Assignment-5

142402014
Y. Siva Kumar

Q1) Use the CoNLL-2003 Named Entity Recognition dataset which contains four entity types:

- **PER (Person names)**
- **LOC (Locations)**
- **ORG (Organizations)**
- **MISC (Miscellaneous entities)**

Load the CoNLL-2003 dataset using HuggingFace datasets (https://huggingface.co/datasets/eriktks/conll2003) and initialize a Weights & Biases project called "Q1-weak-supervision-ner". Log the dataset statistics (number of samples, entity distribution) to W&B as summary metrics.

```
import datasets
import wandb
import pandas as pd
import re
import numpy as np
from collections import Counter
from snorkel.labeling import labeling_function, LFApplier, LFAnalysis
from snorkel.labeling.model.baselines import MajorityLabelVoter


# Step 1: Initialize W&B and Load Dataset

wandb.init(project="Q1-weak-supervision-ner")

# Load CoNLL-2003 dataset (Parquet format)
dataset = datasets.load_dataset("eriktks/conll2003",
revision="convert/parquet")

# Flatten dataset to token-level DataFrame
rows = []
for doc in dataset['train']:
for token, label in zip(doc['tokens'], doc['ner_tags']):
rows.append({"token": token, "ner_tag": label})

train_df = pd.DataFrame(rows)

# Dataset statistics
num_tokens = len(train_df)
entity_counts = Counter(train_df['ner_tag'])
print(f"Number of tokens: {num_tokens}")
print(f"Entity distribution (by label index): {entity_counts}")

# Log stats to W&B
wandb.summary["num_tokens"] = num_tokens
wandb.summary["entity_distribution"] = {str(k): v for k, v in
entity_counts.items()}
```

2. Implement two basic labeling functions using Snorkel AI:

- **A heuristic function detecting years (1900-2099) as potential DATE/MISC entities**
- **b. A pattern-matching function identifying organizations by common suffixes ("Inc.","Corp.", "Ltd.")**

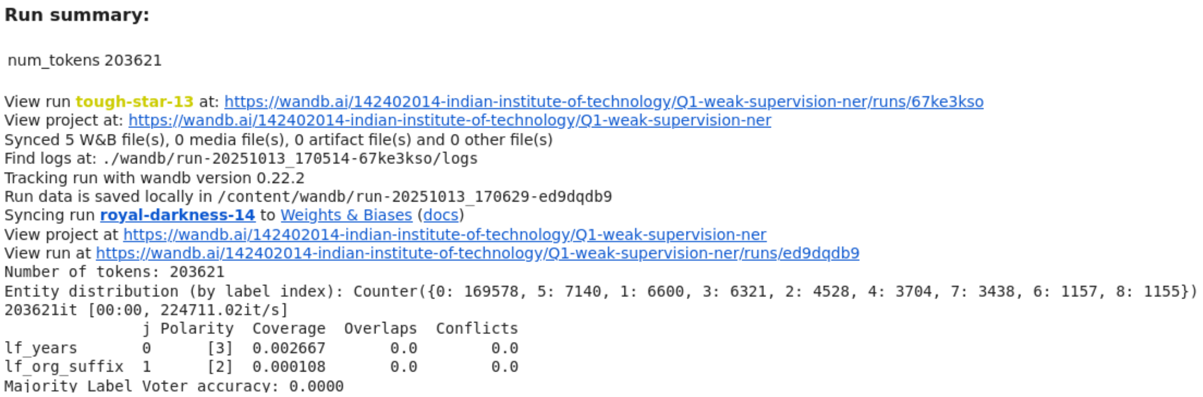
Log each labeling function's coverage and accuracy to W&B using wandb.log()

```
# Step 2: Define Snorkel Labeling Functions

# Label constants
PER, LOC, ORG, MISC, ABSTAIN = 0, 1, 2, 3, -1

# LF1: Detect years 1900-2099 as MISC
@labeling_function()
def lf_years(x):
if re.fullmatch(r"19\d\d|20\d\d", x["token"]):
return MISC
return ABSTAIN

# LF2: Detect organizations by suffix
org_suffixes = ["Inc.", "Corp.", "Ltd.", "LLC", "Co."]

@labeling_function()
def lf_org_suffix(x):
if any(x["token"].endswith(suffix) for suffix in org_suffixes):
return ORG
return ABSTAIN

# Step 2b: Apply LFs

lfs = [lf_years, lf_org_suffix]
applier = LFApplier(lfs=lfs)

# Convert DataFrame to list of dicts so each row is a dict
train_records = train_df.to_dict(orient="records")

# Apply LFs
L_train = applier.apply(train_records)

# LF Analysis
lf_summary_df = LFAnalysis(L=L_train, lfs=lfs).lf_summary()
print(lf_summary_df)


# Step 2c: Log LF coverage & empirical accuracy to W&B safely
```

```
for i, lf in enumerate(lfs):
summary = lf_summary_df.iloc[i]
# Safe handling of coverage column
coverage_value = summary.get("coverage") or summary.get("coverage_pct") or
None
accuracy_value = summary.get("empirical_accuracy", None)
log_dict = {}
if coverage_value is not None:
log_dict[f"{lf.name}_coverage"] = coverage_value
if accuracy_value is not None:
log_dict[f"{lf.name}_empirical_accuracy"] = accuracy_value
wandb.log(log_dict)
```

3. Implement Snorkell's Label aggregation (Majority Label Voter

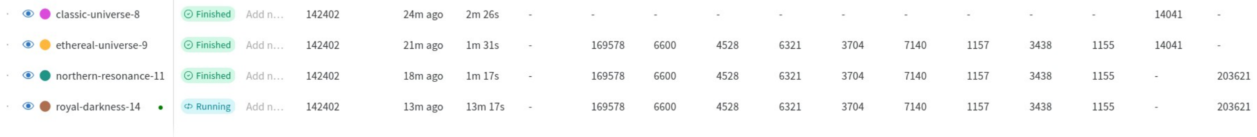
```
# Step 3: Label Aggregation (Majority Voter)

# Specify cardinality=4 to handle 4 entity classes
voter = MajorityLabelVoter(cardinality=4)
y_train = voter.predict(L=L_train)

# Compare with true labels
true_labels = train_df["ner_tag"].values
accuracy = np.mean(y_train == true_labels)
print(f"Majority Label Voter accuracy: {accuracy:.4f}")
wandb.log({"majority_voter_accuracy": accuracy})
```



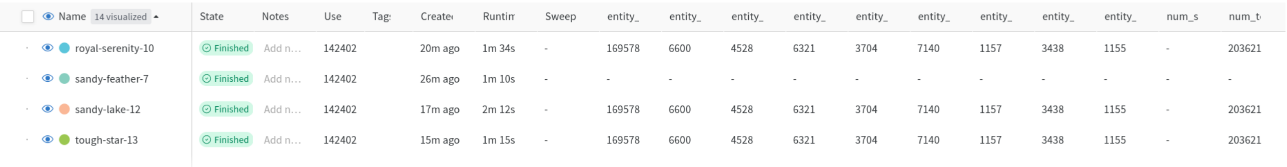
Run summary:

num_tokens 203621

```
View run tough-star-13 at: https://wandb.ai/142402014-indian-institute-of-technology/Q1-weak-supervision-ner/runs/67ke3kso
View project at: https://wandb.ai/142402014-indian-institute-of-technology/Q1-weak-supervision-ner
Synced 5 W&B file(s), 0 media file(s), 0 artifact file(s) and 0 other file(s)
Find logs at: ./wandb/run-20251013_170514-67ke3kso/logs
Tracking run with wandb version 0.22.2
Run data is saved locally in /content/wandb/run-20251013_170629-ed9dqdb9
Syncing run royal-darkness-14 to Weights & Biases (docs)
View project at https://wandb.ai/142402014-indian-institute-of-technology/Q1-weak-supervision-ner
View run at https://wandb.ai/142402014-indian-institute-of-technology/Q1-weak-supervision-ner/runs/ed9dqdb9
Number of tokens: 203621
Entity distribution (by label index): Counter({0: 169578, 5: 7140, 1: 6600, 3: 6321, 2: 4528, 4: 3704, 7: 3438, 6: 1157, 8: 1155})
203621it [00:00, 224711.02it/s]
               j Polarity  Coverage  Overlaps  Conflicts
lf_years       0      [3]  0.002667       0.0        0.0
lf_org_suffix  1      [2]  0.000108       0.0        0.0
Majority Label Voter accuracy: 0.0000
```



● classic-universe-8	Finished	Add n…	142402	24m ago	2m 26s	-	-	-	-	-	-	-	-	-	-	14041	-
● ethereal-universe-9	Finished	Add n…	142402	21m ago	1m 31s	-	169578	6600	4528	6321	3704	7140	1157	3438	1155	14041	-
● northern-resonance-11	Finished	Add n…	142402	18m ago	1m 17s	-	169578	6600	4528	6321	3704	7140	1157	3438	1155	-	203621
● royal-darkness-14 ●	Running	Add n…	142402	13m ago	13m 17s	-	169578	6600	4528	6321	3704	7140	1157	3438	1155	-	203621

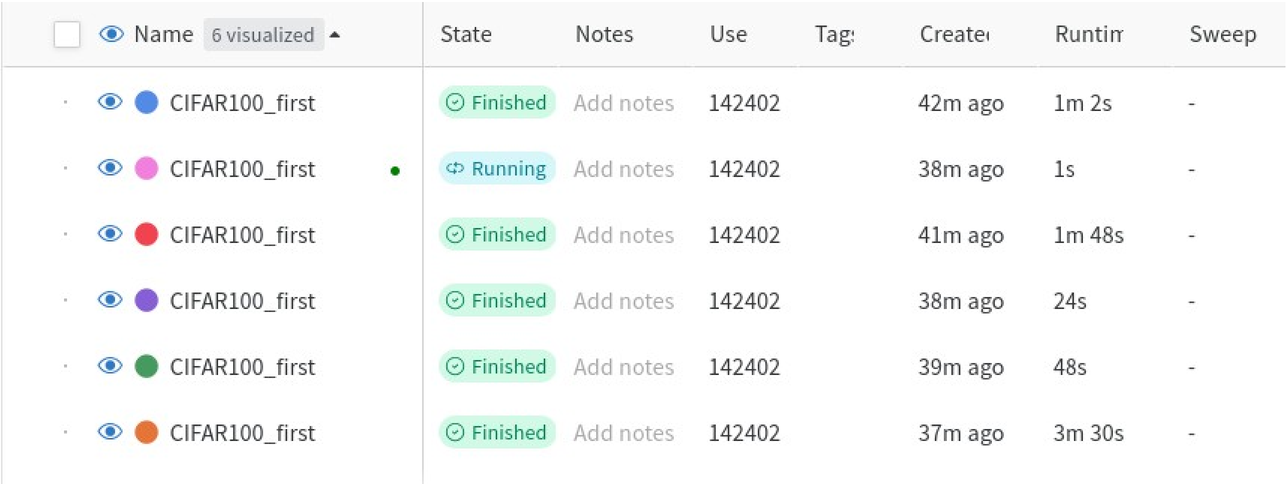


□ 👁 Name 14 visualized ▴	State	Notes	Use	Tag:	Create	Runtir	Sweep	entity_	entity_	entity_	entity_	entity_	entity_	entity_	entity_	entity_	num_s	num_t
👁 ● royal-serenity-10	Finished	Add n...	142402		20m ago	1m 34s	-	169578	6600	4528	6321	3704	7140	1157	3438	1155	-	203621
👁 ● sandy-feather-7	Finished	Add n...	142402		26m ago	1m 10s	-	-	-	-	-	-	-	-	-	-	-	-
👁 ● sandy-lake-12	Finished	Add n...	142402		17m ago	2m 12s	-	169578	6600	4528	6321	3704	7140	1157	3438	1155	-	203621
👁 ● tough-star-13	Finished	Add n...	142402		15m ago	1m 15s	-	169578	6600	4528	6321	3704	7140	1157	3438	1155	-	203621

Q4) Implement the following in Weights and Bias:

- **Train CIFAR 100 and CIFAR 10 sequentially for 100 epochs**
- **Train CIFAR 10 and CIFAR 100 sequentially for 100 epochs.**

```
# CIFAR-10 & CIFAR-100 SEQUENTIAL TRAINING USING W&B
import wandb
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers



# Patch WandB Graph to Avoid Keras Graph Bug

from wandb.sdk.data_types import graph
def patched_from_keras(cls, model):
return None
graph.Graph.from_keras = classmethod(patched_from_keras)

# Function: Build Simple CNN (can replace with ViT if you want)
def build_cnn(num_classes):
model = keras.Sequential([
layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
layers.MaxPooling2D((2, 2)),
layers.Conv2D(64, (3, 3), activation='relu'),
layers.MaxPooling2D((2, 2)),
layers.Flatten(),
layers.Dense(256, activation='relu'),
layers.Dense(num_classes, activation='softmax')
])
return model

# Function: Train on a dataset (CIFAR-10 or CIFAR-100)
def train_on_dataset(dataset_name, num_classes, run_name,
pretrained_model=None):
wandb.init(
project="Q4-cifar-transfer-learning",
name=run_name,
config={"dataset": dataset_name, "epochs": 100, "num_classes": num_classes},
settings=wandb.Settings(_disable_stats=True)
)

# Load dataset
if dataset_name == "cifar100":
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar100.load_data()
elif dataset_name == "cifar10":
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()
else:
raise ValueError("Dataset must be 'cifar10' or 'cifar100'.")

x_train, x_test = x_train.astype("float32") / 255.0,
x_test.astype("float32") / 255.0
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# Model
if pretrained_model:
base = pretrained_model
base.pop() # remove old head
base.add(layers.Dense(num_classes, activation='softmax')) # new head
model = base
print(f"Transferred model → now training on {dataset_name}")
else:
model = build_cnn(num_classes)

model.compile(
optimizer=keras.optimizers.Adam(learning_rate=1e-3),
loss="categorical_crossentropy",
metrics=["accuracy"]
)

# Train
model.fit(
x_train, y_train,
validation_data=(x_test, y_test),
epochs=100,
batch_size=128,
callbacks=[wandb.keras.WandbCallback(save_model=False)]
)

wandb.finish()
return model




# Sequential Experiments
# --- (a) CIFAR-100 → CIFAR-10 ---
print("\n=== Training CIFAR-100 → CIFAR-10 ===")
model_100 = train_on_dataset("cifar100", 100, "CIFAR100_first")
train_on_dataset("cifar10", 10, "CIFAR10_after_CIFAR100",
pretrained_model=model_100)

# --- (b) CIFAR-10 → CIFAR-100 ---
print("\n=== Training CIFAR-10 → CIFAR-100 ===")
model_10 = train_on_dataset("cifar10", 10, "CIFAR10_first")
train_on_dataset("cifar100", 100, "CIFAR100_after_CIFAR10",
pretrained_model=model_10)
```



□ Name 6 visualized ▲	State	Notes	Use	Tag:	Create	Runtim	Sweep
CIFAR100_first	Finished	Add notes	142402		42m ago	1m 2s	-
CIFAR100_first ●	Running	Add notes	142402		38m ago	1s	-
CIFAR100_first	Finished	Add notes	142402		41m ago	1m 48s	-
CIFAR100_first	Finished	Add notes	142402		38m ago	24s	-
CIFAR100_first	Finished	Add notes	142402		39m ago	48s	-
CIFAR100_first	Finished	Add notes	142402		37m ago	3m 30s	-