Set Up

```
!pip -q install git+https://github.com/huggingface/transformers # need to install from github
!pip install -q datasets loralib sentencepiece
!pip -q install bitsandbytes accelerate
!pip -q install langchain
!pip install einops
!pip install tensorflow_probability>=0.13.0
```

```
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: einops in /usr/local/lib/python3.10/dist-packages (0.7.0)
```

```
import warnings
warnings.filterwarnings("ignore")

import nltk
nltk.download('all')
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag

import pandas as pd

from transformers import AutoTokenizer, AutoModelForTokenClassification, TokenClassificationPi

from Utils import *
```

```
[nltk_data]    | Downloading package vader_lexicon to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package vader_lexicon is already up-to-date!
[nltk_data]    | Downloading package verbnet to /root/nltk_data...
[nltk_data]    |   Package verbnet is already up-to-date!
[nltk_data]    | Downloading package verbnet3 to /root/nltk_data...
[nltk_data]    |   Package verbnet3 is already up-to-date!
[nltk_data]    | Downloading package webtext to /root/nltk_data...
[nltk_data]    |   Package webtext is already up-to-date!
[nltk_data]    | Downloading package wmt15_eval to /root/nltk_data...
[nltk_data]    |   Package wmt15_eval is already up-to-date!
[nltk_data]    | Downloading package word2vec_sample to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package word2vec_sample is already up-to-date!
[nltk_data]    | Downloading package wordnet to /root/nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet2021 to /root/nltk_data...
[nltk_data]    |   Package wordnet2021 is already up-to-date!
[nltk_data]    | Downloading package wordnet2022 to /root/nltk_data...
[nltk_data]    |   Package wordnet2022 is already up-to-date!
[nltk_data]    | Downloading package wordnet31 to /root/nltk_data...
[nltk_data]    |   Package wordnet31 is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package ycoe to /root/nltk_data...
[nltk_data]    |   Package ycoe is already up-to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection all
```

Load Data

```
df = pd.read_csv('/content/Train_Data.csv')
```

```
df.head()
```

	ID	Number	Description	Keywords
0	699	71988	A man in a wheelchair and another sitting on a...	['man', 'wheelchair', 'another', 'bench', 'wat...
1	701	193622	A man sits with a traditionally decorated cow	['man', 'traditionally decorated cow']
2	827	52087	A man getting a drink from a water fountain th...	['man', 'drink', 'water fountain', 'toilet']
3	891	119964	A person holding a skateboard overlooks a dead...	['person', 'skateboard', 'dead field of crops']

Next steps: View recommended plots

BERT BASE Model

```
model_name = "QCRI/bert-base-multilingual-cased-pos-english"

# tokenize
tokenizer = AutoTokenizer.from_pretrained(model_name)

# Token Classification
model = AutoModelForTokenClassification.from_pretrained(model_name)

# Pipeline
pipeline = TokenClassificationPipeline(model=model, tokenizer=tokenizer)
```

```
Some weights of the model checkpoint at QCRI/bert-base-multilingual-cased-pos-english were
- This IS expected if you are initializing BertForTokenClassification from the checkpoint
- This IS NOT expected if you are initializing BertForTokenClassification from the checkpo
```

```
def Predict_Keywords(text):
  word_list= [i['word'] for i in pipeline(text) if i['entity'] in ['NN']]

  result = []
  if len(word_list) > 1:
    for index, word in enumerate(word_list):
        if "#" in word:
            try:
              result[-1] += word[2:]
            except:
              pass
        else:
            result.append(word)

  return result


df['BERT_Keywords']=df['Description'].apply(Predict_Keywords)


df.to_csv("BERT_Baseline_Data_Predicted.csv")


df.head()
```

	ID	Number	Description	Keywords	BERT_Keywords
0	699	71988	A man in a wheelchair and another sitting on a...	['man', 'wheelchair', 'another', 'bench', 'wat...	[man, wheelchair, bench, water]
1	701	193622	A man sits with a traditionally decorated cow	['man', 'traditionally decorated cow']	[man, cow]
2	827	52087	A man getting a drink from a water fountain th...	['man', 'drink', 'water fountain', 'toilet']	[man, drink, water, fountain, toilet]
3	891	119964	A person holding a skateboard overlooks a dead...	['person', 'skateboard', 'dead field of crops']	[person, skateboard, field]

```
processed_df = Data_Mapping(df,'Description','Keywords','BERT_Keywords')
```

```
processed_df
```

	ID	Token	Class	Predicted_Class
0	827	A	O	O
1	827	man	Noun	Noun
2	827	getting	O	O
3	827	a	O	O
4	827	drink	Noun	Noun
5	827	from	O	O
6	827	a	O	O
7	827	water	O	Noun
8	827	fountain	O	Noun
9	827	that	O	O
10	827	is	O	O
11	827	a	O	O
12	827	toilet	Noun	Noun
13	827	.	O	O

Next steps: View recommended plots

```
# Define mapping dictionary
label_map = {'Noun': 1, 'O': 0}

# Map values in 'Labels' column
processed_df['Class'] = processed_df['Class'].map(label_map)
processed_df['Predicted_Class'] = processed_df['Predicted_Class'].map(label_map)


from sklearn.metrics import precision_score, recall_score, f1_score,accuracy_score
#accuracy, precision, recall and F1

metrics_df = pd.DataFrame()

# Compute accuracy,precision, recall, and F1 score
accuracy = accuracy_score(processed_df['Class'], processed_df['Predicted_Class'])
precision = precision_score(processed_df['Class'], processed_df['Predicted_Class'])
recall = recall_score(processed_df['Class'], processed_df['Predicted_Class'])
f1 = f1_score(processed_df['Class'], processed_df['Predicted_Class'])

# Print the results
print("Accuracy:", accuracy*100)
print("Precision:", precision*100)
print("Recall:", recall*100)
print("F1 Score:", f1*100)

pred_noun_count = processed_df[processed_df['Predicted_Class']==1].shape[0]
tokens_noun_count = processed_df[processed_df['Class']==1].shape[0]
```

```
covered_area = pred_noun_count/tokens_noun_count
print("the percentage of covered area:", covered_area)
from collections import defaultdict
results = defaultdict(list)
results['Model'].append("BERT_Baseline")
results['Accuracy'].append(round(accuracy*100,2))
results['Precision'].append(round(precision*100,2))
results['Recall'].append(round(recall*100,2))
results['F1 Score'].append(round(f1*100,2))
results['Covered Area'].append(round(covered_area*100,2))

metrics_df = metrics_df.append(results, ignore_index=True)

metrics_df
```

```
Accuracy: 85.71428571428571
Precision: 60.0
Recall: 100.0
F1 Score: 74.99999999999999
the percentage of covered area: 1.6666666666666667
```

	Model	Accuracy	Precision	Recall	F1 Score	Covered Area
0	[BERT_Baseline]	[85.71]	[60.0]	[100.0]	[75.0]	[166.67]