ Solving Rubik's ...
38   Python Developer  â€¢ Operating Systems: Windows â€¢ Others: MS ...
..                ...                                              ...
956   DevOps Engineer  CORE COMPETENCIES ~ Ant ~ Maven ~ GIT ~ Bitbuc...
957    Java Developer  TECHNICAL SKILLS Skills: Java, SQL, PL/SQL, C,...
958  Python Developer  Technical Skills / Responsibilities: â€¢ Hands...
959              Arts  I Other Skills Course/Skill Name Board Year Of...
961               PMO  Skills Exceptional communication and networkin...

[741 rows x 2 columns]
```

In [294…

```
nlp_dataset_copy.drop_duplicates(keep='first', inplace=True)

nlp_dataset_copy.reset_index(drop=True, inplace=True)

print("\nDuplicates remaining:", nlp_dataset_copy.duplicated().sum())

print(nlp_dataset_copy)
```

```
Duplicates remaining: 0
               Category                                             Resume
0            Blockchain  SKILLS Bitcoin, Ethereum Solidity Hyperledger,...
1         Java Developer  Operating Systems Windows XP, 7, 10. Tools/Pac...
2         Java Developer  SKILLS: - 1) Team leading 2) Self-motivated 3)...
3                Testing  COMPUTER PROFICIENCY â€¢ Basic: MS-Office (Pow...
4                 Hadoop  Skill Set: Hadoop, Map Reduce, HDFS, Hive, Sqo...
..                   ...                                              ...
217     DevOps Engineer  TECHNICAL SKILLS â€¢ HP ALM, RTC and JIRA â€¢ ...
218       ETL Developer  Technical Summary â€¢ Knowledge of Informatica...
219  Automation Testing  Technical Skills Summary I have completed "COR...
220               Sales  IT Skills: MS Office. Photoshop. SQL Server.Ed...
221        Data Science  Sai Keerthan Satini  AI & DATA ENGINEER   Choa...

[222 rows x 2 columns]
```

In [296…

```
unique_classes = nlp_dataset_copy["Category"].unique()
num_unique_classes = nlp_dataset_copy["Category"].nunique()
num_resume_per_category = nlp_dataset_copy["Category"].value_counts()

print(f"The unique classes in the df are: {unique_classes}")
print("\n")
print(f"The number of unique classes in the df are: {num_unique_classes}"
print("\n")
print(f"Number of resumes per category is: {num_resume_per_category}")
```

```
The unique classes in the df are: ['Blockchain' 'Java Developer' 'Testing'
'Hadoop' 'Civil Engineer'
 'Business Analyst' 'Data Science' 'Python Developer' 'Arts' 'Database'
 'Advocate' 'Operations Manager' 'Sales' 'DotNet Developer'
 'Automation Testing' 'Web Designing' 'Electrical Engineering' 'PMO'
 'DevOps Engineer' 'ETL Developer' 'HR' 'Mechanical Engineer'
 'Health and fitness' 'Network Security Engineer' 'SAP Developer']


The number of unique classes in the df are: 25


Number of resumes per category is: Category
Java Developer                16
Data Science                  14
Database                      14
HR                            13
Automation Testing            11
Advocate                      11
Blockchain                    10
Arts                          10
Testing                        9
Hadoop                         9
Electrical Engineering         9
DotNet Developer               9
Health and fitness             8
DevOps Engineer                8
Sales                          8
Business Analyst               8
Python Developer               7
ETL Developer                  7
Mechanical Engineer            7
SAP Developer                  7
Web Designing                  6
Civil Engineer                 6
Network Security Engineer      6
Operations Manager             5
PMO                            4
Name: count, dtype: int64
```

Converting every text into lowercase and removing punctuation

In [297…

```python
import string
def lowercase_conversion(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    return text

# Apply cleaning to the Resume column
nlp_dataset_copy["Resume"] = nlp_dataset_copy["Resume"].apply(lowercase_c
```

In [298…

```python
nlp_dataset_copy
```

Out[298...

	Category	Resume
0	Blockchain	skills bitcoin ethereum solidity hyperledger b...
1	Java Developer	operating systems windows xp 7 10 toolspackage...
2	Java Developer	skills 1 team leading 2 selfmotivated 3 hard ...
3	Testing	computer proficiency â€¢ basic msoffice powerp...
4	Hadoop	skill set hadoop map reduce hdfs hive sqoop ja...
...	...	...
217	DevOps Engineer	technical skills â€¢ hp alm rtc and jira â€¢ a...
218	ETL Developer	technical summary â€¢ knowledge of informatica...
219	Automation Testing	technical skills summary i have completed corp...
220	Sales	it skills ms office photoshop sql servereducat...
221	Data Science	sai keerthan satini ai data engineer choa ...

222 rows × 2 columns

Removing Stop Words from the dataset

- Removing stop words through nltk package

In [299...

```
import nltk
from nltk.corpus import stopwordsRe

nltk.download("stopwords") # download the stopwords from nltk
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/saikeerthan/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[299... True

In [300...

```
stopwords = set(stopwords.words('english'))

def stopwords_removal(entries):
    tokens = entries.split()
    enhanced_tokens = [entry for entry in tokens if entry not in stopword
    return ' '.join(enhanced_tokens)

# add the changes to the dataset
nlp_dataset_copy["Resume"] = nlp_dataset_copy["Resume"].apply(stopwords_r

nlp_dataset_copy
```

Out [300…

	Category	Resume
0	Blockchain	skills bitcoin ethereum solidity hyperledger b...
1	Java Developer	operating systems windows xp 7 10 toolspackage...
2	Java Developer	skills 1 team leading 2 selfmotivated 3 hard w...
3	Testing	computer proficiency â€¢ basic msoffice powerp...
4	Hadoop	skill set hadoop map reduce hdfs hive sqoop ja...
...	...	...
217	DevOps Engineer	technical skills â€¢ hp alm rtc jira â€¢ as400...
218	ETL Developer	technical summary â€¢ knowledge informatica po...
219	Automation Testing	technical skills summary completed corporate t...
220	Sales	skills ms office photoshop sql servereducation...
221	Data Science	sai keerthan satini ai data engineer choa chu ...

222 rows × 2 columns

Removing standalone numbers from the dataset

These numbers can just be list items, and it may affect the Text Vectorisation process

In [302…

```
def remove_standlone_num(text):
    tokens = word_tokenize(str(text))
    filtered_tokens = [token for token in tokens if not (token.isdigit()
    return " ".join(filtered_tokens)

nlp_dataset_copy["Resume"] = nlp_dataset_copy["Resume"].apply(remove_stan

nlp_dataset_copy
```

Out[302…

	Category	Resume
0	Blockchain	skills bitcoin ethereum solidity hyperledger b...
1	Java Developer	operating systems windows xp 10 toolspackages ...
2	Java Developer	skills team leading selfmotivated hard working...
3	Testing	computer proficiency â€¢ basic msoffice powerp...
4	Hadoop	skill set hadoop map reduce hdfs hive sqoop ja...
...	...	...
217	DevOps Engineer	technical skills â€¢ hp alm rtc jira â€¢ as400...
218	ETL Developer	technical summary â€¢ knowledge informatica po...
219	Automation Testing	technical skills summary completed corporate t...
220	Sales	skills ms office photoshop sql servereducation...
221	Data Science	sai keerthan satini ai data engineer choa chu ...

222 rows × 2 columns

Apply Word Tokenisation to the dataset

In [303…

```
nlp_dataset_copy["Resume"] = nlp_dataset_copy["Resume"].apply(word_tokeni

nlp_dataset_copy
```

Out[303…

	Category	Resume
0	Blockchain	[skills, bitcoin, ethereum, solidity, hyperled...
1	Java Developer	[operating, systems, windows, xp, 10, toolspac...
2	Java Developer	[skills, team, leading, selfmotivated, hard, w...
3	Testing	[computer, proficiency, â€¢, basic, msoffice, ...
4	Hadoop	[skill, set, hadoop, map, reduce, hdfs, hive, ...
...	...	...
217	DevOps Engineer	[technical, skills, â€¢, hp, alm, rtc, jira, â...
218	ETL Developer	[technical, summary, â€¢, knowledge, informati...
219	Automation Testing	[technical, skills, summary, completed, corpor...
220	Sales	[skills, ms, office, photoshop, sql, serveredu...
221	Data Science	[sai, keerthan, satini, ai, data, engineer, ch...

222 rows × 2 columns

Apply Lemmatisation to the dataset

In [304…

```
from nltk.stem import WordNetLemmatizer
nltk.download("wordnet")
```

```
# Initialise the Lemmatiser from wordnet
lemmatizer = WordNetLemmatizer()

def lemmatize_text(tokens):
    return [lemmatizer.lemmatize(word) for word in tokens]

nlp_dataset_copy["Resume"] = nlp_dataset_copy["Resume"].apply(lemmatize_t

nlp_dataset_copy
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/saikeerthan/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[304…

	Category	Resume
0	Blockchain	[skill, bitcoin, ethereum, solidity, hyperledg...
1	Java Developer	[operating, system, window, xp, 10, toolspacka...
2	Java Developer	[skill, team, leading, selfmotivated, hard, wo...
3	Testing	[computer, proficiency, â€¢, basic, msoffice, ...
4	Hadoop	[skill, set, hadoop, map, reduce, hdfs, hive, ...
...	...	...
217	DevOps Engineer	[technical, skill, â€¢, hp, alm, rtc, jira, â€...
218	ETL Developer	[technical, summary, â€¢, knowledge, informati...
219	Automation Testing	[technical, skill, summary, completed, corpora...
220	Sales	[skill, m, office, photoshop, sql, servereduca...
221	Data Science	[sai, keerthan, satini, ai, data, engineer, ch...

222 rows × 2 columns

Saving the tokenised dataset as a separate CSV

In [305…

```
nlp_dataset_copy.to_csv("/Users/saikeerthan/NYP-AI/NLP/Project/SATINI_SAI
print("Cleaned dataset saved successfully!")
```

```
Cleaned dataset saved successfully!
```

Text Vectorisation

We will take a look at 4 different text-vectorisation methods and compare them with my appended CV through cosine similarity

1. Word2Vec - GLOVE

In [17]:

```
# make an exclusive copy for text-vectorisation

word2_vectorisation_df = nlp_dataset_copy.copy()
```

```
word2_vectorisation_df
```

Out[17]:

	Category	Resume
0	Blockchain	[skill, bitcoin, ethereum, solidity, hyperledg...
1	Java Developer	[operating, system, window, xp, 10, toolspacka...
2	Java Developer	[skill, team, leading, selfmotivated, hard, wo...
3	Testing	[computer, proficiency, â€¢, basic, msoffice, ...
4	Hadoop	[skill, set, hadoop, map, reduce, hdfs, hive, ...
...	...	...
958	Python Developer	[technical, skill, responsibility, â€¢, hand, ...
959	Arts	[skill, courseskill, name, board, year, passin...
960	Sales	[skill, m, office, photoshop, sql, servereduca...
961	PMO	[skill, exceptional, communication, networking...
962	AI & Data Enginner	[satini, sai, keerthan, ai, data, engineer, 65...

963 rows × 2 columns

In [18]:
```
word2vec_model = gensim.downloader.load('glove-wiki-gigaword-50')
```

In [ ]:
```
def vectorize_documents(documents, word2vec_model):
    document_vectors = []
    for document in documents:
        if isinstance(document, str):
            tokens = document.lower().split()
        else:
            tokens = document
        vectors = []
        for token in tokens:
            if token in word2vec_model:
                vectors.append(word2vec_model[token])
        if vectors:
            document_vectors.append(sum(vectors) / len(vectors))
        else:
            document_vectors.append([0] * 50)
    return document_vectors

resume_vectors = vectorize_documents(word2_vectorisation_df['Resume'], wo
```

In [21]:
```
def calculate_cosine_similarity(document_vector1, document_vector2):
    return cosine_similarity([document_vector1], [document_vector2])[0][0


my_cv_vector = resume_vectors[-1]

# Calculate similarity scores for all resumes
similarity_scores = []
for i, resume_vector in enumerate(resume_vectors):
    similarity_score = calculate_cosine_similarity(my_cv_vector, resume_v
    similarity_scores.append((i, similarity_score))
```

```
similarity_scores.sort(key=lambda x: x[1], reverse=True)


ten_similar_resumes = similarity_scores[1:11]


print("Top 10 resumes similar to my resume:")
for index, score in ten_similar_resumes:
    print(f"Resume {index + 1}: Similarity Score = {score:.4f}")
```

```
Top 10 resumes similar to my resume:
Resume 466: Similarity Score = 0.9731
Resume 503: Similarity Score = 0.9731
Resume 704: Similarity Score = 0.9731
Resume 727: Similarity Score = 0.9731
Resume 73: Similarity Score = 0.9726
Resume 489: Similarity Score = 0.9726
Resume 746: Similarity Score = 0.9726
Resume 764: Similarity Score = 0.9726
Resume 265: Similarity Score = 0.9724
Resume 350: Similarity Score = 0.9724
```

2. TF-IDF:

In [22]:
```
tfidf_df = nlp_dataset_copy.copy()

tfidf_df
```

Out[22]:

	Category	Resume
0	Blockchain	[skill, bitcoin, ethereum, solidity, hyperledg...
1	Java Developer	[operating, system, window, xp, 10, toolspacka...
2	Java Developer	[skill, team, leading, selfmotivated, hard, wo...
3	Testing	[computer, proficiency, â€¢, basic, msoffice, ...
4	Hadoop	[skill, set, hadoop, map, reduce, hdfs, hive, ...
...	...	...
958	Python Developer	[technical, skill, responsibility, â€¢, hand, ...
959	Arts	[skill, courseskill, name, board, year, passin...
960	Sales	[skill, m, office, photoshop, sql, servereduca...
961	PMO	[skill, exceptional, communication, networking...
962	AI & Data Enginner	[satini, sai, keerthan, ai, data, engineer, 65...

963 rows × 2 columns

In [335…
```
def calculat_tf(value):
    tf = np.log10(value+1)
    return tf

def calculate_idf(total_documents, value):
```

```
    idf = np.log10(total_documents / value)
    return idf


documents = tfidf_df["Resume"].apply(lambda x: " ".join(x) if isinstance(
vectoriser = CountVectorizer()

X = vectoriser.fit_transform(documents)

tdm = pd.DataFrame(X.toarray(), index=[f"Document {i + 1}"
                                       for i in range(len(documents))], c


total_documents = len(documents)
total_documents_per_word = np.sum(tdm > 0, axis=0)
idf_array = total_documents_per_word.apply(lambda value: calculate_idf(to


tf_matrix = tdm.map(calculat_tf)


tf_idf_matrix = tf_matrix * idf_array.to_numpy()
```

In [26]:
```
tf_idf_array = tf_idf_matrix.to_numpy()


my_cv_vector = tf_idf_array[-1].reshape(1, -1)


similarity_scores = cosine_similarity(my_cv_vector, tf_idf_array).flatten


top_indices = similarity_scores.argsort()[::-1][1:11]


print("Top 10 resumes similar to my resume:")
for i in top_indices:
    print(f"Resume {i + 1}: Similarity Score = {similarity_scores[i]:.4f}
```

```
Top 10 resumes similar to my resume:
Resume 764: Similarity Score = 0.0983
Resume 746: Similarity Score = 0.0983
Resume 489: Similarity Score = 0.0983
Resume 73: Similarity Score = 0.0983
Resume 242: Similarity Score = 0.0868
Resume 203: Similarity Score = 0.0868
Resume 612: Similarity Score = 0.0868
Resume 708: Similarity Score = 0.0868
Resume 667: Similarity Score = 0.0834
Resume 36: Similarity Score = 0.0834
```

3. Term Document Matrix

In [27]:
```
tdm_df = nlp_dataset_copy.copy()
```

In [ ]:
```
# create tdm
vectoriser = CountVectorizer()
documents = tdm_df["Resume"].apply(lambda x: " ".join(x) if isinstance(x,
X = vectoriser.fit_transform(documents)
```

```
tdm = pd.DataFrame(
    X.toarray(),
    index = [f"Document {i + 1}" for i in range(len(documents))],
    columns=vectoriser.get_feature_names_out(),
)
```

In [ ]:
```
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np


tdm_array = tdm.to_numpy()


my_cv_vector = tdm_array[-1].reshape(1, -1)


similarity_scores = cosine_similarity(my_cv_vector, tdm_array).flatten()


top_indices = similarity_scores.argsort()[::-1][1:11]


print("Top 10 resumes similar to my resume:")
for i in top_indices:
    print(f"Resume {i + 1}: Similarity Score = {similarity_scores[i]:.4f}")
```

```
Top 10 resumes similar to my resume:
Resume 203: Similarity Score = 0.5562
Resume 612: Similarity Score = 0.5562
Resume 242: Similarity Score = 0.5562
Resume 708: Similarity Score = 0.5562
Resume 73: Similarity Score = 0.5369
Resume 489: Similarity Score = 0.5369
Resume 764: Similarity Score = 0.5369
Resume 746: Similarity Score = 0.5369
Resume 784: Similarity Score = 0.4379
Resume 234: Similarity Score = 0.4379
```

4. Bag Of Words:

In [34]:
```
bow_df = nlp_dataset_copy.copy()
```

In [ ]:
```
document = bow_df["Resume"].apply(lambda x: " ".join(x) if isinstance(x,
# Create BoW matrix
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(document)
bow_matrix = pd.DataFrame(
    X.toarray(),
    index=[f"Document {i + 1}"
    for i in range(len(documents))],
    columns=vectorizer.get_feature_names_out())
```

In [ ]:
```
bow_array = bow_matrix.to_numpy()

my_cv_vector = bow_array[-1].reshape(1, -1)
```

```
similarity_scores = cosine_similarity(my_cv_vector, bow_array).flatten()

top_indices = similarity_scores.argsort()[::-1][1:11]


print("Top 10 resumes similar to my resume:")
for i in top_indices:
    print(f"Resume {i + 1}: Similarity Score = {similarity_scores[i]:.4f}")
```

```
Top 10 resumes similar to my resume:
Resume 203: Similarity Score = 0.5562
Resume 612: Similarity Score = 0.5562
Resume 242: Similarity Score = 0.5562
Resume 708: Similarity Score = 0.5562
Resume 73: Similarity Score = 0.5369
Resume 489: Similarity Score = 0.5369
Resume 764: Similarity Score = 0.5369
Resume 746: Similarity Score = 0.5369
Resume 784: Similarity Score = 0.4379
Resume 234: Similarity Score = 0.4379
```

BERT

In [124…

```
bert_original_df = nlp_dataset.copy()
bert_original_df
```

Out[124…

	Category	Resume
0	Blockchain	SKILLS Bitcoin, Ethereum Solidity Hyperledger,...
1	Java Developer	Operating Systems Windows XP, 7, 10. Tools/Pac...
2	Java Developer	SKILLS: - 1) Team leading 2) Self-motivated 3)...
3	Testing	COMPUTER PROFICIENCY â€¢ Basic: MS-Office (Pow...
4	Hadoop	Skill Set: Hadoop, Map Reduce, HDFS, Hive, Sqo...
...	...	...
957	Java Developer	TECHNICAL SKILLS Skills: Java, SQL, PL/SQL, C,...
958	Python Developer	Technical Skills / Responsibilities: â€¢ Hands...
959	Arts	I Other Skills Course/Skill Name Board Year Of...
960	Sales	IT Skills: MS Office. Photoshop. SQL Server.Ed...
961	PMO	Skills Exceptional communication and networkin...

962 rows × 2 columns

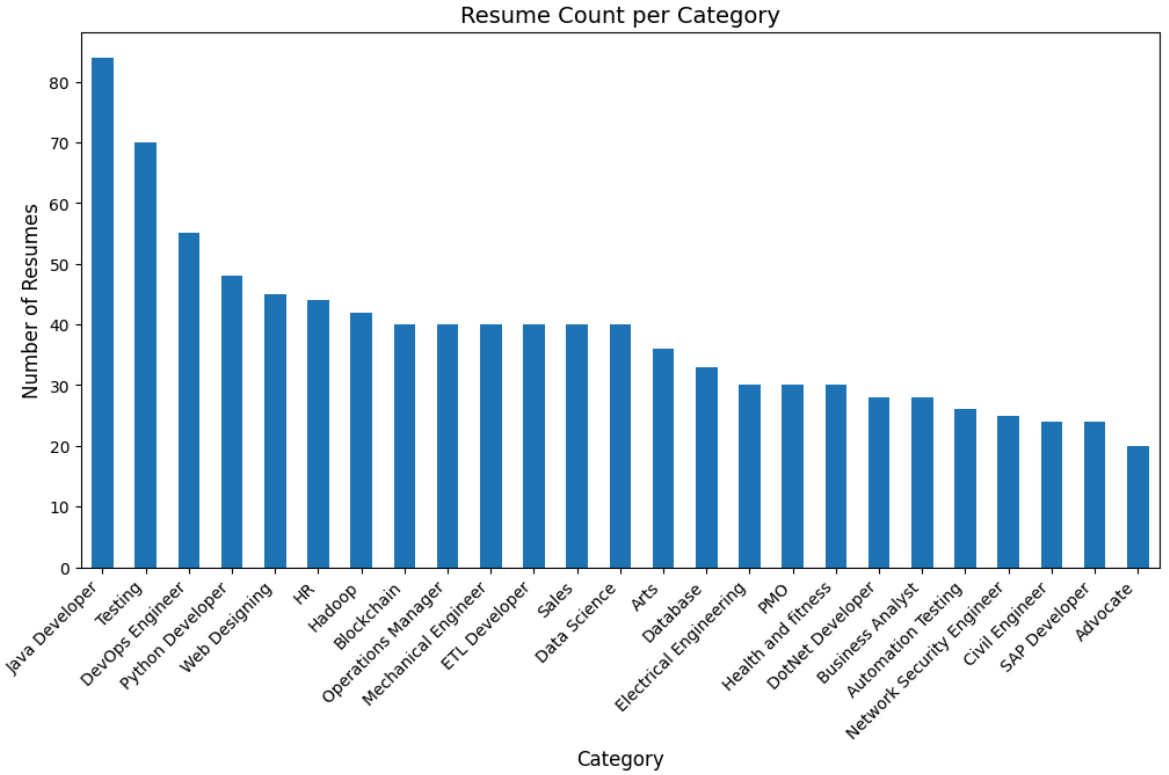
In [125…

```
unique_classes = bert_original_df["Category"].unique()
unique_classes_num = bert_original_df["Category"].nunique()

print(f"Number of unique classes: {unique_classes_num}")
print("\n")
print(f"Unique Classes: {unique_classes}")
```

```
Number of unique classes: 25


Unique Classes: ['Blockchain' 'Java Developer' 'Testing' 'Hadoop' 'Civil E
ngineer'
 'Business Analyst' 'Data Science' 'Python Developer' 'Arts' 'Database'
 'Advocate' 'Operations Manager' 'Sales' 'DotNet Developer'
 'Automation Testing' 'Web Designing' 'Electrical Engineering' 'PMO'
 'DevOps Engineer' 'ETL Developer' 'HR' 'Mechanical Engineer'
 'Health and fitness' 'Network Security Engineer' 'SAP Developer']
```

In [126…

```python
import matplotlib.pyplot as plt

class_counts = bert_original_df["Category"].value_counts()


plt.figure(figsize=(12, 6))
class_counts.plot(kind="bar")

plt.xlabel("Category", fontsize=12)
plt.ylabel("Number of Resumes", fontsize=12)
plt.title("Resume Count per Category", fontsize=14)
plt.xticks(rotation=45, ha="right")

plt.show()
```



In [127…

```python
# add my own resume

import pandas as pd

# get the input of my resume
category = input("Category: ")
resume_text = input("Resume Text: ")

# creates a new entry based on my resume
sai_entry = {
```

```
    "Category": category,
    "Resume": resume_text
}

# append it to the copied version of the original dataset
sai_entry_df = pd.DataFrame([sai_entry])
overarch_bert_df = pd.concat([bert_original_df, sai_entry_df], ignore_ind


print("Resume added successfully to the dataset copy!")
```

```
Resume added successfully to the dataset copy!
```

In [128…
```
overarch_bert_df.to_csv("/Users/saikeerthan/NYP-AI/NLP/Project/overarch-b
```

In [129…
```
overarch_bert_df
```

Out[129…

	Category	Resume
0	Blockchain	SKILLS Bitcoin, Ethereum Solidity Hyperledger,...
1	Java Developer	Operating Systems Windows XP, 7, 10. Tools/Pac...
2	Java Developer	SKILLS: - 1) Team leading 2) Self-motivated 3)...
3	Testing	COMPUTER PROFICIENCY â€¢ Basic: MS-Office (Pow...
4	Hadoop	Skill Set: Hadoop, Map Reduce, HDFS, Hive, Sqo...
...	...	...
958	Python Developer	Technical Skills / Responsibilities: â€¢ Hands...
959	Arts	I Other Skills Course/Skill Name Board Year Of...
960	Sales	IT Skills: MS Office. Photoshop. SQL Server.Ed...
961	PMO	Skills Exceptional communication and networkin...
962	Data Science	Satini Sai Keerthan AI & Data Engineer \t\t ...

963 rows × 2 columns

In [ ]:
```
## converting to lowercase
import string

# function to convert to lowercase
def convert_lowercase(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    return text


overarch_bert_df["Resume"] = overarch_bert_df["Resume"].apply(convert_low

overarch_bert_df
```

Out[ ]:

	Category	Resume
0	Blockchain	skills bitcoin ethereum solidity hyperledger b...
1	Java Developer	operating systems windows xp 7 10 toolspackage...
2	Java Developer	skills 1 team leading 2 selfmotivated 3 hard ...
3	Testing	computer proficiency â€¢ basic msoffice powerp...
4	Hadoop	skill set hadoop map reduce hdfs hive sqoop ja...
...	...	...
958	Python Developer	technical skills responsibilities â€¢ hands o...
959	Arts	i other skills courseskill name board year of ...
960	Sales	it skills ms office photoshop sql servereducat...
961	PMO	skills exceptional communication and networkin...
962	Data Science	satini sai keerthan ai data engineer \t\t 6...

963 rows × 2 columns

In [136…

```
stopwords_set = set(stopwords.words('english'))

def remove_stopwords(entries):
    tokens = entries.split()
    filtered_tokens = []

    for token in tokens:
        if token not in stopwords_set:
            filtered_tokens.append(token)

    return ' '.join(filtered_tokens)

# Apply the function to the dataset
for i in range(len(overarch_bert_df["Resume"])):
    overarch_bert_df["Resume"] = overarch_bert_df["Resume"].apply(remove_


overarch_bert_df
```

Out [136…

	Category	Resume
0	Blockchain	skills bitcoin ethereum solidity hyperledger b...
1	Java Developer	operating systems windows xp 7 10 toolspackage...
2	Java Developer	skills 1 team leading 2 selfmotivated 3 hard w...
3	Testing	computer proficiency â€¢ basic msoffice powerp...
4	Hadoop	skill set hadoop map reduce hdfs hive sqoop ja...
...	...	...
958	Python Developer	technical skills responsibilities â€¢ hands ex...
959	Arts	skills courseskill name board year passing gra...
960	Sales	skills ms office photoshop sql servereducation...
961	PMO	skills exceptional communication networking sk...
962	Data Science	satini sai keerthan ai data engineer 65 818389...

963 rows × 2 columns

In [137…

```
def no_standalone(text):
    return re.sub(r'\b[1-9]\b', '', str(text))

overarch_bert_df["Resume"] = overarch_bert_df["Resume"].apply(no_standalo

overarch_bert_df
```

Out [137…

	Category	Resume
0	Blockchain	skills bitcoin ethereum solidity hyperledger b...
1	Java Developer	operating systems windows xp 10 toolspackages...
2	Java Developer	skills team leading selfmotivated hard work...
3	Testing	computer proficiency â€¢ basic msoffice powerp...
4	Hadoop	skill set hadoop map reduce hdfs hive sqoop ja...
...	...	...
958	Python Developer	technical skills responsibilities â€¢ hands ex...
959	Arts	skills courseskill name board year passing gra...
960	Sales	skills ms office photoshop sql servereducation...
961	PMO	skills exceptional communication networking sk...
962	Data Science	satini sai keerthan ai data engineer 65 818389...

963 rows × 2 columns

BERT Pre-Processing

In [139…

```
# Tokenisation and Vectorisation:
```

```
import torch
import re
import pandas as pd
import numpy as np
from transformers import BertTokenizer, BertModel


def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'\d+', '<NUM>', text)
    text = re.sub(r'[^a-zA-Z0-9 <NUM>]', ' ', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text


overarch_bert_df['Resume'] = overarch_bert_df['Resume'].apply(preprocess_


tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
inputs = tokenizer(list(overarch_bert_df['Resume']), padding=True, trunca


input_ids = inputs['input_ids']
attention_masks = inputs['attention_mask']

torch.save(input_ids, "input_ids.pt")
torch.save(attention_masks, "attention_masks.pt")

print("Tokenization and vectorization complete. Saved input tensors.")
```

```
Tokenization and vectorization complete. Saved input tensors.
```

In [140…

```
overarch_bert_df
```

Out[140…

	Category	Resume
0	Blockchain	skills bitcoin ethereum solidity hyperledger b...
1	Java Developer	operating systems windows xp <NUM> toolspackag...
2	Java Developer	skills team leading selfmotivated hard working...
3	Testing	computer proficiency basic msoffice powerpoint...
4	Hadoop	skill set hadoop map reduce hdfs hive sqoop ja...
...	...	...
958	Python Developer	technical skills responsibilities hands experi...
959	Arts	skills courseskill name board year passing gra...
960	Sales	skills ms office photoshop sql servereducation...
961	PMO	skills exceptional communication networking sk...
962	Data Science	satini sai keerthan ai data engineer <NUM> <NU...

963 rows × 2 columns

In [141…

```
inference_df = overarch_bert_df.iloc[:-1]
```

```
inference_df
```

Out [141…

	Category	Resume
0	Blockchain	skills bitcoin ethereum solidity hyperledger b...
1	Java Developer	operating systems windows xp <NUM> toolspackag...
2	Java Developer	skills team leading selfmotivated hard working...
3	Testing	computer proficiency basic msoffice powerpoint...
4	Hadoop	skill set hadoop map reduce hdfs hive sqoop ja...
...	...	...
957	Java Developer	technical skills skills java sql plsql c c boo...
958	Python Developer	technical skills responsibilities hands experi...
959	Arts	skills courseskill name board year passing gra...
960	Sales	skills ms office photoshop sql servereducation...
961	PMO	skills exceptional communication networking sk...

962 rows × 2 columns

BERT Draft 1:

In [143…

```
import torch
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from torch.utils.data import DataLoader, TensorDataset
from transformers import BertForSequenceClassification, AdamW
from tqdm import tqdm
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report


label_encoder = LabelEncoder()
inference_df['Category'] = label_encoder.fit_transform(inference_df['Cate


input_ids = torch.load("input_ids.pt")
attention_masks = torch.load("attention_masks.pt")
labels = torch.tensor(inference_df['Category'].values)


min_length = min(len(input_ids), len(attention_masks), len(labels))

print(len(input_ids), len(attention_masks), len(labels))

input_ids = input_ids[:min_length]
attention_masks = attention_masks[:min_length]
labels = labels[:min_length]


train_inputs, val_inputs, train_masks, val_masks, train_labels, val_label
```

```
    input_ids, attention_masks, labels, test_size=0.2, random_state=42
)


batch_size = 16
train_dataset = TensorDataset(train_inputs, train_masks, train_labels)
val_dataset = TensorDataset(val_inputs, val_masks, val_labels)
train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuff
val_dataloader = DataLoader(val_dataset, batch_size=batch_size, shuffle=F


model = BertForSequenceClassification.from_pretrained("bert-base-uncased"
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

# Define optimizer
optimizer = AdamW(model.parameters(), lr=2e-5)


patience = 3
min_delta = 0.001
best_val_loss = float('inf')
epochs_no_improve = 0


train_accuracies = []
val_accuracies = []


epochs = 20
for epoch in range(epochs):
    model.train()
    loop = tqdm(train_dataloader, leave=True)

    correct_train = 0
    total_train = 0

    for batch in loop:
        optimizer.zero_grad()

        input_ids, attention_mask, labels = batch
        input_ids = input_ids.to(device)
        attention_mask = attention_mask.to(device)
        labels = labels.to(device)

        outputs = model(input_ids, attention_mask=attention_mask, labels=
        loss = outputs.loss
        loss.backward()
        optimizer.step()

        _, predicted_labels = torch.max(outputs.logits, 1)
        correct_train += (predicted_labels == labels).sum().item()
        total_train += labels.size(0)

        loop.set_description(f"Epoch {epoch+1}")
        loop.set_postfix(loss=loss.item())

    train_accuracy = correct_train / total_train
    train_accuracies.append(train_accuracy)
```

```
    model.eval()
    predictions = []
    true_labels = []
    total_val_loss = 0

    with torch.no_grad():
        for batch in val_dataloader:
            input_ids, attention_mask, labels = batch
            input_ids = input_ids.to(device)
            attention_mask = attention_mask.to(device)
            labels = labels.to(device)

            outputs = model(input_ids, attention_mask=attention_mask, lab
            loss = outputs.loss
            total_val_loss += loss.item()

            _, predicted_labels = torch.max(outputs.logits, 1)

            predictions.extend(predicted_labels.cpu().numpy())
            true_labels.extend(labels.cpu().numpy())

    val_accuracy = accuracy_score(true_labels, predictions)
    val_accuracies.append(val_accuracy)
    avg_val_loss = total_val_loss / len(val_dataloader)


    print("-" * 65)
    print(f"Epoch {epoch+1}: Train Accuracy: {train_accuracy:.4f}, Valida
    print(f"Epoch {epoch+1}: Validation Loss: {avg_val_loss:.4f}")
    print("-" * 65)


    if avg_val_loss < best_val_loss - min_delta:
        best_val_loss = avg_val_loss
        epochs_no_improve = 0
    else:
        epochs_no_improve += 1
        if epochs_no_improve >= patience:
            print("Early stopping triggered. Training stopped.")
            break

model.save_pretrained("bert_resume_classifier")
tokenizer.save_pretrained("bert_resume_classifier")
print("Model training complete. Model saved!")
```

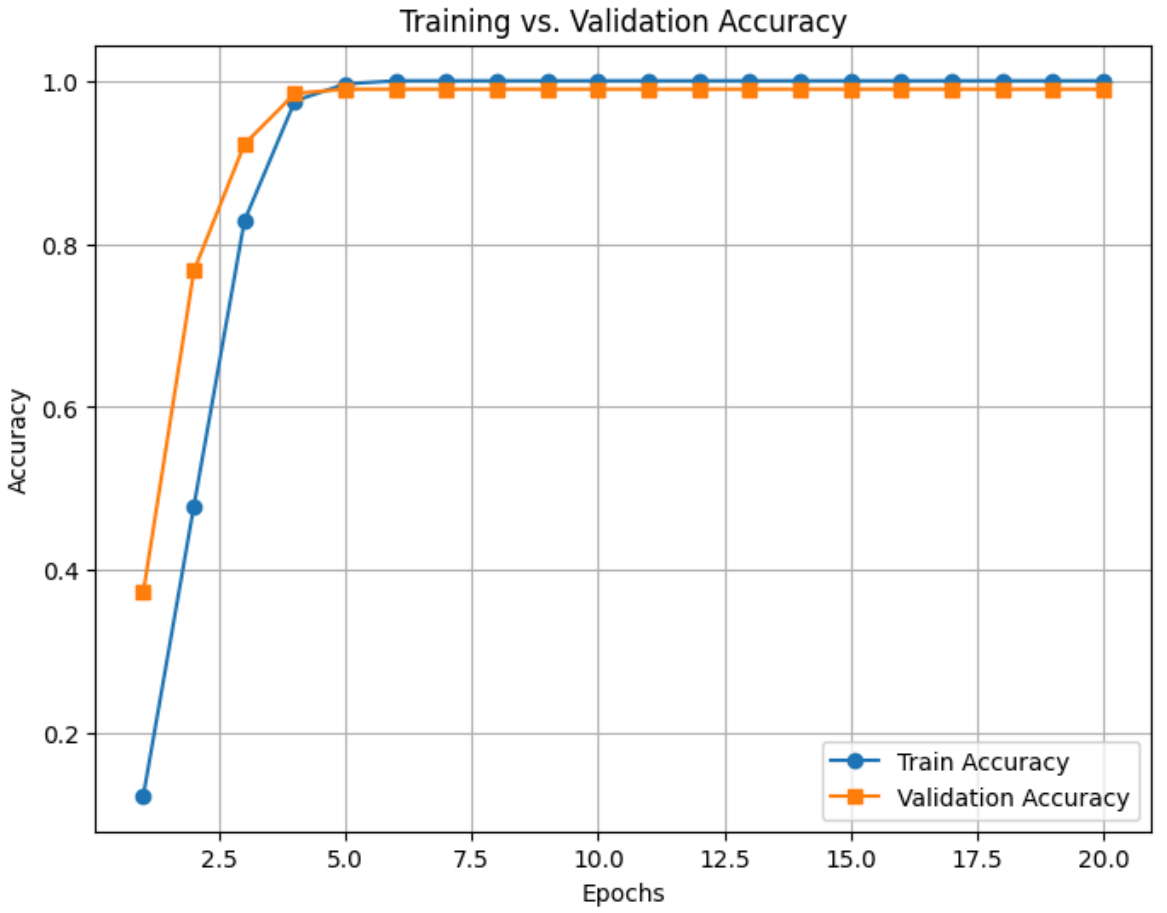
```
/var/folders/wv/lgt7rwvx5dn_v1y74zftwswh0000gn/T/ipykernel_44045/247052085
3.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  inference_df['Category'] = label_encoder.fit_transform(inference_df['Cat
egory'])
963 963 962
```

```
Some weights of BertForSequenceClassification were not initialized from th
e model checkpoint at bert-base-uncased and are newly initialized: ['class
ifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to u
se it for predictions and inference.
/Users/saikeerthan/NYP-AI/NLP/nlp_project/lib/python3.10/site-packages/tra
nsformers/optimization.py:588: FutureWarning: This implementation of AdamW
is deprecated and will be removed in a future version. Use the PyTorch imp
lementation torch.optim.AdamW instead, or set `no_deprecation_warning=True
` to disable this warning
  warnings.warn(
Epoch 1: 100%|██████████| 49/49 [02:18<00:00,  2.82s/it, loss=3.2]
----------------------------------------------------------------
Epoch 1: Train Accuracy: 0.1235, Validation Accuracy: 0.3731
Epoch 1: Validation Loss: 2.7167
----------------------------------------------------------------
Epoch 2: 100%|██████████| 49/49 [03:16<00:00,  4.02s/it, loss=1.77]
----------------------------------------------------------------
Epoch 2: Train Accuracy: 0.4785, Validation Accuracy: 0.7668
Epoch 2: Validation Loss: 1.8768
----------------------------------------------------------------
Epoch 3: 100%|██████████| 49/49 [03:03<00:00,  3.75s/it, loss=1.23]
----------------------------------------------------------------
Epoch 3: Train Accuracy: 0.8283, Validation Accuracy: 0.9223
Epoch 3: Validation Loss: 1.1629
----------------------------------------------------------------
Epoch 4: 100%|██████████| 49/49 [03:14<00:00,  3.97s/it, loss=1.08]
----------------------------------------------------------------
Epoch 4: Train Accuracy: 0.9753, Validation Accuracy: 0.9845
Epoch 4: Validation Loss: 0.7029
----------------------------------------------------------------
Epoch 5: 100%|██████████| 49/49 [03:05<00:00,  3.79s/it, loss=0.27]
----------------------------------------------------------------
Epoch 5: Train Accuracy: 0.9961, Validation Accuracy: 0.9896
Epoch 5: Validation Loss: 0.4343
----------------------------------------------------------------
Epoch 6: 100%|██████████| 49/49 [02:56<00:00,  3.60s/it, loss=0.329]
----------------------------------------------------------------
Epoch 6: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 6: Validation Loss: 0.3030
----------------------------------------------------------------
Epoch 7: 100%|██████████| 49/49 [02:23<00:00,  2.92s/it, loss=0.248]
----------------------------------------------------------------
Epoch 7: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 7: Validation Loss: 0.2228
----------------------------------------------------------------
Epoch 8: 100%|██████████| 49/49 [03:07<00:00,  3.83s/it, loss=0.137]
----------------------------------------------------------------
Epoch 8: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 8: Validation Loss: 0.1771
----------------------------------------------------------------
Epoch 9: 100%|██████████| 49/49 [03:03<00:00,  3.75s/it, loss=0.161]
----------------------------------------------------------------
Epoch 9: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 9: Validation Loss: 0.1457
----------------------------------------------------------------
Epoch 10: 100%|██████████| 49/49 [03:18<00:00,  4.06s/it, loss=0.132]
```

```
----------------------------------------------------------------
Epoch 10: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 10: Validation Loss: 0.1264
----------------------------------------------------------------
Epoch 11: 100%|██████████| 49/49 [03:06<00:00,  3.82s/it, loss=0.143]
----------------------------------------------------------------
Epoch 11: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 11: Validation Loss: 0.1124
----------------------------------------------------------------
Epoch 12: 100%|██████████| 49/49 [02:26<00:00,  2.98s/it, loss=0.0995]
----------------------------------------------------------------
Epoch 12: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 12: Validation Loss: 0.1012
----------------------------------------------------------------
Epoch 13: 100%|██████████| 49/49 [02:12<00:00,  2.70s/it, loss=0.0674]
----------------------------------------------------------------
Epoch 13: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 13: Validation Loss: 0.0940
----------------------------------------------------------------
Epoch 14: 100%|██████████| 49/49 [02:11<00:00,  2.67s/it, loss=0.0881]
----------------------------------------------------------------
Epoch 14: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 14: Validation Loss: 0.0882
----------------------------------------------------------------
Epoch 15: 100%|██████████| 49/49 [02:12<00:00,  2.71s/it, loss=0.026]
----------------------------------------------------------------
Epoch 15: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 15: Validation Loss: 0.0833
----------------------------------------------------------------
Epoch 16: 100%|██████████| 49/49 [02:02<00:00,  2.51s/it, loss=0.0527]
----------------------------------------------------------------
Epoch 16: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 16: Validation Loss: 0.0767
----------------------------------------------------------------
Epoch 17: 100%|██████████| 49/49 [01:53<00:00,  2.33s/it, loss=0.0392]
----------------------------------------------------------------
Epoch 17: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 17: Validation Loss: 0.0757
----------------------------------------------------------------
Epoch 18: 100%|██████████| 49/49 [01:50<00:00,  2.26s/it, loss=0.0316]
----------------------------------------------------------------
Epoch 18: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 18: Validation Loss: 0.0729
----------------------------------------------------------------
Epoch 19: 100%|██████████| 49/49 [02:00<00:00,  2.47s/it, loss=0.0425]
----------------------------------------------------------------
Epoch 19: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 19: Validation Loss: 0.0678
----------------------------------------------------------------
Epoch 20: 100%|██████████| 49/49 [02:04<00:00,  2.55s/it, loss=0.0238]
----------------------------------------------------------------
Epoch 20: Train Accuracy: 1.0000, Validation Accuracy: 0.9896
Epoch 20: Validation Loss: 0.0683
----------------------------------------------------------------
Model training complete. Model saved!
```

In [148…

```
# Plot training & validation accuracy
plt.figure(figsize=(8, 6))
plt.plot(range(1, len(train_accuracies) + 1), train_accuracies, label="Tr
plt.plot(range(1, len(val_accuracies) + 1), val_accuracies, label="Valida
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Training vs. Validation Accuracy")
plt.legend()
plt.grid()
plt.show()
```



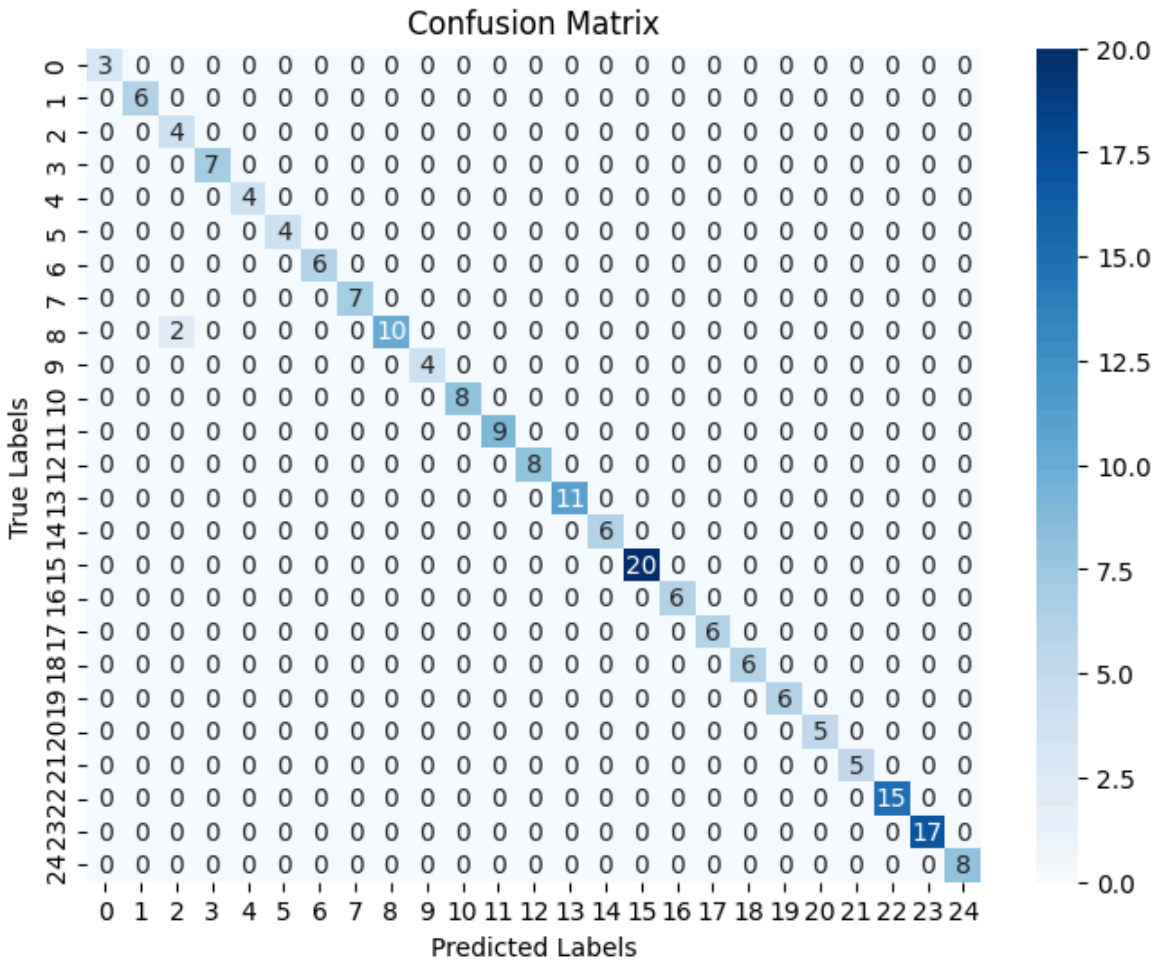
In [ ]:

In [ ]:

```
import torch
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt


model.eval()


predictions = []
true_labels = []

with torch.no_grad():
    for batch in val_dataloader:
        input_ids, attention_mask, labels = batch
        input_ids = input_ids.to(device)
        attention_mask = attention_mask.to(device)
        labels = labels.to(device)
```

```
        outputs = model(input_ids, attention_mask=attention_mask)
        _, predicted_labels = torch.max(outputs.logits, 1)

        predictions.extend(predicted_labels.cpu().numpy())
        true_labels.extend(labels.cpu().numpy())


# Convert target names to strings
class_names = [str(label) for label in label_encoder.classes_]

# Print Classification Report
print("Classification Report:")
print(classification_report(true_labels, predictions, target_names=class_



conf_matrix = confusion_matrix(true_labels, predictions)

# Plot Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=l
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         6
           2       0.67      1.00      0.80         4
           3       1.00      1.00      1.00         7
           4       1.00      1.00      1.00         4
           5       1.00      1.00      1.00         4
           6       1.00      1.00      1.00         6
           7       1.00      1.00      1.00         7
           8       1.00      0.83      0.91        12
           9       1.00      1.00      1.00         4
          10       1.00      1.00      1.00         8
          11       1.00      1.00      1.00         9
          12       1.00      1.00      1.00         8
          13       1.00      1.00      1.00        11
          14       1.00      1.00      1.00         6
          15       1.00      1.00      1.00        20
          16       1.00      1.00      1.00         6
          17       1.00      1.00      1.00         6
          18       1.00      1.00      1.00         6
          19       1.00      1.00      1.00         6
          20       1.00      1.00      1.00         5
          21       1.00      1.00      1.00         5
          22       1.00      1.00      1.00        15
          23       1.00      1.00      1.00        17
          24       1.00      1.00      1.00         8

    accuracy                           0.99       193
   macro avg       0.99      0.99      0.99       193
weighted avg       0.99      0.99      0.99       193
```



In [153…
```
test_bert_df = overarch_bert_df.copy()

test_bert_df
```

Out[153…

	Category	Resume
0	Blockchain	skills bitcoin ethereum solidity hyperledger b...
1	Java Developer	operating systems windows xp <NUM> toolspackag...
2	Java Developer	skills team leading selfmotivated hard working...
3	Testing	computer proficiency basic msoffice powerpoint...
4	Hadoop	skill set hadoop map reduce hdfs hive sqoop ja...
...	...	...
958	Python Developer	technical skills responsibilities hands experi...
959	Arts	skills courseskill name board year passing gra...
960	Sales	skills ms office photoshop sql servereducation...
961	PMO	skills exceptional communication networking sk...
962	Data Science	satini sai keerthan ai data engineer <NUM> <NU...

963 rows × 2 columns

In [154…
```
test_bert_df = test_bert_df["Resume"].iloc[-1]
```

```
print(test_bert_df)
```

```
satini sai keerthan ai data engineer <NUM> <NUM> saikeerthan<NUM>gmailcom
choa chu kang singapore objective fresh graduate diploma ai data engineeri
ng nanyang polytechnic equipped strong data visualization data analytics s
kills proficient tools power bi tableau python sql proven experience desig
ning interactive dashboards etl pipelines adept leveraging analytical insi
ghts drive decisionmaking improve efficiency seeking apply technical inter
personal skills data analyst good job creations singapore pte ltd experien
ce data engineering intern xyz tech solutions singapore september <NUM> ap
ril <NUM> overlooked migration legacy data systems cloudbased solutions au
tomated datacleaning scripts python reducing manual errors <NUM> assisted
creation sql queries databases data extraction analysis collaborated teams
design kpi dashboard improving reporting speed <NUM> key skills data prepa
ration visualisation programming languages data modelling etl development
machine learning models communication teamwork problemsolving education na
nyang polytechnic ang mo kio singapore <NUM> diploma ai data engineer rele
vant modules machine learning data analytics business intelligence tools c
loud computing year sem shopping console python project year sem mitigate
dengue disease data science project utilizing powerbi year sem predict fac
tory environment mqttsql project arduino leadership lead team members data
science project mitigate dengue cases data analytics software like powerbi
presented findings stakeholders demonstrating effective communication team
work delegated tasks monitored progress ensuring timely completion exceedi
ng project goals <NUM> certifications microsoft certified azure ai fundame
ntals <NUM> nvidia certification awards dean s list outstanding academic p
erformance <NUM> languages fluent english telugu conversational hindi
```

In [ ]:
```
print("Label Encoder Classes:", label_encoder.classes_)
```

```
Label Encoder Classes: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
17 18 19 20 21 22 23
 24]
```

In [172…
```
import torch
import pandas as pd
from transformers import BertTokenizer, BertForSequenceClassification
from sklearn.preprocessing import LabelEncoder


model = BertForSequenceClassification.from_pretrained("bert_resume_classi
tokenizer = BertTokenizer.from_pretrained("bert_resume_classifier")


device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
model.eval()


my_resume_text = test_bert_df


inputs = tokenizer(my_resume_text, padding=True, truncation=True, max_len


input_ids = inputs["input_ids"].to(device)
attention_mask = inputs["attention_mask"].to(device)


with torch.no_grad():
    outputs = model(input_ids, attention_mask=attention_mask)
```

```
    predicted_label_index = torch.argmax(outputs.logits, dim=1).item()


print("Actual Class Names in Label Encoder:")
print(label_encoder.classes_)


df = pd.read_csv("/Users/saikeerthan/NYP-AI/NLP/Project/SATINI SAI KEERTH
unique_categories = sorted(df["Category"].unique())


label_encoder.classes_ = unique_categories


class_names = list(label_encoder.classes_)


if 0 <= predicted_label_index < len(class_names):
    predicted_category = class_names[predicted_label_index]
else:
    raise ValueError(f"Error: Predicted index {predicted_label_index} is


print(f"Predicted Category: {predicted_category}")
print(f"Predicted Index: {predicted_label_index} -> {predicted_category}"
```

```
Actual Class Names in Label Encoder:
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24]
Predicted Category: Data Science
Predicted Index: 6 -> Data Science
```

BERT Draft 2:

What has changed in this fine tuning round?:

1. In this fine tuning round, we will be balancing every class such that they have the same number of resumes per each class.
2. We will also be reducing the number of epochs to 4, preventing any overfitting, as a result we will also remove early stopping as it is no longer needed due to the less number of epochs.
3. A new test dataset will be used, to make sure if the BERT model is able to classify every category properly, we will then improve on the category that it is not able to classify properly

In [174…

```
bert_original_df.to_csv("bert_original.csv", index=False)
```

In [ ]:

```
from sklearn.utils import resample


df = pd.read_csv("/Users/saikeerthan/NYP-AI/NLP/Project/bert_original.csv


print("Original Class Distribution:\n", df["Category"].value_counts())


max_samples = df["Category"].value_counts().max()
```

```
better_original_bert = pd.DataFrame()


for category in df["Category"].unique():
    category_df = df[df["Category"] == category]


    category_upsampled = resample(category_df, replace=True, n_samples=ma


    better_original_bert = pd.concat([better_original_bert, category_upsa

better_original_bert = better_original_bert.sample(frac=1, random_state=4


better_original_bert.to_csv("balanced_dataset.csv", index=False)
```

```
Original Class Distribution:
 Category
Java Developer                84
Testing                       70
DevOps Engineer               55
Python Developer              48
Web Designing                 45
HR                            44
Hadoop                        42
Blockchain                    40
Operations Manager            40
Mechanical Engineer           40
ETL Developer                 40
Sales                         40
Data Science                  40
Arts                          36
Database                      33
Electrical Engineering        30
PMO                           30
Health and fitness            30
DotNet Developer              28
Business Analyst              28
Automation Testing            26
Network Security Engineer     25
Civil Engineer                24
SAP Developer                 24
Advocate                      20
Name: count, dtype: int64
```

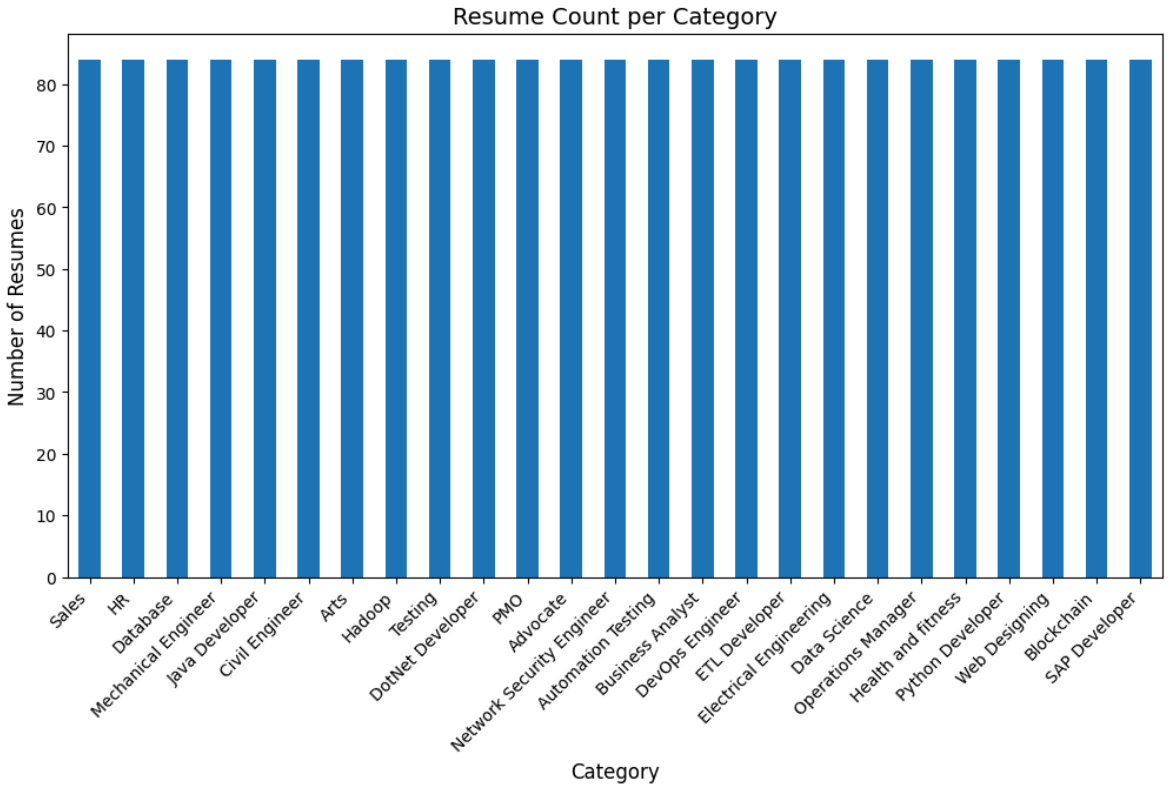
In [179…

```
print("Balanced Class Distribution:\n", better_original_bert["Category"].
```

```
Balanced Class Distribution:
 Category
Sales                          84
HR                             84
Database                       84
Mechanical Engineer            84
Java Developer                 84
Civil Engineer                 84
Arts                           84
Hadoop                         84
Testing                        84
DotNet Developer               84
PMO                            84
Advocate                       84
Network Security Engineer      84
Automation Testing             84
Business Analyst               84
DevOps Engineer                84
ETL Developer                  84
Electrical Engineering         84
Data Science                   84
Operations Manager             84
Health and fitness             84
Python Developer               84
Web Designing                  84
Blockchain                     84
SAP Developer                  84
Name: count, dtype: int64
```

In [180…

```
class_counts = better_original_bert["Category"].value_counts()


plt.figure(figsize=(12, 6))
class_counts.plot(kind="bar")


plt.xlabel("Category", fontsize=12)
plt.ylabel("Number of Resumes", fontsize=12)
plt.title("Resume Count per Category", fontsize=14)
plt.xticks(rotation=45, ha="right")


plt.show()
```



Text Pre-Processing Steps:

In [189…

```
# get the input of my resume
category = input("Category: ")
resume_text = input("Resume Text: ")

# creates a new entry based on my resume
sai_entry = {
    "Category": category,
    "Resume": resume_text
}

# append it to the copied version of the original dataset
sai_entry_df = pd.DataFrame([sai_entry])
better_original_bert = pd.concat([better_original_bert, sai_entry_df], ig


print("Resume added successfully to the dataset copy!")
```

```
Resume added successfully to the dataset copy!
```

In [190…

```
better_original_bert
```

Out[190…

	Category	Resume
0	Sales	key skills â€¢ planning strategizing â€¢ prese...
1	Automation Testing	technical skills summary completed corporate t...
2	Blockchain	skills bitcoin ethereum solidity hyperledger b...
3	Web Designing	skills languages c basic java basic web techno...
4	Python Developer	â€¢ operating systems windows â€¢ others ms ex...
...	...	...
2096	DotNet Developer	technical skills â programming languages cne...
2097	DotNet Developer	education details january 2014 education detai...
2098	Web Designing	technical skills web technologies angular js h...
2099	Advocate	skills â€¢ knows english native speaker ielts ...
2100	Data Science	Satini Sai Keerthan AI & Data Engineer \t\t ...

2101 rows × 2 columns

In [ ]:

```
## converting to lowercase
import string

# function to convert to lowercase
def convert_lowercase(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    return text


better_original_bert["Resume"] = better_original_bert["Resume"].apply(con

better_original_bert
```

Out[ ]:

	Category	Resume
0	Sales	key skills â€¢ planning strategizing â€¢ prese...
1	Automation Testing	technical skills summary completed corporate t...
2	Blockchain	skills bitcoin ethereum solidity hyperledger b...
3	Web Designing	skills languages c basic java basic web techno...
4	Python Developer	â€¢ operating systems windows â€¢ others ms ex...
...	...	...
2096	DotNet Developer	technical skills â programming languages cne...
2097	DotNet Developer	education details january 2014 education detai...
2098	Web Designing	technical skills web technologies angular js h...
2099	Advocate	skills â€¢ knows english native speaker ielts ...
2100	Data Science	satini sai keerthan ai data engineer \t\t 6...

2101 rows × 2 columns

In [ ]:

```
stopwords_set = set(stopwords.words('english'))

def remove_stopwords(entries):
    tokens = entries.split()
    filtered_tokens = []

    for token in tokens:
        if token not in stopwords_set:
            filtered_tokens.append(token)

    return ' '.join(filtered_tokens)


for i in range(len(overarch_bert_df["Resume"])):
    better_original_bert["Resume"] = better_original_bert["Resume"].apply


better_original_bert
```

Out[ ]:

	Category	Resume
0	Sales	key skills â€¢ planning strategizing â€¢ prese...
1	Automation Testing	technical skills summary completed corporate t...
2	Blockchain	skills bitcoin ethereum solidity hyperledger b...
3	Web Designing	skills languages c basic java basic web techno...
4	Python Developer	â€¢ operating systems windows â€¢ others ms ex...
...	...	...
2096	DotNet Developer	technical skills â programming languages cne...
2097	DotNet Developer	education details january 2014 education detai...
2098	Web Designing	technical skills web technologies angular js h...
2099	Advocate	skills â€¢ knows english native speaker ielts ...
2100	Data Science	satini sai keerthan ai data engineer 65 818389...

2101 rows × 2 columns

In [194...

```
def no_standalone(text):
    return re.sub(r'\b[1-9]\b', '', str(text))

better_original_bert["Resume"] = better_original_bert["Resume"].apply(no_

better_original_bert
```

Out[194...

	Category	Resume
0	Sales	key skills â€¢ planning strategizing â€¢ prese...
1	Automation Testing	technical skills summary completed corporate t...
2	Blockchain	skills bitcoin ethereum solidity hyperledger b...
3	Web Designing	skills languages c basic java basic web techno...
4	Python Developer	â€¢ operating systems windows â€¢ others ms ex...
...	...	...
2096	DotNet Developer	technical skills â programming languages cne...
2097	DotNet Developer	education details january 2014 education detai...
2098	Web Designing	technical skills web technologies angular js h...
2099	Advocate	skills â€¢ knows english native speaker ielts ...
2100	Data Science	satini sai keerthan ai data engineer 65 818389...

2101 rows × 2 columns

BERT 2nd Draft Pre-Proecssing:

In [ ]:

```
# Tokenisation and Vectorisation:
```

```
import torch
import re
import pandas as pd
import numpy as np
from transformers import BertTokenizer, BertModel



def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'\d+', '<NUM>', text)
    text = re.sub(r'[^a-zA-Z0-9 <NUM>]', ' ', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text


better_original_bert['Resume'] = better_original_bert['Resume'].apply(pre


tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
inputs = tokenizer(list(better_original_bert['Resume']), padding=True, tr


input_ids = inputs['input_ids']
attention_masks = inputs['attention_mask']


torch.save(input_ids, "input_ids.pt")
torch.save(attention_masks, "attention_masks.pt")

print("Tokenization and vectorization complete. Saved input tensors.")
```

```
Tokenization and vectorization complete. Saved input tensors.
```

In [196…

```
better_original_bert
```

Out[196…

	Category	Resume
0	Sales	key skills planning strategizing presentation ...
1	Automation Testing	technical skills summary completed corporate t...
2	Blockchain	skills bitcoin ethereum solidity hyperledger b...
3	Web Designing	skills languages c basic java basic web techno...
4	Python Developer	operating systems windows others ms excel ms o...
...	...	...
2096	DotNet Developer	technical skills programming languages cnet we...
2097	DotNet Developer	education details january <NUM> education deta...
2098	Web Designing	technical skills web technologies angular js h...
2099	Advocate	skills knows english native speaker ielts over...
2100	Data Science	satini sai keerthan ai data engineer <NUM> <NU...

2101 rows × 2 columns

In [197…

```
inference_df2 = better_original_bert.iloc[:-1]

inference_df2
```

Out[197…

	Category	Resume
0	Sales	key skills planning strategizing presentation ...
1	Automation Testing	technical skills summary completed corporate t...
2	Blockchain	skills bitcoin ethereum solidity hyperledger b...
3	Web Designing	skills languages c basic java basic web techno...
4	Python Developer	operating systems windows others ms excel ms o...
...	...	...
2095	ETL Developer	technicalproficiencies db oracle <NUM>g domain...
2096	DotNet Developer	technical skills programming languages cnet we...
2097	DotNet Developer	education details january <NUM> education deta...
2098	Web Designing	technical skills web technologies angular js h...
2099	Advocate	skills knows english native speaker ielts over...

2100 rows × 2 columns

BERT Draft 2:

In [ ]:

```
import torch
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from torch.utils.data import DataLoader, TensorDataset
from transformers import BertForSequenceClassification, AdamW
from tqdm import tqdm
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
from transformers import get_scheduler


num_training_steps = len(train_dataloader) * epochs
lr_scheduler = get_scheduler(
    name="linear", optimizer=optimizer, num_warmup_steps=0, num_training_
)


label_encoder = LabelEncoder()
inference_df2['Category'] = label_encoder.fit_transform(inference_df2['Ca


input_ids = torch.load("input_ids.pt")
attention_masks = torch.load("attention_masks.pt")
labels = torch.tensor(inference_df2['Category'].values)


min_length = min(len(input_ids), len(attention_masks), len(labels))
```

```
print(len(input_ids), len(attention_masks), len(labels))

input_ids = input_ids[:min_length]
attention_masks = attention_masks[:min_length]
labels = labels[:min_length]


train_inputs, val_inputs, train_masks, val_masks, train_labels, val_label
    input_ids, attention_masks, labels, test_size=0.2, random_state=42
)


batch_size = 32
train_dataset = TensorDataset(train_inputs, train_masks, train_labels)
val_dataset = TensorDataset(val_inputs, val_masks, val_labels)
train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuff
val_dataloader = DataLoader(val_dataset, batch_size=batch_size, shuffle=F


model = BertForSequenceClassification.from_pretrained("bert-base-uncased"
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)


optimizer = AdamW(model.parameters(), lr=2e-5)



train_accuracies = []
val_accuracies = []


epochs = 6
for epoch in range(epochs):
    model.train()
    loop = tqdm(train_dataloader, leave=True)

    correct_train = 0
    total_train = 0

    for batch in loop:
        optimizer.zero_grad()

        input_ids, attention_mask, labels = batch
        input_ids = input_ids.to(device)
        attention_mask = attention_mask.to(device)
        labels = labels.to(device)

        outputs = model(input_ids, attention_mask=attention_mask, labels=
        loss = outputs.loss
        loss.backward()
        optimizer.step()
        lr_scheduler.step()

        _, predicted_labels = torch.max(outputs.logits, 1)
        correct_train += (predicted_labels == labels).sum().item()
        total_train += labels.size(0)

        loop.set_description(f"Epoch {epoch+1}")
```

```
        loop.set_postfix(loss=loss.item())

    train_accuracy = correct_train / total_train
    train_accuracies.append(train_accuracy)


    model.eval()
    predictions = []
    true_labels = []
    total_val_loss = 0

    with torch.no_grad():
        for batch in val_dataloader:
            input_ids, attention_mask, labels = batch
            input_ids = input_ids.to(device)
            attention_mask = attention_mask.to(device)
            labels = labels.to(device)

            outputs = model(input_ids, attention_mask=attention_mask, lab
            loss = outputs.loss
            total_val_loss += loss.item()

            _, predicted_labels = torch.max(outputs.logits, 1)

            predictions.extend(predicted_labels.cpu().numpy())
            true_labels.extend(labels.cpu().numpy())

    val_accuracy = accuracy_score(true_labels, predictions)
    val_accuracies.append(val_accuracy)
    avg_val_loss = total_val_loss / len(val_dataloader)


    print("-" * 65)
    print(f"Epoch {epoch+1}: Train Accuracy: {train_accuracy:.4f}, Valida
    print(f"Epoch {epoch+1}: Validation Loss: {avg_val_loss:.4f}")
    print("-" * 65)

model.save_pretrained("bert_resume_classifier2")
tokenizer.save_pretrained("bert_resume_classifier2")
print("Model training complete. Model saved!")
```

```
/var/folders/wv/lgt7rwvx5dn_v1y74zftwswh0000gn/T/ipykernel_44045/193959847
3.py:21: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  inference_df2['Category'] = label_encoder.fit_transform(inference_df2['C
ategory'])
2101 2101 2100
```

```
Some weights of BertForSequenceClassification were not initialized from th
e model checkpoint at bert-base-uncased and are newly initialized: ['class
ifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to u
se it for predictions and inference.
/Users/saikeerthan/NYP-AI/NLP/nlp_project/lib/python3.10/site-packages/tra
nsformers/optimization.py:588: FutureWarning: This implementation of AdamW
is deprecated and will be removed in a future version. Use the PyTorch imp
lementation torch.optim.AdamW instead, or set `no_deprecation_warning=True
` to disable this warning
  warnings.warn(
  0%|          | 0/53 [00:00<?, ?it/s]/Users/saikeerthan/NYP-AI/NLP/nlp_pr
oject/lib/python3.10/site-packages/torch/optim/lr_scheduler.py:227: UserWa
rning: Detected call of `lr_scheduler.step()` before `optimizer.step()`. I
n PyTorch 1.1.0 and later, you should call them in the opposite order: `op
timizer.step()` before `lr_scheduler.step()`.  Failure to do this will res
ult in PyTorch skipping the first value of the learning rate schedule. See
more details at https://pytorch.org/docs/stable/optim.html#how-to-adjust-l
earning-rate
  warnings.warn(
Epoch 1: 100%|██████████| 53/53 [04:34<00:00,  5.19s/it, loss=2.48]
----------------------------------------------------------------
Epoch 1: Train Accuracy: 0.2357, Validation Accuracy: 0.6143
Epoch 1: Validation Loss: 2.4195
----------------------------------------------------------------
Epoch 2: 100%|██████████| 53/53 [05:17<00:00,  5.99s/it, loss=1.6]
----------------------------------------------------------------
Epoch 2: Train Accuracy: 0.8095, Validation Accuracy: 0.9857
Epoch 2: Validation Loss: 1.3386
----------------------------------------------------------------
Epoch 3: 100%|██████████| 53/53 [05:30<00:00,  6.24s/it, loss=0.81]
----------------------------------------------------------------
Epoch 3: Train Accuracy: 0.9923, Validation Accuracy: 1.0000
Epoch 3: Validation Loss: 0.6258
----------------------------------------------------------------
Epoch 4: 100%|██████████| 53/53 [05:47<00:00,  6.55s/it, loss=0.368]
----------------------------------------------------------------
Epoch 4: Train Accuracy: 0.9994, Validation Accuracy: 1.0000
Epoch 4: Validation Loss: 0.3158
----------------------------------------------------------------
Epoch 5: 100%|██████████| 53/53 [05:56<00:00,  6.73s/it, loss=0.214]
----------------------------------------------------------------
Epoch 5: Train Accuracy: 1.0000, Validation Accuracy: 1.0000
Epoch 5: Validation Loss: 0.1851
----------------------------------------------------------------
Epoch 6: 100%|██████████| 53/53 [06:03<00:00,  6.86s/it, loss=0.164]
----------------------------------------------------------------
Epoch 6: Train Accuracy: 1.0000, Validation Accuracy: 1.0000
Epoch 6: Validation Loss: 0.1260
----------------------------------------------------------------
Model training complete. Model saved!
```

In [ ]:

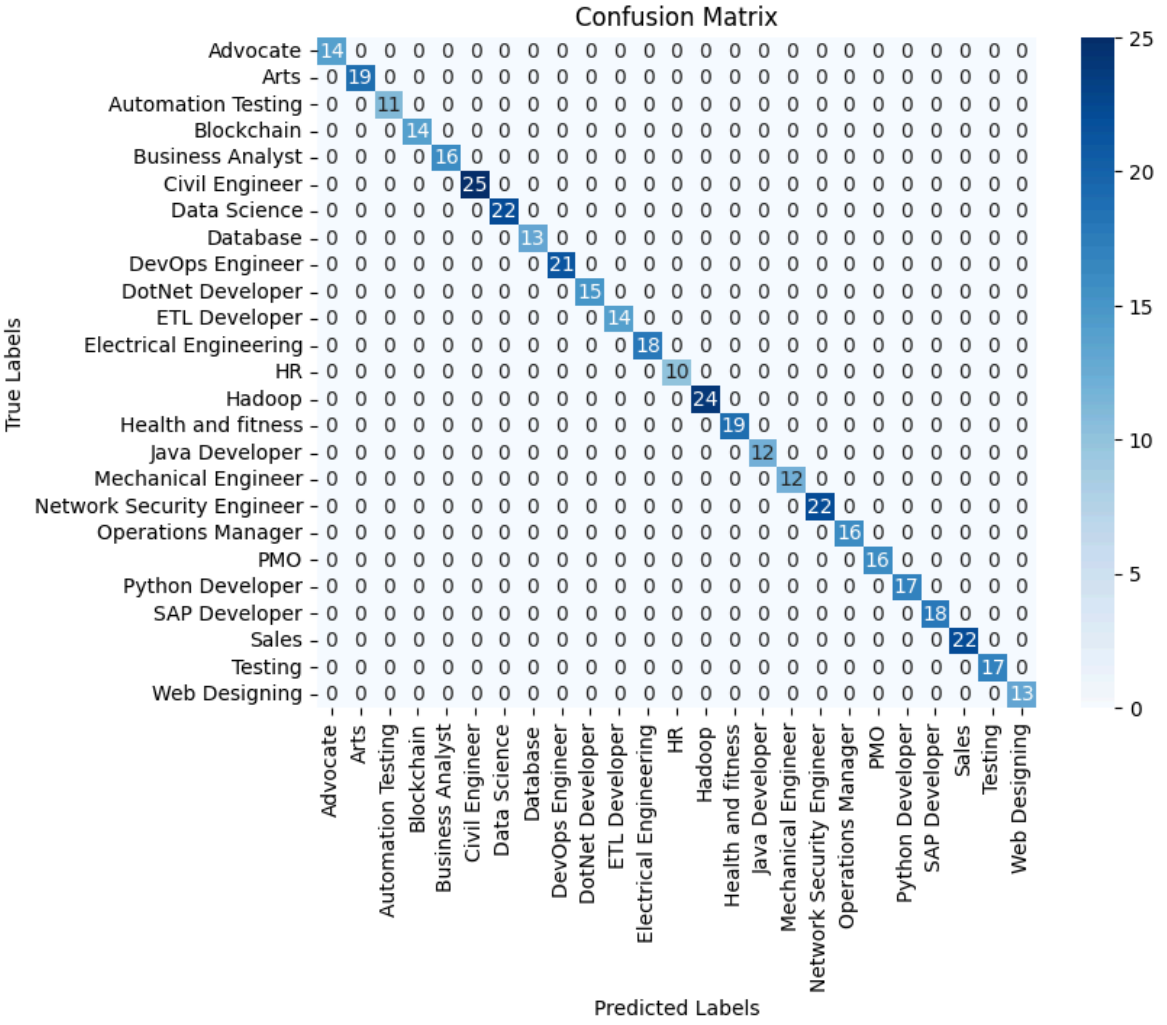
```
plt.figure(figsize=(8, 6))
plt.plot(range(1, len(train_accuracies) + 1), train_accuracies, label="Tr
plt.plot(range(1, len(val_accuracies) + 1), val_accuracies, label="Valida
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Training vs. Validation Accuracy")
```

```
plt.legend()
plt.grid()
plt.show()
```

In [ ]:
```
import torch
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

model = BertForSequenceClassification.from_pretrained("bert_resume_classi
tokenizer = BertTokenizer.from_pretrained("bert_resume_classifier2")

model.eval()

predictions = []
true_labels = []

with torch.no_grad():
    for batch in val_dataloader:
        input_ids, attention_mask, labels = batch
        input_ids = input_ids.to(device)
        attention_mask = attention_mask.to(device)
        labels = labels.to(device)

        outputs = model(input_ids, attention_mask=attention_mask)
        _, predicted_labels = torch.max(outputs.logits, 1)

        predictions.extend(predicted_labels.cpu().numpy())
        true_labels.extend(labels.cpu().numpy())
```

```
class_names = [str(label) for label in label_encoder.classes_]


print("Classification Report:")
print(classification_report(true_labels, predictions, target_names=class_



conf_matrix = confusion_matrix(true_labels, predictions)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=l
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```

```
Classification Report:
                          precision    recall  f1-score   support

                Advocate       1.00      1.00      1.00        14
                    Arts       1.00      1.00      1.00        19
      Automation Testing       1.00      1.00      1.00        11
              Blockchain       1.00      1.00      1.00        14
        Business Analyst       1.00      1.00      1.00        16
          Civil Engineer       1.00      1.00      1.00        25
            Data Science       1.00      1.00      1.00        22
                Database       1.00      1.00      1.00        13
         DevOps Engineer       1.00      1.00      1.00        21
        DotNet Developer       1.00      1.00      1.00        15
          ETL Developer       1.00      1.00      1.00        14
   Electrical Engineering       1.00      1.00      1.00        18
                      HR       1.00      1.00      1.00        10
                  Hadoop       1.00      1.00      1.00        24
      Health and fitness       1.00      1.00      1.00        19
          Java Developer       1.00      1.00      1.00        12
      Mechanical Engineer       1.00      1.00      1.00        12
Network Security Engineer       1.00      1.00      1.00        22
       Operations Manager       1.00      1.00      1.00        16
                     PMO       1.00      1.00      1.00        16
        Python Developer       1.00      1.00      1.00        17
           SAP Developer       1.00      1.00      1.00        18
                   Sales       1.00      1.00      1.00        22
                 Testing       1.00      1.00      1.00        17
          Web Designing       1.00      1.00      1.00        13

                accuracy                           1.00       420
               macro avg       1.00      1.00      1.00       420
            weighted avg       1.00      1.00      1.00       420
```



In [203…

```
draft2_sai_resume = better_original_bert["Resume"].iloc[-1]

print(draft2_sai_resume)
```

satini sai keerthan ai data engineer <NUM> <NUM> saikeerthan<NUM>gmailcom choa chu kang singapore objective fresh graduate diploma ai data engineering nanyang polytechnic equipped strong data visualization data analytics skills proficient tools power bi tableau python sql proven experience designing interactive dashboards etl pipelines adept leveraging analytical insights drive decisionmaking improve efficiency seeking apply technical interpersonal skills data analyst good job creations singapore pte ltd experience data engineering intern xyz tech solutions singapore september <NUM> april <NUM> overlooked migration legacy data systems cloudbased solutions automated datacleaning scripts python reducing manual errors <NUM> assisted creation sql queries databases data extraction analysis collaborated teams design kpi dashboard improving reporting speed <NUM> key skills data preparation visualisation programming languages data modelling etl development machine learning models communication teamwork problemsolving education nanyang polytechnic ang mo kio singapore <NUM> diploma ai data engineer relevant modules machine learning data analytics business intelligence tools cloud computing year sem shopping console python project year sem mitigate dengue disease data science project utilizing powerbi year sem predict factory environment mqttsql project arduino leadership lead team members data science project mitigate dengue cases data analytics software like powerbi presented findings stakeholders demonstrating effective communication team work delegated tasks monitored progress ensuring timely completion exceeding project goals <NUM> certifications microsoft certified azure ai fundamentals <NUM> nvidia certification awards dean s list outstanding academic performance <NUM> languages fluent english telugu conversational hindi

In [ ]:
```
import torch
from transformers import BertTokenizer, BertForSequenceClassification

# Load the trained model and tokenizer
model = BertForSequenceClassification.from_pretrained("bert_resume_classi
tokenizer = BertTokenizer.from_pretrained("bert_resume_classifier2")


device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
model.eval()


print("Extracted Resume Text:\n", my_resume_text)


inputs = tokenizer(draft2_sai_resume, padding=True, truncation=True, max_


input_ids = inputs["input_ids"].to(device)
attention_mask = inputs["attention_mask"].to(device)


with torch.no_grad():
    outputs = model(input_ids, attention_mask=attention_mask)
    predicted_label_index = torch.argmax(outputs.logits, dim=1).item()


print(f"Predicted Index: {predicted_label_index}")


if hasattr(label_encoder, "classes_"):
    class_names = list(label_encoder.classes_)
    predicted_category = class_names[predicted_label_index]
```

```
else:
    raise ValueError("Error: label_encoder does not contain class names."


print(f"Predicted Category for My Resume: {predicted_category}")
```

```
Extracted Resume Text:
 satini sai keerthan ai data engineer <NUM> <NUM> saikeerthan<NUM>gmailcom
choa chu kang singapore objective fresh graduate diploma ai data engineeri
ng nanyang polytechnic equipped strong data visualization data analytics s
kills proficient tools power bi tableau python sql proven experience desig
ning interactive dashboards etl pipelines adept leveraging analytical insi
ghts drive decisionmaking improve efficiency seeking apply technical inter
personal skills data analyst good job creations singapore pte ltd experien
ce data engineering intern xyz tech solutions singapore september <NUM> ap
ril <NUM> overlooked migration legacy data systems cloudbased solutions au
tomated datacleaning scripts python reducing manual errors <NUM> assisted
creation sql queries databases data extraction analysis collaborated teams
design kpi dashboard improving reporting speed <NUM> key skills data prepa
ration visualisation programming languages data modelling etl development
machine learning models communication teamwork problemsolving education na
nyang polytechnic ang mo kio singapore <NUM> diploma ai data engineer rele
vant modules machine learning data analytics business intelligence tools c
loud computing year sem shopping console python project year sem mitigate
dengue disease data science project utilizing powerbi year sem predict fac
tory environment mqttsql project arduino leadership lead team members data
science project mitigate dengue cases data analytics software like powerbi
presented findings stakeholders demonstrating effective communication team
work delegated tasks monitored progress ensuring timely completion exceedi
ng project goals <NUM> certifications microsoft certified azure ai fundame
ntals <NUM> nvidia certification awards dean s list outstanding academic p
erformance <NUM> languages fluent english telugu conversational hindi
Predicted Index: 6
Predicted Category for My Resume: Data Science
```

USE Cohere

In [ ]:

```
# import cohere
# import fastavro.read
# import pandas as pd

# # Initialize Cohere API client (Replace 'YOUR_API_KEY' with your actual
# co = cohere.Client(api_key="YOUR_API_KEY")

# # List of 23 job categories
# categories = [
#     "Data Scientist", "Machine Learning Engineer", "Software Engineer",
#     "Cybersecurity Analyst", "AI Researcher", "Cloud Engineer", "Busine
#     "Front-End Developer", "Back-End Developer", "DevOps Engineer", "Fu
#     "Game Developer", "Embedded Systems Engineer", "IoT Specialist", "C
#     "Natural Language Processing Engineer", "Database Administrator", "
#     "Technical Writer", "Product Manager", "IT Support Specialist", "Ro
# ]

# def generate_resume(category):
#     """
#     Uses Cohere's `chat_stream` to generate a resume for a given job ca
#     """
#     prompt = f"""
```

```
#     Generate a well-structured, professional resume for a {category} po
#     Include sections such as Summary, Skills, Experience, and Education
#     Ensure the resume is formatted clearly and relevant to industry sta
#     """

#     # Initialize the streaming request
#     stream = co.chat_stream(
#         model='command-r-08-2024',
#         message=prompt,
#         temperature=0.3,
#         chat_history=[],
#         prompt_truncation='AUTO'
#     )

#     # Capture the generated text
#     generated_text = ""
#     for event in stream:
#         if event.event_type == "text-generation":
#             generated_text += event.text  # Append generated text

#     return generated_text.strip()

# # Generate resumes for all categories
# resumes = []
# for category in categories:
#     resume_text = generate_resume(category)
#     resumes.append({"Category": category, "Resume": resume_text})

# # Convert to DataFrame
# df = pd.DataFrame(resumes)

# # Display the generated dataset
# import ace_tools as tools
# tools.display_dataframe_to_user(name="Generated Resumes with Cohere", d
```