```
import pandas as pd
import numpy as np
from sklearn.metrics import precision_recall_fscore_support
import logging
import set_ops as setops
import setops_sup
from transformers import AutoConfig, AutoModel, AutoTokenizer, PreTrainedModel
import torch
import random
import os
import argparse
import string

# set up logging info
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

def set_seed(seed):
    """ Set all seeds to make results reproducible """
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False
    np.random.seed(seed)
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)

def id_generator(size=6):
    return ''.join(random.choice(string.ascii_uppercase + string.digits) for _ in
range(size))


def eval_unsup_inter_inter(df_all, df_test, cat1:str, cat2:str, model_name,
data_tempdir, n_sample=20, top_k=1, local_files_only=False):
    # cat1, cat2: categories, in string format
    # step1, determines the number of sentences belongs to both cat1 and cat2
    txt = set(df_all.utterance)
    txt_12 = []
    for t in txt:
        cats = set(df_all[df_all.utterance==t].label)
        if (cat1 in cats) and (cat2 in cats):
            txt_12.append(t)

    # step2, get sample sentences in cat1 and cat2, and get pool
    txt1 = set(df_all[df_all.label==cat1].utterance)
```

```
    txt2 = set(df_all[df_all.label==cat2].utterance)
    sample1 = random.sample(txt1, n_sample)
    sample2 = random.sample(txt2, n_sample)
    n = len( set(txt_12) - set(sample1) - set(sample2) )
    pool_ls = []
    for t in set(df_test.utterance):
        if (t not in sample1) and (t not in sample2):
            pool_ls.append(t)
    # save them in csv
    pd.DataFrame(sample1,
columns=["utterance"]).to_csv(data_tempdir+f"/{cat1}_sample.csv", index=False)
    pd.DataFrame(sample2,
columns=["utterance"]).to_csv(data_tempdir+f"/{cat2}_sample.csv", index=False)
    df_pool = pd.DataFrame(pool_ls, columns=["utterance"])
    df_pool.to_csv(data_tempdir+"/pool.csv", index=False)

    # step3, create sets, do intersection and evaluate
    pool = setops.Set(data_path=data_tempdir+'/pool.csv',
embedding_hf_modelcard=model_name, local_files_only=local_files_only)
    sample_set = dict()
    cossim_set = dict()
    true_label = []
    pred_label = []
    acc_count = 0
    for cat in [cat1, cat2]:
        sample_set[cat] = setops.Set(data_path=data_tempdir+f'/{cat}_sample.csv',
embedding_hf_modelcard=model_name, local_files_only=local_files_only)
        idx, cossim = pool.intersect(sample_set[cat], top_k=top_k, debug=True)
        cossim_set[cat] = cossim
    cossim = cossim_set[cat1] + cossim_set[cat2]
    idx = np.argsort(-cossim).tolist()
    txt_inters = set(df_pool.iloc[idx[:n]].utterance)
    for t in txt_inters:
        t_labels = set(df_test[df_test.utterance==t].label)
        if (cat1 in t_labels) and (cat2 in t_labels):
            true_label.append(cat1+"-"+cat2)
            acc_count += 1
        else:
            true_label.append("-".join(t_labels))
    pred_label = [cat1+"-"+cat2] * n
    res = precision_recall_fscore_support(true_label, pred_label,
average='weighted', zero_division=1.0)
    acc = acc_count / n
    return (acc,) + res[:2] + (2*res[0]*res[1]/(res[0] + res[1]),) + (n,)
```

```
def eval_sup_inter_inter(df_all, df_test, cat1:str, cat2:str, model_name,
data_tempdir, model_tempdir, n_sample=20, top_k=1, train_epoch=60,
local_files_only=False):
    # step 1. finetune model based on the presented intents
    args = setops_sup.Training_args(model_name=model_name, bs=8,
train_epoch=train_epoch, output_dir=model_tempdir)
    trainer = setops_sup.SetCSETrainer(args, local_files_only=local_files_only)
    cats = set(df_all.label)
    sample_ls = dict()
    sample_sets = dict()
    for cat in cats:
        dfi = df_all[df_all['label']==cat]
        dfi.drop_duplicates()
        sample = dfi.sample(frac=1).iloc[:n_sample]
        sample.to_csv(data_tempdir+f'/{cat}_sample.csv', index=False)
        sample_ls[cat] = set(sample.utterance)
        sample_set = setops_sup.Set(data_path=data_tempdir+f'/{cat}_sample.csv')
        sample_sets[cat] = sample_set
    trainer.SetCSEtrain(list(sample_sets.values()))

    tokenizer = AutoTokenizer.from_pretrained(args.model_name)
    model = AutoModel.from_pretrained(args.output_dir)

    # step 2. get n
    txt = set(df_test.utterance)
    txt_12 = []
    for t in txt:
        labels_t = set(df_test[df_test.utterance==t].label)
        if (cat1 in labels_t) and (cat2 in labels_t):
            txt_12.append(t)
    n = len( set(txt_12) - set(sample_ls[cat1]) - set(sample_ls[cat2]) )

    # step 3. do intersection and evaluation
    pool_df = pd.DataFrame(list( set(df_test.utterance) - set(sample_ls[cat1]) -
set(sample_ls[cat2]) ), columns=["utterance"])
    pool_df.to_csv(data_tempdir+"/pool.csv", index=False)
    pool = setops_sup.Set(data_path=data_tempdir+'/pool.csv')

    cossim_set = dict()
    true_label = pred_label = []
    acc_count = 0
    for cat in [cat1, cat2]:
        idx, cossim = pool.intersect(sample_sets[cat], tokenizer, model,
top_k=top_k, debug=True)
        cossim_set[cat] = cossim
```

```
    cossim = cossim_set[cat1] + cossim_set[cat2]
    idx = np.argsort(-cossim).tolist()
    txt_inters = set(pool_df.iloc[idx[:n]].utterance)
    for t in txt_inters:
        t_labels = set(df_test[df_test.utterance==t].label)
        if (cat1 in t_labels) and (cat2 in t_labels):
            true_label.append(cat1+"-"+cat2)
            acc_count += 1
        else:
            true_label.append("-".join(t_labels))
    pred_label = [cat1+"-"+cat2] * n
    res = precision_recall_fscore_support(true_label, pred_label,
average='weighted', zero_division=1.0)
    acc = acc_count / n
    return (acc,) + res[:2] + (2*res[0]*res[1]/(res[0] + res[1]),) + (n,)


def full_eval(df_all, df_test, cat0, cat1, cat2, n_rep, model_name, n_sample,
top_k, local_files_only):
    # unsup eval
    temp_code = id_generator(4)
    data_tempdir = 'data/temp_' + temp_code
    model_tempdir = 'models/temp_' + temp_code
    if not os.path.exists(data_tempdir):
        os.makedirs(data_tempdir)
    if not os.path.exists(model_tempdir):
        os.makedirs(model_tempdir)
    inters_res_unsup = np.zeros([n_rep, 4])
    for i in range(n_rep):
        res1 = eval_unsup_inter_inter(df_all, df_test, cat0, cat1, model_name,
data_tempdir, n_sample=n_sample, top_k=top_k, local_files_only=False)
        res2 = eval_unsup_inter_inter(df_all, df_test, cat0, cat2, model_name,
data_tempdir, n_sample=n_sample, top_k=top_k, local_files_only=False)
        inters_res_unsup[i] = (res1[-1]*np.asarray(res1[:-1]) + res2[-
1]*np.asarray(res2[:-1]))/(res1[-1] + res2[-1])
        print(f'Unsup finished: {i+1}!')
    inters_res_unsup = pd.DataFrame(inters_res_unsup, columns=['acc',
'precision', 'recall', 'F1'])
    inters_res_unsup_mean =
inters_res_unsup.mean().to_frame('Unsup_Intersects').T

    # sup eval
    temp_code = id_generator(4)
    data_tempdir = 'data/temp_' + temp_code
    model_tempdir = 'models/temp_' + temp_code
```

```
    if not os.path.exists(data_tempdir):
        os.makedirs(data_tempdir)
    if not os.path.exists(model_tempdir):
        os.makedirs(model_tempdir)
    inters_res_sup = np.zeros([n_rep, 4])
    for i in range(n_rep):
        res1 = eval_sup_inter_inter(df_all, df_test, cat0, cat1, model_name,
data_tempdir, model_tempdir, n_sample=n_sample, top_k=top_k,
train_epoch=train_epoch, local_files_only=False)
        res2 = eval_sup_inter_inter(df_all, df_test, cat0, cat2, model_name,
data_tempdir, model_tempdir, n_sample=n_sample, top_k=top_k,
train_epoch=train_epoch, local_files_only=False)
        inters_res_sup[i] = (res1[-1]*np.asarray(res1[:-1]) + res2[-
1]*np.asarray(res2[:-1]))/(res1[-1] + res2[-1])
        print(f'sup finished: {i+1}!')
    inters_res_sup = pd.DataFrame(inters_res_sup, columns=['acc', 'precision',
'recall', 'F1'])
    inters_res_sup_mean = inters_res_sup.mean().to_frame('Sup_Intersects').T

    res = pd.concat([inters_res_unsup_mean, inters_res_sup_mean])
    return res

# Step 1, set up experiment parameters
n_rep = 5
n_sample = 20
top_k = n_sample
model_name = "princeton-nlp/sup-simcse-bert-base-uncased"
local_files_only = False
train_epoch = 60

df_all = pd.read_csv("data/github/github_hdl.csv")
df_test = pd.read_csv("data/github/github_hdl_test.csv")
# df = pd.read_csv("data/english_quotes/quotes_all.csv")
cat0 = "help wanted"
cat1 = "docs"
cat2 = "logger"

# Step 2. set random seed
set_seed(42)

# do eval
res = full_eval(df_all, df_all, cat0, cat1, cat2, n_rep, model_name, n_sample,
top_k, local_files_only)
print(res)
```