Natural Prcosessing language(NLP)

Assignment 01
Name: Syed Umer Ali Shah
Roll No:10310
Class:BS-SE 7th
Section:B

Description Of Corpus:
Every day, thousands of companies and individuals turn to LinkedIn in search of talent. This dataset contains a nearly comprehensive record of 124,000+ job postings listed in 2023 and 2024. Each individual posting contains dozens of valuable attributes for both postings and companies, including the title, job description, salary, location, application URL, and work-types (remote, contract, etc), in addition to separate files containing the benefits, skills, and industries associated with each posting. The majority of jobs are also linked to a company, which are all listed in another csv file containing attributes such as the company description, headquarters location, and number of employees, and follower count.

With so many datapoints, the potential for exploration of this dataset is vast and includes exploring the highest compensated titles, companies, and locations; predicting salaries/benefits through NLP; and examining how industries and companies vary through their internship offerings and benefits. Future updates will permit further exploration into time-based trends, including company growth, prevalence of remote jobs, and demand of individual job titles over time.

LINK TO CORPUS: https://drive.google.com/drive/folders/17LvJCFsVtGkuAf7qpiSAU-kpZyL3McQy?dmr=1&ec=wgc-drive-hero-goto
REFERENCE TO WEBSITE:https://www.kaggle.com/datasets/arshkon/linkedin-job-postings
GITHUB REPOSITORY:https://github.com/umersuas/NLP-Assignment

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd

data=pd.read_csv('/content/drive/MyDrive/Dataset/postings.csv')

data.head()
```

	job_id	company_name	title	description	max_salary	pay_period	location	company_id	views	med_salary	...	skills_
0	921716	Corcoran Sawyer Smith	Marketing Coordinator	Job descriptionA leading real estate firm in N...	20.0	HOURLY	Princeton, NJ	2774458.0	20.0	NaN	...	Requirem \n\nW seek College
1	1829192	NaN	Mental Health Therapist/Counselor	At Aspen Therapy and Wellness , we are committ...	50.0	HOURLY	Fort Collins, CO	NaN	1.0	NaN	...	
2	10998357	The National Exemplar	Assitant Restaurant Manager	The National Exemplar is accepting application...	65000.0	YEARLY	Cincinnati, OH	64896719.0	8.0	NaN	...	W cur acce resum FOH
3	23221523	Abrams Fensterman, LLP	Senior Elder Law / Trusts and Estates Associat...	Senior Associate Attorney - Elder Law / Trusts...	175000.0	YEARLY	New Hyde Park, NY	766262.0	16.0	NaN	...	This po requ bas understar
4	35982263	NaN	Service Technician	Looking for HVAC service tech with experience ...	80000.0	YEARLY	Burlington, IA	NaN	3.0	NaN	...	

5 rows × 31 columns

Q1: Choose any corpus of your choice of at least 200 MBs of any domain in NLP and perform the following tasks:

1. Text Preprocessing (Text Cleaning, Stemming / Lemmatization)
2. Word Embedding (using an algorithm like Word2Vec, Glove, FastText)
3. Encoding Techniques (Bag of Words, One – Hot)
4. Parts of Speech tagging.

1. Text Preprocessing (Text Cleaning, Stemming / Lemmatization)

```
# Install necessary libraries
!pip install nltk pandas

import nltk
import pandas as pd
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
import re

# Download necessary NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

# Load the dataset from Google Drive
file_path = '/content/drive/MyDrive/Dataset/postings.csv'
df = pd.read_csv(file_path)

#Applying on one of the column from the corpus"episodeName"
text_column = 'description'
texts = df[text_column].dropna()  # Remove any NaN values in the text column

# Text cleaning function (removes punctuation, converts to lowercase)
def clean_text(text):
    text = re.sub(r'[^\w\s]', '', text.lower())
    return text

# Initialize stopwords, stemmer, and lemmatizer
stop_words = set(stopwords.words('english'))
```

```
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

# Function to process a single text
def process_text(text):
    cleaned_text = clean_text(text)
    tokens = word_tokenize(cleaned_text)

    # Remove stopwords
    tokens = [word for word in tokens if word not in stop_words]

    # Apply stemming
    stemmed_tokens = [stemmer.stem(word) for word in tokens]

    # Apply lemmatization
    lemmatized_tokens = [lemmatizer.lemmatize(word) for word in tokens]

    return {
        "cleaned_text": cleaned_text,
        "stemmed_tokens": stemmed_tokens,
        "lemmatized_tokens": lemmatized_tokens
    }

# Apply the processing to all texts in the dataset
processed_texts = texts.apply(process_text)

# Show some results
for index, result in processed_texts.head().items():
    print(f"Original Text (Row {index}): {texts.iloc[index]}")
    print(f"Cleaned Text: {result['cleaned_text']}")
    print(f"Stemmed Tokens: {result['stemmed_tokens']}")
    print(f"Lemmatized Tokens: {result['lemmatized_tokens']}")
    print('-' * 50)
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.2.2)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.6)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-13-3f12c1b6294f> in <cell line: 55>()
     53
     54 # Apply the processing to all texts in the dataset
---> 55 processed_texts = texts.apply(process_text)
     56
     57 # Show some results

 9 frames
lib.pyx in pandas._libs.lib.map_infer()

/usr/local/lib/python3.10/dist-packages/nltk/stem/porter.py in _apply_rule_list(self, word, rules)
    264                     # Don't try any further rules
    265                     return word
--> 266             if word.endswith(suffix):
    267                 stem = self._replace_suffix(word, suffix, "")
    268                 if condition is None or condition(stem):

KeyboardInterrupt:
```

It took me the one hour to process but did'nt complete the processing. so, i aborted the process.

2. Word Embedding (using an algorithm like Word2Vec, Glove, FastText)

```
# Install necessary libraries
from nltk.stem import PorterStemmer, WordNetLemmatizer
from gensim.models import Word2Vec

# Download necessary NLTK resources
nltk.download('punkt')
nltk.download('stopwords')

# Load the dataset from Google Drive
file_path = '/content/drive/MyDrive/Dataset/postings.csv'
df = pd.read_csv(file_path)

#Applying Word2Vec on the column desription from the corpus
text_column = 'pay_period'
texts = df[text_column].dropna()  # Remove any NaN values in the text column

# Text cleaning function (removes punctuation, converts to lowercase)
def clean_text(text):
    text = re.sub(r'[^\w\s]', '', text.lower())
    return text

# Initialize stopwords
stop_words = set(stopwords.words('english'))

# Function to process a single text
def preprocess_text(text):
    cleaned_text = clean_text(text)
    tokens = word_tokenize(cleaned_text)

    # Remove stopwords
    tokens = [word for word in tokens if word not in stop_words]
    return tokens

# Apply preprocessing to all texts in the dataset
tokenized_texts = texts.apply(preprocess_text)

# Train Word2Vec model on the tokenized data
word2vec_model = Word2Vec(sentences=tokenized_texts, vector_size=100, window=5, min_count=1, workers=4)

# Example: Get the word vector for a word (e.g., 'health')
word_vector = word2vec_model.wv['hourly']

# Example: Find similar words to 'health'
similar_words = word2vec_model.wv.most_similar('hourly')

# Example: Convert a sample text into a vector by averaging word embeddings
def text_vector(text):
    tokens = preprocess_text(text)
    vector = sum([word2vec_model.wv[word] for word in tokens if word in word2vec_model.wv])
    return vector / len(tokens) if tokens else None

# Apply the text vector conversion to the dataset
df['text_vector'] = texts.apply(text_vector)

# Show a sample of the text vectors
print(df[['pay_period', 'text_vector']].head())
print("Most similar words to 'health':", similar_words)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
  pay_period                                        text_vector
0     HOURLY  [-0.0086196875, 0.003665738, 0.0051898835, 0.0...
1     HOURLY  [-0.0086196875, 0.003665738, 0.0051898835, 0.0...
2     YEARLY  [-0.00053622725, 0.00023643136, 0.0051033497, ...
3     YEARLY  [-0.00053622725, 0.00023643136, 0.0051033497, ...
4     YEARLY  [-0.00053622725, 0.00023643136, 0.0051033497, ...
Most similar words to 'health': [('weekly', 0.06797593832015991), ('biweekly', 0.004503030329942703), ('yearly', -0.010839177295565605),
```

3. Encoding Techniques (Bag of Words, One – Hot)

1. Bag of Words

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

# Load the dataset
file_path = '/content/drive/MyDrive/Dataset/postings.csv'
df = pd.read_csv(file_path)

# Display the first few rows of the dataset to identify the text column
print(df.head())

text_data = df['company_name'].fillna('')  # Filling NaNs with empty strings to avoid issues

# Initialize the CountVectorizer for BoW encoding
vectorizer = CountVectorizer()

# Apply Bag of Words encoding
bow_matrix = vectorizer.fit_transform(text_data)

# Convert the BoW matrix to a DataFrame for easier interpretation
bow_df = pd.DataFrame(bow_matrix.toarray(), columns=vectorizer.get_feature_names_out())

# Display the BoW DataFrame
print(bow_df.head())
```

```
      job_id            company_name  \
0    921716   Corcoran Sawyer Smith
1   1829192                     NaN
2  10998357  The National Exemplar
3  23221523  Abrams Fensterman, LLP
4  35982263                     NaN

                                          title  \
0                           Marketing Coordinator
1                Mental Health Therapist/Counselor
2                     Assitant Restaurant Manager
3  Senior Elder Law / Trusts and Estates Associat...
4                              Service Technician

                                     description  max_salary pay_period  \
0  Job descriptionA leading real estate firm in N...        20.0     HOURLY
1  At Aspen Therapy and Wellness , we are committ...        50.0     HOURLY
2  The National Exemplar is accepting application...     65000.0     YEARLY
3  Senior Associate Attorney - Elder Law / Trusts...    175000.0     YEARLY
4  Looking for HVAC service tech with experience ...     80000.0     YEARLY

           location  company_id  views  med_salary  ...  \
0     Princeton, NJ   2774458.0   20.0         NaN  ...
1   Fort Collins, CO         NaN    1.0         NaN  ...
2    Cincinnati, OH  64896719.0    8.0         NaN  ...
3  New Hyde Park, NY    766262.0   16.0         NaN  ...
4    Burlington, IA         NaN    3.0         NaN  ...

                                     skills_desc   listed_time  \
0  Requirements: \n\nWe are seeking a College or ...  1.713398e+12
1                                            NaN  1.712858e+12
2  We are currently accepting resumes for FOH - A...  1.713278e+12
3  This position requires a baseline understandin...  1.712896e+12
4                                            NaN  1.713452e+12

   posting_domain  sponsored  work_type currency compensation_type  \
0             NaN          0  FULL_TIME      USD       BASE_SALARY
1             NaN          0  FULL_TIME      USD       BASE_SALARY
2             NaN          0  FULL_TIME      USD       BASE_SALARY
3             NaN          0  FULL_TIME      USD       BASE_SALARY
4             NaN          0  FULL_TIME      USD       BASE_SALARY

  normalized_salary  zip_code     fips
0           38480.0    8540.0  34021.0
1           83200.0   80521.0   8069.0
2           55000.0   45202.0  39061.0
3          157500.0   11040.0  36059.0
4           70000.0   52601.0  19057.0
```

```
[5 rows x 31 columns]
   000  10  100k  101  1021  1080  10news  10x  11  112  ...  化工猎头  国际人才招聘网  \
0    0   0     0    0     0     0       0    0   0    0  ...     0        0
1    0   0     0    0     0     0       0    0   0    0  ...     0        0
2    0   0     0    0     0     0       0    0   0    0  ...     0        0
3    0   0     0    0     0     0       0    0   0    0  ...     0        0
4    0   0     0    0     0     0       0    0   0    0  ...     0        0

   大江生醫  恒荣律师事务所  熊猫外卖  虎虎保险  Design  Druffel  Landscape  and
```

2. One-Hot

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

# Load the dataset (make sure to provide the correct path)
file_path = '/content/drive/MyDrive/Dataset/postings.csv'
data = pd.read_csv(file_path)

# Assuming 'data' is your DataFrame and 'headline_category' is the column you want to encode
data_to_encode = data[['description']]

# Create the encoder
encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore') # sparse=False for dense output

# Fit and transform the data
encoded_data = encoder.fit_transform(data_to_encode)

# Get feature names and create a DataFrame
feature_names = encoder.get_feature_names_out(['description'])
encoded_df = pd.DataFrame(encoded_data, columns=feature_names)

#To see the output, run the code.
print(encoded_df)
```

```
        description_\n\n\n\nBE PART OF SOMETHING REAL\n \n\nYOUR ROLE\n\nAs the full-time Sales Leader, you support the Store Leaders
0                                                     0.0
1                                                     0.0
2                                                     0.0
3                                                     0.0
4                                                     0.0
...                                                   ...
123844                                                0.0
123845                                                0.0
123846                                                0.0
123847                                                0.0
123848                                                0.0

        description_\n\n\n\nHear from our employees\n on why Eversource is the #1 energy efficiency provider in the nation. \n\n\n\nJ
0                                                     0.0
1                                                     0.0
2                                                     0.0
3                                                     0.0
4                                                     0.0
...                                                   ...
123844                                                0.0
123845                                                0.0
123846                                                0.0
123847                                                0.0
123848                                                0.0

        description_\n\n\n\nJob Description Summary GE Vernova Grid Software develops critical software for control rooms of electric
0                                                     0.0
1                                                     0.0
2                                                     0.0
3                                                     0.0
4                                                     0.0
...                                                   ...
123844                                                0.0
123845                                                0.0
123846                                                0.0
123847                                                0.0
123848                                                0.0

        description_\n\n\n\nJob Description Summary GE Vernova’s Advanced Research business is the central innovation engine for GE’s
0                                                     0.0
1                                                     0.0
2                                                     0.0
3                                                     0.0
4                                                     0.0
...                                                   ...
```

```
123844                                                0.0
123845                                                0.0
123846                                                0.0
123847                                                0.0
123848                                                0.0

        description_\n\n\n\nJob Description Summary The Digital Chief Compliance Officer will have responsibility for managing Digita
0                                                     0.0
1                                                     0.0
2                                                     0.0
```

4. Parts of Speech Tagging

```
# Download the necessary NLTK resource for POS tagging
nltk.download('averaged_perceptron_tagger')

# Sample text
text = "This is a sample sentence for parts of speech tagging."

# Tokenize the text
tokens = word_tokenize(text)

# Perform POS tagging
tagged_tokens = nltk.pos_tag(tokens)

# Print the tagged tokens
print("Parts of Speech Tagging:\n", tagged_tokens)


# Applying POS tagging on a column of dataset
text_column = data['location']

# Create an empty list to store the tagged sentences
tagged_sentences = []

# Iterate through each text in the column
for text in text_column:
    # Tokenize the text
    tokens = word_tokenize(str(text))
    # Perform POS tagging
    tagged_tokens = nltk.pos_tag(tokens)
    # Append the tagged tokens to the list
    tagged_sentences.append(tagged_tokens)


print("Tagged Sentences:", tagged_sentences)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
Parts of Speech Tagging:
 [('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('sample', 'JJ'), ('sentence', 'NN'), ('for', 'IN'), ('parts', 'NNS'), ('of', 'IN'), ('spe
Tagged Sentences: [[('Princeton', 'NNP'), (',', ','), ('NJ', 'NNP')], [('Fort', 'NNP'), ('Collins', 'NNP'), (',', ','), ('CO', 'NNP')],
```

Q2: Perform any two of the basic NLP tasks listed below:

1. Sentiment Analysis (using VADER, TextBlob)
2. Named Entity Recognition (NER)
3. Text Classification (Naive Bayes, Logistic Regression, SVM)
4. Language Models (N-grams, Markov Chains)
5. Topic Modeling (LDA, Latent Semantic Analysis)

1. Sentiment Analysis (using VADER, TextBlob)

1. VADER

```
# Install the vaderSentiment library
!pip install vaderSentiment

# Now import the necessary module
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# Rest of the code
import nltk
import pandas as pd
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re

# Download necessary NLTK resources
nltk.download('punkt')
nltk.download('stopwords')

# Load the dataset from Google Drive
file_path = '/content/drive/MyDrive/Dataset/postings.csv'
df = pd.read_csv(file_path)

# Applying VADER on the column named as show.name
text_column = 'company_name'
texts = df[text_column].dropna()  # Remove any NaN values in the text column

# Text cleaning function (removes punctuation, converts to lowercase)
def clean_text(text):
    text = re.sub(r'[^\w\s]', '', text.lower())
    return text

# Initialize stopwords
stop_words = set(stopwords.words('english'))

# Function to process a single text
def preprocess_text(text):
    cleaned_text = clean_text(text)
    tokens = word_tokenize(cleaned_text)

    # Remove stopwords
    tokens = [word for word in tokens if word not in stop_words]
    return ' '.join(tokens)  # Return as a cleaned sentence instead of tokens for VADER

# Apply preprocessing to all texts in the dataset
cleaned_texts = texts.apply(preprocess_text)

# Initialize VADER sentiment analyzer
analyzer = SentimentIntensityAnalyzer()

# Function to apply VADER sentiment analysis to each text
def analyze_sentiment(text):
    return analyzer.polarity_scores(text)

# Apply sentiment analysis to all cleaned texts
sentiment_scores = cleaned_texts.apply(analyze_sentiment)

# Convert sentiment scores into a DataFrame
sentiment_df = pd.DataFrame(sentiment_scores.tolist())

# Combine the original dataset with sentiment scores
df = pd.concat([df, sentiment_df], axis=1)

# Show the dataset with sentiment scores
print(df[[text_column, 'neg', 'neu', 'pos', 'compound']].head())
```

```
Requirement already satisfied: vaderSentiment in /usr/local/lib/python3.10/dist-packages (3.3.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from vaderSentiment) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.4.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2024.8.30)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
             company_name  neg  neu  pos  compound
0   Corcoran Sawyer Smith  0.0  1.0  0.0       0.0
```

```
1                    NaN  0.0  1.0  0.0       0.0
2  The National Exemplar  0.0  1.0  0.0       0.0
3  Abrams Fensterman, LLP  0.0  1.0  0.0       0.0
4                    NaN  0.0  1.0  0.0       0.0
```

2. Text Blob

```
!pip install nltk textblob

# Download necessary NLTK resources
nltk.download('vader_lexicon')

from textblob import TextBlob

# Function to perform sentiment analysis using TextBlob
def analyze_sentiment(text):
  analysis = TextBlob(text)
  if analysis.sentiment.polarity > 0:
    return "Positive"
  elif analysis.sentiment.polarity == 0:
    return "Neutral"
  else:
    return "Negative"

# Applying Sentiment analysis on 'show.description'  column
if 'pay_period' in data.columns:
  data['Sentiment'] = data['pay_period'].apply(analyze_sentiment)
  print(data[['pay_period', 'Sentiment']])
else:
  print("Column 'description' not found in the dataset.")
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: textblob in /usr/local/lib/python3.10/dist-packages (0.17.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.6)
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-29-cbb98b916a37> in <cell line: 19>()
     18 # Applying Sentiment analysis on 'show.description'  column
     19 if 'pay_period' in data.columns:
---> 20   data['Sentiment'] = data['pay_period'].apply(analyze_sentiment)
     21   print(data[['pay_period', 'Sentiment']])
     22 else:

                                  6 frames
lib.pyx in pandas._libs.lib.map_infer()

/usr/local/lib/python3.10/dist-packages/textblob/blob.py in __init__(self, text, tokenizer, pos_tagger, np_extractor, analyzer, parser,
classifier, clean_html)
    382                 parser=None, classifier=None, clean_html=False):
    383         if not isinstance(text, basestring):
--> 384             raise TypeError('The `text` argument passed to `__init__(text)` '
    385                             'must be a string, not {0}'.format(type(text)))
    386         if clean_html:

TypeError: The `text` argument passed to `__init__(text)` must be a string, not <class 'float'>
```

Next steps: Explain error

2. Name Entity Recognition NER

```
# Install necessary libraries
!pip install spacy pandas

import pandas as pd
import spacy
```

```
import re
file_path = '/content/drive/MyDrive/Dataset/postings.csv'
df = pd.read_csv(file_path)
# Load spaCy's pre-trained English NER model
nlp = spacy.load('en_core_web_sm')

# Specifying the column for NER (e.g., 'show.name')
column_name = 'views'

# Ensure there are no missing values in the specified column
df = df[df[column_name].notna()]

# Text cleaning function (removes punctuation, converts to lowercase)
def clean_text(text):
    text = re.sub(r'[^\w\s]', '', text.lower())
    return text

# Apply cleaning to the specific column
df['cleaned_text'] = df[column_name].apply(clean_text)

# Function to perform NER using spaCy
def perform_ner(text):
    doc = nlp(text)  # Process text through spaCy's pipeline
    entities = [(ent.text, ent.label_) for ent in doc.ents]  # Extract named entities
    return entities

# Apply NER to the cleaned text in the specific column
df['named_entities'] = df['cleaned_text'].apply(perform_ner)

# Show the results (Original text and extracted entities)
for index, row in df[[column_name, 'named_entities']].head().iterrows():
    print(f"Original Text (Row {index}): {row[column_name]}")
    print(f"Named Entities: {row['named_entities']}")
    print('-' * 50)
```